УДК: 794.9:004.928(091)(470+100)
DOI: 10.66523/gici.2026.v1.5

НАУЧНАЯ СТАТЬЯ OPEN ACCESS

LBE-калибровка проще, чем Вы думаете: простейший подход к настройке положения игрока для LBE-проекта на базе гарнитуры Meta Quest 3 и игрового движка Unreal Engine 5

Афанасьев В.А.[1], Тясто. С.А.[1]

[1] ФГАОУ ВО «МГТУ «СТАНКИН», Кафедра автоматизированных систем обработки информации и управления

Автор для переписки: Афанасьев В.А.| afanasiev@gamedev.science

Поступила в редакцию: 28.10.2025 | **Принята к публикации:** 08.02.2026 | **Опубликована:** 28.04.2026

Ключевые слова: развлекательные системы на основе местоположения, виртуальная реальность, разработка игр, шлем виртуальной реальности, пространственная калибровка.

Финансирование: *исследование не имело спонсорской поддержки.*

Конфликт интересов: *авторы заявляют об отсутствии конфликта интересов.*

Аннотация

В данной статье представлен практический подход к пространственной калибровке для проектов развлекательных систем на основе местоположения (LBE) в виртуальной реальности (VR) на основе автономной гарнитуры Meta Quest 3 и фреймворка Unreal Engine 5. Предложенный метод двухточечной настройки выравнивает местоположение игрока на виртуальном уровне с координатной системой физической арены LBE. Используются простейшие геометрические операции и встроенные методы. В качестве промежуточного этапа вводится специальный уровень калибровки, где оператор записывает положения опорных точек и желаемое смещение поворота камеры. Параметры калибровки хранятся локально на каждой гарнитуре и автоматически переключаются между картами. В статье обсуждаются специфические для LBE ограничения безопасности и показано, что рассматриваемый конвейер может быть интегрирован в различные игровые проекты с минимальной перегрузкой.

LBE calibration Is easier than you think: a simple approach to setting up player position for an LBE project using the meta Quest 3 headset and Unreal Engine 5

Afanasiev B.A., Tiasto S.A. Corresponding author: Afanasiev B.A. | afanasiev@gamedev.science

This paper presents a practical approach to spatial calibration for location-based entertainment (LBE) project in virtual reality (VR) based on the autonomous Meta Quest 3 headset and the Unreal Engine 5 framework. The proposed two-point setup method aligns player location in virtual level with the coordinate system of physical LBE-arena. Simplest geometric operations and built-in methods are used. A dedicated calibration level is introduced as interim stage, where an operator records anchor point positions and desired camera rotation offset. Calibration parameters are stored locally on each headset and automatically transitioned between maps. The paper discusses LBE-specific constraints on safety and shows that the considered pipeline can be integrated into different game projects with minimal overload.

Keywords*: location-based entertainment, virtual reality, game development, head-mounted display, spatial calibration.*

Для цитирования: *Афанасьев В.А., Тясто. С.А. LBE-калибровка проще, чем Вы думаете: простейший подход к настройке положения игрока для LBE-проекта на базе гарнитуры Meta Quest 3 и игрового движка Unreal Engine 5. Международное сетевое научное рецензируемое издание "Геймификация, игры, креативные индустрии" 2026. Т. 1, № 1. С. 40-45 https://doi.org/10.1155/10.66523/gici.2026.v1.5 / For citation: Afanasiev B.A., Tiasto S.A. LBE calibration Is easier than you think: a simple approach to setting up player position for an LBE project using the meta Quest 3 headset and Unreal Engine 5. International Peer-Reviewed Online Scientific Journal "Gamification, Games, Creative Industries", 2026, vol. 1, no. 1, pp. 40-45. https://doi.org/10.66523/gici.2026.v1.5*

Введение

В последние годы виртуальная реальность применяется все в большем количестве различных областей: от производства до развлечений. Широкое распространение получил формат LBE (от англ. Location-based entertainment) за счет возможности свободного передвижения игроков в реальном пространстве при сохранении высокого уровня погружения в виртуальную среду. Данный вариант реализации требует точного совпадения цифрового и физического пространств в целях обеспечения, прежде всего, безопасности, а также требуемого качества игрового опыта [5].

Наиболее популярным выбором периферии является гарнитура Meta Quest 3 с автономной работой благодаря отсутствию проводов и внешних базовых станций, как, например, у HTC Vive Pro. Однако следует заметить, что штатные механизмы настройки границ и опции калибровки ориентированы на домашний сценарий и не предоставляют возможность постоянной привязки к координатам игровой арены, особенно в случае сценариев с несколькими устройствами [2].

Цель работы – разработка наиболее простого подхода к двухточечной калибровке пространства для LBE-арен на базе гарнитуры Meta Quest 3 и игрового движка Unreal Engine 5. Предлагаемый способ базируется на использовании простых геометрических вычислений с минимальным количеством сложных математических расчетов.

Практическая значимость подхода заключается в снижении трудоемкости подготовки игровых LBE-сессий и уменьшении ошибок калибровки без вмешательства в низкоуровневые особенности настроек трекинга шлемов виртуальной реальности.

Постановка задачи

LBE-сессия подразумевает перемещение игроков по виртуальному пространству за счет их физического передвижения по игровой арене. Такой подход значительно повышает уровень погружения пользователей в симуляцию, что выгодно отличает его от метода на основе джойстиков. Для обеспечения корректного взаимодействия с виртуальным окружением требуется наиболее точное совпадение позиций игроков на виртуальной сцене с их положением в реальном пространстве в пределах допустимой погрешности. Нарушение этого соответствия повышает риск физических столкновений пользователей и ухудшает качество игрового опыта [1].

Важно отметить, что геометрия виртуальной сцены должна быть спроектирована так, чтобы в точности воспроизводить конфигурацию физической площадки. Следует исключить вероятность угрозы вреда здоровью или целостности оборудования в результате непосредственного контакта.

Для формального описания задачи можно ввести три условных системы координат:

· Sphys – система координат физической площадки (арены). Начало координат выбирается в заданной опорной точке, например, в углу зала, оси направлены вдоль стен. Плоскость XY совпадает с уровнем пола.
· Scalibration – система координат калибровочного уровня в Unreal Engine 5. Представляет собой игровой ассет – карту, на котором фиксируются положения опорных точек. Обычно не содержит никаких игровых объектов для снижения нагрузки на оперативную память, на ней загружаются различные технические виджеты.
· Slevel – система координат игрового уровня в Unreal Engine 5. На данной карте размещаются игровые стены, колонны и различные интерактивные объекты. Геометрия данного уровня должна совпадать с физическими стенами и колоннами в реальности.

Таким образом, задача калибровки сводится к преобразованию системы координат игры для согласования с системой координат физического пространства. Предполагается, что игроки передвигаются по плоской поверхности, что подразумевает настройку поворота камеры, сдвиг по осям и при необходимости масштабирование [3].

Специфика LBE-сценариев накладывает существенные ограничения на дизайн игрового процесса по сравнению с домашним VR-сценарием с использованием контроллеров или жестов. В условиях общей физической арены, когда люди передвигаются самостоятельно, недопустимы резкие телепорты, агрессивные ускорения и дополнительные смещения камеры, поскольку это может привести к потере ориентации в пространстве, непреднамеренным столкновениям и дискомфорту.

В результате целый ряд геймплейных механик, привычных для комнатной или стационарной виртуальной реальности становится категорически неприменим в условиях LBE. Проектирование уровней и игровых возможностей должно исходить из физически реализуемых траекторий движения игроков с ограничениями по скорости и строгих требований к безопасности [4].

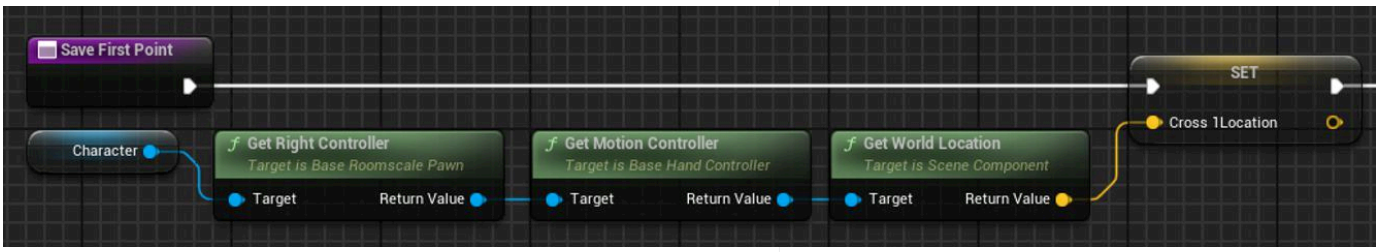
Метод двухточечной калибровки

Предлагаемый метод калибровки основывается на использовании двух опорных точек, расположенных на физической площадке, для которых известны координаты как в реальном пространстве, так и в виртуальной сцене. Отметим, что это предполагает проведение непосредственных измерений арены и согласование размещения всех необходимых маркеров и объектов. Для описания метода необходимо и достаточно набора элементарных геометрических операций и стандартных математических нод Unreal Engine 5, таких как FindLookAtRotation(), ResetOrientationAndPosition(), GetDistanceTo() и прочих.

В системе координат Slevel заранее выбираются две опорные точки D и E, лежащие на одной линии. Эти точки представляются в виде простых Actors на уровне, их положение известно и фиксировано. Затем оператор в ходе калибровки должен последовательно поместить контроллер в соответствующие точки на физической площадке в системе Sphys, чтобы система могла зарегистрировать точки A и B соответственно. Координаты упомянутых точек сохраняются в системе координат Scalibration промежуточного уровня, который загружается перед процессом фиксирования маркеров.

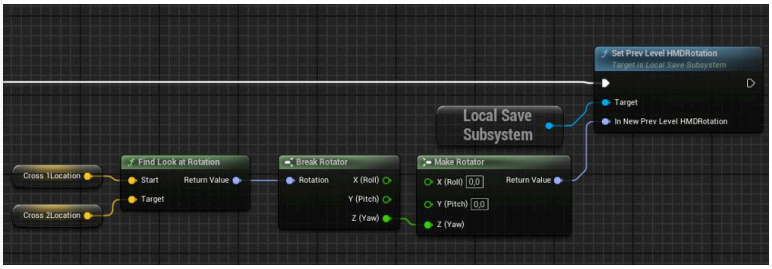
Расстояние между двумя опорными точками на сцене можно вычислить с помощью стандартных средств движка. Система визуального программирования Blueprints предоставляет функцию GetWorldLocation() для определения мирового положения. Отношение расстояний между точками при необходимости используется как коэффициент масштабирования, позволяющий компенсировать возможные различия размеров. Также необходимо запомнить координаты положения камеры игрока (точка C) для дальнейших вычислений.

На рис. 2.1 представлено получение координат опорной точки.



Направление отрезка, построенного по опорным точкам, задает ориентир для выравнивания виртуальной сцены относительно физической. Здесь важно отметить, что не существует понятия «поворот уровня», поскольку карта не имеет какой-либо трансформации. Повороту подлежит камера игрока и прочие элементы в иерархии объекта класса Pawn или Character, который имеет определенное положение в виртуальном пространстве. На данном этапе следует найти компонент Z (Yaw) структуры Rotator между опорными точками с помощью метода FindLookAtRotation(), что пригодится в настройке Rotation на игровом уровне.

На рис. 2.2 представлен расчет направления выравнивания.



Перенос калибровочных данных

Для разделения калибровочной логики и игрового контента используется отдельный промежуточный уровень, предназначенный исключительно для процедуры калибровки. При запуске приложения оператор должен иметь возможность перейти на данную карту, где должны быть размещены интерфейсные подсказки и логика работы с контроллерами. Например, для удобства при фиксировании положения опорных точек можно создавать статические мешки «вешек» в виртуальном пространстве для визуального отклика.
Следует понимать, что данные, загруженные в оперативную память на одном уровне, не будут автоматически передаваться на следующий. Обычно в таких случаях используется GameInstance – синглтон, который создается при запуске приложения и не удаляется между картами, доступен с помощью функции GetGameInstance(). Однако следует обратить внимание, что при перезагрузке виртуальной гарнитуры вся динамическая информация будет стерта, в том числе и данные в GameInstance. По этой причине наиболее простым и удобным способом хранения и передачи калибровочных данных будет встроенная в Unreal Engine система SaveGame.

На каждом шлеме будет храниться локальная конфигурация, настроенная именно для конкретного шлема. Это предполагает проведение отдельных процедур калибровки для разных гарнитур, что значительно увеличивает количество затрачиваемого на подготовку площадки времени. Для ускорения процесса можно загружать настройки на единый сервер, к которому подключены все шлемы, поскольку LBE предполагает синхронизацию данных между устройствами в локальной сети, однако данный метод более трудоемок в реализации.

Применение транформаций

На основе координат, полученных во время этапа фиксирования опорных точек, вычисляется целевое положение объекта Character игрока и требуемый корректирующий поворот вокруг оси Z (Yaw). Начнем с корректировки положения персонажа. Обратим внимание, что на игровом уровне в системе координаты Scalibration должны быть расположены уровневые точки на таком же расстоянии и направлении, на каком были зафиксированы опорные точки в системе координаты Sgame.

Используем следующие обозначения:

- Point A – координаты первой опорной точки.
- Point B – координаты второй опорной точки.
- Point C – положение камеры игрока.
- Point D – координаты первой уровневой точки.
- Point E – координаты второй уровневой точки.
- Point C’ – координаты целевой точки.

На рис. 4.1 представлена схема калибровки.

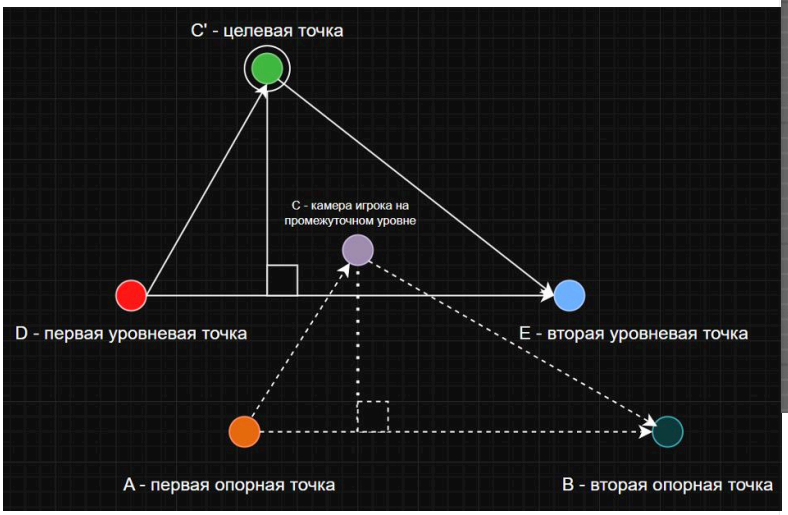


Рис. 4.1 представлена схема калибровки.

Длину вектора можно найти по следующей формуле (в Blueprints – функция VectorLength()):

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \qquad (1)$$

Скалярное произведение векторов (в Blueprints – функция Dot()):

$$\bar{a} * \bar{b} = |\bar{a}| * |\bar{b}| * cos\,\alpha \qquad (2)$$

$$\bar{a} * \bar{b} = a_{x^*} b_x + a_{y^*} b_y \qquad (3)$$

Формула проекции вектора на вектор:

$$\Pi p_{\bar{b}} \bar{a} = \frac{\bar{a} * \bar{b}}{|\bar{b}|} \qquad (4)$$

Определить, с какой стороны от отрезка находится точка, можно с помощью векторного произведения векторов (в Blueprints – функция Cross()):

$$|\bar{a} \times \bar{b}| = |\bar{a}| * |\bar{b}| * sin\,\theta \qquad (5)$$

$$\bar{a} \times \bar{b} = \begin{vmatrix} \bar{i} & \bar{j} & \bar{k} \\ a_x & a_y & a_z \\ b_x & b_y & b_z \end{vmatrix} \qquad (6)$$

AC – отрезок между первой опорной точкой и точкой камеры, AB – отрезок между опорными точками. Проекция вектора AC на AB позволит вычислить расстояние, на которое необходимо сдвинуть целевую точку от первой опорной точки.

На рис. 4.2 представлен расчет длины проекции вектора AC на AB.

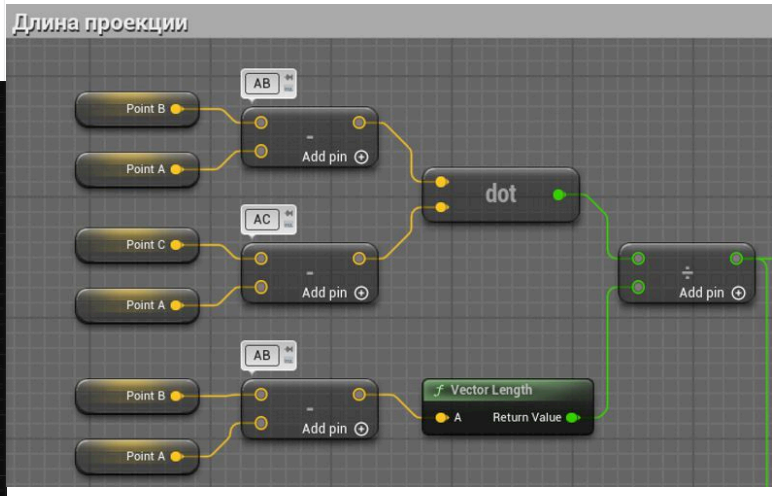


Рис. 4.1. Расчет длины проекции

Сам сдвиг необходимо производить с учетом направления отрезка между опорными точками. Здесь нам поможет значение функции GetUnitDirection(), умноженное на длину проекции.

На рис. 4.3 представлен сдвиг целевой точки с учетом первой опорной точки.

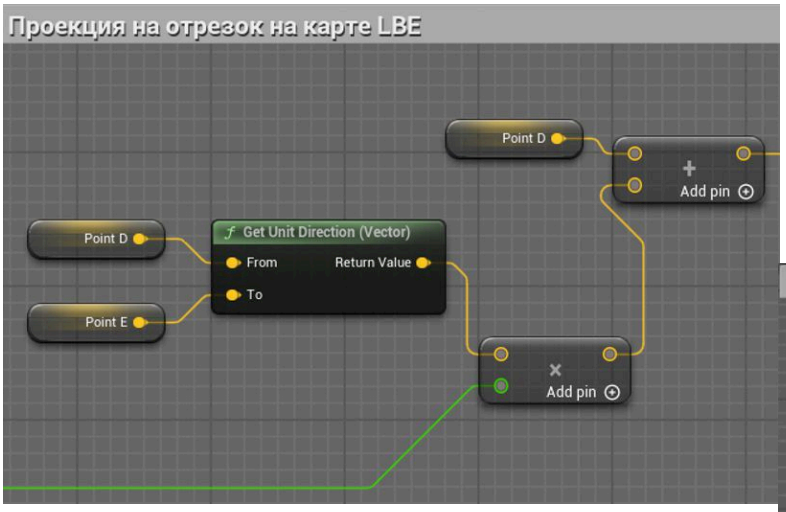


Рис. 4.3. Сдвиг целевой точки с учетом первой опорной точки

Теперь следует опустить перпендикуляр от точки камеры C к отрезку между опорными точками AB. Найдем его длину по упомянутой ранее формуле. Функция Power() – возведение в степень, Sqrt() – извлечение квадратного корня.

На рис. 4.4 представлен расчет длины перпендикуляра к отрезку AB.

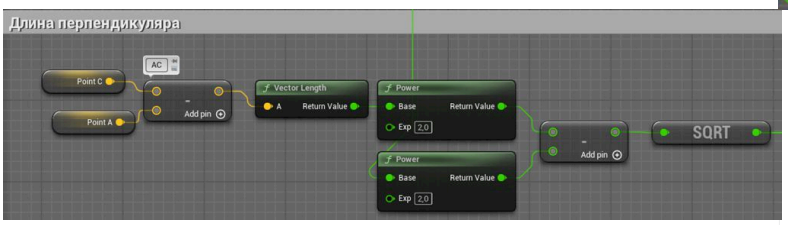


Рис. 4.4. Расчет длины перпендикуляра к отрезку AB

Важным моментом является учет того, с какой стороны от отрезка встает оператор в процессе калибровки. От этого зависит направление сдвига целевой точки от отрезка AB. Blueprints предоставляет функции Cross() и Select() для расчета векторного произведения и выбора потока данных по булевой переменной соответственно. Для инвертирования вектора можно использовать метод NegateVector().

Таким образом:

· Векторное произведение > 0 – точка находится слева.
· Векторное произведение < 0 – точка находится справа.
· Векторное произведение = 0 – точка находится на самом отрезке.

На рис. 4.5 представлен расчет направления сдвига целевой точки по перпендикуляру

Таким образом, ключевым фактором успеха является не только наличие данных, но и способность их адекватного анализа и интеграции в процесс принятия решений.

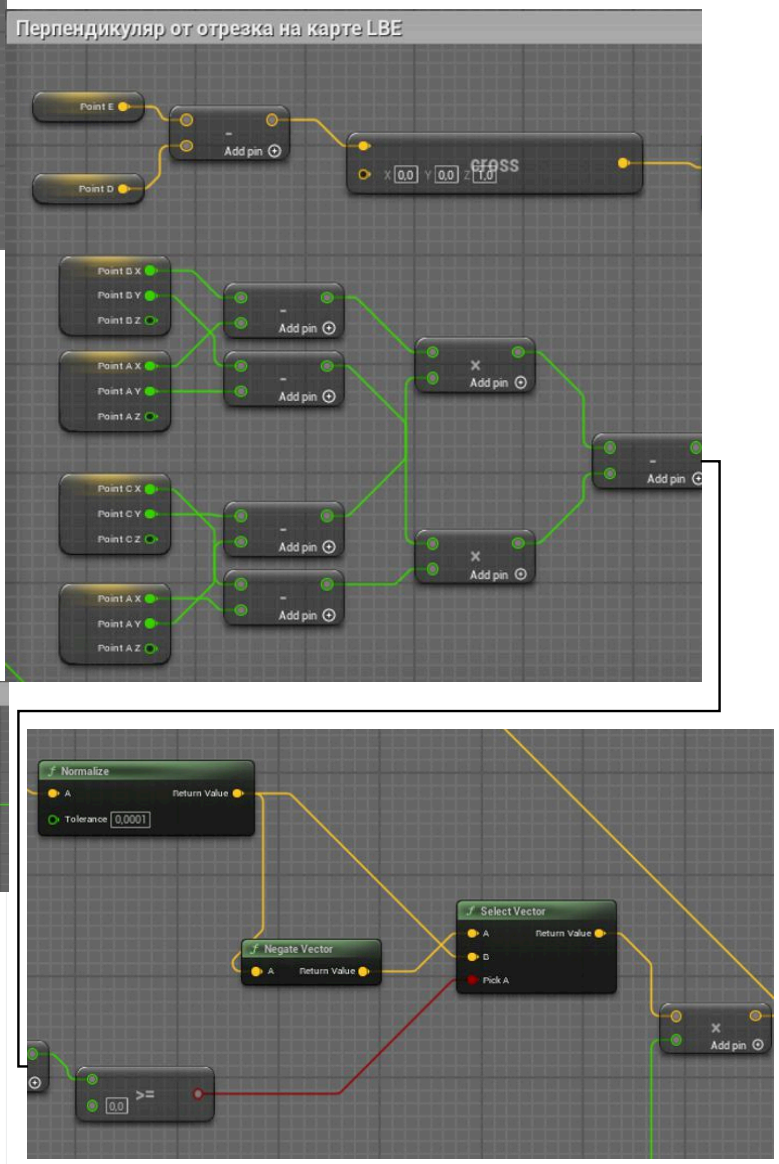


Рис. 4.5. Расчет направления сдвига целевой точки по перпендикуляру

Далее необходимо найти финальные координаты целевой точки с учетом вычисленных ранее сдвигов.

На рис. 4.6 представлен расчет координат целевой точки.

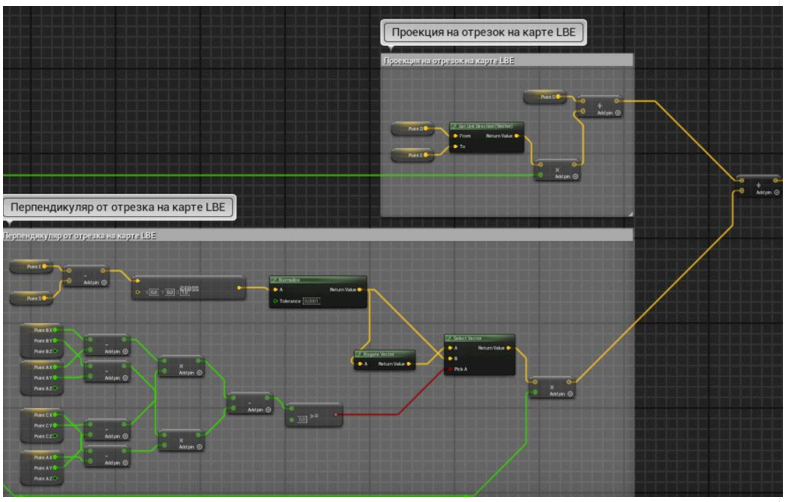


Рис. 4.6. Расчет координат целевой точки

Наконец, остается скорректировать поворот камеры игрока. Направление выравнивания камеры уже было рассчитано ранее и сохранено в промежуточный буфер. С помощью функции GetOrientationAndPosition() получаем текущий Rotator устройства и находим разницу между ним и зафиксированным ранее значением. Метод ResetOrientationAndPosition() позволяет выполнить сброс вида и задать новую точку отсчета поворота для камеры игрока.

На рис. 4.7 представлено выравнивание поворота камеры относительно уровня.

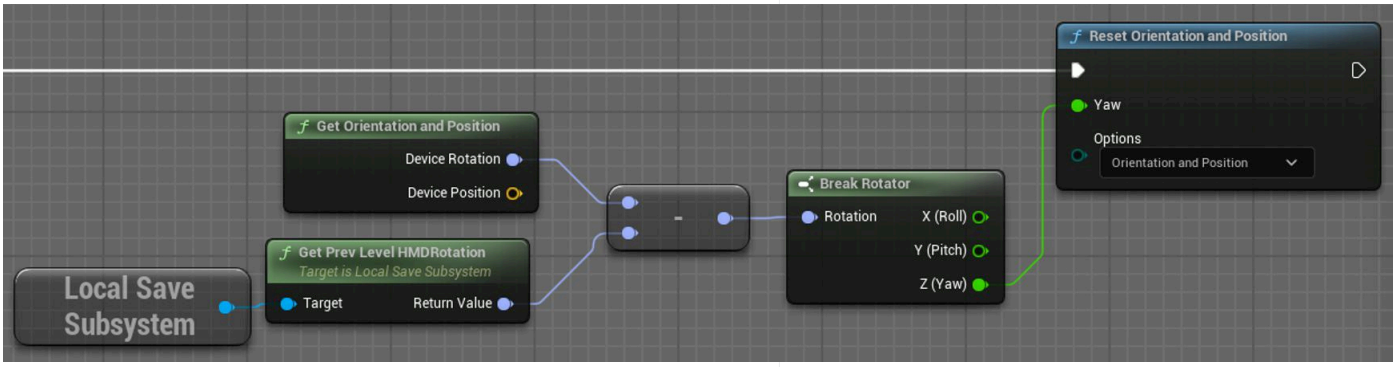


Рис. 4.7. Выравнивание поворота камеры относительно уровня

Теперь следует подтвердить корректность выполненных вычислений и проверить соответствие виртуальных стен и объектов физическим. Если случайных столкновений не произошло, то процесс калибровки можно считать завершенным. Обратим внимание, что настройка роста и положения контроллеров в данной работе не затрагиваются.

Заключение

В данной работе рассматривается наиболее простой подход к калибровке VR-устройств для LBE-сценариев на базе Unreal Engine 5. Предложенный метод двухточечной калибровки опирается на стандартный инструментарий движка и соответствующих XR-плагинов с минимальным применением сложных геометрических вычислений и формул, что значительно упрощает и ускоряет процесс внедрения в проект. Сформирована формальная постановка задачи согласования систем координат Sphys, Scalibration и Sgame. Обоснованы требования и перечислены ограничения геймплейных механик с учетом особенностей игрового процесса в условиях LBE. Реализована система получения и хранения координат опорных точек для корректной передачи данных между уровнями. Вычислены и применены необходимые трансформации для обеспечения соответствия положения игрока в виртуальном пространстве физическому.
Представленная методика может использоваться для построения более сложных систем калибровки, включающих учет погрешностей в процессе фиксирования координат маркеров, более точную настройку поворота камеры и синхронизацию единой конфигурации между несколькими устройствами в пределах локальной сети.

Список литературы

[1] Designing a Free Roam Space That Actually Works // Synthesis VR, Pioneering The VR Location Based Entertainment Industry. - 2026.
[2] Jean-Luc Lugrin, Florian Kern, Constantin Kleinbeck, Daniel Roth, Christian Daxer, Tobias Feigl, Christopher Mutschler, Marc Erich Latoschik A Framework for Location-Based VR Applications // University of Würzburg, Germany. - 2019.
[3] Mark McGill, Jan Gugenheimer, Euan Freeman A Quest for Co-Located Mixed Reality: Aligning and Assessing SLAM Tracking for Same-Space Multi-User Experiences // VRST '20: Proceedings of the 26th ACM Symposium on Virtual Reality Software and Technology Article No.: 26, Pages 1 - 10. - 2020.
[4] Matthias Wölfel, Daniel Hepperle, Andreas Siess, Jonas Deuchler Staging Location-Based Virtual Reality to Improve Immersive Experiences // Faculty of Computer Science and Business Information Systems, Karlsruhe University of Applied Sciences, Karlsruhe, Germany. - 2019.
[5] Ruth West, Eitan Mendelowitz, Zach Thomas, Christopher Poovey, Luke Hillard, Kathryn Hays, Nathaniel Helgesen Designing a VR Arena: Integrating Virtual Environments and Physical Spaces for Social Sensorial Data-Driven Experiences // University of North Texas; Denton, Texas/ USA, Mount Holyoke College; South Hadley, Massachusetts/USA. - 2020.