

MA Commercial Music Production Assignment

x

Tileyard Education,
University of Wales Trinity Saint David

“To create a world.”

DEVELOPMENT ELEMENT 2 - ESSAY

Michael Dahl Rasmussen - 2009231

March 2021

MY STARTING POINT

Video games have always held a dear place in my heart. Long before I had ever thought about playing music, starring in a play, or singing - all parts of my professional vocation today - I spent hundreds, if not thousands of hours playing Nintendo 64, Playstation 2, PC games, Wii, and the list goes on. Indeed, games meant a lot more to me than even listening to music, all the way into adulthood. So it's probably reasonable that what I like in music in general is heavily influenced by the games I've played. Whether it's the epic orchestrations of *The Legend of Zelda*, the romantic melodies of *Final Fantasy X*, or the playful and quirky nature of any *Super Mario* game, it's found a way into my own writing. One of my first dreams was actually to become a video game designer one day, although that dream was overtaken by acting and music etc..

So when I was presented with the task of creating a musical instrument (exciting in and of itself) and one of the options was to create a video game, including music, *from scratch*, I jumped at the chance. Not that I ever thought I would be able to create anything groundbreaking or maybe even artistically interesting given my limited experience in the field. I saw our tutor Ciarán Robinson demonstrate a very simple game scenario in a lesson, and although I was intrigued and encouraged, I expected my own game to be simplistic, with maybe one functioning game mechanic. It would entail learning to use two programmes I had never even heard of, FMOD (for the sound aspect) and Unity (for everything else), including the fact that I would need to code - which is also completely new, and notoriously difficult. I thought I'd just have time to make something that barely demonstrates an idea, not an actual full game.

But I was wrong. In the end I managed to create a game that to me (and my wife) looks beautiful, sounds great, is fun to play, and has a unique concept.

So let me tell you how it came to be.

CONCEPT

It is no secret among my professional peers that I love music theory. I think it's great to be able to understand the music I'm writing and listening to, and to be able to communicate ideas of how to make it with others. But the most important aspect of my theoretical training is in no doubt my ears. The ability to listen to a melody or chord sequence and be able to play it back immediately on my piano is not only fun and useful, but has saved me hours upon hours in a variety of contexts, including transcription, singing teaching, musical directing, composing, and learning new material. But the most important part is arguably that I am able to *interact* with music and other musicians in a completely different way. Or a musical surprise happens in a live context, and I am able to react musically.

Yet, in spite of the seemingly obvious benefits of such a feat, I still hear a lot of reluctance from my peers when it comes to music theory and ear training. Typical remarks go along the lines of "it's boring," or "I'm not very good at it," or the one which baffles me the most: "then I'll lose my creativity!". While the last one requires a longer debate which is beyond the point of this essay, the first two I can definitely sympathise with. When I learned note recognition and intervals as a teenager, I did it because I had to pass a test in order to study Musicology - there was no way around it. I tested myself mainly using simple utilities on musictheory.net which played the notes, and I had to press the right button while keeping track of my score. It was kind of a game, but only at its most basic form. Surely that process could be more fun?

Now, years later (I started my Musicology studies in 2010), there are tons of musical games out there. On one end of the spectrum we have *The Legend of Zelda: Ocarina of Time*, which while having the player use an ocarina (a simple wind instrument made of clay or wood) to perform a variety of feats, the interaction with the instrument was more a question of memorising the right buttons to press rather than using your ears. Plus, the musical systems and scales were not immediately logically applicable to any other instrument outside the game.

On the other end of the spectrum are websites like thetamusic.com. Here, the user can play a series of mini games separate from each other, each training a different musical aspect. The games are simple, with a 2D layout, and do indeed make the process more fun, if not exactly as immersive as an adventure game like *The Legend of Zelda* - not that they are trying to be!

Several other games have of course used some kind of musical interaction as part of the gameplay (*Sing Star* and *Rocksmith*, to name a few), but I had never seen a game which attempted to be truly immersive (so as to make the player forget that they are practicing), while simultaneously practising skills which can be directly applied to a real-life musical context. So I wanted to try.

The first concept that came to me was this: Music is playing in a certain key - there are creatures in the world which sing an interval in the same scale as the music - you need to find the kind of creature that sings the particular interval you have been tasked to find (for example a minor third) - and then *do something with it*. That's it. That eventually got fleshed out as follows: Music is playing in a certain key (C major) - you are equipped with a flute which can play an octave of that scale - the flute can also shoot projectiles - there are dragons in your homeland that you need to get rid of - when you shoot a dragon, it sings a pitch within that octave - you have to play the same note back to it with your flute - if you succeed, the dragon explodes and you get a star - if you don't, you lose hit points - find all the stars to win - on the way you can pick up hearts to replenish your health, and sapphires, because who doesn't like to collect *something*?

Initially, as mentioned, I wanted to have the game be about interval training, but it turned out that the coding for that would be infinitely more complicated than single note recognition, so I stuck with that for now. It still, however, means that we are practicing not only note recognition, but *relative* note recognition, because the notes are always played in the context of the background music in the same key. That way, the player will train hearing what scale degree a certain pitch is in any song (in a major key), rather than just learning pitches and intervals out of context. Seeing as the shooting function plays the root note of the scale, it also means that there is actually a kind of diatonic interval training: If a dragon sings an A, that is a major sixth from the root, which played when you shot the dragon.

This concept can of course be played in many different ways, but I decided to go for a First Person Shooter [FPS] to immerse the player as much as possible in the big world I created. (I would lie if I said that decision was not also influenced by the fact that our tutor Robinson wrote a book, *Game Audio with FMOD and Unity* (Robinson, 2019) [*Game Audio* from now on], on creating a game just like that.)

Of course, a game needs a name. Seeing as the game is about dragons and scale degrees, it seemed obvious: *Dragon Scales*.

MAKING A WORLD

One of the most satisfying parts of composing is creating something out of nothing. Looking at a score or listening to an album, and knowing that that came out of *nothing*, and nobody had ever seen or heard exactly that before you made it. It would soon turn out that that is exactly how it feels when you are creating a game world.

As the first thing in my journey, I designed the visual aspect of the game. I knew I wanted some kind of high platform or mountain from which to start the game and gaze over the world. Through trial and error, and Robinson's book (Robinson, 2019), I managed to create the 3D outline of the world: a mountain in one corner, a high waterfall in the opposite corner, and some slight hills and slopes on the planes underneath. This would hardly make for an appealing game just yet, and rather than populating my world with trees and environment from Unity's stock choices, I decided to venture into buying my own assets for the task. I settled on a package called *Fantasy Forest Environment* by TriForge Assets, because it had a slight cartoony feel to it, and because the style matched that of the dragon assets I had bought previously. Seeing as the package came with environments for different seasons, I decided to have 4 different sections: The mountain, the planes, a winter forest, an autumn forest. This, I thought, would also be a great opportunity to employ

different scores for each section. After what at that point felt like a guilty amount of time (spending 3-4 hours on just the visuals!), I had a world which to me was stunning. I thought, seeing as I'm gonna be spending quite some time in this place, I might as well be thorough and make it pretty. Little did I know how quick it really was, and how easy that whole process was in the context of what's to come, and after having lived through what followed, I'm really glad I took that time.

COMPOSING THE MUSIC

On day two I ventured into coding some dragon behaviour, but after spending 8-9 hours trying to solve one single problem (more on the joys of coding later), I decided to hold off with that, and write the music instead. I knew I wanted it to be one single piece in four different arrangements which would gradually crossfade in and out of each other as you entered the different regions of the game. I also knew I wanted it to be in C major, so that any musical experience gained in the game could as quickly as possible be translated into playing the keys of maybe a piano. Also, there is something nice and calming about beginning your musical adventure in the key of C, with next game levels including more "advanced" scales (imagine the first key you're learning to be for example G# minor - where is the logic in that?).

Seeing as the whole concept is based around the player making melodic content, I knew it couldn't be too busy. It had to be a slow tempo, with few, slow chord changes, so that the player could really feel what impact a certain note has in the context of a certain chord. The chord progression I ended up with was: (Fmaj7, Cmaj7) x 4, F69#11/A, G69add4/B, Fm11/Ab, Cmaj7add4/G. Now, while the second half might look complicated for what's supposed to be a music game for beginners, think for a second about the complexities of the Super Mario theme. I didn't want it to *sound* like a game for beginners in music. I wanted it to still be musically engaging, original, and to inspire the player to play the flute in the game - not just as a means to blow up dragons!

In terms of sounds, I went for long, sweeping pads and noisescapes. Using primarily a synthesizer called Omnisphere, the sound for the four areas are all synth'y sounding - I wanted it to be in contrast to the acoustic nature of the player's bassoon sound (more on that in the next section) and the dragons' howls. There isn't really much that you can call a melody, except for the notes which happen to be on top in each chord (one note has to be on top, alas), again so as not to confuse the player with melodic material mid-game. For the Mountain, the starting area, I set the stage with the long and sweeping sounds. For the Winter area, a lot of the arrangement fades out, and we are left with a more barren, almost howling soundscape. For the Planes, a pulsing arpeggiated bass is driving the energy forwards. And for the Autumn area I went for a lighter, more glittery texture, almost like falling leaves.

At the end of a day which at first seemed to be a complete waste, I was happy to see the music almost writing itself.

DESIGNING THE SOUND

Following along in Robinson's book (Robinson, 2019), included in the book resources there was a host of sound effects ready to use for footsteps, explosions, and various vocalisations. To learn how the whole thing worked, I initially used his examples in my game, but eventually I would replace them with my own. The easiest ones were the vocalisations: It didn't take me long to do my own "jump", "land", "ouch", and other exclamations using my always-handly SM7B, and replacing them with Robinson's.

For explosions I found a website (actionvfx.com, 2021) which in exchange for you signing up for their newsletter would give you a wide variety of explosions for SFX. Perfect.

For the dragon growls I used sounds of tesla dragons and dogs growling from freesound.org. Seeing as I was short for time, and I didn't have a good mic for capturing audio outside, I chose not

to sample my own outdoor sounds, so footsteps, wading in the water, waterfall sounds, and more was sourced from the same website.

For the dragon howls, the ones you have to figure out the pitch of, it wasn't as easy. It had to sound like something which would come from a little dragon, and have a clear and consistent pitch. It also had to stick out from the musical soundscape I had already composed. At first I chose a sample of Tuvan throat singer - it sounded like a growl, but also had a consistent pitch. However, when I started transposing it, it sounded simply awful, and I discarded it. I managed to find a sample of a dog howl, select a snippet of it that sounded more or less consistent, pitch corrected it and smoothed the pitch, then made 7 duplicates and transposed them appropriately. Some occasional DAW editing had to be applied due to the artefacts created from transposing, but I succeeded in the end.

It was easier for the flute sound. I have Spitfire's *BBC Symphony Orchestra Professional* sound library, so I had a good feeling it would be something from there. Initially I went for a clarinet sound, but it lacked a few notes in the bottom range (at first I wanted to have two octaves of range). Then a flute sound, but the low end of a flute is very weak, and I feared it wouldn't penetrate the score as easily. In the end I settled on the upper range of the unconventional-for-the-occasion bassoon. I found it to be a more unique sound, and the nasal quality of its upper octave both stood out in the mix while still sounding pleasing. (The critical player might notice that the instrument wielded looks nothing like a bassoon, but then again they would also have to come to terms with the fact that you are making singing dragons explode with a musical note. Realism has to hold out for another day.)

IMPLEMENTING THE SOUND

For the most part, it was relatively simple to get the sound to work. I struggled for a while with the main theme music to get the transitions between the areas to work smoothly. A problem I had for a long time was the persistent arpeggiated bass of the planes which wouldn't stop playing no matter where I was in the game. By having the sound emanate from four so-called Empty Objects in the game, I tried from both FMOD and Unity to configure the parameters so it made sense, but no matter how small I set the Planes distance parameters, I seemed to be able to hear it from everywhere. While initially giving up on the matter, I found out it had something to do with a gradual lowpass filter as you left the area: the music didn't stop as you left the area - the lower frequencies (including arpeggiated bass lines) kept going. By automating the gain of the area to attenuate gradually to zero as you left the area instead, I made it work.

The foley sounds worked smoothly most of the time, but the waterfall kept stopping after a while. It turned out that with a setting called "async" switched on in FMOD, it didn't loop properly. As another water matter, the wading in the water (or, change of footstep sound), didn't work for me either. I tried creating multiple invisible platforms in the water that would trigger a splashing sound when touched, but it sounded horrendous. It turned out that consulting Robinson's book (Robinson, 2019) was once again the answer - using something called a RayCast, the surface underneath the player is constantly scanned. When it sees that the surface has the tag "SurfaceA", it uses the normal footsteps. When it registers "SurfaceB", it uses the wading in the water. Then I just needed to insert some invisible platforms in the water with the tag "SurfaceB", and it worked like a charm.

I would have liked the main theme music to come under a lowpass filter when an enemy was near, to make the player alert, and to highlight the dedicated "monster is near" rhythms that are playing. I tried it using something called a "snapshot" in FMOD (when entering this area (the area around the monster) a certain effect is applied or music is playing instead of what's normally playing), but failed to make the transition in and out of that area smooth, so it sounded annoying if you were on the border, and it constantly switched back and forth between the normal and low-passed sound in an erratic way. So I scraped the idea this time around.

CODING

“There is no reason why this shouldn’t work,” was a phrase uttered hundreds of times by both myself and my tutor Ciarán Robinson, who was luckily willing to grant me hours of tutorial time in trying to make this work. I knew that coding was gonna be a steep learning curve, with many frustrations, but after spending 9 hours of a day on *one single problem* and it still not working, I really understood what that meant.

Being a complete beginner, at first I simply copied over the examples from *Game Audio* (Robinson, 2019) into my own game, not having a clue what it meant. I thought it would be hopeless to actually grasp what the different lines meant - as long as I copied it over carefully enough, it should work. But I was once again wrong.

On the aforementioned fateful day, I followed a YouTube video (Hibberd, 2020) to get the dragons to wander around when I wasn’t near, and to charge towards me when they saw me, to once again wander around when I was out of sight. As with many things in coding, just because it sounds simple, doesn’t mean it’s simple to code. I painstakingly followed every single syllable that Hibberd demonstrated coded in his game, and it didn’t work. I deleted the script, as it’s called, and started over. I still hit my head against the wall at the same exact spot. The next day, I finally got it to work after spending over 90 minutes with Robinson in a tutorial trying to solve it. I couldn’t tell you how we did it in the end, those 24 hours are still a blur.

After running in to a multitude of problems like this, and experimenting with how to solve it myself, I eventually came to understand the basics of coding in C# (the coding language that Unity uses) - enough to troubleshoot why things weren’t working, and to start getting things to behave the way I wanted. For example, I gave up on having the hearts spawn out of the crates although it seemed so simple, but a few days later I returned, and managed to get the crates to drop both hearts and gems.

Another crucial bit of coding was the essence of the gameplay: Getting the dragons to sing a pitch, “hear” what pitch the player is playing, and to react accordingly. Here I was fortunate enough to have Robinson come up with the code to solve it: When the dragon is hit by a projectile, it sings a random note from 1-8 (equivalent to an octave), and that note then becomes the “right note” - when the player then presses the key for one of their 1-8 notes, the code checks to see whether the two numbers are the same. If the numbers are the same, a series of events is triggered: the dragon vanishes and spawns an explosion effect, it spawns a star, it triggers a celebration sound from the player, and a you-got-it-right musical sequence. If the numbers are not the same, the player loses a hit point, and the player emits a little cry of pain. Further to this we also needed a script (where the code is written) attached to an Empty Object on the dragon to make sure that note-recognition code was only triggered when the player was close to the dragon - or else the player would randomly die from playing notes when there wasn’t a dragon near.

I parked many wishes for the game early on, believing that they would not be possible to achieve this time around, but I surprised myself many times by actually getting them to work (and thus setting my ambitions even higher), not least because of a healthy amount of advice, help, and suggestions from Ciarán Robinson (if you read this, thank you!).

TESTING THE GAME

On day four of my journey I believed I had finished the game for this assignment, and I had my wife play it. To say that she is new to video games would be the understatement of the year, and she has zero musical training or knowledge (apart from what has inevitably stuck after my long rants on the subject), so if she could figure it out, it would be a home run. After demonstrating the basics of an FPS, she walked about and eventually killed her first dragon. Sure, with enough trial and error, everybody could do that. But it didn't take many minutes before a dragon sang an A, and she immediately and instinctively went to press 6, thus killing the dragon in the first try. I exclaimed in elation (it's on video): She had started to learn the basics of ear training without even trying! Proud as I was of my achievement (and hers), she told me that she missed something: when you blew up the crates there was nothing inside (this was prior to a few coding successes), the purpose of the dragon-is-near music wasn't clear at all, and there was a weird latency between button-press and flute sound.

So the next day I went back, and fixed all those things. The dragon-is-near music I made louder and directional, so it was coming from the dragons themselves, and with an energetic piano part (nowhere else in the score is there anything that sounds like a real piano, so it's clear that it's a special situation).

Finally, I added a little text in the beginning of the game to explain the premise and controls.

REFLECTION ON RESULT AND NEXT STEPS

As mentioned several times, I am quite frankly over the moon with how it turned out. A few things could still be improved, however. I think the sound of the dragons' howl could be clearer, and also match the exact octave of the flute - currently it is an octave above. It would also be great to have a little piano graphic show when you play a note to indicate what note you are playing on the keyboard¹. Likewise, a subtle and gradual introduction to real music theory and ear training concepts is obviously also in the scope.

Furthermore, to make it a full-scale video game, developing the gameplay and adding levels. This could include making the patterns to be recognised extend to several notes or maybe chords, and the music played be following different scales. Maybe even add boss dragons, which require more to kill. To take it to the next level, it could be integrated with a MIDI keyboard, and the notes could be played on that to optimise learning (and maybe fun).

However, I want to involve other people in the project. I believe the game has entertainment and educational potential, and next steps would crucially be finding a game development team to take it to new heights, with me focusing on concept development and composing the music. I have never worked with game developers, though, so some research is ahead of me.

I have personally never seen a first-person adventure game which combines the elements of an FPS with playing music as an integral part of the gameplay, and I hope it can spawn a whole new field in the game scene: Explorative and adventurous music games, rather than the already plentiful arcade-style music games where the focus is only on getting it right as much and quickly as possible.

Overall this has been a fun and rewarding experience, and I highly doubt this is the last of my involvement with game music and design.

¹ I may have added this by the time I submit.

References

ActionVFX (2021) <https://www.actionvfx.com/collections/free-explosion-sound-fx> [Accessed March 25th 2021].

Brackeys (2020) *How to make a HEALTH BAR in Unity!* [online video]. Available from: youtube.com/watch?v=BLfNP4SciA&t=848s [Accessed March 25th 2021].

Freesound (2021) freesound.org [Accessed March 25th 2021].

GameDevTraum (2020) { *How to DISPLAY TEXT on screen in Unity* } - *Text Canvas Component* [online video]. youtube.com/watch?v=LuAAzNDhVko [Accessed March 25th 2021].

Hibberd (2016) *Unity Tutorial - Monster AI* [online video]. Available from: youtube.com/watch?v=8DZAXuIUgvY&t=14s [Accessed March 25th 2021].

Robinson, C. (2019) *Game Audio with FMOD and Unity*, Oxford: Routledge

Zotov (2017) *How to create a simple coin counter in your Unity game? Easy Unity 2D tutorial.* [online video]. Available from: youtube.com/watch?v=-EIXQHxoicg [Accessed March 25th 2021].