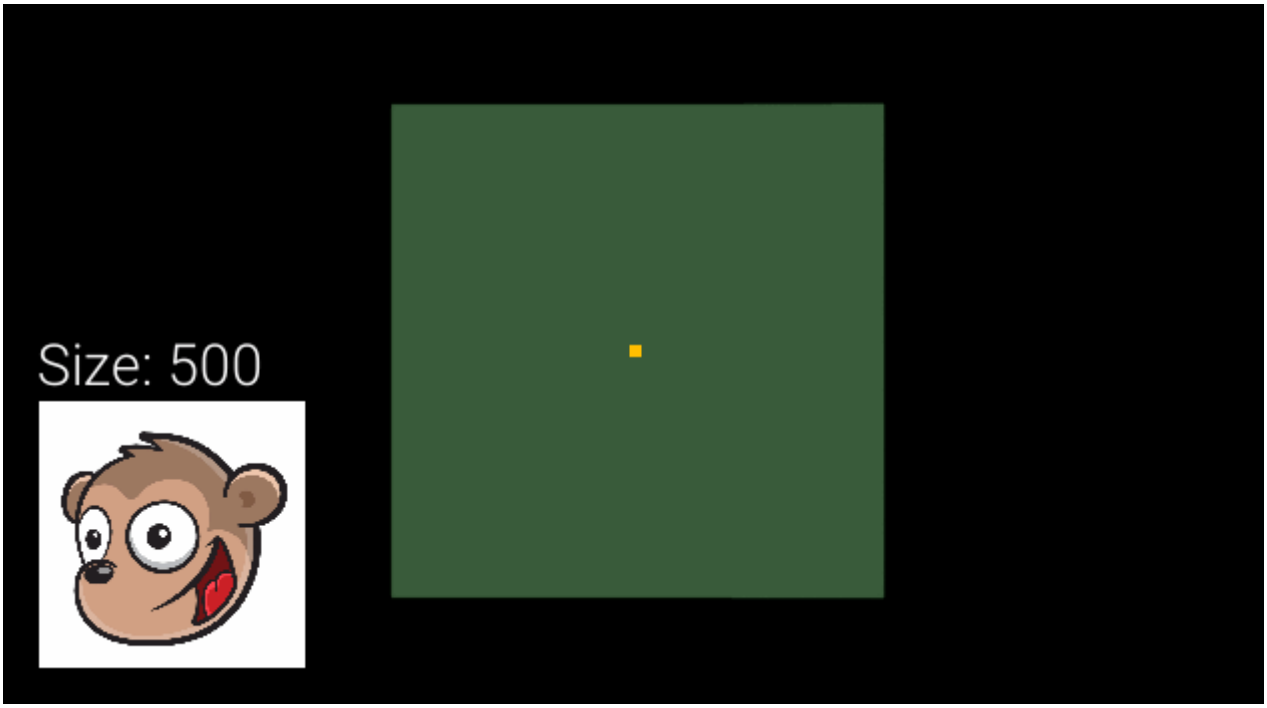
Рисование с помощью Render Targets в Unreal Engine



Render target — это, по сути, текстура, в которую можно выполнять запись во время выполнения приложения. С точки зрения движка они хранят такую информацию, как базовый цвет, нормали и ambient occlusion.

С точки зрения пользователя render target в основном используются в качестве своего рода дополнительной камеры. Можно задать захват сцены (scene capture) в какой-то точке и сохранять изображение в render target. Затем можно отобразить render target на меше, например, для симуляции камеры наблюдения.

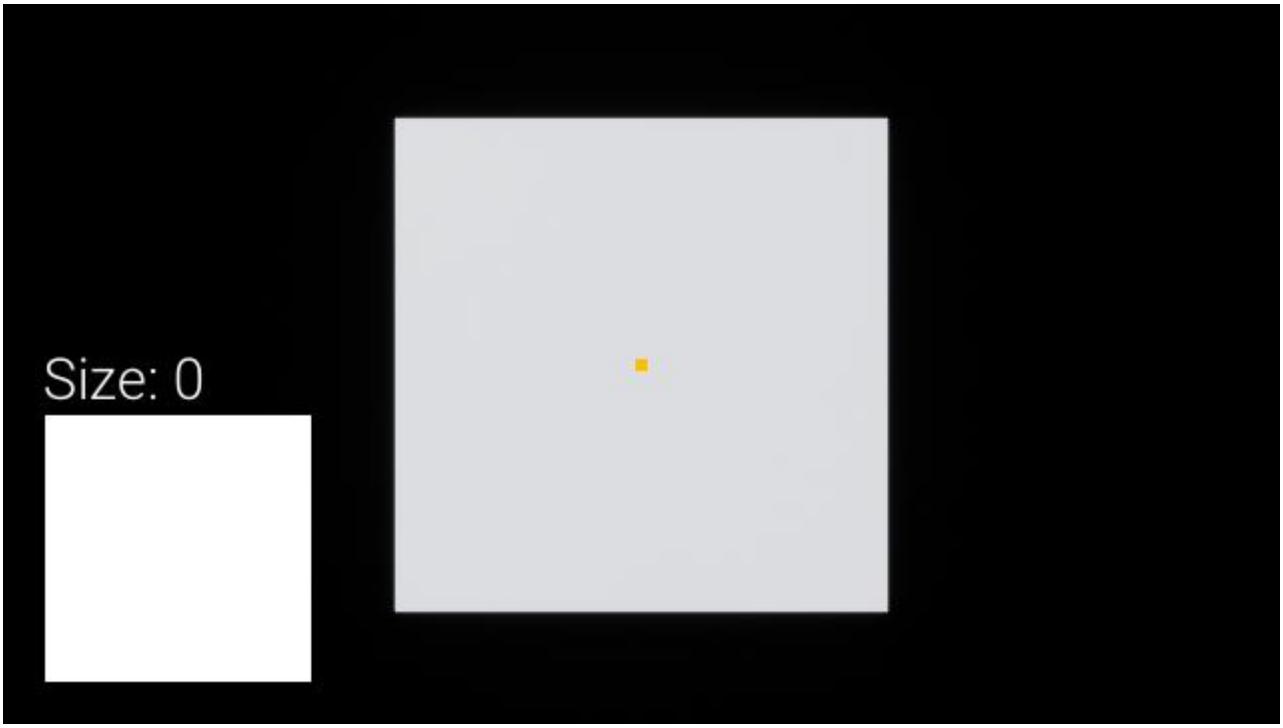
После выпуска версии движка 4.13 компания Epic добавила возможность отрисовки материалов непосредственно на render target с помощью блюпринтов. Эта функция позволяет создавать сложные эффекты, например, симуляцию жидкости и деформацию снега. Звучит потрясающе, правда? Но прежде чем переходить к таким сложным эффектам, лучше всего освоиться с чем-то простым. Что может быть проще, чем рисование на render target?

В этом туториале вы научитесь следующему:

- Динамически создавать render target с помощью блюпринтов
- Отображать render target на меше
- Рисовать текстуру на render target
- Менять размер кисти и текстуру во время игрового процесса

Приступаем к работе

Начнём с загрузки материалов для этого туториала (взять их можно https://koenig-media.raywenderlich.com/uploads/2018/05/CanvasPainter.zip). Распакуйте их, перейдите к *CanvasPainterStarter* и откройте *CanvasPainter.uproject*. Если вы нажмёте *Play*, то увидите следующее:



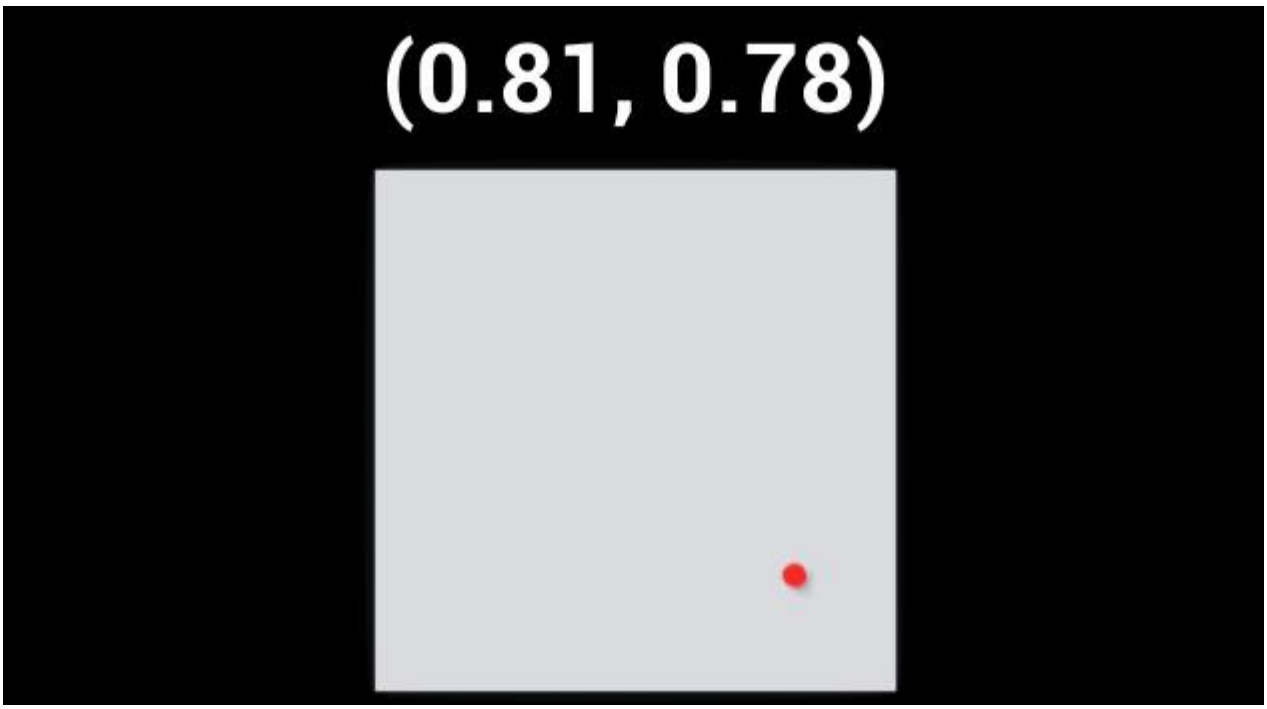
Квадрат посередине (canvas) — это то, на чём мы будем рисовать. Элементы UI слева будут текстурой, которой мы будем рисовать, и её размером.

Для начала давайте разберёмся со способом, который используется для рисования.

Способ рисования

Первое, что нам нужно — это render target, используемый в качестве холста (canvas). Для определения того, где выполнять рисование на render target, мы оттрассируем прямую, выходящую из камеры вперёд. Если прямая пересекает холст, то мы можем получить место пересечения в UV-пространстве.

Например, если холст имеет идеальную привязку UV-координат, то пересечение в центре вернёт значение *(0.5, 0.5)*. Если прямая пересекает холст в правом нижнем углу, то мы получим значение *(1, 1)*. Затем можно использовать простые вычисления для расчёта места рисования.



Но зачем получать координаты в UV-пространстве? Почему бы не использовать координаты настоящего пространства мира? При использовании пространства мира нам сначала придётся вычислять место пересечения относительно плоскости. Также придётся учитывать поворот и масштаб плоскости.

При использовании UV-пространства все эти вычисления не требуются. На плоскости с идеальной привязкой UV-координат пересечение с серединой всегда возвращает *(0.5, 0.5)*, вне зависимости от расположения и поворота плоскости.

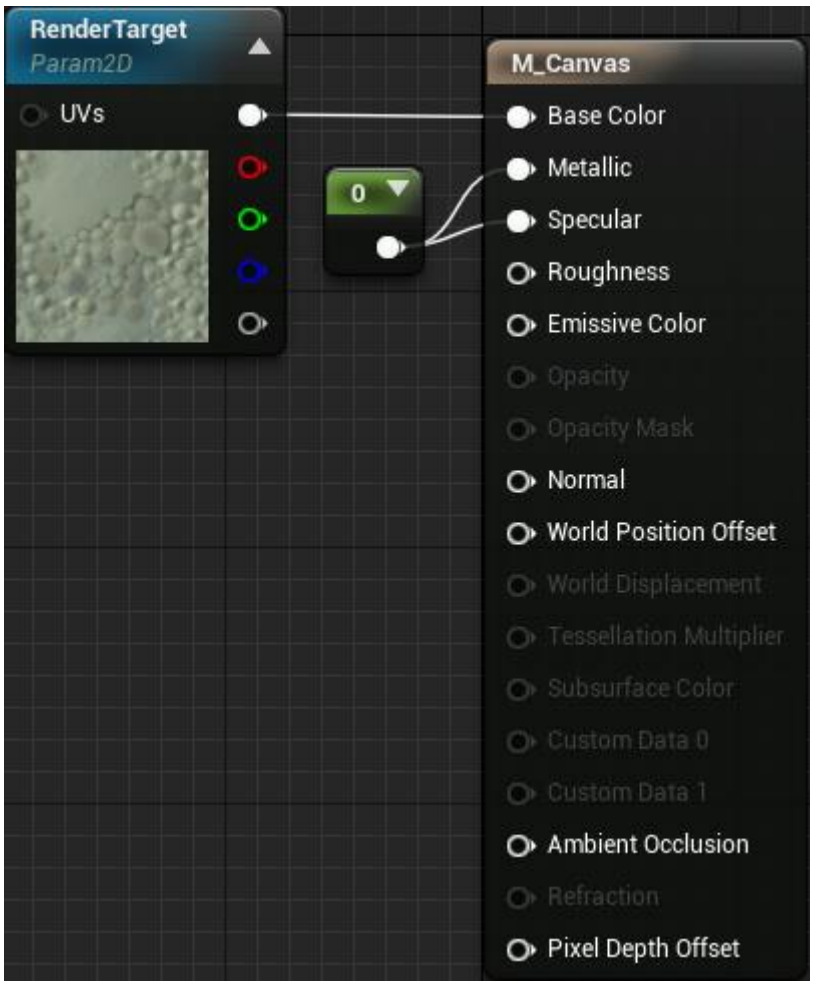
Примечание: рассматриваемый в этом туториале метод в общем случае работает только с плоскостями или похожими на плоскости поверхностями. Для других типов геометрии требуется более сложный метод, который я рассмотрю в другом туториале.

Сначала мы создадим материал, который будет отображать render target.

Создание материала холста

Перейдите в папку *Materials* и откройте *M_Canvas*.

В этом туториале мы будем создавать render target динамически, с помощью блюпринтов. Это значит, что нам придётся настроить текстуру как параметр, чтобы можно было передавать его render target. Для этого создадим *TextureSampleParameter2D* и назовём его *RenderTarget*. Затем соединим его с *BaseColor*.



Пока не беспокойтесь о выборе текстуры — мы займёмся этим дальше в блюпринтах. Нажмите *Apply*, а затем закройте *M_Canvas*.

Следующим этапом будет создание render target, после чего мы используем его в качестве материала холста.

Создание Render Target

Существует два способа создания render target. Первый: создание в редакторе нажатием на *Add New\Materials & Textures\Render Target*. Этот способ позволяет удобно ссылаться на один и тот же render target нескольким акторам. Однако если нам понадобятся несколько холстов, то придётся создавать render target вручную для каждого холста.

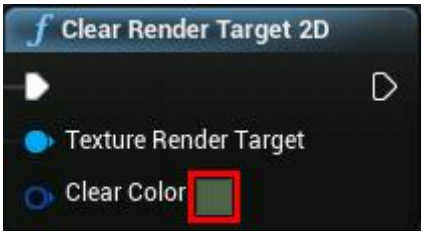
Поэтому лучше создавать render target с помощью блюпринтов. Преимущество такого подхода в том, что мы создаём render targets только при необходимости и они не раздувают объём файлов проекта.

Для начала нам нужно создать render target и сохранить его как переменную

для дальнейшего использования. Перейдите в папку *Blueprints* и откройте *BP_Canvas*. Найдите *Event BeginPlay* и добавьте выделенные узлы.

Присвойте параметрам *Width* и *Height* значение *1024*. Так мы изменим разрешение render target на *1024×1024*. Чем больше значения, тем выше качество изображения, но и больше затраты видеопамяти.

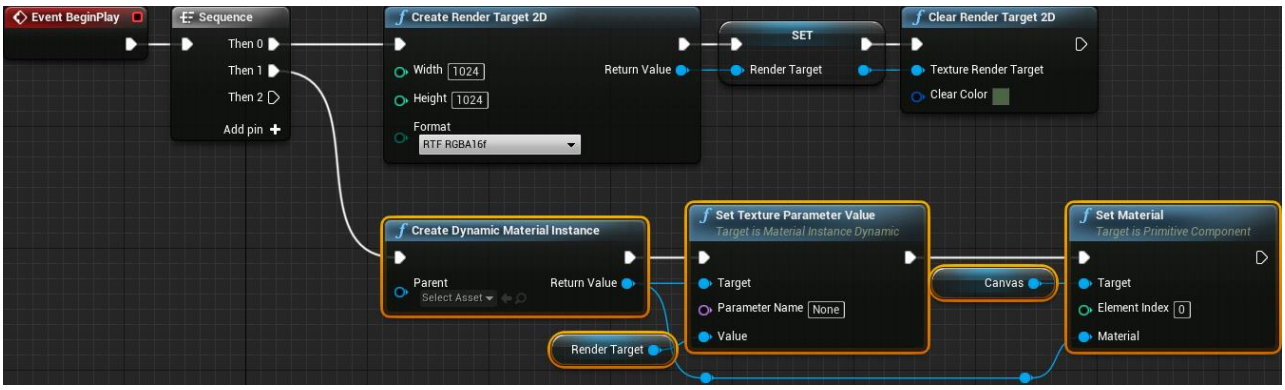
Далее идёт нод *Clear Render Target 2D*. Мы можем использовать этот нод для задания цвета render target. Задайте *Clear Color* значение *(0.07, 0.13, 0.06)*. При этом весь render target заполнится зеленоватым цветом.



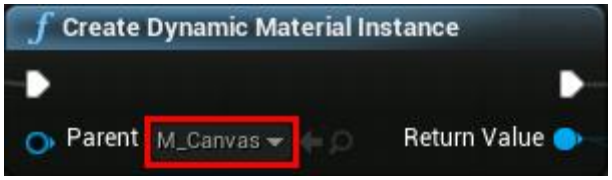
Теперь нам нужно отобразить render target на меше холста.

Отображение Render Target

На данном этапе меш холста использует материал по умолчанию. Для отображения render target нужно создать динамический экземпляр *M_Canvas* и передать ему render target. Затем нужно применить динамический экземпляр материала к мешу холста. Для этого мы добавим выделенные ноды:

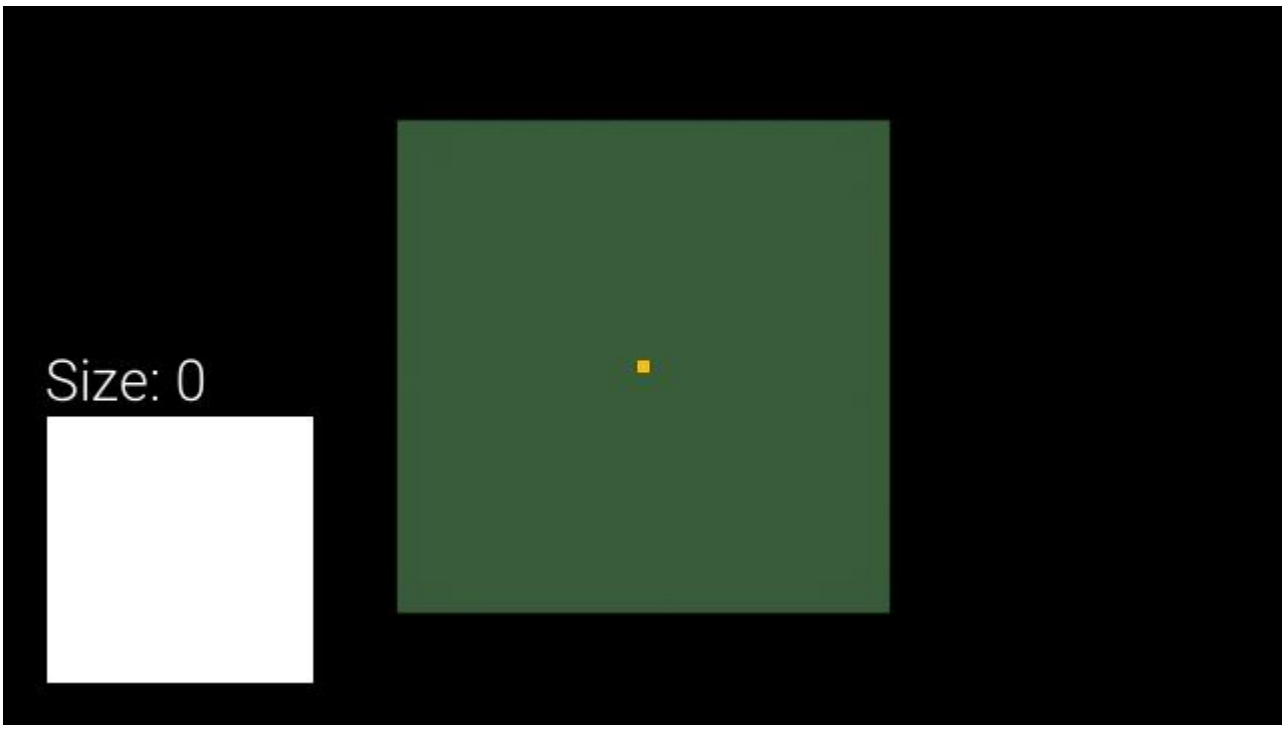


Сначала перейдём к ноду *Create Dynamic Material Instance* и зададим в качестве *Parent* значение *M_Canvas*. Так мы создадим динамический экземпляр *M_Canvas*.



Далее перейдём к ноду *Set Texture Parameter Value* и зададим для *Parameter Name* значение *RenderTarget*. Так мы передадим render target созданному ранее параметру текстуры.

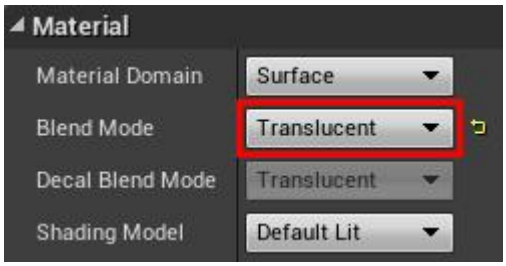
Теперь на меше холста будет отображаться render target. Нажмите на *Compile* и вернитесь в основной редактор. Нажмите на *Play*, чтобы увидеть, как холст изменит цвет.



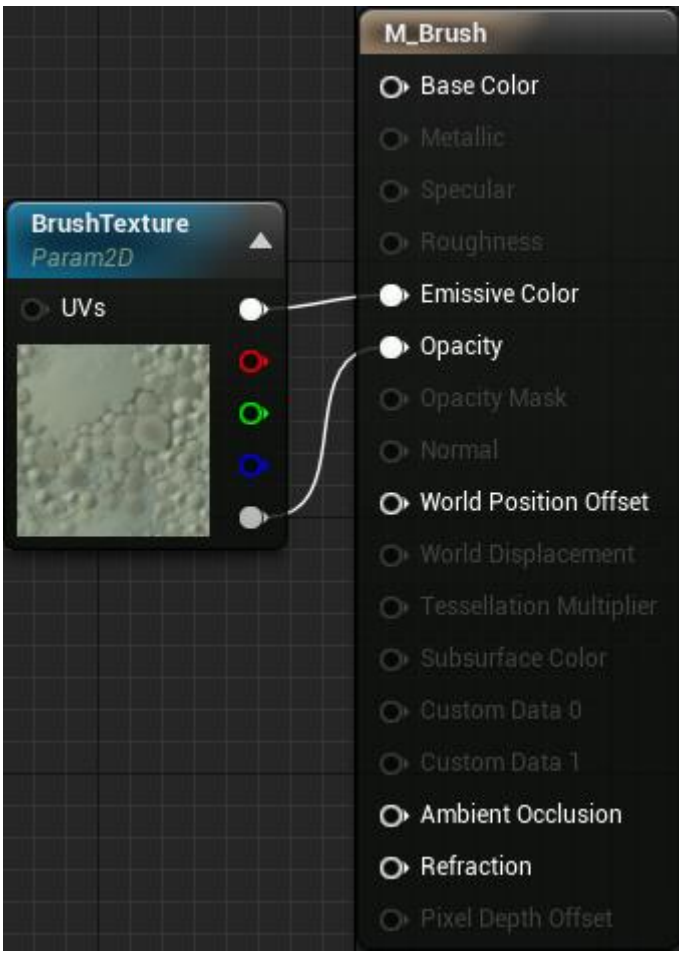
Теперь, когда у нас есть холст, нам нужно создать материал, который можно использовать в качестве кисти.

Создание материала кисти

Перейдите в папку *Materials*. Создайте материал *M_Brush* и откройте его. Сначала задайте для *Blend Mode* значение *Translucent*. Это позволит нам использовать текстуры с прозрачностью.

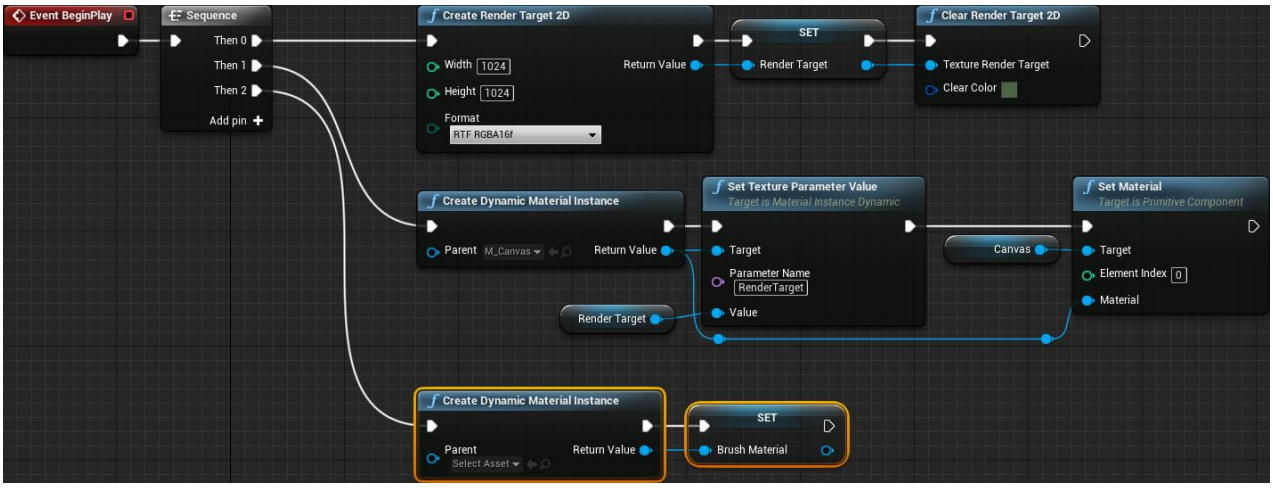


Как и в случае с материалом холста, мы задаём текстуру для кисти в блюпринтах. Создайте *TextureSampleParameter2D* и назовите его *BrushTexture*. Соедините его следующим образом:

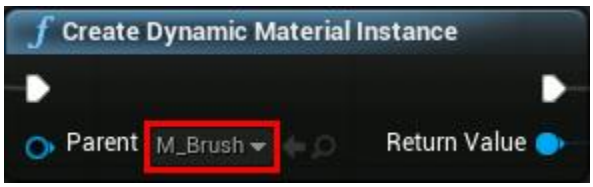


Нажмите на *Apply*, а затем закройте *M_Brush*.

Следующее, что нужно сделать — создать динамический экземпляр материала кисти, чтобы можно было менять текстуру кисти. Откройте *BP_Canvas* и добавьте выделенные узлы.



Далее перейдите в нод *Create Dynamic Material Instance* и задайте для *Parent* значение *M_Canvas*.

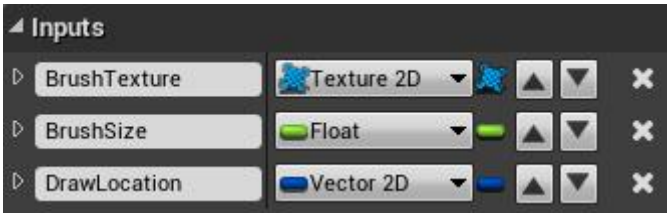


Мы создали материал кисти, и теперь нам нужна функция для рисования кистью на render target.

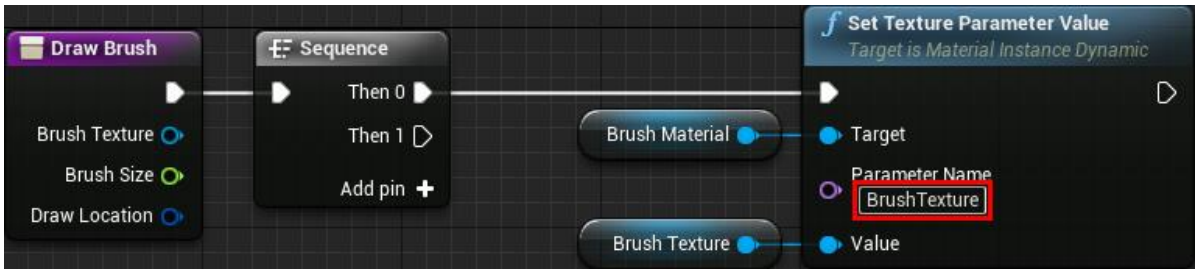
Рисование кистью на Render Target

Создайте новую функцию и назовите её *DrawBrush*. Сначала нам понадобятся параметры: используемая текстура, размер кисти и место для рисования. Создайте следующие входные входные данные:

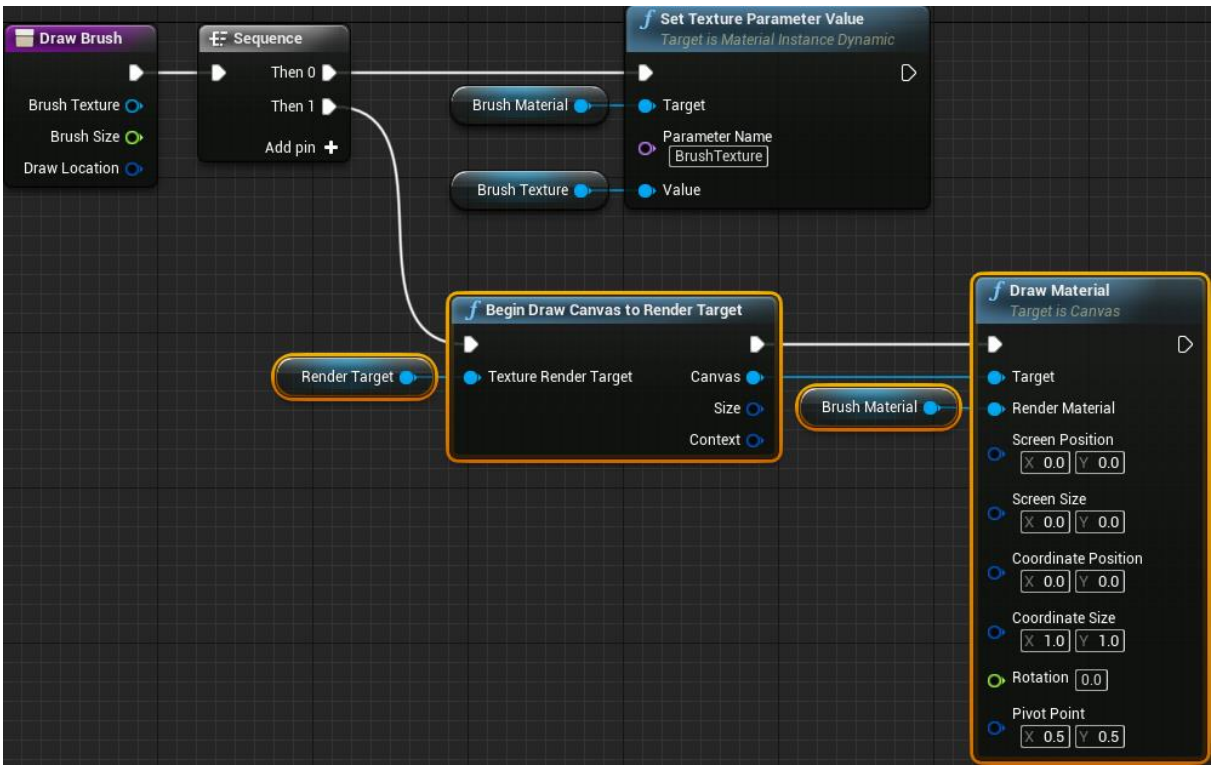
- *BrushTexture:* выберите тип *Texture 2D*
- *BrushSize:* выберите тип *float*
- *DrawLocation:* выберите тип *Vector 2D*



Прежде чем рисовать кисть, нам нужно задать её текстуру. Для этого создадим схему, показанную ниже. Убедитесь, что в качестве *Parameter Name* выбрано значение *BrushTexture*.



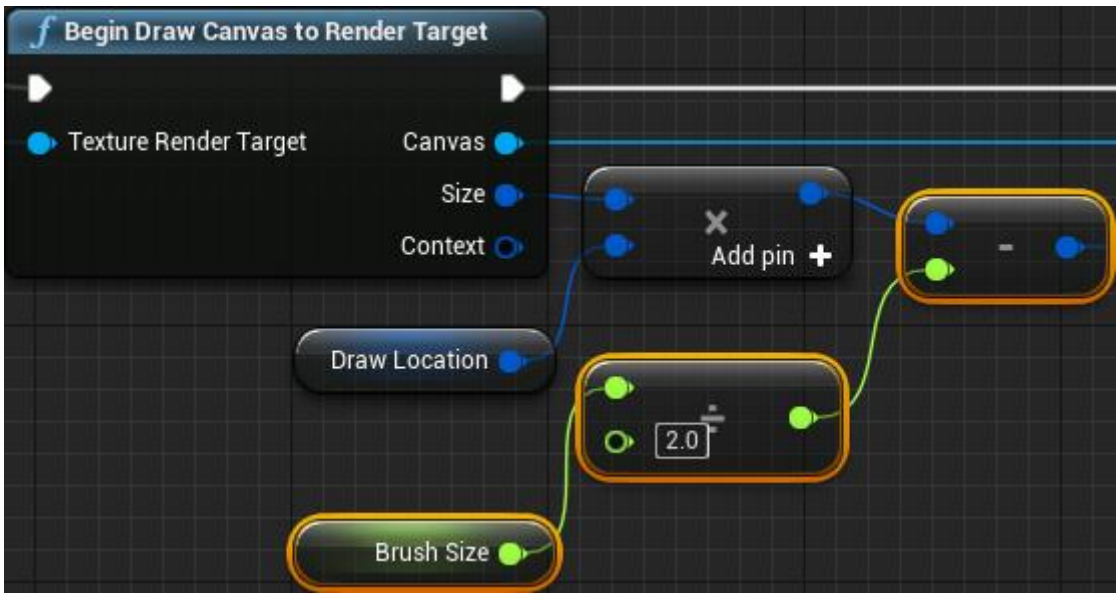
Теперь нам нужно выполнять отрисовку в render target. Для этого создадим выделенные ноды:



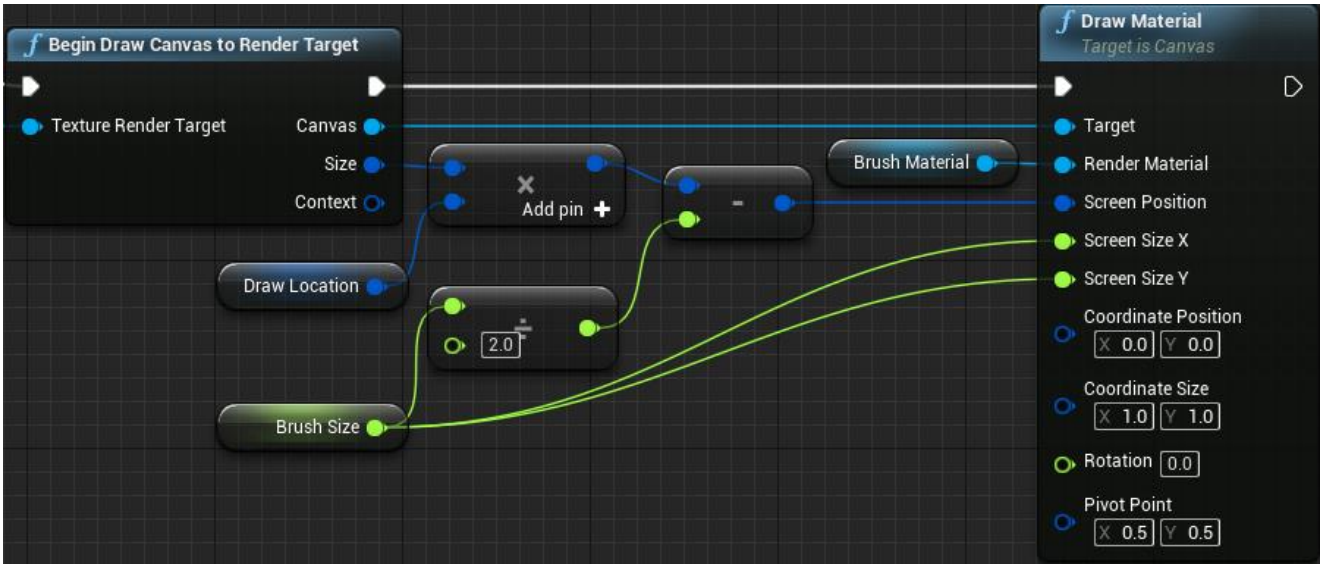
Begin Draw Canvas to Render Target позволит движку узнать, что мы хотим начать отрисовку в определённый render target. *Draw Material* позволит отрисовывать материал по указанному местоположению с выбранным размером и поворотом.

Вычисление позиции отрисовки — это двухэтапный процесс. Сначала нам нужно отмасштабировать *DrawLocation*, чтобы уместиться в разрешение render target. Для этого умножим *DrawLocation* на *Size*.

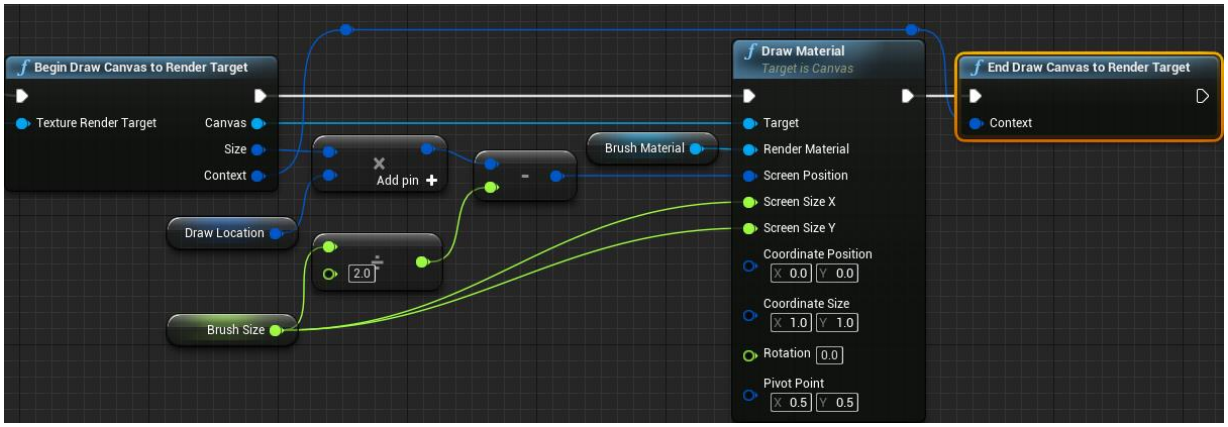
По умолчанию движок будет отрисовывать материалы, используя в качестве исходной точки верхний левый угол. Поэтому текстура кисти не будет центрирована нам, где мы хотим выполнить отрисовку. Чтобы исправить это, нам нужно поделить *BrushSize* на *2*, а затем вычесть результат из предыдущего этапа.



Затем соединим всё следующим образом:



Наконец нам нужно сообщить движку, что мы хотим остановить отрисовку в render target. Добавим нод *End Draw Canvas to Render Target* и соединим его следующим образом:

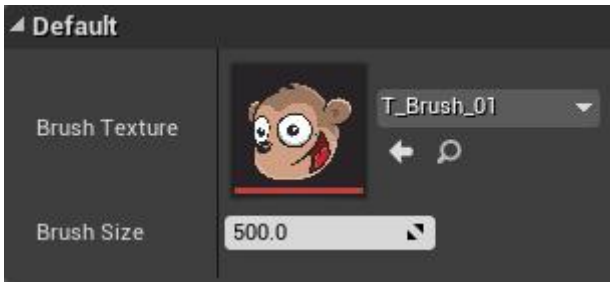


Теперь при каждом выполнении *DrawBrush* она будет сначала устанавливать в качестве текстуры для *BrushMaterial* передаваемую текстуру. Затем она будет отрисовывать *BrushMaterial* в *RenderTarget*, пользуясь переданными позицией и размером.

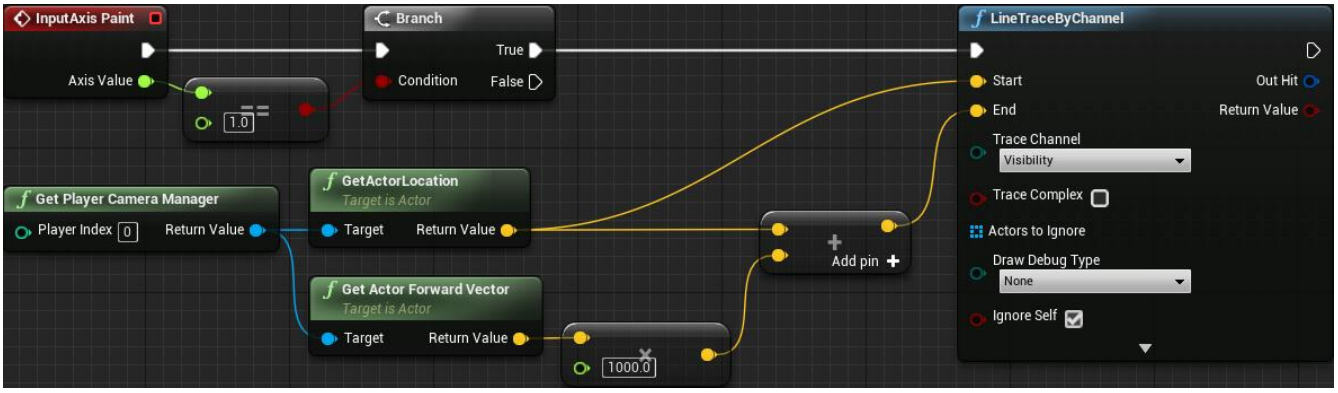
И на этом функция отрисовки готова. Нажмите на *Compile* и закройте *BP_Canvas*. Следующим шагом будет трассировка прямой из камеры и рисование в том месте холста, где произошло пересечение.

Трассировка прямой из камеры

Прежде чем рисовать на холсте, нам нужно указать текстуру кисти и размер. Перейдём в папку *Blueprints* и откроем *BP_Player*. Затем присвоим переменной *BrushTexture* значение *T_Brush_01*, а переменной *BrushSize* значение *500*. Так мы назначим кисти изображение обезьяны размером *500×500* пикселей.

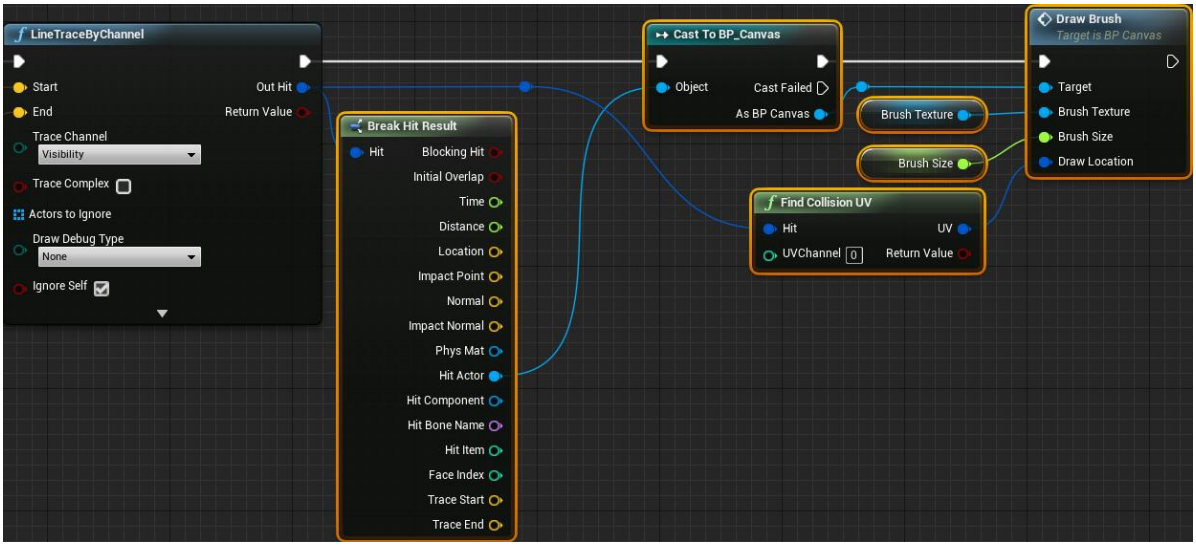


Далее необходимо выполнить трассировку прямой. Найдите *InputAxis Paint* и создайте следующую схему:



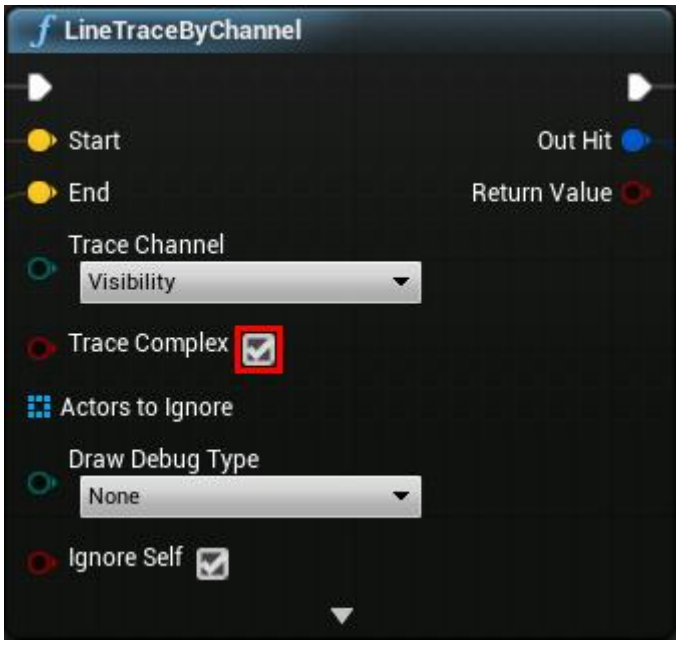
Так мы будем выполнять трассировку прямой, направленной из камеры прямо, пока игрок будет удерживать клавишу, назначенную для *Paint* (в нашем случае это *левая клавиша мыши*).

Теперь нам необходимо проверить, пересекла ли прямая холст. Добавим выделенные ноды:



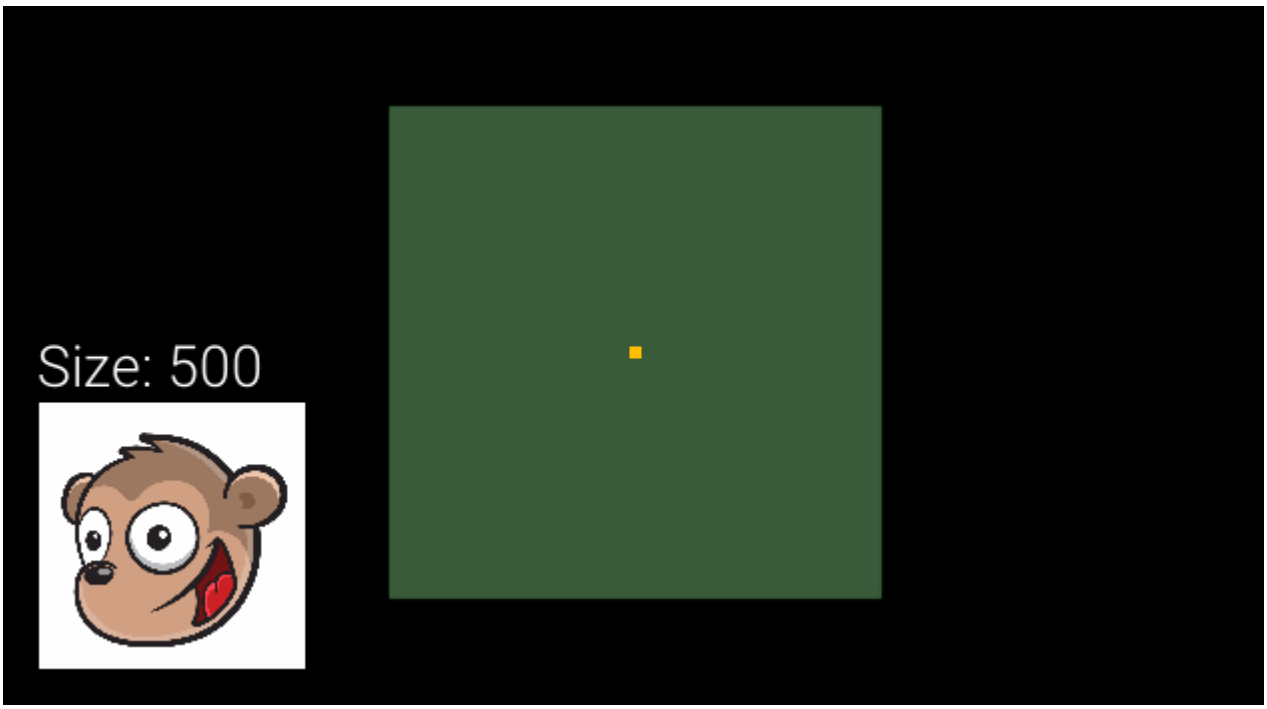
Теперь при пересечении прямой и холста будет выполняться функция *DrawBrush*, использующая переданные ей переменные кисти и UV-координаты.

Чтобы нод *Find Collision UV* заработал, нам нужно изменить два параметра. Во-первых, перейдём в нод *LineTraceByChannel* и включим *Trace Complex*.



Во-вторых, перейдём в *Edit\Project Settings*, а затем в *Engine\Physics*. Включим *Support UV From Hit Results* и перезапустим проект.

После перезапуска для рисования на холсте нажмите *Play* и *левую клавишу мыши*.

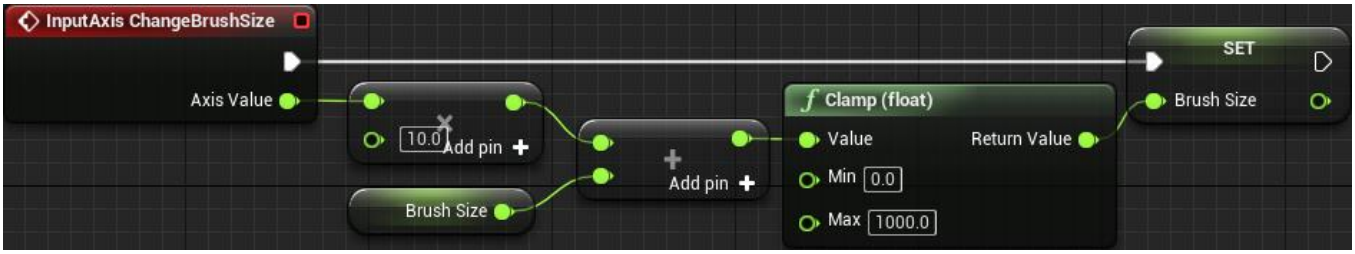


Можно даже создать несколько холстов и рисовать на каждом из них отдельно. Это возможно, потому что каждый холст динамически создаёт собственный render target.

В следующем разделе мы реализуем функционал изменения игроком размера кисти.

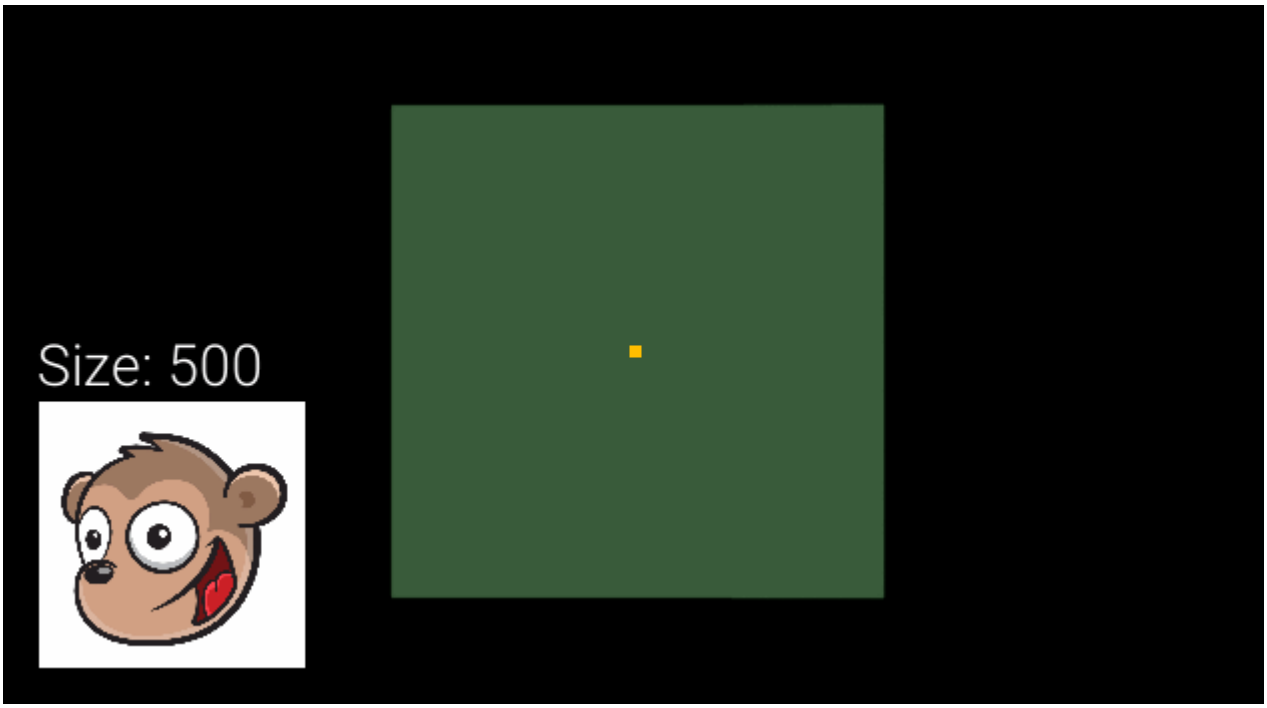
Изменение размера кисти

Откройте *BP_Player* и найдите нод *InputAxis ChangeBrushSize*. Эта привязка оси настроена на использование *колеса мыши*. Для изменения размера кисти нам достаточно менять значение *BrushSize* в зависимости от *Axis Value*. Для этого создадим следующую схему:



Она будет выполнять прибавление или вычитание из *BrushSize* при использовании игроком колеса мыши. Первое умножение определяет скорость прибавления или вычитания. В качестве меры безопасности добавлено *Clamp (float)*. Он гарантирует, что размер кисти не станет меньше *0* или больше *1000*.

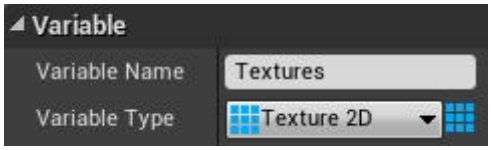
Нажмите на *Compile* и вернитесь в основной редактор. Покрутите *колесо мыши*, чтобы изменить размер кисти при рисовании.



В последнем разделе мы создадим функционал, позволяющий игроку менять текстуру кисти.

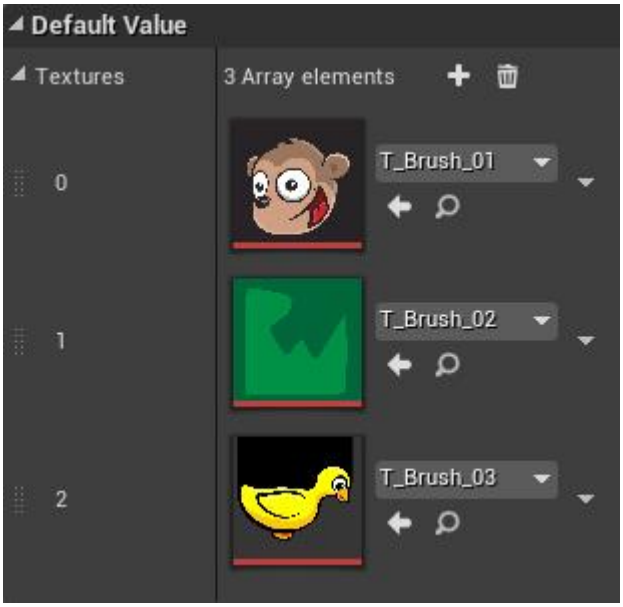
Изменение текстуры кисти

Для начала нам нужен массив для хранения текстур, которые может использовать игрок. Откройте *BP_Player* и создайте переменную *array*. Выберите для неё тип *Texture 2D* и назовите её *Textures*.



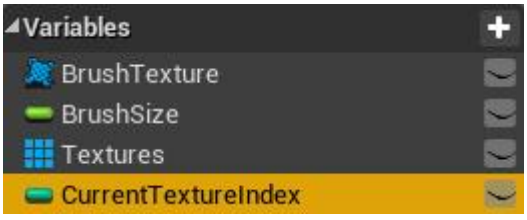
Затем создайте *три* элемента в *Textures*. Присвойте им следующие значения:

- *T_Brush_01*
- *T_Brush_02*
- *T_Brush_03*



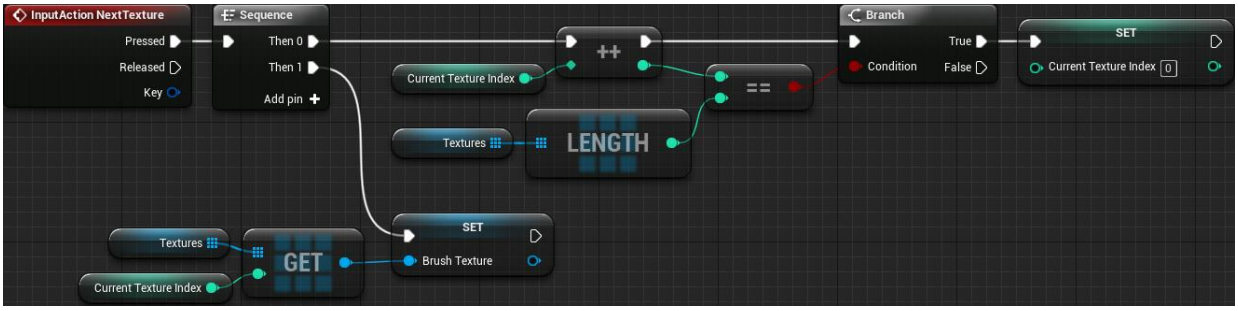
Это будут текстуры, которыми сможет рисовать игрок. Для добавления новых текстур достаточно добавить их в этот массив.

Далее нам необходима переменная для хранения текущего индекса массива. Создайте переменную *integer* и назовите её *CurrentTextureIndex*.



Далее нам необходим способ обхода в цикле всех текстур. Для этого туториала я настроил привязку действия (action mapping) под

названием *NextTexture* и привязал её к *правой клавише мыши*. Когда игрок нажимает эту клавишу, должен выполняться переход к следующей текстуре. Для этого найдите нод *InputAction NextTexture* и создайте следующую схему:



Эта схема будет увеличивать *CurrentTextureIndex* при каждом нажатии на *правую клавишу мыши*. Если индекс достигает конца массива, то он снова сбрасывается на *0*. Наконец, *BrushTexture* задаётся соответствующая текстура.

Нажмите на *Compile* и закройте *BP_Player*. Нажмите *Play* и щёлкайте *правой клавишей мыши* для переключения между текстурами.