Introduction

For my final project I created a small forest scene in a 2.5D style (2D images within a 3D space). This style has piqued my interest for a few months now, but I was always too intimidated to attempt it myself. Mainly because I am unfamiliar with working in 3d. With all the practice I received this semester working with new coding styles, I decided now would be the time to finally attempt this style.

I designed a small and confined space within a forest where the user can move around a fairy. The scene also contains a river and a bridge. There are deer that react whenever the user moves within range of them. The scene includes forest and river background sounds.

Environment

For this project I used the GameMaker engine. I chose it because it's an IDE I am comfortable using and felt would support this larger scope project. GameMaker uses a language called GML, which is nearly identical to languages such as Java or C++. My skills in p5.js also translated well into this environment because the bulk of coding in GML takes place in a create event and a draw event. Similar to how p5.js uses a setup and draw function.

I also used this engine because I am hoping I can use the code I developed for hobby projects in the future.

I would say most of the project was created via code. Game engines provide many ways to circumvent using code, that was not the case for this project. I prefer coding everything myself because it gives me finer refinement and is less of a black box.

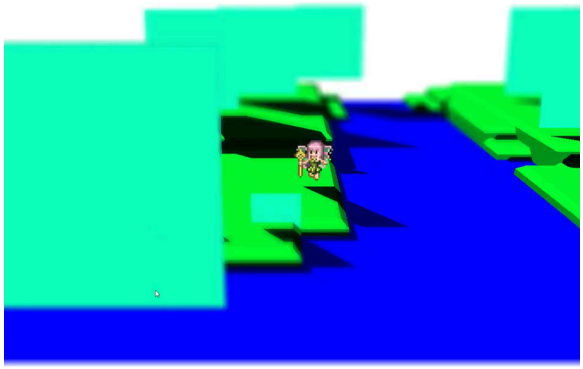
Finally, I used a bit of AI in this project to aid me with more simple tasks. Such as what number I need to calculate to match two different rotations, how I would increase contrast via a shader, how I could repeat one part of code within a for loop, etc. The rest of the code I all either wrote myself, followed from tutorials, or found on online resources such as GitHub or Stack Overflow.

The folders containing the bulk of code are the obj_camera object, obj_player, obj_tree, and all shaders. There are various scripts but I found most of these online. Everything else is more or less a bunch of meta data. But please feel free to explore.

Setting up 3D

The first step of this project was to set up a 3D world. This engine is primarily used for 2D but still provides functionality for 3D. For this part of the project I had to give myself a crash course on how 3D objects are generated onto a screen. It took about two days to get some simple 3D objects up and running onto the screen. Luckily, there were various online resources and documentation to aid me. The time spent practicing with some more low level coding topics was valuable and grew my coding skills. And by understanding the basics (there is still A LOT I don't know or understand), I was able to more easily import 3D objects later on in the project.

I also want to note that the 3D assignment we had also aided me a lot in this step of the process. It acted as a nice stepping stone between no knowledge and the more in depth stuff I was learning.



Initial 3D scene without any textures

All objects in this world can technically be considered 3D objects. The terrain and bridge are obvious. But all the 2D sprites are actually 3D objects. The sprites are textures I draw onto a vertical standing wall. This is notable when viewing the video of week 1. At that point in time I had not added any textures, and various standing walls can be seen. This method is used for all trees, bushes, flowers, and the player. Initially I had some difficulty directly displaying the transparency from the textures. I was able to fix all issues by discarding any transparent pixels via a shader.

The water also uses a similar technique as the sprites. It is a flat plane with a repeating texture. The movement comes from a shader trick that utilizes random noise.

All the terrain consists of simple 3D cubes with diagonal edges. I coded these by hand to gain familiarity with how vertexes work. All cubes also only have 4 of their 6 faces. As

the camera is always looking forward, the back and bottom of the cubes are never seen. Discarding these faces adds a bit of optimization when the code is running.

Lighting, and Shadows

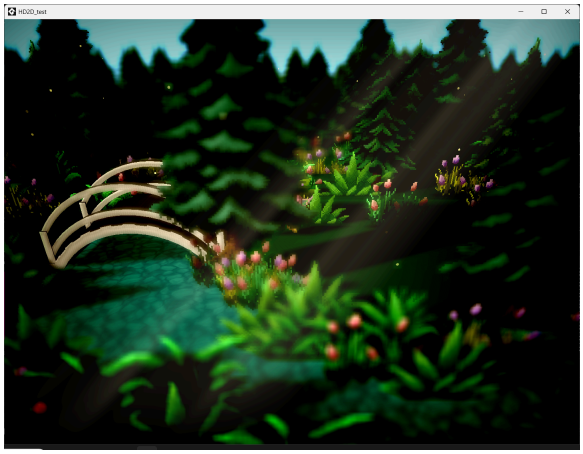
The hardest aspect of this project was setting up lighting and shadows. I had to watch many video tutorials and went through loads of trial and error until I got lighting and shadows properly working. I'm still not a pro but I'll do my best to describe the method used.

I set up a camera in place of where the sun would be. The direction the camera is pointing represents the direction the light is shining. I then create a surface that draws the scene from this camera's perspective. I put the surface into a shader that does some math to calculate the distance between the camera and each object, creating a depth map. (A surface in GameMaker is essentially a screenshot storing whatever is drawn on screen. You draw onto the surface and then draw the surface onto the screen.)

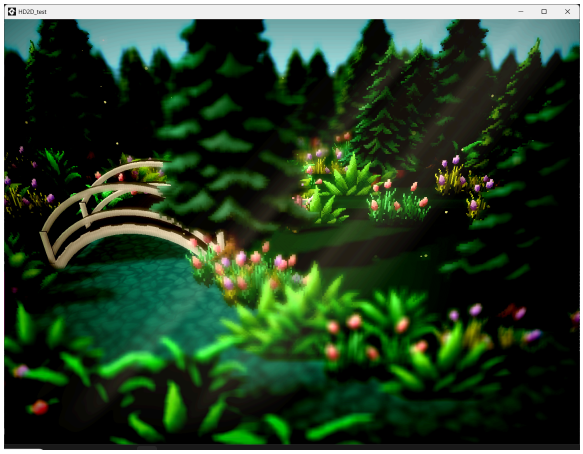
Once the depth is calculated I then draw everything again on another surface. This time using a camera from whatever perspective I want the final screen to be using. I pass the depth map into a shader alongside this new surface. The shader does some more math to calculate all shadow locations, and wherever there's a shadow I decreased the rgb values to appear dark.

There is a lot of math involved that I don't fully understand, but in the end it works and looks pretty! While the system is not perfect, for my ability and familiarity, I'm proud of the system I was able to achieve.

I revisited these shadows near the end of the project as I noticed they were all rectangles. Recall how the 2D sprites were a texture on a vertical, rectangular wall. This was the culprit. I edited my shader that creates the depth map such that it discards information wherever a pixel was transparent. This made all shadows the proper shapes. I also noticed it increased the framerate a bit.



Old shadows



New shadows

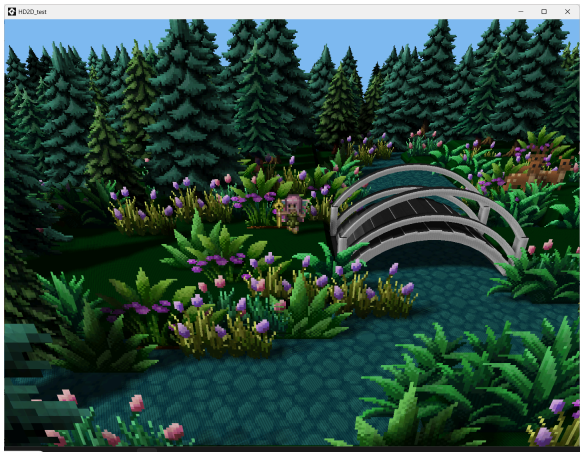
Post Processing

The remaining time of this project was spent adding many different post processing effects. I watched one video that outlined all the different effects one needed to add to the 2.5D style to achieve a striking visual identity. I used this list as a checklist of what effects to add.

This step was easier to implement than the previous two as my code was designed in a way that could easily implement new effects and shaders. Most effects are in a fragment shader called shd_postProcessing. However some effects needed further shaders to achieve their desired outcome. Each shader was either something I coded myself, found online, or made in a previous class assignment.

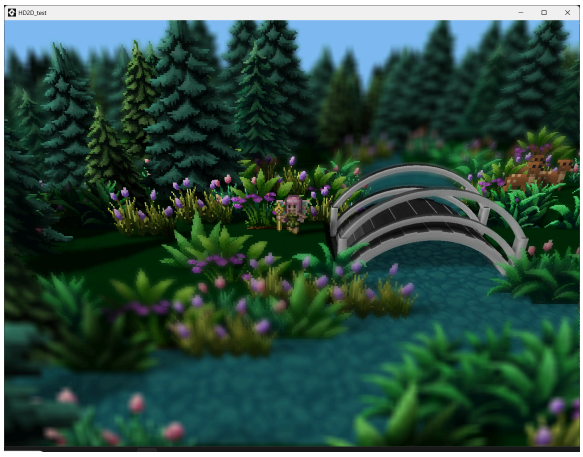
The hardest effect, and most time consuming, to implement was Depth of Field. As such, it was one of the first one I implemented. Luckily, there were various video tutorials to aid me.

The bloom shader was interesting to revisit as I found it quite easy to implement compared to when I made one for the 3D assignment. I did the hard work then and was able to quickly translate it to this project.

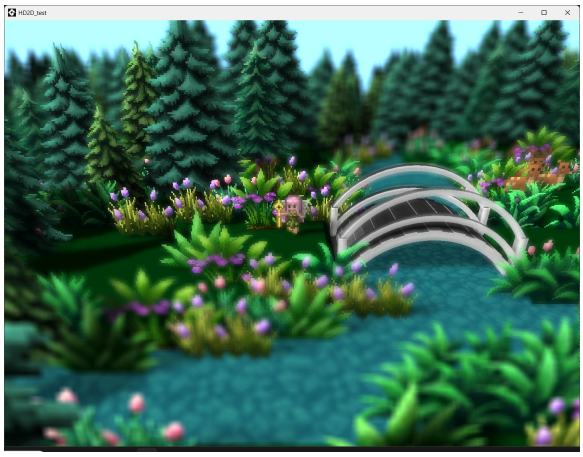


No effects

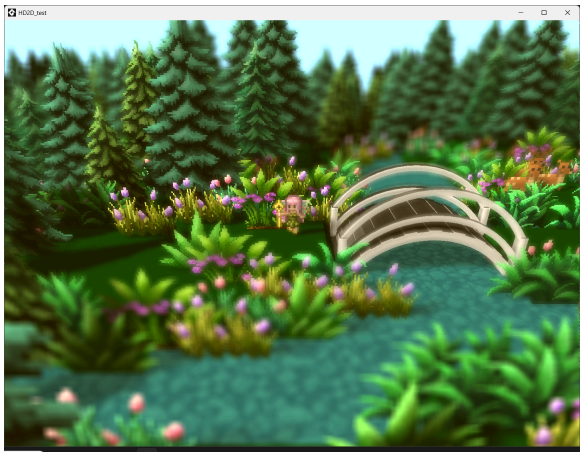
Moving water



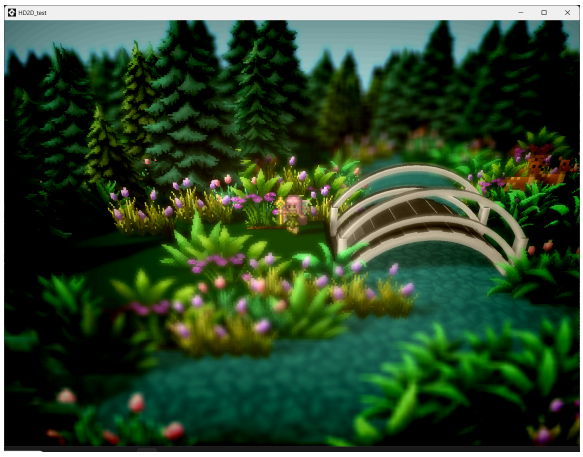
Depth of field



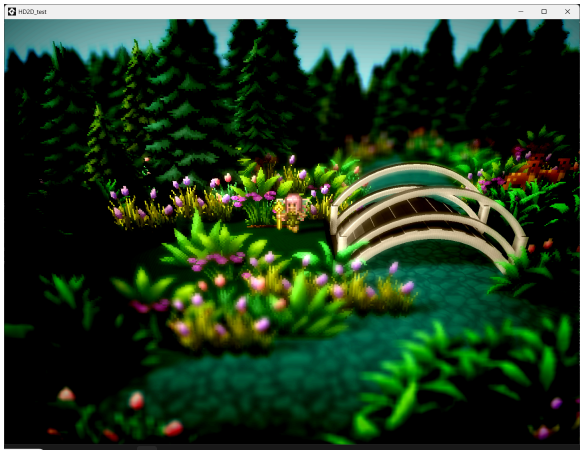
Bloom



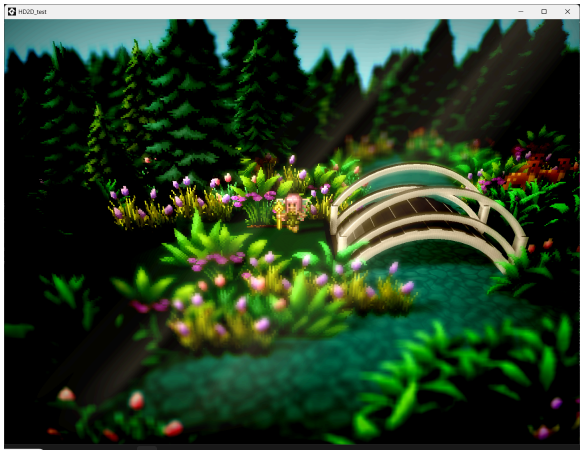
Color grading



Vignette



Contrast



Light rays

The amount of pop all of these effects added greatly enhances the entire scene despite about half of them being few lines code. I think the contrast is a good example of what I mean. It is just a single line of code but hugely enhances the visuals of the scene.

The last thing I added was a very simple particle effect. They are made just like any other 2D sprite in this scene. I generate a very small dot with a random size, speed, and direction. It’s another small addition that adds great visual interest to the scene. There is some more fiddling I could probably do with these particles but they are good where at.

Particles

Conclusion

I'm super proud of how this project turned out and I'm super impressed that it's something I actually made. I wanted to try this technique months back but within a few minutes got intimidated and abandoned the idea. This class really pushed my creative coding skills. I develop games as a hobby so I've done some creative code in that capacity. However I often stuck to what I was familiar with. The structure of this class allowed me to experiment with and practice so many new techniques I would have never dared try. In turn making me a stronger programmer. And techniques that were once too intimidating became something I felt comfortable diving head first into.

I am hoping to utilize what I made in future projects. During development, I put a lot of effort into making sure the code was clean, readable, and scalable so that I can use it beyond just this small scene.

Videos

Video final:
https://drive.google.com/file/d/1-pLeA3GSGyKlzdVR6K7JxZwuHQCyFNtS/view?usp=sharing

Video week 2:
https://drive.google.com/file/d/132ddEMuopnsDrNGS3j3R9upW0WzugyAJ/view?usp=sharing

Video week 1:
https://drive.google.com/file/d/15TaiStUJfceNYtZCc51U6uIk9-BG3bvn/view?usp=sharing

Assets

2D Sprites
Player: Me
Trees: https://karsiori.itch.io/spruce-tree-pack-pixel-art-animated
Bushes: https://karsiori.itch.io/free-pixel-art-bush-pack
Flowers: https://karsiori.itch.io/free-pixel-art-flower-pack
Deer: https://lyaseek.itch.io/miniffanimals

3D Objects
Bridge: https://sketchfab.com/3d-models/bridge-7d3d65090eb945e2996c23ac98adec70

Sounds
River: https://freesound.org/people/Akacie/sounds/73716/
Forest: https://freesound.org/people/klankbeeld/sounds/577488/