РЕАЛИЗАЦИЯ СИСТЕМЫ УСИЛЕНИЙ С ИСПОЛЬЗОВАНИЕМ SCRIPTABLE OBJECT ПРИ РАЗРАБОТКЕ ИГР НА ДВИЖКЕ UNITY

Аверичев Никита Константинович

бакалавр, Дальневосточный Федеральный Университет, РФ, г. Владивосток

IMPLEMENTATION OF A POWER-UP SYSTEM USING SCRIPTABLE OBJECT WHEN DEVELOPING GAMES USING THE UNITY GAME ENGINE

Nikita Averichev

Bachelor of Science, Far Eastern Federal University, Russia, Vladivostok

Аннотация. При разработке компьютерных игр у разработчиков часто появляется потребность в создании системы усилений, с помощью которой игрок сможет увеличивать урон, здоровье или другие характеристики своего персонажа или вовсе получать новые способности. Причем для удобства разработчиков эта система должна быть понятной, эффективной и легко расширяемой. В этой статье описывается реализация такой системы с использованием инструмента Scriptable Object движка Unity.

Abstract. When developing computer games, developers often need to create a power-up system that allows players to increase damage, health or other features of their character or even gain new abilities. Moreover, for the convenience of developers, this system should be understandable, effective and easily extensible. This article describes the implementation of such a system using the Scriptable Object tool of the Unity game engine.

Ключевые слова: разработка игр, программирование, Unity, C#.

Keywords: game development, programming, Unity, C#.

Сложно представить современную компьютерную игру, в которой у игрока нет возможность тем или иным образом усилить своего персонажа. Усиления могут являться основополагающим элементом для некоторых жанров игр. Например, в играх жанра «Roguelike» по мере прохождения, игрок может получать различные усиления, но теряет практически весь прогресс при поражении. Чтобы сохранить интерес игрока, система усилений должна быть богатой и разнообразной. А в играх жанра «Metroidvania», игрок по мере изучения мира находит новые способности, которые позволяют ему попадать в недоступные ранее места. Без усилений исследование игрового мира было бы скучным и однообразным.

Создать единоразовое усиление персонажа не составляет труда для разработчиков, однако создание богатой системы разнообразных усилений, которые будут встречаться множество раз по мере прохождения игры, может вызывать трудности. При реализации такой системы

нужно учесть удобство внедрения различных усилений: от простого увеличения значений характеристик, до открытия новых способностей и возможностей для персонажа. Так же важна расширяемость системы: у разработчиков должен быть простой способ добавления новых усилений без потребности в исправлении существующего кода. Создать такую систему в игровом движке Unity можно с помощью инструмента Scriptable Object.

Scriptable Objects

Scriptable Object в Unity — это класс, который позволяет создавать объекты с пользовательскими данными и хранить их вне сцены или префаба (шаблон для объекта в игровом движке Unity). Они представляют собой контейнеры данных, которые могут быть использованы для хранения, организации и передачи информации между различными системами и компонентами игры. Scriptable Object может использоваться для различных целей при разработке игры, в этой статье описывается их использование для создания системы усилений.

Абстрактный класс для усилений

Первым шагом при создании системы усилений является создание абстрактного класса, от которого будут наследоваться все добавляемые в игру усиления. Для этого в проект добавляется новый скрипт, однако в отличие от других скриптов, этот будет наследоваться не от «MonoBehaviour», а от класса «ScriptableObject». Класс делается абстрактным, поскольку не потребуется создавать экземпляры этого класса, однако от него будут наследоваться классы конкретных усилений, таких, как например, увеличение здоровья или урона. В этом классе должны быть описаны поля, которые будут присущи любому усилению. Обязательным является объявление функции, которая будет срабатывать при получении усиления (функция «Apply» на рис. 1) Эта функция принимает на вход игровой объект, к которому будет применен эффект (обычно это сам игрок). Так же в примере ниже объявлено поле «Tier» – уровень усиления, может использоваться, чтобы по мере прогресса игрока выдавать ему более сильные бонусы. На рисунке 1 изображен пример этого класса, конкретная реализация может отличаться в зависимости от потребностей конкретной игры.

```
using UnityEngine;

public abstract class PowerupEffect : ScriptableObject
{
    public int tier;
    public abstract void Apply(GameObject target);
}
```

Рисунок 1. Абстрактный класс усилений

Создание усилений

Следующий этап – создание конкретных усилений. Для этого нужно создать новый класс, который будет унаследован от созданного ранее абстрактного класса. В нем необходимо реализовать объявленные в абстрактном классе функции, в нашем случае – это функция «Apply». В пример на рисунке 2 показана реализация усиления характеристик: объявлены поля для значений, на которые должна увеличиться каждая из характеристик и реализована функция «Apply», внутри которой соответствующие значения прибавляются к характеристикам игрока.

```
[CreateAssetMenu(menuName = "PowerUps/StatBuff")]
public class StatBuff : PowerupEffect
{
    public float healthAmount;
    public float damageAmount;
    public float fireRateAmount;
    public float bulletSpeedAmount;
    public float sizeAmount;

    public override void Apply(GameObject target)
    {
        target.GetComponent<CharacterContorl>().health += healthAmount;
        target.GetComponent<CharacterContorl>().maxHealth += healthAmount;
        target.GetComponent<Weapon>().fireRate += fireRateAmount;
        target.GetComponent<Weapon>().bulletSpeed += bulletSpeedAmount;
        target.GetComponent<Weapon>().damage += damageAmount;
        target.GetComponent<Weapon>().size += sizeAmount;
    }
}
```

Рисунок 2. Класс усиления характеристик

Для удобства разработки к функции так же можно добавить атрибут «CreateAssetMenu». Например, после добавления строчки «[CreateAssetMenu(menuName = "PowerUps/StatBuff")]», создавать экземпляры усилений можно будет прямо из меню Unity, открыв соответствующую вкладку. В круглых скобочках указывается название подменю для созданных усилений.

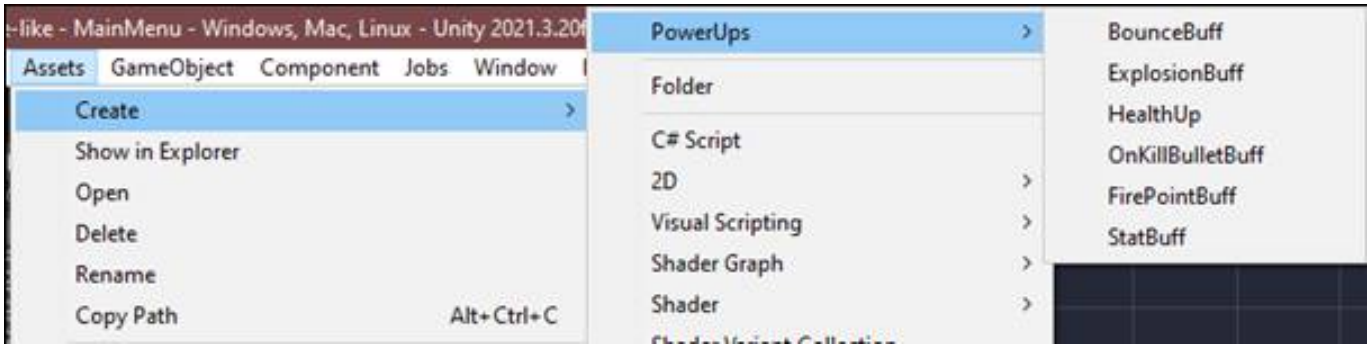

Рисунок 3. Добавление усилений из меню

Далее остается только создать конкретные экземпляры усилений. После выбора соответствующего пункта в меню, в проект добавляется объект усиления. Настроить значения объявленных ранее полей можно внутри панели Inspector.

Рисунок 4. Настройка значений

Аналогичным образом можно создать различные усиления, которые будут уникальным образом влиять на персонажа, а при создании нескольких экземпляров усилений одного класса, можно варьировать их числовые параметры чтобы создавать слабые или сильные бонусы. Далее разработчику остается только реализовать получение этих усилений игроком. Один из способов – создание на сцене объекта, при взаимодействии с которым, выполняется функция «Apply» соответствующего усиления, однако конкретная реализация будет зависеть от потребностей разрабатываемой игры.

Сложные усиления

При реализации системы усилений таким образом, можно создавать не только простые бонусы, увеличивающие характеристики, но и более сложные. Разработчики могут взаимодействовать с любыми элементами игры через функцию «Apply» нужным им образом. Так же можно реализовать срабатывание эффектов не только при получении бонуса, но и при других условиях, создав интерфейс для классов усилений. На рисунке 5 показан пример интерфейса «IOnHit». Классы, реализующие этот интерфейс должны определить функцию «OnHitEffect», которая будет срабатывать, когда игрок попадает снарядом по противнику.

```
using UnityEngine;

public interface IOnHit
{
    void OnHitEffect(GameObject bullet, Collision2D collision, float chance);
}
```

Рисунок 5. Пример интерфейса

Далее, при создании новых классов усилений, некоторые из них могут реализовывать один или несколько таких интерфейсов, что позволит создавать более сложные и интересные бонусы для игрока. Разнообразие таких усилений ограничивается только фантазией

разработчика.

Выводы

Система усилений для компьютерных игр на движке Unity созданная с помощью Scriptable Objects не только облегчает процесс разработки игры, предоставляя простой и удобный способ добавления бонусов, но и дает разработчикам возможность создавать сложные, разнообразные усиления, способные удовлетворить потребности любой игры.

Список литературы:

1. Документация Unity [Электронный ресурс] URL: https://docs.unity3d.com (дата обращения: 22.07.2024)