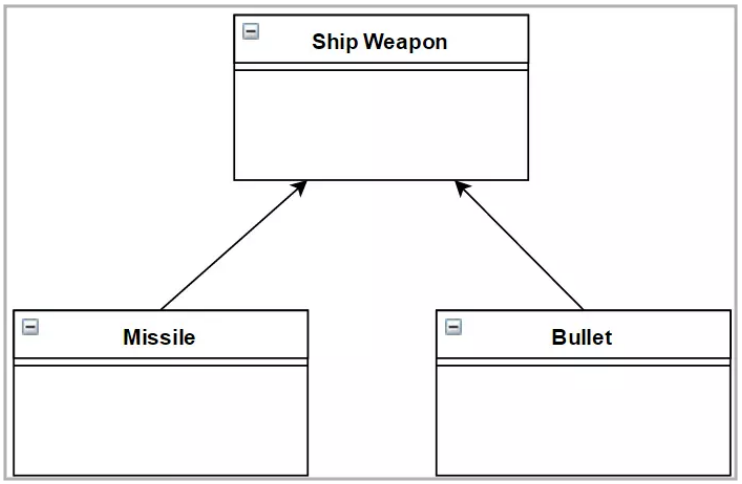
Strategy Pattern trong Unity

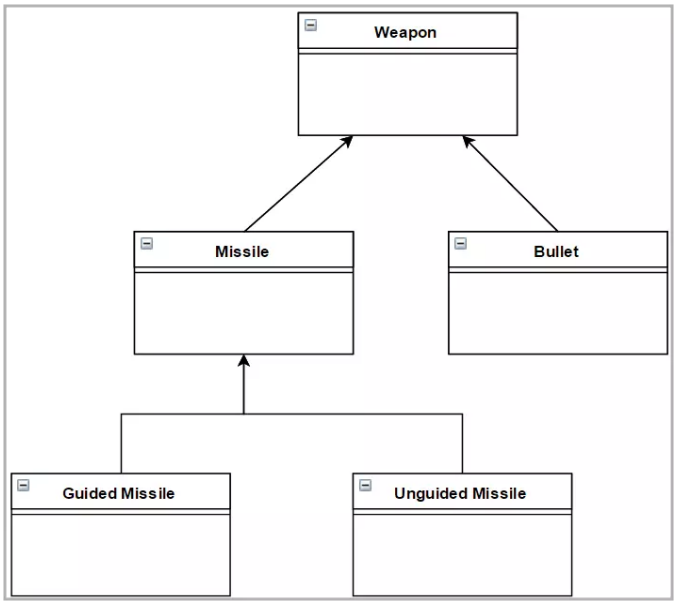
Chào các bạn. Hôm nay mình xin giới thiệu về Strategy Pattern trong Unity. Bài này ta sẽ giúp trả lời những câu hỏi:

Strategy Pattern là gì ?
Tại sao ta phải dùng Strategy Pattern và nó mang lại được gì cho việc dev game? Oke chúng ta bắt đầu vô bài.
Strategy Pattern là gì ? Đây là một behavioral pattern, định nghĩa một tập những thuật toán mà có thể chuyển đổi lẫn nhau để thực hiện một tác vụ cụ thể nào đó. Lý thuyết sơ qua vậy thôi, mình đưa vào ví dụ để dễ hiểu hơn. Bài này mình lấy ví dụ là mình có nhiều con tàu chiến mỗi tàu thì có từng loại vũ khí khác nhau: như boom, tên lửa, đạn,...

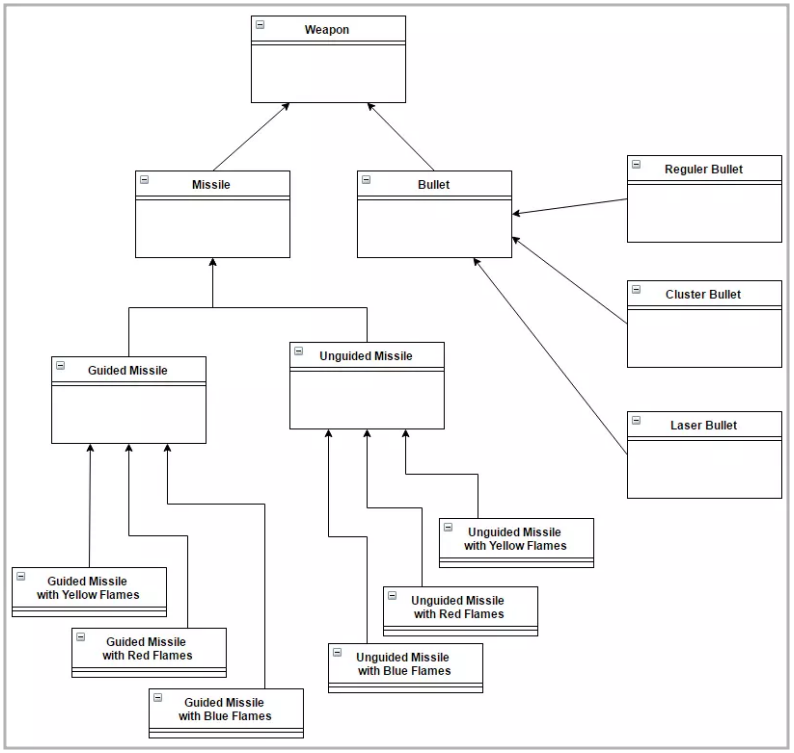
Tại sao ta phải dùng Strategy Pattern và nó mang lại được gì cho việc dev game? Như cách thông thường đó là dùng OOP dùng tính kế thừa ta sẽ có được sơ đồ



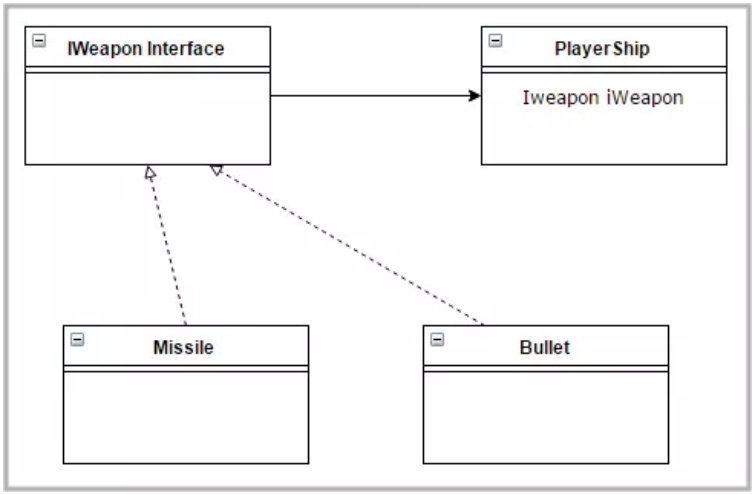
Rồi tiếp tục kế thừa thêm các loại vũ khí nữa



Và tiếp tục cho đến khi nó phì to như thế này



ta thấy ở hình 1,2 thì quản lý cho việc kế thừa này dường như vẫn còn dễ dàng. Nhưng đến hình cuối cùng thì đó đã là 1 vấn đề lớn còn cho ta phải dừng lại suy nghĩ có nên tiếp tục dùng cách kế thừa hay không. Chính vì vậy Strategy Pattern được đưa ra như 1 giải pháp giúp bạn giải quyết vấn đề này.



Vậy thì thay vì kế thừa thì ta tạo ra 1 interface cho vũ khí đó và nếu bạn muốn tạo 1 loại vũ khí mới cứ việc kế thừa từ interface này như hình. Ta sẽ khi phải bận tâm khi tạo ra rất rất nhiều loại đạn.

Bây giờ ta đi vào coding. - Tạo scene:Tạo 1 con tàu chiến trong không gian, với 2 loại đạn: tên lửa và đạn. - WeaponManager Script

```
using UnityEngine;
using System.Collections;
public class WeaponManager : MonoBehaviour {
}
```

```
public interface IWeapon{
   void Shoot();
}
public class Bullet: MonoBehaviour, IWeapon{
   public void Shoot(){

      Vector3 initialPosition = new Vector3 (this.transform.position.x, this.transform.position.y
+1f,0);
      GameObject bullet =  Instantiate(Resources.Load("BulletPrefab",typeof(GameObject)))
as GameObject;
      bullet.transform.position = initialPosition;
      bullet.GetComponent<Rigidbody2D>().velocity = new Vector2(0f,3f);
   }
}
public class Missile: MonoBehaviour, IWeapon{
   public void Shoot(){

      Vector3 initialPosition = new Vector3 (this.transform.position.x, this.transform.position.y
+ 1f,0);
      GameObject missile =
Instantiate(Resources.Load("MissilePrefab",typeof(GameObject))) as GameObject;
      missile.transform.position = initialPosition;
      missile.GetComponent<Rigidbody2D>().velocity = new Vector2(0f,3f);

   }
}
```

Interface IWeapon có phương thức Shoot() và có 2 kế thừa từ đó là 2 class Missile và Bullet để tùy chỉnh 2 hàm Shoot() khác nhau cho riêng nó.

FlameManager Script

```
using UnityEngine;
using System.Collections;
public class FlameManager : MonoBehaviour {
}
public interface IFlame{

   void ShowFlame();
}
public class BlueFlame:MonoBehaviour,IFlame{

   public void ShowFlame(){
      GameObject blueFlame =
Instantiate(Resources.Load("Flame-blue",typeof(GameObject))) as GameObject;
      blueFlame.transform.parent=transform;
   }
}
public class RedFlame:MonoBehaviour,IFlame{
```

```
   public void ShowFlame(){
      GameObject redFlame =
Instantiate(Resources.Load("Flame-red",typeof(GameObject))) as GameObject;
      redFlame.transform.parent=transform;
   }
}
```

Tương tự như WeaponManager Script có tùy chỉnh hàm ShowFlame() ở 2 class kế thừa.

ShipController Scritp

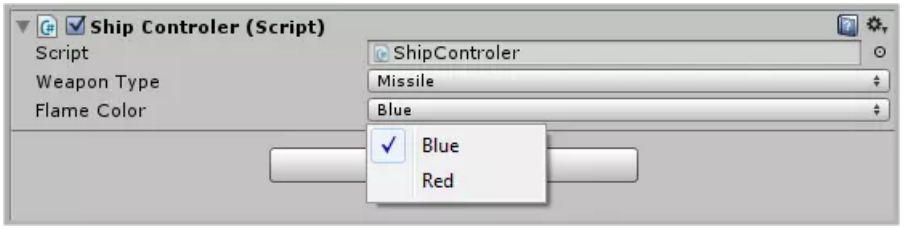
```
using UnityEngine;
using System.Collections;

public enum WeaponType{
        Missile,
        Bullet
}

public enum Flame{
        Blue,
        Red
}

public class ShipControler : MonoBehaviour {

        #region variables
        public WeaponType weaponType;
        public Flame flameColor;

        private IWeapon iWeapon;
        private IFlame iFlame;
        #endregion

        private void HandleWeaponType(){

                //To prevent Unity from creating multiple copies of the same component in
inspector at runtime
                Component c = gameObject.GetComponent<IWeapon>() as Component;

                if(c!=null){
                        Destroy(c);
                }

                #region Strategy
                switch(weaponType){

                        case WeaponType.Missile:
```

```
                iWeapon = gameObject.AddComponent<Missile> ();
                break;

            case WeaponType.Bullet:
                iWeapon = gameObject.AddComponent<Bullet> ();
                break;

            default:
                iWeapon = gameObject.AddComponent<Bullet> ();
                break;
        }
        #endregion
    }

    public void HandleFlameColor(){

        Component c = gameObject.GetComponent<IFlame>() as Component;

        if(c!=null){
            Destroy(c);
            iFlame.DestroyFlame(); // so that number of flame objects remains
one
        }

        #region Strategy
        switch(flameColor){

        case Flame.Blue:
            iFlame = gameObject.AddComponent<BlueFlame> ();
            break;

        case Flame.Red:
            iFlame = gameObject.AddComponent<RedFlame> ();
            break;

        default:
            iFlame = gameObject.AddComponent<BlueFlame> ();
            break;
        }
        #endregion

    }

    public void Fire(){
        iWeapon.Shoot();
    }

    void Start(){
```

```
            HandleWeaponType(); //to check the value of weaponType in the inspector initially
            HandleFlameColor();
            iFlame.ShowFlame();
        }

        void Update () {

            if (Input.GetKeyDown(KeyCode.Space)){
                Fire();
            }

            //to check the value of weaponType in the inspector while in play mode
            if(Input.GetKeyDown(KeyCode.C)){
                HandleWeaponType();
            }

            if (Input.GetKeyDown(KeyCode.F)){
                HandleFlameColor();
                iFlame.ShowFlame();
            }
        }
}
```

Bạn có thể thấy enum WeaponType và enum Flame cho mỗi loại mình cần để dựa vào đó chính ta có thể tạo sao các loại đạn và flame khác nhau.

Cuối cùng là add ShipControler vào từng con tàu đó.



Và ta được kết quả