**Title**

**しゅうちゃく**(Shuuchaku Obsession)

**Process Overview**

Upon starting this project I didn’t truly take into account the amount of work and expectations that I was placing on myself, so I ended up getting a *lot* less done than I originally planned to do. To add wind to the flames I also lost a substantial amount of progress due to processing crashing after my blitz run on phases 2 and 3, so that made things even worse. Another big factor that contributed to me not being able to finish the game were the incredibly vast amount of errors and bugs that I came across during development. Fixing these bugs often took around 20-30 minutes of thinking, trial, and error, with the more substantial errors taking me at least 2 hours to realize what the problem was. Overall if I were to rate my success out of 10, I would give it a 5.

**Detailed Process Reflection**

**Ideation**

The ideation phase was quite fast actually. Originally I didn’t know what I was going to do, but a couple days later while talking with friends about how Goku would play Sans’ role I thought, “What if this was an actual fangame?” I began to think about how the premise would change if it was in a dragon ball environment, and all of the creative juices started to flow from then on.

**Design**

When starting designs for the game, I settled on the fact that the assets would have to be pretty high quality(and not just 8bit simple pixel art). I then, with the help of several pictures and a nice new mouse that let me adjust my DPI, I drew a bunch of Goku sprites in paint.net. Thanks to tools such as mirror mode I was able to make the drawings look pretty accurate(though if you zoom in you can see some jagged, sloppy lines). After drawing the gokus I planned out the FSM on a piece of paper and planned out all of the attacks in a google docs document. Finally, I thought up of some dialogue(with the help of some friends) on another piece of paper. All of this took around 2 weeks to complete due to me having to keep up with other classes. Most of this was done around a week or 2 before the due date. I never got around to making any music for the game and resorted to other sources for the sound effects(no music but cliche effects).

**Implementation**

Ah yes, the most “fun” part of the project. To start it off I used some of the basic structure from my last game. This included the minim library, the title screen, most of the controls code(changed to have somewhat better conditionals), and functions for fading backgrounds. From there I decided to focus on the gameplay of the game, since I knew it would take the longest to complete. A good thing was that while messing around with text before I even finished the design, I figured out a way to not only animate text, but a method of moving through an array of PImages at a speed independent of everything else and without adding several constructions of the same image to increase time between each frame. I did this by creating a variable(framespeed) and setting it equal to the framecount divided by a number, and then setting that variable equal to a variable(i) that represented a frame of the animation(or in the case of text, a character within a string). The bigger the number I divided the framecount by, the slower the animation was. Each time framespeed was equal to i, I would increase i by 1 which in turn goes to the next frame in the animation. That made animation a lot easier to handle and allowed me to use a lot less code. Once I implemented that, all I needed to do was worry about the attacks. I also had just learned about all the labels of the unit circle in trig, so I was able to get a deeper understanding of rotations and use radians instead of degrees(which is way more convenient). There were a lot of bugs I had concerning hitboxes and duplicating attacks to the player 2 box that I fixed while I was speed blitzing phase 2 and 3(same day I lost all of that progress). Fixing those was mostly just a matter of using math functions. I used the distance function to calculate slopes and I was able to simply draw the same attacks at an extended x location in order to duplicate attacks for player 2(since the y values would be the same). The most interesting wall I ran into though was when I was coding the guard mode, a mode that relied heavily on intervals of time. This was what kept me from turning this in on time, and I really hate that I ended up losing all of it. The goal of the mode is to press the buttons in time with a *ding* sound in order to parry them and avoid taking damage. I used the framespeed method to control the dings. I set the framespeed to I think framecount/40, and I used the *i* variable to handle a countdown in the form of a string(3 2 1). Once *i* was equal to the final number of the countdown, both players would have an attack(punch, kick, ki blast, or kamehameha) thrown at them. At this point I used a conditional to say that if the desired button was pressed and the framecount was less than 90, a shield would appear around the player blocking them from the attack. From there it was just a copy + paste fest for each time I used the mode. The problem was that the conditional wouldn’t work and I could not figure out the reason why for the life of me. The way I figured it out was actually hilarious. I vented out my frustration in my discord server after 2 days of on and off debugging, and talked about how the universe must hate me(which I still firmly believe it does now), but then immediately after that I decided to see if the actual problem was that the framecount wasn’t resetting by resetting it for each time framespeed equaled *i*. Just like that the problem was fixed with a single line of code. I couldn’t even express how stupid I felt in that moment.

**Iteration**

*Hardcore iterating in progress…*

So I actually got my mom and some of my friends to test run the game prior to making the adjustments you suggested. When observing both my mom and my friends one very apparent similarity between their experiences is that instructions were clearly lacking. My mom’s first intuition was to use the mouse in order to click the options, in which I had to tell her that the game functions with the keyboard. After being told that she picked up on maneuvering through the UI pretty quickly, but more problems arrived when it came to selecting options, going back, and even playing the game itself. Of course when playing a game the amount of possible keys that could perform these functions are endless so it makes sense that she didn’t know what to do from as early as the title screen. My friends voiced the same problem to me when they played, in which they encountered the same problems as my mom. Afterwards I simply added text notices that indicated which buttons did what and just like that the problem was solved.

Another problem I sort of had but didn’t really put much thought into was getting text to scroll while the background is constantly looping. The way I went about animating is using the framecount method(which I now see is bad) and having each character appear individually from a preset string value. It basically runs through an array of characters to create a string. The problem I saw with this is since this value is constantly changing, the previous value isn’t being saved. So if I were to run a background or anything it would draw over the previous character that was animated. I couldn’t think of an efficient way to handle this issue(the best idea I could think of was individually having each character of the string that had been shown so far printed in its place but that sounds like a lot), as the way I was doing the dialogue was already not that efficient. It would be interesting to learn of any techniques that would fix that. I also had many problems with adjusting hitboxes for some of the attacks(especially for player 2, they’re far from good, it’s kinda sad). More specifically I just could not think of a way to accurately calculate the hitbox for a diagonal kamehameha attack. I thought about doing something with the attacks overall area but I don't really know how I would check for overlapping areas when one of them is diagonal.

**Final Analysis:**

As I said before, I would give this about a 5/10 success rate. My main issue was my lack of understanding as to how much work I was piling onto myself. I don’t know why I thought I could code a fully functional 3 phase game of this level in the incredibly long *2 weeks* of time that I gave myself. As the game came along the quality of the mechanics, my creativity, and my will to code decreased heavily. This led to me poorly planning steps of development, such as spending too much time on assets, and not being able to hit the deadline for the project in time, as well as me enduring long periods of frustration over errors and an immeasurable amount of anger and sadness after losing 66% of what I spent so long on. Even if I didn’t lose all of that work, at the point of phase 3 I was starting to throw away a lot of the ideas I planned to implement in favor of being able to finish the game at a presentable time(which was already very late). I think I should’ve chosen a much less ambitious idea like creating a standard platformer, which would’ve allowed me to spend a lot less time on assets and more time on coding. The standard platformer would also be nowhere near the level of quality that this idea demanded, so I wouldn’t have as many things to code either. This might have allowed me more time to draw backgrounds, work on music, or add more interesting mechanics to the game. This game didn’t have to be super omega high quality in order for me to successfully convey everything I learned this year. This project taught me that when given a due date, especially one that’s not that far away, it’s important to understand yourself and to know your capabilities so as to not overload yourself with something you thought you would have been able to handle.