ChunkMaster

Manipulating Blocks

Index

---

1. Adding Blocks

a) The SetBlock method

Now that we've successfully created our world, we can now start adding blocks. This can be achieved by using the SetBlock method. Below we have an example on how to set a block in world space.

```
Vector3 worldPoint = new Vector3(15.4f, 64.4f, -12f);
myWorld.SetBlock(worldPoint, 0);
```

You'll notice that the SetBlock method requires two parameters, a point in world space as a Vector3 and also any variable of your choice. In the example above we used a simple integer value zero. Keep note that any value can be placed inside the second parameter. See below for more examples.

```
myWorld.SetBlock(worldPoint, 0);
myWorld.SetBlock(worldPoint, "grass");
myWorld.SetBlock(worldPoint, 12f);
myWorld.SetBlock(worldPoint, true);
myWorld.SetBlock(worldPoint, 2.2);
```

As you can see you may input any value you like as a second parameter.

Congratulations! You've successfully set a block at Vector3(15.4f, 64.4f, -12f) in world space with a value of zero!

See the figure below for results in the unity inspector.

See Figure - 1

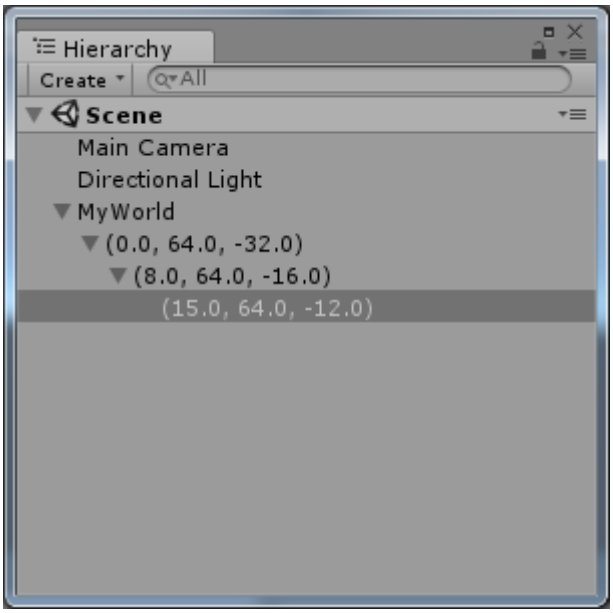

Figure - 1

You can see in the unity results that our MyWorld object now has some children created. The first child is what we like to call a "Sector", the second child is a "Chunk", and the third child is the block itself. See the diagram below to see the world hierarchy.

See Figure - 2

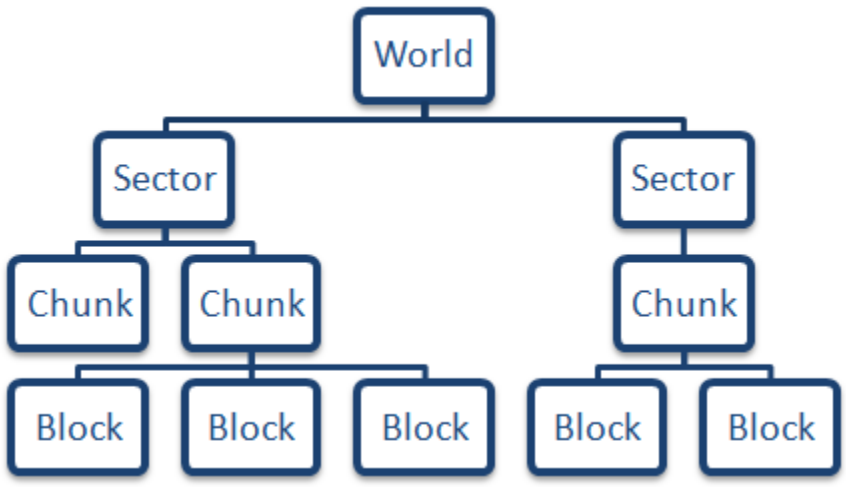

Figure - 2

Below is what your code should look like.

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

using ChunkMaster.Unity.Framework;

public class WorldController : MonoBehaviour {

    public static UnityWorld myWorld;

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

        Vector3 worldPoint = new Vector3(15.4f, 64.4f, -12f);

        myWorld.SetBlock(worldPoint, 0);
    }
}
```

I Completed This Section

---

2. Removing Blocks

a) The ClearBlock method

Removing a block is even easier than setting a block. You can remove a block by using the ClearBlock method. Using the a point in world space to remove the desired block.

```
        myWorld.ClearBlock(worldPoint);
```

Nothing too complicated here. Below is what your unity inspector should look like.

See Figure - 3

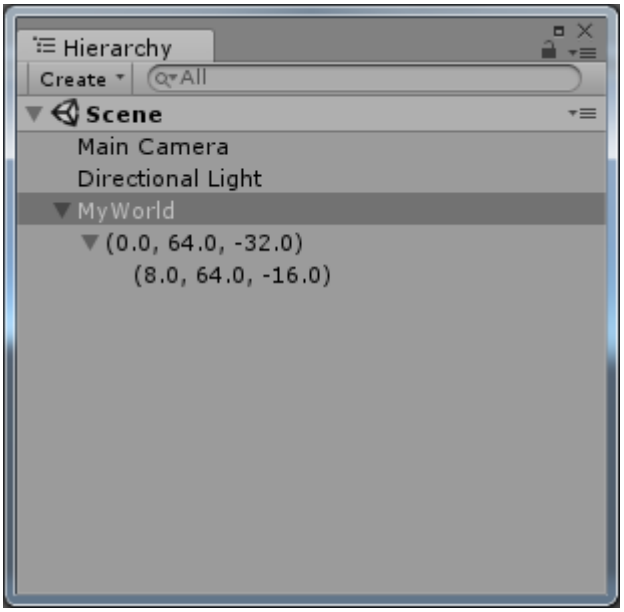

Figure - 3

As you can see in the figure above, we have successfully removed the block we've placed in our world. Below we have our current code.

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

using ChunkMaster.Unity.Framework;

public class WorldController : MonoBehaviour {

    public static UnityWorld myWorld;

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

        Vector3 worldPoint = new Vector3(15.4f, 64.4f, -12f);

        myWorld.SetBlock(worldPoint, 0);

        myWorld.ClearBlock(worldPoint);
    }
}
```

I Completed This Section

3. Checking if a space is occupied

a) The IsOccupied method

At some point in your code you will want to check and see if there is a block at a certain coordinate. You can find out if a block is occupied with another block by using the IsOccupied method. Like the other methods, this one also requires a point in world space.

```
myWorld.IsOccupied(worldPoint);
```

The code above will return a boolean value. In our case the result should be false, since we have removed the previous set block. Lets print this statement and see the results in unity.

```
print(myWorld.IsOccupied(worldPoint));
```

See Figure - 4

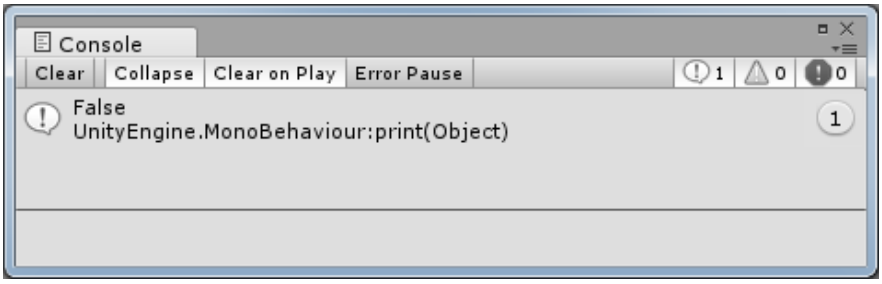

Figure - 4

As expected, we have a false value, Is the space occupied? false/Nay.

Below is what your code should look like by this point.

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

using ChunkMaster.Unity.Framework;

public class WorldController : MonoBehaviour {

    public static UnityWorld myWorld;

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

```
    {
        myWorld = new UnityWorld(transform);

        Vector3 worldPoint = new Vector3(15.4f, 64.4f, -12f);

        myWorld.SetBlock(worldPoint, 0);

        myWorld.ClearBlock(worldPoint);

        print(myWorld.IsOccupied(worldPoint));
    }
}
```

I Completed This Section

---

4. Getting a block

a) The GetBlock method

Every block in the ChunkMaster framework uses a data type of "UnityBlock". You can retrieve a block by using the following code snippet.

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

Now that you've retrieved your block, you can retrieve many of it's data members such as it's location, gameObject, content etc... Continue reading for more!

b) The GetContent method

If you ever need to retrieve a block's information you can use the GetBlock method. Just like any other method you will need to specify the block location in world space. See the code snippet below.

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

Since we've initially set the block using an integer data type with a value of zero, now if we would like to retrieve the value, we will need to cast the output to the input data type, in our case we will cast to an integer using (int).

c) Retrieving the world location

You can retrieve a block's current location by using the following code.

```
// The block's world position.
Vector3 worldPosition = block.worldPosition;
```

Luckily for us, this data property returns as a Vector3, which means we don't need to cast or create a new Vector3 object in order to retrieve it's world position.

d) Retrieving the block's gameObject

Every block has an assigned gameObject. You can retrieve the block's gameObject by using the following code snippet.

```
// The block's gameObject.
GameObject blockGameObject= block.gameObject;
```

After all this, you are probably wondering what my code should look like before continuing? Well here below is what it should represent.

```
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);

        // 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;
    }
}
```

I Completed This Section

Up Next Saving & Loading->

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