УДК 004.415.25

РЕАЛИЗАЦИЯ ВЕРТИКАЛЬНОГО ПЕРЕМЕЩЕНИЯ В ПРИЛОЖЕНИИ ВИРТУАЛЬНОЙ РЕАЛЬНОСТИ

М.А. СЕРГЕЕВ
(Представлено: канд. техн. наук, доц. А.Ф. ОСЬКИН)

В статье представлен практический способ реализации механизма вертикального перемещения в приложениях виртуальной реальности разрабатываемых на Unreal Engine 4. Решена задача реализации перемещения игрока по вертикальным поверхностям.

Ключевые слова: *информационные технологии, виртуальная реальность, Unreal Engine 4.*

Введение. Игровые приложения виртуальной реальности часто базируются на взаимодействии с мелкими деталями и объектами виртуальных миров. Примеров взаимодействия игрока с игровым пространством в более глобальном смысле, имеется не так много. В основном это связано с ограничениями, накладываемыми реализацией vr-приложений. Кроме того, необходимо снизить риск дезориентации игрока в пространстве, что чревато возникновением эффекта «укачивания» игрока.

Однако, возможность взаимодействия игрока с таким глобальным объектом игрового мира, как он сам, вполне возможна и без укачивания.

В данной статье будет представлена надежная реализация перемещения игрока в виртуальной реальности по вертикальным поверхностям.

Реализация вертикального перемещения. Игровым движком для реализации приложения был выбран Unreal Engine 4. Таким образом, разработка приложения будет вестись при помощи Blueprints, графического языка программирования.

Приложение будет состоять из нескольких функциональных частей, которые взаимодействуют между собой. Компонент контроллера, хранящий в себе 3d-модель контроллера и его анимации BP_MotionController [1], компонент игрока MotionControllerPawn [2], хранящий в себе весь основной функционал пользователя, интерфейс, связывающий игрока и компоненты, с которыми он взаимодействует, и компонент поверхностей для вертикального перемещения BP_ClimbObj, хранящий в себе модель примитива поверхности для вертикального перемещения, а также специально созданный материал для упрощенного визуального восприятия объектов этого типа. Помимо этого, компонент BP_ClimbObj хранит в себе логику перемещения по вертикальным поверхностям (рисунки 1–3).

Логика подобного перемещения следующая: у игрока имеется два контроллера, каждый из которых взаимодействует с объектом определенного типа (специальной стеной для перемещения). В момент начала взаимодействия, вся модель игрока начинает двигаться относительно точки, в которой началось взаимодействие контроллера и объекта, данное движение происходит с помощью контроллера, модель которого зафиксирована в вышеуказанной точке.

На рисунке 1 представлено событие, определяющее точку отсчета, относительно которой двигается игрок. Помимо этого, в данном событии также обновляется переменная, в которой ведется счет количества контроллеров, который взаимодействуют с объектом.

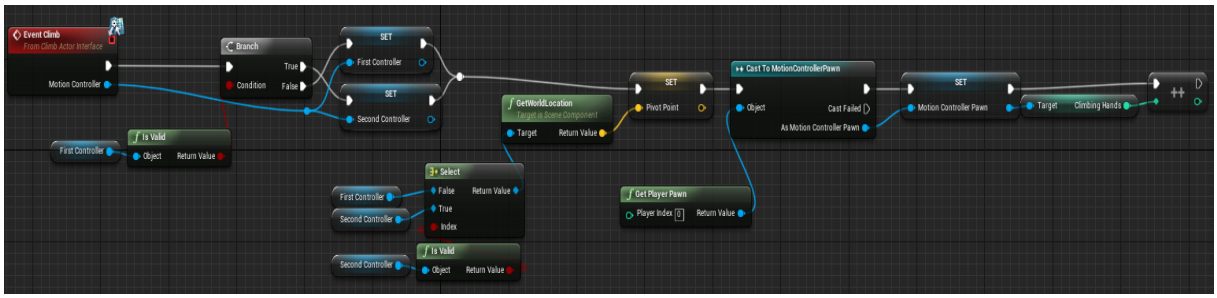

Рисунок 1. – Событие, определяющее точку отсчета

На рисунке 2 представлено событие, определяющее момент, когда взаимодействие прекращается. В нем также предусмотрен вариант развития событий, при котором во время взаимодействия одного контроллера с поверхностью, происходит взаимодействие другого контроллера. Соответственно, точка отсчета меняется, и переменная-счетчик взаимодействующих контроллеров обновляется снова.

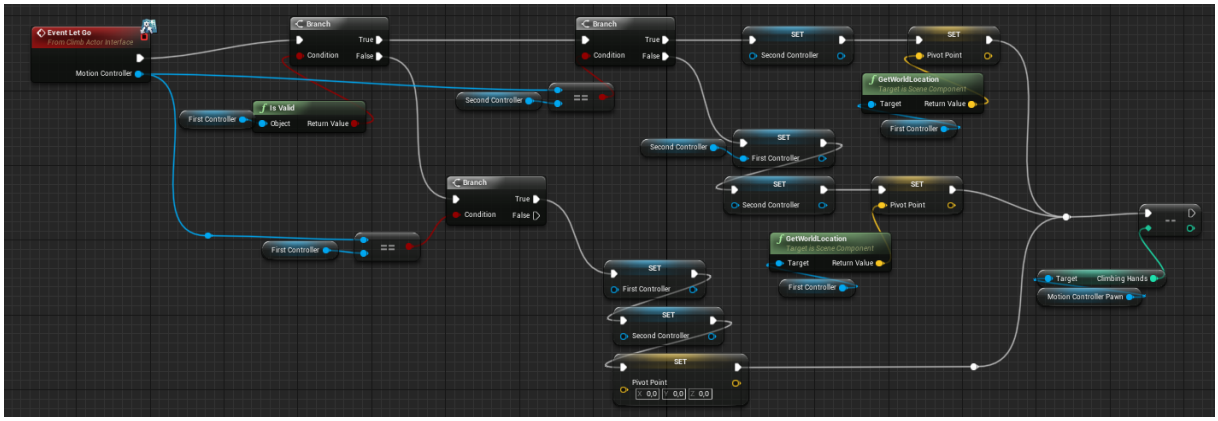

Рисунок 2. – Событие прекращения взаимодействия

На рисунке 3 продемонстрировано обновление позиции игрока в реальном времени. Оно проверяет переменную Is Climbing, принимающую значение true, в момент взаимодействия контроллера с поверхностью, и хранящуюся в компоненте MotionControllerPawn и валидность контроллера и самого игрока. После данной проверки инициируется перемещение модели игрока при помощи узла AddActorLocalOffset, который за точку отсчета берет точку, в которой началось взаимодействие контроллера с поверхностью. Узел же, который следует за ним, возвращает контроллер игрока в эту точку.

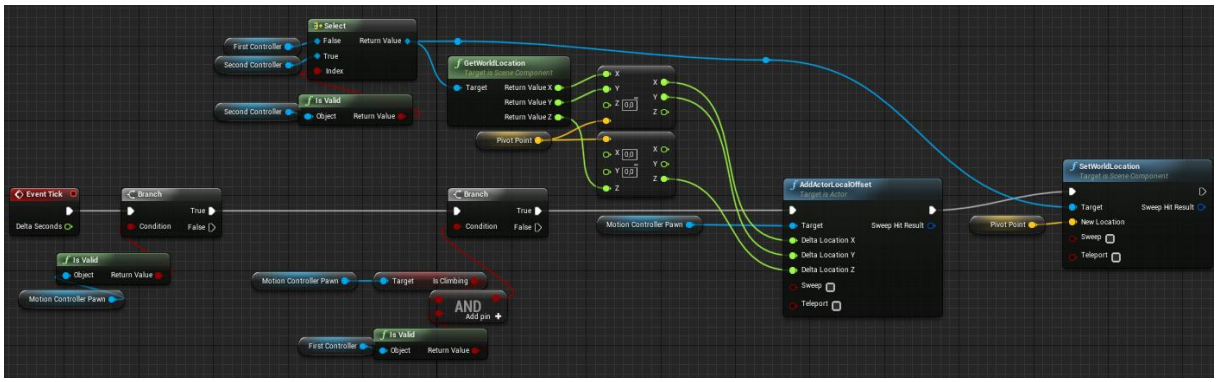

Рисунок 3. – Логика обновления позиции игрока в 3d-пространстве

На рисунках 4-6 продемонстрирован код падения игрока, после окончания взаимодействия с поверхностью перемещения. Располагается данная часть кода в компоненте MotionControllerPawn.

На рисунке 4 продемонстрированы проверки, определяющие взаимодействует ли хоть один контроллер с поверхностью, либо же нет. В случае, если взаимодействующих контроллеров больше нуля, переменная IsClimbing присваивает значение true, а IsFalling – false. В ином случае, идет дополнительная проверка вышеуказанных переменных, приводящая либо к присваиванию переменной взаимодействующих контроллеров значение 0, либо присвавивание переменной IsClimbing значение false, а IsFalling – true. При повторной проверке значение переменной IsFalling true проинициирует начало падения.

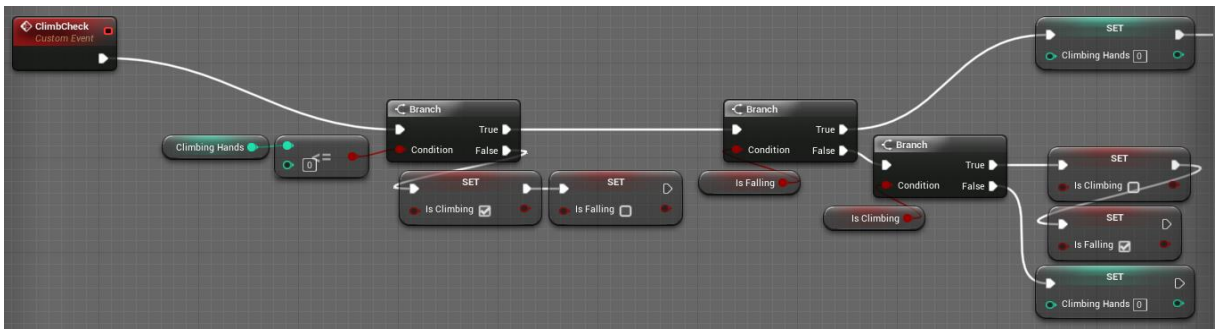

Рисунок 4. – Событие падения ч. 1

На рисунке 5 продемонстрирована проверка, определяющая, находится ли игрок уже на поверхности при помощи узла LineTraceByChannel. В случае, если игрок уже находится на такой поверхности, IsClimbing и IsFalling принимают значение false, а скорость падения (falling speed) становится равной нулю.

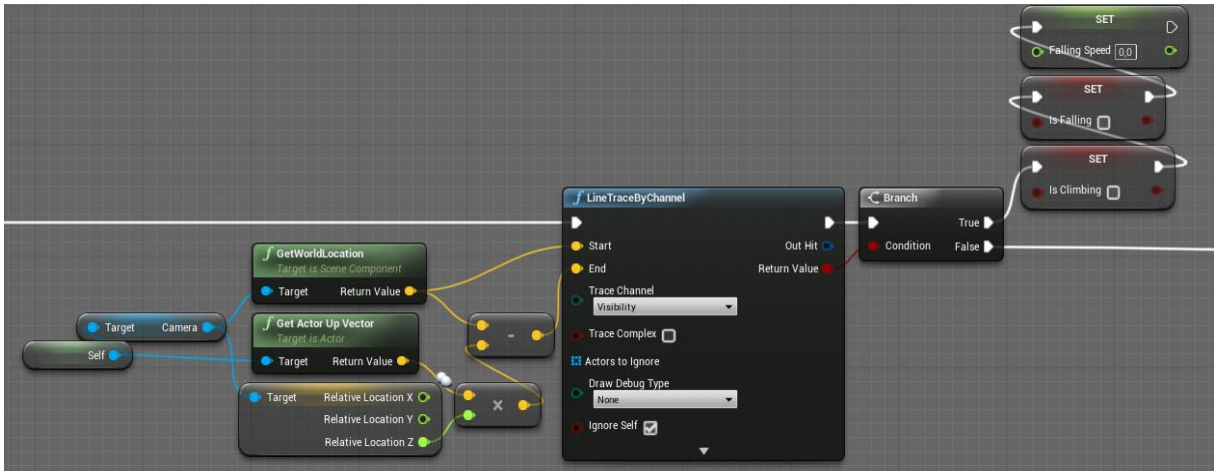

Рисунок 5. – Событие падения ч. 2

На рисунке 6 продемонстрированна математическая часть реализации падения виртуальной модели игрока. Переменной Falling Speed, пределяющей значение скорости падения присваивается ускорение, вычисляемое по значениям земной гравитации. Данная переменная ограничивается значением 300,0 и умножается на значение времени. Полученное число перемножается с инвентированным верхним вектором игрока, что переходит в узел AddLocalOffset, который, в свою очередь, приводит в движение модель игрока.

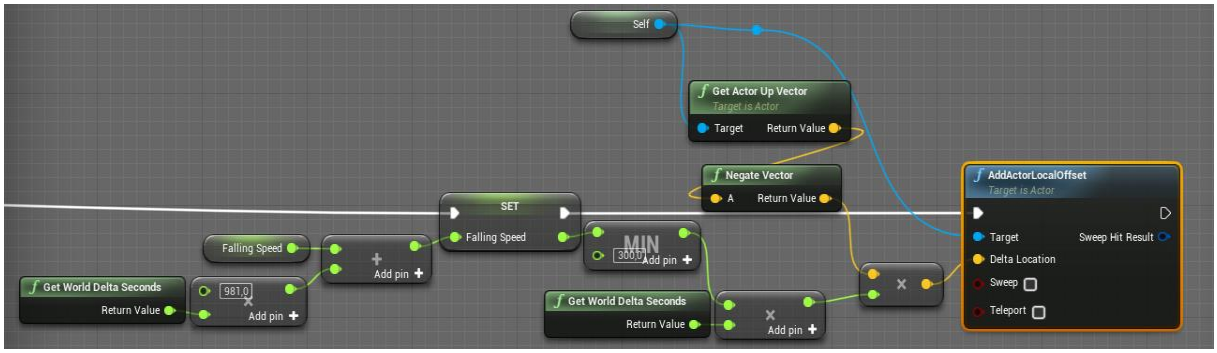

Рисунок 6. – Событие падения ч. 3

Заключение. Приложения виртуальной реальности могут базироваться на обширном количестве геймплейных концепций, влияющих на восприятие пользователем игрового мира, а также, затрагивающих, в том числе, столь базовую вещь, как тип передвижения пользователя в виртуальном мире. В данной статье была продемонстрирована реализация самой новаторской концепции перемещения модели игрока, а именно, физический тип перемещения, согласно которому игрок перемещает свою модель не только при помощи нажатий клавиш контроллера, но и при помощи реальных физических действий, совершаемых его телом [3].

Рассмотрены инструменты игрового движка Unreal Engine 4, позволяющего создавать приложения практически любого уровня сложности, а также удобного в создании приложений виртуальной реальности. По итогам данной работы, был описан метод создания модуля физического вертикального перемещения игрока.

ЛИТЕРАТУРА

1. Unreal Engine 4 Documentation [Электронный ресурс]. – Режим доступа: https://docs.unrealengine.com/en-us/ – Дата доступа: 10.09.2019.

2. Unreal Engine 4 Документация [Электронный ресурс]. – Режим доступа: https://uengine.ru/docs – Дата доступа: 10.09.2019.
3. Unreal Engine VR для разработчиков / Митч Маккефри; [пер. с англ. Н.И. Веселко, О.В. Максименковой, А.А. Незнанова]. – Москва : Эксмо, 2019. – 256 с.