



БАЗОВОЕ УПРАВЛЕНИЕ ПЕРСОНАЖАМИ И КАМЕРОЙ В 2D КОМПЬЮТЕРНЫХ ИГРАХ

В. Д. Забурдин, Д. В. Коновалов, А. В. Шахомиров

Санкт-Петербургский государственный университет аэрокосмического приборостроения

В статье рассмотрены аспекты управления камерой и игровыми персонажами в среде разработки Unity. Целью работы являлась возможность предоставить конкретные примеры реализации, которые помогут осуществить необходимое движение персонажей в игре, описать использование базовых механик управления движением игровых персонажей и камеры с помощью мыши.

Ключевые слова: компьютерная графика, управление персонажем, управление камерой, Unity, игровое пространство.

Для цитирования:

Забурдин, В. Д. Базовое управление персонажами и камерой в 2D компьютерных играх / В. Д. Забурдин, Д. В. Коновалов, А. В. Шахомиров // Системный анализ и логистика. – 2024. – № 1(39). – с. 30 – 36. DOI: 10.31799/2077-5687-2024-1-30-36.

BASIC CHARACTERS AND CAMERA CONTROLS IN 2D COMPUTER GAMES

V. D. Zaburdin, D. V. Konovalov, A. V. Shakhomirov

St. Petersburg State University of Aerospace Instrumentation

The article deals with the aspects of controlling the camera and game characters in the Unity development environment. The purpose of the work was to provide specific examples of implementation that will help to realize the necessary movement of characters in the game, to describe the use of basic mechanics of controlling the movement of game characters and the camera with the mouse.

Keywords: computer graphics, character control, camera control, Unity, game space.

For citation:

Zaburdin, V. D. Basic characters and camera controls in 2D computer games / V. D. Zaburdin, D. V. Konovalov, A. V. Shakhomirov // System analysis and logistics. – 2024. – № 1(39). – p. 30 – 36. DOI: 10.31799/2077-5687-2024-1-30-36.

Введение

В мире приложений и игр эффективное управление объектами и камерой играет критическую роль в обеспечении удобства пользователя. В приложениях это обычно связано с навигацией по интерфейсу и визуальным взаимодействием с контентом [1]. В играх же управление персонажем и камерой выступает как один из основных строительных блоков геймплея, определяя вовлекающий опыт и влияя на восприятие игрового окружения [2]. Создание эффективных и приятных в использовании механик управления становится ключевым элементом разработки, влияя на общее впечатление от приложения или игры.

Задача и среда разработки

В качестве задачи выступает разработка движения камерой и управления персонажем для игрового приложения в среде Unity 2D [3] жанра Roguelite [4] - Shoot 'em up.

Выбранным вариантом движением персонажа является управление мышью. Такой тип контроля встречается в моба и стратегиях в реальном времени, например, Dota 2 или Warcraft 3. Однако в рамках разрабатываемого приложения не требуется управление несколькими персонажами одновременно, а также отсутствует поиск пути, что делает управление ближе к варианту, представленному в игре Magica.

Камера не закреплена за персонажем, а также управляется мышью, что позволяет большее время держать ее в статическом положении, акцентируя внимание пользователя на происходящих событиях игрового окружения. Такой вид реализации можно встретить помимо



вышеупомянутых игр в Civilization 5 – пошаговая стратегия, и Dysco Elisium – ролевая игра.

Реализация

Управление персонажем

Перед рассмотрением непосредственно программы движения персонажа, рассмотрим входные параметры. В качестве примера рассмотрим сцену представленную на рисунке 1. На сцене располагаются три объекта образованных от общего Prefab [5] Player, обладающего стандартными параметрами: скорость движения и булевой переменной “дружелюбен”. Наследуемые от него объекты на сцене имеют следующие атрибуты, представленные на рисунке 1.

Так программа управления персонажем должна обладать возможностью переключения между персонажами, исключая противника. А также осуществлять движение объектом основываясь на значении его скорости.



Рис. 1. Сцена игрового пространства.

В данной работе рассмотрена базовая механика движения, реализованная в скрипте MoveToMouse, который закреплен за объектом Prefab Player.

Для проверки отслеживания выбранного персонажа используется список moveableObjects. В случае если выбранный объект является “другом” (isFriendly) происходит очистка списка и добавление выбранного персонажа.

Листинг 1. Скрипт MoveToMouse. Выбор персонажа

```
private void OnMouseDown() {
    if (this.GetComponent<StandardVariables>().isFriendly) {
        selected = true;
        gameObject.GetComponent<SpriteRenderer>().color = Color.green;
        foreach(MoveToMouse obj in moveableObjects) {
            if(obj != this) {
                obj.selected = false;
                obj.gameObject.GetComponent<SpriteRenderer>().color=Color.white
            }
        }
    }
}
```



Рис. 2. Осуществление выбора персонажа

Движение персонажа происходит в каждом графическом кадре в методе Update(). Если текущий объект выбран и нажата правая кнопка мыши, то осуществляется преобразование координат мыши (mousePosition) в мировые координаты, и данное значение записывается как цель (target). Далее осуществляется движение персонажа используя метод MoveTowards класса Vector3 с учетом скорости движения персонажа (speed).

Листинг 2. Скрипт Update. Движение персонажа

```
void Update() {  
    //...//  
    if(Input.GetMouseButton(1) && selected) {  
        target = Camera.main.ScreenToWorldPoint(Input.mousePosition);  
        target.z = transform.position.z;  
        Debug.Log(target);  
    }  
    speed = this.GetComponent<StandardVaruables>().movementSpeed;  
    transform.position = Vector3.MoveTowards(transform.position, target,  
        speed * Time.deltaTime);  
}
```

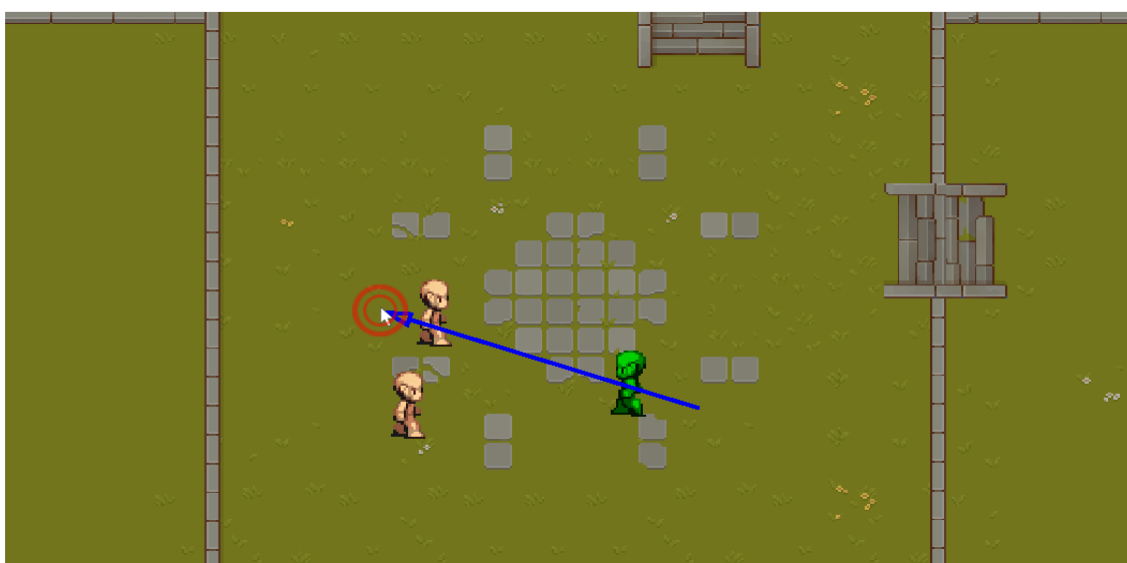


Рис. 3. Осуществление движения персонажа



Управление камерой

Движение камеры реализовано через пользовательский интерфейс среды Unity (UI) [6]. На сцене к камере прикреплен холст (canvas), на котором располагаются четыре объекта типа image. Они закреплены к краям родительского холста, а также указаны в инспекторе камеры.

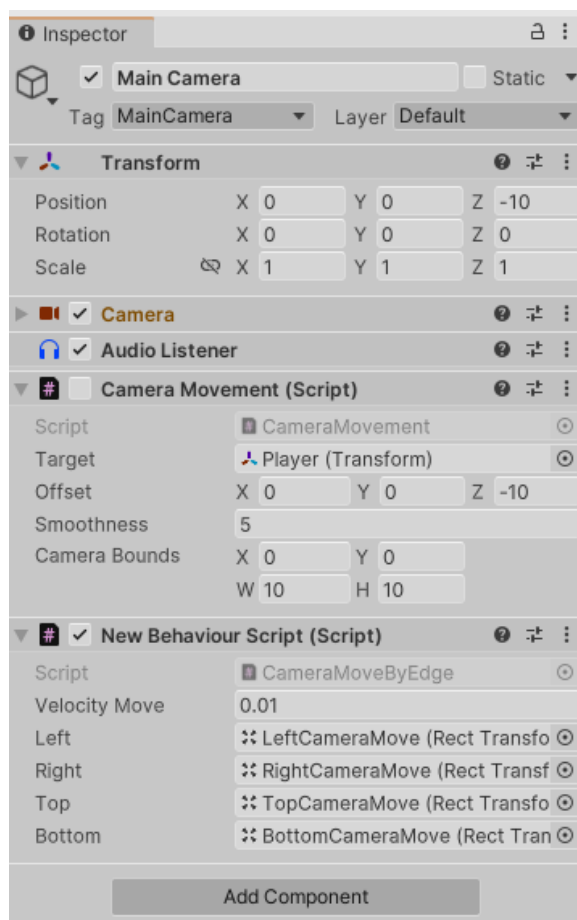


Рис. 4. Параметры камеры



Рис. 5. Объекты на холсте (Canvas)



Непосредственно управление и перемещение камеры происходит в методе Update(). Каждый кадр координаты мыши (mousePosition) [7] преобразуются в координаты игрового пространства, и, если они по абсолютному значению превышают координату одного из полей, осуществляется движение камеры.

Листинг 3. Скрипт Update. Движение камеры

```
void Update() {  
    Vector3 mousePosition;  
    mousePosition = Camera.main.ScreenToWorldPoint(Input.mousePosition);  
    mousePosition.z = transform.position.z;  
    Debug.Log(mousePosition);  
    if (mousePosition.x < Left.position.x)  
    {  
        transform.position += new Vector3(-VelocityMove, 0, 0);  
    }  
    if (mousePosition.x > Right.position.x)  
    {  
        transform.position += new Vector3(VelocityMove, 0, 0);  
    }  
    if (mousePosition.y > Top.position.y)  
    {  
        transform.position += new Vector3(0, VelocityMove, 0);  
    }  
    if (mousePosition.y < Bottom.position.y)  
    {  
        transform.position += new Vector3(0, -VelocityMove, 0);  
    }  
}
```

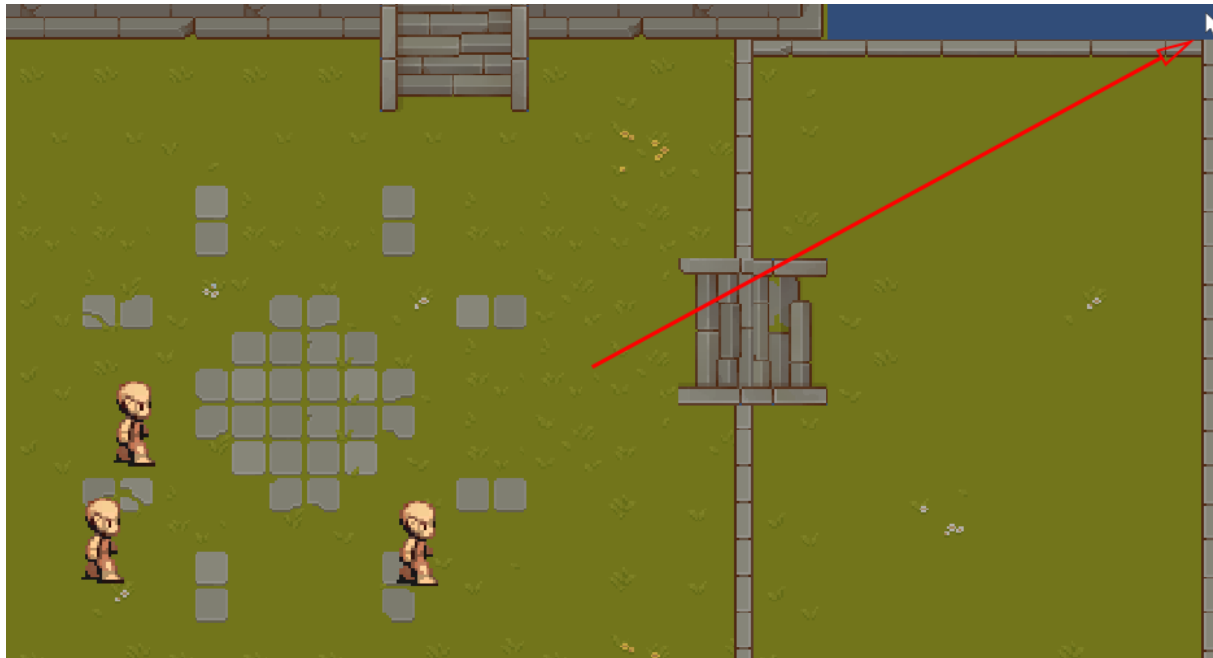


Рис. 6. Движение камеры

Заключение

С использованием современных сред разработки Unity и языка программирования C#, реализованы алгоритмы, которые обрабатывают перемещение игровых персонажей и камеры на сцене игрового пространства. Планируется дальнейшая разработка и реализация алгоритмического поведения персонажей игры, управляемых непосредственно игроком либо



компьютером, в плане обхода препятствий на игровом поле, и поиска путей прохода из одной части игрового лабиринта в другую.

СПИСОК ЛИТЕРАТУРЫ

1. Ферроне Х. Изучаем C# через разработку игр Unity. 5-е издание / Ферроне Х.; пер. с англ. А. Павлов. — СПб.: Питер, 2021. – 279 с.
2. What is "3C" in the game development and design? [Электронный ресурс]. – URL: <https://www.programmingsought.com/article/47897859912/> (дата обращения: 05.12.2023).
3. Хокинг Д. Unity в действии. 2-е международное издание / Хокинг Д.; пер. с англ. И. Рузмайкина. — СПб.: Питер, 2019. – 16 с.
4. Roguelite [Электронный ресурс]. – URL: <https://plarium.com/en/glossary/roguelite/> (дата обращения: 05.12.2023).
5. Unity Documentation: Prefab [Электронный ресурс]. – URL: <https://docs.unity3d.com/Manual/Prefabs.html> (дата обращения: 05.12.2023).
6. Unity Documentation: User Interface [Электронный ресурс]. – URL: <https://docs.unity3d.com/Manual/UIToolkits.html> (дата обращения: 05.12.2023).
7. Unity Documentation: Mouse events [Электронный ресурс]. – URL: <https://docs.unity3d.com/Manual/UIE-Mouse-Events.html> (дата обращения: 05.12.2023).

ИНФОРМАЦИЯ ОБ АВТОРАХ

Забурдин Вячеслав Дмитриевич –

Студент кафедры аэрокосмических компьютерных и программных систем
Санкт-Петербургский государственный университет аэрокосмического приборостроения
190000, Санкт-Петербург, ул. Большая Морская, д. 67, лит. А
E-mail: zaburdinslv@mail.ru

Коновалов Даниил Владиславович –

Студент кафедры аэрокосмических компьютерных и программных систем
Санкт-Петербургский государственный университет аэрокосмического приборостроения
190000, Санкт-Петербург, ул. Большая Морская, д. 67, лит. А
E-mail: danil.konovalev.70@gmail.com

Шахомиров Андрей Викторович –

Доцент кафедры аэрокосмических компьютерных и программных систем, кандидат технических наук
Санкт-Петербургский государственный университет аэрокосмического приборостроения
190000, Санкт-Петербург, ул. Большая Морская, д. 67, лит. А
E-mail: shakhomirov@gmail.com

INFORMATION ABOUT THE AUTHORS

Zaburdin Vyacheslav Dmitrievich –

Student of the Department of Aerospace Computer and Program Systems
St. Petersburg State University of Aerospace Instrumentation
67 Bolshaya Morskaya St., St. Petersburg, Lit. A, 190000, St. Petersburg, Russia
E-mail: zaburdinslv@mail.ru

Konovalev Daniil Vladislavovich –

Student of the Department of Aerospace Computer and Program Systems
St. Petersburg State University of Aerospace Instrumentation
67 Bolshaya Morskaya St., St. Petersburg, Lit. A, 190000, Saint-Petersburg
E-mail: danil.konovalev.70@gmail.com



Shakhomirov Andrey Viktorovich –

Associate Professor of the Department of Aerospace Computer and Program Systems, Ph.
St. Petersburg State University of Aerospace Instrumentation

67 Bolshaya Morskaya St., St. Petersburg, Lit. A, 190000, St. Petersburg, Russia.

E-mail: shakhomirov@gmail.com