Сфера игрока – префаб игрока на сцене

BasePlayer

Движение сферы игрока: есть анимация, описывающая вертикальное нелинейное движение игрока. Горизонтальное движение игрока в плоскости (XoZ) линейно.

Координаты сферы игрока на платформе – точка в которую прыгнул игрок (существует даже если игрок в полёте).

Как происходит ход: пользователь кликает по экрану на платформы или вникуда. Из экранной точки, в которую ткнул игрок получаем координаты точки в физическом мире, из этих координат и координаты сферы игрока на текущей платформе получаем координаты сферы игрока на следующей платформе, и формируем из них прыжок.

Прыжки хранятся в очереди, если последний добавленный прыжок ведёт в никуда – больше прыжки не добавляются, пока игрок не улетит в никуда и данный прыжок не проиграется.

В каждый момент времени выполняется какой-то прыжок, игрок перемещается со своей скоростью вдоль линейной траектории прыжка, так же выполняется вертикальное нелинейное движение, описанное в наследуемом классе в переменной VerticalMove. Комбинация этих двух движений даёт параболу по которой перемещается сфера игрока.

После окончания каждого прыжка, вызывается событие OnFinishedJump, в котором из очереди прыжков подтягивается следующий. Если такой отсутствует, сфера игрока прыгает на месте.

PlayerSpeed - текущая скорость игрока

CurrentPlatform - текущая платформа игрока

m\_QueuePlayerJumps - очередь прыжков текущего игрока

m\_CurrentJump - текущий прыжок игрока

PositionOnMoveBetweenPlatforms - текущая позиция игрока в трёхмерном пространстве (движение по параболе)

VerticalMove - трек вертикального движения игрока

**OnFinishedJump** – событие вызываемое после окончания прыжка. На него подписывается камера, чтобы пересчитать направление своего движения. и игрок, чтобы получить трек движения до следующей платформы

GoNextCurveJump() - получить следующий прыжок

UpdatePlayerOnMoveBetweenPlatforms() - обнновить положение сферы игрока между двумя точками [0;1] (интерполяция). Вызывается каждый кадр

CreateJumpFromPointAndEnqueueFromQueueJump() - Получить экранные координаты при клике игрока по игровому полю и получить из них прыжок.

UpdatePositionSpherePlayer() - обнновить позицию сферы игрока в трёхмерном пространстве. Вызывается каждый кадр

MultiPointLerp() - текущее положение вдоль трека точек [0;1] (интерполяция)

BaseThemeSettings : MonoBehaviour

**PlayerDefaultOffset** – Отступ префаба игрока от точки своего положения на платформе. Абстрактный член, требуется определение в классе потомке

DefaultPlayer : Player

**VerticalMove** - Вектор вертикального перемещения сферы игрока в локальных! Координатах. определение абстрактного члена из класса Player

DefaultThemeSettings : BaseThemeSettings

PlayerDefaultOffset - определение абстрактного члена из класса BaseThemeSettings

GlobalSetings

ThemePathFromAssets - путь до папки с текущей темой от папки Assets

hemePathFromResources - путь до папки с текущей темой от папки Resources

ForwardVector – вектор вдоль которого с отклонениями задаваемыми в (disperionAngleMaxX и disperionAngleMaxY) рендерятся новые платформы

disperionAngleMaxX - максимальный угол отклонения при вращении ForwardVector вдоль оси X

disperionAngleMaxY - максимальный угол отклонения при вращении ForwardVector вдоль оси У

MaxPlatformsOnScene - максимальное количество отрендеренных в данный момент платформ на сцене

CountUnvisiblePlatformsBeforeVisibles - Максимальное количество невидимых платформ в данный момент на сцене находящихся до видимых платформ

CountUnvisiblePlatformsAfterVisibles - Максимальное количество невидимых платформ в данный момент на сцене находящихся после видимых платформ

ConstMovingSpherePlayer - константа при расчёте скорости сферы игрока

StartSpeed - минимальная скорость сферы игрока. Она же его скорость в самом начале игры.

CameraHorisontalPositionAbovePlatform - отступ камеры над текущей платформой в горизонтальной проекции экрана пользователя

CameraVerticalPositionAbovePlatform - отступ камеры над текущей платформой в вертикальной проекции экрана пользователя

DefaultCameraRotation - изначальное вращение камеры. Не использовать вращение вокруг оси У (вокруг этой оси происходит вращение при наведении на следующие платформы)

ReLoadThemeSettings() - перезагружает настройки темы (использовать при смене темы)

LocalStorage

Класс хранящий настройки по умолчанию, юзер сможет изменять их через логику игры (переключатели и прочее).

**ThemeCurent** – текущая тема пользователя которую он выбрал либо дефолтная

Platform

Collider - компонент Collider UnityEngine, используемый при определении видимости объекта в камере (она одна) и при проверке на удаление платформ со сцены

IsVisible - видно ли объект в камере

PlayerJump

MedianeLengthOneJump - средняя длина прямой соединяющей точки старта прыжка и его конца (From, To)

VerticalMove - трек вертикального движения

From - начало прыжка

To - конец прыжка

PositionBetweenPlatforms - текущее положение между From и To. Прямая. Показывает насколько полностью был пройден путь по прямой от 0 до 1. Используется для интерполяции.

PositionOnMoveBetweenPlatforms - Текущая точка в которой находится сфера игрока. (Учитывая вертикальное движение и стандартный отступ сферы игрока указанный в настройках)

JumpToNowhere - является ли прыжок прыжком в никуда.

PlayerJump() - прыжок игрока.

FromHereToThere() - проложить прыжок из текущей конечной точки в новую точку

QueuePlayerJumps

m\_Queue - список си шарповый

IsEmpty - пуста ли очередь

LastJumpIsNowhere - является ли последний добавленный прыжок прыжком в никуда

GetLast - получить последний элемент очереди

EnqueueJump() - добавить прыжок в очередь

DequeueJump() - удалить прыжок из очереди

RenderCamera

Движение происходит так: есть константа (**CameraHorisontalPositionAbovePlatform** / **CameraVerticalPositionAbovePlatform**), задающая отступ камеры от интерполяции пути между платформами. Движение происходит с этим отступом по интерполяции пути от позиции сферы игрока на текущей платформе, до точки, в которую летит сфера игрока со скоростью движения игрока. Направление камеры – среднее между позициями следующих 3 платформ.

m\_CameraPlanes - плоскости, ограничивающие зону видимости камеры. Требуются для определения видимости из камеры платформ.

m\_Camera - компонент камера

PlaneIsVisible() - определить видна ли платформа

**PlatformsClickedZoneClick()** – вызывает событие **OnClickGameZone**, на которое подписан **MainPlayer** после чего совершает свой ход.

ProcessPositionCamera() - вычисление позиции и поворота камеры

RootPlatforms

Обновление платформ: находим первую видимую платформу, если за ней платформ больше, чем допускается в GameSetings.CountUnvisiblePlatformsBeforeVisibles - очищаем лишнее из начала. Если в конце платформ меньше, чем в GameSetings.CountUnvisiblePlatformsBeforeVisibles – добавляем платформы.

**m\_PrefabsPlatforms** – префабы платформ определённого стиля

**m\_Platforms** – платформы отрендеренные на сцене

**ResourcesPlatforms** – путь к папке platforms от папки resources

**AssetsPlatforms** – путь к папке platforms от папки assets

**FirstVisiblePlatform** – первая платформа которую видно на камере

**LastVisiblePlatform** – последняя платформа которую видно с камеры

**LastPointPosition** – координаты последней отрендеренной точки

**LoadPrefabsPlatforms()** – загружает префабы платформ из ресурсов из файлов

**UpdatePlatforms()** – проверяет какие платформы надо удалить (из начала если их много) а какие добавить (в конец если их мало)

**ReMathFirstLastVisiblePlatforms ()** – находим первую и последнюю видимые в камере платформы

**CreateAndPushLastPlatformOnScene()** – создаём объект из префаба и добавляем его на сцену

**DeleteFirstPlatformOnScene()** – удалить первую платформу на сцене

**GetRandomPlatform()** – получаем рандомную платформу из префабов загруженных в **LoadPrefabsPlatforms()**

**GetForvardVectorRandomedDispersion()** – берём вектор вдоль которого должны следовать платформы, немного смещаем его

RootPlayers

MainPlayer – основной игрок. Тот который играет от первого лица

LoadMainPlayer – загрузить основного игрока.

Прыжок

Игрок тыкает по экрану, в пространство пускается луч, если у этого луча нет коллизий ни с одной платформой – берётся платформа, на которую должен прыгнуть игрок и берётся высота её стартовой точки, в неё и полетит игрок. Если платформа, пролетая, пересекает луч, финальная тока перевычисляется и игрок летит в неё. Если в финальной точке отсутствует платформа, игрок летит дальше по траектории (последние 2 точки его траектории задают путь), так он пролетает заданное в настройка расстояние и погибает. Возрождение происходит посередине последней платформы, на которую наступил игрок (она замирает в текущей точке и не отмирает пока игрок не прыгнет далее)

Плавность движения игрока: вертикальная составляющая движения всегда плавная, а значит требуется регулировка только горизонтальной составляющей (XoZ). Во время движения в PlayerJump вычисляется текущая точка, по мимо процента пройденности пути (то что я называл интерполяцией), при каждом изменении финальной точки запоминается текущая точка на момент смены финальной (новая стартовая точка) и значение процента пройденности пути (тоже становится стартовым)

Вся логика отвечающая за прыжок кроется в PlayerJamp. Классу отдали луч – класс сам всё построил и всё проследил, передал сообщение что прыжок выполнился в точке такой то либо что прыжок уебался.

Во время доставания прыжка из очереди запускаем корутину переподсчёта позиции игрока, в ней же будет проверяться координата итоговой точки, луч будет чекаться на коллизии с префабами.

Надо переписать функцию получения платформ вокруг заданной, передавать будем только “радиус” необходимое количество платформ до и после текущей Эта функция будет использоваться для определения предполагаемой высоты.

Конструкторы PlayerJamp: прыгать на одной точке (передаётся только точка).

Нахождение EndPoint в случае изначального промаха игрока

Возьмём 3 платформы вокруг следующей платформы игрока (спереди и сзади по одной). Найдём и среднюю координату y – . Построим уравнение прямой: O – origin вектор луча, D - dirrection

. Подставляя в исходное уравнение, находим .

Вот такие дела.