Celine de Wijs
3013898

PLATFORMER EDITOR TOOL

DOCUMENTATION

What is the Platformer Editor Tool?
The Platformer Editor Tool is a tool for quickly creating and editing platformer levels. It's based on drawing pixel art maps of your level and then mapping each color to a GameObject. The tool will detect each color in your texture and shows this in a list, from where you can select prefabs. After clicking on "Generate Level" the prefabs will be placed on the exact same spot as their matching colors. The tool is mainly for 2D editing.

The tool is made for Unity.

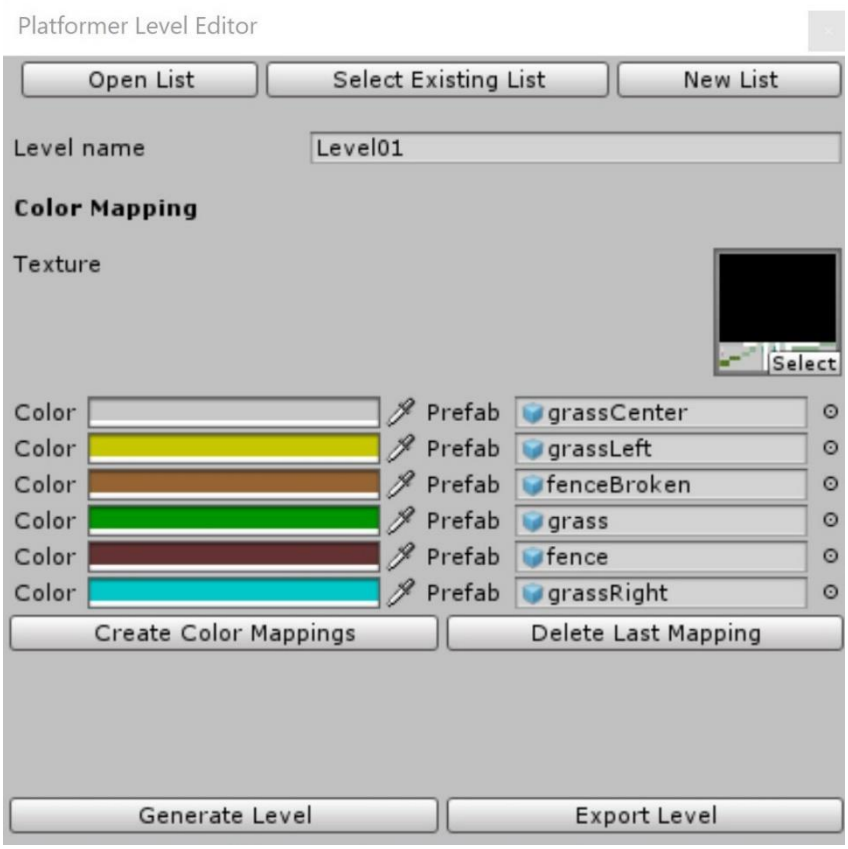
How it works
For editing levels, I created an editor window. In this window you can create a "Level Data" list. This is a ScriptableObject which contains: the name of your level, the texture of your level, the ColorMappings and it can also store a prefab of your generated tiles.

Once you created a new list, or opened an already existing one, you can edit the name and the texture of your current level. Once you select a texture, you can create your ColourMappings. The ColorMappings consist of a color and a GameObject. These colors are detected with nested for-loops (x and y). In this for loop, the tool iterates over every pixel in the texture and saves every unique color to a list. For every color in the list, a ColorMapping is created with this color. If you need to, you can remove ColorMappings by clicking on "Delete Last Mapping". Only the last ColorMapping will be removed.

After mapping the colours to prefabs, you can generate your level. The tool will instantiate each prefab in editor mode in your scene. It will place these gameobjects as children of (levelName + " Generated Tiles") in the Hierarchy.

You can choose to export your level, which will create a prefab from the Generated Tiles object and places it in your LevelData ScriptableObject.

Celine de Wijs
3013898



Screenshot of the Platformer Editor Tool

Process

At first the idea was to include a pixel editor in the tool, but I realized this was too ambitious for the time I had available. The texture that I used in my demo was thus created in Photoshop.

I chose to use ScriptableObject for my data, because this was the obvious choice for what I needed. The data that is created in this tool can only be used in the tool itself and it's only usable in Unity. It also contains prefabs that need to exist in your project to use them. That's why I didn't choose for other method's like XML and JSON for example.

In the first version of the tool, the colors weren't filled in for you yet, so you had to find the exact same colors as in your texture yourself. This activity was very annoying and time consuming, so I made the tool smarter and let it fill in the colors for you.

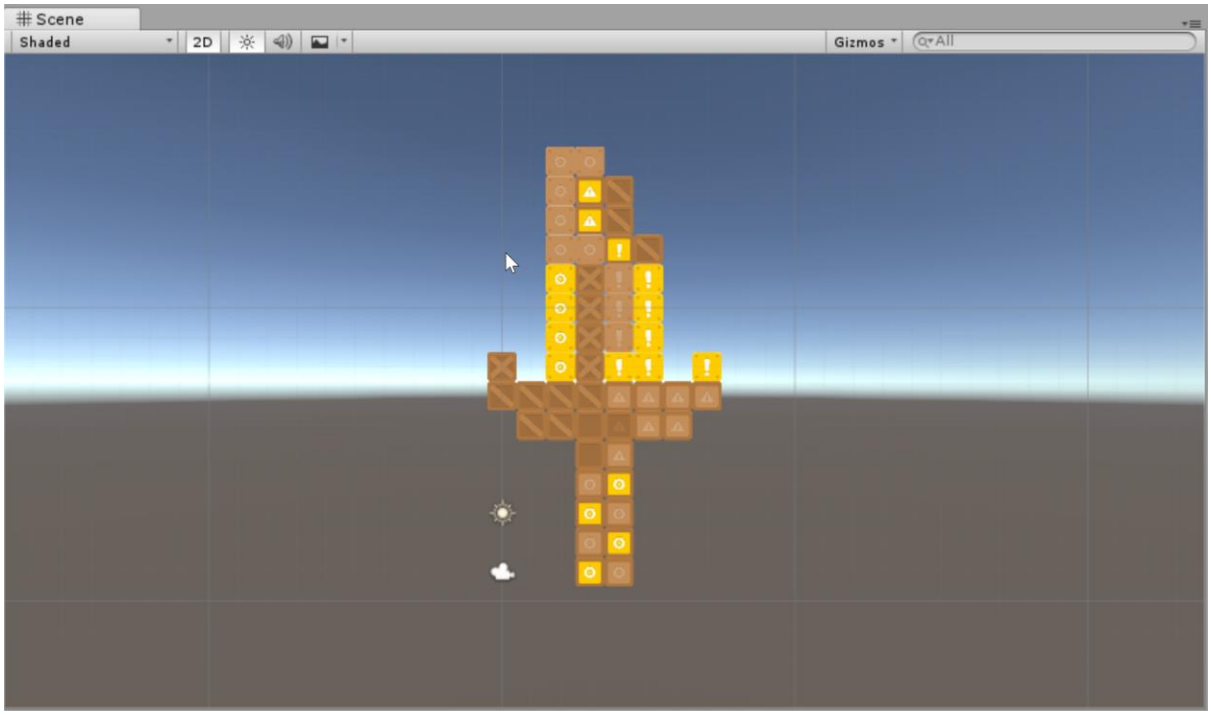
I also want to say something about my script formatting. In the beginning of the year we learned some conventions, we had to use, but I think my scripts are way more readable if I place my brackets on the next line. I've used this consistenly.

Buddy System

For the buddy system assignment I worked with Ronald van Egdom. He made a Weapon Generator Tool, which creates a texture. I used this texture in my own tool.

The first problem I ran into was that I got a very, very long list of colors. This was because there were a lot of colors in the texture with different alpha's. I fixed this by changing the threshold of the alpha to > 0.9. When I tried to get the ColorMappings again, I got a list I could work with.

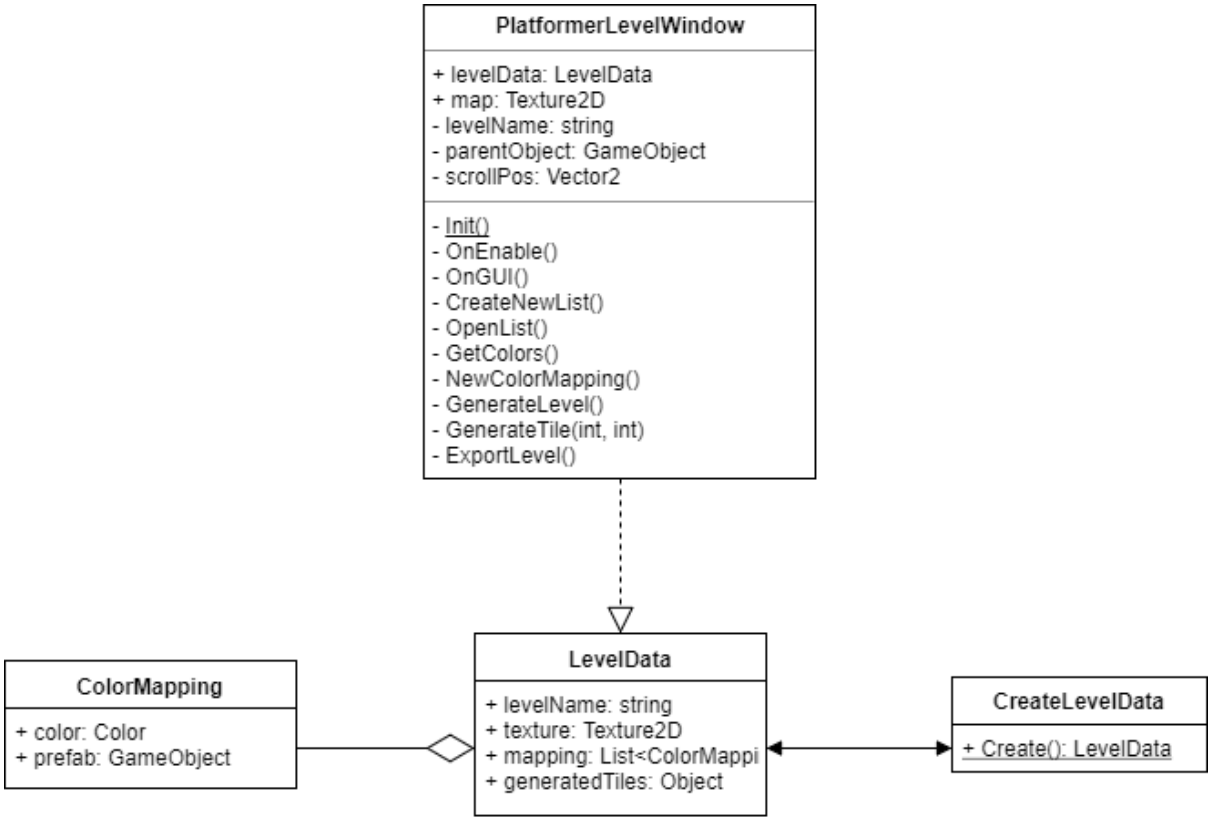
The second problem was that these colors were in illogical places for a simple platformer level. I didn't know what else to use than a bunch of boxes, but it worked.



The result of throwing the texture from Ronald's Weapon Generator into my Platformer Editor.

Celine de Wijs
3013898

UML Diagram



Celine de Wijs
3013898

Activity Diagram