Development Blog

April 8,
Project Title: Windows XPerience
Team member: Junior, Ju Hee, Adham
For this project, we will go back in time to revisit a staple piece of technology: a Windows XP computer with the classic green valley and blue sky background. Our inspiration came after a rigorous brainstorming session were we touched base on a lot of ideas that interested us. After presenting some of the ideas to the class and after further deliberation amongst our group, we decided that this idea provided the best balance between abstract storytelling and realistic implementation considering the timeframe and themes available. The theme our piece will be based on is: “close the door” and we hope to do so in an interesting way, giving the user the possibility to decide the ending of a story where, inevitably, the door will be closed.
As seen in the Scene 1, the user will be in a room with his old-style windows computer appearing in front of him/her. As he/her clicks on it, he will be sucked inside the computer, shadowing the idea presented by Richard Moore in the Disney Animated film Wreck it Ralph 2. In order to escape the computer, the user will have to click on a set of icons in a predefined sequence. Each icon will transport the user into a different scene, and it is up to the user to click on the right set of icons to escape.
Possible Assets:

- Windows XP Icons
- Furniture for the room
- Laptop
- Paint brushes (Paint Program)

Interactions to Design and Code:

- Clicking on the icons
- Scene Manager to move from one scene to another

Sound:

- God-like voice that tells the user the situation he is trapped in and gives him clues to possibly escape the computer
- Sound effects for clicking on different icons
- Sound effect of getting absorbed into the monitor
- Peaceful background music when the user is in the green valley

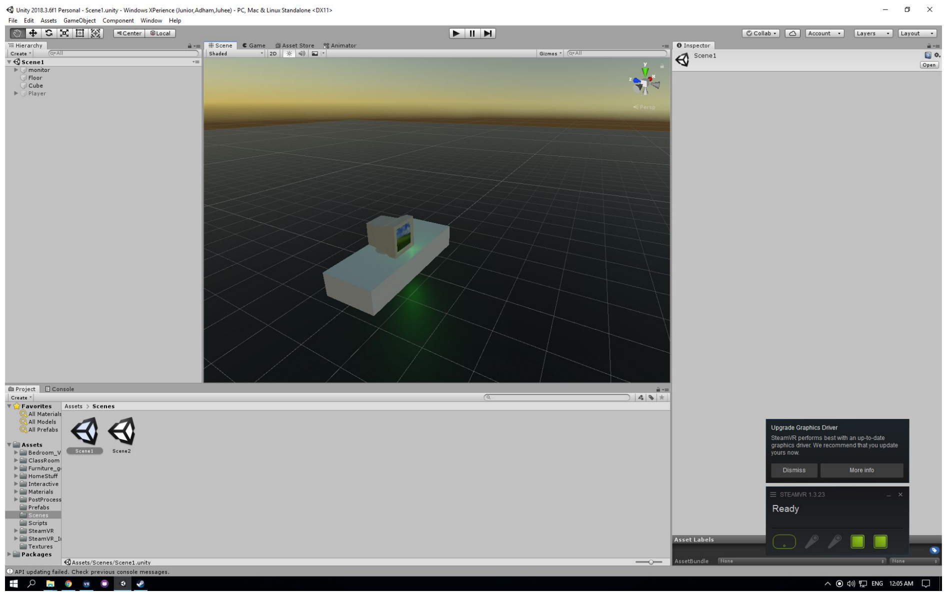
Lighting:

- Stark contrast in lighting between the room and the inside of the computer.

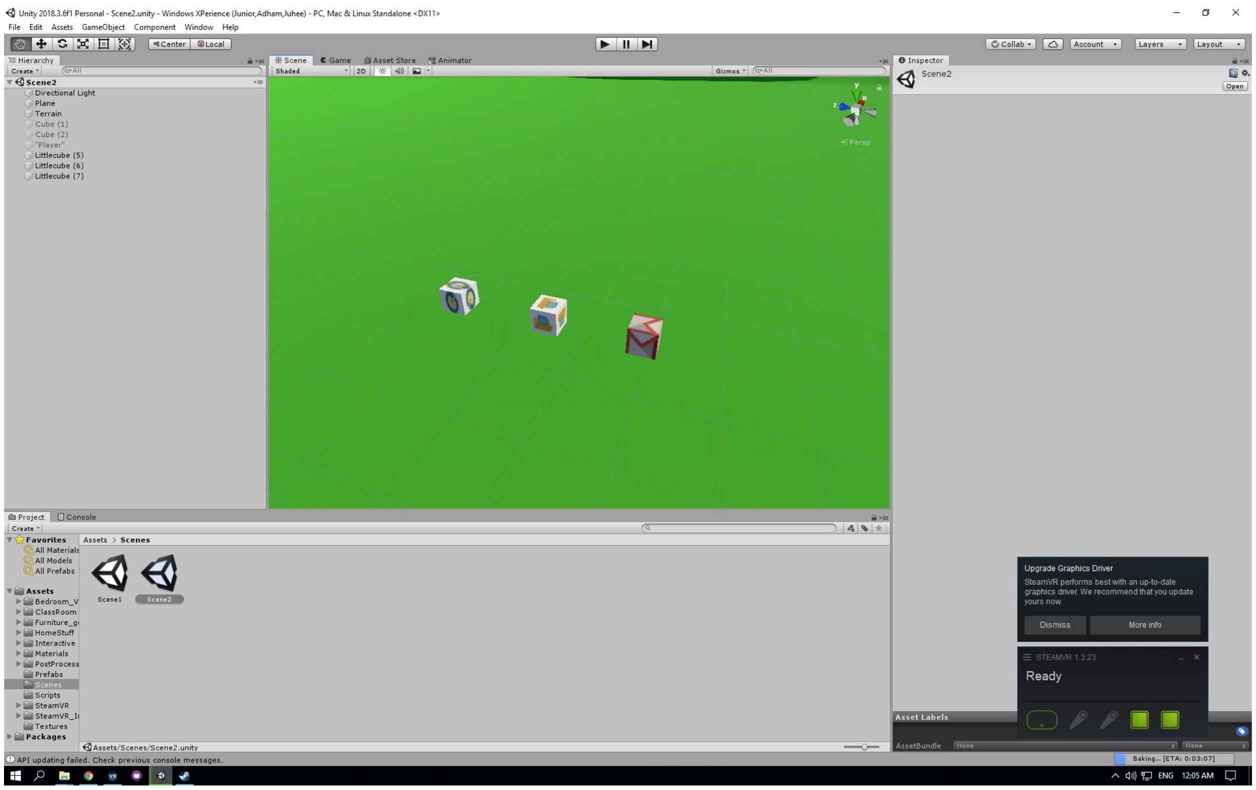
April 12
In preparation to this project, I decided to start looking into different logistical things that need to be figured out later on but will be beneficial to do so early in order to maximize workflow efficiency amongst all of the team members. First off, I thought it would be a good idea to think of ways we could use github in order to work simultaneously on different portions of the project. As such, I created a github repository to ease collaboration:
https://github.com/jgarcia1599/Windows-XPerience
Link I used to setup the github repository:
https://www.studica.com/blog/how-to-setup-github-with-unity-step-by-step-instructions

Something else I decided to start looking into was the different options we could look into when designing the User Interface of our project. The UI can be manifested in different ways, but one of the clearer options is to add a beginning and end screen to add a conclusive narrative to our piece.
Useful links:
https://blog.teamtreehouse.com/make-loading-screen-unity

April 14
In preparation to the rough prototype deadline tomorrow, we decided to build a rough version of our two initial scenes in order to show the class the direction our project might go into.



We started with a blank scene, and we added a monitor we got from the asset store. One of the problems the monitor was giving us was that its screen had a predefined animation sequence. We had to remove that and change the canvas of the screen to the Windows XP background.



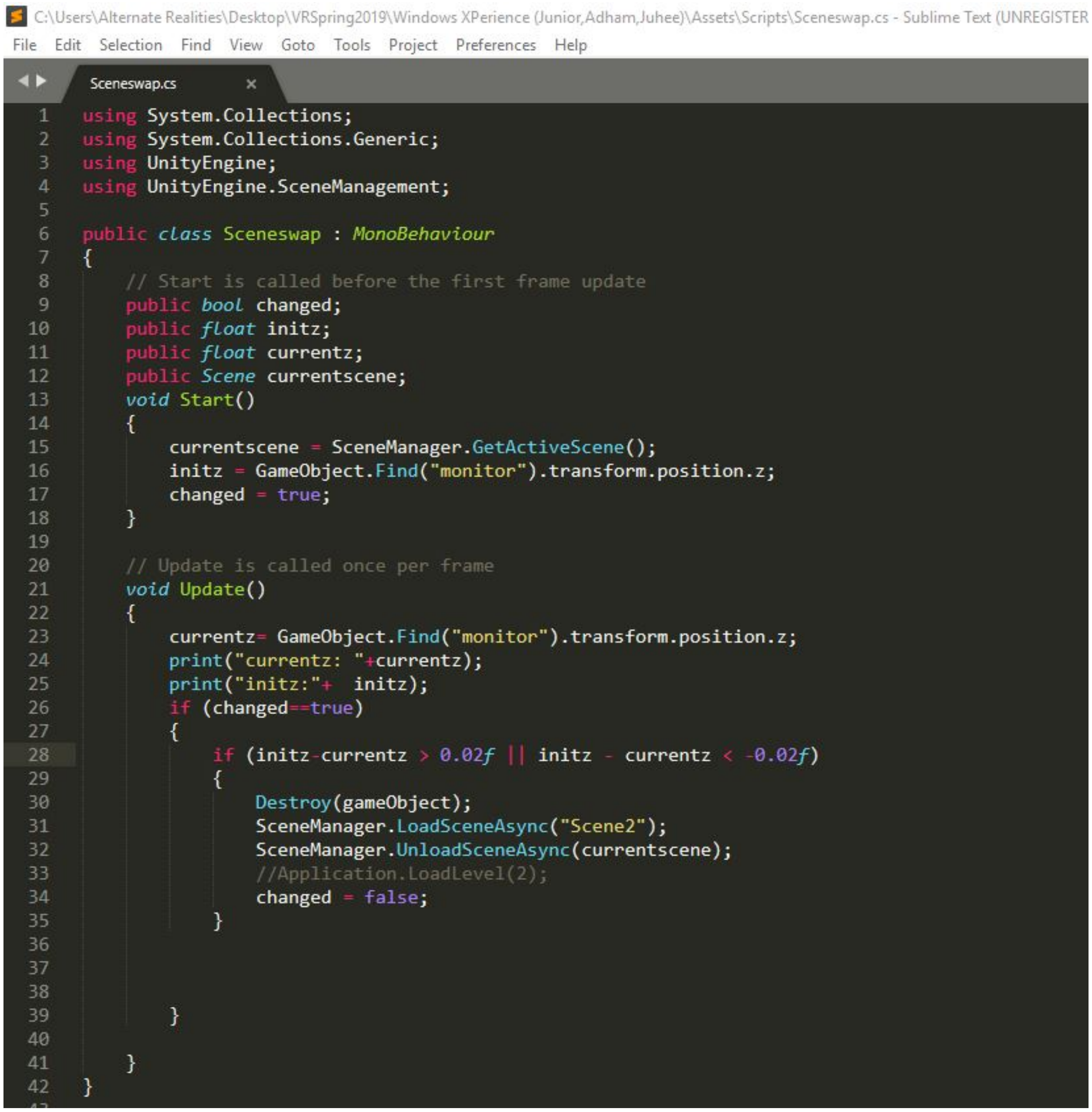
Upgrade Graphics Driver
SteamVR performs best with an up-to-date graphics driver. We recommend that you update yours now.

API updating failed. Check previous console messages.

Baking... [ETA: 0:03:07]

We also created the second scene, which was composed of a terrain object we modified by changing its color to green and by playing with the edit height/edit texture tools to create mountains (really bad ones as it is a prototype).

We also added the interactable script (and all of its dependent scripts) from the SteamVR library to the monitor and the icons. We also added a sceneswap script to the monitor. This script checks a change in the transform.position.z (any coordinate variable could be used to make this logic work) of the monitor and unloads the current scene and loads the new scene whenever this change occurs.


C:\Users\Alternate Realities\Desktop\VRSpring2019\Windows XPerience (Junior,Adham,Juhee)\Assets\Scripts\Sceneswap.cs - Sublime Text (UNREGISTER
File Edit Selection Find View Goto Tools Project Preferences Help
Sceneswap.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class Sceneswap : MonoBehaviour
{
    // Start is called before the first frame update
    public bool changed;
    public float initz;
    public float currentz;
    public Scene currentscene;
    void Start()
    {
        currentscene = SceneManager.GetActiveScene();
        initz = GameObject.Find("monitor").transform.position.z;
        changed = true;
    }

    // Update is called once per frame
    void Update()
    {
        currentz= GameObject.Find("monitor").transform.position.z;
        print("currentz: "+currentz);
        print("initz:"+  initz);
        if (changed==true)
        {
            if (initz-currentz > 0.02f || initz - currentz < -0.02f)
            {
                Destroy(gameObject);
                SceneManager.LoadSceneAsync("Scene2");
                SceneManager.UnloadSceneAsync(currentscene);
                //Application.LoadLevel(2);
                changed = false;
            }



        }

    }
}
```

April 15
Playtest Session #1
In class, we were able to get meaningful feedback from the class from our Playtest session. One of the problems our project has is that it can go into many different directions. Since there is a lot of staple icons we could recreate in our project, each of them would require an ample amount of work to the point where each one could be a project in it of its own. For example, recreating a Paint icon that would allow users to create any form of Paint-generated artwork could be a project that exploits the artistic capabilities of Paint in a 3D virtual-reality environment. As such, we heavily rely on Ideation and Feedback sessions like this one in order to narrow the focus of our project.

This is some of the feedback we received in class after playtesting our prototype:

- Keep interactions and everything in one scene
- Virus takes over the computer. You need to defeat the virus
- Maybe after you defeat the virus you become king

Generally, people would like to fight against some of the iconic enemies you can find in a computer (a.k.a a virus). Maybe giving the user a predefined task like this one could narrow the focus of our project as it is really open-ended as of right now.

April 17
Having Sarah Rotberg in class was definitely useful as it brought to our attention something that we did not previously considered.She recommends her students in her VR class that their projects should keep the users entertained for a minimum of three minutes. Even though we didn't have time in class to present our project idea to her, this feedback is useful as we are looking ways of narrowing the focus of our project. Maybe giving the user a task to do could be too short of an experience. The idea that we have right now is to make the user submit an assignment. However, in terms of the VR context of our project, how could we recreate a physical manifestation of such an internet-based task that is engaging enough to keep the user entertained for a minimum of three minutes?

April 21st
Playtest Session #2
In order to have a better, more complete idea of the way we would like to direct the user towards a sequence of actions he/she must complete in our project, we decided to playtest our prototype with our friends without giving them any clue of what our project is about. Attached is a video our playtest session:

ATTACH VIDEO HERE

After letting our friends play our prototype, this is some of the feedback we received from them:

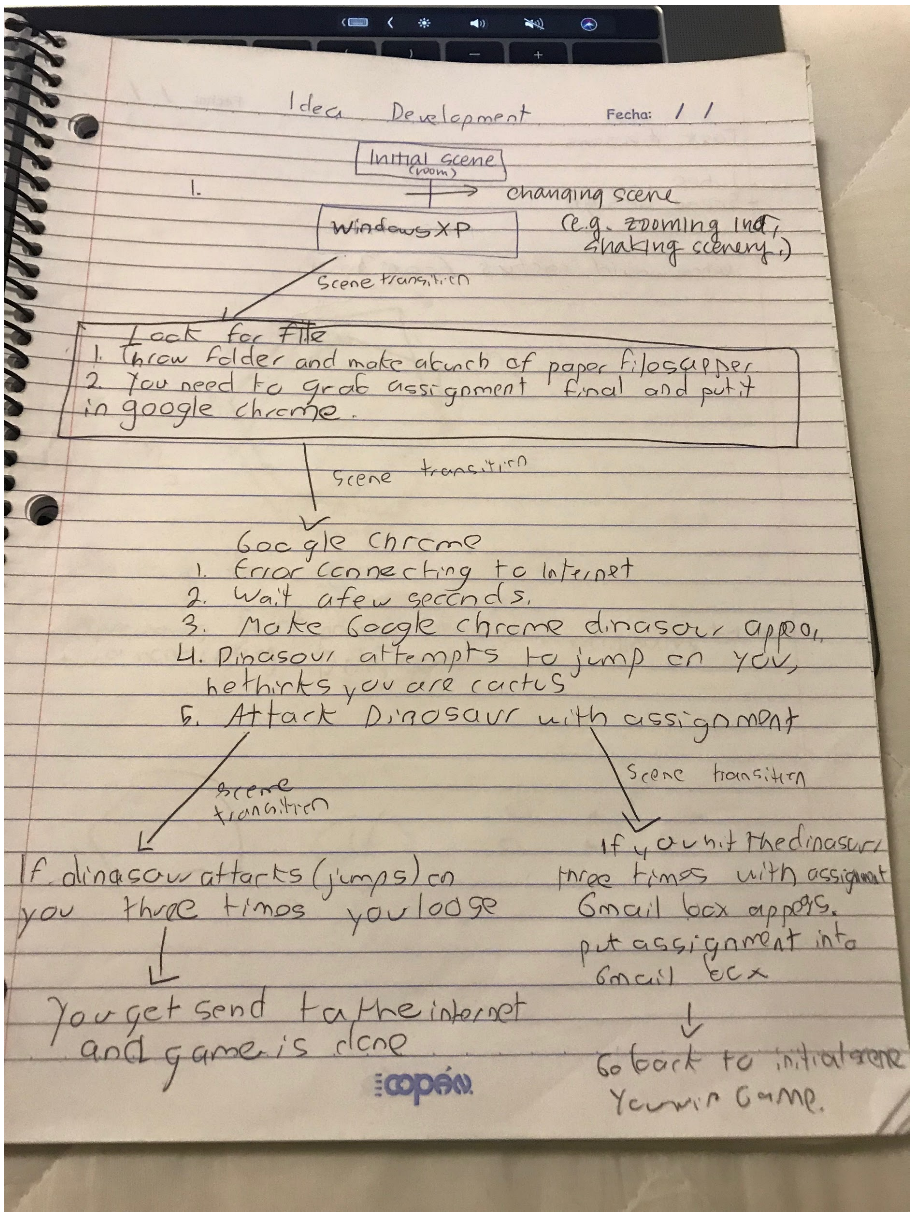
- Things they liked:
    - They liked playing with the boxes
    - They had the sense that they were getting inside the monitor

- Things they didn't fully understand:
    - The transition from the monitor scene to the inside the monitor scene wasn't clear. Maybe have a loading screen or a shaking of the camera that creates a pause and indicates a transition from the room to the inside of the computer.

- Things we could do to make the experience better:

- Interact with something iconic on the internet (error 404 guy, google chrome dinosaur)

Based on our playtester's feedback and our previous idea of submitting an assignment, we would like to combine both ideas into one. The iconic character our users would be interacting with is the jumping dinosaur in google chrome. We chose this character in hopes that it is iconic enough for any common user to familiarize with and to incorporate the internet on a controlled environment that is both technically feasible and significant enough to send our message across. This message, we hope, is too allow users to physically engage in computer-based task in a way that takes full advantage of Virtual Reality.



Idea Development Fecha: / /

Initial Scene (room)

1\.

changing scene (e.g. zooming ind, shaking scenery.)

WindowsXP

Scene transition

Look for File
1. Throw folder and make abunch of paper filesapper.
2. You need to grab assignment final and put it in google chrome.

Scene transition

Google Chrome
1. Error connecting to Internet
2. Wait afew seconds.
3. Make Google chrome dinasour appear,
4. Dinasour attempts to jump on you, he thinks you are cactus
5. Attack Dinosaur with assignment

Scene transition

If dinasour attacks (jumps) on you three times you loose

You get send to the internet and game is done

Scene transition

If you hit the dinasour three times with assignment Gmail box appears. put assignment into Gmail box

Go back to initial scene You win Game.

copén

Task Division:

Junior (Interactions):
Scene swap (done)
Throwing objects (done)
Make a bunch of papers appear from the folder after its thrown
Grab one of the papers and put it close to the google chrome icon
Make a dinosaur appear that jumps
Dinosaur wants to get close to you and jump on you
Attack the dinosaur with the folder

Juhee (Scene Design):

- Room scene
- Green valley (inside the computer) scene
- Stuck in the internet scene
- Google chrome scene

Adham (User Interface and sounds):

- Beginner and ending screen(2) (one for each ending)
- Instruction to users on what to do on each step of our story (either by an UI canvas or through sound)
    - When inside the room, user must touch the computer
    - When in the green valley, user must throw the folder to make a bunch of papers appear, grab the final version of the paper, and put it in google chrome
    - When inside google chrome, user must attack the dinosaur with the assignment three times and he shouldn't let the dinosaur jump on him
- If the user wins, then he should go back to the original scene and he should be notified(somehow) that he has successfully submitted the assignment
- If the user loses, then he should be indicated that he has been stuck in the internet.
- Sound effects that indicate transition from one scene to the next.

April 24
Today I had a really useful Idea-critique session with Sarah. She mentioned a couple of things that we must bear in mind as we try to finalize our project's concept:

1. Google Chrome dinosaur signifies lack of connectivity: This is an important piece of information that we must keep in mind. Our original premise was that after the dinosaur killed the user, the user would be stuck in the Internet and loose the game. How can the user be stuck in the Internet if the dinosaur only appears when there is no internet connection in the monitor? As such, we fixed this logical mishap by making the user restart the game. This not only fixes our storyline's logic, but also extends the playtime to the user as it makes him/her play the game a couple of times until they are able to beat the dinosaur and submit the assignment.

2. Kill the dinosaur with a cactus: Before, we wanted the user to kill the dinosaur with the assignment. However, Sarah recommended that we could give the user a cactus to kill the dinosaur as that is what actually kills the dinosaur in the real-life version of the game.

April 27-28
One of the main things that we must accomplish is to make the dinosaur jump and follow the user. As such, I created a script that follows the user (namely the target) and starts jumping after if it reaches a certain distance from the user. In every frame, I calculate the vector3 distance and the magnitude between the dinosaur and the player and I make the dinosaur aim at that direction and change its transform every frame with a predefined speed.
https://www.youtube.com/watch?v=QRp4V1JTZnM

Made the jumping dinasour script

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class followTarget : MonoBehaviour
{
   public GameObject target;//the target dinosaur will go to are going too

   public float speed;//the speed of enemiess
   public float jumpingdistance;//distance between object and target
   public bool jumped = false;
   public float jumping_height;
   void Start()
   {
   }

   // Update is called once per frame
   void Update()
   {
      Vector3 path = target.transform.position - transform.position;//calculate the path to target
      path = path / path.magnitude;//calculate only the direction of the path
      if (Vector3.Distance(target.transform.position, transform.position) <= jumpingdistance &&
jumped == false) //&& to_jump == false && jumped == false)
      {
         path.Set(path.x, jumping_height, path.z);//set the path to the jumping height
         transform.forward = path;//make enemies look forward to the target
         transform.position = transform.position + speed * path;
         jumped = true;//use jumped boolean to avoid infinite jumps
      }
```

```
        path.Set(path.x, 0.0f, path.z);//set the path on the ground
        transform.forward = path;//make enemies look forward to the target
        transform.position = transform.position + speed * path;//move enemies with speed "speed

    }
    void OnCollisionEnter(Collision collision)
    {
        if(collision.gameObject.name==target.name)
        {
            Vector3 backwards = new Vector3(transform.position.x - jumpingdistance, 0f,
transform.position.z - jumpingdistance);
            transform.position = backwards;//make the enemy go backwards if it collied with the
player
            jumped = false;// set the boolean backl to false to restart the jumping action

        }
    }
}
```

I also made the destroyer script that destroys the player after it has been jumped on 3 times. Instead of destroying, the player will be sent back to the initial scene. To start all over

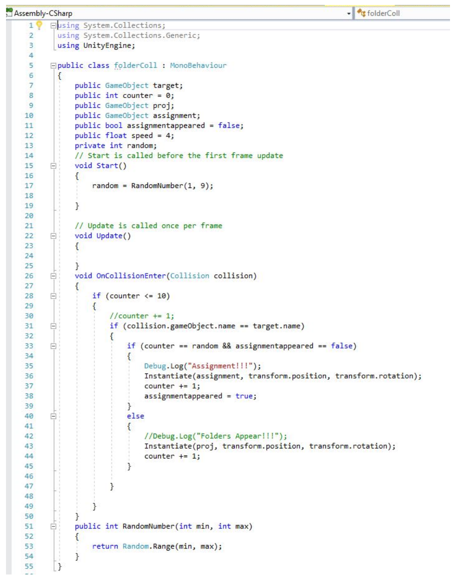
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class objectDestroy : followTarget
{
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        if (jumpcounter==3)
        {
            Debug.Log("You died");
            Destroy(target);

        }

    }
}
```

One of the problems I encountered while making this script was that the user could not be detected during game time. This comes as a result of using the Scene Manager, which has two simultaneous scenes running: the scene the user is currently in and the “DontDestroyonLoadScene” which carries the user (and any other objects the user was carrying from the previous scene) into the new scene.I didn't fully comprehend the complexity of this problem as I was prototyping the jumping interaction in my computer with boxes.

[ATTACH VIDEO HERE]

Given the little time there is between today and the project submission, one solution that was proposed to me by my classmate Max is to include all of the scenes into one and just change the transform.position of every object into the position of the new scene.

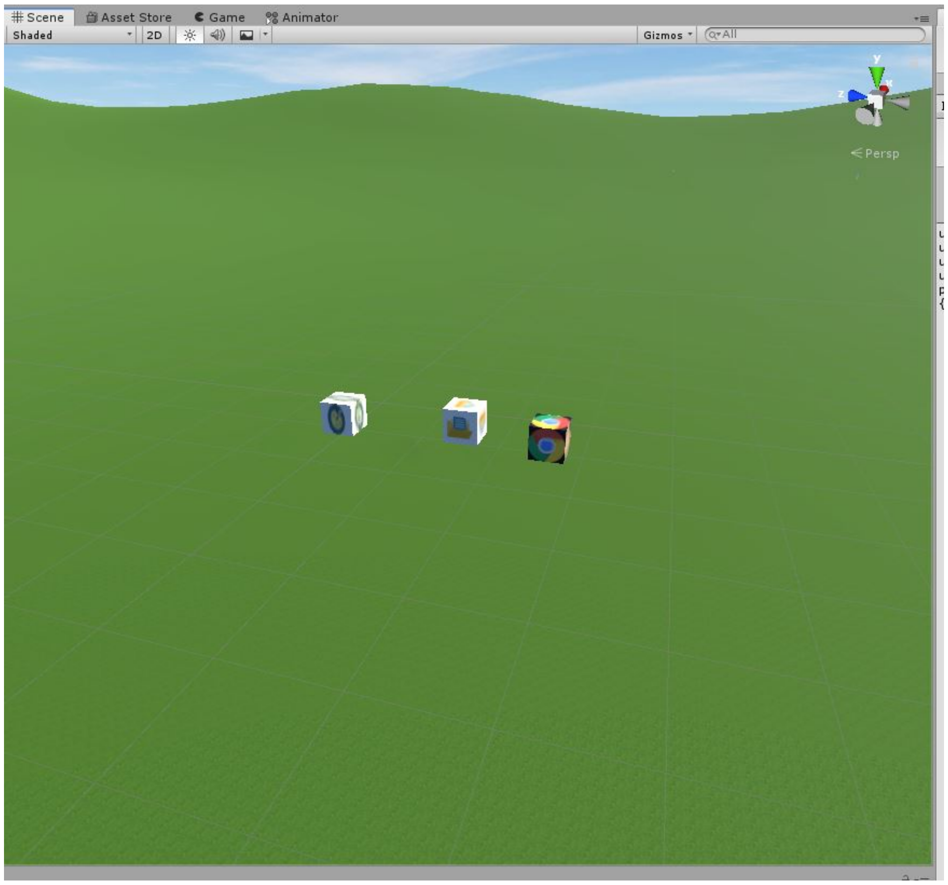
May 3-5
This weekend I worked on Instating Paper objects whenever the folder Icon collides with the ground. We also worked in finishing up the green valley scene and in uniting all 3 scenes into one to bypass the “DontDestroyonLoad” Issue.



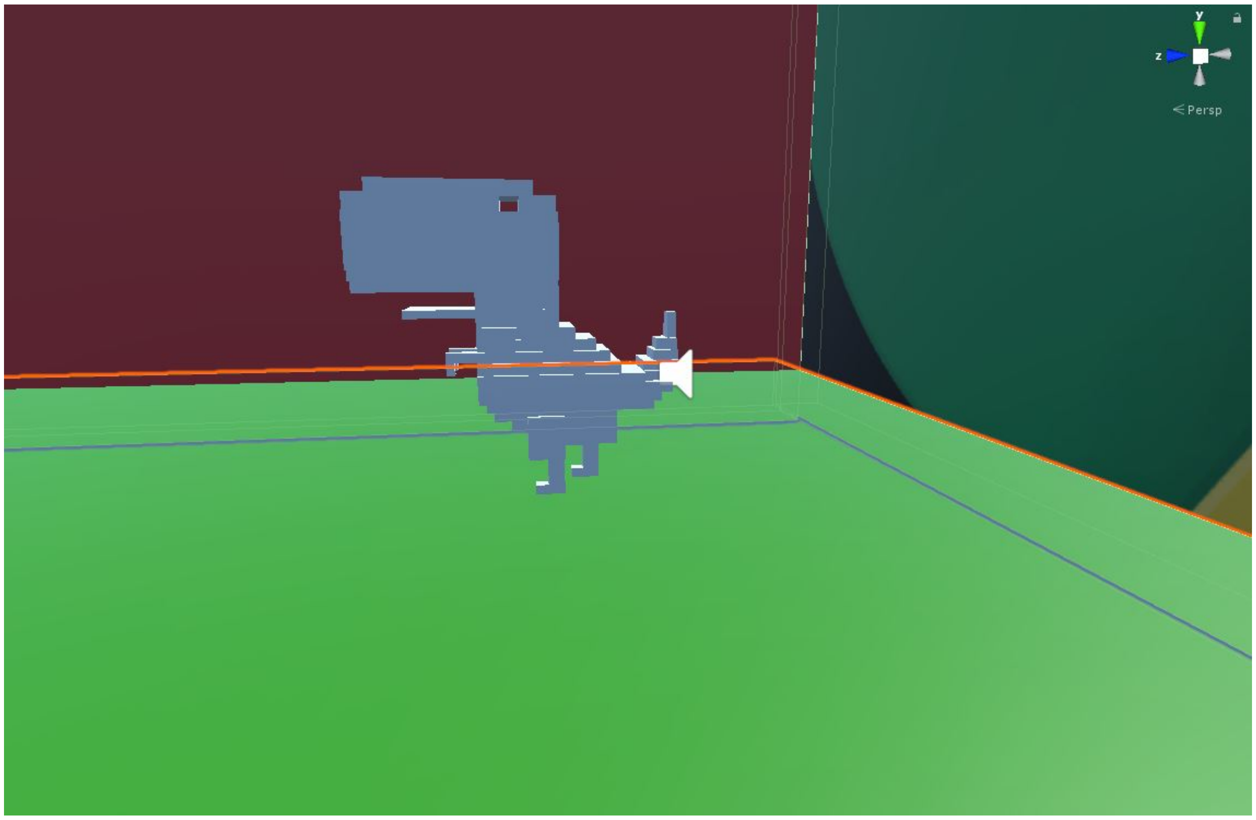
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class folderColl : MonoBehaviour
{
    public GameObject target;
    public int counter = 0;
    public GameObject proj;
    public GameObject assignment;
    public bool assignmentappeared = false;
    public float speed = 4;
    private int random;
    // Start is called before the first frame update
    void Start()
    {
        random = RandomNumber(1, 9);

    }

    // Update is called once per frame
    void Update()
    {

    }
    void OnCollisionEnter(Collision collision)
    {
        if (counter <= 10)
        {
            //counter += 1;
            if (collision.gameObject.name == target.name)
            {
                if (counter == random && assignmentappeared == false)
                {
                    Debug.Log("Assignment!!!");
                    Instantiate(assignment, transform.position, transform.rotation);
                    counter += 1;
                    assignmentappeared = true;
                }
                else
                {
                    //Debug.Log("Folders Appear!!!");
                    Instantiate(proj, transform.position, transform.rotation);
                    counter += 1;
                }

            }

        }
    }
    public int RandomNumber(int min, int max)
    {
        return Random.Range(min, max);
    }
}
```

What this script does is that it instantiates a new paper Game Object after every collision the folder has with the the terrain. The paper was created out of a an empty cube object that was then given its paper look by adding a paper material on it. There is also a counter that limits the number of papers that can be created after every collision with the terrain (10 collisions maximum). I also created a Random Number generator that creates a random number between a range of number (for this project, this range is set between 1 and 10) so that an assignment paper can be instantiated only once and randomly. The assignment paper is the same as the other paper objects with the addition that a canvas was added on top of it with the text "Assignment".

Me and Juhee also worked on doing the final touches in our Green Valley scene. We fixed the mountains and changed the skybox to one with more clouds that are vanishing as we thought it gave the scene a more invigorating, powerful feel to it. The mountains were also carefully designed for, not too big or not too small, in order to best recreate the vintage Windows XP look.

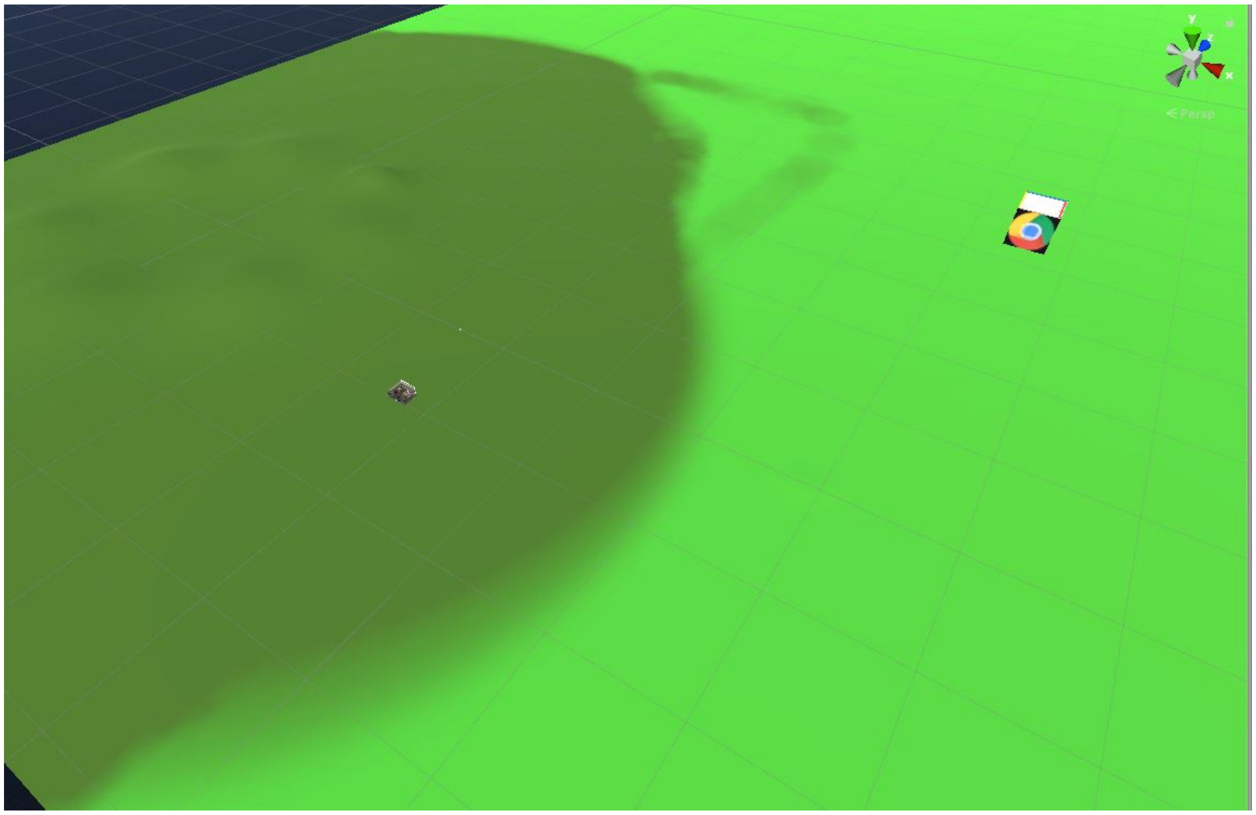


Juhee also spend a fair amount of time designing the dinosaur object. She made it out of a lot of cube objects to give it its iconic, pixelated look.

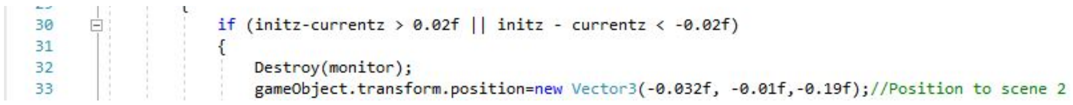


We also decided to create a Google chrome world, one that sends the message across that the user just landed in Google Chrome but it an abstract way. We also wanted to give the world an appearance similar to that of a combat studio as the user would be confronted, 1v1, with the dinosaur. As such, we went with a confined box environment, which creates the solitary, depredatory aura that we want the user to feel. We also decorated the box with Google Chrome's iconic green,yellow, red and blue colors on the sides of the cube, with one side being the google chrome icon itself in order to let the user know were they are at all times.

Finally, we decided to merge all three scenes into one scene in order to bypass the "DestroyonLoad" issue that we had as our player traversed one scene with the other. In order to do so, we made all the Game Objects in the 1st scene and the 3rd scene as children of one empty game object respectively. Then, to avoid any issues as this is a major change, we copied the entire Project File to perform all the major changes in a copy. We then copied the two game object's from the Project copy's assets' folder in the into the Original Project's Assets Folder. This allowed us to drag and pull the entire scene from our projects asset's folder inside the 2nd scene(Green Valley Scene).



This merging also required us to change the sceneswap script we had. Instead of using the SceneManager, we just changed the transform of the object to the position we wanted them to be inside the scene.



```
        if (initz-currentz > 0.02f || initz - currentz < -0.02f)
        {
            Destroy(monitor);
            gameObject.transform.position=new Vector3(-0.032f, -0.01f,-0.19f);//Position to scene 2
```

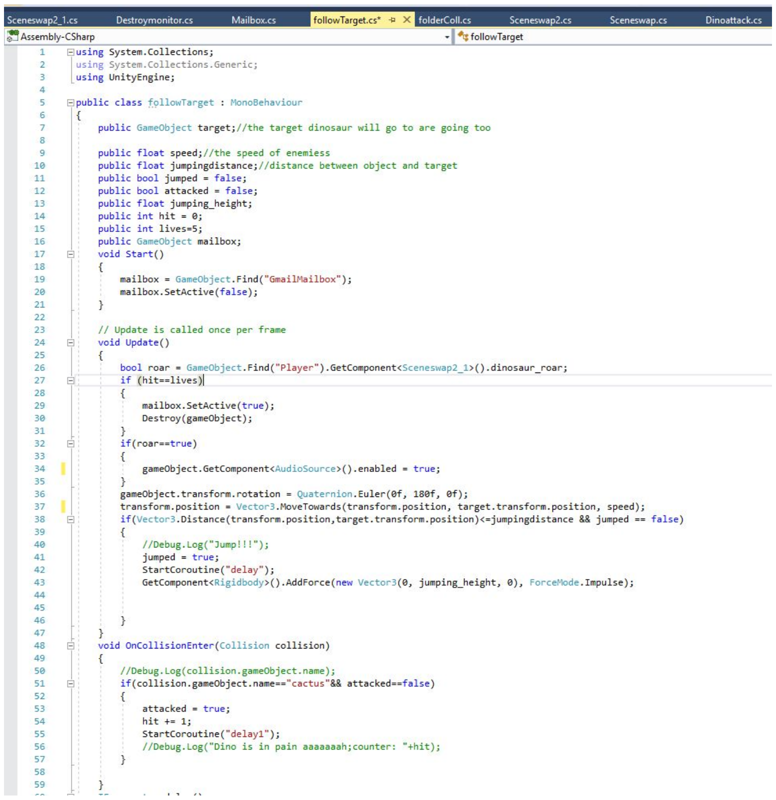
May 10-12
This weekend, we focused on finalizing the final interactions necessary for our game to be complete. So far, we have the assignment instantiation, the scene swap, and all three scenes merged into one. Now, we need to finalize the winning and restart conditions, finish the dinosaur script and fix the monitor's destruction (the object is still carried by the user as it moves from scene 1 to scene 2).

To fix the destruction of the monitor's script, a simple script attached to the mailbox sufficed.

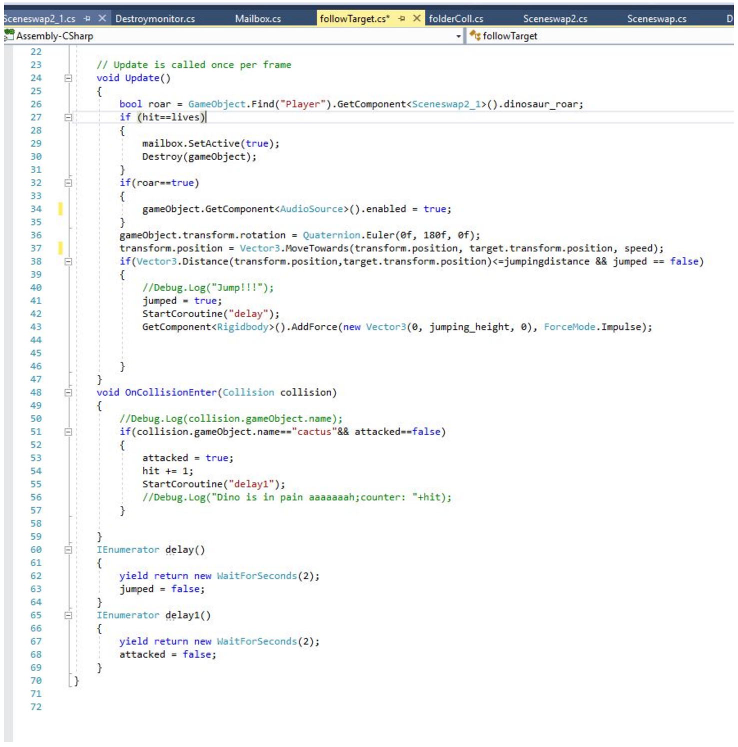
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Destroymonitor : MonoBehaviour
{
    public bool changed;
    public float initz;
    public float currentz;
    // Start is called before the first frame update
    void Start()
    {
        initz = gameObject.transform.position.z;
        changed = true;

    }

    // Update is called once per frame
    void Update()
    {
        currentz = gameObject.transform.position.z;
        if (changed == true)
        {
            if (initz - currentz > 0.02f || initz - currentz < -0.02f)
            {
                Destroy(gameObject);
                changed = false;

            }
        }
    }
}
```

As highlighted above, after there is a change in the transform.position of the monitor, the object will be destroyed.

Another major change that we had to achieve to achieve our project’s full fruition is the dinosaur script. This script has a lot of major components that deal with major parts of our project and such will be explained further in the upcoming paragraphs.



```csharp
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class followTarget : MonoBehaviour
{
    public GameObject target;//the target dinosaur will go to are going too

    public float speed;//the speed of enemiess
    public float jumpingdistance;//distance between object and target
    public bool jumped = false;
    public bool attacked = false;
    public float jumping_height;
    public int hit = 0;
    public int lives=5;
    public GameObject mailbox;
    void Start()
    {
        mailbox = GameObject.Find("GmailMailbox");
        mailbox.SetActive(false);
    }

    // Update is called once per frame
    void Update()
    {
        bool roar = GameObject.Find("Player").GetComponent<Sceneswap2_1>().dinosaur_roar;
        if (hit==lives)
        {
            mailbox.SetActive(true);
            Destroy(gameObject);
        }
        if(roar==true)
        {
            gameObject.GetComponent<AudioSource>().enabled = true;
        }
        gameObject.transform.rotation = Quaternion.Euler(0f, 180f, 0f);
        transform.position = Vector3.MoveTowards(transform.position, target.transform.position, speed);
        if(Vector3.Distance(transform.position,target.transform.position)<=jumpingdistance && jumped == false)
        {
            //Debug.Log("Jump!!!");
            jumped = true;
            StartCoroutine("delay");
            GetComponent<Rigidbody>().AddForce(new Vector3(0, jumping_height, 0), ForceMode.Impulse);


        }
    }
    void OnCollisionEnter(Collision collision)
    {
        //Debug.Log(collision.gameObject.name);
        if(collision.gameObject.name=="cactus"&& attacked==false)
        {
            attacked = true;
            hit += 1;
            StartCoroutine("delay1");
            //Debug.Log("Dino is in pain aaaaaaah;counter: "+hit);
        }

    }
```



```csharp

    // Update is called once per frame
    void Update()
    {
        bool roar = GameObject.Find("Player").GetComponent<Sceneswap2_1>().dinosaur_roar;
        if (hit==lives)
        {
            mailbox.SetActive(true);
            Destroy(gameObject);
        }
        if(roar==true)
        {
            gameObject.GetComponent<AudioSource>().enabled = true;
        }
        gameObject.transform.rotation = Quaternion.Euler(0f, 180f, 0f);
        transform.position = Vector3.MoveTowards(transform.position, target.transform.position, speed);
        if(Vector3.Distance(transform.position,target.transform.position)<=jumpingdistance && jumped == false)
        {
            //Debug.Log("Jump!!!");
            jumped = true;
            StartCoroutine("delay");
            GetComponent<Rigidbody>().AddForce(new Vector3(0, jumping_height, 0), ForceMode.Impulse);


        }
    }
    void OnCollisionEnter(Collision collision)
    {
        //Debug.Log(collision.gameObject.name);
        if(collision.gameObject.name=="cactus"&& attacked==false)
        {
            attacked = true;
            hit += 1;
            StartCoroutine("delay1");
            //Debug.Log("Dino is in pain aaaaaaah;counter: "+hit);
        }

    }
```

```csharp
    IEnumerator delay()
    {
        yield return new WaitForSeconds(2);
        jumped = false;
    }
    IEnumerator delay1()
    {
        yield return new WaitForSeconds(2);
        attacked = false;
    }
}

```

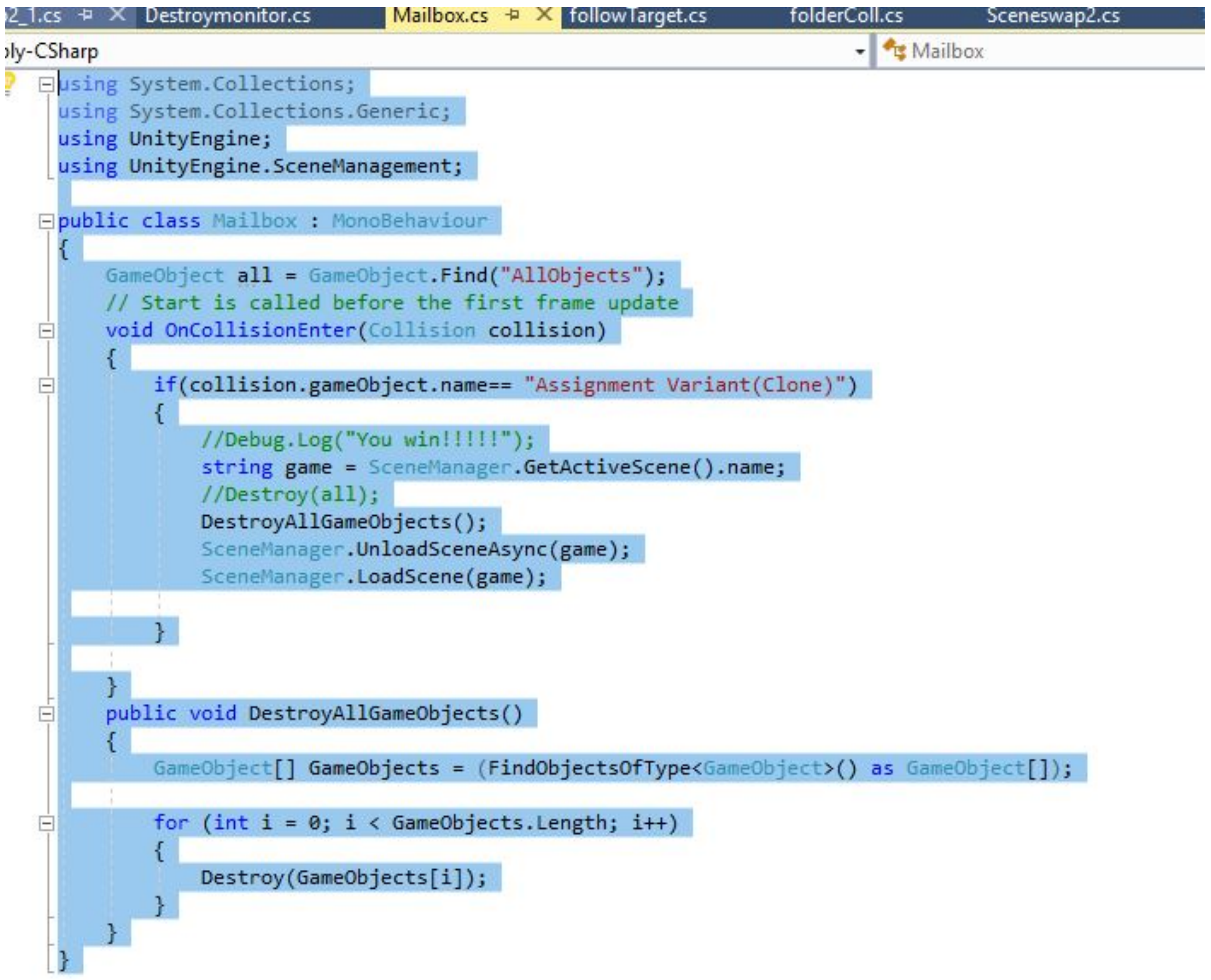
Vector3.Movetoward: This function simplified my code immensely. It boils down a lot of the code that I did before to make the dinosaur follow the target as it literally makes the object follow the object with an incremental change in position after every frame and it also changes the rotation of the object so that it looks at the target.

Rigidbody.addforce: Before, I was trying to literally hardcode the position the dinosaur had to be when it jumped, which seemed unnatural as the dinosaur would disappear and then appear somewhere in the air and then fall down. By setting the ForceMode to Impulse and determining the position in space you want the dinosaur to be after the jump, this line of code creates a natural jump of the object to the desired height.

Quaternion. Euler: Something that was happening after the force was added to the rigidbody was that the dinosaur would move amongst its own axis in space. In order to stop this, I froze the rotation of the Game object using the quaternion.euler function in the Update function so that it remains in place within its own axis after every jump.
Lives and hit counter: The lives and hit counters are an essential variable in the dinosaur that determine a lot of functionalities in my project. Whenever the number of hits is equal to the number of lives assigned to the dinosaur in the inspector, the mailbox appears. The mailbox then detects a collision between itself and the assignment to determine the win condition. This can be further appreciated in the mailbox script I attached to the mailbox.

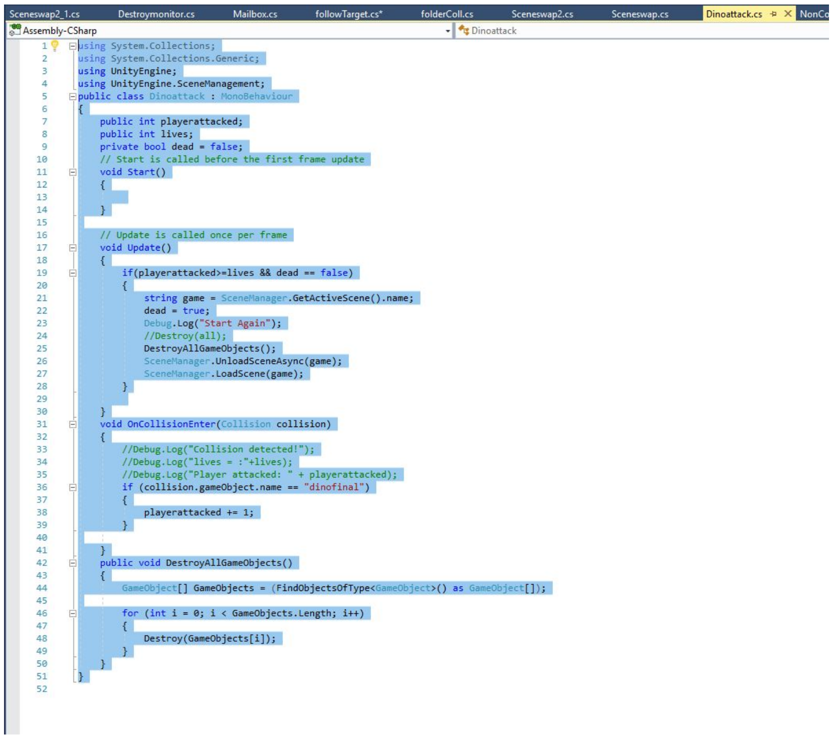
Coroutine and Delay function: I also learned about coroutines and implemented them in the script as well. I used them to cause a delay between every jump as the jumps were happen very frequently frame after frame. I also added the coroutine on the collision detection so that the number of hits between the dinosaur and the player were reduced as a lot of collisions were happening frame after frame too.

The winning and losing conditions of our project were determined by two scripts attached to the mailbox and the dinosaur respectively.



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class Mailbox : MonoBehaviour
{
    GameObject all = GameObject.Find("AllObjects");
    // Start is called before the first frame update
    void OnCollisionEnter(Collision collision)
    {
        if(collision.gameObject.name== "Assignment Variant(Clone)")
        {
            //Debug.Log("You win!!!!!");
            string game = SceneManager.GetActiveScene().name;
            //Destroy(all);
            DestroyAllGameObjects();
            SceneManager.UnloadSceneAsync(game);
            SceneManager.LoadScene(game);

        }

    }
    public void DestroyAllGameObjects()
    {
        GameObject[] GameObjects = (FindObjectsOfType<GameObject>() as GameObject[]);

        for (int i = 0; i < GameObjects.Length; i++)
        {
            Destroy(GameObjects[i]);
        }
    }
}
```

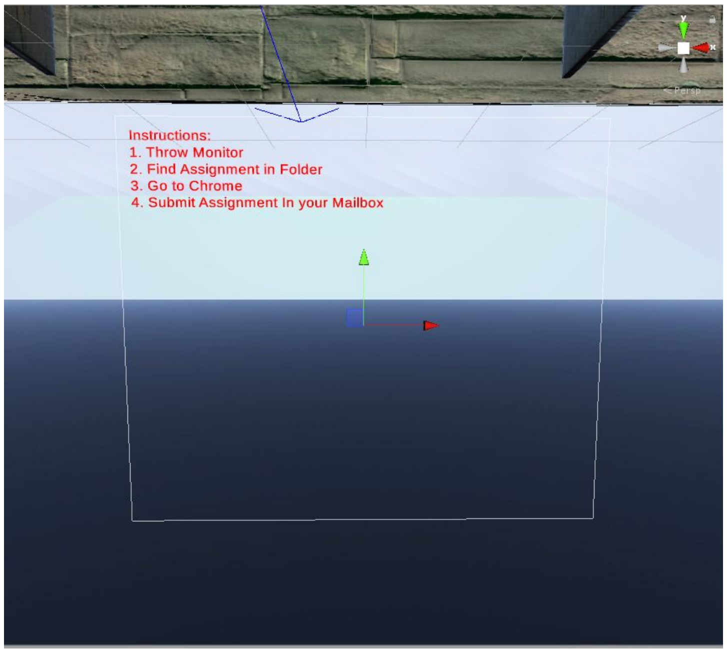
This script detects the collision between the mailbox and the assignment. If this happens, the entire scene is reloaded as the player won the game. Dealing with the SceneManager was definitely one of the biggest challenges I had with Unity this semester. Something that I learned after hours of testing different strategies in order to restart an entire scene was that the best way to achieve this is by deleting all of the game objects of the scene (including the player) and then loading the new scene. As such, I found this function “DestroyAllGameObjects()” that iterates through the entire hierarchy and deletes every single game object. I call this function before unloading the current scene and loading the same scene in order to achieve the scene restar condition.



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
public class Dinoattack : MonoBehaviour
{
    public int playerattacked;
    public int lives;
    private bool dead = false;
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        if(playerattacked>=lives && dead == false)
        {
            string game = SceneManager.GetActiveScene().name;
            dead = true;
            Debug.Log("Start Again");
            //Destroy(all);
            DestroyAllGameObjects();
            SceneManager.UnloadSceneAsync(game);
            SceneManager.LoadScene(game);
        }

    }
    void OnCollisionEnter(Collision collision)
    {
        //Debug.Log("Collision detected!");
        //Debug.Log("lives = :"+lives);
        //Debug.Log("Player attacked: " + playerattacked);
        if (collision.gameObject.name == "dinofinal")
        {
            playerattacked += 1;
        }

    }
    public void DestroyAllGameObjects()
    {
        GameObject[] GameObjects = (FindObjectsOfType<GameObject>() as GameObject[]);

        for (int i = 0; i < GameObjects.Length; i++)
        {
            Destroy(GameObjects[i]);
        }
    }
}
```

Similarly, this script also detects for a collision between the user and the dinosaur. If the number of attacks supersedes the number of lives alloted to the player, then the scene is restarted.The restart process entails the same logic of destroying all game objects and unloading and loading the scene.

I also added a simple UI element to my project. I added a set of instructions that followed the player as most playtest sessions required me to explain to the user what to do.



Instructions:
1. Throw Monitor
2. Find Assignment in Folder
3. Go to Chrome
4. Submit Assignment In your Mailbox

Final Scripts

1. **Destroy Monitor**

Attacghed to monitor in initial scene to ensure it is destroyed after the player is moved to the next scene.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Destroymonitor : MonoBehaviour
{
    public bool changed;
    public float initz;
    public float currentz;
    // Start is called before the first frame update
    void Start()
    {
        initz = gameObject.transform.position.z;
        changed = true;

    }

    // Update is called once per frame
    void Update()
    {
        currentz = gameObject.transform.position.z;
        if (changed == true)
        {
            if (initz - currentz > 0.02f || initz - currentz < -0.02f)
            {
                Destroy(gameObject);
                changed = false;

            }
        }
    }
}
```

2. Mailbox Script:

Script attached to0 the mailbox that ensures the game is restarted after the user wins the game by submitting the assignment to the mailbox

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class Mailbox : MonoBehaviour
{
    GameObject all = GameObject.Find("AllObjects");
    // Start is called before the first frame update
    void OnCollisionEnter(Collision collision)
    {
        if(collision.gameObject.name== "Assignment Variant(Clone)")
        {
            //Debug.Log("You win!!!!!");
            string game = SceneManager.GetActiveScene().name;
            //Destroy(all);
            DestroyAllGameObjects();
            SceneManager.UnloadSceneAsync(game);
            SceneManager.LoadScene(game);

        }

    }
    public void DestroyAllGameObjects()
    {
        GameObject[] GameObjects = (FindObjectsOfType<GameObject>() as GameObject[]);

        for (int i = 0; i < GameObjects.Length; i++)
        {
            Destroy(GameObjects[i]);
        }
    }
}
```

3. Followtarget: Dinosaur script

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class followTarget : MonoBehaviour
{
    public GameObject target;//the target dinosaur will go to are going too
    public float speed;//the speed of enemiess
    public float jumpingdistance;//distance between object and target
    public bool jumped = false;
    public bool attacked = false;
    public float jumping_height;
    public int hit = 0;
    public int lives=5;
    public GameObject mailbox;
    void Start()
    {
        mailbox = GameObject.Find("GmailMailbox");
        mailbox.SetActive(false);
    }

    // Update is called once per frame
    void Update()
    {
        bool roar = GameObject.Find("Player").GetComponent<Sceneswap2_1>().dinosaur_roar;
        if (hit==lives)
        {
            mailbox.SetActive(true);
            Destroy(gameObject);
        }
        if(roar==true)
        {
            //gameObject.GetComponent<AudioSource>().enabled = true;
        }
        gameObject.transform.rotation = Quaternion.Euler(0f, 180f, 0f);
        transform.position = Vector3.MoveTowards(transform.position, target.transform.position,
speed);
        //Debug.Log("Distance between Player and dino: " + Vector3.Distance(transform.position,
target.transform.position));
        if(Vector3.Distance(transform.position,target.transform.position)<=jumpingdistance &&
jumped == false)
        {
            //Debug.Log("Jump!!!");
            jumped = true;
            StartCoroutine("delay");
            GetComponent<Rigidbody>().AddForce(new Vector3(0, jumping_height, 0),
ForceMode.Impulse);


        }
    }
    void OnCollisionEnter(Collision collision)
    {
        //Debug.Log(collision.gameObject.name);
        if(collision.gameObject.name=="cactus"&& attacked==false)
        {
            attacked = true;
            hit += 1;
            StartCoroutine("delay1");
            //Debug.Log("Dino is in pain aaaaaaah;counter: "+hit);
        }

    }
    IEnumerator delay()
    {
        yield return new WaitForSeconds(2);
        jumped = false;
    }
    IEnumerator delay1()
    {
        yield return new WaitForSeconds(2);
        attacked = false;
    }
}
```

4. Foldercoll:
Script that instantiates papers and the assignment when the folder icon touches the ground

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class folderColl : MonoBehaviour
{
    public GameObject target;
    public int counter = 0;
    public GameObject proj;
    public GameObject assignment;
    public bool assignmentappeared = false;
    public float speed = 4;
    private int random;
    // Start is called before the first frame update
    void Start()
    {
        random = RandomNumber(1, 9);

    }

    // Update is called once per frame
    void Update()
    {

    }
    void OnCollisionEnter(Collision collision)
    {
        if (counter <= 10)
        {
            //counter += 1;
            if (collision.gameObject.name == target.name)
            {
                if (counter == random && assignmentappeared == false)
                {
                    Debug.Log("Assignment!!!");
                    Instantiate(assignment, transform.position, transform.rotation);
                    counter += 1;
                    assignmentappeared = true;
                }
                else
                {
                    //Debug.Log("Folders Appear!!!");
                    Instantiate(proj, transform.position, transform.rotation);
                    counter += 1;
                }

            }

        }
    }
    public int RandomNumber(int min, int max)
    {
        return Random.Range(min, max);
    }
}
```

5. Sceneswap 2_1:

Changes the user to the chrome world scene

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Sceneswap2_1 : MonoBehaviour
{
    bool  checkassignment;
    bool changed;
    public bool dinosaur_roar = false;
    // Start is called before the first frame update
    void Start()
    {
       changed = true;
    }

    // Update is called once per frame
    void Update()
    {
        checkassignment =
GameObject.Find("chrome").GetComponent<Sceneswap2>().assignmentcollide;

        if (checkassignment==true && changed ==true)
        {
            //Debug.Log("Go to chrome!!!");
            gameObject.transform.position = new Vector3(75.5263f, 0.47f, 46.372f);
            changed = false;
            dinosaur_roar = true;
        }
```

6. Sceneswap 2:
Detects collision between assignment and chrome icon and sets the assignment collide boolean to true, which is then referenced be Sceneswap 2_1 to change the transform of the user to the next scene

```
    }
}
```

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class Sceneswap2 : MonoBehaviour
```

```
{
    public bool assignmentcollide = false;
    void OnCollisionEnter(Collision collision)
    {
        if (collision.gameObject.name=="Assignment Variant(Clone)")
        {
            assignmentcollide =true;
        }

    }



}
```

7. Scene swap :
Changes the user from the room scene to the inside windows xp scene

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class Sceneswap : MonoBehaviour
{
    // Start is called before the first frame update
    public bool changed;
    public float initz;
    public float currentz;
    GameObject monitor;
    //public Scene currentscene;
    void Start()
    {
        monitor = GameObject.Find("monitor");
        //currentscene = SceneManager.GetActiveScene();
        initz = GameObject.Find("monitor").transform.position.z;
        changed = true;
    }

    // Update is called once per frame
    void Update()
    {
        currentz= GameObject.Find("monitor").transform.position.z;
        //print("currentz: "+currentz);
        //print("initz:"+  initz);
        if (changed==true)
        {
            if (initz-currentz > 0.02f || initz - currentz < -0.02f)
            {
                Destroy(monitor);
                gameObject.transform.position=new Vector3(-0.032f, -0.01f,-0.19f);//Position to scene
2
                //SceneManager.LoadSceneAsync("Scene2");
                //SceneManager.UnloadSceneAsync(currentscene);
                //Application.LoadLevel(2);
                changed = false;
            }



        }

    }
}
```

8. Dino attack
Script that keeps track of the number of times the dinosaur attacks the player. If the lives are exhausted, the user loose and the game is restarted

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
public class Dinoattack : MonoBehaviour
{
    public int playerattacked;
    public int lives;
    private bool dead = false;
    // Start is called before the first frame update
    void Start()
    {

    }
    // Update is called once per frame
    void Update()
    {
        if(playerattacked>=lives && dead == false)
        {
            string game = SceneManager.GetActiveScene().name;
            dead = true;
            Debug.Log("Start Again");
            //Destroy(all);
            DestroyAllGameObjects();
            SceneManager.UnloadSceneAsync(game);
            SceneManager.LoadScene(game);
        }

    }
    void OnCollisionEnter(Collision collision)
    {
        //Debug.Log("Collision detected!");
        //Debug.Log("lives = :"+lives);
        //Debug.Log("Player attacked: " + playerattacked);
        if (collision.gameObject.name == "dinofinal")
        {
            playerattacked += 1;
        }

    }
    public void DestroyAllGameObjects()
    {
        GameObject[] GameObjects = (FindObjectsOfType<GameObject>() as GameObject[]);

        for (int i = 0; i < GameObjects.Length; i++)
        {
            Destroy(GameObjects[i]);
        }
    }
}
```

Bugs to fix:
In chromeworld, assignment goes far away
In green valley, papers fall through terrain collider
Add teleportation to allow for world exploration
Initial scene is too big
Scaling problem: The entire world is too small

Assignment can sometimes be thrown really far away. Add invisible colliders to prevent this from happening