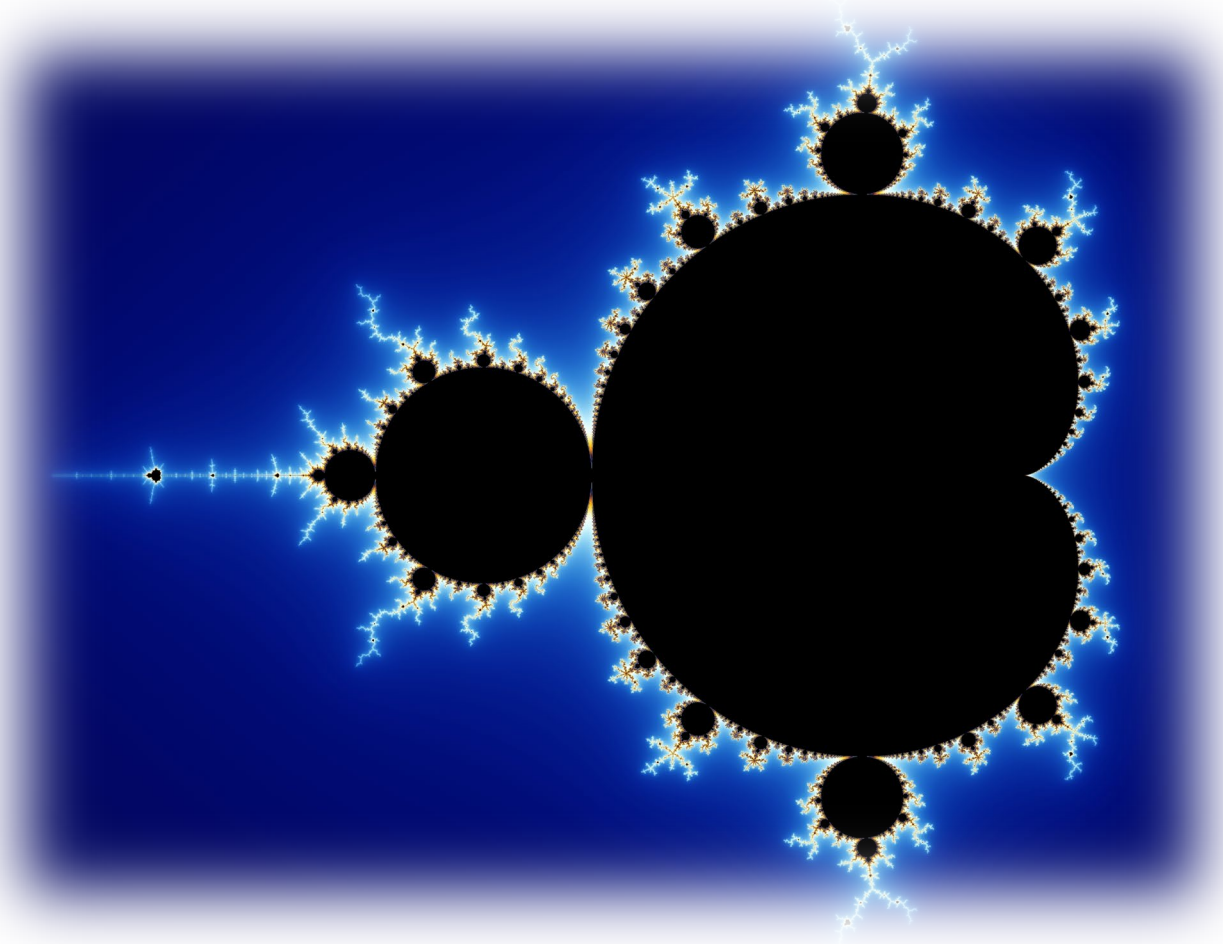


Creating GameObjects on the fly

Using prefabs

Using code to spawn prefabs

Prefabs started in array

Prefab ID's as enums

Using a loop to spawn any number

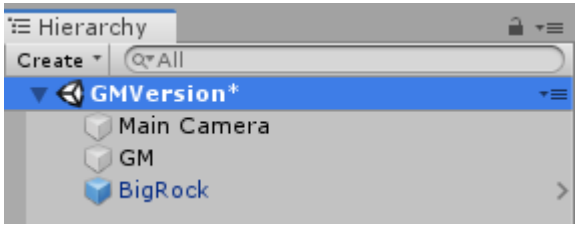
```
[SerializeField]
private GameObject[] SpawnablePrefabs; //Prefabs we will be able to spawn

public enum SpawnIDs { //Just makes up nice labels starting from 0
    Player      //0
    , AsteriodBig //1
}
```

```
void CreatePlayer() {
    SpawnPrefab(SpawnIDs.Player, RandomOnScreenPosition(), 0); //Show player
}


void CreateAsteroids(int vCount) {
    while(vCount-- !=0) { //Make as many asteroids as needed
        SpawnPrefab(SpawnIDs.AsteriodBig, RandomOnScreenPosition(), Random.Range(0,360)); //Show asteroid with random rotation
    }
}
```

Reminder

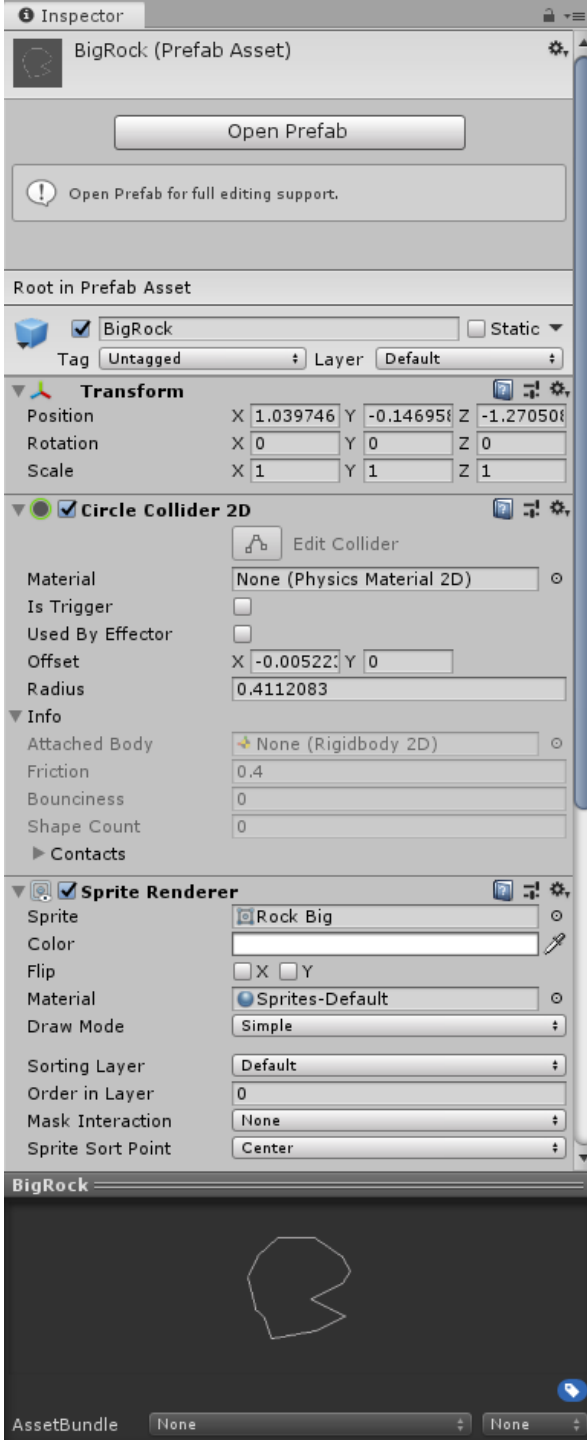
A prefab is a game object template, it needs to be added to a scene either in the IDE or with code

Prefabs have a blue tint



If you edit the prefab in the Scene you are editing the instance, not the template

If you edit the prefab from the asset window you will be editing the template



Making a bullet

FakePhysics based

Single pixel

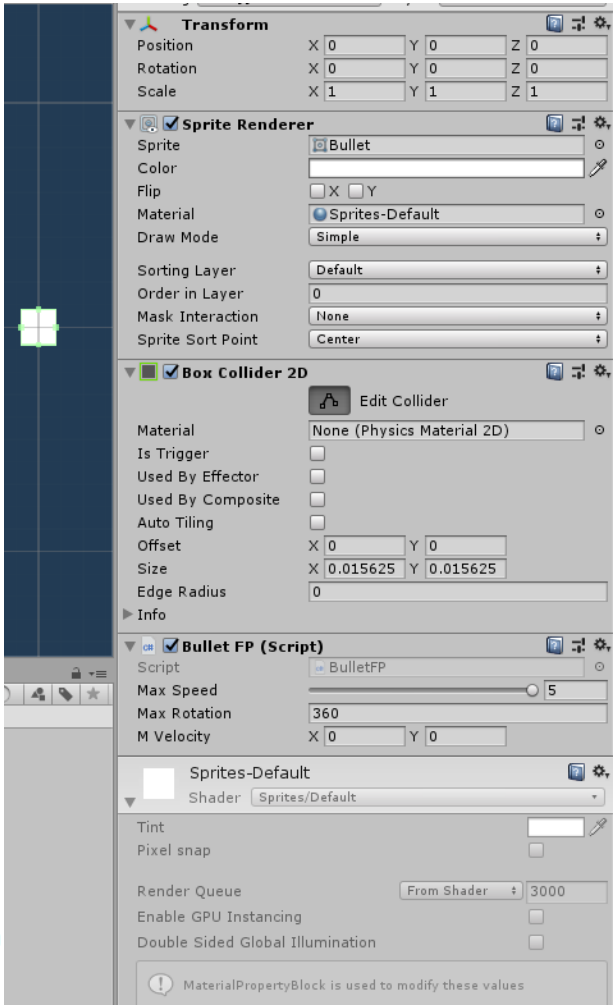
Box collider

BulletFP.cs script

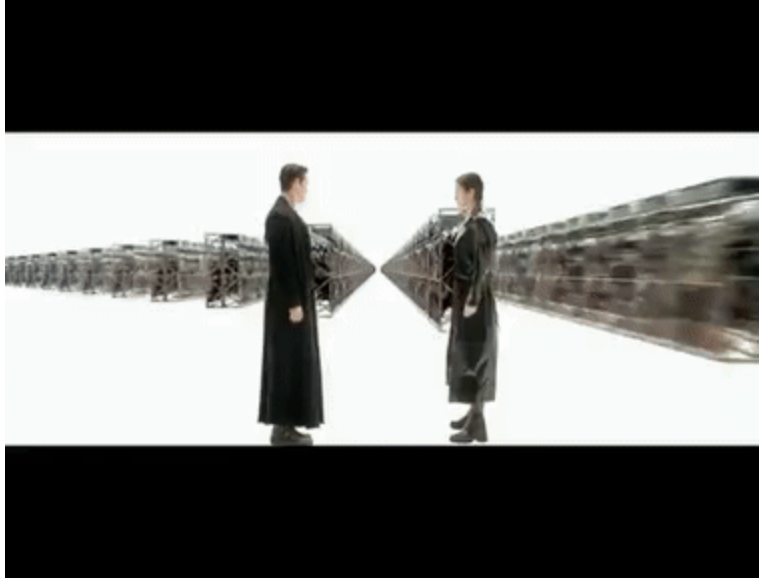
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class BulletFP : FakePhysics
{
    protected override void Start() { //Start Bullet
        base.Start();
        Destroy(gameObject, 2.0f);      //Destroy bullet after 2 seconds
    }
}
```

However we also need a change to FakePhysics.cs as we need to set the velocity from the Gun script

```
public Vector2 mVelocity = Vector2.zero; //Now public as we need to use it in Gun
```



We needs guns

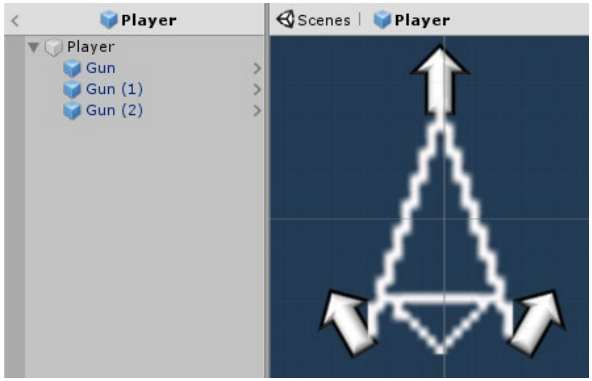


Maybe even lots of guns?

Why make one gun when you can make many?

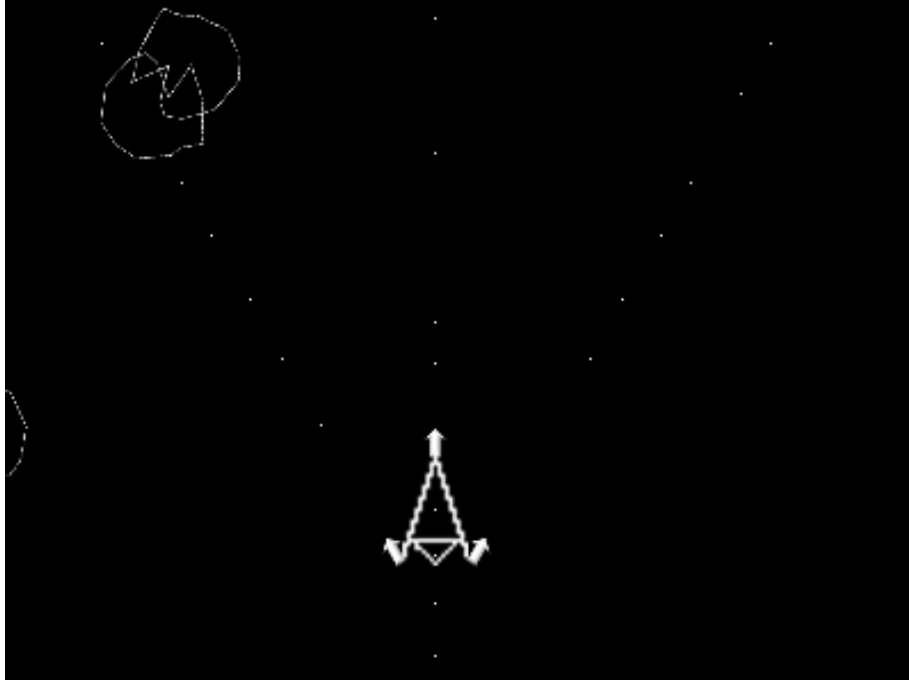
The gun itself does not really exist, its just a point where the bullet spawns

Empty GO with a Gun.cs script , except to make it show up I added a sprite




```
public class Gun : MonoBehaviour
{
    [SerializeField]
    float BulletSpeed=5.0f;
    void Update()
    {
        if(Input.GetButton("Fire1")) {
            BulletFP tBullet = GM.SpawnPrefab(GM.SpawnIDs.Bullet, transform.position, 0).GetComponent<BulletFP>();
            FakePhysics tParentFP = GetComponentInParent<FakePhysics>();
            tBullet.mVelocity = (Vector2)transform.up * BulletSpeed;  //Initial Velocity
            tBullet.mVelocity += tParentFP.mVelocity; //Player ship velocity
        }
    }
}
```

Result



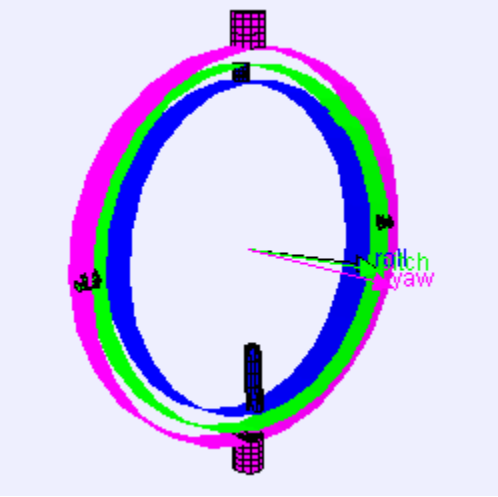
As many guns as we want and they can be rotated and placed anywhere on the ship

The trick with the bullet is to calculate its initial velocity

- Player ship + Direction of gun * Bullet Speed
- We also need the bullet to have a TTL (Time to live) which is easy as we can override Start() from FakePhysics to add this

```
public class BulletFP : FakePhysics
{
    protected override void Start() { //Start Bullet
        base.Start();
        Destroy(gameObject, 2.0f);      //Destroy bullet after 2 seconds
    }
}
```

Rotation interlude



We are used to Euler angles such as Rotate(0,0,15), which is a rotation around the Z axis by 15 degrees

Whilst we see Rotations in Unity as (X,Y,Z), internally they are stored and calculated as Quaternion.

This avoids GimbalLock, you cannot just look at a quaternion to see what the rotation of an object is as it's a 4D object with complex number components. NB to add Quaternions you multiply them

```
//Rock's own movement
protected override void DoMove() {
    Quaternion tRotation = Quaternion.Euler(0, 0, -mRotation * MaxRotation * Time.deltaTime);
    transform.rotation *= tRotation; //Using Quartonians for rotation
    base.DoMove();
}
```

By Lookang many thanks to Fu-Kwun Hwang and author of Easy Java Simulation = Francisco Esquembre - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=15837410

Casting

Sometimes you may wish to turn one class into another such as;

- Vector2 to Vector3
- FakePhysics to PlayerFP

You can do this by using a cast which is

WhatIWantClass tNewClass = (WhatIWantClass)otherClass

The () is the cast and it tries to cast BUT THIS MAY NOT WORK if there is no valid conversion

```
transform.position = (Vector3)tRotation;
```

(local variable) Quaternion tRotation

Cannot convert type 'UnityEngine.Quaternion' to 'UnityEngine.Vector3'

You can up (parent to child) & downcast (child to parent)

Shortcuts & code explanation

- transform.rotation *= tRotation; //Using Quartonians for rotation
  - *= is the same as writing transform.rotation = transform.rotation * tRotation;

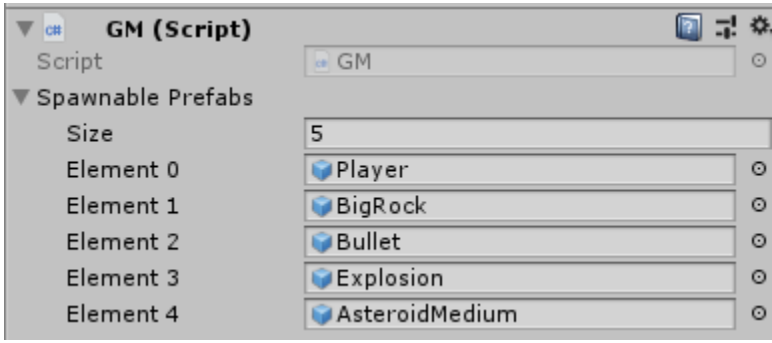
- Vector2.up (0,1,0) but will rotate if object is rotated, for 2D up is used in place of forward as forward is (0,0,1) and meant for 3D only

- Vector2 tAt45Degrees = Quaternion.Euler(0,0,45) * Vector2.up;
  - Make a Vector at 45 Degrees

- Checking for a class being a certain type

```
if (vOtherFF is BulletFP) {  //Were we hit be a bullet
}
```

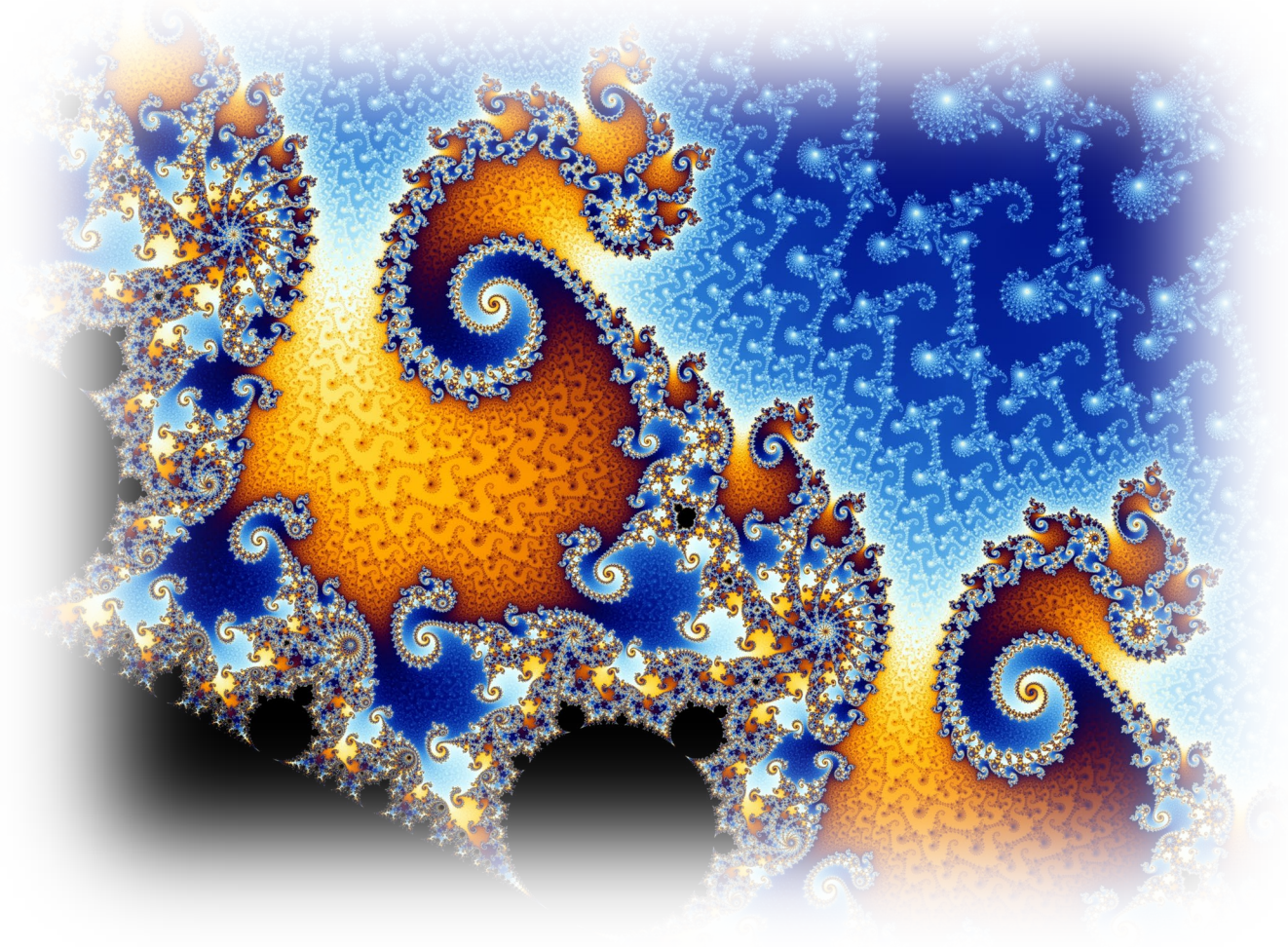
Dealing with collisions



- Big Rock explodes into 2 Medium ones & dies

```
protected override void CollidedWith(FakePhysics vOtherFF) {
    //We do not call parent as we will handle
    if (vOtherFF is BulletFP) {  //Were we hit by a bullet
        Destroy(gameObject);    //Destroy Asteroid
        Destroy(vOtherFF.gameObject);    //Destroy Bullet
        GM.SpawnPrefab(GM.SpawnIDs.Explosion, transform.position, Random.Range(-180, 180));// Exposion
        GM.SpawnPrefab(GM.SpawnIDs.AsteroidMedium, transform.position, Random.Range(-180, 180)); //Medium Rock
        GM.SpawnPrefab(GM.SpawnIDs.AsteroidMedium, transform.position, Random.Range(-180, 180)); //Medium Rock
    }
}
```

- Medium Rock explodes into 3 Small ones & dies
- Small rock dies



Workshop

Adding the other asteroids

1. Grab the explosion prefab & add to you project
2. Review the Gist https://tinyurl.com/yxjfgote & make changes to you code
3. If stuck Grab the New player Prefab, which has the guns
4. If really stuck grab the entire project https://github.com/RLTeachGit/UnityRocksOOP
5. Using the BigRock as a guide add code to make the Medium and Small Rocks work