TIPS FOR BUILDING AN INDIE STUDIO

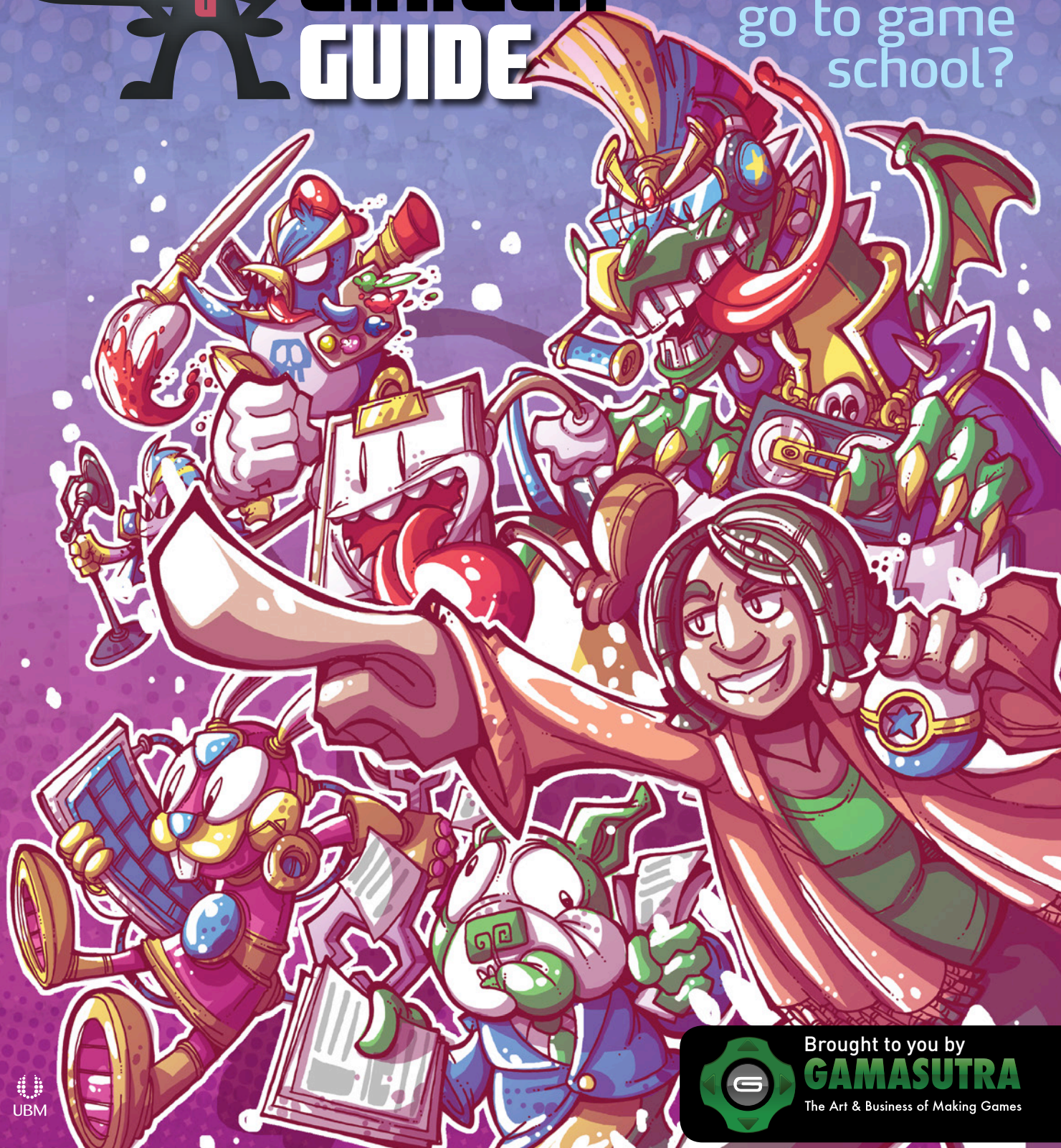
JUNE 2015

GAME DEVELOPER MAGAZINE

GAME CAREER GUIDE

START MAKING GAMES TODAY!

Should you go to game school?



Brought to you by GAMASUTRA
The Art & Business of Making Games

UBM

LCAD.EDU

Matthew McKeown (Junior, Game Art BFA with 3D Character Emphasis)
Orc Warrior, Digital, 2015

ART OF GAME DESIGN MFA

LAGUNA COLLEGE OF ART + DESIGN

Join the next generation of game designers, developers and entrepreneurs in advancing the art, science and culture of play.

"LCAD's Game Art program has a great reputation for producing top quality artists. Their Art of Game Design MFA empowers artists and programmers to take their skills to the next level."

unity

JUNE 2015

GAME CAREER GUIDE

CONTENTS

DEPARTMENTS

STUDENT POSTMORTEM_Justin Ng, Dexter Chng, Bryan Teo, Soo Zhong Min

FEATURES

WELCOME TO THE GAME CAREER GUIDE 2015!

VIDEO GAMES ARE GREAT. YOU KNOW THAT, AND THAT'S WHY YOU'RE READING THIS! But while the industry has been opening up to more people, and tools are now more accessible than ever, making good video games is still very hard. And that is precisely why you should make them. As Theodore Roosevelt said, "Nothing in the world is worth having or worth doing unless it means effort, pain, difficulty... I have never in my life envied a human being who led an easy life. I have envied a great many people who led difficult lives and led them well."

Starting out in any industry, there's bound to be some sacrifice. I don't want to sugarcoat it—whether you are working as an independent developer or working for a company, you're not going to start out at the top. As an independent game developer myself, I know how difficult it can be to separate "regular" life from work. I work from my house, and all my coworkers are in different time zones, so quite often I'm working at very odd hours, and it can be difficult to turn off my working brain. For those joining companies, you'll start with little power, you'll do lots of grunt work, and won't get much pay. That's the fun of being an unproved hire! But should you still make video games? Absolutely! Just don't expect immediate success or for your life to get magically easy.

Starting out

Here's a big question that many ask: How should you get started? Should you go to a game school? Should you work a full time job and do games on the side? Get a job at a major studio? Quit your job and go full indie? As with all things, it depends on who you are. If you simply "want to make games," but have no particular idea what you want to do within that framework, school might be right for you. But school is more expensive than ever, so keep in mind that you may be committing to debt as well, and you've no guarantees of success, even after you graduate.

If you've worked in a related industry and have some coding or art chops, you might be able to get a job at a big studio somewhere, and gain experience. But don't expect to get a job as a neophyte, unless that job is in testing/QA. Being a tester is still a good road to becoming a producer or game designer, so you could choose worse paths than this. You get paid to learn as you go—but it's tough to get out of QA once you're in it, unless you're very ambitious.

For all levels of experience, starting or joining an indie studio can be creatively rewarding, but money will be hard to come by for quite a while (possibly forever). You need the right idea, enough time and runway before you must get paid, and a whole lot of luck if you're going to succeed as an indie. While it's true that there are more and more venues for getting money, like Kickstarter, new game platforms, and the traditional publishers, each of these venues gets increasingly crowded over time, as more indie studios crop up around the world. The lower barrier for entry makes game development more accessible, which is great! But that also means there's a lot more competition.

Just do it

All that sounds pretty negative, I realize. I don't want to discourage you, I just want you to know there's no established formula for success, aside from hard work and inventiveness. So where does that leave new game developers? Ultimately the same place they've always been. The game industry has always been competitive, like all creative industries. The only thing you can do, if you're committed to making games, is to make

games, any way you can.

The nice thing about the current state of the industry is you can get started even if you're not an artist, or a coder, or a musician. You can make a text-based game in Twine. You can learn to make games of basic competence in GameMaker in a few weeks. You can use Unity or Unreal's scripting tools and asset stores to make something that should be beyond your initial ability. It just takes time and dedication.

No matter whether you're going to school, applying for a job, or starting a studio, having some game making experience under your belt is a good idea. It'll give you a taste of what's to come, and you'll have some practical skills you can apply going forward.

There's some amount of difficulty involved in committing yourself to any art form, but the personal rewards are very compelling. You'll have to work and toil and grind away at your game before it really becomes anything—and this is normal—but once it all starts to come together and your ideas come to life before you, the rewards feel very real.

Make your own path!

No other industry comes as close to actual world creation. Game developers are the masters of their own universe, and the more new ideas, new feelings, and new opportunities there are, the better off the entire industry will be. So we welcome you to game development, no matter what your path may be! And if you don't see something that works for you within these pages, go out and forge a path of your own!

It's our hope that this guide will help you get started faster, and to avoid some common development pitfalls. This is a very friendly industry, so feel free to reach out to any of our authors with non-searchable questions, and share this guide with others. Let's all make something wonderful!

Brandon Sheffield, editor
www.twitter.com/necrosofty

illustration by Juan Ramirez

STARTING A COMPANY: GIVING IT (AND YOU) THE BEST CHANCE TO SURVIVE

GAMES ARE BUILT ON CONSTRAINTS AND LIMITS. STARTING A COMPANY IS A LOT OF WORK—I KNOW, I'VE DONE IT. I can't tell you how to make a successful company (and here's my first tip, don't trust anyone who promises they can), but what I can tell you is how to start one that will survive, and one that you'll survive starting.

Have the best reasons

It sounds idiotic, I'm sure. But if you're going to start a company, you have to make sure you're doing it for the right reasons—and be sure that you are the right person to be starting it. You're deciding to become a businessperson and a boss, and that means a lot of hard decisions, paperwork, and leadership. If you don't want to do that—let's say you just really want to make games—then this isn't for you, and you're going to have to find people you trust to partner with. However, the nice thing is that you can stop reading this article now and turn the pages to the next one!

But seriously, some good reasons to start a business include:

- You've got an idea that you want total ownership of. No one will force you to do anything, and you're willing to take on a lot of work that doesn't involve the idea.
- You can't work for someone else! I'll be honest and make it clear that I'm bad at that. No authority figure is going to tell me what to do!
- You're working with other people and you're the best candidate for the job. Sometimes necessity is the mother of businesspeople, as well as invention. (She mostly talks about the invention, though. She's a bad mother, with favorites.)
- You just really, really love forms and spending thousands of dollars on lawyers, and you're not in a mental institution already.

Take it seriously

If you're going to do all this work, you're going to have to make sure that it's going to be worth it. The reason you're doing this is because you're sensible enough to know that you can't just, say, start making a game with a bunch of friends, and expect everything to work out and the game to get released. You've recognized that you need proper, written agreements (contracts!) that will protect everyone involved (not just yourself, not just the company—you also can't view this selfishly) and the work they have done or will performeddo.

So, if you're starting a game development company, it's going to be important from the onset to have that first project in mind—maybe it's a passion project! Maybe it's a jam game!

Also make sure you have the people you need to start that project lined up. Then...

Register as a corporation

This probably sounds like the most insane overreaction—after all, there are many options such as limited liability partnerships—and so you might want to talk to an actual business person before going in on the deep end, but a corporation protects everyone from liability when it comes to bankruptcy and litigation (as long as you've got proper contracts) and can be important when dealing with your local government for grants, tax-breaks, and so on. Yes, even if you're a tiny, one-person indie who is only thinking of using some contractors.

After registering the corporation, you're going to have to designate shareholders and sell them shares—I'm not going to go into detail on this, because that will be up to you. If you're looking for personal details, I run a corporation with a single shareholder, myself, to further protect myself from having to do any "buy-outs" early in the company's life, if someone changes their mind and wants out. However, if you want a different structure, you can have all company staff own shares, or sell shares to someone else who was is looking to invest in a fresh video game company.

Get contractual

Having "employees" as an indie is incredibly difficult—not merely in terms of the workload, but also the legal requirements. It's highly likely that you'll hire everyone as contractors unless you reach a certain level of success quickly; although anyone who owns the company won't be able to be a contractor—they/you will probably be "pulling dividends." (Either way, alongside the expensive lawyer that you're going to have to spend money on, you're going to have to spend money on an accountant, which really isn't optional.)

Everyone, however, even owners, will have a contract with the company. Why? Because the company, a corporation, needs to know what's going on with the work that is being produced, and the people working for the company need to know exactly what they're doing and what they're going to get as a result.

And, most importantly, they need to know that everything will be okay if things don't work out; if they want to leave the company, or need to leave the company for some reason.

That means you're going to have to hire a lawyer, or get one on pro- bono, or use (within reason, and hopefully with someone with a legal eye to look over it) a web service like docontract() or Shake. Generally, this should state the corporation owns everything produced for it [i.e., work done on this video game you're making) or that it holds in-perpetuity rights for the work within that context (a more complex idea, but good for musicians or sound designers who'd like to sell soundtracks, or use their work elsewhere).

Why? Well, imagine the situation where your AI programmer, without a contract, decides to quit and take all of their work with them. Now all of your enemies stand there like lemons and there's nothing you can do except get litigious, which is going to cost money, and time, and not even guarantee anything, so all you can actually do is hire someone to do all the work again.

Compare that to a situation where your AI programmer leaves, but the corporation still owns all the work they performed for it. You've still got an AI system, and maybe it isn't finished, but you can hire someone else to work on the code that's already there.

And while maybe you and the AI programmer are the best of pals, and you feel they'd never do that, without a contract it's similarly hard for them to leave. How do they know you're not going to try and force them to finish the project even if they leave their code behind, or something equally ludicrous?

Don't spend your own money

This is business rule number one, and probably the one that seems most unattainable. Beyond meaningful start-up costs (incorporating, buying your own shares) you really don't want to be putting your own money into the company to keep it afloat. For owners, time and your own flexibility are your currency—cut back in your own life, spend more of your own time—but do not consider anyone working on contract to be "flexible." Pay them as fairly as you can—even if this involves a contractual obligation to pay them revenue share in the future (another reason why contracts are so important).

Explore every avenue for funding: Platform platform holders, publishers, local government

funding in the form of arts grants or loans, the obvious examples of Kickstarter or other crowd-funding platforms (even Patreon), and so on. But any money must go to contractors first.

Don't be afraid to ask for help

"Explore every avenue for funding?" But what if you don't know any platform holders?! Here's probably the most important piece of advice I can give you: Don't be afraid to ask for help. Some people will have this easier—maybe you already live in a city with an established game development community, for example. Find the nearest game development community, or nearest established developers, and talk to them! Not everyone is going to have time for you, but making those connections will be a massive step in establishing yourself, and as "schmoozy" as it sounds, you won't just make business associates that can help, you'll make friends that will help. We're all in this together.

Know how long it's going to be before the company starts making money

You've tightened your belt and know you can get away with only spending so much money per month—but for how long? What if the AI programmer quits, or someone promised they could do X in Y amount of time, but it's actually taking Z? The worst aspect of starting your business is, surprisingly, starting your business's first project, because it requires something basically no one likes doing: project management. (Sorry to all the project managers out there!)

It's hard to "do it all." Be wary as the CEO of a fresh corporation of taking on too much. It's highly likely you'll be so busy with business "stuff" that project management will slip massively by the wayside, especially if you're the creative director and designer at the same time (because you'll want all the cool stuff in the game that you originally wanted, and something's gotta give.). Under no circumstances should you try and to run an unmanaged project. It's a death sentence. You may need to hire someone. I'm sorry.

Schedule everything with clearly estimated times until the end of the project. Don't know how long something is going to take? Make your best guess. Then, in every situation, double it—and that's already after scheduling with the worst case in mind. You want to be smashing your targets, not slipping by them. Why? Because if you budget for targets you miss, you're in a terrible place.

targets you miss, you're in a terrible place.

The problem, of course, is that you'll believe that your new schedule is wildly unrealistic, scheduling your game for release in about 2028. It isn't! Before you've gone into production, you now know how long things could take and (gulp) can start to cut down into something manageable if your funding won't stretch that far. It goes all the way back to my first point: You have to take it seriously. But ultimately, there's one important thing to remember above all.

Don't be afraid you'll become "The Man"

I admit it. This article is dry. It's a demonic turkey sandwich. It's a lot of stuff that honestly just doesn't sound like it's got anything to do with video game development—incorporating? Contracts? Project management? But the unfortunate reality is that if you want ownership and control of your video games, you have to step up, and start a business, and run a business (unless you're happy with it as a hobby, which is fine too!).

The most difficult thing about it is the feeling that if you start a company, and you start telling people that you need work done by a certain time, and that they have to sign an contract... even if you're not making much (or any!) money, you feel like "the man." You might even feel like why you got into video games—to enjoy working in a field you love—is slipping away.

I've got to say, however, that you won't be "the man" by just running a company in a clear, sensible, and proven fashion. And you'll be doing all the hard work that's required to make sure you can keep working in the field you love, because if you can start a company and get that first project out without breaking the bank, frustrating your collaborators, or any number of things that can go wrong without this kind of effort, you are much, much more likely to be able to start work on a second project.

And that will make it all worth it.

Mathew Kumar is the CEO of MK-ULTRA Games, developer of PC and OUYA title Knight & Damsel. He made a lot of mistakes in the process of starting his company—and he really doesn't want you to make them too.

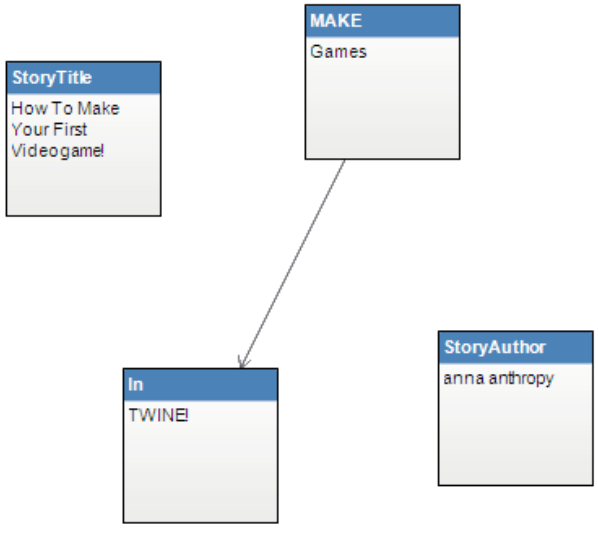

In the world of game development
EXPERIENCE
MATTERS
Offering degree programs in the following subject areas:
GAME DESIGN & DEVELOPMENT
COMPUTER SCIENCE & ENGINEERING
DIGITAL ART & ANIMATION
MUSIC & SOUND DESIGN
SEE WHAT YOU CAN ACCOMPLISH
DigiPen Student Artwork » Joseph Biwald
www.digipen.edu
DigiPen
INSTITUTE OF TECHNOLOGY

TWINE: AN EASY WAY TO START MAKING GAMES WITH WORDS

THERE ARE A FEW MAJOR OBSTACLES TO GAME-MAKING WITH THE TRADITIONAL MODEL: (1) Need to know how to program; (2) Need other people to contribute assets like images, models, or sound; (3) Need to be able to afford game-making software.

Twine is free, text-only, and requires no programming. It's a very easy-to-use platform for creating hypertext games and stories. Plus, it's easy to distribute: Finished games are published as HTML files, which can then be uploaded to a site like www.philome.la. As a bonus, Twine games are more player-accessible than most, since they're played with just the mouse and generally require no dexterity.

Ready to make a Twine game? It's simple. Here's everything you need to know.



Making games with Twine

You can download Twine at twinery.org. There are actually two versions: Twine 2, the online version, and Twine 1, for Windows or OSX. I use the downloadable version myself. The differences are slight.

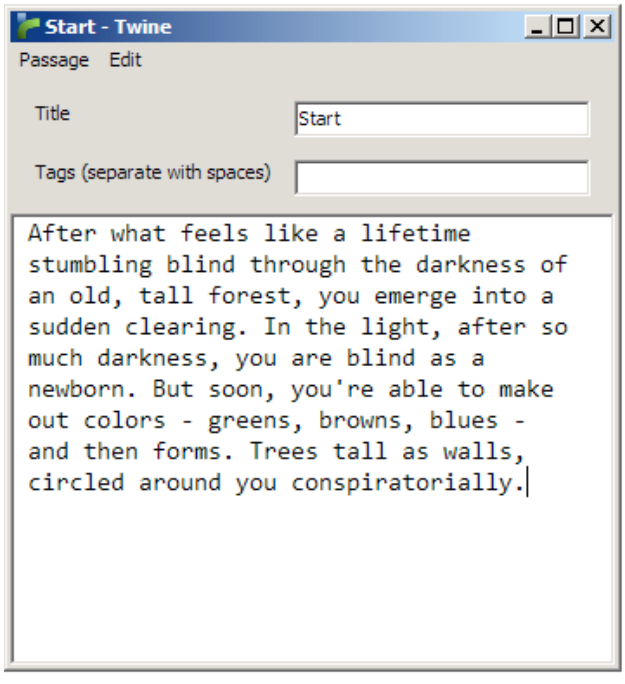
When you open up Twine you should see three boxes, marked "StoryTitle," "StoryAuthor," and "Start." These are called "passages." In the first two, you can type the name of your story and your own name. The "Start" passage is the very first passage the player will ever see. Don't ever rename it!

Instead, double-click it and let's start writing!

A window should pop up that you can type in. Type whatever you want! We are the music makers, and we are the dreamers of dreams!

Great! Now, here's the most important thing to know: How to let your reader travel to other passages in the story. In Twine's text editor, a link looks like this:

[[We are the dreamers of dreams.|Dreamland]]

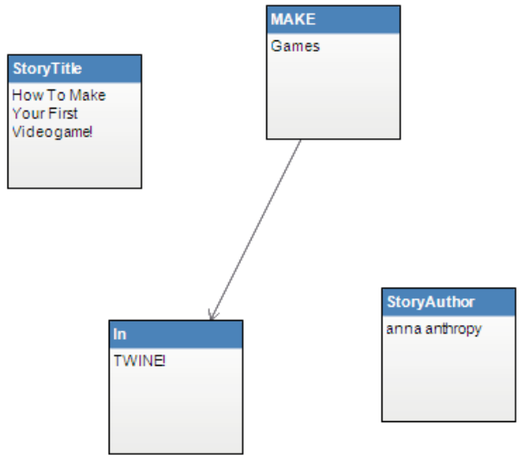


TWINE: AN EASY WAY TO START MAKING GAMES WITH WORDS

THERE ARE A FEW MAJOR OBSTACLES TO GAME-MAKING WITH THE TRADITIONAL MODEL:
(1) Need to know how to program;
(2) Need other people to contribute assets like images, models, or sound; (3) Need to be able to afford game-making software.

Twine is free, text-only, and requires no programming. It's a very easy-to-use platform for creating hypertext games and stories. Plus, it's easy to distribute: Finished games are published as HTML files, which can then be uploaded to a site like www.philome.la. As a bonus, Twine games are more player-accessible than most, since they're played with just the mouse and generally require no dexterity.

Ready to make a Twine game? It's simple. Here's everything you need to know.



Making games with Twine

You can download Twine at twinery.org. There are actually two versions: Twine 2, the online version, and Twine 1, for Windows or OSX. I use the downloadable version myself. The differences are slight.

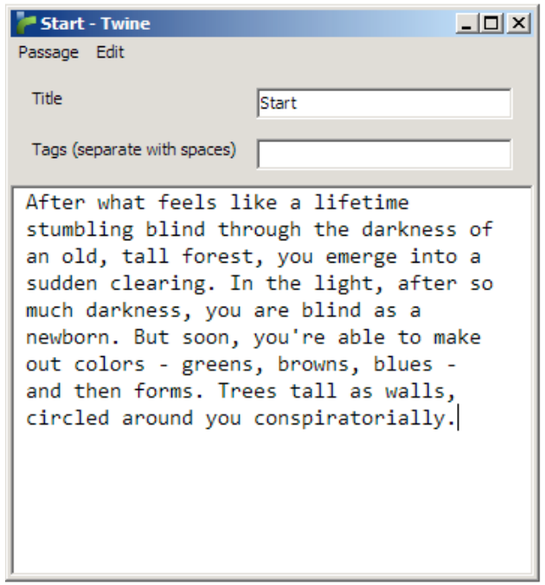
When you open up Twine you should see three boxes, marked "StoryTitle," "StoryAuthor," and "Start." These are called "passages." In the first two, you can type the name of your story and your own name. The "Start" passage is the very first passage the player will ever see. Don't ever rename it!

Instead, double-click it and let's start writing!

A window should pop up that you can type in. Type whatever you want! We are the music makers, and we are the dreamers of dreams!

Great! Now, here's the most important thing to know: How to let your reader travel to other passages in the story. In Twine's text editor, a link looks like this:

[[We are the dreamers of dreams.|Dreamland]]



Double brackets. Vertical line in the middle—it's on the same button as the backslash on most keyboards. Remember that.

The text to the left of the | is the text the player sees, what the actual link looks like. When she clicks on it, she goes to the passage matching the name to the right side of the |. In this case, it's "Dreamland."

In a newer version of Twine, when you close this passage, Twine should offer to create the new Dreamland passage for you. If not, you'll have to click the "new passage" button and name your new passage "Dreamland." You should see an arrow pointing from one passage to another, letting you know they're successfully linked.

To make sure it works, you can right click on your starting passage and select "Test play from here." Your game will open up in your browser.

Those are the basics! Go back and add some additional links and passages and you've created choices for your reader.

Distributing your game

Click on the "Build" drop-down menu at the top of the screen, then pick "Build Story..." You'll be asked to name the file. Hit Save and voilà! Your game has been saved as an HTML file, and it should pop up in your browser to play.

In the future, whenever you want to update this file, you can just choose "Rebuild Story" from the Build menu and the HTML file will be quietly rebuilt.

Note that building a distributable version of your story is different than saving your Twine source so you can work on it again later. You do that by going to File --> Save (which produces a .TWS file). Make sure to save often!

Once you've created your finished HTML file, you can upload it for free to somewhere like www.philome.la or neocities.org, or your own website. Then folks can play it in their web browsers!

A few more tricks

Do you have some text you want to appear in more than one place in the story? Instead of writing it twice, use display! Here's what it looks like:

←←display 'name of passage'→→

Those are the pointy brackets, not the square ones. And single quotes, not doubles. The passage you name should appear in that passage.

Here's another thing you can do: Randomized text using either:

"Gimme a Shirley Temple ←←print either("on the rocks","and make it dry")→→."

You can even combine display and either to display a random passage:

←←display either("Dreamland","Mundan ia")→→

There are a few more helpful things Twine can do, like tracking variables or displaying images. If you know CSS, you can customize how your story looks. For an intro to those sorts of things, see my online Twine tutorial:
http://auntiepixelante.com/twine/

An example game

Try Star Court. It's a game about being on trial in space:
http://auntiepixelante.com/starcourt/

It uses a lot of eithers to randomize text and influence chance, tracks a ton of variables, and features a central game loop with multiple endings. When you're ready, download the source and poke around in it:
http://auntiepixelante.com/starcourt/

Anna Anthropy is a play designer, game historian, and the author of Rise of the Videogame Zinesters and ZZT. Find more of her work at auntiepixelante.com.

WE'RE HONORED

TO BE NAMED ONE OF FORTUNE'S 100 BEST COMPANIES TO WORK FOR IN 2015. WE'D LIKE TO THANK EACH AND EVERY ONE OF OUR GREAT EMPLOYEES WHO BUILD THE WORLD'S MOST BELOVED ENTERTAINMENT FRANCHISES.

ACTIVISION® | BLIZZARD®

FORTUNE 100 BEST COMPANIES TO WORK FOR 2015

UE4 VS. UNITY 5 AN ENGINE COMPARISON FOR NON-PROGRAMMERS

UNITY AND UNREAL ENGINE HAVE BEEN COMPETING FOR GAME DEVELOPERS' HEARTS, EYES, AND MONEY FOR THE LAST SEVERAL YEARS (MAINLY SINCE UE4 CAME OUT). Now both parent companies are aggressively targeting indie studios, touting design tools that reduce the need for a programmer. It can be hard to know which engine to choose! So, we've broken down the differences in terms of price, UI, content pipeline, premade assets, support, and more to help you make a more informed decision before you jump in.

Price

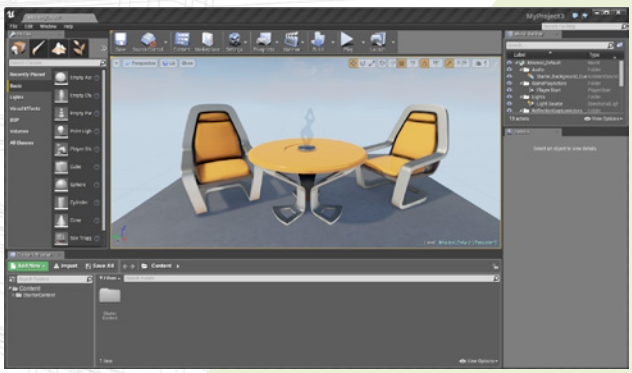
When you're an indie developer, money is always a problem—so your first question might be: How much do these things cost? There was a huge difference in price between UE4 and Unity 4 at launch. UE4 was $19 per month—for that, you got the ability to cancel at any time plus the full-package engine, including its source code. Unity 4's pro version was comparatively expensive—a high one-time fee of $1,500, and users had no access to the source code. Of course, many people used the free version, but access to higher functions was limited.

Now, after GDC 2015, everything has changed. UE4 is now completely free (and still includes the full source code), and Unity has released version 5, which has a "personal" free version of the software. This contains all the "pro" features, but will show "powered by Unity" splash screens on every build. You still don't get source code access. The price of the "pro" version is still a one-time fee of $1,500, or $75 per month, and if you want to get iOS and Android pro versions, you'll need to pay $4,500. But before you start thinking UE4 is the better deal all around, consider also that UE4 takes 5 percent revenue off the back-end of your title, whereas you pay up front for Unity.

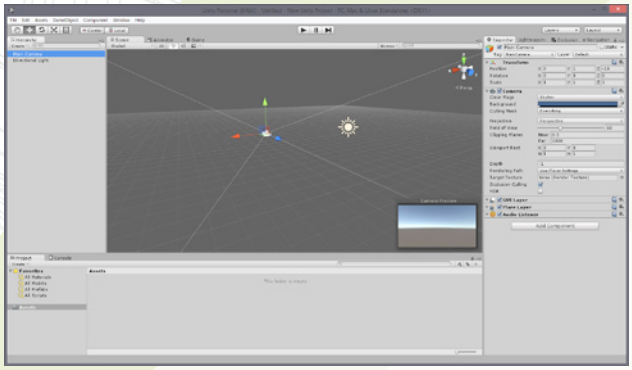
The UI

Whichever engine you choose, the first thing you'll likely notice is the UI. Unity 5's UI is quite simple and straightforward. Many of its advanced features are automatically hidden for new users. UE4, on the other hand, appears comparatively complicated when you first open it (and it is, but still simpler, I'd say, than CryEngine 4!).

To me, the UE4 UI feels like a professional triple-A engine, while Unity's UI is simpler and therefore much easier to get started with. Tutorials are available to guide new Unreal users through the UI. But you should also note that UE4 comes with much higher hardware specs—its makers recommend 8GB of RAM. Unity's requirements are much lower.



The Unreal workspace.



The Unity 4 workspace.

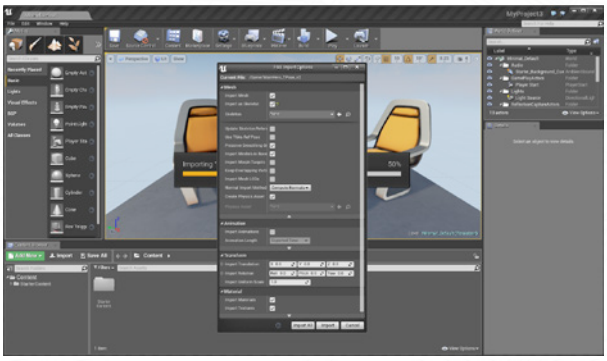
Content pipeline

Once you've got a handle on the UI, you'll be wondering how to get content into the engine. In the past, you would have had to use specific content formats in order to feed your assets into your engine, but both UE4 and Unity 5 allow you to drag and drop most asset formats into the engine. The engine will convert them into their desired format without any extra input.

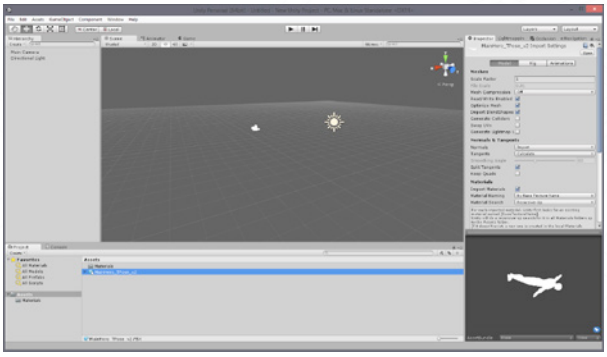
There's a small difference between how the two engines import the files. Unity copies your original asset's format into the project folder while the engine processes the asset into its own format—so you wind up with two copies of your asset, the original format (PNG, for example), and the processed format.

UE4, on the other hand, processes your asset (that PNG, say) and stores only the processed file in your project folder with information about the source path to the original file, plus the option to reimport it.

Another important difference to consider is which original asset formats each engine supports. If you go through the documentation, you'll see that UE4 supports fewer formats while Unity is able to process a wide variety.



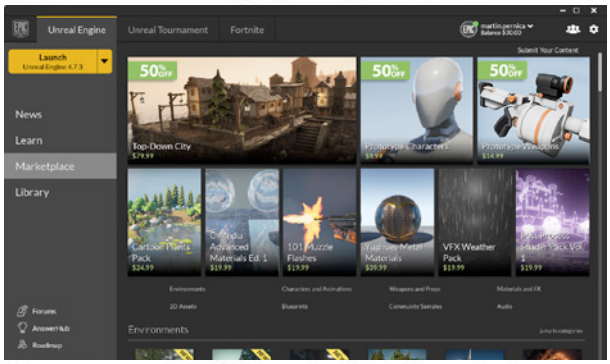
Importing assets into Unreal.



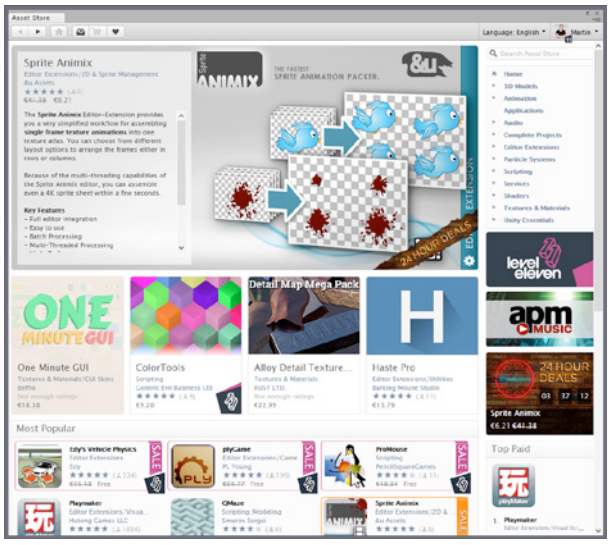
Importing assets into Unity.

Purchasing premade assets

Sometimes you can't (or don't want to) create every asset or system in your game, opting to use prefabricated 3D models, VFX, animations, or scripts instead. To facilitate this, both engines have a store where you can purchase assets piecemeal for use in your game. Unity's Asset Store is much bigger than UE4's right now (mainly because it has been around longer), and the process of adding new assets to the store is simple and straightforward. The UE4 Marketplace is young, and the process of adding new assets to it is quite slow and strict; Epic approves only a certain standard of assets. As a result, Marketplace assets are often of higher quality, but there are fewer of them. On the other hand, a search in the Asset Store may turn up many asset options of varying quality—you'll need to sort out for yourself which one will work best for you.



The Unreal asset store.



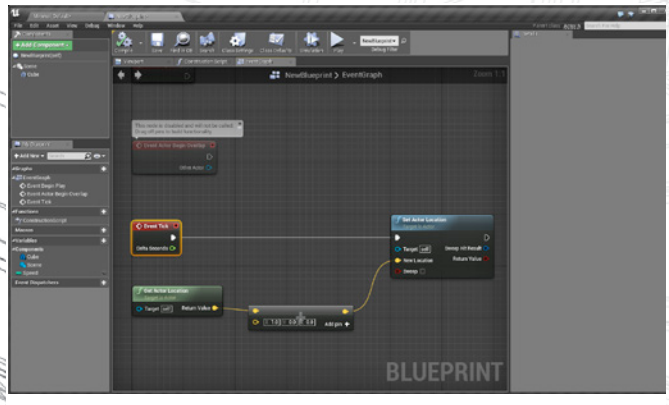
The Unity asset store.

Bring your assets to life!

Once you've got your assets processed and imported, you can start dragging them to the scene window to start building levels. When you've got static ones in place, you'll want interaction, so you'll need to get your player characters moving, whether that's with a game pad, a keyboard, or touch controls.

If you're a programmer, this is simple—Unity supports programming in C#, and UE4 will accept C++ code. But if you're not a programmer, the situation is quite different. You can't just learn C# or C++ in a couple weeks, so what should you do?

In UE4, the solution is called Blueprints. This is basically a visual programming tool that allows you to connect nodes of logic together so you can proceed without learning a programming language. You just need to understand how basic concepts like variables, functions, and so on work.



Unreal's Blueprints.

Using Blueprints is still much simpler than programming in C++ or C#, so you should definitely give it a look. Blueprints isn't particularly limited, either—in fact, you could do an entire game using Blueprints, foregoing a traditional programming language! Check out the "Tappy Chicken" sample project Epic provides (essentially a Flappy Bird clone), which was created entirely by one artist using Blueprints.

So what about Unity, then? You'll run into some problems here, as Unity doesn't have a default visual scripting tool. With it, you've got two choices: learn C# or buy a scripting plug-in, such as Playmaker, in the Asset Store. But you'll need to pay for it, and it's not a core feature of the engine like Blueprints is for UE4.

Unity requires third party plugins for similar functionality.

Materials

As you're building your game, you'll come across something called materials—pieces of code under the umbrella of shaders, which describe how things look in your game. For example, a wall may appear more "plastic-y," while a car will look more metallic. You can paint everything in your own textures, but that's not the best solution due to lighting differences and other post-processing variables. So, you'll need to use materials.

Both engines support a modern shading model called physical based shading (PBS). The main goal of PBS is to simulate the real-world behavior of materials.

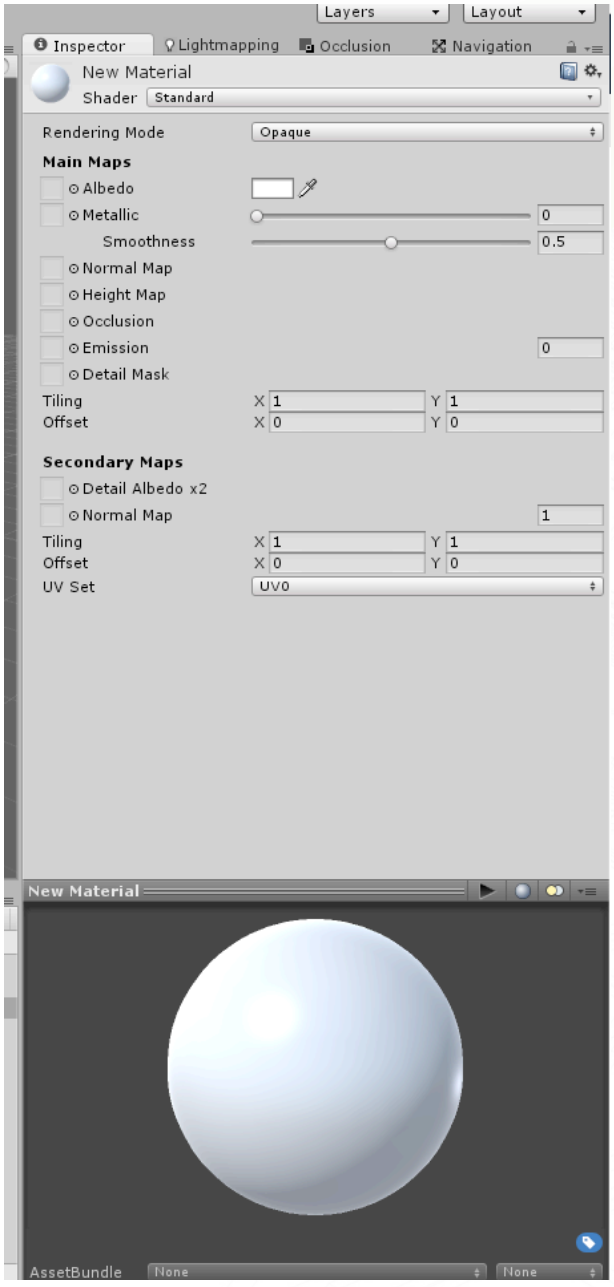
In Unity 5, the engine uses something called the "standard shader" when you create new materials. This standard shader has some input variables, like Albedo, Metallic, Normal Maps, and so forth. You can connect your textures to these inputs and create your own materials from them. But when you need a bit more magic in your Unity 5 materials, you'll need to dive into shader programming, which is quite complex.

In UE4, you'll have some very similar inputs to what you find in Unity 5 when you create new materials, but you can do a lot more than just connect to your textures in the materials editor. You can work your magic with materials without any knowledge of shader language, because the UE4 materials editor is similar to the Blueprint editor. You just drag and drop nodes of logic to connect them.

As you may have guessed, materials editors exist in Unity as well, but they exist as plug-ins in the asset store, many of which you must pay for.

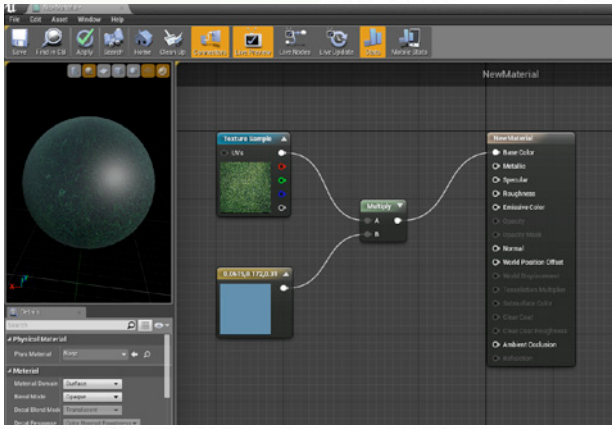
Learning resources

Engines are huge pieces of software, and they're



Unity Materials.

difficult to learn by yourself. Unity and UE4 both have a ton of material from which to learn—video tutorials, documentation, community tutorials, books, and more. Unity has been open to indies for longer, and as such, it has a much larger community and more resources to draw from. The disadvantage is that you'll sometimes find outdated tutorials. Still, new ones are always being made, and there's plenty of support from both official channels and unofficial forums.

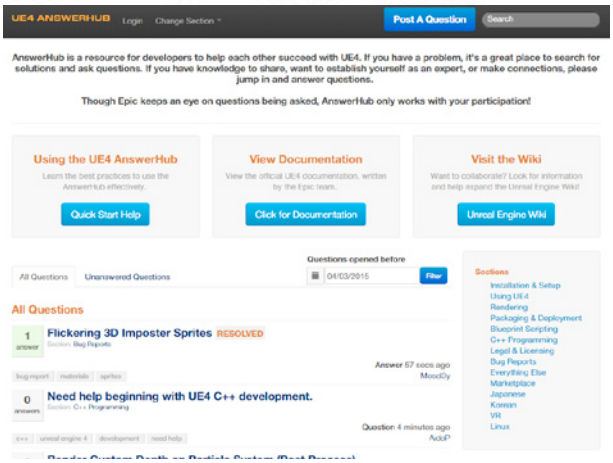


Unreal Materials.

Community and support

Finally, when you choose your engine, you become a part of the user community and receive support from other users. Both engines have official forums and answer hubs where you can submit problems and discuss them. Unity's community is huge, and you're often able to get responses from the community before the Unity team even sees your question. UE4's community isn't as well-established, so answers from fellow developers may come slower, but you're more likely to get answers from Epic employees.

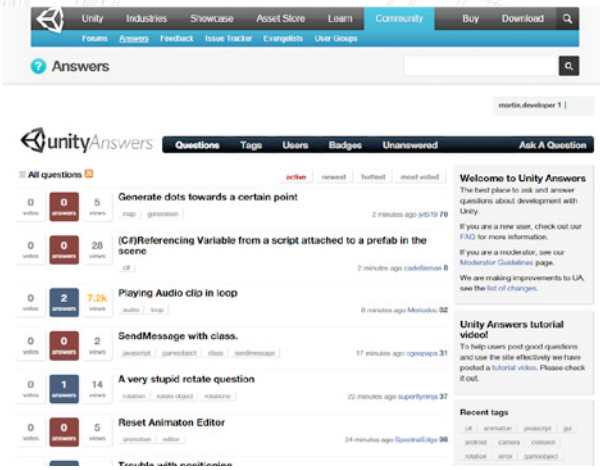
You can also use each engine's official email support. From my personal experience, Epic gives more thorough responses and is much quicker to respond than Unity is.



Unreal support interface.

So, which to choose?

This isn't actually the right question to ask,



Unity support interface.

necessarily. Perhaps the more appropriate question is this: What game you are making? If you're making a 2D game, Unity currently has much better 2D support than Unreal does. But if you're trying to create the next Skyrim, UE4 is more likely the better choice, as it is tailor-made for big games and big teams.

Even if you're working alone, though, UE4 can support you with Blueprints and its materials editor. On the other hand, if you're hoping to use some ready-made assets, like models, scripts, or VFX, the Unity store is much more robust. And if you're hoping to make a smaller game for mobile devices, Unity has the edge because of its smaller hardware requirements.

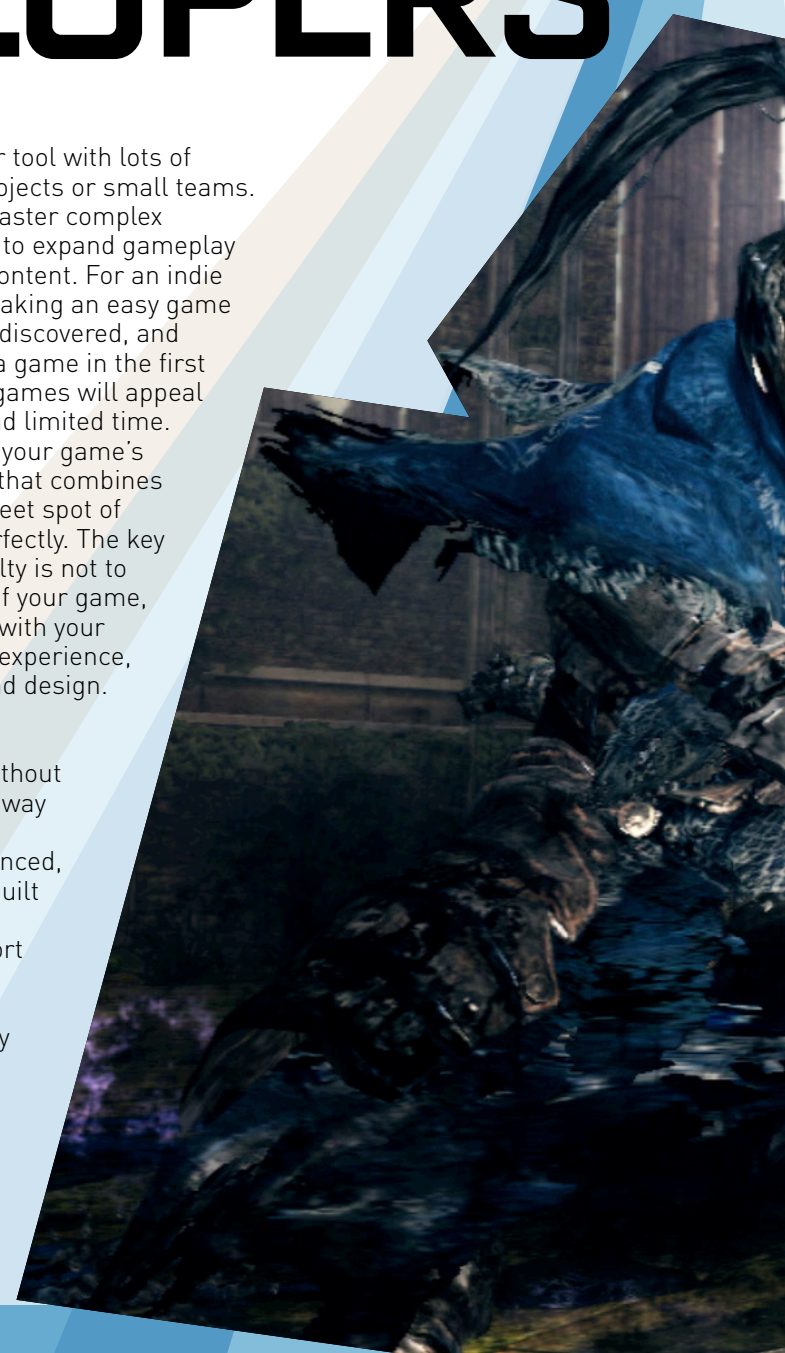
Of course, neither is the silver bullet for all game development needs. Both have their ups and downs, and both take time to master. Whatever engine you use, what really matters is the game you choose to make with it!

Martin Pernica is lead programmer and co-founder of young game studio Soulbound Games in the Czech Republic, currently working on Renoir. He also teaches classes on game development and mobile applications at a local university.

THE DIFFICULTY WITH DIFFICULTY: A GUIDE FOR NE DEVELOPERS

A game's difficulty is a powerful developer tool with lots of practical benefits, especially for solo projects or small teams. A difficult game can entice players to master complex systems, and allow development teams to expand gameplay time even if they can't afford a lot of assets or content. For an indie developer, these are important positives. But making an easy game has its upsides, too. It's hard for an indie to get discovered, and people are more likely to play and recommend a game in the first place if it's easy to get into. Not only that, easy games will appeal to a wider audience with more diverse tastes and limited time.

The goal of this article is not to help you get your game's difficulty just right, hitting a perfect sweet spot that combines accessibility and depth. There is no one true sweet spot of difficulty that makes every game hum along perfectly. The key to finding the right place for your game's difficulty is not to imagine it as a separate, abstract layer on top of your game, but to think of it as something deeply entwined with your game—something that supports its theme and experience, and is supported in turn by your game's tone and design.



Holistically Hard

It's impossible to write an article on difficulty without mentioning Dark Souls, so let's get it out of the way right here. Dark Souls is praised so often for its difficulty, not because it's particularly well-balanced, but because it's a perfect example of difficulty built into the game's core. Dark Souls' difficulty is a centerpiece, and the game is designed to support and highlight its absurd challenges. Enemies jump out from intentionally unfair hiding spots, and archers are placed at the end of ridiculously long, narrow bridges. The system of gathering souls and bringing them back to your base is designed to punish players for dying, but it also allows them to slowly bank progress and start to push back against the game. The attacks are intentionally slow and deliberate, making it clear to players that they must master this

EWER

James Lantz

The Souls series is a modern example of difficulty done right.

system, and making the combat easier to understand and learn. The "You Died" screen is the cherry on top, cementing its difficulty not just as a set of balance choices but as a central part of a holistic design.

Difficulty is ingrained in every element of Super Hexagon, making it holistically hard.

Similarly, the excellent Super Hexagon bakes obscene difficulty into its tone, music, theme, and mechanics. Each difficulty level, which, at launch, escalated from Hard to Hardestest, is designed for a new player to fail within 2-3 seconds, slowly graduating to 5 seconds, and then eventually 10 seconds, and so on. These wildly short rounds of play fit perfectly with this incredibly and consistently hard game.

The point is, if you want to make your game superhard, you shouldn't just balance it so that the player loses a lot and it's tough to beat. You should build difficulty into its heart and soul. If you're making players master your system, make sure it's a system worthy of their time and mastery. Make sure the system is transparent, so it's clear why they lost and how they can do better next time. Make sure your game isn't asking players to learn and master every mechanic at once, or keep track of a complex story and learn tons of mechanics at the same time.

Still, while these are good rules of thumb for any difficult game, all game design rules are made to be broken. A great design could combine a supercomplex story with a ridiculous number of mechanics and make it work spectacularly, as long as it takes full ownership of those ideas and supports them with its theme, tone, presentation, and the rest of its design.

A good example of a game failing to design difficulty holistically is Call of Duty 2's single-player campaign. The game works well at the Regular (normal) difficulty, but lost its focus on the highest difficulty level, Veteran. Difficulty is balanced toward the hard end without the support of the rest of the design—enemies throw a ridiculous number of grenades and often shoot you instantly on sight, making progression slow and usually random. This doesn't fit at all with the game's focus on spectacle and experience, and really only exists for replay value. It doesn't sink the game as a whole experience, but is still a good example of a difficulty level that wasn't designed with the rest of the game's support.

Difficulty Levels

When designing difficulty, one of the most important factors is the core unit of experience in your game. Are you creating a linear story, intended to be played once by most and twice by dedicated fans? Are you creating a complex system intended for absolute mastery? Are you creating a game with procedural and random elements, intended to be played over and over? These qualities affect the types and level of difficulty you want in your game.

For a complex game that's all about deep systems mastery, you'll usually want to create a gradual difficulty curve that eases players into a deeper


Call of Duty 2's difficulty falls apart at higher levels.

Infinifactory introduces new elements one block at a time.

understanding of your game. You can still have an interesting game with low difficulty because your complex systems are carrying most of the weight and soul of the game. But players probably have enough on their plate learning and mastering a deep web of interesting systems, and don't need to deal with cranked-up encounters. You, as a designer, know that the most interesting part of your game is when the deep systems meet the interesting challenges, but often you have to lead the player there slowly, because if you throw them in the deep end right away, they'll probably just bounce off.

Infinifactory is a great example of this principle at work. Every idea that Infinifactory presents feels fresh the first time you see it, and so the game takes hours and hours introducing its blocks, which represent new mechanics, one clever puzzle at a time instead of throwing you into the deep end. Because Infinifactory is so fundamentally challenging, the difficulty curve ramps very slowly so you don't just bounce off immediately. Infinifactory's early levels also find a sweet spot between tutorial and interesting gameplay, which is important for any systems-based game.

On the other hand, if your game is built on standard tropes, you may want to frontload some of your difficulty to show players right away what makes your game different or particularly interesting. Pillars of Eternity is a classic RPG built by veterans of the genre, and it immediately demonstrates its difficulty with quick deaths to those who mismanage their party or push forward hastily.

How Players Experience Difficulty

Indie developers don't have a lot of time, money, or manpower for playtests—often leaving just you and a handful of close friends to figure out how players feel when they play your game. One solution to this problem is finding a community through a Kickstarter or Steam Early Access, but if that's not the direction you want to take your game, then you have to develop the ability to extrapolate from your own experience with your game's difficulty. To do that, you have to learn to put yourself in the shoes of a new player.

As a new player, your perception of a game's difficulty has less to do with how high the numbers are tuned on a boss and more to do with the opaqueness of the game's systems, how much the game is asking you to learn, and how quickly it's asking you to learn it. Frequent failures can be frustrating, but what's more frustrating is hitting several failure states when you don't understand what you're doing wrong, how to do it better, or worst of all, are missing whole chunks of understanding about critical systems.

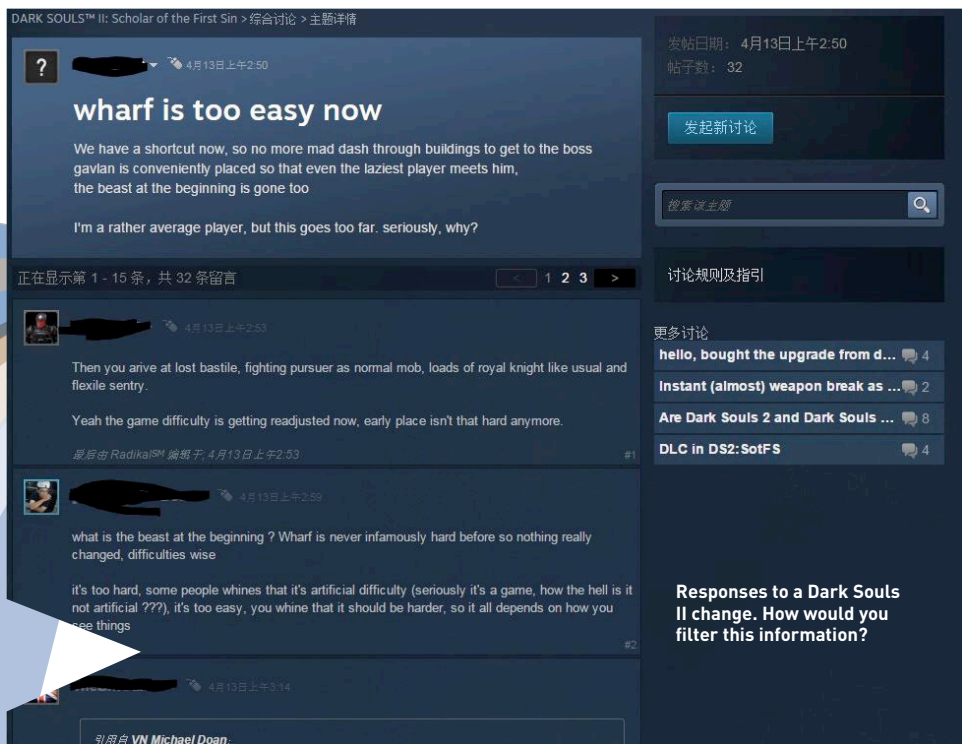
Halo is difficult without being frustrating (most of the time).

Halo is a good example of a game that is difficult without being frustrating, except in a few confusing levels. When you first encounter tougher Covenant enemies, you'll probably hit several checkpoints where you die and fail over and over again. But each time you die you feel like you have a complete understanding of what happened—you just failed to execute. That's because Halo's systems and UI are clear, transparent, intuitive, and simple. When you get shot and die behind cover, it's because an enemy flanked you, not because they had some hidden bonus you forgot about after the tutorial. This means that every time you die, you'll have a coherent theory about how to approach that situation next time with a near complete understanding of the game rules that keeps you from getting too frustrated.

Of course, not every game should be as straightforward as Halo. But every game can work hard to give its players a working model of the game's systems and make those systems as transparent and intuitive as they can be, so that when players hit the difficulty wall, they have the tools to stay grounded and understand what to try next time, giving them a sense of improvement even after their failure.

How to Work With Feedback on Difficulty

Absorbing and using player feedback regarding difficulty is the hardest part of balancing a game. First off, it's important to realize that players are going to be cruel and they're going to hurt your feelings. Whether it's because they're insecure about feeling stupid, they want to puff up their ego, or they simply have different tastes than you, players and playtesters are going to write epic stories about how terrible your game's balance is and how much of an idiot you are. What's important at this point is to



Responses to a Dark Souls II change. How would you filter this information?

develop the ability to take in that feedback, throw out the really vitriolic stuff, and then be able to get as much value as you can from what's left.

A lot of players like to use words like "unfair," "imbalanced," "artificial difficulty," and so on. They'll also make a lot of very specific claims about what they think are the exact balance tweaks that need to happen. Sometimes they're right, and maybe your game can actually be unfair or imbalanced, or maybe that one boss is actually too hard. But more often than not, these player frustrations are indicative of a deeper problem that's adding up invisibly and wearing down on the player until they quit or give up.

To figure out where these deeper problems are, you have to figure out what parts of the game the player is understanding and what parts of the game are poorly explained or too tough to understand. Once you've figured that out, you can attack those problems from any angle that fits with the soul of your game. You can improve or add more tutorialization, you can clean up overly complex UI elements, you can improve user experience and menu flow, you can switch around the placement of things on your HUD, and so on. If players are getting stuck on a particular mechanic or system, don't be afraid to remove it entirely and see what happens. Often removing a mechanic that you are in love with as a designer, though incredibly tough, can make you realize that it wasn't critical to the core of the game, and removing it makes your entire experience stronger and less confusing.

At the end of the day, you aren't going to be able to please everyone. Just try to incorporate feedback as best you can to end up with a level of difficulty that remains a core part of the soul of your game without being thoughtless, inaccessible, or obtuse.

To sum up: Design difficulty holistically, make sure it speaks to the soul of your game, try to understand players, and learn from feedback without letting it grind you down.

James Lantz is a designer and programmer with a strong passion for systems and mechanics. He has launched several small indie games, including the user-generated Mercury, and currently works at Klei Entertainment as a technical designer and core designer on the recently-released Invisible Inc.

THE PRINCETON REVIEW
#3
TOP GRADUATE SCHOOLS FOR VIDEO GAME DESIGN

MAKE GAMES WITH THE PROS

- Over **650 graduates** working at over **200 studios**
- Student portfolios include **3+ fully playable games**
- **Faculty with experience** at more than 40 top studios
- Earn a **Master of Interactive Technology** in 2 years

Your greatest achievement is one move away!
Apply for Spring 2016 at smu.edu/guildhall/applynow

Those who apply for Spring 2016 will have their application fee waived and get an expedited admission decision.

ART CREATION • LEVEL DESIGN • PRODUCTION • PROGRAMMING

SMU | GUILDHALL
GAME DEVELOPMENT EDUCATION

COLORING INSIDE THE LINES: USE LIMITATIONS TO MAKE GREAT ART.

BROAD RULES APPLY WHEN MAKING GAME ART, BUT THE BIGGEST ONE FOR MY MONEY IS THIS:

Games are built on constraints and limits. Whatever the art style or studio size, being aware of those limits will free you to be awesome. From triple-A titles to smaller indie games, all the artists I've polled have told me the same thing: Setting a cap to time and scope is a must.

Set limits within your reach

You can't break barriers or defy expectations without first defining those barriers and expectations. A limit gives you something attainable. Setting reasonable goals means you can accomplish your task.

I made all the art for the puzzle platformer Escape Goat 2, as well as some level and story design. Starting out with the genre of "puzzle platformer," I could already begin thinking about limits. First, the game would be 2D, presented from a side view. Then, I could start asking other questions: Is it close in or zoomed out? How big will characters be, and how detailed?

I also knew it's a game about a goat, and animating a quadruped is a challenge. Ian Stocker, the designer, wanted controlling the goat to feel like controlling Alucard from Castlevania: Symphony of the Night. One of the reasons Alucard is fun to control is because each movement has a unique animation. Turn left, turn left from a run, turn left from a crouch. So, how many animations does my character need? Run, turn, standing idle, jump, land. Limits are valuable because they allow you to focus on the big-picture tasks. For me, the goat was definitely one of these.

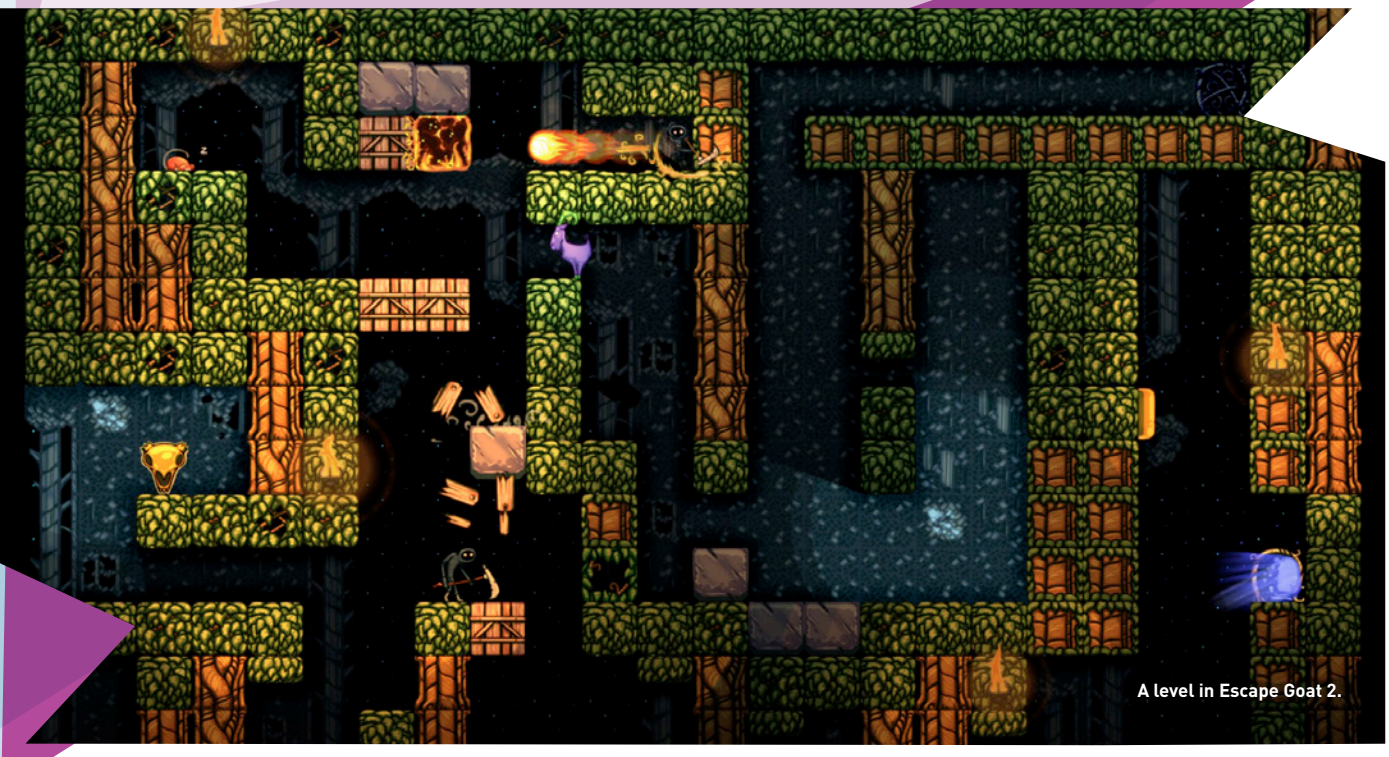
When you start a game, you may want to think big, but you should narrow your focus—and cut, cut, cut—as quickly as possible. Is your role to create the enemies for a dungeon crawler? Sketch 30 bad guys, choose three ideas, and work the hell out of them.

There's a tendency to want a huge number of enemies or characters—but wouldn't you rather create one kick-ass troll than 10 mediocre goblins? How many major triple-A games have more than five cool enemies that you can remember? Skyrim has a wonderful, huge world. And Bethesda, one of the larger companies out there, made around 30 unique creature models, with maybe half as many skeletons/rigs. It's quality over quantity!

Ultimately, it's important to set goals. You don't have to hit them all, and some you might surpass, but it's important just to have goals to strive to meet! It'll feel great when you do.

Game art is a tool

Most art is functional in a game. In Minecraft, every block has some purpose. Dirt mostly gets in the way, but even that is used for farming. Now think of stone, coal, wood, iron, and animals!



A level in Escape Goat 2.

Even pigs are functional. Imagine if pigs were hard to see—if they blended into the background, say. Suddenly, spotting and hunting animals would be a big challenge, and players would reap the rewards that come with those actions. That would change Minecraft as an experience. Art is as much about being pretty as it is about communicating function.

What does your art show players? What doesn't it show? You can aim to hide the function of your art, if you want. Does your explosive barrel look explosive? Do you want people to be surprised the first time a barrel blows up? After 50 times, will players just be annoyed when that boring-looking object explodes in their face once again?

As an artist, you may want to make an explosive barrel that's not red and obvious. But game art is a tool, so there is more to consider than just aesthetics.

Set standards

Knowing that game art is a tool, you should set standards—that is to say, the clearest and most common version of an object, which you can iterate throughout your game. This can be quite fun, and it makes work easier later. You want to communicate with the player, and if you show something twice, the player will start to see a pattern.

Spend time getting one version of fire working, for example. Then you can use the same colors, shapes, and styles again and again. In Escape Goat 2, I only had to make one collectible bounding object. Then I could just put anything inside, and players would know it was collectible. Or hopefully they eventually figured that out, anyway!

Set a standard piece of art for exits and doors or similar colors and shapes for dangerous things. This means less work later. You should use the same colors, animation styles, and shapes for the same concepts.

You should also limit the number of colors and textures. Design one method for creating normal maps, one method for lighting—maybe two if you're ambitious. I used one shade of blue/green for all shadows in Escape Goat. I used only two lighting colors. On the other hand, if you have busy art, that busy-ness should be consistent. Once you've developed a method for creating chaotic art, you can reuse that method. And then, when you decide to change the color of your lighting, players will take notice. When you take that flat texture and finally add some little detail in some corner, it will be more striking.

Games have a lot of art. You should do anything you can to reduce time spent thinking and working later on. This leads to my next point.

You don't need much variety to flesh out a space

We had a single 512 x 512 tileset for each region of the game, plus a "gadget" tileset for all the pieces (such as razors, ice blocks, and keys) that could be placed in any region.

Ian had a system in place for the original Escape

Goat in which there were four versions of blocks: basic, vertical, horizontal, and fancy. In Escape Goat 2, I believed the forest zone needed more vertical pieces for tree trunks, while the Egyptian-style zone should have more horizontal blocks. It was important to me that the forest tiles not be locked into the same rules every other tileset. Ian wanted the same rules for each tileset so he could swap levels into different zones easily.

As I worked on Escape Goat 2, I came to appreciate the rules and limits that Ian had set. By the end of the project, whenever I started a new tileset, I knew exactly how much art I needed. The game was manageable. The majority of elements in the game had only one or two versions of each piece of art. There was a fancy key and a basic key—a fancy candle, and a crude torch.

It's in your best interests to make art that you can reuse over and over. Every block should have a purpose. I left one space open on the tileset to have a fancy block for each region. I barely used these blocks in the end, and they ended up feeling like a waste of time and space.

Iterate

The reason you want to set limits and to make as little art as possible is iteration. With no limits, and 40 different characters, when are you done? If you set a reasonable limit, you can reach it, and then you either move on or you say your work is not good enough and double-down on it.

In Escape Goat 2 and Waking Mars (which I also did most of the art for), I cannot tell you how many iterations the first areas went through. Each had at least four complete re-dos and countless subtle changes within those do-overs.

In Escape Goat 2, I had 10 different environments to draw. A couple I was happy with immediately, but most needed overhauls. I redid the cathedral, first zone, and stained glass zones over and over.

I had to finish the goat once to know that he didn't look good. Then I drew him again. And again, and again. Now he looks pretty great, I think. I spent days working on one version of his animation, and it looked terrible compared to the final version.

External limits

Everything I've talked about up to this point has been self-imposed. We enforce these limits ourselves in order to save time and help us finish the game through iteration. But there are plenty of limits that are not put there by choice.

First, there are technical limitations. Learning to scale to your target devices can make a huge

How does a green pig in Minecraft change the user experience?

difference in the final look of your game. For example, Waking Mars was made for mobile. We had to make a game that would play on any iOS device all the way back to the iPhone 3s. The variety of screens and processors across these devices created huge issues we had to account for.

To optimize this process, I had to learn about draw calls. A draw call refers to every time the computer has to look for an image. The computer writes the image to the screen, finds the next, and then the next. I recommend learning how these work, especially if you're doing 2D games, and especially for mobile.

You should also learn about lighting. You might be able to show thousands of polygons on screen for mobile, or millions for console. But you'll then watch dynamic lighting take your frame rate to the floor. If your lights don't need to move—if your perspective is fixed—you can bake the light onto surfaces. Every time you ask the computer to do something involving lighting, you're adding a lot of work. Learn about static versus dynamic objects in Unity. If you don't need to move an object, the computer will be happy to save processing power.

Learn about resolution, as well. Unless you're doing vector art, you need to know how big your textures and images are going to show up. Don't spend tons of time on a 3D texture only to have it be the backside of a small prop. And it's probably a good idea to think about using vectors for anything that might be marketing material and will need to scale to a variety of sizes.

Explore your tools. Each tool is there to make your work look professional and polished. Find what works for you and then polish and refine how you work, but keep learning. My friend Amber Okamura uses the same two brushes for almost all of her concepts, because they make everything look fantastic. She often defaults to the same set of Photoshop layers, knowing it'll look good. Professional environment artist Jeff Parrot [environmentartist.com] defaults to Maya for modeling/UV mapping, 3D Coat/Photoshop for texture painting, and Marmoset Toolbag for presentation.

What have the big games done to limit their art?

Just Cause 2 uses red to draw you to destructible objects. The Walking Dead uses a very consistent texturing style. Look at screenshots for a game like Ori and the Blind Forest and note how many times you see the same spike art, tree, or bush. It reuses immense amounts of art, but it's gorgeous. Darkest Dungeon has a tight camera that focuses attention on character art, which was a big selling point. This also means the game doesn't need an abundance of environment art. Far Cry 4 has the same rock texture pretty much everywhere. Monument Valley uses the same shapes and flat shading with soft gradients, which means new patterns stand out.

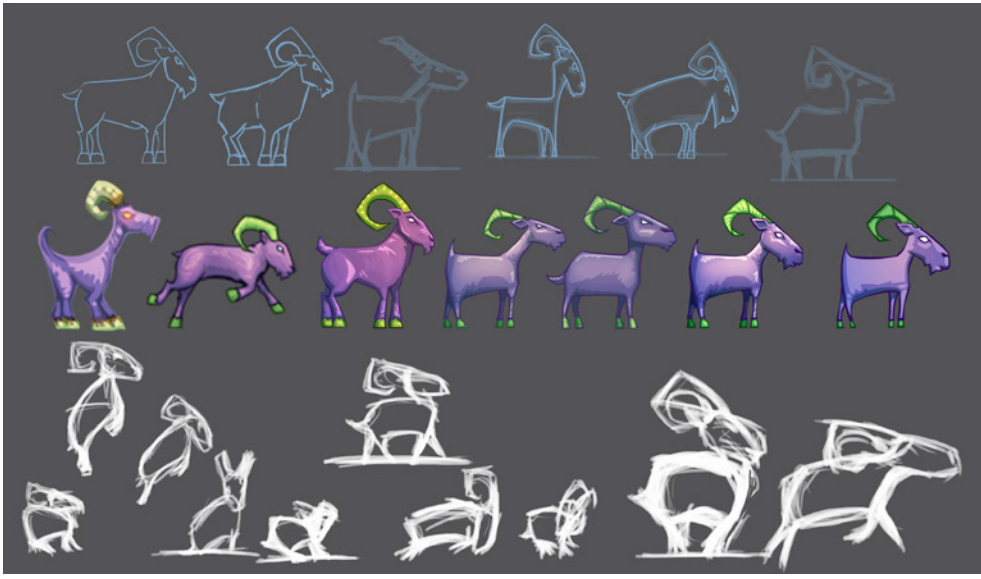
Skyrim has clear color palettes and consistent lighting. Most potions stand out on screen. Potions would never be such obnoxiously bright colors if they didn't have to draw the player's attention. If you're not convinced that the big teams limit themselves too, consider this: Skyrim's story is all about dragons. And yet, there's pretty much just one dragon with a few textures and heads. There's so much necessary art in that game that it'd be wasteful to do more.

Make what you love

Enjoy the process. Model, sketch, and design levels when you can. Make stuff you want to make. Do what you love when you're able, because when you start out, you'll be asked to emulate others. Unless you're hired for your art style (which is rare), the person paying you will want your work to look like something established. You'll gain experience by absorbing the techniques of those artists you're asked to emulate. Do they use line work? Is it sketchy? Are the models hard surfaces or organic? As you get good at this over time, people will begin to seek you out for your own style.

Do the work you say you're going to do

My number one tip for artists? Finish what you say you'll finish. If you promise something, do it. It doesn't have to be the greatest work, but it does have to be done. I was a relatively decent artist when I graduated college. But whenever I got a gig, I did the work. I wasn't amazing, but I was reliable. And I got more gigs because I had finished my work. And I got a little better at art with every gig I did. If you're reliable, you will have opportunities. Reliable artists get snatched up by good companies. A lot of people learn art as a hobby, as a personal form of



Color and animation tests for the goat in Escape Goat 2.

expression. If you want to have a career, you need to learn to make art as a craft—as a job. Finishing stuff is how you make a living.

And don't forget to be nice. Unless you're making a game by yourself, you'll have to work with a team. Nice people get invited back. You've got to be an incredible artist to get away with being a jerk. I mean that. A good partnership is more valuable than two incredible people who can't get along. You can clash, but you should admit when you're wrong and be willing to learn from everyone. Finish your stuff, and don't be a jerk about it. And you'll get rehired, and you'll get better. You'll make friends. And get rehired!

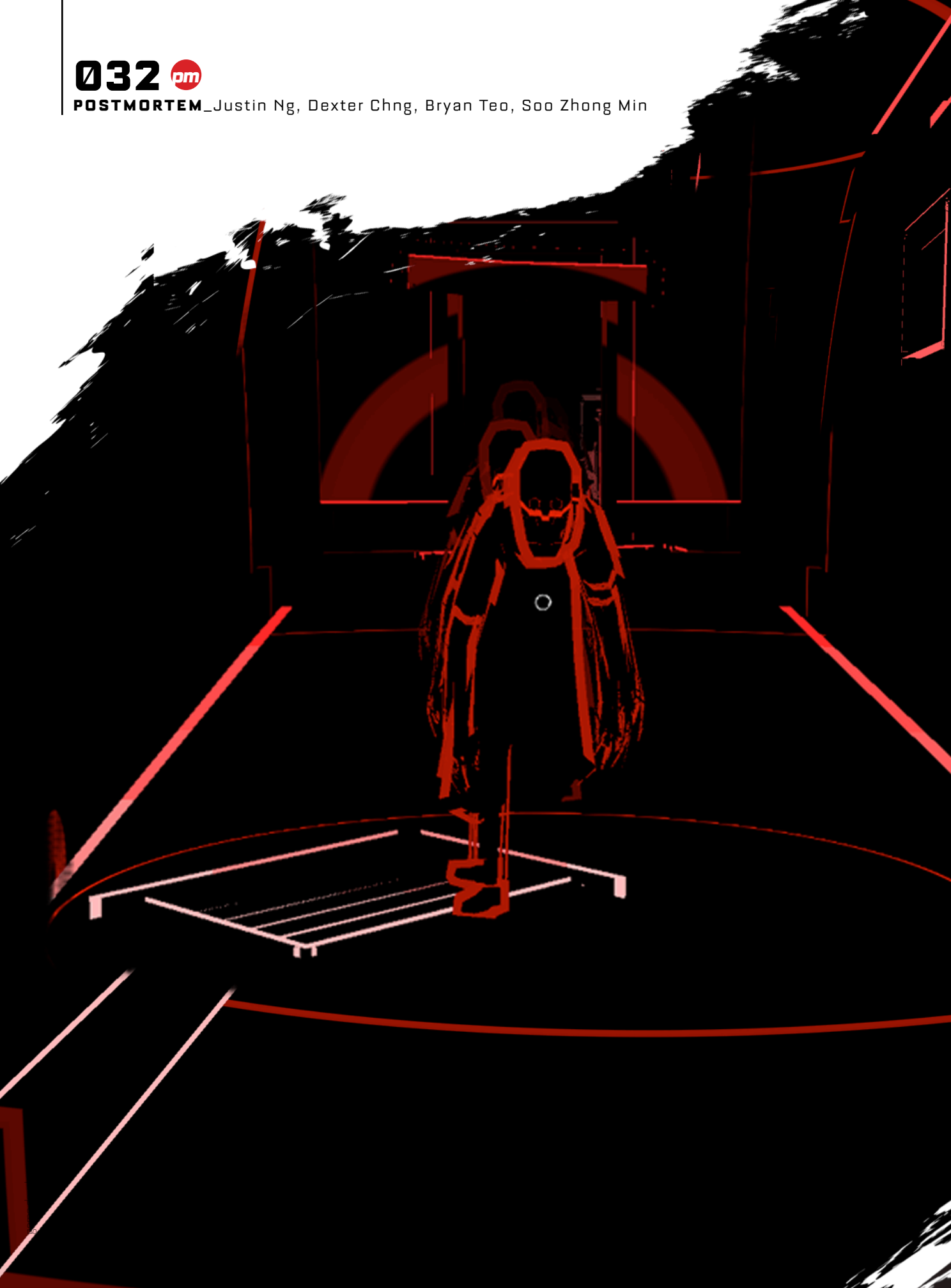
Escape velocity

Standards and limits are your friends. Aim small, set standards, and iterate. We want to see that killer character, that perfect user interface, that gorgeous sky temple. It works for me—Escape Goat 2 started small, I worked at it, I iterated, and then one day I realized I had drawn an entire game.

Randy OConner is an independent 2D and 3D artist and designer who has worked on games such as Waking Mars, Escape Goat 2, and his own game Dead End HD.

032 pm

POSTMORTEM_Justin Ng, Dexter Chng, Bryan Teo, Soo Zhong Min



LURKING

pm 033

POSTMORTEM

Justin Ng, Dexter Chng,
Bryan Teo, Soo Zhong Min

Lurking is a sound-based horror game in which sound creates pulses in an otherwise totally dark world. These pulses enable players to see outlines of the world around them. As the name suggests, there are enemies lurking in the darkness, and they are attracted to any sound made, creating a push and pull on players' desires to see and to remain unnoticed by the lurkers.

Lurking was the final major game project in our studies at Digipen Institute of Technology, Singapore, and we had two semesters (six months) to finish it,

034 pm

POSTMORTEM_Justin Ng, Dexter Chng, Bryan Teo, Soo Zhong Min

that we had little time to solve the many problems we faced, in both conceptualization and execution. We were very fortunate to be noticed and nominated for awards at Sense of Wonder Night (Tokyo Game Show 2014), IGF China 2014, and SXSW Gamer's Voice 2015. We received the Best Student Game and Excellence in Technology awards at IGF China 2014, for which we feel quite honored.

As with all game projects, there are things that we felt went well and others that could have gone better. Hopefully in sharing this with you, your game projects will go more smoothly!

WHAT WENT **RIGHT**

1. SOUND CORE CONCEPT Right from the start, we wanted to make a game that had something to do with sound generation as a primary interaction. This became the core of the game, and we sought to build everything else around that idea. We first established what the game would look like—how sound would generate visuals via pulses and what the world would look like when "lit."

At this point, we realized we got a lot of "free" tension simply because players couldn't see beyond their sound pulses, so we decided to go with horror as our genre. It was at this point that the game really began to take shape.. Making sound pulses is the only way players can navigate through the pitch-black environment, which generally makes them feel safer. This is juxtaposed with the idea that monsters lurk in the shadows, and they are attracted to the same sound players make to see. This makes players wary of making too much noise. The addition of microphone input that translates player-made sounds into the game works to further enhance this tension, making it a very immersive and unique experience.

What we learned: For what we were making, we didn't need to have everything figured out at the start. It is perfectly fine to start off with a simple core idea, and to explore and improve upon it.

2. THE ART We decided on the look of the game pretty early: Objects in the game would be represented by simple white outlines. Our intent was to create a stylized look that was both evocative of echolocation and simple to produce. Once our programmers figured out how to create the outlined look with shaders, we had a relatively easy time creating art assets for the game. As most 3D artists know, one of the most time-consuming parts of creating 3D assets is texturing. Thanks to the art style and shader technology, we didn't have to unwrap and texture most of our assets, saving us a large amount of time.

What we learned: Aim for an art style that best represents the feel of the game, but keep in mind the speed at which you can produce assets of a consistent quality. The overall aesthetic is always more important than graphical fidelity, especially for projects with tight time constraints.

3. USAGE OF UNITY For students like us, building our own game engine for a single game, though possible, wasn't necessarily the best use of our meager time. From the start, we decided to make use of a pre-existing game engine that was accessible and flexible enough for whatever game idea we decided to go with. We settled on using Unity, and, in hindsight, this was one of the few major decisions that set us up for success.

We only had four people in our team, two of whom were programmers. Without having to worry about building a full game engine, our two programmers could focus on building multiple iterations of systems during our prototyping phase, giving us a better idea of what worked and what didn't in a short amount of time. This was a lifesaver, as we went through multiple models of the game to figure out the best way to proceed with our core concept.

What we learned: If you don't have a concrete plan for the game in mind or much time for development, it is much faster to work with a pre-existing game engine.

4. GREAT SOURCE OF FEEDBACK Getting feedback is one of the most important parts of game development, and we were very fortunate to have access to a great source; our fellow designers and developers-to-be at Digipen Singapore. Through all the feedback garnered in play-test sessions and presentations, we were better able to fine-tune our game and improve upon it. There is much to be gained from multiple perspectives, and we all were able to learn from the testing process.

For example, we learned that every individual's idea of what makes a "good" game is vastly different, and it is folly to try to implement every single one of those ideas. We learned to pick out suggestions that were both feasible time-wise and consistent with our idea of the game.

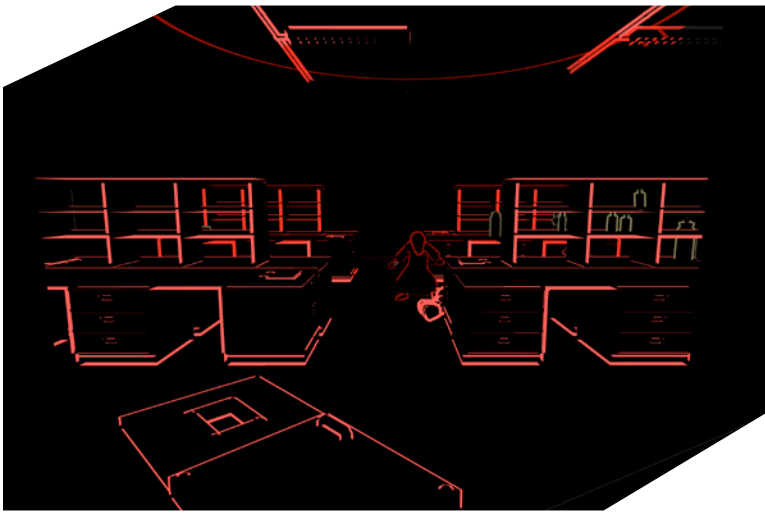
What we learned: Learn to take feedback(as much as possible), but never lose sight of your core idea.

5. GREAT TEAM It can never be understated how important it is to have a great team working together on a game. We were a team of four: two programming students and two design students. We knew each other's limits and capabilities, which allowed maximum creativity during the idea-generation phase while having fair expectations about the project at hand. We didn't always agree with each other, but we always sat down to talk through our thoughts. Everyone knew what they were working toward, and being on the same page led to a stronger game overall.

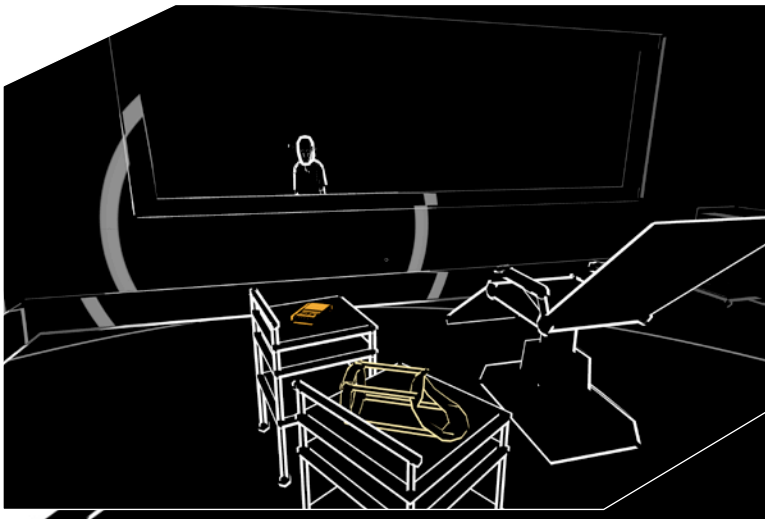
What we learned: Know your team, know yourself. Communicate, communicate, communicate!

WHAT WENT WRONG

1. TOO MUCH FEEDBACK Wait. What? Yes, this might sound a bit contradictory. Multiple times in our testing, we got feedback that there isn't much to do in the game besides throwing objects to distract the enemy. We spent countless hours brainstorming different gameplay methods and additional features for players to interact with, from adjusting pitch and tone to including puzzles in the game world. In the end we realized we didn't really have enough time for big changes. We were behind already, so we had to just leave it be. In that feedback process, we changed the pacing



Enemies attack when they hear sounds.



Each sound makes a radar-like pulse that shows the surrounding area.

Intro to Lurking.

036 pm

POSTMORTEM_Justin Ng, Dexter Chng, Bryan Teo, Soo Zhong Min

shouldn't have. A horror game is all about pacing, and we realized later that we should have focused on and fine-tuned that aspect of the game instead of padding the experience with other not-so-important gameplay extras.

What we learned: Be discerning of which ideas to implement when you get feedback. Never ever lose sight of what makes your game unique and interesting.

2. LACK OF PROPER PLANNING We spent the first three months building game assets without proper knowledge of their purpose. By the time we had a better idea of what we wanted the final game to be, we found ourselves with a large number of unusable assets and had to rework many from scratch. Granted, this was also a symptom of having milestones to meet for school, as we had to periodically show progress of our work. This hammered home the importance of not rushing straight into execution and having a clear plan before starting. Scrapping work is part and parcel of game development, but you can always plan ahead to reduce wasted work and time.

What we learned: Plan better and don't rush. Don't get too attached to your work, and be prepared to scrap a huge amount of it. We did.

The Lurking team.

3. THE HORROR OF HORROR We decided to make a horror game, but neither of our designers were familiar with the genre. We scrambled to do research by watching movies and playing horror games to familiarize ourselves with its elements and execution methods. Still, this may have been insufficient, and, had we had more beginning familiarity with the genre or more time for research, the game may have been brought closer to its full potential.

What we learned: Consider picking a genre you are familiar with. Do tons of research to figure out what makes games in your chosen genre work and how that is accomplished.

4. TUTORIAL Lurking has a number of game mechanics that are not immediately apparent to players. One is that crouching makes softer sounds, so enemies will not follow you if you are crouching and out of line of sight. In order to teach that, we created an area in which players are forced to crouch under a set of tables near a caged enemy. The intent was to show that the enemy stops following you while you are crouching. However, it ended up not working as well as we hoped it would—the crouching portion of the tutorial was pretty short and most players did not pick up on the mechanic.

What we learned: Test your tutorials on new players to make sure they work as intended.

RIGHT AROUND THE CORNER...

Lurking ultimately ended up as kind of a snapshot showcase of an idea, and it is meant to be taken as such. The horror aspects could have been better emphasized and better paced. With that in mind, we (three of our original four members) are currently working on a spiritual successor to the idea called Stifled (www.stifledgame.com), and we hope that Stifled will be a better representation of what we know the original game idea could be. If you are interested in Lurking, it can be downloaded for free via the links on the game's website (www.lurking-game.com).

Justin Ng, Dexter Chng, Bryan Teo, and Soo Zhong Min made Lurking while at the Digipen Institute of Technology Singapore. They are now working on the game's spiritual successor, Stifled.

Our diverse programs, and passionate, experienced professors will give you the skills to succeed. As a grad, you'll join a vibrant Sheridan alumni community who work as animators, filmmakers, designers and creative professionals around the world.

gamecareer.sheridancollege.ca

Sheridan | Get Creative

ADVANCING TECHNOLOGY STUDIES
Business Technology
Technology Leadership
Technology Studies

CYBER SECURITY
Network Security
Network Engineering
Technology Forensics
Information Assurance

DIGITAL ARTS
Digital Media
Digital Video
Virtual Modeling and Design
Web Design

GAME STUDIES
Game Art and Animation
Game Design
Game Production and Management
Game Programming
Serious Game and Simulation

HARDWARE CREATION
Digital Maker and Fabrication
Emerging Technologies
Human Computer Interaction
Robotics and Embedded Systems

PROGRAMMING
Advancing Computer Science
Artificial Life Programming
Enterprise Software Development
Open Source Technologies

GAME THE SYSTEM.

University of Advancing Technology
UAT
Learn. Experience. Innovate.

Please see www.uat.edu/fastfacts for the latest information about degree program performance and costs.

UAT.EDU/MAJORS

MONEY TROUBLE: AN INCREMENTAL FUNDING MODEL FOR SMALLER GAME DEVELOPERS

GETTING FUNDING IS ONE OF THE MOST DIFFICULT THINGS FOR ANY DEVELOPER, LARGE OR SMALL. UNLESS you're making games purely as a hobby, you'll need money to live, for software tools, and for licenses–and so will the people you work with.

While there's definitely no sure-fire way of getting money, I propose a relatively new model that gets publishers and platform holders to pay for you to make the games you want, while you keep your IP – you just may not get to finish your dream project in one fell swoop. It'll take time, and incremental steps, as you work gradually toward a final product.

But first, some backstory

The game industry is always changing. Once consoles were king in the U.S. and Japan, and PC ruled Europe. Now mobile is crown prince of Asia, and Steam makes up a lion's share of most independent game developers' revenues in the West. One of the bigger recent changes, as far as I'm concerned, is the rise of the independent developer. The "bedroom programmer" was common in the Amiga days, but small studios were deemed unsustainable for quite some time as console games got bigger, and teams needed to grow to match.

This meant the large teams, making games like your Call of Dutys or Assassin's Creeds, kept getting bigger. The mid-sized developers, most of whom were working on licensed or budget titles, wound up finding they had no business. After the Wii ceased to dominate, there was no more room for a mid-tier game.

But at the same time, indies rose up, as digital platforms increased in popularity. Suddenly it was possible to be a team of around 5, and make a game like Braid, Castle Crashers, or Minecraft, and do very well for yourself. That was around the early days of the Xbox 360 and PlayStation 3, in the mid-late 2000s.

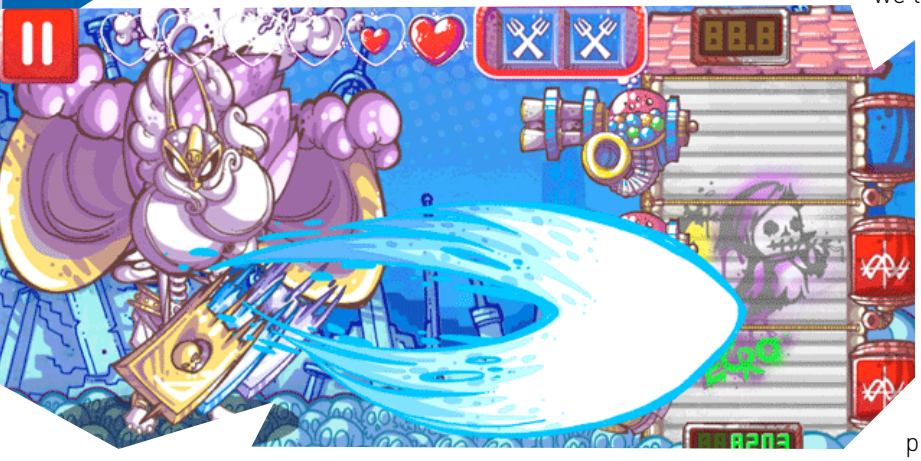
Now, indies seem to be everywhere, and we hear as many stories of abject failure as we do of smashing success. But there's a good side-effect of all this—right now indies are a bit of a buzzword. That's gross, I acknowledge that! But it can be used to your advantage. For one thing, platform holders, especially new ones, see the appeal of indie games, and want to capitalize on that. Think about the PlayStation 4 announcement at E3 a couple years back. They were considered to have "won" E3 in that particular year, in large part because they seemed so human and reasonable. Part of that was because they brought a lot of indies, for whom consumers have some sympathy, up on stage.

Platform holders also know that indies are relatively affordable, in terms of budgets, so they may be more interested in getting into funding, with the potential of a larger payout later. While you're not as likely to get big money out of Sony or Microsoft as you might have three years ago, newer, smaller platforms show up all the time, hoping to get their slice of the pie.

And then, on the other hand, you have the larger indies (like The Behemonth, indiefund, or Mojang), who now sometimes fund other indies, using the spoils of their earlier success. Likewise, smaller publishers like Devolver Digital have risen to publish and promote smaller indie games, to great success. It's these types of folks to whom you have to appeal, if you want to get funding in this day and age as a smaller developer.

A case study

I'll explain the model by way of example, using a game that we at Necrosoft Games managed to get funded twice (so far). Our first game, Gunhouse, was funded from its inception by Sony, for its fledgling (now defunct) PlayStation Mobile platform. We had prototyped Gunhouse as a game jam game, and Sony needed games quickly for its new platform, so it was a good fit.



We didn't exactly finish Gunhouse quickly, but that's another story—what matters is Sony funded the project, and we kept the IP, and were able to self-publish. While we didn't make revenue from the project, we now had a complete game that we could do whatever we wanted with. So, after the exclusivity period of two months ended, we brought the game to Windows Phone, through a grant program that Microsoft had at the time, called AppCampus. Personally I think we should get some sort of achievement for getting both Sony and Microsoft to fund the same game, but there's nobody to award it. Too bad. Maybe we should make ourselves one.

Anyway, we pitched Gunhouse to Microsoft as, essentially, Gunhouse 1.5. It would be called the same thing, but it would have a lot of improvements. We improved the store, and the weapon selection feature. We added enemies, stages, and feedback, such as screenshake. We improved the way the game plays in a bunch of ways, but not enough to call it a sequel. Essentially, it was like Microsoft paid for a version up patch. In the process, we also brought the project from Sony's PSM Studio SDK to Unity, meaning the game was now much more portable. We still got to keep the IP and self-publish.

Why would Microsoft do this? Well, at the time, Windows Phone also needed more games, and PlayStation Mobile wasn't a huge market, so they weren't concerned that the market for the game had already been exhausted.

Now, we're trying to get the "final" version of the game funded, by a third party, in which case we'll release the game on iOS, Android, and PC. But while we wait for that final round of funding, we've ported the existing version of the game to Amazon's phone and tablets. It's another small market that doesn't cannibalize potential future sales, because it's a relatively isolated platform, but one where the platform-holder can promise deck placement or marketing if you pitch them properly. Since the game is now in Unity, it only takes a couple days to port and integrate new platform features.

All of this is essentially the model I'm suggesting to you now. I'll sum it up – you should target smaller (funding) platforms first, and release the core of your game there. Not every feature you'd love to get in there, but the very core of the gameplay and featureset. From there, target additional small platforms to fund larger versions of the game, building on that core, assuring them that the market for your game is larger than the one you just released it to. Then once you have a version you're satisfied with, push that to the wider platforms – iOS, PC, Web, PlayStation 4, whatever is appropriate for your game.

In our case, we are also porting the existing version, which we still want to improve, to some other platforms as we anticipate more funding

for additional versions of the game. If you can get some marketing push behind your game, this can be worth it. But it should be noted – even if we don't ever get more money for the game, we can still pull the trigger and release the version we have on iOS, Android, and PC, because we own it. The game is good, we just want to do more with it. If that opportunity doesn't arise, we still have a full game we can release to an audience that hasn't seen it yet.

Funding in your sights

Now who should you target? Any new platform is a good potential source of funding. Not all of them fund, but many of them do. Some friends of mine kickstarted a game called XXX, which didn't make its funding goal, but caught the eye of Intel and their RealSense camera, which gave them $100k to continue development. I targeted PlayStation Mobile and Windows Phone, but others I know targetd the Nvidia Shield around the same time. If you had a game that showed off their 3D Tegra chip, they would potentially fund you.

Today, of course, the companies that are funding are different, but they're out there. Just do some research—what are the new platforms from companies with money, but who haven't necessarily been in the field before? That's where you look first. And look for platforms with large potential audiences, if you can't get funding. Spry Fox released a game on the original Kindle, before anyone else. But they knew lots of people had the device, so there was a high potential for return there.

So where do you find these people? You can look online, and send people cold emails, but if you have the opportunity, go somewhere like GDC or E3 and try to meet these folks in person. A face-to-face meeting really helps establish that you mean business.

After looking at platforms, you should look into publishers—from smaller ones like Devolver Digital and new companies like Raw Fury, to more established publishers such as Square Enix and 505 Games. These folks can also be contacted by email, but you should do your best to impress them. Publishers look for something with market viability within their area of expertise first and foremost, whereas platform holders may take a wider variety of games just to have a spread of content. So if you have some nice art, an interesting story, or can show an interesting vine of your prototype, and get some attention with it, you've got a better shot at catching someone's eye without having to do as much cold-calling.

Lastly, there are the indies that have gotten big. Our current game, Gunsport (no relation to Gunhouse—confusing naming convention, I know), is being funded by Iron Galaxy, which had success as a porting house for quite a while, til they broke into indie success with Divekick. Now they're making Killer Instinct Season 2, and are doing quite well. Divekick's success got them interested in doing some funding and publishing themselves, and now they've got three multiplayer indie games in their stable.

Developers are a bit tougher to get hold of, and they may in fact have to contact you. The Behemoth has a fund, but you can't petition them. The choice is all on their side. But if you can get them to notice your game, who knows! Indiefund is similar—they have scouts who look for interesting games to fund, so your best bet is to make a splash on the internet and hope to catch their eye. Going to Indiecade and talking to people is a pretty decent way to make this happen.

Talking to funders

If you're ready to give this a try, there's a bunch of stuff you should know first. For one thing, you should have something playable that demonstrates what is interesting about your game before you approach anyone. If you've just got hopes and dreams, with no gameplay, or visual or audio target, you'll be viewed as a much greater risk, and are less likely to get funded.

Now, when you're talking to folks about funding and the future of your game, I advise you to keep these things in mind:

1) Never give up your IP. It's all you've got, and if you let someone else own your game, you may as well do work for hire. It makes more money, and can sometimes be less stressful. But if you want to make something for yourself, you need to keep it. Plus, if you don't keep your IP, you can't sell an improved version of the game without extreme difficulty.

2) Ask for timed exclusivity, not permanent, from platform holders. Most platforms will want to keep your game forever. Unless that platform

is somehow completely ubiquitous (no platform is), or they're giving you an absolutely massive amount of money, this is not worth it. So you should make sure your game only needs to be exclusive to their platform for a set amount of time, after which you can bring your game wherever you want. And then, once you've got timed exclusivity locked down, ask for it to be shorter. In the case of Gunhouse, Sony had a standard 4 months of exclusivity for PSM. We asked for it to be two months, and they said "okay." That's it! Never hurts to ask.

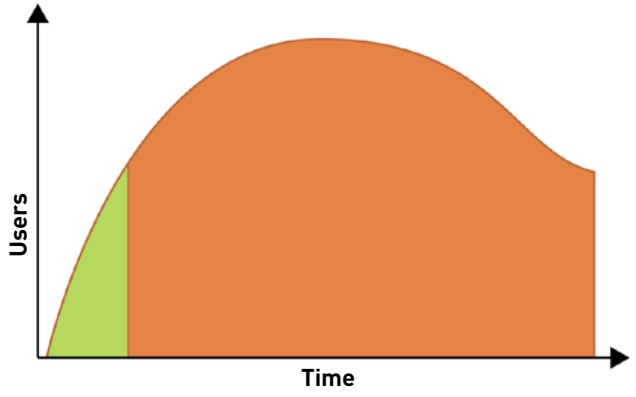
3) Ask for more money than you think you need, but not much more than others are getting. Game development is hard to predict, and your time estimates will probably turn out to be optimistic, and thus wrong. You'll need a bit more money. But if you can, try to find out how much games are getting in terms of funding from the same platform or publisher. Try to find this out from other developers, of course—the publisher won't tell you. But they may give you a range, and if they do, aim for the top. They'll make a counter offer, and then you can make a third that may please both of you.

4) Ask for higher revenueshare, in stages. Indiefund has a publicly established model for this, so there's a precedent. Indiefund gives you a certain amount of money, which they expect to recoup upon initial sales of the game. Once the game recoups, then the revenueshare split is 70/30, in your favor ←←CHECK THIS→→. If the game then doubles indiefund's investment, 100% of the revenue reverts to you. Now, most publishers won't give you 100% after you make double their investment, but they may well give you something like 90%. See what you can get in terms of revenueshare stages. This makes you play the long game, even if the publisher does a fire-and-forget release.

5) Seek out good/wide distribution partners. For some companies, like the aforementioned Spry Fox, distribution is more important than funding (whereas for me, at the current state of my company, funding is more important). As Spry Fox's Daniel Cook says, "A small trickle of steady revenue is more valuable [to us] long term than a burst of short-term revenue."

6) Try to get success early, one way or another, so that the platform holder cares about you once they move to an extraction model. At the beginning of a platform's lifecycle, the platform holder will be in an investment phase. That is to say, they know they need users, and for users they need game, so they will pay to have games on their platform so that they can get those users. But if that platform becomes successful, and they reach a sustainable number of users, now they don't want to give funding anymore, because they know that developers will want to release their games where the users are, regardless of funding.

This is another reason why you want to target new platforms early—but you should also try to get in good with them in the early stages, if you believe in the future of the platform, so they will still care about you when they make it big. If Ouya had made it big, for example, they'd still care about Towerfall, because Towerfall gave people a reason to talk about the Ouya at a critical time.



New platforms operate under an investment model (green) at firs, and then an extraction model (orange) once they reach a tipping point of users.

For our part, since Microsoft paid for Gunhouse's Windows Phone version, we felt we'd gotten enough out of it, and so released it for free, in the hopes that it'd make an impression on Microsoft for the future. I don't generally advise releasing your game for free, but, in that case we felt it would help. (It did help, but we screwed up our launch and had to re-launch later without promotion... but again that's another story.)

Sustainability

Not everyone can follow this model. It requires you to be very cheap, and complete games on budgets that hover around $50-$75,000 dollars,

or often less. We made the Windows Phone version of Gunhouse for less than 20k, and had some left over.

Essentially, you've got to save money wherever you can. Cook at home. Live as cheaply as you can. I personally have $400 per month for all food and entertainment, and I live in the Bay Area of California, one of the most expensive places in the US. Don't have an office if you really don't need it. Don't meet at coffee shops, unless they're extremely cheap. Offer more revenueshare versus up front money. People should be paid, but if you can promise more on the back end, you can work for cheaper. Eventually, all this scrimping and saving can pay off, because if you get a hit, your recoup will be so low that you'll just start making money on day one.

We at Necrosoft Games are a globally distributed team. This has logistical problems, but financially works out very well. Most of us are in eastern europe, which has a much lower cost of living. To work with us, you have to be able to live as cheap as us. We have a core group of people who are paid a monthly salary, and everyone else is a contractor – however, all contractors also get revenueshare or bonuses if they complete their full contract.

You also need to make sure you think about marketing, especially if you're working with a platform holder rather than a publisher. Nobody knows your game as well as you do, so taking vines and gifs, writing stories about your team, sharing information, all of this helps raise your company's profile. Pitch your game and your story to press, and try to get your game and your company name out there. Speaking engagements can sometimes lead to publishing deals, or new partners, as well, so leave all your options open. It's difficult to think about marketing when you're focusing on making your game, but in this day and age it's necessary to stay ahead of the crowd.

After you've made the game

Once you've released the thing, there's still more

you can and should do! This is when you try to get new platforms or publishers to pay for ports and upgrades. If the platform you've chosen has a weird SDK or tools, you'll want someone to pay you enough to where you can rewrite the game in Unity or C++, so that you can then bring it anywhere. And then once you can bring it anywhere (the "final" version of your game), do so, remembering to angle for marketing deals or deck placement by contacting the platform holders first. You never know where your game will hit.

Get into bundles once your game has kind of had a sunset in terms of sales. Bundles will give you much less money than you'd get from selling it normally, but once you're not selling much normally anymore, this can squeeze a bit more blood from the stone. This is another reason why you want to own your IP – if you don't, you'll have to talk to a publisher before you go into a bundle, or onto a new platform, and if they don't want to, well, that's that!

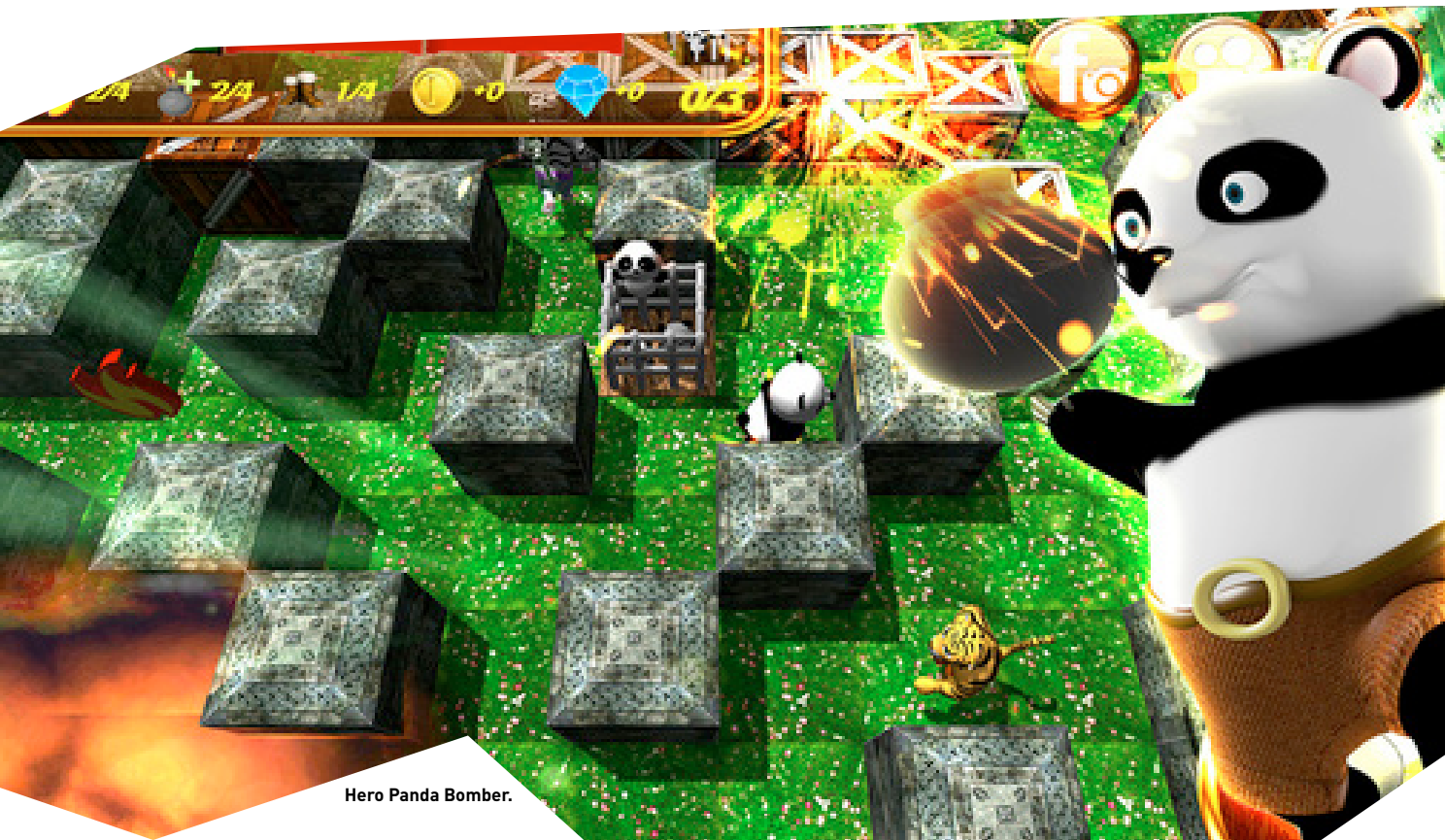
And again, make sure you give revenueshare or sales bonuses to your contractors. Of course you could burn out a bunch of college kids or friends, but that's not sustainable, and hurts the industry at large.

More examples

Here are a few more examples of this model, or variants thereof. Eliteapps in Bratislava Slovakia released a game called Hero Panda Bomber. It's basically a Bomberman clone in 3D. They did it as a proof of concept to see if they could make something nice in 3D for the Nvidia Shield. And it worked. While even they admit the concept is far from stellar, being essentially a clone, Nvidia needed games, saw theirs, and offered marketing. Once the game was marketed heavily on Nvidia's store, other android consoles with controllers started asking them to bring their game over, again in exchange for marketing.

While this wasn't a huge revenue blast, it was a nice trickle, and got the company some recognition in the industry, all because they targeted new platforms, and didn't port all at once.

In the second example, a company in the Czech Republic took my advice and walked around the show floor of GDC Next in Los Angeles, and pitched all the technology providers there. They wound up getting $25k to make a small game that



Hero Panda Bomber.

Listing 1: Companies that are funding or offering marketing support

Indie Fund - funding, eventually you get all the revenue. http://indie-fund.com

Devolver Digital - the whole indie package. http://www.devolverdigital.com

Double Fine Presents - publishing with revshare, not so much into funding. http://www.doublefine.com

Paradox Interactive https://www.paradoxplaza.com

Team 17 http://www.team17.com

Adult Swim - Small original games, you own IP. Small budget, so small teams are ideal. http://games.adultswim.com

Curve Digital - http://www.curve-studios.com

Chucklefish http://www.chucklefish.org

Finji - http://www.finjigames.com

Versus Evil http://vsevil.net

nkidu - http://www.nkidu.com – all about marketing.

Reverb http://reverbinc.com/triplexp

Mastertronic http://www.mastertronic.com

Nicalis - likes to adapt games to their engine. www.nicalis.com

Positech http://positech.co.uk

Surprise Attack – http://surpriseattackgames.com

Headup Games - http://www.headupgames.com

Wadjet Eye Games - http://www.wadjeteyegames.com

Evolve PR http://www.evolve-pr.com – marketing, obviously.

STEAKSTEAK - http://steaksteak.com

Kongregate - looking for longer term F2P partners. http://www.kongregate.com/

The Behemoth - funding some indies, though there's no way to petition. http://www.thebehemoth.com/

Raw Fury - new indie publisher out of Finland www.rawfury.com

Soedesco - new publisher from the Netherlands www.soedesco.com

Work for hire:

Nickelodeon - funds work for hire game based on its IP, and is looking to buy games with new IP. Reasonable budgets! http://www.nick.com/games/

Microsoft - Work for hire based on MS IP for Win10, bonus for cross-play.
(Some of these thanks to http://gambrinous.com/)

they were then going to use to supplement the development of their larger projects.

Lastly, there's someone who did the opposite. I spoke to a developer in Austria who was offered 50k Euro to make his game. The game was level-based, and was sort of a platformer. He said he didn't take the money because he couldn't make the full game for that much. But I pressed him—couldn't you make the core of your game for that much, though? And he said sure, but wouldn't that be cheating the platform holder? You could make an argument for it, but I say no. They want a game—any game. You should of course do your absolute best to make the best game possible. But if you give them a polished one hour experience instead of a three hour one, nobody is going to complain. Had he taken the money and made the core of his game with it, he would have been able to then get another round of funding to bring it to completion. As it was, the game remained unfunded, with money just coming out of his own pocket.

The power and danger of risk

Publishers are afraid of risk, but they want new ideas. This push and pull has existed in the game industry for its entire lifetime. But if you're cheap, you can take a risk that a publisher can bank on. Their only risk is a small amount of money, the risk is on you to recoup. I highly suggest trying risky, interesting titles with

Table 1: Know your chance of success.

	SUCESS RATE
Big companies with amazing development pipelines: They have more expensive pre-release processes, but releases are so well proven that they tend not to fail. Or if they do, they can turn them around (See Plants vs Zombies 2).	50% (if only because their private betas are as expensive as full launches for other groups)
Small professional team: These are the new midtier studios. Fan base, high quality product, clear monetization. Klei, Super Giant, Vlambeer (on mobile).	30-40%
Prolific indies: They make tons of small games, initially with no focus on making money. They get a ton of practice and when they try to make money, they at least have a competent game.	Maybe a 20% success rate?
Non-prolific indies: They make a few small games with no focus on making money. They may take 2 or 3 years per game.	1-2%
New entrepreneurs: They haven't made a lot of games, but their goal is to make money.	Maybe 5%
Students: They haven't made a lot of games and don't know how to make money.	←1%

this model, because it may be your best chance to explore them.

In my continued discussion with Daniel Cook about this subject, he shared with me his ideas for potential success rates of certain kinds of companies, saying that they should be aware of their chances of succeeding. These are rough guidelines, and should not be taken literally, but are interesting to think about.

Necrosoft Games is somewhere between the non-prolific indies, and the prolific indies. So let's say our chances of success are maybe 10-15% per game. But this model I propose is not about increasing your chance of success, directly. It's about lowering your chance of failure. If you're making games with someone else's money, and you budget to where you definitely can survive long enough to make that game, plus a bit longer, you've got no chance of failure. You can't go bankrupt unless you're spending your own money.

Put the fun in funding!?

To sum up, I say: do take risks, but don't do it with your own money. Do keep your IP, no matter what. And if you're interested in trying this model, do it now. The window is closing on publishers funding, but new sources of money come every couple months. New platforms launch. New publishers arise. Larger indies start funds. The money is out there, if you're hungry enough to take it. Just make sure you make a real good game along the way. Because if you're in this industry just for the money, you'll be better off making advertising software!

Brandon Sheffield is director of Necrosoft Games (follow www.twitter.com/necrosoftgames), senior contributing editor to Gamasutra, and advisor to multiple conferences and competitions. Currently working on Gunsport (among others) at www.gunsport.tv.

So You Want to Pitch a Publisher

Bryant Francis

You've begun work on your game. If you look to the north, you see the daunting technical challenges, creative barriers, and timeline to completion ahead of you. But you're ready for that. To the west and east, you have your friends, family, and teammates. And to the south—well, to the south, you still have one giant unanswered question: How can you get your game out into the world?

For some developers—and maybe you—the answer lies in the almighty publisher. Those great entities with fabled QA teams, marketing leads, and the ability to shuffle you through the technical requirements for your platform of choice. But how do you pitch your game to the publisher? How do you catch the attention of companies like Adult Swim Games, Devolver Digital, or 505 Entertainment, all known for supporting indie developers?

Luckily, at each of these companies, there's someone whose job it is to talk to you. At Adult Swim that person is Steve Gee, who says his process of discovering games is very similar to the path a game's potential audience takes. "My job in particular is to scour the internet, talk to my dev friends, and see what people are making out there, to find great games to publish across Steam, console, and mobile. I'm always on the internet, I'm scouring TIGsource and Kickstarter and whatnot. If I see something I like, I will email a developer, and go 'Hey! Your game looks freaking awesome. Let's chat!'"

Gee sits on Adult Swim's business development team, which means he and others like him in the industry are very focused on the pitching process. Over the course of a given week, he's looking for 10-20 games to take to his producing team. And your pitch could be the thing that gets your game on the table. At Adult Swim, that pitch can be as simple as a two-page document, but can be enhanced by a gameplay video, screenshots, or even a .gif. You have various options for presenting your idea. "It could be a playable, it could be a one- to two-page write-up," Gee says. "For a write-up, we'll want something describing what the game is, the game flow, the bullet-point list of features, and mockups of screenshots or art would be great."

And how does the company choose what to produce? "Adult Swim is obviously known for its bizarre sense of humor, but in the gaming space, it's not about being funny per se for us," Gee says. "It's about a great game different than what's

Death's Gambit is published by Adult Swim Games

Skylines is published by Paradox Interactive.

out there, and great style. Great games are different. You see them, you see a screenshot, you see a gif, you see a trailer, like 'Whoa, what is that? I want to know more.' Rain World, I saw a gif. Jazzpunk, I saw a trailer."

It's important to remember, though, that publishers aren't always looking for the same thing. Devolver Digital has a similarly laid-back process, according to producer Andrew Parson, but Nils Brolin, acquisition associate at Paradox Interactive uses structure based on their Game Pillars, and Justin Berenbaum, head of busines development at 505 Games needs to get his hands on a prototype to get the ball rolling.

"Generally we want to see a first playable, a prototype, a demo, something up and running," says 505's Berenbaum. "Pitching on paper is awfully difficult. As anyone who's been in the industry knows, it's really hard to convey your concept, your idea, what you're going for, on paper. Very rarely have I seen a paper pitch that made me go 'Oh, I want to sign that game, or I want to play that game.'"

If Berenbaum were to describe a pitch that took his breath away, he'd point to Three One Zero's Adrift, which 505 recently signed. "It was our last meeting on the last day of DICE [a game conference in Las Vegas]. We went into a meeting room and sat down with Adam [Orth] and a couple other people and they brought it up and started showing us the demo. At that time, the demo was just a prototype that was put together in three or four weeks, just so we have something to see and touch and feel. And it just caught us. The fact that our breathing changed just by playing it, we actually had to catch ourselves and realize we're not in space. It was things like that. A whole picture's worth a thousand words."

But you're not just pitching the game that comes out at the other end to a publisher, you're also pitching your ability to finish the game. Gee says this part of the process usually belongs to the VPs and producers on his team, but he still has some insight. He says,

> "When we do like a game, it does get to a process where we get farther along, and we bring our producers in, and we start talking to the team, and one of the first things we ask for is a mini game design document kind of outlining what the schedule will be. And our producers will assess, you know, 'Is this a realistic plan to produce this game?' But generally our production team will get to it and go, 'Cool, you know, they're asking for—they got these milestones spaced out, this is legit, this is realistic.'"

Andrew Parson is a producer at Devolver Digital.

Titan Souls is published by Devolver Digital.

Westerado is published by Adult Swim Games.

"That being said, at Adult Swim, we are—I don't want to say lenient, but we are very understanding that games do take time," Gee says. "Games will be done when they're done. We communicate often, on a daily if not a weekly basis. And we kind of know what their builds, what their progress is. We give developers time they need to make the game they need to make."

Brolin makes it clear that Paradox's objectives are best met by developers who can not only match their pillars, but who also have the capability to execute their vision and know how to put work beyond that first pitch. "If a developer has an outstanding game on their hands, the game will speak for itself. How the game is pitched is not really that important. The most important part is that it's a great game with great potential, and this potential is communicated in an understandable way. Most great games start with an idea that's later realized—but there's a difference between innovation and invention. The idea itself is not worth a lot, but it's the team and effort behind the idea that makes it into an actual invention—in our case, a game—in the end."

"Recently, Cities: Skylines proved to be an excellent example of a game hitting our Game Pillars Hard, and we were involved with the original concept being developed [and pitched]," he says.

But many publishers need to see something playable right away, as Devolver Digital's Parsons explains. "Gameplay is everything for Devolver, so the sooner you can get your hands on something, the better. People tend to get a bit wrapped up in the capabilities of their technology or fancy art, whereas really the most crucial thing should only ever be: 'Is this fun?' Often a prototype can answer that question in a matter of minutes. Prototypes shouldn't just be created as a pitching tool, though—they can help you iron out so many of your fundamental mechanics. I always think they're worth doing."

In 2015, for publishing mobile games, special consideration does need to be given to games based on alternative pricing plans. At a recent GDC talk, designer Ethan Levy made an argument for mobile designers working with free-to-play and in-app-purchase transaction systems to begin thinking about this design from day 1, not toward the end of their process. To build off that idea, Berenbaum explained that such business considerations are crucial when pitching at 505. "I think honestly that goes for just about any game," he says. "We're looking for developers that have some understanding of the business side. It makes our relationship with them much easier and smoother, because they understand where the money's going, what's happening, how economy of the video game world works. I think mobile has become such a crazy hard-to-predict beast, especially on the free-to-play in-game transaction side. If somebody who pitches a game that's F2P but they haven't done a F2P game before, that makes it a very tough pitch to accept. Just because that has become so much alchemy—art and science mixed. It's really hard to pull that off from scratch on your first game."

Gee's a bit more easygoing, mostly because Adult Swim hasn't shied away from the premium model even as IAP games buy multimillion-dollar Super Bowl ads. "We definitely discuss IAP, or whether it's right for the game," he says. "Obviously it depends on the game. If the game is a premium-type game, it is what it is. You

make the business model fit the game, as opposed to the other way around. You don't just shoehorn in-app purchases in if it doesn't really work in a game. So the first question is, what kind of game are we making? If it isn't a free-to-play game, then it's not.

"That being said, if we do do a free-to-play game, then we do discussions," he adds. "'Okay, it is free-to-play. How can we make money off of this?' It's gonna be in-app purchases, it's gonna be ad-supported, it's a mixture of both, whatever. IAPs—are they consumables, are they durables, and all that stuff. But first things first, what is the game, and is it a good game that we want to play? The game dictates the business model for us. Our titles lately have been free-to-play, but that's not to say we wouldn't do another premium title. There are some that we're actually considering."

From there, the long road to shipping begins. If you're looking for what absolutely not to do, Berenbaum has one bit of advice that keeps him and his team from running for the hills: "One of the things that will make us run away faster than anything else is when someone says 'This is for everybody!' So we really like devs to have an idea. 'This is who I'm going for, this is the audience,' and at 505, we're not afraid of a niche. We like what we call defendable niches. Sniper Elite was a defendable niche. It was a subset of a shooter genre, very hardcore, but one of those things where a super-big publisher would have a hard time going at that market because it's not going to sell 10 million units. But it was a big investment. Brothers is one of those games, again. How do you explain that game to a consumer in a sound bite? You can't. It's a really hard game to explain to people until they sit down and play it. We like those experiences."

And if you plan to pitch to Gee, remember this story: "The guys at Rain World—on that one, I saw a TIGsource forum post, and I saw some gifs, and I was like 'Oh my god, what is this? What is this game?' Eager to learn more, Gee scoured the internet—but was unable to find the developer's contact information. But it turned out okay in the end, he says. "Eventually they had a Kickstarter page, and so finally I could talk to someone. I found James [Q: last name?], we chatted, and we signed them to a deal—but #1 thing: Please have an email."

Bryant Francis is a freelance writer for a variety of publications.

TWITCH STREAMS: THE NEW MEDIA BLITZ

ON SEPTEMBER 8, 2014, TINYBUILD'S FLAGSHIP TITLE SPEEDRUNNERS RECORDED ITS BEST SALES DAY EVER. This spike wasn't due to the usual suspects, like a Steam sale or flash deal, or a big YouTuber putting out a video that featured the game. Rather, the game's biggest sales spike came from a Twitch livestreaming partnership.

In conjunction with internet celebrity PewDiePie and eSports company Gfinity, the SpeedRunners PewDiePie Challenge saw players buying the game in spades, simply so they could play against "Pewds" via his Twitch livestream.

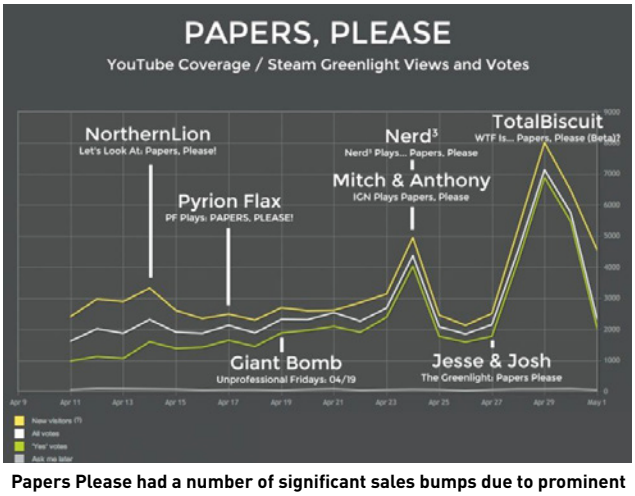
I'm sure you're well aware of Twitch already, but I'll describe it just in case: Twitch lets people livestream themselves playing games online from their PC, Xbox One, PlayStation 4, and other devices. Other players can watch these livestreamers, comment on the action taking place, and subscribe to streamers if they deem the content interesting enough.

It's a movement that many game industry observers are still trying to wrap their heads around. The idea that tens of millions of players want to simply sit and watch other people play games for hours on end may sound absurd, yet sit and watch they do—and in droves. At last count, Twitch now regularly receives a million concurrent viewers and gets more than 100 million unique viewers every month.

Livestreaming complements the YouTuber uprising perfectly. Fans of big-name video game YouTubers, like the aforementioned PewDiePie, can more easily interact with their heroes on Twitch and feel like they're taking part in a spectacle. Certain games are feeling the effects of the Twitch boom already: League of Legends, The Binding of Isaac, Hearthstone, Counter-Strike: Global Offensive and others can attribute a good portion of their ongoing success to the sheer number of people streaming these games.

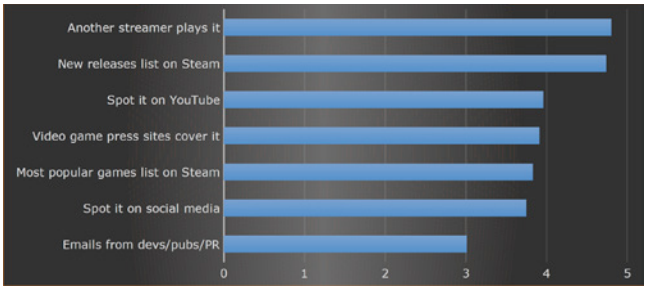
More and more big names on YouTube are taking to Twitch too, such as the highly lauded Geek & Sundry network (who, in fact, started their very first livestream by playing SpeedRunners!), and the movement is bound to grow stronger thanks to eSports, celebrities, and big Twitch presence at high-profile video game events around the world. Twitch is even planning its own conference, slated to take place for the first time in September 2015.

To ignore this stampede would be careless for any game developer, especially for smaller independent studios trying to keep their heads above the huge wave of indie titles crashing against game industry shores right now. At tinyBuild, our focus on Twitch streamers has proven most fruitful during the last year. We have been featured regularly on Twitch's front page, sometimes with hundreds of thousands watching.

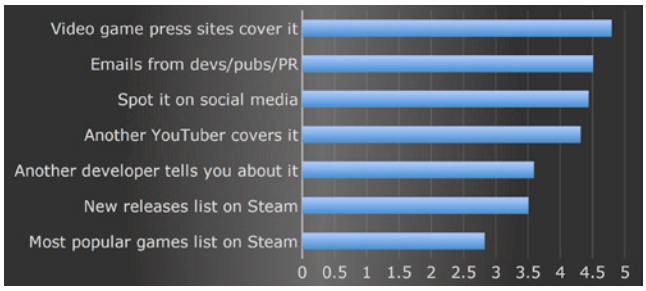


Papers Please had a number of significant sales bumps due to prominent streamers.

But how do Twitch livestreamers choose the games they play? And how do you get them to latch onto your game? That's what I mean to tell you here. I'll also explore how Twitch streamers differ and agree with their fellow YouTubers in their broadcasts—oh, and I'll make your life a guaranteed thousand percent easier by providing a handy list of contact information for hundreds of livestreamers.



Where Livestreamers learn about new games.



Where Youtubers learn about new games.

As a general note, all of the information about the games press, YouTubers, and Twitch streamers you'll find in this article comes directly from the horses' mouths. I surveyed hundreds of people from each group to discover what makes them tick, and the following data is the result of my digging through the mass of submissions I received.

The chain of discovery

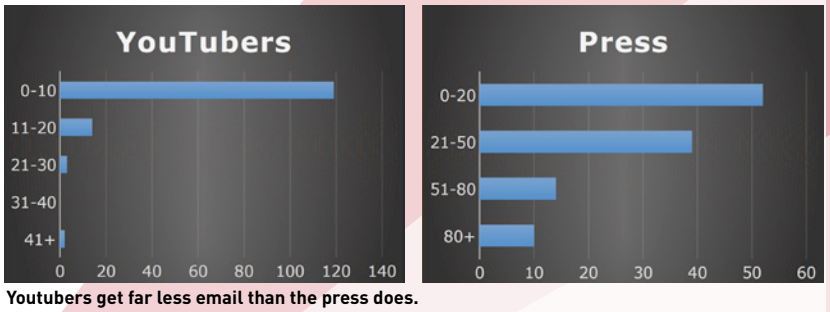
If you were hoping to skip straight past traditional written press and fire your press releases directly at YouTubers and Twitch livestreamers without bothering with the likes of Kotaku and Gamasutra, you might want to reconsider. While it's true that these massive video celebrities can bring more traffic to your game than most written news sites combined, you should be asking where exactly these people are finding the games they play.

The answer, as it turns out, comes in the form of a chain of sorts. YouTubers, for the most part, are choosing to make videos of games that read about, as well as games they see other YouTubers playing. Meanwhile, Twitch streamers are mainly finding their games from—you guessed it—YouTubers and other livestreamers. As such, getting traction in the written press is still as important as it has ever been, just for different reasons.

It's notable that the vast majority of both YouTubers and livestreamers state they rarely play games they receive via email. At first glance this might suggest that it's not worth emailing them, but digging deeper into survey responses reveals that these people rarely play games they receive via email because they barely receive email in the first place. Roughly 85 percent of the livestreamers I surveyed told me that they received no more than 10 emails a day, and I'd wager most receive none. Even the biggest livestreamers are receiving a maximum of 20 emails a day, maybe 30 at a push. This compares to the 30-a-day average that journalists and critics receive. It's fair to say that if you email a Twitch streamer, they're much more likely to spot your email.

The problem is, well, there's a reason Twitch streamers don't receive emails—it's nigh on impossible to find contact information for them. There's no directory of Twitch streamers, no email addresses listed on the sidebar of Twitch profiles. Contacting livestreamers involves a lot of tedious back-and-forth via Twitch messaging, hoping they spot your message in the first place, and then hoping they want to fire a message back, preferably with a contact email attached.

But you don't want to go through that slog, right? You want to be making video games, and rightly so. That's why I've gone through the hours, days, and weeks of slog for you: If you visit tinybuild.com/twitch, you'll find a full list of contact information for hundreds of livestreamers, all for free, updated on a weekly basis. You're welcome!



Youtubers get far less email than the press does.

What do livestreamers want?

So if email is your "in" to the world of livestreamers, how do you make a good impression and get streamers on your side?

A lot of Twitch streamers are very particular about their craft and care a great deal about whether the people soliciting them genuinely know their work or are just feeding them bull to get some free marketing. What this means is that before you even begin to contact livestreamers, you need to become incredibly familiar with the Twitch platform and how its users work.

Many streamers want to make a career of livestreaming, and they're putting their all into it, streaming eight hours daily and building an audience through their personality. If you take the time to learn how many of these individual personalities tick, it will stand you in much better stead.

Livestreamers are looking for very specific things when it comes to working with developers. Everything they do needs to expand their audience. As such, apart from giving them the standard—a code to download your game and a link to a YouTube video that shows how it looks in motion—you'll also want to be thinking about the social aspect of it.

Code giveaways are a notable way that livestreamers grow their subscriber count, so if you can provide a bunch of codes for them to throw at their viewers while they play your game, there's far more chance they're going to take you up on that offer. Meanwhile, a developer who can provide an additional boost to a Twitch streamer's audience, whether it's through sharing a link to their stream via Twitter, Facebook, Twitch, or a newsletter blast, is a developer that they are more likely to want to work with.

What livestreamers want

What Youtubers want.

Many Twitch streamers also love to show their audience something new, maybe even exclusive, that their viewers aren't going to be able to see anywhere else. In the run-up to the launch of your game, you may want to consider sending special preview builds to specific livestreamers—or simply make sure that streamers have codes for your game days in advance of the launch day, so they can go out all guns blazing as your game drops on Steam.

And you should also consider...

It might sound obvious, but video recording and streaming issues are a major factor in whether or not a streamer or YouTuber chooses to play your game. If they have problems actually putting your game on their platform, or it's simply a hassle, they're just not going to bother.

The solution is simple: Make sure your game works in every goddamn piece of recording and streaming software available. Open Broadcaster Software, Bandicam, XSplit, Nvidia ShadowPlay, Fraps, Dxtory... there are numerous programs available, and it'll be a pain to get them all downloaded and tested properly. But let's be honest—it can't be any worse than, say, console localization, right?

If you're developing a PC or console game, working with livestreamers is going to be far easier than pouring your heart and soul into a mobile game. Around 90 percent of livestreamers have no plans to cover mobile games—mainly due to how difficult it is to stream mobile games because of hardware requirements and technical issues. Many Twitch streamers also appear to be of the opinion that the majority of mobile games simply aren't worth streaming.

But that's not to say that mobile game devs should stop reading now and give up the chase. There are still plenty of livestreamers who do play mobile games if you can just track them down, and there are a number of mobile games that have seen success on Twitch. Search for Clash of Clans on Twitch, for example, and you'll find that there's always at least a dozen or so people streaming the popular Supercell game.

If your mobile game is cross-platform and already has a PC version that's available to stream, you can use this to your advantage as well—sending out Steam codes to Twitch streamers on the day that your mobile version launches can potentially cause some cross-promotion for both versions.

Getting involved with Twitch directly

Building relationships with Twitch streamers is one avenue of attack when it comes to approaching the behemoth platform, but you may want to also consider talking to the Twitch team directly and building a name for yourself and your studio.

We've been Twitch fans for a while now, and we're constantly in talks with Twitch regarding online and physical events that we can get involved with. At PAX East this year, we were granted our own hour-long show on the front page of Twitch, with tens of thousands of people watching, to announce two new games we're publishing. The company has now asked us to be involved with their front page livestreams direct from E3, PAX Prime, and more, simply because we're showing so much interest and faith in the movement.

There are numerous other game studios who are also getting in deep with Twitch, but still plenty who aren't yet exploring the possibilities. Positioning yourself as a developer worth featuring on the front page, in full view of hundreds of thousands of people on a daily basis, is clearly advisable.

It's especially advisable if you plan to lean heavily on eSports. Twitch has been a godsend for eSports, allowing players to follow tournaments and their favorite players from various online games with much greater ease. More and more big-name titles now have Twitch streaming built directly into them, and it appears to be proving rather fruitful for these titles.

Amazon's stamp on the future of Twitch

Twitch was acquired by Amazon last year for nearly $1 billion, and, as you might expect, Amazon is wasting no time exploring ways to make the platform better.

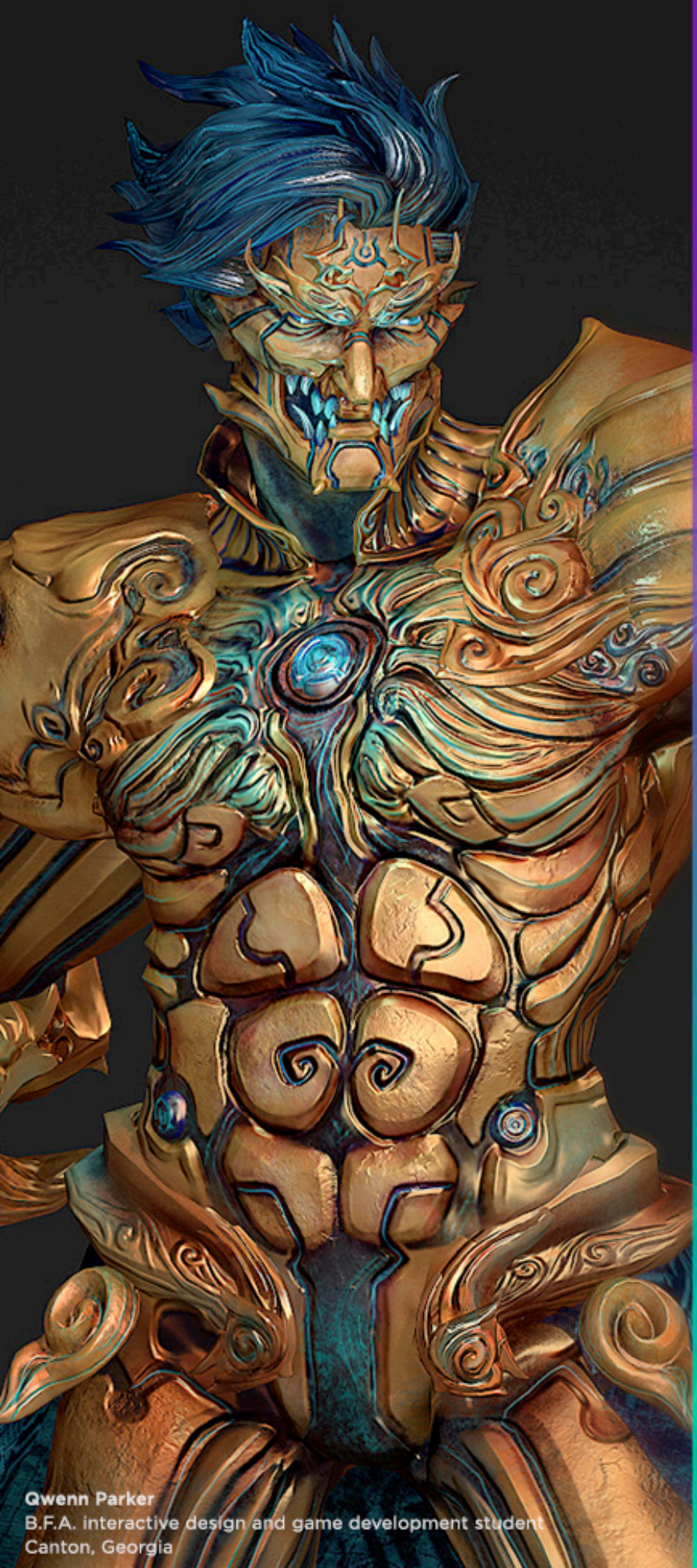
For example, discovery on Twitch, as with every game platform under the sun, is a big issue, particularly for developers looking to work with streamers. As mentioned previously, there is no directory of streamers, or indeed any real way to contact them outside of my own list. I recently chatted with one of the Amazon directors about this very problem, and Amazon is currently exploring different solutions to this issue, a number of which sound rather promising.

In any case, Amazon's involvement with the platform can only signal that being involved with livestreaming is going to be highly necessary in the coming years—and now, with Google reportedly working on their own video game streaming platform to rival Twitch, the space is only going to become more lively.

Twitch to life

It's clear that Twitch can't be ignored as a discovery, marketing, and even playtesting vehicle for your games. But you can't just fire and forget—you've got to carefully plot your plan of attack and put in the necessary time to make it work. I never said it was going to be easy!

Mike Rose was previously a game critic of eight years for outlets like Gamasutra, Kotaku, and Pocket Gamer. These days he's the resident British guy at developer and publishing house tinyBuild, where he spends his days finding new games to publish.



Qwenn Parker
B.F.A. interactive design and game development student
Canton, Georgia

INTERACTIVE DESIGN AND GAME DEVELOPMENT

Launch your creative career as a professional:

- Game designer
- Concept artist
- Online game developer/designer
- Interface designer
- Modeler/texture artist
- Level designer
- Web designer
- Environmental artist
- UI/interface designer

SCAD
The University for Creative Careers

LEARN MORE ›

MODERN 2D TECHNIQUES IN UNITY FOR INDIE TEAMS

Benoit Fouletier

THERE ARE MANY WAYS TO MAKE 2D GAMES THESE DAYS, FROM 8-BIT-STYLE PIXEL ART (SHOVEL KNIGHT, FOR EXAMPLE) TO FULL 3D WITH 2D GAMEPLAY (DONKEY KONG COUNTRY RETURNS) AND EVERYTHING IN BETWEEN.

My company, Swing Swing Submarine, has made two games so far, both in 2D. Blocks That Matter was made in six months by the two founders, more or less from scratch in XNA. Its spiritual sequel, Tetrobot and Co., involved four people and was made over about a year using Unity. As the titles suggest, both games are about blocks, and they use orthographic rendering of isometric sprites.

Visually, our upcoming game Seasons After Fall is much more ambitious. We're aiming for lush graphics and a very organic feel, both in the world's ambiance and the level design. We're giving everything lots of curves. So we knew old-school blocks and tiles were out, and for various reasons (both creative and pragmatic ones, e.g., our team's skills), we knew we didn't want any 3D elements. The production itself is, for us, quite ambitious?we're about 15 months in with nine months left (fingers crossed), and this time, our team

A scene from Seasons After Fall in the editor (looks better without the GIF compression!)

consists of six people: one artist, one animator, one musician, one designer, and two programmers.

THE STATE OF 2D

At my previous job at Ubisoft Montpellier, I worked on Rayman Origins and Rayman Legends, for which we developed the UbiArt Framework that was also later used to create Child of Light and Valiant Hearts. Other recent examples of what I vaguely call the "modern 2D" movement are Broken Age, Ori and the Blind Forest, and many more. What these games have in common is they use "real" (flat) 2D, but they are rendered in perspective in a 3D world using a 3D engine, and they usually play a lot with depth, which instills life into the environments.

Another characteristic of these games is that they tend to require a lot of high-res textures, which usually takes a lot of manpower to produce. And manpower is what we're lacking (six guys, remember?). To make things worse, the premise of Seasons After Fall is that you play as a fox that is given the power to change seasons. So all our environments can be played in any of the four seasons...which means every asset in the game has to have four painted versions! That's just ridiculous, and we must be out of our minds.

Having only one artist means we have to use highly reusable assets. Having only one animator means we have to create an animated character (what we call a rig or puppet) as simply and quickly as possible.

All in all, we knew we had to build a significant amount of tech to get the job done. Our tools should be focused on level design, with instant iteration loops. We put level design at the forefront, automating level art as much as possible while still retaining authoring freedom. We need to be able to modify gameplay situations as late as possible in production, when the finishing touches of level art finally lock in.

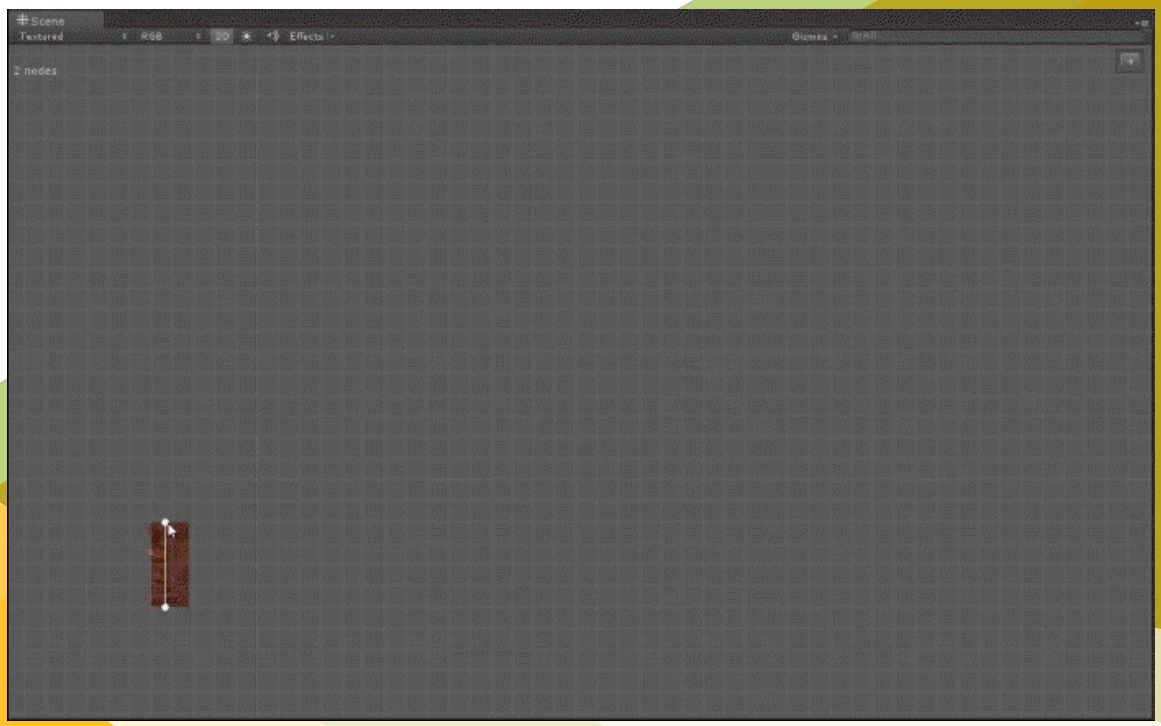
If there's one area in which our artists excel, it's Photoshop. So instead of spending time rigging a 3D model in Maya or Max, the idea was to work as much as possible in Photoshop, export all the metadata we can, and have tools that can transform it into assets that are ready for level design, level art, and animation with as little friction as possible.

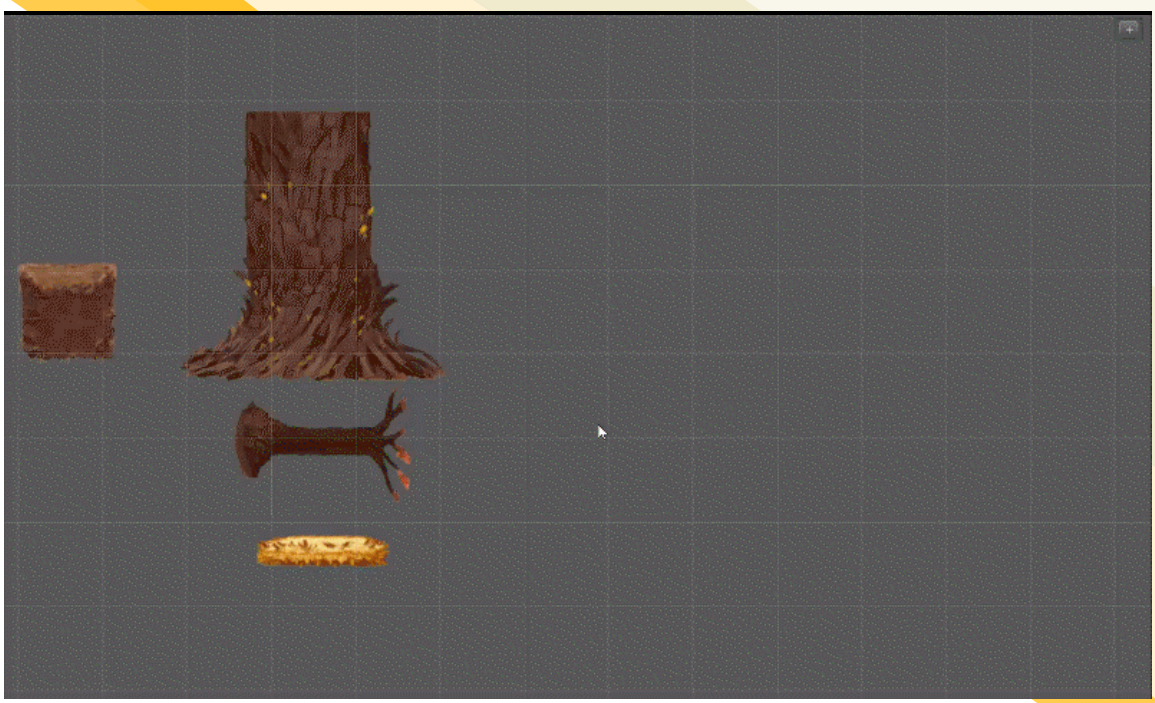
CARPETS AND RIBBONS

One of the first tools we worked on is very much inspired by the UbiArt Framework. It's a mix of vector graphics and textured graphicsÐa kind of smart texture that can be stretched along a path (straight lines and/or splines) to create level design or level art by the kilometer, all from a single source texture.

We have two categories here: carpets and ribbons. Carpets are used to cover surfaces and for closed shapes like terrain, water, fog, or even lighting (which, unfortunately, I won't have time to get into here). Ribbons are used for linear, open shapes, like crawling vines, pipes, trees/branches, and so forth.

Authoring these smart textures is a bit like creating a tileset, albeit a little more tricky but once you have them, you get a lot of bang for your buck! For


Scene
Textured
RGB
2D
Effects
Gizmos
2 nodes

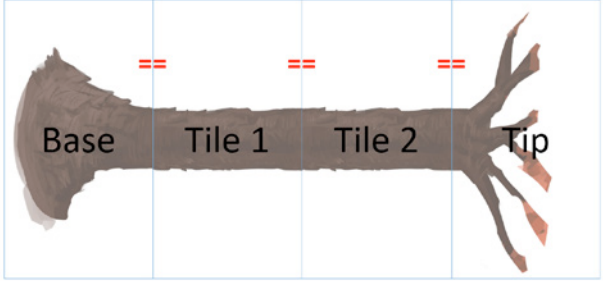


example, with just three elements—trunk, branch, and foliage—you can draw a lot of diverse trees.

PRETTY RIBBONS

So how do we make these assets? Let's start with the ribbon, because it's simpler. Your texture is divided into three parts: a base, a tiling part, and a tip (base and tip are actually optional; it depends on your design).

Also optionally, you can subdivide the tiling part, to get more variety.

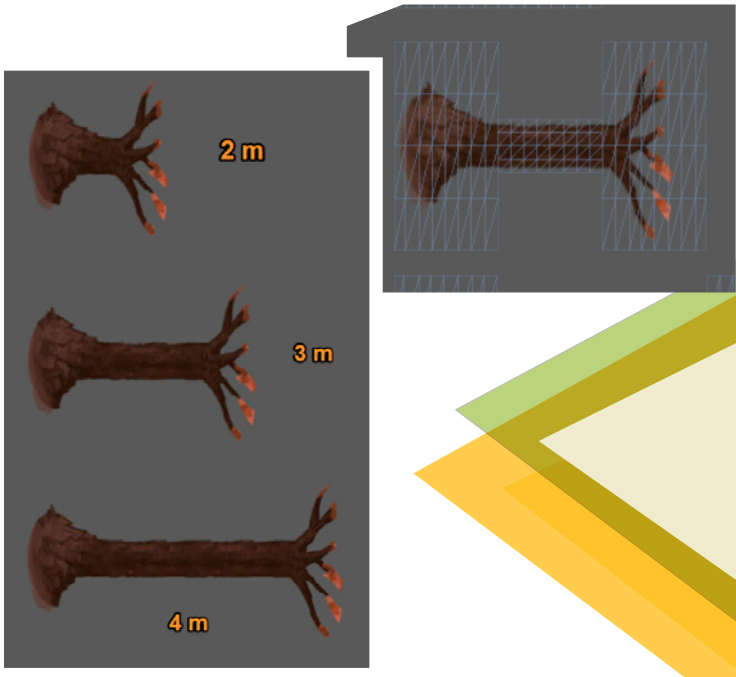


Here we have two subtiles, but you could have four, or eight, or however many you feel like painting. But, importantly, all the edges with the red "equals" sign must match: The base matches with the left side of the tiling part, obviously, but also with the left-middle side, and with the tip. And the tip matches with the right side of the tiling part, and the right-middle part, and with the tip.

Now when you extrude that texture along a spline, you still want the base and tip to match, so you need an integer number of repetitions of the tile (you don't have that constraint if you don't have a base or a tip).

So if each part is one meter long, the minimal combination you can have is the base and tip alone (two meters long), or a branch with only "tile 1" (three meters long), or with both "tile 1" and "tile 2" (four meters long). Obviously, then, you can unroll as much as you want. So we tile by increments of one meter, and if the length is in between, it will simply stretch, rounding to the nearest integer.

If you look at the texture, you'll notice that the middle (tiling) part has more white space, so we can trim the geometry.

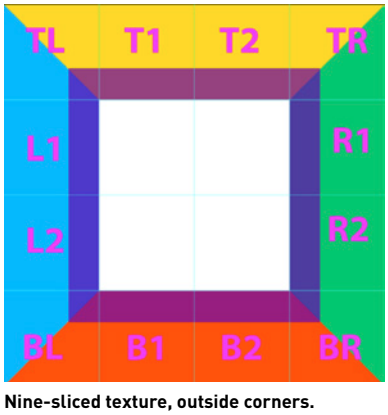


We still have wasted texture space, but at least it's less overdraw.

CARPET 101

Now, on to the carpets! They're a bit trickier. They are made up of two parts. The first is a filling texture (the inside of the surface), which is just a regular tiling texture that tiles on both axes. Nothing fancy here. The other part is the outline, which is quite similar to the ribbon. We have several variants of these depending on the design, but the most typical (and complex) variant is to have a different outline for the ground, walls, and ceiling, with nice corners to link them all together.

This is similar to a nine-sliced texture, which is what most UI packages use to make (for example) a stretchable button with rounded corners. And, like we did with the ribbon, we can also get fancy and subdivide the tiling part to get more variety. And now... behold the awesome programmer art debug texture!



Nine-sliced texture, outside corners.

VANCOUVER CANADA'S MASTER OF DIGITAL MEDIA PROGRAM

MDM Graduate Students Benefit From:

Learning in a Top Video Game Hub

Vancouver has one of the top Video Game clusters in the world. Companies such as EA, Hothead Games, Sega and Microsoft all have studios in Vancouver.

Startup Business Support

The MDM program has strong roots in Vancouver's startup culture. You have the option to do a Venture Internship which mimics the first year of starting a business.

Competitive Scholarship Opportunities

In the September 2014 intake, **+60%** of students received scholarships or other funding assistance.

As a Master of Digital Media graduate student, you will develop the essential professional skills for a career in the Games industry. **Fast track to become a Video Game Producer, Development Director, Producer, Project Manager or Entrepreneur.**

> I cannot stress enough how the lessons we learned at MDM gave us the tools to become successful leaders in a company as exciting and innovative as Hothead Games.

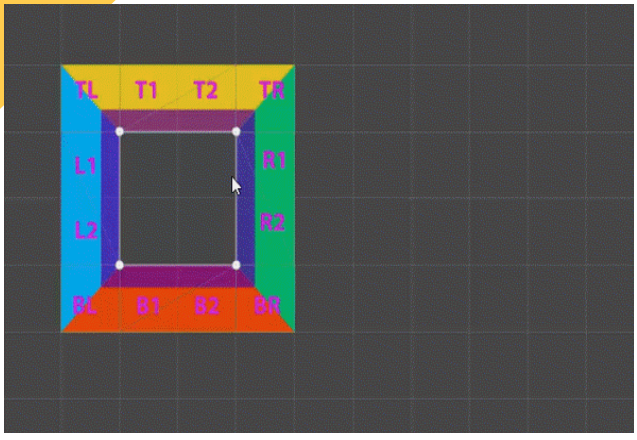
Ryan Wong
MDM Alumni
Project Manager, Hothead Games

CENTRE FOR DIGITAL MEDIA

a collaboration between

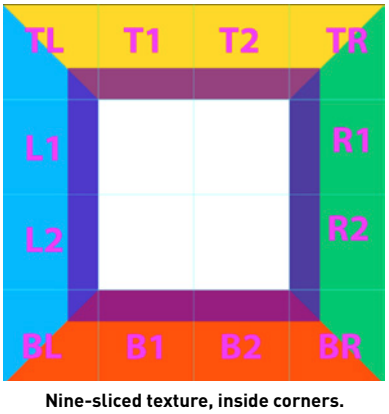
UBC SFU emily carr university of art+design BCIT

learn more ▸ **thecdm.ca**

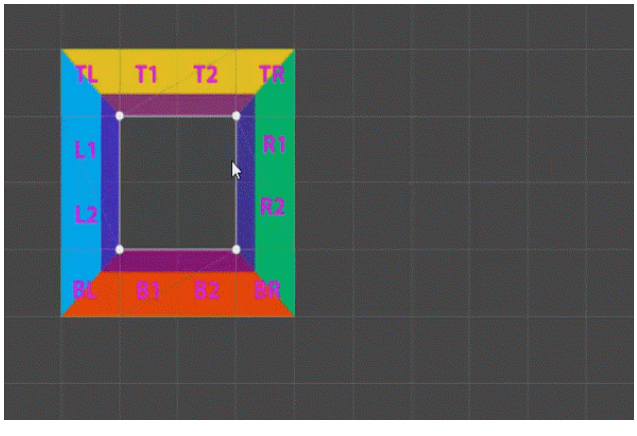


Note how the top-right edge can land either on T1 or T2 depending on the length.

Unlike UI buttons, level design requires assets to have "inside" corners, as well. Whaaat? Imagine you're coming up a cliff, and you reach the top, which is a flat ground. The cliff and the ground form an "outside" corner (90° angle). Now if you're coming down a cliff, the cliff and the ground form an "inside" corner (270° angle). So we can make another nine-sliced texture for the inside corners:

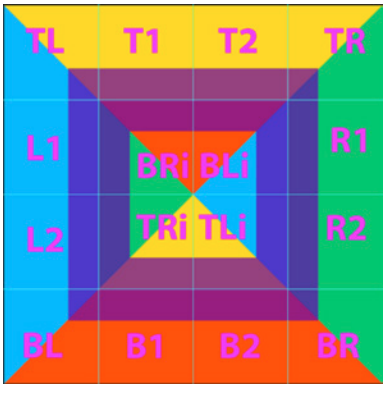


Nine-sliced texture, inside corners.



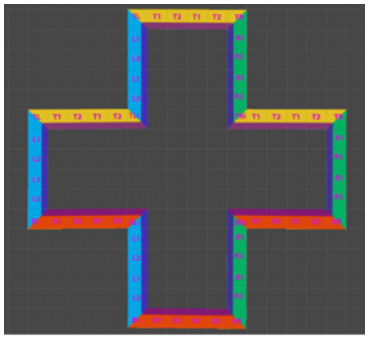
It's not as obvious as it should be here, because I forgot to put a filling texture, but imagine the square in the first gif is filled on the inside, and the second one is filled on the outside (it's actually a hole).

Now this all seems awfully complex, but the good news is that the borders on both textures are actually the same, so we can combine the two textures and get the double-nine-sliced texture! Which kinda sounds like a burger. But we call it a doughnut. Go figure.



The double-nine-sliced texture, aka The Doughnut.

Done this way, the artist has the asset thing in one texture that's relatively easy to author. As with the ribbon, everything has to tile with everything, so you have to use your usual Photoshop tricks for painting tiles, but eventually you get there.



All the corners you'll ever need!

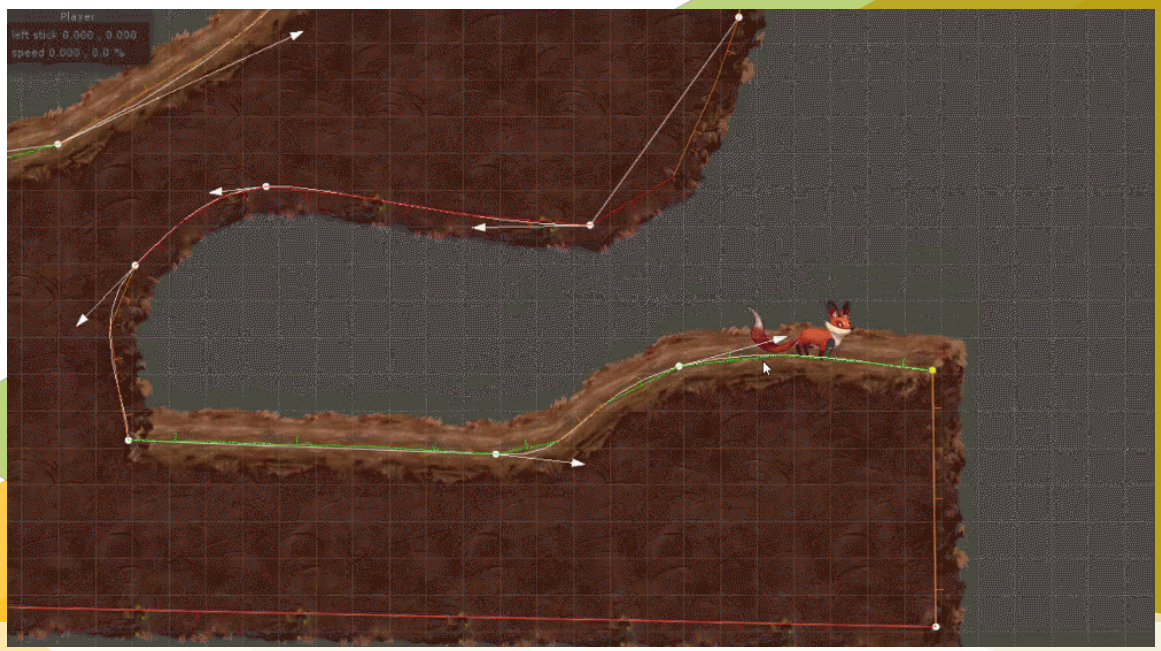
Side-note: As a programmer, I'm kind of stunned at how awkward it is to paint tiles in Photoshop. Hey Adobe, how about some Unity-style extensible canvas with gizmos and EditorWindows and stuff?

And... here's how it looks on a real asset!

Now do you see the donut?

DON'T TRIP ON THE CARPET

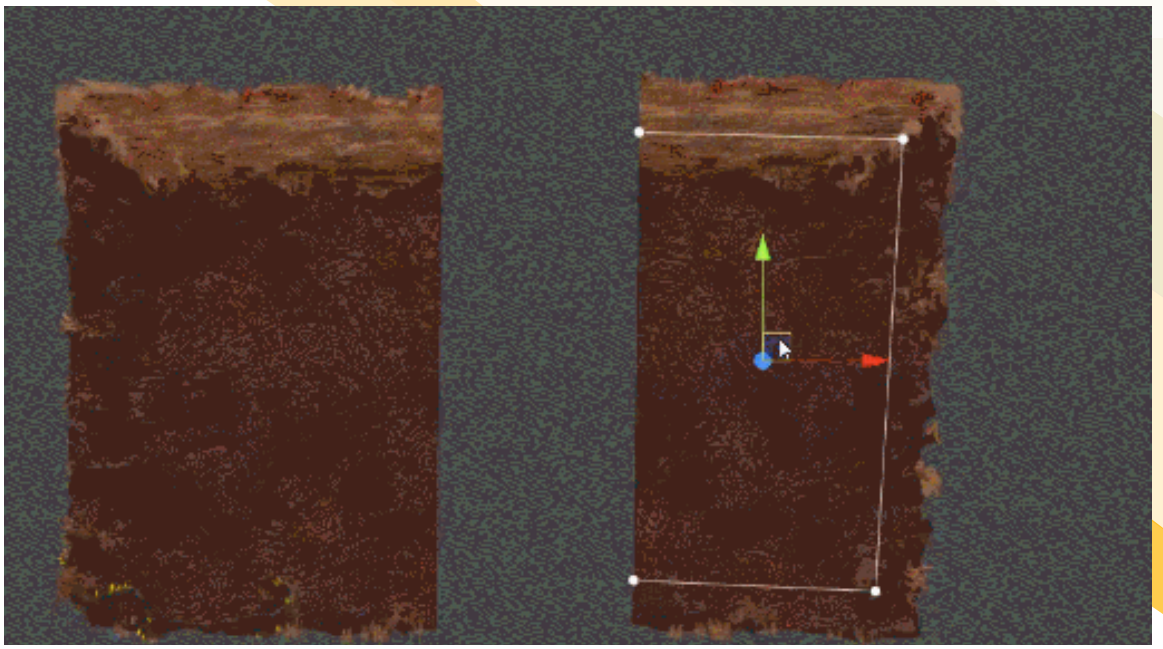
A nice side effect of using vectors to define your levels is that you can reuse that data in all sorts of waysÐto generate collision, for example. From a level-design perspective, that means you can instantly change the shape of your level,



and both the collisions and visuals will remain in sync (though it becomes harder and harder as you layer in more and more level art).

With the same spline that's used to draw the geometry, we can generate the collider, which is the colored line you see here. Green is for ground, orange is for walls, and red is for ceilings, so the level designer has instant feedback. You see here that the edge in front of the fox is orange, so it cannot climb and has to jump: Bad level design! On carpets, the collision is usually closed, while on ribbons, it's usually one-sided (the player can jump through foliage and land on it).

And one last pro tip for filling the carpets: The inside mesh is generated from the spline using a standard ear-clip decomposition (we got the algorithm from Farseer). From there, computing the UVs is pretty straightforwardÐyou just use the vertex



If you didn't understand a word I just said, that's only proof of your sanity. But if you tried it, you'd get it instantly, because it's pretty neat.

coordinates. But if you use the vertex's world-space coordinates, you can snap together several carpets and their UVs will match automatically.

MASTER OF PUPPETS

Next stop: animation! This is another topic with many different schools: pixel art, traditional animation (flipbook), skeletal animations in 2D (puppet), or full 3D rigs. Each has its pros and cons. Nothing beats traditional animation visually (after all, each frame is hand-painted!), but the HD texture size will kill you, and it can be very time-consuming to author. Full 3D is very efficient in terms of production (you get limitless camera angles for free, whereas for an isometric game, you'd have to create four- or eight-direction sprites), but the modeling/rigging/UV-unwrapping/texturing process is very technical and can get pretty daunting. A 3D rig is also not ideal for transparency or fur, and the mesh topology has to be spatially accurate. This can be a blessing (for example U-turn animations are a no-brainer in 3D, but can be surprisingly tricky in 2D), or a curse (e.g., if you want crazy deformations). Check out the new 3D Looney Tunes rigsÐthey're pretty impressive, but have you got what it takes to create that? As for puppet animation (2D sprites jointed together), it's trivial to create and animate, but it can also look rather stiff and awkward if done on the cheap.

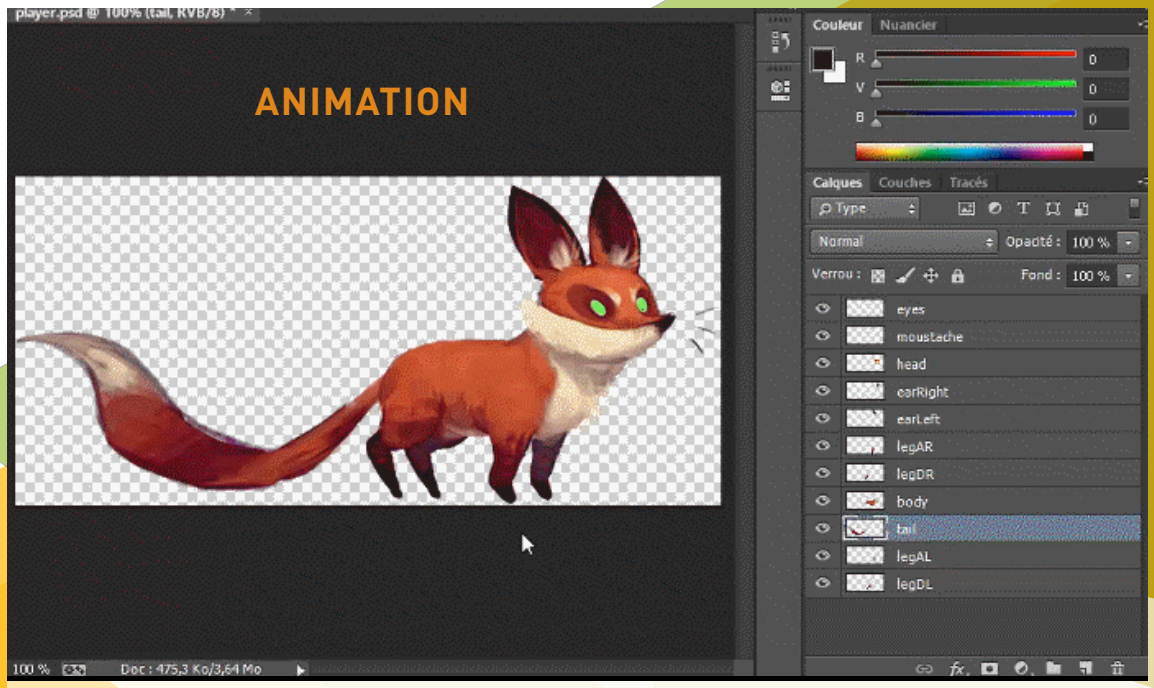
With more recent 2D games, hybrid techniques have emerged. Almost everyone seems to converge on some form of skeletal animation, because the texture cost of frame-by-frame flipbook animation is simply too high at 60fps, 1080p. Curve-based animation allows for fluid results that are way more cost-effective, and it also allows for blending, something that is simply not possible in traditional animation because it forces you to author an insane amount of transitions.

So with all these options at our disposal, we had to start by defining our requirements more precisely. We won't have lots of protagonists in the game, partly because the world and story dictates that, but also because we wouldn't have the resources to create tons of complex animated characters. This is a constraint that we set very early, but the characters we do have will be rather detailed. On the other hand, we want the environment to feel very alive, so every cloud and tree and even stone should move in some way if we can make it do so (Ori and the Blind Forest is a stunning example of that, and is rather intimidating for us).

This doesn't mean we'll create custom animations down to the leaf (there are procedural ways to cut corners, which I won't go into here), but it does mean our system should not be designed with only four-legged animals in mind; it has to scale down to simple stuff very easily. What we wanted was to be able to create scenes with static sprites, but anytime we want to put some more life into an element, we should be able to convert it into a puppet and deform it with simple animations very quickly (the kind of stuff that's trivial to do in Flash, for example).

On the technical front, again a lot depends on what resources you have in house. To create Broken Age, Double Fine had plenty of animators and programmers who were well-versed with 3D packages, so it made sense for them to just rig their characters in Maya with mostly flat meshes (sprites), while benefiting from the array of tools (weighted bones, blend shapes, and so on) of standard 3D animation. They complemented this with flipbook animation (sprite swapping) on faces or other body parts, enabling another characteristic of Broken AgeÐtheir characters are very detailed, with nuanced facial expressions and lots of close-ups, much more so than in the Rayman games, or even Valiant Hearts, which is more character-driven.

We liked the overall principle (and the results!), but we didn't like the idea of rigging in Maya, both because of our skill sets and because I didn't think it would scale down gracefully for our "static tree sprite to animated tree puppet" requirement. Lastly, from past experience with the UbiArt Framework, where the animation tool was the only piece of software that wasn't integrated into the engine, I knew that we would come up with some fun surprises when going from external tool to engineÐsurprises along the lines of "Hey, when I animated it in the tool, the guy's



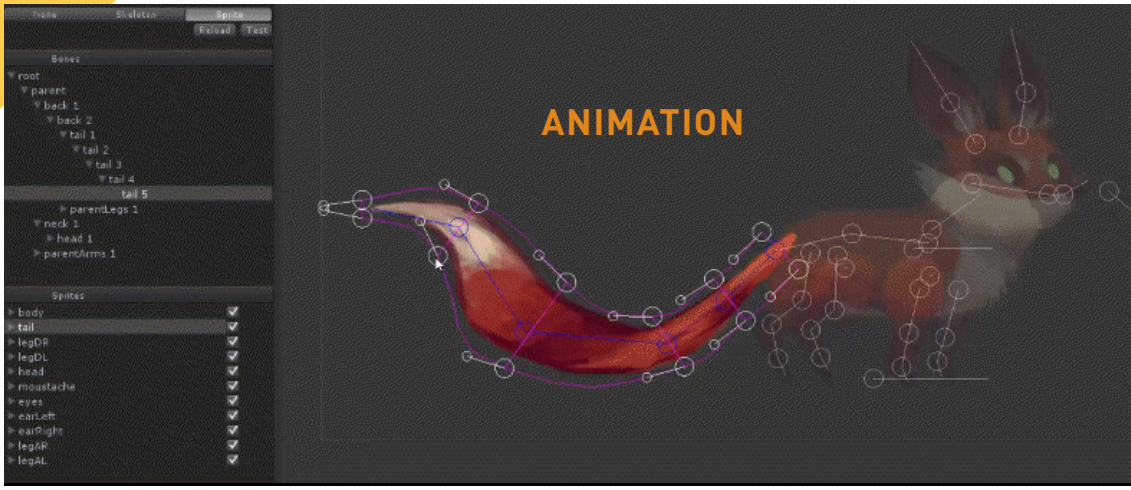
The fox in Photoshop.

leg was in the front, but in the game it's in the back." These are surprises that animators love to discover, and programmers love to debug (or maybe not).

So, the solution we landed on is a hybrid of a hybrid, inspired both by the UbiArt Framework, which is closer to a basic sprite puppet where each sprite is deformable, and Broken Age with its support for flipbook animation on body parts. Instead of creating a sprite hierarchy with pivot points directly, we do have bones; we just automate the process of creating a mesh topology.

Basically, our artist is able to simply draw a character, our animator cuts it up in Photoshop to create the body parts (layers) he needs, and then the rest all happens inside Unity: The rigging is quite straightforward, and we use the engine's built-in animation view to author the clips and Mecanim for the animation tree and blending. This also means that when we have actual in-game rendering/lighting/whatever, we can see the character being animated in the context of a scene with other characters around if we want to. Plus, we can attach particle effects to the bones and animate them, as well as animate script variables, events, and more.

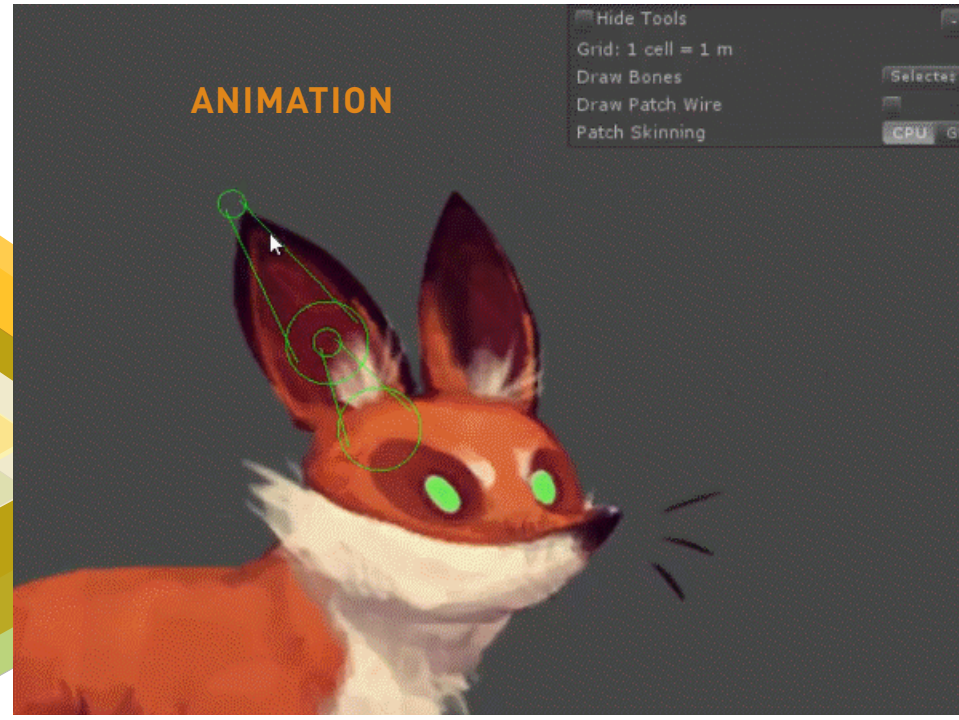
So this is all pretty neat. We lean on our tools (Photoshop and Unity) as much as we can, but one tricky part remains: rigging! Obviously, we're not trying to replicate the power of Maya, because a) that would be insane, and b) the goal is to steer away from complexity anyway. That means we don't model



faces or paint bone weights directly. Our animator imports the Photoshop layout (we autogenerate sprites for each layer, while keeping the original positions from the PSD) and draws the skeleton on top of it, which is rather trivial.

The animator then proceeds to attach each sprite to one or several bones and draw the outline of each sprite using bezier curves (or splines), which results in a surface called a bezier patch. This is a semi-tedious step in the process, and one my overengineering self really would love to automate (it seems quite doable since recent Unity versions give us access to the outline of a sprite as a polyline, so all that's left to do is approximate that as a bezier curve). But so far, we've just crossed it off as "good enough for government work."

With this setup in place, we can deform sprites



Deforming a bezier patch.

using multiple bones, group sprites together in a body part, and make flipbook animations easily (we even reuse Unity's built-in sprite preview in the dopesheet view). We show/hide sprites depending on the stance, fade them out, and more.

A side note on bezier patches: Similar to subdiv surfaces, they describe a vector surface that can be tessellated to a mesh (vertices and triangles). But instead of skinning that mesh with the bones directly, we instead only skin the bezier vertices, with their orientation (tangent) being transformed as well. This gives us a new outline, which we tessellate to a mesh on the GPU (incidentally, the math is super simple and fast). The advantage is that bezier patches usually deform quite naturally, resulting in pleasing curves that work great for organic designs. The word "usually" is key here: There is a tradeoff when we forego classic explicit mesh topology. We give up some control in exchange for simplicity.

2D OR NOT 2D

As a very small indie team, the level of polish we're aiming for with Seasons After Fall is very ambitious. Even though it's "just" a 2D game, we believe investing in tools is crucial to any production of a significant size. But of course, we cannot do everything, and we must pick our battles.

Choosing Unity as an engine is only the first step: You then have to leverage its insane extensibility to create a pipeline suited to the needs of your game and of your team. Recognizing your team's strengths and matching them against the unique aspects of your project will help crystallize your design goals. We all love some good tech, but it has to serve the design first and foremost.

Benoit Fouletier is a gameplay and tools programmer, oscillating between triple-A and indie, depending on the weather. He's currently walking the indie line at Swing Swing Submarine, in sunny Montpellier, France, and previously worked on Rayman Origins and Rayman Legends at Ubisoft Montpellier. He vaguely tweets @benblo42.

DO MORE WITH SYSTEMS: GAME DESIGN TOOLS FOR INDIES

LIKE ALL ARGUMENTS IN VIDEO GAMES, THE QUESTION OF GAME DESIGN STARTS AND ENDS WITH DEFINITIONS. FOR SOME PEOPLE, "game design" means determining content—levels, scripting, and so on. For others it means creating the technical aspects. For still others, it equates to optimization, balancing, and very sophisticated testing. Everyone's a game designer, as Jesse Schell says, because the label essentially means "I make games, and in some capacity I influence the direction of the games I make."

But there's a fourth meaning to "game design" and "game designer." It's a meaning that stands apart from the visuals, the technology, and the data, and instead focuses on the design of systems. By this, I mean the design of actions and rules, of user interface in the abstract, of patterns of play and psychology. That kind of game design is a very specific skill, one that's hard to explain but is nonetheless very real. I hesitate to use the word "pure" to describe it, because that has all sorts of connotations. Instead, I call it "core game design." And I suppose those of us who do that kind of work are core game designers.

But core game design isn't easy to convey. Generally, people who gravitate toward it are of an academic bent, which means that they're bad at agreeing on anything. In the academic world there are many nuances to every question, which is useful in the study of games, but not so much in the practice of making them. Everything is relative, everything is a point of view, and indeed some of the key texts that teach core game design (such as Rules of Play or The Art of Game Design) actively encourage you to think of games relatively. This is fine for designers to do internally, in and of themselves, but in the industry such contemplation tends to fall on deaf ears. The deliverables of core game design struggle to be rendered into something more workmanlike, something that industry-folk might apply in their work.

Game design often—quite rightly I think—gets no respect because game designers can't get their act together. This leads many other people in the industry to conclude that game design is a bit of a fool's errand, and that the only way to design a game is to evolve, iterate, and essentially improve genres. Are they right? Nope. The task of the game developer is not just to build a better jumping game, and I don't mean that solely for high-minded reasons (although I have those too). Rather, it's that genre thinking is ultimately narrow, self-limiting, and what business experts would call red-ocean. If your sole view of game design boils down to making better versions of what you've seen before, you will run out of steam.

Can games be "designed"? Yep. But let's be clear what that means. I'm not here to tell you that core game design is about having ideas. I'm not anti-creative, but in the context of core game design that's not the job I'm describing here. I view my role as game designer as helping teams minimize waste. I help plan features, mechanics, narrative structures, or win mechanisms to assist teams in getting to where they want to be faster than if they did it on their own. The magic that I bring is not wonderful dreams and sparks of imagination so much as a keen sense of what doesn't work and helping them to avoid that fate. That way I help arty developers make better arty games, fun developers make better pure-gameplay games, or whatever.

How? I've been fortunate in life to acquire a lot of experience and a knack for good communication. The first gives me a good bullshit sensor while the latter helps me explain why bullshit is bullshit without offending the bullshitter. But really, a lot of my secret sauce comes down to good design tools. Not software applications per se, but conceptual tools, by which I mean sanity checkers and modeling tools that

help to center an idea and give it some legs.

I can't go through the whole suite of my tools in these few pages (there'll be a book next year), but I thought I'd describe four major tools that I use every day, and which anyone could use to improve their designs. These tools ask hard questions of a project early, demand that a modicum of structure be attached to an idea, and then question whether it still stands up or is lacking. They are most effective in the formative stages of a project, not a month from shipping, when the team realizes that the game they've made contradicts itself in every possible way. So use them early and see if they work for you.

Before we get into the nitty-gritty, I recommend that you designate someone on your team to wear the game design hat. On small teams especially, it's often the case that there's no full-time role for core game design. But it's good practice for someone to wear that hat at least part-time, in order to keep an eye on the bigger picture as well as the day-to-day stuff.

Tool 1: Wireframes

When I design a game, one of the first things I always do is draw it. I might use whiteboards, a sketchpad, or a notebook, but I start by drawing boxes and noting interface or mechanic ideas. I'm not an artist by any stretch of the imagination. Most of my sketches are crude boxy affairs at best, and most of the time they have arrows leading in and out of them to notes that

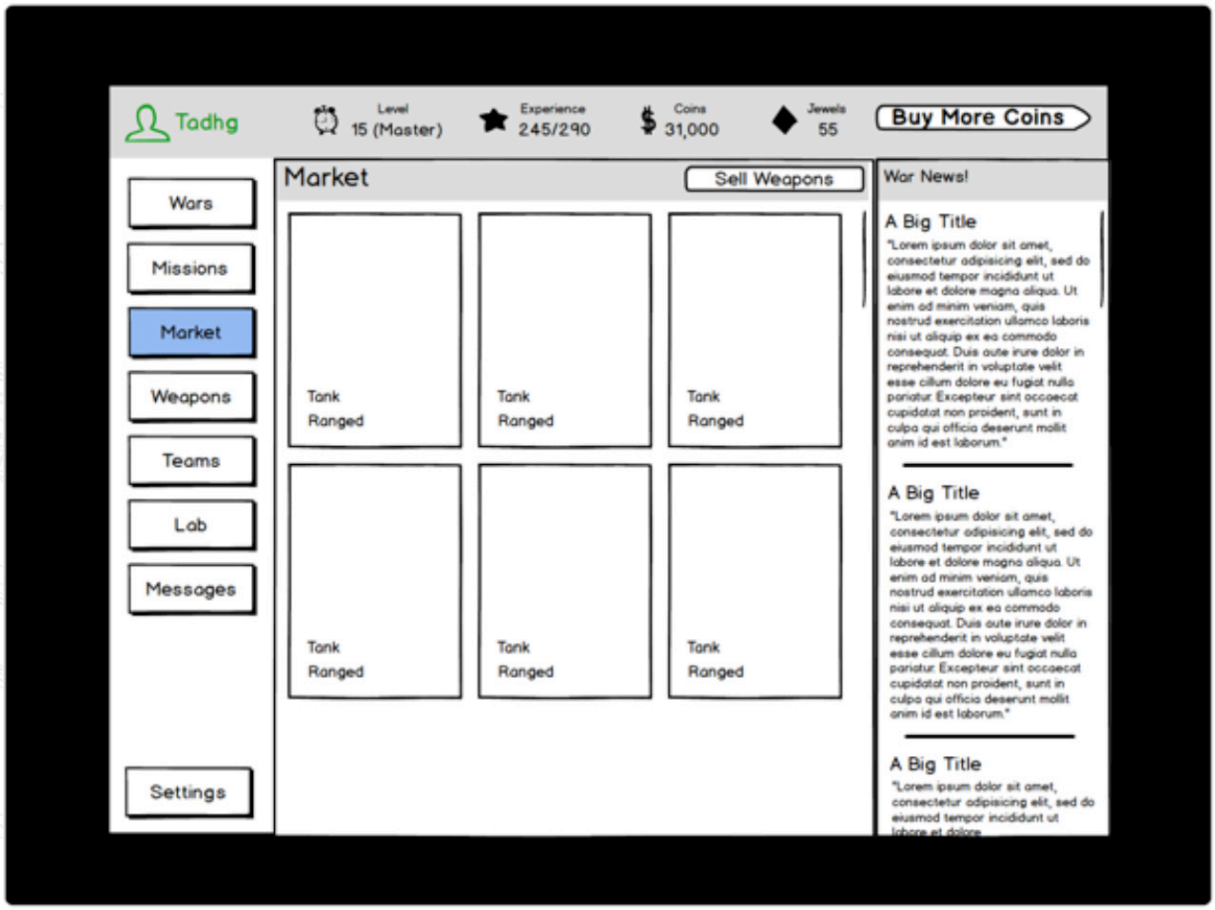


Figure 1: a UI wireframe.

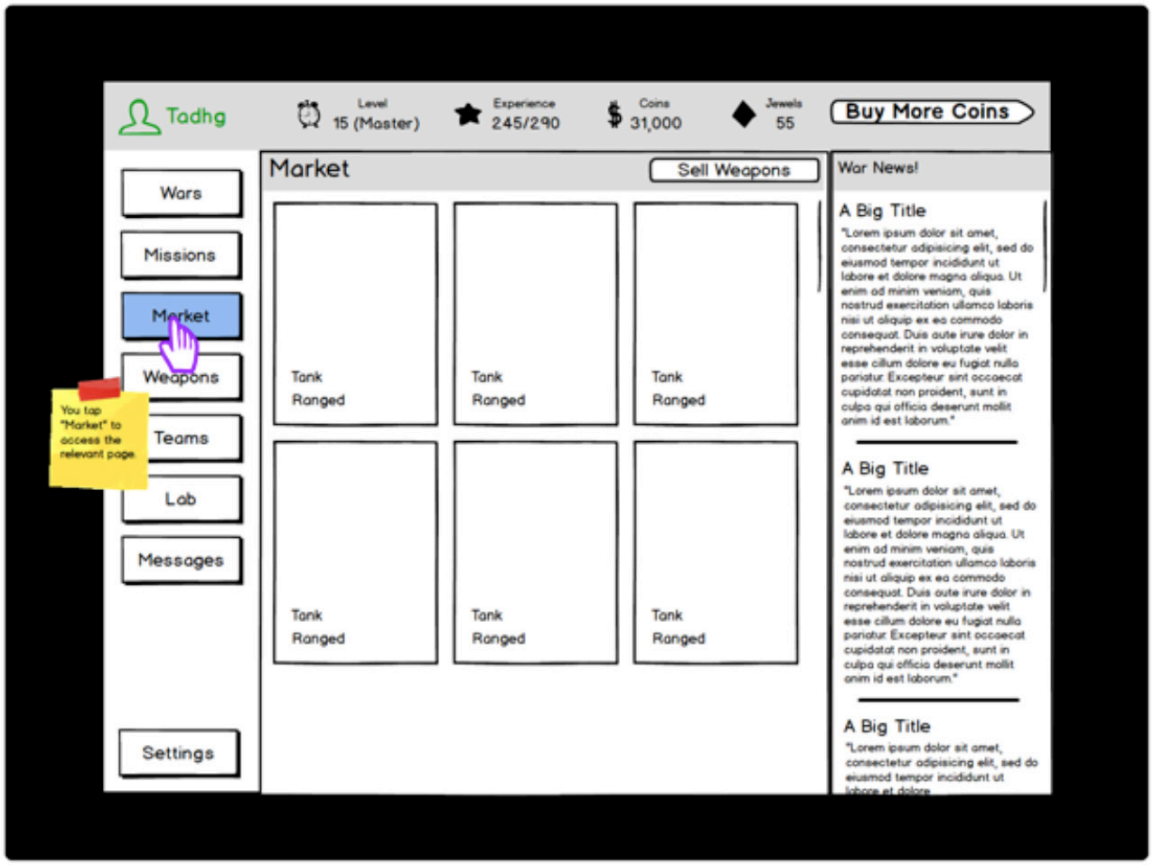


Figure 2: Showing key UI interactions in a wireframe.

only I can understand (I have terrible handwriting, too). Where do I go from there? Wireframes.

A wireframe is a sketch created in an application like Balsamiq, OmniGraffle, or Visio. It's the equivalent of a highly simplified architectural diagram for a piece of software whose purpose is to delineate spacing and functionality. Ideally the cruder this stage is, the better—so as not to confuse the task of core game design with that of making a user interface look pretty.

I think about game design in this visual form very early because it's the best way to force myself to remember that the inner systems of the game need to translate to tangible outcomes on the screen. If they don't, then there's a high risk that those systems will be hard for the player to understand and thus self-defeating. In creating wireframes I find that I make two distinct types. The first is a user interface (UI) wireframe, like this in **Figure 1**.

Figure 1 shows my wireframe for the store interface for a strategy game. I'm describing general layout, position of number indicators, framing, and highlight states for buttons. I tend to take this kind of wireframe and map it out a little like a comic, showing key interactions, as in **Figure 2**.

The second type of wireframe that I do is in-world. What does that mean? The vast majority of game modes have essentially two visual layers. There's a HUD that floats over the world and then the world itself. HUD elements tend to stay toward the edges of the screen (typically top and bottom) while the world occupies the middle. When wireframing in-world I'm usually accounting for both, and again mapping out the key interactions.

For example, **Figure 3** shows a wireframe of an endless runner game.

Figure 3 shows several critical pieces of information, such as the avatar's motion, the distance factors (very important for any jumping

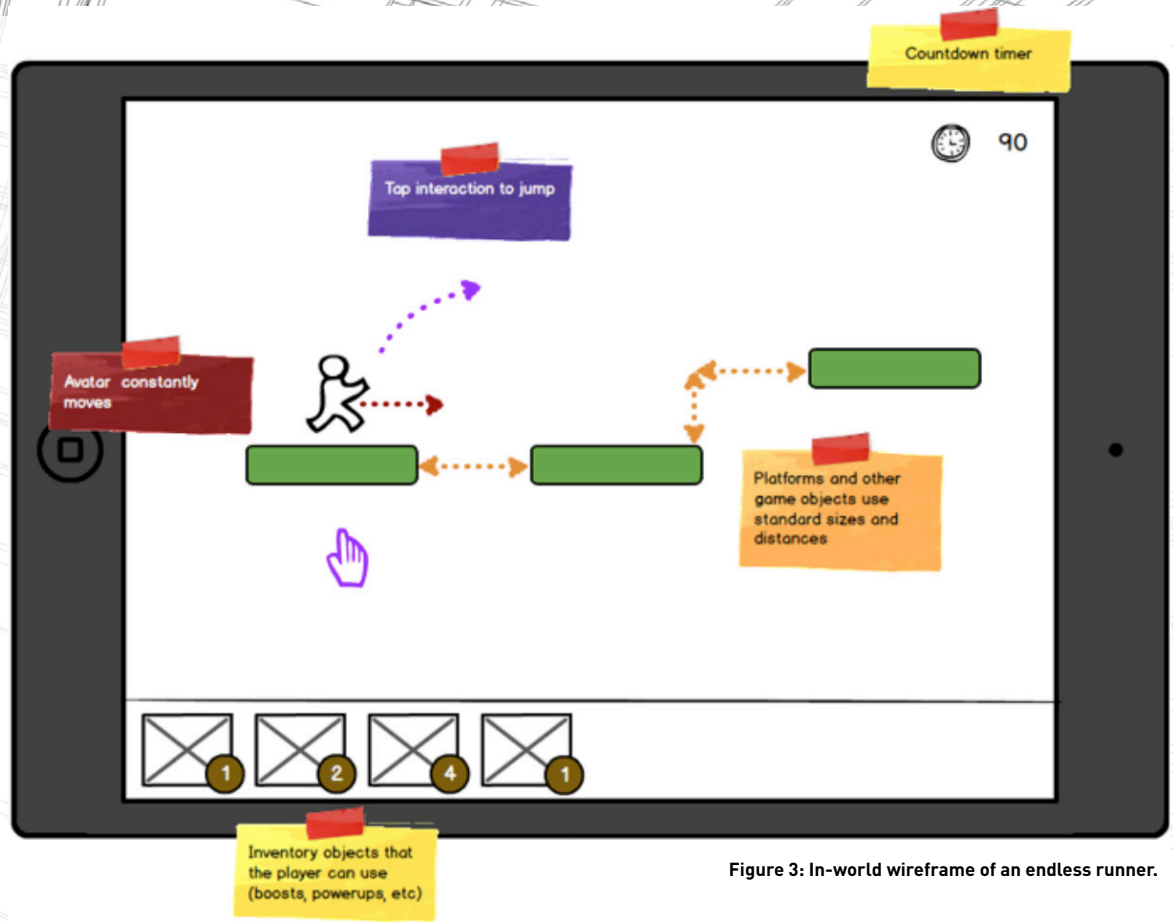


Figure 3: In-world wireframe of an endless runner.

game), the core interactions, and the HUD elements. The use of color is significant. I typically designate up to three colors in my wireframes to show interactions (purple), forces (deep red), and measurements (orange) as notation within the scene. Everything else is actually in-world.

When visualizing the game early like this you should start to anticipate a variety of design questions, such as:

1. **The likely components of the design.** A huge amount of game design is modular, about figuring out a consistent use of components across a game—ideally as few as needed. If you can do this, it means you can create a design that players will find easier to learn, and which is also likely to be comprehensive.

2. **A sense of which mechanics** are likely to visually make sense, and conversely those that will be hard to communicate. Then, you can rule out many potential mechanics that won't easily be understood. Or alternatively, if you're married to a particular mechanic and determined to make it work, wireframes force you to figure out how.

3. **A sense of proportion and placement.** A wireframe is almost never exact in its positioning, but it should give a sense of user interface overload. You'll notice that you unconsciously fall into thinking about your game's layout in rules-of-three, and this is good (this refers to the idea that the viewer unconsciously splits in a screen into a 3x3 grid and finds placement along the dividing lines inherently pleasing). You'll see where components might stack awkwardly and take action to move them, to make them make sense visually. This means that when the game interface gets to development, you'll only have to iterate a couple of times to get it right.

4. **A sense of schema, of interaction patterns** and the permutations that will likely result from them. If, for example, you establish a visual rule for left/right proportions over an entire interface and then realize that you keep making exceptions, maybe your initial rule was bad. Now you can fix it before it moves into code and becomes much more expensive to change.

The other thing to note about wireframes is that I don't just do one. With respect to any one-page-game-design fans out there (and tipping my cap to Stone Librande here), I find that sequences of wireframes tend to illustrate a design far better than single images. It's often important to show the motion of interface elements, the pressing of buttons, the tapping of onscreen components and similar. Wireframes become more like a comic than a poster. I typically generate 10-20 wireframes for any significant piece of design work for a project, and that means I need to be able to generate them quickly.

However, I would advise against sophisticated linking. Some wireframe software lets you virtualize interface elements by creating image hyperlinks between pages, but honestly they are more trouble than they're worth. Just draw your game and export it as PNGs with a simple naming sequence and you'll be 90 percent of the way there.

Tool 2: Modal Map

Game reviews often rate games in terms of their single-player and multiplayer modes, which seems pretty straightforward. Yet as a core game designer, you'll quickly become aware that game modes can get much more complicated than that. At a bare minimum every game has one mode (see Desert Golfing, for example) but conventionally they usually have at least three. Some—like the Wario Ware series—have dozens. Some have strong separations between modes, whereas others are far more interchangeable. They can be a real nightmare.

What is a mode? Any time a game significantly shifts its rules of play, presentation, or both means it changes mode. Grand Theft Auto V has numerous modes, for example, such as in-vehicle, on-foot, in mini-game, on mission, in-menu, in-map, in-mobile, and so on. When you play that game you're traveling back and forth between modes all the time and yet not getting lost. Why? Mostly because of good design.

To try to achieve a similar good order, I use a tool called a modal map (many other designers use something similar called a "user flow diagram"). This is a diagram that starts with the access point to enter the game (at the top), charts the modes through which a player travels during play, and then diagrams the paths along which she travels. A modal map might look something like **Figure 4**.

Simple enough right? It should be, but the important part to pay attention to is those directional arrows. In this map I have a clear flow going from story through to gameplay activities, plus a diversion out to a mini-mode, all of which lead back to a map. Around and around the game mostly goes without being rigid.

Good modal maps help a designer to achieve three things:

1. **A good sense of what travels from one mode to the next.** By this I mean resources, numbers, and game states. Does success in one mode unlock a portion of the next, for example? I often keep notes of what travels between modes and then try to minimize them where possible. Doing so reduces likely compound problems in the design and gives each mode a clear purpose.

2. **A good sense of hierarchy:** Do modes operate in parallel or are they hierarchical? Do they significantly affect one another, and if so how? Be very clear on this question as it's one of the easiest ways for projects to accidentally explode in scope.

3. **A good sense of transitions.** What does getting into and out of a mode look like? What does it need to cover up while code and graphic changes are loading?

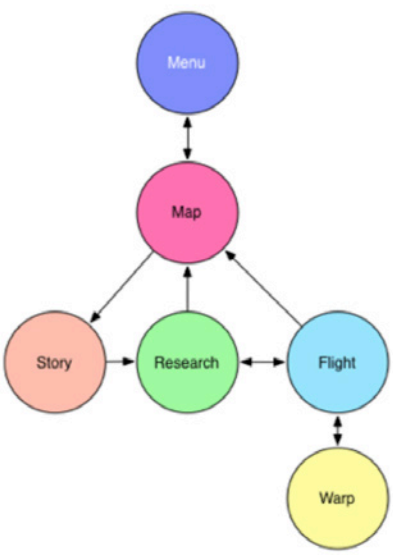


Figure 4: a modal map.

Mostly though, a modal map helps put a sense of framework to your game that becomes invaluable later on, as well as assisting greatly in project planning. They don't take long to do, but they really make you think.

Tool 3: Resource Table

Some designers like to put together large math-y spreadsheets that compute things like hit ratios, and assign variable ranges. Personally, I view that as a waste of time, because when it comes to the actual implementation in code my engineers usually have a better idea of how their engine handles math than I do. Also, by making big spreadsheets I'm treading on their chance to be creative on the project, which I personally feel is arrogant. My job is to help reduce waste, rather than add to it, after all.

Setting good objectives for developers is infinitely more useful. It's not important that I explain the inner workings of the game to them, but it is important that I explain what the outputs should appear to do. In this context I need to explain the resources of the game and their relationship. I need them to get how everything low-level fits together in the core of the game to make it a game. For this I use a resource table.

A resource table is a simple three-column table that lists the name of each resource, how it's earned, and what I use it for. Here's a simple example from Tetris:

Resource	Earned By	Used For
Tetromino	Prior placement	Row creation, Score (small)
Row	Placement of tetrominos	Score (big), Field clearance
Field	Game start	Holding tetrominos
Score	Placing tetrominos, rows	Achieving high scores

The purpose of this tool is to expose the economy of the game, and also to identify what the gameplay is really about. It might sound reductive, but all games are essentially about the conversion of initial resources into end resources through a series of buckets, whether that's guns and ammo or unlock locations for maps. Even Twine games are about making choices that unlock game geography, and that in itself is a resource.

Ultimately what I'm looking for is a final output. Most of the entries in a resource table should lead to other resources (gold leads to items, which leads to better kills, which leads to experience points, and so on). One of these resources usually stands out as the final output—and that's the one that determines true player progress, and therefore purpose. By pushing you into making a resource table what I'm actually doing is forcing you think about that end goal rather than simply about what the player can do.

Note that in this context "resource" doesn't just mean tokens and in-game money. Territory is a resource, as is the player's avatar. A resource table can get pretty lengthy as a result, so it often makes sense to break it up into sub-tables separated by mode (see above) and link them, rather than use one giant table. Combining this tool with the previous one, I might create a set of tables and a rule that each is only permitted to pass one value out from mode to mode. This helps my game acquire a more solid structure, and a sense of specific challenge starts to emerge, which is more likely to prove fun. Understanding that will make you a thousand times better as a game designer.

Tool 4: The Bullet Point

The biggest afflictions of game designers are vagueness and verbosity, being opaque or obscure—essentially being the frustrated sage who complains that nobody understands him. That's because nobody does, and they don't because he's not doing enough to make his work relate to them on their terms rather than his. If your design materials don't speak of utility rather than theory you are simply wasting everyone's time.

As a core game designer you need to know how to write, but also how to write functionally and effectively. You need to know how to be brief, to the point, and declarative. You need to get out of that bad habit of presenting four options for the team to choose from in a spec. You need to drop the equivocation, the tendency to write essays, and the urge to streak ahead. You need to learn the difference between active and passive voices and stop using the latter.

This is why my final tool deals with writing. While many of the skills of a core game designer focus on insight and working out important details in a way that makes sense, all of them can be undone by poor expression skills. Effective game design writing is not writing a big stupid design document. Nobody reads them, everyone hates them, and you'll spend the rest of your time on a project running around after your team. Nor, in case you're wondering, is it writing a big game wiki. There was a phase several years ago when wikis were all the rage, but they're just not great. Key information gets buried due to a lack of maintenance, and a tendency for terminology among multiple contributors to drift over time.

Instead, an effective designer focuses on producing short disposable specs. An ideal spec is a two- or three-page document that covers all the essential detail to get a specific game system working in one or two iterations. It starts with a summary to describe what the mechanic is, but from there it dives into pure bullet points. For example:

Tab Pages in Game X

- A tab page consists of
    - o A Back button
    - o Up to five tab selectors
    - o More cards

• When loaded, the tab page immediately defaults to
    - o The leftmost tab selector
    - o The first card on the page
    - o The carousel showing the second card

See how functional that is? As a game designer, 95 percent of everything you ever write should be blunt like this. Writing in bullet point form forces you to break all your high-minded thinking down into operations and rules, things that the game does and does not. It also gets you thinking about permutation, and how your creativity will need to fit into these rules that you've laid out. The result is something lean, systemic, and, as a result, much more programmable. A developer or artist can immediately understand and execute your desired constituent parts when you bullet-point. And so, for that matter, can you.

Good luck!

Tadhg Kelly is a video game designer, producer, creative director, columnist, and consultant. He has held roles at various video game development, technology and publishing companies. He is currently consulting out of Seattle for a variety of companies under the banner of Tadhg Kelly Game Design, as well as writing a book named Core Game Design. You can reach him at tadhgk@gmail.com.

FREE TO PLAY: AN INDIE DEVELOPMENT PRIMER

Peter Van Dyke

illustration by Dami Lee

Free-to-play (F2P) games have been called everything a piece of media can be called: the "new thing," the "growth engine," the "scourge," and the "darling" of the industry. Free-to-play is praised for creating new markets and expanding the industry, and at the same time ridiculed for a heavy-handed approach to monetization. Many independent teams have found success in the space... perhaps you've even found it intriguing yourself. If so, this article is for you.

The free-to-play business (and design) model benefited from the explosive growth of the smartphone, as well as from access to players and markets where the majority of internet users accessed their first webpage via their mobile phones. Free-to-play was also the first platform-genre of game to employ the comprehensive user metrics analysis pioneered by the online (massively multiplayer) game industry, but with access to data from significantly more users.

There is a lot of information created and compiled by producers and designers seeking to educate the uninitiated with the top two or three "need-to-know" lessons about the free-to-play market. This information is often focused on monetization and presents conveniently descriptive definitions of success, backed by experience with their own data. This often creates a complex definition of "free-to-play" characterized by player-targeting and compulsion mechanics.

In that vein, F2P games can present some of the most demanding and frustrating design problems in the world today, not just because of the tenuous relationship between "free" and "financial success," but also because your audience can be overwhelmingly wide or completely different from your intention at the outset of your project.

Fundamentally, these issues are no different from any other game project. The most structurally significant difference between F2P titles and more traditional projects is defined by the implications of "free" with regard to how players view and approach your game.

I've written this summary as a very high-level overview of how I approach free-to-play games and their market, in the hopes that it'll provide a little guidance and perspective to designers, producers, and small teams that are interested in creating or servicing a F2P game.

Free to Play

For decades, the retail purchasing process defined players' relationships with the games they played. Users would purchase a game, and then play it. This initial cash outlay was an investment that created an opportunity cost equal to the purchase price, giving game makers some additional time to ensnare and compel players. More pessimistically, it added insulation for poor, confusing, uninformed gameplay or interface design. The barrier to entry created by the purchase-first system also restricted access to games: Once people made a purchase, they were tied to that purchase until they were motivated enough to make another.

With the advent of free-to-play, that cushion isn't there to ensure that players give games a chance. The opportunity cost of a purchase has been replaced by an opportunity cost quantified by the amount of time invested into your game. At first install, that opportunity cost is virtually zero. That's a scary thought, one that has wide-ranging ramifications when it comes to everything from game and UX design to narrative and world building. But, the reality about free-to-play games is that they are, first and foremost, games!

Don't forget you're making a game here!

Your game should always be fun. This is a truism often lost in the data-driven design processes that have governed many of the F2P games

released over the last several years. Because players' interaction with your game is based on progress and time spent, the actions your players need to take to progress should be fun! These actions should be repeatable and yield exciting but only somewhat predictable results. For example, tapping to harvest a bale of corn may not be as interesting as swiping across a field (think, Hay Day). These types of simple actions are how your players interface with your game—the more unique and compelling you can make them (without overcomplicating your interface, of course), the more they will help to define your game in a positive way.

Compulsion Loops

Compulsion loops are core to the design of your game. They're not just the system you're designing; they're also an important way for your game to communicate with your players. Compulsion loops are the actions that your users take to make their way to the outcomes that define "progress" in your title. There are three basic types of compulsion loop, defined by the (abstracted) amount of time it takes for a player to complete one loop: short, medium, and long. Short-term compulsion loops are extremely simple, and should take a player between seconds and an hour to complete—things like battle-reward or sow-grow-harvest.

Mid-term compulsion loops are significantly trickier, because they should be one of the first communicators of real game progress to your user. Formally, they expand upon the oft-repeated actions that define your short-term loops, but have a completion time of several hours to several days. They should allow players to use the resources (both in-game, and in terms of skill) that they gather, along the common path of "create, receive, win, discover." They use these resources to build and gain new abilities in your game.

New actions or abilities are a very important reward in F2P games because they ensure that your game remains fresh in the eyes of the user and, more importantly, they are a signifier of progress. Users should be able to complete one go-around of a mid-term compulsion loop in your title within their first or second session to get a taste of what your game has in store for them long-term.

Long-term compulsion loops are similar to the end-game content that plagues Massively Multiplayer games and RPGs. They build upon all of the rewards and abilities that users receive on an hourly, daily, and weekly (even monthly) basis, giving players new ways to use their collected resources. There is one other defining characteristic of long-term loops that differentiate them from mid- and short-term loops: Users should only need to complete short-term and mid-term loops a few times to receive their expected outcome, but they should be required to complete a long-term compulsion loop many times to receive their desired reward. Whereas short-term loops should have outcomes that are easily anticipated by users (or "rare" rewards that are encountered fairly often), long-term loops should help users see what they want from the loop, but not always reward them in the way that they expect. Gacha systems (based on Gatchapon, in which players essentially enter a lottery for a chance to win the best items, are an excellent example of this.

New vs. Proven

It's very tempting to simply emulate an existing, popular F2P mechanic while designing your title, and there is often a compelling business case for doing so. When a mechanic reaches a top-grossing spot on the App Store, it means there are enough players enjoying and playing that type of game to support another game that's similar.

But games take time to make, and simply copying something comes with several significant disadvantages that make this an uphill battle: First, you'll have to either convince players to leave a game they've already invested time in or attract new players—neither of which is an easy task. Second, the game that's already on the market is providing its team with oodles of data they can use for improvements and polishing. This is data to which you do not have access, and which will inform all of their design decisions, but not yours. Third, the time required to develop your game is also time that other teams will be using to improve their own!

The best way forward, in my opinion, is to establish the basis for what you expect of your game, and to learn as much as you can from other games on the market. What did they do right? What did they do wrong? How can those decisions inform your process? Use their experience and outcomes to your advantage in any way you can—there is a ton of expertise and competition in the F2P market, and it's free, right there in front of you! There's always something valuable to learn.

Masking Mechanics

"Masking Mechanics" doesn't mean simply taking an existing game and skinning it. There are many loops that already exist—look for ones that match the mechanics you find interesting and let them inform your design! Gacha and the evolution mechanics that are currently so popular in battle and action RPGs are fundamentally collection mechanics, expanded and transformed to enhance the positive feelings elicited by completing a collection. Many of the battle games that have been released in the last year have character mechanics that are based on card battle games; now they're just 3D. When starting a project,

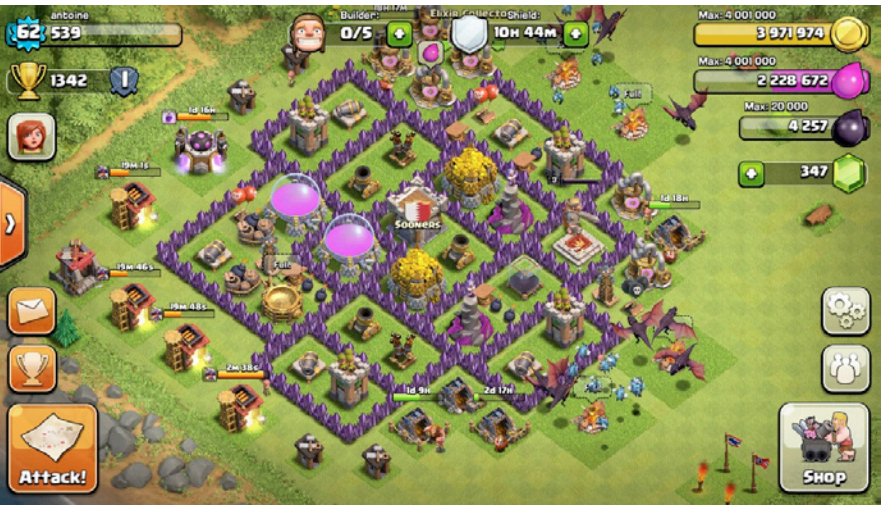
think about ways to give players a sense of reward and progress or incentivize repeated actions. Look to existing games for insights: How can the mechanics you see in fun games already on the market be adapted or transformed to reinforce the vision you have for your project? Make them your own.

Interface Design is About Expectation

When you release your game, it will be let loose into an ocean of existing games, many of which your target audience will be familiar with. You can use that! While simply copying an existing interface design would be poor form, the best interfaces find an elegant way to incorporate trends into a design that is subtly familiar to players while still tailored to your game's unique mechanics.

Interface design is one of the most difficult parts of mobile game development, and if users expect a function to be in one place but they can't find it, they will often eschew that feature in lieu of something that is more readily accessible or understandable. Make sure you do focus testing, and pay close attention to actions that testers take repeatedly as a result of assumptions they make about where to find things. Try to

Games like Puzzle & Dragons use a player's desire to collect as a compulsion loop.



The Clash of Clans-style interface has evolved across the original Clash of Clans, to Boom Beach, to Rival Kingdoms, each taking learnings from the previous iteration.

minimize these frustration points as much as you can, because they can result in players being confused or unable to decide upon their next action. Players that can't see a clear path forward will often close your game and never look back.

In addition, your interface represents the best way to give users visual feedback. Visual design and feedback are important to navigation and usability, but the most overlooked type of visual feedback is the call to action. Calls to action give players next steps, which are integral to ensuring that they begin to intuitively understand the compulsion loops that make up your primary gameplay. A call to action can be many things: an animation floating over something ready to harvest, a badge on a unit or pet of which you have more than one, or even something as simple as a badge on a store tab that indicates the number of purchasable buildings. A good call to action can mean the difference between a player being bored, and a player being so compelled by your game's mechanics that they come back again and again!

Data Design

Mobile games, with their extremely wide user base, contain copious amounts of user data, and that's good for developers. Strong hooks in your game and the ability to analyze in-game metrics will allow you to see which types of player are playing your game, and how.

Retention is King

Metrics are often what drive conversations around mobile F2P games. Day 1/3/7/14/30 retention, average revenue per daily active user or per paying user, lifetime value (affectionately known in the industry by Dn Retention, ARPDAU, ARPPU, and LTV)... these are the measure by which a game is judged as either a success or a failure from a business perspective. The majority of this data is descriptive, but one can be prescriptive: retention. Retention is the metric most strongly correlated with spending propensity, and it's also one of the only ones of these metrics that you can directly

affect with design and content polishing.

Exit Points and Wow Moments

To make the most of your game's potential, make sure that you can track how long users play per session and overall, as well as your users' final actions before they leave your game. Combined with user-specific progress data (like level, play time, and tutorial progress), these metrics help you to identify common exit points in your game and give you information about how to eliminate them. Points of frustration, boredom, loss, disappointment—each of these can be positive or negative, and user data can help you distinguish between the two. Note that simply eliminating all the pressure and frustration points in your game isn't the right way to go about polishing; instead, it's important to identify whether they're positive or negative in terms of your game's pacing and allow that to inform your decision to either eliminate or showcase those moments to your user.

"Wow moments" are just as essential to your mobile F2P game as they are to a triple-A console title or a Hollywood blockbuster. Making sure your game surprises players in a positive way during their first play session goes a long way toward bringing them back to your game later, and it gives players an easy way to communicate about your game with their friends.

My favorite wow moments in F2P games generally come from a revelation regarding mechanics or player-unique content (like the unique characters in Tiny Tower or horses in Derby Days). If you can show a player that the content they have is unique, players will feel more comfortable establishing an emotional bond with your content and a personal relationship with your game.

Mechanic revelations that surprise and awe users can hint at deeper fun and the things users will be able to do in the future; this can be exciting, and will put users on the path to uncovering other new features and ways to play.

Live Operations and User Management

There is a saying in many creative industries: "The last 10% of the project is 90% of the work," implying that the majority of the work when making something comes from polishing. For mobile games, it would be more accurate to say that at least half your work comes after the game's launch. Imagine your launch as a playtest, with thousands (or millions) of players all giving you feedback, all the time. Further, your

Derby Days' unique content comes in the form of rare horses, which players want to collect.

game is alive in the hands of users. There are a number of things to keep in mind for after you launch.

Content Updates

The regularity and quality of your post-launch content updates can be one of the biggest factors in determining the long-term success of your F2P title. Content updates generally fall into three categories: content additions, feature updates, and balance updates. Never forget that content updates are the best way to show users how much you care about them and your game. The moment that dedicated users feel like a game has been abandoned, or the user community is no longer lively, they will start looking for other games to play.

Content updates can be many things. For expressly content-driven games (like Dragon Vale or Puzzle & Dragons), in-game units are one of the most important measures by which users can see their own and their friends' progress. For these types of games, it is imperative to stay ahead of users in terms of their content consumption—if player progress outpaces content updates, you run the risk of losing your most dedicated players. Content updates can include event and seasonal content, new stages... essentially any type of content that users collect or consume. For games that we manage at SK, we try to ensure that our update cadence is bi-weekly or, if possible, weekly, with more substantial content and feature updates once every four to six weeks.

Feature updates require significantly more design and development time, but they also represent the best way to showcase your ongoing support to your players. You can do this with major or minor features, which can take many forms. Small user-facing features and interface tweaks are always good, but giving your team new ways to balance and update the game in real time or to communicate with your players can be invaluable to the user experience. As a side note, the types of features that I like to prioritize are features that can transform your players into content for other players. The simplest example of this is a cooperative multiplayer or PvP system: Players cooperating (or competing) with each other extend the life of your game significantly by giving users a new way to consume the same content.

For competitive games, balance updates are extremely important to the user community and should accompany both content and feature updates if necessary. Paying attention to your players and responding to concerns are key, especially if your game involves any type of direct or indirect competition between players. This doesn't mean that you should always bend to your players, but you should use their feedback to identify potential issues and opportunities for improvement.

Balance updates are also important to the lifecycle of massively single-player games, but they're less guided by player opinion than by your design team's ability to analyze how people are playing your game.

An Open Dialogue

It's very important to maintain an open dialogue with your user community, and to give them as many avenues for social interaction both inside and outside of your game as you can. This is beneficial for you for two reasons: First, it gives users a non-game space in which to dedicate time to your game, and second, it gives you and your team a way to directly respond to and interact with your users, ensuring that they feel heard and valued. These things all contribute to the perception that your game feels alive to your most passionate players.

Events

Live operations are essential to maintaining the liveliness of your community, and events are the most effective tool that you have to boost and sustain retention. They have three primary goals: They bring your game to the player's mind, they give users an additional reason to start and play your game, and they give your community regularly scheduled infusions of material for discussion. They can also give users a new reason to interact inside and outside of your game.

Events can take many forms: Sales, specials, competitive/cooperative events, community events, boosts and power-ups... essentially any time-limited change to the game's rules. Events can be run weekly, daily, or hourly, but simply running an event will not effectively galvanize your community; your team needs to do a good job of informing, promoting, and incentivizing users to participate.

Your most dedicated users will jump at the chance to receive prizes or limited-time bonuses, and these players are often the loudest voices in your community; design a great event, and they will repeat and amplify your announcements by discussing them, lauding them, or criticizing them.

In addition, the importance of planning events to coincide with each other—a competitive event with a sale, for example—will dramatically increase your short-term revenue, and cannot be overstated.

Great Tools

Your backend infrastructure is essential to the successful management of your F2P project after launch, and should be a consideration from

preproduction onward. Great tools empower your live operations team (if you have one!), enabling them to respond to, reward, or compensate players, run events, or promptly and efficiently verify user data in the case of a customer service request. Tools need to be both powerful and flexible, and they should give your team the ability to search for and view both general and user-specific KPIs (key performance indicators) and metrics data. They should also give your support team the ability to message users directly, gift items or currency, and verify user purchases easily and accurately. Great tools are integral to the long-term success of your F2P game!

In terms of data, your backend should provide what both the financial- and design-types on your team require to make strategy decisions for marketing and design/content production scheduling. It should be displayed visually, and the data visualization requirements of your game will be mechanic-specific, so make sure that your designers have plenty of input into what types of data they will need to verify their design decisions and polish after launch. It's also very important for the system that you use to display the data to be flexible, allowing your team to view data granularly or generally, over predefined or custom timelines.

There are plenty of companies developing products that harness and visualize this type of data, and if you're working on a small team, it may be a good idea to look for third-party solutions instead of rolling out your own. Creating a fast, secure system for collecting and displaying data can be a big task, and diverting your engineering resources away from game development and polishing will have an associated non-trivial opportunity cost.

Customer Service and VIP Players

The happiness of your players is essential to the lasting success of your game. Prompt, efficient, responsive, region-specific customer service is a must. Region-specific support is very important; in Korea, for example, having a phone number for your users to call embedded right in your app alongside an email address is essential! Players who have billing issues prefer to make a phone call, and providing that number can help bolster a relationship of trust and keep your most dedicated players in your game. While this isn't the case in the United States necessarily, it is important to be sensitive to your community's wants and needs and to tailor your customer service to your users. For smaller teams, finding a partner in local markets can make this a lot easier.

There are some players, often called "whales" in the industry, who will play your game day-in-day-out and spend gobs of money. Large companies have teams dedicated to identifying players like this who are at risk of leaving (or who have left) and enticing them to stay or return with VIP benefits like unique items. In some cases, they'll even offer these players starter packs for other games in their portfolio in order to keep them within the company's ecosystem. Your game will have these players, and being able to identify and incentivize them to stay in your game (or to move to your new game, if you have one), can have a tangible effect on your long-term financial outlook.

Marketing and User Acquisition

"But what about marketing?" Marketing and user acquisition (which is more specific to F2P), are important tools that can help jumpstart, sustain, or revitalize your game and its community. Knowing when, and how, to properly spend marketing dollars can have a dramatic effect on the efficacy of your campaign and help you to make every penny count.

The most important rule in the user acquisition game is this: Be prepared. This means many things, but at the very least, it means making sure you have the ability to handle increased traffic, and a tracking system in your game that allows you to compare the effectiveness of each of your campaigns in real time. This is important because it allows you to track the origin of your players, identify which are most engrossed in your game (via your retention numbers), and track that data back to identify which sources bring in the most interested players. If you can divine that users from one campaign are more likely to log in several times or even complete your tutorial, it definitely makes sense to allocate more money in that direction versus keeping it on a campaign that only nets short-term users.

Even a minimally granular tracking system will allow you to spend a little money on acquisition to generate valuable user data for analysis, which you can use to improve your game before its big launch. It can also be used to push your game up

in the charts and increase exposure a day or two after release. Both are viable strategies, but they should not overshadow the usefulness of strategic spending during live operations. Facilitating a fresh influx of new players into a stagnating user community can revitalize your commons and extend the life of your game significantly, giving your existing high-level players new compatriots (or enemies, depending on your design).

I'm a team of two. Can I do any of this?

Of course! There is a whole ecosystem of third-party platforms and software that have sprung up to help address the needs of developers building networked mobile games. Many of these tools are monetized based on server calls, so they'll be free at first and only start costing you after you've achieved a modicum of success.

In terms of operations and content updates, the smaller your team, the more planning you should do before you launch. The most effective way to stay ahead of the content update requirements for your live game is to launch the game in your primary market with four to six weeks (or as much as you can muster) of content updates ready to go. That way, you have a head start on upkeep, which will ensure that development bandwidth remains available for the inevitable problems you'll encounter at launch.

For a small team, one other great way to alleviate the pressure for post-launch updates for your F2P game is to try and design features that can turn your player base into a resource for other players! Clash of Clans and Game of War do this elegantly, but you don't necessarily need to create direct competition systems; it's also possible for you to create simpler asynchronous gameplay features that feature other players' characters or likenesses in useful or fun ways. Working with friends to build something, friend-based leaderboards, inviting friends' characters into battle... any features that can turn new players into new content will help keep dedicated players happy with fewer, less frequent updates.

Free-For-All

The world of free-to-play games on mobile is full of potential. Games made for mobile have access to the largest user base of any existing platform, and the number of potential users is increasing every day. That may seem daunting, and designing for something free to be financially viable is tricky, but it's important to remember that above all else, you're making a game! Your goal should be to ensure that your players love everything about your game; that it compels them, surprises them, and at the right times, frustrates them.

Players should be subsumed into your game world within the first minute of gameplay, and you should do everything you can to ensure that each of the first several times they open your game, they're impressed or surprised by something new and exciting. As they progress, they should be looking forward not only to new content, but also to new things to do and actions to take—all of these are important markers of progress.

As your game matures, always plan and execute on content and feature updates. Find ways to let your users help you improve your game with metrics analysis. Spend your marketing money wisely and track where your players come from and how they play. Endeavor to never let your game feel like it's aging or being deprecated.

If you can make a game that's fun, easy to learn, and compelling, users will give you their time and their money, if you give them the opportunity. Your players' relationship with your game is different from that of a player with a triple-A console title—you have to ensure that you give them every opportunity to love and appreciate your project.

Free-to-play games aren't easy to make or maintain, but they are an incredible opportunity for an ambitious team to make their mark on the lives of millions of players.

Peter Van Dyke currently runs production for the Korean branch of npnf, a subsidiary of SK Planet in Seoul, Korea. npnf Korea is responsible for localization, culturalization, operations, and live service for npnf games released in Korea.

WB GAMES IS HIRING!

WB GAMES™

Production • Design
Art • Engineering

TO FIND OUT MORE...

WBGAMESJOBS.COM
FACEBOOK.COM/WBGAMES

MONOLITH™ · NETHERREALM STUDIOS · WB GAMES MONTREAL · turbine® · rocksteady™ · WB GAMES SAN FRANCISCO™

Get Schooled on University Game Programs

A guide for college hopefuls

R. Bryant Francis

If you're a young person thinking about a career in video games, you've probably been sold one of several dreams about getting a college education. Maybe you've been told it's the inevitable next step after high school. Maybe you're the first one in your family to think about attending college. Maybe your parents are alumni of a big university, and have been pressuring you to follow in their footsteps. With more and more schools offering game design programs, your chance of starting your career with a college degree in game studies may be better than ever. But with the average college tuition ranging from $16,000 to $33,000 per semester and beyond, the cost of that dream has never been more expensive. So here's a good question: Should you even go to game school? And if so, why?

Tracy Fullertom

Big universities

If you're still in high school, you've probably been taught to look at that question through the lens of degrees and skills training. But if you talk to people in and around North American game design programs, you'll find a different focus—one on community, collaboration, and portfolio building.

Tracy Fullerton is the director of the University of Southern California Games program, a collaboration between USC's School of Cinematic Arts and School of Engineering. To her, the value of USC's game design program is that it prepares students for a long career with a strong identity and a good network—it doesn't just shuttle you into a job. "In general, the reason that one might choose to go to a game program is there are a lot of critical skills, and there's a lot of general knowledge, and a lot of specific knowledge before jumping into pretty much any career," she says. "So I think it's really important that people go to university, that they round out their education, and become really well-educated citizens.

"I think that if you want to ask what each program offers uniquely, you're gonna look to their culture, you're gonna look to the values that are held in their communities," she continues. "When you look at what USC does, we have a really strong value around creativeness, and around innovation, and thinking about games as something that everyone plays, and making games for people beyond the traditional audience."

Students that came out of USC's game program,

such as Disney Interactive game designer Lauren Careccia, point out that just going for the degree and the skills won't really cut it. "When I was there, there wasn't a lot of focus on programming," she says. "There was a programming class. But if you wanted to learn those particular skills you had to go on your own to learn those. That makes it very difficult when you're going to look for a level-designer job, or anything that requires systems knowledge, or anything that requires any sort of interaction with code.

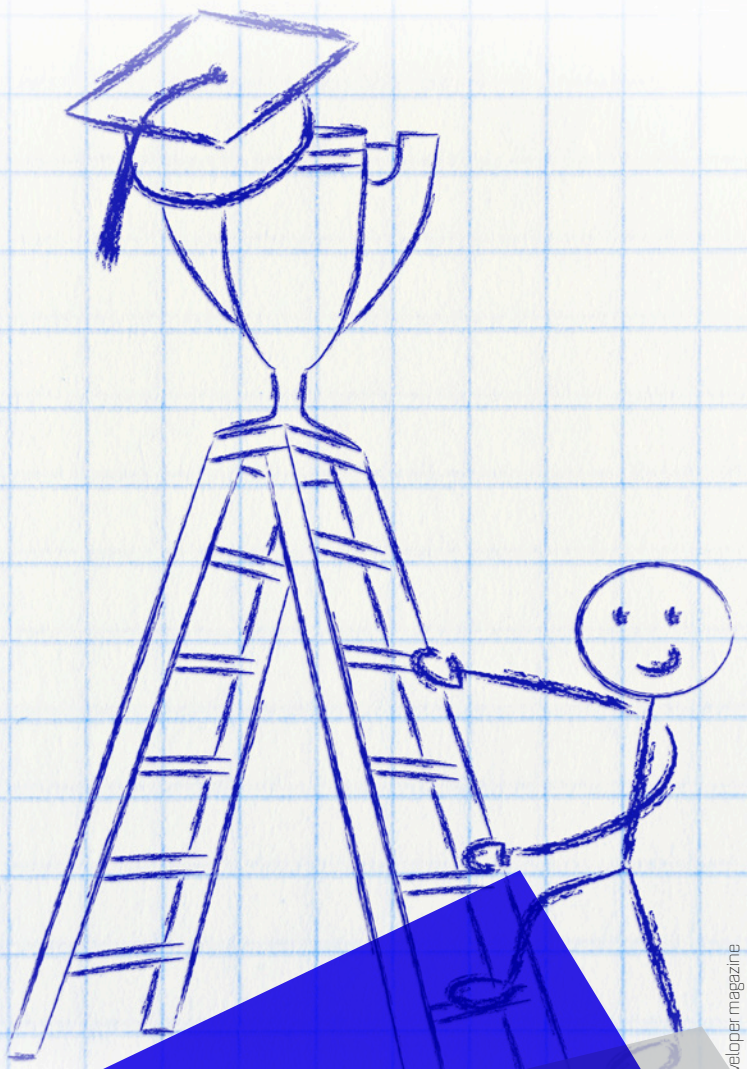
"One thing I missed out on that would have been helpful was the advanced game project. That's usually something done in the fourth year, where you go through an entire game development process. They bring in industry professionals as panelists to determine which ones become the advanced game projects. Usually, they get submitted at the Independent Games Festival. So it's a really, really awesome way to get a fantastic portfolio piece and get your game and your work out there."

Project focus

To get a bit more expansive when thinking about your education, take on a global mindset. Phillip Beau and Daniel Goffin, a German programming duo and winners of the IGF student showcase with Symmetrain, say that the realities of their cheaper German education (500 euros per semester!) mean that their respective undergraduate degrees valued project-based output over skills training. As Beau puts it, "I just think for myself—if I meet a person, and this person tells me 'I got a degree for game design from the University of America,' that would tell me he or she is interested in games, but I really don't know if that would qualify [for] anything. For me the only real qualification is, 'This is the game I did, so what do you think?' And then I can judge from what I see."

That project-based focus is the emphasis of other programs as well, including Carnegie Mellon's Entertainment Technology Center. Its graduate program offers less in the way of instructional

Lauren Careccia



courses, but gears its students to build those projects and portfolio pieces out of the gate. Walt Desler, former Carnegie Mellon student and IGF finalist designer on Way, paints this picture when describing his time there: "For the first semester, the first class you take is called building virtual worlds, and in that class, they take four or five students for a couple weeks at a time and they just kind of make something, like a game or an interactive piece of some sort. And after the first semester, [for] the remaining three semesters, you're put on a team of six to eight people, and you're working on some fairly real-world projects."

It's worth noting that Desler didn't pursue game design as an undergraduate. He majored in computer science and theater, two skills he said were invaluable as he moved through Carnegie Mellon's program, designed Way with Chris Bell, and eventually went to work for Schell Games. "I think my computer science education did a good job of rounding me out as a programmer. From a pure job-getting standpoint, that certainly was a good thing to have."

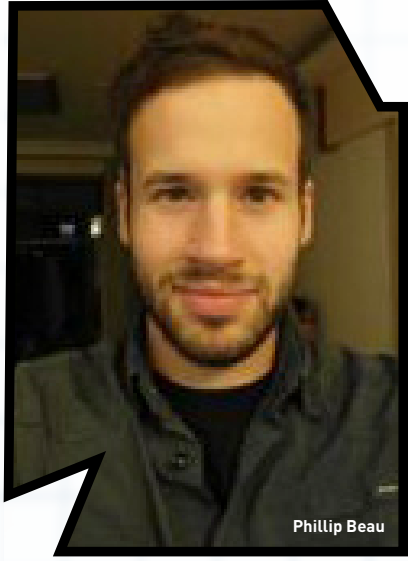

Walt Desler

But can you get work?

Carnegie Mellon, like its nearby cousin Drexler University, emphasizes co-op education—paid work-placement programs that substitute a semester's worth of learning for time spent in the professional field. But these are costly, competitive programs. And game designer Liz England, a graduate of SMU's Guild Hall program and system designer at Insomniac Games, tries to emphasize that when she speaks to students about pursuing game careers. "The number of jobs available is much smaller than the number of people with degrees. And most of the degrees aren't preparing students for realistic roles at companies with realistic skills involved. Usually what I tell students is, unless you're in the top 10 percent or 5 percent of your class, maybe this isn't the direction you should go. Not to go toward a game studies-specific degree.

"If you're graduating out of 30 students and you're all looking for design jobs, you have to be able to look to your peers and say 'Well, is my work better than my fellow students' work?'" she adds. "We might be best friends but now, we're competing for jobs."

England also wants students to think very carefully about how they interpret their potential relationship with their professors when researching game programs. "When you talk about professors, the last thing a lot of game studies programs will advertise is the quality of professors, she says. "What games they've worked on, what games they've shipped... And unfortunately, students will judge programs on that, and that's a bit misleading. There's only a handful of professors at game studies programs that have long résumés—but also where I would say this person's probably an excellent teacher. The first person who comes to mind is Brenda Romero, who's teaching at UC Santa Cruz. That's an example of someone where you know enough about her, and you know what kind of work she does, and she's written on the topic of game design, and she'd probably be a great professor. A lot of other professors are advertised based on games they've worked on, and a lot of these times it's older games—and that's fine—but it's problematic when students go to pick stuff. I've noticed that those students are like 'They worked on this game and it was very good,' or 'They worked on a very bad game and it got poor reviews.'"


Phillip Beau

Know your professor

England's examples can been seen at USC, and that professional background is something Fullerton touches on in her explanation of the program, with game developers such as Naughty Dog alum Rich Lemarchand holding full-time faculty positions, and guest speakers (such as Shane Liesegang being a staple part of the program. Fullerton does, however, also counsel students to consider the professors no matter what school they attend. "The first thing I would say is to make sure there are people who have been in the field, who have real experience in making and publishing games," she says. "A lot of schools won't have those professors in tenured positions, which also makes a lot of difference.

"At USC, we value the experience of these professionals who are coming in to teach," she continues. "We have people who are tenured game designers, who are the core of the teaching faculty, and then we have people who are in charge of practice, and we have the part-time people who come in and teach one or two classes per semester. And those tend to be all working professionals. We have a real emphasis that everyone who teaches a program has either been in the industry or is currently in the industry. And I think that's a really important thing to look for when you're looking at programs."

A geography lesson

In the United States, there is another important consideration for your education: location, location, location. While it's obviously going to matter if you prefer to freeze in New York City or shake, rattle, and roll in a West Coast earthquake, your career prospects will be influenced dramatically by how geographically close you are to a game industry hub. Fullerton points out that USC has a major advantage in this regard. "To be in California, and specifically Los Angeles, not only [gives you] an edge in game development, but also in media in general. It means that there are opportunities here to really look at internships, and interact with people who are thinking of games as they fit into our entertainment landscape, and our media landscape beyond entertainment.

"Being able to interact with those people on a daily or weekly basis is advantageous, and that has to do with the location," she adds. But she also has a take for students who may not be able to afford to study that far away. "There are people who can't or don't want to move. But more and more, it's becoming possible in this world to work from home, or to digitally commute, so if you are for whatever reason locked in a place, you may be able to make it work. It can make it harder, because you are going to a small school in a place where the industry is not central. It certainly makes it harder. When you're more established, it becomes easier to move back to the place where you want to be, but it's hard to break in. So going to school in a place that has deep connections, that has a lot of opportunities, I think gives you a head start, and then if you want to change it up later on, you've already got that structure going around you."

So maybe you could take the indie route? Remember, you're going to school with those $16,000 to $33,000 per semester price tags, and your student loans are still going to be there when you graduate. England cautions against that as well. "Talking to a lot of students, a lot of students felt the dichotomy between getting a triple-A job or going indie, and I feel like a lot of students are saying, 'If I don't get a job, I'll go indie.' But I don't think those students are thinking about the bill that they have to pay when it comes to their student loans and—just in general—the fact that going indie is not something that necessarily sells. Of the people who go indie, it's probably closer to 10 percent. It's probably less than that. So it's a very risky proposition. [Q: Does she mean 10 percent of grads go indie? 10 percent sell something? Better to just clearly state what her point is and quote sparingly] A lot of students are in it for the art of making games, and not so much for the business of making games, and so there's a little bit of a disconnect there."

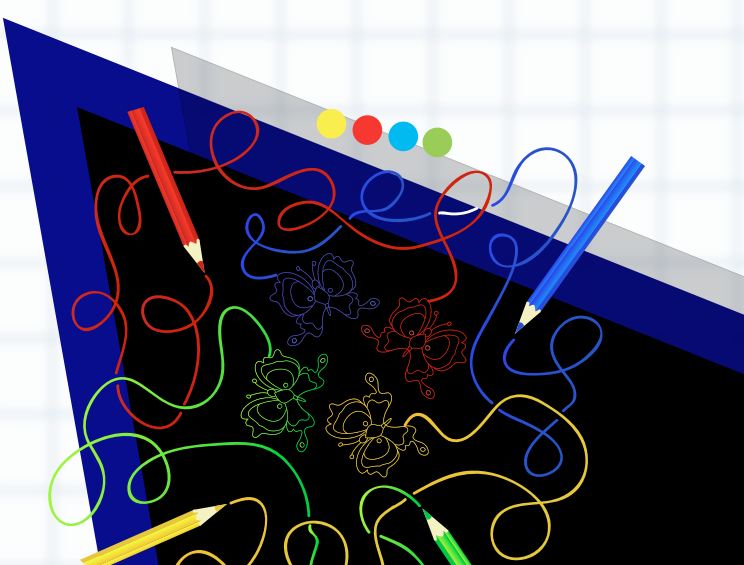
Should you go to game school?

Ultimately, the smartest way to approach your game design school decision might be to, well, think like a game designer. Consider and weigh the variables: the time, cost, networking opportunities, and skill opportunities. Maybe it's not a game design degree you need, but a computer science degree with a personal focus on level design, or a theater degree with a minor in programming. Consider your location, and whether your school is geographically centered in a region where people are actually working in games. Or maybe, with that portfolio you've already started building, a game design degree is exactly what you need. Or maybe what you need to do is learn to make games on your own and avoid a degree altogether. Many game developers are self-taught, and that's still a viable way to get into the industry: simply make games.

But if you do decide to go to school, Tracy Fullerton has one last bit of critical advice: Find your people.

"I can't stress enough that when you're looking at a school, it's as much about how you feel there," she says. "You should sit in a class, you should talk to a professor, you should talk to students. You should walk around the campus, you should get a sense of the community. That community is probably the most important thing to your happiness when you go to a school. I think that's really important for a person to find, especially at the undergraduate level—to find a community that fits their goals and their personalities. I know a lot of kids who say 'Oh, it's too expensive to go visit.' Well, it's gonna be a lot more expensive if you don't like it when you get there. So wherever you go, whatever schools you're looking at, you have to find your people, because those are the people you're going to make your games with."

Bryant Francis is a freelance writer for a variety of publications.



Game School Directory

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 possibly tell you all you need to know, so you should use this directory to find schools you're interested in, and then research them 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 of schools by location, degree offerings, online study options, and more to find the game development school that's right for you!

TITLE	URL	LOCATION	PROGRAM OFFERED
3D Training Institute	www.3dtraining.com	New York, NY	3D Foundation Workshop, 3D Project Based Course
3dmx Digital Design University	www.3d.com.mx	Zapopan, Jalisco, MX	Videogames Development
Abilene Christian University	www.acu.edu/academics/sitc/index.html	Abilene, TX	Digital Entertainment
Academy of Art University	www.academyart.edu	San Francisco, CA	Web Design 1, Visual Development, 3D Animation, Storyboard, Game Environments, Character Animation, Visual Effects, 3D Modeling, 3D, 2D, Character/Tech, VFX/Compositing, Maya, Game Design, Computer Graphics, Background Painting, Modeling
Academy of Interactive Entertainment Sydney	www.aie.edu.au/sydney	Ultimo, NSW, AU	Advanced Diploma of Professional Game Development (Software Development)
Academy of Interactive Entertainment, Lafayette	www.theaie.us	Lafayette, LA	Advanced Diploma of Professional Game Development: Game Programming, Advanced Diploma of Professional Game Development: Game Art & Animation, Advanced Diploma of Screen & Media: 3D Animation & VFX for Film
Academy of Interactive Entertainment, Melbourne	www.aie.edu.au	Melbourne, VIC, AU	Certificate IV in Screen, Certificate III in Screen, Advanced Diploma of Game Development (Art), Advanced Diploma of Screen, Certficate IV in IT, Certificate III in Screen (Architectural Visualisation),
Academy of Interactive Entertainment, Seattle	www.theaie.us	Seattle, WA	Advanced Diploma of Professional Game Development: Game Programming, Advanced Diploma of Professional Game Development: Game Art & Animation, Advanced Diploma in Screen & Media: 3D Animation & VFX for Film
Academy of Interactive Entertainment, Watson	www.aie.edu.au/	Watson, ACT, AU	Diploma of Computer Game Development , Diploma of Screen (for television and film), Certificate IV in 3D Animation for Games and Film

TITLE	URL	LOCATION	PROGRAM OFFERED
Acadia University	cs.acadiau.ca	Wolfville, Nova Scotia, CA	BCSS Game Development
Alberta College of Art & Design	http://www.acad.ca/	Calgary, Alberta, CA	Visual Communications Design Program: Charcter Design Stream
Algoma University	http://www.algomau.ca/computer-science/computer_games_technology/	Sault Ste. Marie, Ontario, CA	Master of Science (Computer Games Technology)
Algonquin College	www3.algonquincollege.com/mediaanddesign/program/game-development/	Ottawa, Ontario, CA	Game Development
American University	www.american.edu/gamelab/	Washington, DC	MA in Game Design
Angelo State University	www.angelo.edu	San Angelo, Texas	Computer Science
Animation Gurgaon	http://www.frameboxx.in/	Gurgaon, Haryana, IN	Redboxx-I
Animation Mentor	www.animationmentor.com	Emeryville, CA	Diploma in Animation Fundamentals , Diploma in Advanced Character Animation Production, Diploma in VFX Fundamentals, Diploma in Advanced Animal and Creature Animation Production
AnimSchool	www.animschool.com	Orem, Utah	3D Animation Program, 3D Character Program
Anne Arundel Community College	www.aacc.edu	Arnold, MD	Game Art and Design Transfer AS Degree, Game Interface Design AAS Degree, Interactive Technologies AAS Degree
Ã–rebro University	www.oru.se	Ã–rebro, SE	Simulation and Game Technology
Art Center Design College - Albuquerque	http://www.suva.edu/	Albuquerque, NM	Animation
Art Center Design College - Tucson	http://www.suva.edu/	Tucson, AZ	Animation
Art Institute of California - Los Angeles	https://www.artinstitutes.edu/los-angeles	Santa Monica, CA	Game Art & Design
Art Institute of California - Orange County	https://www.artinstitutes.edu/orange-county	Santa Ana, CA	Visual & Game Programming, Media Arts & Animation, Game Art & Design
Art Institute of California - San Diego	https://www.artinstitutes.edu/san-diego/	San Diego, CA	Game Art & Design
Art Institute of California - San Francisco	https://www.artinstitutes.edu/san-francisco	San Francisco, CA	Visual & Game Programming, Game Art & Design
Art Institute of Charlotte	www.artinstitutes.edu/charlotte	Charlotte, NC	Web Design & Interactive Media, Graphic Design, Digital Filmmaking & Video Production
Art Institute of Las Vegas	www.artinstitutes.edu/lasvegas	Henderson, NV	Game Art & Design
Art Institute of Pittsburgh	www.artinstitutes.edu/pittsburgh/	Pittsburgh, PA	Game Art & Design, Interactive Media Design, Entertainment Design, Media Arts & Animation
Art Institute of Seattle	www.artinstitutes.edu/seattle	Seattle, WA	Media Arts & Animation, Web Design & Interactive Media, Audio Production, Game Art & Design, Animation Art & Design, Audio Design Technology

TITLE	URL	LOCATION	PROGRAM OFFERED
Art Institute Online	www.aionline.edu/information/programs/animation/game_art_design/?cid=GCARG_121508_Profile_Lin	Pittsburgh, PA	Game Art & Design
Art Institutes International Minnesota	www.artinstitutes.edu/minneapolis	Minneapolis, Minnesota	Interactive Media Design, Design Management, Media Arts & Animation, Photography (BFA), Visual Effects & Motion Graphics
Artcode Game Academy	artcode.la/	San Salvador, San Salvador, SV	DAP2D, Creación de Personajes, DAP3D, Programación de Videojuegos con Java
Asian Institute of Gaming and Animation - AIGA	www.aiga.in	Bangalore, Karnataka, IN	Game Art, Game Programming
Attend iD Tech Camps	www.iDTech.com	Campbell, CA	Game Design & Dev Academy, Game Design Game Development, 3D Animation 3D Modeling
Austin Community College	www.viscom.austincc.edu/	Austin, TX	Visual Communication, Game Development Institute with specialization in Programming, Art or Design
Backstage Pass, School of Gaming	www.backstagepass.in	Hyderabad, Andhra Pradesh, IN	Game Development, Game Art and Design
Baker College Online	www.baker.edu/	Flint, Michigan	Bachelor of Computer Science, Game Software Development
Becker College	www.becker.edu/gamedev	Worcester, MA	Game Development and Programming, Game Design
BES La Salle - Universitat Ramon Llull	www.salle.url.edu/	Barcelona, Catalonia, ES	Grau en Enginyeria Multimèdia - Menció en Videojocs
Blekinge Institute of Technology	gamescience.bth.se	Karlshamn, Blekinge, SE	Digital Game Development, Master of Game Design (Magister i speldesign)
Bloomfield College	www.bloomfield.edu/academics/Divisions/Cre_Art_Tech.aspx	Bloomfield, NJ	Game Design, Game Programming, Graphics, Animation, Music Technology
Boston University Center for Digital Imaging Arts	www.cdiabu.com/	Washington , DC	3D Animation + Interactive Media, Game Art & Character Animation, Animated Short
British Columbia Institute of Technology, School of Computing	www.bcit.ca/study/programs/825hbtech	Burnaby, British Columbia, CA	Bachelor of Technology - Computer Systems Technology - Games Development Option
Broadview Entrainment Arts University	beau.broadviewuniversity.edu/	Salt Lake City, Utah	Digital Video and Media Production, Media Business, Entertainment Design, Music Business, Game Art, Sequential Imaging
Brown College	contact.browncollege.edu/game-design-development.aspx/?src=60246	Mendota Heights, Minnesota	Bachelor of Science in Visual Communications - Graphic Design Emphasis, Bachelor of Science in Visual Communications - Multimedia Emphasis
Brown University	www.cs.brown.edu	Providence, RI	Computer Science
Brunel University - School of Arts	http://www.brunel.ac.uk/courses/undergraduate/games-design-ba	Uxbridge, Middx, GB	MA Digital Games: Theory & Design
Bryan College	http://bryanu.edu/locations/springfield-missouri/	Springfield, Missouri	Gaming and Robotics , Gaming and Robotics Specialist

TITLE	URL	LOCATION	PROGRAM OFFERED
C.W. Post Campus of Long Island University	http://liu.edu/post/gamedesign	Brookville, NY	Master of Arts in Digital Game Design and Development
California Institute of the Arts	www.calarts.edu	Valencia, CA	Character Animation, Experimental Animation
California State University Channel Islands	www.cs.csuci.edu/	Camarillo, CA	Master of Science in Computer Science, Bachelor of Computer Science, Interdisciplinary Minor in Game Design and Development
California State University, Fullerton	www.fullerton.edu	Fullerton, CA	Entertainment Art/Animation
Camden County College	www.camdencc.edu	Blackwood, NJ	Game Design & Development, Computer Science, Computer Graphics
Cañada College - Redwood City - CA	www.canadacollege.edu/multimedia	Redwood City , CA	Multimedia Art & Technology
Capilano University	www.gradshow.com/	North Vancouver, British Columbia, CA	Digital Animation
Carnegie Mellon University Entertainment Technology Center	www.etc.cmu.edu	Pittsburgh, Pennsylvania	Entertanment Technology Center
Centennial College	http://www.centennialcollege.ca/	Toronto, ON, CA	Game Design & Development
Center for Distance Education	www.cd-ed.com	Sydney, Nova Scotia, CA	3D Advanced: Character Animation , 3D Animation, 3D Game Artist
Centre for Arts and Technology - Fredericton Campus	www.digitalartschool.com	Fredericton, New Brunswick, CA	Audio Engineering & Production, Animation for Game, Film & Visual Effects, Digital Filmmaking, Graphic Design & Web Development
Centre for Arts and Technology - Halifax Campus	www.digitalartschool.com	Halifax, Nova Scotia, CA	Event & Promotions Management, Graphic Design & Web Development, Digital Filmmaking, Animation for Game, Film & Visual Effects, Audio Engineering & Production
Centre for Arts and Technology - Kelowna Campus	www.digitalartschool.com	Kelowna, British Columbia, CA	Animation for Game, Film & Visual Effects , Digital Filmmaking, Graphic Design & Web Development, Audio Engineering & Production
Centre for Digital Media	www.thecdm.ca	Vancouver, BC, CA	Master of Digital Media program
CENTRE NAD - National Animation and Design Centre	www.centrenad.com	Montreal, Quebec, CA	Bachelor in 3D animation and digital design (Accredited undergraduate program)
Centro de Entrenamiento Alcance Digital	www.alcancedigital.com	Leon, Guanajuato, MX	Producción profesional de videojuegos, Animación 2D, Cinematografía Digital, Producción Creativa, Composición musical digital, Estereoscopia 3D, Animación 3D, diseño Digital, Efectos visuales
Centro de Informatica - Universidade Federal de Pernambuco	www.cin.ufpe.br	Recife, Pernambuco, BR	Game Design and Development
Champlain College	gamestudio.champlain.edu	Burlington, Vermont	Production, Game Art and Animation, Game Design, Game Programming
Charles Sturt University	www.csu.edu.au	Bathurst, NSW, AU	Bachelor of Computer Science (Games Technology), Bachelor of Arts (Animation and Visual Effects)

TITLE	URL	LOCATION	PROGRAM OFFERED
Clover Park Technical College	http://www.cptc.edu/	Lakewood, WA.	Media Design and Production - 3D Art and Animation
Cogswell College	www.cogswell.edu/info/game.php	Sunnyvale, CA	Entrepreneurship & Innovation , Software Engineering, Digital Audio Technology, Digital Arts Engineering, Game Design & Development, Digital Media Management, Digital Art & Animation
Colegio Universitario IES21	www.ies21.edu.ar	Córdoba	Desarrollo de Simulaciones Virtuales y video-juegos
Coleman University	www.coleman.edu	San Diego, CA	Game Programming Development and Design
College for Creative Studies	www.collegeforcreativestudies.edu	Detroit, MI	Advertising Design, Entertainment Arts, Illustration, Illustrationi
College of Lake County	http://www.clcillinois.edu/	Grayslake, Illinois	Computer Information Technology (Game Development)
Cologne Game Lab	www.colognegamelab.de	Cologne, NRW, DE	Game Development & Research
Columbia College Chicago	www.colum.edu/game	Chicago, IL	Game Design (Game Development), Game Programming (BA), Game Programming (BSc), Game Design (Game Art), Game Design (Game Sound)
Concordia University Centre for Continuing Education	http://sarno.concordia.ca/conted/departments/program.aspx?program_id=298DeptName=Computer%20InstituteDeptCode=CI	Montreal, Quebec, CA	Graphics & Visualization, Game Programming Certificate
Confetti ICT	www.confetti-ict.com	Nottingham, Nottingham-shire, GB	Interactive Gaming
Conservatory of Recording Arts & Sciences	www.audiorecordingschool.com	Gilbert, AZ	MRPII
Cornell University	gdiac.cis.cornell.edu	Ithaca, NY	Game Design Minor
Creajeux	www.creajeux.fr	Nimes, FR	Concepteur 3D, Programmeur
DADIU	http://www.dadiu.dk/	1437 Copenhagen K, Den-mark, DK	Game Production
Dakota State University	www.dsu.edu	Madison, South Dakota	Computer Game Design
Daniel Webster College	www.dwc.edu	Nashua, NH	Video Game Programming
DAVE School (The Digital Animation & Visual Effects School)	www.daveschool.com/	Orlando, FL	Visual Effects, Game Production
De Montfort University	www.dmu.ac.uk/faculties/art_and_design/ug_courses/game_art.jsp	Leicester, Leicestershire, GB	Game Art Design
Department of Digital Technology and Game Design, Shu-Te University	http://en.main.stu.edu.tw/	Kaohsiung , Taiwan , TW	Digital Game Design, Multimedia Program Design, Information Scinece and Design

TITLE	URL	LOCATION	PROGRAM OFFERED
DePaul University - GameDev. DePaul.edu	CDM.DePaul.edu	Chicago, IL	Animation - Game Art, Computer Science, Game Development - Programming, Digital Cinema, Animation, Animation - Technical Art, Game Development - Game Design
Derby University - School of Computing and Mathematics	http://www.derby.ac.uk/engineering-technology/undergraduate/	Derby, UK, GB	Computer Games Modelling and Animation, Computer Games Programming, Computer Graphics Production
DeSales University	www.desales.edu/	Center Valley, PA	Computer Science - Game Programming Track
Design3 - Learn game design, app development and 3D animation	www.design3.com/	Online	Engines and SDKs, Theory and Design, 2D Art and Design, 3D Art and Animation, Audio, Web
DeVry University	www.devry.edu	Downers Grove, IL	Game & Simulation Programming
DeVry University - Dallas Metro Campus	www.devry.edu	Irving, TX	Game and Simulation Programming
DeVry University - Oakbrook Terrace, IL	www.devry.edu	Oakbrook Terrace, IL	Game and Simulation Programming
DigiPen Institute of Technology	www.digipen.edu	Redmond, WA	The Bachelor of Arts in Music and Sound Design, Bachelor of Science in Computer Science in Real-Time Interactive Simulation, Master of Fine Arts in Digital Arts, Bachelor of Science in Computer Engineering, Bachelor of Science in Computer
DigiPen Institute of Technology Europe-Bilbao	www.digipen.es	Zierbena, Bizkaia, ES	Digital Arts and Animation, Computer Science in Real-Time Interactive Simulation
DigiPen Institute of Technology Singapore	https://singapore.digipen.edu/	Singapore, SG	Digital Arts and Animation, Game Design, Computer Science and Game Design, Computer Science in Real-Time Interactive Simulation
Digital Film Academy	www.DigitalFilmAcademy.edu	New York, NY	Digital Filmmaking Conservatory
Digital Media Academy	www.digitalmediaacademy.org/	Campbell, CA	Pro Series Training Courses, Educator Series
Digital Media Arts College	https://www.dmac.edu/	Boca Raton, Florida	Bachelor of Fine Arts - Computer Animation Game Art Concentration, Bachelor of Fine Arts - Computer Animation, Master of Fine Arts - Visual Effects Animation
Doña Ana Community College	dabcc.nmsu.edu/	Las Cruces, NM	Game Design
Drexel University, Antoinette Westphal College of Media Arts and Design	http://drexel.edu/westphal/undergraduate/GMAP/	Philadelphia, PA	Digital Media, Graphic Design, Photography, Film & Video, Screenwriting & Playwriting
Edinboro University of Pennsylvania	www.edinboro.edu	Edinboro, Pennsylvania	Computer Animation, Game and Virtual World Development
Edison Community College	http://catalog.edisonohio.edu/preview_program.php?catoid=198&poid=3231&returnto=1284	Piqua, OH	Computer Games and Simulation Programming and Design
Edith Cowan University	www.ecu.edu.au/	Perth, Western Australia, AU	Bachelor of Science (Digital Media), Bachelof of Computer Science (Games Programming), Bachelor of Creative Industries (Game Design and Culture), Master of Games and Simulation Programming, Graduate Diploma of Games Programming

TITLE	URL	LOCATION	PROGRAM OFFERED
Edmonds Community College	www.edcc.edu/	Lynnwood, WA	Game Development
Emily Carr University of Art + Design	www.ecuad.ca	Vancouver, BC, CA	Bachelor of Fine Arts , Master of Applied Arts - Media Stream
ENJMIN (National School of Video Game and Interactive Media)	www.enjmin.net	Angoulême, FR	graphics, project management, computer science, ergonomy, production, game design, sound design
Escuela Da Vinci	www.escueladavinci.net/	Buenos Aires, Buenos Aires, AR	Diseños de Videojuegos / Videogame Development
EuNoia Animation School	www.eunoia.tv	Mexico City, Mexico, MX	Traditional and 3d Animation
Expression College for Digital Arts	www.expression.edu	Emeryville, CA	Interaction Design, Game Art and Design, Interactive Audio, Digital Filmmaking, Animation and Visual Effects, Motion Graphic Design, Sound Arts
EzetriX-Gaming and Animation Institute	www.ezetrix.com	Pune, Maharashtra, IN	Diploma in Game Art & Design
Facultad de Ingeniería - Universidad Nacional AutÃ³noma de MÃ©xico	www.ingenieria.unam.mx/	México D.F., México D.F., MX	Ingeniería en Computación/Computación Gráfica
Faculty of Informatics/Complutense University of Madrid	www.videojuegos-ucm.es/	Madrid, Madrid, ES	Master en desarrollo de videojuegos (diseño), Master en desarrollo de videojuegos (programaciÃ³n)
Fanshawe College	www.fanshawec.ca/programs-courses/full-time-programs/gdp1	London, Ontario, CA	Game Development - Advanced Programming
Fatec Sao Caetano do Sul	www.fatecsaocaetano.edu.br	Sao Caetano do Sul, SP, BR	Tecnologia em Jogos Digitais
Ferris State University	www.ferris.edu/dagd	Big Rapids, MI	Digital Animation and Game Design
Fingerlakes Community College	www.flcc.edu/academics/gameprogramming/index.cfm	Canandaigua, NY	Game Programming and Design
Florida Interactive Entertainment Academy at UCF	http://www.fiea.ucf.edu/	Orlando, Florida	Art Track, Programming Track, Production Track
Foothill College	www.foothill.edu	Los Altos Hills, CA	Music Technology
Friends of Design - Academy of Digital Arts	www.friendsofdesign.net	Cape Town, Western Province, ZA	Game Technology and Multimedia Entertainment
Full Sail University	www.fullsail.edu/index.cfm?fa=landing.	Winter Park, FL	Game Development, Game Art, Computer Animation, Game Design
FuturePoly	www.futurepoly.com	Bellevue, WA	3D Modeling for Games, Digital Painting, Character Modeling with ZBrush, Character Animation for Games, Texturing for Games, Concept Art
FZD School of Design	http://fzdschool.com/	Singapore, Singapore, SG	Character Art, Environment Art, Color Key Art, Production Art, Game Assets Art
Game Academy	www.gamea.com.cn	Shanghai, CN	Game Design, NextGen Game Art
Game Character Academy	www.gc-academy.net	Online	Character Rigging/Technical Art, Character Art
Game Institute, Inc.	www.gameinstitute.com/	New York, NY	Game engine, 3D graphics development

TITLE	URL	LOCATION	PROGRAM OFFERED
GAMES ACADEMY	www.games-academy.com	Berlin, Germany, DE	Game Production, Game Design, Film Art & Animation, Game Art & Animation, Game Programming, Interactive Audio Design
Gemini School of Visual Arts & Communication	www.geminischool.com	Cedar Park, TX	Diploma in Visual Arts & Communication
Generando IT	www.generandoit.com/	Buenos Aires, Buenos Aires, AR	Iniciación Juegos Multiplayer, Unity 3D Fundamental, Flash Games
George Brown College	http://www.georgebrown.ca/G4@5-2@15-2@16/	Toronto, Ontario, CA	Game Design
George Mason University, ACS	www.cs.gmu.edu/~acsgame/	Fairfax, VA	ACS Game Design
George Mason University, BFA	game.gmu.edu	Fairfax, VA	Computer Game Design
Georgia Institute of Technology	www.gatech.edu	Atlanta, GA	Artificial Intelligence, Graphics & Visualization
Glasgow Caledonian University	www.gcu.ac.uk/	Glasgow, GB	BSc Computer Games (Design), BSc Audio Technology with Multimedia, BA 3D Computer Animation, BSc Computer Games (Software Development), BA Computer Games (Art & Animation)
Glasgow Caledonian University, School of Engineering and Computing	www.gcu.ac.uk/sec/study/	Glasgow, Scotland, GB	Computer Games (Software Developemnt), Computer Games (Design)
Glendale Community College	www.gc.maricopa.edu/	Glendale , AZ	AAS Multimedia Animation
Glyndwr University	www.glyndwr.ac.uk	Wrexham, Wrexham Borough, GB	Computer Game Development, Digital Art for Computer Games
Gnomon School of Visual Effects	www.gnomonschool.com	Hollywood, CA	Highend Computer Graphics Certificate Program, Maya Fast Track Program
Goldsmiths, University of London	www.gamesgoldsmiths.com	London, England, GB	MSc Computer Games & Entertainment
Great Northern Way Campus	mdm.gnwc.ca	Vancouver, British Columbia (Canada) , CA	Masters of Digital Media Program
Guildhall at SMU	www.smu.edu/guildhall	Plano, TX	Art Creation, Software Development, Level Design, Production
Hagerstown Community College	www.hagerstowncc.edu/	Hagerstown, MD	Simulation and Digital Entertainment
HAL Institute of Computer Technology - Nagoya	http://www.hal.ac.jp/	Nagoya, Aichi-ku, JP	Multimedia, Music and General information processing
HAL Institute of Computer Technology - Osaka	http://www.hal.ac.jp/	Osaka, JP	Multimedia, Music and General information processing
Hamburg University of Applied Sciences (HAW)	www.gamesmaster-hamburg.de/en/	Hamburg, Hamburg, DE	Time-based Media / Sound - Vision - Games
Hamidrasha Art College, Beit Berl Academic College, Kalmaniya	www.beitberl.ac.il	Beit Berl, Near Kfar Sava, IL	Software Development , Animator, Copywriting , Graphic Designer, Content Design
Harper College CE	http://goforward.harpercollege.edu/	Schaumburg, IL	Flash Game Designer CE Certificate
Harvard Extension School	www.extension.harvard.edu/	Cambridge, MA	Understanding and Developing Multimedia, Introduction to Computer Graphics, Digital Multimedia Art

TITLE	URL	LOCATION	PROGRAM OFFERED
Herzing University - Madison WI	https://www.herzing.edu/career-programs/undergraduate-degrees/design/game-development	Madison, WI	Graphic Design, Game Development
Herzing University - Milwaukee	https://www.herzing.edu/career-programs/undergraduate-degrees/design/game-development	Milwaukee, WI	Graphic Design, Game Development
High Point University	www.highpoint.edu/	High Point, NC	Game and Interactive Media Design
Hochschule Darmstadt University Of Applied Sciences	www.h-da.de	Darmstadt, Hessen, DE	Digital Media, Media, Media Direction
Hochschule Mittweida (FH) University Of Applied Sciences	www.hs-mittweida.de/	Mittweida, Sachsen, DE	Media Informatics and Interactive Entertainment
Homer College	www.omiros.gr	Athens, Attiki, GR	MultiMedia Computing
Hong Kong Polytechnic University - Multimedia Innovation Centre (MIC)	www.mic.polyu.edu.hk	Hong Kong, Kowloon, HK	Multimedia & Entertainment Technology, Traditional & Interactive Media, Animation & Games
Howest - University College West-Flanders	www.digitalartsandentertainment.com	Kortrijk, West-Flanders, BE	Digital Arts & Entertainment
HTW- Hochschule fÃ³r Technik und Wirtschaft	gamedesign.htw-berlin.de/	Berlin, Berlin, DE	Interaction Design / Game Design
Humber College - School of Media Studies	humber.ca	Toronto, ON, CA	3D Animation, Art and Design, 3D Modelling and Visual Effects Production, Game Programming
iD Game Design & Development Academy for Teens	www.idtech.com/teens/game-design-development/	Held at Stanford, Harvard, Emory, the University of Washington, Vassar, the University of Denver, Villanova, TCU, and Lake Forest College	3D Modeling & Animation with Autodesk Maya, Game Development for iPhone and Android with Unity & Javascript, 3D Level Design Unreal Engine
Illinois Institute of Art - Chicago	www.ilic.aii.edu	Chicago, IL	Media Arts & Animation, Game Art & Design, Digital Media Production, Visual Effects & Motion Graphics
Image College of Arts, Animation & Technology (ICAT)	www.icat.ac.in	Chennai, TN, IN	Visual Effects, 3D Animation, Game Development, Game Design
Indiana University	games.indiana.edu/	Bloomington, IN	Game Design - Art, Game Design, Game Design - Programming, Game Design - Management, Game Design - Audio
Informatics Professional Development Centre	http://www.informatics.edu.sg/	Singapore, SG	Gaming and Animation Technology
Institute Desgraff	www.institutdesgraff.com	Sherbrooke, Quebec, CA	3D digital arts and animation for video games
Institute of Technology Carlow	www.itcarlow.ie	Carlow, Carlow, IE	Computer Games Development
Instituto Tecnología y de Estudios Superiores de Monterrey - Campus Estado de México	www.cem.itesm.mx	Atizapán de Zaragoza, Estado de México, MX	Animación y Arte Digital/Computer Graphics Animation

TITLE	URL	LOCATION	PROGRAM OFFERED
Inter-Dec college	www.interdeccollege.com	Montreal, Quebec, CA	Video games, 3D animation
ISART Digital	www.isartdigital.com/	Paris, FR	CG Animation, Production, Visual Effects, Game Art, Game Design and Programming , Sound Design, Game Design, Game Programming
Istanbul Yildiz Technichal University - Communications Design And Multimedia	http://www.yildiz.edu.tr/en/	Istanbul, Marmara, TR	Digital-interactive media
Istituto Europeo di Design	www.ied.it	Rome, RM, IT	Masters in VIdeo Game Design and Development
IT University of Copenhagen	www.itu.dk	Copenhagen, DK	Media, Technology and Games
ITP at New York University	itp.nyu.edu	New York, NY	Interaction Design
ITT Tech - Greenville, South Carolina	itt-tech.edu/	Greenville , SC	Digital Entertainment and Game Design Bachelor of Science Degree, Visual Communications Associate of Applied Science Degree
ITT Technical Institute - Arnold, Missouri	www.itt-tech.edu/campus/school.cfm?lloc_num=91	Arnold, MO	Digital Entertainment and Game Design
Kajaani University of Applied Sciences	www.kajak.fi/?deptid=13937	Kajaani, FI	Game Programming
Karelia University of Applied Sciences	tiko.ncp.fi/gameprogramming/index_en.html	Joensuu, Finland, FI	Game Programmer
Keiser University	www.keiseruniversity.edu	Fort Lauderdale, FL	Video Game Design
Lake Washington Technical College	http://www.lwtech.edu/	Kirkland, WA	Print/Web, Digital Games and Media, Video Production
Lancaster University UK	www.dcs.lancs.ac.uk/admissions/postgraduate_courses.php?course_id=008623	Lancaster, Lancashire, GB	Mobile games development
Langara College Continuing Studies	www.langara.bc.ca/cs	Vancouver, BC, CA	Business & Computer Technology
Living Arts College @ Living Arts College	www.living-arts-college.edu	Raleigh, NC	Digital Audio Production & Design, Digital Filmmaking , Interactive Media Arts, Digital Photography, Animation & Game Design
Long Island University	www.liu.edu/post/gamedesign	Brookville, NY	Digital Game Design and Development
Los Angeles Film School - Game Production	www.lafilm.edu	Hollywood, California	Game Production
Lost Boys Learning	www.lostboys-learning.com	Courtenay, BC, CA	Visual Effects

TITLE	URL	LOCATION	PROGRAM OFFERED
Louisiana State University	www.lsu.edu	Baton Rouge, Louisiana	Digital Media Minor (DMTEC) - Undergraduate, Digital Media Minor (DMART) - Undergraduate
Louisiana State University Digital Media Arts & Engineering	dmae.lsu.edu	Baton Rouge, LA	Digital Media Arts & Engineering
LSC - Kingwood	http://www.lonestar.edu/kingwood.htm	Kingwood, TX	Game Design & Simulation Designer, 3D Animation
Madeira Interactive Technologies Institute/University of Madeira	www.m-iti.org	Funchal, R. A. Madeira, PT	Masters of Entertainment Technology, partnership with ETC/Carnegie Mellon University - USA
Madison Media Institute | College of Media Arts	www.mediainstitute.edu	Madison, WI	Recording & Music Technology, Video And Motion Graphics, Game Art & Animation
Massachusetts Institute of Technology (MIT)	web.mit.edu/admissions/	Cambridge, MA	Electrical Engineering and Computer Science, Sloan School of Management, Comparative Media Studies
Massasoit Community College	massasoit.edu/	Brockton, MA	Computer Information Systems - Programming Option
Max the Mutt Animation School	www.maxthemutt.com	Toronto, Ontario, CA	Classical & Computer Animation and Production, Illustration for Sequential Arts: Comic Books and Graphic Novels, Concept Art
McGill University, School of Computer Science	www.cs.mcgill.ca/	Montreal, QC, CA	Major in Computer Science Computer Games Option
MD.H Mediadesign-Hochschule	mediadesign.de/	Berlin, DE	Game Design
Media Design School	www.mediadesignschool.com	Auckland, NZ	Bachelor of Art and Design (3D & VFX), Bachelor of Media Design, Bachelor of Software Engineering (Game Programming), Graduate Diploma in Creative Technologies, Bachelor of Creative Technologies (Game Art)
Mesa Community College	http://www.mesacc.edu/	Mesa, AZ	Assosciate of Applied Science in Game Technology
Michigan State University	seriousgames.msu.edu	East Lansing, Michigan	Game Design & Development Specialization, Serious Game Design MA
Middlesex University	www.mdx.ac.uk	London, GB	3D Animation and Games, Multimedia Computing, Animation, Computing, Graphics and Games
Mildred Elley - School of Digital Media Arts	www.mildred-elley.edu	Albany, NY	Game Design & 3D Animation, Digital Graphics & Multimedia Design, Ditial Graphics & Multimedia Design
Milwaukee Area Technical College	www.matc.edu	Milwaukee, WI	Animation
Minneapolis Media Institute	http://www.mediainstitute.edu/minneapolis-media-institute	Edina, MN	Game Art & Animation
Missouri State University - West Plains	wp.missouristate.edu/Academics/default.htm	West Plains, MO	Computer Graphics & Programming

TITLE	URL	LOCATION	PROGRAM OFFERED
Mohawk Valley Community College	https://www.mvcc.edu/academic-programs/degrees/computer-science	Utica, NY	Computer Science
Monash University - Berwick School of Information Technology	www.infotech.monash.edu.au/berwick	Berwick, Victoria, AU	Bachelor of Information Technology and Systems - Games Development, Bachelor of Information Technology and Systems - Multimedia Applications
Montgomery College	www.studygaming.com	Rockville, MD	Computer Gaming and Simulation - ProgrammingTrack, Computer Gaming and Simulation - Production & Design Track, Computer Gaming and Simulation - Art & Animation Track, Internet Gaming and Simulation - Certificate
Montgomery County Community College	mc3.edu	Blue Bell, Pennsylvania	Digital Design: Computer Graphics, Electronic Game and Simulation Design, Digital Design: Multimedia, Digital Audio Production, Software Engineerin
Motherwell College	www.motherwell.ac.uk/	Motherwell, North Lanarkshire, GB	National Certificate/Diploma
Mount Ida College	www.mountida.edu	Newton, MA	BS in Game Art and Animation
Mt. Sierra College	www.mtsierra.edu	Monrovia, CA	Media Arts & Design, Game Arts & Design
Murdoch University	www.games.murdoch.edu.au/	Perth, Western Australia, AU	Games Technology, Games Software Design and Production
Musitechnic	musitechnic.com/en/	Montreal, Quebec, CA	Audio production technique
Napier University	http://www.napier.ac.uk/Pages/home.aspx	Edinburgh, GB	Interactive Media Design, Interactive Media Design
National Centre for Computer Animation, Bournemouth University	ncca.bournemouth.ac.uk	Poole, Dorset, GB	MA/PD, Computer Animation Arts, BA, Digital Effects, MA/PD, Software Development for Animation, Games and Effects, BSc , Computer Visualisation & Animation, BA , Computer Animation & Visual Effects, MSc
National Film and Television School	www.nfts.co.uk/courses/games-design-and-development	Beaconsfield, Buckinghamshire, GB	MA Games Design and Development
National University - School of Media & Communication	www.nu.edu/Academics/Schools/SOMC.html	La Jolla, CA	Digital Entertainment and Interactive Arts, Video Game Production and Design
NBCC Miramichi	www.nbcc.ca/miramichi	Miramichi, NB, CA	Applied Arts - Electronic Game - Design (programming), Applied Arts Diploma of Advanced Studies, Applied Arts - Animation and Graphics, Applied Arts - Electronic Game - 3D Graphics
Nescot - National Diploma Game Development	www.nescot.ac.uk/	Epsom, Surrey, GB	National Diploma Game Development

TITLE	URL	LOCATION	PROGRAM OFFERED
Neumont University	www.neumont.edu/	South Jordan, Utah	Computer Science
New England Institute of Technology	www.neit.edu	East Greenwich, RI	Video and Audio Production Technology, Software Engineering Technology, Video Game Design, Digital Recording Arts, Graphics, Multimedia and Web Design, Game Development and Simulation Programming Technology
New Jersey Institute of Technology	www.njit.edu	Newark, New Jersey	Information Technology
New York Film Academy	www.nyfa.edu/game-design-school/	New York, NY	Game Design
New York University	gamecenter.nyu.edu/	Brooklyn, New York	Game Design M.F.A, Game Design B.F.A
New York University - School of Continuing and Professional Studies	www.scps.nyu.edu/dcom	New York, NY	Digital Communications and Media
NHTI Concord Community College	https://www.nhti.edu/	Concord, New Hampshire	Animation and Graphic Game Programming
NHTV Breda University of Applied Science	www.nhtv.nl/made	Breda, NL	International Game Architecture and Design
North Carolina State University - College of Design	http://design.ncsu.edu/	Raleigh, NC	Multimedia & Digital Imaging, 3D Animation, Interactive Design, Ideation & Illustration, Game Design
Northeastern University - College of Professional Studies	http://www.cps.neu.edu/	Boston, MA	Interactive Design, Digital Video, Game Design, 3D Animation
Northeastern University - Creative Industries (Game Design & Interactive Media)	http://www.cps.neu.edu/degree-programs/graduate/masters-degrees/masters-digital-media.php	Boston, MA	Interactive Media and Digital Art, Interactive Media and Music Technologies, Creative Industries Minor, Game Design and Digital Art, Game Design and Computer Science, Interactive Media and Graphic Design,
Northern Oklahoma College	http://www.noc.edu/	Tonkawa, OK	3D Animation and Post-production
Northumbria University	www.northumbria.ac.uk/sd/academic/ceis/	Newcastle upon Tyne, Tyne & Wear, GB	Games Programming, Computer Games Design & Production
Norwegian School of Information Technology	www.nith.no	Oslo, NO	Game design, Game programming
Norwich University College of the Arts	www.nuca.ac.uk	Norwich, Norfolk, GB	Games Art and Design
Nova Scotia Community College	www.nscc.ca	Truro, NS, CA	Interactive & Motion Graphics - 3D Modelling & Motion Capture, Interactive & Motion Graphics - Visual Effects, Interactive & Motion Graphics - Game Design
NTI Birmingham - Nano	www.bcu.ac.uk/pme/nti/gamercamp/courses/nano	Birmingham, GB	Gamer Camp: Nano
NTI Birmingham - Mini	www.bcu.ac.uk/pme/nti/gamercamp/courses/mini	Birmingham, GB	Gamer Camp: Mini - PgCert/PgDip/MA/MSc

TITLE	URL	LOCATION	PROGRAM OFFERED
NTI Birmingham - Pro	www.bcu.ac.uk/pme/nti/gamercamp/courses/pro	Birmingham, GB	Gamer Camp: Pro, Video Games Development - MA / MSc
Ohio University	http://mediaschool.ohio.edu/ga	Athens, OH	Digital Media: Special Effects, Games, and Animation
Oklahoma Christian University	www.oc.edu/art	Edmond, OK	Oklahoma Christian U. Gaming + Animation
Oklahoma City Community College	www.occc.edu	Oklahoma City, Oklahoma	Computer Aided Technology- Game Design Option
Oklahoma Panhandle State University	www.opsu.edu	Goodwell, OK	BFA Computer Graphics, BTEC Game Art Design
Otis College of Art and Design - Digital Media Department	http//www.otis.edu	Los Angeles, CA	game design, animation, interactive design, motion graphics, visual effects
Parsons the New School of Design: School of Art, Media and Technology	cdt.parsons.edu	New York, NY	Design & Technology
Pennsylvania College of Technology	www.pct.edu	Williamsport, PA	Information Technology Sciences: Gaming & Simulation
Pennsylvania State University	ist.psu.edu	University Park, PA	Information Sciences and Technology
Pensacola State College	http://www.pensacolastate.edu/	Pensacola, FL	Simulation and Game Design
Piedmont Community College	www.piedmontcc.edu	Roxboro/Yanceyville, NC	Digital Effects and Animation Technology
Pinnacle College	www.pinnaclecollege.edu	Alhambra, CA	Audio for Games and Interactive Media
Platt College - Digital Media Design	www.platt.edu	San Diego, CA	Media Arts (Video Production), Media Arts (Visual Effects & Compositing), Multimedia/Animation, Media Arts (3D Animation), Graphic Design, Web Design, Media Arts (Web Design), Media Arts
PlaygroundSquad	www.playgroundsquad.com	Falun, SE	Programming, Game Design, 3D Graphics
PowerUp Games	www.powerupgames.com/game-tester-course.html?cid=7&pid=5	Jacksonville, FL	Game Testing Certification
Pyramind: the Institute for Advanced Digital Audio Training	www.Pyramind.com	San Francisco, CA	Music and Sound for Picture and Video Games
QANTM College - Australia	www.qantmcollege.edu.au	Brisbane, QLD, AU	Bachelor of Interactive Entertainment (with a Major in Games Programming), Bachelor of Interactive Entertainment (with a Major in Animation), Bachelor of Interactive Entertainment (with a Major in Games Design),
QANTM College - Munich	www.qantm.de	Munich, 81737 DE, Bavaria, DE	Bachelor of Arts, Interactive Animation, Bachelor of Science, Games Programming
Queen's University Belfast - School of EEECS	www.qub.ac.uk/eeecs	Belfast, Antrim, GB	Computer Games Design and Development
Queensland University of Technology - Kelvin Grove, QLD	www.creativeindustries.qut.edu.au	Kelvin Grove, QLD, AU	Business, Game Development and Entertainment

TITLE	URL	LOCATION	PROGRAM OFFERED
Queensland University of Technology -Brisbane City	www.qut.edu.au/scitech	Brisbane, QLD , AU	Master of Information Technology - Games Design/Production, Bachelor of Games and Interactive Entertainment
Quinnipiac University	www.quinnipiac.edu/	Hamden, CT	Game Design and Development
Rasmussen College	www.rasmussen.edu/degrees/technology-design/game-design/	Online and at 25 campuses in 5 states	Multimedia Technologies, Game & Simulation Programming, Digital Design and Animation, Information Systems Management
Ravensbourne College Of Design & Communication	www.rave.ac.uk	Chislehurst, Kent, GB	Interactive Digital Media*, Computer Visualisation and Animation, Animation
Recording Arts Canada	recordingarts.com/programs/game-design	Toronto, Ontario, CA	Game Design
Rensselaer Polytechnic Institute	www.gsas.rpi.edu	Troy, NY	Game and Simulation Arts & Sciences
Ringling College of Art and Design	www.ringling.edu	Sarasota, Florida	Game Art & Design, Computer Animation
Rochester Institute of Technology	www.rit.edu	Rochester, NY	New Media Interactive Development, Information Technology, Software Engineering, Computer Science, 3-D Digital Design, Film and Animation, Game Design & Development, New Media Design , Visual Communications Design
S4G School for Games	www.school4games.net	Berlin, Berlin, DE	Online Game Graphics, Online Game Development, Online Game Engineering
Sacred Heart University	www.sacredheart.edu/pages/35_computer_science_information_technology.cfm	Fairfield, CT	Game Design and Development
SAE Institute (Singapore Campus)	sae.edu.sg	Singapore, Singapore, SG	Diploma of IT, Bachelor of Science (Hons) in Games Programming
SAE Institute Amsterdam	amsterdam.sae.edu	Amsterdam, Noord-Holland, NL	Audio Engineering Course
SAGE	http://www.sageinfolabs.com	New Delhi, New Delhi, IN	Game Programming
Sam Houston State University	www.shsu.edu/~animate	Huntsville, TX	Computer Animation
San Jacinto College Central	http://www.msjc.edu/Pages/default.aspx	Houston , TX	3D Animation courses (I,II,and III)
Santa Ana College	http://www.sac.edu/	Santa Ana College, CA	3D Animation Certificate
Santa Monica College - Academy of Entertainment Technology	academy.smc.edu	Santa Monica, CA	Game Development, Post Production, Visual Effects, Animation
Savannah College of Art and Design	www.scad.edu	Savannah, GA	Sound Design, Visual Effects, Interactive Game Design & Motion Graphics, Animation

TITLE	URL	LOCATION	PROGRAM OFFERED
School of Animation, Communication University of China	animation.cuc.edu.cn/	Beijing, CN	Animation Design, Animation, Animation, Digital Film and Television Production, Digital Media Art, Digital Media Art, Game Programming direction of Game Design, Animation Writing and Directing
School of Arts and Humanities, University Campus Suffolk	www.ucs.ac.uk	Ipswich, Suffolk, GB	Computer Games Design
School of Communication American University	www.american.edu/soc/	Washington, DC	MA In Game Design
School of Computer Science, University of Windsor	www.cs.uwindsor.ca	Windsor, Ontario, CA	Game Development Specialization
School of Computing & Mathematical Sciences, Liverpool John Moores University	www.cms.livjm.ac.uk/	Liverpool, GB	Interactive Media Design, Animation for Film and Games, Computer Games Technology
School of Computing and Intelligent Systems, Faculty of Computing and Engineering, University of Uls	www.ulster.ac.uk	Derry, County Derry, GB	BSc (Hons) Multimedia Computer Games, BEng (Hons) Computer Games Development
School of Multimedia Systems	http://www.monash.edu.au/pubs/handbooks/aos/games-development/	Berwick, Victoria, AU	Games Development
School of Video Game Audio	School.VideoGameAudio.com	Vancouver, CA	Wwise Demo Reel, FMOD Demo Reel
School of Visual Arts	www.sva.edu	New York, NY	BFA Computer Art, Computer Animation and Visual Effects, MFA Computer Art, BFA animation, BFA Computer Art
Seamedu - Media School	www.seamedu.com	Pune, Maharashtra , IN	BTEC Level 5 HND Creative Media Production (Computer Game Design), Diploma in Game Development
Seattle Central Community College	www.learnatcentral.org	Seattle, WA	3D Animation / Design & Gaming Program
Seneca College - Game Art & Animation	http://www.senecacollege.ca/fulltime/GAA.html	Toronto, Ontario, CA	Animation Advanced Diploma Program, 3D Animation Graduate Certificate Program (dan), 3D Gaming Graduate Certificate Program (GAM)
Serious Game Design Institute	http://seriousgamedesigninstitute.org/	Santa Barbara, CA	Serious Game and Simulation Design
Sessions College for Professional Design	www.sessions.edu/certificate-programs/game-art	Tempe, AZ	Accredited Game Art Certificate
Shawnee State University	www.shawnee.edu	Portsmouth, OH	Game Programming, Game Graphics
Sheffield Hallam University	www.shu.ac.uk/multimedia/games/	Sheffield, South Yorkshire, GB	MA Animation for Computer Games, BSc Game Software Development, MSc Game Software Development
Sheridan College	www.sheridancollege.ca	Oakville, Ontario, CA	Game Development - Advanced Programming, Bachelor of Game Design, Game Level Design, Bachelor of Interaction Design
Sierra College	cs.sierracollege.edu	Rocklin, CA	Computer Science
Simon Fraser University - School of Interactive Arts and Technology (SIAT)	www.siat.sfu.ca	Surrey, British Columbia, CA	Undergraduate Program: Media Arts, Design & Informatics Concentrations, Graduate Program

TITLE	URL	LOCATION	PROGRAM OFFERED
Southbank Institute of Technology	http://tafebrisbane.edu.au/	South Brisbane, QLD, AU	Diploma of Information Technology, Games Development
Southbank Institute of Technology	http://tafebrisbane.edu.au/	Brisbane, Queensland, AU	Diploma of Interactive Digital Media - Game Design
Southern Adventist University - School of Visual Art and Design	art.southern.edu/	Collegedale, TN	Animation, Fine Art, Interactive Media, Film Production, Graphic Design
Southern New Hampshire University SNHU	www.snhu.edu	Manchester, NH	Computer Information Technology - Game Design & Development, Game Design & Development, Business Studies - Game Design & Development, Information Technology - Game Design & Development
Southern Polytechnic State University	games.spsu.edu/	Marietta, GA	Computer Game Design and Development, Software Engineering, Computer Science, Information Technology
Springfield College	www.spfldcol.edu/	Springfield, MA	Game Design
St. Edward's University,	www.stedwards.edu/business/graduate/mbad/index.htm	Austin, TX	Digital Media Management
St. Petersburg College-Seminole Campus	http://www.spcollege.edu/	Seminole , FL	Digital Media Video production, Digital Media Production, Video Game Foundations
Staffordshire University	www.staffs.ac.uk	Stafford, GB	Computer Games Programming, Multiplayer Online Games Programming, Computing: Games Development
Stony Brook University	www.cs.stonybrook.edu	Stony Brook, NY	Computer Science Specialization in Game Programming
Stroud College in Gloucestershire	www.stroud.ac.uk/	Stroud, Gloucestershire, GB	Multimedia - BTEC National Diploma
Swansea Metropolitan University - School of Digital Media	www.sihe.ac.uk/	Swansea, Swansea, GB	Computer Games Development, BSc Computer Games Development, BA Creative Computer Games Design
TAFE - Tea Tree Gully	www.tafesa.edu.au/game-art-studies.aspx	Modbury, SA, AU	Game Art VET, Game Art Program
TAFE N.S.W. Hornsby	-	Hornsby, New South Wales, AU	19010 - Programming, 19050 - Games Development (Diploma), 7872 - Digital Media
TAFE NSW - Wollongong West	www.illawarra.tafensw.edu.au	Wollongong, NSW, AU	Digital and Interactive Games (Programming),
Tallahassee Community College	https://www.tcc.fl.edu/College/TypeOfProgram/Pages/2158.aspx	Tallahassee, Florida	Computer Game Design
Temasek Polytechnic (Temasek InfoTech School)	http://www.tp.edu.sg/	Singapore, Singapore, SG	Game & Entertainment Technologies
Texas State Technical College	www.waco.tstc.edu	Waco, Texas	Game Programming and Design, Game & Simulation

TITLE	URL	LOCATION	PROGRAM OFFERED
The Academy of Entertainment and Technology at Santa Monica College	academy.smc.edu	Santa Monica, CA	Animation, Game Design
The Art Institute of Vancouver	www.artinstitutes.edu/vancouver/	Vancouver, British Columbia, CA	Visual & Game Programming, Game Art & Design
The Art Institutes System of Schools	www.artinstitutes.edu/game-design-and-programming-2602.	Pittsburgh, PA	Game Art & Design, Visual Effects & Motion Graphics, Game Programming, 3D Modeling for Animation & Games, Media Arts & Animation, Animation Art & Design, Visual & Game Programming, VFX for Film & Television, Computer Animation
The College of Westchester	www.cw.edu	White Plains, New York	Multimedia Development & Management
The Game Assembly	www.thegameassembly.com	Malmö, SE	Game Art
The Rydan Workshop	www.therydanworkshop.com	North Vancouver, BC, CA	Intro Digital Concept Art, 2D Matte Painting, Digital Character and Creature Design, Intro to Maya Rigging, Game Character Pipeline, Real-Time Environments with Unreal Engine, Photoshop Intro for the Entertainment Industry,
Train2Game	www.train2game.com	Nationwide, United Kingdom, GB	TIGA Diploma in Games Design, TIGA Diploma in Games Development
Training Center Alcance Digital	www.alcancedigital.com	León, Guanajuato , MX	Digital Cinematography, Video Games Programming, Digital Musical Composition, 3D Animation, Digital Design, Visual Effects
Tribeca Flashpoint Media Arts Academy	www.tfa.edu	Chicago, IL	Animation + Visual Effects, Recording Arts, Game & Interactive Media, Film + Broadcast, Design + Visual Communications
Trinity College Dublin	www.cs.tcd.ie/courses/msciet/5	Dublin	MSc in Interactive Entertainment Technology
TriOS College	www.getintothegame.ca	London, Kitchener, Hamilton, Mississauga and Toronto, Ontario, CA	Video Game Design and Development + Internship
Tulsa Tech	http://tulsatech.edu/Pages/Default.aspx	Tulsa, Oklahoma	Information Technology-3d Design & Animation
Tyler Junior College	www.tjc.edu	Tyler, TX	Gaming and Simulation Development--Graphics, Gaming and Simulation Development--Programming
Universidad Iberoamericana Leon	www.leon.uia.mx	León, Guanajuato, MX	Digital Design
Universidade Anhembi Morumbi	www.anhembi.br	São Paulo, São Paulo, BR	Game Design
Universidade do Vale do Rio dos Sinos	http://www.unisinos.br/	São Leopoldo, Rio Grande do Sul, BR	Digital Games
University Campus Oldham	www.uco.oldham.ac.uk/	Oldham, Greater Manchester, GB	Digital Arts Practice (Games Art)
University for the Creative Arts at Farnham	http://www.ucreative.ac.uk/	Farnham, Surrey, GB	BA Computer Games Arts

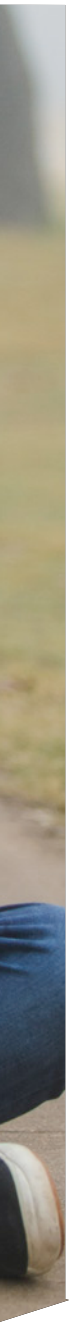
TITLE	URL	LOCATION	PROGRAM OFFERED
University of Abertay Dundee	www.abertay.ac.uk	Dundee, Scotland, GB	Computer Games Technology, Game Design & Production Management, Computer Game Applications Development, Computer Arts, Game Art & Animation, Creative Sound Production, Games Development
University of Advancing Technology	www.uat.edu	Tempe, AZ	Game Art and Animation, Serious Game and Simulation , Game Design, Game Programming, Game Production and Management
University of Applied Sciences Salzburg / FH Salzburg	www.fh-salzburg.ac.at	Puch bei Hallein , Salzburg, AT	Master MultiMediaArt / Producing, Master MultiMediaArt / audio - content creation, Master MultimediaTechnology, Master MultiMediaArt / visual content creation
University of Applied Sciences Technikum Wien	www.technikum-wien.at/en/	Vienna, Wien, AT	Computer Science, Game Engineering and Simulation Technology
University of Baltimore	www.ubalt.edu/games	Baltimore, MD	Simulation and Digital Entertainment
University of Birmingham	www.cs.bham.ac.uk	Birmingham, UK, GB	BSc Computer Science, BSc Artificial Intelligence and Computer Science, MSci Computer Science
University of Bradford - School of Informatics	www.inf.brad.ac.uk/	Bradford, West Yorkshire, GB	Interactive Systems and Video Games Design, Artificial Intelligence for Games , Design for Computer Games
University of California, Santa Cruz	games.soe.ucsc.edu	Santa Cruz, CA	Computer Game Design Ph.D., Digital Arts & New Media / Playable Media, Computer Game Design
University of California San Diego | Extension Digital Arts Center	dac.ucsd.edu	San Diego, CA	Video & Editing, Graphic & Web Design, Mobile Applications Development
University of California, Irvine	www.ics.uci.edu	Irvine, CA	Computer Science, Computer Game Science, Digital Arts
University of Central Florida (FIEA)	www.fiea.ucf.edu	Orlando, FL	Production Track, Art Track, Programming Track
University of Central Lancashire	www.uclan.ac.uk	Preston, Lancashire, GB	Games Design, Computer Games Development
University of Colorado, Colorado Springs	http://www.uccs.edu/	Colorado Springs, CO	Bachelor of Innovation(TM) in Game Design and Development, Game Design and Development Minor
University of Denver	www.gamedev.cs.du.edu	Denver, CO	Animation and Game Development, Computer Science, Electronic Media Arts Design, Studio Art, Digital Media Studies
University of East London	http://www.uel.ac.uk/	London, GB	Computer Games Design (Story Development)
University of Glamorgan	www.glam.ac.uk/coursedetails/685/53	Pontypridd, RCT, GB	Computer Game Development

TITLE	URL	LOCATION	PROGRAM OFFERED
University of Gotland	http://www.campusgotland.uu.se/en	Visby, Gotland, SE	International Game Production Studies Degree, International Game Production Studies II, International Game Production Studies I
University of Houston	games.cs.uh.edu	Houston, TX	Game Development
University of Houston-Victoria	www.uhv.edu/asa	Victoria, Texas	BAAS Digital Gaming Simulation, BS Computer Science-Digital Gaming and Simulation
University of Hull - Department of Computer Science	www.net.dcs.hull.ac.uk/	Hull, E Yorkshire, GB	Games Programming
University of Idaho Virtual Technology & Design	http://www.uidaho.edu/	Moscow, Idaho	Virtual Technology and Design
University of Lincoln	http://www.lincoln.ac.uk/home/	Lincoln, UK, GB	Advanced Games Programming, Games Computing
University of Maryland, Baltimore County	gaim.umbc.edu/	Baltimore, Maryland	Visual Arts + animation and interactve media, Computer Science + game development track
University of Miami - Music Engineering	www.music.miami.edu/programs/mue/	Coral Gables, FL	Music Engineering
University of Michigan - EECS Department	www.eecs.umich.edu	Ann Arbor, MI	Computer Science, Computer Engineering
University of Montevallo	www.montevallo.edu/	Montevallo, AL	Game Studies and Design
University of Montreal	www.dej.umontreal.ca	Montreal, Quebec, CA	Graduate Diploma in Game Design
University of Ontario Institute of Technology	www.uoit.ca/	Oshawa, Ontario, CA	Game Development and Entrepreneurship
University of Otago	www.otago.ac.nz	Dunedin, Otago, NZ	Computer Game Design, Artificial Intelligence, Computer Graphics, Software Engineering
University of Pennsylvania	www.cis.upenn.edu/grad/cggt/	Philadelphia, PA	Digital Media and Design, Human Modeling and Simulation, Computer Graphics & Game Technology
University of Portsmouth - School of Creative Technologies	www.port.ac.uk/games	Portsmouth, Hampshire, GB	Computer Games Technology
University of Prince Edward Island	www.upei.ca/csit/	Charlottetown, PE, PEI, CA	BSc CS with Specialization in Video Game Programming
University of Skövde	www.his.se	28 Skövde, Högskolevägend, SE	Graphics, Programming, Serious Games, Game Writing, Design, Sound/Music, Media, Aesthetics and Narrative
University of Southern California - Interactive Media Division	cinema.usc.edu	Los Angeles, California	Interactive Media, Interactive Entertainment
University of Technology Sydney (UTS)	www.uts.edu.au/	Sydney, NSW, AU	Master of Interactive Multimedia, Master of Animation, Bachelor of Science in Games Development
University of Teesside	http://www.tees.ac.uk/	Middlesbrough, Cleveland, GB	Interactive Computer Entertainment, Computer Games Art, Computer Games Design, Virtual Reality, Visualisation, Computer Animation
University of Texas at Austin	gamedev.utexas.edu	Austin, TX	Game Development Program

TITLE	URL	LOCATION	PROGRAM OFFERED
University of Texas at Dallas - Arts and Technology program	atec.utdallas.edu	Richardson, TX	Arts and Technology
University of the Philippines IT Training Center	http://itdc.up.edu.ph/	Quezon City, NCR, PH	Game Development Track
University of Utah: Entertainment Arts and Engineering Master Games Studio	eae.utah.edu/	Salt Lake City, UT	Entertainment Art and Engineering Master Games Studio
University of Utrecht	www.uu.nl/	Utrecht, NL	Game and Media Technology
University of Verona	www.mastergamedev.it/	Verona, Italy, IT	Master in Computer Game Development
University of Wales, Newport	http://www.southwales.ac.uk/	Newport, South Wales, GB	Computer Games Design
University of Washington - Bothell	www.bothell.washington.edu	Bothell, WA	Interactive Media Design, Media and Communication Studies
University of Washington - Professional and Continuing Education	www.pce.uw.edu	Seattle, WA	Game Development, Virtual Worlds, 3D Animation for Games & Digital Media
University of Waterloo	www.cs.uwaterloo.ca/	Waterloo, Ontario, CA	Computer Science and Computer Engineering
University of West Scotland	www.uws.ac.uk/	Paisley, Renfrewshire, GB	Computer Game Technology
University of Western Ontario	www.csd.uwo.ca	London, Ontario, CA	Computer Science with Minor in Game Development
University of Winnipeg - Professional, Applied and Continuing Education	www.UWinnipegCourses.ca	Winnipeg, Manitoba, CA	Serious Games Certificate
University of Wisconsin-Stout	www.uwstout.edu/programs/bsgdd/index.cfm	Menomonie, WI	Game Design & Development-Art, Game Design & Development-Computer Science
University of Wisconsin Whitewater	uww.edu/games	Whitewater, Wisconsin	Media Arts and Game Development
UPC School of Professional and Executive Development	www.talent.upc.edu/professionals/presentacio/codi/20801200/	Barcelona, Barcelona (Spain), ES	Videogame Design and Creation
Utrecht School of the Arts/ Faculty of Art, Media & Technology	www.hku.nl/	Hilversum, CL, NL	Game Design & Development
Vancouver Animation School	www.vanas.ca	Vancouver, British Columbia, CA	Concept Art, 3D Computer Character Animation
Vancouver Film School	www.vfs.com	Vancouver, BC, CA	Classical Animation, Sound Design, Programming for Games, Web & Mobile, 3D Animation, Digital Character Animation (Maya), Game Design
Vancouver Institute of Media Arts (VanArts)	www.vanarts.com	Vancouver, BC, CA	Web Development & Interactive Design, Game Art & Design, 2D/3D Character Animation, Visual Effects
Victoria University	computergraphics.ac.nz	Wellington, NZ	Computer Graphics
Visual College of Art & Design	www.vcad.ca	Vancouver, British Columbia, CA	3D Modelling Animation Art & Design

Rebekah and Adam Saltsman with their two boys.

GAMES AND PARE
HOW DO YO



NTHOOD: U HANDLE IT?

Rebekah Saltsman

Making games is tough. Making games as a parent is even tougher.

Over the years, many words have been used to describe me. I have been a daughter, a sister, a teammate, a captain, a student, a teacher, a coach, a girlfriend, a fiancée, and a wife. All of these titles have had a tremendous impact on what makes me, well, me. In 2011, I added one more important title—one that will never expire. I became a mom.

Motherhood is a loaded term. There are a million ways to be a mother, and, unfortunately, you will be told there is only one way to do it right by every passing stranger, friend, family member, and acquaintance. The mommy wars are real, mommy guilt is par for the course, and juggling the isolation of new motherhood and a near-complete change in your priorities and values is daunting to even the best-prepared parent.

Let's add a bit to that though—I am also the CEO of Finji, a small independent game company in Austin, Texas. I run this small studio with my more famous other half, Adam "Atomic" Saltsman. We develop our own internal IP and help bring other games created by small independent teams to market through our Partner Publishing branch. I am integral to the daily operations of the company. I contribute to design, handle finances, legal affairs, and console partnerships, and monitor and co-mentor those Partner Publishing projects.

I am also the "default parent" to our two boys, Kingsley and Finnegan. This is the term I use to define the parent who performs most of childcare every day. I wake up with the boys, I cook the majority of the meals, give the baths, and I make sure they get to preschool, gymnastics, dance, and swim classes. I am the parent who drives around for daily errands, like going to the grocery store. Because I'm the "default parent," the kids usually come to me first for skinned knees, bloody lips, night terrors, and, of course, mediating fights.

My days start at 6:30 a.m. (if I am really, really lucky) and end at 8:30 p.m. (if I am mostly lucky). From the moment I leave my bed to the moment the boys stop leaving theirs, I am a mom first and a CEO second. I have limited dedicated daily work time set aside, and most of it is scheduled from 9-11 p.m., especially when we are late in development on our internal projects and Adam needs more computer time.

Each day, I am attached to my devices. I take my laptop and smartphone everywhere. I check and respond to email while sitting at swim class, while filling the bathtub, even while making dinner with a 2-year-old strapped into the child carrier on my back.

Along with the emotional consequences of one parent taking on the bulk of the parenting responsibilities, we also have to navigate the challenges of scheduling and budgeting our time, finances, and home life. We have to be creative and efficient. I hope that the preparations we made to be both parents and indies will help you as you navigate working in the industry yourselves.

How we budget

One of the questions I frequently receive when talking about how to have a family while running a small company is often gasped with a heavy note of incredulity: "How in the world can you afford to have kids and make indie video

games?" There is no easy answer to this question—but I can tell you how we got to the point where we are now, and how we maintain our financial stability.

First, I always know exactly how much we need to maintain something we call our "monthly burn rate." We have several of these "monthly burn rate" numbers that we reference as we plan our finances each month, quarter, and year. We have calculations for how much we pay in advances and expenses at the company level. Next, and perhaps most importantly, we know the amount we need for monthly personal expenses. I have several tiers to our personal monthly burn rate, including the bare minimum amount of money we need to not go hungry or move to a new home.

The total number I use is essentially the amount of money I need each month to live comfortably. This is the number I need to put in the checking account on the first day of the month that will cover the mortgage, utilities, food, lessons, entertainment, and fun family adventures.

Coming by our monthly burn rate was not dumb luck, nor was it a few minutes of extra work. It was a lengthy process in which I tracked every single cent we spent for three years. I analyzed our spending, found trouble spots in our budget, and fixed them. I discovered that I spent three times the amount budgeted for Target when I shopped there because I wandered the store instead of arriving with a list. I discovered that we were spending over $800 a month on food (and we didn't have kids yet!), because we were too lazy to cook and dined out despite buying groceries to cook meals every night. I discovered exactly how much it cost us to have two pugs, a breed that is notoriously expensive to maintain because of their health problems. This information is still completely invaluable. It showed me how we interacted with money on a day-to-day basis. And most importantly, it showed us how to rein it all in—how to live smart, lean, and debt-free.

I did this all *before* either of us quit our "real jobs." I knew our spending habits and got our financial house in order before taking the plunge into the uncertainty of freelancing and indie development. Most importantly, I learned exactly how to tighten our belt financially when and if the need ever arose. I also made it a point to practice these methods before the need ever arose. For example, if you need to cut your food budget in half, can you do that? What does that look like? What kind of food can you buy and cook? In an effort to learn more, I regularly practice household austerity measures. I often take $80 into the grocery store on the day where the weekly deals overlap (so food is cheaper) and buy enough food to feed us for a week. This process still gives me confidence we can weather our worst-case financial scenario.

It is imperative that we view our monthly burn rate as non-negotiable. Unless we need or want to change that baseline, it is exactly the amount we need to live every single month. It is the upper limit of what you allow yourself to spend. PERIOD.

Having a rule in place that you designed and fully understand brings a sense of comfort and stability to

Playtime is a shared activity.

a very uncertain lifestyle. One of the most concrete examples is removing the uncertainty of deciding what freelance contracts are "worth it." If you want to make enough on a project to support you for one month and your monthly burn rate is $2,000 and you have an effective tax rate of 20 percent, then you must make $2,400 on the project, and it must take you less than one month. You now know that you would want to start negotiations above $2,400 and you won't settle for less than that.

If you want a project to bring in six months of living expenses, you can calculate your monthly burn rate, add your effective tax rate, and voilà! You have your absolute minimum amount of pay you are willing to accept on a contract project. This will get more complicated when you are also looking at funding future months and projects with contract work, but the principle is the same. How many months do you want to fund? What are your active burn rates?

If you want to go indie, especially if you have a family to support, do not improvise your financial well-being. Get in the habit of tracking your spending and knowing exactly how much you need to support the lifestyle you want to have.

Along with our monthly burn rate, we plan for the worst. When Adam and I talk about this, we say things like, "If we need to get a 'real' job, how long will that take us?" This depends on the job market and the city we are in. For me, that time is usually around four to six months. Adam, who has a wider base of marketable skills, would take much less time to find a normal software development job. We regularly re-evaluate our worst-case scenarios. We talk about whom we would call, we keep our professional résumés up-to-date, and we keep in contact with friends outside of the game industry.

In the event of a catastrophic business failure, we have established what we call the War Chest. This is a savings account we built to contain exactly our burn rate for a specified number of months. Oftentimes, this will be between three and six months of savings. We do not dip into the War Chest for frivolous things. This is for living expenses, and funds from it are used only when needed and always replenished as soon as possible. This War Chest was much smaller before we had children. Now it includes costs for monthly health care premiums, a maxed-out health savings account (HSA), a larger grocery bill, and more. We planned and built it slowly by modifying our monthly burn rate.

This is all well and good, you think, but Finji is an established indie development studio! You guys have been around forever! How does this apply to someone just starting out? It is so important to understand that this *is* how we started out. Adam worked full-time as a software developer for two full years before his freelance clients provided enough for him to quit his well-paying job. He took a $35,000+ pay cut when he quit. I supported him for another two years while working full-time with health benefits before I was able to leave my position. It took us over four years to ramp up production, and this was still before the release of *Canabalt*, one of our best-known games. We built up a War Chest, knew exactly how much we needed each month to survive, paid off all debt except our home mortgage, and worked with those constraints in mind. We planned for this life for years.

Even with our internal development on games like *Overland* and our Partner Publishing projects, we still seek out contracting jobs that pay well and are interesting. We use these funds to bolster the bottom line both personally and professionally. *Overland* has been 100 percent funded by two outside contracts. The last two financial years for us personally were funded by two separate freelancing jobs. Our revenue from our existing games does not always support our monthly burn rate, and we work hard to make sure we always have back-up plans in place to make sure our family is financially OK.

There is nothing wrong with contracting, and there is nothing wrong with working a day job and developing your dream project at night. What is important is that you can create. In order to do that, you have to be able to eat.

How we live

As you have noticed, I am a planner. I always have 10 back-up plans ready to put into motion if something fails. This includes all aspects of our life, from the way we raise our two boys, to the house we live in, and the car we drive.

We try to keep our home life simple. We live in the suburbs because the house was affordable and we needed a studio space and a backyard. We drive a paid-off Honda Fit that we bought in 2008. We try to max-out our HSA with scheduled monthly transfers so that in the event of a medical emergency, we have our insanely high deductible cost available. Both Adam and I cook a lot, and we rarely eat out. We have made our home a place where we live, work, and play. We strive to make our rather boring surburban home a magical place for adventure and imagination for our boys.

There are a million ways to set up your home as a base of operations for indie development. We have worked out of our bedroom, from a desk in the living room, from a small studio that shared a space with the guest bedroom, to the extra bedroom-turned-studio in our current house. It is not important where the studio space is, rather that the studio space is intentional.

What does that even mean? How do you make an intentional working space? First, I loathe working at a desk, and I have a constant desire to sit in a bright room looking outside. I rarely work in our studio space at home because it has one window looking out on a cedar privacy fence on the northwest side of the house. My desk faces a wall. In a pinch, if I really need to escape the chaos of children and pugs, it works. But my best work is done sitting on the couch by the front windows, or at the café table in the afternoon sunlight, or at the patio table in the backyard. Adam needs a studio space with a desk, a window, his books, his music, and a door to keep out our mammalian riffraff. I need open space and airflow and things to look at when my mind wanders.

We have the luxury of a large house, but you can figure out your own ideal workspace based on your available resources. Where do you work now? What environment will best allow you to focus and create? Map out this place in your mind and strive to build it for yourself. If you need a door, put your workspace in a place where you can keep the other people in your house at bay. Working with kids in the house is intense, and there are some days I need isolation to hear my thoughts clearly. But often, there is work that can be done in the midst of the chaos kids create, and as a parent you have to be able to work in both environments.

How we schedule

At Finji, work time is not divided equally. Specifically, if we take the total number of hours that Adam and I can work in a week, we do not allocate half to Adam and half to me. Our company responsibilities are

very different. I do not code or manage the internal development team. I do not usually need to participate in the daily nitty-gritty design discussions. Adam is better at this than I am, and this is the work that makes him happy. I manage things. I work with my computer. I send emails. I measure numbers, read contracts, and map out development schedules, releases, marketing, and festivals. I participate in design outside of Adam's main work hours and during mine. Because our work hour needs are so different, Adam takes the majority of hours during the week to create our games. I fit my work time in when we can.

This means that I schedule my work time around our children's schedule. I marathon emails and phone calls during nap times and spend nearly every night from 9-11 p.m. making sure everything else is done.

This schedule works for us right now. It will change again in a few months because our development schedule will change, our children will be on different preschool schedules, and our personal needs will warp into something new. This is normal. Set yourself up to work in an evolving environment. Managing the needs of your partner (if you have one), your children, and your work will always and constantly change. It is in your best interest to be communicative with your family and create a space that works for everyone.

There are a lot of things I do to keep up with work since I do not get superfocused dedicated work time. I have a lot of things going on at once, and it is very easy to let things slip through the cracks.

First, I use a calendar system that works for me. This has been one of the more frustrating parts of working as an indie. I have tried everything. I have tried applications on my laptop that sync with my phone. I have tried every calendar on my phone, computer, and email client. None of these systems stuck. I hated using them. What works for me is a physical calendar and datebook! It took me over two years to figure this out.

My datebook contains my to-do lists, the kids' schedules, travel dates, and so on. I highlight and color and scribble—I use all of the space and even take notes on it. I mix business and personal. It is all in my calendar, and it works for me. Adam uses the Remember the Milk app and text files. The important thing is not what scheduling tool you use, it is that you're using one. Find a system that works for you and start writing things down.

Once you know what you have to do, getting your tasks done in the allotted time is so much easier, which then makes it easier to figure out when and how to do something else. There are so many great consequences that come from planning like this. Adam finds that solutions to game design or buggy code will come to him during these off-times. I find that some of my best ideas about marketing,

specifics of console relationship needs, or future Finji business planning are clearer when I am not trying to brute force the ideas or the work.

Don't waste time pounding away on something you don't understand when you can get something else done. You could spend three hours not fixing that buggy code, or you can step away, get some fresh air, and fix it in 10 minutes the next morning after you get some sleep. Prioritize your tasks, have a game plan for work time, and don't get hung up on something you don't understand. Be flexible.

Second, we use a system called zero-inboxing to remove to-dos from taking up space in our brain and our email. In order to zero-inbox, you archive and remove emails from your inbox while making notes in your schedule of when you need to take care of those emails. It seems circuitous—why in the world wouldn't you just take care of the emails right then? But think about what you do now. You read an email and you leave it in the inbox until you get around to answering it. Those to-dos in your inbox pile up until you are looking at hours of just email correspondence or project tasks, and that yields unnecessary stress. Read the email, make a note of what is needed in your calendar, and then get that out of your inbox. I promise, you will thank yourself for doing this.

Finally, I am going to talk a little bit about travel. I do not travel much for game industry events. Since 2011, I have attended GDC in San Francisco twice. Adam has been to Australia, China, Iceland, Argentina, France, the UK, Canada, and all over the United States. And I am not gonna lie—I am a bit jealous that he has seen so much of the world while I have been at home with our boys. And, in contrast, he is often quite envious of my ability to stay home.

Over the years we have come up with rules for what kind of travel we will do for business and for fun. For business, we do travel sprints when we have something

important to say. In fall of 2014, Adam traveled to two countries and five North American cities. He spoke at universities, game jams, festivals, and conferences. This was the first conference season since we reformed and announced Finji, and there was an interest and a need to talk about the company, our internal IP development, and our Partner Publishing initiative.

The tradeoff for going to these various places around the world—the effect it had on our family—was worth it at the time. In contrast, in 2013, the year our second son was born, Adam traveled very little. In both of these situations, we made the right decision for our family, and our rules for travel adapted as our life situation evolved. There is always a tradeoff for travel when you don't have children, but the consequences of travel are compounded when there are kids and a stressed-out at-home partner in the picture. You will not travel as much when you have children, so when you do, make it count.

Our "travel for fun" trips are reserved for family vacations to see our far-flung siblings and parents. Right now, we cannot travel to places where we are required to be completely off the grid, and we never leave the gear behind. We make sure that we have available office space, even if it is in a coffee shop, wherever we travel.

When you run your own studio, there is no one else to make up for the time you take off. You are it. So when you plan on having a vacation, and your development is continuous, then you should plan around the logistics of how to work while traveling. We have taken beach vacations, gone to week-long family reunions, and taken short weekends away with our kids. We can always find time to do the necessary work on the road.

Why in the world do we do this?

There are a lot of tradeoffs to this life, and being a parent and an indie developer at the same time is a new thing. There's no older generation to use as an example of what to do and what not to do.

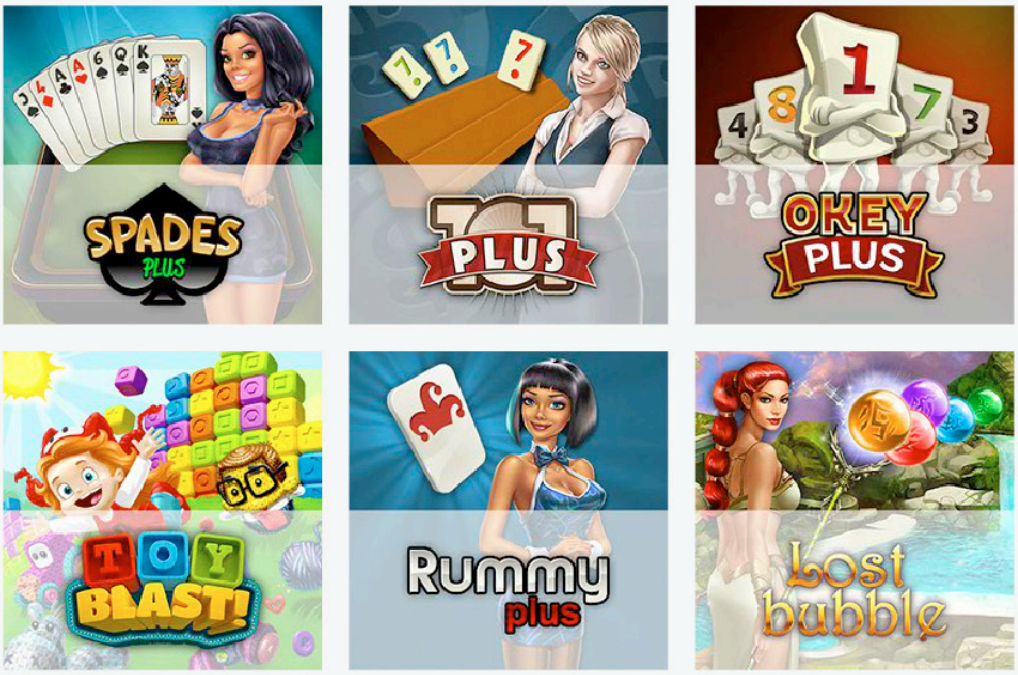
We take our available time and ration it out across our work tasks and family responsibilities, and sometimes that means things like making a new game or preparing a contract will take longer. Emergencies happen, and tasks will get delayed. There are things we want to do—we want to learn to play the piano, to write novels, to watch movies, to build a game that won't appeal to a larger audience but will fulfill us emotionally. Our tradeoff doesn't mean that those things are unimportant, or that we remove those things from our life. Instead, we re-evaluate our time and interests regularly, because everything happens in a season. And sometimes the season isn't right. And that is OK.

Right now we are parents of young children. Our time is at a premium, and we have arranged our life to accommodate that absolute reality. Someday, we will both travel to industry events. Someday, we will pick up our hobbies again—like playing instruments and writing. Someday, we will go out with our friends more often. Our goal is to survive this time. We want to be both amazing parents and successful indies.

We stay in this industry because games are in our blood. Our co-workers and competitors are creative, smart, and beautiful people. They challenge us with the things they make, influencing the games we make and the way we think. We can't think of a better way to raise our children. Or a better way to live our dream as a family.

Rebekah Saltsman is the CEO of Finji (among other roles), and mother two two boys with her husband Adam Saltsman. Finji is currently working on Overland, among many publishing projects.

LET YOUR CAREER REACH THE PEAK!



Come join us to shape and drive the future of technology and work with a talented and motivated team

JOIN US

FREE DEVELOPMENT TOLLS 2015

GAME DEV DOESN'T HAVE TO COST A MINT!

Shane Marks

Game tools can definitely help you get your games done faster, but a lot of the best ones cost a lot of money. Still, games don't have to cost millions of dollars to make! If you're willing to put in a little extra time to learn a new interface and workflow, free tools can be just as useful as the paid ones. That's why we've collected some of our favorites, to help you make informed choices when working with free tools, updated for 2015. Here you'll find some suggestions, ideas and possibly inspiration. (Note that while these tools are free or offer a free versions, they all vary in licensing and revenueshare, so please make sure they're compatible with your project.)

Development Tools/engines

This section is devoted to tools for people who want to implement game systems via code or editors.

MonoGame

monogame.net

MonoGame is a C# framework that allows developers to target desktop, handhelds, mobile, and consoles using a single codebase. The framework is written on top of Mono, an open source implementation of the .NET framework. The framework was originally designed to implement Microsoft's XNA 4.0 API, but Microsoft's decision to abandon XNA has meant that the MonoGame project has started to grow beyond the stagnated XNA implementation.

Developers can choose to use MonoDevelop or Visual Studio as their IDE (integrated development environment), though to build content to the xnb format you will also need install Windows Phone SDK 7.1.1 (installing XNA 4.0 doesn't work anymore), which means you will need a Windows machine to build the content. Once you've built it though, you can continue using whatever OS you were working in. You can read more about the specifics of these issues on Gamasutra or on the MonoGame wiki.

Although there are some issues present, the MonoGame Team is working on solutions. To date the framework has been used in multiple successful titles such as Bastion, FEZ, Mercenary Kings, TowerFall Ascension, and Transistor.

You should also be aware that Ethan Lee, the developer behind the ports of games such as Escape Goat 2, FEZ, Rogue Legacy, and TowerFall Ascension, has been working on a fork of the project know as FNA, which aims to reimplement MonoGame using SDL2 (see below), and can be found here: https://github.com/flibitijibibo/MonoGame

Simple DirectMedia Layer (SDL)

libsdl.org

SDL is a cross-platform library written in C, designed to act as a wrapper around operating system specific functionality. Some of this functionality includes audio, input devices, graphics hardware, file access, timing, and threading. SDL allows developers to write more generalized code, rather than worrying as much about the underlining specifics of the system they're targeting.

Beyond this, any extra functionality can been added through libraries that sit on top of SDL. Some of these libraries include support for multiple image formats, audio mixing, networking, TrueType font, Rich Text Format support and many more, via third party libraries.

SDL provides bindings for several languages such

as C#, Lua, Python, Lisp, Haskell, and others. It's also commonly used as the base code for other game libraries such as LÖVE, PyGame, ScummVM, and even the CryEngine, as well emulators such as MAME and ZSNES.

SDL is probably the most commonly used games library I know of - you'll see it in games such as Amnesia: The Dark Descent, FTL, Portal, Psychonauts, Team Fortress 2, The Cave, The Walking Dead and many, many more.

Unity

unity3d.com

Unity is a game engine developed by Unity Technologies that allows developers to target desktop, web, mobile, handhelds, and consoles using a single codebase. It comes with a fully-fledged game editor as well as the MonoDevelop IDE, which provides code auto-completion and debugging.

The engine provides scripting through Mono, and allows developers to write games in C#, a JavaScript-like language that most people call UnityScript, or a Python-like language called Boo. While some new developers to Unity start off using UnityScript or Boo due to its perceived lower barrier of entry the majority of developers use or eventually switch to C#.

Unity comes with full support for 3D and 2D games, and generally anything that you feel is lacking can usually be found through their online Asset Store or somewhere else online. Games like Kentucky Route Zero, Night in the Woods, Superhot, Shadowrun Returns, and Gone Home have all been made in Unity, showing that as an engine it's versatile enough to be adapted to any number of genres.

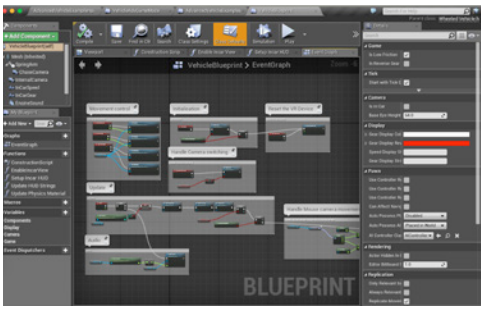
While the free version should allow a lot of developers to write their entire game without the need for Pro features, they should be aware that there are restrictions, which you can read about here: unity3d.com/unity/licenses. A developer can use the free version provided their annual gross revenues does not exceed US$100,000, otherwise they will have to purchase a license for Unity Pro, which will cost $1,500 or $75/month.

Check out unitypatterns.com for some useful tips and tutorials, as well as the tutorial in this issue.

Unreal Engine

unrealengine.com

Unreal Engine is probably the most well know game engine in the industry. Unreal 3 powered the majority of the Triple-A titles on the PlayStation 3 and Xbox 360—with the release of Unreal 4, Epic Games looks set to keep that going. Titles such as the upcoming Street Fighter V, Tekken 7, and Kingdom Hearts III are all being powered by the engine.



But Epic Games, developer of the Unreal Engine, has been trying to reach out to the indie community as well. The company recently announced that the engine has become totally royalty-based. This change has now made the engine a viable option for people looking to keep development costs down, with royalties set at 5% after the first $3,000 of revenue per product per quarter.

Unreal comes with a powerful editor that you can use to configure every aspect of your game. Developers can write their games using C++, or use a visual scripting language called Blueprint. While visual scripting might sound restrictive, and counter to what most other frameworks or engines do, it actually can be quiet expressive. I've found on several occasions that it was quicker to implement or test a feature in Blueprint, then to jump into C++ afterward. My only criticism of Blueprints would be if you're not very conscious of space and cleanup, then your Blueprints can quickly end up looking like a mess of spaghetti, and when you come back later you might think "What the heck was I doing here!?"

One of the biggest advantages Unreal has over other big engines is that their full source code is available. That means you don't have to wait weeks or more for bugs to be fixed or even spotted, because you can just do it yourself!

Unreal currently provides support for Mac, Linux, Windows, PS4, Xbox One, iOS, Android (limited support, currently), and WebGL. The engine comes with so many features that I couldn't possibly go over them here, but if you're interested in seeing where the engine shines, check out it's Rendering, Materials Editor, Particle Editor, Networking, and Level Streaming.

Top Alternatives Libraries/Frameworks

If none of the above suggestions appeal to you then maybe check out one of the following.

Cocos2d

cocos2d.org

Cocos2d is a 2D framework designed for mobile and desktop, but the majority of games made with it are for mobile. It's divided into multiple branches, the most popular being cocos2d-x, cocos2d-swift, and cocos2d-html5. Each branch is written in a different language, though they do try to keep some consistency. Cocos2d also comes with an editor called CocosBuilder, which can be used for designing UI, and setting up scenes and events without code.

LÖVE

love2d.org

LÖVE is an *awesome* framework written in Lua design for making 2D games. The nice thing about LÖVE is that it's well documented, there are loads of tutorials, there's an active community, and a lot of cool libraries built on top of LÖVE to help you add features you feel are missing. The fact that LÖVE is written in Lua is also a big plus to me since it's an extremely popular scripting language in the game industry. So even if you later decide LÖVE isn't for you, you can still apply that Lua knowledge somewhere else.

OpenFL

openfl.org

OpenFL is a Flash API-compatible library designed for desktop, mobile and web. OpenFL is built on top of a library called Haxe, so if feel like you want lower level access then check that out. One of the cool things about OpenFL is that because it's compatible with Flash you can also use popular Flash libraries such as Flixel and FlashPunk if you feel more comfortable using those libraries. OpenFL has been used in such titles as rymdkapsel and the IGF winner Papers, Please.

Phaser

phaser.io

Phaser is a HTML5 framework designed for desktop and mobile, built on top of Pixi.js. Although Phaser is relatively new, it supports a lot of features such preloading, animations, tilemaps, sounds, physics and WebGL rendering, that are not commonly found in most competing JavaScript frameworks.

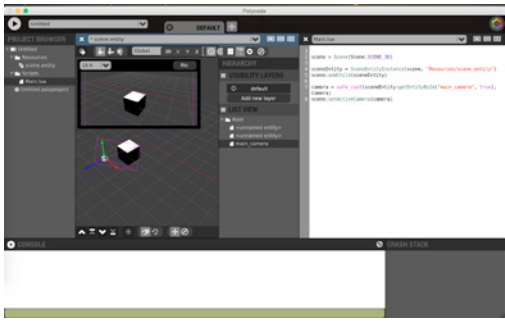
SFML

sfml-dev.org

SFML is a C++ library similar to SDL, but with additional features such as networking within the core library. Like SDL, SFML comes with additional third party libraries, and also supports bindings to multiple other programming languages.

Polycode

polycode.org



Polycode is a free and open source framework created by Ivan Safrin. While it's been around for a long time, it only recently had it's first official release. It's written in C++ and supports scripting via Lua. It currently exports to Mac, Linux, and Windows, with Android and iOS support in the works.

The framework comes with an IDE which you can use to edit your scene, modify your game assets, or write Lua code.* The degree to which you utilize this IDE depends on your configuration of the framework.

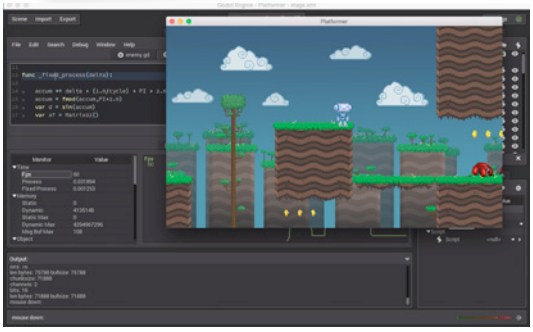
If you're interested in a framework which gives you similar functionality to something like Unity, but also has the benefit of access to the source code, then Polycode might be for you.

Godot Engine

godotengine.org

Godot is a free and open source game engine created by Juan Linietsky and Ariel Manzur. The engine comes with an editor and a custom Python-like scripting language called GDScript. The scripting language was built from the ground up to work efficiently with the engine. Developers can also use C++ for performance critical features.

The Godot team has recently rewritten the entire 2D part of their engine, so defintely check out their demos to see the latest features. It currently exports to Mac, Linux, Windows, Android, iOS, PS3, and PS Vita (with license from Sony), and support is in development for browsers and Windows Phone.



Art/Graphics

Here are some free art tools that can be used to create various types of visual content for games.

Blender

blender.org

Blender is a cross-platform open source 3D graphics suite developed by the non-profit Blender Foundation in collaboration with hundreds of others around the world.

It supports modeling, animation, rigging, texture mapping, UV wrapping, compositing, morph targets, sculpting, simulations such as clothes and hair physics, and several other features. Any features you feel are missing can be added using Blender's Python API, and generally any custom features that become popular enough in the community end up being part of future Blender releases. The most recent well know addition has been Cycles, adding a new powerful rendering engine to Blender.

There are plenty of books and online tutorials about Blender, and several new books have been released that show off some of the newer features available to users. You can check out projects made using Blender in their showcase reel on their website.

GIMP

gimp.org

GIMP is an open source raster graphics editor originally started by Spencer Kimball and Peter Mattis in 1995. Today it is developed by The GIMP Development Team, and has gone to become one of the most popular graphic editors in the world.

GIMP supports various paint tools common to most raster graphics editors such as Photoshop. It also has support for animation, layers and channels, path tools, quick masks, exports to many formats and can be enhanced via scripts and plugins. GIMP also supports Photoshop brushes, so you don't have to worry about looking for GIMP-specific brushes.

One of the most interesting things about GIMP is that because it's open source, it can be repackaged by its users with different default settings and plugins, which allows it to be configured by for a more specific kind of task. This had led to the popularisation of configurations like GIMP Paint Studio.

Inkscape

inkscape.org

Inkscape is cross-platform open source vector graphics tool similar to Adobe Illustrator. It's designed to be used in creating icons, logos, diagrams, maps, web graphics, and much more. It does this through SVGs, implementing an open standard set by the W3C.

Inkscape supports advanced drawing tools, object manipulation, styling, text manipulation, as well as many other features. It can also export to several formats such as SVGZ, AI, PDF and PNYesG.

Probably one of the most useful things I've done with Inkscape is generate game terrain. This is extremely useful when creating terrain similar to games like World of Goo, or Worms.

Top Alternatives for Art/Graphics

Graphics Scale *humanbalance.net/gale/us/index.html*

This is a popular tool designed specifically for pixel art. It supports all the features commonly found in pixel art tools, and while there is a pro version, chances are you won't need that since the free version supports all the features commonly needed.

Hexels

hexraystudios.com/hexels

Hexels is a grid-based art program that lets users paint with shapes. The specialised nature of this tool means that artists may find it easier to distinguish their game. If you're interested in seeing the kinds of things possible with Hexels, check out madeinhexels.tumblr.com

Piskel

piskelapp.com

This is a pretty new open source web-based pixel art

tool. It supports layers, frames, previews, imports, and all the other basic functionality you'd expect from a pixel art program. One of the coolest things about Piskel is the fact that because it's web-based, it allows its users to create accounts and save their progress online. The most obvious advantage of this is being able to pick up where you left off without carrying the files around with you, or worrying about the machine you're currently using having the software installed.

Spriter
brashmonkey.com
Spriter is a relatively new animation tool which was created though Kickstarter backing. It allows you to set up advanced animation sequences using an editor, without the need to worry about which development environment you are using. It also allows an artist to create, preview and tweak animations without the need of a programmer, which can remove a lot of the overhead in getting a game looking right. Spriter was used on Gunhouse, made by this magazine's editor (Brandon Sheffield) and his team.

Wings3D
wings3d.com
This is a 3D modeling software program similar to Blender. While Blender supports more features then Wings3D, people often prefer Wing3D because of its simpler UI and refined feature set.

Audio

While there are plenty of free tools for development and art, for whatever reason I've always found it hardest to find good free audio tools. Here are some tools that I think are pretty useful for editing audio or creating music and sound effect.

Audacity
audacity.sourceforge.net
Audacity is a cross-platform open source audio editor and recording tool created by Dominic Mazzoni and Roger Dannenberg in 1999. Several years later, it's still being developed, with new improvements coming every release.

You can use Audacity to record live audio, record computer playback, perform noise removal, add audio effects, convert, edit or mix multiple types of audio formats, as well as export to multiple formats. Audacity also supports plugins and allows custom effects to be made using a programming language called Nyguist.

Some of the most basic uses of this software would be to remove lag in podcasts where the speakers are in different locations. This would be done by every participant recording their own audio stream using Audacity, and afterward stitching every audio stream together in the editor while accounting for the lag that happened while recording.

Another simple use for this software would be creating sound effects, or splicing up an audio track in case you needed to remove parts of it to make it better loop in your game.

Bfxr
bfxr.net
Bfxr is a browser-based synth program used for generating sound effects. Bfxr is based on Sfxr, a tool that was written during the 10th Ludum Dare, but has been extended over time with additional features.

Bfxr has added new waveforms and filters, expanded the pitch and jump parameters, as well added a mixer and visualizer. It also allows for users to lock specific parameters during randomization and mutation for better control.

Once you've created the sound effects you want, you can then download them as .wav files.

Top Alternative Audio Tools

Audio
audiotool.com
Audio is a web-based audio workstation which can be used to create music for your game. It also supports midi devices, and since it works within a browser you don't have to worry about installing any kind of software. The only downside you might find is that you have to publish your tracks to a public page. You still retain the copyright, but it does mean that the tracks will be seen by everyone.

LLMS
lmms.sourceforge.net
LLMS is an open source cross-platform audio workstation that allows you to produce music. It supports midi devices, as well as a plugin system to

extend the software.

Famitracker
famitracker.com
This is tracker specially designed for creating NES/ Famicom audio. This can be useful if you're interesting in creating a game that's trying to be a throwback to that era, or you're simply interested in seeing what you can produce with a more limited set of options.

MilkyTracker
milkytracker.org
Milky is a popular cross-platform tracker used to create sound effects and music tracks for games. Milky exports to .mod, which is supported by popular game engines such as Unity.

OpenMPT
openmpt.org
OpenMPT is a tracker for Windows which allows you to create music and effects for your game. OpenMPT also supports a plugin system which allows it to extend the functionality of the software.

Misc
Finally, I've decided to throw in a few pieces of software I use on a day-to-day basis, or have found to be useful at various times.

Document/Spreadsheets
Google Docs
docs.google.com
This comes in pretty handy if you're trying to put together documents for your project and want to collaborate with someone else.

Libre Office
libreoffice.org
Libre Office is useful if you'd rather store all your documents locally, or know you're going to be without a connection and need to work on those spreadsheets!

Source Control
Bitbucket
bitbucket.org
BitBucket provides Git and Mercurial source control, a wiki, and issue tracker. Free accounts are provided for up to five users, but after that you'll need to start paying.

Github
github.com
Github provides source control, a wiki, and issue track. Unfortunately, unlike BitBucket, all free accounts are publicly viewable unless you pay for a private account. However, a pretty neat feature in Github is Github pages. These allow users to create a static website for the projects they host there. A lot of users use this feature to host their own blogs. You could do the same thing for your game website and save yourself some money.

Project Management
There are many project management tools out there, but unfortunately most of them are bloated, or filled with junk I don't think is particularly useful. Here are some free ones I've found myself going back to.

Slack
slack.com
This software is used for team communication. It allows users to separate out conversations into different channels or topics to help keep track of conversations and issues. It also allows users to hook in different pieces of software, to provide different kinds of notifications to teams. To those who have used IRC it's essentially an IRC web front end with a bot.

Trello
trello.com
Trello is used for creating, organizing and tracking tasks. This comes in really handy when collaborating with other people.

Freedcamp
freedcamp.com
Freedcamp is designed for the managing of tasks. The core features allow you to assign tasks, and set up milestones and other significant events. It also provides the ability for a team to discuss and iterate on a task, as well as allowing them to set up timelines for tasks on a shared calendar. The free plan can be extended with paid add ons.

Shane Marks is a game developer from Ireland, who has worked on titles for console, desktop and mobile. He is currently working at Necrosoft Games on a variety of projects. Contact him on twitter at @d92008.