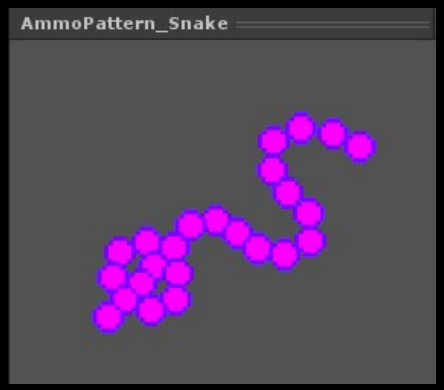
Enemy Ammo Pattern Concepts

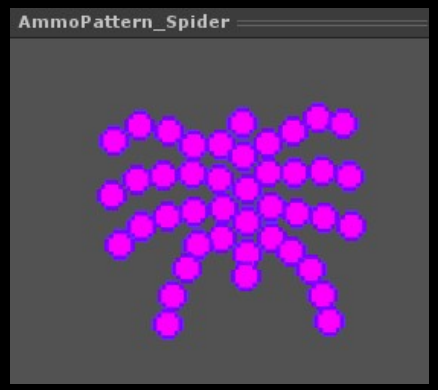
Enemy Ammo Pattern Concepts

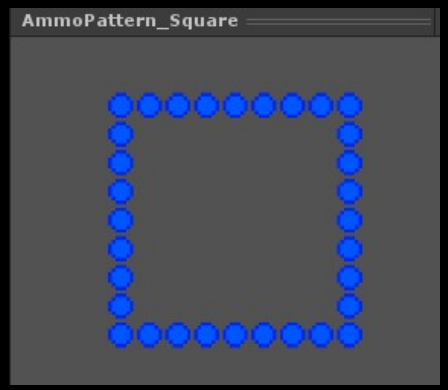
What Is An Ammo Pattern?

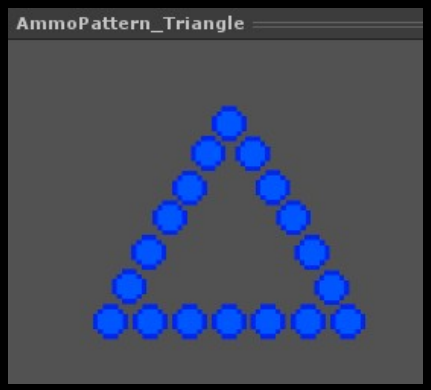
So for some of the enemies, for example Slizzard and Hedusa, rather than firing a weapon, they will fire patterns of ammo.

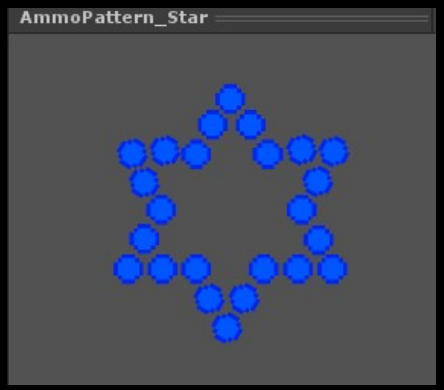
An ammo pattern is a prefab that can be ‘fired’ that is made up of multiple instances of regular ‘Ammo’ but arranged in a pattern.











Enemy Ammo Pattern Concepts

What Is An Ammo Pattern?

Enemy Ammo Pattern Concepts

What Is An Ammo Pattern?

So how are we going to implement ammo patterns?

This isn't going to be too difficult, since when we designed the weapon firing system we used an interface call 'IFireable'.

As long as we get the ammo patterns to conform to the IFireable interface we should be able to implement them within our existing weapon and ammo systems!

Enemy Ammo Pattern Concepts

IFireable Interface

When we designed the weapon system, we used an interface called IFireable that Ammo classes have to implement. This interface contains a method called 'InitialiseAmmo', which Ammo classes need to implement. At the moment the 'Ammo.cs' class is the only class that implements the IFireable interface.

```
using UnityEngine;

4 references
public interface IFireable
{
    4 references
    void InitialiseAmmo(AmmoDetailsSO ammoDetails, float aimAngle, float weaponAimAngle,
      float ammoSpeed, Vector3 weaponAimDirectionVector, bool overrideAmmoMovement = false);

    2 references
    GameObject GetGameObject();

}
```

Enemy Ammo Pattern Concepts

FireWeapon, Ammo & IFireable

The ‘InitialiseAmmo’ method is used by the FireWeapon class to initialise ammo that it is about to fire.

It retrieves an Ammo component which implements IFireable from the object pool based on the specified ammoPrefab. Knowing that Ammo must implement the IFireable interface, the ‘InitialiseAmmo’ method can be called by the FireWeapon class to initialise the ammo.

The ‘InitialiseAmmo’ method retrieves the ammoDetails scriptable object, sets the ammo fire direction, sets the ammo sprite, sets the ammo material, sets the ammo range and sets the ammo speed.

FireWeapon.cs

Assembly-CSharp | FireWeapon | FireAmmoRoutine(AmmoDetailsSO currer

```
170            // Get Gameobject with IFireable component
171            IFireable ammo = (IFireable)
                 PoolManager.Instance.ReuseComponent(ammoPrefab,
                 activeWeapon.GetShootPosition(), Quaternion.identity);
172
173            // Initialise Ammo
174            ammo.InitialiseAmmo(currentAmmo, aimAngle, weaponAimAngle,
                 ammoSpeed, weaponAimDirectionVector);
```

Enemy Ammo Pattern Concepts

FireWeapon, Ammo & IFireable

The Ammo class is then responsible in it's update method to move itself in the initialised direction and speed until it reaches it's maximum range, at which point it disables itself.

Ammo.cs

Assembly-CSharp | Ammo | Update()

```
Unity Message | 0 references
private void Update()
{
    // Ammo charge effect
    if (ammoChargeTimer > 0f)
    {
        ammoChargeTimer -= Time.deltaTime;
        return;
    }
    else if (!isAmmoMaterialSet)
    {
        SetAmmoMaterial(ammoDetails.ammoMaterial);
        isAmmoMaterialSet = true;
    }

    // Don't move ammo if movement has been overriden - e.g. this ammo is part of an ammo pattern
    if (!overrideAmmoMovement)
    {
        // Calculate distance vector to move ammo
        Vector3 distanceVector = fireDirectionVector * ammoSpeed * Time.deltaTime;

        transform.position += distanceVector;

        // Disable after max range reached
        ammoRange -= distanceVector.magnitude;

        if (ammoRange < 0f)
        {
            if (ammoDetails.isPlayerAmmo)
            {
                // no multiplier
                StaticEventHandler.CallMultiplierEvent(false);
            }

            DisableAmmo();
        }
    }
}
```

Enemy Ammo Pattern Concepts

FireWeapon, Ammo & IFireable

The Ammo class also has the Physics2D OnTriggerEnter2D method which handles the Ammo collision and then disables itself.

Ammo.cs

Assembly-CSharp | Ammo

```
Unity Message | 0 references
private void OnTriggerEnter2D(Collider2D collision)
{
    // If already colliding with something return
    if (isColliding) return;

    // Deal Damage To Collision Object
    DealDamage(collision);

    // Show ammo hit effect
    AmmoHitEffect();

    DisableAmmo();
}
```

Enemy Ammo Pattern Concepts

Ammo Patterns

By using the IFireable interface it makes ammo patterns relatively easy to implement.

We are going to create another class called 'AmmoPattern' that is also going to implement the 'IFireable' interface. The AmmoPattern class will hold an array of regular Ammo components, that have been positioned in the ammo pattern prefab into the ammo pattern that we want.

The AmmoPattern will have an 'InitialiseAmmo' method which will initialise the pattern including the fire direction, speed, rotation speed, and range for the ammo pattern as a whole.

The AmmoPattern 'InitialiseAmmo' method will also iterate through all the individual Ammo components for the pattern held in the ammo array and initialise each one of them using it's own 'InitialiseAmmo' method – but passing in a flag to override the individual ammo movement – since that is being controlled by the ammo pattern.

Enemy Ammo Pattern Concepts

Ammo Patterns

So the Fire Weapon class will be able to fire ammo patterns without any amendments to it's code.

It can still retrieve an AmmoPattern component from the object pool since it also implements the IFireable interface. For ammo patterns the ammoPrefab will be an ammo pattern prefab rather than an ammo prefab (which will be specified in the ammo details scriptable object).

Also, since the ammo pattern implements the IFireable interface and will have it's own implementation of the 'InitialiseAmmo' method, the FireWeapon class can continue to call 'ammo.InitialiseAmmo' because this will work whether it's an Ammo component or an AmmoPattern component.

FireWeapon.cs

Assembly-CSharp | FireWeapon | FireAmmoRoutine(AmmoDetailsSO currer

```
170            // Get Gameobject with IFireable component
171            IFireable ammo = (IFireable)
                 PoolManager.Instance.ReuseComponent(ammoPrefab,
                 activeWeapon.GetShootPosition(), Quaternion.identity);
172
173            // Initialise Ammo
174            ammo.InitialiseAmmo(currentAmmo, aimAngle, weaponAimAngle,
                 ammoSpeed, weaponAimDirectionVector);
```

Enemy Ammo Pattern Concepts

Ammo Patterns

The end result will be an ammo pattern that can be fired by weapons like regular ammo.

The ammo pattern will comprise of individual ammo prefabs that move as a pattern, but whose individual elements collide and deal damage like individual ammo.