Bonds: UI and Menu Research / Implementation

Charlotte Weiss

Triangulation: Problem Analysis, Best / Good Practices, Root Cause Analysis

Reflection

My biggest takeaway from this was we did not allot ourselves enough time with Unity. It is a program none of us knew how to use with a language most of us had never even touched. I severely underestimated the amount of work it would take and I truly believe if I had even a few days more I could have figured something out. I am still proud of some things I learned, such as working with capsule colliders and learning some basic C#. It was simply an issue of underestimating how difficult it was, since I even said I would handle this set of tasks by myself, which was not a smart move. Next time, I want to allow myself more time to work on implementation, as well as asking for help if it's something I have never done before - Even if I think I can do it, I need to ask for help.

Summary

I tried to create a Menu UI and player interaction within Unity for our game bonds. I researched how others go about tackling a similar problem, then tried to analyze the best way to implement for our project, and finally looked for the root issues with the way I went about this particular problem

Sections are as follows: Problem description, research / resources used, version control issues, collider interaction set up, UI creation, and problems finalizing implementation.

Problem

I have been tasked with creating the menu UI and player interaction UI in Unity for our game Bonds. We need a menu that allows a player to Continue, start a New Game, and Quit. We also need a player interaction between player and NPC that allows dialogue to show on screen. Once one player had interacted with an NPC though, I needed to lock the interaction to be in line with our game idea so that only one interaction per NPC.

Research

I first attempted to see if something like this had been done before, so I could learn and follow what others had done previously. I have no prior experience with Unity or C# so I needed a tutorial that would break things down for me. I found three points of reference for possible implementation. My plan was to use this video as a point of reference for player interaction, this video to do the kind of menu UI I was looking for, and then fall back on this tutorial if I could not get the second one working. I assumed that I could use a similar menu UI method used for dialogue options in the second one to create a menu UI that executes a specific function per button interacted with. Continue would close the menu, new game would restart the scene, and quit would end the scene / game. From all my research the methodology seemed simple enough.

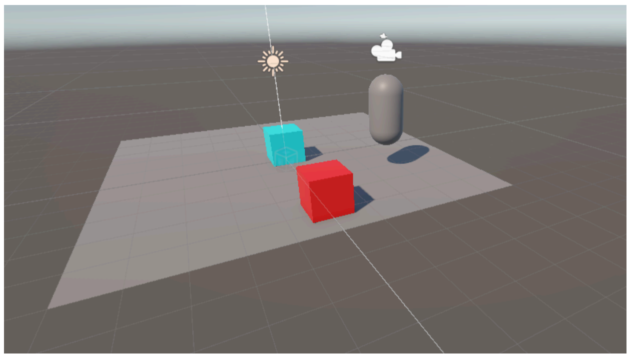
Attempted Implementation

Version Control Test

I first tried to set up version control for Unity. Unity *technically* has version control for a subscription price, and while I could have it work on the smaller section of the project I was doing, it would make it very difficult for others to access it. Github was also an issue, as the file sizes contained in Unity in any section of the project are far too large and cause immense issues when connecting to GitHub. Nieck was able to get a GitHub branch working for the project using gitignore.io which was at the recommendation of Maikel to all of our team. However, I was unable to get the branch working on my end in a reasonable amount of time. As I was working on this section by myself, I felt it imperative to focus my efforts on working on the product first.

Player Interaction

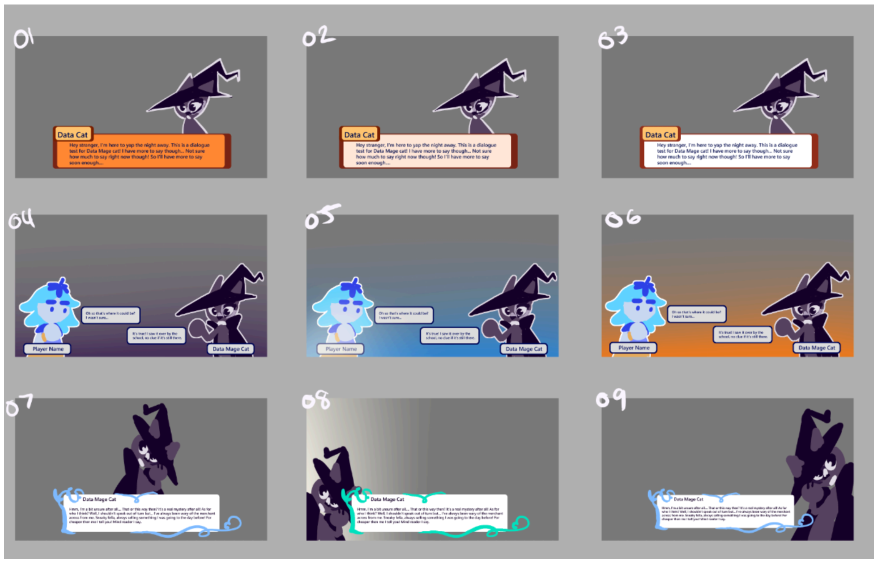
Using the player controller developed by Nieck, I devised a simple setup to test interactions and develop the player interaction system. The capsule is the player, and the two cubes are NPC's. I attached any changes or scripts I made to materials, since if I created any work as a material, it could easily be dragged onto any NPC we created.



I started with section one of the first tutorial, how to talk to NPC's and created a detection and collision system on both the player and the NPCs. I just added capsule colliders to the player and NPCs and ran a script that detects when a player is near an interactactable object. Then, if a player presses E, it acknowledges that you interact with an NPC in the console. This is simply done with those colliders acknowledging each other at a certain distance, and then pressing the button. This part was relatively easy for me to set up.

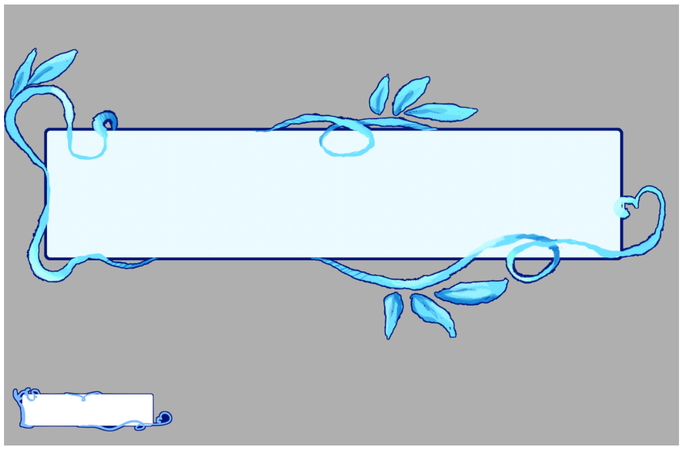
UI

I created very quick mockups for testing purposes of the UI that would be used. I wanted designs that would still show off gameplay behind it in case Yoshua wanted to do animations during dialogue. I also was trying to make sure that it would be clear when dialogue was on screen. I showed these to both Jan and my group and received the following feedback

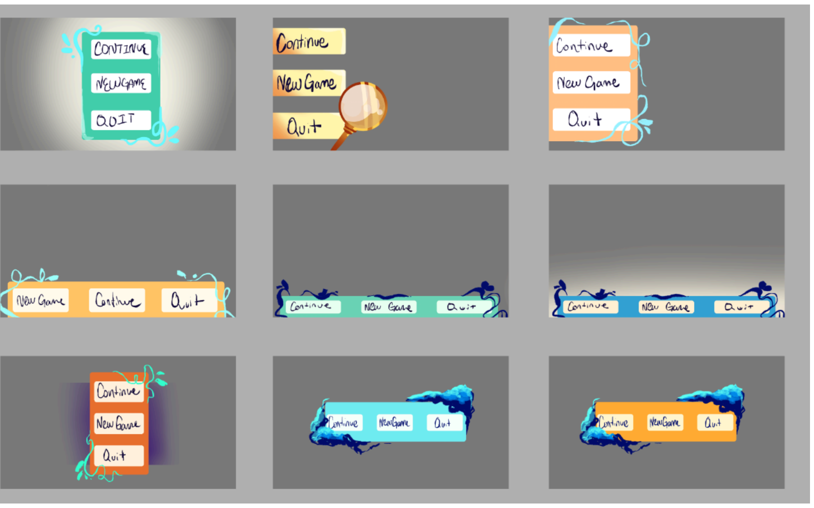


- Top three feel too generic. Too many games do this and it doesn't really show off the style of our game.
- The middle ones are nice, but it may be difficult to read the text, especially when we won't have dialogue yet for the player it feels unnecessary
- The bottom three are the best conceptually
    - 08 the green doesn't mesh well with the overall palette, and the gradient is a bit much.
    - 07 and 09 are nice as well, but the character feels overwhelming especially since we will be able to see the models behind the UI

In the end I created the ' final ' text box to be used. It takes the style in text boxes 07 - 09 and



brightens up the saturation to make it easier to read. The outline also gives it differentiation from the background of the game, that way text is easily readable. While this wasn't a concern that was brought up, I noticed the game's palette is very saturated, so for a game text box to not only stand out but be easily readable as separate, there needs to be separation in values, which the outline provides. I also adjusted where the vines go, so there was more room for text and it would not be cramped with small margins. It gives the text more space to breathe. I also got rid of the character sprite and the gradients entirely.



I also created Menu UI in a similar style to the user interaction, to try and keep consistency in designs. I did try a few other things to differentiate between text box and menu UI. However, these were unfinished as I wanted to prioritize interacting with NPC's, but I kept these as what logical next step I would take with the concept.

<u>Issues with Implementation</u>

I did try continuing and going to finalize user interaction with NPC's however I could not get the method I had in place to work. Something with the way I have interactions set up would not allow it to recognize the next step in showing the UI on a canvas. It would not recognize anything would need to happen and only show things in the Console, alongside a notification that I had multiple of some kind of selector over and over again, even though I did not have that to the best of my knowledge. I asked my friend who works in GoDot for help, but we were unable to find a solution in the way I originally intended.

To try and pivot at the last minute, I found a Dialogue Editor asset that helped you create dialogue options within the asset itself. This was very easy to set up within Unity, and used branching paths to create dialogue options that when interacted with using my collision system SHOULD have worked. I followed a tutorial by the dev as well on how to use the asset, which can be found here. However, after some trial and error and having Unity delete all my dialogue branches, I found that the asset is not compatible with the current version of Unity and has not been updated for a few years. I called the same friend to see if we can figure out a way to troubleshoot, however after tinkering with settings and the asset itself we found that it would take significant time and effort to get the asset to work with the current version of Unity.

At this point, there would not have been enough time to try to not only fix the asset, but then implement it and get it working into our version of the game. Ashley created a website for dialogue options that would be played on a separate monitor just so we would have dialogue for the sake of testing. As I also couldn't figure out issues showing the UI and making it interactable in Unity, I was simultaneously unable to create the menu since I believe whatever method I would have used for one would have been able to be used for the other.