**федеральное государственное автономное образовательное учреждение высшего образования**



**МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**(ВЫСШАЯ ШКОЛА ПЕЧАТИ И МЕДИАИНДУСТРИИ)**

**(Факультет информационных технологий)**

***(Институт Принтмедиа и информационных технологий)***

***Кафедра Информатики и информационных технологий***

**направление подготовки**

**09.03.02 «Информационные системы и технологии»**

**ЛАБОРАТОРНАЯ РАБОТА № 2**

**Дисциплина:** Шаблоны проектирования

**Тема: Система игровых событий**

**Выполнил(а): студент(ка) группы 221-3711**

Морозов К.А.

(Фамилия И.О.)

**Дата, подпись** ***\_\_\_\_\_\_\_\_\_\_\_***

(Дата) (Подпись)

**Проверил: \_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_***

(Фамилия И.О., степень, звание) **(Оценка)**

**Дата, подпись** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  ***\_\_\_\_\_\_\_\_\_\_\_***

(Дата) (Подпись)

**Замечания: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Москва2024**

Лабораторная работа №2

Система игровых событий

Цель: Создайте систему событий, в которой различные игровые объекты могут подписываться и реагировать на игровые события с использованием определенного шаблона проектирования.

Описание: В играх часто происходят различные события: от перемещения игрока до завершения задания. Для управления такими событиями и реакцией на них различных компонентов игры необходима эффективная система. Использование подходящего шаблона проектирования может сделать эту систему более гибкой и удобной.

**Шаги:**

Определение игровых событий:

- Определите набор игровых событий, которые вы хотите реализовать. Это могут быть события, связанные с движением игрока, завершением уровня, сбором предметов и т. д.

Реализация системы событий:

- Создайте "издателя" событий, который будет генерировать события.

- Разработайте механизм подписки, который позволит другим игровым объектам "подписываться" на интересующие их события.

- Игровые объекты, "подписанные" на события, должны реагировать на них соответствующим образом.

Тестирование:

- Запустите вашу игру и проверьте, как различные компоненты реагируют на игровые события в реальном времени.

**Листинг:**

1. EventManager

Статический класс управляет событиями добавления очков, сбора монет и завершения уровня. Он включает методы для вызова этих событий и делегатов для обработки этих событий другими скриптами.

* **RaiseAddPoints(int points)**: Вызывает событие добавления очков.
* **RaiseCoinCollected()**: Вызывает событие сбора монеты.
* **RaiseLevelComplete()**: Вызывает событие завершения уровня.

|  |
| --- |
| * using UnityEngine.Events; * public static class EventManager * { * public static event UnityAction<int> OnAddPoints; * public static event UnityAction OnCoinCollected; * public static event UnityAction OnLevelComplete; * public static void RaiseAddPoints(int points) => OnAddPoints?.Invoke(points); * public static void RaiseCoinCollected() => OnCoinCollected?.Invoke(); * public static void RaiseLevelComplete() => OnLevelComplete?.Invoke(); * } |

1. Player

Скрипт отвечает за передвижение игрока на основе ввода с клавиатуры. Он получает входные данные по осям (горизонтальной и вертикальной) и перемещает игрока в соответствии с ними.

|  |
| --- |
| using UnityEngine;  public class Player : MonoBehaviour  {  public float speed = 5f;  void Update()  {  float moveHorizontal = Input.GetAxis("Horizontal");  float moveVertical = Input.GetAxis("Vertical");  Vector2 movement = new Vector2(moveHorizontal, moveVertical);  movement = movement.normalized;  transform.Translate(movement \* speed \* Time.deltaTime);  }  } |

1. LevelController

Скрипт подписывается на событие завершения уровня и выводит сообщения в консоль, когда уровень завершен. Он также может инициировать завершение уровня.

* **OnEnable()**: Подписывается на событие завершения уровня.
* **OnDisable()**: Отписывается от события завершения уровня.
* **HandleLevelComplete()**: Обрабатывает событие завершения уровня и выводит сообщение в консоль.
* **CompleteLevel()**: Инициирует завершение уровня.

|  |
| --- |
| * using UnityEngine; * public class LevelController : MonoBehaviour * { * private void OnEnable() * { * EventManager.OnLevelComplete += HandleLevelComplete; * } * private void OnDisable() * { * EventManager.OnLevelComplete -= HandleLevelComplete; * } * private void HandleLevelComplete() * { * Debug.Log("Level Completed!"); * Debug.Log("Transition to new level"); * } * public void CompleteLevel() * { * EventManager.RaiseLevelComplete(); * } * } |

1. ScoreController

Скрипт отслеживает текущие очки, собранные монеты и текущий уровень. Он обновляет соответствующие UI элементы и обрабатывает события добавления очков и сбора монет. При достижении определенного количества очков он добавляет бонусные очки. При сборе всех монет в уровне инициирует завершение уровня.

* **OnEnable()**: Подписывается на события добавления очков, сбора монет и завершения уровня.
* **OnDisable()**: Отписывается от событий добавления очков, сбора монет и завершения уровня.
* **HandleAddPoints(int points)**: Обрабатывает событие добавления очков, обновляет текущий счет и UI. Добавляет бонусные очки при достижении определенного количества очков.
* **HandleCoinCollected()**: Обрабатывает событие сбора монеты, обновляет счетчик монет и UI. Инициирует завершение уровня, если собраны все монеты.
* **HandleLevelComplete()**: Обрабатывает событие завершения уровня, обновляет текущий уровень и UI. Выводит сообщение о переходе на новый уровень.

|  |
| --- |
| * using UnityEngine; * using UnityEngine.UI; * public class ScoreController : MonoBehaviour * { * public int currentScore = 0; * private bool bonusGiven = false; * public int coinsCollected = 0; * public int totalCoinsInLevel = 5; * public int currentLevel = 1; * public Text scoreText; * public Text coinsText; * public Text levelText; * private void OnEnable() * { * EventManager.OnAddPoints += HandleAddPoints; * EventManager.OnCoinCollected += HandleCoinCollected; * EventManager.OnLevelComplete += HandleLevelComplete; * } * private void OnDisable() * { * EventManager.OnAddPoints -= HandleAddPoints; * EventManager.OnCoinCollected -= HandleCoinCollected; * EventManager.OnLevelComplete -= HandleLevelComplete; * } * private void HandleAddPoints(int points) * { * currentScore += points; * scoreText.text = "Score: " + currentScore; * if (currentScore >= 50 && !bonusGiven) * { * bonusGiven = true; * EventManager.RaiseAddPoints(100); * Debug.Log("Good job!"); * } * } * private void HandleCoinCollected() * { * coinsCollected++; * coinsText.text = "Coins: " + coinsCollected; * if (coinsCollected >= totalCoinsInLevel) * { * EventManager.RaiseLevelComplete(); * } * } * private void HandleLevelComplete() * { * currentLevel++; * coinsCollected = 0; * levelText.text = "Level: " + currentLevel; * Debug.Log("Transition to new level"); * } * } |

1. CoinController

Скрипт отвечает за уничтожение монеты при столкновении с игроком. Он вызывает события добавления очков и сбора монет, когда игрок касается монеты.

* **OnTriggerEnter2D(Collider2D other)**: Обрабатывает столкновение с игроком, вызывает события добавления очков и сбора монеты, уничтожает объект монеты.

|  |
| --- |
| * using UnityEngine; * public class CoinController : MonoBehaviour * { * [SerializeField] private int points = 10; * private void OnTriggerEnter2D(Collider2D other) * { * if (other.CompareTag("Player")) * { * EventManager.RaiseAddPoints(points); * EventManager.RaiseCoinCollected(); * Destroy(gameObject); * } * } * } |

**Ход работы**

Для выполнения данной лабораторной работы была создана игровая сцена с управляемым игроком и пятью монетами для сбора. Игрок управляется с помощью клавиш и может перемещаться по игровому полю, собирая монеты. При каждом сборе монеты счетчик очков увеличивается на 10, а счетчик монет – на одну единицу. Когда все монеты собраны, игрок получает бонус в 100 очков и в консоль выводится сообщение о переходе на новый уровень.

Система событий была создана для управления игровыми состояниями и обновления пользовательского интерфейса (UI). Основные события включали OnAddPoints, которое активировалось при увеличении очков, OnCoinCollected, отслеживающее сбор монет, и OnLevelComplete, управляющее переходом на новый уровень.

EventManager управлял всеми событиями в игре, обеспечивая их активацию и обработку. Player скрипт отвечал за движение игрока. CoinController обрабатывал столкновение с игроком, вызывал события добавления очков и сбора монет, а также уничтожал монету после ее сбора. ScoreController обрабатывал добавление очков, обновление счетчиков монет и уровней, а также выводил сообщения в консоль при достижении определенных условий. LevelController отвечал за завершение уровня и вывод сообщения о переходе на новый уровень.

Использование паттерна «Наблюдатель» (Observer) позволило удобно связывать реакции различных объектов на события в игре. Это сделало систему более гибкой и легко модифицируемой, позволяя добавлять новые реакции на события без значительных изменений в коде. События позволили легко подписывать и отписывать различные объекты на определенные действия, что значительно упростило управление игровыми состояниями и поведением объектов.

Ссылка на проект в Github: <https://github.com/Sollimba/Hablon2>