Jordan Schwichtenberg ID: 911860526

CSc 631 Project 1 Client

Client Development Documentation

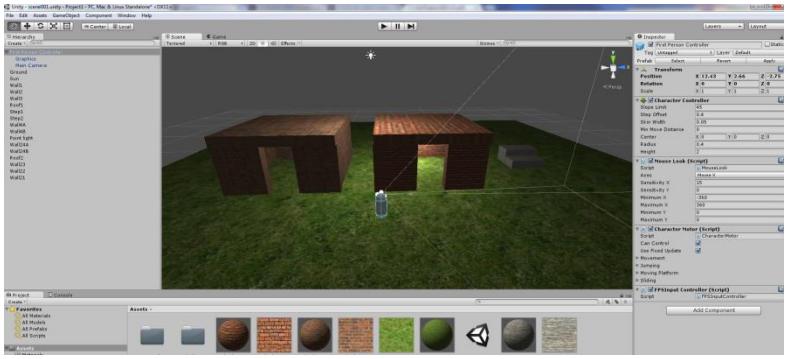
Play with Unity3D – going through tutorial and make a simple interactive scene

Learned terms:

- Point light
  - A game object from which light emanates
- Rigidbody
  - Control of an object's position through physics simulation
    - A component that you add to a GameObject
    - Without adding any code, a Rigidbody object will be pulled downward by gravity and react to collisions with incoming objects if the right Collider is also present
- Prefab
  - Allows you to store a GameObject complete with components and properties. The prefab acts as a template from which you can create new object instances in the scene.
    - Any edits made to a prefab are immediately reflected in all instances produced from it
    - Can also override components and settings for each instance individually
- Quaternion
  - They are used to represent rotations. They are based on complex numbers and are not easy to understand.

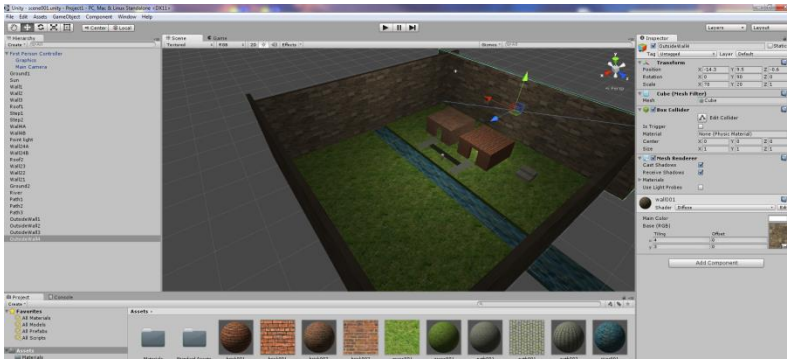
Things I've learned how to do:

- Create a material asset
  - To apply a texture to a game object, it's better to create a texture and then apply that texture to a material first, and then apply that material to the game object, as it gives you a little more control over how the texture appears in the game world
  - Apply a texture to that material



  - Apply a material to a game object, and scale the tiling so it looks decent

- First person controller
    - The game is played through a first person perspective
- Write a C# script that animates the river object I created
    - I used renderer.material.mainTextureOffset = new Vector2 (offsetX, offsetY)
        - I used this method in the update() method, which is called every frame
        - Every frame I calculate a new offset using offsetX = Time.time * scrollspeedX



- How to use empty GameObjects to organize your assets folder
- I used transform.Rotate() to make a golden coin (which was created from a cylinder) rotate, like some games in which you collect coins to gain points
- Physics components
    - Add a Rigidbody component to the player camera, select the Is Kinematic option
    - Add a Box Collider component to the player camera, select the Is Trigger option
        - This allows a script to be run when the player collides with a coin
    - Add the Rigidbody component to the golden coin object, select the Is Kinematic option
    - Used OnTriggerEnter(Collider other) which is called when the player collides with the coin, and Destroy(gameObject) makes the coin vanish, like the player picked up the coin
    - When you play the game, colliding with each coin causes that token's gameObject to be deleted from the scene view
- Use Particle System to create a glowing torch

Creating a Scene

I created a script called “TestSceneScript”; I also created the MassaiMara prefab and a first person camera.

In the script, the original prefab, which exists in the scene before the game is launched, I store both the position as a Vector3 and the rotation as a Quaternion. I then load the prefab from the Resources/Prefabs directory (I moved the prefab) and I told the script to create the object at the original object’s position and rotation. I then destroy the original object. The new object shows up as “MassaiMara(Clone)” in the hierarchy view during gameplay. The code I wrote lives in the void Start() function, which runs when the game first starts.

When you play the game, the original land is destroyed, and a new one (a clone) is created in its place.

Populating Scene with Species

I created a script called “TestSceneScript2”, in which in the Start() method I get five animal textures using Resources.Load() as Texture2D; . I also set a default scale for all the animals using a Vector3, and for each animal I use GameObject.CreatePrimitive(PrimitiveType.Cube) to create a cube with a z-axis of 0 to make them like sprites (not the best way, but it works).

When the script first runs five species are instantiated in random locations. To determine the random location, to make sure the animal is created within the terrain and not out in space, I first put the Prefab MassaiMara to 0, 0, 0, which then allowed me to see that the x and z dimensions fall roughly in the range of 20-1950. So, I created two methods, getRandX() and getRandY() that returns a random int within that range.

In the update method, there are three if (Input.GetKeyDown(KeyCode)) statements, which run when that particular key is pressed down. In each method, I instantiate a different species in a random location.

Creating a Simple GUI Element

To add a button to the Login scene to switch to the TestScene I simply added some more space to the rectangle that contained the text fields and submit button using GUILayout.Space(), and then I simply added a new button just like the previous Submit button, and when it is clicked it loads the test scene.

To add a button in the TestScene to go back to the login scene, I did the same thing, but I put the button code in the void OnGUI() in my TestScene2 script.

Using Collision Rays to activate mouse click on the scene

I created a script called CreateSpeciesOnClick which I attached to the First Person Controller (I use this as camera so player can look around). It uses if (Input.GetMouseButtonDown(0)) in the update method, which creates a new Ray called ray that takes the Camera and the mousePosition and creates a ray. I also create a RaycastHit called hit.Then, I use Physics.RayCast, with ray and out hit as the parameters to create a Raycast in the direction from the camera’s view to where the mouse was clicked.

With a third if statement, I use if (hit.collider.tag == “Ground”) to test if the Ray intersected the terrain (which I gave a tag called ‘Ground’). I then used the same method I used in my TestSceneScript2 to create a species, but instead of giving a random location I use hit.point.x, hit.point.y, hit.point.z, to instantiate the species where the player clicked.