ПРОГРАММНЫЙ КОД ДЛЯ РЕАЛИЗАЦИИ ПОВЕДЕНИЯ RTS КАМЕРЫ НА UNITY

А.О. Савельев, к.т.н., доц. ОИТ ИШИТР,
А.А. Сиротин, студент гр. 8К71 ОИТ ИШИТР
Томский политехнический университет
E-mail: aas244@tpu.ru

Введение

Сектор игровой индустрии является одним из крупнейших на данный момент, а также становится всё более доступным для начинающих разработчиков игр. Существуют целые площадки, куда пользователи могут выкладывать свои разработки для их дальнейшего использования в создании собственных программ и игровых приложений. В частности, для игр, жанра стратегия необходима реализация RTS (Real Time Strategy) камеры.

При создании игрового приложения в жанре стратегия на игровом движке Unity не был найден скрипт, описывающий поведение RTS камеры, который бы удовлетворял всем потребностям игрового приложения. Именно поэтому целью данной работы являлось написание программного кода для перемещения RTS в пространстве с гибкой системой настроек параметров.

Описание алгоритма

Основные требования к RTS камере:

- Удобный способ обзора плоской поверхности с изменением положения в пространстве
- Масштабирование
- Поворот камеры по одной или двум осям координат
- Дополнительные требования RTS камеры для разрабатываемого приложения:
- Возможность изменения угла обзора камера (от 30 до 150 градусов)
- Автоматически настраиваемые границы перемещения камеры в зависимости от её положения в пространстве
- Возможность сохранения положений камеры в пространстве для быстрого переключения между ними
- Возможность управления мышью и клавиатурой
- Настройка всевозможных параметров камеры, такие как: скорость перемещения, скоростью вращения, скорость зума, задание клавиш управления, угол наклона к плоскости, режимы вращения (вокруг собственной оси, вокруг точки в пространстве)

Главным отличием от аналогов является наличие автоматически настраиваемых границы камеры в зависимости от положения в пространстве и возможность сохранения положений камеры, которых не было обнаружено ни в одном из существующих аналогов для Unity на площадке AssetStore.

Для реализации передвижения камеры при помощи мыши использовалась система из 4-х прямоугольных областей (Рисунок 1), при попадании курсора в одну из областей камера начинает двигаться в соответствующем направлении параллельно горизонтальной плоскости.

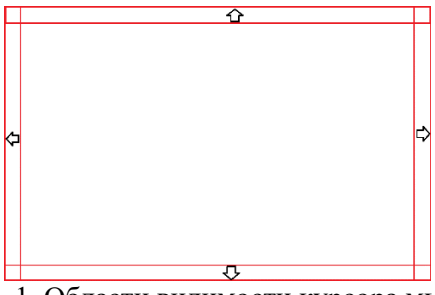

Рис. 1. Области видимости курсора мыши

Для реализации перемещения с помощью клавиатуры использовался стандартный функционал Unity, такой, как Input.GetAxis(), который получает в качестве аргумента строку, которая указывает на то, вдоль какой оси будет осуществляться перемещение.

Масштабирование реализовано через базовые методы пакета Unity - Mathf. В частности, использовались методы Clamp и Lerp. Метод Clamp необходим для ограничения максимальной и

минимальной величины масштабирования. Метод Lerp необходим для линейной интерполяции высоты камеры.

Для вращения камеры используется базовый функционал Unity, такой, как Rotate и RotateAround, для вращения вокруг собственной оси и вокруг точки в пространстве соответственно. Точка в пространстве, вокруг которой происходит вращение камеры определяется посредством использования стандартного класса Unity – Ray и преобразованием координат центра экрана камеры в луч (ScreenPointToRay), направленный к плоскости. Величина луча вычисляет по теореме Пифагора.

Для изменения величины границы камеры, в зависимости от масштаба, использовалось подобие треугольников. Для этого берётся пересечение области профильной плоскости камеры и линии горизонта и из теоремы подобия высчитывается величина, на которую может переместиться камера в горизонтальной плоскости. При этом границы положения камеры смещаются в пространстве в зависимости от угла наклона к плоскости, чтобы этого не происходило используется формула для определение ближней границы камеры (1).

$$b = \frac{(hmax - h)}{tg(\beta)} \quad , \tag{1}$$

где b – величина ближней границы камеры
$hmax$ – максимальная высота камеры
h – текущая высота камеры
β – угол между нижней границей обзора камеры и плоскостью

Для определения границ камеры при повороте используется уравнение вращения точки относительно другой в пространстве (2) и изменение границ на величину вращения. При этом вычисление границ камеры и её текущего положения выполняется уже после её вращения и перемещения с использованием метода Clamp, о котором говорилось ранее.

$$\begin{aligned} x' &= x_0 + (x - x_0)\cos\alpha - (y - y_0)\sin\alpha \\ y' &= y_0 + (x - x_0)\sin\alpha + (y - y_0)\cos\alpha, \end{aligned} \tag{2}$$

где x – начальная позиция точки по оси x
y – начальная позиция точки по оси y
x_0 – позиция точки вращения по оси x
y_0 – позиция точки вращения по оси y
x' - новая позиция точки по оси x
y' - новая позиция точки по оси y
α – угол поворота камеры

Для сохранения информации о положении камеры используются горячие клавиши на клавиатуре, которые пользователь может сам установить, которые сохраняют информацию о позиции и вращении объекта, для этого используется Vector3 и Quaternion, которые являются базовыми классами в Unity.

Заключение

В статье изложены наработки по написанию программного кода для реализации поведения RTS камеры, которая расширяет базовый функционал стандартной RTS камеры. Отличительные способности позволяют удобней взаимодействовать с камерой и направлены на игровые приложения в жанре стратегия, где есть необходимость просматривать всю карту целиком и устанавливать точки с положением камеры в пространстве.

Список использованных источников

1. Официальная документация по программирования на Unity [Электронный ресурс]. – URL: https://docs.unity3d.com/ru/530/ScriptReference/index.html
2. Площадка для публикации программных продуктов для Unity [Электронный ресурс]. – URL: https://assetstore.unity.com/