[PACKT] PUBLISHING

HOW DO I GET STARTED MAKING VIDEO GAMES?

John Horton

Everybody has at least one game in them. I believe this is the 21st century equivalent of "everybody has a book in them". Games are powerful; they can tell a story, entertain, persuade and bring joy or sadness. To their creator, video games offer satisfaction, educational advancement and even personal wealth.

What more powerful reasons do we need to get that game out from within us and onto the Google Play, Apple App store, Steam, XBOXLive Arcade or where ever we think our video game should be?

THE PROBLEM

The problem of course is that you want to make video games but you just don't know where, how or the best way to start.

This brief article was written for you if any of the following 3 questions are going round in your mind and you have so far not managed to find an answer:

1. **Which is the best language (C++, C#, Java, Python, Objective C, HTML5, etc.) to learn?**
2. **Which is the best platform (PC, Android, iOS, Mac, SteamOS etc.)?**
3. **Which Engine (UnrealEngine, Unity, GameMaker, Cocos, LibGDX, AndEngine, CryEngine, etc.)?**

The first thing to point out is that there is no "best" platform, engine or language and anyone who tells you there is, is either biased, blinkered or just plain wrong.

THE SOLUTION

The answer to all these questions can be much more easily found by talking about you and your game. It is desperately important to have this discussion with yourself because if you head off down the wrong path you could blow a serious amount of time before you realise you should have done things differently.

If you ever get that sinking feeling knowing you have just burnt an unrecoverable hour of your life on Facebook or Candy Crush, trust me, that is nothing compared to learning a programming language which appeared to offer so much but turns out it can never deliver what you want. Furthermore, using a scatter-gun approach and trying to learn a bit of everything will make progress very slow and possibly cause confusion.

TALKING ABOUT YOU AND YOUR GAME

To make sure you get it right first time, write down on a piece of paper or in a text editor, your answers to all the following questions. Wherever possible, elaborate a little so at the end of this short exercise you will have a few paragraphs that detail everything about you and your game. Make sure to do this before we move on to the part of the tutorial that will allow you to match your goals to languages, platforms and engines.

Q1: Where are you starting from?

Are you already a programming guru in one or more languages or are you a complete beginner with absolutely no programming experience at all? Perhaps you are somewhere in the middle. Write it down and then move on to the next question.

Q2: Where do you want to end up and when?

What do you see as a successful conclusion to your efforts at learning to make games? Do you want to be the lead programmer at Rockstar or Infinity Ward? Perhaps you have seen Indie Game the Movie and have a passionate drive to become an indie dev'. Maybe you just want to have fun? Perhaps you are just looking for the absolute easiest path to getting published or simply finishing your game for yourself? How much time are you prepared to put in to this? A weekend, a year, as long as it takes?

Q3: How do you like to learn?

Do you want to learn the absolute 'proper' way without any shortcuts or useful tricks? You want a fully comprehensive a-z learning pathway with zero shortcuts- no matter how much fun the shortcuts might be. Do you want the polar opposite of this and want to get to the games straightway or maybe your way is somewhere in between the two.

Q4: Do you have a preferred target platform?

You might not have an answer to this one; you might have several platforms in mind. It is even possible you absolutely must develop your game for every platform. Whatever the case, write it down before moving on.

Q5: Do you know what type of game you want to make?

There are so many different types of game and which one you want to make will certainly steer you towards different learning pathways, engines and languages. Write a sentence or two about the game you want to make. Be sure to mention the genre, perhaps, 3d, 2d, FPS, RPG, survival, retro arcade, multiplayer sandbox or mobile match-three. Obviously the preceding list is not exhaustive and might not have mentioned the type of game that you want to make.

Q6: The important question

Which of the above aspects about you and your game is the most important to you? Some choices are occasionally hard to reconcile together. Often some kind of compromise of goals is necessary. For example, how important is it that you make your game for your favourite platform/genre compared to how fast you want to see results, etc.

YOU AND YOUR GAME CONCLUSION

Hopefully the above questions will have left you with a statement about you and your game, perhaps something like the following:

"I did a little bit of programming at school but it is probably best to start again at the beginning. I have a strong desire to be a successful indie dev' and I am prepared to do whatever it takes to achieve this but I must be able to learn alongside my existing job which pays the bills. I want to learn everything thoroughly but I also want to be building games as fast as possible. I wouldn't mind making games for any or even all platforms but most of all I would like to make my game for desktop PC's and, one-day, get my new game green-lit on Steam. That would be a real buzz! I want to make a 2d game with retro graphics but it must feel new and exciting to play. I don't have all the details yet but I have loads of ideas. Maybe a platform stealth, rogue-like set in a dystopian world run by an evil dictator and the player has to make his way through the world taking on progressively tougher enemies and bosses before the final show-down with the dictator himself. The most important thing is to get it on Steam, anything else is a bonus."

MATCHING YOUR GOALS TO REALITY

By now you should have a personal statement which clearly details your game building goals. Take a look over the following table. Use the priorities from your statement to target the appropriate column and then study the row (which represents the attributes of a specific game building solution) to see if it also matches your other priorities or requirements. This will also start to make clear the language that would be most appropriate.

There will always be a match! But, there will always be compromises to be made. Study the table and then we can talk more.

TOOL	SUITS GENRES/ PERFECT USE CASE	MAIN LANGUAGE(S) USED	PLATFORMS DEPLOYED TO	DIFFICULTY ASSESSMENT	PRO	CON
UE4	3d* Want to build a AAA game and don't care how long it takes.	C++, Blueprint visual scripting	All Inc Consoles**	Very easy to begin building 3d worlds but piecing together full games eventually becomes complex. Simpler, visual scripting alternative to programming. Want to work for AAA consider starting here.	Totally free and fully featured to learn and develop. Arguably the best and fastest 3d engine. Amazing for designing 3d worlds. Loads of up-to-date beginner tutorials	5% Gross royalty payable on turnover for released titles.
CryEngine	3d*	C++, Lua	Desktop and Consoles**	Very easy to begin building 3d worlds but piecing together full games eventually becomes complex. Want to work for AAA consider starting here.	Arguably the best looking graphics of any game engine. Amazing for designing 3d worlds. No royalties!	Small monthly fee around $10 Less complete beginner tutorials compared to Unity and UE4.

TOOL	SUITS GENRES/ PERFECT USE CASE	MAIN LANGUAGE(S) USED	PLATFORMS DEPLOYED TO	DIFFICULTY ASSESSMENT	PRO	CON
Unity	3d*	C#, Javascript	All	Build your first, working 3d game in around an hour. 2d games are way more complicated than on 2d focussed solutions.	Massive, beginner-friendly community. Asset store beats all the competitors.	Although free there are lots of ways Unity can make you "need" to pay for an upgrade. Despite a recent upgrade, generally considered a less professional solution compared to UE4 and CryEngine Some of the easier to find beginner tutorials need updating
Android Studio	2d, 3d	Java, C++	Android	3d games are probably not for beginners Even 2d games are slightly tougher than using solutions like LibGDX.	Very easy publish to a vast market that is virtually free to enter ($25 lifetime). Perfect to show your betas to friends and family without jumping through hoops.	Expect to have to market hard to give your game a start.

TOOL	SUITS GENRES/ PERFECT USE CASE	MAIN LANGUAGE(S) USED	PLATFORMS DEPLOYED TO	DIFFICULTY ASSESSMENT	PRO	CON
XCode	2d, 3d	Objective C, Swift	iOS & Mac	3d games are probably not for beginners. Build games for all Apple devices using really quick to grasp Swift or the more challenging (but flexible) Objective C.	Perhaps the nicest development environment. Swift option is arguably the best beginner game programming language.	Significant chance of having submissions declined (especially amateur games). Awkward to show unpublished games to friends and family $99- $299 + tax annual fee to submit games. Must have a Mac to develop.
Cocos2d x	2d	C++, JavaScript, Lua	Mobile and Windows	Going to need to learn C++ first.	Totally free. Loads of successful games to point to.	Mobile focussed. Better off with SDL or SFML for desktop.
Game Maker	2d	GameMaker script	All	Very easy to get started with a range of video tutorials available. 3d is nearly impossible. Don't get this for 3d.	Although the language is proprietary it is very similar to other language basics like C, C++ and Java and a great way to get comfortable coding. Free for windows games.	2d games will be noticeably less smooth and playable compared to other 2d focussed solutions Expect to pay between $150 and $800 to get some fairly necessary 'advanced' features or to be able to support mobile and Mac.

TOOL	SUITS GENRES/ PERFECT USE CASE	MAIN LANGUAGE(S) USED	PLATFORMS DEPLOYED TO	DIFFICULTY ASSESSMENT	PRO	CON
SFML	2d, simple 3d	C++	Windows, Mac, Linux	C++ is arguably one of the harder languages to learn and get started but SFML can actually ease this. Need to learn C++ first.	Very fast. Modern OOP language. Possibly the fastest, smoothest 2d games for desktop can be made using SFML.	Smaller user base to SDL, less online tutorials available. 3d work is not for the beginner. Community can be slightly unforgiving to complete beginners.
SDL	2d, simple 3d	C	Windows, Mac, Linux, Mobile	Need a beginner C programming book first.	Very fast. Perfect for porting old code written in C to mobile.	Uses out-of-date language. Mobile implementation is not a good place to start for beginners.
LibGDX	2d, 3d	Java	All	Need to learn Java first, preferably in an Android context.	Totally free. Very beginner friendly support community.	3d is much less capable/flexible than Unity, Ue4, CryEngine. Arguably slower performance on iOS. Very significantly slower performance on desktop compared to SDL or SFML. 3d has very little documentation.

TOOL	SUITS GENRES/ PERFECT USE CASE	MAIN LANGUAGE(S) USED	PLATFORMS DEPLOYED TO	DIFFICULTY ASSESSMENT	PRO	CON
RPGMaker MV	2d	Ruby	All	Getting started is possible for any determined beginner.	Perfect for retro RPG's especially with turn based fighting. Free trial available. Loads of character and scenery packs available for a fee.	Won't do anything else without lots of effort. Even this latest version makes the games look a bit clunky on large HD screens. Full version $70 (watch out for Steam sales).

* Perfectly capable of 2d but with significant disadvantages to using a 2d focussed solution(avoidable complexity, increased program size.)

** Separate developer licence required from each console manufacturer.

Hopefully building a clear view of exactly what it is you want to achieve will make the table of options super-useful in identifying the best path for you.

START BY COPYING/CLONING SIMPLE GAMES

You know which game you want to build and you know which solution and language you are going to use. Exactly what should you do next? Immediately designing and implementing your target game could be very challenging if you are a beginner so a suggested progression path could be as follows.

2d Games and features progression

Start with the most simple implementation of each game and progressively add features like scoring, high scores, inventory, achievements, animated characters, parallax backgrounds, particle effects, multiplayer.

As you progress, consider the features your game will need and try to implement them in the simplest possible manner. When you get bored of the first game (perhaps Pong), provided you have achieved your goals, move onto the next (perhaps Arkanoid). Each of these features and games is a new learning experience and trying to do them all immediately could end in frustration, unless you are really dogged. And even if you are, dogged isn't fun.

Pong - Implement a game that has the simplest possible moving objects- a bat and a ball. Make the ball bounce off the bat, the ball and the walls. Lose a life when the ball hits the bottom of the screen, gain a point when the ball hits the top.

Breakout/Arkanoid - In this game you still have a bat and a ball but this time you also need a bunch of bricks that must disappear when they are touched by the ball and the ball must bounce off of a brick. This is a great game for adding lots of features and making it your own. Consider multiple balls, multi-color bricks, different strength bricks, increasing ball speed, decreasing bat size, extra life pick-ups etc.

Space Invaders - This is probably the most basic game you can implement where there is some element of AI (artificial intelligence.) Deciding when the invaders will shoot, move left and right or down is a challenge but one you will be easily ready for by this point.

Platformer with fixed view - A platformer introduces new elements like a more versatile player character, perhaps one with walking and jumping animations, maybe the player can have a weapon as well. Add some simple physics so that the player can fall when he is not on a platform and some awkward jumps that must be timed just right.

Shooter/platformer with scrolling world and intelligent enemies - Now it's time to use everything you have learnt so far and add something new as well. Scrolling is the first step to building rich, exciting game worlds worthy of the player's time to explore. Learning how to centre the scene based on the players location in a world that is bigger than the screen at this stage is achievable and extremely rewarding for the aspiring developer.

By this stage you will be able to consider almost any 2d game and begin to work out for yourself how to go about implementing it.

3D GAMES AND FEATURES PROGRESSION

As with the 2d games start with the simplest possible implementation and make sure you finish the game. Be sure to add things like score, lives and victory/defeat conditions.

Maze escape - Use your chosen 3d environment to build a straightforward set of walls that can act as a maze then add a controllable character who can walk around and try and find his way out. Perhaps add a time limit where the player will fail if the time expires. Carefully test how difficult or easy it is to succeed and then ask a friend or family member to try out your first game.

Walking simulator - Now it's time to get more creative with your environment builder. Try sculpting a modestly sized landscape with hilly areas, plains and areas with buildings. Try adding trees, street lights (that work), buildings that you can go inside and structures you can jump on and ascend. Perhaps make exploration the goal of your game. Scatter pick-ups around the world and provide a HUD showing how many the player has found and how many more are needed.

Fps with one enemy type - Time to add some enemies. Make a single enemy and make him home-in and fire at you. Once you have one dangerous enemy learn how your chosen game engine can help you clone him into dozens of enemies with very little extra work. Have a score, perhaps a kill-count and a victory condition as well.

Multiplayer fps - By the time you have learnt to do all these things you might be surprised how straight forward adding multiplayer support is. Engines like UnrealEngine and CryEngine provide seriously good support for adding what might at first seem desperately complicated.

By this stage you will be able to plan most game types for yourself.

TIME TO MAKE THAT GAME

Once you have built a few games with a few different features, stop! At this point, which could be between a month or a year later, you will realize that not only do you know how to make a game with all the features you want but you also have a good idea of how to structure your project. You will likely have the confidence to start on your masterpiece. You will probably even feel confident enough to outsource the parts of your game you don't want/aren't able to complete yourself, perhaps, graphics, sound, marketing or even parts of the programming.

It is now that you might get this funny feeling in your stomach. A feeling of anticipation, excitement and urgency. You will absolutely know- without any doubt that not only have you got a game inside of you but you can get it out too.

Author Bio

John Horton is a coding and gaming enthusiast based in the UK. He has a passion for writing apps, games, books and blog articles about coding, especially for beginners.

He is the founder of Game Code School, which is dedicated to helping complete beginners get started game coding using the language and platform which is best for them.

John sincerely believes that anyone can learn to code and that everybody has a game or an app inside of them; but they just need to do enough work to bring it out.

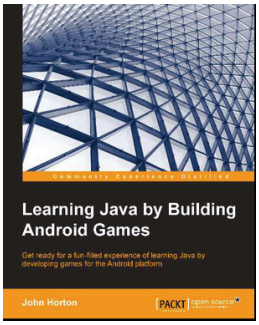
He has authored around a dozen technology books most recently the following:

Android Programming for Beginners:

https://www.packtpub.com/application-development/android-programming-beginners

Android Game programming by Example

https://www.packtpub.com/game-development/android-game-programming-example



Learning Java Building Android Games

https://www.packtpub.com/game-development/learning-java-building-android-games