УДК 004

Создание интерактивной головоломки в Unity

Андриенко Иван Сергеевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье описывается процесс создания интерактивной головоломки в среде разработки Unity. Головоломка представляет собой игровое поле, состоящее из кубиков, которые меняют цвет при взаимодействии с ними. Для реализации данной головоломки используется игровой движок Unity. Результатом работы является интерактивная головоломка, способная захватить внимание и вызвать интерес у игроков.
Ключевые слова: интерактивная головоломка, Unity, игровое поле, куб, C# скрипт.

Creating an interactive puzzle in Unity

Andrienko Ivan Sergeevich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article describes the process of creating an interactive puzzle in the Unity development environment. The puzzle is a playing field consisting of cubes that change color when interacting with them. The Unity game engine is used to implement this puzzle. The result of the work is an interactive puzzle that can capture the attention and arouse the interest of players.
Keywords: interactive puzzle, Unity, playing field, cube, C# script.

1 Введение

1.1 Актуальность

Создание интерактивной головоломки в Unity представляет собой актуальную тему исследования, заинтересовывающую разработчиков игр. Эта тема имеет практическую значимость, поскольку позволяет создавать увлекательные игровые опыты, требующие логического мышления и стратегического подхода. Разработка такой головоломки в Unity позволяет использовать инструменты и возможности движка для создания уникальных геймплейных механик, визуального оформления и звукового сопровождения. Кроме того, данная тема открывает возможности для исследования алгоритмов решения головоломок и создания адаптивных игровых сценариев.

1.2 Обзор исследований

А.С. Дедюкина, Д.Г. Мандарова описали создание приложения для углубленного изучения повседневного быта якутской культуры при помощи межплатформенной среды разработки Unity [1]. Д.Ю. Тазабеков, Н.В. Бужинская исследовали особенности разработки обучающей компьютерной игры в среде Unity. Данная игра предназначена для обучения школьников информатике в занимательной форме. Графическая оболочка игры создавалась в Adobe Photoshop CS6, игровая механика - на игровом движке Unity. Для написания скриптов использовался язык программирования C# [2]. А.В. Бардин, Т.Н. Томчинская описали процесс создания интерактивной сцены в игровом движке Unity. Объяснен путь получения модели для экспорта в программном обеспечении Autodesk Infraworks. Модель создана на основе воспроизведенного ранее рельефа поверхности части Зеленского съезда Нижнего Новгорода и его близлежащей инфраструктуры. Рассмотрены способы импорта полученной модели в сцену Unity с дальнейшим взаимодействием с ней [3]. Н.Г. Дюкина, А.С Шутов рассмотрели разработку интерактивного приложения на платформе UNITY. Интерактивное приложение рассматривается как компьютерная игра для мобильных устройств [4]. Козлов С.В., Меженцев И.О. описали проблемы создания видеоигр, которые, в свою очередь, подталкивают к созданию новых решений для развития компьютерной науки. В качестве примера рассматривается разработка видеоигры с помощью межплатформенной среды Unity. Для взаимодействия со средой разработки используется язык программирования C#, который имеет свои особенности в данной среде [5].

1.3 Цель исследования

Цель исследования – разработать и реализовать интерактивную головоломку в Unity.

2 Материалы и методы

Для реализации интерактивной головоломки была использована интегрированная среда разработки Unity версии 2021.2.3f1. Языком программирования для создания скриптов кубиков и управления игровым процессом был выбран C#.

3 Результаты и обсуждение

Суть головоломки заключается в создании игрового поля, состоящего из кубиков. Изначально все кубики имеют красный цвет. Цель игры состоит в том, чтобы изменить цвет всех кубиков на зеленый. Игрок может взаимодействовать с кубиками путем прикосновения или столкновения с ними. При таком взаимодействии выбранный кубик изменяет свой цвет на зеленый, а также изменяются цвета соседних кубиков. При повторном соприкосновении кубики становиться опять красными. Соседними кубиками являются кубики, расположенные вокруг, то есть 8 кубов. Игрок должен использовать свои логические и пространственные навыки, чтобы определить

правильную последовательность взаимодействия с кубиками и добиться того, чтобы все кубики стали зелеными. Возможно, потребуется несколько шагов и стратегическое планирование, чтобы решить головоломку. Данная головоломка будет интегрирована в VR игру, где игрок должен взаимодействовать с кубиками контроллером.

Для реализации головоломки создадим в новой сцене поле с кубиками. Поле может быть любых размеров, но оно должно быть решаемым. В данном примере будет использоваться поле 4 на 4. Для создания куба создаем 3D объект и размешаем их рядом с друг другом (рис. 1).

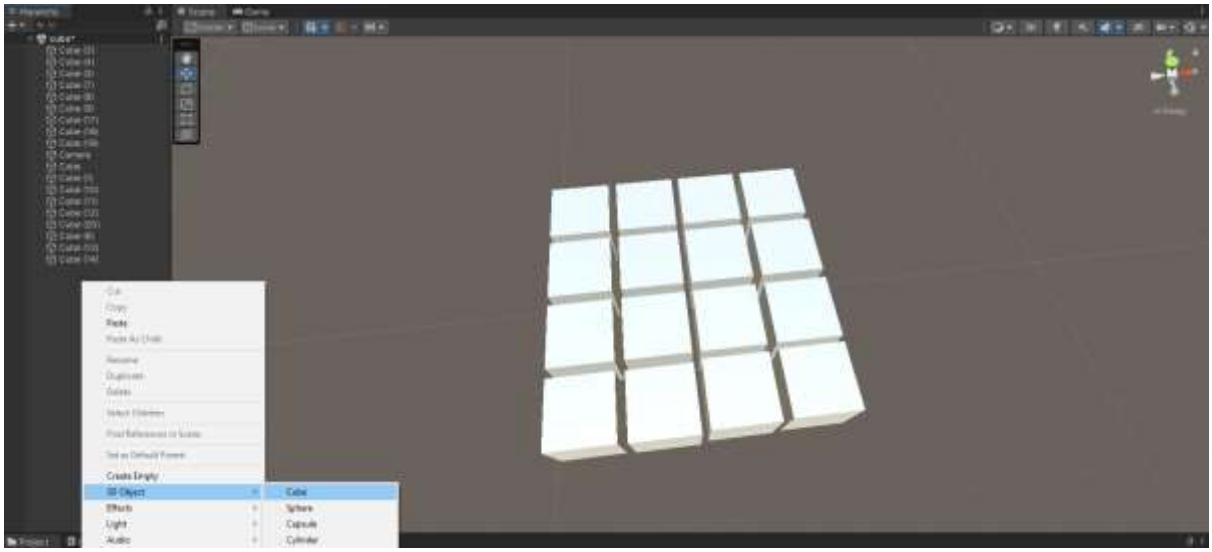

Рисунок 1 – Создание игрового поля

Переходим к созданию скрипта для головоломки. В папке «Assets» создаем папку для скриптов. В новой папке создаем C# скрипт «CubeColorController» (рис. 2).

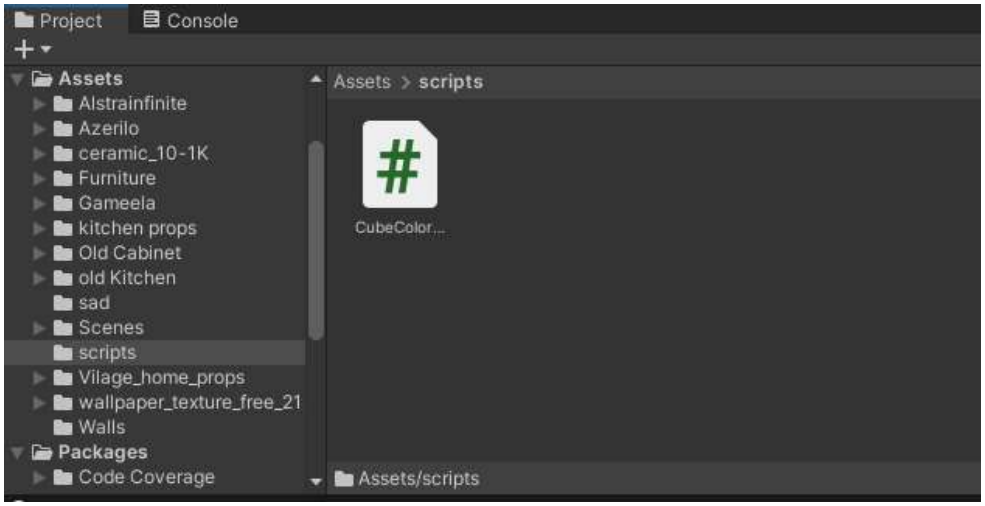

Рисунок 2 – Создание C# скрипта

Пропишем логику для скрипта. В начале кода, создаем класс CubeColorController, отвечающий за управление цветом кубика. В начале

объявляем переменные, включая «cubeRenderer», который представляет собой компонент Renderer для доступа к отображению кубика. «initialColor» хранит начальный цвет кубика, а также «activeColor» и «inactiveColor», определяющие цвета для активного и неактивного состояний кубика соответственно. Также имеется переменная «isWin», которая указывает, достигнута ли победа в игре. Значение по умолчанию для «isWin» - «false», что означает, что победы пока не было (рис. 3).

CubeColorController.cs

```
1  using UnityEngine;
2
3  public class CubeColorController : MonoBehaviour
4  {
5      private Renderer cubeRenderer;
6      private Color initialColor;
7      private Color activeColor = Color.green;
8      private Color inactiveColor = Color.red;
9
10     private bool isWin = false;
11
```

Рисунок 3 – Создание переменных

Далее в функции Start() получаем компонент «Renderer» для текущего объекта и сохраняем его в переменную «cubeRenderer». Затем сохраняем начальный цвет кубика в переменную «initialColor», получая его из материала кубика. В функции «OnCollisionEnter(Collision collision)» вызываем несколько методов. Сначала вызываем метод «ToggleCubeColor()», который изменяет цвет текущего кубика. Затем вызываем метод «ToggleNeighborCubeColors()», который изменяет цвет соседних кубиков. Наконец, вызываем метод «CheckWinCondition()», который проверяет условие победы в игре. Этот код выполняется при столкновении кубика с другим объектом и отвечает за изменение цветов кубиков и проверку условия победы (рис. 4).

```
11
12     private void Start()
13     {
14         cubeRenderer = GetComponent<Renderer>();
15         initialColor = cubeRenderer.material.color;
16     }
17
18     private void OnCollisionEnter(Collision collision)
19     {
20         ToggleCubeColor();
21         ToggleNeighborCubeColors();
22         CheckWinCondition();
23     }
24
```

Рисунок 4 – Реализация бесконечного цикла

Определяем два метода. В методе ToggleCubeColor() происходит изменение цвета текущего кубика. Проверяем текущий цвет кубика и, в зависимости от него, устанавливаем новый цвет. Если текущий цвет равен неактивному цвету (inactiveColor), то устанавливаем активный цвет (activeColor), и наоборот. Метод ToggleNeighborCubeColors() отвечает за изменение цветов соседних кубиков. Необходимо использовать функцию «Physics.OverlapBox()» для определения всех коллайдеров, которые пересекаются с областью, охватывающей текущий кубик. Затем проходимся по каждому найденному коллайдеру и проверяем, не является ли он самим текущим кубиком. Если это не текущий кубик, то получаем компонент «Renderer» соседнего кубика и устанавливаем его цвет равным текущему цвету кубика. Таким образом, эти два метода отвечают за изменение цветов кубиков и их соседей (рис. 5).

```
24
25      private void ToggleCubeColor()
26      {
27          cubeRenderer.material.color = cubeRenderer.material.color == inactiveColor ? activeColor : inactiveColor;
28      }
29
30      private void ToggleNeighborCubeColors()
31      {
32          Collider[] colliders = Physics.OverlapBox(transform.position, Vector3.one, Quaternion.identity);
33
34          foreach (Collider collider in colliders)
35          {
36              if (collider.gameObject != gameObject)
37              {
38                  Renderer neighborRenderer = collider.GetComponent<Renderer>();
39                  neighborRenderer.material.color = cubeRenderer.material.color;
40              }
41          }
42      }
```

Рисунок 5 – Метод ToggleCubeColor() и ToggleNeighborCubeColors()

Для того чтобы игрок мог выиграть, пропишем метод CheckWinCondition(), который отвечает за проверку условия победы в игре. Сначала получаем все объекты «CubeColorController» при помощи функции «FindObjectsOfType<CubeColorController>()» и сохраняем их в массив «allCubes». Затем мы выполняем итерацию по каждому кубику в массиве и проверяем его цвет. Если хотя бы один кубик имеет цвет, отличный от активного цвета, происходит возврат из метода, так как еще остаются деактивированные кубики. Если все кубики загорелись зеленым, мы изменяем цвет всех кубиков на черный и устанавливаем значение переменной «isWin» в «true», указывая на успешное завершение игры. В качестве победы кубики будут загораться черным цветом. В дальнейшем можно сделать другое событие победы, например, воспроизведение музыки (рис. 6).

```
43
44      private void CheckWinCondition()
45      {
46          CubeColorController[] allCubes = FindObjectsOfType<CubeColorController>();
47
48          foreach (CubeColorController cube in allCubes)
49          {
50              if (cube.cubeRenderer.material.color != activeColor)
51              {
52                  return; // Еще есть невыигранные кубики
53              }
54          }
55
56          // Все кубики загорелись зеленым, изменяем цвет всех кубиков на черный
57          foreach (CubeColorController cube in allCubes)
58          {
59              cube.cubeRenderer.material.color = Color.black;
60          }
61
62          isWin = true;
63      }
```

Рисунок 6 – Создание метода CheckWinCondition()

Создадим метод ResetCubeColor(), который отвечает за сброс цвета кубика. При вызове этого метода происходит проверка переменной «isWin». Если она имеет значение «true», то цвет кубика возвращается к исходному цвету («initialColor»), а значение «isWin» устанавливается в «false», чтобы сбросить состояние победы. Этот метод используется для восстановления исходного состояния кубика после завершения игры (рис. 7).

```
65      public void ResetCubeColor()
66      {
67          if (isWin)
68          {
69              cubeRenderer.material.color = initialColor;
70              isWin = false;
71          }
72      }
73  }
74
```

Рисунок 7 – Создание метода ResetCubeColor()

Тестируем головоломку. В качестве нажатия на куб будет падать другой объект (рис. 8). Тестируем событие победы (рис. 9).

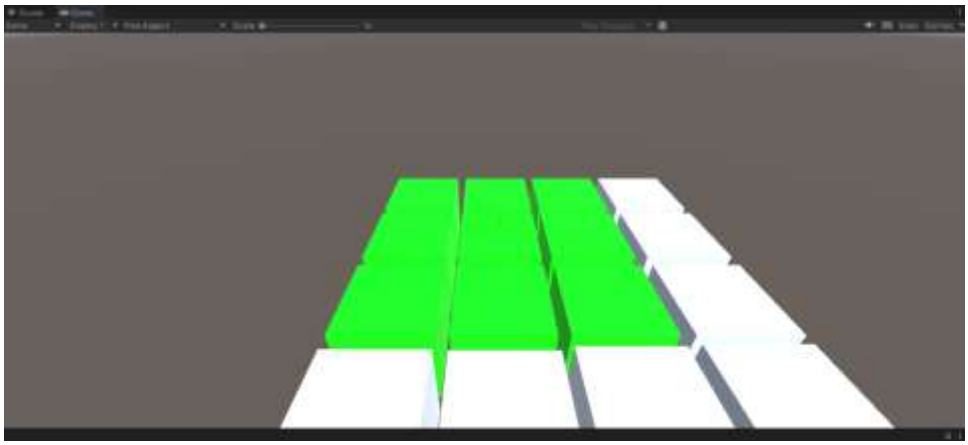

Рисунок 8 – Нажатие на куб

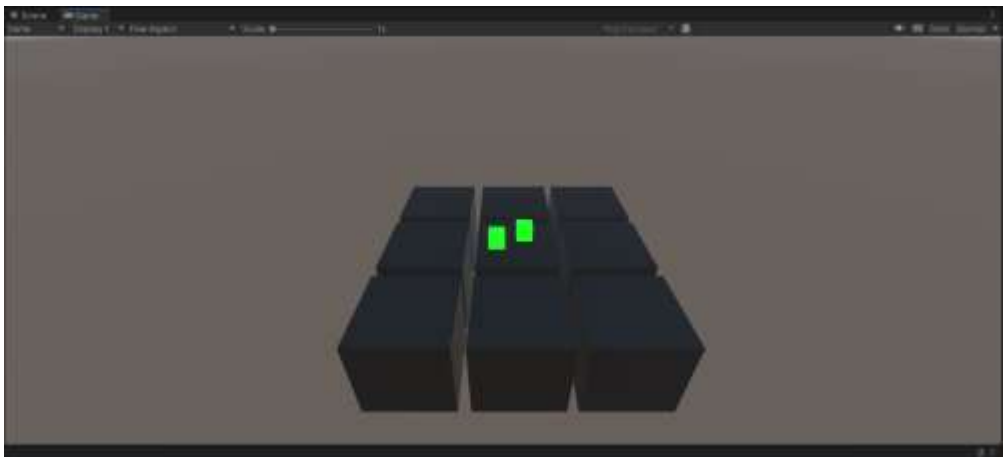

Рисунок 9 – Прохождение головоломки

Выводы

В данной работе была разработана интерактивная головоломка в среде Unity. Была создана система, основанная на изменении цветов кубиков и их взаимодействии при столкновениях. Реализованная функциональность позволяет пользователю взаимодействовать с головоломкой, изменяя состояние кубиков и стремясь достичь выигрышного условия. В процессе работы был использован язык программирования C#. Полученные результаты подтверждают успешное создание интерактивной головоломки в Unity.

Библиографический список

1. Дедюкина А.С., Мандарова Д.Г. Разработка vr приложения «Якутский балаган» с помощью межплатформенной среды разработки unity // DIGITAL EDU. Цифровые компетенции в образовании. Сборник материалов Всероссийского научного форума с международным участием. Киров, 2022. С. 228-231.
2. Тазабеков Д.Ю., Бужинская Н.В. Применение среды unity для разработки обучающих компьютерных игр // Тенденции развития науки и образования. 2020. № 58-2. С. 34-37.
3. Бардин А.В., Томчинская Т.Н. Разработка интерактивной сцены в игровом движке unity для автомобильного симулятора автошколы // КОГРАФ-2022. сборник материалов 32-й Всероссийской научно-практической конференции по графическим информационным технологиям и системам. Нижний Новгород, 2022. С. 144-152.
4. Дюкина Н.Г., Шутов А.С Разработка интерактивного приложения на платформе unity // Вестник педагогического опыта. 2022. № 53. С. 21-24.
5. Козлов С.В., Меженцев И.О. Взаимодействие языка программирования c# с межплатформенной средой разработки unity // Системы компьютерной математики и их приложения. 2019. № 20-1. С. 193-199.