Exploding Cube

1. Create a floor

First start off by creating a Plane for the cube to lay on, make sure to name the Plane as floor.

Create a material and add it to the Floor to make it stand out.

Afterwards add a **boxcolloider** to the floor

1. Create a cube

Add a 3D Cube and position it high in the air and on position X: 0 and z: 0.

Then add a **rigidbody** to the cube.

Afterwards add a boxcollider and **enable IsTrigger** in the boxcollider component**.**

1. Vanishing Object

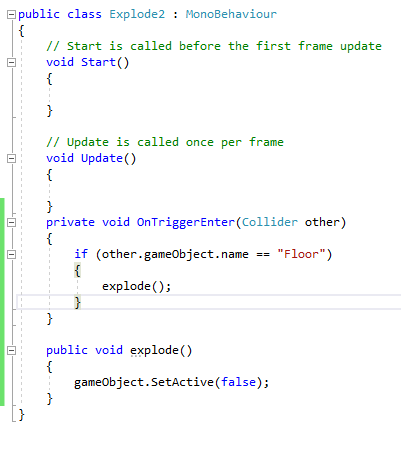
Add a new C# script and call it “**Explosion**” then add it to the Cube by dragging it onto the Cube.

Afterwards open the script; in that script you will create a private void function under the update function, use an **OnTriggerEnter** function for detection with trigger collider.

Inside it create a function called “**explode**” then create a new function outside of the OnTriggerEnter and make it a public void with the function explode which we created previously.

Inside the public Variable make sure to disable the game object by setting the “**SetActive**” as **False** this will ensure that when the object is triggered by the collider it will then disappear.

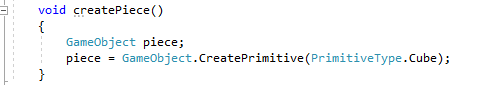
Lastly you want to specify that the object is destroyed only when it touched the object called floor so you need to write inside of the OnTriggerEnter “**if (other.gameObject.name == “Floor”)**” and afterwards put the “**explode**” function inside that.

The end script should look like this:

1. Spawning Small Cube

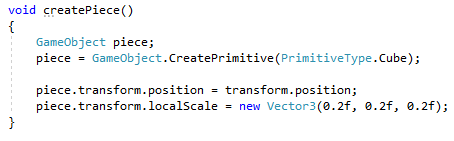
Create a new variable labelled “**createPiece** inside that function we will set it so the small cube spawns by creating a game object called “**Piece”** and underneath it we will define so piece is equal to creating a primitive cube, which the code will look like this: “**piece = GameObject.CreatePrimitive(PrimitiveType.Cube);**”

Create Primitive means is that it creates a new 3D object with all the properties it requires, such as Mesh and Boxcollider.

It will look like this:

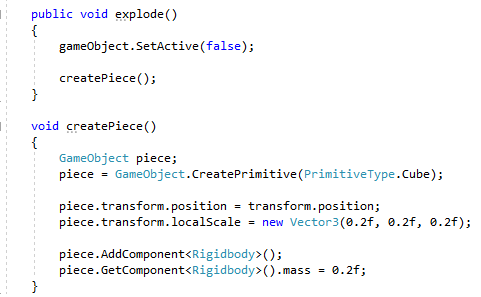
1. Defining small cube size and weight

First we will define position and scale the first will be a **transform position** for the piece which will equal to **transform.position**.

Underneath that one write down the scale which will be a **new Vector3** equalling every axis to **0.2f**, it will look like this:

Second we will add the **rigidbody** and set a mass to the piece, we first use **AddComponent** to add the rigid body and afterwards we use the **GetComponent** to access the rigidbody and create a mass of 0.2f.

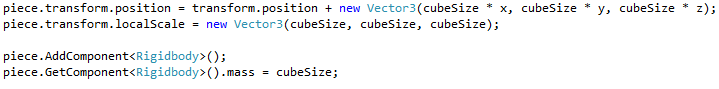
After finalizing this, we will call our createPiece function from the explode function, which will look like this.



1. Spawning several Cubes

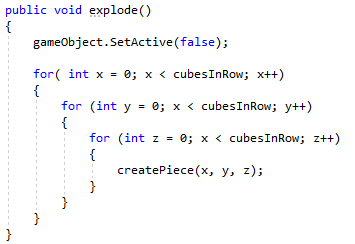
At the start of our scrip create two public variables, one **float** type called **cubeSize** with a value of 0.2f and an **integer** called **CubesInRow** which is equal to 5.

In the area create three dimensional cubes by adding **Int** for all the three dimensions inside of the **void function**, as such.

Next modify the transform position functions inside the **createPiece** by adding a **new Vector3** function to the transform variable and **multiplying** the **cubeSize** times every axis and in addition replace every **0.2f** to **cubeSize**, as demonstrated here.

1. Spawning cubes in a 5x5x5 area

Remove the createPiece function and replace it with 3 different loops for each coordinate, this will define the Int value for each axis and make it so that when the CubesInRow exceed the specified axis then it will stop looping, in total this will happen 3 times creating 125 pieces.



1. Adjusting Pivot

Start by creating variables to calculate pivot, create a new vector3 and call it cubesPivot.

Afterwards add a float variable called cubesPivotDistance.

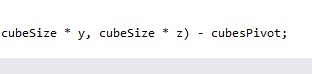
Then inside the start function start by calculating the pivot distance, multiply the size of the cube times the cubesInRow divided by two.



Below the function, this one will be used for creating the pivot Vector, instead of putting the x, y and z values they will be replaced with the pivot distance of the cube, as such.



Then subtract the cubesPivot from the piece.transform.position.



1. Explosion Force

Now we will add the explosion force, first inside the explode function we create a new vector3 called explosionForce and equal it to transform.position



Afterwards add a collider list and call it colliders, add a physics.overlapsphere, inside the parenthesis specify the position of the explosion and the radius.



Afterwards we will add an explosion force to all colliders in the overlapsphere by creating a foreach which checks for colliders hitting colliders.



Then we will retrieve the rigidbody from the collider objects by doing a simple get component line of code as such.



And lastly we will add the force to the explosion by setting rigidbody to affect the addExplosionForce, then inside the parenthesis add explosionForce, transform.position, explosionRadius and explosionUpward.



If errors appear for explosionForce, explosionRadius and explosionUpward then just specify them as public variables at the start of the script.

1. End of Tutorial