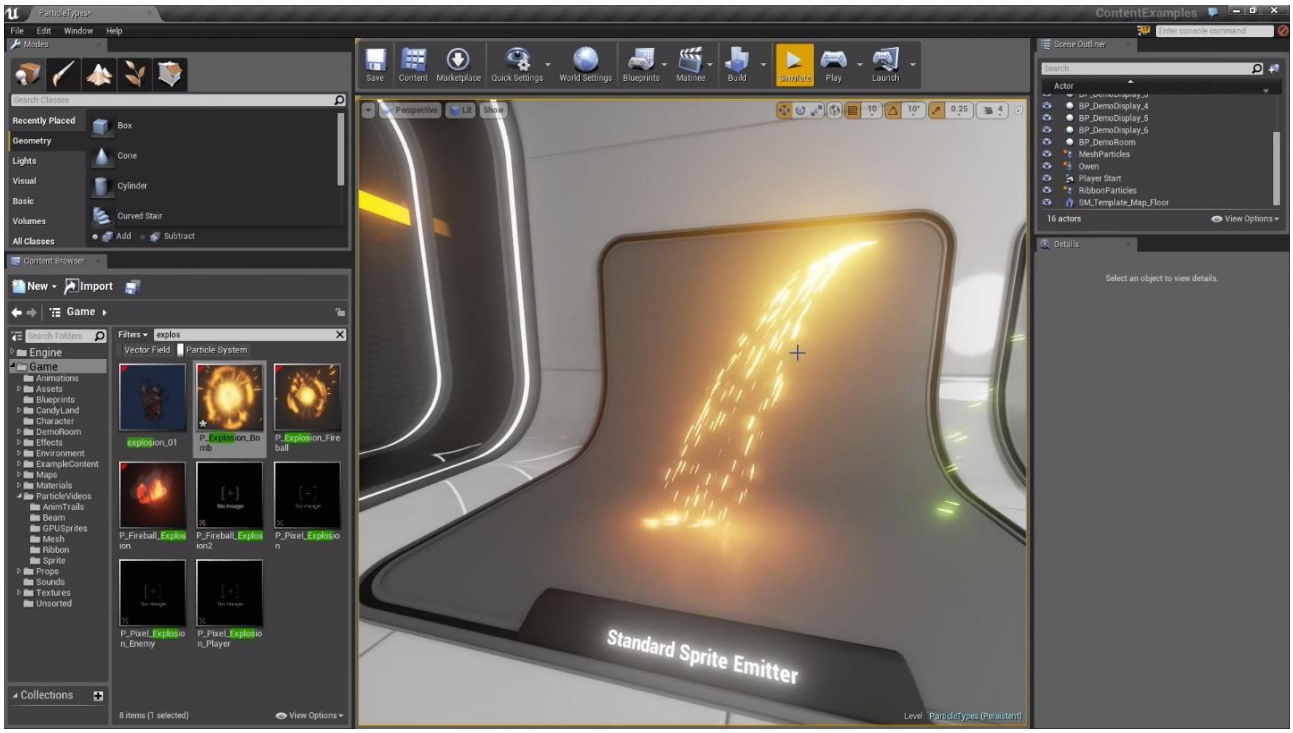
Туториал по Unreal Engine. Часть 8: Системы частиц



Системы частиц — важнейший компонент визуальных эффектов. Они позволяют художникам создавать такие эффекты, как взрывы, дым и дождь.

В Unreal Engine 4 есть надёжная и удобная система под названием Cascade для создания эффектов частиц. Эта система позволяет создавать модульные эффекты и легко управлять поведением частиц.

В этой части туториала вы научитесь следующему:

- Создавать системы частиц
- Задавать скорость и размер частиц
- Изменять частоту спауна частиц
- Масштабировать размер частиц в течение срока их существования с помощью кривых
- Задавать цвет частиц с помощью Cascade
- Активировать и деактивировать систему частиц с помощью Blueprints
- Задавать цвета частиц с помощью Blueprints

Приступаем к работе

Скачайте заготовку проекта и распакуйте её. Перейдите в папку проекта и откройте *SpaceshipBattle.uproject*.

Нажмите на *Play*, чтобы запустить игру. Удерживайте *левую клавишу мыши*, чтобы стрелять, и перемещайтесь клавишами *W*, *A*, *S* и *D*.

В этом туториале мы создадим два эффекта частиц. Один для двигателей корабля, а второй — для его взрыва. Для их создания мы воспользуемся *системами частиц*.

Что такое «система частиц»?

Как можно понять из названия, система частиц — это система для создания *частиц* и управления ими. Частица — это просто точка в пространстве. С помощью систем частиц можно управлять внешним видом и поведением частиц.

Системы частиц состоят из одного или нескольких компонентов, называемых *эмиттерами*. Они выполняют спаунинг частиц.

У эмиттеров также есть компоненты, называемые *модулями*. Модули управляют отдельными свойствами частиц, создаваемых эмиттером, например, материалом и начальной скоростью частицы. В примере ниже использованы два модуля для придания каждой частице материала красного круга и случайной скорости.

Также можно изменять цвет частицы с течением срока её существования. В этом примере цвет частицы изменяется с красного на синий:

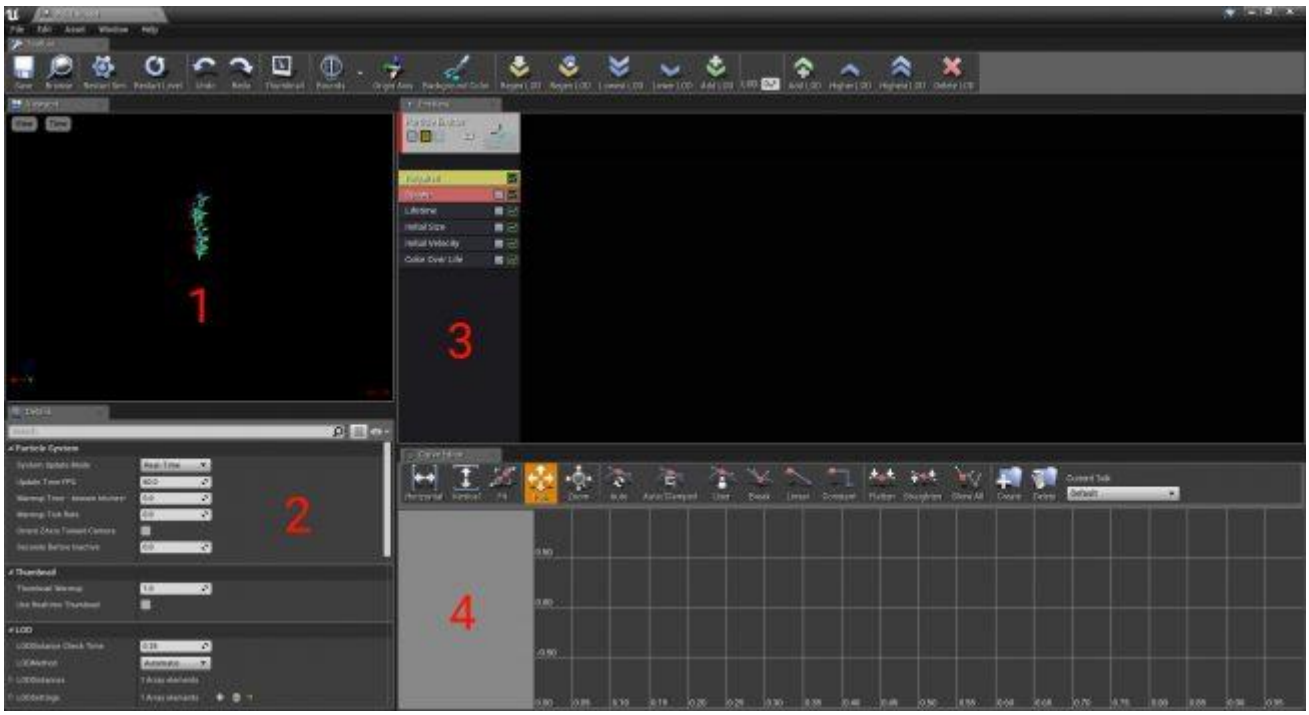
Теперь мы знаем, что такое система частиц, и мы можем создать её для двигателей корабля.

Создание системы частиц

Перейдите в папку *ParticleSystems* и нажмите на *Add New\Particle System*. Переименуйте систему частиц в *PS_Thruster* и откройте её.

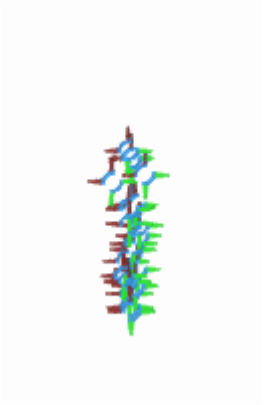
Cascade: редактор систем частиц

Cascade состоит из четырёх основных панелей:



1. *Viewport:* в этой панели отображается внешний вид системы частиц. Можно поворачивать её, удерживая *правую клавишу мыши* и двигая ею. Для перемещения удерживайте *правую клавишу мыши* и нажимайте клавиши *WASD*.
2. *Details:* здесь отображаются все свойства выбранных компонентов (эмиттеров, модулей и т.д.). Если ничего не выбрано, то здесь отображаются свойства системы частиц.
3. *Emitters:* в этой панели отображается список эмиттеров слева направо. У каждого эмиттера показан список его модулей.
4. *Curve Editor:* редактор Curve Editor позволяет визуализировать и изменять значения кривых модулей. Не все свойства модулей поддерживают кривые.

Пока наша система использует материал частиц по умолчанию.



Для начала заменим материал частиц на материал круга.

Применяем к частицам материал

Перейдите в панель Emitters и выберите модуль *Required*.

Модуль Required содержит необходимые свойства, такие как материал частиц и длительность работы эмиттера. Модуль Required должен быть у каждого эмиттера.

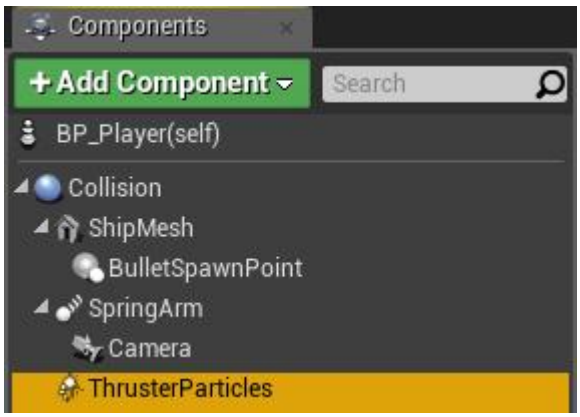
Для изменения материала перейдите в панель Details и задайте для *Material* значение *M_Particle*. При этом частицы станут оранжевыми кругами.

Теперь мы присоединим систему частиц к кораблю игрока.

Присоединение системы частиц

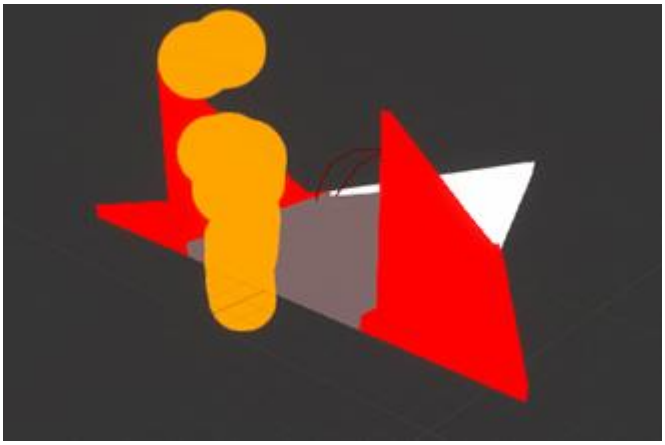
Вернитесь в основной редактор и зайдите в папку *Blueprints*. Откройте *BP_Player* и перейдите к панели Components.

Для использования системы частиц можно применить компонент *Particle System*. Создайте его и переименуйте в *ThrusterParticles*. Соедините его с компонентом *Collision*.

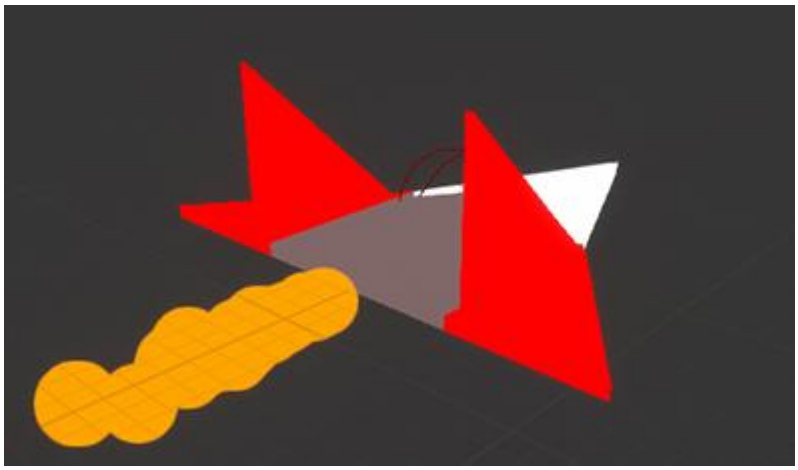


Для задатия системы частиц перейдите в панель Details и найдите раздел *Particles*. Задайте *Template* значение *PS_Thruster*.

Затем задайте для для *Location* компонента *ThrusterParticles* значения *(-80, 0, 0)*. При этом частицы расположатся за кораблём.



Наконец, задайте *Rotation* значения *(0, 90, 0)*. Это направит систему частиц таким образом, что частицы будут удаляться от корабля.



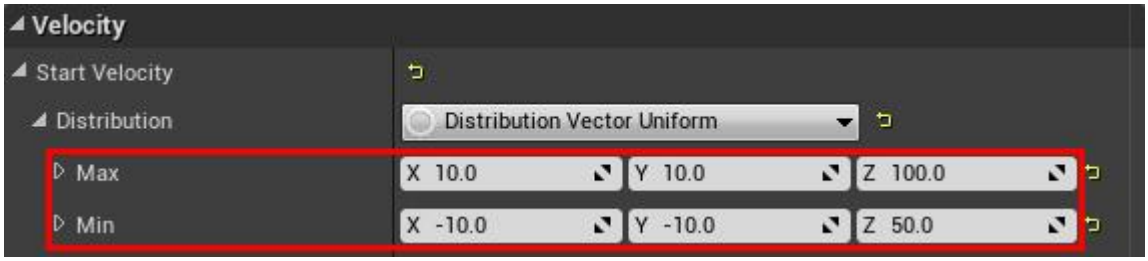
Нажмите на *Compile* и вернитесь в основной редактор. Нажмите на *Play*, чтобы увидеть систему частиц в действии.

Система частиц работает, но частицы движутся слишком медленно и они очень малы. Это можно исправить, задав начальную скорость и размер частиц.

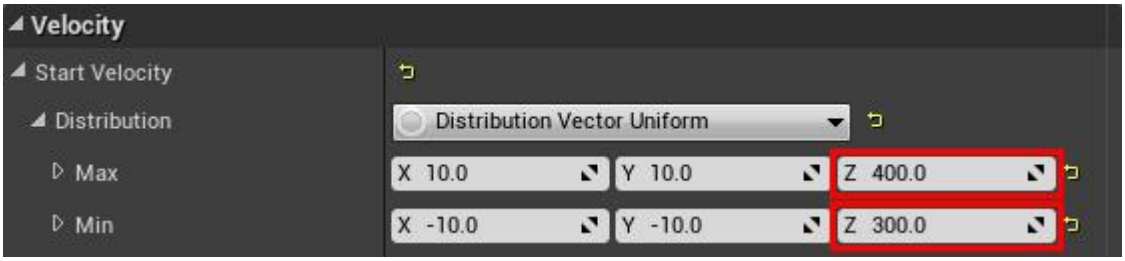
Задание скорости и размера частиц

Сначала мы зададим начальную скорость частиц. Откройте *PS_Thruster* и выберите модуль *Initial Velocity*. Затем разверните *Start Velocity\Distribution*.

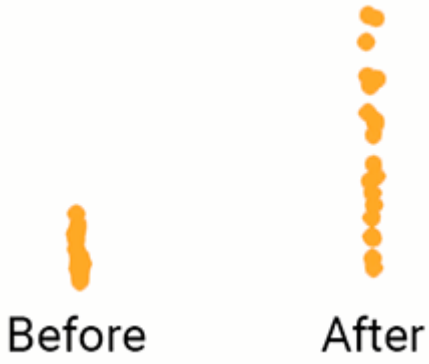
По умолчанию начальная скорость частиц находится в интервале от *(-10, -10, 50)* до *(10, 10, 100)*.



Чтобы частицы удалялись от корабля быстрее, нам нужно увеличить скорость по *Z*. Присвойте *Min Z* значение *300*, а *Max Z* — значение *400*.



Вот сравнение исходной и новой скоростей:

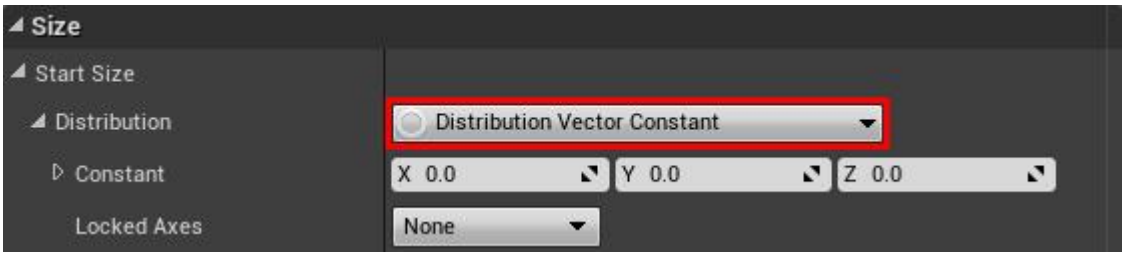


Далее нам нужно задать начальный размер частиц.

Задание размера частиц

Выберите модуль *Initial Size* и перейдите в панель Details. Затем разверните *Start Size\Distribution*.

Как и в модуле Initial Velocity, в Initial Size тоже есть интервал минимальных и максимальных значений. Однако в этом туториале мы зададим постоянный размер частиц. Для этого выберите для *Distribution* значение *Distribution Vector Constant*.



Примечание: распределения (Distributions) позволяют задавать постоянные значения, значения в интервале или на кривой. Также можно задавать значения с помощью Blueprints. Чтобы узнать больше, изучите страницу Distributions в документации Unreal Engine.

Затем задайте *Constant* значения *(70, 70, 70)*. Вот иллюстрация сравнения размеров:

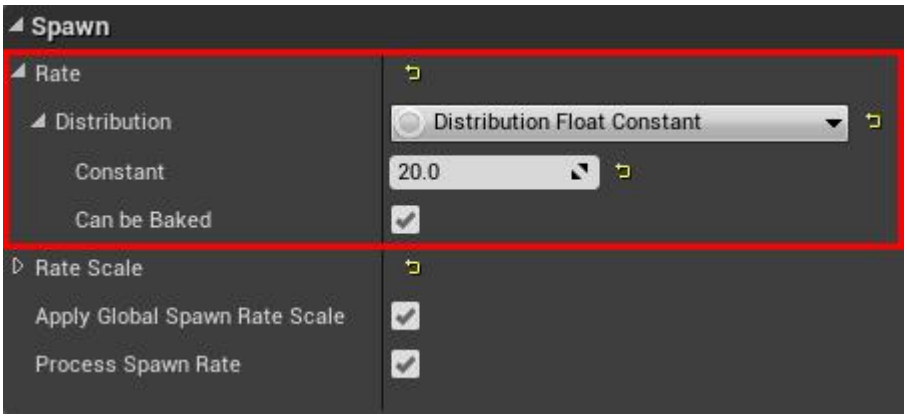
Вернитесь в основной редактор и нажмите на *Play*.

Частицы выглядят лучше, но расстояние между ними по-прежнему слишком большое. Так получилось из-за слишком длительного интервала между спауном частиц. Чтобы исправить это, мы можем увеличить скорость спауна.

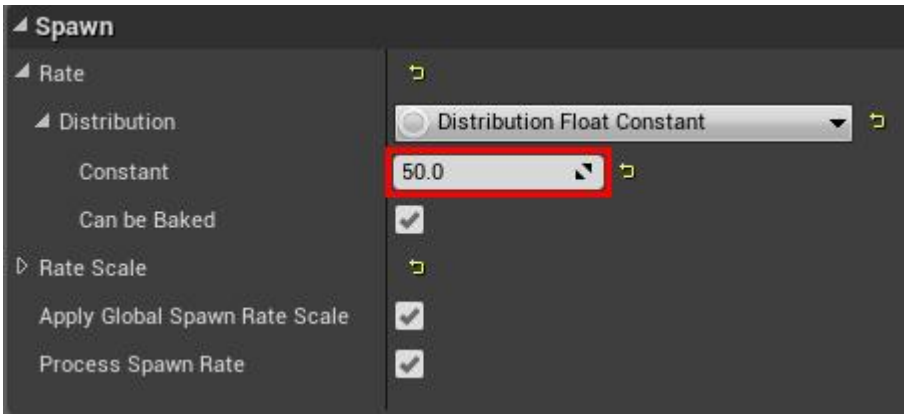
Увеличение скорости спауна частиц

Для увеличения скорости спауна нам нужно воспользоваться модулем *Spawn*. Этот модуль управляет скоростью спауна частиц эмиттером. Вместе с Required каждый эмиттер должен иметь модуль Spawn.

Откройте *PS_Thruster* и выберите *Spawn*. Перейдите в панель Details и разверните раздел *Spawn\Rate*.



Задайте *Constant* значение *50*. Это увеличит скорость спауна до 50 частиц в секунду.



Вернитесь в основной редактор и нажмите на *Play*.

Как вы видите, теперь частицы больше напоминают след. Чтобы сделать частицы больше похожими на пламя двигателя, можно уменьшить срок их жизни.

Уменьшение срока существования частиц

Откройте *PS_Thruster* и перейте в панель Emitters.

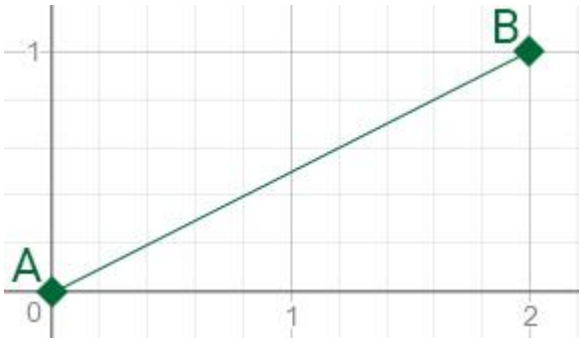
Чтобы уменьшить время существования частиц, нужно воспользоваться модулем *Size By Life*. Этот модуль применяет множитель размера частицы в течение срока её существования. Создайте его, *нажав правой клавишей мыши* на пустом пространстве в эмиттере и выбрав *Size\Size By Life*.

По умолчанию это не повлияет визуально на размер частиц, потому что по умолчанию множитель всегда равен 1. Чтобы уменьшить частицу, нам нужно изменить *кривую* модуля, чтобы множитель размера со временем уменьшался. Но сначала нужно разобраться, что же такое кривая?

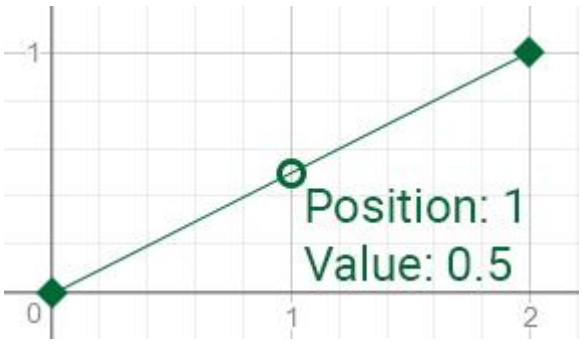
Что такое «кривая»?

Кривая (curve) — это множество точек. Каждая точка обладает двумя свойствами: положением и значением.

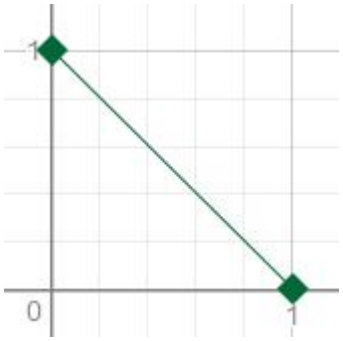
Когда у нас есть две или более точек, то они создают линию. Ниже представлен пример простой линейной кривой. *Точка A* имеет положение и значение, равные *0*. *Точка B* имеет положение *2* и значение *1*.



Если сэмплировать линейную кривую в любой точке, то это работает как линейная интерполяция. Например, если сэмплировать кривую в точке *1*, то мы получим значение *0.5*.



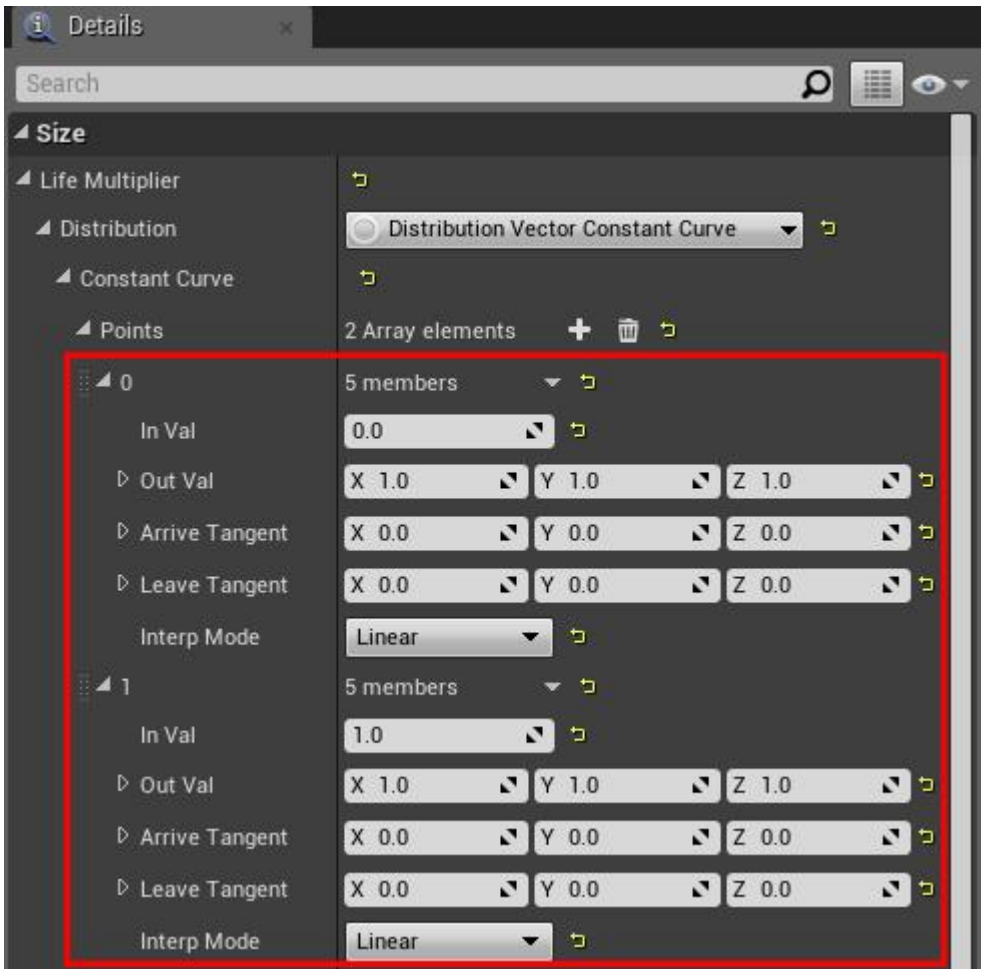
Если создать убывающую кривую, то получаемое значение постепенно будет уменьшаться. Именно такую кривую мы хотим использовать для модуля Size By Life.



Теперь мы создадим показанную выше кривую в Cascade.

Изменение кривой модуля

Выберите *Size By Life* и перейдите в панель Details. Разверните *Life Multiplier\Distribution\Constant Curve\Points*. Здесь представлен список точек кривой *Life Multiplier*.

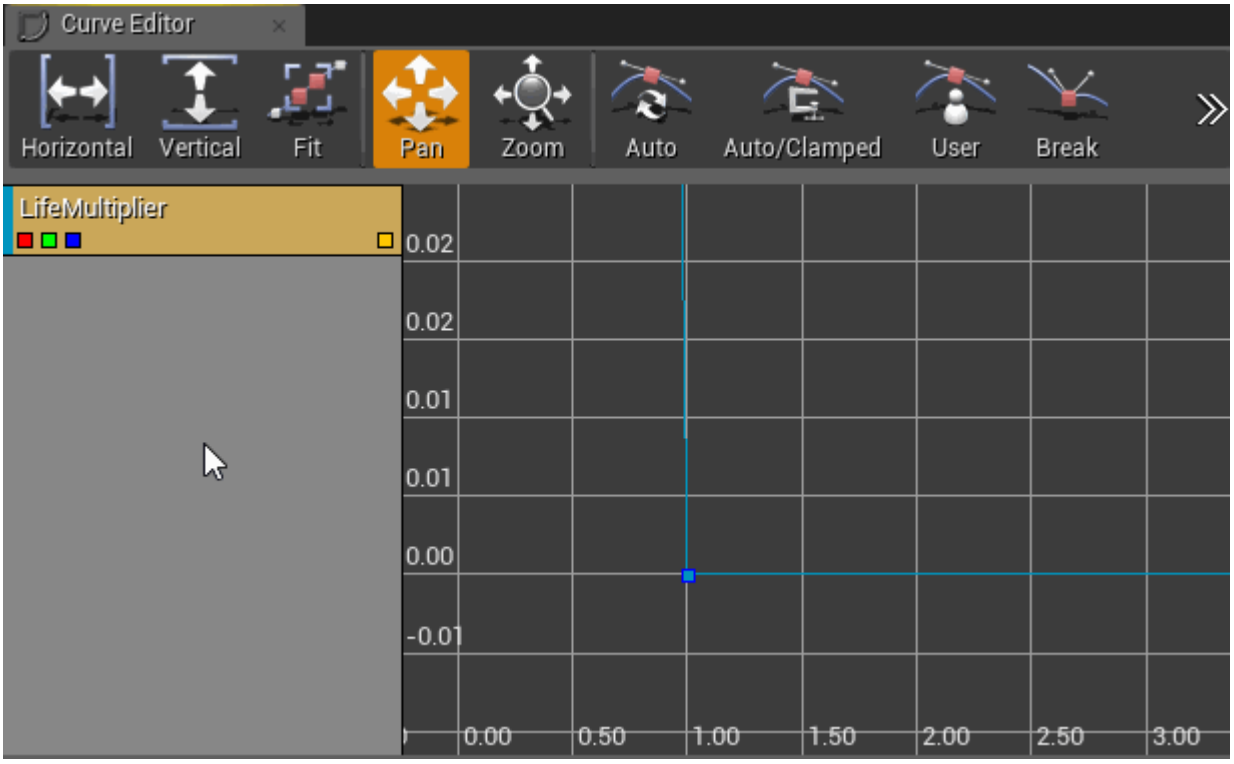


In Val — это положение точки на кривой. Для *Size By Life* значение *0* обозначает *начало* срока жизни частицы. Значение *1* обозначает *конец* срока жизни частицы.

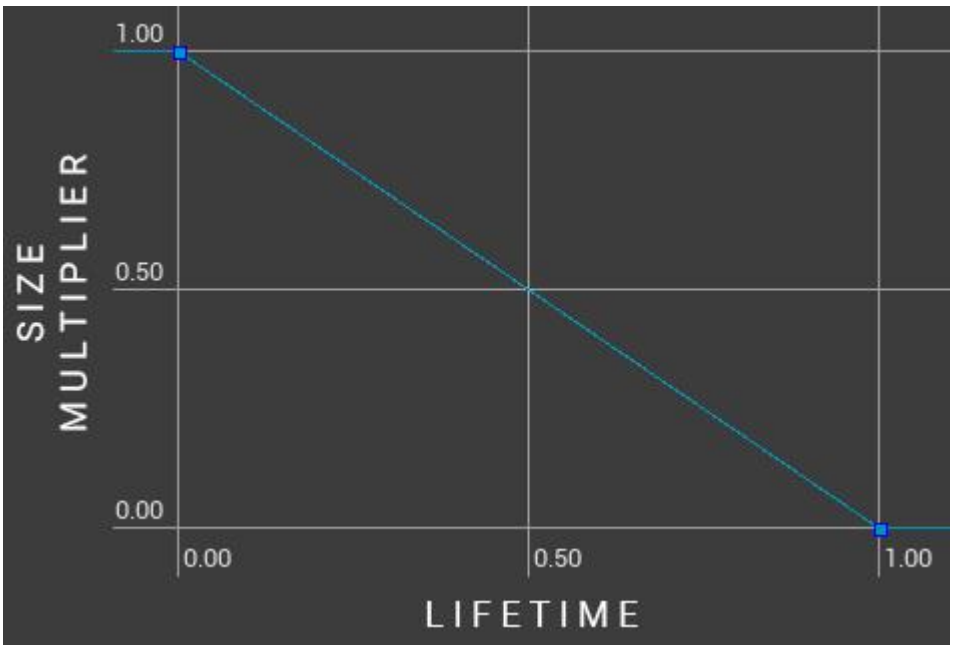
Для уменьшения множителя размера со временем нам нужно уменьшить *Out Val* второй точки. Задайте для *Out Val точки 1* значение *(0, 0, 0)*. Это постепенно снизит размер частицы до 0.

Наглядно увидеть кривую *Life Multiplier* можно в Curve Editor. Для этого нажмите на *значок графа* модуля *Size By Life*.

Это добавит *Life Multiplier* в Curve Editor. Чтобы кривая помещалась в окно, нажмите на *Fit* в Curve Editor.



Как вы видите, множитель размера уменьшается за срок жизни частицы с 1 до 0.

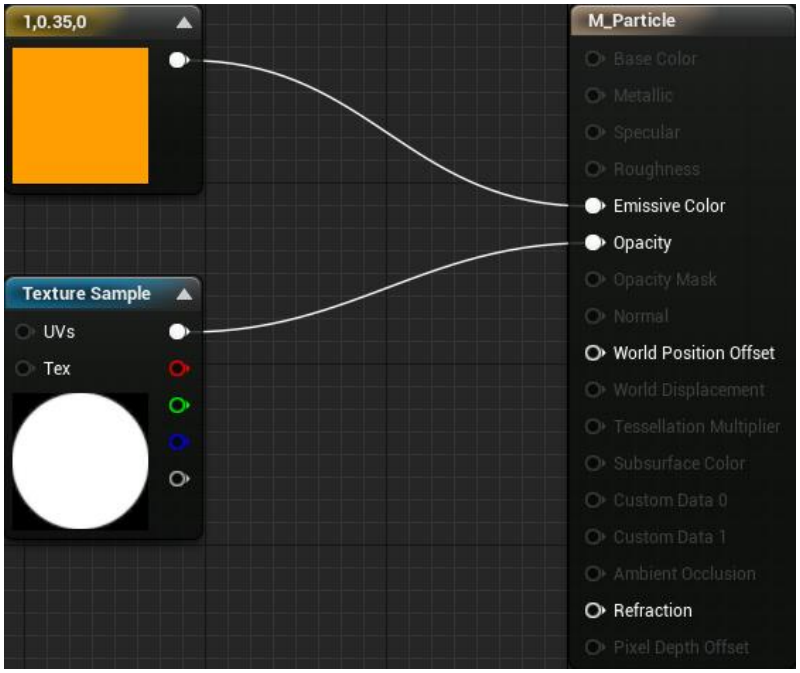


Вернитесь в основной редактор и нажмите на *Play*

Теперь частицы больше походят на пламя! Последнее, что мы добавим к этой системе частиц — цветовые вариации.

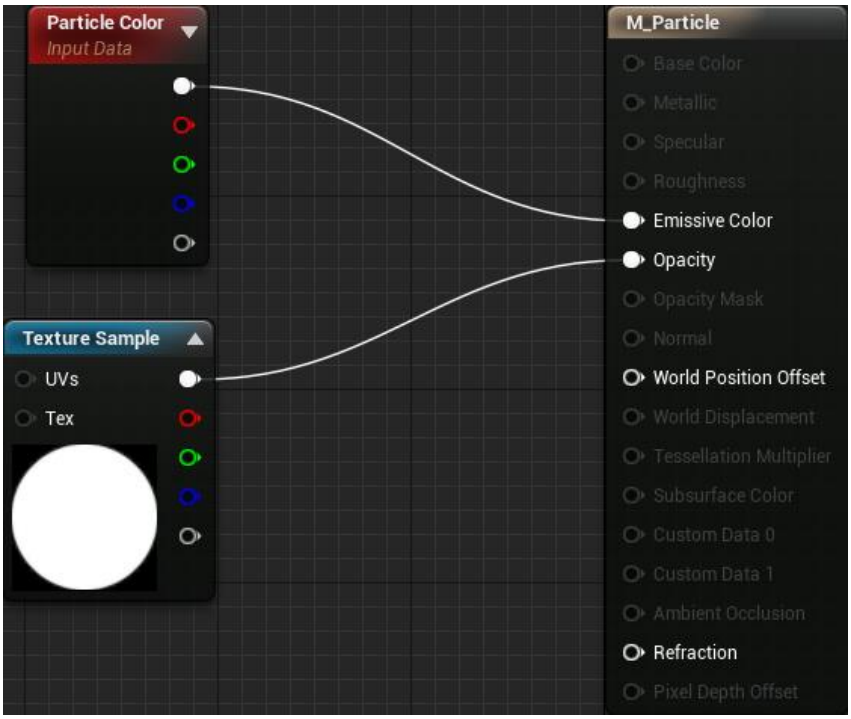
Добавление цветовых вариаций

Чтобы задать цвет частицы с помощью Cascade, нам нужно правильно выбрать материал частиц. Перейдите в папку *Materials* и откройте *M_Particle*.



Сейчас цвет задан в материале. Чтобы использовать цвет из системы частиц, нам нужно применить нод *ParticleColor*.

Во-первых, удалите нод, соединённый с *Emissive Color*. Затем добавьте нод *ParticleColor* и соедините его следующим образом:

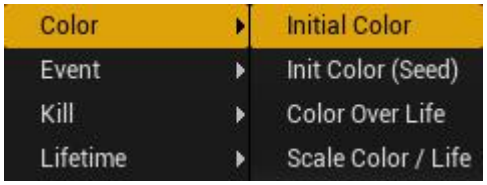


Дополнительно

Нажмите на *Apply* и закройте *M_Particle*.
Чтобы задать цвет частицы, можно использовать модуль *Initial Color*.

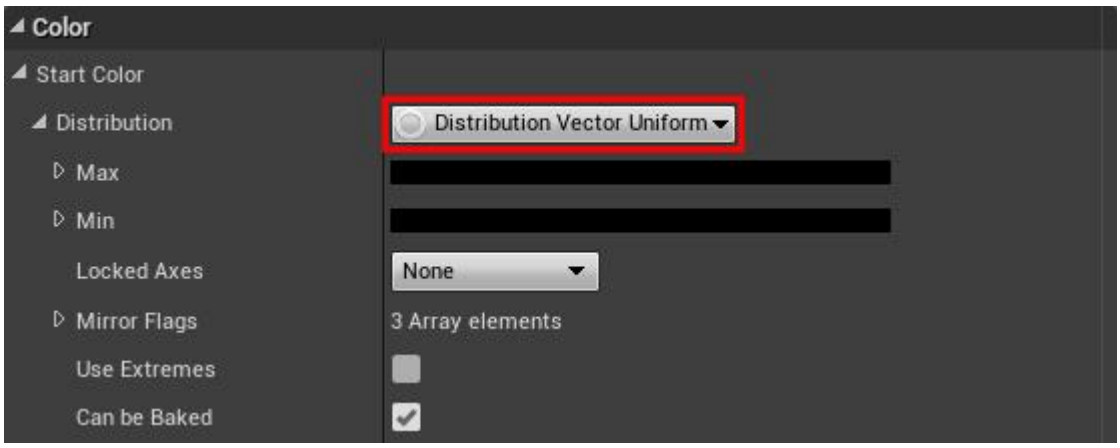
Модуль Initial Color

Откройте *PS_Thruster* и добавьте модуль *Initial Color*. Его можно найти в категории *Color*.

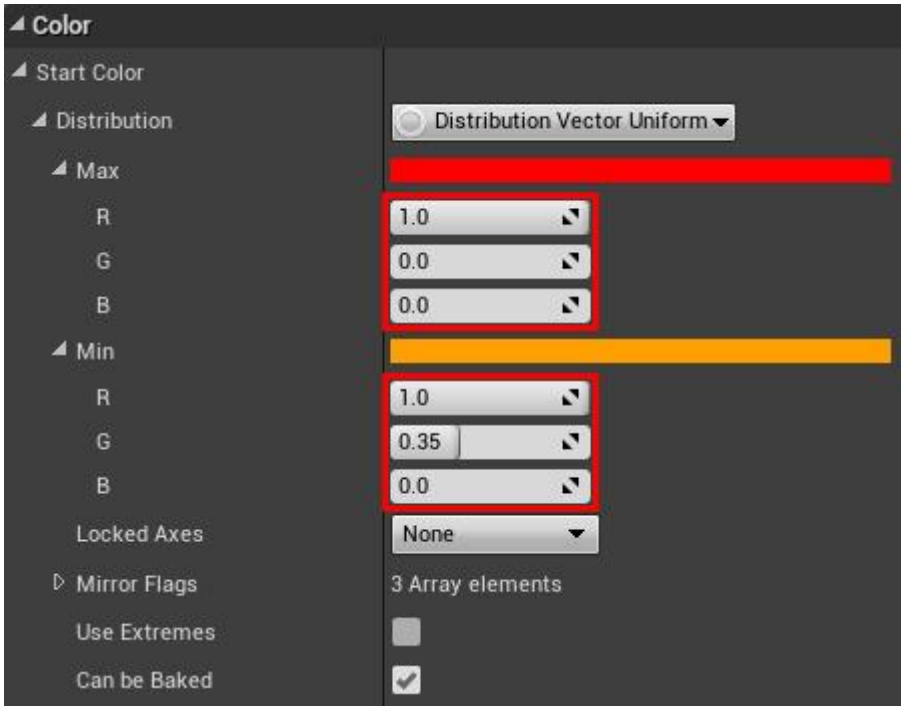


Чтобы добавить цветовые вариации, нам нужно задать интервал, в котором может находиться цвет. Для этого можно воспользоваться распределениями.

Выберите *Initial Color* и перейдите в панель Details. Разверните раздел *Start Color* и измените *Distribution* на *Distribution Vector Uniform*. Это позволит нам указать интервал для каждого цветового канала.



В этом туториале цвет будет находиться в интервале от оранжевого до красного. Для этого нужно задать *Max* значения *(1.0, 0.0, 0.0)*, а *Min* — значения *(1.0, 0.35, 0.0)*.



Если вы посмотрите на Viewport, то увидите, что цвет ведёт себя странно.

Так происходит потому, что модуль *Color Over Life* постоянно делает цвет белым. Чтобы исправить это, выберите *Color Over Life* и нажмите *Delete*. Теперь ваш список модулей будет выглядеть так:

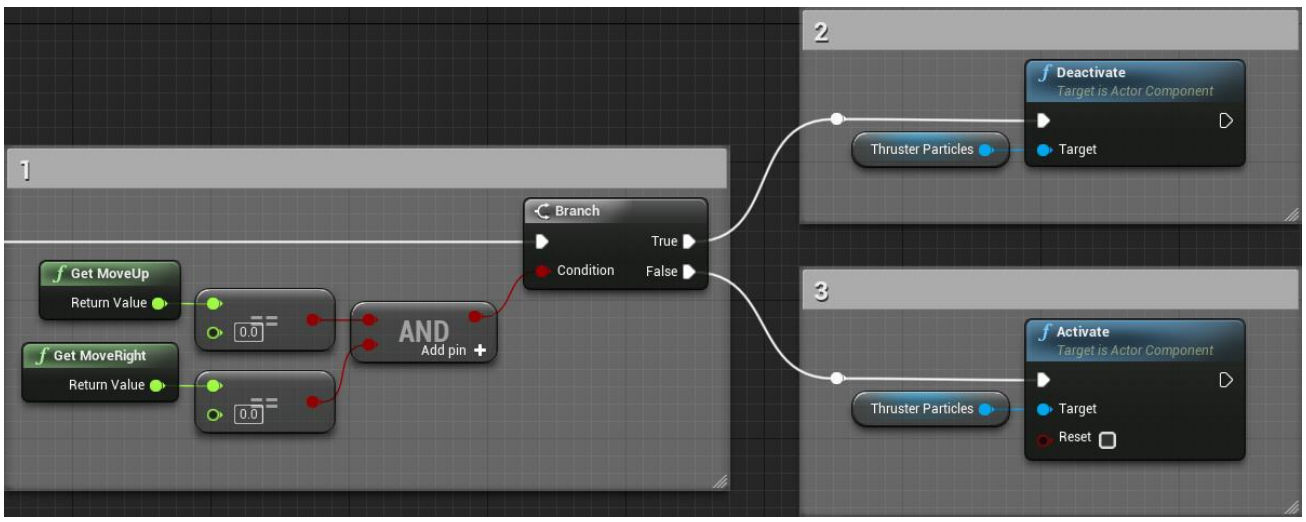
Закройте *PS_Thruster* и нажмите на *Play* в основном редакторе. Полюбуйтесь на эти языки пламени двигателя!

Теперь нам нужно научиться переключать систему частиц в зависимости от того, движется ли корабль.

Включение/отключение системы частиц

Чтобы проверить, движется ли корабль, мы можем проверять, нажимает ли игрок клавиши движения.

Откройте *BP_Player* и найдите нод *Event Tick*. Добавьте следующую схему в конце цепочки нодов:



Давайте разберёмся с тем, что делает эта схема:

1. Она проверяет привязки осей *MoveUp* и *MoveRight*. Если обе возвращают *0*, то это значит, что игрок не нажимает клавиш движения.
2. Если *Branch* возвращает *true* (игрок не нажимает клавиши движения), то деактивируется *ThrusterParticles*

3. Если *Branch* возвращает *false* (игрок нажимает клавишу движения), то активируется *ThrusterParticles*

Нажмите на *Compile* и закройте *BP_Player*. Нажмите на *Play*, а потом нажимайте и отпускайте клавиши движения, чтобы увидеть разницу.

Теперь настало время для самого интересного: создадим систему частиц взрыва!

Создание эффекта взрыва

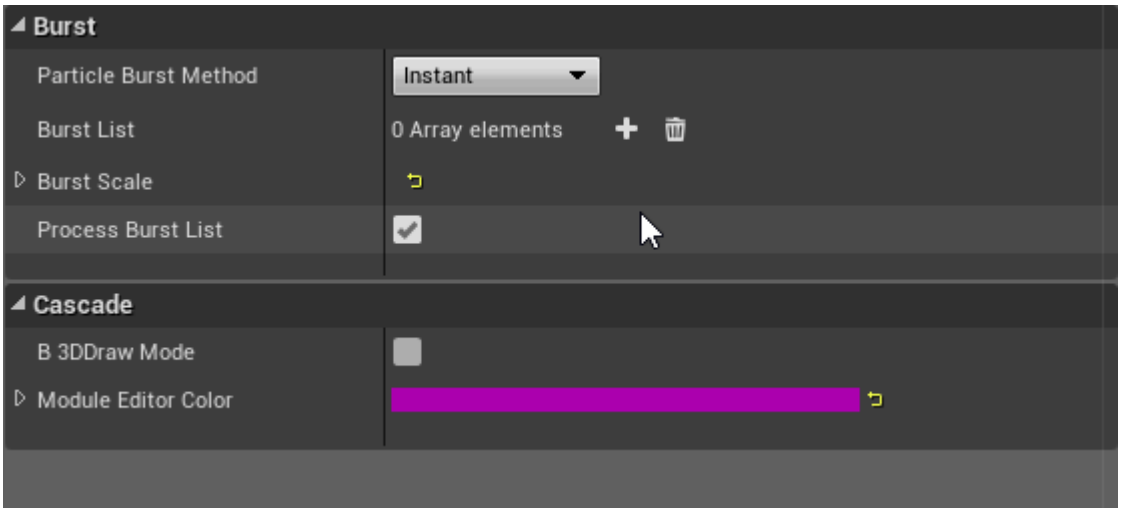
Вместо создания новой системы частиц, мы дублируем частицы двигателя. Перейдите в папку *ParticleSystems, нажмите правой клавишей мыши* на *PS_Thruster* и выберите *Duplicate*. Переименуйте дубликат в *PS_Explosion* и откройте его.

Для взрыва все частицы должны спауниться одновременно, а не одна за другой. Этот эффект называется *импульсное испускание*.

Создание импульса

Для начала нам нужно задать скорость спауна, равную нулю, потому что мы не хотим использовать поведение спауна по умолчанию. Выберите модуль *Spawn* и задайте *Spawn\Rate\Distribution\Constant* значение *0*.

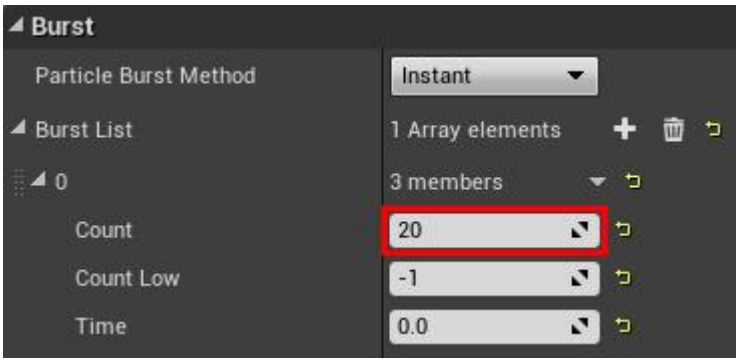
Далее нужно сообщить эмиттеру, что мы хотим создать импульс. Прокрутите вниз, к разделу *Burst* и добавьте новый элемент в *Burst List*. Это можно сделать, нажав на значок +.



Каждый элемент будет содержать три поля:

1. *Count:* количество создаваемых частиц. Укажите значение *20*.
2. *Count Low:* если больше или равно 0, то количество создаваемых частиц будет изменяться в интервале от *Count Low* до *Count*. Оставим здесь значение *-1*.
3. *Time:* момент спауна частиц. Значение 0 обозначает начало срока жизни эмиттера. Значение 1 обозначает конец срока жизни эмиттера. Оставим здесь значение *0.0*.

Примечание: срок жизни эмиттера находится в модуле *Required*. Он указан как *Emitter Duration* в разделе *Duration*.

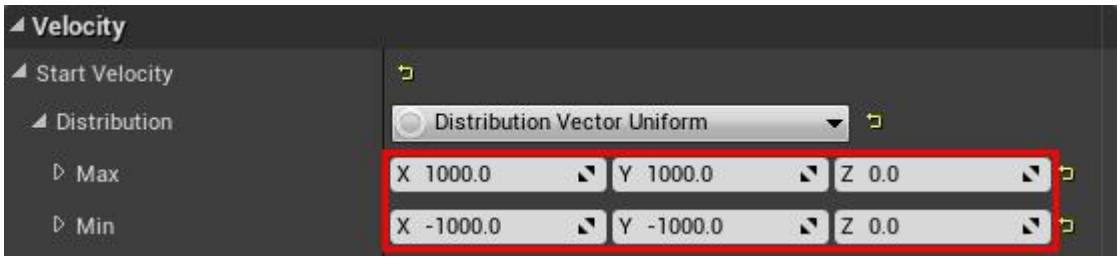


Это значит, что эмиттер создаст *20* частиц в *начале* своей жизни.

Чтобы сделать импульс похожим на взрыв, нам нужно добавить скорость, с которой частицы будут разлетаться.

Разлетание частиц

Поскольку игра имеет вид сверху, нам нужно указать только скорости по X и Y. Выберите модуль *Initial Velocity* и разверните *Start Velocity\Distribution*. Задайте *Max* значение *(1000, 1000, 0)*, а *Min* — значение *(-1000, -1000, 0)*.



Мы указали интервал от отрицательных до положительных значений, поэтому частицы будут разлетаться от эмиттера.

Теперь нужно задать количество повторов эмиттера.

Задание повторов эмиттера

По умолчанию эмиттеры повторяются бесконечно. Это отлично подходит для таких эффектов. как дым и огоно, но импульс должен проигрываться только один раз. Чтобы исправить это, нам нужно сообщить эмиттеру, что он должен повторяться только один раз.

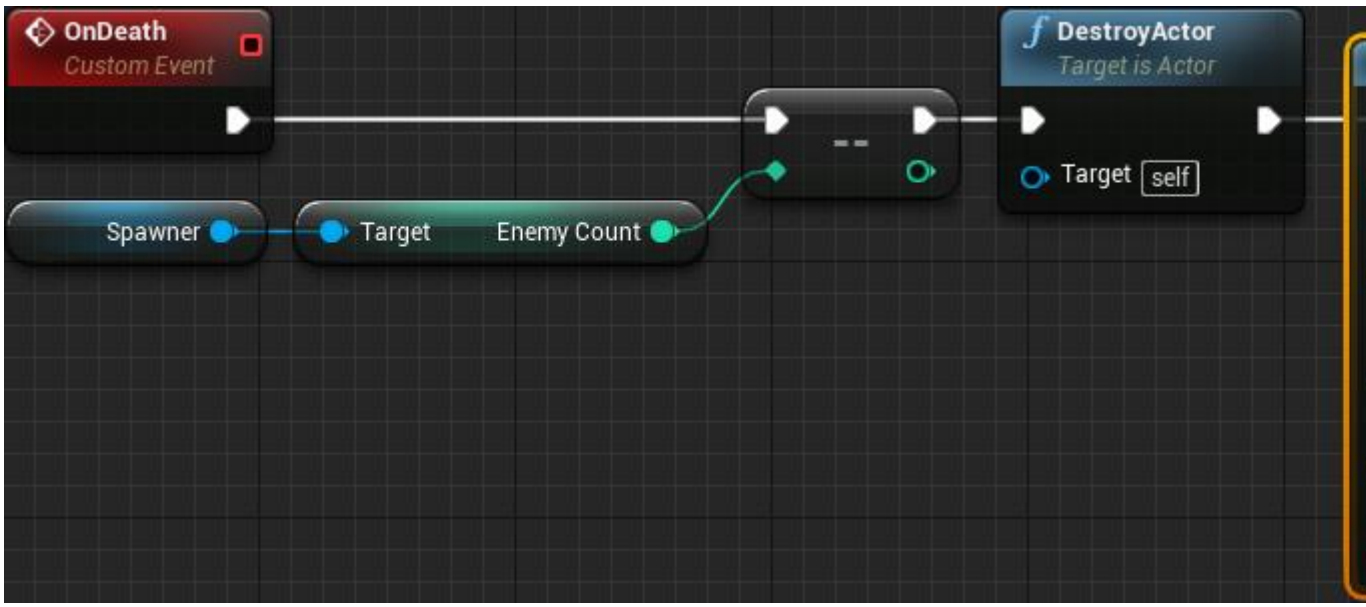
Выберите модуль *Required* и найдите раздел *Duration*. Задайте *Emitter Loops* значение *1*.

Теперь мы будем воспроизводить взрыв при смерти врага.

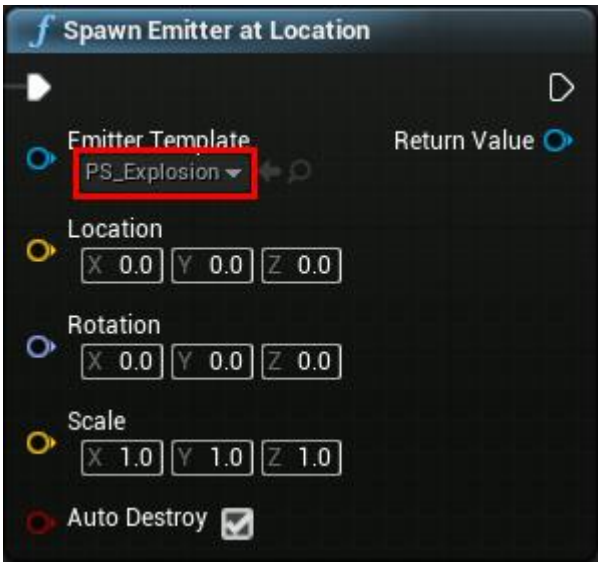
Создание частиц при смерти противника

Вернитесь в основной редатор и перейдите в папку *Blueprints*. Откройте *BP_Enemy* и найдите событие *OnDeath*.

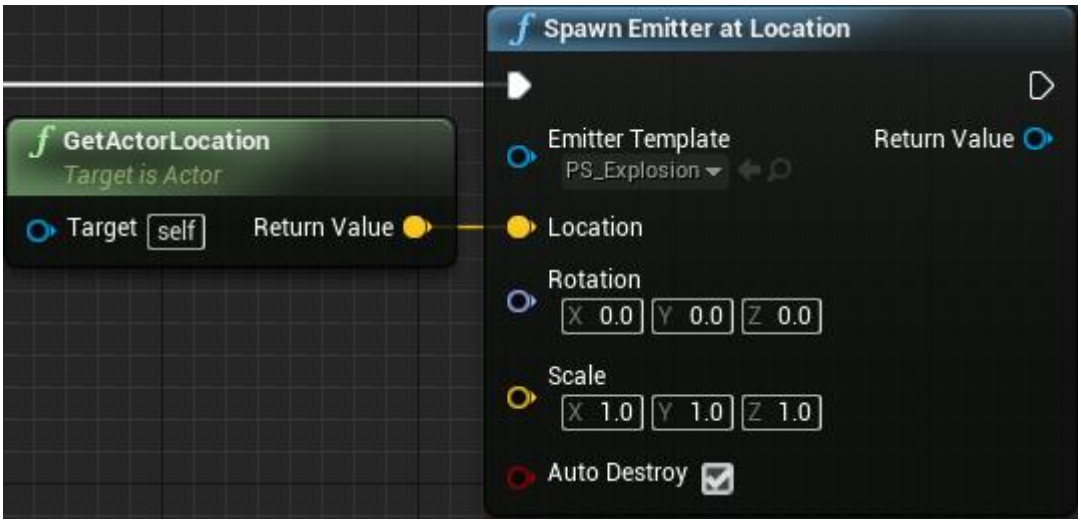
Чтобы заспаунить систему частиц, можно использовать нод *Spawn Emitter at Location*. Создайте его и соедините с *Destroy Actor*.



Затем задайте *Emitter Template* значение *PS_Explosion*.



Наконец, создайте *GetActorLocation* и соедините его с контактом *Location*.



Теперь при смерти врага событие будет создавать экземпляр *PS_Explosion* в точке расположения врага.

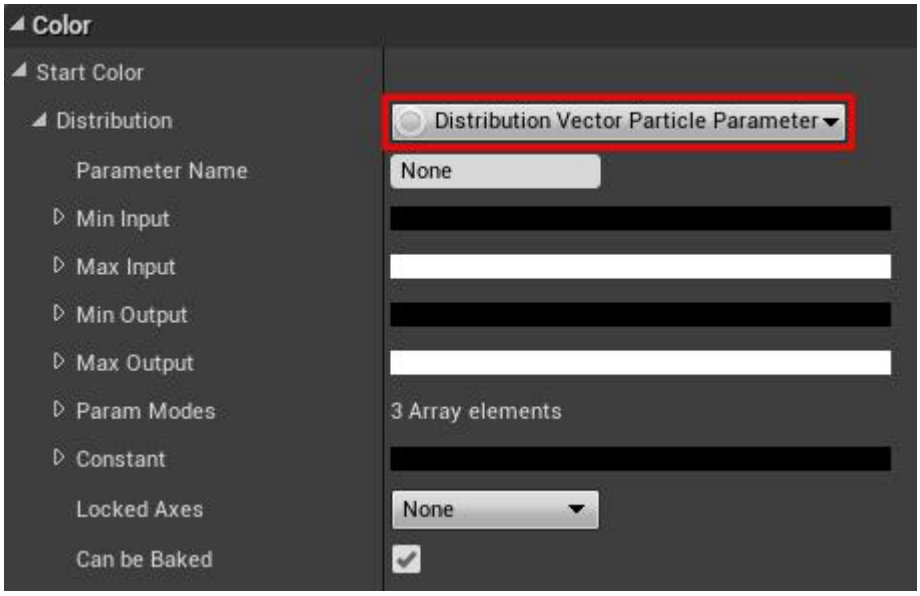
Нажмите на *Compile* и вернитесь в основной редактор. Нажмите на *Play* и начните расстреливать врагов.

Посмотрите, вот и взрывы! Теперь мы сделаем их интереснее, придав им тот же цвет, что и врагам.

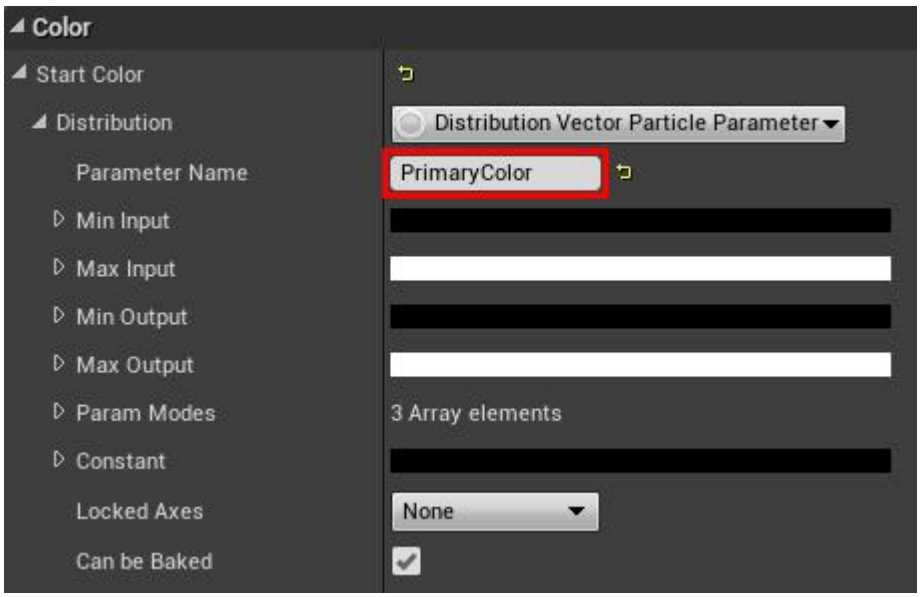
Замена цвета взрыва на цвет врага

Чтобы воспользоваться цветом, нам нужен способ получения такой информации из Blueprints. К счастью, в Cascade есть тип распределения, позволяющий это сделать.

Откройте *PS_Explosion* и выберите модуль *Initial Color*. Задайте *Start Color\Distribution* значение *Distribution Vector Particle Parameter*.



Это даст нам параметр, который мы сможем изменять с помощью Blueprints. Дайте *Parameter Name* название *PrimaryColor*



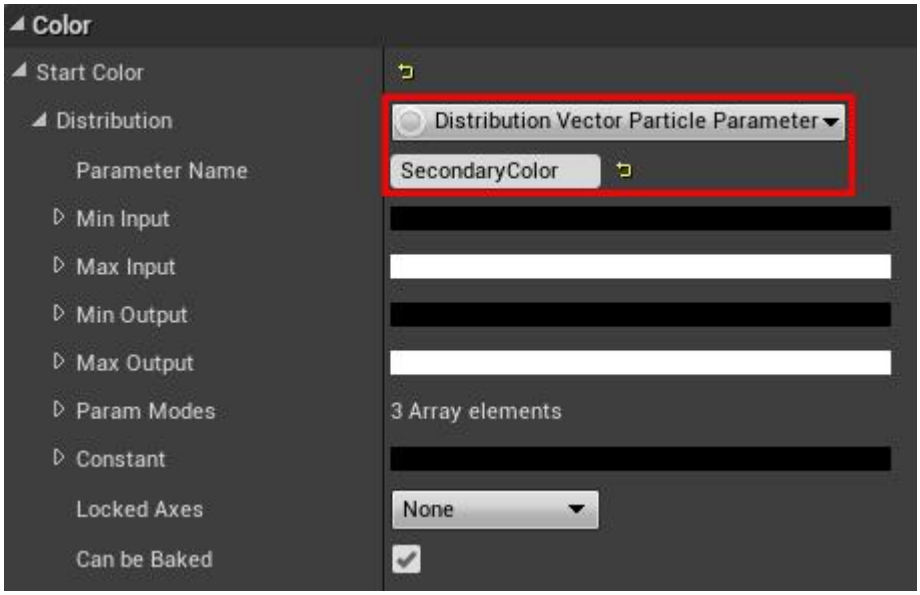
Для взрыва мы будем использовать оба цвета врага. Для использования второго цвета нам нужен ещё один эмиттер. *Нажмите правой клавишей мыши* на пустом пространстве в эмиттере и выберите *Emitter\Duplicate and Share Emitter*. Так мы дублируем эмиттер.

Вы заметите, что у каждого модуля теперь есть значок +. Благодаря использованию *Duplicate and Share Emitter* вместо *Duplicate*, мы связали модули, а не скопировали их. Все изменения, вносимые в один модуль, будут отражаться на том же модуле другого эмиттера. Это полезно, если мы хотим изменять свойства во всех эмиттерах, например, размер.

Единственный модуль, который нам нужно изменять — это *Initial Color*. Однако если мы внесём изменения, то они отразятся на обоих эмиттерах. В этом случае нам не нужно, чтобы модули были связаны, потому что им нужны отдельные названия параметров. Простейший способ отключения их связи заключается в удалении дублированного модуля *Initial Color* и создании нового.

Примечание: на момент написания статьи встроенных методов для разрыва связи между модулями не существует.

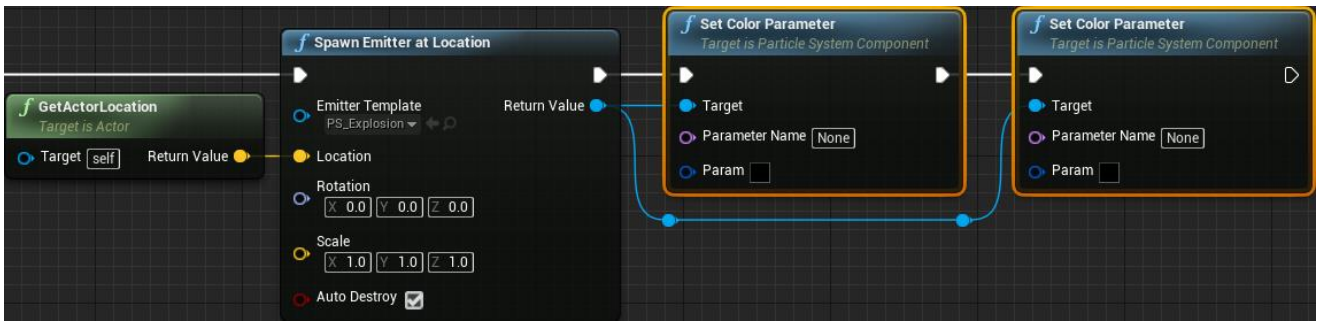
Выберите новый *Initial Color* и задайте *Start Color\Distribution* значение *Distribution Vector Particle Parameter*. Затем задайте *Parameter Name* значение *SecondaryColor*.



На этом этапе система частиц уже готова. Закройте *PS_Explosion*.
Далее нам нужно будет задавать параметры с помощью Blueprints.

Задание параметров частиц с помощью Blueprints

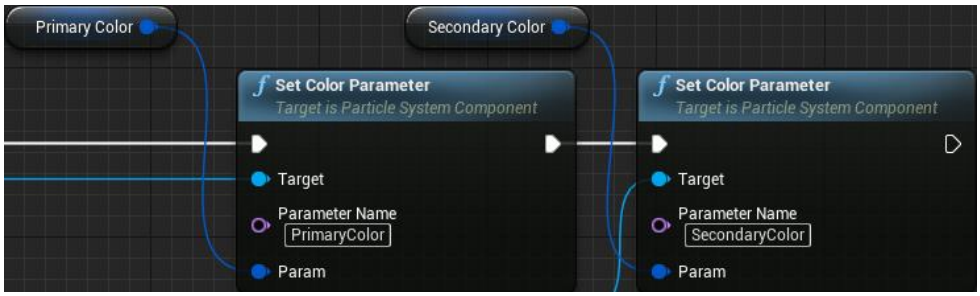
Откройте *BP_Enemy* и добавьте после *Spawn Emitter at Location* выделенные ноды:



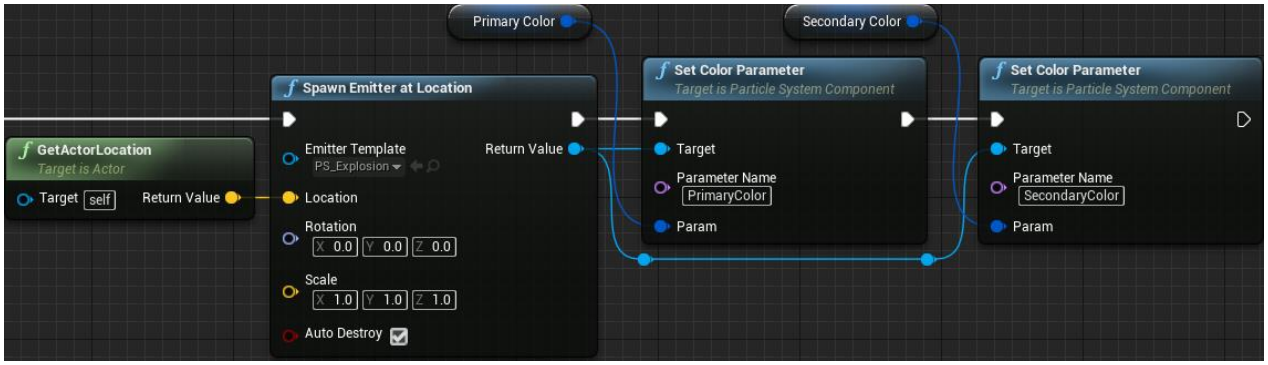
Это позволить изменять два параметра *PS_Explosion*.

Теперь нам нужно дать параметрам правильные названия. Задайте в качестве *Parameter Name* первого *Set Color Parameter* значение *PrimaryColor*. Задайте в качестве *Parameter Name* второго *Set Color Parameter* значение *SecondaryColor*

Наконец, нам нужно передать цвета. Чтобы упростить рабту, мы уже сохранили цвета в переменные *PrimaryColor* и *SecondaryColor*. Соедините каждую переменную с соответствующим нодом:



Вот, что у вас в итоге должно получиться:



Давайте разберёмся в событиях по порядку:

1. Когда враг умирает, он спаунит экземпляр *PS_Explosion* в точке своего местоположеня
2. Задаётся значение параметра *PrimaryColor PS_Explosion*
3. Задаётся значение параметра *SecondaryColor PS_Explosion*

Нажмите на *Compile* и закройте *BP_Enemy*. Нажмите на *Play* и начните расстреливать врагов, чтобы увидеть взрывы.