Oliver Qiu

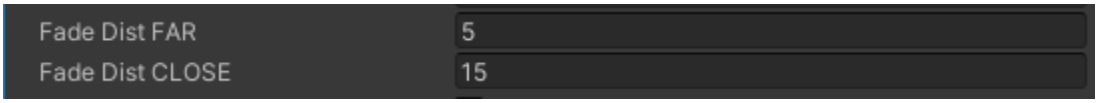
CAP3020 (Ambient Narrators) Dev Journal #1

As a great professor once said, “make everything a variable.” Well, I’m sure a good number of teachers have said this advice over my computer science years but now scripting in Unity, it rings true on more than principal.

This week I’ve been working on our inventory and interaction system. Previously, for the Alpha I had implemented a few simple scripts that would enable the player to take an item (into their inventory), as well as dynamic labels that would fade in/out text (or 2d icons later) based on the player’s distance to them.

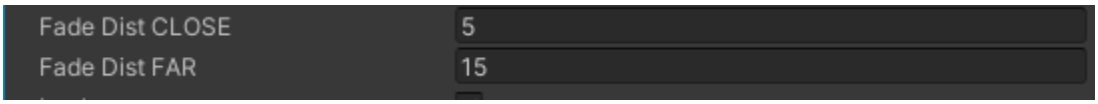
In this journal I’d like to focus on small details and refinement outside of relatively straightforward object-oriented programming and interaction. (Straightforward as in implementing these systems logic wise hasn’t raised as interesting questions.)

1. The Order of Variables



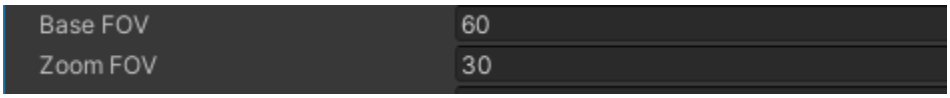
In adding variables to an item label script, I introduced variables to determine the fade distance/thresholds. I could not understand why my simple code wasn't working – turns out putting your max before your min creates unforeseen problems.

```
public float fadeDistCLOSE, fadeDistFAR;
```



Especially working in a team, simple quality of life changes like this just makes sense, but have never occurred to me before. Entering the low then high value feels natural and looks more "correct."

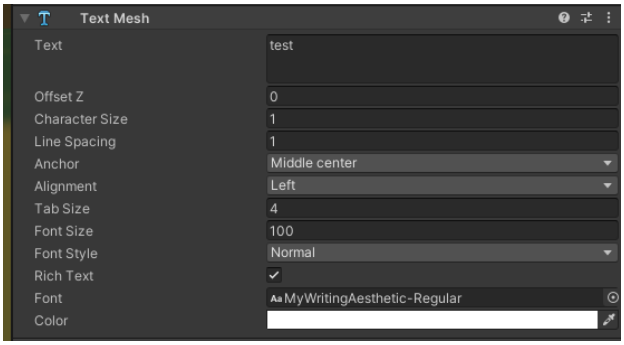
Another example could be with my right click zoom functionality:



Intuitively it makes sense to enter our default/base value before entering modifications.

2. Scripting vs. Objects

In determining whether to create/modify item text through script or the editor, I came across a lot of opinions on Reddit, Unity Answers, Stackexchange, and more. I generally understand the idea that scripting is more useful for procedurally generated or large numbers of objects while using the inspector works for more specific objects.



Procedural changes:

```
if (Dist > item.fadeDistCLOSE)
        {
            textmesh.color = new Color(0, 0, 0, 0);
        }
        if (Dist < item.fadeDistFAR && Dist > item.fadeDistCLOSE)
        {
            float temp = scale(item.fadeDistCLOSE, item.fadeDistFAR, 1, 0, Dist);
            textmesh.color = new Color(1, 1, 1, temp);
        }
        if(Dist < 2)
        {
            textmesh.color = new Color(1, 1, 1, 1);
 }
```

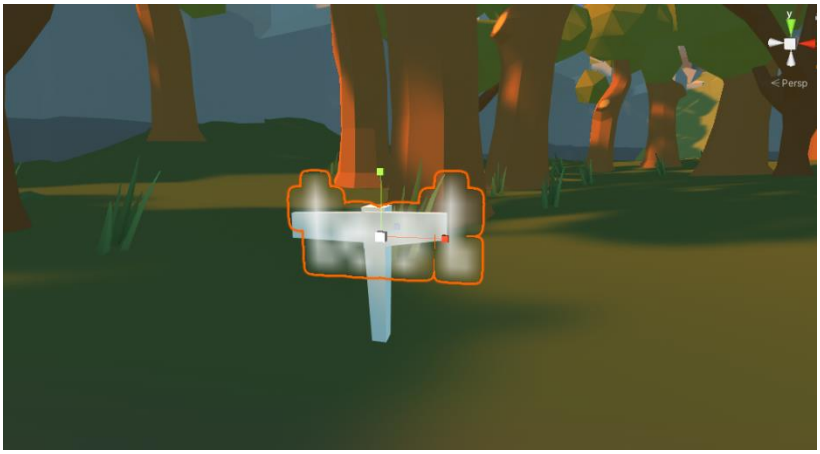
This question further breaks down into if we need to know what components/children/anything related to an object has beforehand. Without a Textmesh component here, knowing what we can/should change with scripts would be slightly trickier as it wouldn't (initially) exist in the editor.

For our game most all of our interactable objects will be hand placed and well considered, so focusing on the editor with scripts used to do things unable to be done in the editor seems good so far.

3. Editor Quirks

Dealing with Unity3d's sometimes primitive behavior leads to what would otherwise be band aid fixes but according to other users are accepted solutions.

A very specific example is when displaying text in the worldspace with Textmesh, the font doesn't automatically alias/scale.



Font Size 10 @ Scale 1

Font Size 100 @ Scale 0.1

In summary, this week creating basic systems that I might've practiced in principal (OOP) have shown me firsthand there's more to coding that isn't math or physics or some other science, but basic usability and common sense. I've built my systems so artists can drop in UI or 3d assets and can't wait to keep on developing!