you can make games now!

ANNUAL

GAME CAREER GUIDE

BROUGHT TO YOU BY gd GAME DEVELOPER MAGAZINE

FALL 2012

INSIDE:
HANDS-ON HOW-TOS
FOR THE UNREAL
DEVELOPMENT
KIT, THE SOURCE
ENGINE, AND MORE!

UBM

ANNUAL

GAME CAREER GUIDE

FALL 2012

CONTENTS

BROUGHT TO YOU BY gd GAME DEVELOPER MAGAZINE

postmortem

features

departments

COVER ILLUSTRATION BY KEN TAYA

GAME PLAN // BRANDON SHEFFIELD

THE GAME OF MAKING GAMES

WELCOME TO THE WILD WORLD OF VIDEO GAME DEVELOPMENT!

//////// Welcome to *Game Developer* magazine's 2012 Game Career Guide! If you like games and think you want to make them, we're here to get you started. At the very least you'll be able to determine whether making games is something you truly want to do.

Within these pages, we have tutorials that can take you from zero to game with no cost, and minimal learning curve. Whether you want to try out level design in Unreal Engine 3 (page 23), learn to prototype game ideas in 3D space (page 51), chart gameplay metrics in Excel (page 56), or make RPG quests in the Source engine (page 7), we've got you covered. And if you're worried that you're not technical or talented enough to make games, don't worry—with enough time and the right attitude, anyone can! (See more about this on page 58.)

Once you've gotten your feet wet, we have an extensive list of many (though not all!) free game development tools you can use to start making games (page 91), and game engines you can freely mod (page 17). If you want more tutorials, check out our past game career guide issues, where we have basic guides to making games with Flash and Unity, setting up a simple artificial intelligence, and making retro-styled pixel art. You can find those (and lots more stuff, like an interactive and extended version of our school directory on page 64) at www.gamecareerguide.com.

WHAT KIND OF DEVELOPER AM I?

» Some people prefer to work solo, or in small groups. This sort of person is probably best suited to the path of the independent developer: Make small games for small money, but love what you do. If you want to work on a game like Call of Duty or Halo, you'll need to join a large team. Regardless of which path you take, you'll find some information about potential salaries and compensation in our Salary Survey (page 33). These are not guarantees, of course, but they do tell you what developers early in their career are reporting income-wise.

No matter how you want to make games, having some projects under your belt is the best way to show a potential employer (or a potential indie team) that you have what it takes to work with them. It's that chicken-and-egg scenario: You need experience before you can get a job, but how do you get experience making games if you can't get a job in the industry? Answer: Start making them on your own, or with associates at school, or however you can!

OKAY, WHAT NOW?

» Game schools can be a good way to network with future developers, and gain a solid foundation of knowledge across all disciplines. But even if you can't afford school, you can work on your own and make your way in the world of games.

The key when making your own projects is to scale small. Maybe you want to create a huge, sprawling action RPG—but there's a reason it took Diablo III so long to come out. Large projects require a level of polish that's often very difficult when first starting out, especially with a small team.

Start by figuring out what you can realistically achieve, and try to make innovative small projects before you take on that big first-person shooter. If you have some specific area of interest, or some small feature you always wished you could see in games, then go for that! Minecraft, for example, wasn't envisioned as some grand-scale project that would enchant millions. Minecraft's creator Markus Persson just made the game he wanted to play, and it turned out other people did too.

You may worry that people won't latch on to your idea. But none of us is unique, as much as we might like to think so. There's almost certainly someone else out there that likes the things you do. And even if your initial games don't set the world on fire, they'll prove to yourself and to everyone else that you can do it. The more you learn from your failures (and there will be some!), the more you'll improve as a game developer. It's often a good idea to do a postmortem of your project, to figure out how you can improve next time (see page 44). Fail early, fail often!

If you're not confident going it alone, that's no problem. Game jams (see page 63) are a great way to meet other developers that are at or above your level, and try to make some games together with them in a team. In a jam environment, you form a team on the spot and try to make a game in (usually) 48 hours. The time pressure may sound troublesome, but it's a great introduction to creative problem-solving and keeping a project small. Game jams are happening all the time, all over the world, so there's a good chance you'll be able to find a jam near you.

WELCOME TO THE CLUB!

» We want you to make games, and we want you to have fun doing it. We look forward to seeing what you come up with, and hope you'll join us as we all, together, define the industry for years to come. And don't forget to subscribe to *Game Developer* magazine, where you'll get very deep-level tips year-round (check our student discount subscription rates at gdmag.com/subscribe). Happy game making!

—Brandon Sheffield
twitter: @necrosofty

GAME DEVELOPER MAGAZINE
WWW.GDMAG.COM

UBM LLC.
303 Second Street, Suite 900, South Tower
San Francisco, CA 94107
t: 415.947.6000 f: 415.947.6090

SUBSCRIPTION SERVICES

FOR INFORMATION, ORDER QUESTIONS, AND ADDRESS CHANGES
t: 800.250.2429 f: 847.763.9606
e: *gamedeveloper@halldata.com*
www.gdmag.com/contactus

EDITORIAL

PUBLISHER
Simon Carless e: *scarless@gdmag.com*
EDITOR-IN-CHIEF
Brandon Sheffield e: *bsheffield@gdmag.com*
EDITOR
Patrick Miller e: *pmiller@gdmag.com*
MANAGER, PRODUCTION
Dan Mallory e: *dmallory@gdmag.com*
ART DIRECTOR
Joseph Mitch e: *jmitch@gdmag.com*
CONTRIBUTING WRITERS
Tom Curtis, Ben Evans, Sjoerd de Jong, Matthew Weise, Luke Parkes-Haskell, Ira Fay, Robin de Jongh, Matthew Burns, Mare Sheppard
ADVISORY BOARD
Mick West **Independent**
Brad Bulkley **Microsoft**
Clinton Keith **Independent**
Brenda Brathwaite **Loot Drop**
Bijan Forutanpour **Sony Online Entertainment**
Mark DeLoura **THQ**
Carey Chico **Globex Studios**
Mike Acton **Insomniac**

ADVERTISING SALES

GLOBAL SALES DIRECTOR
Aaron Murawski e: *aaron.murawski@ubm.com*
t: 415.947.6227
MEDIA ACCOUNT MANAGER
Jennifer Sulik e: *jennifer.sulik@ubm.com*
t: 415.947.6227
GLOBAL ACCOUNT MANAGER, RECRUITMENT
Gina Gross e: *gina.gross@ubm.com*
t: 415.947.6241
GLOBAL ACCOUNT MANAGER, EDUCATION
Rafael Vallin e: *rafael.vallin@ubm.com*
t: 415.947.6223

ADVERTISING PRODUCTION

PRODUCTION MANAGER
Pete C. Scibilia e: *peter.scibilia@ubm.com*
t: 516-562-5134

REPRINTS

WRIGHT'S MEDIA
Jason Pampell e: *jpampell@wrightsmedia.com*
t: 877-652-5295

AUDIENCE DEVELOPMENT

AUDIENCE DEVELOPMENT MANAGER
Nancy Grant e: *nancy.grant@ubm.com*
LIST RENTAL
Peter Candito
Specialist Marketing Services
t: 631-787-3008 x 3020
e: *petercan@SMS-Inc.com*
ubm.sms-inc.com

UBM
WWW.UBM.COM

WHO TO KNOW AND WHAT TO DO IN THE GAME INDUSTRY

Video-game events and trade shows are an excellent resource for learning about the industry and making connections with professional developers. Here are a few events you should know.

GAME DEVELOPERS CONFERENCE

www.gdconf.com

» The Game Developers Conference (GDC) brings together developers from all walks of life: Indies, social game devs, and console and PC game developers rub shoulders with students and game-development hopefuls. GDC features lectures, summits, workshops, and more, all designed to make you better at your job (or help you land one). Of particular interest for students is the Game Career Seminar, in which industry luminaries give lectures to aspirational game developers. This popular event continues to grow and supports the show's Game Career Pavilion, where attendees can meet face-to-face with some of the industry's top studios and hand out a few resumes.

GDC Europe.

GDC also hosts the annual Independent Games Festival, which showcases some of the finest games in the indie-development space. It has a special Student Showcase, which displays 10 specially chosen student projects and puts them in front of thousands of attendees (and potential publishers).

(GDC is owned and operated by UBM TechWeb, which also owns *Game Developer* magazine. It next takes place in 2013 in San Francisco.)

GDC ONLINE

www.gdconline.com

GDC Online is similar to GDC but (as you may have guessed!) has more of an online focus. It's usually held in Austin, Texas, a hub of game development in the southern United States. But that's not all the show covers—mobile and social titles sit next to free-to-play and subscription-based MMOs, not to mention offline game development. The next GDC Online is October 9–11 2012.

GDC EUROPE

www.gdceurope.com

» GDC Europe follows the structure of the San Francisco-based GDC, but it turns its gaze toward European developers and regional market trends. The show takes place in conjunction with the consumer-focused trade show Gamescom and has cheaper student passes that allow access to all content. It takes place in Cologne, Germany from August 13–15 2012.

GDC CHINA

www.gdcchina.com

» GDC China, which is November 17–19 2012 in Shanghai, hosts talks on business and management, game design, production, programming, and visual arts, but it has a bent toward the Asian market. The show also hosts its own IGF competition, much like the primary GDC, and is open to Oceania as well as all Asian countries.

GAMESCOM

www.gamescom.de/en/gamescom/home

» Be sure to book some extra vacation time after GDC Europe, because Cologne, Germany hosts the consumer-oriented Gamescom on August 15–19 2012. Gamescom is Europe's largest trade fair for games, and the show invites the public and industry professionals to check out upcoming titles and interact with the developers behind them.

DEVELOP CONFERENCE

www.develop-conference.com

» The annual Develop Conference, held in Brighton, U.K., hosts talks on production, coding, audio, design, art, and business. The show also features the Evolve Conference, which focuses on the emergence of new platforms, technologies, and markets. Develop usually takes place during the summer; the 2012 conference is July 10–12.

INDIECADE

www.indiecade.com

» The annual IndieCade Festival brings together independent games and developers from around the world to celebrate the growing indie space. The event allows developers to showcase their titles and network with peers, and the event organizers host an awards ceremony to honor the industry's standout indie games and studios. The next IndieCade will take place October 4–7 in Los Angeles.

PENNY ARCADE EXPO

www.paxsite.com

» The biannual Penny Arcade Expo (PAX), held in Seattle and Boston, builds its reputation on celebrating video games and the culture that surrounds them. PAX's fan-oriented focus makes it the most accessible of all the shows we've mentioned. The show features a range of tournaments and other social events for attendees, as well as a number of panels from industry professionals. The show's expo floor offers a

PHOTOS COURTESY OF THE GAME DEVELOPERS CONFERENCE

GDC 2012 at Moscone Center in San Francisco, CA.

collection of upcoming titles as well as a showcase of top indie games known as the PAX 10. The Seattle-based PAX Prime event takes place August 31–September 2, and the next PAX East takes place in Boston in early 2013. A neat part of the PAX lineup is a smaller, developer-focused event called PAX Dev, which runs right before PAX Prime in Seattle on August 29–30.

D.I.C.E.

www.dicesummit.org

» The D.I.C.E. (Design, Innovate, Communicate, Entertain) Summit is more about the business end of the industry, bringing the higher-level executives together as they discuss developing trends and opportunities within the game industry. The show is organized by the Academy of Interactive Arts & Sciences and hosts the annual Indie Game Challenge as well as the Interactive Achievement Awards. The last D.I.C.E. Summit took place in February 2012 in Las Vegas—details for the 2013 event have yet to be announced.

E3

www.e3expo.com

» The Electronic Entertainment Expo (E3), organized by the ESA, is one of the largest and most extravagant U.S. game events in the industry. The show brings together publishers and hardware manufacturers as they reveal their upcoming projects and build excitement for the ever-important fourth quarter. While the show is only open to professionals, E3 drew in more than 45,700 attendees in 2012. E3 usually falls in early June and is held in downtown Los Angeles.

TOKYO GAME SHOW

www.tgs.cesa.or.jp/

» The Tokyo Game Show (TGS) is Japan's largest video-game trade show, where the region's biggest publishers exhibit their upcoming titles to journalists, fans, and the industry. The show features a number of workshops and conferences for developers and serves as a venue for the Computer Entertainment Suppliers Association's (CESA) Japan Game Awards. TGS's Sense of Wonder Night allows developers to showcase their new or experimental projects and ideas. The first two days of TGS are industry-only, though the final two days are open to the public. The event is co-organized by CESA and Nikkei Business Publications, Inc., and the 2012 show is September 20–23 at the Makuhari Messe in Chiba City, Japan.

REZZED

www.rezzed.com

» Rezzed is a relatively new, U.K.-based PC and indie games showcase event hosted in Brighton by game publications Eurogamer and Rock, Paper, Shotgun. Rezzed is more of a player-focused show, and its biggest draws are its preview builds of PC games, but its indie showcase could offer local devs exposure. Rezzed typically takes place in early July.

ORGANIZATIONS

These organizations support and regulate the game industry, and a number of them are open to student and consumer members.

IGDA

www.igda.org

» The International Game Developers Association (IGDA) aims to connect developers worldwide. This nonprofit organization has chapters in North America, Asia, and Europe, and it sponsors a number of events and parties at the industry's various developer conferences. Members of the organization gain access to a wealth of community tools and discussions that allow developers to connect with each other and share information and ideas. Annual memberships are available to students for $30 and to professionals for $48, with other options available for qualified individuals. They even have a group health care plan!

ESRB

www.esrb.org

»The Electronic Software Ratings Board (ESRB) was created by the ESA in 1994. It rates the content of games released in North America and ensures that the industry follows a common set of marketing guidelines. The organization receives support from publishers and retailers alike.

ESA

www.theesa.com

»The Entertainment Software Association (ESA) represents the interests of U.S. game publishers via antipiracy programs, business and consumer research, government outreach, and intellectual property protection efforts. The organization also owns and operates E3 and funds several charities through its ESA Foundation.

GAME AUDIO NETWORK GUILD

www.audiogang.org

»The Game Audio Network Guild provides a community for game audio professionals to connect with peers, share ideas, and advance the implementation of audio in video games. Members of the organization gain benefits, including access to the G.A.N.G. social network, as well as a number of other opportunities to network with like-minded audio professionals. Membership is available at a variety of levels for both students and professionals.

VIDEO GAME VOTERS NETWORK

www.videogamevoters.org

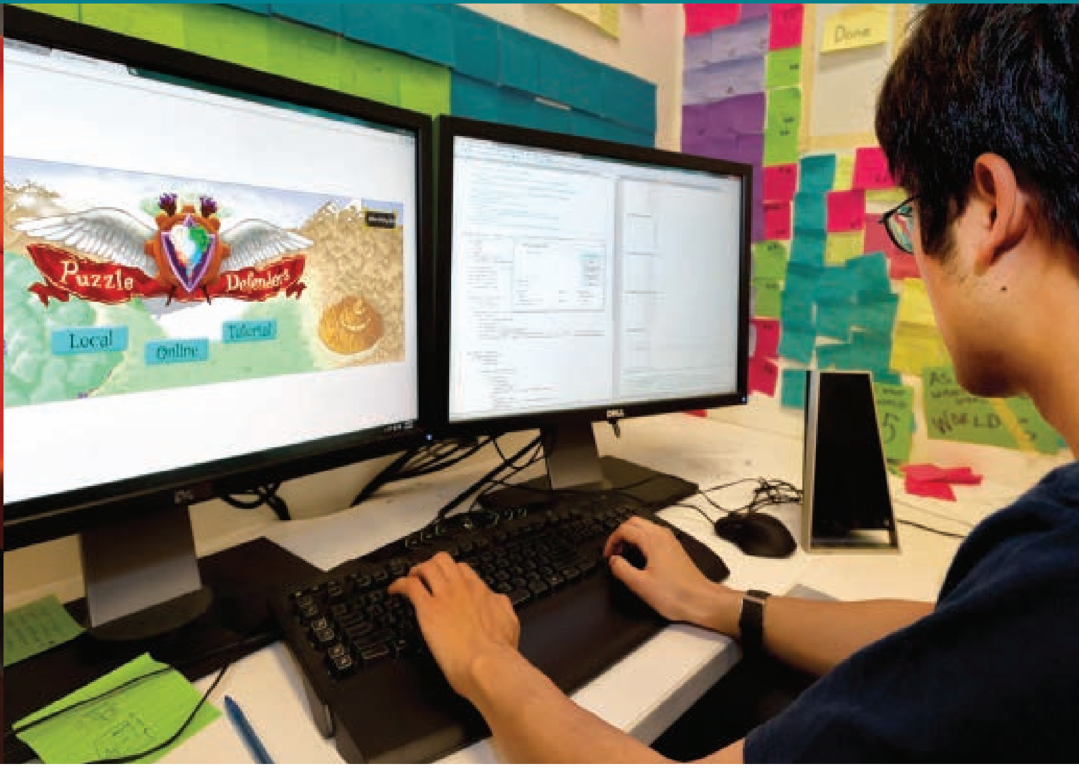
»The Video Game Voters Network helps leads the charge against legislation that threatens the video game industry. Sponsored by the ESA, the organization aims to keep voters educated about issues that could affect game regulation and encourages outreach to local, state, and federal officials.

AIAS

www.interactive.org

»The Academy of Interactive Arts and Sciences (AIAS) is a nonprofit organization that celebrates the developers behind the industry's most influential games. The AIAS hosts the annual D.I.C.E. summit as well as the event's Interactive Achievement Awards.

Invent the Future of Games.



At UC Santa Cruz, we push the boundaries of games as an expressive medium.

We imagine new game experiences that haven't been invented yet. We wrestle with technical questions and come up with innovative answers. We design new genres of games. And we build the technologies to make them real.

+ The largest game research group in North America
+ Game design lab with industry-standard resources, including 3D motion capture
+ Near the epicenter of innovation and entrepreneurship, Silicon Valley
+ Interdisciplinary, with courses spanning art, engineering, and digital media studies
+ Graduates now working at Microsoft Studios, EA, Sony, Zynga, Blizzard, and others
+ A research university located in a redwood forest overlooking the Pacific Ocean

Become part
of the future.

THE UNIVERSITY OF CALIFORNIA · LET THERE BE LIGHT · 1868

UC SANTA CRUZ

B.S. in Computer Science/
Computer Game Design
cs.ucsc.edu/games
admissions.ucsc.edu

DIY RPG

MAKE YOUR OWN RPG WITH THE SOURCE ENGINE

by Ben Evans

If you want to get into game development, modding an existing game is a good way to start exploring design ideas and building prototypes without reinventing the wheel (see our modding primer on page 17). In this tutorial, we're going to use Valve Software's Source Engine (the same game engine that powers Portal, Left 4 Dead, Half-Life, and Team Fortress 2) to prototype a mod with RPG-esque quests within Half-Life 2: Episode 2.

/// Note that this tutorial assumes you have a basic knowledge of Source SDK's Hammer Editor and Face Poser tools, so it's definitely an intermediate-level project. You should be comfortable with building and compiling maps, placing entities, and setting up triggers before following this tutorial, and if you want to add custom dialogue, you'll also need to know how build and save your own scenes in Face Poser. It's okay if you've never used the Source SDK tools before—there are a ton of free resources online that you can use to learn how to use them. Start with the Valve Developer Wiki **(Resource 1)** which hosts a lot of very useful tutorials and references, and then check out design3 **(2)** (disclaimer: I am a design3 contributor) for step-by-step instructional videos on the Source Engine. The minimum requirements to run these tools are the same as the requirements for Half-Life 2: Episode Two: a 1.7 GHz processor, 512MB RAM, DirectX® 8.1 level graphics card, and Windows XP or newer. You will also need a microphone that plugs into your PC.

GEARING UP

/// Before we get started, let's make sure we have everything we need. First, you'll need a copy of Half-Life 2: Episode Two installed. We're going to use this game because it's the most up-to-date and flexible Source engine single-player game available for modding. If you don't have Episode Two, Half-Life 2 will be compatible with most of the elements in this tutorial as well.

If you want to add custom dialogue to your RPG, you'll need some audio editing software. Audacity **(3)** is a free, open-source audio-editing application that we're going to use to record your own dialogue for your mod.

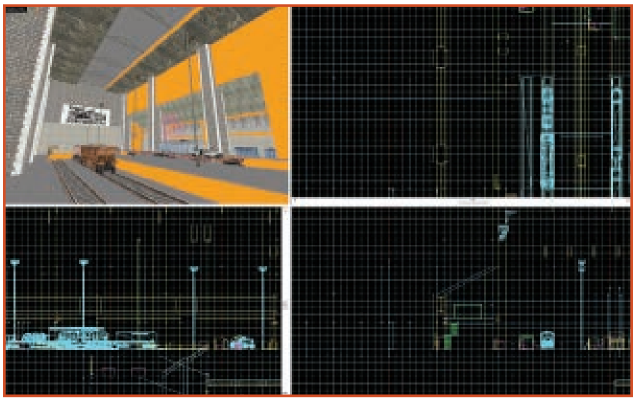
We'll also need a Phoneme Extractor patch **(4)** that will help make Face Poser work better with Windows 7/Vista. Download it and follow the installation instructions. Without that patch, the tool that associates dialogue text strings with sound files won't work.

Next, grab GCFScape **(5)**. GCFScape allows us to open up the cache files that come with Steam games so we can view and extract the files within them. We'll be using it to extract audio for our mod.

If you haven't already, you'll need to install the Source SDK, so open up the Steam Client, go to the Tools tab, and install it. You will need your own map to work with; we'll refer to it as "main map." It can be anything from just a few simple rooms up to a wide-open landscape. **Figure 1** is a screenshot of the level I use in this tutorial—if you'd like to use this level for the tutorial, go ahead and download it at URL http://gdmag.com/resources/code.php

Right now our level is just a rough prototype that hasn't been fully textured or detailed yet. It's always best to start rough and refine the details as you finalize the level, so just come up with a basic level structure to start with for your own RPG. Also, remember to save regularly! Some of the new features in Hammer are not fully supported in the Episode Two version and may crash your editor.

Figure 1: **Our sample level in the Hammer Editor.**



OUTLINING THE QUEST STRUCTURE

/// For this tutorial, let's focus on the questing aspect of a role-playing game—building a character-progression system with experience points would have to be a completely separate guide. Our game level will have three non-player characters (NPCs) giving two quests each. We can use them to give quests, provide information, fight against the player, or just add life to the environment.

To keep our levels organized, let's use a naming convention prefix for our separate groups of entities, as explained in **Figure 2**.

This will make finding the entities much easier, especially when working on individual quests. For example, adding the Level_prefix to entities that only affect the level keeps them all in one location in the list and out of the way of the quest entities.

Before we start making quests, we need to break them down into smaller components. **Figure 3** is a table that lists the order in which the components are used in each quest. Note that "<Quest Components>" and "<Next Quest Activation>" listings refer to the actual game logic within the quests and the trigger to activate the following quest (if there is one).

Don't worry about figuring out all of this yet—we're only making a list of what we need right now. This is a list for just one quest, so you can tell we'll have a lot of entities to keep track of by the time we're done. To keep our viewports (and minds) uncluttered, we're going to use Visible Groups (VisGroups) to separate and hide each quest. VisGroups let you define groups of brushes and entities and hide/show them quickly. If the VisGroup is hidden when the map is compiled, everything in that group will be skipped in the process and won't appear in the game. This is very useful because it allows you to selectively view individual groups of objects by themselves so that you can focus on them without all the other stuff getting in the way. We'll start using them in the next section.

BUILD A QUEST TEMPLATE WITH INSTANCES

/// We're going to use a helpful tool known as instancing within the Hammer Editor. Instancing allows your main map to reference other maps and includes them in the build. We'll be making a "quest instance," which will be the template for every other quest we add later. Once we have our first instance, we can quickly duplicate it, place it in the map, and tweak its settings for each new quest. We can use an instance many times and change properties to make each one unique in order to save a lot of time during development. These instances will control the gameplay and logic of our RPG because they will contain the dialogue and character actions in the game. Unfortunately, Episode Two doesn't fully support instances, so we have to convert them into the level before compiling (which we'll cover when we're ready to compile at the end of this tutorial).

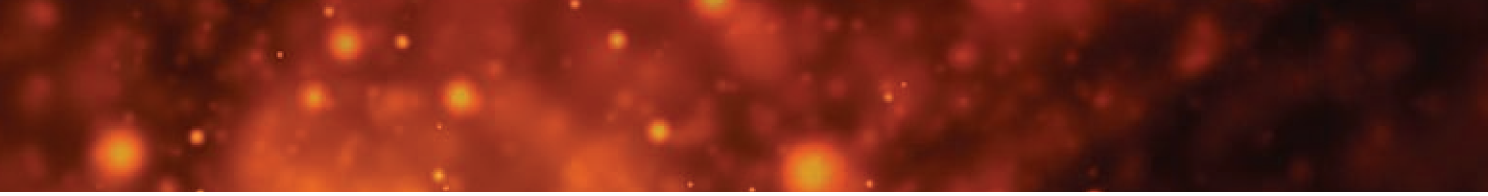
In order to set up our quest instance, we're going to need a new map file. Within Hammer, go to File, New to create a new map, and then save the map as "quest_instance" in a folder named "instances" within the same location as your main map.

Figure 4 shows how our quest_instance should look after we add all the necessary entities. We'll cover those necessary entities next.

It's best to place your entities as close to the origin coordinates

Figure 2: **This naming convention will make it easier for you to stay organized.**

DESCRIPTION	PREFIX	EXAMPLE
Quest	Quest_<n>_<desc>	Quest_1_text
Level	level_<desc>	level_rope
Global Variables	Global_<desc>	Global_logic



(0,0,0) as possible. We also want to have them sitting on top of the X/Y axis because when the instance is imported into the main map it will be oriented around the func_instance entity, which means you might lose sight of entities under the world if they are too low. You can check the location of your entities by using the top and side viewports to determine where they are. If needed, use the Selection Tool to place them above the X/Y axis. Our instance also gives all the entities inside it a prefix of our choice. We'll leave the triggers until the end of placing all the entities and leave some values out until later, so don't worry if it looks like we missed something.

We're going to need all of the entities mentioned in the list in **Figure 3**. Let's start by placing a NPC. With the Entity Tool selected, find npc_citizen on the drop-down list and place it at the origin coordinates. This will be our quest giver. Hit Alt + Enter to open his properties and make his name "giver." Also change Prevent picking up weapons? to Yes. Hit Apply and then go into his Flags panel and check Not Commandable, or else the quest giver will follow your player around the map. We'll also want to enable Don't drop weapons and Ignore player push (don't give way to player).

Next we're going to need a trigger that tells us when the player has approached our quest giver. Create a brush that is 64 units tall, 32 units wide, and 4 units thick, and then place it just in front of our NPC. We'll change its texture to "nodraw" so that it doesn't get rendered in the game. To change the texture, just select the brush, switch to the Toggle texture application tool, click the Browse button, filter for nodraw, double-click the nodraw texture, and click Apply. Then tie the trigger to an entity by pressing Ctrl + T and selecting func_button from the class list. Let's name it "trigger_start" and change the speed to 0. Hit Apply to save the changes we've just made.

Next, add a sprite that will alert the player that the NPC has a quest. Place an env_sprite entity above your NPC's head and open up the Object Properties window by hitting Alt + Enter. Name it "Sprite." We're also going to need to change the render mode so that the sprite renders properly in the game. Within the Object Properties window, locate the Render Mode option box and select World Space Glow from the drop-down menu. You can change the sprite image in this menu if you have one that you want to add to your game, but for now the standard sprite will do. Also, make sure that Start on is checked in the Flags tab.

Figure 3: **The anatomy of a quest.**

COMPONENT	DESCRIPTION
Trigger Start	Activates the quest
Sprite	Shows where the quest can be started
Giver	The NPC who is giving the quest
Scene Start	The dialogue explaining the quest
Sound Start	An audio cue to notify the player of the start of the quest
Text Start	A text cue with a brief description of the quest
Relay Content	A relay that activates the main parts of the quest, such as npc_spawners
<Quest Components>	<This is the main operational part of the quest and will trigger the Relay Complete when finished>
Relay Complete	Enables the next set of entities for the following quest
Text End	Text that explains how to end the quest
Sprite	Reactivated to show which NPC to talk to
Trigger End	The trigger to activate the end of the quest
Scene End	The ending dialogue
Sound End	Audio cue that will signal the end of the quest
<Next Quest Activation>	<If there is a next quest it is activated here>

Now we're going to add some dialogue. Find logic_choreographed_scene in the entity list and place it next to our NPC. We're going to need at least two of these for each quest, one for the start dialogue and one for the end dialogue, so copy and paste a second one above the first. Name one scene_start and the other scene_end. We won't add the actual dialogue just yet, but we'll cover that soon.

We will also need an audio notification that the player has accepted the quest and that it's begun. Create an ambient_generic entity, place it near your NPC, and name it sound_start. If you don't already have one you want to use, try using elevbell1.wav by typing "plats\elevbell1.wav" into the Sound Name field. Then go to the Flags tab and make sure all three options are checked.

After that we're going to add the text that explains the quest to the instance. Add two separate game_text entities and name them "text_start" and "text_end." The Hold Time for both will need to be a high value, like 99999, so that the message won't disappear until we kill that entity. We also want the text to be off to the side of the screen where it won't obstruct the player's view, so set the X and Y fields to 0.1 for both game_text entities. Text_start will be the message that displays during the quest and text_end will be the message that tells the player to return to the quest giver or a certain area.

We're going to need some relays to keep our triggers organized. A relay is an entity that activates other entities when it gets triggered. They're used by level designers to keep triggered events that occur at the same time together in one place, and they can easily be enabled and disabled. Create two logic_relay entities and place them near your NPC. Name them "relay_content" and "relay_complete." These will be the relays that activate all of the entities in the quest. The specifics of the actual quest are up to you to create for the player to complete. You'll need to set up a condition for the player to meet, like "get to a specific location" or "collect a certain number of objects." You can use entities such as math_counters, or brush entities such as trigger_once to check whether the condition has been met. If those entities find that the condition has been met, they will then trigger the relay_complete entity that we've set up. The relay_complete will then trigger all the entities needed to run the end of the quest. Make sure that "relay_complete" is triggered when your quest section is completed.

Now we need the audio notification for the end of the level. Add another ambient_generic entity

Figure 4: **Our basic quest instance template.**

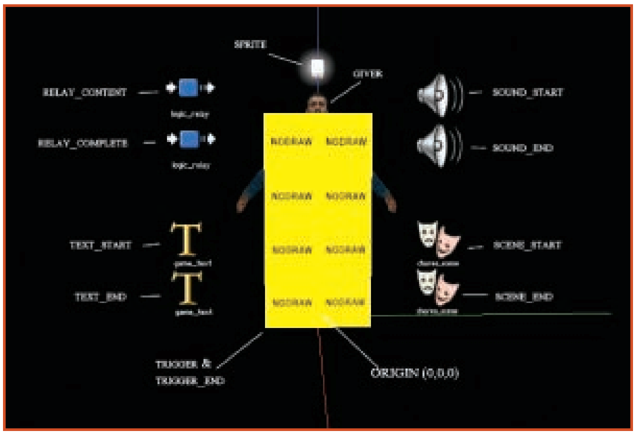
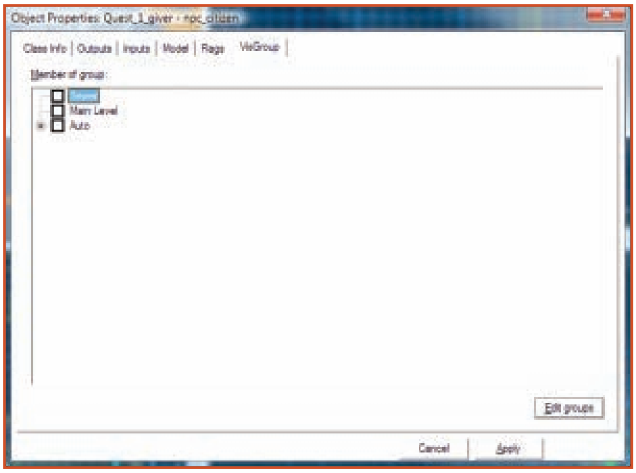


Figure 5: Our trigger listing.

ENTITY SELECTED	MY OUTPUT NAMED	TARGET ENTITIES NAMED	VIA THIS INPUT
trigger	OnPressed	scene_start	Start
	OnPressed	trigger	Kill
scene_start	OnCompletion	relay_content	Trigger
relay_content	OnTrigger	**[Quest entities]**	**[Functions]**
	OnTrigger	sound_start	PlaySound
	OnTrigger	relay_content	Trigger
	OnTrigger	sprite	ToggleSprite
[End of quest]	**[Completed]**	relay_complete	Trigger
relay_complete	OnTrigger	trigger_complete	Enable
	OnTrigger	sound_complete	PlaySound
	OnTrigger	text_start	Kill
	OnTrigger	text_end	Display
	OnTrigger	sprite	ToggleSprite
trigger_complete	OnTrigger	scene_end	Start
	OnTrigger	trigger_complete	Kill
	OnTrigger	level_quest_ complete	Display
scene_end	OnCompletion	**[Next quest]**	Enable
	OnCompletion	sprite	ToggleSprite

Figure 6: Organize Visible Groups in the Object Properties window.



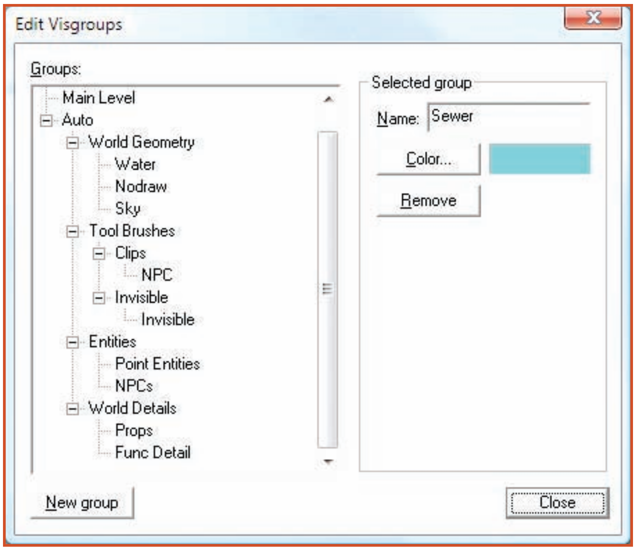
and name it “sound_complete.” Set it to a sound that indicates the quest has been completed, such as a bell or cash register. To keep it simple we’ll use “plats\elevbell1.wav” again for now.

We’ll need to add one more trigger in front of the NPC. This will be the same as the starting trigger we made earlier, so just copy and paste trigger_start. Name this new trigger “trigger_end” and make sure that Starts locked is checked in the Flags tab. Make sure both triggers are not overlapping each other at all and that trigger_start is in front by using your top and side viewports to place them. Also, make sure that they are in front of the NPC’s hitbox, or else the game cannot tell if you are selecting the trigger or button. To check the hitbox location, just select your NPC and a yellow wireframe bounding box should appear around it.

Let’s set up all of the triggers listed in **Figure 5**. For now, you don’t have to worry about the triggers in [brackets] because we’ll add them later. Select the listed entities, open their Object Properties, go to the Outputs tab, and add the designated outputs to each entity using the table below for reference. Be sure to hit Apply to save each output!

That’s how we set up our quest instance. Now we can take that instance and put it into different places around our map. Right now, our instance is essentially a basic quest template that we can easily modify and add new elements to.

Figure 7: Assign brushes and entities to Visible Groups.



PLACING INSTANCES

/// Now that we have our instance set up, let’s add copies of it at different points in our main map so we can start making our actual quests. Place a func_instance where you want your first quest to start. Open up its Object Properties window and under VMF Filename browse for your quest_instance map file. (Note that browsing can be a bit buggy, so you may have to type in the relative file path to “quest_instance” yourself.) You can now copy and paste these entities around your level. Go ahead and duplicate the instance until you’ve got as many quests as you want.

Select your first quest instance and collapse it by going to Instancing, Collapse, Selection. Hammer will then import all the entities and give them a prefix, normally “AutoInstance<number>-”. Everything will be selected when it has been collapsed, so we’re going to organize them into VisGroups to make the quest-building process easier. Bring up the Object Properties window and click on the VisGroup tab. You should see something like **Figure 6**.

Don’t worry about the “sewer” and “main level” entries—I’m using them in my own map to break up the space, but you don’t have to have them in yours. Hit the Edit groups button and you’ll see the window shown in **Figure 7**.

From here you can create new VisGroups to use in your own map. Press the New group button and you’ll see a new entry on the list. Click on it and then rename it using the name field to Quest_n, where n is the number of your quest. Click Close after you’ve done that and you should see the new VisGroup appear as a new option on the main list. Check the box to add entities to that group, hit Apply, and click Close. You can now hide/unhide that group by checking or unchecking it in the VisGroup control panel, as seen in **Figure 8**. Do this for the rest of your quests as well.

IMPORTING YOUR OWN AUDIO

/// Our RPG is going to be extremely boring without any recorded dialogue to convey the story, so we’re going to use Audacity to record your own dialogue. You must have a microphone for this part. Due to the way the Face Poser application works, you're going to want to record one sentence at a time. Recording audio in Audacity is easy. All you need to do is get your mic set up correctly, hit the red button to record, and save your audio as a .wav file when you're finished. If you have any problems, Audacity's help menus have lots of useful information. When you're done recording your dialogue, create a folder named "RPG" and save each sentence as a .wav file into it. Move that RPG folder to C:/Program Files/Steam/steamapps/<your username>/half-life episode two/ep2/sounds/. It's important to save the audio into a folder one level above the sounds folder because the engine will not read the sound file otherwise.

Next, let's create a sound_script file so that the engine will be able to reference our sounds in the editors and the game. Launch GCFScape, go to File, Open, go over to your steamapps folder, and open up the "episode two content.gcf" file. After GCFScape has loaded that file, we'll need to find a file named "game_sounds_manifest.txt," which is located in the /steamapps/<your username>/ep2/scripts/ folder. Right-click on the file and click Extract, and place the manifest within /steamapps/<your username>/half-life 2 episode two/ep2/scripts. We're done with GCFScape, so you can close it now.

Go to the scripts folder and open up the game_sounds_manifest.txt file we just extracted. In this text file we can see all of the sound scripts that the game will use when it loads, and we need to add our own to this list. Under the last line of code, within the last bracket, add your own entry like **Listing 1**.

Listing 1

```
//RPG FILES
     "PRECACHE_FILE"                    "SCRIPTS/RPG_SOUNDS.TXT"
```

Now we're going to create the "rpg_sounds.txt" file that we are referencing, so save the manifest file and close it now.

In the scripts folder, right-click and select New, Text Document. Rename this new text file to "rpg_sounds.txt." After you rename the text file, open it up and add the code in **Listing 2**.

Listing 2

```
"RPG_SOUNDS.FEMALE_01"
{
     CHANNEL   CHAN_AUTO
     VOLUME         VOL_NORM
     PITCH          PITCH_NORM
     SOUNDLEVEL     SNDLVL_NORM

"WAVE"         "CITIZENHOMIE/FEMALECITIZEN_LINE1-TRY2 - PROCESSED.WAV"
}
```

The first line is the name of the sound that will appear in Face Poser: "rpg _sounds" is the prefix, and the second part is the name. You can change this naming convention to whatever you like. The next four lines tell the engine how to deal with the sounds, and we'll leave them as they are for now. The last line is the .wav file it will load and contains a relative path to the "sound" folder, located in /steamapps/<your username>/half-life 2 episode two/ep2/sound/. All you need to do is put the folder with your audio files into that "sound" folder and make sure the file names and extensions match what it says in the script file. Save and close the text file when you are done.

If you're running Windows Vista or Windows 7, we'll need to help Face Poser out a little bit. As mentioned earlier, you'll need to download and install the Phoneme Extractor patch in order to do this. Follow the instructions and replace the necessary files. This patch will allow Face Poser to correctly translate the audio files and sound scripts into the phoneme facial expressions that will be animated on the characters. It's important to note that Source SDK refreshes itself each time it is launched, so this patch fix needs to be done each time Source SDK is opened. Yes, it is a bit of a pain, but it's way easier and faster than trying to line up phonemes to the audio manually.

With Face Poser open, we're going to start a new scene. Under the Choreography menu tab click "New," which will create a new scene to work with. It'll prompt you to save the scene, so name it something you can remember, and then place it in /steamapps/<your username>/half-life episode two/ep2/Scenes/RPG_MOD/. It'll make it easier to find later on if you give the scene name a prefix, such as "RPG_". Next, Face Poser will ask you to name your first Actor. Name your actor "!_target<n>" where <n> is a number between 1 and 8, because we can have up to eight individual NPCs in each scene. Hammer will use whichever NPC you select in the editor based on this name. We currently only have one character, so name the actor "!_target1." Right-click on the Actor name that has appeared in the Choreography window and then go to New, Channel to bring up the Create Channel box. Name this channel "Audio." We'll need to make a second channel for animations, so repeat the last step and name it "Anim."

Now we're going to sort out the audio. In the list of tabs at the bottom of the Face Poser screen, double-click Phoneme Editor and it will bring up a new window. Click on Load and select the .wav file you want to add to the character. The file will be displayed as a waveform diagram in the Phoneme Editor window after it is loaded. Press the Re-extract button, and Face Poser will prompt us to write out the sentence text of the same dialogue audio clip we selected. Write out the line and then press OK. The words you entered should now be placed at the appropriate points over the waveform, so right-click on the waveform and hit Commit extraction to apply the phonemes to your waveform. If you play it by pressing your space bar you should see the Actor in the window move their mouth. Hit Save on the Phoneme Editor window and then close it.

Figure 8: **Toggling Visible Groups.**

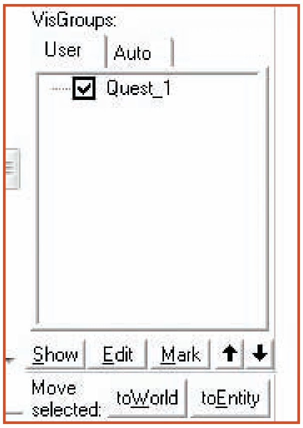
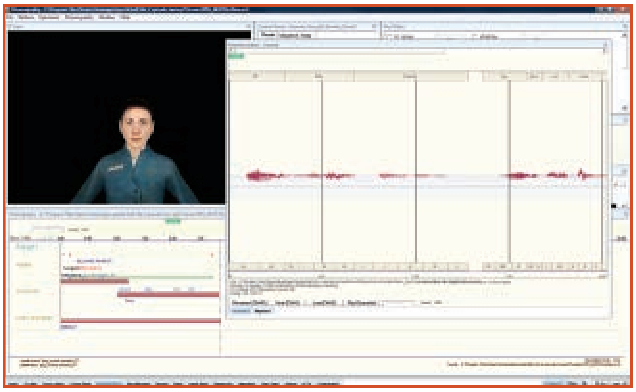


Figure 9: **Once your dialogue animation is set up in Choreography, it will look like this.**



We need to add that .wav file to our actor, so right-click on the Audio channel and click WAV File, and then check Show All Sounds. We need to locate the sound name that we gave our .wav file in the sound script file; in our example it will be named "CitizenHomie/FemaleCitizen_Line1-try2 - Processed.wav," but you should look for whatever name you added into your own sound script. You can use the filter field to help you find it. Once you find it, select it and hit OK. This will place it on your Audio channel. When you play through the sound, it should make your actor's mouth animate to speak the dialogue. Look at **Figure 9** for an example of what everything should look like once you're done.

If you wanted to add gestures, facial expressions, and other animations, you'd do that now. We won't cover that in this tutorial, but there are plenty of resources online that explain how to do those things, such as design3's Source Engine section **(6)**. Next save your scene by going to Choreography, Save. Now that we have our scene saved, we're ready to attach it to our NPC.

APPLYING CUSTOM AUDIO TO NPCS

/// Now we need to set our dialogue up in our level. We're going to add our scene to the scene_start and scene_end entities we created while building our instance earlier. Hit Alt + Enter to open up the Object Properties window of the scene_start entity and you'll see an option named "Scene file." Select this option and then browse for your scene under the "scenes" folder. We'll need to assign an NPC to use the scene, so select the Target 1 option and choose the NPC with the name of your actor in Face Poser. Hit Apply to save these changes. Then repeat that same process for the scene_end entity.

All we need to do now is trigger the scene to start playing. We'll need to trigger the scene from another entity at the point where we want the scene to be acted out. For example, you can use a trigger_once brush with the output of "On trigger, entities named scene_start, Start." Our quest instance that we set up earlier has two button brushes that should already start our scenes, so just double-click on the brush to open up the Properties panel and change the Scene File field to the scene you want your NPC to act out.

TRIGGERING EVENTS

/// In order for this RPG to work, we need a system that allows each quest to know when the player has completed a specified task. This is so the game can progress and won't just stop dead in its tracks. The best thing to do is have a single entity that performs all the necessary outputs once the quest is completed. We'll use our relay_complete entity for this task.

Here's an example of a quest:

- ♦ Player needs to kill five enemies.
- ♦ On each kill, a math_counter entity increases by one.
- ♦ When it hits its max limit, five in this example, it outputs a trigger to our logic_relay.
- ♦ logic_relay runs all necessary outputs like stopping enemies spawning, activating the quest giver, activating sprites, and so on.

Using relays keeps the number of outputs low and in one place instead of spread across multiple entities, which can get very confusing on complex maps.

SETTING UP AUTOSAVE

/// Everybody hates it when a single crash erases their last hour of gameplay, so it's vital to have an autosave feature just in case it happens to our players. Fortunately, autosave is easy to implement. All we need is a logic_autosave entity. Find it on the entity list and place it in your map near any other logic entities you have, such as environmental lighting or auto_logic. The placement isn't too important, but it's good to have similar entities located close to together to make them easier to find. Let's name it "Global_autosave." The autosave entity works off inputs from other entities, so we're going to make our end relays for each quest output a trigger to our autosave in order to save the game. Open up the Outputs tab on our relay_complete and add this output: "On Trigger" the entities named "Global_autosave" via this input "Save." And that's all there is to it! Now the game should save the player's progress every time we finish a quest.

GAME ENDING

/// Our game is going to have to end at some point, and it'll look bad if it just cuts back to the main menu, so we're going to use another group of entities to roll the credits first. Create the entities shown in **Figure 10** and place them in a suitable place in the level, somewhere that is easy to access and remember.

Once you've made the core entities for the ending credits, all you need to do is link them with triggers. Open up the output tab on your end_relay and create these new outputs: On trigger, env_fade, fade, 0 seconds On trigger, end_text, display, 1 seconds On trigger, point_clientcommand, disconnect, 5 seconds This will make the screen fade to black, your credits will fade in, and then a few seconds later the game will disconnect the player using the client command.

Now you just need to compile your map. Compiling is the process of packaging everything and making a .BSP file that the game can read. To compile, just go to File, Run Map. This will bring up the Run Map options window with some different compile options, such as launching the game after the compile finishes. Just hit OK to compile the map and you're good to go!

QUEST COMPLETE

/// Now you should have the basic tools you need to build your own RPG prototype. We've shown you how to create quests, incorporate your own dialogue, and use logic to enhance your game, so now it's your job to keep testing and refining those quests, level, and dialogue decisions. You can make any sort of quest you can possibly imagine, or add a multiplayer co-op component like the Synergy mod, or even use elements from other Valve games like Left 4 Dead. Get in there and start modding!

Ben Evans *is a game developer with experience in level design, 3D art, and coding. He's a frequent design3 contributor, especially when it comes to Source engine tutorials. He is also the current lead programmer at Stupor Studios and can be found on Twitter as @The_BenEvans.*

Special thanks to Ross Scott, the director of the Machinima series Civil Protection *and* Freeman's Mind, *for providing dialogue writing and audio recording.*

Figure 10: **You'll need to use these entities to make your ending credits sequence.**

ENTITY NAME	NAME TO GIVE	FIELDS TO CHANGE
point_clientcommand	end_client_command	n/a
env_fade	end_fade	n/a
logic_relay	end_relay	n/a
game_text	end_text	Message: Created by <Your name> Hold time : 100

resources

1. Valve Developer Wiki https://developer.valvesoftware.com/wiki/Main_Page

2. Design3 http://www.design3.com/

3. Audacity http://audacity.sourceforge.net

4. Phoneme Extractor patch www.moddb.com/members/varsity/downloads/windows-vista7-phoneme-extractor-13

5. GCFScape http://nemesis.thewavelength.net/index.php?p=26

6. Design3 Source Tutorial design3.com/training-center/engines-sdks/source

COMPETITIVE SPIRIT

Indie game competitions can be a useful resource for student developers. Events such as the Independent Games Festival and showcases like the PAX 10 can help rookie developers gain exposure and raise awareness about their games. These events also help incentivize efficient development and timely release windows by forcing teams to meet deadlines and produce finished games.

IGF Student Showcase Competition

www.igf.com

Entry Requirements: Open to all student developers worldwide. Entrants must be at least 13 years old Entry Fee: None Organizers, Sponsors: UBM TechWeb

» As part of GDC's Independent Games Festival, the IGF Student Showcase encourages students to submit their games or mods for judging by a panel of roughly 60 leading independent and professional developers. The winning titles join the other games featured at the IGF Pavilion on the GDC show floor, and each team receives $500 to help fund a trip to the show. At the Independent Games Festival Choice Awards, one of these finalists will win a $3,000 prize for Best Student Game. IGF Student Showcase finalists from 2012 include The Snowfield, Nous, Way, and One and One Story.

Dream-Build-Play Challenge

www.dreambuildplay.com

Entry Requirements: Open to individuals who are at least 18 years old and who are legal residents in their country of entry Entry Fee: None Organizers, Sponsors: Microsoft

» The Microsoft-hosted Dream.Build.Play Challenge tasks entrants with creating a high-quality title using XNA Game Studio 4.0. Winners of the competition split a prize pool of $75,000 and secure a publishing contract and exposure for their game on the Xbox Live Marketplace.

Sense of Wonder Night

www.expo.nikkeibp.co.jp/tgs/2012/exhibition/english/exhibit/sense.html

Entry Requirements: The game must invoke a sense of wonder! Entry Fee: None Organizers, Sponsors: Computer Entertainment Suppliers Association (CESA), Nikkei Business Publications, Inc., IGDA Japan

» The Tokyo Game Show's Sense of Wonder Night provides developers with an opportunity to present new and experimental game ideas to an audience of professionals. Winning applicants get 10 minutes to present their idea or prototype to an audience at the show and can exhibit their project at the SOWN Pavilion on the TGS show floor.

Indie Game Challenge

www.indiegamechallenge.com

Entry Requirements: Open to individuals or teams worldwide (except to those located in U.S.-embargoed countries) Entry Fee: $100 (fee is waived for residents of certain states in the U.S.) Organizers, Sponsors: The Academy of Interactive Arts and Sciences; GameStop, Inc.; The Guildhall at Southern Methodist University

» This game development competition, hosted alongside the Interactive Achievement Awards at D.I.C.E., allows student developers to compete for up to $100,000 in prizes. Finalists in the competition receive $500 to help pay for travel to the awards ceremony, where the challenge's grand prize winner is announced. These finalists also receive the opportunity to talk to a number of game companies about potential publishing deals.

indiecade

www.indiecade.com

Entry Requirements: Submitted games must not have funding from a major publisher Entry Fee: Processing fees range from $35 to $70 and are based on time of entry Organizers, Sponsors: Creative Media Collaborative

» IndieCade offers developers a chance to gain exposure for their games by granting winners a spot in the IndieCade Festival showcase in Culver City, California. The games shown at IndieCade are selected by a panel of over 160 jurors, and the show welcomes indie titles of all types, including exotic genres such as virtual toys and Alternate Reality Games (ARGs). Entrants may submit works in progress, but all games should include at least one finished, playable level.

PAX 10

www.prime.paxsite.com/pax10.php

Entry Requirements: Open to all independent developers without a traditional publisher-developer relationship in place Entry Fee: $50 Organizers, Sponsors: Penny Arcade

» The PAX 10 showcase at both the Seattle and Boston shows allows indie developers to demo their titles on the PAX show floor. These developers receive exposure for their work and get the chance to interact with the show's attendees.

Independent Propeller Awards

www.indiepubgames.com/contest.php

Entry Requirements: Open to all applicants more than 13 years old Entry Fee: None Organizers, Sponsors: indiePub, Zoo Games, South By Southwest (SXSW)

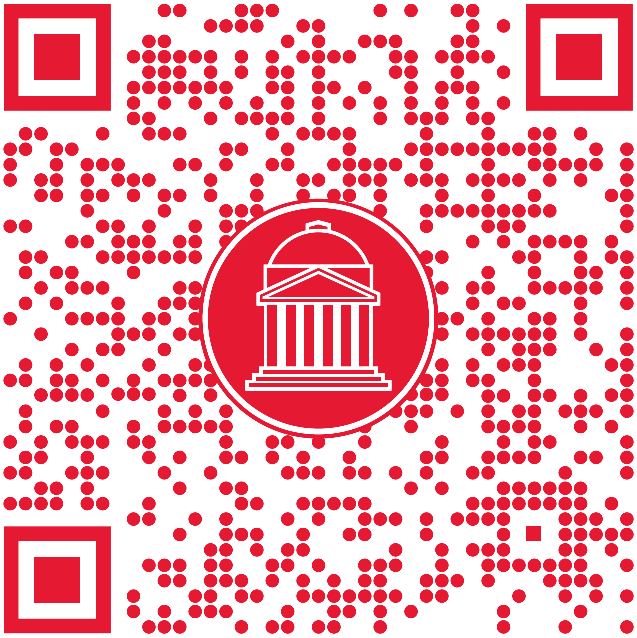
» The Independent Propeller Awards, presented by the indiePub community, offers indie developers the chance to win a $50,000 grand prize, with $25,000 going to the winners of four subcategories: best art, audio, design, and technical excellence. The awards are held annually at South by Southwest Interactive's ScreenBurn, the conference's video game-based track.

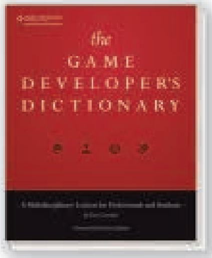
G.I.R.L. (Gamers in Real Life)

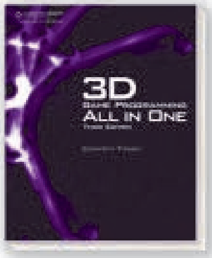
www.soe.com/girl

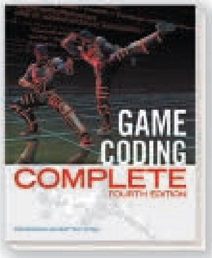
Entry Requirements: Students over 18 enrolled in a game-related program—check site for specifics Entry Fee: None Organizers, Sponsors: Sony Online Entertainment

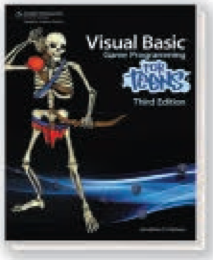
» The G.I.R.L. Scholarship Program is a different kind of competition—instead of submitting a game for review, students in game-related fields (including art, design, animation, programming, visual effects, and so on) can submit a piece of concept art or 3D art, and a short essay on how the applicant can help make games that more women will want to play, for a shot at a $10,000 scholarship and a 10-week internship with Sony Online Entertainment in San Diego. The concept art is usually tied to a Sony Online game (the 2012 competition had entrants submit art related to Planetside 2 and EverQuest II), and the application deadline usually falls in early March.

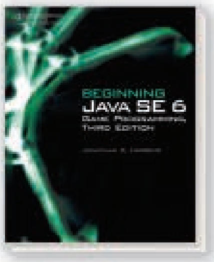


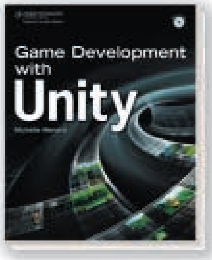


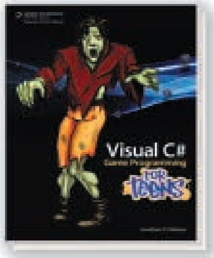


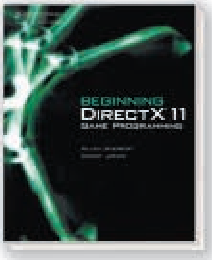


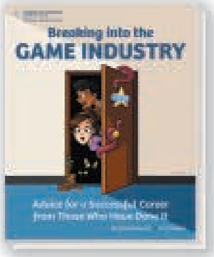


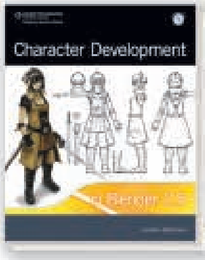








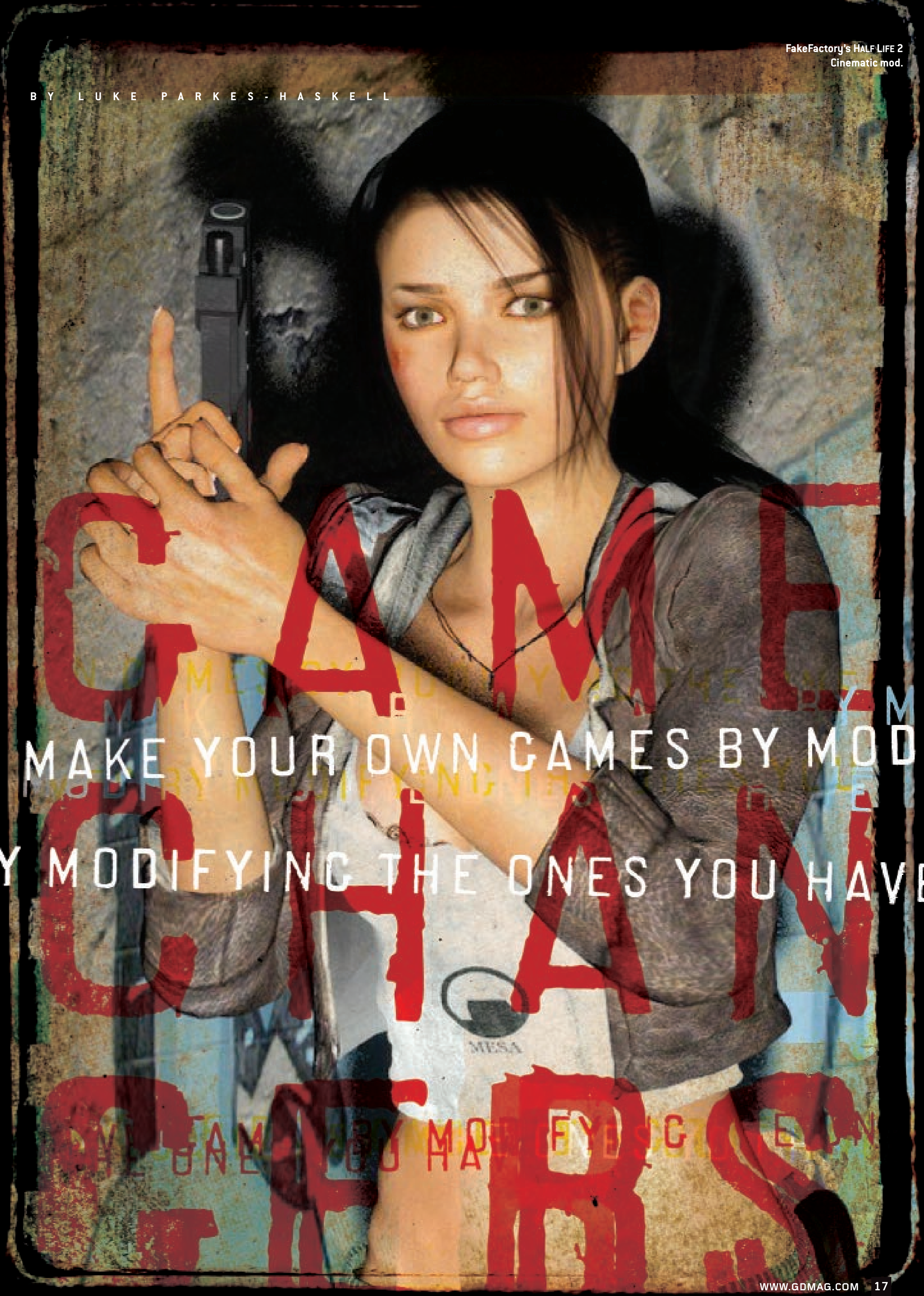




BY LUKE PARKES-HASKELL

GAME CHANGERS

MAKE YOUR OWN GAMES BY MODIFYING THE ONES YOU HAVE



FakeFactory's HALF LIFE 2 Cinematic mod.

GAMECHA

Have you ever wanted to make changes to a game while you were playing it? Perhaps you thought your weapons were too weak, or you were frustrated by your incompetent AI teammates. Maybe you simply weren't satisfied with something more fundamental. Well, you're not alone—there are hundreds of thousands of players out there who like to make modifications ("mods") to the games they own, ranging from humble subtle changes and bug fixes to complete overhauls that leave very little of the original game intact. Modding isn't just fun and games, though! The skills you learn while making a game mod can be the beginning of your career in game development. In this article, we're going to walk you through the basic information about modding games; we'll discuss modding history, popular games and engines, and a few stories of modders-turned-developers—to encourage you to start making your own.

WHY MOD?

/// If you're interested in making your own games, building a few mods is a great way to get your feet wet. When you're trying to make a game from scratch, it's easy to get stalled early on—especially if you're working by yourself and don't have the skills or time to do all the art, sound, coding, designing, and testing all by yourself. On the other hand, if you decide to modify an existing game, you already have an excellent base to start from, and you get to jump straight to the game-making process.

Also, modding can also be a good way to start building your portfolio for when you're ready to start looking for work as a game developer, since you can start practicing your level design, 2D and 3D art skills, sound design, and other game development disciplines without having to make a game from scratch first.

A BRIEF HISTORY OF MODS

/// Game mods have been around for nearly as long as games themselves have; Steve Russell's Spacewar! (1962) had plenty of third-party additions, including cloaking generators, different weapons, and even a first-person perspective. Ms. Pac-Man started out as a third-party Pac-Man conversion kit called Crazy Otto. And one of the earliest games to feature after-market mod support was actually Microsoft Flight Simulator 4.0's Aircraft and Scenery Designer expansion pack (1990).

Most of modern modding activity can trace itself back to id Software's Wolfenstein 3D release in 1992; its codebase was particularly easy for enthusiasts to play with, so many people made new level packs that extended the life of the game. Id noticed this and built Doom with deliberate flexibility in mind by separating the code, media, and levels into modular .wad packages to make it easier for modders to distribute their changes, and would continue to encourage modders to work with the Quake series by including modding tools and eventually making their engines open-source.

Since then, we've seen many major games grow from humble game-mod origins into full-fledged triple-A titles—or in some cases, entirely new genres. Counter-Strike, one of the most legendary online games in history and the forefather of practically every realistic FPS out there, started out as a Half-Life mod. League of Legends, Heroes of Newerth, and other games in the emerging multiplayer online battleground (MOBA) genre all share one common ancestor: Warcraft III mod Defense of the Ancients. If you have a great idea for a mod, you might find that it can eclipse the original game you modified!

THE BIG THREE

UNREAL ENGINE

[EPIC GAMES]

/// The Unreal Engine is ubiquitous. Its current incarnation (Unreal Engine 3, at the time of this writing) powers more than 200 titles, and it's a rare week when an Unreal Engine game is not

Unreal Tournament 3.

NCERS

Counter-Strike: Global Offensive.

present in the current charts. However, most of these games cannot be edited by the mod community (in fact, only Unreal Tournament 3 boasts complete mod support, although some other games have level editors available).

The Unreal Engine has been around in one form or another for nearly 15 years, and with this longevity comes a wealth of documentation and a community that is both uniquely helpful and uniquely creative. Unreal boasts a wide selection of proven industry standard tools, but does have its drawbacks; you can't access the Unreal Engine code at any level other than UnrealScript, which is enough to change gameplay mechanics but not really powerful enough to build completely new features or integrate external libraries (without a lot of extra work, at least).

Nowadays, many modders are using Epic's Unreal Development Kit (UDK), a more up-to-date development platform for the Unreal Engine that is free for noncommercial use and comes with indie licensing terms and multiplatform support. (For more on the Unreal Development Kit, read our hands-on tutorial on making a game with the UDK on page 23 of this issue.) For designers or those who just want to mess around with levels and concepts though, working with Unreal Tournament 3 can be a good way to see the nuts and bolts of both the game and engine in action before moving over to the UDK.

Picking a Game to Mod

HOW DO YOU DECIDE WHAT GAME YOU WANT TO START MODDING? There are so many moddable games out there that it can be tough to choose. Here are three steps this modder would follow:

• Look at your game shelf. There's no need to go out of your way to acquire a moddable game if you're interested in dabbling, since chances are you probably own several anyway—unless you're not much of a PC gamer. Notably it tends to be the slightly older games (late 1990s onward) that are both easy to work with and well documented. Some older games may have once been well documented, but information is all too easily lost to the depths of the web.

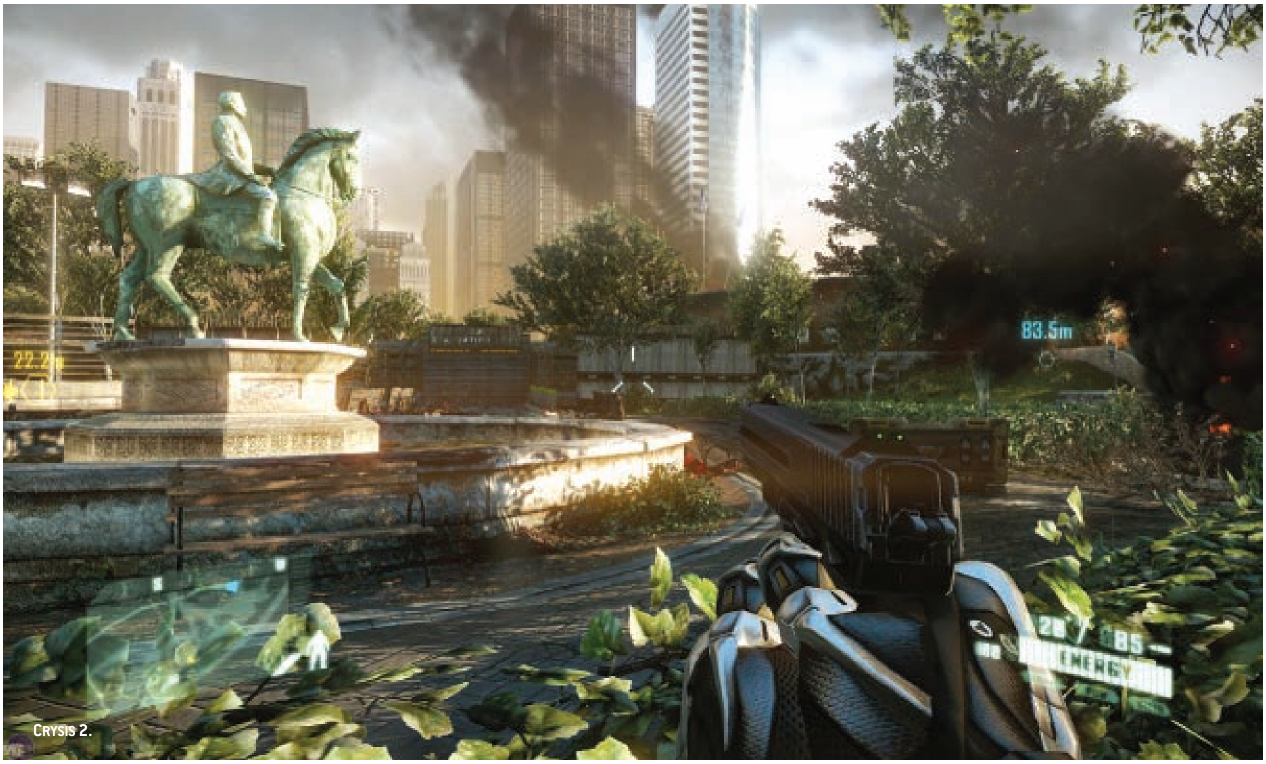
• Do an internet search. If there is any information or developer tools to be found for your games, they'll be found relatively easily on the internet, often alongside community web sites which usually have at least a small number of helpful souls around.

• Check out existing mods and see what other people are doing. Web sites like ModDB (www.moddb.com) have a very large and dedicated modding community who keep a fairly comprehensive database of mods. It's worth searching through it to see what kinds of mods have already been achieved for the games you have at hand. If there aren't many, it may be worth asking in the forums why this may be the case and just what is possible.

MODDABLE GAMES: Unreal Tournament 3, Roboblitz LEVEL EDITOR-ONLY: Gears of War, Mirror's Edge (unsupported)

Source Engine
[Valve Software]

/// Valve's Source Engine is by far the most long-running and popular choice with mod-makers. The engine has impressively powered 16 of Valve's own projects since its 2004 introduction



Crysis 2.

as the engine behind Half-Life 2. In order to make a mod with the Source Engine, you don't need to own a specific game (unless you want to use specific content from a game)—it's free to all Steam users.

While the Source Engine's large community and excellent documentation make it a great place to start, it can be notoriously awkward to work with thanks to its rather dated tool set and relatively weak asset pipeline. There are some fantastic modern works using Source, but the tools hinder more than they help. And since Source is feeling a little dated, we wouldn't be surprised if Valve had a new version in the works.

MODDABLE GAMES: Just about any game that uses the Source Engine: Half-Life 2, Left 4 Dead, Portal, and so on.

CRYENGINE

(CRYTEK)

/// CryTek's CryEngine (used first in Far Cry) is newer than both Source and Unreal, and has a smaller community base, but its latest incarnation (version 3, as of this writing) is every bit a competent rival. During Far Cry's release days, CryEngine wasn't a popular tool with mainstream modders because the tools weren't well supported, but after Crysis was released, CryTek made sure to offer a solid set of tools.

The Crysis mod community is similar to the Unreal Tournament 3 modding community insofar as most Crysis modders have started working with the CryEngine 3 SDK, which is offered with comparable terms to those of UDK (although notably lacking the UDK's initial royalty-free $50,000 for commercial software). CryEngine 3 has superior landscaping tools and access to reasonably flexible C++ game code, but Unreal offers strong baked lighting and an excellent material editor. Unfortunately, the free CryEngine 3 SDK currently has some DRM restrictions in place that prevent anyone but you from opening levels you're working on, which makes it hard to collaborate with others.

GAMES: Crysis, Crysis 2

/// This article has mostly focused on the big first-person shooter releases—but these are not the only games that can be modded (nor the only shooters). They are simply the games that have proven to be some of the most popular and retain active communities to this day. Real-time strategy games are another popular target for game mods; the Command and Conquer and Total War series have fairly robust modding communities, and Starcraft 2's map editor is so powerful that enterprising modders have used it to make all kinds of games, from Tetris clones to first-person shooters.

TESTIMONIALS: FROM MODDER TO PRO

LUKE PARKES-HASKELL

(Microsoft Game Studios)

"I took an early interest in games, as did many who work in the industry, but it wasn't until Creatures 2 that I got interested in mods. It was the norm for players of that game to share not only their creature files, but also scriptable objects known as COBs. It's a lesser-known community today, without the grand history of Doom and the like, but at its peak was very large for the time. I also found myself tinkering with Command and Conquer: Tiberian Sun, editing the units and creating new levels, and playing around with Quake III and its source code during high school. My departure for university almost went in a completely different direction; I had initially chosen to work in aerospace and engineering, but instead elected to take a newly introduced course in computer game programming at the University of Derby.

Command and Conquer: Tiberian Sun.

NCERS

I spent two years there under a fairly challenging regimen that introduced me to some of the higher-level concepts to do with rendering pipelines, though at the time I was more interested in gameplay and much of it proved difficult to grasp under the accelerated pace with which it was delivered. Nevertheless, it dramatically improved my grasp of "how games work," especially after I attempted to write my own crude software renderer as part of an assignment. That course also introduced me to Unreal Tournament 2004, which I tinkered with during the summer prior to my supposed placement year. That tinkering landed me a job working with an architectural visualization firm in London as the sole programmer dealing with a very early Unreal Engine 3. This is where I really learned to program and earned my wings—left to my own devices with a partially documented and very much more complex solution than I'd ever had to deal with before.

I attempted to return to university for my third and final year, but family things didn't quite work out, so I had to start picking up assorted freelance work to keep me afloat. That work was so time-consuming I couldn't complete the course. I eventually bagged enough money to head to a new university (Central Lancashire) and took full advantage of the availability of a new student loan scheme. I never did stop working with Unreal during this time, and I lent a hand to one or two mod teams at the time to develop my skills with the engine. Two years later, Microsoft Game Studios offered me a job working on Fable: The Journey, which I took (and put my degree on hold); I knew I'd be foolish to pass up the opportunity. It did cost me five years of my education, but it is undoubtedly the best move I could have made.

LUKE PARKES-HASKELL *is a programmer at Microsoft Game Studios and presently working on* Fable: The Journey.

CHRIS HOLDEN

(Illfonic)

"I was placed in a class for "gifted" students in school, despite not really being gifted. Since the NES had just been released, I tried to recreate the Super Mario Bros. experience. The computers we used in class didn't have enough graphical power to pull that off, but it did give me the bug for making games.

My most life-changing game experience came with the release of Doom and its robust community, readily available tools, and map-sharing via early internet dial-up BBSes. I began building levels, sprites, and gameplay mods immediately and it wasn't long before I had an entire episode of levels with new monsters, weapons, and so on.

After graduating from high school, I found myself with enough money to buy my own PC, and a friend pointed me in the direction of the latest thing to drop on the world—Quake Test. I took my first steps into 3D modeling with Quake Test just as the internet started coming into its own, and I soon found myself making skins for Quake clans, creating mods, and even running a Quake Skins and Total Conversions site on PlanetQuake.com.

QUAKE 2.

In 1997 I received a contract to work on a Quake 2 add-on, and one of the mods I worked on was to be included in the Quake II: Extremities commercial mod package. A game journalist knew that a company that needed an artist, and hit me up based on my earlier work. It was a surreal experience: I was brought in as "the professional" even though all I had done was make a few mods.

From there I moved to fulfill a dream along with a fellow animator and went to work on a PlayStation 2 game, then later went further with a GameCube and Xbox release of Goblin Commander. That same animator now runs his own studio and introduced me to educational games. Using the Serious Sam engine, we made a demo that was actually pretty sweet. Networking is really just another word for friendship.

After that project I went to Mythic to work on Warhammer Online: Age of Reckoning, and it was networking that helped to get my name in the basket (and a big thanks to 3D art site Polycount). Mythic was one of the best work experiences of my life because I got to work with so many good people, and it was full of Polycounters and kids from Full Sail. When the game shipped, the layoffs came with it, and I took a sabbatical before eventually taking a role as lead artist over at Illfonic. It's been great, and I've been working with top-level technology and traveling the world for shows and training.

CHRIS HOLDEN *is lead artist at Illfonic and recently released* Nexuiz. *http://chrisholden.net*

BENEDIKT PODLESNIGG

(Massive)

"I started out creating levels for Doom II using an editor I found on a shareware CD, and when Duke Nukem 3D was released I did a load of levels for that. I was always experimenting and trying to push things a little bit further than the given stuff.

Around the same time I also got Klik and Play, which let me make 2D games without any coding knowledge. I made a lot of small games with that.

My first experiences with proper 3D software came after I saw the first Toy Story in the cinema. I installed a version of 3D Studio MAX R3 about an hour after the movie had finished and started to play with it. The stuff I did was completely forgettable, but I certainly found the taste.

I later became quite involved in the German Half-Life modding scene, mostly doing weapon models and a few levels. The mod I was primarily working on had lots of potential but unfortunately failed as the coder jumped ship—which was pretty bad given that we were a four-man team. I continued contributing to mods and did a load of stuff here and there, but I never thought about being good enough to do it professionally.

I spent a year doing civil service (it's compulsory in Austria to spend either a year in the army or do civil service) and during that time didn't do any game art, but worked on my portfolio for the university in Vienna—which rejected me three times. I went on to study history of arts instead, purely out of interest. I had completely given up hope of getting into the game industry.

It was at university that I met my lovely wife, who pushed me toward opportunities as an artist. I found a small ad for a job doing game-spec buildings on a freelance basis, and took it on, working for the company for over a year. When I'd finished working there, I posted my work on CGTalk, where out of the blue I was contacted by an outsourcing company in Vienna. I started two weeks later. I spent two years there and worked on a load of interesting projects with a great team, but I always felt I wanted to get into development and learn more.

I moved to the UK and started working as an environment artist at Free Radical Design where I spent a year before it closed down, then I went to Bizarre Creations where I spent another two years before that too closed down. After that, I did a three-month stint at Rockstar North before going to Massive in Sweden to work as a level artist.

I've been working in the game industry now for seven years professionally, and I still enjoy it every day. I've been quite lucky to work with so many talented people—learning from them, teaching them in turn, and doing so many different things to improve myself.

BENEDIKT PODLESNIGG *is an environment artist and has worked on projects such as* Blur, James Bond: Bloodstone, *and* Star Wars: Battlefront 3. *www.zaplab.at/podlesnigg*

DAE
DIGITAL ARTS & ENTERTAINMENT

BY SJOERD DE JONG

TEST DRIVE THE UNREAL ENGINE

GET STARTED WITH THE BIGGEST GAME ENGINE IN THE BUSINESS

Batman: Arkham City is just one of the blockbuster games powered by the Unreal Engine.

/////// The Unreal Engine is the tech that drives Batman: Arkham City, Borderlands 2, Mass Effect 3, the Gears of War series, and several other major games—including yours, if you'd like it to. The Unreal Development Kit (UDK) is free for individual, noncommercial use, so just download it from www.udk.com/download (Windows PCs only), and I'll walk you through the basics of making a simple third-person game in Unreal. We'll start out with the basics of installing and setting up the engine itself, then we'll introduce UnrealScript and start building your first level.

STARTING WITH UNREALSCRIPT

/// Once you've downloaded and installed the UDK, navigate to the UDK install folder (the default folder at the time of this writing is C:/UDK/UDK-2012-05). You should see four folders in the UDK-2012-05 folder: Binaries, Development, Engine, and UDKGame. The Binaries folder contains the executables and all of the files that form the compiled engine. The Development folder contains the UnrealScript source code. The Engine folder contains the most essential content and configuration files. And the UDKGame folder contains everything else: compiled game code, levels, sounds, textures, and so on. You'll be doing most of your work in the Development and UDKGame directories—don't touch the files in Binaries and Engine for now.

Open the Development folder and then open the Src folder. You should find 16 folders within the Src folder—these folders contain your game's source code. Note that this is not the source code of the actual Unreal Engine itself, just the code that handles the gameplay in the game. (The engine source code is off-limits for UDK developers.)

From the Src folder, go to the UDKBase folder and open the Classes folder inside. There should be about 120 files in the Classes folder, each with a .uc extension. These files are UnrealScript source files, and they're just TXT files with a different extension, so you can open any of them in Notepad or your preferred text editor. Go ahead and look around for a minute.

Now you're going to write your own UnrealScript and set up an UnrealScript package. Navigate back to Development/Src and create a new folder. Name the folder "MyFirstGame." Inside MyFirstGame, make another folder named "Classes." Next, create a new empty text document inside Classes by right-clicking in the folder and going to New, Text Document. Name the new text document "MyGameInfo" and change the extension from ".txt" to ".uc". (If you don't see any file extensions, go to Control Panel, Folder Options, View, and uncheck Hide extensions for known file types). Open MyGameInfo.uc in Notepad, and type the following code:

```
class MyGameInfo extends UTDeathmatch;

defaultproperties
{
	PlayerControllerClass=class´MyPlayerController´
	DefaultPawnClass=class´MyPawn´

	Name="Default__MyGameInfo"
}
```

For our game, we're going to use MyGameInfo to hold the rules of the game. We won't go into building game rules for this tutorial, but once you're more comfortable with the UDK, you would put your win/loss conditions in here. For now, we're making our game adopt the rules found in UNREAL TOURNAMENT'S Deathmatch mode (specified in UTDeathmatch).

UnrealScript is somewhat similar to Java: Files always have a header/title at the very start of the file that identify the file by name (MyGameInfo) and specify the file's parent class (in this case, it's "UTDeathmatch"—go ahead and find the file UTDeathmatch.uc in the Development folder and see what it looks like). Children files take over all functionality of their parent files, and the parent of the parent, and so on, so you only have to write code for things that were not already covered at some point by one of the parent or grandparent files. Note that you must make sure that the file name matches the header inside the file exactly (without the extension); if you've named this file something besides MyGameInfo, you'll need to change the header, too.

A typical UnrealScript file has the actual UnrealScript code below the header, and a section called "defaultproperties" below that which contains the default settings for the class or actor that corresponds to your file. For this file there is no actual UnrealScript code present because we're just using this file to change a few default properties.

Make another text file in the Classes folder, name it "MyPlayerController.uc" and enter the following code in that file. Note that the bolded text is just describing what the code does—you don't need to type that in.

```
/**
 * MyPlayerController - This class allows players to control pawns.
 */
class MyPlayerController extends UTPlayerController;

/**
 * Called when the controller should possess a pawn in the world
 *
 * @param  inPawn       Pawn to possess
 * @param  bVehicleTransition  If true, then the controller is actually
possessing a vehicle

event Possess(Pawn inPawn, bool bVehicleTransition)
{
 // Call the parent versions of this function
 Super.Possess(inPawn, bVehicleTransition);
 // Set the camera view mode to behind view, so that we are looking
behind the pawn
 SetBehindView(true);
}

// Default property block
defaultproperties
{
}
```

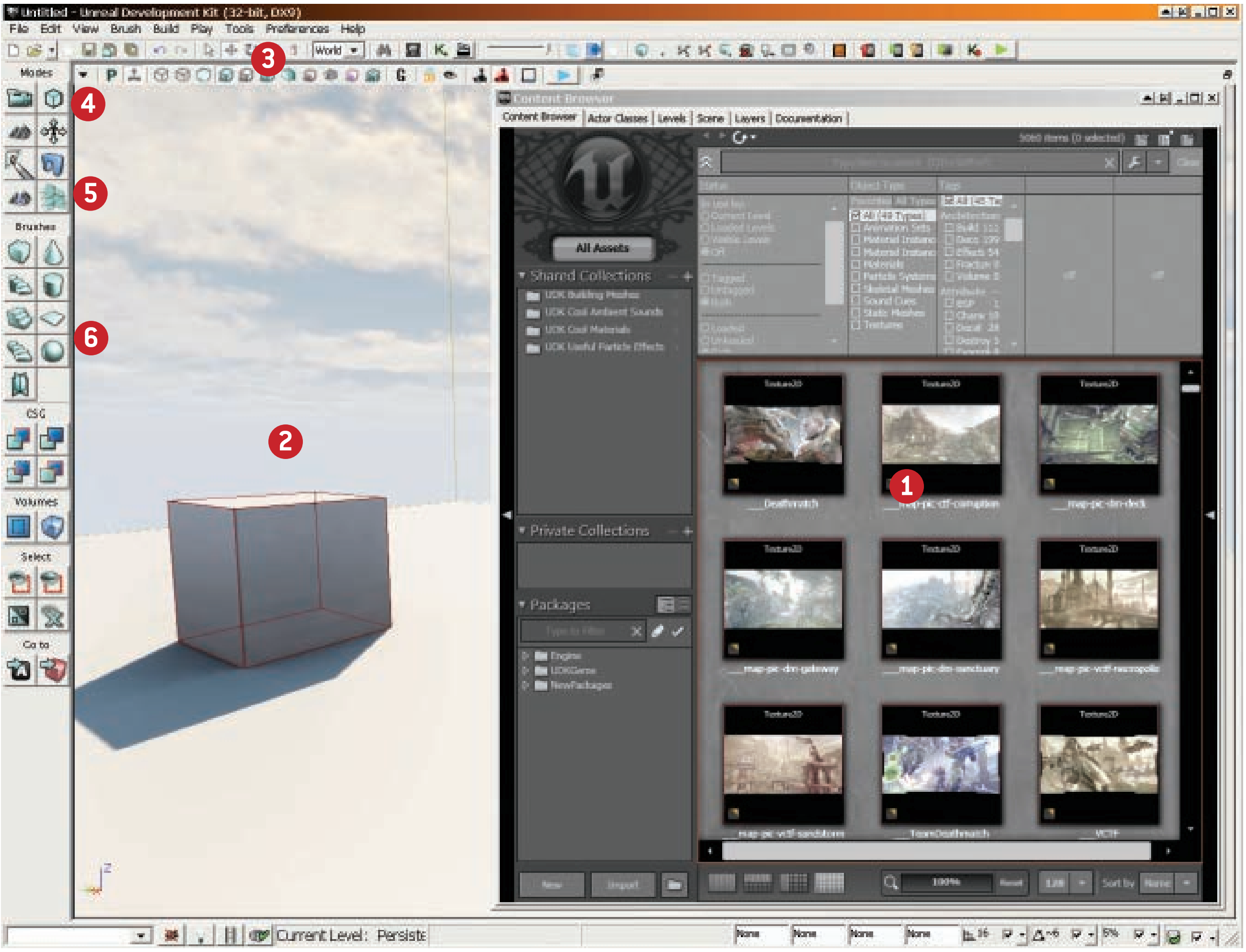
As you can see, we have a small amount of actual UnrealScript this time that specifies how you control and view the character.

Now make one more text file in the Classes folder, name it "MyPawn.uc" and enter the code in **Listing 1** (except the bolded comments). Once you're done with that, you'll have the absolute bare essentials required to start building a game with the UDK, but we need to compile it into something that the engine can use in the game and in the editor. In order to do this, we must tell the Unreal Engine that our new code package exists, which means we have to change the engine's configuration files. Here's how you do it.

First, go to UDKGame/Config, which contains all the .ini files that control how the game behaves. You can edit .ini files in a text editor, so go ahead and open one of them in Notepad.

LISTING 1

```
/**
 * MyPawn - This class is the physical representation of a player and is usually
controlled by a player controller or an AI
 */
class MyPawn extends UTPawn;

/**
 * Called when the pawn should calculate the camera location and rotation
 *
 * @param  fDeltaTime  Time that has passed since the last time this function was
called
 * @param  out_CamLoc  Output camera location
 * @param  out_CamRot  Output camera rotation
 * @param  out_FOV   Output camera field of vision
 * @return       Returns true if the camera should use the output calculated from
this function
 */

simulated function bool CalcCamera(float fDeltaTime, out vector
out_CamLoc, out rotator out_CamRot, out float out_FOV)
{

  local vector start, end, hl, hn;
  local actor a;

 // Perform a trace to check if positioning the camera behind the pawn will
intersect with world geometry. This is
 // done to prevent the camera from being embedded within a static mesh,
another pawn, or other objects
 // Get the starting location of the trace
  start = Location;

 // If there is a controller, then calculate the end location of the trace based on the
Controller's rotation

  if (Controller != none)
  {
    end = Location - Vector(Controller.Rotation) * 192.f;
  }
 // Otherwise, calculate the end location of the trace based on my rotation
  else
  {
    end = Location - Vector(Rotation) * 192.f;
  }

 // Perform the trace between start and end

  a = Trace(hl, hn, end, start, false);
 // Check if the trace has detected an actor. If it has then relocate the out camera
location to the hit location

  if (a != none)
  {
    out_CamLoc = hl;
  }
 // Otherwise, output where we've calculated the trace end
  else
  {
    out_CamLoc = end;
  }

 // Calculate the camera rotation so that the camera is looking at me at all times

  out_CamRot = Rotator(Location - out_CamLoc);

 // Return true to get the camera to use the camera location and rotation we have
just calculated
  return true;
}

// Default property block
defaultproperties
{
 // Set bOwnerNoSee to false so that the player can always see the pawn's
skeletal mesh
  Begin Object Name=WPawnSkeletalMeshComponent
    bOwnerNoSee=false
  End Object
}
```



The UDK main editor window.

The Unreal Engine uses all the .ini files with “Default” in the name to generate the .ini files with “UDK” in the name. In other words, if you were to delete UDKEngine.ini, and then start the game, it would automatically re-create UDKEngine.ini based on the information it found in DefaultEngine.ini. We’re going to edit the default .ini files to build our game.

Open DefaultEngine.ini and look for the section [UnrealEd.EditorEngine]. Add this line at the bottom of that section:

```
"+EditPackages=MyFirstGame".
```

It should look like this:

```
[UnrealEd.EditorEngine]
+EditPackages=UTGame
+EditPackages=UTGameContent
+EditPackages=MyFirstGame
```

“MyFirstGame” refers to the name of the folder you made in the Development folder, so if you named your folder something else, you’ll need to put that name here instead.

Next, locate the section [Engine.Script Packages] and add `"+NonNativePackages=MyFirstGame"` to the bottom of that section.

For Example:

```
[Engine.ScriptPackages]
+NonNativePackages=UTGame
+NonNativePackages=UTGameContent
+NonNativePackages=MyFirstGame
```

Save and close the .ini file, and start the editor via the Windows Start Menu shortcut. If you’ve done everything correctly, a dialog window will pop up that says “Scripts are outdated. Would you like to rebuild now?” Click Yes.

After UDK has finished rebuilding the scripts, go to the UDKGame folder and open up the Script folder within. This folder contains the compiled code. Every file here corresponds to a folder in the Development folder. To verify that your new code compiled, see if you can locate the file MyFirstGame.u. If that file is there, start up the Unreal Editor again; you’re ready to start building a level.

BUILDING YOUR FIRST LEVEL

/// Now that your code is compiled, you can start building a level in the Unreal Development Kit. Let’s start by explaining the interface.

1 The window on the right is the Content Browser. This contains all of the textures, models, sounds, and so on that are in your game. It also has all placeable actors, gives access to sublevels, layers, and so on.

2 On the left is the viewport, where we can work on the level itself. There are two different ways to navigate within the viewport: You may hold the right mouse button and use WASD or the arrow keys to move around, or you may left-click and drag in the viewport to turn and move forward/backward, right-click and drag to look around, and hold both mouse buttons to drag to pan the screen. The latter way is faster but takes a little getting used to.

GOT GAME?

ENTERTAINMENT ARTS & ENGINEERING AT THE UNIVERSITY OF UTAH

If you've got game, we have the program to turn your obsession into a career.

UNDERGRADUATE EMPHASIS

Film and Media Arts Emphasis BA
Computer Science Emphasis BS

MASTER GAMES STUDIO

Game Arts MFA
Game Engineering MS
Game Production MFA

OUR GRADUATES WORK FOR:

Microsoft
EA
Disney Interactive
Zynga
Turbine
And many more!

Ranked No. 1 among public universities in the U.S. for game development by the Princeton Review, and No. 3 among all universities nationwide.

With developers like ***EA, Disney Interactive, Epic, eat sleep play, Smart Bomb Interactive, Ninjabee*** and many more based here in the Salt Lake area, you would have unprecidented access to industry leaders.

Come study in Salt Lake City, voted one of the happiest cities for new grads* and top places to start a new business**!

*http://www.forbes.com/sites/jacquelynsmith/2012/05/18/the-happiest-cities-for-job-seeking-college-grads/

**http://m.heraldextra.com/news/local/utah-among-top-in-nation-for-starting-small-businesses/article_cccb3963-7f05-57e5-8064-bcff8729f325.html

THE UNIVERSITY OF UTAH

eae.utah.edu

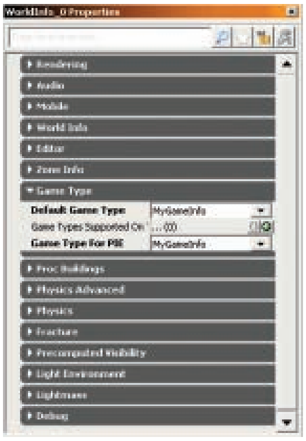
fb.com/EAEUofU | @EAEMGS
Salt Lake City | UT | 801.585.6491

3 The small cube buttons right above the viewport allow you to change the viewport into wireframe or unlit mode, among other display modes. You can use these different display modes to look at a level in different ways (such as with or without lighting) in order to better see what you're building. The "P" button in that toolbar allows you to switch between different viewports (top-down, side view, front view, and perspective).

4 The leftmost button on the viewport toolbar (the small arrow pointing downward) contains the viewport settings. Sometimes, you might accidentally hit a key that triggers a viewport option that makes something disappear. Don't panic! When that happens, press this button and go to Show, Reset to Defaults to restore your viewport to its default settings.

5 You may access a few special editing modes from the top part of the toolbar on the left. The most important one is the default camera tool, at the top-left, which you use to navigate the viewport and view a level while you're making changes to it. You can also access the brush edit mode, landscape mode, foliage mode, and other special editing modes that allow you to make advanced modifications to specific elements within a level.

6 The Brushes and CSG parts in the left toolbar allow you to make brushes (simple geometric primitives used for creating very simple environments). We won't go into detail about these tools in this tutorial; a lot of the buttons in this section are remnants of old versions of the engine, and I advise you to leave them alone and stick to using only "static meshes."



WorldInfo_0 Properties

- Rendering
- Audio
- Mobile
- World Info
- Editor
- Zone Info
- Game Type
  - Default Game Type: MyGameInfo
  - Game Types Supported On: ... (0)
  - Game Type For PIE: MyGameInfo
- Proc Buildings
- Physics Advanced
- Physics
- Fracture
- Precomputed Visibility
- Light Environment
- Lightmass
- Debug

Configuring UDK to use your new gametype.

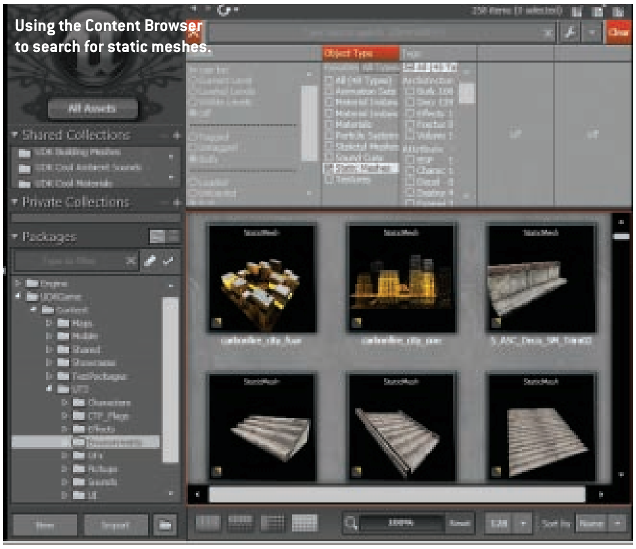
Now that you're acquainted with the editor's basic interface, click the green arrow on the right side of the top toolbar. This is the Play button. It opens the game window and allows you to walk around in the default scene. Go ahead and walk around for a bit. Cool, right? But it's not using your MyFirstGame code you set up earlier, so let's fix that. Go to the View menu, pick World Properties, and click Game Type. Set both Default Game Type and Game Type For PIE to MyGameInfo.

The gametype controls what pawn/player type is used and how that player is controlled. So by setting the gametype to MyFirstGame, UDK will automatically use MyPlayerController and MyPawn as well.

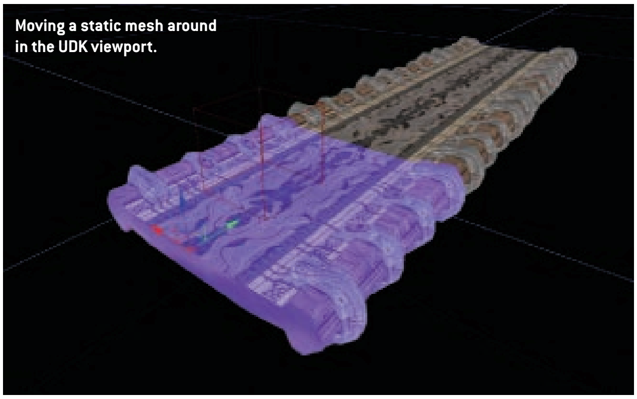
Now click the Play button and try to play the level again. If everything worked correctly, you should get a third-person camera, which means UDK is now using your MyFirstGame code.

As you can see, your character is a robot that is carrying a gun (which can shoot!). Earlier, you wrote MyGameInfo to extend UTDeathmatch, and since UnrealScript files take the functionality of their parents, MyFirstGame is now basically a third-person Unreal Tournament Deathmatch game. If you want to change or disable things, you would either want to extend another existing gametype instead (if you use "SimpleGame" instead of UTDeathmatch, and you'll just spawn a simple character with no game rules), or write code yourself that controls how the player looks and what he can and can't do.

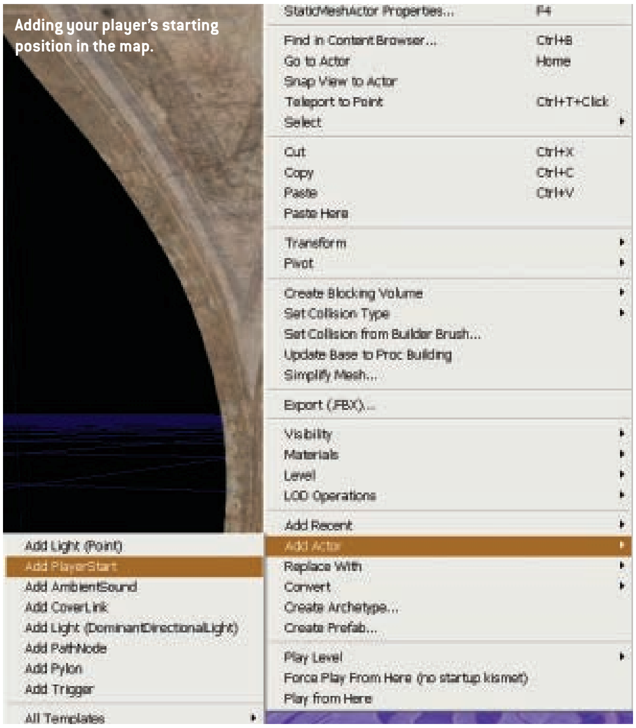
Now we can start putting the level together. Go to the File menu, New, and choose the Blank Map template, and you'll see an empty space in dire need of some filling. Go to the Content Browser (if you closed it earlier, you can reopen by going to View, Browser Windows, Content Browser) and expand UDKGame in the bottom-left panel, then expand Content. Each listing



Using the Content Browser to search for static meshes.



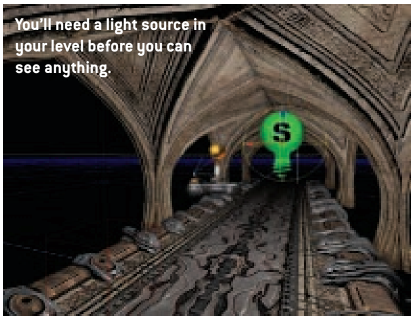
Moving a static mesh around in the UDK viewport.



Adding your player's starting position in the map.

in this window is either a content package (sounds, particle effects, menus, materials—pretty much anything) or a game level. Take a minute to browse around a bit and explore.

You'll find a lot of useful stuff in the UDKGame\Content\UT3\Environments folder, so open that in the Content Browser, select the Environments folder, and check the Favorites panel in the upper-middle section of the Content Browser. The Favorites panel lists various types of assets such as Animation Sets, Material Instances (Constant), Material Instances (Time Varying), Materials, and so on. Select Static Meshes from the list, so the Content Browser only shows assets tagged as static meshes. Static meshes are your standard building blocks for a level—90 percent or so of a level is generally made up of static meshes.


You'll need a light source in your level before you can see anything.

Find something that looks as if you'd be able to walk on it (such as "S_NEC_Floors_SM_Walkway_STR1"), select it in the Content Browser, and then drag it from the Content Browser to the empty space in the viewport. You've now added a static mesh to your level. You can move it around using the three colored arrows (click and drag one of them). If you want to rotate the static mesh, press the spacebar on your keyboard once to transform the three colored arrows into three colored curves, and click and drag to rotate an object. Press spacebar yet again, and you will get three red blocks connected by three lines. This is the scale tool. Press spacebar once more to return to the move tool.

Assemble a little environment by adding more static meshes to your level and moving them around. Once you have something that resembles a basic level, you're ready to set it up so it's playable and apply some lighting.

Before you try to play through the level, you'll need to define the player's start location—the Play button won't work until you do this. Right-click one of your models in the viewport (preferably on the floor, since that is where you want to start the game) and pick Add Actor, Add Playerstart.

You should see a little joystick icon appear in the level. If you instead see a square that says "bad size," that means the Playerstart does not have enough empty space around it, so try moving it to a more spacious location. Also, make sure you have already set the gametype to MyGameInfo in the View, World Properties menu.

You can now press the Play button in the top toolbar, and your game should work—but it'll be pitch black. That means we need some lighting.

Make sure the viewport is set to Lit mode by using the viewport toolbar (or pressing Alt + 4 for the keyboard shortcut), and then right-click somewhere on one of the static meshes in your level and choose Add Actor, Add Light (Point). You should now see a lightbulb icon, and your level should now have light. Move this lightbulb around to where you want to have it.

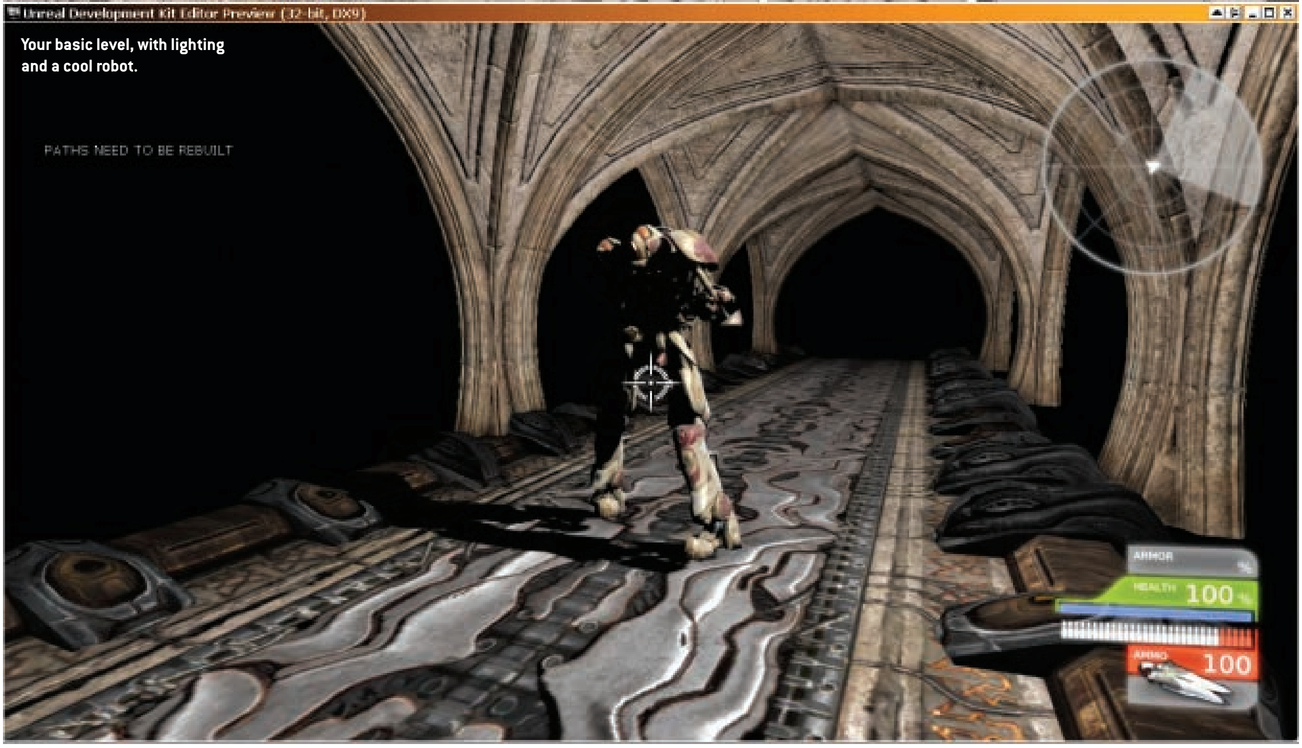
You're still not ready to play your level yet, because the lighting has not yet been compiled and baked into the level. The editor gives you a preview of dynamic lighting, but the game itself cannot do that. So go to Build, Build Lighting, and press OK in the Lighting Build Options window that pops up. Once it's finished, press Play to walk around your newly lit level. (You may get an in-game message that says "Paths need to be rebuilt." To get rid of that message, go to the Build menu and choose AI Paths.)

You can change the color, brightness, and radius of a light by modifying its properties. Double-click the light bulb in the viewport to open up the Properties window. You can find the Radius property in Light, Light Component, Point Light Component, and you can find Brightness and Light Color in Light, Light Component, Light Component.

Note that every time you move or modify a light actor, you must build the lighting again or it will not show up in the game.

BRINGING YOUR LEVEL TO LIFE

/// You can run around in your level, but that's about it—so we're going to add a bit of interactivity. The Unreal Engine has a very easy-to-understand


Unreal Development Kit Editor Preview (32-bit, DX9)

PATHS NEED TO BE REBUILT

Your basic level, with lighting and a cool robot.

visual scripting tool called Kismet, which handles all level scripting.

To start, add a trigger actor somewhere in the level by right-clicking the floor and selecting Add Actor, Add Trigger. Next, add a standard pointlight above the trigger by right-clicking and choosing Add Actor, Add Light (Point) like you did before, and then right-click the light you just made and go to Convert Light, Pointlights, PointlightToggleable. This will convert your light into a toggleable light, which will let us toggle the light on and off using the trigger you've just made.

Now it's time to get the light and the trigger into Kismet. Select the trigger in the viewport, and then open Kismet by clicking the green "K" button in the top toolbar. Right-click the central gray area in Kismet and in the right-click menu pick New Event Using Trigger_0, Touch.

This will bring your trigger actor into Kismet as a touch event. Next, add a Toggle action by right-clicking the central space again and choosing Action, Toggle, Toggle. Connect the output Touched from the Trigger event to the Turn Off input on the Toggle action by clicking the black connector box on either of the two and dragging to draw a line between the two. Then select the toggleable light in the viewport, and in Kismet right-click the central area once more, but this time choose New Object Var Using PointLightToggleable_0. You now have your light in Kismet as an object. Finish the setup by connecting the PointLightToggleable_0 object to Target from the Toggle action. If you rebuild the lighting and play the game, you should be able to walk into the trigger to turn the light off.

If you want the trigger to turn the light back on, go back into Kismet and connect Untouched to Turn On, so that when you leave the trigger the light will be toggled on again. You'll need to make sure that trigger can be used more than once, so click the Trigger_0 Touch event in Kismet. You'll see Trigger_0's properties pop up in the bottom-left corner of the Kismet window. In here, set Max Trigger Count to 0 to make it trigger an infinite number of times.

WRAPPING UP

/// We're just about done with this tutorial, but we need to walk you through a few steps to finish your game first. Save your level by going to the File menu and choosing Save Current Level. Make sure to save the level within the UDKGame/Content folder or it won't work. You can save it anywhere within the Content folder, but for organization's sake, save it into UDKGame/Content/Maps. Also, the Unreal Engine does not like spaces or nonstandard characters in the filename, so keep it simple and stick to letters from the English alphabet.

You'll need to add a prefix to the level name, too, because the Unreal Engine uses it to determine what gametype the level corresponds to, so call your level MY-Firstlevel. The MY is the prefix in that case.

Once you're satisfied with your level, close the editor and return to the UDKGame/Config folder—we have to edit another .ini file to make sure that your level is the default level the game uses. Make sure the editor is closed before you edit any .ini files!

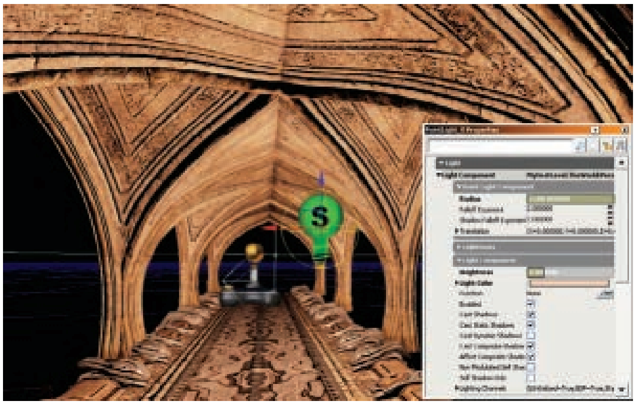
Open DefaultEngine.ini, and at the top change Map=UDKFrontEndMap.udk into Map=MY-Firstlevel.udk (or whatever the name of your level was), and once more for the line "LocalMap=". Save and close.

We need to teach the game to recognize the new prefix we made, too. You can do this by opening DefaultGame.ini and finding the "+DefaultMapPrefixes=" lines at the top. Add the following line under the existing four DefaultMapPrefixes lines.

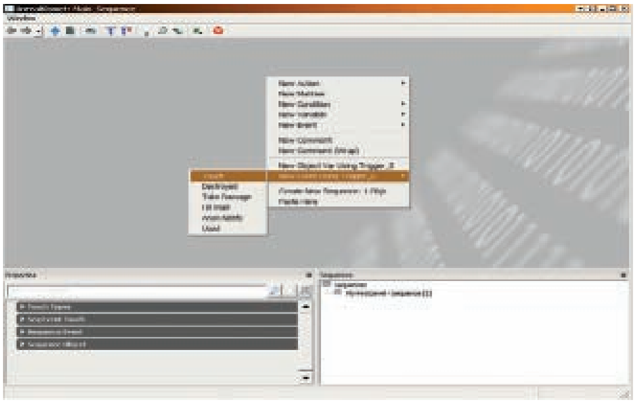
```
+DefaultMapPrefixes=(Prefix="My",bUsesCommonPackage=FALSE,GameType="MyF
irstGame.MyGameInfo")
```

Save and close, and then start the game via the Windows Start Menu shortcut. It should automatically load your level with the correct third-person gametype. Congratulations, you've made your first basic Unreal Engine game.

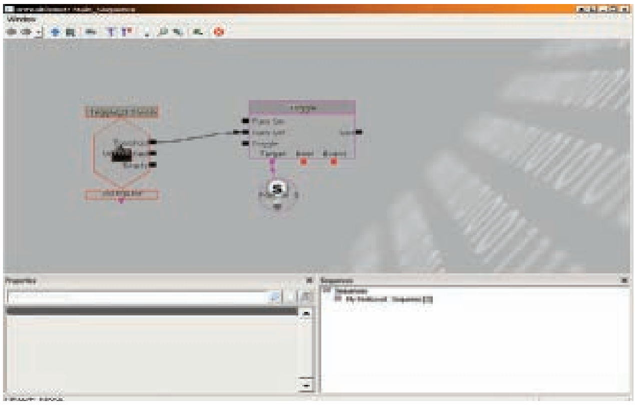
Of course, there is a whole lot more to the engine than the few things that we covered in this introduction tutorial, so experiment and try things out. The Unreal Engine is very artist/designer-friendly, so even if you're not a programmer, you can still accomplish a whole lot in the engine, especially with the Material Editor, Kismet, Matinee, and other powerful subeditors. Good luck!



Modifying the light source's properties.



Setting up a basic touch trigger in Kismet.



Using Kismet to set up a light switch trigger.

SJOERD DE JONG *is an artist and designer who started out in the mod community around 1998 by making levels for* UNREAL. *Since then he has worked for companies such as Starbreeze Studios, Guerrilla Games, and Epic Games, and is currently employed by his own company, Teotl Studios (*THE BALL*). He is also an Unreal Engine development consultant, tutor, and lecturer, and has many more UDK tutorials available at www.Hourences.com.*

Thanks to James Tan for the example code.

Sheridan

get your **game** on!

Turn your skill and passion
INTO CAREER SUCCESS!

Sheridan offers three programs that will give you an edge to break into this exciting field:

GAME DESIGN BACHELOR'S DEGREE (starting in September 2013)
POST-GRAD GAME LEVEL DESIGN
POST-GRAD GAME DEVELOPMENT – ADVANCED PROGRAMMING

Employers want savvy, creative grads who can create memorable virtual experiences that truly engage players. Sheridan can help you become that grad!

gaming.sheridancollege.ca

ELEVENTH ANNUAL SALARY SURVEY

PATRICK MILLER

Every year, we ask thousands of *Game Developer* and Gamasutra readers to tell us how much money they made last year, along with a slew of related questions. We get everyone from established developers to newbies to tell us their base salaries, benefits, additional compensation, and other work information so we can show you what to expect if you decide to pursue a career in the game industry.

Interestingly, the average salaries for the entry-level crowd have increased across the board, which stands in contrast to the industrywide trend of small-but-steady growth. Whether you're looking to start a career in quality assurance or business, you'll see salaries rising rather impressively—though some disciplines are paid much more than others.

This year, we learned that the average salary across the entire game industry is $81,192, hovering near the same level as 2010's $80,817 reported average. What that number doesn't tell you is that the industry was significantly more stable this year than it has been in the past several. Only 13 percent of respondents were laid off in 2011, compared to 14 percent in 2010 and 19 percent in 2009, and those that received layoffs were 6 percent more likely to find a new job elsewhere in the game industry (58 percent, up from 52 percent in 2011).

Having a little more money and stability in turn made developers feel more optimistic about their careers as well as the industry as a whole. A majority of developers—65 percent—said they felt "satisfied" or "extremely satisfied" with their potential career path (up 4 percent from 2010), 34 percent believed that there were more jobs in the industry than the year before (up 5 percent), and 54 percent felt that there were more opportunities for game developers than before (up 7 percent).

Nevertheless, there's still plenty of change going on. Independent devs in particular made a big step in 2011: Individual indie developers reported an average $23,549 in primary compensation, more than double 2010's $11,379, while members of independent developer teams made an average of $38,239, up $11,459 from 2010's $26,780. If you are looking to break into the industry, more avenues are available for you than ever before. Here's to living in the clouds.

gd

programmers

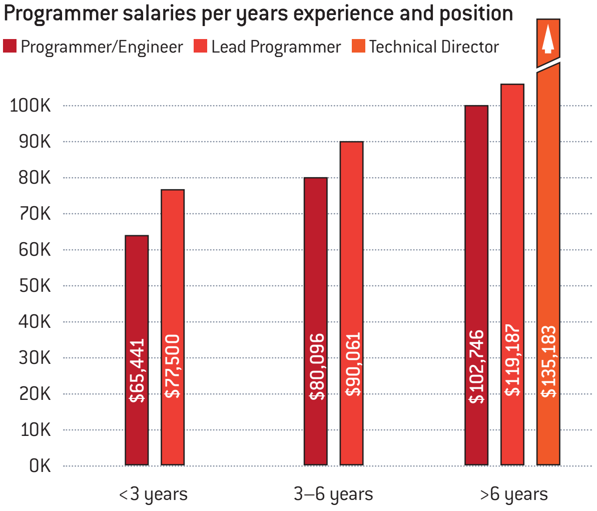
AVERAGE SALARY 3 YEARS OR LESS **$66,116**

Programmers are in high demand, even at the entry-level; the average salary for programmers in their first three years is higher than that of all other disciplines except business, and it's actually up $10,700 over last year's average of $55,426. Considering we had more programmers respond to the survey than we did last year, it seems that the industry simply can't hire talented programmers fast enough.

But Canadian developers didn't see the same boom—their $74,970 (USD) average across all skill levels was only up about $500. European programmers reported an average of $46,801 (USD), down about $1,400 from 2010.

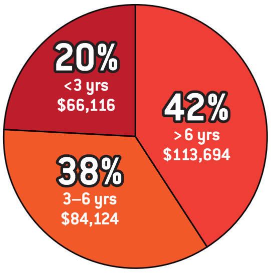
While women were underrepresented in the workforce even more than usual (2.9 percent in 2011, down from 4 percent in 2010) , they did report an average increase of about $8,800, compared with their male counterparts' $7,000 increase. However, the wage gap is still alive and well; the average female programmer's salary ($83,333) is nearly $10,000 lower than the average male programmer's ($93,263).

Programmer salaries per years experience and position



ALL PROGRAMMERS AND ENGINEERS

YEARS EXPERIENCE IN THE INDUSTRY



Percent receiving additional income:	83%
Average additional income:	$21,009

Type of additional compensation received

Annual bonus	46%
Pension/Employer contribution to Retirement plan	38%
Profit sharing	16%
Project/title bonus	20%
Royalties	8%
Stock options/equity	35%

Percent receiving benefits:	94%

Type of benefits received

Medical	84%
Dental	80%
401K/Retirement	72%

GENDER STATS FOR PROGRAMMERS

Gender	Percent Represented	Average Salary
Male	97%	$93,263
Female	3%	$83,333

artists and animators

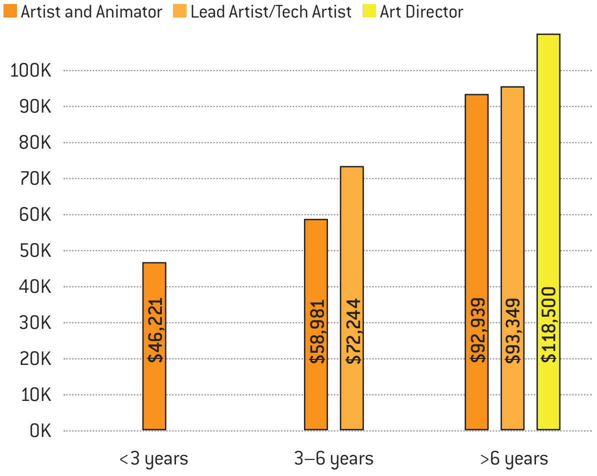
AVERAGE SALARY 3 YEARS OR LESS **$49,481**

Artists and animators with less than three years of experience saw a respectable increase of about $3,700 over the previous year, but their average still places them in the lower end of the spectrum compared to programmers, producers, and businesspeople. Also, would-be game artists and animators should take note: The game industry is kind of a notorious grind on artists, who are often expected to quickly develop a technical specialty and worked hard until they burn out, so be careful when choosing where and how to enter the industry.

Artists and animators in Canada received an average salary (across all skill levels) of $66,651 (USD), up about $3,300. Artists in Europe didn't fare so well, however. Their $35,887 (USD) average salary fell about $5,000 from 2010.

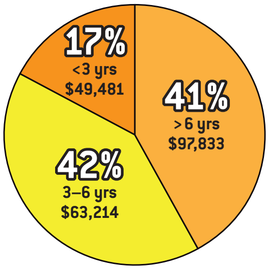
Women comprised 13 percent of our surveyed artist pool, up 2 percent, though their average salary across all skills levels ($52,875) actually decreased about $6,800. Men in art and animation, on the other hand, made $79,124 on average, which is up about $6,200 from 2010.

Artist and Animator salaries per years experience and position



ALL ARTISTS AND ANIMATORS

YEARS EXPERIENCE IN THE INDUSTRY



Percent receiving additional income:	83%
Average additional income:	$16,163

Type of additional compensation received

Annual bonus	40%
Pension/Employer contribution to Retirement plan	36%
Profit sharing	11%
Project/title bonus	30%
Royalties	14%
Stock options/equity	28%

Percent receiving benefits:	95%

Type of benefits received

Medical	80%
Dental	76%
401K/Retirement	65%

GENDER STATS FOR ARTISTS

Gender	Percent Represented	Average Salary
Male	87%	$79,124
Female	13%	$52,875

game designers

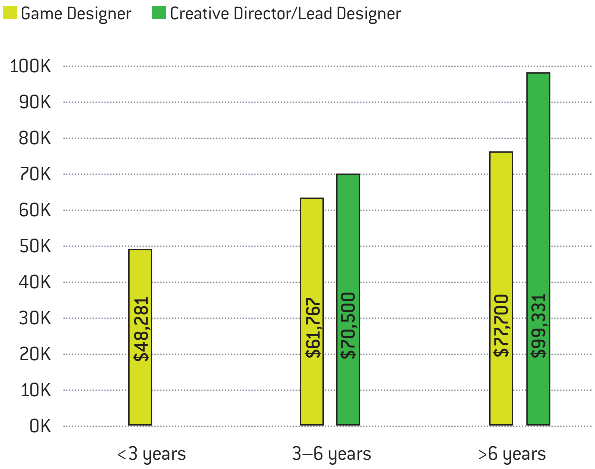
AVERAGE SALARY
3 YEARS OR LESS
$50,375

The average salary for entry-level game designers and writers is $4,100 higher than the previous year's, and it stands roughly in the middle of the industry's salary spectrum. This discipline isn't always the easiest one to get into, though, meaning there can be a lot of competition for relatively few jobs. Also, if you have no prior game-industry experience, you probably won't be able to land a design gig without having at least made a few of your own games.

Canadian game designers across all skill levels averaged $60,240 (USD), up about $1,950 from 2010, while European game designers averaged $38,281 (USD), which is down about $3,000.

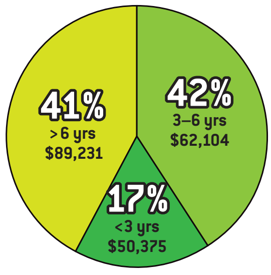
Female designers made up 10.9 percent of our designer responses, compared to 7 percent from 2010, and their average salary across all skill levels of $67,000 is up $2,850 from 2010. Male designers, on the other hand, made an average of $74,180 in 2011—up $3,500 from 2010.

Game Designer salaries per years experience and position



ALL GAME DESIGNERS

YEARS EXPERIENCE IN THE INDUSTRY



GENDER STATS FOR DESIGNERS

Gender	Percent Represented	Average Salary
Male	89%	$74,180
Female	11%	$62,000

Percent receiving additional income:	79%
Average additional income:	$15,216

Type of additional compensation received

Annual bonus	35%
Pension/Employer contribution to Retirement plan	34%
Profit sharing	13%
Project/title bonus	23%
Royalties	10%
Stock options/equity	31%

Percent receiving benefits:	96%

Type of benefits received

Medical	84%
Dental	80%
401K/Retirement	70%

producers

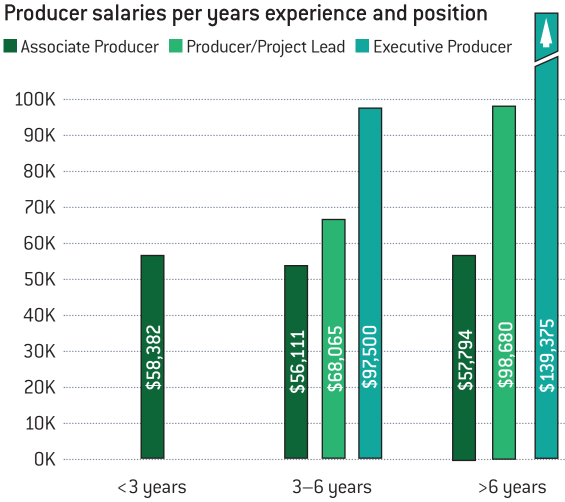
AVERAGE SALARY
3 YEARS OR LESS
$55,893

Entry-level producers received an average of about $3,600 more in 2011 than they did in 2010, and their overall average of $55,893 is on the higher end for the game industry. This stands in contrast to the overall average for producers in 2011; they saw a $2,850 salary cut compared to 2010, though they still draw a higher salary than all other departments except programming and business.

Canadian game producers across all skill levels averaged $71,500 in 2011, down $1,000 from 2010. European producers' salaries increased $3,500 to $56,346. Interestingly, producers in Europe are paid more than any other department of game development in the region.

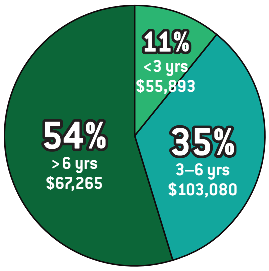
Women are well represented in the producer ranks this year (16 percent), down 1 percent from 2010. Interestingly, men absorbed most of the salary drop; their average salary across all skill levels dropped from $90,744 to $87,119, while women producers' salaries actually rose slightly from $77,870 to $78,354.

Producer salaries per years experience and position



ALL PRODUCERS

YEARS EXPERIENCE IN THE INDUSTRY



GENDER STATS FOR PRODUCERS

Gender	Percent Represented	Average Salary
Male	84%	$87,119
Female	16%	$78,354

Percent receiving additional income:	85%
Average additional income:	$19,050

Type of additional compensation received

Annual bonus	49%
Pension/Employer contribution to Retirement plan	37%
Profit sharing	13%
Project/title bonus	23%
Royalties	6%
Stock options/equity	37%

Percent receiving benefits:	98%

Type of benefits received

Medical	61%
Dental	60%
401K/Retirement	54%

mobile

Proppa is an international collaboration between students from DePaul and the Trident College of Computing in Japan.

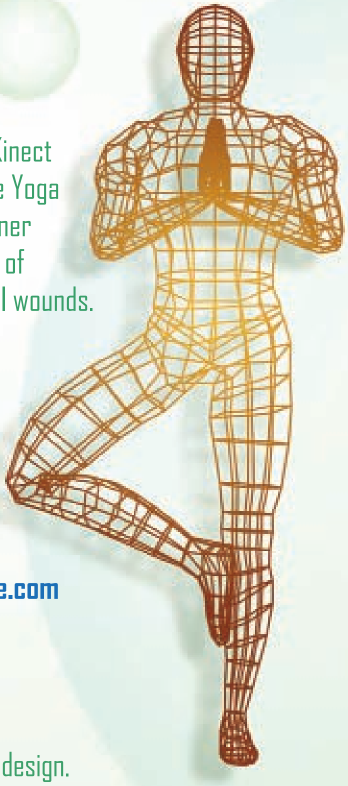
augmented reality

Kaiju Kazoo is an iOS game created by DePaul game students, alumni, and faculty. The game blurs the boudaries between the game world and reality.

explore

kinect

Zombie Yoga is an Xbox Kinect game in which players use Yoga poses to unleash their "inner light" to both fight hordes of undead and heal emotional wounds.



indie

Octodad: Dadliest Catch is the creation of Young Horses, Inc., an indie game company formed by DePaul alumni to produce a commercial sequel to their IGF Student Showcase game, Visit **OctodadGame.com**

GameDev.DePaul.edu

At DePaul you'll work with faculty who have extensive game development experience, not to mention our faculty experts in animation, AI, cinema, game studies, screenwriting, and sound design. We offer graduate and undergraduate programs in **game art**, **programming**, **design**, and **animation**.

DEPAUL UNIVERSITY®
COLLEGE OF COMPUTING & DIGITAL MEDIA

audio professionals

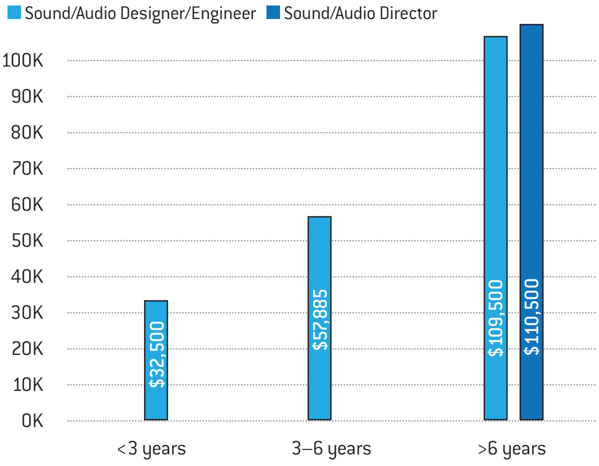
AVERAGE SALARY
3 YEARS OR LESS
$32,500

Audio jobs in the game industry are a tricky thing to monitor because there aren't nearly as many jobs for audio professionals, and most of those jobs are offered as contract gigs instead of salaried work. Full-time, salaried audio jobs are often only open to accomplished developers at the higher end of the experience (and salary) spectrum, while entry-level gigs are scarce and not particularly well paid. (This, by the way, makes it harder to collect reliable data for our survey—the $32,500 average for entry-level audio workers comes from a grand total of four respondents.)

Audio professionals working in contract roles also made an extra $3,200 in 2010. Salaried audio workers were the least likely to receive extra compensation for their work out of any discipline, though the $9,875 they received is up $2,200 from 2010.

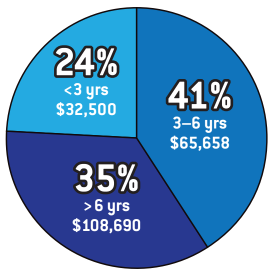
Canada-based audio professionals received an average salary across all skill levels of $67,955 in 2011, down $600 from 2010. Unfortunately, we didn't collect enough responses from European audio workers to make any significant conclusions.

Audio Developer salaries per years experience and position



ALL AUDIO DEVELOPERS

YEARS EXPERIENCE IN THE INDUSTRY



Percent receiving additional income:	74%
Average additional income:	$9,875

Type of additional compensation received

Annual bonus	40%
Pension/Employer contribution to Retirement plan	47%
Profit sharing	12%
Project/title bonus	23%
Royalties	9%
Stock options/equity	28%

Percent receiving benefits:	90%

Type of benefits received

Medical	79%
Dental	79%
401K/Retirement	63%

GENDER STATS FOR AUDIO DEVELOPERS

Gender	Percent Represented	Average Salary
Male	93%	$83,963
Female	7%	$72,500

qa testers

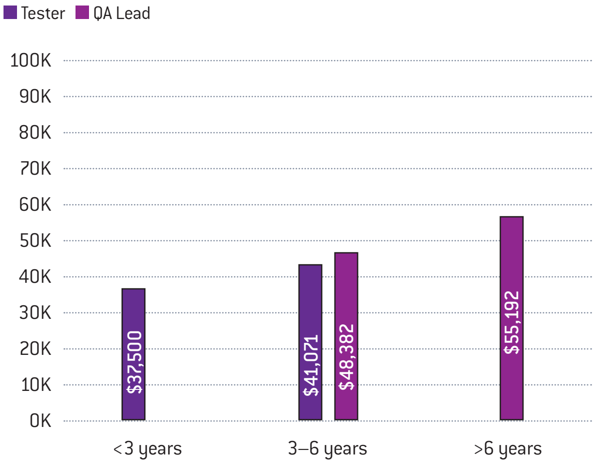
AVERAGE SALARY
3 YEARS OR LESS
$37,000

Quality-assurance professionals are the lowest-paid people in the game industry, especially the entry-level hires. On the bright side, they did manage to make an extra $800, on average, over the previous year. If you're trying to break into the game industry without any trade skills like programming or art, you will probably find QA the easiest place to start out. But not many studios offer a QA career track that is as developed as the other disciplines, so talented QA leads often end up moving over to design or production instead.

Most QA employees are hired on a contract basis, with salaried positions mostly given to QA leads. QA contractors across all skill levels made an average of $27,065 in 2011, up about $4,150 from 2010.

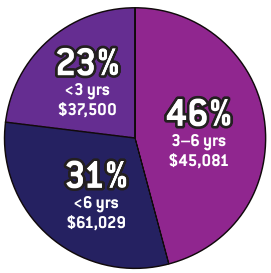
Interestingly, average salaries for testers with less than three years of experience rose about $6,200 from 2010, and QA leads with 3 years to 6 years of experience saw a $3,700 boost. Canadian QA professionals across all skill levels made an average of $43,125 (USD), up $5,100 from 2010. European QA workers made $32,500 (USD), which was about $6,750 less from 2010.

QA Tester salaries per years experience and position



ALL QA TESTERS

YEARS EXPERIENCE IN THE INDUSTRY



Percent receiving additional income:	77%
Average additional income:	$12,640

Type of additional compensation Received

Annual bonus	49%
Pension/Employer contribution to Retirement plan	33%
Profit sharing	7%
Project/title bonus	16%
Royalties	5%
Stock options/equity	30%

Percent receiving benefits:	95%

Type of benefits received

Medical	68%
Dental	70%
401K/Retirement	64%

GENDER STATS FOR QA TESTERS

Gender	Percent Represented	Average Salary
Male	87%	$49,196
Female	13%	$39,375

business and legal people

AVERAGE SALARY
3 YEARS OR LESS

$71,818

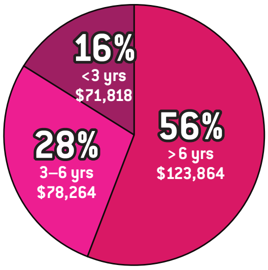
The “Business and Legal People” category includes chief executives and executive managers, community managers, marketing, legal, human resources, IT, content acquisition and licensing, and general administration staff. Younger people on the business side made an average of $71,818 in 2011, $14,000 up from $57,778 in 2010.

Business professionals received the highest average salary (across all skill levels) in the industry ($102,160), as well as the most additional nonsalary income ($24,874). However, both those numbers are actually down from 2010 by $3,300 and $4,000, respectively.

Women were relatively well represented in the business side of the game industry, reaching 17.6 percent in 2011, which is 7.6 percent higher than average across the entire industry this year, and 3.6 percent higher than in 2010.

ALL BUSINESS AND LEGAL PEOPLE

YEARS EXPERIENCE IN THE INDUSTRY



GENDER STATS FOR BUSINESSPEOPLE

Gender	Percent Represented	Average Salary
Male	82%	$108,402
Female	18%	$73,534

Percent receiving additional income:	**82%**
Average additional income:	**$24,874**

Type of additional compensation Received

Annual bonus	46%
Pension/Employer contribution to Retirement plan	30%
Profit sharing	19%
Project/title bonus	15%
Royalties	11%
Stock options/equity	44%

Percent receiving benefits:	**95%**

Type of benefits received

Medical	82%
Dental	78%
401K/Retirement	62%

LAYOFFS

JOBS IN THE GAME INDUSTRY APPEAR TO BE GETTING SLIGHTLY MORE stable. Of 3,100 respondents, 13 percent had been laid off in 2011, compared to 14 percent in 2010 and 19 percent in 2009.

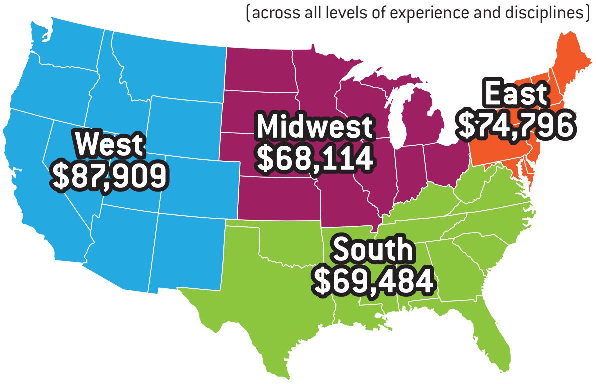
From those people who were laid off, 58 percent found new employment in the games industry, 19 percent went into contracting or consulting, 10 percent founded a new company, 13 percent went into independent games development, and 13 percent haven't found new game development work. (Note that for this survey question, multiple responses were allowed.)

A significant amount of respondents reported being laid off and rehired by the same company, either as a contractor, or as a salaried employee with a different job title (but the same responsibilities).

Fewer laid-off developers opted to start their business, join an independent studio, or go into contract work this year. This could be a sign of a slowing bubble in the social and mobile sectors, which were giving away massive amounts of cash in recent years.

AVERAGE SALARY BY U.S. REGION

(across all levels of experience and disciplines)



TOP 10 STATES WITH HIGHEST AVERAGE SALARIES

(across all levels of experience, excluding states with low sample size)

		AVERAGE SALARY	PERCENT WHO OWN HOMES	AVG. SALARY OF HOMEOWNERS
1	**California**	$93,696	32%	$118,513
2	**Washington**	$89,674	49%	$108,479
3	**North Carolina**	$82,500	61%	$93,382
4	**Texas**	$79,017	49%	$95,761
5	**Massachusetts**	$78,567	44%	$94,688
6	**Maryland**	$75,521	44%	$94,688
7	**Utah**	$74,722	70%	$78,816
8	**Illinois**	$71,319	47%	$84,167
9	**New York**	$69,697	29%	$85,395
10	**Florida**	$67,136	33%	$90,000

AVERAGE SALARY BY U.S. REGION BY DISCIPLINE

	EAST	MIDWEST	SOUTH	WEST
Programmer	$84,167	$73,397	$82,740	$101,387
Art and Animation	$64,235	$68,676	$71,818	$81,229
Game Design	$69,575	$61,818	$52,616	$82,204
Production	$77,813	$87,045	$68,889	$90,395
Audio	$60,625	$55,00	$68,333	$97,885
QA	$47,500	$44,167	$39,643	$49,545
Business	$97,870	$87,500	$83,750	$108,056

AVERAGE SALARY FOR HOMEOWNERS VS. NON-HOMEOWNERS BY U.S. REGION

	EAST	MIDWEST	SOUTH	WEST
Homeowners	$93,662	$81,378	$89,415	$111,842
Non-Homeowners	$66,602	$58,777	$58,223	$79,148

AVERAGE SALARIES IN THE U.S., CANADA, AND EUROPE

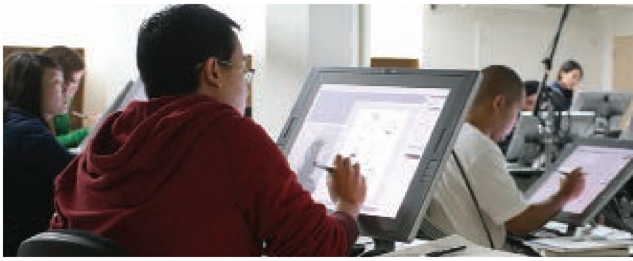
(across all levels of experience, by discipline, given in USD)

	U.S.	CANADA*	EUROPE**
Programmer	$84,124	$74,970	$46,801
Art and Animation	$63,214	$66,651	$35,887
Game Design	$62,104	$60,240	$38,281
Production	$67,265	$71,500	$56,346
Audio	$65,658	$67,955	$25,500
QA	$45,081	$43,125	$32,500
Business	$79,269	$100,938	$47,222

*Most Canadian respondents were from British Columbia, Quebec, and Ontario.
**Most European respondents were from the United Kingdom (24%), Germany (12%), France (11%), Spain (9%), Poland (10%), Sweden (5%).

ACADEMY *of* ART UNIVERSITY®

FOUNDED IN SAN FRANCISCO 1929 BY ARTISTS FOR ARTISTS



TAKE CLASSES ONLINE OR IN SAN FRANCISCO

Acting*
Advertising
Animation & Visual Effects
Architecture
Art Education
Fashion
Fine Art
Game Design
Graphic Design
Illustration
Industrial Design
Interior Architecture & Design
Landscape Architecture
Motion Pictures & Television
Multimedia Communications
Music Production & Sound Design for Visual Media
Photography
Web Design & New Media

ENROLL NOW

EARN
YOUR AA, BA, BFA, MA, MFA OR M-ARCH ACCREDITED DEGREE

ENGAGE
IN CONTINUING ART EDUCATION COURSES

EXPLORE
PRE-COLLEGE SCHOLARSHIP PROGRAMS

WWW.ACADEMYART.EDU
800.544.2787 (U.S. Only) or 415.274.2200

79 NEW MONTGOMERY ST, SAN FRANCISCO, CA 94105

Accredited member WASC, NASAD, CIDA (BFA-IAD, MFA-IAD), NAAB (M-ARCH)

**Acting degree program is not currently available online.*

Visit www.academyart.edu to learn about total costs, median student loan debt, potential occupations and other information.

Photo credit: Molly Maloney, Chris Haejin Chu

AVERAGE SALARY BY EDUCATION LEVEL AND DISCIPLINE

(across all levels of experience)

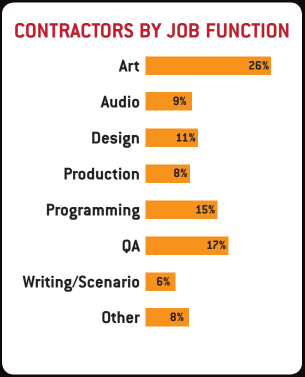
	PROGRAMMING	ART	DESIGN	PRODUCTION	AUDIO	QA	BUSINESS
High school/GED	—	—	$79,500	—	—	—	—
Some College	$104,907	$95,379	$77,500	$94,643	—	$45,000	$113,250
Associates Degree	$93,676	$79,737	$69,583	—	—	—	—
Bachelors Degree	$88,649	$71,680	$73,703	$81,685	$81,364	$49,667	$88,684
Some Graduate	$105,109	$71,667	$59,423	$91,667	—	—	$106,731
Masters Degree	$90,291	$59,868	$68,438	$90,870	—	—	$121,288
Some Doctoral	$93,864	—	$67,500	$107,500	—	—	—
Doctoral Degree	—	—	—	—	—	—	—

THE INDIE REPORT

THIS IS THE THIRD YEAR WE'VE collected data for our indie report, where we survey individual independent developers, independent teams, and individual contractors for their perspective on the industry. Out of those three groups, independent contractors made the most, though both individual indies and members of indie teams pulled in significantly more in 2011 than they did in 2010.

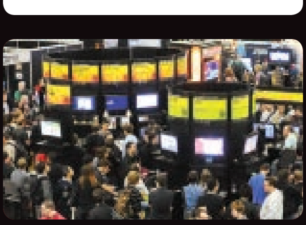
Independent contractors averaged $56,282 in 2011 (up $800 from 2010), individual independent developers averaged $23,549 (up from $11,379 in 2010), and members of independent developer teams averaged $38,239 (up from $26,780 in 2010). As the indie game community continues to mature and grow financially, it also seems to be consolidating somewhat. Compared to 2010, more independent developers are working in teams rather than going solo.

Indie games made a bit more money in 2011, too. 48% of independent developers made less than $500 from the sale of their game, down from 55% in 2010. 16% of independent developers made over $60,000 from the sale of their game in 2011, compared to 8% in 2010. Meanwhile, non-game revenue streams (non-game DLC/additional content, sponsorship or ad opportunities, and awards/grants), remained relatively hard to obtain—79% didn't receive any additional income whatsoever (down 2% from 2010). The developers that did cash in through non-game revenue streams generally didn't make a whole lot, either; 44% made under $1,000, compared to 35% from 2010. In general, it appears that the developers who are good at designing games to take advantage of non-game revenue streams are able to pull in a decent amount; in both 2010 and 2011, 40% of developers with these sources of income were able to make over $5,000.



INDIES BY JOB FUNCTION

Art	41%
Audio	18%
Design	52%
Production	37%
Programming	40%
QA	31%



JOB FUNCTIONS

While we survey both indie game developers and contractors for their job function in the game industry, we structure the survey differently to adjust for the difference between the two sectors. Developing a game in a small independent team means most developers don't have completely specialized roles—usually people are wearing multiple hats, so asking an independent games developer to only report one discipline wouldn't be completely accurate. On the other hand, that's not the case for most contract game developers. As such, the indie chart should be read as "what percentage of independent developers do at least this job function," rather than "how many independent developers do this job exclusively."

More and more indie developers are finding themselves in roles involving design (59%, up 7%), programming (53%, up 13%), QA (43%, up 12%), and production (47%, up 10%), while art and audio have declined slightly (40% and 17%, each down 1% from 2010). Considering our number of independent developer responses overall were roughly equal, this means independent developers are wearing more hats than ever before—a good indie team member is someone who can code, test, design, produce art, and manage a production schedule.

METHODOLOGY

NOW IN ITS ELEVENTH YEAR, the Game Developer Salary Survey was conducted in February 2012 for the fiscal year January 1, 2011 through December 31, 2011 with the assistance of Audience Insights. Email invitations were sent to *Game Developer* subscribers, Game Developers Conference attendees, and Gamasutra.com members asking them to participate in the survey.

We gathered 4,132 responses from developers worldwide but not all who participated in the survey provided enough compensation information to be included in the final report. We also excluded salaries of less than $10,000 and the salaries of students and educators. The small number of reported salaries greater than $202,500 were excluded to prevent their high numbers from unnaturally skewing the averages. We also excluded records that were missing key demographic and classification numbers.

The survey primarily includes U.S. compensation but consolidated figures from Canada and Europe were included separately. The usable sample reflected among salaried employees in the U.S. was 1,742, for Canada 403, and for Europe 339; and 524 for indies and independent contractors who provided compensation information worldwide.

The sample represented in our salary survey can be projected to the U.S. game developer community with a margin of error of plus or minus 2.4% at a 95% confidence level. The margin of error for salaried employees in Canada is plus or minus 5%, and is 5.4% for Europe.

LITERALS

In order to hear what developers are saying about the industry right now, we allowed space at the end of our survey for direct comments. Here are some of the more notable responses.

THE BAD

“The attitude toward work life balance is absolutely terrible. It is not an honor to work in games. Engineers are treated like garbage, especially when compared to the treatment they can easily get outside of games.”

“The desire for games to be fun, creative, and unique is slowly being diminished by the never-ending need for money, money which is best acquired by making derivative boring titles often seen in the social games market. The game industry is going the way of the movie industry with constant remakes and prequels and no new innovation besides independents who often give themselves too much credit.”

PHOTO COURTESY OF GAME DEVELOPERS CONFERENCE

“The longer I work in the industry, the less I can relate to gamers. The vocal minority is more annoying than ever.”

“This is an industry I've at times wanted to leave. I've often come very close to it. Because, plain and simple, it's not easy being a woman in this industry. Thankfully I've recently joined a company where my gender isn't used against me as an argument to dismiss me, and it's been a really empowering experience. Also, a lot of companies are closing, many are hiring. There's always a job open somewhere. But we need to stop the firing of employees once a game has shipped as a viable 'saving the company money' measure, and we need to find ways to welcome students within our ranks better, as opposed to always ask for only experienced personnel, thus not giving newcomers a chance to learn.”

“I'm still surprised by how much sexism/racism the industry exhibits, and by how difficult it is to change perspectives on it. While the industry as a whole is slowly improving, I frequently find myself trying to explain to coworkers why certain content—however hilarious they find it—might offend certain groups of people. There still seems to be a “boys' club” atmosphere in the office sometimes and many women are put in the unenviable and unfair position of political correctness enforcer.”

“It's been a terrible year in the U.K. There have been hardly any new design positions, and so many studios have closed.”

“Audio designers in the game industry are often treated as a disposable resource, and the opportunities for us, both inside and outside the game industry, are incredibly scarce, as sound is very much misunderstood and undervalued as a craft. There are bright glimmers, but we have to really band together, speak up, and make a lot of noise about examples of great sound design to help continue to validate our work. Great games where audio plays a prominent role like BASTION and PORTAL 2 really help us to move forward.”

“The outsourcing of art jobs has made this a less than desirable position for starting artists. Why spend thousands of dollars on an art degree, and countless hours perfecting your skills, only to come into a studio and do lackey clean up on work received from the outsourcers? A lot of talented artists don't even really create art anymore, just mundane clean-up tasks.”

THE GOOD

“I think the advent of crowd-funding and self publishing is going to see a huge shift in the coming years in terms of what games get made and who makes them. Indie game studios once again possess the tools and funding necessary to retake the game development industry much like they did in the '80s and early 1990's.”

“We're at a big turning point. The digital distribution model, along with the explosion of mobile gaming, is ushering in a new age of smaller studios and quicker development cycles. The age of triple A , 2–3 year games is coming to an end.”

“We're in a time of great change, not only in business models and distribution platforms, but also in ethics. I'm optimistic for the future—more people than ever are playing games and when this recession is over, we're going to see some incredible revenue across all aspects of the industry.”

“It seems that the industry is on the precipice of a creative revolution as SDKs and self-publishing options are getting more accessible, creative direction will come less from the publisher, who relies more on proven methods, and more from wild-eyed developers who would rather experiment than replicate.”

“Developers that have been able to adapt to the rise of social and mobile gaming have done well despite an overall economic decline in the world. Mobile and social gaming is exciting because it allows a developer to focus on simple, tight, and polished gameplay. In that regard I feel the recent trend is somewhat of a return to the 'golden age of games'.”

DON'T FEAR SOCIAL GAMES

“There is a lot to learn about game design and how to appeal to a wide audience. Think about the average time until someone who has never played a game before starts having fun in your game—social games do this better than most traditional games. This is an incredible and exciting thing. If some of the early companies in social games were unscrupulous, don't let this keep you from learning exciting lessons from social gaming and let this improve your own designs.”

“Video games!!!!!!!!”

NOW HIRING

ART

Studio Art Director / Senior Character TD /
Senior Character Artist / Senior Environment Artist

AUDIO

Audio Lead

DESIGN

Lead Level Designer / Senior Designer

ENGINEERING

Senior A.I. Engineer / Senior Multiplayer Engineer /
Senior Graphics Engineer / Senior Core Engineer /
Senior Backend Engineer

Apply online at **jobs.lucasfilm.com**

LUCASARTS
A LUCASFILM COMPANY

MAKE YOUR MARK IN GAMES

Full Sail University game degree grads have taken their talents to the titles below and then some. If you want to make an impact in the world of gaming, Full Sail's industry-focused degree programs can show you what it takes to get there.

DEGREE PROGRAMS IN:

- Game Art
- Game Design
- Game Development

FULL SAIL UNIVERSITY

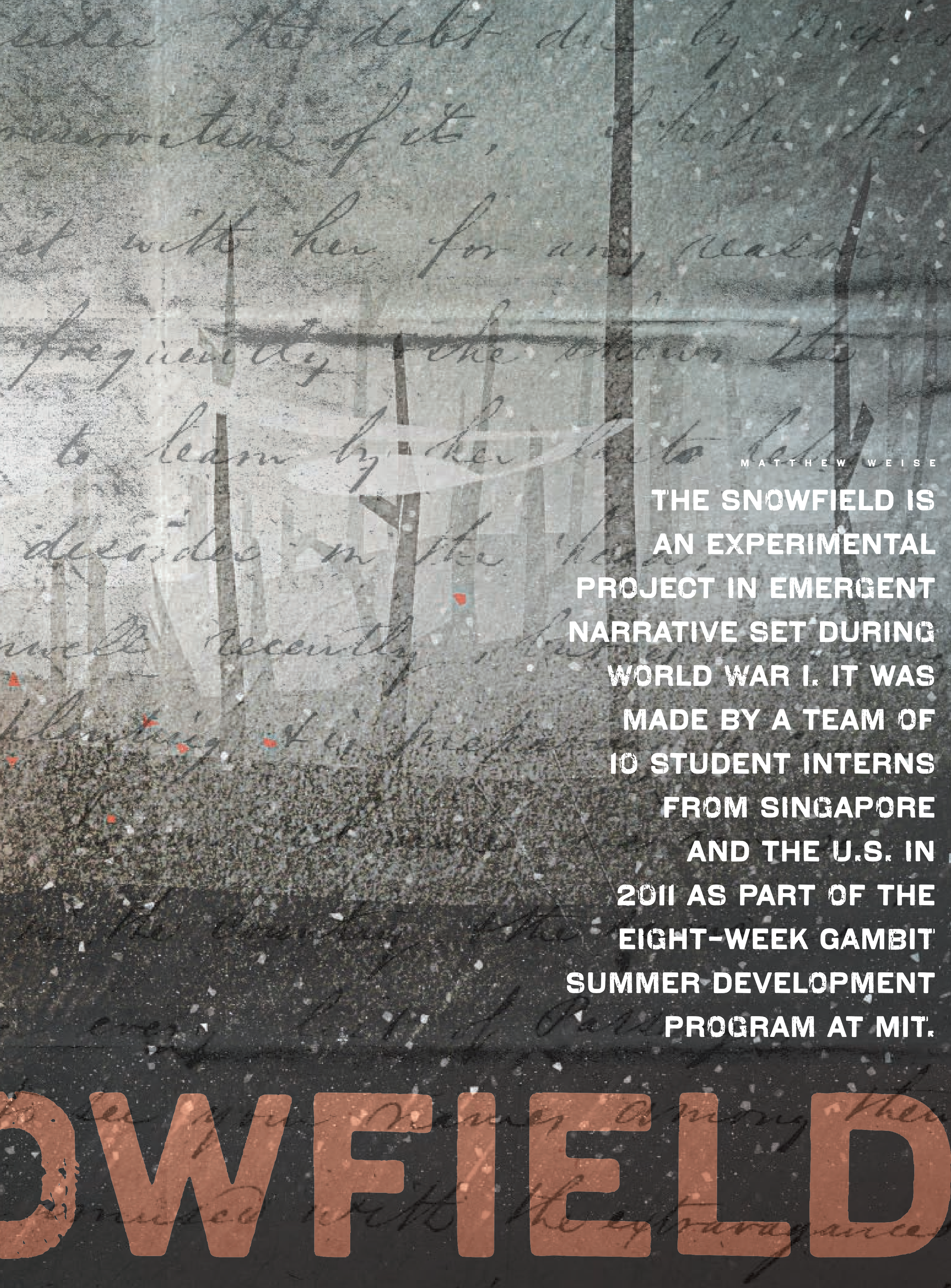
800.226.7625
fullsail.edu

3300 University Boulevard • Winter Park, FL

Financial aid available for those who qualify • Career development assistance • Accredited University, ACCSC
To view detailed information regarding tuition, student outcomes, and related statistics, please visit fullsail.edu/outcomes-and-statistics.

POSTMORTEM

THE SNO

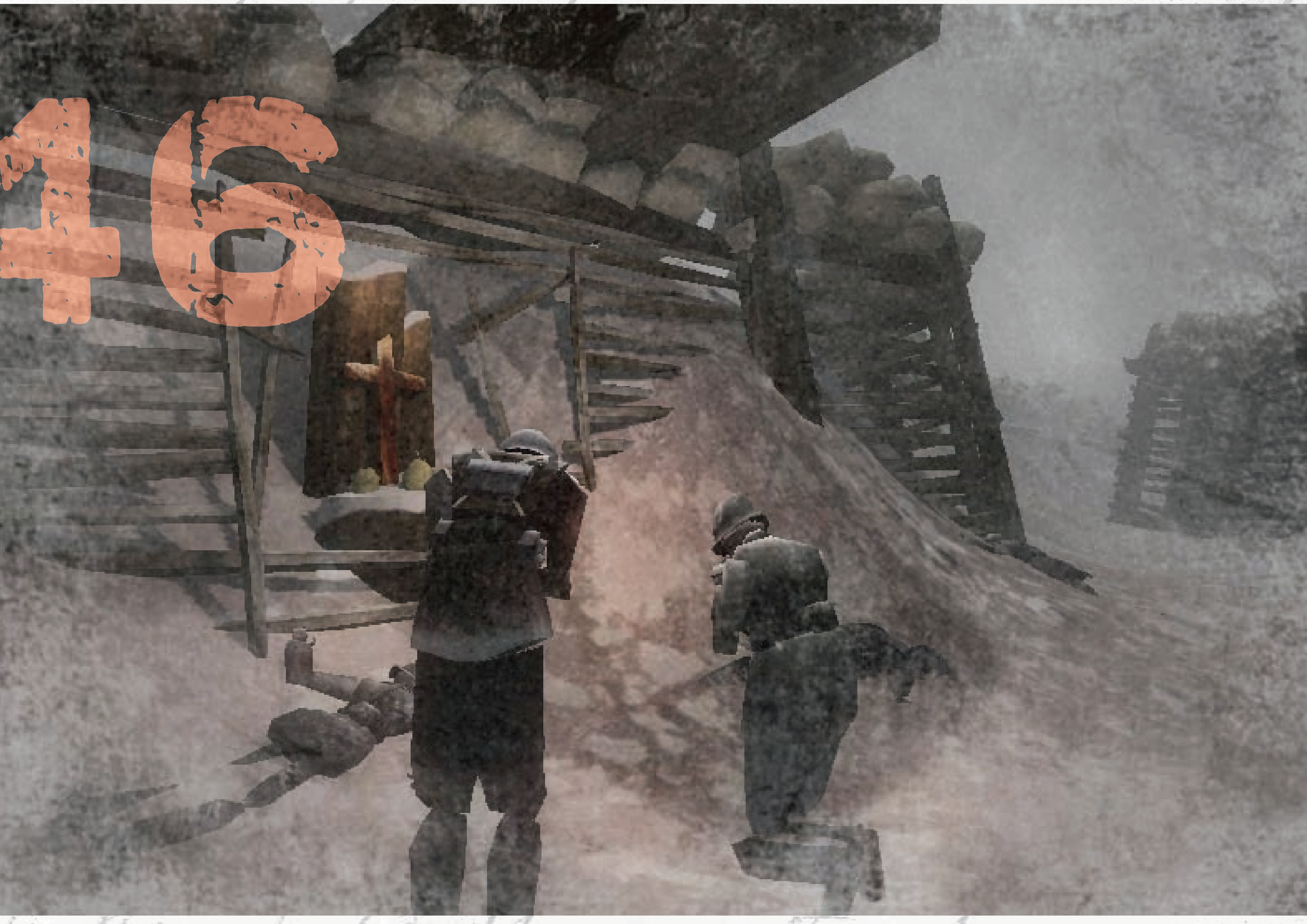


MATTHEW WEISE

THE SNOWFIELD IS AN EXPERIMENTAL PROJECT IN EMERGENT NARRATIVE SET DURING WORLD WAR I. IT WAS MADE BY A TEAM OF 10 STUDENT INTERNS FROM SINGAPORE AND THE U.S. IN 2011 AS PART OF THE EIGHT-WEEK GAMBIT SUMMER DEVELOPMENT PROGRAM AT MIT.

OWFIELD

THE SNO



I was the product owner for THE SNOWFIELD, as it was based on theories of narrative design I had developed over several years as GAMBIT's game design director. Our goal was to take these theories, turn them into a clear method for development, and test that method with a game. THE SNOWFIELD was a response to the idea, in both industry and academia, that there are principles of good storytelling that we need to systematize in order to marry video games with narrative effectively. This has led to ideas like "drama management," which assumes that emergent narratives require an A.I. storyteller who controls events to ensure the emerging story is satisfying.

Such notions are built on an assumption that "good storytelling" is somehow an objective, with rules we can apply to achieve consistent results. Storytelling, as any storyteller who isn't a Hollywood huckster will tell you, is not science. I wanted THE SNOWFIELD to illustrate how we could build an emergent, emotionally rich, player-driven narrative experience from the ground up by relying solely on art, audio, and character A.I. (refined via player testing), instead of trying to identify formulaic notions of "good" storytelling and work them into the game beforehand. I wanted the team to develop an open simulation with strong emotional hooks and minimal dissonance and see if the user's imagination would do the rest.

WHAT WENT RIGHT

1 ambition One of the best things we did with THE SNOWFIELD was to make it a fully 3D game with a semi-traditional over-the-shoulder camera. This greatly contributed to the atmosphere and general emotional impact of the game, which ended up being our saving graces when other aspects did not turn out as planned.

Early in development this was considered a risky move. Traditionally, the GAMBIT summer program advises interns to avoid 3D whenever possible, since it adds an extra layer of complexity for things like collision, modeling, texturing, and lighting. Using 3D also can put interns in the frame of mind that they might achieve results similar to a triple-A game in terms of detail, which means they're likely to spend more time wrangling technology than actually cutting their teeth on game design.

We talked a lot about whether to go 2D or 3D during the first few weeks of brainstorming and paper prototyping. As the product owner, I was against 3D. I felt our narrative goals might only be achievable in the eight-week time frame if we used a highly abstract art style, like pixel art or stick figures. I thought this would allow us time to generate enough assets to sufficiently populate the large number of narrative events and situations we were planning.

In retrospect, I may have been right—our decision to go full 3D did have a major impact on our ability to implement a large number of narrative content. However, I'm not convinced the game would have gotten the recognition it did had it been a highly complex pixel art game. The immediacy of the horrific audiovisual design of a WWI battlefield was largely cited as the defining feature of the game, and the main source of its emotional impact.

Also, by making The Snowfield a 3D game with an over-the-shoulder perspective, it became a sort of commentary on traditional triple-A war games like Gears of War, which sport similar conventions. Since we wanted to depict war in a way that counterpoints most commercial games, speaking their language had certain advantages; The Snowfield could play upon the player's expectations to drive additional narrative impact.

2 unity Choosing Unity was absolutely the right move for this project. Even though it had certain drawbacks and caused certain problems, without it we would not have gotten the results we did in 3D within the same time frame.

Unity's easy-to-use interface and readily available tutorials allowed even people who'd never used it before to produce on-screen tests and prototypes quickly. It is very easy to create a 3D space, place objects with behaviors, set lights and collision, and go. In fact, one of the reasons the team decided to go 3D was because Unity made it so easy to take our prototypes and move the camera around to see how it would look from a variety of angles.

Special kudos have to be given to Unity for making our sound design pipeline a snap. Because it is very easy to move sound sources around in Unity, our sound designer easily filled the world with an emotionally rich, haunting soundscape without fighting the tech or taking time away from programmers, which played a large part in making The Snowfield as effective as it was.

3 artist in charge One of the myths about narrative we wanted to discredit was the idea that plot structures or archetypes are what make players care about game stories—in other words, what makes players react emotionally. Traditionally, game developers assume narrative creates emotion. We believed emotion creates narrative, and that emotional touchstones are what cause players to perceive a set of system outputs as a story.

To me this meant artists should be the backbone of whatever narrative impact we were creating. Just as the right bit of acting or music creates an emotional effect in film, I felt the right character animation or sound is what creates emotional effect in a game. This is why I chose to partner with Jason Beene, GAMBIT's art director. As the project's game director, he mentored the team's management process, but he also set the tone for the project as an art-focused project, led brainstorming, and heavily emphasized The Snowfield's art from beginning to end.

Jason is one of the main reasons the project went 3D, since he championed the emotional possibilities of adopting the over-the-shoulder view. He also pushed back heavily on my desire to include text (both descriptive prose and dialogue) in the game. My initial vision for The Snowfield was more like a comic book or a Japanese RPG from the 1990's, which would use icons, text, and characters in various combinations to produce different effects. Jason pushed for a more singular, simple approach because he felt it would be more emotionally direct. While I do think combining styles can work well, I do admit that Jason's simple approach was the right choice; text certainly would have distracted from the simplicity and directness of the final product.

Jason also pushed the team to be the best artists they could be. He insisted they do proper reference gathering and concept creation before getting started on assets, and had them post the images that inspired them around the room. Jason wanted the artists to work in an environment of imagination, where you could get an instant impression of the project just by looking at the walls, and it was a great asset to The Snowfield's process and the final result. Art is often not leveraged the way it could be in game development. Artists need to be treated like drivers and collaborators of game design and narrative, not just illustrators for decisions that other departments make.

4 veto power The Snowfield was a student intern project guided by two professional game development staff mentors. Jason and I therefore had to ask ourselves when (if at all) we wanted to play the authority card when we felt it would greatly benefit the project. This only happened once, but in hindsight we made the right choice.

At the end of our extensive brainstorming process, in which the entire team participated equally, we put up all the ideas and had a vote. It came down to two ideas: one about being the last survivor of a WWI battle, and the other about being a child living on a surreal, tiny planet (à la *Le Petit Prince*). When we first voted, the tiny planet idea beat the WWI idea by a single vote.

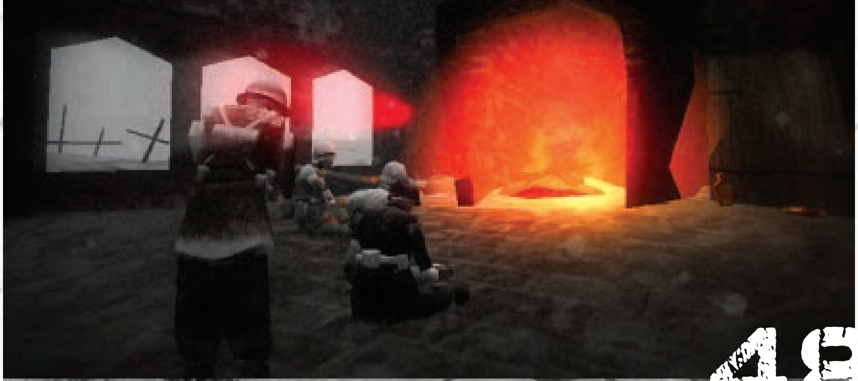
Though we also liked the tiny world idea, Jason and I felt the WWI idea was the smarter choice for this particular project. We felt that because the goal was to create emotion with as minimal development work as possible, the WWI concept had the advantage of having most of its dramatic conflict and emotional context built into the setting itself. The core situation suggested drama right out of the gate, whereas with a surreal world the team would have to work harder to set up context, establish world rules, and then create situations for characters to react against.

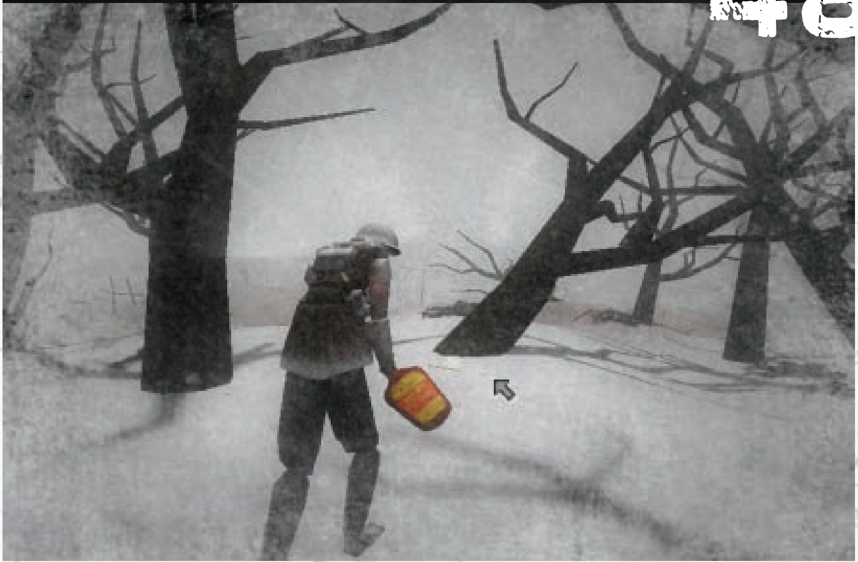
We decided to flat-out tell the team to use the WWI idea, and hope that the morale hit would not be too great (which it was not). Living up to the obvious dramatic possibilities of an existing world is a very different challenge than trying to create dramatic possibility in an imaginary world. We thought the historical setting would take the pressure of creating an entire world off the team so they could get right to modeling and simulating dramatic elements.

The WWI scenario also helped The Snowfield get more recognition due to the unusual nature of its content. There are many whimsical, surreal games out there, especially in the indie/student space. There are virtually no games about WWI, and almost no war games that don't involve shooting. Though whimsy felt like a more "fun" theme to spend the summer on, the sanity-shattering horror of the most awful war in history had the benefit of making a more striking impression. Ultimately we wanted this game to affect people.

5 follow your own advice Near the end of the project we were running into serious problems getting in all the narrative content we had planned. We had prototyped on paper many different character types, object types, and behavior types. Though simplistic, this made a lot of sense in a straightforward way, was thematically relevant, and allowed us to at least finish the game by giving it a goal structure, all

> I HAD GRAND VISIONS OF A DEEP, RICH, EMOTIONAL SIMULATION, WITH HUNDREDS OF POSSIBLE DRAMATIC SITUATIONS—ALL SOMEHOW CONCEIVED, PROTOTYPED, BUILT, IMPLEMENTED, TESTED, RE-IMPLEMENTED, TESTED AGAIN, REFINED, TUNED, AND POLISHED IN EIGHT WEEKS WITH AN ALL-INTERN TEAM. I KNEW THAT WASN'T POSSIBLE, BUT I THOUGHT THAT BY AIMING HIGH WE'D BE ABLE TO RELEASE SOMETHING MODERATELY DEEP. I WAS WRONG...





of which interacted in complex ways to produce emergent narratives. After all, we wanted to prove that we could make an emotional, replayable narrative. The art and sound was intended to achieve the emotion, while the systems depth was intended to achieve the replayability.

One of our paper-prototypes involved a shell-shocked woman who would follow you, a crazed soldier who stole things, and land mines that would explode if you stepped on them. During one prototyping session this generated a story where the player befriended the woman and began searching houses with her for food. But every house they went into had no food, and eventually they found out the thief was traveling to all the houses and stealing it. They found him, assaulted him once, and chased him away. Then, later, when the woman had found some food, the thief appeared again and kidnapped her. (Since his behavior was "pick up food" and she had food, he picked her up!) This resulted in a chase that ended spectacularly when the player stepped on a land mine and blew up, causing the thief to drop the woman, who then escaped with the food.

These were the kinds of emergent narratives we wanted our system to generate, but we had implemented almost none of them by the final two weeks. We had a nice atmospheric demo, but it was a mood piece, not a game. There was no goal, no end state, and we had only implemented one character type (the soldier) and one behavior type (following). Given that we had prototyped dozens of behaviors, and that the project itself in theory hinged on achieving a critical density of behaviors to even generate narrative, this was a potential disaster.

We had four days, a short grace period before code-freeze, to turn what we had into a game. We could not add features. We could only connect existing features in ways they had not been connected. Since our feature set was so small, we had to be radically creative. One of our artists suggested designing the game entirely around the "follow" behavior, since it was the only one we had, and make the game about leading stray freezing soldiers back to the fire

We made a similar decision earlier that summer about THE SNOWFIELD'S freezing-to-death mechanic. Originally, freezing was just a gating mechanic, to prevent players to reaching the edge of the map without resorting to invisible walls, but when testers responded so emotionally to it (nearly all of them said it made them feel "cold and alone" in the heat of July) we decided to embrace that simple emotion and build the game around it. This was an outgrowth of our "emotion creates narrative" philosophy, as something that guided development as well as in-game player narrative.

THE SNOWFIELD may not have been the deep emergent system I had hoped, but the fact that we were able to look at even the smallest number of mechanics we had and find ways to leverage even their emotional and thematic value was ultimately an illustration of how hard it is to follow your own advice sometimes. The goal was to recognize whatever emotions happened to be created by the features we had, and tailor the user experience to them. The fact that we were forced to do this with an insanely small pool of features was frustrating, but it allowed the team to finish the project without compromising our method entirely.

WHAT WENT WRONG

1 personality problems We learned early on that the people on the team with the loudest ideas were not necessarily the people with the best ideas. GAMBIT historically does not assign a strict hierarchy of creative control within teams; we prefer to see who tends to emerge as a strong visionary and then encourage them to accept that role, whether they be designer, artist, producer, QA, sound designer, or programmer.

On THE SNOWFIELD this approach backfired a bit. One of the team members, who was very outgoing and with a strong personality, ended up asserting his will in ways that were not especially in line with the project's overarching goals. Meanwhile another team member, who was shy and soft-spoken, really understood the goals well and had excellent instincts but tended to defer to the louder, more aggressive team member.

Jason dealt with this problem the best he could, by giving the soft-spoken team member more explicit authority, but after a certain point it really comes down to the people themselves, how willing they are to accept responsibility and work things out on their own. We could have easily stepped in and made decisions for the team, but since GAMBIT's goal is both training and quality, we struggled (as is typical in GAMBIT) to find a balance between letting the team learn from failure while ensuring we got an acceptable final product.

In the end, the team got through the summer intact, but Jason and I also found out afterward—after the interns had left—that tensions on the team ran even deeper than we had realized. Had we known earlier we might have been able to reallocate or swap team members to find a better balance of personality types. On the other hand, sometimes the only way to discover these things is to watch them play out. You can never be sure.

2 unity Though Unity basically made the project possible, it also caused a lot of grief that we had not anticipated. The version we were using got very finicky with code merging near the end of the eight weeks, just when things began to really heat up. Features and fixes that our frazzled, already-struggling team spent hours implementing would disappear for no apparent reason. For a while team members thought they were going (more) crazy, until we realized it was a merging problem.

There was no time to fix this, so we adjusted our pipeline so that only one of our 10 team members could ever have the project open on a single machine at a time. This was a major pain to say the least, turning the team's process into a tedious relay race. It was a struggle to find effective use of other people's time (especially programmers) while they were not allowed to work directly on the build. In some cases people just ended up crowding around a single machine for hours, or taking turns writing code on a single machine. The team got through it in the end, but it was a monkey wrench on top of everything else at exactly the wrong moment. If we'd had proper version control, the team would have had fewer setbacks and might have accomplished more.

3 two drivers Jason and I had worked together, as artist and designer, on GAMBIT internal projects prior to THE SNOWFIELD. These were game prototypes developed entirely by GAMBIT staff, not involving student interns. Our collaboration was very healthy and involved, and we collaborated closely. I respected his design input, and he respected my art input. We thought that co-managing a student project with me as product owner and him as game director would be a cinch.

Then we found out how different our management styles were. Jason's job prior to joining GAMBIT had been an art manager at a game company, and mine had been producer at a game company. We therefore had our own instincts on how to run a team, and this created some pretty distressing friction between us before long.

To make matters worse, the producer on the team was supposed to report directly to Jason, but he had developed a rapport with me and tended to come directly to me when he wanted advice or direction. Because the project represented my theories of narrative design, I was personally invested and determined to get the results I wanted. For this reason I tended to grant the producer advice, which was often contrary to Jason's.

Jason and I eventually clashed over this, and I decided to back down and let him run the show. This is what I should have done originally, because that's how the product owner / game direction relationship is designed to work. We should have worked out our differing approaches and boundaries of responsibility more clearly beforehand, and when the difference became clear I should have brought it up with him immediately, rather than going behind his back. It would have been better for us, and less confusing for the team.

4 too much brainstorming On THE SNOWFIELD we had a very organic brainstorming process. We literally spent days sitting in a circle, writing down ideas, sifting through them, and waiting for a consensus to emerge. While I don't think this is a bad process, it did slow the team's momentum early on. Certain team members were itching to get started, especially once they saw other teams begin to prototype environments and mechanics at the beginning of the second week. We certainly wanted to instill in the team a healthy appreciation for proper brainstorming, but we might have overdone it a little bit.

I tend to think of the eight-week GAMBIT summer program as a long game jam, where lost seconds can mean lost features. While I think the brainstorming process we put the team through was necessary, I do believe we could have probably gotten as good results if we had spent a day or two less, and also let the team members who seemed less invested get to work on things like tech exploration sooner.

5 vague theories If I had to trace back most of these problems (especially the excessive brainstorming) to a deeper underlying problem, it would be that I failed to make my method of narrative design clear early on. I had a collection of notions surrounding narrative I wanted to explore, but they were not as organized or refined as they should have been to give both Jason and the team proper direction in the first few weeks.

To make matters worse, I had an entire semester before the summer where I explored the narrative ideas I was interested in, using a different student team. This semester was intended as prep and exploration for the summer, and the goal was precisely to refine my theories into a coherent method I could give to the summer team when they arrived.

While the spring team did very good work, and helped me refine some of my ideas, I did not sit down and look at all the data they collected on my own until a few weeks before the summer. At that point I did not have enough time, before traveling to Singapore for the two-week intern orientation, to properly organize what I had into a clear set of methodical guidelines. I told myself that because I was product owner, I would be on hand to clarify things for Jason and the team, but this was just a rationalization for being lazy. The fact was I had not adequately clarified my ideas to myself.

Never allow yourself to believe that, just because you feel like you have a strong intuitive grasp of something complicated, you truly do. If you can't write it out, draw it out, or otherwise express it cleanly in some media form, you do not really understand it.

BREAKING THE ICE

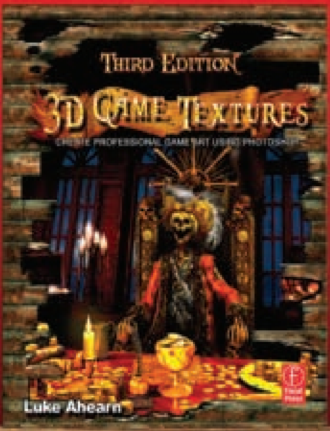
Given THE SNOWFIELD'S ambition and various difficulties, I'm surprised that the project did not self-destruct. I had grand visions of a deep, rich, emotional simulation, with hundreds of possible dramatic situations—all somehow conceived, prototyped, built, implemented, tested, re-implemented, tested again, refined, tuned, and polished in eight weeks with an all-intern team. I knew that wasn't possible, but I thought that by aiming high we'd be able to release something moderately deep. I was wrong, but the team kept things together enough to release a version of THE SNOWFIELD that still illustrated the theory/method. The fact that this minimal version retained such impact is a testament to the team's heroic commitment to creating an intense audiovisual experience in spite of everything.

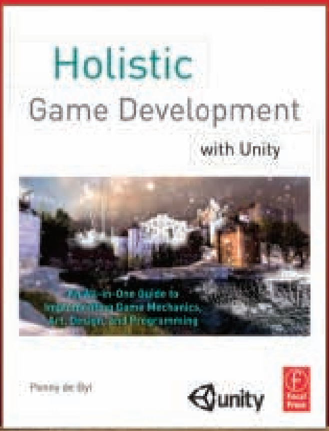
THE SNOWFIELD team at work.

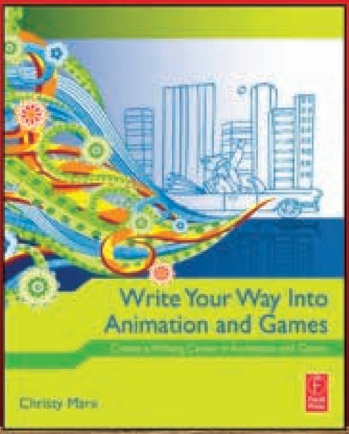
One is supposed to admit failure, since analyzing failure will help lead an entire community of practice toward gradual understanding. While THE SNOWFIELD was a successful emotional experience, a successful mood and atmosphere piece, and a successful virtual evocation of World War I, it was not a successful illustration of my theories on narrative. The critical mass of interacting behaviors necessary to create a dense possibility space was never achieved, nor was a flexibly abstract-yet-emotional visual style to go along with it. I'd still love to see that kind of combination happen, though I now feel it might be better suited to something like a MINECRAFT mod than a from-scratch student project.

Still, getting a haunting WWI survival sim out of it is itself something to be proud of. The video game industry has put way too much energy into making war fun, when there's a million other things we can and should be saying about it, as only digital games can. Successful narrative experiment or not, if The SNOWFIELD at least managed to sucker punch a few people, we've done a public good.

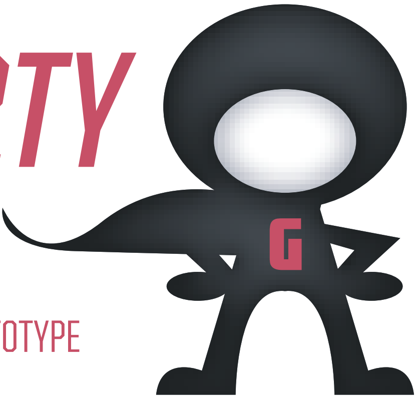
MATTHEW WEISE *worked for five years as game design director for the Singapore-MIT GAMBIT Game Lab. His writing on narrative game design has appeared in various books, online publications, and can be found regularly on his blog at outsideyourheaven.blogspot.com. He now works as a narrative designer at Harmonix Music Systems.*


Third Edition
3D Game Textures
Create Professional Game Art Using Photoshop
Luke Ahearn
Focal Press


Holistic
Game Development
with Unity
An All-in-One Guide to Implementing Game Mechanics, Art, Design, and Programming
Penny de Byl
unity
Focal Press


Write Your Way Into
Animation and Games
Create a Writing Career in Animation and Games
Christy Marx
Focal Press

QUICK AND DIRTY GAME ART



MAKE PLACEHOLDER ART FOR YOUR BRILLIANT GAME PROTOTYPE

BY ROBIN DE JONGH

So you've downloaded an amazing game-development SDK, or you've learned the basics of a scripting language, or you're already a coding genius. You're all ready to work, except for one problem: You have no art assets to work with.

When you're making a game intended to show off your talent, it's important that you don't draw attention to your weak points. If you are pitching yourself as a coder, your game shouldn't look like it came from a four-year-old's sketchbook (unless it's supposed to). Your code might be immaculate, but your audience will be too distracted by the hideous visual style to care. You can get around this by making art that looks visually appealing, clean, and finished—but still temporary. Stylization is your friend!

Let's say you have an idea for a great game and a clear idea of how it should look in your mind's eye. Now, take one item at a time from that imaginary ensemble and strip it of color, shading, and fine details until you're left with a form that represents your object. It is not your object—your object will be far more detailed, colorful, and alive than this. Take this stripped-down object and quickly draw it with a pencil. What you have now is a concept asset, and if you keep all of your game art in this style, anyone playing your game will see that all it needs is a hotshot artist.

Of course, you can't use a pencil-and-paper sketch for your prototype's art, because you'll need the geometry and size to be consistent in order for you to handle collision detection, physics, and so on. So we're going to use SketchUp to build our assets instead.

STARTING WITH A SKETCH

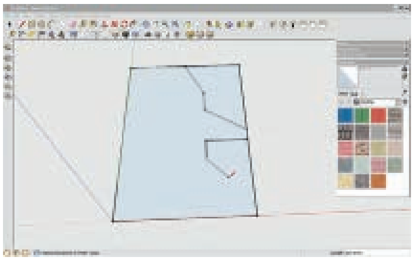
» SketchUp (sketchup.google.com) is a free, easy-to-use tool that's well suited to making 3D models and 2D art. We're using SketchUp for this article because it has display features that make it easy to make your art look like a preliminary sketch, which is important for reminding your audience that the game is still a work in progress. Yet the assets you create will be geometrically identical to the finished asset you might commission in the future, so you can develop and test your game code as you would the finished article.

(Note: At the time of this writing, SketchUp is in the process of being acquired by Trimble. Trimble plans to keep the free version of SketchUp, so the contents of this tutorial shouldn't change, but the download URL might.)

SPRITE SHEET IN 20 MINUTES

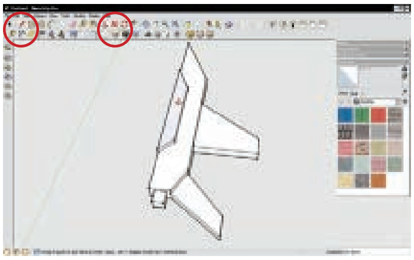
» Every game developer should know how to create her own sprites. Downloading someone else's stuff from the Internet is just going to limit what you can do with your prototype's visual style and how you can move a sprite, and you don't want anything to impede your creativity.

Download SketchUp (the free version), install it, and fire it up. In the first window, click Choose Template and select a template that suits you (Millimeters, Meters, or Feet and Inches), then click Start Using SketchUp. Once the main SketchUp window opens, select the Rectangle Tool on the left-hand side of the toolbar, and draw a rectangle that you're going to use as a box that borders your new sprite. Note that depending on which template you select, you will either start with a top-down or isometric view. Click the top view icon or use the middle mouse button to rotate the view so that you easily see the red and green axes. We're going to draw a spaceship next.

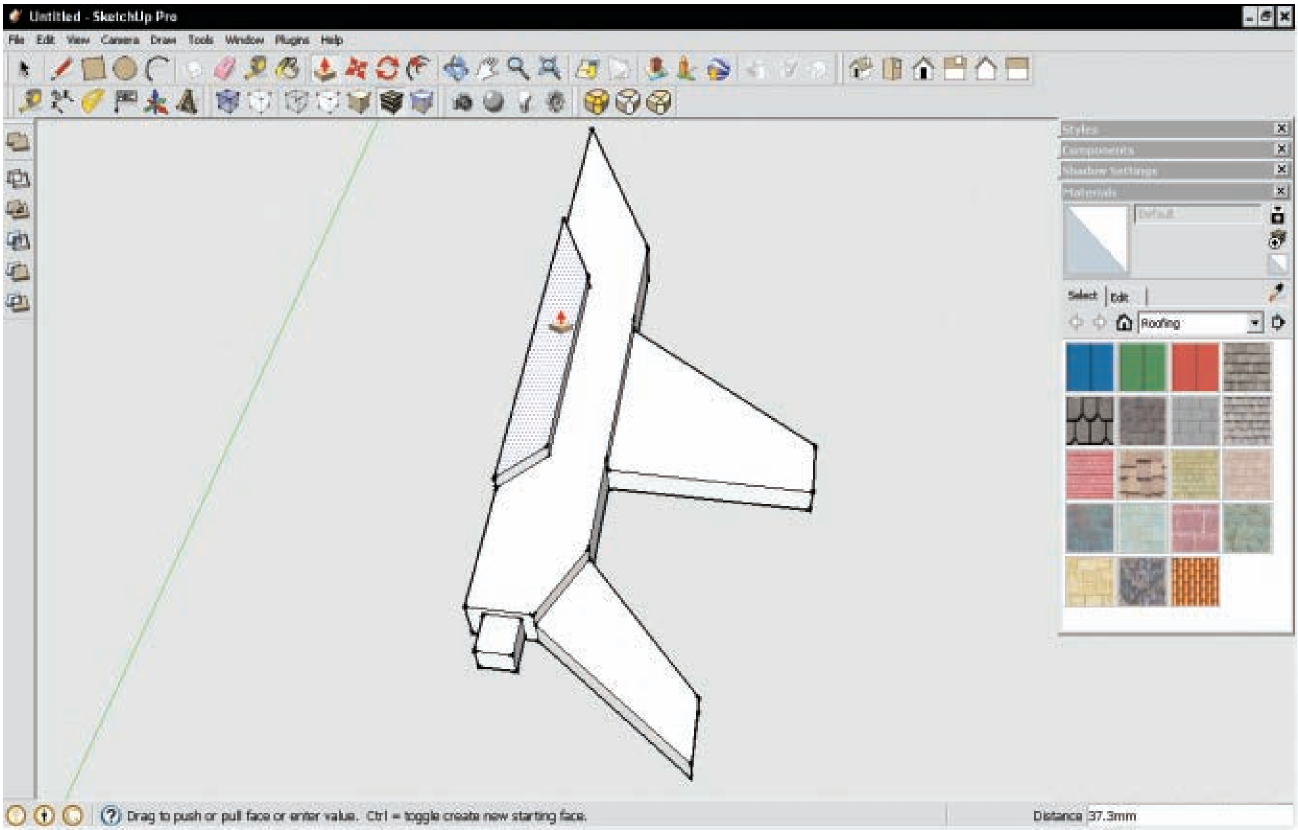


Draw an outline of a spaceship in SketchUp.

Select the Pencil tool, and start drawing a rough outline of half your spaceship at the middle of the top edge of the rectangle. Erase the rest of the rectangle, and then use the PushPull tool (a cube with a red arrow pointing upward) to give the shape some thickness. You can continue to use the pencil tool to draw in a little more detail here and there, and whenever you draw a closed shape, just use the PushPull tool to add depth.



Use the Pencil and PushPull tool to draw shapes and give them depth.



Drag to push or pull face or enter value. Ctrl = toggle create new starting face.

Distance 37.3mm

Set a background color to isolate the sprite for the sprite sheet.

Now triple-click on your half spaceship to select the entire thing, and then right-click and choose Make Component. Use the Move tool and hold Control to drag a copy of your half-ship to the side (Option-click if you're on a Mac), and then right-click and select Flip Along, Red Direction. Move this half to join it with the original half. You now have a finished ship shape! Select both of these halves, right-click again, and choose Make Component again, and then select Set Component Axes and move the axis to the middle rear of the ship. This allows you to rotate it more easily, which will come in handy later.

LOOKING SKETCHY

» We need to make sure your spaceship looks like placeholder art, so go to Window, Styles, and select Sketchy Edges from the drop-down list. Select any of the styles you like, and then go to the Edit tab, select the Background icon (the one with the empty cube in the middle of the Styles palette), and click Background, and then select a blue or green color. This will make it easy to isolate the sprites in your image-editing software later.

Now we're ready to build your sprite sheet. Go to Camera, Standard Views, choose the Top View, and then go to Camera and choose Parallel Projection.

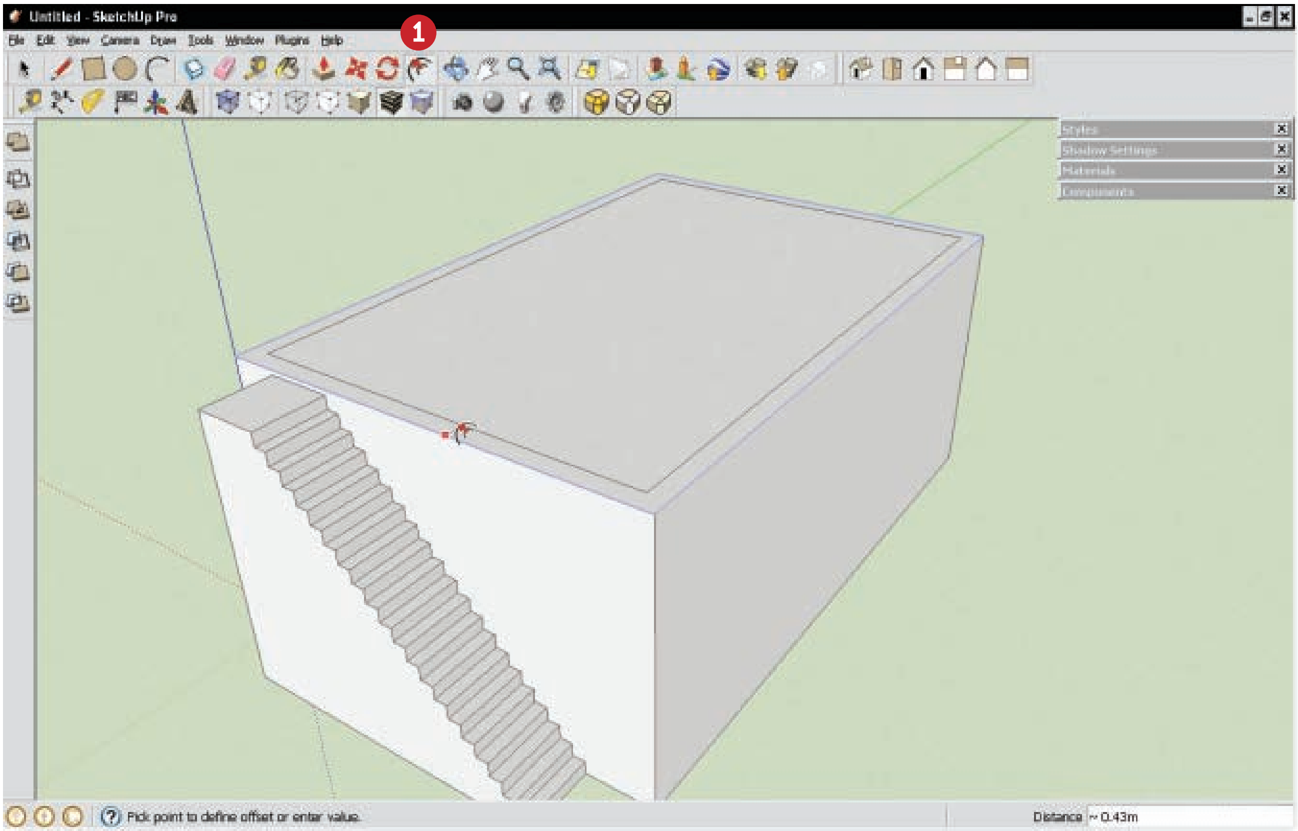
From here, you can create your sprite sheet by moving, copying, and rotating with the Move tool. Hover the mouse over the sprite and you'll see some red crosses. Click one and move the mouse to rotate the spaceship. This is better than the Rotate tool in this instance because the Move tool always rotates around the sprite's center point. Control-click (Option-click if you're on a Mac) and drag the mouse to copy it. Rotate your ship from the top, rear, or side camera views to achieve the rolling, banking, or turning effect you want.

Export your image as a .png file (untick the anti-alias box in Settings), open it in your image editor, and then select and delete the blue background to make its background transparent. Now, all you need to do is use a sprite extraction tool like ShoeBox (http://renderhjs.net/shoebox) to create individual numbered sprites for use in Cocos2D, Corona SDK, or other 2D game development tools.

BUILDING BASIC 3D-LEVEL GEOMETRY

» These days, many smartphones feature a quad-core processor and a 720p widescreen display, both of which are perfectly capable of playing full 3D games—so it's important that you supplement your programming résumé with some 3D asset creation skills. And it's not just good for job hunting, either. If you have some basic 3D props and scenery modeling skills, you're more able to tweak game mechanics on the fly. Maybe you need to narrow a gap between two obstacles a player needs to jump over or modify the height of a wall a player can shoot over. You'll need to be able to judge distances, sight lines, hearing range, and other similar factors you can't do with a pencil and paper. Let's walk through building some simple 3D level geometry in SketchUp.

First, open up SketchUp and select the Google Earth Modeling-Meters template. Use the Rectangle tool and click on the Origin point (where the red, blue, and green axes intersect). Draw a rectangle from here, and then type in "10, 15" and press Enter. Use the PushPull tool to give this rectangle some height (about six meters). This is your building. Use the Pencil tool to draw a line the height of a single step (riser), and then another line for the depth. Select these, and



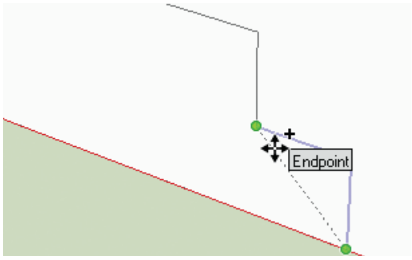
Untitled - SketchUp Pro

Pick point to define offset or enter value.

Distance ~0.43m

Build an opening in the roof with the Offset tool.

then use Move to select the bottom of the riser, and hit Control (Option on the Mac) to copy, and click where the next riser should start. Cut a single step into the side of the building, and then copy it several times (you can type in 20x and press Enter to create a whole staircase). Delete any excess lines at the top of the building with the Eraser tool, and then draw a line down from the edge of the building to complete the side of the staircase. Then use the PushPull tool to extrude the steps out.



Use the PushPull tool to extrude a set of steps out of a building.

Double-click the roof and use the Offset tool **(1 above)** to create a rim inside the roof. PushPull this up to create a roof parapet, and then use the Rectangle tool and PushPull to create an opening in the parapet by the top of the stairs. Go to Window, Components and select a person, and then place him on the roof.

Now you have a building with some steps and a parapet that you can import and test in your game engine. You can now work out whether your character can walk up the steps, decide at what height you wish to trigger damage (or end the game) if he falls off the steps, decide whether the parapet is low enough to look over but too tall to jump over, and so forth. You can test your game scripting decisions and environment rules before asking someone to model game assets. This is called "grey boxing" or "blocking out" in the industry.

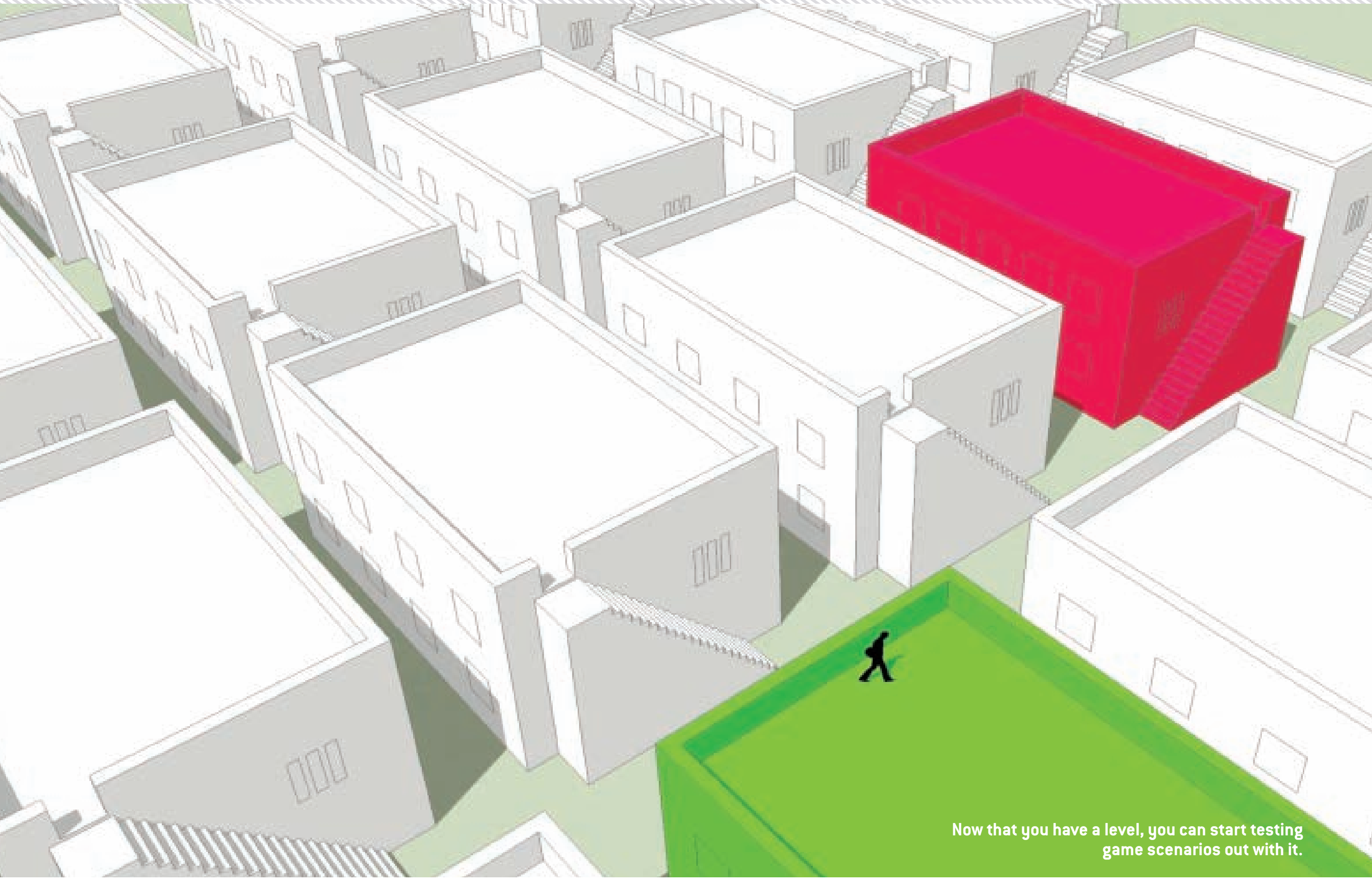
Triple-click on the building and turn it into a Component, and then copy it around a few times to create a block of buildings. Double-click on a component to edit it, and then draw some rectangular windows with the Rectangle tool. These should now show up on each and every building.

Add windows to your buildings.

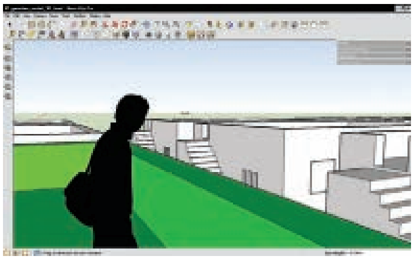
None of this looks very polished, which is important: Keep assets simple during the development process, and you and your team can focus on more important geometric and dimensional level design issues until you're ready to pass your SketchUp files on to an artist who can finish them using your geometry as a base. This way, the artist can add visual appeal but won't be able to "break" the development work you've already done.

TESTING GAMEPLAY WITH YOUR NEW LEVEL

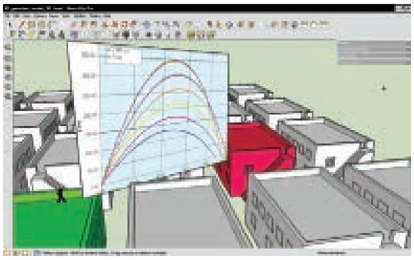
» Now that you have a level, you can use it to test out potential in-game scenarios. Let's say your character has found the end boss in the building



Now that you have a level, you can start testing game scenarios out with it.



Adjust the camera so you're accurately reading the player's perspective.



Map out the different possible projectile trajectories on a graph and paste it into your level and you can quickly determine how your player can hit the shot from afar.

shown in red. Your game is in a historical or post-apocalyptic setting, and the player could have a bow and arrow (at best) or a rock (at worst) to knock the boss out with. Imagine that the level-design specification requires that any of these weapons should be able to render the boss unconscious, which then triggers an end-of-level cutscene. The player should be able to finish off the game with a lucky shot from the green rooftop through the window of the red building while the boss is pacing around the top floor, so you'll need to know the range, flight-path, and decay rate for each weapon to work it out with your 3D mockup.

First, make sure there is a line-of-sight connection from the green rooftop to the red window by going to Camera, Position Camera and clicking the corner of the roof of the green building. Now left-click and drag the mouse to look around. Look at the person you placed on the roof. If his eye level is on the horizon, you're at the right height. Otherwise type in a height and press Enter until it feels correct.

From here, look at the red building and check if you can see the window. You can walk around the map as if you were in your finished game by right-clicking and choosing Walk. Move the target building or window if you need to. Now select the Tape Measure tool (or press T), click on the parapet where you're standing, hold Control, hover over the target window, and note the distance. Then create a trajectory path by using the Pencil tool to draw a line from the parapet to the window, move the pencil up, snap to the blue axis, do the same at the other end, and then complete the rectangle. You can now draw any trajectory you like on this canvas using the Arc tool, or if you're really clever, do some math and paste the resulting graph on it. Will the incoming angle of the projectile clear the window frame to hit the boss on the head or just bruise his big toe?

You can see where we're going with this: It's easy to do the hard work of level design before you ever touch the code—if you know what you're doing. Now you can import your new mockup assets straight into your game engine and continue development while a 3D artist is polishing it up at the same time (or build perfectly functional assets for your prototype so you can get hired). Maybe you won't need those accounting classes after all.

ROBIN DE JONGH *is the author of two books on SketchUp, the latest being* Google SketchUp for Game Design: Levels and Props. *He works as an acquisition editor on game development and 3D art books for Packt Publishing.*

BLIZZARD®
ENTERTAINMENT
WORLD OF WARCRAFT®
DIABLO®
STARCRAFT®

BY IRA FAY

MICROSOFT EXCEL FOR GAME DESIGNERS

USING EXCEL FOR DATA ANALYSIS

When people ask me what tools I use as a game designer, they're usually surprised when I tell them how much I use Microsoft Excel (or more recently, Google Docs). Excel can be a powerful data analysis tool for any level of game designer, and it takes some practice to master. Feed playtest results into your spreadsheet and suddenly you have all of the information you could ever want about your game. But it's important to remember that Excel can't work on its own: We need to analyze data thoughtfully. It's also worth remembering that Excel can help with text-based games. Here are some Excel tips and tricks for aspiring game designers with real-life examples taken from games that I've designed.

MICROSOFT VS. GOOGLE

» First, let's talk about Microsoft Excel vs. Google Docs spreadsheets. While these two options are generally interchangeable, there are some key differences. Google Docs is free and cross-platform (it runs in your web browser), and it's great if you want multiple people to edit documents simultaneously. Unfortunately, you need to be connected to the Internet to create and save your documents, and it's not easy to track changes. Excel documents can be easily integrated into your version-control system, can be saved and edited without an internet connection, and include advanced features such as countifs() and sumifs(). But Excel isn't free, and multiple people can't work simultaneously on the same document.

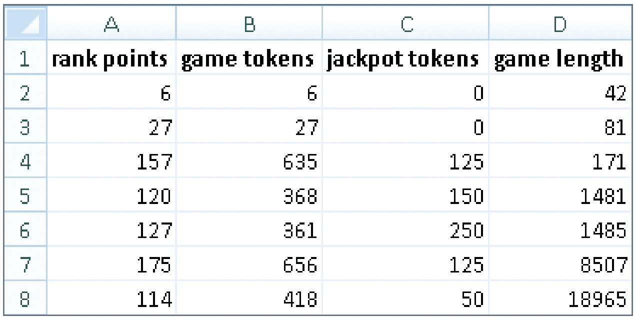
In most cases, you can use either one, though it's useful to be aware of the differences so you can choose the tool that's best suited to your needs.

I won't spend time covering the basics of using spreadsheets, which you can learn in a few minutes on your own. I'll let you know how to use these functions: sum(), average(), max(), min(), and len(). Also, you should know how to sort your data ascending and descending and employ freeze rows and columns so you can always see row and column headers while scrolling your document. If any of that feels unfamiliar to you, spend a few minutes playing around or searching the Internet for answers.

SETTING UP YOUR DATA

» When starting a game project, I make sure to consider what data I'll need to gather. Plan to gather the right data early on and you'll ensure you have the right tools to balance the game later. When I was designing a casual bingo/lottery scratcher card game called Lottso Express (http://pogo.com/games/lottsoexpress), I worked with the game's engineer to output certain metrics into an Excel spreadsheet that was generated every day, filled with data from our beta testers. If you want to play along, you can find the raw Excel files at http://gdmag.com/resources/code.php.

For Lottso Express, I started by looking at the rank points (experience points) a player earned during a game, the game tokens (in-game currency) the player earned, and the jackpot tokens the player earned from a slot machine-like bonus at the end of some games. All these metrics were measured for a single game, which was recorded in the game length column—make sure to note the units! Are we measuring in seconds, minutes, or something else? I asked the engineer, and game length was being measured in seconds, so the entry in D8 (see **Figure 1**) is actually a single game that took more than five hours (18,965 seconds / 60 seconds in a minute / 60 minutes in an hour =~5.27 hours). Since this is a casual game designed to last 1 minute to 15 minutes, I knew something was wrong. I asked the engineer again, and we were actually measuring time from game start to game over. If players pause the game for a long time, we get very long game lengths, which throw off the metrics we're trying to balance.



	A	B	C	D
1	rank points	game tokens	jackpot tokens	game length
2	6	6	0	42
3	27	27	0	81
4	157	635	125	171
5	120	368	150	1481
6	127	361	250	1485
7	175	656	125	8507
8	114	418	50	18965

FIGURE 1: Game metrics from a few sessions of Lottso Express

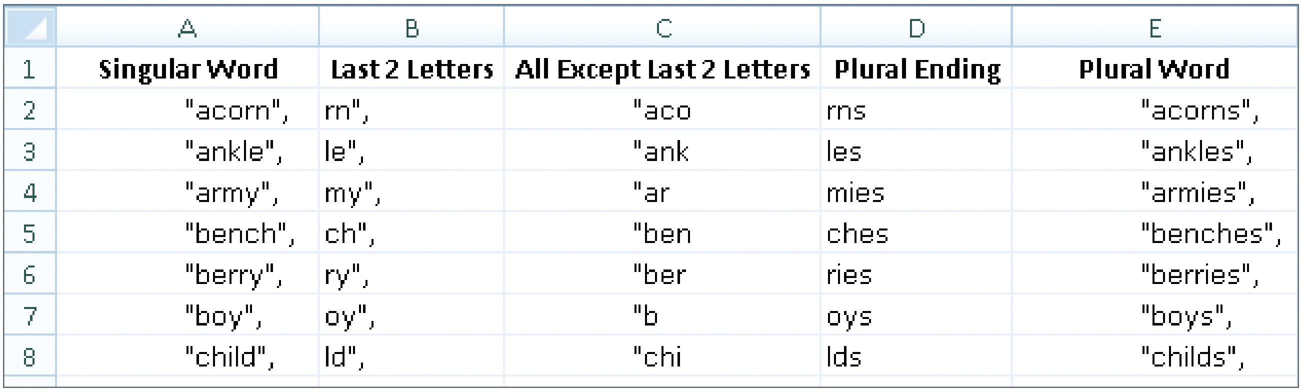
This brings up our first important Excel lesson: Don't blindly trust the data. Data is important, but you must use your powers of analysis and reason to determine the correct course of action. Also, you'll want to make sure you have good communication with the engineering team, since they're the people feeding you the data you need to do your job.

ANALYZING YOUR DATA

» Now that we have some statistics that show how people are playing our game, we can start to figure out what we should fine-tune. The tuning goal for this game was 3,000 tokens per hour, with 2,500 coming from the game itself and 500 coming from the jackpot slot machine. The following formulas get us started:

```
TOTAL GAME TOKENS EARNED (IN B9):        =SUM(B2:B8)
TOTAL JACKPOT TOKENS EARNED (IN C9):     =SUM(C2:C8)
TOTAL GAME LENGTH (IN D9):               =SUM(D2:D8)
```

Instead of typing each formula by hand, only create the formula for game tokens (in B9, the cell below the current data). Then, select B9, C9, and



	A	B	C	D	E
1	Singular Word	Last 2 Letters	All Except Last 2 Letters	Plural Ending	Plural Word
2	"acorn",	rn",	"aco	rns	"acorns",
3	"ankle",	le",	"ank	les	"ankles",
4	"army",	my",	"ar	mies	"armies",
5	"bench",	ch",	"ben	ches	"benches",
6	"berry",	ry",	"ber	ries	"berries",
7	"boy",	oy",	"b	oys	"boys",
8	"child",	ld",	"chi	lds	"childs",

FIGURE 2: You can use Excel to generate a list of plural words, which is useful for word games.

D9, and use the Fill Right command (Control + R). You'll save yourself countless hours of typing if you learn and properly employ the Fill Right and Fill Down commands.

Now we can use the initial sums to calculate tokens per hour:

```
GAME TOKENS PER HOUR:        =B9 / (D9 / 3600)    RESULT: 290
JACKPOT TOKENS PER HOUR:     =C9 / (D9 / 3600)    RESULT: 82
```

It's important to remember the units of your data, which is why we must divide D9 by 3,600, since there are 3,600 seconds in an hour. Given those results, the game is paying out less than 400 tokens per hour, which is way off from our goal of 3,000 per hour. Of course, that analysis includes some really long games, which are skewing the results. Therefore, we can look more closely and exclude abnormally long games. To do so, I recommend the function sumif(), as follows:

```
GAME TOKENS EARNED IN GAMES <30 MINUTES:=SUMIF(D2:D8, "<1800",
B2:B8)
JACKPOT TOKENS EARNED IN GAMES <30 MINUTES:=SUMIF(D2:D8,
"<1800", C2:C8)
TOTAL GAME LENGTH IN GAMES <30 MINUTES:=SUMIF(D2:D8, "<1800")
```

In this case, if you tried the Fill Right shortcut, Excel would properly change "B2:B8" to "C2:C8", but "D2:D8" would also change to "E2:E8", which would mess up the whole formula. Therefore, to tell Excel to keep the column "D" fixed even when using Fill Right, we add a dollar sign ($) in front of the D, like this:

```
GAME TOKENS EARNED IN GAMES <30 MINUTES:=SUMIF($D2:$D8, "<1800",
B2:B8)
```

With that formula, we could Fill Right and the formula would remain correct. Since this example only has a few columns, Fill Right doesn't matter too much since we could type three formulas by hand. When you're working with many columns of data, Fill Right becomes vital to avoid typing out tens or hundreds of slightly different formulas, when Excel could do it all for you with Control + R.

Now that we've eliminated the very long games, let's look at the results of the formulas to get our modified rate-of-tokens-per-hour: Game tokens come out to 1,543 per hour, and jackpot tokens come out to 580. We're still not up to 3,000 per hour, but we got a lot closer by intelligently updating our model of player behavior. To properly balance a game, you need accurate data and data analysis. One other thing to note with this example is that the sample size is very small and only contains seven entries. In a normal beta-testing situation, we would have hundreds or thousands of data points per day.

STRING MANIPULATION

» Although we tend to think of Excel as a tool for numeric data, I've also used Excel to manage text in many games. Recently, I was working on a vocabulary game where I had a dictionary of only singular words, and I needed plural words, too, so I created a table like the one in **Figure 2**.

The data in Column A is taken directly from the dictionary file used in the game. The function in B2 is simply =right(A2, 4). Then I selected B2 through B9 and used the Fill Down command (Control + D). C2 contains =left(A2, len(A2)–4). The "–4" is in the formula because I want to split the word apart, leaving the last two letters (and quote and comma) separate.

Unfortunately, the rules for plural words in English are tricky. First, I started with something simple, like =LEFT(B2, 2)&"s", which simply appends an "s" to the end of the word. Then I realized I should detect words that end in "y," so I used an IF() statement, like this:

```
=IF(MID(B2,2,1)="Y", LEFT(B2,1)&"IES", LEFT(B2,2)&"S")
```

After that, I realized I should actually append an "s" instead of "ies" if the letter preceding the "y" is a vowel. Therefore, I updated the formula to look like:

```
=IF(MID(B2,2,1)="Y",
  IF( OR(LEFT(B2,1)="O", LEFT(B2,1)="A", LEFT(B2,1)="U",
      LEFT(B2,1)="E", LEFT(B2,1)="I"),
    LEFT(B2,2)&"S",
    LEFT(B2,1)&"IES"),
  LEFT(B2,2)&"S")
```

As you can see, the actual algorithm for finding the plural spelling of an English word gets complicated quite quickly. Nonetheless, if you build your formula one step at a time, you'll be able to create powerful results. Start small, and work your way up in complexity. At some point, you may reach diminishing returns for rare corner cases, and it's okay to handle those by hand. You can see the word "child" turns into "childs," even though the correct plural form is "children." I simply dealt with that case by hand.

These are only a few of the ways you can use Excel as a game design and data mining tool. You can also create probability and game simulations, advanced text processing, complex asset tracking combined with easy integration into game scripts, and much more. For anyone who wants to practice their Excel skills, I created several Excel puzzles that are available here: http://irafay.com/Excel. Enjoy!

IRA FAY *is an assistant professor of game design at Quinnipiac University and is the founder of Fay Games, a development studio primarily focused on educational games. He was previously a senior game designer at Electronic Arts (Pogo.com).*

BY MARE SHEPPARD

YOU CAN MAKE GAMES!

NEVER GIVE UP. TRUST YOUR INSTINCTS.

You probably have a passion for playing video games. Why else would you be reading this? Games take you on a journey through their world, immersing you in their language and rules as you explore. It's almost magical how games can inspire you to learn their mechanics and behaviors and to develop your skills, growing as a player along the way. And if you enjoy playing games, you may have started thinking about how to make them. Contemplating how games are composed and designed, and how people built their worlds, characters, and interfaces, can help you understand how games work, what makes them fun, and what you might do differently if you had the chance.

If the above resonates with you, I have good news: You can make games. I can say that with confidence, even though I have no idea who you are. You could be a 67-year-old grandmother, an 18-year-old guy with an interest in music, a 23-year-old woman just finishing her college education, a 45-year-old firefighter—whoever you are, whatever your interests, you can make games.

WHY MAKE GAMES?

» Make games because you love them. For me, I make games as part of a two-person indie developer called Metanet Software because there are games we really want to play that don't yet exist beyond our imaginations. So we create them ourselves! We like working as part of a small team because we needed to have complete creative control in order to bring those game ideas to life, and because we each also happen to have a variety of skills, such as game design, art, programming, and so on.

But you don't have to make games in small teams. If you would rather specialize in one particular area, such as 3D modeling or database programming, you could be an integral part of a larger development team—maybe at a big-budget triple-A team of hundreds or at a leaner indie studio with a couple dozen employees. If you're a designer, or if you want to first sketch out some ideas until you have a feel for what you can do and what you most enjoy, you might start out on your own. Making games means different things to different people: It can mean single-handedly constructing the entire game from start to finish, or it can mean contributing just one small, carefully crafted part. It's all about experimenting until you discover what satisfies you the most.

PHOTOS BY VINCENT DIAMANTE

HOW CAN YOU BE SURE I CAN MAKE GAMES?

» Because anyone can, if they try. Your age, race, sex, religion, social status, height—none of this matters. All you really need to get started is a passion for games. Even having no prior experience in games can be a good thing; if you're new, you won't necessarily be operating within the established "rules," which allows you to explore unconventional solutions with creativity and innovation. No one is born a perfect game developer, fully formed. They get better by practicing their skills over time, and all of the skills you need to make games are naturally attainable by every single human on this planet. If you're concerned about what you bring to the table, especially in such a saturated industry, consider this: What distinguishes you from everyone else is the exclusive mélange of your intrinsic talents, your opinions, and your skills, cultivated over the course of your life and shaped by your unique circumstances. Everyone benefits from having diverse opinions and people in a field, and you can contribute to that diversity with input no one else can give. Whatever your circumstances, whatever your stats, making games is something you can do.

Lobby Bar
MEET US AT BOOTH 639
By the time you finish this sentence, we'll have connected 42 more users to apps.
W3i POCKET GAMER PARTY!
Windows

WHAT IF I'M NOT A GEEK?

» If you're worried that you don't fit in or aren't right for this industry, let me assure you: You definitely do belong here. The clunky old "geek" stereotype you may feel you aren't part of is actually not as representative of game developers (or gamers) as some might say, and you certainly don't have to be a geek to work in this field. Looking around at game-related conferences and events, it's obvious that game developers are far more varied than the stereotype suggests. Unfortunately, this isn't well known outside of the industry. Eventually, our culture will catch up, and by ignoring stereotypes and working where you want to, not where it's implied that you fit better, you will help to change them. The things that matter are your attitude and capability, not some outdated definition of who you "should" be.

WHAT IF I CAN'T AFFORD TO GO TO SCHOOL?

» It's hugely beneficial to have educational resources and tutorials available, but experience—playing, learning, experimenting, and spending time working—is even better. School is useful, but you already have everything you need to start learning on your own, and for free or cheap, on the internet. The internet is an incredible resource with the answers to pretty much any question you could ever have. Also, the best way to learn how games work is by experimenting on small projects by yourself or with others.

If you're worried that your skills aren't at the level they need to be to work in the game industry, there's good news: It's never too late to learn, and the great thing about skills is that they improve with practice. Plus, there are so many tools out there to help you (many of which are explained in this very magazine). In game development, your potential isn't limited by what you know or what you can do right now but by how willing you are to learn and try new things and how well you persevere through difficulty.

WHAT IF I CAN'T CODE?

» You can always learn. Knowing how to code is a useful skill, especially on a small team where you will have to work in a variety of roles. You could also explore modern game-making tools, such as Stencyl, GameMaker, and even Flash, which remove a lot of the difficulty inherent in programming but still let you create interesting and fun games, like CANABALT, SPELUNKY, and TUNING.

If you're talented in another way, then you can also team up with others and collaborate. Collaborations often lead to many innovative, exciting projects. PC and iOS hit SUPERBROTHERS: SWORD & SWORCERY is the result of Jim

Guthrie (a musician) and Craig Adams (an artist) teaming up with Capybara (a talented game developer). Each contributed their own perspective, creating a compelling world and an immersive playing experience. It's like CHRONO TRIGGER, where different party members can team up to execute different special moves; instead of attacking enemies, you're making a game with different people and skills that you can combine in remarkable ways.

WHAT IF MY GAME SUCKS?

» Making games is not an easy career path—whatever you do and however you do it, you are going to fail repeatedly. The trick is anticipate it and not let it discourage you. The development of SOUND SHAPES began with many prototypes that Jon Mak showed at a Hand Eye Society social in Toronto. Each prototype was a distinct, intriguing grain of an idea that could grow into a game in its own right, and the only qualities they all shared were a unique mix of action and music-making and a failure to capture the magic Jon was looking for. He could've given up and succumbed to the idea that he would never get it right, but he continued to search for a game that matched his vision by learning from the mistakes of each failed prototype. Failure makes it very difficult to stay motivated and persevere, but if you can recognize when a project just isn't working and shift your focus to developing your skills and experimenting, odds are good that you'll eventually create something wonderful.

WHAT IF NO ONE LIKES MY GAME?

» Rarely does anyone get anything in games right on the first try. The important part is to keep honing and refining your ideas and execution. Surrounding yourself with other people who are interested in making games definitely helps, since you can share knowledge with experienced peers and provide feedback on each other's games to iterate and improve your games. Plus, you'll need a strong support network that'll champion your efforts and help you through those rough patches.

Maybe your game isn't as great as you thought, and you're disappointed. Use that disappointment to inspire yourself to iterate and improve upon your game idea. If you're not getting players or press interested in your game, it probably needs more work. Get feedback from players, listen to it, and keep building and distilling until everyone can see what you find so compelling. To survive the gauntlet of emotional trauma that is game development (or really, any creative endeavor), you need determination, flexibility, humility, curiosity, and most of all, love for what you're doing. If you don't like your game, you probably won't be motivated enough to finish it.

WHAT IF I NEVER MAKE A GAME THAT SELLS A MILLION COPIES?

» Selling millions copies of a game is an exceedingly rare event. Moreover, huge sales numbers are only one way to measure success, and they aren't always the best or most accurate way. BRAID, arguably one of the most successful indie games, has sold "only" 450,000 copies on XBLA, but to call it a failure because it didn't sell a million-plus copies does it a disservice: BRAID's real worth comes from its innovative and uniquely compelling brain-bending gameplay mechanics and lasting impression on its players. For most games, the number of sales is just a number; the best measure of success is your own personal standard. If success for you equates to "spending my life learning and doing what I love," it's much easier to attain than "being rich and famous" (and is likely to be much more gratifying, too).

WHAT IF I START TO HATE WHAT I'M WORKING ON?

» You need to be able to give your project, and yourself, the freedom to be something else. Flexibility in general is one of the most important skills you can have. You may become aware that something you're working on is more or less fun than expected but feel that changing it is daunting since so much work/time/money has been spent on it. We initially wanted the game N to be a slow-paced stealth game, until we discovered how fun it was to run and jump around the level at top speed—a huge divergence from our plans. Having the courage to change gears and maximize or minimize new discoveries can lead to some wonderfully unconventional outcomes and a better game. The history of games tells many such stories, everywhere from the triple-A to indie games, from SPACE INVADERS to SOUND SHAPES to GRAND THEFT AUTO. That concurrence illuminates the fundamental goal of game developers: We are trying to make great games.

That's what I believe should ultimately unite each person working in the game industry: We care about making games, and we're genuinely concerned with their content. Good games are fun, exciting, engaging, and intriguing, and are often the result of a combination of careful contemplation, planning, and happy accidents—but they are incredibly hard to create. When you make games because you believe in them and want to play more of them, and because you enjoy exploring creative ideas, then regardless of critical or commercial results, this can be the most enjoyable and satisfying work you could ever do. The payoff is immense: You get to watch your ideas come to life in surprising and dynamic ways, and you get to play an awesome game when you're done.

If you take away only one thing from this article, make it this: Your focus should be on making great games, or making games great. Make something that you care about and believe in. This industry really needs more creative and passionate individuals who are ambitious and enthusiastic and love what they do—those are the most important qualifications you could have. If that's you, then welcome! I can't wait to see you get started, and am very much looking forward to playing your games. It will be a difficult journey, but you can do it.

MARE SHEPPARD *is one half of Metanet Software Inc., an indie game developer based in Toronto, and is a founding member of the Hand Eye Society, a Toronto-based coalition relating to video games. Mare fills her days doing half of everything at Metanet, and in her spare time enjoys creating Metanet's merchandise, playing video games, chilling with Toronto indies and encouraging more people to make and play games.*

further inspiration

http://livelyivy.com/gravityghost-com/5-alternatives-to-a-game-design-doc

www.wildbunny.co.uk/blog/2012/01/06/how-to-make-games/

www.flight404.com/blog/?p=49

Autodesk®

BY PATRICK MILLER

GAME JAMS

GAME DEVELOPMENT SPEED DATING

Ever try to make a game over a single weekend? Game jams are a great opportunity to make quick-and-dirty game prototypes, play with new and unconventional design themes, and network with fellow enthusiastic developers. And since you can participate in many of these jams remotely, you don't have to be limited by your local developer scene, either. In a game jam, a bunch of people, who often have never met each other, and who sometimes have never even made games before, get together to work on game projects. Most game jams have a loose theme for the developers to work on, and a time limit, but that's all the structure you get! This is a great way to meet developers, gain some skills, and learn about team dynamics without the threat of losing money or getting a bad grade. It's a great way to experiment, and to learn by doing.

Global Game Jam

www.globalgamejam.org

» The Independent Game Developers Association's Global Game Jam is the largest game jam in the world (in fact, it's getting added to a special edition of the Guinness Book of World Records). It takes place every year in late January, and last year's jam saw thousands of developers in 242 different locations across the world make over 1,500 games—just over the course of a weekend! Don't be intimidated, though; everyone is welcome, not just professionals, and you're encouraged to come and find a team at the venue rather than bring your own. If there is a school near you with a game-development program, odds are pretty good they're participating in the Global Game Jam, so be sure to search the official site for local jams.

TOJAM

www.tojam.ca

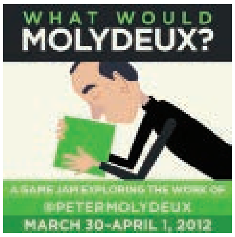
» The Toronto Game Jam (TOJam) aims to foster collaboration and experimentation among developers by hosting a three-day programming marathon for hobbyists and professionals. Attendees bring their own computers, tools, and sleeping bags to collaborate with their peers and make the best game they can—from scratch—over the course of a weekend.

The event is not a competition but rather an opportunity for amateur and professional developers to test their skills in a fun environment while surrounded by like-minded peers. Since the first TOJam in 2006, the event has produced well over a hundred complete games. The next TOJam will take place in May 2013.

Experimental Gameplay Project

www.experimentalgameplay.com

» The Experimental Gameplay Project is a long-running game jam headed up by the indie devs behind World of Goo, Crayon Physics Deluxe, and many other games. It's a virtual jam, so all you have to do is submit a game by posting a link on the EGP's web site that follows three rules: Your game must be made by one person and in less than seven days, and it must be built around the EGP theme of the month. Recent themes include "Rotation," "101 Things," and "Economy."



What Would Molydeux?

www.whatwouldmolydeux.com

» What Would Molydeux? This irreverent game jam (called "Molyjam" for short) owes its genesis to Twitter account "Peter Molydeux" (@molydeux), a parody account that gently pokes fun at notable game designer Peter Molyneux's (Fable, Populous) high-flying concepts for emotionally evocative video games. A few developers and journalists (including *Game Developer* editor-in-chief Brandon Sheffield) took the idea and managed to organize a worldwide game jam with over 900 developers in almost 35 cities in under a month. (For more on Molyjam's origins, read the postmortem in the May 2012 issue of *Game Developer*.)

Would-be developers of all experience levels are encouraged to participate, either in person at a Molyjam site or remotely, and the only required theme is that your game must be inspired by a specific @molydeux tweet, such as "I wish there was a game where you played a two-year-old, alone in a creepy house during a terrible storm." or "What if to create a save point you had to give birth to it? What if the save point followed you but could easily get upset and run away?" The first Molyjam happened in 2012, and the second hasn't been scheduled yet but will probably be in spring 2013.

Ludum Dare

www.ludumdare.com

» The Ludum Dare competition/jam is another weekend-long online game-development event that has been running for about 10 years. There are two different ways to participate in Ludum Dare. You can elect to submit your game to the competition, which means you have to make the game by yourself, over 48 hours, and all game content (with certain exemptions) needs to be created within those 48 hours. Or you can participate in the jam, which means you can take up to 72 hours to make a game, and you can work with a team if you like. Games are judged on their innovation, fun, visual and audio quality, how well they explore the theme, and other criteria. Ludum Dare is currently approaching its 24th session, scheduled for August at the time of this writing.

Klik of the Month Klub

www.glorioustrainwrecks.com

» The Klik of the Month Klub is a monthly online jam hosted by Glorious Trainwrecks on the third Saturday of every month at 4 p.m. Pacific time. It's focused on the venerable Windows 3.1–era game development tool Klik & Play (though you can use whatever development platform you like), and unlike most jams, your time limit is only two hours. And yes, in case you were wondering: "Glorious Trainwrecks" refers to all the wonderful, terrible games people create for the Klik of the Month Klub.

Make Your Own Jam!

» If none of these appeal to you, form your own jam in your local development community or university. The Global Game Jam site has hints about how to set up a jam, and there are many postmortems of the Molyjam floating around the web. Game jams show that truly anyone can make games!

GAME CAREER GUIDE.COM

BROUGHT TO YOU BY GAMASUTRA

SCHOOL DIRECTORY

SCHOOL NAME	ADDRESS
3D Training Academy	11410 NE 124th St. #406
A.I.E. - The Academy of Interactive Entertainment	305 Harrison St Suite 405
A.I.E. - The Academy of Interactive Entertainment	537 Cajundome Blvd. Suite 209
Academy of Art University	79 New Montgomery St.
Academy of Interactive Entertainment, Melbourne	Level 8, 14 Queens Rd.
Academy of Interactive Entertainment, Sydney	Level 2, Wentworth Park Grandstand
Academy of Interactive Entertainment, Watson	Canberra Technology Park
Acadia University	Jodrey School of Computer Science 27 University Ave.
Algoma University	1520 Queen St. E.

There are tons of options out there in terms of viable game schools, and this list is just the starting point to get you acquainted with the schools near you (or far from you, if that's what you prefer!). No list could possible tell you all you need to know, so you should use this directory to find schools you're interested in, and look them up with our comprehensive directory at GameCareerGuide.com/Schools. Over there you'll find all sorts of information, including:

- In-depth school profiles
- Listings of specific game development programs and degrees offered
- Tuition and financial aid information
- Student-to-faculty ratio
- Online class offerings
- Actual playable student games

While you're there, you can use our Digital Counselor app (GameCareerGuide.com/digital_counselor/) to search through the full database for schools by location, degree offerings, online study options, and more search criteria to find the game development school that's right for you!

CITY	STATE	ZIP	COUNTRY	PHONE	CONTACT EMAIL
Kirkland	WA	98034	U.S.	877-540-3DTA (3382)	info@3dtrainingacademy.com
Seattle	WA	98109	U.S.	206-428-6350	madeleinee@aie.edu.au
Lafayette	LA	70506	U.S.	337-735-5483	madeleinee@aie.edu.au
San Francisco	CA	94105	U.S.	800-544-2787	info@academyart.edu
Melbourne	VIC	3004	AU	61 3 9820 8201	melb@aie.edu.au
Ultimo	NSW	2010	AU	612 8514 8800	sydney@aie.edu.au
Watson	ACT	2602	AU	61 2 6162 5131	canberra@aie.edu.au
Wolfville	Nova Scotia	B4P 2R6	CA	902-585-1331	cs@acadiau.ca
Sault Ste. Marie	Ontario	P6A 2G4	CA	888-ALGOMAU	msc@algomau.ca

SCHOOL NAME	ADDRESS
Angelo State University	ASU Station #10909
Animation Mentor	1400 65th St.
Asian Institute of Gaming and Animation - AIGA	No 41/1, H.A.L. Airport Rd. Murugeshpalya
Austin Community College	Visual Communication Department 11928 Stonehollow Dr.
Bloomfield College	467 Franklin St.
British Columbia Institute of Technology, School of Computing	3700 Willingdon Ave.
Broadview Entrainment Arts University	240 East Morris Ave.
Centennial College	951 Carlaw Ave.
Centre for Arts and Technology - Fredericton Campus	130 Carleton St.
Centre for Arts and Technology - Halifax Campus	1577 Barrington St
Centre for Arts and Technology - Kelowna Campus	Landmark Technology Centre III 1632 Dickson Ave. Ste. 100
Centre for Digital Media - Masters of Digital Media Program	577 Great Northern Way
CENTRE NAD - National Animation and Design Centre	405 Ogilvy Ave.
Champlain College	163 So Willard St.
Cogswell College	1175 Bordeaux Dr.

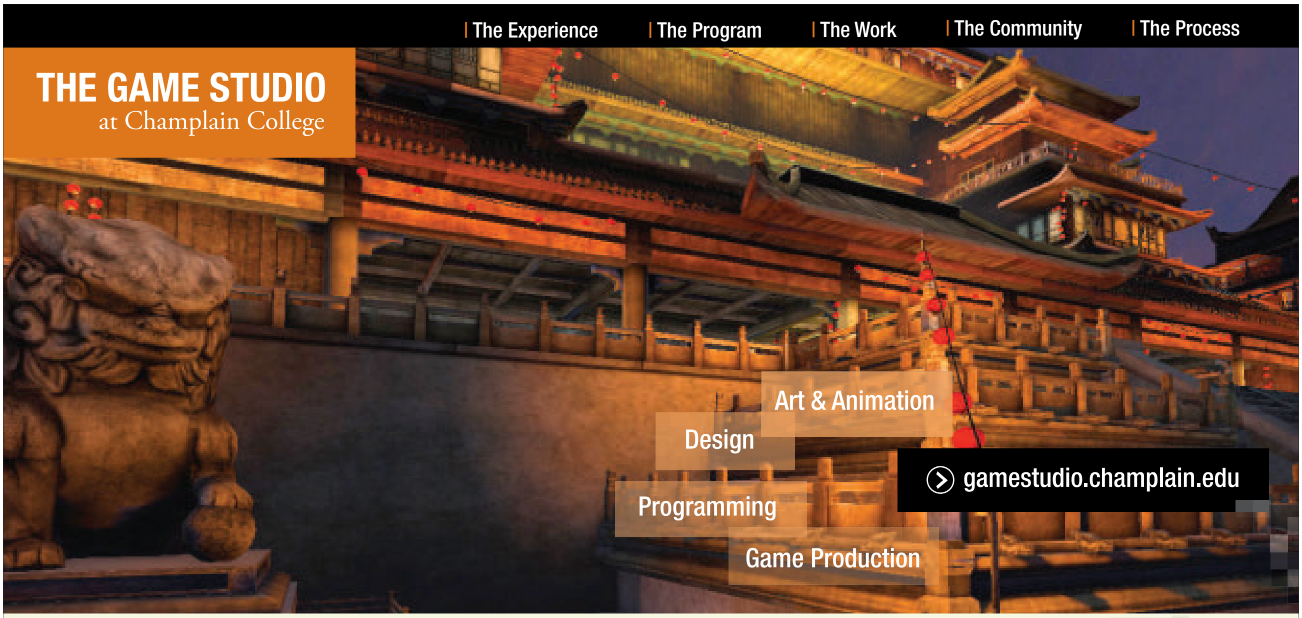
CITY	STATE	ZIP	COUNTRY	PHONE	CONTACT EMAIL
San Angelo	TX	76909	U.S.	325-942-2101	troden@angelo.edu
Emeryville	CA	94608	U.S.	877-326-4628	admissions@animationmentor.com
Bangalore	KA	560 017	IN	91 80 40480900	info@aiga.in
Austin	TX	78758	U.S.	512-223-4802 or 4830	cblanken@austincc.edu
Bloomfield	NJ	07003	U.S.	973-748-9000	admission@bloomfield.edu
Burnaby	BC	V5G 3H2	CA	604-432-8644 or 604-453-4021	cstbtech@bcit.ca
Salt Lake City	UT	84115	U.S.	801-300-4300 or 877-480-3335	N/A
Toronto	ON	M4K 3M2	CA	416-289-5000 x8637	knoble@centennialcollege.ca
Fredericton	NB	E3B 3T4	CA	506-460-1280	inquire@digitalartschool.com
Halifax	NS	B3J 1Z7	CA	866-429-1847 or 902-429-1847	inquire@digitalartschool.com
Kelowna	BC	V1Y 7T2	CA	866-860-2787 or 250-860-ARTS (2787)	inquire@digitalartschool.com
Vancouver	BC	V5T 1E1	CA	778-370-1010 or 1001	yasmeen_awadh@gnwc.ca
Montreal	QC	H3N 1M3	CA	514-288-3447	info@nadcentre.com
Burlington	VT	05402-0670	U.S.	802-860-2727 or 800-570-5858	admissions@champlain.edu
Sunnyvale	CA	94089	U.S.	800-264-7955 or 408-498-5100	admissions@cogswell.edu

SCHOOL NAME	ADDRESS
Columbia College Chicago	600 South Michigan Ave.
Cornell University	4130 Upson Hall
Creajeux	1105 Ave Mendes France
Dakota State University	820 North Washington Ave.
DAVE School (The Digital Animation & Visual Effects School)	2500 Universal Studios Plaza Soundstage 25
DePaul University	243 South Wabash Ave.

CITY	STATE	ZIP	COUNTRY	PHONE	CONTACT EMAIL
Chicago	IL	60605	U.S.	312-369-7750	info@iam.colum.edu
Ithaca	NY	14853	U.S.	607-255-9196 or 607-255-0980	gdiac@cornell.edu
Nimes	LR	30 000	FR	04.66.35.56.20	creajeux@creajeux.fr
Madison	SD	57042	U.S.	605-256-5819	skg@dsu.edu
Orlando	FL	32819	U.S.	855-328-3839 or 407-224-3283	admissions@daveschool.com
Chicago	IL	60604-2287	U.S.	312-362-8381	admission@cdm.depaul.edu

SCHOOL NAME	ADDRESS
DeVry - Dallas	4800 Regent Blvd.
DeVry University	3005 Highland Pkwy
DigiPen Institute of Technology	9331 Willows Rd. NE
DigiPen Institute of Technology Europe—Bilbao	Bilbao, Bizkaia 48008
DigiPen Institute of Technology Singapore	Pixel Building #01-01
Edinboro University of Pennsylvania	219 Meadville St.
ENDI (École nationale en divertissement interactif)	Édifice de la Fabrique, 295, boul. Charest Est

CITY	STATE	ZIP	COUNTRY	PHONE	CONTACT EMAIL
Irving	TX	75063	U.S.	800-633-3879 or 972-929-6777	inquiries@devry.com
Downers Grove	IL	60515	U.S.	866-338-7934	crubin@devry.edu
Redmond	WA	98052	U.S.	866-478-5236 or 425-558-0299	admissions@digipen.edu
Bilbao	Bizkaia	48008	ES	(94) 4706400 or (94) 4706500	admissions.es@digipen.es
Singapore	—	138649	SG	65 - 6577 1900	singapore@digipen.edu
Edinboro	PA	16444	U.S.	814-732-2760 or 1171	http://www.edinboro.edu/home/contact_form.dot
Québec	QC	G1K 3G8	CA	418-656-7033	info@endi.qc.ca



SCHOOL NAME	ADDRESS
ENJMIN (National School of Video Game and Interactive Media)	ENJMIN 121 rue de Bordeaux
Entertainment Technology Center, Carnegie Mellon University	700 Pittsburgh Technology Center CMU-ETC, Room 5305
Escuela Da Vinci	Av. Corrientes 2037
Expression College for Digital Arts	6601 Shellmound St.
Friends of Design	186 Bree St.
Full Sail University	3300 University Blvd.
FuturePoly	12708 Northup Way Suite 202
Games Academy	Rungestr. 20, 10179
George Mason University, ACS	Department of Computer Science 4400 University Dr. MSN 4A5
George Mason University, BFA	4400 University Dr.
Glendale Community College	6000 West Olive Ave.
Goldsmiths, University of London	Goldsmiths College, University of London Department of Computing, New Cross
Great Eastern Technology	282 Main St.
Guildhall at SMU	5232 Tennyson Parkway Bldg #2
Herzing College	1616 Rene Levesque

CITY	STATE	ZIP	COUNTRY	PHONE	CONTACT EMAIL
Angoulême	PC	16000	FR	33-(0)545386568	contact@enjmin.fr
Pittsburgh	PA	15219	U.S.	412-268-5791	RL3J@andrew.cmu.edu
Buenos Aires	BS	1045	AR	54-11 5032-0076 or 5032-0055	info@escueladavinci.net
Emeryville	CA	94608	U.S.	877-833-8800 or 510-654-2934	admissions@expression.edu
Cape Town	WC	8001	ZA	27 21 402 0303	info@friendsofdesign.net
Winter Park	FL	32792	U.S.	800-226-7625 or 407-679-0100	admissions@fullsail.com
Bellevue	WA	98005	U.S.	206-331-1573	info@futurepoly.com
Berlin	—	10179	DE	Berlin: +49 30 29779120 or Frankfurt: +49 69 42 69 64 60	info@games-academy.de
Fairfax	VA	22030	U.S.	703-993-1530	csadmin@cs.gmu.edu
Fairfax	VA	22030	U.S.	703-993-5734	gmugame@gmu.edu
Glendale	AZ	85302	U.S.	623-845-3000	michelle.blomberg@gcmail.glendale.edu
London	—	SE14 6NW	GB	44 (0)2079197850	ffl@gold.ac.uk
Salem	NH	3079	U.S.	800-875-0025 or 781-937-0300	hjelev@greateasterntech.com
Plano	TX	75024	U.S.	214-768-9950	rarchamb@smu.edu
Montreal	QC	H3H 1P8	CA	514-935-7494	info@mtl.herzing.edu

SCHOOL DIRECTORY

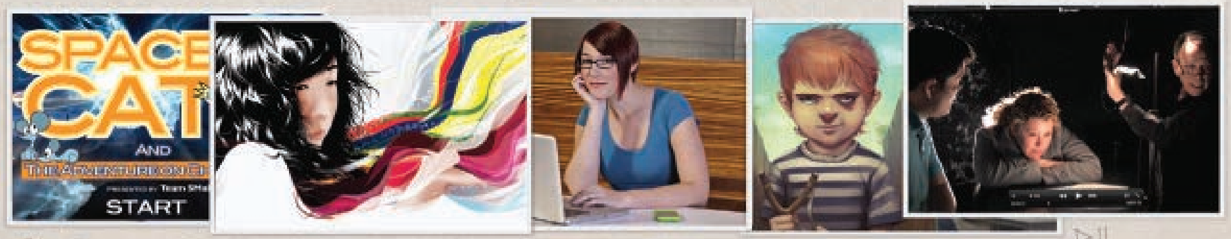
GAME CAREER GUIDE.COM

SCHOOL NAME	ADDRESS
Hochschule Mittweida (FH) University Of Applied Sciences	Technikumplatz 17
Howest - University College West-Flanders	Graaf Karel de Goedelaan 5
HTW - Hochschule für Technik und Wirtschaft	Campus Wilhelminenhof Wilhelminenhofstraße 75A
ISART Digital	2, rue de la Roquette
Max the Mutt Animation School	952 Queen St. West
Media Design School	92 Albert St.

	CITY	STATE	ZIP	COUNTRY	PHONE	CONTACT EMAIL
	Mittweida	SA	D-09648	DE	49 (0) 3727 58-0	kontakt@hs-mittweida.de
	Kortrijk	West-	8500	BE	32 56 24 12 11	info@digitalartsandentertainment.be
	Berlin	Berlin	12459	DE	(030) 5019 - 0	N/A
	Paris	Île-de-France	75011	FR	0033 148 075 848	informations@isartdigital.com
	Toronto	Ontario	M6J 1G8	CA	877-486-MUTT (6877) 416-703-6877	admissions@maxthemutt.com
	Auckland	NZ	1141	NZ	64 9 303 0402	admissions@mediadesignschool.com

67% of American Households Play Video Games
The average American spends 97,864 hours at work over the course of a lifetime. Shouldn't you have a job you love?
Ringling College of Art + Design
2700 North Tamiami Trail | Sarasota, Florida 34234-5895
941.351.5100 | www.ringling.edu
"Among Skies"
Brian Harries, Daniel Martinez, Noah Ortega, Clarke Nordhauser, Game Art & Design, class of 2012

SCHOOL NAME	ADDRESS
Middlesex University	The Burroughs Hendon
Montgomery College	51 Mannakee St.
New England Institute of Technology	2500 Post Rd.
NHTV Breda University of Applied Science	Reduitlaan 44
North Karelia University of Applied Sciences	Lansikatu 15
Northeastern University - College of Professional Studies	College of Professional Studies 360 Huntington Ave.



CITY	STATE	ZIP	COUNTRY	PHONE	CONTACT EMAIL
London	—	NW4 4BT	GB	44 (0)20 8411 5555	enquiries@mdx.ac.uk
Rockville	MD	20850	U.S.	240-567-5000	deborah.solomon@montgomerycollege.edu
Warwick	RI	02886-2266	U.S.	401-739-5000	info@neit.edu
Breda	—	4814 DC	NL	31 (0) 76 533 2770	secretary.ade@nhtv.nl
Joensuu	—	FI-80110	FI	358 50 3381126	eero.monkkonen@pkamk.fi
Boston	MA	02115-9959	U.S.	617-373-2400 or 5824	cpsadmissions@neu.edu

SCHOOL DIRECTORY

GAME CAREER GUIDE.COM

SCHOOL NAME	ADDRESS
Northeastern University - Creative Industries (Game Design & Interactive Media)	360 Huntington Ave.
Norwegian School of Information Technology	Schweigaardsg 14
NTI Birmingham	15 Bartholomew Row
Otis College of Art and Design - Digital Media Department	9045 Lincoln Blvd.
Parsons the New School of Design: School of Art, Media and Technology	2 West 13th St. 10th Floor
Pennsylvania State University	332 Information Sciences and Technology Building
Piedmont Community College	P.O. Box 1197/ P.O. Box 1150
QANTM College - Amsterdam	Johan van Hasseltweg 31
Queensland University of Technology - Brisbane City, QLD	Gardens Point Campus 2 George St.
Queensland University of Technology - Kelvin Grove, QLD	Creative Industries Precinct Musk Ave.
Rasmussen College	Multiple Campuses
Rensselaer Polytechnic Institute	110 8th St.
Richland College	12800 Abrams Rd.
Ringling College of Art and Design	2700 North Tamiami Tr.

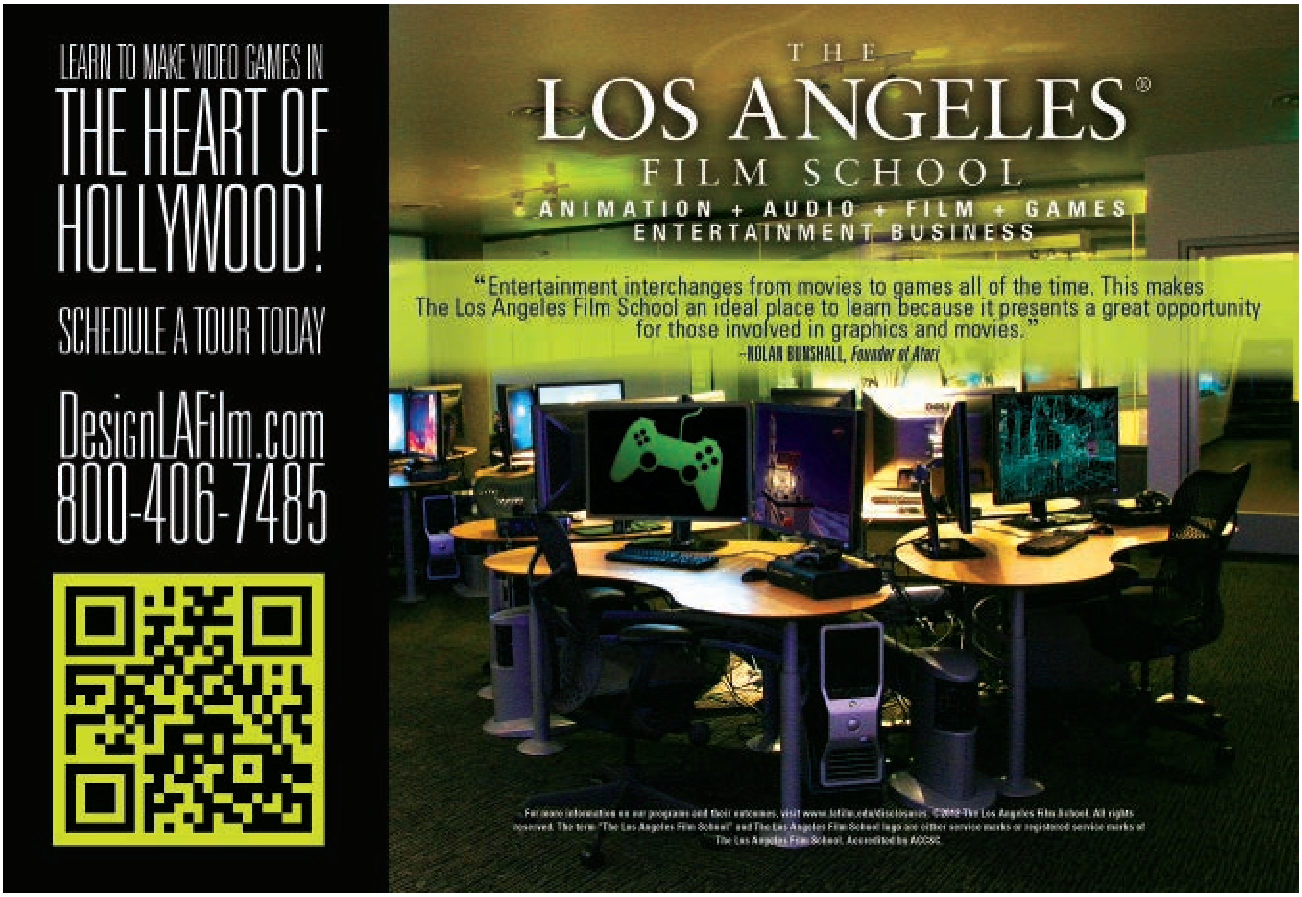
CITY	STATE	ZIP	COUNTRY	PHONE	CONTACT EMAIL
Boston	MA	02115	U.S.	617-373-2000 or 7645	t.masson@neu.edu
Oslo	—	N-0185	NO	4722059999	oslo@nith.no
Birmingham	—	B5 5JU	GB	0121 202 4822	iain@gamercamp.co.uk
Los Angeles	CA	90045	U.S.	3100-665-6800 or 6982	zharris@otis.edu
New York	NY	10011	U.S.	800-252-0852	cdt@parsons.edu
University Park	PA	16802	U.S.	814-865-3528	futurestudents@ist.psu.edu
Roxboro/Yanceyville	NC		U.S.	336-694-5707 or 336-599-1181	hindmap@piedmontcc.edu
Amsterdam	Noord-	1021 KN	NL	31 (0)20 6 228 790	info@qantm.nl
Brisbane	QLD	4000	AU	61 7 3138 2782	sef.enquiry@qut.edu.au
Kelvin Grove	QLD	4059	AU	61 7 3138 8114	creativeindustries@qut.edu.au
Currently Offered Online and at 21 campuses in 5 states.		Multiple Locations	U.S.	888-549-6755	http://www.rasmussen.edu/techdesign/Contact-Us/default.asp
Troy	NY	12180	U.S.	518-276-6575	destem@rpi.edu
Dallas	TX	75243	U.S.	972-761-6830	N/A
Sarasota	Florida	34234-5895	U.S.	800-255-7695 or 941-351-5100	admissions@ringling.edu

SCHOOL NAME	ADDRESS
Rochester Institute of Technology	60 Lomb Memorial Dr.
Savannah College of Art and Design	P.O. Box 2072
School of Visual Arts	209 East 23rd St.
Seattle Central Community College	1702 Harvard
Seneca College - Game Art & Animation	70 The Pond Rd.
Serious Game Design Institute	Santa Barbara City College 721 Cliff Dr.
Sessions College for Professional Design	Main Campus 398 S. Mill Ave. Ste. 300

CITY	STATE	ZIP	COUNTRY	PHONE	CONTACT EMAIL
Rochester	NY	14623-5608	U.S.	585-475-6631	admissions@rit.edu, gradinfo@rit.edu
Savannah	GA	31402-2072	U.S.	800-869-7223 or 912-525-5100	admission@scad.edu
New York	NY	10010-3994	U.S.	800-436-4204 or 212-592-2100	admissions@sva.edu
Seattle	WA	98122-2413	U.S.	206-587-5448 or 206-344-4412	commed@sccd.ctc.edu
Toronto	Ontario	M3J 3M6	CA	416-491-5050 x3850	aac@senecac.on.ca
Santa Barbara	CA	93109	U.S.	805-965-0581	kiggens@sbcc.edu
Tempe	AZ	85281	U.S.	800-258-4115 or 480-212-1704	admissions@sessions.edu

SCHOOL NAME	ADDRESS
Sheridan College Institute of Technology & Advanced Learning	1430 Trafalgar Rd.
Simon Fraser University - School of Interactive Arts and Technology (SIAT)	250 -13450 102 Ave.
Southern New Hampshire University - SNHU	2500 N. River Rd.
Southern Polytechnic State University	1100 S Marietta Pkwy.
Staffordshire University	Faculty of Computing, Engineering and Technology Beconside
Texas State Technical College	3801 Campus Dr.

CITY	STATE	ZIP	COUNTRY	PHONE	CONTACT EMAIL
Oakville	Ontario	L6H 2L1	CA	905-845-9430	infosheridan@sheridaninstitute.ca
Surrey	BC	V3T 0A3	CA	778-782-7444	siat_advising@sfu.ca
Manchester	NH	03106	U.S.	800-668-1249	enroll@snhu.edu
Marietta	GA	30060	U.S.	678-915-4982 or 3571	jpreston@spsu.edu
Stafford	—	ST18 0AD	GB	44 (0)1785 353370	fcet@staffs.ac.uk
Waco	TX	76705	U.S.	800-792-8782 x2360	marcus.balch@tstc.edu



SCHOOL NAME	ADDRESS
The Art Institutes System of Schools	The Art Institutes is a system of more than 45 schools located throughout North America
The Game Assembly	Kalendegatang
The Rydan Workshop	1051 Heritage Blvd.
Training Center Alcance Digital	Blvd. Juan Alonso de Torres no. 2401 col. Panorama
Tribeca Flashpoint Media Arts Academy	28 North Clark St.
triOS College	Campuses in London, Kitchener, Hamilton, Mississauga, and Toronto
Universidade Anhembi Morumbi	Av. Roque Petroni Jr, 630
University of Abertay Dundee	Bell St.
University of Advancing Technology	2625 West Baseline Rd.
University of Baltimore	1420 N. Charles St.
University of California , Santa Cruz	1156 High St.
University of California San Diego - Extension Digital Arts Center	6256 Greenwich Dr.
University of California, Irvine	Donald Bren School of Information and Computer Sciences
University of Central Florida (FIEA)	500 West Livingston St.
University of Colorado, Colorado Springs	1420 Austin Bluffs Pkwy.
University of Denver	2360 S. Gaylord St.

CITY	STATE	ZIP	COUNTRY	PHONE	CONTACT EMAIL
Pittsburgh	PA	15222-2603	U.S.	888-233-6898	http://www.artinstitutes.edu/email-us.aspx
Malmö		205 10	SE	N/A	info@thegameassembly.com
North Vancouver	BC	BC V7J	CA	206-905-9644	info@therydanworkshop.com
León	—	37160	MX	477 2187112/477 2187112	ad@alcancedigital.com
Chicago	IL	60602	U.S.	312-332-0707	info@tribecaflashpoint.com
London, Kitchener, Hamilton, Mississauga and Toronto	ON	N8T 1E9	CA	888-282-3893	info@trios.com
São Paulo	São Paulo	04707-000	BR	55-11-50955605 or 55-11-50955636	gamedesign@anhembi.br
Dundee	—	DD1 1HG	GB	44 1382 308080	sro@abertay.ac.uk
Tempe	AZ	85283	U.S.	800-65805744 or 602-383-8228	admissions @uat.edu
Baltimore	MD	21201	U.S.	410-837-5473 or 6222	sde@ubalt.edu
Santa Cruz	CA	95064	U.S.	831-459-5840	advising@soe.ucsc.edu
San Diego	CA	92122	U.S.	858-534-6705 or 6731	dac@ucsd.edu
Irvine	CA	92697-3430	U.S.	949-824-5156	ucounsel@uci.edu
Orlando	FL	32801	U.S.	407-823-2121	info@fiea.ucf.edu
Colorado Springs	CO	80918	U.S.	719-255-3150	chamillard@cs.uccs.edu
Denver	CO	80208	U.S.	303-871-2458	leut@cs.du.edu

SCHOOL NAME	ADDRESS
University of Gotland	Cramergatan 3
University of Houston	University of Houston Department of Computer Science 501 Philip G. Hoffman Hall
University of Houston - Victoria	3007 N. Ben Wilson
University of Miami - Music Engineering	Frost School of Music 1314 Miller Dr.
University of Michigan - Dearborn	Computer and Information Science 4901 Evergreen
University of Montevallo	Hill House, Station 6505
University of Ontario Institute of Technology	2000 Simcoe St. North

CITY	STATE	ZIP	COUNTRY	PHONE	CONTACT EMAIL
Visby	—	621 57	SE	46 0498 299900	igspinfo@hgo.se
Houston	TX	77204-3010	U.S.	713-743-3350 or 1606	ugradinfo@cs.uh.edu
Victoria	TX	77901	U.S.	361-570-4201	ArtsSciences@uhv.edu
Coral Gables	FL	33124	U.S.	305-284-8221	admission.music@miami.edu
Dearborn	MI	48128	U.S.	313-436-9145	bmaxim@umich.edu
Montevallo	AL	35115	U.S.	205-665-6505	games@montevallo.edu
Toronto	ON	L1H 7K4	CA	905-721-8668	admissions@uoit.ca

INFORMING, ENGAGING, AND EMPOWERING THE INDUSTRY
gamasutra.com
the art and business of making games
UBM TechWeb

SCHOOL DIRECTORY

SCHOOL NAME	ADDRESS
University of Ontario Institute of Technology	2000 Simcoe St. North
University of Pennsylvania	Department of Computer & Info Science 3330 Walnut St. Levine Hall
University of Prince Edward Island	Dept. of Computer Science & Information Technology
University of Southern California - Interactive Media Division	University Park Campus 900 W. 34th St., Ste. 222
University of Technology Sydney (UTS)	FEIT, UTS P.O. Box 123
University of Utah	375 S. 1530 E. Rm. 257B
University of Utah: Entertainment Arts and Engineering Master Games Studio	370 South 1530 East
University of Verona	Strada Le Grazie 15
University of Washington - Professional and Continuing Education	4333 Brooklyn Ave NE
Vancouver Animation School	206-1551 Johnston St.
Vancouver Film School	200 - 198 West Hastings St.
Vancouver Institute of Media Arts (VanArts)	600-570 Dunsmuir St.
Wake Technical Community College	9101 Fayetteville Rd.
Worcester Polytechnic Institute	100 Institute Rd.
Zurich University of the Arts	Ausstellungsstrasse 60

CITY	STATE	ZIP	COUNTRY	PHONE	CONTACT EMAIL
Toronto	ON	L1H 7K4	CA	905-721-8668	admissions@uoit.ca
Philadelphia	PA	19104	U.S.	215-898-8560	cggt@cis.upenn.edu
Charlottetown, PE	PEI	C1A 4P3	CA	902-628-4330	csit@groupwise.upei.ca
Los Angeles	CA	90089-2211	U.S.	213-821-4472	usc_interactive@cinema.usc.edu
Sydney	NSW	2007	AU	61 2 9514 2666	feit@uts.edu.au
Salt Lake City	UT	84112	U.S.	801-581-5127	darci.berg@utah.edu
Salt Lake City	UT	84112	U.S.	801-585-6491 or 801-581-7110	corrinne.lewis@utah.edu
Verona	Italy	37134	IT	045 802 7069 or 045 802 7069	game_dev@sci.univr.it
Seattle	WA	98195-9485	U.S.	206-685-8936	info@pce.uw.edu
Vancouver	BC	V6H 3R9	CA	888-677-8827 or 604-612-1171	info@vanas.ca
Vancouver	BC	V6B 1H2	CA	604-685-5808	admissions@vfs.com
Vancouver	BC	V6B 1Y1	CA	800-396-2787 or 604-682-2787	info@vanarts.com
Raleigh	NC	27603	U.S.	919-866-5390	crfoster@waketech.edu
Worcester	MA	01609	U.S.	508-831-4977 or 5000	imgd@wpi.edu
Zurich	ZH	8005	CH	41 43 446 32 54	verena.romanens@zhdk.ch

BY PATRICK MILLER

MAKE GAMES FOR FREE

You don't need to spend a cent to start making games. We've rounded up a list of professional-caliber tools and assets for would-be coders, artists, soundsmiths, game designers, and producers so you can build your portfolio without breaking the bank.

INTEGRATED DEVELOPMENT ENVIRONMENTS (IDES)

Visual Studio Express

www.microsoft.com/visualstudio

» If you're building games for Windows, Windows Phone, or Xbox 360, you should be coding in Microsoft Visual Studio. The Express version of Visual Studio is available as a free download from Microsoft's web site and should be more than enough to get you started building your dream game in C++, C#, or even Visual Basic. Also, if you're working on a game in Visual Studio Express, you can take advantage of Microsoft's free game-specific toolset, XNA Game Studio (http://msdn.microsoft.com/en-us/library/bb200104.aspx), which includes several game templates to help you get started. Keep in mind, though, that XNA Game Studio support gets tricky once Windows 8 comes out.

Win

Xcode (Apple)

www.developer.apple.com/xcode

» If you want to make a game for an Apple platform (Mac OS or iOS), you'll be coding with Xcode. All you need to use Xcode is a Mac and a free Apple Developer account (free registration on the Xcode web site). Unlike Visual Studio Express, Xcode doesn't really have a game developer-friendly toolset like XNA, though the Apple Developer Library (https://developer.apple.com/resources) does have sample code that can help you get up to speed with things like drawing 2D/3D images to the screen or working with iOS Game Center.

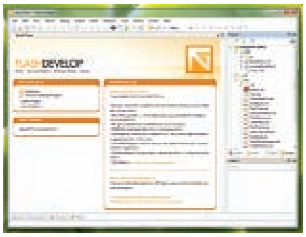
Mac

Eclipse (Eclipse Foundation)

www.eclipse.org

» Eclipse is a free, open-source IDE predominantly used by Java developers, though you could extend it to work with Python, C/C++, and JavaScript, among other languages. Eclipse is actually a Java-based application itself, so if you like working with Eclipse, you could use it on a Mac, Windows, or Linux PC. It's worth noting that Eclipse is a popular IDE for building Android games with the Android SDK (http://developer.android.com/sdk/index.html), so if you're looking to make mobile games, it can't hurt to have some Eclipse experience.

Win Mac Linux



FlashDevelop.

FlashDevelop

www.flashdevelop.org

» Flash is still a very popular platform for game development, and FlashDevelop is a very popular open-source IDE for Flash developers because of its speed and customizability (and that it's free, of course, compared to $700 for Adobe Flash Professional). Unlike Adobe Flash Professional, FlashDevelop is code-only; it lacks a user-friendly drag-and-drop interface. Fortunately, FlashDevelop is so popular that there are loads of easy tutorials for making Flash games out there (including this one on the Kongregate Flash game portal www.kongregate.com/games/Paltar/flashdevelop-tutorial). Once you're used to working with FlashDevelop, you can take advantage of game-specific code libraries such as FlashPunk (http://flashpunk.net) and Flixel (http://flixel.org) to make games with help from some of the biggest Flash game devs out there.

Win

GAME ENGINES

GameMaker Lite

www.yoyogames.com

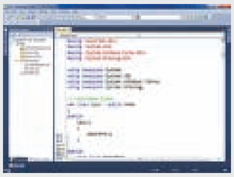
» Want to make games even though you're not much of a

CHOOSING YOUR CODING TOOLS

/// It's not always easy to figure out which programming tools you should use when starting a new project—but it's also not easy to change tools midproject once you've realized you've made a mistake. Here are a few tips on picking the right IDE and setting up source control for your game-to-be.

Picking a good IDE can help speed up coding a lot. The right IDE will streamline the way you write and compile code, and possibly even hook into other tools in the development pipeline.

Your IDE will be determined in part by the language you want to code in and the platform you're developing for. If you're using a Microsoft family of languages (C#, for instance), you'll want to use Microsoft's Visual Studio—Express is fine, though if you're still a student you should check and see if your school is part of the DreamSpark Premium program (formerly known as MSDN Academic Alliance), because you might be able to get Visual Studio Pro for free. For iOS/Mac development, you pretty much have to use a Mac with Xcode. For Java development, Eclipse is a widely used open-source IDE that is very refined. While Eclipse is largely known for its excellent Java support, it does work with other languages as well.

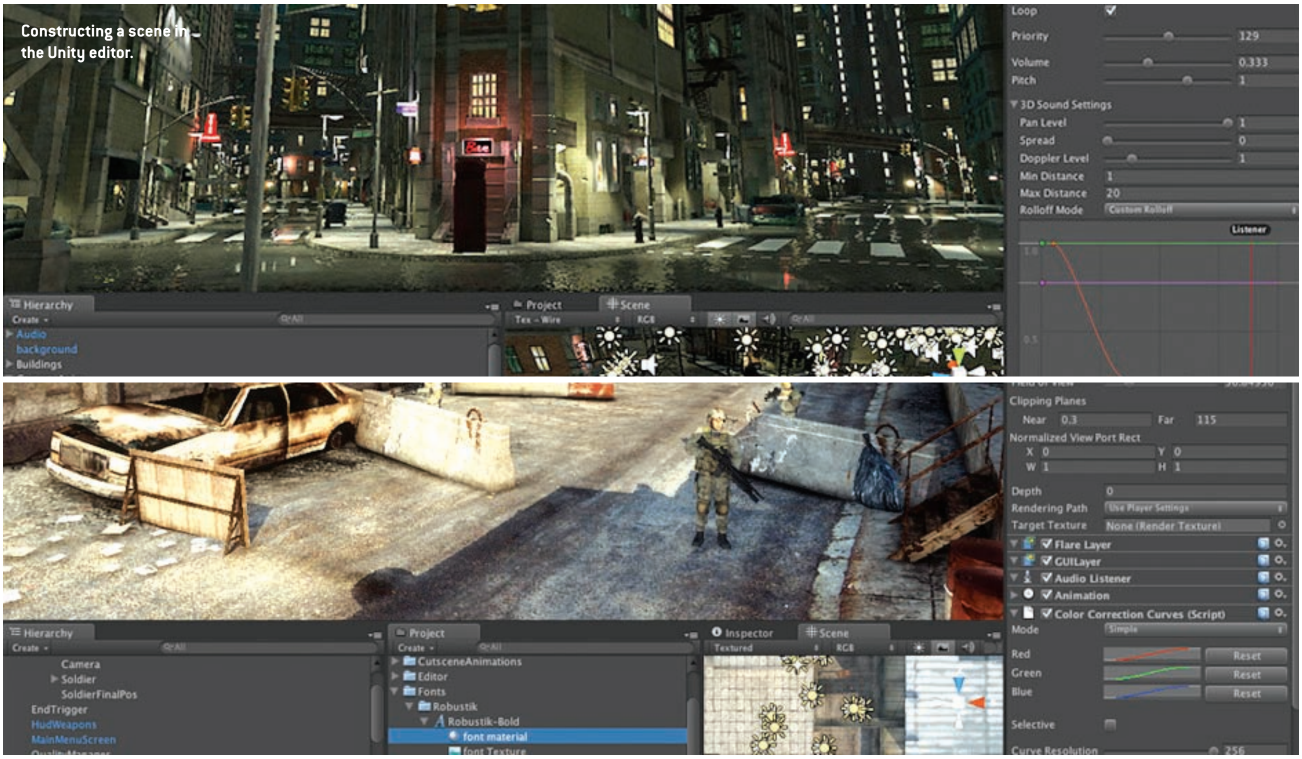


If you simply can't find the right IDE for your workflow, you might want to try going old-school and sticking to a text editor. VIM, Notepad++, and Emacs are all excellent text editors for writing code. They don't offer the same one-stop solution for coding that you get from an IDE, and they can take more effort to set up, but with the right editor you can create your own versatile, personalized coding workflow.

Next up is source control, which is an absolute necessity whether you're working alone or with a team. Most people use source control to manage multiperson projects where different programmers need to work on the same project simultaneously and merge their changes. However, solo developers can benefit from a good source-control program as well—it lets you keep backup snapshots of your program at various stages of development and revert to previous versions of the game in case something goes wrong. Working without source control is usually inviting some kind of disaster.

There are lots of different source control programs that exist. Subversion (http://subversion.apache.org) is a good one, and I often recommend it to people without much source-control experience. Tortoise SVN (www.tortoisesvn.net) is an excellent SVN client for Windows that makes using SVN a lot less intimidating for newcomers. For OSX, XCode has built-in SVN support that's pretty good, too. Just make sure you have something in place before you start any serious coding!

—Elijah O'Rear



Constructing a scene in the Unity editor.

Loop ☑
Priority 129
Volume 0.333
Pitch 1
3D Sound Settings
Pan Level 1
Spread 0
Doppler Level 1
Min Distance 1
Max Distance 20
Rolloff Mode Custom Rolloff

Clipping Planes
Near 0.3 Far 115
Normalized View Port Rect
X 0 Y 0
W 1 H 1
Depth 0
Rendering Path Use Player Settings
Target Texture None (Render Texture)
☑ Flare Layer
☑ GUILayer
☑ Audio Listener
☑ Animation
☑ Color Correction Curves (Script)
Mode Simple
Selective □
Curve Resolution 256

coder? Give GameMaker a shot. With GameMaker, you can build simple games from an easy-to-use drag-and-drop interface. You can then graduate to working with the underlying GameMaker Language (GML) when you're ready. The free GameMaker Lite is a bit limited in that you can't take advantage of any 3D graphics or hardware acceleration features, and every game you make will have a GameMaker watermark. But if you like it enough to move up to the full version, it'll only cost you $40.

Win Mac

GameSalad Basic

www.gamesalad.com

» GameSalad is another user-friendly game development program, like GameMaker, though it's much newer. Unlike GameMaker, GameSalad has no underlying editable code whatsoever; everything in GameSalad happens from the drag-and-drop interface. Once you get the hang of it, you can build game prototypes very quickly, which is handy for quickly testing ideas out and making changes early on, even if you intend to code your final version. The basic version is free and lets you build games and publish them for the Mac, iOS, and the web (a Windows version is currently in an open beta-testing phase as well). The Pro version costs $300 a year and gives you access to features that let you add In-App Purchases to your game, iAds for iOS games, Android support, and other tools meant to help developers make a few bucks.

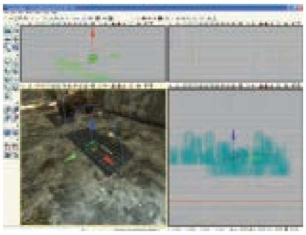
Win Mac

Unity

www.unity3d.com

» Unity's popularity has been surging, especially among smaller development teams that have outgrown more basic tools like Game Maker but don't want to create an entire game engine from scratch. Unity combines a designer-friendly visual game editor with solid support for JavaScript, C#, and Boo (a Python variant), and the free version gives you access to everything you'd need to build and compile a 2D or 3D game. You might not be able to afford the Unity Pro license yourself—at $1,500, it's kind of a stretch for a novice developer—but spend some quality time with the free version and your Unity chops might just get you a job working on Republique or the next Triple Town.

Win Mac



UDK.

Unreal Development Kit

www.udk.com

» The Unreal Development Kit powers some of the top games in the biz, including Mass Effect, Batman: Arkham Asylum, and, of course, the Unreal series. Why not get some experience building your own 3D games with the full-featured UDK? You don't have to pay a dime to work with the UDK until you plan on selling something you made with it, so it's a pretty good way to get your feet wet with the tools you very well might be working with in your first game-development job. Also, don't forget to check out our UDK tutorial on page 23 of this issue.

Win

2D ART

GIMP

www.gimp.org

» Photoshop is great. Photoshop also costs $600. The open-source GNU Image Manipulation Program ("GIMP"), on the other hand, is every bit as full-featured an image editor and won't eat into your rent money for the month. It's no Photoshop clone, so if you're a Photoshop power user, you'll have to spend some time getting used to a different interface. That said, GIMP simply cannot be beat when it comes to power and feature selection—especially at the $0 price point.

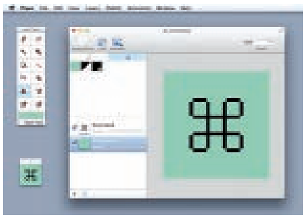
Win Mac Linux

Paint.net

www.getpaint.net

» GIMP and Photoshop are handy for industrial-strength image editing, but you should also keep Paint.net handy for more simple work. Paint.net is lightweight and easy to use—in fact, it feels like a buffed-up version of MS Paint—but it still has advanced features such as layers, numerous advanced selection tools, and many different effects.

Win



Pixen.

Pixen

www.pixenapp.com

» Pixen is a Mac-only app specifically designed for making retro-styled pixel art sprites that make your game look like it came from the NES/Genesis/SNES golden age of 2D. Detail-oriented game artists will undoubtedly appreciate Pixen's compatibility with pressure-sensitive drawing tablets and its support for custom color palettes, which makes it easier for you to mimic the color limitations of earlier game consoles.

Mac

GrafX2.

GrafX2

www.code.google.com/p/grafx2

» Windows and Linux users can't play with Pixen, but they can get GrafX2 instead—it's another bitmap paint program meant for pixel art, and it draws heavily from classic Amiga painting apps Deluxe Paint and Brilliance. GrafX2's color-palette management features let you work with different color systems and generate custom palettes for each image you're working with, which can come in handy for making authentic retro-style sprites. There is also a dual-view mode that lets you work on your sprite in a zoomed-in view while keeping an actual-size view to see what your changes look like.

Win Linux

Inkscape

www.inkscape.org

» Bitmap image editors are nice, but a well-rounded game artist should keep a vector-based illustration program on hand as well, especially if you'll be making games in Flash. Inkscape is a free, open-source alternative to Adobe Illustrator ($600), and while it doesn't completely duplicate Illustrator's feature set, many of the missing features are mostly specific to working with illustrations meant for print, not video games. Also, two particularly enthusiastic Inkscape users have posted over 100 screencasted video tutorials—check them out at www.screencasters.heathenx.org.

Win Mac Linux

ShoeBox

www.renderhjs.net/shoebox

» If there is one tool from this list that you absolutely must grab, it's ShoeBox. This app is useful because it streamlines a few game dev-specific tasks that can be a tremendous pain for a budding game artist, such as building sprite sheets out of animated GIFs or Flash animations, extracting tile maps out of a game screenshot, or extracting textures and animations from a Flash document. It's like a Swiss Army knife for a 2D artist.

Win Mac

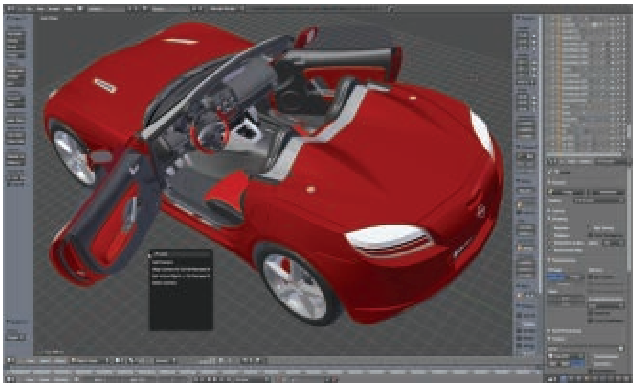
3D ART

Blender

www.blender.org

» Blender is a free, open-source 3D art program that is nearly as powerful as Autodesk's 3ds Max or Maya—and without the $3,500 price. Within Blender, you can build a 3D model, rig it, animate it, render it, and export a video clip of that animation just like you can with a paid 3D modeling program. It even has a neat built-in game engine you can use to actually make playable games from within Blender itself—try Yo Frankie! (www.yofrankie.org) to see how the Blender Game Engine works for yourself.

Blender.



Win Mac Linux

SketchUp

www.sketchup.google.com/download

» SketchUp (formerly Google SketchUp; as of this writing, SketchUp is still hosted by Google but has been acquired by Trimble) is a free 3D modeling and blueprinting application that is relatively easy to pick up and play with. You can use SketchUp to make finished 3D models, but this isn't really its strong suit; it's much better for quickly making polished-but-temporary placeholder models and even level geometry. For more on SketchUp, read our Quick and Dirty Game Art guide on page 51 of this issue, which walks you through the process of prototyping basic game art assets with SketchUp.

Win Mac

MeshMixer.

MeshMixer

www.meshmixer.com

» MeshMixer's web site describes it as "a free tool for making crazy-ass 3D stuff without too much hassle." With MeshMixer, you can import existing 3D models and resculpt them with a variety of tools and then attach them together to

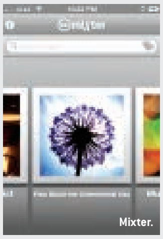
FREE SPRITES AND SOUNDS

You can't quite do your brilliant game prototype justice with stick figures and system sounds. Here's where you can grab plenty of free art and sound assets (usually licensed under various Creative Commons agreements) to use for your games. Just make sure to give credit where it's due.

ccMixter (www.ccmixter.org) hosts tons of songs, samples, remixes, and a cappella tracks you can use in your game.

OpenGameArt (www.opengameart.org) is another repository for art of all kinds: 2D, 3D, sound, music—even textures and UI elements.

freesound (www.freesound.org) specializes in sound bites: Rolling thunder, bottles opening, mood-setting instrumental tracks—it's all here. The random sound of the day is a neat way to keep up on what people are adding to freesound's growing database, too.



Mixter.

pdsounds (www.pdsounds.org) is another free sound repository that uses a tag-sorted database to organize its clips, so you can search for specific types of sounds very easily. You can even subscribe to RSS feeds for each tag, in case you absolutely need to know when a new explosion gets added to the list.

create a new model. MeshMixer probably won't replace Blender in your arsenal, but it's handy for generating quick-and-dirty models you can refine later on.

Win Mac

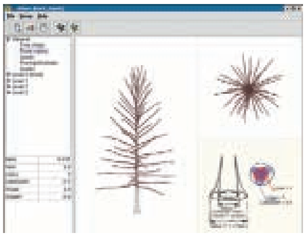
Makehuman.

Makehuman

www.makehuman.org

» Why build a human model from scratch if you don't have to? Makehuman is a free, open-source program specifically meant for making human 3D models within minutes. Just start with a base human mesh, modify it by tweaking various parameters (age, gender, weight, height, ethnicity, and so forth), and add hair, skin, or clothing from an included library, and you'll have a simple human model ready for Blender within minutes.

Win Mac Linux


Arbaro.

Arbaro

www.arbaro.sourceforge.net

» Just because you can't afford to drop a few grand for SpeedTree doesn't mean you're stuck modeling each and every larch or weeping willow in your upcoming tree-based FPS. (Adventures of Bayou Billy reboot, anyone?) Arbaro can easily generate all kinds of tree models from a few basic parameters and export them in a Blender-friendly format. It'll certainly come in handy for your Johnny Appleseed expansion pack.

Win Mac Linux

MUSIC AND SOUND

Audacity

www.audacity.sourceforge.net

» Audacity is a free and open-source sound editor that you can use to record, splice, and modify sounds for use in your game engine of choice. As a budding game sound designer, most of your work will probably start in Audacity, whether you're editing recorded dialogue, adjusting volume levels, or cleaning up your sound samples before passing them on to your game engine of choice.

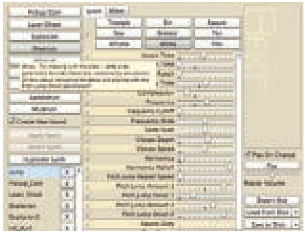
Win Mac Linux

Linux MultiMedia Studio (LMMS)

www.lmms.sourceforge.net

» There's at least one Grammy-winning game soundtrack inside you somewhere, and with LMMS you can extract it without having to shell out $200 for FL Studio or more for a high-end digital audio workstation. With LMMS, you can compose and edit songs (ideally with a MIDI keyboard), mix effects, and more. Bonus: LMMS comes with a set of free instrument plug-ins, including ones that emulate the Game Boy's and Commodore 64's signature sounds.

Win Linux

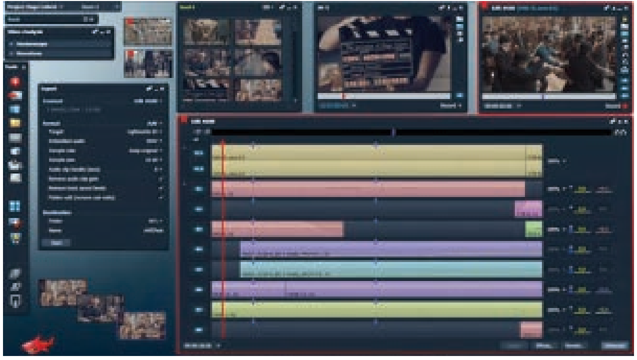

Bfxr.

Bfxr

www.bfxr.net

» Looking for some bleeps and bloops to fill your game's sparse soundscape? Bfxr is an easy-to-use web app that can generate whatever synthetic sound effects you need. You can either use Bfxr by starting with a general category of sound, like "Powerup" or "Explosion," and modifying the synth's parameters to make your own unique sound, or just mash on the Randomize button to roll the dice and see what sounds you come up with. Not a bad way to start your sound design on a shoestring budget.

Win Mac Linux


Lightworks.

MISCELLANEOUS

TargetProcess

www.targetprocess.com

» Small game-dev teams need producers, too—and if your game is an independent side project, your team might be even less able to afford losing time in development hell than a big team would. TargetProcess is a web-based agile project-management tool meant for small- to medium-sized teams, and they offer a free version for teams of five or less at targetprocess.com/free. If production buzzwords like "Scrum" and "Kanban" are what you need to make your ship date, give TargetProcess a shot.

Win Mac Linux

Lightworks

www.lwks.com

» Just because your game is on a low budget doesn't forgive the fact that your game's cutscenes and trailers reek of Windows Movie Maker. Grab the free version of Lightworks, an open-source nonlinear video editor similar to Final Cut Pro or Avid DS, and you'll be cutting Oscar-winning films in no time. (Well, it worked for *The King's Speech* and *Pulp Fiction*, anyway.) Be forewarned that you might need to shell out $60 for the Pro version if you need to use a few proprietary video codec packs, but even that is still pretty cheap for an incredibly powerful tool.

Win

Notepad++

www.notepad-plus-plus.org

» Everyone needs a good text editor, whether you use it for source code or grocery lists. Notepad++ is a venerable Windows text editor with tabbed windows for multiple documents, powerful syntax highlighting and folding features for coders, and macro recording features. You can even write your own Notepad++ automation scripts in Javascript, Python, or PHP.

Win

Tiled

www.mapeditor.org

» If you're making a 2D game, Tiled could very well be an indispensable part of your arsenal: It's a 2D map editor that lets you easily make your own tile-based maps (from either an isometric or top-down view) and save them in its own Tile Map XML (TMX) map format, which you can then integrate into your game. That way, you don't need to worry about building your own development tools before you start building your game. Instead, just bake TMX support into your game (the Tiled wiki includes sample code and tools for integrating TMX support into C++, Java, Flash, HTML5, Unity, and many other game-development environments) and you can jump straight to building game levels.

Win Mac Linux

Image courtesy of Sky Legends

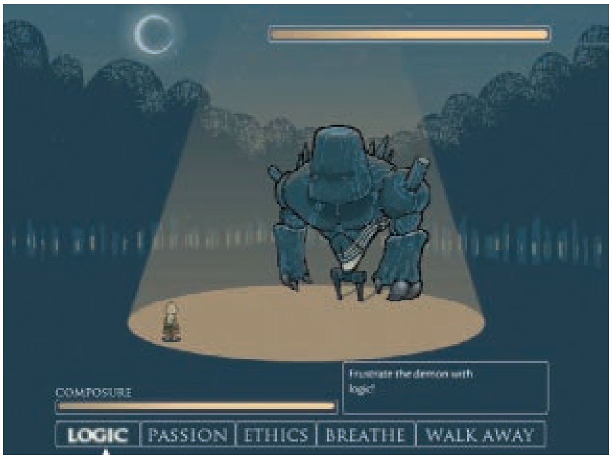
EDUCATED PLAY

Every issue of *Game Developer* magazine features a special section called Educated Play! which interviews the student developers or teams behind particularly interesting or noteworthy games. Here is a compilation of the six most recent Educated Play! installments, with student games from DigiPen, MIT Gambit Game Lab, and independent student developers.

A CLOSED WORLD

HTTP://GAMBIT.MIT.EDU/LOADGAME/ACLOSEDWORLD.PHP

IT'S RARE TO SEE GAMES WRESTLE WITH COMPLEX, AND PERHAPS CONTROVERSIAL SOCIAL ISSUES. A CLOSED WORLD, WHICH COMES FROM A TEAM OF RESEARCHERS AT THE SINGAPORE MIT GAMBIT GAME LAB, DOES JUST THAT, AND OPENLY EXPLORES LGBT AND QUEER THEMES IN A JRPG FORMAT. IN THE GAME, PLAYERS TAKE CONTROL OF A YOUNG PERSON WHO DECIDES TO ENTER HIS VILLAGE'S FORBIDDEN FOREST. ALONG THE WAY, THE PROTAGONIST MUST BATTLE INTERNAL DEMONS THAT QUESTION WHAT WE CONSIDER "NORMAL" AND WHAT IT MEANS TO BE TRUE TO ONESELF. WE SPOKE TO PRODUCT OWNER TODD HARPER TO LEARN MORE ABOUT THIS INTROSPECTIVE TITLE.



Frustrate the demon with logic?

Tom Curtis: *Where did the inspiration for A Closed World come from?*
Todd Harper: In September of 2010 there were a number of queer youth and bullying victim suicides in the news, including the story of Tyler Clementi. The mission of GAMBIT, the lab where I work, is to use game design to do research, and my background is both in game studies and queer theory. So I thought, "How would a game deal with talking about this stuff?" The eventual end product didn't have all that much to do with that issue, but that was part of the spark.

TC: To what extent did the team draw from personal experience when creating the game?
TH: This is a difficult question to answer. I know my own past experience went into discussions I had with the team about things like coming out and issues like that, as did one of the interns who identified as gay. I don't think that the game's content came as much from their actual personal experience as it did from the team making a big effort to come to grips with what the issues involved with being a queer person even are.

The room they were in at GAMBIT has a glass wall that can be written on with dry erase marker. The team drew this amazing brainstorming thing that was dubbed The LGBT Spider, and it took up the entirety of the eight-foot high wall by the end of the project, having morphed into the LGBT Kraken. There's this big, amorphous concept of "the LGBT experience" that they—like any other dev team—had to wrap their heads around.

TC: In general, how have audiences responded to the title? Were you surprised by how people reacted?
TH: Responses have run the gamut. There are players who really like everything, from story and gameplay on down. There are those who thought the gameplay was too simplistic, and the story was too cliché. In general, feedback told us that the combat system was an interesting idea, but perhaps too simple, which I'd expect from something with an eight-week development cycle. Some people were big fans of the story, others thought it was too simplistic—observe Anna Anthropy's parody game A CLOSED MIND (www.auntiepixelante.com/?p=1276) on that front. The really consistent thing, though, regardless of opinions on the game, is that this topic and this idea are important. Lots of people who weren't the biggest fans of the game still said "I'm glad someone tried this."

TC: While the team seems to openly acknowledge A CLOSED WORLD's underlying themes, the game itself is fairly subdued and ambiguous. Was this a conscious decision on the team's part?
TH: I'd say so. We had this constant battle with what we called "the hammer," where we come down with some top-heavy, word of God message about insert-LGBT-theme-here. Nobody likes that in a story or a game. Plus, our big conceit was this procedurally generated gender concept, where the genders of the people you meet (and in the backstory) are just picked for you at random. So, we needed a narrative that supported that, but that still got across the message we wanted to send, which is this idea that being true to yourself is hard, and might require sacrifice, but in the end is a much better option than the alternative.

Team Members
Todd Harper product owner
Abe Stien game director
Sophia Yuen Shu Hui producer
Lex Johnson QA Lead
Praveen Namasivayam designer
Bree Westphal artist
Peter Tan artist
Kevin Laughlin artist
Tran Hung Nguyen programmer
Jovi Tan programmer
Casey Merhige audio designer

TC: The game seems to focus more on its themes and message than it does gameplay itself. Why did you choose to emphasize narrative so heavily over the mechanics?
TH: Before the summer we had a six-month prototyping period, where we worked over various ideas about the shape of the game. At first we really wanted to do all the work in a procedural way. We spent a lot of time looking at various parts of what a queer individual's experience might be and saying "Okay, how can we turn this into a gameplay mechanic?" The real problem is we couldn't find one that was satisfying to us, and part of that was really just the huge amorphousness of grasping "a queer experience" in the first place. So in the end, we decided to let the narrative do the heavy lifting, ideologically speaking, and tried to focus on having the gameplay deliver and reinforce the narrative.

DEITY

HTTP://WWW.DEITY-GAME.COM

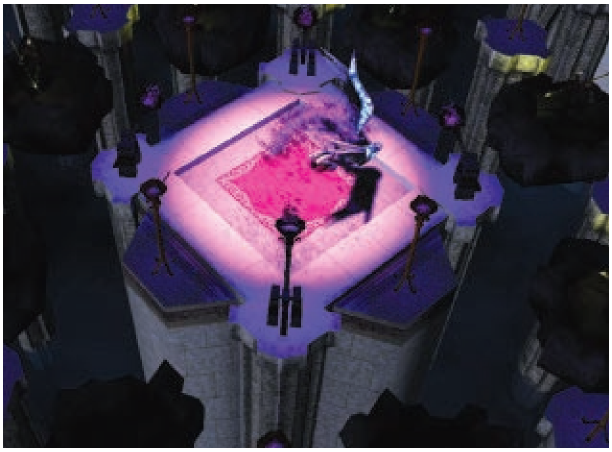
THERE'S NO DENYING THAT STUDENT PROJECT DEITY HAS AMBITION. THE GAME TAKES CUES FROM THE LIKES OF DIABLO, ASSASSIN'S CREED, AND BATMAN: ARKHAM ASYLUM, BLENDING ISOMETRIC ACTION WITH STEALTH-BASED STRATEGY. PLAYERS WORK THEIR WAY THROUGH A DANGEROUS MEDIEVAL CASTLE, ALL THE WHILE HUNTING GUARDS, LURKING IN THE SHADOWS, AND OTHERWISE ESCAPING CERTAIN DOOM. GAME DEVELOPER RECENTLY SPOKE TO THE DIGIPEN STUDENT TEAM BEHIND THE GAME TO LEARN MORE ABOUT ITS INSPIRATION AND ORIGIN.

***Tom Curtis:* How did you all come up with the concept for DEITY?**
During the early stages of the project, most of our ideas focused on creating a stealth game emphasizing light and shadows. The first playable prototype used a third-person camera, where the player had to manipulate the environment's lights and hunt guards holding flashlights, but we settled on the isometric perspective because we identified a nostalgic charm attached to it—it also removed the need for additional camera controls.

***TC:* For a game of this scope, DEITY is surprisingly complex, both aesthetically and in terms of its stealth- and action-based gameplay. Was this part of the original intent? How did you all coordinate/delegate the workload to get everything done on time?**
We definitely [wanted] to make DEITY feel strategic, but our main priority was to ensure that the game was fun. Continuous playtesting resulted in the team making gradual improvements to the mechanics and level design that revolved around our central ideas. Oftentimes, we would make drastic gameplay changes several days before milestones because we realized that it just wasn't as fun or engaging as we hoped. Most of DEITY's artistic inspiration came from dungeon crawlers, and we wanted to maintain that same gritty look and feel that most of these games feature.

The team would have an in-depth meeting after each milestone to decide what our individual goals were for the next project milestone. We would also officially meet twice a week during the times that our schedules didn't conflict (conducting SCRUM-like sprints so we could quickly discuss our progress, problems, and work plans).

Having to deal with schoolwork and a full game project wasn't easy for us, and we would often have to endure unforeseen delays. As such, most of our ideas and concepts were planned with scalability in mind. Our art assets were reusable (environment pieces were modular and tile-able) and our gameplay features could be cut.



***TC:* What games were your greatest influences, and how did they impact the final project?**
BATMAN: ARKHAM ASYLUM and ASSASSIN'S CREED were some of our sources of inspiration for the stealth experience. Being able to swoop in and out and eliminate your targets without being caught was something we loved about those games. The idea was to bring a similar experience to DEITY: You had visible superiority over your enemies, but your movement and attack strategies determined your success. We wanted to deliver a unique experience while preserving a similar stealth feel; gameplay mechanics and level designs had to be constantly tweaked to achieve that.

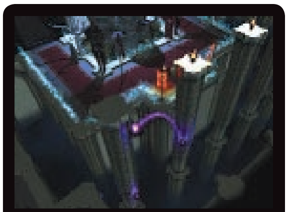
DIABLO and TORCHLIGHT had an influence on our artistic and interface design for DEITY. Those references helped us to identify the potential camera viewing problems inherent with our viewing angle (tall objects and walls had to be avoided) and we gained a greater understanding of darker, organic atmospheres. The artists on the team did a phenomenal job executing our vision, and really made the game stand out visually.

As for the controls, it was initially modeled after a combination of WORLD OF WARCRAFT and DIABLO, but we realized that the added complexity could have been devastating for players who were unfamiliar with those titles, so we streamlined the controls onto the mouse. This had a tremendous effect on lowering the learning curve, which led to non-gamers picking up and enjoying our game.

***TC:* Any lessons you took from DEITY's development?**
One of the most important lessons we took away from the project was that developing an engaging, polished game requires teamwork. Being able to trust in each of our abilities to deliver, respecting personal schedules, and valuing the opinion of every team member was crucial to improving productivity.

We learned to ask what each of us could do and not to impose tasks. The various roles in the game's production were volunteered for, not assigned, which led to every person putting in amazing effort without being compelled by others to work.

DEITY has been the most rewarding and successful project for all of us on the team, and a large part of what made that happen was the synergy and perseverance we managed to uphold.



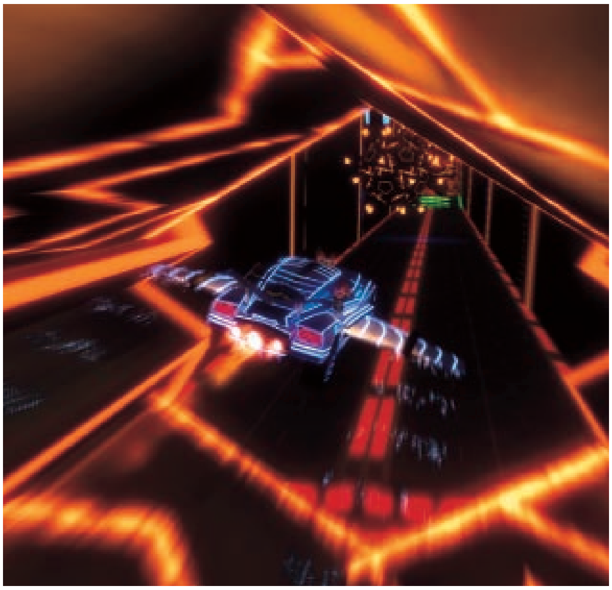
Developer/Publisher: DigiPen Institute of Technology
Release Date: 12/16/11
Platform: Windows PC
Number of Developers: 8
Length of Development: 15 months
Budget: $0
Lines of Code: Over 70,000 lines of C++ code
Fun facts:
Everyday Value Slams @ Denny's weekly consumption: 10
The Longest Day: 52 hours
Instant noodle packs consumed: Approximately 500

Team Members
Ryan Chew
Caroline Sugianto
Christopher Mingus
Ryan Hickman
Michael Travaglione
Ying Liu
Matt Frederick
Aariel Hall

NITRONIC RUSH

HTTP://NITRONIC-RUSH.COM

NITRONIC RUSH IS A BIT OF AN ODDITY WHEN IT COMES TO DRIVING GAMES. IT'S NOT ABOUT RACING OTHER DRIVERS, BUT SIMPLY ABOUT MAKING IT TO THE FINISH LINE IN ONE PIECE. THE GAME THROWS PLAYERS INTO A VIBRANT, TRON-LIKE WORLD WHERE THEY MUST AVOID TRAPS, SPEED BETWEEN OBSTACLES, AND EVEN USE A SET OF DEPLOYABLE WINGS TO FLY ACROSS THE TRACK. THE GAME'S UNIQUE DESIGN AND IMPLEMENTATION EARNED IT AN AWARD AT THIS YEAR'S INDIE GAME CHALLENGE, AND AN HONORABLE MENTION AT IGF 2012. WE CHATTED WITH THE DIGIPEN STUDENT TEAM BEHIND THE PROJECT TO LEARN ABOUT ITS PRODUCTION AND EVENTUAL SUCCESS.



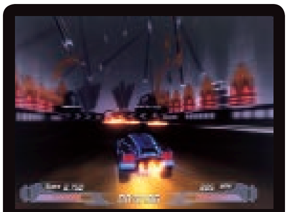
***Tom Curtis:* What were some of your biggest influences on the project? I'm guessing** Tron: Legacy **played a pretty big role?**
Kyle Holdwick (executive producer, gameplay programming, obstacle logic): Overall, we were definitely inspired by a number of '90s arcade racing games like Rush and Hydro Thunder. Visually, we always wanted to go with a futuristic cyber-based style, and we certainly did look at *Tron: Legacy* for reference. Musically, we were influenced by retro game music and the influx of the modern electro house movement.

***TC:* The game is surprisingly full-featured for a student project (with achievements, multiple modes, and so forth). What was your strategy for implementing all of these various systems on time?**
Jordan Hemenway (audio director, music composition, sound design, web site development): Scope was definitely a challenge for the team, but we did our best to prioritize between needs and wants. Oftentimes during development it seemed like there was a monumental list of features waiting to be implemented. Despite this, we made sure the foundation for new features and mechanics was well laid for us to build upon. Thanks to that, we ended up being largely content-driven during the last couple months of development, adding as much music, levels, and polish into the game as we could.
KH: In many ways we took the development of Nitronic Rush day by day, allowing our inspirations at the moment to guide us. This allowed for our design methodology to be very flexible and natural. When we would have a design meeting, we would look at what our playtesters wanted and figure out what we wanted to work on together. This approach allowed for a number of creative decisions to be made by many different team members throughout the entire development cycle of the game. Since our strategy was to be self-driven, motivation was usually high and finishing tasks on time was easier.

***TC:* You worked on the game for 17 months, correct? Is that sort of development cycle typical for other student projects at DigiPen?**
JH: Usually a DigiPen game project takes two semesters (fall and spring) with a possible additional semester during the summer to add polish for competitions. Nitronic Rush's development, however, lasted five semesters in total (from May 2010 to November 2011).

We actually started a semester early to build tech for a shared architecture called Superdyne (used in another game Kyle and I worked on called Solstice), and due to teacher encouragement we continued the project through the end of the year. It's definitely atypical for a DigiPen game, but with how ambitious the design was I'm glad we were able to see it to the end.

***TC:* What were the biggest challenges during development?**
Andy Kibler (game designer, level design): Physics is by far one of the hardest elements of a racing game. The car physics have to be solid in order for anything to happen on-screen. Jason Nollan did a great job, and it definitely shows.
JH: In terms of the overall team, I agree that the physics and overall car controls were probably the most challenging problems we had to solve. Otherwise, I'd say that having a polished beginning and ending to every piece of the project was definitely a challenge. Oftentimes we'd get stuck working on the core part of a level, menu, or story mode arc while leaving the loose ends unpolished until later on.



Developer/Publisher: Team Nitronic
Release Date: 11/11/11
Platform: Windows PC
Number of Developers: 11
Length of Development: 17 months
Budget: $0
Lines of Code: 132,551
Fun fact:
Two members of Team Nitronic are actually twin brothers, Andrew Nollan and Jason Nollan.

***TC:* If you were to go back and do one thing differently on this project, what would it be?**
KH: If we were to go back and change some things, we would definitely add both competitive and cooperative multiplayer to the game. We would add more vehicles with a broader range of abilities. We would also add more levels with more of an emphasis on wall riding and flying.
AK: I should have made more levels for the game. I know I made quite a few, but I know I could have made more!
JH: Fleshing out the story quite a bit more would be have been really interesting. Everyone had visions of a fairly grandiose story early on in development, but it wasn't until the very end that we created the cut scenes and started laying out the actual story within the game. If we had more time, it would have been great to spend some proper time creating a more elaborate story arc throughout the game's story mode.

NOUS

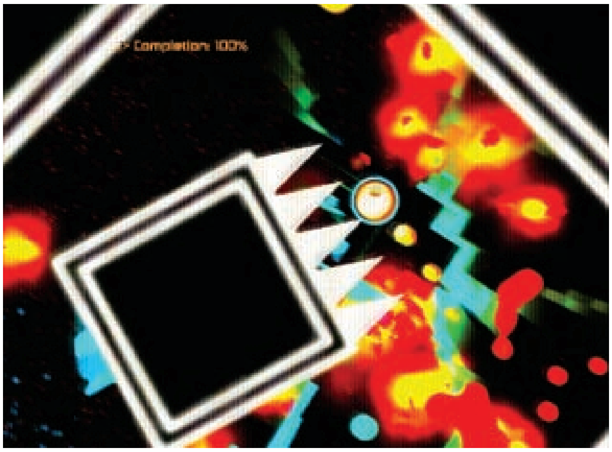
WWW.WHATISNOUS.COM

IT'S HARD TO PUT A LABEL ON AWESOME SHARK VOLCANO'S IGF STUDENT SHOWCASE FINALIST NOUS. AT FIRST GLANCE, IT LOOKS LIKE A SIMPLE ARCADE GAME WITH A RATHER RETRO AESTHETIC—UNTIL YOU ENCOUNTER THE UNHINGED AI NARRATOR. READ ON TO FIND OUT HOW FOUR DIGIPEN STUDENTS MANAGED TO MAKE NOUS —AND HOW THEY MIGHT HAVE GOTTEN A LITTLE BIT UNHINGED IN THE PROCESS.

***Patrick Miller:* Tell me a little bit about the team behind Nous. How'd you meet, and who did what?**

Pohung Chen: We're a team of four students at DigiPen Institute of Technology. Nous was our sophomore year game project. I wrote physics code for the custom engine. A lot of the physics code didn't end up being used in the final game beyond basic collision detection and resolution. I also helped the team with scheduling, figuring out where we're going, and ran playtest sessions. Treb Connell worked on the core engine and wrote our Lua binding.

Jason Meisel: I was the graphics programmer on the project as well as the "co-designer" and eventually the artist—that one surprises me. I came on to the project wanting to learn how to do cool graphics effects, and I think I was pretty successful. I also made many of the prototypes that we experimented with over the course of development as well as a good amount of random gameplay code.



Brett Cutler: I was the last to join the team and was handed the clay of our ideas, tasked with shaping it into a game as the designer. That involved scripting the gameplay, the levels, writing the script, running playtesting, and polishing everything player-side.

***PM:* Where'd the idea for Nous come from? Was the AI "personality" there from the beginning, or did it come after other game mechanics were set down?**

BC: We were a team driven to create something great, but we lacked a killer idea. We spent most of our development period throwing out prototypes. Before Nous, we had "Dr. Gravity and the Invention of Gravity;" before that, "Spaceburnium." By the end of our regular production, we were known mostly as the game that kept changing. So we made that the focus of the game.

The frustration and indecision we'd fought against for the whole year was poured into a manipulative, confused, and insane narrator. The AI personality grew into this space, personifying our development and exploring the concept of identity; namely, though our game had changed radically, we still felt it was the same project. Could we build a character with radically different personalities that felt unified? What composes a character—or an identity?

***PM:* What inspired Nous's visual style?**

JM: As our project evolved, so did its visuals. Throughout the project we struggled with mixing 2D gameplay with 3D visuals, and for a long time the game looked like a random assortment of sprites and effects rather than a cohesive piece. When we figured out the theme of Nous, we figured out an art style to go with it, one that would utilize the effects in more complementary ways. There was some definite inspiration from Geometry Wars, being simple and computeristic. The color palette was inspired by Portal, or any DVD cover, and after consulting with some of the artists at DigiPen, we were able to really make it stand out.

BC: There's a stage in any creative's development where they just want to provoke a reaction. Visual noise, flashing lights, and loud sounds feature heavily in the game, inspired by the opening credits of the movie Enter the Void. The chaos is punctuated by silence—by pure, clean letters on a black void.

***PM:* What did you use to prototype and develop Nous?**

PC: We wrote our engine using C++ and Lua. Integrating Lua into our game really sped up our code-iteration speed. We were able to use it to prototype a lot of different ideas fairly quickly.

Because we were only four people building the entire game from scratch and we each had four to five other classes to take care of, some things were left in a barely good-enough state. One good example of this was our level editor. It was thrown together quickly and wasn't very usable. We didn't end up making it user-friendly enough to be all that useful. A lot of the content was tediously thrown together using a combination of the editor and Lua code.

BC: I still wonder about the prototypes we trashed. Could they have been better than Nous, given time to shine? How do you know when you've developed a prototype far enough to test it, or whether you're abandoning it just before you would have hit a breakthrough that would unlock the fun?

***PM:* Is your team going to continue to work together on anything else?**

JM: Pohung and I have been working on another project for about a year, and it's given me a whole new perspective on game development. We're expecting to release it within the year.

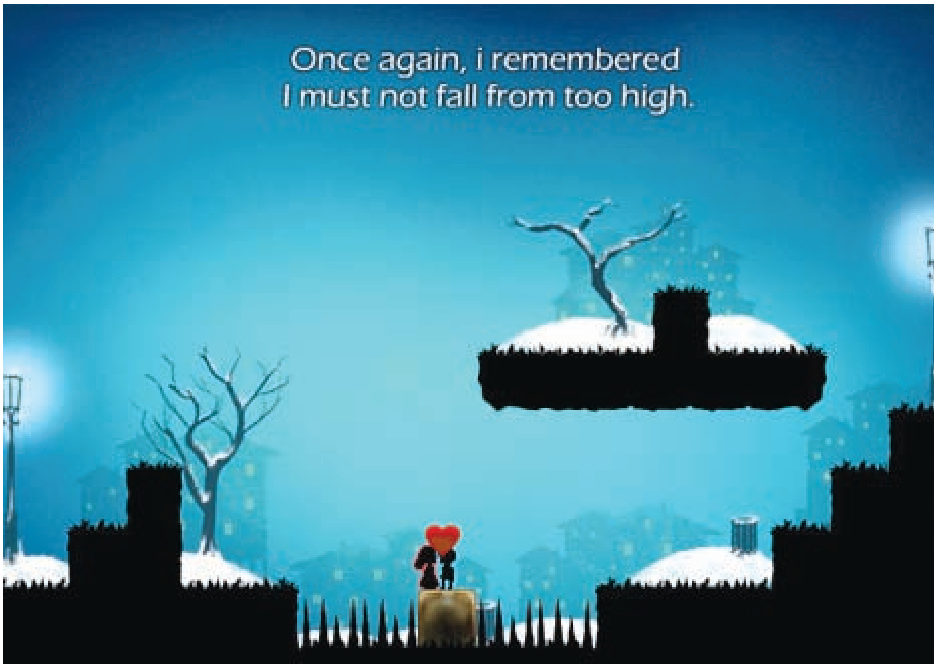
BC: Teams shift every year at DigiPen, and we try to work with a mixture of new people and old friends. Treb, Pohung, and I have been working on another project—and trying to build a game very different from Nous. It involves rhinoceroses.

Team Name:
Awesome Shark Volcano (DigiPen)
Release Date
10/11
Development time:
About a year
Development Budget:
$0.00
Lines of code in the game:
40,000
A fun fact about the development process:
Composer on the cheap—some of the music in Nous was made with Paul Stretch, a program that slows down classical music to a quarter of its normal speed.

ONE AND ONE STORY

WWW.MATTIATRAVERSO.COM/GAMES

PLENTY OF NEW GAME DEVELOPERS CHOOSE TO START OUT WITH A SIMPLE 2D PLATFORMER, BUT FEW WOULD THINK TO MAKE A 2D PLATFORMER BASED ON A ROMANTIC GAME MECHANIC LIKE IGF STUDENT SHOWCASE FINALIST ONE AND ONE STORY. WE TALKED WITH ONE AND ONE STORY DESIGNER AND PROGRAMMER MATTIA TRAVERSO ABOUT THE DEVELOPMENT PROCESS AND THE TRANSITION TO IOS.



Release Date: 10/10/2011
Length of development: 3 months. 90% of the game was finished in a month, but we spent two months polishing the final 10%.
Budget: One half-eaten lollipop. That's it.
Lines of code: 4,000+
Fun Fact: An early build of the game had an ugly brown background, and one playtester said "Those five minutes were the worst of my life, I was just wandering around a brown nothing," which prompted Gabriele to make the great art we used in the final build. Guess we should thank that tester today.

***Patrick Miller:* Tell me about the team behind ONE AND ONE STORY. Who worked on it, and how did you all meet?**
Mattia Traverso: I made the core of the game, (with very rough graphics) in about a month, and then I started looking for artists and musicians. David was easy to find because he was very well known in the Flash game scene, but before finding Gabriele I wasted two months with various artists around the web who couldn't get the mood of the game in their drawings and mock-ups.

I realized a few days ago—maybe seven months after the development of the game—that I have never talked directly with them! We used Skype to text, but we've never heard each other's voices. Strange, huh?

***PM*: How did you come up with the design for ONE AND ONE STORY? Did you draw upon any life experiences or memories?**
MT: I think the best way to describe the game is that I had a relationship, but it was full of crates and spikes, so we had to break up!

Joking apart, the design phase worked in a similar way to Jenova Chen's design process, but I'm not saying I'm that good, of course! His design philosophy is different from the common one: Normally, you take a mechanic and make sure it is fun. But Jenova does not start with a mechanic; he starts from an emotion, from a feeling that he wants to transmit to the player.

And that's how we worked on ONE AND ONE STORY, exploiting the game mechanics to work toward the narration and therefore the emotions of the player.

***PM:* Did you decide to do ONE AND ONE STORY in Flash from the beginning, or did you start with something else?**
MT: Sadly, I was forced to use Flash because I'm just starting my developer career and that's the only tech I know. Since then, I have come to know a lot of programmers. But back then, nobody would have listened to a boy with such a simple idea. We are now remaking the game for iOS devices thanks to a very good C++ programmer, Tommaso Checchi.

***PM*: In March, you found an unauthorized copy of your game on the iOS App Store. How did you respond to that? Were you able to find anyone who could help you deal with the legal matters and get the copy pulled off the store?**
MT: When I discovered the copy, I shouted some not-so-polite words on Twitter. Surprisingly, that helped a lot: I was retweeted by around 100 indie developers, and my problem got spread around the web very quickly. I got a lot of contacts willing to help me, to name a few: Adam Atomic (CANABALT creator) sent me the DMCA module to take the app down, IGDA and the Vlambeer guys (SUPER CRATE BOX) contacted Apple. I got a fantastic wave of support from the community. A few lawyers also contacted me even though I never intended to sue the "thieves." After two weeks or so, the app was removed from the store, though I haven't heard anything else from Apple!

***PM:* Now you're working on your own iOS port. Is it hard to make ONE AND ONE STORY work well on a phone and tablet?**
MT: Yes! Since it's a platformer, the controls are a major issue. I personally don't like games with a lot of buttons on the screen, because that's just a way to adapt a game that was not originally designed for iOS. I'm trying to rethink the control scheme to work without a single button on-screen. The system is up: Now we just need to playtest it!

Another problem is that the game is too short to be in a paid form: We need to add content, but without ruining the atmosphere and the mood, which is tough.

Besides that, I should thank Apple: The art now needs to be redone twice as large for the new iPad!

VOID

WWW.DIGIPEN.EDU/?ID=1170&PROJ=23876

IN VOID, PLAYERS MUST COMPARE DIFFERENCES BETWEEN A RUINED WORLD OF THE PRESENT AND A PERFECT PAST WORLD IN ORDER TO PROGRESS THROUGH THE WORLD. IT'S A NEW KIND OF FIRST PERSON PUZZLER—AND ONE THAT HAD A LOT OF POTENTIAL ROOM FOR EXPLOITS. THE TEAM, WHICH WON BEST STUDENT GAME AT THE IGF CHINA 2011 AWARDS, DISCUSSES THE MAKING OF THIS UNUSUAL TITLE.

Tom Curtis: Can you give a little background on the origin of the project? How did you come up with the unusual design?

Mark Ravindran (designer): VOID was actually a student project created for the production phase of our course at DigiPen. I think its eventual design was actually a product of our limitations as well as some technological breakthroughs that we had while designing the project itself. For example, we didn't have much time [to work on] the project, and with a small team size, we figured that designing enemies with a substantial amount of AI would actually be rather difficult, so we kept things simple and created a first-person puzzler. Other things, like the dimension rip bubble, were a result of programmers discovering new ways of adjusting the engine.

TC: What technical challenges did you encounter?

Chan Sin Huan (programmer): The biggest technical challenge was the Source SDK itself. While we had access to the code that made HALF-LIFE the game it is today, the SDK only provides the code and assets for the game layer itself (the AI, weapons, monsters, and levels); it does not provide any access to the internals of the Source Engine itself. As the Source Engine is a "black box," this posed many challenges to us as we could not modify the engine to better suit our needs.

Leau Tat Sin (programmer): I was the graphical programmer on the team, in charge of creating the bubble effect. The first challenge I had was trying to figure out how to create custom shaders on the Source SDK. There were little bits of documentation, and most of them were a little outdated, since the Source SDK has undergone multiple updates since its release. There was a lot of trial and error involved.

Creating the effect also proved to be much more difficult than first expected. The basic effect was easy enough to create, but there were a lot of unforeseen artifacts, which took up most of the development time to fix. There were still a few issues that I never did solve, but unfortunately we ran out of time.

TC: Any design challenges?

Tan Chee Ming (designer): The key challenge was the doubled workload for everything we wanted to design and create. For every level or room we created, the designers had to create two layouts, and the artists had to model two versions of assets: a past (perfect) version and a present (destroyed) version.

It was also a huge challenge for both Mark and I as designers, since we had to ensure the level design for both past and present complemented each other, and that there was not any easy bypass or walkthrough breaks by changing time dimensions.

TC: If you could go back in time and do one thing differently on this project, what would it be?

LTS: I would have gone with the two-world approach right from the start. We essentially wasted the first half of the production time when we rebooted and threw everything away. The extra development time would have allowed us to fix the remaining issues, or at least find some kind of workaround.

TCM: On my part, I would have liked to engage this project with a clearer vision, so that the team wouldn't have had to go through so much trouble. Also, I would have liked to learn to filter feedback properly, since we received so much feedback that it actually caused us to become confused and lose our vision.

CSH: As this was part of a student course, for most of us, it was our first time creating games in an actual production environment, and it would definitely have been better to approach the whole project more methodically.

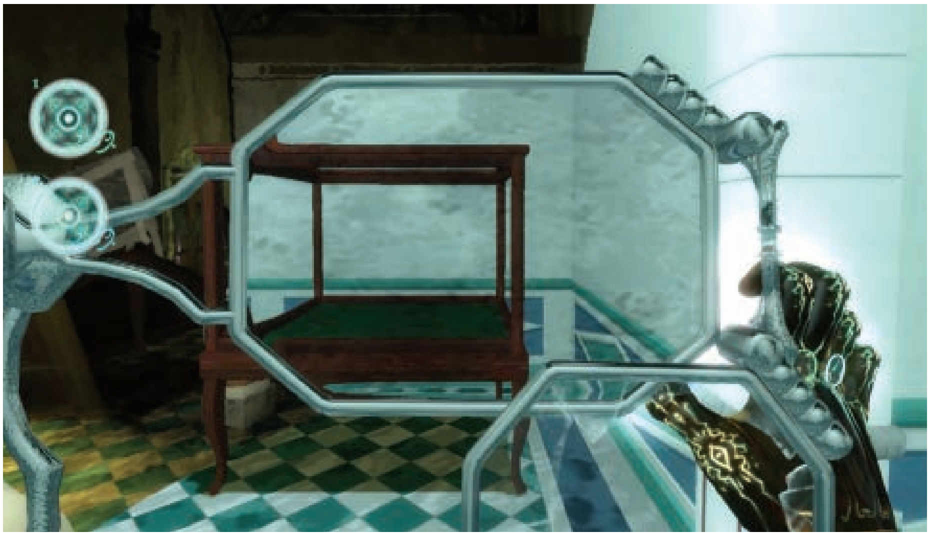
Zou Xinru (animator): I do sometimes wish we had the full five months to concentrate on the final iteration of VOID. It could have also gone down the same road of self-destruction, or it could have been much more awesome, with more levels, a third dimension, and maybe some monsters clawing at your back. At least, I would like to imagine that would be the case.

TC: How does it feel to earn the Best Student Game award at the IGF China? What's next for the team?

LTS: It's surprising, really. To be honest, throughout development, we all thought that we performed badly. For the longest time, at least until the reboot, the game simply sucked. In fact, we were in danger of failing the course. Good thing it all worked out!

MR: It definitely makes up for all the pain we had with redesigning and scrapping the ideas (*Laughs*).

TCM: Now that the project is finished, all of us are at separate companies right now, but I definitely wouldn't mind coming back together to further develop VOID if someone or some company expressed interest in it.



AIE ACADEMY OF INTERACTIVE ENTERTAINMENT

Welcome to the World of AIE

At the Academy of Interactive Entertainment (AIE), innovation is its own reward, our success is measured by your success, and experience counts for everything. Level up by completing industry-style projects taught by teachers with hundreds of years of collective experience making games and films. When you join us, you enter a realm of possibilities where you learn how to create new worlds, stories, and experiences. Welcome to AIE.

AIE is a non-profit educator established in 1996 to grow and support the game industry. Our unique curriculum is offered in the United States and Australia, and provides you with the training needed to join our other graduates as entrepreneurs, leaders, and innovators in the video game industry.

WE'VE GOT THE INDUSTRY CONNECTIONS.

AIE is one of the oldest, most experienced colleges in the industry, and has built strong ties to many world-famous companies. Our curriculum advisory board members come from such companies as Valve, Hidden Path, Zombie Studios, and 5th Cell. Our graduate destinations include: 2K Games, Atari, Bioware, Blizzard Activision, Electronic Arts, Rockstar Games, SEGA, Sony, THQ, Ubisoft, and many more.

CURRICULUM FOCUSED ON INDUSTRY SKILLS AND PLACEMENTS.

Our programs are taught in a hands-on, studio-style setting that simulates actual industry environments. During 1,800 contact hours with AIE faculty, students create showreels, complete team projects, and learn everything they need to know through a task-based learning system.

TEACHERS WHO HAVE "BEEN THERE AND DONE THAT."

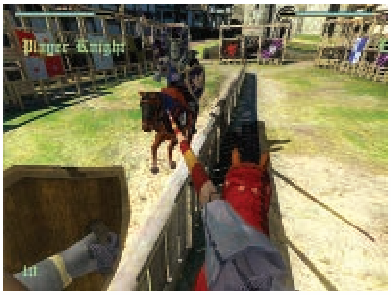
AIE faculty have countless years of industry experience, hundreds of game titles shipped, and numerous published papers and books. They offer invaluable insight into the world of game development and animation.

WE ARE WHERE THE ACTION IS.

AIE's Seattle campus is situated in the Seattle Center, where the iconic Space Needle can be seen from our classrooms. Seattle has long been a hub for the video game industry, home to Microsoft, Bungie, ArenaNet, Nintendo of America, Valve and many emerging independent game studios. Our Lafayette campus is located in the LITE Center, a state-of-the-art facility home to both industry and academic institutions where collaboration happens every day. Lafayette is the heart of Cajun culture and is an emerging center for the next wave of technologies for use in the film, military, and medical industries.

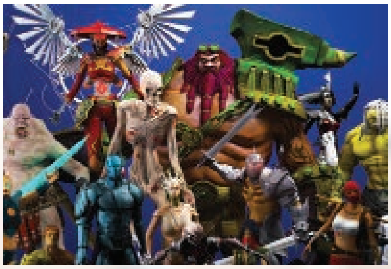
KNOWN AROUND THE WORLD.

Our newest international campuses are in Seattle, WA, and Lafayette, LA, and join our oldest ones in Sydney, Melbourne, and Canberra, Australia. We offer our students the opportunity to study abroad at any of our campuses.



"I can definitively say that AIE was the main factor that got my foot in the door and allowed me to build a career in games. More so than any university degree the AIE taught me the skills needed to make it in the industry that I love."

Andrew Hung, EA



"I just get so excited about our chosen career field that I leave the building to go home and kind of skip around, or grin like a fool...It's really amazing to think about how little I knew about this whole thing just a few months ago."

Katherine Walker, current AIE student

→ **ACADEMY OF INTERACTIVE ENTERTAINMENT**

Seattle, Lafayette, Canberra, Melbourne, Sydney
206.428.6350 - Seattle Campus
337.456.1848 - Lafayette Campus
www.theaie.us

ART SCHOOL

Still shot and character study from the 2012 senior capstone game project, *Water Aloft the Ridge*.

> *"The atmosphere of Columbia's Game Design major is what makes it unique. You're learning so many more skills than just what's taught in your concentration – through the variety of subjects offered, the interesting students that share your classes, the high-tech game lab, and, of course, the awesome teachers. The department feels like a second home to me."*
>
> **– Shimika West, Game Design, 2013**

Columbia
COLLEGE CHICAGO
INNOVATION IN THE VISUAL, PERFORMING, MEDIA, AND COMMUNICATION ARTS

→ **COLUMBIA COLLEGE CHICAGO**

600 S. Michigan Ave.
Chicago, IL 60605
312-369-7130
admissions@colum.edu

For more information about Columbia's Game Design Program, visit:
colum.edu/gamer

Game Design at Columbia College Chicago

Game Design is among the most rapidly evolving fields of our time. It is a dynamic and expansive discipline, producing some of the most popular and exciting forms of entertainment of the 21st century and on the leading edge of real-world practical applications in the fields of education, science, and business.

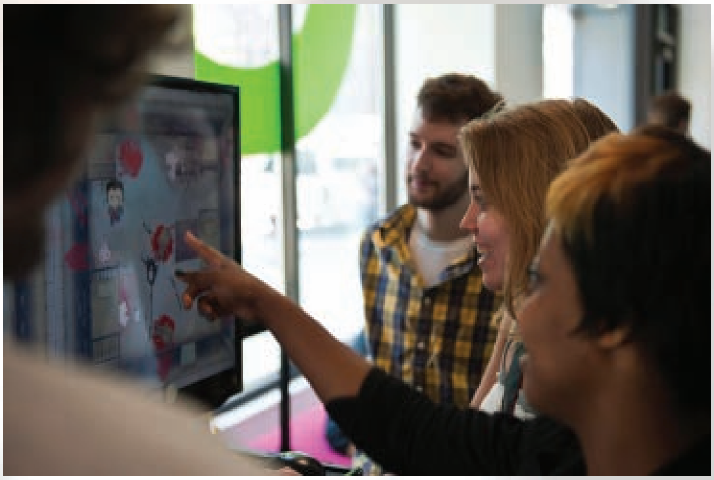
Columbia College Chicago's Game Design program is designed to mirror contemporary industry practices. Students learn how to create a shared vision and develop games in a strong, collaborative production environment. Game Design students explore the aesthetics and culture behind effective game design, game theory, and concept development. Columbia's state-of-the-art facilities, expert faculty who currently work in the field, strong cross-disciplinary partnerships across the college, and required Liberal Arts & Science core are essential elements of this creative learning environment and push our students to produce content that is both original and daring.

Columbia's curriculum offers both a core of Interactive Arts & Media classes as well as instruction in one of four Game Design concentrations (Game Art, Development, Programming, Sound) that provide the in-depth specialization that the game industry demands, and prepares them for careers in the many different professional disciplines found throughout the field. As majors study in their chosen concentration they are expected not only to broaden their understanding of the game design process and hone their skills, but also to build a body of work that leads to their final senior capstone experience—an advanced production class that brings together multiple teams of student specialists and culminates in the production of a working game.

To learn more about our program and to see student work, visit **colum.edu/gamer**

ABOUT COLUMBIA COLLEGE CHICAGO

Columbia College Chicago is the largest and most diverse private non-profit arts and media college in the nation. We offer a four-year liberal arts education specifically tailored for a community of highly motivated students who want to turn their creative talents into rewarding careers. Our student body is comprised of approximately 12,000 students who come from all 50 states and 41 foreign countries who we provide with the rigorous academics and unparalleled resources necessary to be successful in a highly competitive 21st century marketplace. The College is located in the Chicago's historic South Loop neighborhood, providing our students with a dynamic, eclectic urban environment in which to study and live. Convenient public transportation allows Columbia's faculty and students to utilize the whole city as a social, cultural, educational, and professional resource—effectively turning the entire city into our campus.



PRIVATE COLLEGE

DigiPen
INSTITUTE OF TECHNOLOGY

A LEADER IN GAME DEVELOPMENT EDUCATION

DigiPen students don't just learn about game development – they live it. Each year, students from across DigiPen's degree programs in computer science, game design, digital art, sound design, and computer engineering come together to create fully playable games that are innovative, polished, and, most importantly, fun. But their work doesn't end when they receive their grades – DigiPen students release their games to the public and continue to hone their projects based on player feedback. It's this passion, dedication, and drive to improve that has helped DigiPen students win more awards at the Independent Games Festival than any other school – and helped DigiPen alumni produce groundbreaking games like Portal and Portal 2.

CLASSROOM THEORY MEETS REAL-WORLD PRACTICE

Since it began offering the world's first four-year degree in game development, DigiPen's mission has been to produce graduates who are creators of technology, not simply users. The school's seven undergraduate and two graduate degree programs help students reach this goal by offering them a strong foundation in the academic fundamentals of their field, then challenging them to apply their knowledge in yearly team projects in a studio environment. In these projects, students create 2D and 3D games, soundtracks, animated films, and even game hardware – all from their own designs, and all from scratch. Visit www.digipen.edu/gallery to see examples of DigiPen student work.

LOCATED IN THE HEART OF THE GAME INDUSTRY

DigiPen's Redmond, Washington, location is an ideal place to study game development. With more than 350 interactive media companies in the Seattle metropolitan area, including industry giants like Microsoft Game Studios, Nintendo of America, and Valve Software, DigiPen students have ample opportunities to begin building their professional network from their first semester of classes onward. DigiPen hosts frequent guest lectures by developers and artists from local companies like Bungie, ArenaNet, and PopCap Games, giving students an insight into how game industry professionals solve real-world problems. DigiPen students also have the ability to intern for credit at local game studios, giving them an insider's perspective of the game industry – and, often, a foot in the door at their first job.

Screenshot from student game *Nitronic Rush*.

INDUSTRY PERSPECTIVES OF DIGIPEN:

"Over the years [Valve has] been in business, we've come to believe the best way to learn how to make games is to see how real players react when they're put in front of real games. To see that process being replicated inside an academic institution is really exciting. ... Our company today is stronger than it's ever been thanks, in part, to our DigiPen graduates."

Robin Walker, Creator of Team Fortress and Developer at Valve Software

"[DigiPen students] are able to hit the ground running. We can actually hire them as associate-level engineers, and they're already as experienced as our mid-level engineers are in a lot of ways. So it's a huge win for us."

Erik Gross, Core Technology Team Manager at Monolith Productions

→ **DIGIPEN INSTITUTE OF TECHNOLOGY**

9931 Willows Road NE
Redmond, WA 98052
866.478.5236
info@digipen.edu

www.digipen.edu

Top Undergrad Game Design Program
The Princeton Review
2012

→ **ENJMIN**

121 rue de Bordeaux
16000 Angoulême
France
Phone : +33(0)5 45 38 65 68
E-mail : contact@enjmin.fr
www.enjmin.fr

le cnam

The graduate school of games and interactive media enjmin

A WELL-KNOWN FRENCH SCHOOL

You may know Ecole Polytechnique (Engineering), ENA (Public Administration), ENST (Telecomunication), FEMIS (Cinema). Enjmin is «THE» school of games in France and one of the leading training structure in Europe.

OUR SCHOOL IS A PUBLIC ORGANIZATION

- **A public institution of higher education and research**
- **Postgraduate recognized** in the world of digital media
- Founded in **2005** as a department of the Cnam (National Conservatory of Arts and Crafts), one of the oldest public university in France
- **More than 40 awards and prizes** awarded to students for games developed as part of their training
- **Collaboration** with 8 universities and research centers **worldwide** and welcoming foreign students

MISSIONS

- Provide the digital industries with **trained professionals**, researchers and artists of the highest caliber
- Trained and support **research** in fields related to entertainment computing
- Provide **upskilling** to those already in the audiovisual sector, or **re-qualification** for those changing their career plans
- **Support students and alumnae** (application for start-ups grants, game production financing)
- Create an **international hub** for an industry learning and exchange network open to students, active games professionals and researchers

WHAT WE DO?

- **Master Degree in Games and Interactive Digital Media** accredited by the Cnam and the University of La Rochelle
- **Post-Master Degree in Interactive Digital Experiences** degree of Cnam Enjmin, in partnership with GOBELINS
- **Inter-company internships** short cycles from 5 to 15 days
- **Customized in-company trainings**
- **Accreditation of prior experiential learning** (VAE) : if you have more than 3 years professional experience within the degree framework, you can get the Master Degree

WHY JOIN OUR SCHOOL?

- Thanks to our **international network** we are able to give you a high level of education: professionals of game and interactive media, famous researchers, conferences with worldwide experts (Peter Molyneux from Lionhead Studio, Trip Hawkins founder of Electronic Arts, Gonzalo Frasca from Powerful Robot Games...).
- You will have the opportunity to get an **internship in famous companies**, such as Ubisoft (all over the world), Electronic Arts, Phoenix Interactive, Lexis Numerique, Darkworks...
- **Video game "gurus"** tutor our students to give them expert advice : Jordan Mechner (Prince of Persia), Eric Viennot (In Memoriam), Frédérick Raynal (Alone in the Dark), Serge Hascoët (Chief Creative Officer - Ubisoft), Viktor Antonov (The Building Studio).
- You will have all the material needed to create and **develop a pre-production of a game**, working in team of 8 or 9 students. Most of the time these projects are presented during conferences or events and obtain rewards : Puddle, Abstrakt, Attract Touch...
- Possibility to join a **PhD position** at the end of a master in one of the 2 laboratories associated with Enjmin (CEDRIC, L3i)

SCHOOL FEES

- **Master Degree in Games and Interactive Digital Media**
  - Students : € 450 per year (it includes the price of the training, French social security and all advantages of the student status in France)
  - Professionals : training = € 5,000 per year / registration fees = € 245
- **Post-Master Degree in Interactive Digital Experiences**
  - With an individual funding : € 8,000
  - With a firm funding : € 11,000
- **Accreditation of prior experiential learning** : € 2,740

Enjmin is in the creative media campus

Pôle Image Magelis Angoulême

FULL SAIL UNIVERSITY

With more than a decade of experience in video game education, Full Sail University – located in Winter Park, Florida – has been the launching point for many exciting career paths in the game industry. Full Sail's educational approach is centered around collaboration, immersive learning, and real-world experience, so it makes sense that its campus is equipped with the same kind of working environments found in today's game studios. These recently upgraded and expanded facilities are specifically designed for the needs of programmers, artists, and producers as they collaborate to build a playable video game.

THE GAME PRODUCTION STUDIO – WHERE PROJECTS COME TO LIFE

The Game Production Studio is where Game Development and Game Art students create and finish their final projects. The studio has areas specifically designed for audio, graphics, and technical development, as well as a game console timeline (complete with vintage hardware) and a VIP graffiti wall. At the center is an open workspace where students can work on completing their final projects together during class time as well as in their spare time. This large-scale studio also offers a private conference room for team meetings, plus comfortable seating areas to unwind after a long day of coding. In the studio's testing room, students are able to play their in-progress games, test for bugs, and of course, just play video games.

COLLABORATIVE SPACE IN THE GAME DESIGN AND DEVELOPMENT CENTER

The Game Design & Development Center offers a private conference room for team meetings, as well as comfortable seating areas to unwind after a long day of coding. Classic arcade cabinets are also scattered through the halls, giving students the chance to grab a quick game break between classes, or even pick up inspiration for their own projects.

DAVE ARNESON'S BLACKMOOR STUDIOS

Named for *Dungeons & Dragons* co-creator and former Game Development faculty member Dave Arneson, these studios commemorate Dave's contributions to Full Sail and to the world of gaming as a whole. The facility features an exhibit with memorabilia from his nearly four-decade career as a game developer – as well as eight game studio offices, four game development studios, two asset development rooms, a research room, and a conference room.

TRADITIONAL ART STUDIOS IN THE 3D ARTS BUILDING

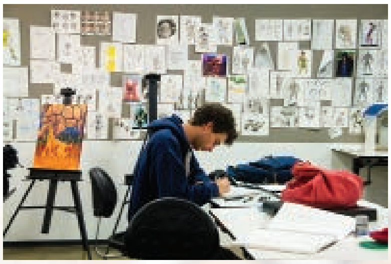
Since an understanding of art and animation fundamentals is key to a game artist's skill set, Full Sail Game Art and Computer Animation students have access to traditional art, 2D, and 3D animation studios in the 3D Arts building. The building also features student galleries that showcase models and other student artwork.

IMMERSIVE ONLINE ENVIRONMENTS

Through the Project LaunchBox™ program, every Full Sail student gets a laptop pre-loaded with industry software, allowing them to work and create from anywhere. Having a cohesive online platform means that campus programs are supplemented with dynamic, multimedia coursework, and online programs offer an exceptional level of collaboration and connectivity to instructors and classmates.

LEARN MORE

For more details on Full Sail's game degrees, including Game Art, Game Development, and Game Design, call **1.800.226.7625** or visit **www.fullsail.edu**. To view detailed information regarding tuition, student outcomes, and related statistics, please visit fullsail.edu/outcomes-and-statistics.



→ FULL SAIL UNIVERSITY

Full Sail University
3300 University Blvd
Winter Park, FL 32792
800.226.7625
fullsail.edu

View student gallery:
gameproject.fullsail.com/gpgames

ART SCHOOL

"The FuturePoly Professional Program is an invaluable experience, with a focus on education directly relevant to the game industry. Taught by industry vets, it was exactly what was needed to transform my art into a hire-able portfolio. I can't recommend FuturePoly enough to anyone who is thinking about breaking into the industry."

– Philip Oullette (graduate)

"No school can grant you what comes from hard work and practice, however, FuturePoly truly hands you all the ingredients you need to compete. Before the three month course was over I had a job at a AAA studio. If you're looking for a piece of paper, this school is not for you. If you're looking to be a professional video game artist, this school utterly eclipses all other options, and that I believe as firmly as I believe in gravity."

– Danny Weinbaum (graduate)

"My first college gave me a lot of knowledge, but ultimately left me unprepared for the real working world. FuturePoly's focused teaching was able to fill in the gaps and gave me the tools I needed to prove myself as a game artist. The instructors were able to show exactly what needs to be done to be competitive in this industry and motivate the students to push themselves regardless of their level."

– Kelly Shipman (graduate)

→ **FUTUREPOLY**

Email: info@futurepoly.com

www.futurepoly.com

Danny Weinbaum

FuturePoly is a digital arts training studio specifically geared toward video game development. At FuturePoly, every one of our instructors currently works in the video game industry and this guarantees that students are receiving the most up-to-date and relevant curriculum available. Basically you're not going to learn from some retired artist how they used to model barrels back in '92.

Our instructors know how to get a job in the industry and are here to share this knowledge with our hard-working and eager students. By keeping each class to 18 students or less, instructors are able to engage in one-on-one instruction. We believe the more a student puts in to the assignments the more they should get back. For example if a student comes in every class with new progress to show, then the instructor will continue to give suggestions until they believe it is ready for the students portfolio.

We know that as an artist, your portfolio is your degree. At FuturePoly, our courses are directly geared towards creating and fine tuning students' skill set and portfolio. An artists' ability cannot be displayed with a degree or resume. This is why we believe in having a clean, simple portfolio showcasing the exact skills the studio you are applying for needs. Because of this focus on building a professional portfolio, our courses also have an emphasis on career guidance. "Looking back on the last 12 years of my career, I've never once been asked what degree I have. Either your work looks good or it doesn't." – Jason Stokes, founder of Futurepoly.

Learning from industry professionals also provides students with a one-of-a-kind networking opportunity. If a student is excelling on all of the assignments and an instructor knows of an opening at Bungie, Valve, Arena Net, Microsoft, etc., it is a natural step to recommend that student.

FuturePoly provides four main core programs, 3D Modeling for Games, Digital Art for Games, Animation for Games, and ZBrush for Games. Each core program is three months long and offered year round. These main classes are split up between our Seattle and Bellevue campuses.

Other course offerings are available throughout the year and are meant to supplement and/or compliment the core programs. Our Intro to Game Art course is for those who are interested in pursuing this field, but might have little to no experience. We also hold workshops and other special events and are working to build our online program. Students can choose which programs they are interested in to register for, and can leave those that aren't a good fit behind. More information can be found on our courses page at www.futurepoly.com or by joining our mailing list.

Breaking into video games can be difficult and while there is no substitution for learning on the job, building a strong portfolio under the guidance of seasoned industry veterans is an advantage few other programs can offer. To find out more info, visit www.futurepoly.com or contact us at info@futurepoly.com.

Jason Stokes

{ ADVERTISEMENT }

PUBLIC UNIVERSITY

Digital Arts and Entertainment (DAE) is a three year full-time **Bachelor's degree program** taught in English at Howest University College in Kortrijk, Belgium. It consists out of **180 ECTS** points. Consequently, it is a European accredited and **internationally recognized** Bachelor's degree program. The curriculum provides a complete overview of the game development pipeline – from concept art to the final interactive game. We want students to feel, see, breathe, create, and ... be games. Students will soon find themselves making some cutting-edge, professional-quality content, ready to put in their portfolio!

UNIQUE PROFILE

Most students still graduate with a more traditional art or programming profile and this is where Digital Arts and Entertainment differs from all the rest. We compiled our degree with the input of the major players in the international games and interactive media industry. Together with these partners, we created a "technical artist profile" with an expertise in either 3D Arts or Game Development. In the course of three years, students learn all the skills, techniques, principles and insights to create complete interactive 3D environments. Our unique program contains both artistic 3D and technical programming courses and produces highly talented and wanted "technical artists", who are recruited throughout Europe and far beyond.

INDUSTRY-DRIVEN AND AWARD-WINNING

As mentioned above, DAE was the result of several years of national and international research. Our curriculum is taught by experts using state-of-the-art equipment and industry-standard practices. Students complete real-life and team-based, industry assessed projects. They enjoy guest lectures by key industry speakers, go on internships and experience a practical training. Our students enter worldwide competitions and show their work in the most renowned publications on digital art. We are proud to be the only university worldwide to have reached the Game Design finals of the global Microsoft Imagine Cup competition for five years in a row.

INTERNATIONAL FOCUS

As a DAE'er, the world is your playground. Right from the first semester, students are submerged in an international context, as applicants come from all over the world. It goes without saying that an exploration of the world's leading companies in the field of digital arts and entertainment, is part of the curriculum. Study trips in Europe and in the United States are a must. In the sixth semester, an internship at an internationally renowned company helps students become those sought after, top-of-the-list creative geniuses they always wanted to be.

FROM THE HEART OF EUROPE

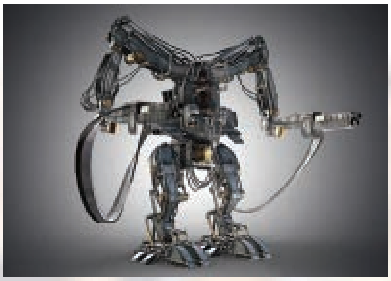
Digital Arts and Entertainment is a part of Howest University, located in Belgium, one of the countries with the highest standard of education in Europe.

Howest University has four campuses situated in Kortrijk and Bruges, one of the most beautiful cities in Europe. Walking along the maze of winding cobbled alleys and romantic canals, you imagine yourself to be in medieval times. What a source of inspiration!

Belgium with its rich cultural heritage, vibrant nightlife and artistic cities like Ghent, Antwerp and Brussels – the capital city of Europe – is also close to other famous European places such as Paris, London, Amsterdam, Berlin, Prague and Barcelona.

SUCCESS STORIES

We see our students working in a broad range of industries: game development, animation, VFX, media, architectural visualization, advertisement,... and holding a broad range of different positions: character modeler, environment modeler, rigging artist, level designer, 3D Artist, compositor, animator, 3D programmer, AI programmer, Technical artist, console programmer, Game tester, Game engine programmer, Post producer, Game designer, VFX artist, game developer,...They find employment or do their internship at companies like Electronic Arts, Crytek, Codemasters, Spicy Horse, Larian Games, Splash Damage, DICE, RockStarNorth, Guerrilla Games, PostPanic,...



"DAE is a high-quality education which, together with your determination and dedication, can help launch your career in the game development industry. I felt that its goal is not only to teach and hand out degrees, but also stimulate people in becoming better developers/artists, which is exactly what the industry needs."

Jeroen Maton (Graduate DAE)
Environment Artist - Dice - Sweden

howest
University College West Flanders
Member of the Ghent University Association

→ **DIGITAL ARTS AND ENTERTAINMENT AT HOWEST UNIVERSITY**

Graaf Karel de Goedelaan 5
8500 Kortrijk – Belgium – Europe
Phone: +32 56241264
Email: info@digitalartsandentertainment.com
www.digitalartsandentertainment.com

MASTER'S PROGRAM

"The MDM Program has the unique advantage of training students within the structure of a real-world production environment. It's education in a live-fire setting – a must for fostering the next generation of talent for the interactive entertainment industry."

Dave Fracchia, Vice President of Technology, Radical Entertainment

"In a live social game, our production cycle is all about speed, prototyping and rapid iteration. These are all skills I learned on group projects at Centre for Digital Media. There I was able to find my strengths and readied myself for the challenge of a fast paced, competitive space like Silicon Valley."

Jennifer Dowding, Associate Producer, Playdom, MDM graduate

→ **CENTRE FOR DIGITAL MEDIA**

577 Great Northern Way
Vancouver, BC V5T 1E1
Canada

SPEAK WITH AN ADVISOR:
Yasmeen Awadh
Senior Student Recruitment Coordinator
Tel: +1 778.370.1010
Toll free: 1.855.737.2666
Email: Yasmeen_awadh@gnwc.ca

VISIT US ONLINE AT: cdm.gnwc.ca

CENTRE FOR **DIGITAL MEDIA**

UBC SFU emily carr university of art + design BCIT

WHO WE ARE The Centre for Digital Media offers the Masters of Digital Media (MDM) and the Executive Masters of Digital Media (EMDM), both of which are offered as a collaborative effort between The University of British Columbia, Simon Fraser University, Emily Carr University of Art + Design, and the British Columbia Institute of Technology. The MDM and EMDM are full- and part-time professional graduate programs offering team-based learning in close collaboration with the digital media industry.

The programs are offered at the Centre's brand new, 49,000 sq. ft. state-of-the-art facility in Vancouver, British Columbia – a burgeoning digital media city that is home to major studios such as Electronic Arts, Microsoft, Relic, Rockstar and DeNA. The new facility, which opens September 2012, also houses Apartments @ Centre for Digital Media – flexible and affordable live/work spaces for students and visiting faculty.

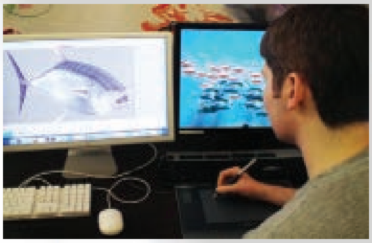
OUR PROGRAMS The MDM program offers a challenging and experiential 16-month program (including internship) that draws on students' imagination, skills and creativity. Guided by top-level faculty and industry mentors, students work closely with clients and peers on team-based, industry-supported projects. They learn how to communicate their ideas, and apply their knowledge and skills to produce results. The EMDM program features an identical curriculum, but is offered part-time, mostly online and is tailored for working professionals.

Over the course of three semesters, Masters of Digital Media students will learn the foundations of digital media from business, creative and production standpoints. Working in close-knit project teams throughout this period, students will also collaborate on real-world projects for key industry partners, learning asset management, leadership, and interpersonal communication skills in the process. Past industry clients have included Microsoft, Zynga, Electronic Arts, Mozilla and many others. Following these three semesters of course and project work, students embark on a four-month internship with a leading organization in a range of sectors to hone their newly-acquired skills in a real work environment.

Housing and cash scholarships are available for students who have demonstrated exceptional portfolios.

WHY STUDY AT THE CENTRE FOR DIGITAL MEDIA? Motivated students from a range of backgrounds choose the MDM and EMDM programs because they have a passion for digital media and a vision for how to turn this passion into something great. The program helps them achieve this by providing a learning environment that blends bold experimentation with real world experience. It also allows them to build their business network and discover new opportunities. Graduates have the know-how and confidence to work at the highest level in the industry as creators, producers, innovators, or entrepreneurs.

Students graduate from the Centre for Digital Media with a master's degree from Canada's first professional graduate program in digital media. Degrees also bear the seals of four of the top academic institutions in Western Canada. Alumni from the program have gone on to form successful start-ups, teach at the post-secondary level, and work for large, well-respected organizations such as Google, Disney Playdom, Microsoft and Electronic Arts.



For further information visit **http://cdm.gnwc.ca**

WORLD-CLASS EDUCATION IN A FULLY IMMERSIVE ENVIRONMENT

Media Design School is New Zealand's most-awarded tertiary institute for digital and creative technology qualifications.

The Auckland-based school offers degrees and diploma qualifications in 3D computer animation and visual effects, game development, media design, creative advertising and digital creativity foundation studies.

The School is highly regarded for its cutting-edge courses in digital design, which prepare students to be outstanding contributors to the creative industry by encouraging professionalism and pushing creative boundaries.

Programmes are developed in collaboration with industry, giving students unparalleled project experience using industry-standard software in a studio environment. The School also hosts several high-profile guests throughout the year to present lectures and workshops, give students live industry briefs or provide feedback on work.

Media Design School offers pathways into future education with foundation courses and diploma programmes providing an opportunity for future learning across a range of disciplines.

The School is part of the Laureate International Universities, the world's largest private university group with including 60 universities in 29 countries, including dedicated design schools in San Diego, California, and Santa Fe, New Mexico, USA; Berlin and Milan, Italy.

GAME ART AND PROGRAMMING DEGREES

Media Design School offers New Zealand's only degrees specialising in game development — Bachelor of Creative Technologies (Game Art) and Bachelor of Software Engineering (Game Programming).

The comprehensive three-year programmes have been developed in close collaboration with local and international game studios to ensure graduates are primed to enter the industry. Intense project-based learning replicates the studio environment, giving students first-hand experience of game production processes.

Media Design School CEO Frances Valintine said offering world-class degrees in this high growth digital sector was a natural and exciting progression.

"Global demand for game content across all platforms — console, mobile, PC and online — continues to grow and the projected future demand for game programmers and game artists exceeds the level of qualified graduates who enter the sector each year," said Valintine.

CURRENT QUALIFICATIONS

BACHELOR DEGREES: Bachelor of Art and Design (3D Animation and Visual Effects); Bachelor of Creative Technologies (Game Art); and Bachelor of Software Engineering (Game Programming).

DIPLOMAS: Diploma of Digital Creativity; Diploma of Graphic Design; Diploma of Creative Advertising; Diploma of Digital Media.

GRADUATE QUALIFICATIONS: Graduate Certificate of Creative Technologies, and Graduate Diploma of Creative Technologies.

For more information visit: www.mediadesignschool.com

Tim Shepherd, Media Design School '12

> *"Media Design School played a huge part in my progress with an excellent set of staff, tutors and the team-based learning. Even after graduating Media Design School helped a great deal by sending out local job openings when they come up. But when it comes down to it, I'd have to say the tutors are why Media Design School is such a great place to learn 3D."*

Thomas Hobman,
Motion Picture Company London

> *"Working at Weta for over 10 years, I've seen what our film industry can do. I've been lucky enough to work alongside some of the most ridiculously talented people in film. But it's an even rarer treat to see up-and-coming film artists from Media Design School showing what the future of visual effects in New Zealand holds."*

Greg Broadmore – Weta Workshop
(Art Department *The Adventures of Tintin*, *District 9*, Special Effects *Avatar*, *King Kong* and *The Chronicles of Narnia*.)

→ MEDIA DESIGN SCHOOL

Media Design Centre
92 Albert Street
Auckland 1010
New Zealand
Phone +64 9 303 0402
Skype media.design.school
Email admissions@mediadesignschool.com
Facebook www.facebook.com/mediadesignschool
Twitter www.twitter.com/mdsnz
Vimeo http://vimeo.com/user743798

www.mediadesignschool.com

ART SCHOOL

"SCAD offers a balance of programming and design, so I can stay true to my aesthetic while I work on my programming. I feel well-rounded with multi-faceted skills to take the industry head-on."

Whitney Taylor (SCAD M.F.A. graduate, 2011), User Experience Designer, Zynga

"The time I spent at SCAD provided the training and experience I needed to build a solid foundation for a successful career within the gaming industry. I utilize the wide variety of knowledge and skills my professors provided me on a daily basis to excel in a career that I am very passionate about."

Jimmy Barnett, (SCAD B.F.A. graduate, 2009), Environment Artist, Gearbox Software

"I recently gave a presentation at SCAD's 2011 GDX. Afterward I was able to speak to the students and tour the SCAD facilities. I have to say that I was not only blown away by the talent the students showed but also by the amount of resources that were made available in the facilities. Anyone thinking of getting an education in a creative field should definitely check out SCAD. What they offer is pretty incredible."

Lina Halper, Senior Programmer, Epic Games

→ **SAVANNAH COLLEGE OF ART AND DESIGN**

P.O. Box 2072
Savannah, GA 31402-2072
Phone: 912.525.5100 or 800.869.7223
Email: admission@scad.edu
scad.edu/interactive

SCAD®

The University for Creative Careers®

AWARD-WINNING GAME DESIGN EDUCATION IN A WORLD-CLASS SETTING.
The SCAD interactive design and game development graduate program was ranked in the Top 10 in the United States by the *Princeton Review* in 2011. Each year, SCAD brings together game developers, educators and students with experts who share their behind-the-scenes knowledge at the Game Developers eXchange. According to the Los Angeles Times, the SCAD interactive design and game development program ranks among the Top 10 in the United States mentioned by industry recruiters.

SCAD IS THE UNIVERSITY FOR CREATIVE CAREERS IN GAME DEVELOPMENT.
With triple-threat credentials as artists, designers and programmers, SCAD graduates are in demand. Annually, more than 500 national and international companies and organizations recruit SCAD students and alumni. Employers of SCAD graduates include Apple, Blizzard Games, Disney Interactive Studios, Electronic Arts, Firaxis Games, Red Storm Entertainment, Rockstar Games, Snowblind Studios, Sony Pictures Imageworks, Vicarious Visions and Vivendi Games.

DIVERSE DEGREES IN A DYNAMIC LEARNING ENVIRONMENT.
Students can pursue either undergraduate or graduate degrees through SCAD's B.A., B.F.A., M.A. and M.F.A. programs in interactive design and game development. The department is available at each of SCAD's four locations: in Atlanta and Savannah, Georgia; in Hong Kong; and online via innovative eLearning. Students also have the opportunity to immerse themselves in the culture of Provence at SCAD's study-abroad location in Lacoste, France. Internships, collaborative projects, personal attention from acclaimed professors and interaction with iconic professionals all contribute to the unique learning environment creative students need to thrive.

AN ADAPTABLE PROGRAM RELEVANT TO TODAY'S RAPIDLY EVOLVING GAME AND DESIGN CULTURE. SCAD students of interactive design and game development fuse artistic and technical knowledge to design and develop digital and non-digital games, rich-media Internet and mobile applications, virtual and physical computing environments, and emerging forms of human/computer interaction. The two distinct tracks of the program—interactive design, game development—enable student to pursue their passions in the context of a holistic education in game art and interactive media.

ACCESS TO ADVANCED TECHNOLOGY AND PROFESSIONAL-LEVEL EQUIPMENT.
SCAD students use the same high-end hardware and software used throughout the profession: a full motion capture lab, green screen and Wacom Cintiq displays; as well as Autodesk Maya, Autodesk Mudbox, Unity 3D, CryEngine, the Unreal Engine, ZBrush, 3ds Max; and Adobe Photshop, Illustrator, Premiere, After Effects, Flash, C++ and ActionScript. In addition to SCAD's annual Game Developers eXchange, the department participates in SIGGRAPH, GDC, Austin GDC and SXSW, among other industry-specific conferences, garnering students valuable connections in the industry.

Learn Game Design at the World's Leading School for Animation, Gaming and Design*

CANADA'S LARGEST ARTS SCHOOL

Founded in 1967, Sheridan College's Faculty of Animation, Arts and Design is the largest arts school in Canada. The wide range of programming offered – from animation to gaming, illustration to music theatre, photography to crafts and design – results in a dynamic and stimulating learning environment.

A TRADITION OF CREATIVITY AND INNOVATION

Sheridan is renowned internationally for its animation programs, which have been producing top talent for the film and television industries for over 40 years. Our graduates include Oscar nominees and winners, who have helped fuel the surge in popularity of animated films by studios including Pixar, Dreamworks and Nelvana. Six of the films nominated for 2012 Academy Awards included 43 Sheridan graduates on their creative and production teams.

GET YOUR GAME ON AT SHERIDAN

Sheridan's new gaming programs build directly on its success and expertise in the animation field. The Game Level Design certificate is a one-year, postgraduate program intended for students with a bachelor's degree or college diploma in a visual art or design-related discipline. It prepares students with the skills to bring together all the different aspects of the game – narrative, characters, rules and mechanics – to create the virtual environment in which all of these elements play out.

The curriculum focuses on applied, real-world skill development. Assignments and class work focus on case studies, simulations and lab projects. Students learn both 2D and 3D game design, while gaining industry-specific skills in:

- Story and scripting
- Cinematography and sound
- Multi-player systems
- Game documentation
- Play testing
- Project management

A unique feature of the Game Level Design program is its collaborative relationship with the Game Development – Advanced Programming graduate certificate program offered by Sheridan. Together, they offer students the unique opportunity to experience a realistic industry environment, with designers and programmers working together to create a video game in the third semester of their one-year program.

Launching in the Fall of 2013, Sheridan's Bachelor of Applied Arts in Game Design is a four-year degree that will offer a comprehensive grounding in all aspects of game development, including:

- Game design and art
- Narrative and script development
- 2D and 3D Animation and Rendering
- Programming and Technology
- Artificial intelligence
- Psychology
- Marketing and distribution
- Development and project management

Students will learn the complete life cycle of game design, from idea through development, production and marketing. A 14-week practicum after Year 3 provides real-world job experience, while 4th year students will design and produce a game with a team of collaborators, simulating the project team setting found in industry.

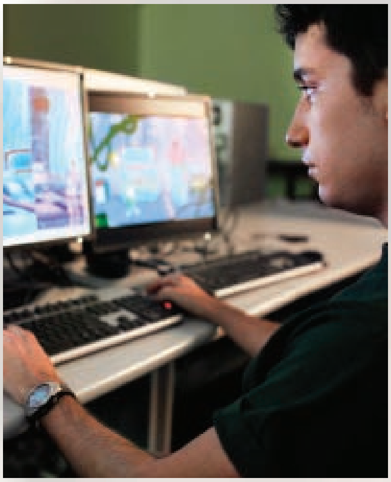
Graduates will also be qualified for graduate studies in game design.

*** Animation Career Review has named Sheridan College number one in the world for animation, gaming and design following a comprehensive analysis of school reviews, industry-reviewed lists like the Princeton Review, and through a survey of animation industry professionals and recruiters from companies like Pixar and DreamWorks.**

Sheridan

→ **SHERIDAN COLLEGE INSTITUTE OF TECHNOLOGY AND ADVANCED LEARNING**

PO Box 2500, Stn Main
Oakville, Ontario, Canada L6L 7T7
Phone: 905-845-9430
Email: infosheridan@sheridaninstitute.ca
http://gaming.sheridancollege.ca



With your creativity and our technology, you can develop the talents you need to launch a future in game design.

Learn to capture motion—and unleash your creativity on the world of gaming as you prepare for exciting career opportunities.

→ **THE ART INSTITUTES**

Administrative Office
210 Sixth Avenue, 33rd floor
Pittsburgh, PA 15222
1.800.894.5793
Gamer.aii.edu

Ai The Art Institutes®

CREATE TOMORROW

With the right education, imagine what you could create.

Explore the Game Art & Design programs at The Art Institutes schools.

AN OPPORTUNITY TO BRING YOUR IDEAS TO LIFE.

Creative people are pushing the gaming industry forward with new technologies, new characters, and new gaming experiences. If you want to join them, the first step is the right creative education. Through the Game Art & Design programs at The Art Institutes schools, we can help you develop and nurture the talents you need to go from game player to gaming professional.

It's a practical, career-focused education that's focused on the work, the students who create it, and the instructors who guide them every step of the way.

A COLLABORATIVE, SUPPORTIVE LEARNING ENVIRONMENT.

In our Game Art & Design programs, you'll find yourself in a creative community, working with other students who share your passion. You'll use professional-grade technology. And you'll be guided by talented, experienced instructors who know where the industry is, where it's heading, and what skills you'll need to meet it head-on.

Once you've completed the fundamental courses, you'll start to build skills in drawing, color, design, computer applications, and creating lifelike characters. You'll explore image manipulation, cinematography, creative storytelling, storyboarding, and 2D and 3D modeling techniques. And you'll use hardware including Apple and HP workstations; game engines such as Microsoft XNA, Unity 3D, and Unreal SDK; Adobe Master Collection, including Flash Professional, Photoshop, and After Effects; and Autodesk Entertainment Creation Suite, which includes Image Studio, Maya, and MotionBuilder. Technology may vary by curriculum and/or school.

START OUT WITH A FOCUSED EDUCATION. REAL-WORLD EXPERIENCE. AND A PROFESSIONAL PORTFOLIO.

As you get ready to enter the job market—and join our alums who are out there bringing their ideas to life—we'll help you assemble a digital portfolio to showcase your talents, both in individual interviews and at portfolio shows, where you can show your work to companies that are familiar with our program and our graduates. It's all about getting your foot in the door with an entry-level job such as game tester/analyst, game designer, level designer, texture artist, cinematic artist, 2D artist, or 3D artist, working in a software, game design, or education company.

TALK TO US TODAY.

Contact us and we'll help you get started toward a career in game art & design. And launch a future where you can create your own tomorrow.

The Art Institutes is a system of over 50 schools throughout North America. Financial aid is available to those who qualify. Program offerings and credential levels vary by school.

See **aiprograms.info** for program duration, tuition, fees, and other costs, median debt, federal salary data, alumni success, and other important info. The Art Institutes is a system of over 50 schools throughout North America. Programs, credential levels, technology, and scheduling options vary by school, and employment opportunities are not guaranteed. Financial aid is available to those who qualify. Several institutions included in The Art Institutes system are campuses of South University or Argosy University. OH Registration # 04-01-1698B; AC0165, AC0080; Licensed by the Florida Commission for Independent Education, License No. 1287, 3427, 3110, 2581. Certified by SCHEV to operate in Virginia. Administrative office: 210 Sixth Avenue, 33rd Floor, Pittsburgh, PA 15222. ©2012 The Art Institutes International LLC.

GRADUATE SCHOOL

Florida Interactive Entertainment Academy

Part of the University of Central Florida, the **Florida Interactive Entertainment Academy (FIEA)** offers a graduate gaming education in a world-class facility in downtown Orlando. It teaches artists, programmers, and producers the techniques, tools and skills that you don't get in a traditional degree program.

You'll be exposed to an industry-based curriculum that mimics the production environments of successful gaming studios. You'll work on student teams with real-world projects to learn the skills and software of video-game design and development.

In 16 months, you'll earn a Master's of Science in Interactive Entertainment.

Five Reasons to Choose FIEA

FACULTY

Our faculty are industry veterans with real-life answers. Collectively, FIEA faculty have shipped more than 45 of today's biggest games and films and have worked in studios like EA, Sony, Disney, Microsoft, High Moon Studios and Take Two. And our 10-to-1 student-to-faculty ratio ensures you'll get the attention you need.

INDUSTRY SUCCESS

Bioware. Google. Zynga. LucasArts. Ubisoft. Telltale Games. Neversoft. Cartoon Network. YouTube. Bethesda. Electronic Arts. FIEA grads are everywhere making games and building their careers.

Recent games FIEA grads have worked on include Call of Duty Black Ops & MW3, The Elder Scrolls V: Skyrim, Madden NFL Football 13, Eve Online, NBA Live 13, Words With Friends, Star Wars: The Old Republic, God of War: Ascension, Disney Epic Mickey 2 and Dead Space 3.

MASTER'S DEGREE

Unlike some schools, FIEA offers a fully accredited Master's of Science in Interactive Entertainment from the University of Central Florida. So you not only get a cutting-edge education but also a degree that's recognized and a valuable marketplace commodity. And you get it in only 16 months.

THREE TRACKS

Our production, programming and art tracks teach you all you need to know to become industry ready. As you specialize, you'll go deep into your chosen discipline to learn all the techniques and tools used to make games and films. But because we never offer two classes at the same time, you can also "minor" in one of the other specialties to learn valuable additional skills.

You'll apply these skills in our team-based curriculum. From the day you enter, you'll be put on a game team with other artists, programmers and producers and be required to display the communication and commitment necessary to bring a game to fruition.

THE SPACE

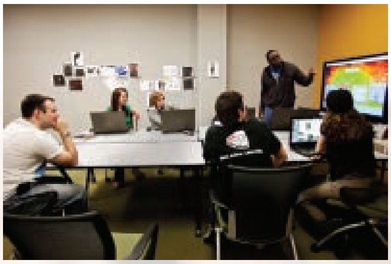
FIEA is located in UCF's Center For Emerging Media where FIEA students collaborate with student musicians, filmmakers, architects and animators on games and other interactive projects. The 114,000 sq. ft. building features a soundstage and motion capture facility that host student and industry shoots.

Learn more at **www.fiea.ucf.edu**

At 3,500 sq. ft., FIEA's Mocap Studio is one of the largest on the East Coast.

> *"I can say without hesitation that FIEA has been the best decision I've made for pursuing my career as a video game artist."*
>
> **Ingrid Aguero, Environment Artist**
> **Electronic Arts, Tiger Woods PGA TOUR 10, 11, 12, 13, Madden NFL 11, PGA TOUR Golf Challenge, NBA LIVE 13**

FIEA games have won awards from MTV, EA, IGF and Shockwave.



Artists, programmers and producers working together is the heart of a FIEA education.

In 2011 and 2012, *The Princeton Review* ranked FIEA as one of the top 4 video game design programs in the world.

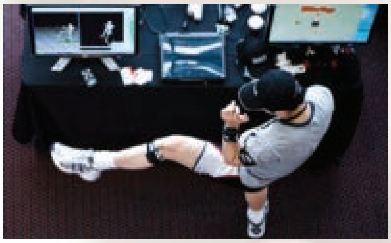
UCF

Center for Emerging Media

FIEA Florida Interactive Entertainment Academy

→ **FLORIDA INTERACTIVE ENTERTAINMENT ACADEMY**

Center for Emerging Media
500 West Livingston St., Orlando, FL 32801
407-823-2121 • info@fiea.ucf.edu
facebook.com/UCFFIEA • twitter.com/FIEA
www.fiea.ucf.edu



“Coming in from VFS, I was ready to hit the ground running. VFS prepared me very well for the volume and type of work that I do, and to produce the kind of gameplay that I can be proud of.”

David Bowring, VFS Game Design Graduate
Gameplay Designer, *Saints Row: The Third*

“VFS was really instrumental in me being successful today.”

Armando Troisi, VFS Game Design Graduate
Narrative Director, *Halo 4*

VANCOUVER FILM SCHOOL
Results Matter

VANCOUVER FILM SCHOOL

198 West Hastings St.
Vancouver, BC V6B 1H2
Canada
Phone: 604.685.5808 or 800.661.4101
inquiries@vfs.com
vfs.com/gamecareer

Vancouver Film School

The industry is in our DNA. Everything about the one-year Game Design program at Vancouver Film School is built to closely mirror the real world of game development. That means every student creates games for one intense year, lives a real-world design and production pipeline, and graduates with a powerful loadout of skills and training.

There's a reason the *LA Times* called VFS one of the top 10 schools favored by game industry recruiters and VFS was the only one-year program on the *Princeton Review*'s 2012 list of Top Game Design Programs.

Whether you want to contribute to massive AAA titles or make groundbreaking indie games, at VFS you learn everything you need, including:

- > Interactive Narrative
- > Interface Design
- > Scripting
- > Level Design
- > Preproduction
- > Project Management
- > Flash
- > Mobile & Handheld Design
- > Game Audio
- > Analog Games
- > The Business of Games

LED BY INDUSTRY

In VFS Game Design, you're mentored by a faculty of respected game industry professionals – men and women who've spent their careers in game development and who become your first crucial connections to the professional world. At the helm is Dave Warfield, an industry veteran who helped produce and design EA's *NHL* franchise for 10 years, among many other credits. The program's curriculum is developed and kept current in conjunction with an Advisory Board of game luminaries from companies like Activision, Microsoft, LucasArts, and Nokia.

A STUDIO ENVIRONMENT

Your year at VFS is modeled after a real-world studio environment and production pipeline. Want to break into the industry? Any pro will tell you to do the same thing: *make games*. That's exactly what you'll do at VFS. You'll work in teams to take your games from concept to completion, then present them to an audience of industry reps and recruiters. It'll be your first job interview.

LIVING & CREATING IN VANCOUVER

There are few better places in the world to learn about game design than Vancouver, BC, Canada. It's a hub of development, full of leading developers and cutting-edge independent studios. VFS Game Design is right at its heart. Vancouver is also widely considered to be one of the best places in the world to live: a beautiful and cosmopolitan city that just so happens to be the perfect place to launch your game design career.

THE RESULTS

VFS Game Design graduates have gone on to key design and production roles around the world. A small selection of their recent and upcoming credits includes *Halo 4*, *Prototype 2*, *Star Wars: The Old Republic*, *SSX*, *Max Payne 3*, *Sleeping Dogs*, and *Mass Effect 3*.

gd
HAUNT
GAME DEVELOPER MAGAZINE
FEBRUARY 2012 INSIDE
10 YEARS OF SALARY SURVEYS
DEUS EX HUMAN REVOLUTION
INDIANA JON

ADVERTISER INDEX

gd Game Developer (ISSN 1073-922X) is published monthly by UBM LLC, 303 Second Street, Suite 900 South, South Tower, San Francisco, CA 94107, (415) 947-6000. Please direct advertising and editorial inquiries to this address. Canadian Registered for GST as UBM LLC, GST No. R13288078, Customer No. 2116057, Agreement No. 40011901. ***SUBSCRIPTION RATES:*** Subscription rate for the U.S. is $49.95 for twelve issues. Countries outside the U.S. must be prepaid in U.S. funds drawn on a U.S. bank or via credit card. Canada/Mexico: $59.95; all other countries: $69.95 (issues shipped via air delivery). Periodical postage paid at San Francisco, CA and additional mailing offices. ***POSTMASTER:*** Send address changes to Game Developer, P.O. Box 1274, Skokie, IL 60076-8274. ***CUSTOMER SERVICE:*** For subscription orders and changes of address, call toll-free in the U.S. (800) 250-2429 or fax (847) 647-5972. All other countries call (1) (847) 647-5928 or fax (1) (847) 647-5972. Send payments to *gd Game Developer*, P.O. Box 1274, Skokie, IL 60076-8274. Call toll-free in the U.S./Canada (800) 444-4881 or fax (785) 838-7566. All other countries call (1) (785) 841-1631 or fax (1) (785) 841-2624. Please remember to indicate *gd Game Developer* on any correspondence. All content, copyright *gd Game Developer* magazine/UBM LLC, unless otherwise indicated. Don't steal any of it.

JOB INTERVIEW DON'TS

SITUATIONS TO AVOID WHEN INTERVIEWING AT A GAME STUDIO

Congratulations! You've gotten the call back, and now you're headed to an in-person interview for your dream job at (insert your favorite developer studio here)!

What should you say? How should you act? Well, nobody can tell you exactly what to do, other than to relax and be your awesome self. But why don't we talk about what not to do? (That's more fun.) Here are some things you should not do when going to your big interview.

Drink Too Much Caffeine

» The appointment is probably early in the morning, so it's natural to want to ensure you're as bright and alert as can be. But don't go overboard on the espresso shots and Red Bulls before you start. Otherwise, you'll vibrate as you talk, giving everything you say a not-very-confident-sounding warble. You'll also probably have to pee really badly just as someone starts asking you about fluid physics.

Exaggerate Your Previous Role

» Being confident in your abilities means accurately describing what you did on past projects without resorting to hyperbole. If you think you need to oversell your experience, then you're probably not applying for the right job. Also, your interviewers can tell when you do it, because it tends to sound like this:

"Would you describe what you did at your last job?"

"I basically designed and programmed World of Warcraft."

"Oh, okay. It doesn't say 'Blizzard' on your resume...."

"Yeah, I was actually a contractor for Blizzard's photocopier service company at the time. They took all my work and stole all the credit from me. I'm over it now, though."

And who knows—maybe that's totally true! Just be aware of how that sounds to the people tasked with judging your truthfulness. Plus, if that really did happen, you'd probably make more money selling your story to 60 Minutes or Kotaku.

Tell Them How to Do Their Jobs

» "How can we improve on our previous game [x]?" is a common question. Tread carefully here: Obviously, you want to work at a studio because you want to make their products better, and that's great. But it doesn't mean that you already understand all of the factors and decisions that went into making the game the way it was. If you're asked for criticism, be constructive.

ILLUSTRATION BY CHARLIE MOK

Don't tell BioWare, "I can't believe you guys decided to end Mass Effect 3 the way you did! Let's face it: That was completely terrible. No offense—just saying you pretty much need to hire me, 'cause I wouldn't have let that happen."

That won't fly in the interview room. That does work as a forum post, though, so if that's all you're after, flame away!

Have "No weakness"

» Another talking point that often comes up is your self-assessment of strengths and weaknesses. Strengths are usually pretty easy to talk about—you can always mention how good-looking you are, for example. The weaknesses part is the trickier one, because you're being asked to willfully mention something that could be negative about yourself. Pick carefully. The following weaknesses never work, so don't try them:

- You are so amazing that you make others around you feel bad.
- You just don't know when to stop working. If it were up to you, you'd work all day! For free!
- Your weakness is that you have no weakness.

Be Creepy

» Sometimes studios want to talk about your "passion" for games, just to make sure you're really dedicated to this medium and won't mind the adverse situations that come up from time to time in the course of the job.

Passion is great, but don't go overboard. When asked if you're passionate about the games you play, definitely say yes—but don't be the one talking about how you were once so upset by a balance change in Call of Duty: Black Ops that you drove two hours to Treyarch's offices in Santa Monica and screamed incoherently in the parking lot until security escorted you off the premises.

I mean, that's a fun story and all, but save it for later. At least until after you've been hired by Treyarch.

Bring-a-Pie Bribe

» You don't need to bring gifts or any other kind of bribe to your interview. Especially not a home-baked pie. Pie-bearing candidates are clearly aiming to please just a little too hard. There's really no world where providing a pie at the interview will result in anything but people feeling a little put off. Well, unless the game in development actually does center on pies—then I suppose it could be appropriate.

Also, if you do bring one, do not bake cash or illegal drugs into the pie.

Forget to Wear Pants

» Put a little sticky note on the inside of your door the night before the interview: "ARE YOU WEARING ALL OF YOUR CLOTHES?"

I'm not saying you're going to forget. You just want to be extra sure, that's all. You don't want to be like that "friend" of mine.

MATTHEW WASTELAND *writes about games and game development on his blog, Magical Wasteland (www.magicalwasteland.com). email him at mwasteland@gdmag.com.*

UNREAL ENGINE NEWS

SUDA 51 OFFERS NEW TAKE ON ZOMBIE SHOOTERS WITH UNREAL ENGINE 3

In the video game business, there are few creators who more routinely conjure completely original action adventures than Japanese game designer Goichi "Suda 51" Suda. With *Lollipop Chainsaw*, the game maker and his team at Grasshopper Manufacture have introduced a new heroine to gaming lore. Juliet Starling is a high school cheerleader who has a secret life as a zombie killer. When the undead invade her San Romero High School, it's time to replace those pompoms with a chainsaw and assorted weapons to take out the monsters.

"We wanted to create a fun and exhilarating zombie action game that brings the concept of 'pop, cute and violent' to gamers," said Suda 51.

Suda 51 is delivering an extremely different action-based interactive creature feature this time around, showcasing just how much breadth of scope is available to Unreal Engine 3 (UE3) developers. *Lollipop Chainsaw* comes on the heels of Grasshopper's UE3-powered *Shadows of the Damned*, published by Electronic Arts. And the two games couldn't be more different.

Although both games share plenty of blood and gore, *Lollipop Chainsaw* takes an over-the-top '80s approach to its third-person perspective action. Starlet wears her Knights cheerleader uniform with pride but delivers deadly combos with accuracy when armed and dangerous.

"Some of the basic methods for taking out enemies are chainsaw attacks and cheerleading attacks," said Masahiro Yuki, co-producer on *Lollipop Chainsaw*. "How those attacks can be mixed to slash zombies is entirely up to the player's creativity. We implemented lots of combination attacks in the game so that users can take full advantage of the variety of moves, such as the chainsaw dash and chainsaw bluster, while strategizing their game plan."

The technology behind this mayhem benefited from the developer's multiple UE3 titles.

When it came to choosing UE3 for its games, Tomas Rovina Roquero, technical art director at Grasshopper Manufacture, said the company plans to maintain technologies and know-how that they have built upon over multiple projects. "The provided support system that Epic Games offers was enticing enough to choose Unreal Engine 3 for our development," he added.

Nobuhiro Obata, a programmer at Grasshopper Manufacture, said the Unreal Kismet visual scripting system is used by the team in many ways. "We were able to create an open development environment where staff members from different divisions were able to easily participate in the game's development," he said.

Unreal Cascade and post-processing functions allowed the team to create cartoonish and cute, yet violent effects and expressions for Juliet Starling.

"We were able to flexibly implement halftone and 'Sparkle Hunting' in the post-processing, which flawlessly depict the interpretation of the world of *Lollipop Chainsaw*," explained Obata. "Our game planners and artists were pleased that we met this requirement."

Roquero said the team took advantage of Epic's Unreal Developer Network (UDN) throughout the development process. In fact, by the end of the game's development they had exchanged more than 350 emails through the UDN site, including many questions regarding new functionality that Grasshopper created for this game.

The end result is a zombie game unlike anything out there, and yet another example of the creativity that comes from Suda 51 and Grasshopper Manufacture. With each new game from this inventive Japanese studio, the capabilities of the Unreal Engine continue to expand.

*Canadian-born **Mark Rein** is co-founder and vice president of Epic Games, based in Cary, NC. Unreal Engine 3 has won eight Game Developer magazine Front Line Awards, including entry into the Hall of Fame. Unreal Engine 3 has also received four consecutive Develop Industry Excellence Awards. Epic is the developer of the bestselling Unreal, Gears of War and Infinity Blade series of games. Follow @MarkRein and @UnrealEngine on Twitter.*

WWW.UNREAL.COM

UPCOMING EPIC ATTENDED EVENTS

Develop
Brighton, UK
July 10-12, 2012

GDC Europe
Cologne, Germany
August 13-15, 2012

gamescom
Cologne, Germany
August 15-19, 2012

Please email **licensing@epicgames.com** for appointments.

For recruitment opportunities, please visit **www.epicgames.com/careers**.