Learning from another's game

Game development will be better, faster, and more successful when the examples of successful games are followed, showing that anyone can get into this high-demand field

DAKODA KOZIOL

Learning from another's game

Game development will be better and faster when examples of successful games are followed, showing that anyone can get into the high demand field.

Dakoda Koziol

Mrs. Voigt

Senior Project Research Paper

October 20, 2017

Most everyone, at one point, have played a videogame, from hardcore PC gamers to kids on mom's smartphone, from the arcade to the console. Even the elderly are playing blitz games on Facebook. The gaming industry manages to touch everyone. Games are fun. Obviously. One might say that they make the world go round. However, as great an experience it is, many would argue that *developing* the game is even more fulfilling than completing it. That's right, it isn't just big, money-hungry AAA game development companies that are making the games, there are hundreds if not thousands of independent game developers building unique games that they personally love. Games like *Spelunky*, *Cuphead*, *No Man's Sky*, *Rocket League*, and *Kerbal Space Program* are all made by indie developers. Even the huge hit *Minecraft* began its life as an indie game, though after its monumental success it can't be really be considered "indie" anymore. But how did indie game developers make a dent in this crowded market? They made a fun game of course! No one can resist the thrill of a good game, and luckily a new developer has a lot of examples to pull from. Game development can be better and faster when examples of successful games are followed, showing that anyone can get into the high demand field. Originality is fantastic, one should never blatantly copy after all, but usually somebody else has already come up with a good, tried, and true solution to a particular problem a developer is facing.

The controls are arguably the most important element of a player's experience, and are just as important to the general feel as graphics and story are. They take the player's input and translate it into commands for the game to obey. A well-designed control scheme can do this accurately, easily, and predictably. This is relatively easy to pull off on PC and console because of their reliable, tactile buttons. The touchscreen, however, is a different story, and can require quite a bit of ingenuity to get right. This is because a touch screen, unlike a gamepad or keyboard, is only a flat slab of glass. Without physical buttons to feel, a player's thumbs tend to stray from the controls, crippling the players surety in the controls. That’s no good! To cope with this, the best mobile games strive to have the simplest control scheme possible. Take the ill-fated, viral game *Flappy Bird* for example. This game only requires one finger to play. Furthermore, the player isn’t restrained to a little virtual button; that one finger can be tapped anywhere on the display to register. This makes it just about impossible for a clumsy thumb to miss, resulting in tight, agile control over a game.

Another great game with a great control scheme, if not as well known, is *Skiing Yeti Mountain*. This game also requires one thumb, but this time to guide a skier down a slalom course by rotating the skier's body, and hierarchly, his skis. Most developers would attempt to mimic physical controls — in this case the joystick — to control the character. In my experience, however, this isn’t always the most elegant solution. As Adam Smith, author for VentureBeat, puts it:

> “these movements do not feel smooth or natural because the environments that they are built on are more suited to a console game, where the player will have a two joysticks – one for the camera, and the other for movement. While the virtual joystick can solve some of these issues, it is a design mechanic that is seeking to recreate what is not instinctive to the hardware” (Adam Smith)

Skiing Yeti Mountain’s control scheme is proof of just this concept. Rather than settling for a joystick, the developers allow control of the skier by sliding his/her thumb across the bottom of the screen, giving the player easy and more precise maneuverability over the little skier. Although a joystick would've worked, it wouldn't feel quite as tight as the current control scheme is. In addition, this control scheme nails the general feel of skiing. As Anna Marsh of Lady Shotgun Games states: “Movements that you use are reminiscent of real life and familiar because of muscle memory". She continues with an example: "what you do with *Angry Birds* in the game resembles using a real catapult" (Leigh, Alexander). In *Skiing Yeti Mountain*, the back and forth motion the player makes when his/her thumb is dragged across the screen is very reminiscent of the motions skiers make as they tack and carve down a slope. Skiers are, in this case, the game's target audience. Anyone who's been skiing knows that to speed up a skier should be parallel to the direction of the slope, to stop a skier has to turn his/herself perpendicular to the slope, and to maintain speed a skier has to tack back and forth at appropriate intervals. These techniques apply to the game as well, making for predictable controls and a satisfying experience.

So far it would seem that mobile is a bit of a downgrade from its PC and console counterparts. Though to most this wouldn't necessarily be wrong, mobile does have it's perks, performing some tasks just as well or even better than a PC or console could. One particular strong suit of mobile devices is the abundance of sensors on board. With the exception of the iconic Wii Remote, most popular gamepads don't have built in accelerometers and gyroscopes. On the other hand practically *every* smart device has these sensors. Many popular games make use of them, especially racing games. *Traffic Racer*, *Asphalt 8: Airborne*, *Riptide GP2*, and probably a few hundred others all use the accelerometer to steer something, and do so with impressive precision. Secondly, the touchscreen, although often troublesome when precision is required, opens whole new dimensions of gameplay. The player can swipe, like in *Fruit Ninja* or *Prune*, or the player can use multiple fingers, like in *Cut the Rope* or *Clash of Clans*. Mobile may not excel when it comes to tactile input, but it is unmatched when it comes to its huge variety of input methods, many of them not found anywhere else.

But enough with controls, let's get to the gameplay. This is the backbone of the entire project, because without gameplay there is no game. Often a game can be successful by borrowing another games gameplay. Because of this, you can find entire genre's of games dedicated to specific types of games. There are loads of first-person shooters out there. Real-time strategy games will never die. Clickers are a dime a dozen. Tower defence? They're everywhere. Everybody's played an endless arcade game. Casual puzzles? Check. And platformers, classic. There are many more game genres out there, some more specific than others. Many avoid using these overdone game genres, and I don't blame them. But if profit is your goal, these are tried and true strategies that are easy to sell. There's a reason that the top charts on the Play Store and App Store are full of these types of games. However, there is often a unique one out there, a diamond in the rough begging to be someone else's inspiration. Case in point: *Agar.io*. This viral multiplayer may or may not have been the first game of it's kind, but it was certainly the most influential. Since it's release, similar multiplayer games have been released by the bucketload, including (but definitely not limited to) *Slither.io*, *Diep.io, Paper.io, Arrow.io*, and *Hexar.io*.

But what can a developer do when he/she doesn't want to be a greedy sell out, but still can't come up with something original? The fact is, the gaming industry is already very saturated; just about every idea a developer can come up with has been already been tried at some level. That's no excuse to totally copy and paste another games gameplay, but if only *specific* elements of gameplay are used as inspiration amazing things can come about. *BADLAND*, for example, is a side-scroller where the character flies through a course, navigating by gaining boosts in altitude with the tap of a finger. Many elements in the game have been done before, however, *BADLAND* differentiates itself not only with its tasteful and refined art direction, but also with multiplayer gameplay, diverse power ups scattered throughout the levels, and diverse and clever terrain. Single device multiplayer isn't new for mobile, though it wasn't nearly as popular before the release of *BADLAND* as it is now. In fact, *BADLAND* is likely the reason for the recent burst of single device multiplayer games. The concept of power-ups isn't unique at all, however the power-ups found in *BADLAND* are difficult to find elsewhere at all, let alone in one well-polished game. Most games implement cliché power-ups like speed boosts and score multipliers. But *BADLAND*? One type of power-up changes tap sensitivity, another slows down or speeds up time, some make the character spin, some change the characters size, and finally, one clones the character. Then there's terrain. Admittedly, there's nothing like it. This game is an intense, often frantic rush. Most side scrollers seem to be platformers, but not *BADLAND*. It's in a league of its own. (BADLAND)

Finally, graphics and audio are there to fuel the experience. If gameplay can't define a game, graphics and audio often will. It was brought up earlier that *BADLAND* is known for its art design, and for good reason. *BADLAND* creates a stark contrast between the background and foreground, which not not only improves the player's comprehension of the gameplay but also stands as a bold art direction. The background is made up of beautiful, parallaxing, hand-drawn artwork that not only gives the game hints of a story, but is also wonderfully immersive. The foreground is what the player(s) actually interacts with, and is made up of only silhouettes, making it easy for the player to tell what is or isn't an obstacle. The artwork, according to the official *BADLAND* website, was based on nature. To be specific, the art was based on the forests of Finland, where *Frogmind*'s studio is located. However, many reviews in *BADLAND*'s listing of the *Google Play Store* point out graphic similarities to games such as *LIMBO*, which is also known for its powerful atmosphere and silhouetted foreground. Because of these two games in particular, dark foregrounds have become fairly common in games, offering both potential for good graphics and better gameplay. As for the audio, *BADLAND* opts against a theme song or background music like the vast majority of games, and instead provides ambient sounds of the forest environment, along with all of its dangers. The graphics are already eye-catching, but the sound effects make the whole experience surreal and immersive. This is true of every game, but *BADLAND* really nails it. When the player hears the screaming of saws and the pounding of machine guns in slow motion and all other variations of space-time, the adrenaline rushes in and the player forgets that it's only game. For the record, *LIMBO* gives the player similar feelings of tension and fear. (BADLAND)(LIMBO)

In the bigger picture, and perhaps less poetically, there is the game engine. The game engine is not an element of a game, but rather the core itself. A game engine typically has the tasks of calculating physics, applying graphics, generating lighting, utilizing API's, and providing an interface for the developer to work with. Many AAA game studios (and even a few indie devs for that matter) opt to build their own game engine from scratch. This solution offers flexibility and optimization for a studios general game style, along with all the benefits of direct ownership over licensing. However, this solution takes a lot of time, work, and money. A *lot*. Even with a fairly large team, we're talking *months*, and that's just to get something that's vaguely usable. It really isn't an option for the average indie developer, but luckily there's another. Commercial game engines like Unity and Unreal are a huge boon to the world of indie game development. They essentially democratized game development. Before, if someone wanted to build a game they required a well-funded and well-staffed company, but with the rise of commercial game engines most of the hard work has been done. Many game engines offer a free version alongside their already affordable subscriptions, making it so that all a future developer needs is a half-decent computer and the will to learn to get started in game development. And to ice the cake, many game engines offer a marketplace for developers to sell assets such as sprites, models, scripts, tools, or anything to build a game, providing excellent opportunities to cut down development time. This paper isn't meant to read like an advert, but it's just about impossible to write about indie game development without mentioning the rise of the commercial game engine. (Unity)

Game development is challenging, almost as challenging as writing seven page papers, however it doesn't have two be. There are so many tools, helps, and examples to pull from that it simply can't be said that game development is for companies. It's for everyone. As stated before, game development can be better and faster when examples of successful games are followed, showing that anyone can get into the high demand field. If new developers take advantage of their resources, they can build games for everyone from the hardcore PC gamers to the kids on mom's smartphone, from the arcade monkeys to the console gamers, from the elderly to the developer himself/herself.

Works Cited

Alexander, Leigh. *Nailing the perfect feel for touchscreen controls*. *Gamasutra*, Gamasutra, July 10, 2013, www.gamasutra.com/view/news/195982/Nailing_the_perfect_feel_for_touchscreen_controls.php. Oct. 20 2017.

Android Apps on Google Play. *Google*, Google, Jan. 6 2012, play.google.com/store/apps. Oct. 20 2017.

BADLAND. Computer Software. *Google Play Store*. Vers. 3.2.0.29. *Frogmind*, April 4, 2013. play.google.com/store/apps/details?id=com.frogmind.badland. Oct 19, 2017.

BADLAND - The Game of the Year -Winning action adventure. *BADLAND - The Game of the Year -Winning action adventure*, Frogmind, badlandgame.com/.

Flappy Bird. Computer Software. *APKMirror*. Vers. 1.3. .*GEARS Studios*, May 24, 2013. https://www.apkmirror.com/apk/gears-studios/flappy-bird/flappy-bird-1-3-release/flappy-bird-1-3-android-apk-download. Oct 20, 2017.

LIMBO. Computer Software. *Google Play Store*. Vers. 1.16. *Playdead*, July 21, 2010. play.google.com/store/apps/details?id=com.playdead.limbo.full. Oct 20, 2017.

Skiing Yeti Mountain. Computer Software. *Google Play Store*. Vers. 1.2. *Featherweight*, 2015.

play.google.com/store/apps/details?id=com.featherweightgames.skiiing. Oct 20, 2017.

Smith, Adam. *Mobile gaming is a growing market, but what makes a good mobile game? VentureBeat*, Apr. 28, 2014, venturebeat.com/community/2014/04/28/mobile-gaming-is-a-growing-market-but-what-makes-a-good-mobile-game. Sept. 28, 2017.

Unity Game Engine.

Unity

unity3d.com. Oct. 20, 2017.