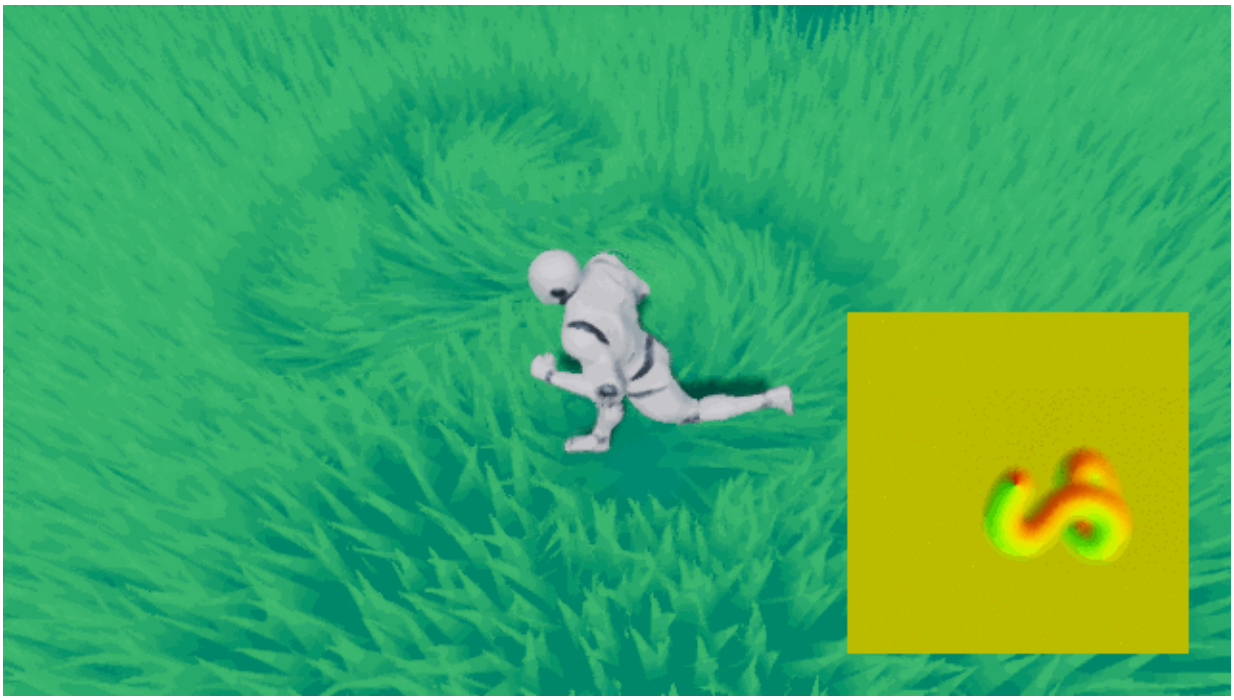
Создание интерактивной травы в Unreal Engine



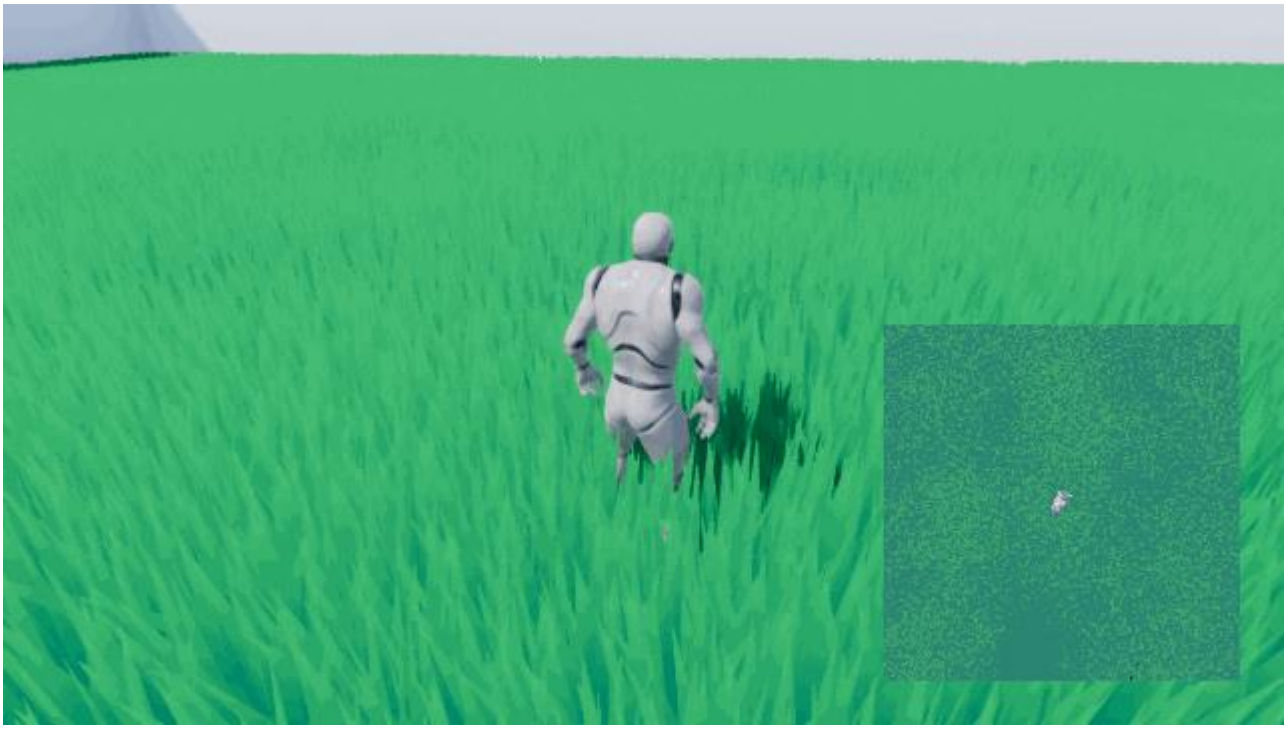
До недавнего времени трава в играх обычно обозначалась текстурой на земле, а не рендерингом отдельных стеблей. Но с увеличением мощности железа появилась возможность рендерить траву. Замечательные примеры такого рендеринга можно увидеть в играх наподобие *Horizon Zero Dawn* и *The Legend of Zelda: Breath of the Wild*. В этих играх игрок может бродить по травяным лугам, и, что более важно, трава *реагирует* на действия игрока.

К счастью, создать такую систему не очень сложно. На самом деле, статья научит вас именно этому! В этом туториале вы научитесь следующему:

- Создавать векторное поле с помощью захвата сцены (scene capture) и системы частиц
- Сгибать траву от игрока на основании векторного поля

Приступаем к работе

Начнём с загрузки материалов для этого туториала (их можно скачать https://koenig-media.raywenderlich.com/uploads/2018/07/InteractiveGrass.zip). Распакуйте их, перейдите в *InteractiveGrassStarter* и откройте *InteractiveGrass.uproject*. Вы увидите небольшое поле травы, которое будет темой этого туториала. Также я создал виджет для отображения render target захвата сцены.



Прежде чем приступить, вам стоит изучить туториал по созданию следов на снегу, потому что я буду пропускать часть информации. Стоит учесть, что в этом туториале тоже используется захват и проецирование. Для экономии времени я подготовил блюпринта захвата, похожий на блюпринт из туториала по следам на снегу.

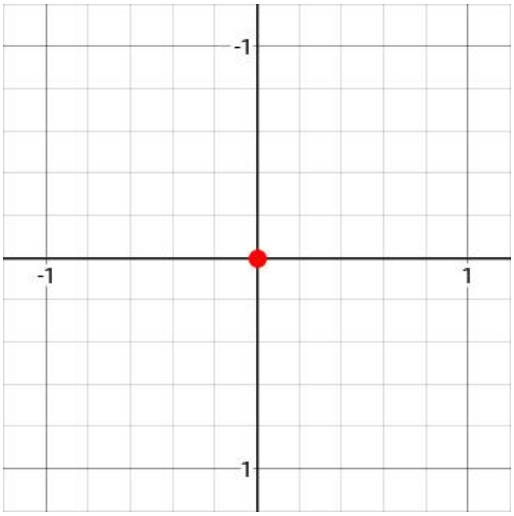
Для начала давайте рассмотрим другой способ создания интерактивной травы. Наиболее распространённый способ заключается в передаче координат игрока материалу травы и в использовании сферической маски для отгибания травы в определённом радиусе от игрока.

Хотя этот подход вполне хорош, он плохо масштабируется, когда мы хотим добавить больше влияющих на траву акторов. Для каждого добавляемого актора к материалу придётся добавлять ещё один параметр координат и сферическую маску. Более масштабируемый метод заключается в использовании *векторного поля*.

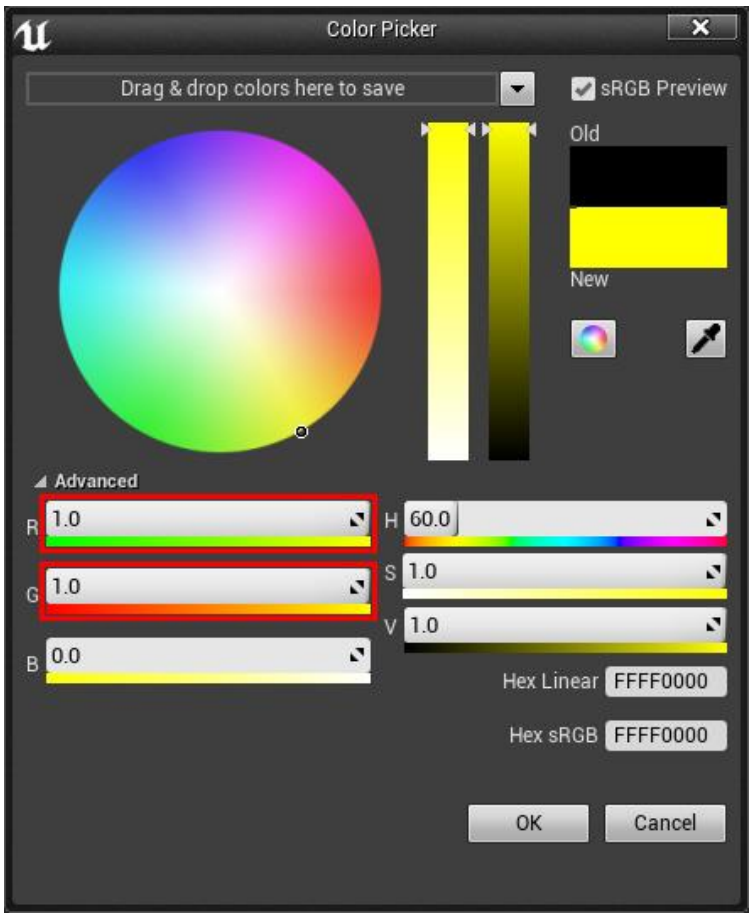
Что такое векторное поле?

Векторное поле — это просто текстура, каждый пиксель которой соответствует направлению. Если вы раньше работали с картами потоков, то они аналогичны. Но вместо перемещения UV мы будем с помощью контакта World Position Offset перемещать вершины. В отличие от решения со сферической маской, для получения направления сгибания материалам достаточно только *один раз* сэмплировать векторное поле.

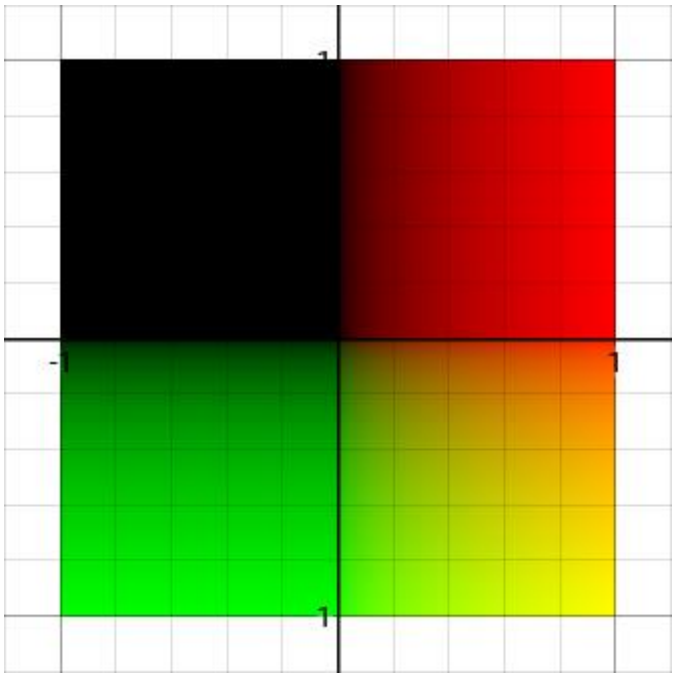
Давайте узнаем, как можно сохранять направления в текстуру. Посмотрите на эту сетку:



Допустим, красная точка — это объект, который мы хотим переместить. Если мы переместим его в правый нижний угол, то какой вектор будет обозначать это перемещение? Если вы ответили *(1, 1)*, то вы правы! Как вы вероятно знаете, можно также представить векторы в виде цветов, и таким образом сохранить их в текстуру. Давайте вставим этот вектор в color picker Unreal и посмотрим, какой цвет он вернёт.



Как видите, направление *(1, 1)* возвращает жёлтый цвет. Это значит, что если мы захотим согнуть траву по направлению положительных осей XY, то нам придётся использовать этот цвет текстуры. Давайте теперь посмотрим на цвета всех векторов.

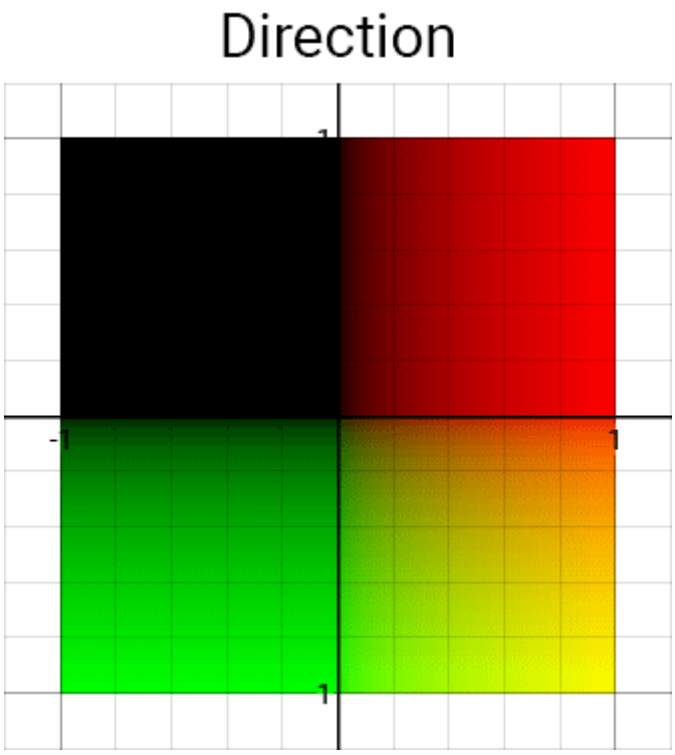


Правый нижний квадрат выглядит довольно хорошо, потому что имеет градиенты по обеим осям. Это значит, что мы можем хранить в этом квадранте в виде цвета любой вектор, потому что каждый вектор имеет уникальный цвет.

Но с другими тремя квадрантами возникает проблема. У нас есть градиент только по одной оси, или нет градиента вообще. Это значит, что несколько векторов будут иметь один цвет. Например, мы никак не сможем различить векторы *(-1, 1)* и *(0, 1)*.

Эти три квадранта не имеют уникальных цветов для каждого вектора потому, что мы можем представить цвета только с помощью значений от 0 до 1. однако в этих трёх квадрантах используются отрицательные значения, находящиеся вне этого интервала.

Решение заключается в перераспределении векторов таким образом, чтобы все они поместились в интервале от 0 до 1. Это можно сделать, умножив вектор на *0.5* и прибавив *0.5*. Вот визуализация того, что мы получим:



Теперь у каждого вектора есть уникальный цвет. Когда нам нужно использовать его для вычислений, мы просто перераспределяем его обратно в интервал от -1 до 1. Вот несколько цветов, и соответствующие им направления после перераспределения:

- *(0, 0):* отрицательные X и Y
- *(0.5, 0.5):* нет движения
- *(0, 1):* отрицательный X и положительный Y
- *(1, 0):* положительный X и отрицательный Y

Теперь давайте узнаем, как создать векторное поле в Unreal.

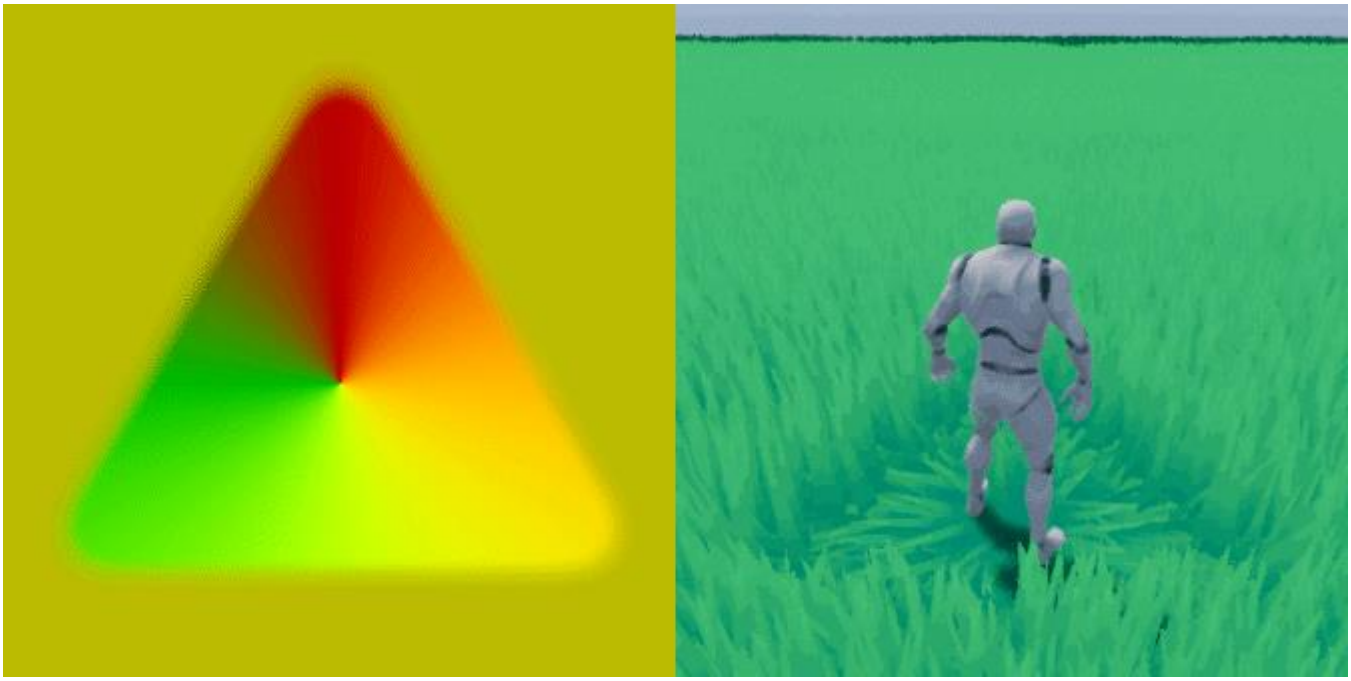
Создание векторного поля

В отличие от создания следов на снегу, мы не будем выполнять захват формы объектов. Вместо этого мы будем рисовать на render target с помощью «кистей». Это будут просто изображения выбранного векторного поля. Я буду называть их *кистями направлений*.

Вместо рисования на render target с помощью блюпринтов, мы можем использовать *частицы*. Частицы будут отображать кисть направлений и испускаться из игрока. Для создания векторного поля нам достаточно использовать захват сцены (scene capture) и выполнять захват только частиц. Преимущество этого метода в том, что создавать следы очень просто. Кроме

того, он позволяет с лёгкостью управлять такими свойствами, как длительность сохранения следов и их размер. Кроме того, частицы создают временно сохраняющиеся следы, потому что они существуют после ухода из области захвата и возврата в неё.

Ниже представлено несколько примеров кистей направлений, которые мы можем использовать, а также их влияние на траву. Заметьте, что в показанном ниже примере частицы невидимы.

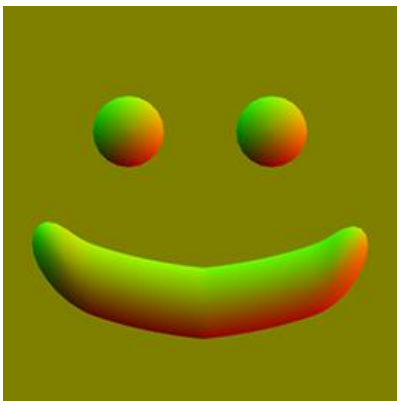


Для начала давайте создадим материал, который будет отображать кисть направлений.

Создание материала для направлений

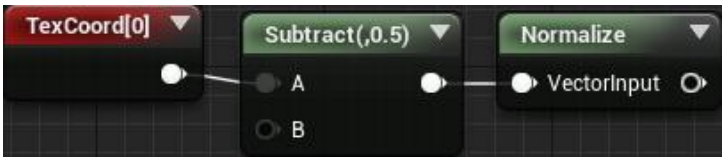
Существует два способа создания кисти направлений:

1. *Математический:* направления и форма задаются внутри материала. Преимущество его в том, что он не требует стороннего ПО и удобен для простых форм.
2. *Преобразование карты нормалей:* создание карты нормалей нужных направлений и формы. Для преобразования карты в подходящее векторное поле нам достаточно удалить синий канал. Преимущество этого метода в том, что можно очень просто создавать сложные формы. Ниже представлен пример кисти, которую сложно будет создать математически.



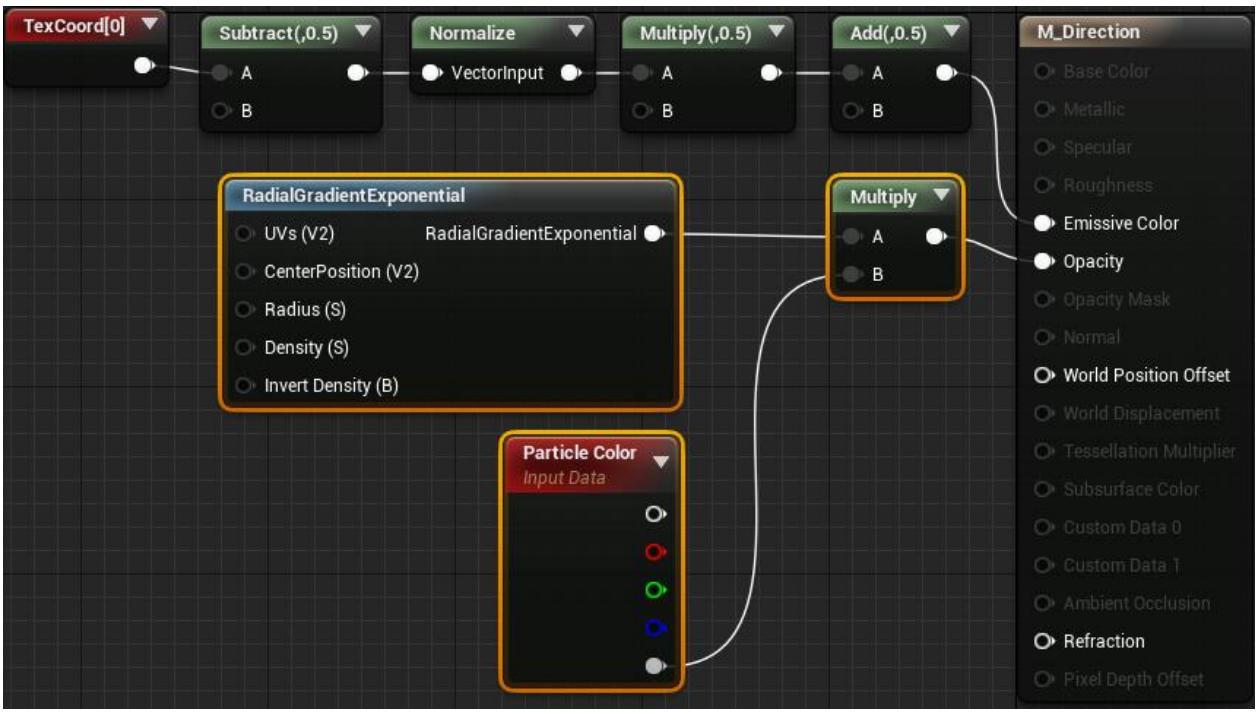
Для этого туториала мы создадим кисть математически. Перейдите в папку *Materials* и откройте *M_Direction*. Заметьте, что для этого материала выбрана модель затенения (shading model) *Unlit*. Это важно, потому что позволяет захвату сцены захватывать частицы без влияющего на них освещения.

Чтобы не усложнять, мы создадим материал, который заставит траву отодвигаться от центра частицы. Для этого создадим следующую схему:



Теперь нам нужно выполнить перераспределение. Для этого добавьте выделенные ноды и соедините всё следующим образом:

Теперь давайте придадим кисти круглую форму. Для этого добавим выделенные ноды:



RadialGradientExponential управляет размером и резкостью окружности круга. Умножение его на *Particle Color* позволяет управлять непрозрачностью частиц из системы частиц. Подробнее я расскажу об этом в следующем разделе.

Вот, как выглядит кисть:

Нажмите *Apply* и закройте материал. Теперь, когда мы создали материал, настало время приступить к системе частиц следов.

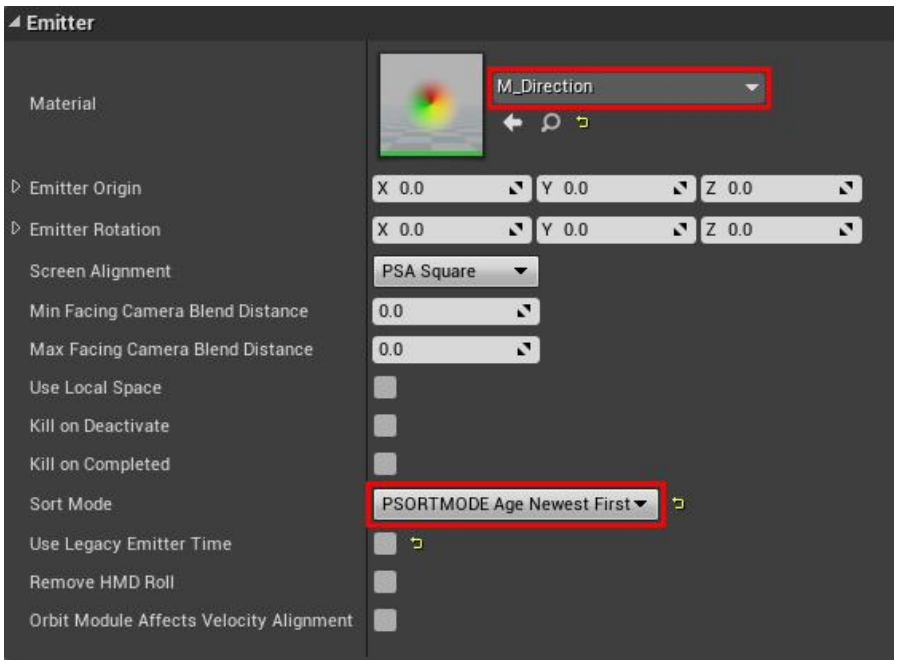
Создание системы частиц следов

Перейдите в папку *ParticleSystems* и откройте *PS_GrassTrail*. Для экономии времени я уже создал все необходимые модули.

Вот, как каждый модуль влияет на следы на траве:

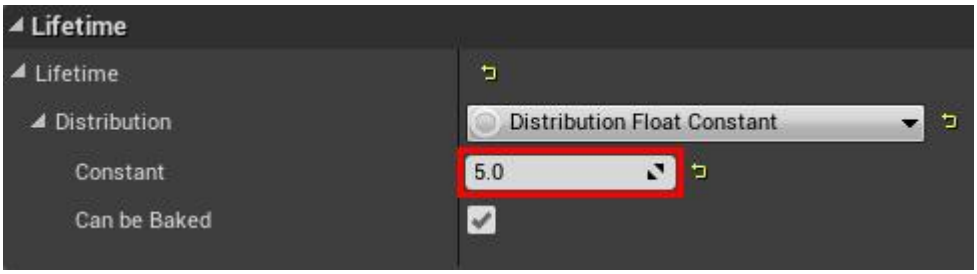
- *Spawn:* частота создания влияет на плавность следов. Если следы выглядят прерывистыми, то стоит увеличить частоту создания. В этом туториале мы оставим значение по умолчанию (20).
- *Lifetime:* время существования следа до возвращения травы к исходному состоянию
- *Initial Size:* размер следа
- *Color Over Life:* поскольку мы используем в материале Particle Color, здесь можно управлять непрозрачностью. Также можно изменять альфа-кривую для управления пропаданием следа. Например, можно выбрать линейное пропадание, easing in и/или easing out. В этом туториале мы оставим настройку по умолчанию, то есть линейное пропадание.
- *Lock Axis:* используется для того, чтобы частицы были направлены в сторону захвата сцены
- *Initial Rotation:* используется для того, чтобы частицы были ориентированы по правильным осям (подробнее об этом ниже)

Сначала нам нужно задать материал. Выберите модуль *Required* и задайте для *Material* значение *M_Direction*. Также задайте для *Sort Mode* значение *PSORTMODE Age Newest First*.

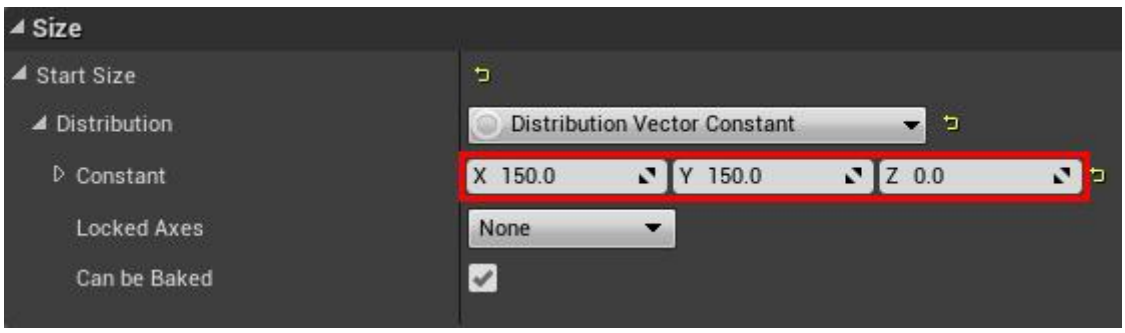


Этот режим сортировки позволяет рендерить новые частицы поверх старых. Если этого не сделать, то на траву будут влиять не новые, а старые частицы.

Далее идёт длительность существования следа. Выберите модуль *Lifetime* и задайте *Constant* значение 5. Благодаря этому след будет пропадать в течение пяти секунд.



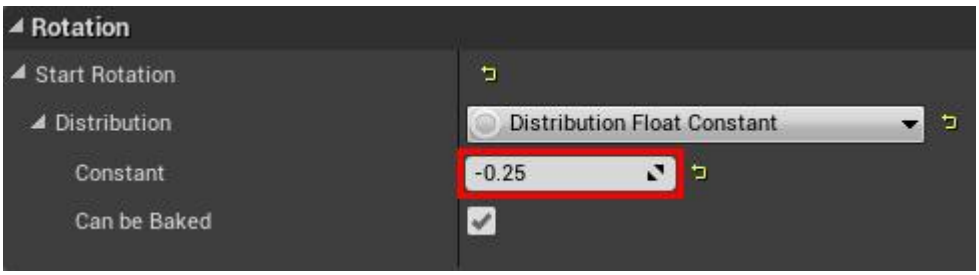
Теперь перейдём к размеру следа. Выберите модуль *Initial Size* и задайте для *Constant* значение *(150, 150, 0)*. Благодаря этому каждая частица будет покрывать область 150×150.



Теперь нам нужно сделать так, чтобы смотрели в направлении захвата сцены. Поскольку захват сцены выполнятся из вида сверху, то частицы должны

смотреть в направлении положительной оси Z. Для этого выберите модуль *Lock Axis* и задайте для *Lock Axis Flags* значение *Z*.

Наконец, нам нужно задать поворот частиц. На текущий момент цвета кисти не выровнены с направлением, которое они представляют. Так получилось потому, что по умолчанию система частиц применяется с поворотом в 90 градусов. Чтобы исправить это, выберите модуль *Initial Rotation* и задайте для *Constant* значение *-0.25*. Это повернёт частицы на 90 градусов против часовой стрелки.

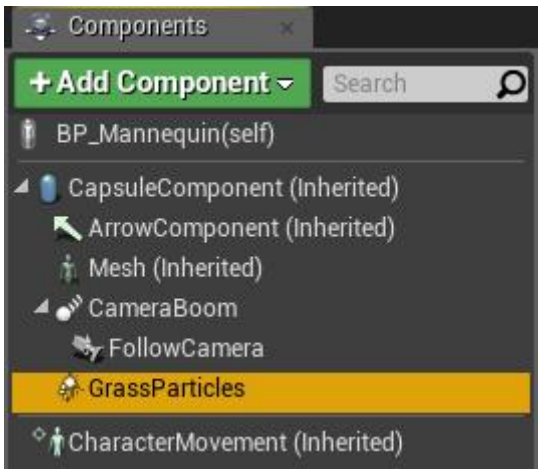


И это всё, что нам нужно для системы частиц, поэтому давайте закроем её.

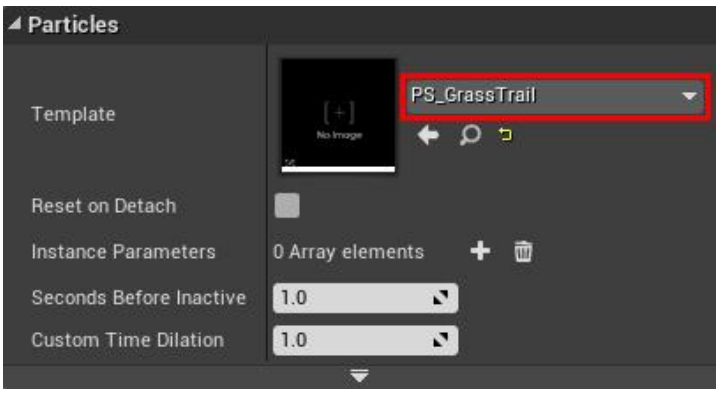
Далее нам нужно прикрепить систему частиц к тому, что должно создавать следы. В нашем случае нужно прикрепить её к персонажу игрока.

Прикрепление системы частиц

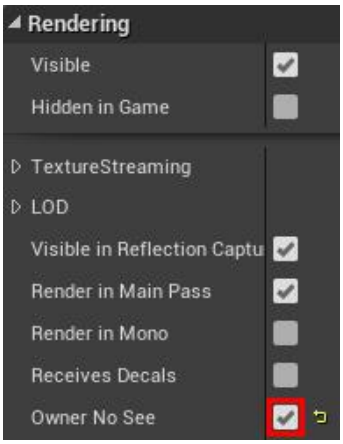
Перейдите в *Characters\Mannequin* и откройте *BP_Mannequin*. Далее создайте компонент *Partice System* и назовите его *GrassParticles*.



Далее нам нужно задать систему частиц. Перейдите в панель Details и задайте для *Particles\Template* значение *PS_GrassTrail*.

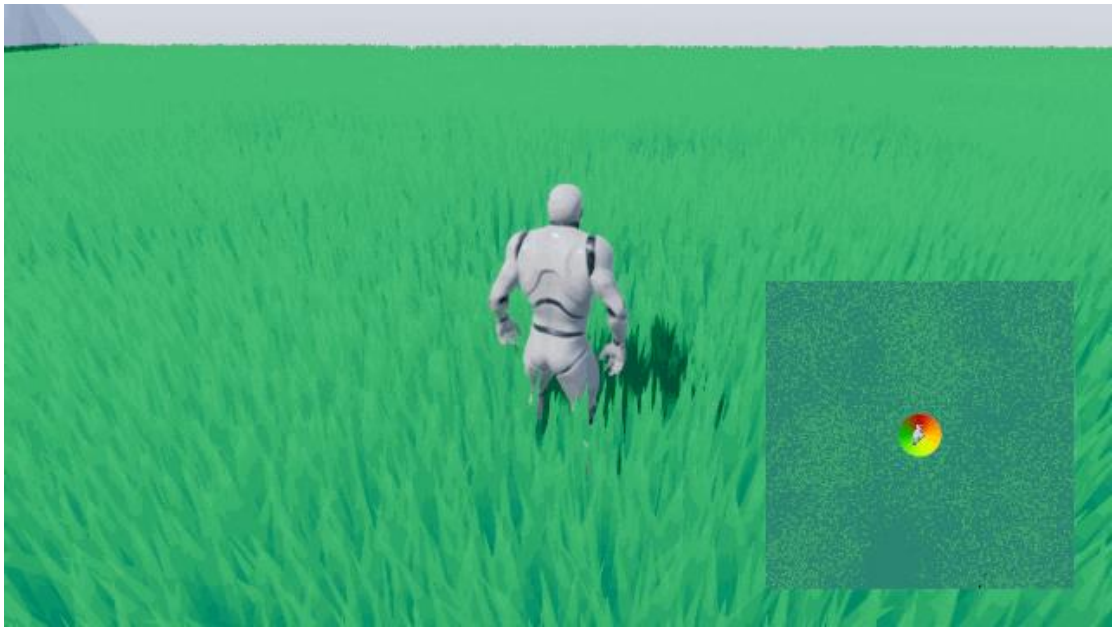


Было бы странно, если бы игрок мог видеть след в игре, поэтому стоит скрыть его. Для этого включим *Rendering\Owner No See*.



Поскольку система частиц прикреплена к игроку (owner), то игрок не увидит её, но она будет видима всему остальному.

Нажмите *Compile*, а затем нажмите *Play*. Заметьте, что частицы не появляются для камеры игрока, но отображаются на render target.

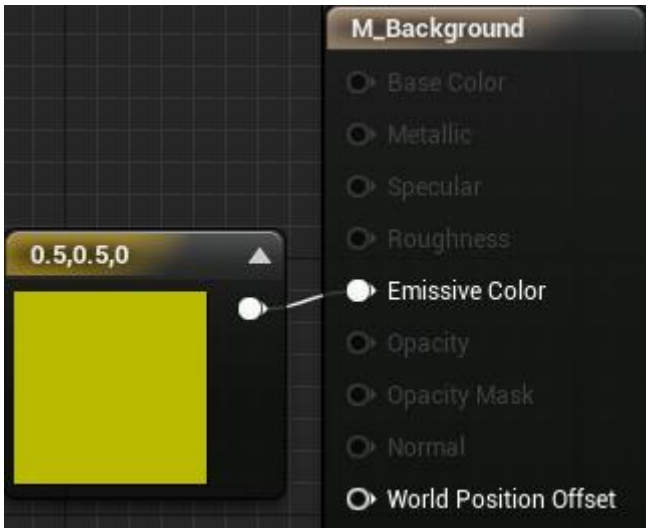


Пока захват сцены настроен на захват *всего*. Очевидно, что это нам не подходит, потому что на траву влияют только частицы. В следующем разделе мы узнаем, как выполнять захват только частиц.

Захват частиц

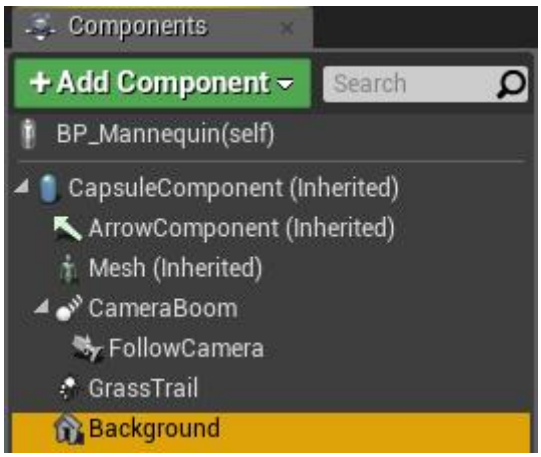
Если мы будем захватывать частицы сейчас, то будет выполняться ненужное нам сгибание в областях без частиц. Так происходит потому, что фоновым цветом render target является чёрный. Сгибание происходит потому, что чёрный обозначает движение по направлению к отрицательным осям XY (после перераспределения). Чтобы пустые области не содержали движения, нам нужно сделать так, чтобы фоновым цветом render target был *(0.5, 0.5, 0)*. Проще всего это сделать, создав огромную плоскость и прикрепив его к игроку.

Сначала создадим материал для фона. Вернитесь в Content Browser и откройте *Materials\M_Background*. Затем соедините константу *(0.5, 0.5, 0)* с *Emissive Color*.



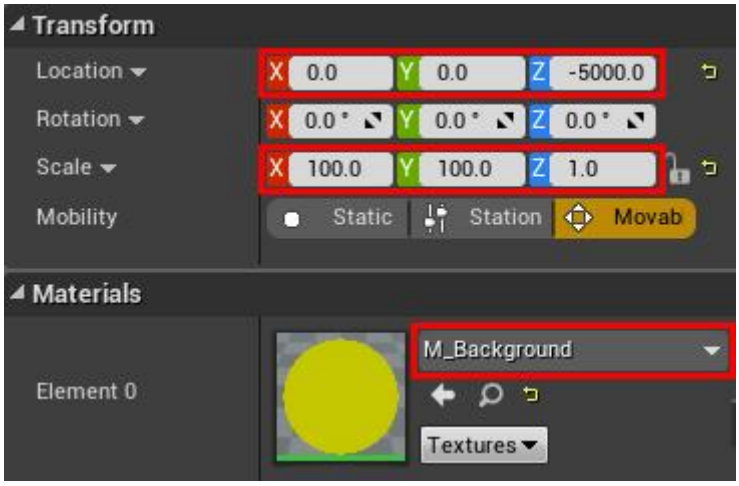
Примечание: как и в случае с материалом частиц, любой материал, который мы будем захватывать, должен иметь модель затенения *unlit*.

Нажмите *Apply* и закройте материал. Вернитесь к *BP_Mannequin*, а затем создайте новый компонент *Plane*. Назовите его *Background*.

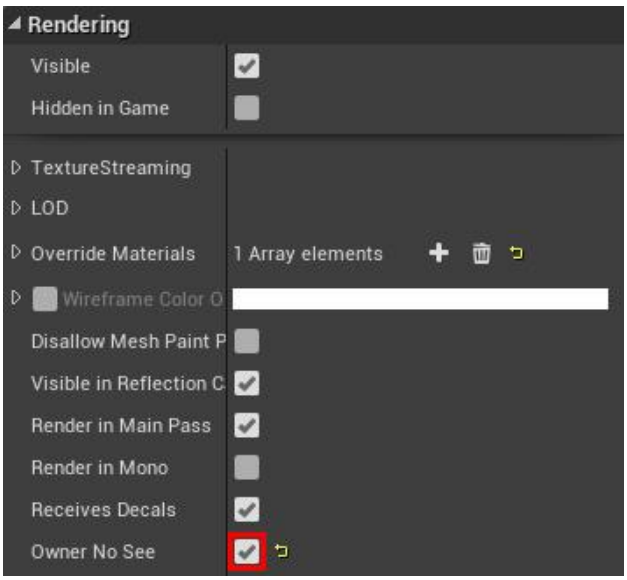


Далее задайте следующие свойства:

- *Location:* (0, 0, -5000). Мы размещаем плоскость так низко, чтобы она не перекрывала никакие частицы.
- *Scale:* (100, 100, 1). Так мы отмасштабируем до размера, достаточного для покрытия всей области захвата.
- *Material:* M_Background



Как и в случае с частицами, было бы странно, если бы игрок видел под собой огромную жёлтую плоскость. Чтобы скрыть её, включите *Rendering\Owner No See*.



Теперь, когда мы настроили фон, настало время для захвата частиц. Мы можем сделать это, добавив систему частиц к списку *show-only list* захвата сцены. Это список компонентов, которые будет захватывать захват сцены.

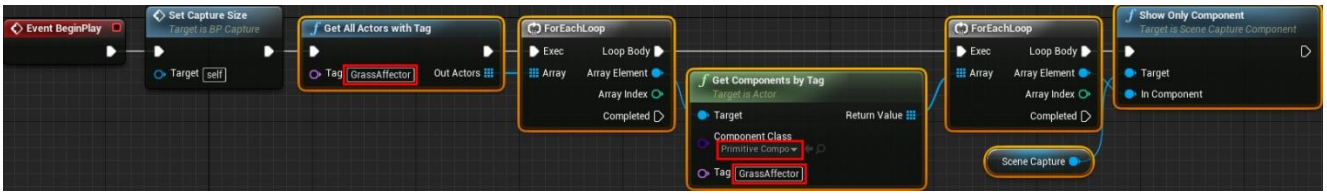
Использование Show-Only List

Прежде чем мы получим возможность добавлять в show-only list, нам необходим способ получения всех влияющих на траву акторов. Один из способов их получения — использование *тэгов*. Тэги — это простые строки, которые можно присваивать акторам и компонентам. Затем можно использовать нод *Get All Actors With Tag* для получения всех акторов с соответствующим тэгом.

Поскольку актор игрока должен влиять на траву, ему нужен тэг. Для добавления тэга нажмите на кнопку *Class Defaults*. Затем создайте в *Actor\Tags* новый тэг и назовите его *GrassAffector*.

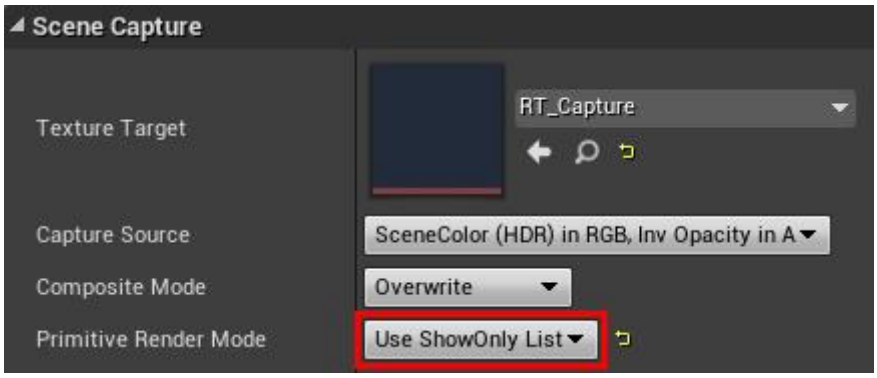
Поскольку в список show-only list можно передавать только *компоненты*, нам нужно добавить тэги и к влияющим на траву компонентам. Выберите компонент *GrassParticles* и добавьте новый тэг, расположенный в разделе *Tags*. Назовите его тоже *GrassAffector* (*необязательно* использовать именно этот тэг). Повторите то же самое для компонента *Background*.

Теперь нам нужно добавить влияющие на траву компоненты в show-only list захвата сцены. Нажмите *Compile* и закройте *BP_Mannequin*. Затем откройте *Blueprints\BP_Capture*. Перейдите к *Event BeginPlay* и добавьте выделенные ноды. Убедитесь также, что подсоединены обозначенные контакты.



Эта схему будет обходить в цикле всех акторов с тэгом *GrassAffector*. После чего она будет проверять, есть ли у актора компоненты с таким тэгом, и добавлять их в show-only list.

Далее нам нужно сказать захвату сцены, чтобы он использовал только show-only list. Выберите компонент *SceneCapture* и перейдите в раздел *Scene Capture*. Задайте для *Primitive Render Mode* значение *Use ShowOnly List*.

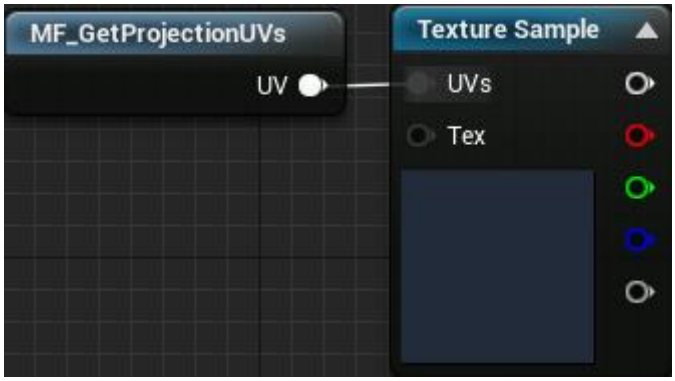


Нажмите *Compile* и закройте блюпринт. Если нажать на *Play*, то вы увидите, что render target теперь выполняет захват только частиц и плоскости фона.

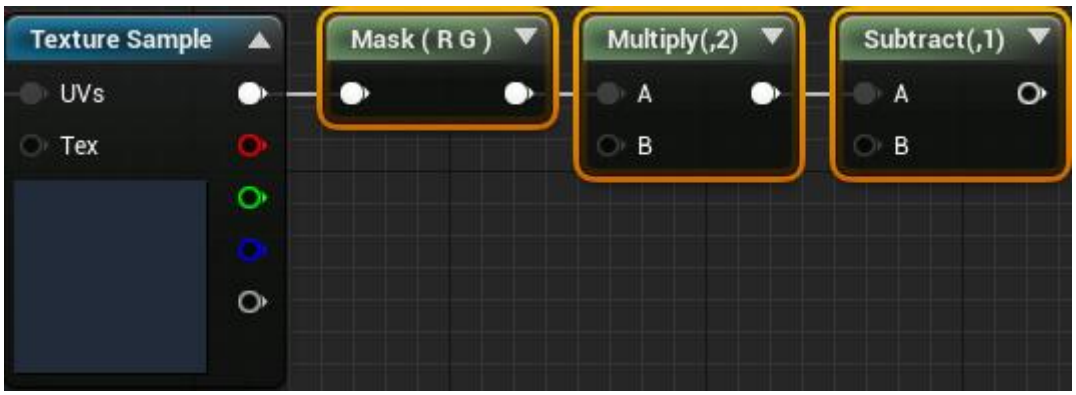
В следующем разделе мы доберёмся до того, чего ожидали. Настало время научить траву сгибаться!

Сгибаем траву

Сначала нам нужно спроецировать render target на траву. Перейдите в папку *Materials* и откройте *M_Grass*. Затем создайте показанные ниже ноды. Задайте в качестве текстуры *RT_Capture*.

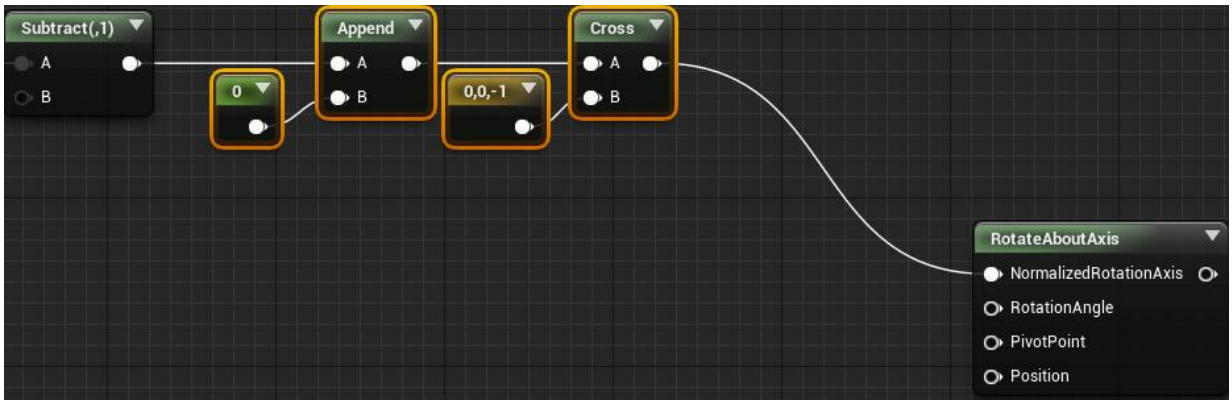


Поскольку мы перераспределили цвета в интервал от 0 до 1, то перед использованием их нужно перераспределить обратно в интервал от -1 до 1. Для этого добавим выделенные ноды:

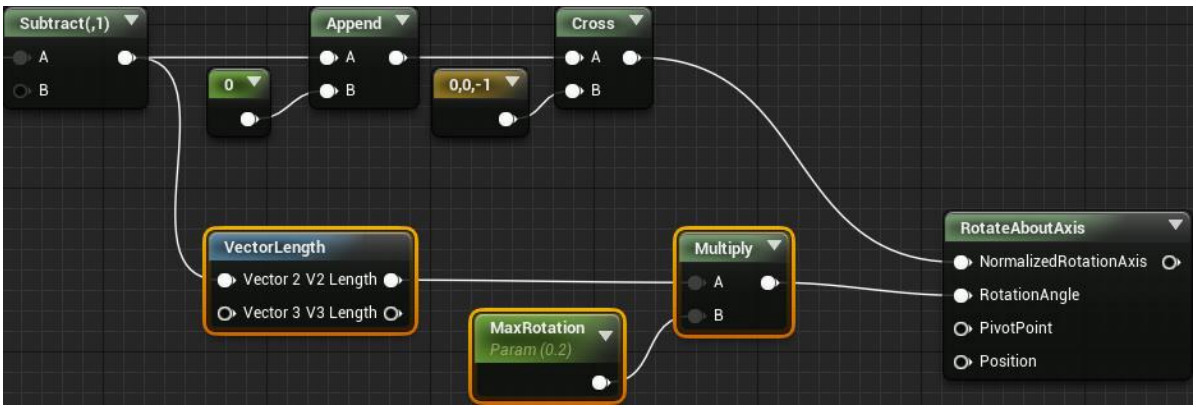


Теперь, когда у нас есть направление сгибания, нам нужен какой-то способ повернуть траву в этом направлении. К счастью, для этого существует нод под названием *RotateAboutAxis*. Давайте создадим его.

Давайте начнём с контакта *NormalizedRotationAxis*. Как понятно из названия, это ось, вокруг которой будет поворачиваться вершина. Для вычисления нам всего лишь нужно векторное произведение направления сгибания на *(0, 0, -1)*. Для этого нам нужно добавить выделенные ноды:



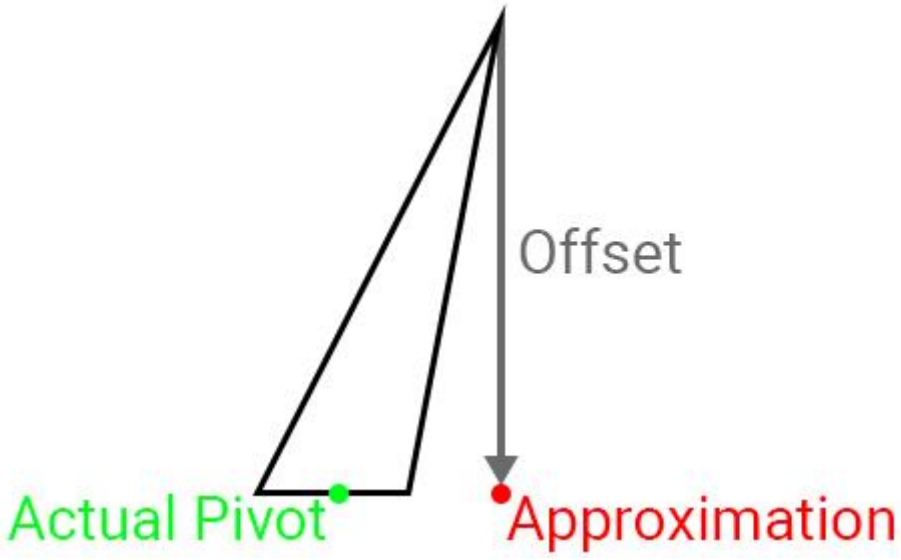
Также нам нужно указать *RotationAngle*, то есть величину поворота вершины относительно точки вращения. По умолчанию значение должно находиться в интервале от 0 до 1, где 0 — это 0 градусов, а 1 — это 360 градусов. Для получения угла поворота мы можем использовать длину направления сгибания, умноженную на максимальный поворот.



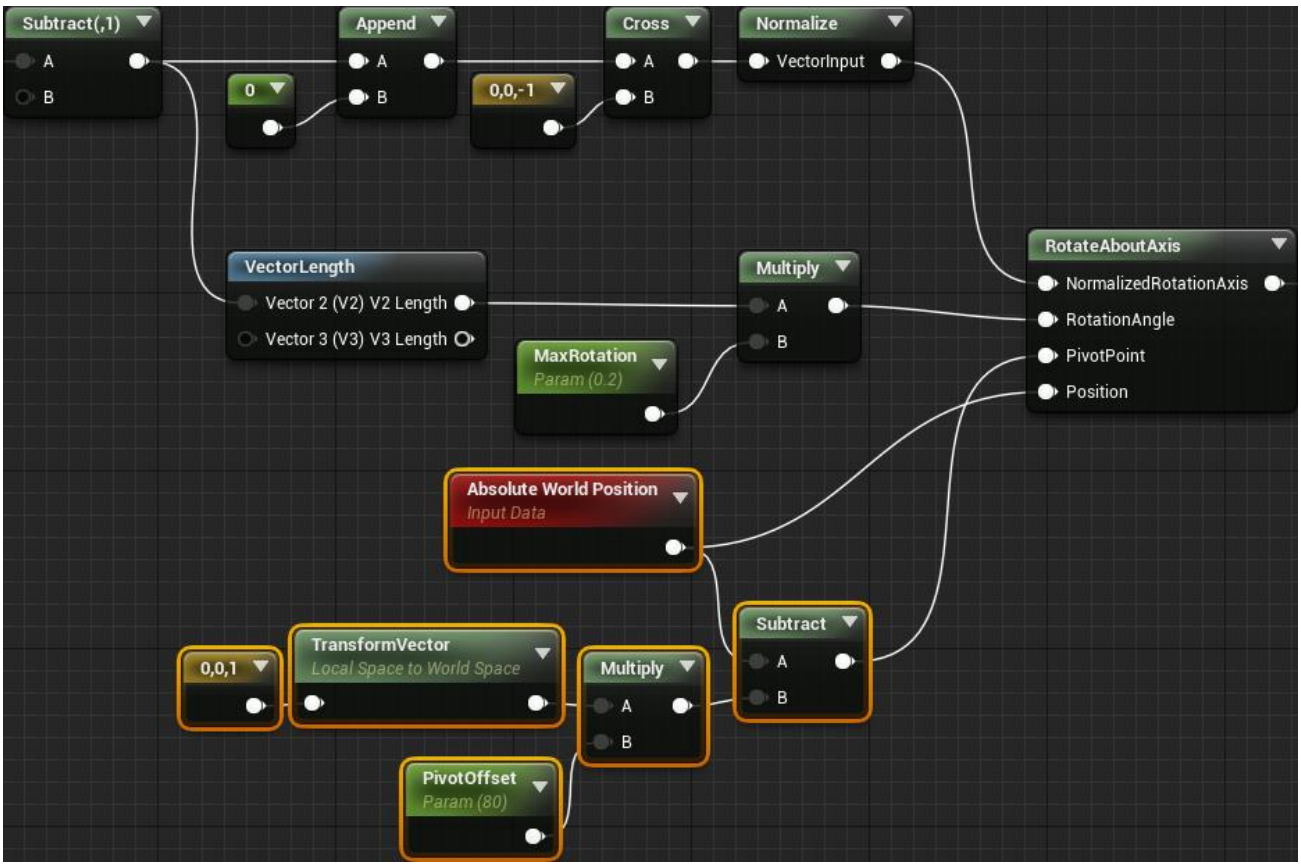
Умножение на максимальный поворот в *0.2* будет означать, что максимальный угол поворота равен *72* градусам.

Вычисление *PivotPoint* — чуть более сложная задача, потому что один меш травы содержит несколько стеблей. Это значит, что мы не можем использовать что-то вроде нода *Object Position*, потому что он будет возвращать одну точку для всех стеблей травы.

В идеале следует использовать сторонний 3D-редактор для сохранения точек вращения внутри UV-каналов. Но для этого туториала мы просто аппроксимируем точку вращения. Это можно сделать, передвинувшись с вершины *вниз* на определённое смещение.



Для этого добавим выделенные узлы:

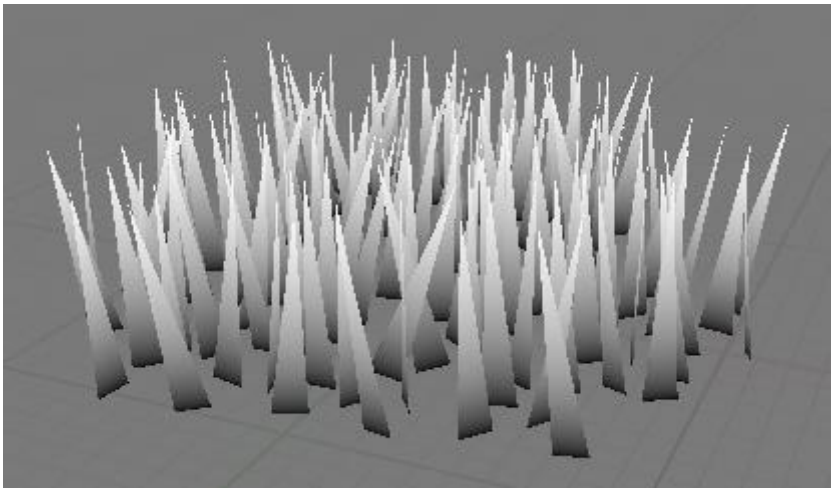


В этом туториале трава высотой примерно *80* единиц, поэтому я задаю для *PivotOffset* это значение.

Далее нам нужно выполнить две маски. Первая маска гарантирует, что корень стебля не будет двигаться. Вторая маска гарантирует, что на траву за пределами области захвата не будет действовать векторное поле.

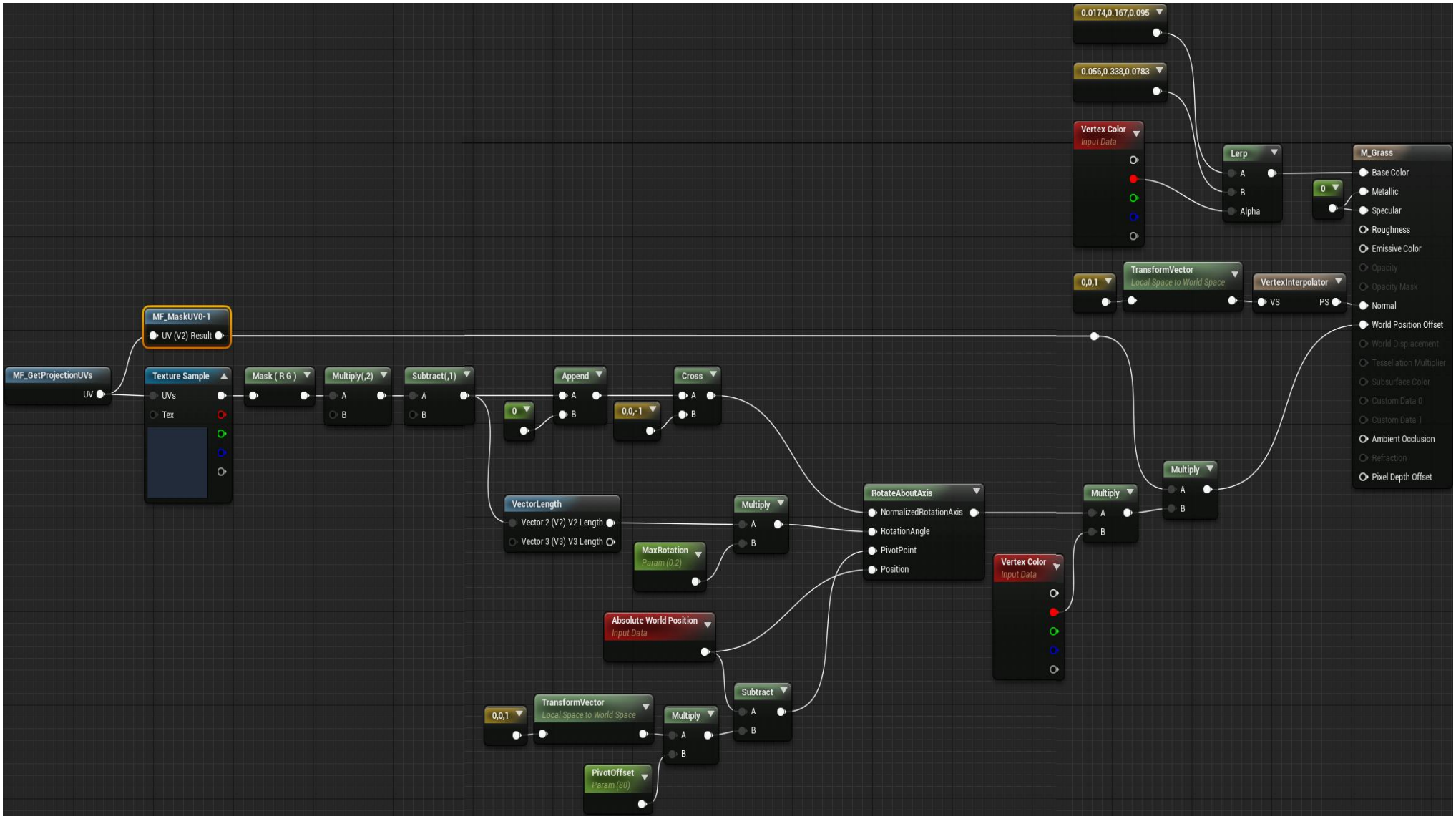
Маскирование

Для этого туториала я задал цвета вершин травы таким образом, что нижние вершины имеют чёрный цвет, а верхние — белый.



Для маскирования корней мы просто умножим результат *RotateAboutAxis* на нод *Vertex Color*.

Для маскирования травы за пределами области захвата мы умножим предыдущий результат на выделенный нод:



Нажмите *Apply* и закройте материал. Нажмите *Play* и побегайте по траве, чтобы оставить на ней следы!