Getting Started
Creating Your World
Manipulating Blocks
Saving & Loading
Rendering Blocks
Rendering a World
Special thanks

ChunkMaster

Rendering a World

Index

- Creating the SpawnChunk method
- Creating the SpawnSector method
- Having fun

1. Creating the SpawnChunk method

a) Understanding what chunks are

By this point you are probably wondering how you can spawn more than just a single block in space. Before we continue we will need to understand what a chunk actually is.

A chunk is a designated 3 dimensional indexed array of blocks. What does this mean? Well if we have have a chunk size of 8 blocks, we know that our chunk can contain 8 x 8 x 8 = 512 blocks in a single chunk. Sounds like a lot! Here is a visual representation of a chunk.

See Figure - 1

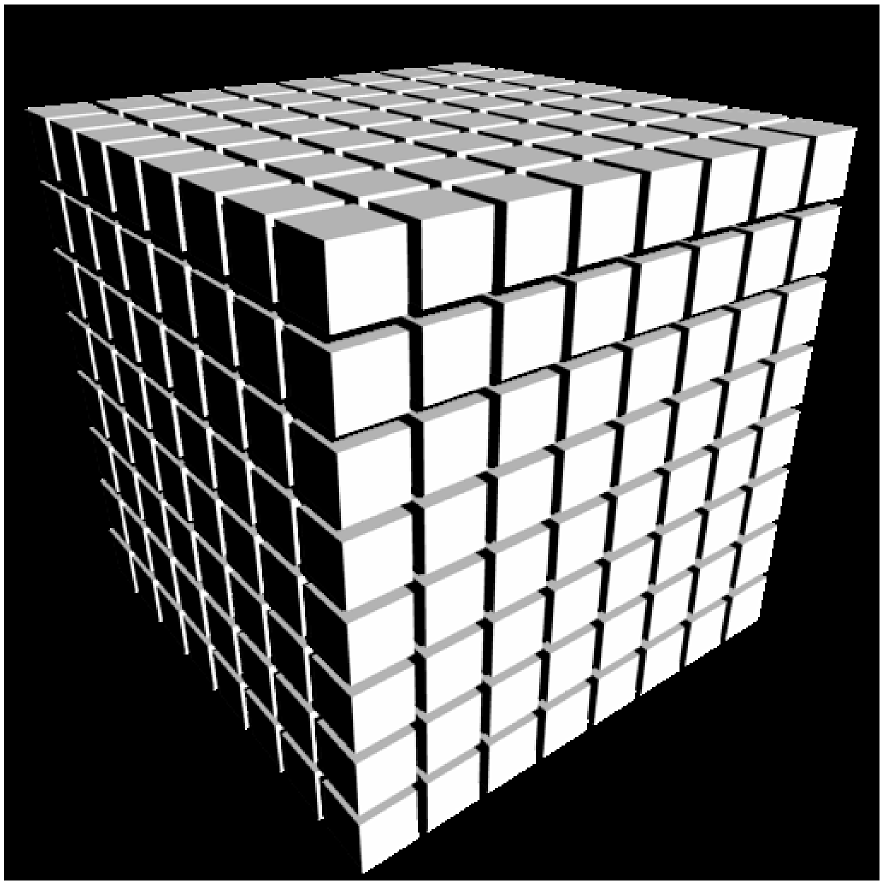

Figure - 1
512 blocks doesn't look like very much. Can you image what a sector looks like? We will see an illustration of this later on.

b) Retrieving a chunk

In order to spawn a chunk we will need to retrieve it. ChunkMaster has a data type "UnityChunk" and we can retrieve a chunk by using the following code snippet.

```
        // Retrieving a chunk
        UnityChunk chunk = myWorld.GetChunk(worldPoint);
```

As you can see we can retrieve a chunk with a single line of code, ignoring the comment above of course. The GetChunk method retrieves a chunk of UnityChunk data type. Now that we have the chunk acquired successfully, we can now create the SpawnChunk method.

c) Creating the SpawnChunk method

The spawn chunk method will be a very simple one. What we will need to do is iterate through all the chunk's blocks. We can then spawn each block in the unity 3D space. Lets consider the method below.

```
    // Spawn chunk
    public void SpawnChunk(UnityChunk chunk)
    {
        foreach(UnityBlock block in chunk.blocks)
        {
            if (block != null)
                SpawnBlock(block);
        }
    }
```

Not what you were expecting right? How can this small method do so much? Well because we've already created the SpawnBlock method, SpawnPrefab method, and setup a lot of other statements, it is really easy for us to manipulate objects. This is the key in programming, every time you have a chance to make your life easier in the future, you should go for it.

Enough chatter, As you can see in the method above, we iterate through all of the chunk's blocks, we check if the block isn't null and finally we spawn it. The UnityChunk object contains an array of blocks which can easily be accessed through one loop. The blocks are placed in a 1 dimensional array.

You can now use this method inside the start method. The result will essentially be the same as spawning our single block. 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;

    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:\\ChunkMasterTutori

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

        // Retrieving a chunk
        UnityChunk chunk = myWorld.GetChunk(worldPoint);

        // Spawn Chunk
        SpawnChunk(chunk);
    }

    // Spawn chunk
    public void SpawnChunk(UnityChunk chunk)
    {
        foreach(UnityBlock block in chunk.blocks)
        {
            if (block != null)
                SpawnBlock(block);
        }
    }

    // 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);
    }
}
```

Your result in the unity engine.

See Figure - 2

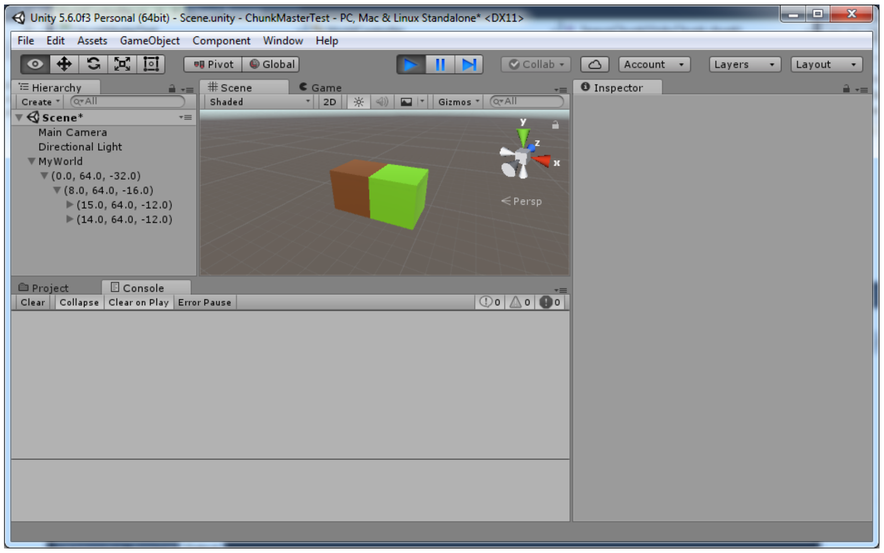
Unity 5.6.0f3 Personal (64bit) - Scene.unity - ChunkMasterTest - PC, Mac & Linux Standalone* <DX11>
File Edit Assets GameObject Component Window Help
Pivot Global
Collab Account Layers Layout
Hierarchy Scene Game Inspector
Create Shaded 2D Gizmos
Scene*
Main Camera
Directional Light
MyWorld
(0.0, 64.0, -32.0)
(8.0, 64.0, -16.0)
(15.0, 64.0, -12.0)
(14.0, 64.0, -12.0)
Persp
Project Console
Clear Collapse Clear on Play Error Pause

Figure - 2

I Completed This Section

2. Creating the SpawnSector method

a) Understanding what a sector actually is

A sector is what your world is composed of. A sector can contain chunks. Each sector has a size of 4, which in tern is how many chunks a sector can store. In a 3 dimensional sense, a sector can store 4 x 4 x 4 = 64 chunks. How many blocks can a sector hold? Consider the following equation: 4 x 4 x 4 x 8 x 8 x 8 = 32,768 blocks. Now that is a lot of data.... Here is what 32k blocks looks like. I hope my computer can handle it....

See Figure - 3

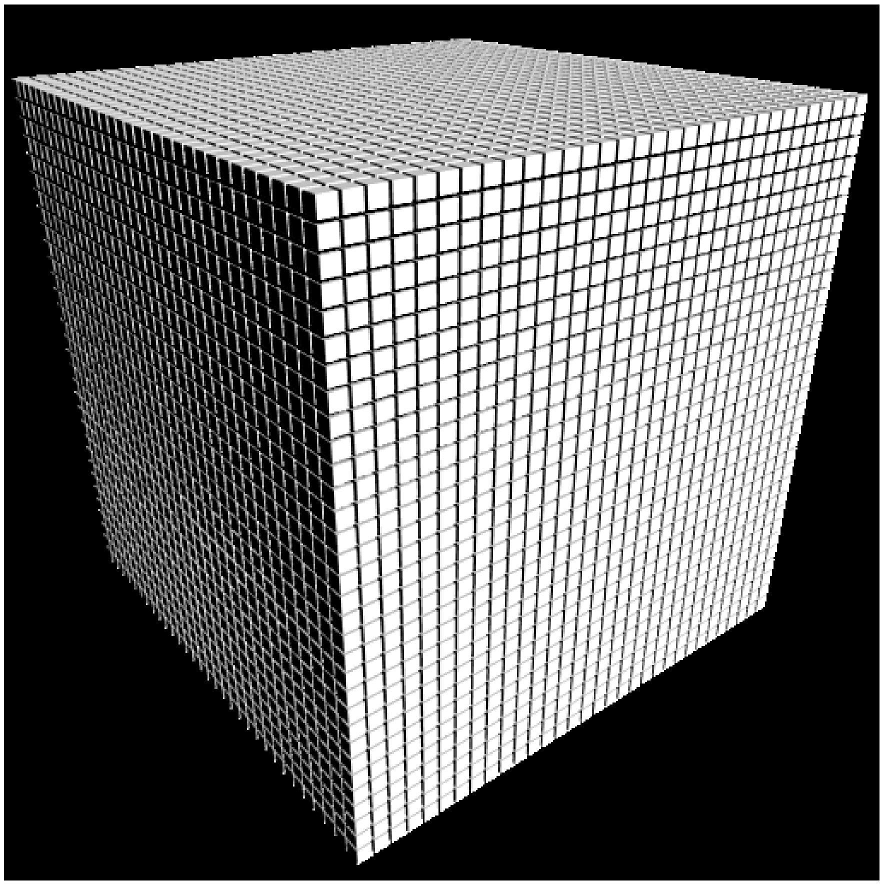

Figure - 3
Actually that wasn't so bad. We can render 32k blocks without any issues, just make sure to weld or combine all your meshes into a single object when manipulating a large number of cubes.

b) The GetSector method

If you ever need to retrieve a sector, you can use the following method to get a sector.

```
        // Retrieving a sector
        UnitySector sector = myWorld.GetSector(worldPoint);
```

There are no surprises here, we have a UnitySector data type that we assign using the GetSector method given a point in world space.

c) Creating the SpawnSector method

Next we will want to spawn a sector, this should look the same as our SpawnChunk method, we will basically copy paste the

SpawnChunk method and make a few changes. Here is what the method should look like.

```
    // Spawn sector
    public void SpawnSector(UnitySector sector)
    {
        foreach (UnityChunk chunk in sector.chunks)
        {
            if (chunk != null)
                SpawnChunk(chunk);
        }
    }
```

Here we iterate through all the chunks associated to the sector object. A sector has data member named chunks, which just like the chunk.blocks you can iterate through each chunk in a sequential search. The chunks data member is a 1 dimensional array which means you can easily iterate through each chunk using one single loop. Using our previously make SpawnChunk method we can spawn each chunk easily.

Finally we can Use what we've learned to spawn a sector in the start method.

```
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:\\ChunkMasterTutori

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

        // Retrieving a sector
        UnitySector sector = myWorld.GetSector(worldPoint);
```

```
            // Spawn Sector
            SpawnSector(sector);
        }

        // Spawn chunk
        public void SpawnChunk(UnityChunk chunk)
        {
            foreach(UnityBlock block in chunk.blocks)
            {
                if (block != null)
                    SpawnBlock(block);
            }
        }

        // Spawn sector
        public void SpawnSector(UnitySector sector)
        {
            foreach (UnityChunk chunk in sector.chunks)
            {
                if (chunk != null)
                    SpawnChunk(chunk);
            }
        }

        // 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);
        }
    }
```

And the result in the unity engine.

See Figure - 4

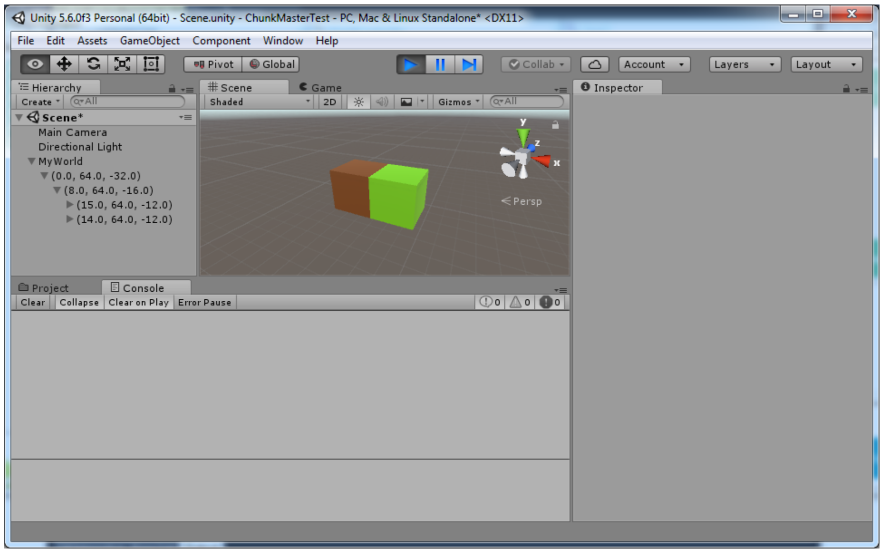
Unity 5.6.0f3 Personal (64bit) - Scene.unity - ChunkMasterTest - PC, Mac & Linux Standalone* <DX11>
File Edit Assets GameObject Component Window Help
Pivot Global
Collab Account Layers Layout
Hierarchy
Create
Scene*
Main Camera
Directional Light
MyWorld
(0.0, 64.0, -32.0)
(8.0, 64.0, -16.0)
(15.0, 64.0, -12.0)
(14.0, 64.0, -12.0)
Scene Game
Shaded 2D Gizmos
Persp
Inspector
Project Console
Clear Collapse Clear on Play Error Pause

Figure - 4

It looks like I keep copying the same picture over and over, but no this is the result! We only have two blocks placed, so we will be spawning the same two blocks every time.

I Completed This Section

3. Having fun

a) Spawning random cubes

Lets make things interesting and spawn many blocks at different locations just for the fun of it. Lets create a method called PopulateWorld. This method will generate a random number of blocks at random locations, we will then render these blocks.

See Figure - 5

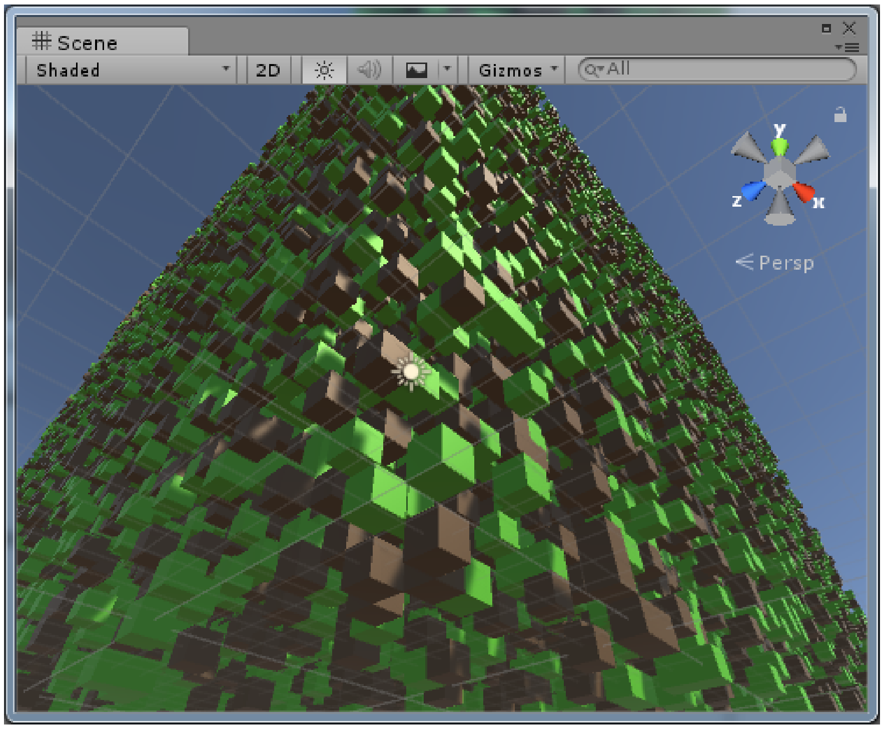

Figure - 5

Well this is impressive. 100 x 100 x 100 = 1,000,000 blocks, which is generated using the following code snippet.

```
// Populate with random blocks
public void PopulateWorld()
{
    int size = 100;
    Vector3 worldPosition;
    for (int x = 0; x < size; x++)
    {
        for (int y = 0; y < size; y++)
        {
            for (int z = 0; z < size; z++)
            {
                worldPosition = new Vector3(x, y, z);
                SpawnRandomBlock(worldPosition);
            }
        }
    }
}

// Spawn a random block.
public void SpawnRandomBlock(Vector3 worldPosition)
{
    UnityBlock block;
    if (Random.Range(0, 1f) < 0.1f)
    {
        block = myWorld.SetBlock(worldPosition,
            Random.Range(0, blocks.Length));
        SpawnBlock(block);
    }
}
```

We have two methods here. One method iterates through a million positions, and the second method generates a random cube, whether it's empty, grass, or dirt.

Our final code snippet is as follows.

```
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);
        PopulateWorld();
    }

    // Populate with random blocks
    public void PopulateWorld()
    {
        int size = 100;
        Vector3 worldPosition;
        for (int x = 0; x < size; x++)
        {
            for (int y = 0; y < size; y++)
            {
                for (int z = 0; z < size; z++)
                {
                    worldPosition = new Vector3(x, y, z);
                    SpawnRandomBlock(worldPosition);
                }
            }
        }
    }

    // Spawn a random block.
    public void SpawnRandomBlock(Vector3 worldPosition)
    {
        UnityBlock block;
        if (Random.Range(0, 1f) < 0.1f)
        {
            block = myWorld.SetBlock(worldPosition,
                Random.Range(0, blocks.Length));
            SpawnBlock(block);
        }
    }

    // Spawn chunk
    public void SpawnChunk(UnityChunk chunk)
    {
        foreach(UnityBlock block in chunk.blocks)
```

```
            {
                if (block != null)
                    SpawnBlock(block);
            }
        }

        // Spawn sector
        public void SpawnSector(UnitySector sector)
        {
            foreach (UnityChunk chunk in sector.chunks)
            {
                if (chunk != null)
                    SpawnChunk(chunk);
            }
        }

        // 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 Special thanks->

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