ChunkMaster

Rendering Blocks

Index

---

1. Preparing prefabs

a) Preparing the grass and dirt prefabs

Well this is all great and all that we could create, save, load and remove blocks from and empty world, but how can we display them inside the unity engine?

The first thing we will need to do is to create our block prefabs. Lets go ahead and create a few blocks shall we? We will create a dirt block and a grass block. Right click in the inspector and create a cube.

See Figure - 1

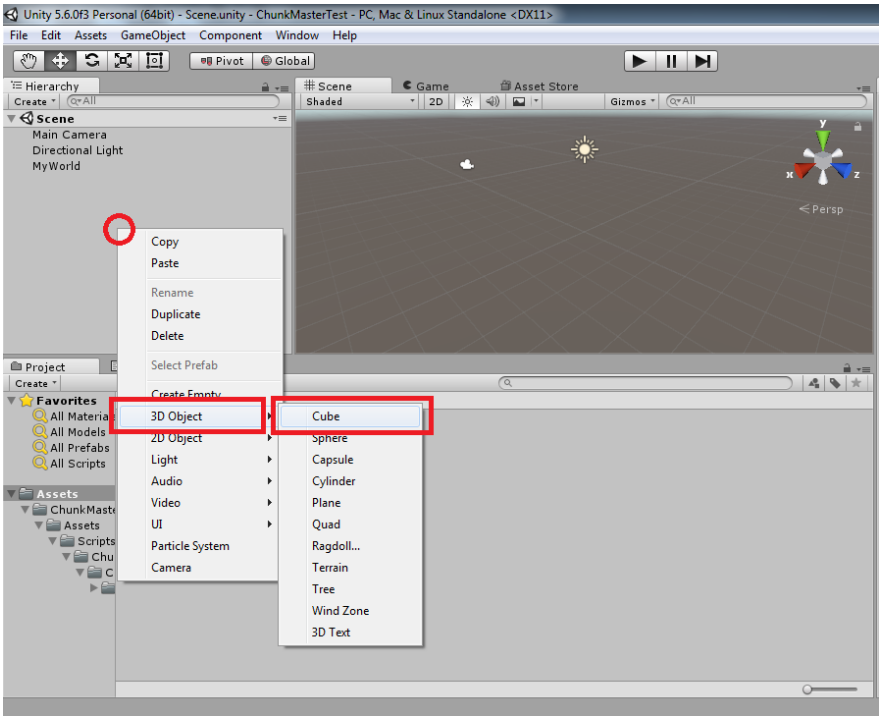

Figure - 1

Next we will rename the cube to "Grass" as seen in the figure below.

See Figure - 2

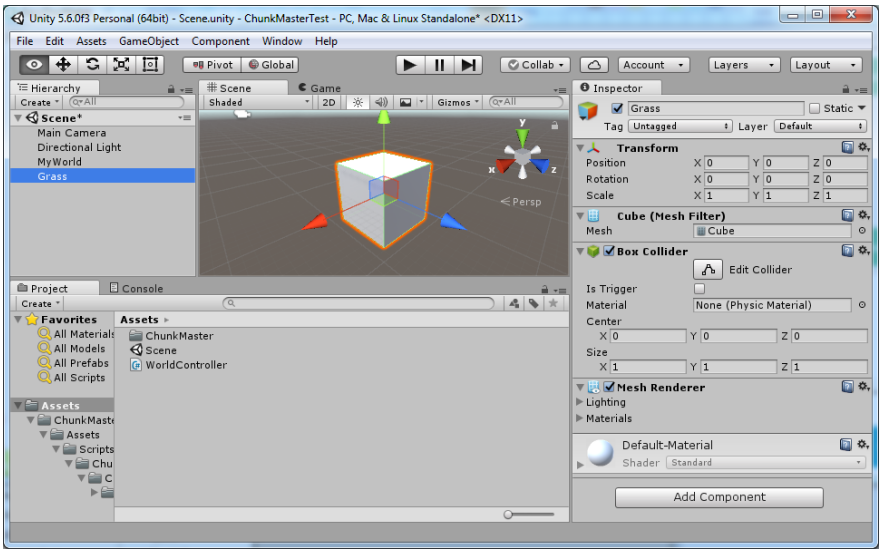

Figure - 2

We will create a new material by right clicking in the project browser and selecting "Material"

See Figure - 3

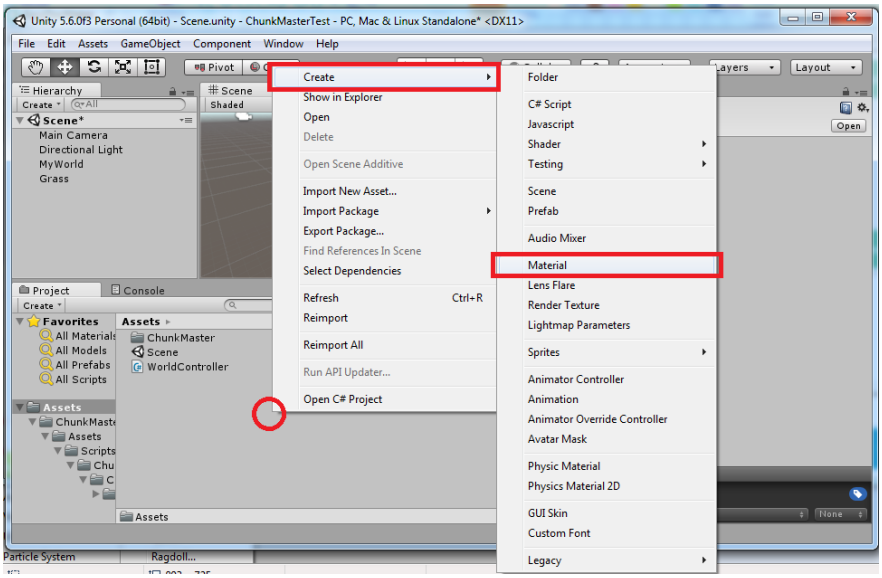

Figure - 3

Rename the material to "Grass" and set it as a green color as indicated in the figure below.

See Figure - 4

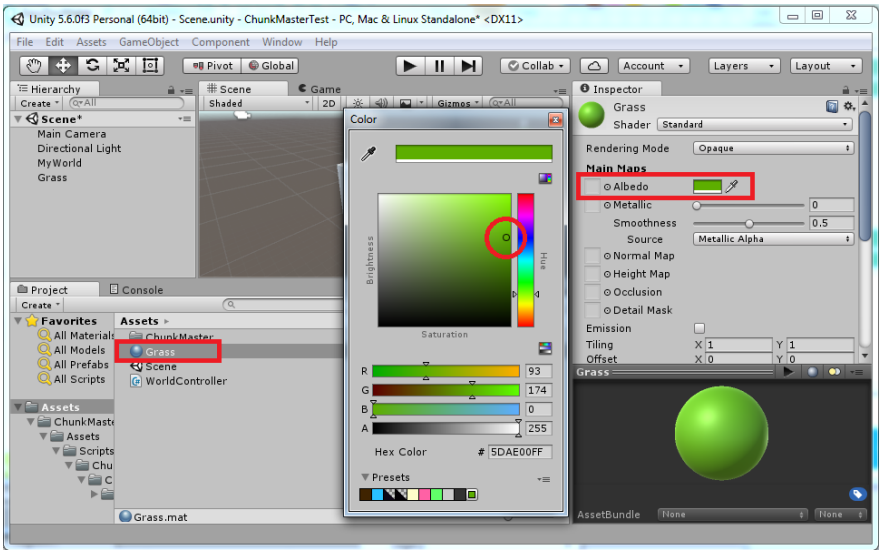

Figure - 4

Next we will assign the material to our cube.

Create a folder named "Prefabs" and drag and drop your "Grass" cube into the prefabs folder.

See Figure - 5

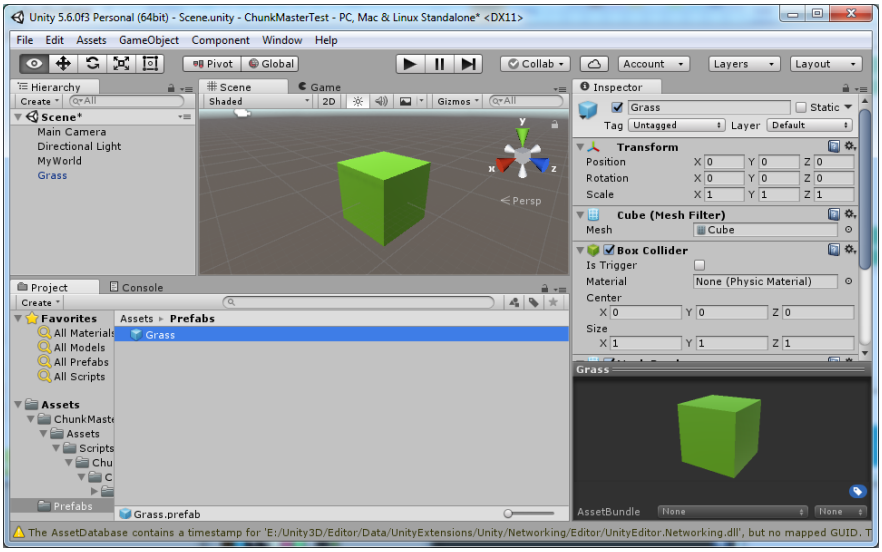

Figure - 5

Duplicate the grass cube in the inspector and rename it to dirt. Repeat the previous steps to the dirt cube, i.e. create a new material, name it "Dirt", change the color to a brownish color, assign the material to your dirt cube, and drag the dirt cube into the prefabs folder. Your unity editor should look something like this.

See Figure - 6

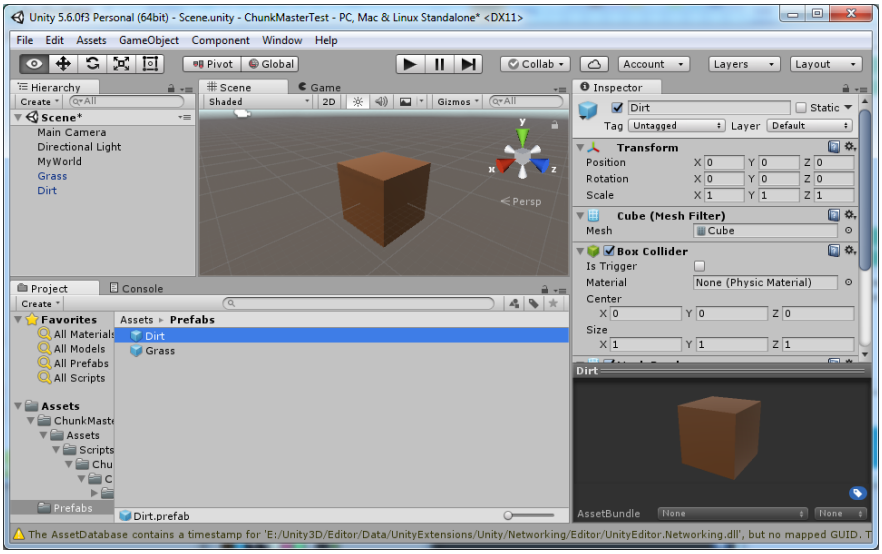

Figure - 6

Delete the cubes in the 3D view inspector, and your project should be ready for spawning blocks! Below is a final look at what your editor should look like.

See Figure - 7

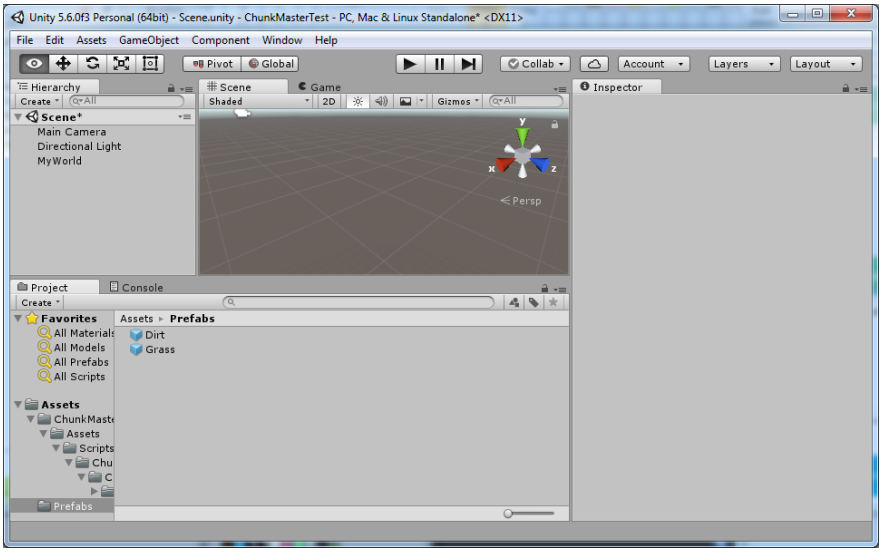

Figure - 7

I Completed This Section

2. Spawning blocks in code

a) Creating the spawn method

Now back to coding.

Before we can spawn our prefabs, we will need to declare some variables. Add the following code to the top of your class. We will be creating an array of Transforms with a variable name "blocks". This is where we will be placing our cube prefabs into for spawning/instantiating.

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

using ChunkMaster.Unity.Framework;

public class WorldController : MonoBehaviour {

    public static UnityWorld myWorld;

    public Transform[] blocks;

    // Use this for initialization
    void Start ()
```

```
    {
        // Creating a myWorld instance object
        myWorld = new UnityWorld(transform);

        // Declaring a random point in space
        Vector3 worldPoint = new Vector3(15.4f, 64.4f, -12f);

        // Loading a sector
        myWorld.LoadSector(worldPoint, "E:\\ChunkMasterTutoria
    }

}
```

Next we will drag and drop our prefabs into the unity editor inspector where our WorldController is located. Your inspector should look something like this.

See Figure - 8

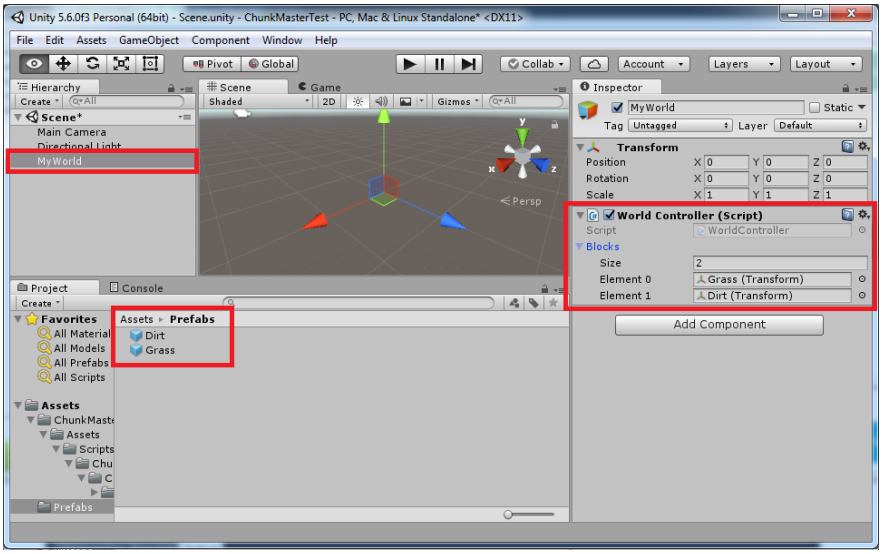

Figure - 8

We will now create the SpawnPrefab method. In order to spawn our cubes in the correct location we will need to instantiate them using a world coordinate, a parent to attach to, and the correct prefab itself. See the code below for a simple spawn prefab method.

```
    // Spawns a prefab
    public void SpawnPrefab(Transform prefab,
        Vector3 worldPosition,
        Transform parent)
    {
        Transform tmp = Instantiate(prefab);
        tmp.position = worldPosition;
        tmp.SetParent(parent);
    }
```

As you can see there is nothing too complicated here, we are simply creating a method with three parameters and assigning the formal parameters to our instantiated prefab.

Next we will need to create the SpawnBlock method. Creating this method will require one single formal parameter, a "UnityBlock" data type.

```
// Spawn Block
public void SpawnBlock(UnityBlock block)
{
    // The block's world position.
    Vector3 worldPosition = block.worldPosition;

    // The block's content.
    int blockIndex = (int)block.GetContent();

    // The block's parent object.
    Transform parent = block.gameObject.transform;

    // Spawn block.
    SpawnPrefab(blocks[blockIndex], worldPosition, parent
}
```

Using what we've learned in "Manipulating Blocks", we can create the above method with using as little statements as possible.

b) Spawning your block

Now that we have our methods ready to go, we can add more statements to the start method. We will need to retrieve the block given a point in world space, next we will spawn the block in the unity world 3D space. See the code below for an example on how to accomplish this.

```
        // Spawn your block
        SpawnBlock(block);
```

One statement? Wow now that is really simple! Let's see what the result is in the unity world space.

See Figure - 9

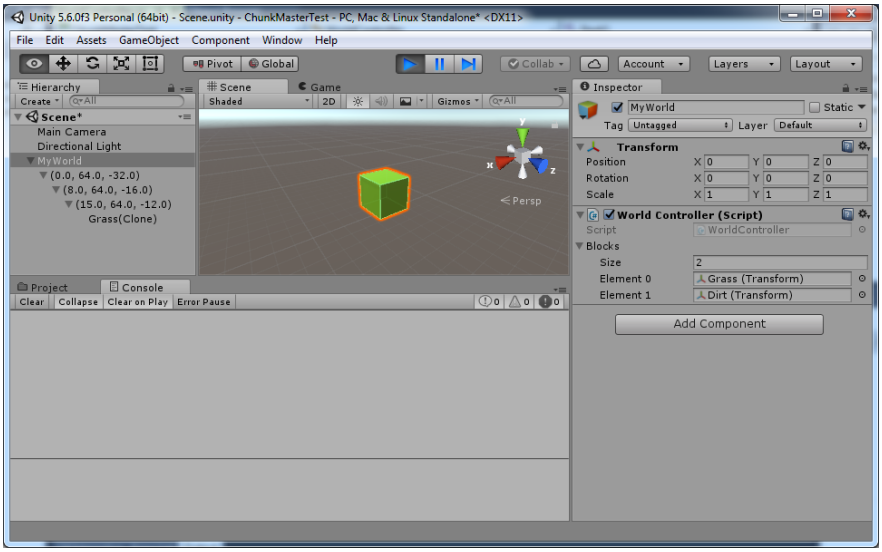

Figure - 9

Now I have to say this is pretty neat. Lets spawn a dirt block right next to our grass block!

```
        // Set a dirt block to the left side of our grass blo
        UnityBlock dirtBlock = myWorld.SetBlock(worldPoint + V

        // Spawn dirt block
        SpawnBlock(dirtBlock);
```

As you can see we've added two statements, SetBlock and SpawnBlock. Did I forget to mention that the SetBlock method returns the UnityBlock itself? You can use the output of this method to spawn and or manipulate the block. Here is the result inside the unity engine.

See Figure - 10

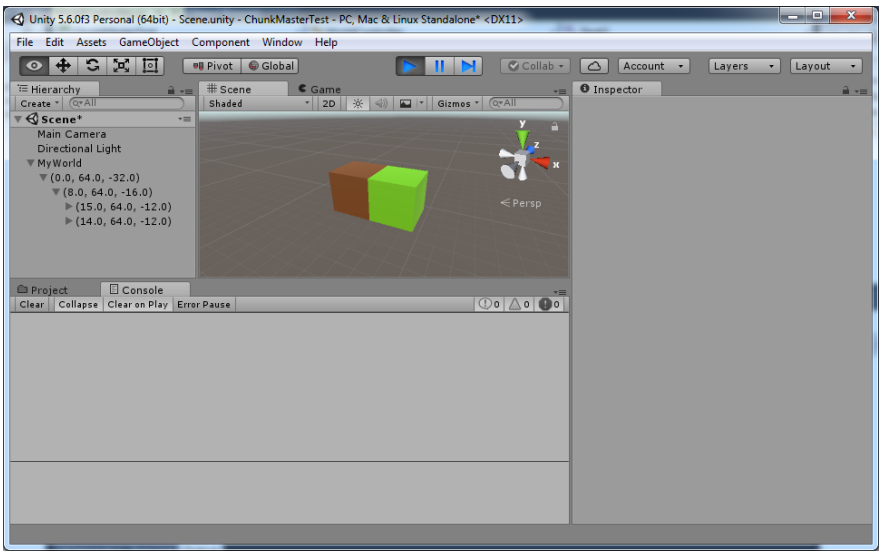

Figure - 10

Your code should look like the code snippet below.

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

using ChunkMaster.Unity.Framework;

public class WorldController : MonoBehaviour {

    public static UnityWorld myWorld;

    public Transform[] blocks;

    // Use this for initialization
    void Start ()
    {
        // Creating a myWorld instance object
        myWorld = new UnityWorld(transform);

        // Declaring a random point in space
        Vector3 worldPoint = new Vector3(15.4f, 64.4f, -12f);

        // Loading a sector
        myWorld.LoadSector(worldPoint, "E:\\ChunkMasterTutoria

        // Retrieving your block
        UnityBlock block = myWorld.GetBlock(worldPoint);

        // Set a dirt block to the left side of our grass blo
        UnityBlock dirtBlock = myWorld.SetBlock(worldPoint + \

        // Spawn your block
        SpawnBlock(block);

        // Spawn dirt block
        SpawnBlock(dirtBlock);
    }

    // Spawn Block
    public void SpawnBlock(UnityBlock block)
    {
        // The block's world position.
        Vector3 worldPosition = block.worldPosition;

        // The block's content.
        int blockIndex = (int)block.GetContent();

        // The block's parent object.
        Transform parent = block.gameObject.transform;

        // Spawn block.
        SpawnPrefab(blocks[blockIndex], worldPosition, parent
    }

    // Spawns a prefab
    public void SpawnPrefab(Transform prefab,
        Vector3 worldPosition,
        Transform parent)
    {
```

```
            Transform tmp = Instantiate(prefab);
            tmp.position = worldPosition;
            tmp.SetParent(parent);
        }
    }
```

I Completed This Section

Up Next Rendering a World->

Developed by Corey St-Jacques
Questions please contact Corey_stjacques@hotmail.com