Lab Week 3

50.033 Game Design and Development

1004147 --- Riley Riemann Chin

Are you participating in the Weekly Lab competition? ~~Yes~~/No

Provide the YouTube/other platform link to your screen recording:
https://youtu.be/15W8SjMf1UU

Provide the link to your lab repository:
https://github.com/rileychin/50.033-Game-Dev-Labs/tree/lab3

Describe what you have done to achieve the desired checkoff requirement for this lab:
[Your **high level** description here]

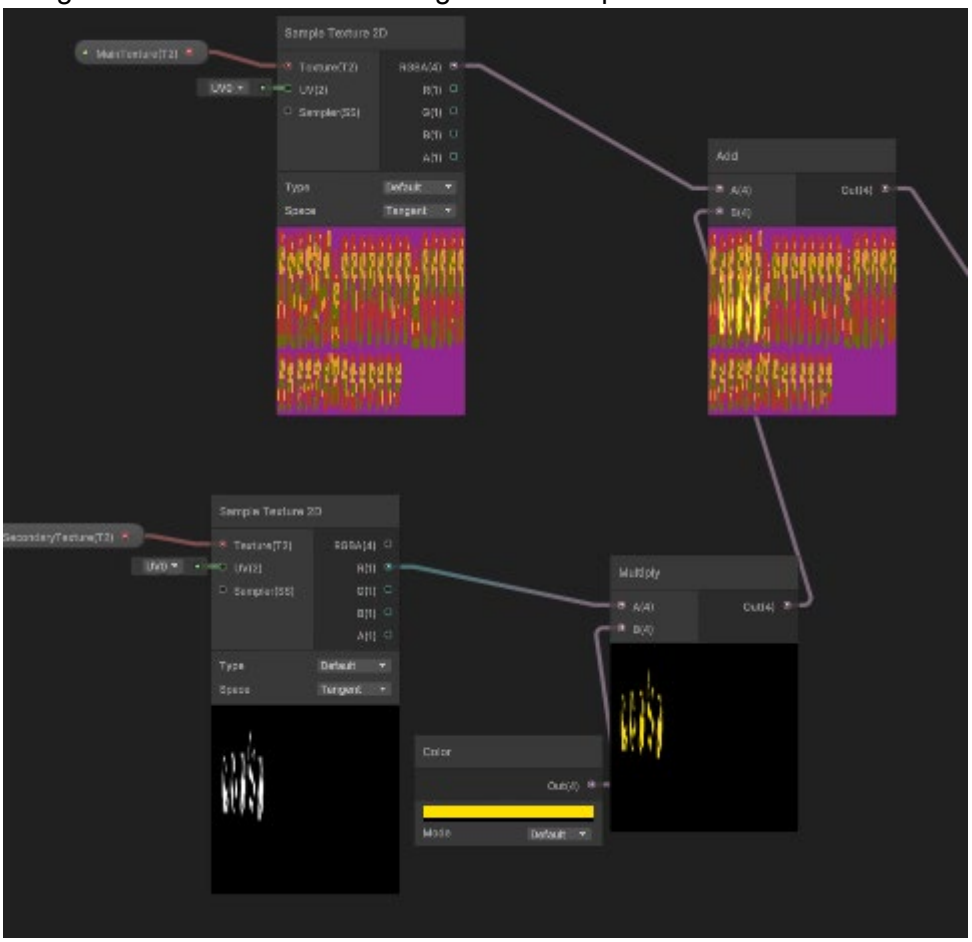
- You don't need to be too specific. This is just to assist us when we check your repository
- Scripts added, Assets added if any
- General modifications that you have done: eg animating the enemies, implementing FSM for the NPCs, etc.

- Breakable bricks (you are free to define what breakable means, that is if the debris has collisions with the ground, tiny animations, etc).
    - To make breakable bricks I followed the tutorial provided in the handout. I first had to create the necessary gameobject with a brick sprite with sorting layer and layer set to “Obstacles”, and a child gameobject with an edgecollider component attached with the BreakBrick.cs script.

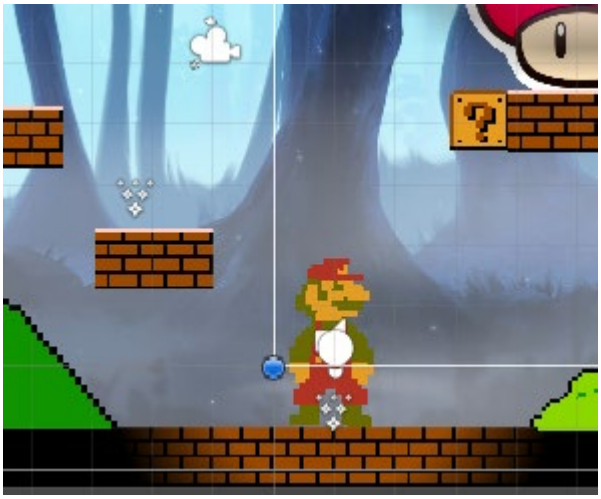
```
void  OnTriggerEnter2D(Collider2D col)
{
    if (col.gameObject.CompareTag("Player") &&  !broken){
        broken  =  true;
        // assume we have 5 debris per box
        for (int x =  0; x<5; x++){
            Instantiate(prefab, transform.position, Quaternion.identity);
        }
        gameObject.transform.parent.GetComponent<SpriteRenderer>().enabled  =  false;
        gameObject.transform.parent.GetComponent<BoxCollider2D>().enabled  =  false;
        GetComponent<EdgeCollider2D>().enabled  =  false;
    }
}
}
```

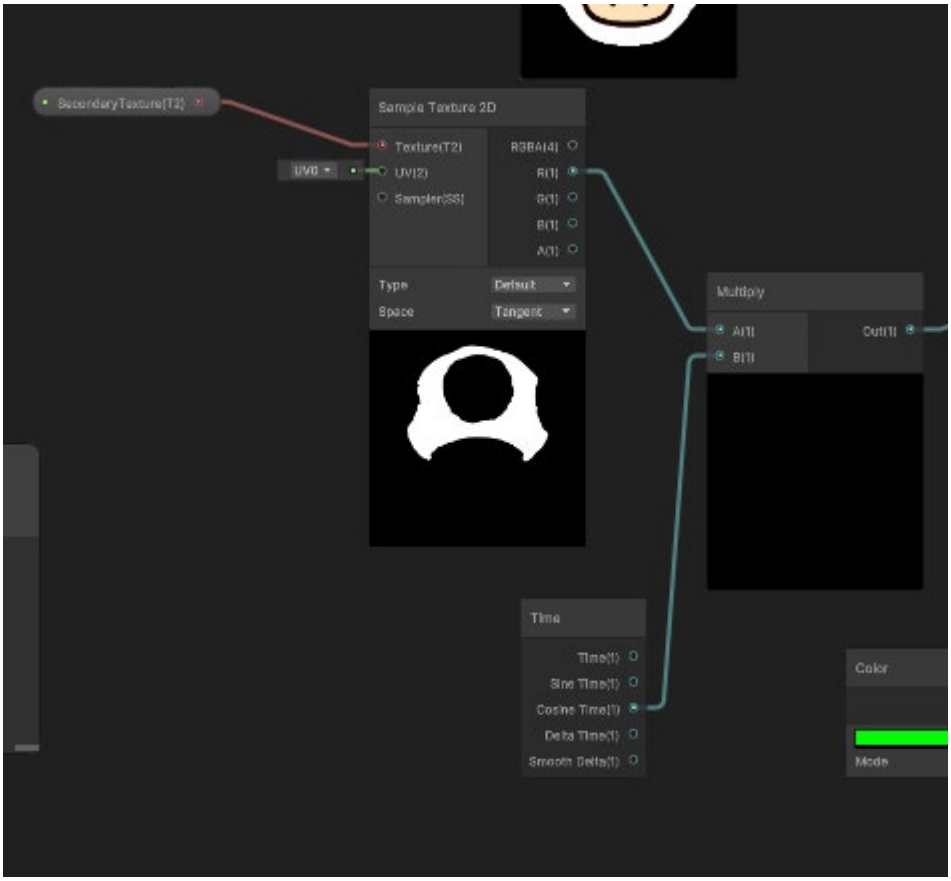
    - Here is the trigger event for what happens when the player touches the edge collider. Essentially upon touching it, it will spawn the debris prefab which will just shrink the box down to nothing.

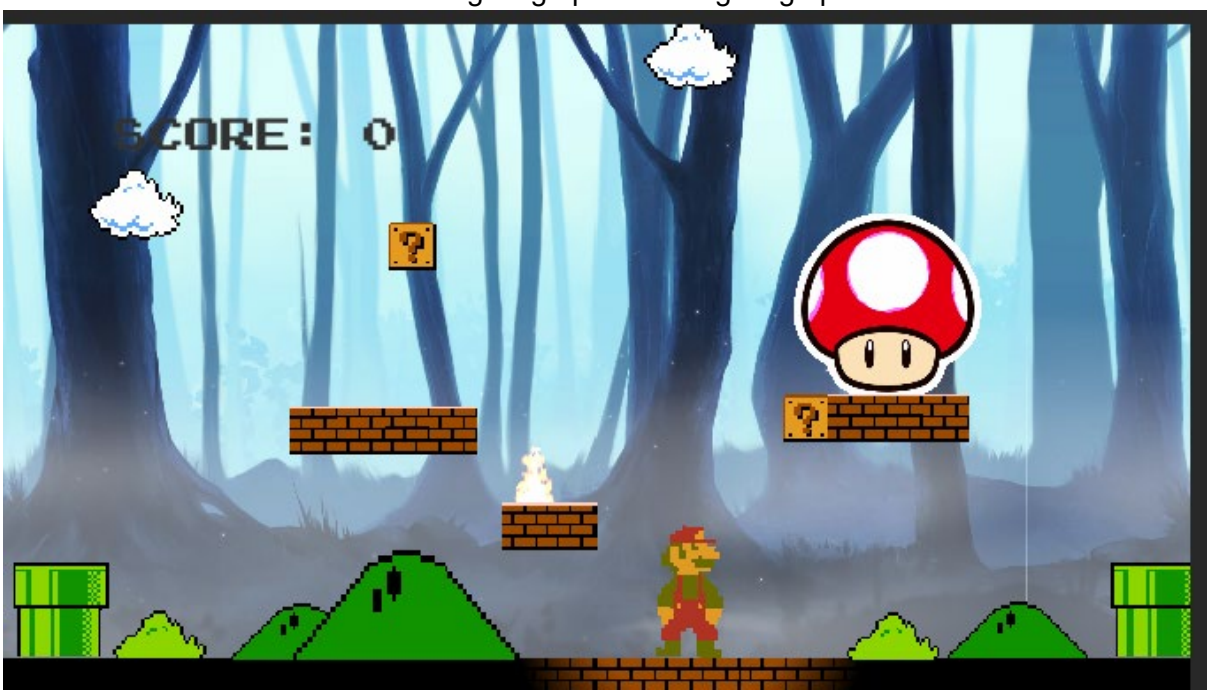
- The Debris.cs script is then attached to the Debris prefab which does the box scaling down to nothing and destroys the original game object sprite renderers and box colliders.

- Glowing mario shirt when he is running, or jumping, or doing anything.
  - Similar to glowgraph for the castle game, I included the same shader graph design to mario's shirt and the original mario sprites.
  - 
  - The thing here is that the silhouette of mario's shirt when he is running is given on a white background so I had to change it to black so that the sampletexture 2d was able to see it properly
  - This was done using some online tool that managed to change the background.
  - After that I just added a spotlight2d gameobject to mario player
  - 

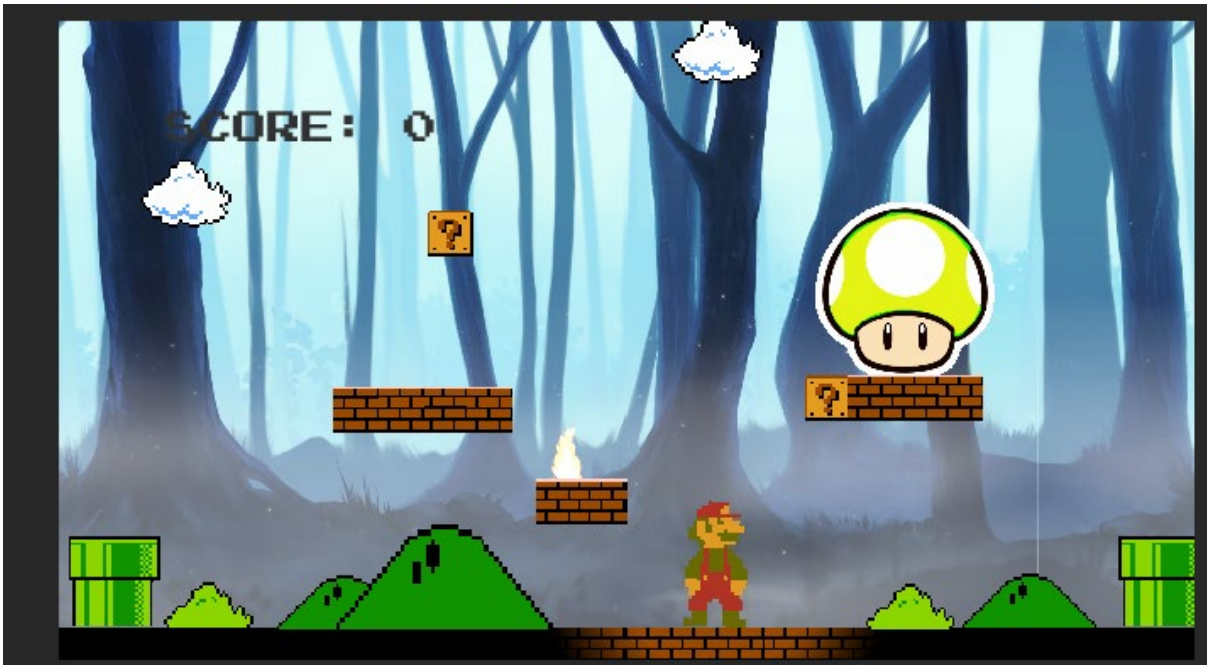
  - And then changed the material of the ground to lit so that it had that "glowy" feel when he runs or walks

- 
- Use particle system to create any interesting object, you must utilise textured sheet animation.
  - 
  - This was done using just a simple particle system to create the fire, the fire texture was already split into an 8x8 grid so all I had to do was set the settings for the particle systems component.
- Create your own simple graph shader for anything in the scene that changes at runtime (hence you need to use the time node).
  - Here I used a simple mushroom sprite taken online, then I traced the portions of it where it was red in color and then set that as white with a black background as the secondary texture

- 

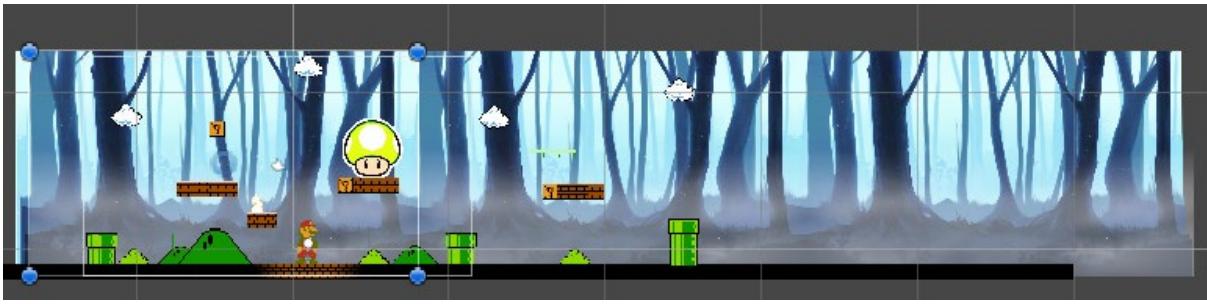
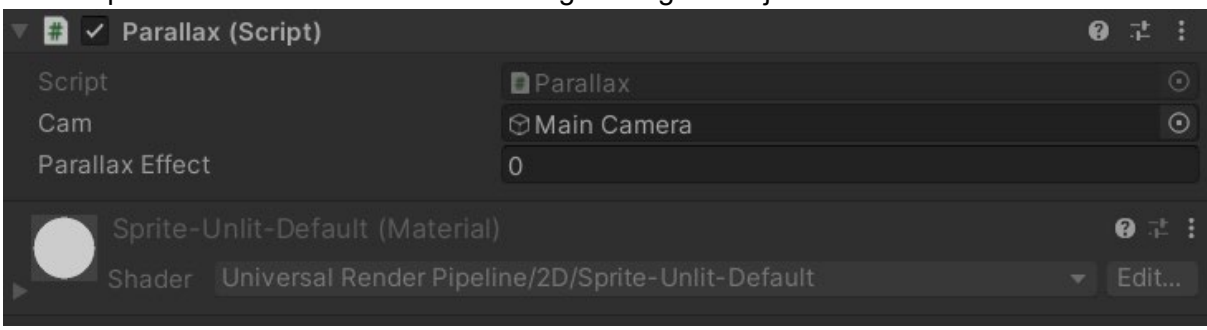
- The time node then was implemented using along with the multiply node so that the shader oscillates between lighting up and not lighting up.
- 

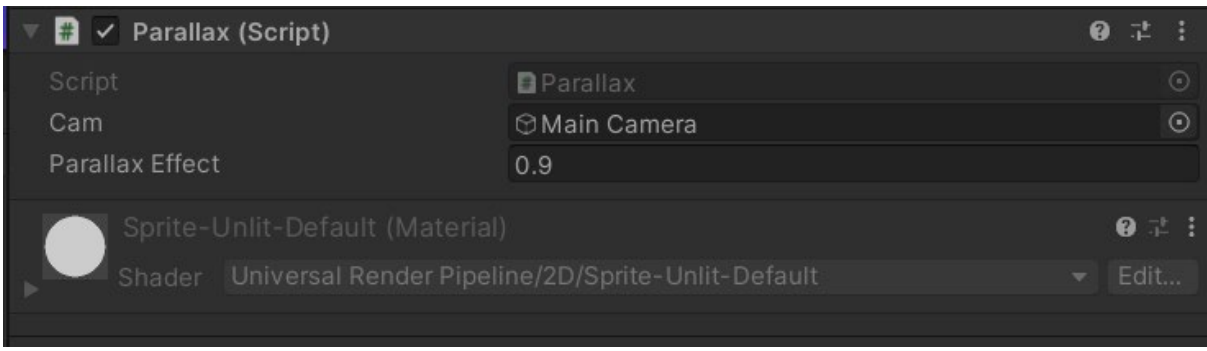
- At time t=0

- 

  - At time t=1
  - The color will switch during run time
- Implement parallax background effect
  - The parallax background tutorial in the handout didn't work for me because I could not apply more than 2 base cameras in the scene. So instead I followed another tutorial from: https://www.youtube.com/watch?v=zit45k6CUMk&ab_channel=Dani
  - So what I did was to first create the sprites for the background with sprite renderer on an unlit default material
  - 

  - I did this for all layers and just stacked multiple layers sequentially in the scene

- 
- After that I created a script to control the moving background called Parallax.cs
- The main thing is to shift the background according to a certain speed value based on the order of the layer. So the front mist layer will move along with the camera, while the background layers will move more slowly.
- The script is then attached to ALL the background gameobjects
- 

- This is for BG1
- 

- And this is for BG10.
- Overall it creates a nice smooth effect of depth in the background.