Outrealm Post-Mortem, Part 1

Blogs
by Nick Thandi on 02/06/18 09:42:00 am Featured Post

Introduction

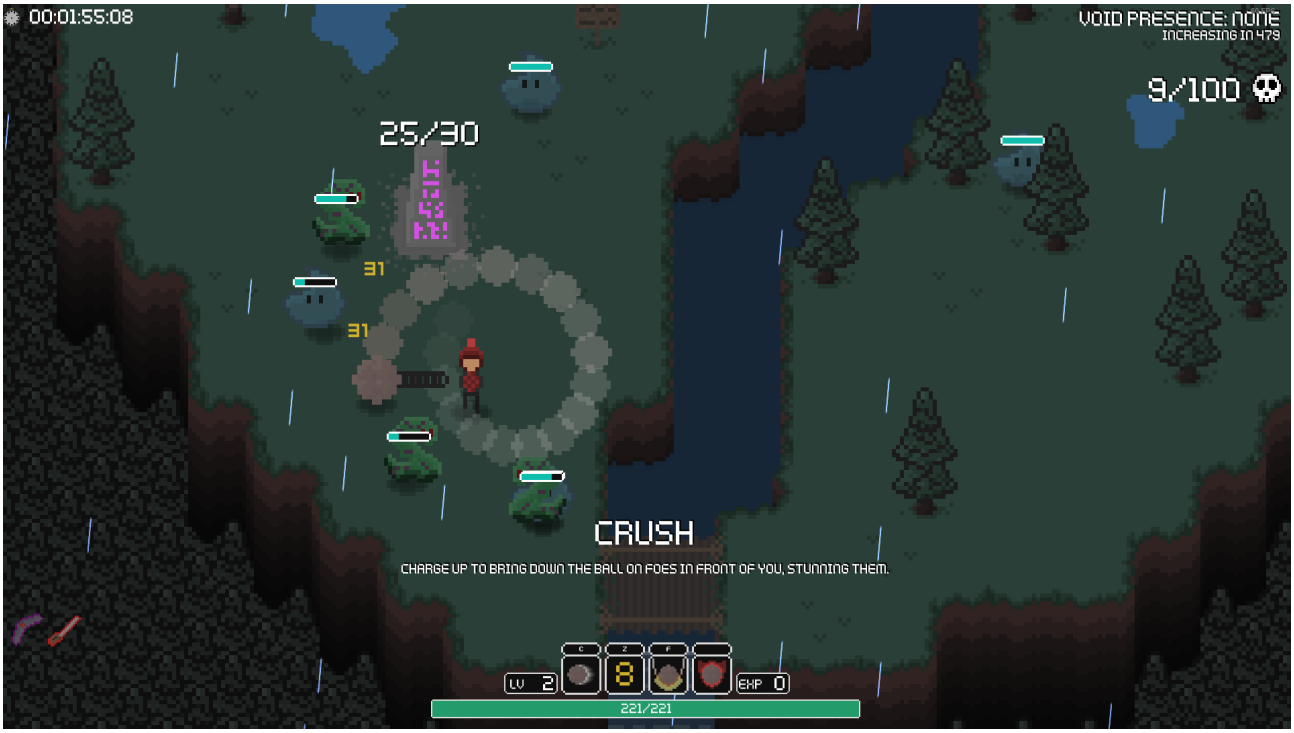
Hello there. I am the developer of a game called Outrealm, which released on Steam recently. If you want to check out the game, you can do so here. This was my very first commercial project: I have done freelance work for some games, but never have I actually gone about making my own product. Development time was around eleven months, with many extended breaks where I worked on other projects. However, I've been working consistently for the last five months, giving a total condensed time of around six months.

This Post-Mortem is divided into two parts. Part 1 focuses on myself, the realisations I had, and the growth that I experienced personally while working on Outrealm. Part 2 is focused on the product and the trials and tribulations of development. To me it only seemed natural to separate these concepts, because personal growth is a long journey worthy of its own examination.

I thought a lot about how I wanted to write this part, and settled on the idea that I wanted to focus directly on the mistakes I made, and the things I learned. Mistakes are really the most important thing. I made mistakes on this project that I couldn't have without making other mistakes in the past on other projects. Hopefully I can make a new set of mistakes on every project I work on!

If you're a newbie dev, I sincerely hope that you can learn something from my mistakes. Although that might be wishful thinking. I've always been the kind of person who has to experience failure firsthand before I learn the lesson. But even if I can get you to think about a different perspective or appreciate a certain way of thinking, that will make me very happy. I hope you can get something valuable from this!

And to those who decided to play Outrealm, I couldn't be more thankful. Whether you loved it or hated it, I feel honoured that you decided to take a chance on me. I hope you have fun playing! And please, leave a comment, rating or review that lets me know how you feel. Don't pull your punches. Every developer has to go through the process of recognising and fixing their mistakes. That's what writing these things are all about.



Lesson One: Solo development is inherently flawed.

Let's start with the biggest, most glaring mistake I made when developing Outrealm. Everything else pales in comparison to this one. Trying to do it all by myself. And I mean ALL. Everything from the coding, art, design, music and even some SFX (Although I did mostly use creative commons sounds!).

Before I get into why this is an issue, let me clarify that I certainly don't think it is impossible. Just, it's not worth it. Developing many solo projects in rapid succession is one of the fastest ways to improve your skills. And there are many examples of solo developers creating amazing commercial games. However, doing so will force you to make crucial concessions.

Let's say that traditional game development has three factors: Time, Cost and Quality. However it is impossible to develop a game while scoring well in all three; one is always sacrificed. Most AAA game development sacrifices cost or time, meaning budgets blow out and deadlines are missed. And if you've ever been a part of a buggy game release, you know what it's like to sacrifice quality. So what about a one man team? Quality is still a relevant factor, so it stays the same. Devs like me can make games for no financial cost, so cost really just means time. The last factor is sanity. Don't laugh: I'm serious. So if every type of development team has pitfalls, why is it so much worse for solo devs? Because every one of those pitfalls will doom your project in one way or another.

Sanity & Time: Sacrifice Quality: “It doesn’t have to be the greatest thing ever”. The flaw with this type of development is that ultimately, nothing matters in the end if your project is poor. Adding quality to a game takes a lot of time, meaning that if you choose to do things quickly, your project won’t represent your best work. And if you set out with the aim to achieve something only average, you will very likely achieve that.

Sanity & Quality: Sacrifice Time: “I’ll take the time to do things right”. It is possible to simply develop over a long period of time, however this is exceptionally risky. Technology and platforms move very quickly. Console generations may pass. More things can happen to you: Personal loss, accidents etc. Someone might produce a game of similar type and quality to yours, capturing your market niche. Most of the learning and growth that you experience will happen in the earlier stages of development. Focusing on one project for an extended time may slow down your growth as a developer. And if you do grow significantly, you will be constantly updating old work to your new standard.

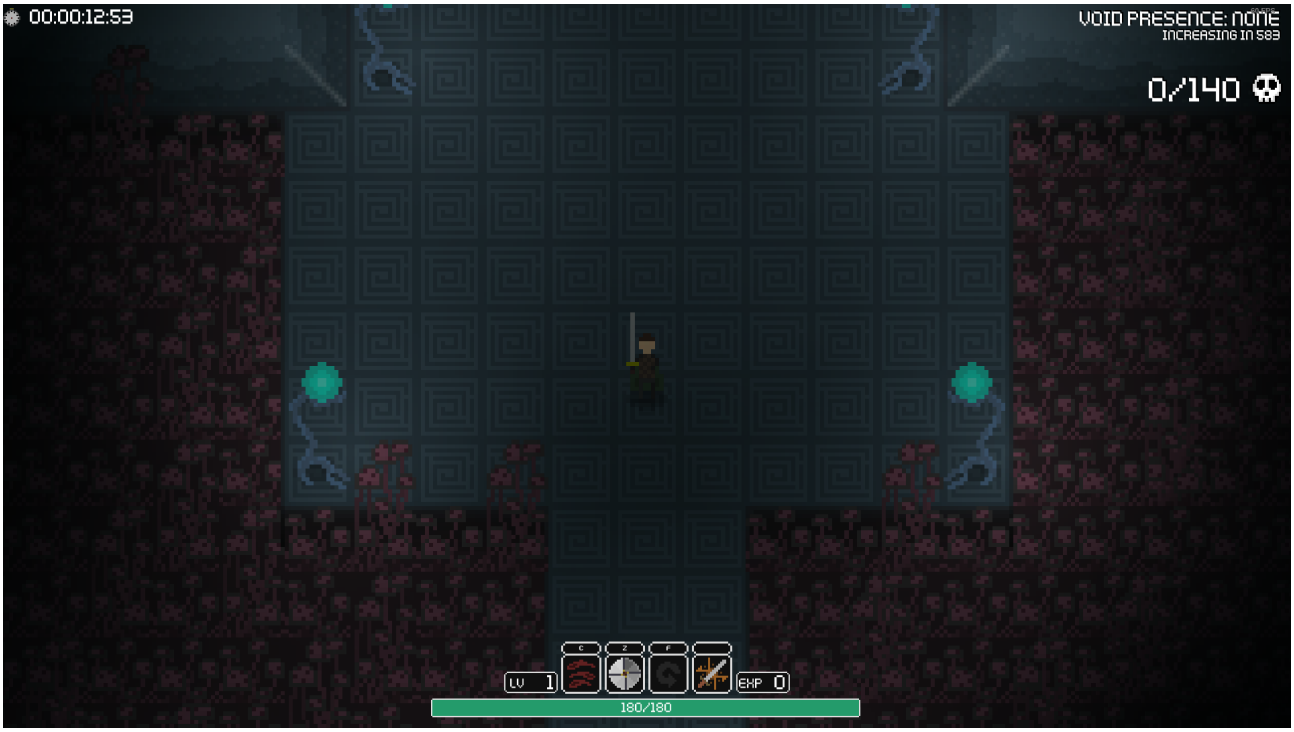
Time & Quality: Sacrifice Sanity: “I’ll just work harder”. This is when you decide to hermit in one room for sixteen hours a day and do nothing but development. If you do this, your quality of life will suffer. You will have to cover every aspect of your game in meticulous detail, and in a very short time frame. This is putting enormous pressure on yourself, especially if your time limit is based on your finances.

The only way to escape this fate is to work with other people. Imagine if it was two of you working on your project. You can cut down the time by a significant margin. Suddenly, you don’t have to handle every minute element of the project. You can work on the stuff you are strongest at, and so can your partner. You can more easily help each other with your design blindness by critiquing each other’s work. Sharing the burden means less stress, more time and better quality work.

There are other things, too. When developing Outrealm, I felt like I was spreading myself too thinly. I have always felt that broader work is more difficult than deeper work. I would rather do a large amount of one task, as opposed to smaller amounts of more varied tasks. I’m not sure if this is the same for everyone, so I’m chalking it down as a lesson purely about myself.

Unless you are just starting out, or doing this as a hobby only, I would advise against going it solo. If you’re just tinkering with a new engine, or making your very first ever game project. However, if you’re in any way serious about game development or the product you’re making, then I would highly advise you to work with others. Those others could be anyone. A friend of yours. A publisher. But getting help will lessen the load by a huge margin.

So in summary: If you are a newbie dev looking to get your feet wet, by all means start developing solo. But if you are looking to develop and sell a commercial product, then understand what that means, and the concessions you’ll have to make by going it alone. Don’t think you can just start coding and it’ll all end up rosy, it likely won’t!



Lesson Two: Your game is a product, and should be treated as such.

This one might sound obvious to those experienced with commercial game development, but it's my first one, alright? Games have always been a passion of mine, and developing them is no different. However, there is a point where the rubber meets the road: At the end of the day, you must think of your game as a product. You aren't just developing a piece of software: you're creating a product that goes on a shelf, and must compete with others for market share and success.

This is not as easy as it seems. Spending six months nestled in with a project gives you a very different perspective than outsiders with no attachment. This is similar to "Design Blindness" (being unable to see the flaws in the work you create), but not quite the same. It takes a cold, cutting and critical eye to ask tough questions about your own game. Why would anyone buy my game? What stands out about it? Is this an attractive product? It is a valuable purchase?

Lastly, thinking about your game as a product will get you thinking about things such as cost, placement and marketing material. These are often inconvenient thoughts that get pushed back to later. Speaking for myself, I was so caught up in the actual development that I never considered these things until much later.

The easiest way to get this opinion is to ask someone else. Would you buy my game? Would you recommend it? Why? Why not? Start marketing early: as early as you have something worth showing off. Reddit, Imgur and Discord are all places to make yourself known. By going public early, you also create accountability for yourself. Let the expectations of others drive you forward.

Now, full disclosure. I did basically none of the things in the above paragraph. Learn from my mistakes! (And get advice from people much better at marketing than me).

Lesson Three: You'll have to compromise with yourself eventually.

This is probably the most positive lesson I learned. When you see someone every day, you don't notice how much they change. But if you don't see them for a year, you notice almost immediately! This project taught me just how much I could improve my skills. Near the end of the project, I started to feel that some of the earliest work was subpar. Of course, what had happened was that I had grown. When I completed that work back then, it was representative of my best.

Now, this leads to another problem. If you are constantly growing and improving, then your work standards are also rising. So if you work on a project for long enough, you will eventually feel the need to redo your older work. This is a problem I had never encountered before, since I only worked on multiple smaller projects. As a commercial project, you have to release at some point. Which means that at some point, you will have to compromise with yourself.

Nothing you create will ever be finished, it will never be perfect and it will never be representative of what you feel is your "best work". That's something we all have to accept eventually, otherwise we will keep working on the same project forever.

Lesson Four: Start with a Theme.

When I began working on Outrealm, the first thing that came to mind were the basic game mechanics, progression and control scheme. I whipped open my program of choice and start madly producing code. In a few days, I had a very rough prototype, a "proof of concept" to help me understand if this is something I want to develop further. "This game should be Risk of Rain meets A Link to the Past."

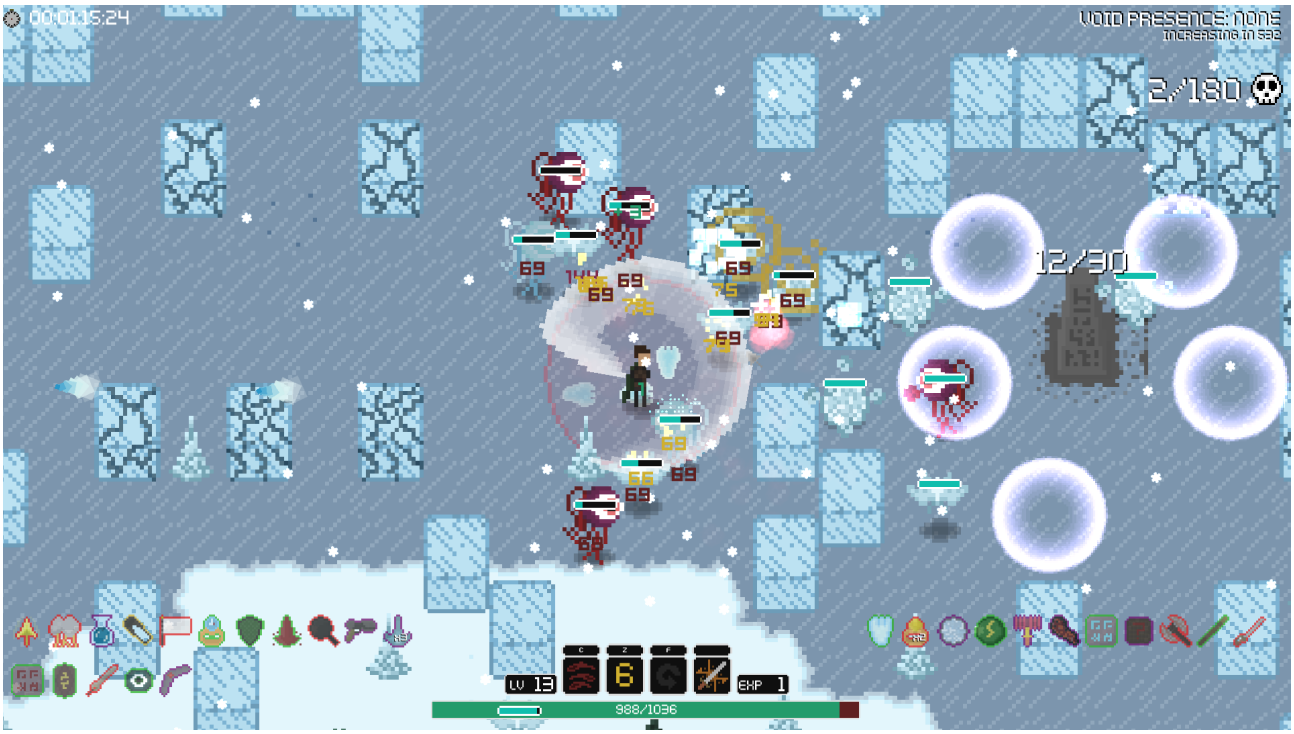
While these sorts of descriptions are useful for describing how a game plays, they miss something very important. Take the recently released, critically acclaimed Super Mario Odyssey. When you think of that game, what jumps into your mind? What is the one critical element that ties in everything? If you guessed it, hats off to you.

Super Mario Odyssey is an example of what a game looks like when designed around a theme. Hats permeate every part of its design and aesthetics. Every NPC in that game wears a hat. Mario interacts with friends, foes and the environment through his hat. The game begins in a "Hat Kingdom". Many of the bosses in the game are defeated by removing their hat. They didn't say halfway through "Hey, did anyone notice there are a lot of hats in our game?" It was a decision from the start, and showcases the power of a game designed around a theme and not vice versa.

A powerful theme ties every element of a game together. In many ways, a theme is the most impactful part of a game. It is a powerful form of branding, and what identifies your game in the mind of others. Elements of your game that tie in to a central theme make sense in the mind of the player. Mario throwing his cap makes sense, because everything is about hats.

And theme can be just about anything. It does not have to be an object. It can be a concept, an idea a feeling, or a belief. The theme of my favourite game of all time Metroid Prime 2: Echoes is Light & Dark. The theme of Overwatch is Being a Hero. This is just as important for narrative-based games. If you have an important message within your game, then that message is your theme.

Beginning a game with a theme is an incredibly powerful way to go about development. From there, you can decide on how to tackle elements such as story, mechanics, UI and music. With Outrealm, I only started thinking about theme around two-thirds into development. At that point, I already had a game and many of the elements finalised. Outrealm is fun, but not particularly memorable.



Lesson Five: Don’t make excuses for yourself. Your work should stand up regardless of context.

Another trap, one that I fell completely into is the “Indie Pass” trap. This is where low-quality work is excused because it was developed by an indie team. Now, I have always been adamantly against this kind of attitude, which makes the following all the more ironic.

One of the reasons I wanted to make a game solo is to show exactly what I could do. I wanted to see just how good of a game I could make by myself. When I word it like that it sounds good, right? Challenging myself to accomplish greater things. But even though I didn’t want to

admit this to myself, really what I was doing was giving myself an indie pass. By limiting myself to a team of one, I was secretly hoping that when people looked at my game, they would say "Well, it's pretty good for a game made by one guy".

And heck, perhaps there isn't anything wrong with that. No matter how hard we try to avoid bias, we always factor in context when appraising things. We have higher expectations of AAA studios because they have more resources. So by the same token, is it really wrong to expect less from a dev team with less?

Well what I think is that the answer doesn't matter. I was using the fact I was a solo dev to legitimise failure in my own mind. Starting a game by making excuses for future failures is no way to approach this industry. At the end of the day, your work stands on its own. Can you imagine plastering the following message on your game's steam storefront "By the way, this game was made by just one person". As if it would smugly blow people's minds when they found out.

So what does this have to do with putting myself out there? It means from day one, I was afraid to put myself out there. Really show my skills with nothing to fall back on. I was absolutely terrified to show off my work. Thinking about it now, instead of giving myself excuses for failure, I should've given myself reasons to succeed. I had to stop sabotaging myself.

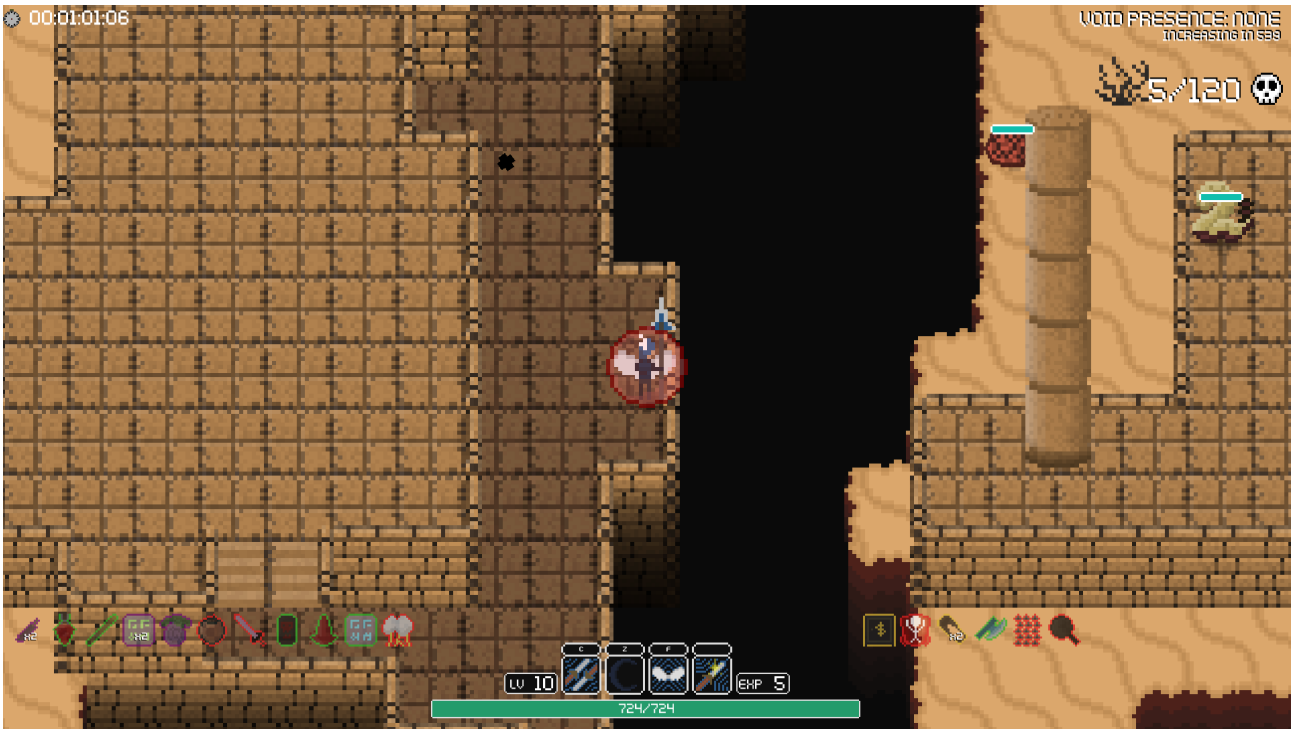
Lesson Six: Make an educated and calculated decision about tools, particularly engine choice.

There are going to be some people that read this and disagree, but I stand firm in my opinion. Something I didn't realise is just how important choosing the correct game engine is. Your game engine shapes your entire development and limits what you can and can't do. In this matter, I chose poorly.

This isn't a jab at Game Maker Studio: The software I used. It just means that I, the developer, chose poorly. In my case, picking GMS limited me in two major ways. It stopped me from being able to release Outrealm on the Nintendo Switch, and it stopped me from implementing multiplayer. Not that GMS does not support multiplayer, quite the opposite. But I didn't have the skills or time to implement it. Choosing a different engine (Unreal 4, for instance) would have let me, me specifically, achieve those two goals.

GMS was perfect for me when I was first learning how to create games. It lets you rapidly prototype game concepts. You can take an idea and turn it into a game quickly and easily. It's very strong for developing single player 2D games for PC, Xbox, PS4 and Mobile. And even with my criticism, GMS made laying a lot of the early groundwork very easy. I traded early development strength for later limitations. At the time I felt that it was a good decision, but now I am paying the price near the end of development.

This also applies to all tool choice. Choose wisely! Don't take the easiest path, make the best decision for the long-term. Take some time and make an educated, calculated choice. Of course, actually being able to make the correct choice all comes from experience.



Lesson Final: Understanding Pride

Pride is a real double-edged sword. Humans have a strange understanding of it, as we are both encouraged to be "proud of people" while at the same time, recognise it as one of the seven deadly sins. I wrestled with pride while developing Outrealm, and came to what I believe is a valuable conclusion.

First of all, It is extremely difficult to be proud of my own work. How can I be when such good work, better work, already exists? People say my art looks good, but all I think about is how much better Hyper Light Drifter looked. This attitude might be seen as preferable to the person who is proud of anything they make. But of course, both have serious flaws.

I can't be proud of most of the work I make. Not yet. But, there is something I am very proud of. And that's my personal journey as a developer. When I first opened Game Maker Studio in 2014 after two whole years of doing absolutely nothing with my life, I had one very simple goal. Make each game better than the last. That was my only rule. Going back to look at those old projects, and I am shocked by just how much I have improved. It's staggering to see such a radical change in yourself.

And just to clarify, there is nothing wrong with being proud of your work. I think it's quite healthy, and important in shattering feelings of self-doubt. When you're surrounded by top-quality game releases big and small, it's hard to feel like you've done anything worthy. At least that's how I feel. That's why for me, It's easier to focus on my journey as opposed to the stuff I'm actually making.

There's quite a few things I'm proud of in Outrealm. I'm proud of the Smasher and Lancer classes, both are challenging and rewarding to play. I'm proud of two songs in particular, themes from the Snowy and Desert Realm. I'm proud of the fact I was able to integrate into Steam's systems and implement working achievements. But all of those pale in comparison to how proud I am of my personal journey. Outrealm represents my biggest step forward yet as a Game Designer. A journey that began in 2010 when I attended my first class of University.

End

Thanks for reading through part 1 of this multi-part sequence. Originally I was going to leave things here, but I received many questions about development while writing this. Part 2 will be finished and released sometime within the next two weeks.

At nearly 3500 words, I doubt many will bother to read this far. But that's fine, because at the end of the day, I wrote this for myself. It's one thing to experience a lesson, and another to learn a lesson. I'll look back on this in the future and hopefully, will smile at the now-obvious lessons I had yet to learn.
