Адаптер - це структурний патерн проєктування, який дає змогу об'єктам із несумісними інтерфейсами працювати разом.

В моєму проекті він використовується багато де, але у всіх випадках використання, конвертер є вбудованим. Тобто юніті, як платформа та бібліотека має у себе дуже багато корисних та часто потрібних конверторів. Що є дуже зручним.

Але все таки я маю клас, який хоч і не прямо, можна назвати адаптером. Цей клас адаптує рух мишки та введення до кутів, які потім передаються нашому об’єкту, в мене це зброя. Тобто координати мишки по осі абсцис та ординат переводяться в кути нахилу, а потім за ними задається трансформація нашого об’єкта.

Клас має назву WeaponSway.

Як він працює:

Спершу ми задаємо швидкість обертання та максимальний нахил.

В методі OnEnable (який працює коли об’єкт включається в сцені) ми створюємо та включаємо нову систему, яка зчитує введення з мишки нашого гравця.

```
[Header("Sway Settings")]
public PlayerInput _input;
[SerializeField]
private float rotateSpeed = 4f;
[SerializeField]
private float maxTurn = 3f;

Unity Message | 0 references
private void OnEnable()
{
    _input = new PlayerInput();
    _input.Enable();
}
```

В методі Update ми фіксуємо введення та передаємо їх в функцію яка конвертує координати в кути, а також відразу застосовуємо їх до трансформації.

```
private void Update()
{
    Vector2 mouseInput = _input.OnFoot.Look.ReadValue<Vector2>();
    ApplyRotation(GetRotation(mouseInput));
}
```

Функція, яка переводить Vector2 (вектор 2 координатами x та y) у кут нахилу та повертає його значення.

```
Quaternion GetRotation(Vector2 mouse)
{
    mouse = Vector2.ClampMagnitude(mouse, maxTurn);
    Quaternion rotX = Quaternion.AngleAxis(-mouse.y, Vector3.right);
    Quaternion rotY = Quaternion.AngleAxis(mouse.x, Vector3.up);
    Quaternion targetRot = rotX * rotY;
    return targetRot;
}
```

Функція, яка приймає та застосовує цей кут нахилу до нашого об’єкта с урахуванням часу.

```
private void ApplyRotation(Quaternion targetRot)
{
    transform.localRotation = Quaternion.Slerp(transform.localRotation, targetRot, rotateSpeed * Time.deltaTime);
}
```

Цей клас в мене використовується для того, щоб моя зброя була не статичною, а динамічною, щоб вона трусилась при повертанні камери і т.д. Цей ефект змушує гру відчуватись більш реалістичною (Хоч повного реалізму я і не прагну).

Також хочу показати, як можна було зробити використання різних здібностей противниками за допомогою адаптера. Але ця система не реалізована в моєму проекті, тому що в мне вже є інша, яка використовує інші патерни.

Клас UniqueEnemy, не реалізує жодний інтерфейс, але має метод, який щось робить (в даному випадку визиває якусь здібність).

```
class UniqueEnemy
{
  public string UseSkill()
  {
     return "Using Skill";
  }
}
```

Інтерфейс IEnemy в якому є метод атак. Але в нашому UniqueEnemy, немає цього метода та він не реалізує цей інтерфейс. Що робити?

```
interface IEnemy
{
  string Attack();
}
```

Використовувати клас UniqueEnemyAttack, який є адаптором між класом UniqueEnemy та інтерфейсом IEnemy.

```
class UniqueEnemyAttack: IEnemy
{
  UniqueEnemy uEnemy = new UniqueEnemy();
  public string Attack()
  {
     return uEnemy.UseSkill ();
  }
}
```

Ось таким чином можна використовувати адаптер в юніті.

Скрипти можете знайти за шляхом:

Assets/Scripts/Weapon/ShootingWeapon/MainScripts/WeaponSway.cs