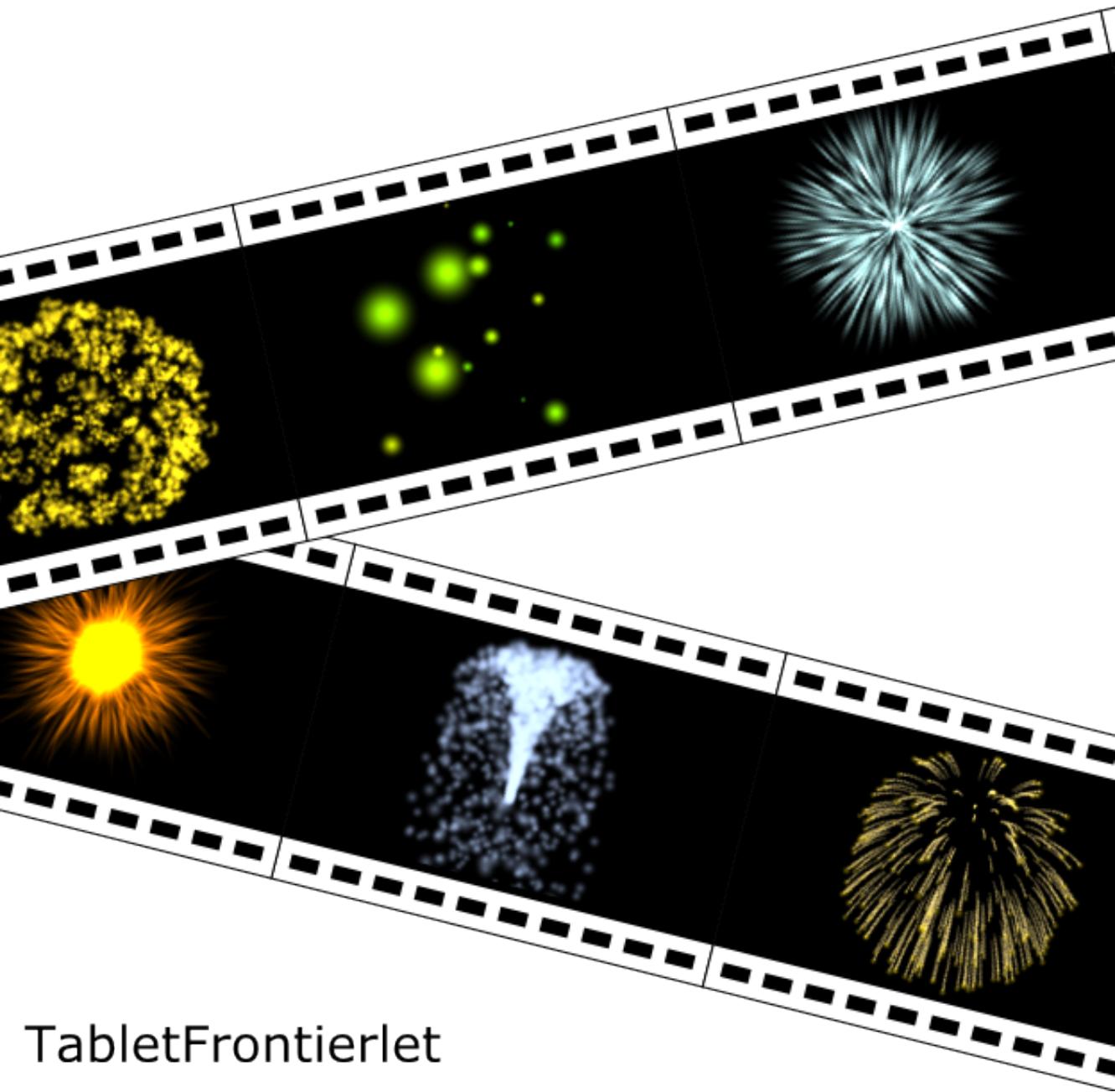


できる！ エフェクト for Unity



目次

はじめに.....	3
Particle System 分析編.....	4
Particle System で出来ること.....	5
Particle System のしくみ.....	6
各モジュールの解説.....	9
Initial.....	10
Emission.....	12
Renderer.....	13
Shape.....	16
Velocity over Lifetime.....	17
Limit Velocity over Lifetime.....	18
Force over Lifetime.....	18
Color over Lifetime.....	19
Color by Speed.....	19
Size over Lifetime.....	20
Size by Speed.....	20
Rotation over Lifetime.....	21
Rotation by Speed.....	21
Collision.....	22
Sub Emitters.....	25
Texture Sheet Animation.....	26
エフェクト実装編.....	28
チャージ・ワープ(収束エフェクト).....	29
蛍・爆発(多重描画による発光効果).....	31
たき火・ロウソク.....	34
煙.....	36
打ち上げ花火.....	38
滝・噴水.....	40
周回軌道、スパイラル(公転するエフェクト).....	43
クレジット.....	45

はじめに

Unity3.5 から「Shuriken」というパーティクルシステムが追加されましたが、既存のパーティクルシステムと比べて大きく仕様が変わり「よく分からぬから使ってない」という話を何度か聞きました。個人的にはとても便利になり、表現力も大きく上昇したため気に入っているのですが…。このような話を聞くたびに歯痒い思いをするので、今回はその魅力を伝えるために僭越ながら筆をとることにしました。

前置きはこれぐらいにしておいて、本書の構成の説明をば。本書は大きく分けて **Particle System 分析編** と **エフェクト実装編** に分かれています。

Particle System 分析編 では Shuriken の特徴や各モジュールの解説をしています。「仕組みが分からぬから使ってない」「あのモジュールのあのプロパティの意味が分からぬ」という方はこちらを見てもらうと解決するかもしれません。

エフェクト実装編 では実際に Shuriken を使って様々なエフェクトを作る方法を説明します。「どんなものが作れるか知りたい」という方はこちらを見てもらえると幸せになれるかもしれません。

また、**エフェクト実装編** で触れたエフェクトについてはサンプルプロジェクトを収録しています。「実際の動きを早く見たい!」という方はこちらもどうぞ。ちなみにエフェクトについては自由に使っていただいてかまいませんが、無断転載やそのまま再配布することは許可できませんのでよろしくお願ひいたします。

本書は「Version 3.5.4f1」での動作をもとに記述しています。異なるバージョンでは異なる動作をする、プロパティが変化している場合があるかもしれません、ご理解いただけるようお願ひいたします。また、あくまで独自研究をもとにしているため、記述内容に誤りが存在する可能性もありますが、その場合は申し訳ありません。よろしければそれを何らかの手段で教えていただけると大変助かります。

本書を読んでいただけた方の、何らかの役に立つことができれば幸いです。

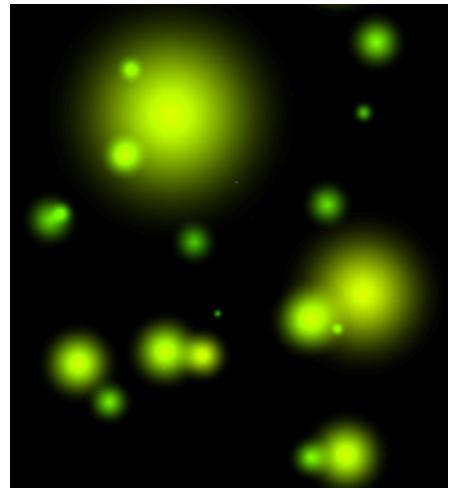
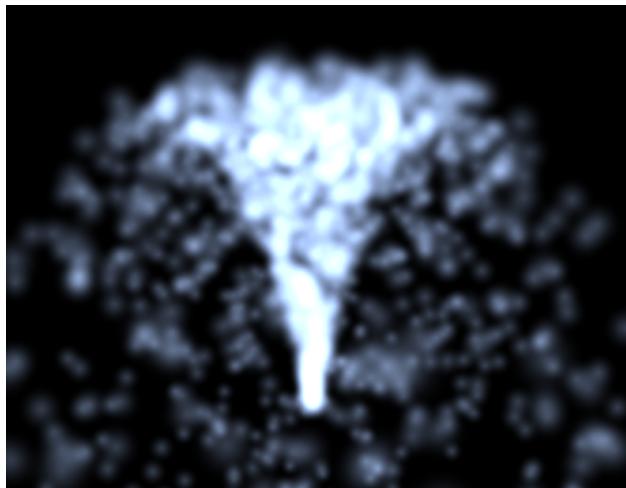
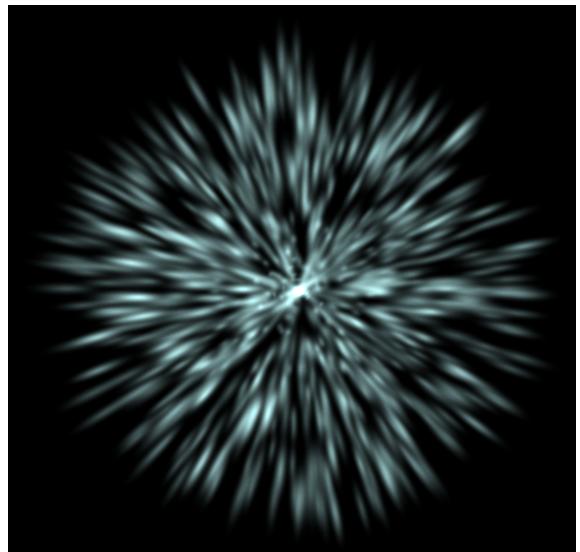
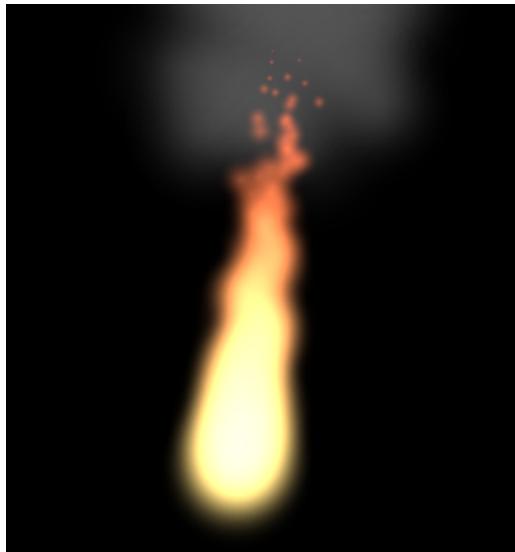
Tablet Frontierlet
よもぎ

※本書の一部または全部を無断で転載、複製、再配布することを固く禁じます。

Particle System 分析編

Particle System で出来ること

Particle System(以下システム)はパーティクル(粒子)を生成・噴射するコンポーネントです。パーティクルの噴射形式や噴射後のパーティクルの振る舞いを設定することで、様々な視覚エフェクトを表現することができます。



※各エフェクトの具体的なパラメータ設定例についてはエフェクト実装編(P.28)で説明します。

Particle System のしくみ

- 複数の噴射形式

パーティクルは、(1)1つずつ連続で噴射させる[図 1]以外に、(2)一度に複数のパーティクルを噴射させたり[図 2]、(3)システムの移動量に応じてパーティクルを噴射させる[図 3]ことができます。また、(1)と(2)を併用することもできます。

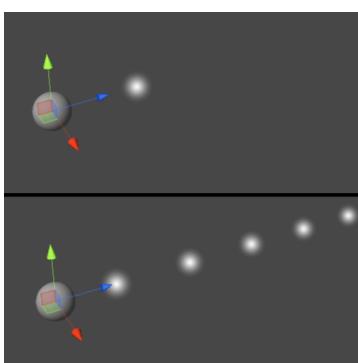


図 1:連続噴射

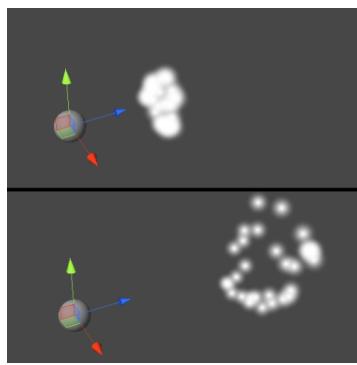


図 2:一斉噴射

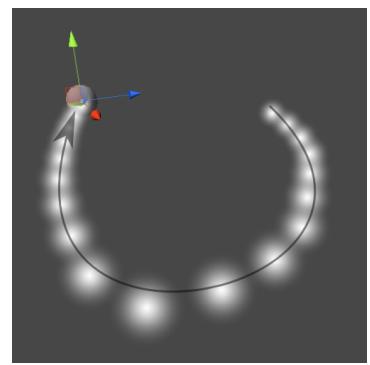


図 3:移動に応じて噴射

※システムの位置がわかりやすいように球を表示しています。

- パーティクルの形状

パーティクルは、任意のテクスチャを貼り付けたビルボード[図 4]もしくはポリゴンモデル[図 5]から選択することができます。ビルボードは常に特定の方向を向く(一般的にはカメラ方向)板状ポリゴンですが、設定によって正面を向ける方向や形状を変化させることができます。さらに、パーティクルを回転させたり大きさを動的に変化させることもできます。

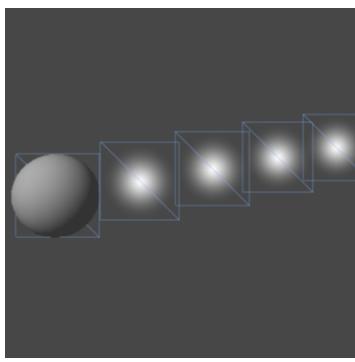


図 4:ビルボード

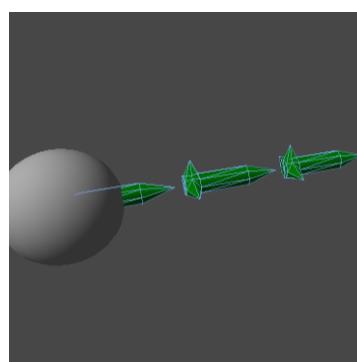


図 5:ポリゴンモデル

※デバッグ用にワイヤーフレームを表示しています。

- モジュールによるシステムやパーティクルの設定

システムは複数のモジュールによって構成されています。噴射速度などのシステムそのものの設定のみでなく、パーティクルの色や大きさを動的に変化させたり、他のシステムとの連携を設定するモジュールも存在します。

システムのインスペクタを図 6 に示します。モジュール名のクリックでインスペクタの折り畳み、左の白丸をクリックすることでモジュールのオンオフを変更できます。また、赤枠の部分はモジュール名が表示されませんが、公式マニュアルによると「Initial」という名前が付いているようです。

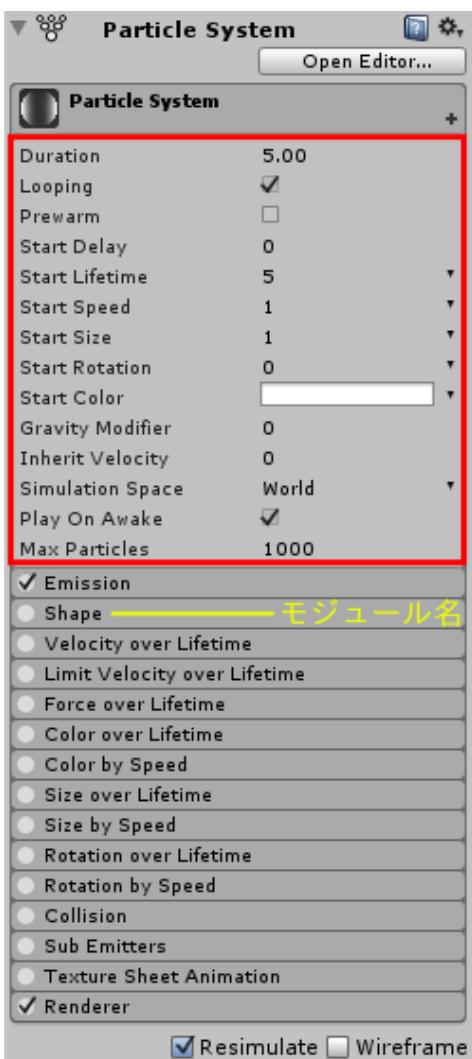


図 6:システムインスペクタ

右端に▼がついているプロパティは値を複数のスタイルで入力することができます[図 7]。例えば数値で指定するプロパティであれば、

- Constant(定数)
- Curve(曲線による関数定義)
- Random Between Two Constants
(範囲を指定した乱数)
- Random Between Two Curves
(乱数の振れ幅を曲線で定義)

といった選択肢から好きなスタイルを選んで値を入力することができます。

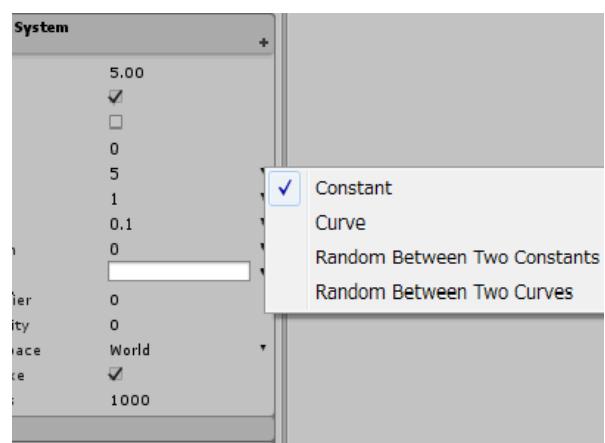


図 7:入力スタイルの選択

- Curve Editor や Gradient Editor を使った入力

Curve や Random Between Two Curves を選択した場合は、インスペクタの下に表示される Curve Editor で曲線を設定します。縦軸、横軸の単位はプロパティによって違いますが、縦軸の値の最大値はこの Editor 上で変更できます。

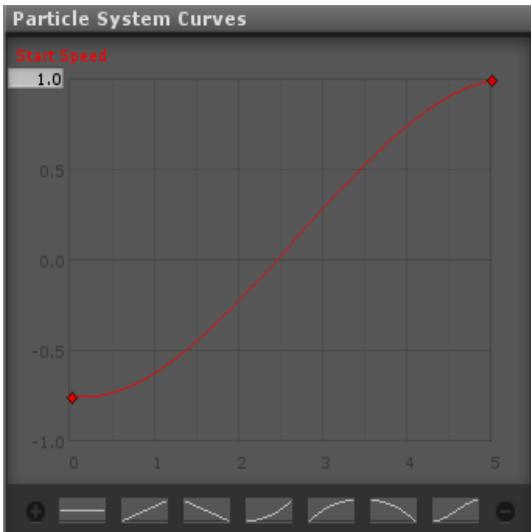


図 8:Curve を選択した場合

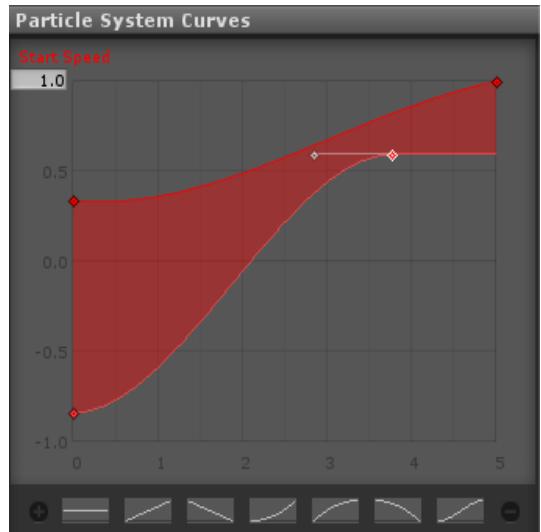


図 9:Random ~ を選択した場合は
2つの曲線で乱数の範囲を設定

色を指定するプロパティでは Gradient(グラデーション)や Random Between Two Gradient(グラデーションの乱数版)を選択できますが、グラデーションの設定にはプロパティをクリックすると表示される Gradient Editor を使います。上端と下端にそれぞれ不透明度と色を指定するマーカを配置し、好みのグラデーションを作ることができます。

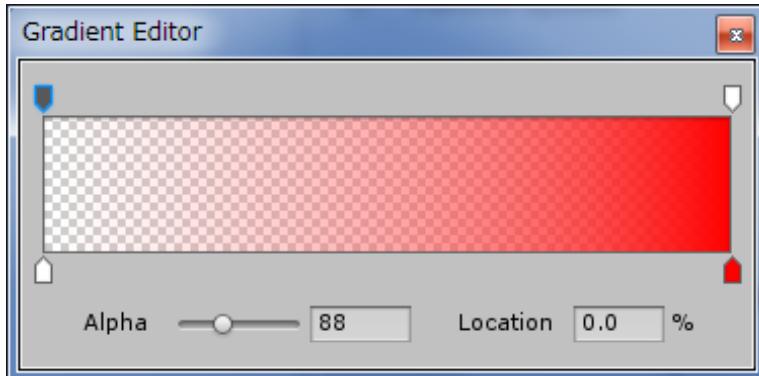


図 10:Gradient Editor によるグラデーションの設定

各モジュールの解説

ここでは各モジュール毎にプロパティや設定のコツを解説します。

モジュール一覧(解説の都合上、エディタ上での順番とは異なります)

モジュール名	機能
Initial	パーティクルの速度や色の初期値や物理的性質を設定
Emission	パーティクルの噴射形式と噴射量を設定
Renderer	パーティクルの形状を設定
Shape	システムの噴射口の形状、噴射方向を設定
Velocity over Lifetime	経過時間に応じてパーティクルの速度変化させる
Limit Velocity over Lifetime	経過時間に応じてパーティクルの速度を制限する
Force over Lifetime	経過時間に応じてパーティクルに外力を加える
Color over Lifetime	経過時間に応じてパーティクルの色を変化させる
Color by Speed	速度に応じてパーティクルの色の変化させる
Size over Lifetime	経過時間に応じてパーティクルの大きさを変化させる
Size by Speed	速度に応じてパーティクルの大きさを変化させる
Rotation over Lifetime	経過時間に応じてパーティクルの回転速度を変化させる
Rotation by Speed	経過時間に応じてパーティクルの回転速度を変化させる
Collision	パーティクルと平面の衝突の設定
Sub Emitters	他の Particle System との連携の設定
Texture Sheet Animation	テクスチャを用いたアニメーションの設定

プロパティの値について

複数のスタイルで設定可能なプロパティについては「プロパティ一覧」の中で以下の記号を使うことにより使用可能なスタイルを記述しています。

記号 意味	記号 意味
Co 定数を指定	Col 色を指定
Cu 曲線による指定	Gra グラデーションによる色の変化の指定
RCo 範囲を指定した乱数	RCol 色の範囲を指定したランダムな色
RCu 曲線により範囲を指定した乱数	RGra ランダムな色のグラデーション版

Initial

パーティクルの速度や色の初期値や物理的性質を設定

■ プロパティ一覧

プロパティ名 型[単位]	内容
Duration 定数[s]	パーティクルを噴射する時間の長さ
Looping True/False	噴射をループさせる
Prewarm True/False	時刻 0 でパーティクルを噴射済みにする ※Looping が有効であるときのみ設定可能
Start Delay 定数[s]	噴射を指示してから実際にパーティクルの噴射が開始されるまでの時間 ※Prewarm が無効であるときのみ有効
Start Lifetime Co,Cu,RCo,RCu[s]	パーティクルの寿命 パーティクルは発生後、この時間が経過すると消える
Start Speed Co,Cu,RCo,RCu[m/s]	パーティクルの速度 大きさのみで、向きは Shape で設定する
Start Size Co,Cu,RCo,RCu[scale]	パーティクルのサイズ
Start Rotation Co,Cu,RCo,RCu[deg]	パーティクルの傾き 回転軸は形状(Rendererで設定)によって異なる
Start Color Col,Gra,RCol,RGra	パーティクルの色
Gravity Modifier 定数[scale]	パーティクルへの重力の影響度(=質量[g]) 負値を設定することも可能
Inherit Velocity 定数[scale]	パーティクルへのシステムの速度の影響度(=質量[g]) 負値を設定することも可能
Simulation Space	噴射後のパーティクルの位置とシステムに依存関係 Local(依存),World(非依存)から選択[図 11]
Play On Awake True/False	自動的に噴射を開始する
Max Particles 定数	一度に噴射できるパーティクルの上限

■ 注意点やメモ

- 他のモジュールとの関係

実際のパーティクルの色は、**Start Color** の色に、他のモジュール(**Color over Lifetime** や **Color by Speed**)の色を乗算した色になる。

Start Size は **Size over Lifetime** や **Size by Speed** のプロパティの基準値になる。

- Simulation Space** は Play モード時のみ有効

Simulation Space の設定は Play モード(ゲームを操作できる状態)でのみ有効で、そうでないときは Local と同様の状態になる。よって、効果を確認する為には Play モードで確認する必要がある。

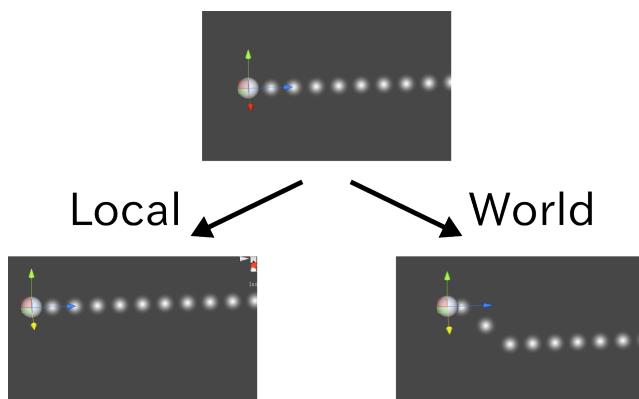


図 11:Simulation Space による違い。

Local にすると、システムの移動と同時にパーティクルも移動する。

- Inherit Velocity** は **Simulation Space** が **World** であるときのみ有効

Simulation Space が **Local** であるとき、システムが移動するとパーティクルも同時に移動する。このとき、**Inherit Velocity** の値は無視される。よって **Inherit Velocity** を使うときは **Simulation Space** を **World** にする必要があり、また前述の理由により、Play モードでないと効果を確認できない。

- Max Particle** の値による影響

シーンに存在するパーティクルが **Max Particle** を超えると、システムは一時的に噴射を止める。パーティクルが消滅し、シーン中のパーティクルの数が減ると噴射は再開される。

Emission

パーティクルの速度や色の初期値や物理的性質を設定

■ プロパティ一覧

プロパティ名 型[単位]	内容
Rate Co,Cu[個/s or 個/m]	<p>基準に Time を選択した場合 1 秒当たりのパーティクルの噴射量</p> <p>基準に Distance を選択した場合 1m 当たりのパーティクルの噴射量</p>
	噴射の基準を Time と Distance から選択する
Burst	<p>※基準に Time を選択した場合のみ有効 一斉噴射の設定 噴射する時刻と噴射する量を設定する</p>

■ 注意点やメモ

- Distance にする場合は **Simulation Space** を World にする必要がある
Simulation Space を World にしておかないと、Distance での噴射はされません。しかし、この場合は Play モードでなくても確認可能です。
- Rate の Curve Editor の横軸は **Duration** に対するスケール
Rate の設定に Curve を使う場合、横軸は Initial の Duration に対応しています。つまり、0~1 が噴射の開始～終了に対応しています。ループをする場合、1 ループ中での変化を設定することになります。

Renderer

パーティクルの速度や色の初期値や物理的性質を設定

■ プロパティ一覧

プロパティ名 型[単位]	内容
Render Mode	<p>パーティクルの形状</p> <ul style="list-style-type: none"> • Billboard 常にカメラに正面を向ける正方形のポリゴン • Stretched Billboard 辺の長さを変更可能なビルボード • Horizontal Billboard 常にWorld座標のY軸方向に正面を向けるビルボード • Vertical Billboard XZ平面内でカメラに正面を向けるビルボード • Mesh ポリゴンモデル
Material True/False	使用する Material
Sort Mode	<p>パーティクルの描画順序</p> <p>None(指定なし)</p> <p>By Distance(距離に依存)</p> <p>Youngest First(新しいパーティクルから描画)</p> <p>Oldest First(古いパーティクルから描画)</p>
Sorting Fudge	システム間の描画順序の指定 値が小さいシステムのパーティクルから順番に描画される
Cast Shadows True/False	パーティクルが影を落とすようになる(Unity Proのみ)
Receive Shadows True/False	パーティクルに影が落ちるようになる(Unity Proのみ)
Max Particle Size 定数[scale]	<p>パーティクルの大きさの制限</p> <p>スクリーンの大きさに対する割合を指定 (例:0.5=スクリーンの半分)</p>

■ 注意点やメモ

- モジュールを無効にしても描画がされないだけでパーティクルの生成はされる
このモジュールを無効にするとパーティクルは描画されなくなりますが、移動や生成/死亡/衝突の判定は行われ、これを利用したテクニックもあります。パーティクルの噴射そのものを止めたい場合には **Emission** を無効にします。
- Render Mode と回転の関係
Initial の **Start Rotation** や **Rotation over Lifetime** の **Rotation by Speed** によって設定する回転の回転軸は **Render Mode** によって違います
 - 各種 Billboard の場合
ビルボードの正面が回転軸です。また、**Stretched Billboard** は回転しません。
 - Mesh の場合
ポリゴンモデルの場合、さらに **Shape** の設定により軸が変わります。一部の場合を除いて、生成時のパーティクルの回転、およびその回転軸はほぼランダムになります。よって、パーティクルに指定するモデルはなるべく点対称なもの(向きが存在しないもの)を選択することが無難です。
- Stretched Billboard の補足
Stretched Billboard は短辺と長辺、パーティクルの速度方向に長辺を向けています。短辺に対する長辺の長さは、以下の 3 つのプロパティで設定します。
 - Camera Scale
カメラの速度の影響度。0 で影響なし。
 - Speed Scale
パーティクル自身の速度の影響度。正の値で長くなり、負の値で短くなる。
 - Length Scale
短辺に対する長辺の長さ。1 で正方形になる。

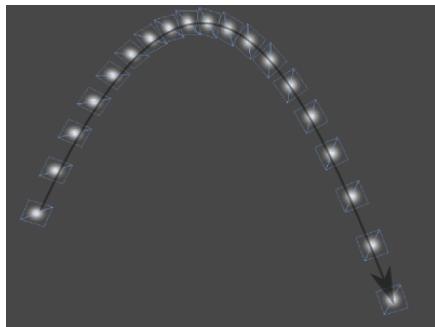


図 13:Length Scale=1 のみ

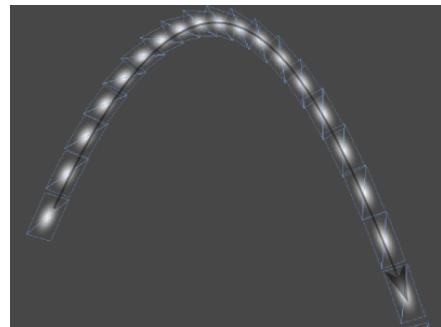


図 12:Speed Scale を設定

- **Material の Shader** は **Particles** から選択するのが無難

Material は「Default-Particle」のように、Particles に属するシェーダを使っているものを選択した方が無難です。そうでない場合、システム上で色を変更することが不可能になり(Material そのものの色を変更する必要)、不便です。

Shape

システムの噴射口の形状、噴射方向を設定

■ プロパティ一覧

プロパティ名 型[単位]	内容
Shape	噴射口の形状 Sphere(球)、Hemisphere(半球)、Cone(円錐)、Box(直方体)、Mesh(ポリゴンモデル)から1つ選択します。 Meshではさらに、Vertex(頂点)、Edge(辺)、Triangle(面)のどこから噴射するかを選択します。
Radius,Angle など	噴射口の各部位の大きさ
Random Direction	パーティクルを噴射する方向をランダムにする ※Coneでは無効
Emit from Shell True/False	パーティクルを表面からのみ噴射 無効であるときは、球(半球)の領域全てから噴射 ※Sphere、Hemisphereのみ

■ 注意点やメモ

• モジュールが無効である場合

このモジュールが無効であるとき、システムの噴射口は点になり、システムの存在する座標からシステムの正面(Z軸方向)に向けて噴射します。

• 噴射方向について

パーティクルの噴射方向は噴射口の形状に依存します。

- Sphere, Hemisphere, Mesh の場合

噴射口の法線方向に噴射します。

- Cone, Box の場合

システムの正面(Z軸方向)に噴射します。Coneの場合、Angleの範囲内でランダムになります。

Velocity over Lifetime

経過時間に応じてパーティクルの速度変化させる

■ プロパティ一覧

プロパティ名 型[単位]	内容
X,Y,Z Co,Cu,RCo,RCu[m/s]	パーティクルの速度
Space	X,Y,Z で設定した値の座標系 Local(システムの座標系)、World(ワールド座標系)から 選択

■ 注意点やメモ

- **Curve Editor** の横軸はパーティクルの寿命に対応

このモジュールに限らず「～over Lifetime」という名前のモジュールで曲線による設定を行う場合、その横軸はパーティクルの一生に対応します。つまり 0 のときにパーティクルが生まれた瞬間、1 のときが消滅する瞬間です。

- **Force over Lifetime**との使い分け

徐々に加速するパーティクルを作りたいときは、**Force over Lifetime**を使うと便利です。逆に、急に進行方向を変えるようなパーティクルにはこのモジュールが向いています。

Limit Velocity over Lifetime

経過時間に応じてパーティクルの速度を制限する

■ プロパティ一覧

プロパティ名 型[単位]	内容
Separate Axis True/False	速度制限を成分ごとに行う 無効である場合、速度方向に対して速度制限を行う
Speed or X,Y,Z Co,Cu,RCo,RCu[m/s]	速度制限の閾値
Dampen 定数	超過速度の減衰率

■ 注意点やメモ

- 閾値を超えたときの挙動

パーティクルの速度が閾値を超えた場合、以下の式で結果を算出する模様。

(処理後の速度) = (粒子の速度) - (粒子の速度 - **Speed**)×**Dampen**

つまり、**Dampen=1** の時、パーティクルの最高速度は **Speed** になります。

Force over Lifetime

経過時間に応じてパーティクルに外力を加える

■ プロパティ一覧

プロパティ名 型[単位]	内容
X,Y,Z Co,Cu,RCo,RCu[m/s]	パーティクルにかける外力の各成分
Space	X,Y,Z で設定した値の座標系 Local(システムの座標系)、World(ワールド座標系)から選択
Randomize True/False	外力を角フレーム毎にランダムにする 無効であるとき、パーティクル毎にかかる外力はランダムであるが、1つのパーティクルにかかる力は一定 ※X,Y,Z で「Random~」を選択している時のみ有効

Color over Lifetime

経過時間に応じてパーティクルの色を変化させる

■ プロパティ一覧

プロパティ名 型[単位]	内容
Color Gra,RGra	寿命に対するパーティクルの色の変化

■ 注意点やメモ

- グラデーションの横軸はパーティクルの寿命に対応
- パーティクルの最終的な色は、各モジュールの算出した色を乗算した色です

Color by Speed

速度に応じてパーティクルの色の変化させる

■ プロパティ一覧

プロパティ名 型[単位]	内容
Color Gra,RGra	速度に対するパーティクルの色の変化
Speed Range 定数[m/s]×2	グラデーションを適応する速度の範囲 最小値と最大値を設定する

■ 注意点やメモ

• **Speed Range** の補足

パーティクルの速度が **Speed Range** の範囲外の場合は、最小値もしくは最大値の色が適応されます。

Size over Lifetime

経過時間に応じてパーティクルの大きさを変化させる

■ プロパティ一覧

プロパティ名 型[単位]	内容
Size Cu,RCo,RCu[scale]	寿命に対するパーティクルの大きさの変化

■ 注意点やメモ

- **Size** は **Start Size** に対するスケール

Curve Editor の縦軸の値は **Initial** の **Start Size** の大きさを 1 とした倍率です。つまり、**Size=2** であるとき **Start Size** の 2 倍の大きさになります。

Size by Speed

速度に応じてパーティクルの大きさを変化させる

■ プロパティ一覧

プロパティ名 型[単位]	内容
Size Cu,RCo,RCu[scale]	速度に対するパーティクルの大きさの変化
Speed Range 定数[m/s]×2	変化を適応する速度の範囲 最小値と最大値を設定する

■ 注意点やメモ

- **Color by Speed** の大きさ版です

Rotation over Lifetime

経過時間に応じてパーティクルの回転速度を変化させる

■ プロパティ一覧

プロパティ名 型[単位]	内容
Angular Velocity Co,Cu,RCo,RCu[deg]	パーティクルの回転速度

■ 注意点やメモ

- 他の「～ over Lifetime」と違い、定数を設定できます

初期回転速度を設定するプロパティが存在しない為だと思われます。

- 回転軸は **Renderer** の設定に依存

Rotation by Speed

経過時間に応じてパーティクルの回転速度を変化させる

■ プロパティ一覧

プロパティ名 型[単位]	内容
Angular Velocity Co,Cu,RCo,RCu[deg]	速度に対するパーティクルの回転速度
Speed Range 定数[m/s]×2	変化を適応する速度の範囲 最小値と最大値を設定する

■ 注意点やメモ

- 他の「～ by Speed」との違い

このモジュールの **Angular Velocity** では定数値を指定できます。つまり
「速度がある範囲に含まれている場合のみ回転させる」ということができます。

- 回転軸は **Renderer** の設定に依存します

Collision

パーティクルと平面の衝突の設定

■ プロパティ一覧

プロパティ名	型[単位]	内容
Plane		衝突判定を持たせる平面の座標系(複数設定可能)
Dampen	定数	衝突によりパーティクルが失う速度の大きさ
Bounce	定数	衝突後のパーティクルの、平面の法線方向成分の大きさ
Lifetime Loss	定数	衝突によるパーティクルの寿命への影響度
Min Kill Speed	定数[m/s]	この速度より小さいパーティクルは衝突時に消滅する。
Visualization		シーンエディタ上での衝突判定の見た目 Grid[図 14],Solid[図 15]から選択
Scale Plane	定数	シーンエディタ上での平面の見た目の大きさ

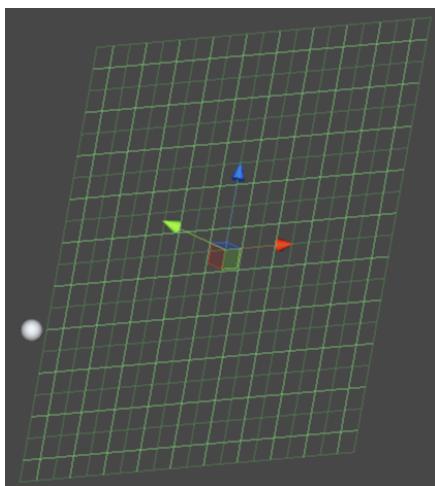


図 14:Visualization=Grid

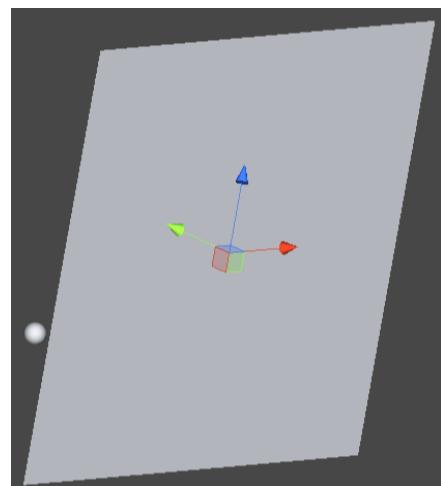


図 15:Visualization=Solid

■ 注意点やメモ

- 衝突判定について

Plane に既存の Game Object の座標系または新たに生成した座標系を割り当てるにより、パーティクルとの衝突判定を設定することができます。衝突判定は指定した座標系で定義される XZ 平面であり、衝突後のパーティクルは基本的に反射します。(衝突後の振る舞いは他のプロパティに依存します)
また、座標系の Y 軸正方向からの衝突を前提としており、平面の表と裏ではパーティクルの挙動が微妙に変化します。

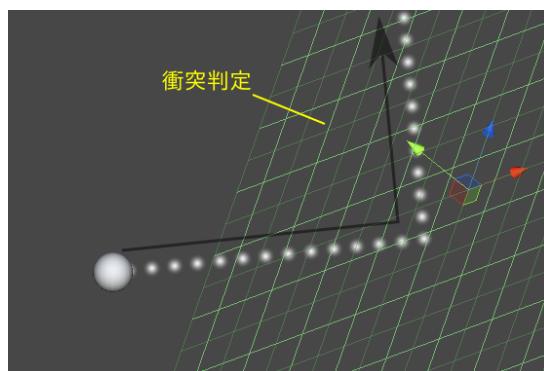


図 16:パーティクルの衝突

- 衝突判定は無限平面

Scale Plane で設定する平面の大きさはあくまで見た目上の大きさであり、実際の判定は描画される平面の外にも存在します。[図 17]

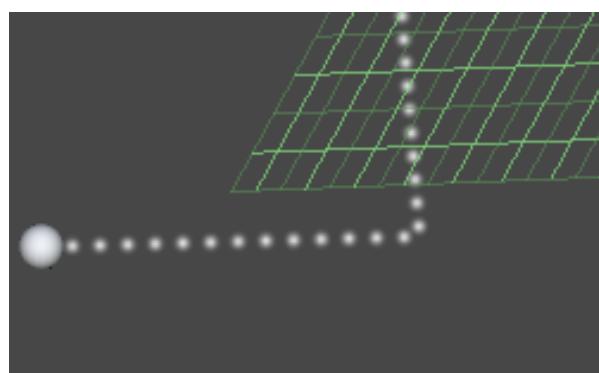


図 17:衝突判定は無限平面

- **Dampen の補足**

衝突後にパーティクルから奪う速度の大きさを示します。

$$(衝突後の速度) = (1-\text{Dampen}) \times (\text{衝突前の速度})$$

0~1 の範囲では単純な減速、1~2 の範囲では逆方向への加速になります。

0~1 の値を指定することで壁のような[図 18]、1~2 の値を指定することでレンズのような挙動[図 19]をパーティクルに対して与えることができます。

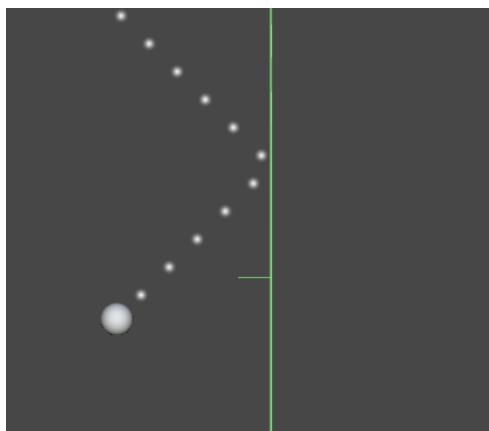


図 18;Bounce=1,Dampen=0

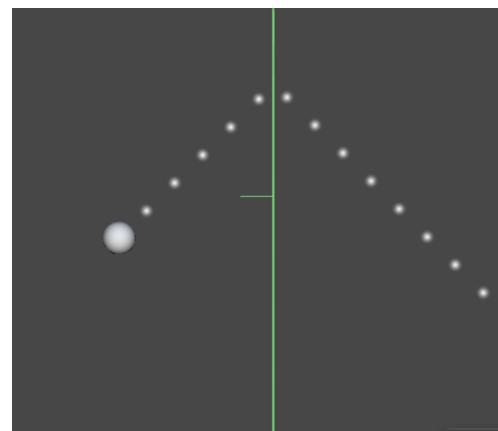


図 19:Bounce=1.Dampen=2

- **Bounce の補足**

パーティクルの速度の、「平面の法線方向成分」の速度に影響します。

Bounce=1 の時、衝突後と衝突前の法線方向の速度の大きさが等しくなり、

Bounce<1 では大きさが減り、**Bounce>1** では大きさが増えます。速度の全ての成分に影響する **Dampen** と違い、平面の法線方向成分のみ作用します。

- **Lifetime Loss の補足**

衝突時、パーティクルの残りの寿命は減らされます。そのとき減らされる時間は **Lifetime Loss** と **Start Lifetime** に依存します。

$$(衝突後の残り寿命) = (\text{衝突前の寿命}) - \text{Start Lifetime} \times \text{Lifetime Loss}$$

例えば、**Lifetime Loss=0** の場合、衝突で寿命は減らず、**Lifetime Loss=1** の場合、衝突時によって必ず消滅します。衝突により寿命が減ると、その分「~over Life Time」系のモジュールによる変化が適応されます。これを利⽤することによって、衝突時に大きさや色を変化させることもできます。

Sub Emitters

他の Particle System との連携の設定

■ プロパティ一覧

プロパティ名 型[単位]	内容
Birth	パーティクル生成時に起動するシステム
Death	パーティクル消滅時に起動するシステム
Collision	パーティクルが衝突したときに起動するシステム ※Collision の設定が必要

■ 注意点やメモ

- **Birth** の補足

Birth にシステムを設定すると、各パーティクルにそのシステムの噴射口にすることができます。

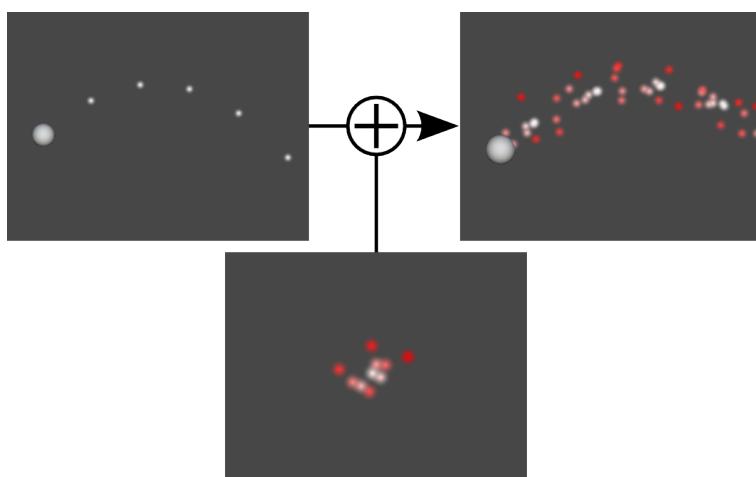


図 20:Sub Emitter - Birth

指定したサブシステムは普通のシステムとほぼ同様の挙動示しますが、以下の特徴があります。

- **Burst** による噴射はループしない(一度きり)
- **Prewarm** は無効
- 設定に関わらず **Simulation Space=World** 同様の挙動を示す。

- **Death** の補足

Death にシステムを設定すると、パーティクルが消滅した際に、消滅した場所でシステムに噴射をさせることができます。ただし、有効なのは **Burst** による一斉噴射のみで、**Burst** の Time=0.00 の噴射のみ有効です。

- **Collision** 補足

衝突時にシステムに噴射させることができます。**Death** と同様に、**Burst** による噴射のみ有効です。

Texture Sheet Animation

テクスチャを用いたアニメーションの設定

■ プロパティ一覧

プロパティ名 型[単位]	内容
Tile X, Y 定数×2	テクスチャに含まれる行と列の数
Animation	どちらか一つを選択する Whole Sheet テクスチャ全体を 1 つのアニメーションパターンとする Single Row 1 行を 1 つのアニメーションパターンとする
Random Row True/False	アニメーションをパーティクル毎にランダムにする ※Single Row のときのみ有効
Row 定数	アニメーションに用いる行 ※Single Row のときのみ有効
Frame over Time Co,Cu,RCo,RCu	実際に表示するフレームのインデックス
Cycle 定数	アニメーションのリピート回数

■ 注意点やメモ

- アニメーションパターンの指定方法

まず **Tile** でテクスチャを縦と横にそれぞれ何分割するかを指定します。そして、**Animation** で、アニメーションパターンの区切りを指定します。複数のアニメーションパターンを 1 枚のテクスチャに収める場合は、アニメーションパターンのフレーム数を統一し、1 つのアニメーションパターンを左から右に配置します。フレームや行は、左から右に、上から下に順番にインデックスがつきます。**X=Y=2**、**Animation=Whole Sheet** の場合は図 21 の順番にフレームが並んでいるとみなされます。

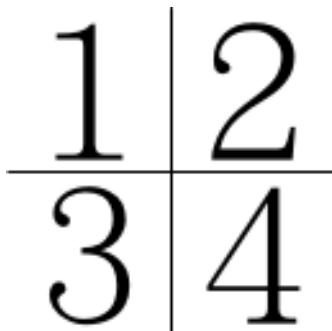


図 21

$X=Y=2$, Whole Sheet

- フレームの遷移について

Frame over Time では、1 ループのアニメーションでのフレームの遷移を指定します。Curve Editor の横軸はループの開始から終了までに対応し、パーティクルの一生で何回アニメーションをループするかは **Cycles** で指定します。

- アニメーション以外の活用法

Frame over Time に定数(Constant)を指定すると、パーティクルはアニメーションしません。この時、**Animation=Single Row** とすると、**Row** と **Frame over Time** で任意の部分を選択することができます。これを利用して、複数のパーティクルを 1 つのマテリアルで取り扱うことができます。

エフェクト実装編

Particle System の機能を活用したエフェクトの設定例とポイントを解説していきます。

同梱の設定済みサンプルエフェクトを実際に触りながら読むと各パラメータの効果を実感できます。

チャージ・ワープ(収束エフェクト)

Legacy Particle System と比較した Shuriken の大きな利点の 1 つは「パーティクルを一点に収束させることができる」ことだと思います。発散するエフェクトは従来のシステムでも作れましたが、収束するエフェクトは困難でした。ここでは、サンプルエフェクト「Charge」[図 22]の説明を通してそういったエフェクトの作り方を解説します。

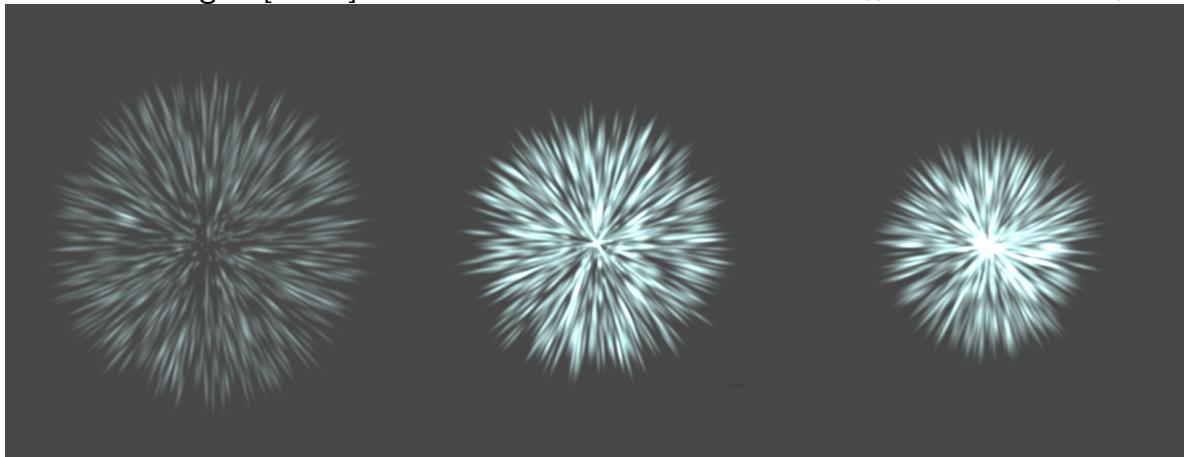


図 22:サンプルエフェクト「Charge」

1. 一点にパーティクルを収束させる

Shape を Sphere、Radius=5 に、**Initial** の **Start Speed** = -1 にします。これだけで、パーティクルが球の中心へと収束します。さらに **Shape** の **Emit from Shell** を有効にすると、パーティクルの発生点と焦点の距離を一定にすることができます。

2. 生成タイミングの変更

Emission で、**Rate**=0 に **Burst** で **Time**=0.00、**Particle**=100 とすることで、一度にパーティクルが生成されるようにします。

3. エフェクトの大きさの調整

Shape の **Radius** でエフェクトの半径を好みの大きさに調整しつつ、**Initial** の **Start Speed** や **Start Lifetime** を設定して、中心でパーティクルが消滅するようにします。

4. パーティクルの形状を調整

円形のパーティクルでも悪くはありませんが、収束感を出すために細長くします。

Renderer の Render Mode を Stretched Billboard にし、Speed Scale と Length Scale を適当な値に設定します。値が大きいほどパーティクルが細長くなります。

5. フェードイン/アウトで発生と消滅を滑らかにする

現在、パッとパーティクルが現れてパッと消滅し、嘘臭さがあります。そこで、フェードイン/アウトするようにします。Color over Lifetime で図 23 のように、Alpha を両端を 0 に、中心を 255 にします。これでフェードイン/アウトします。

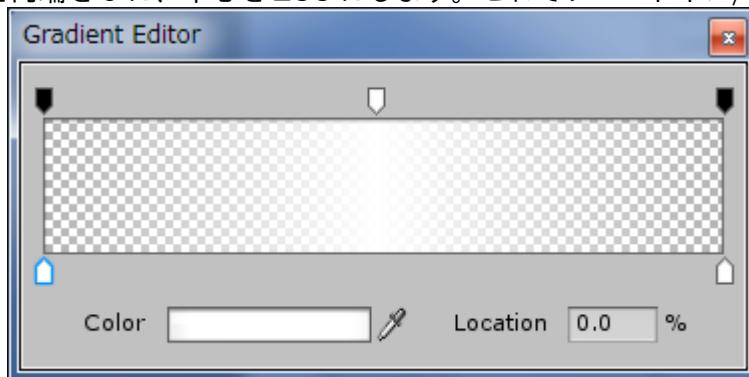


図 23:不透明度の変化でフェードイン/アウト

6. 色味の調整

最後に、Initial の Start Color で全体の色を調整します。好みの色を選択しましょう。発光するタイプのエフェクトは Alpha を 50%ぐらいにするとそれっぽくなります。

✓ 応用アイデア

- 今回は Burst で噴射していますが、Rate のみの連続噴射にするとまた違ったエフェクトになります。(サンプルエフェクト「Warp」)
- 他にも Size over Lifetime や Color over Lifetime で色や大きさを変化させたり、テクスチャを変更して形状を変えると色々なバリエーションができます。

蛍・爆発(多重描画による発光効果)

プリセットされている「Default-Particle」は全体的にぼやけていて、そのままでは強い発光を表現するのにはあまり向きません[図 24]。しかし新たにテクスチャを用意するのはなかなか大変です。今回は「Default-Particle」を使いつつ、しかし強い発光[図 25]を実現する方法を紹介します。

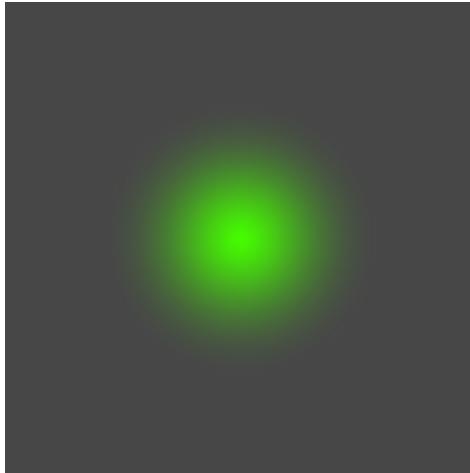


図 24:Default-Particle
そのままでは強い発光には
あまり向かない

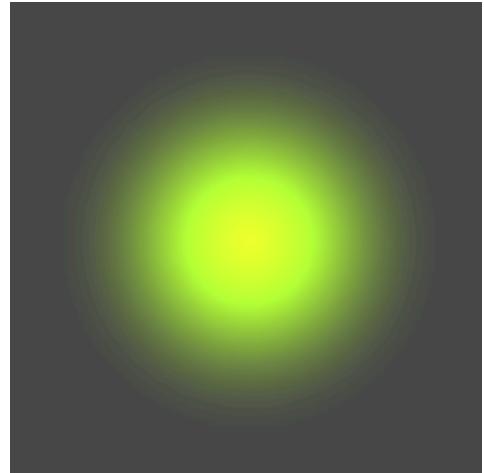


図 25:同じく Default-Particle
ちょっとしたテクニックで
発光感を強くできる。

1. 多重描画

手品の種はとても簡単です。「同じ座標に複数枚描画する」これだけです。

Shape を無効にして **Initial** の **Start Speed** を 0 にすると、全てのパーティクルがシステムの原点に描画されます。**Emission** の **Rate** を 1 にすると、元々のテクスチャよりも程よく明るくなります。例として、図 26 と図 27 は描画回数以外は全く同じ設定ですが、明らかに見た目が違います。

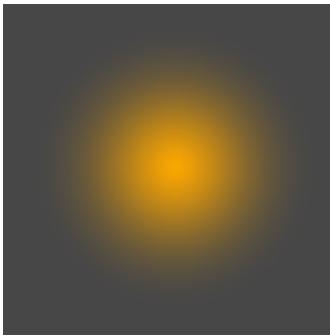


図 26:普通に描画

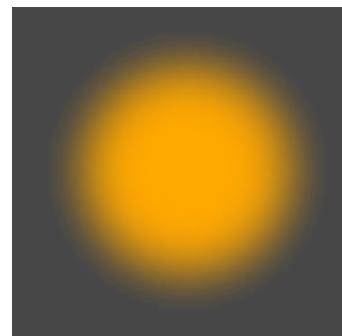


図 27:多重描画

このままだと少しマットな感じがあり、求める発光感とは程遠いですが、**Initial** の **Start Color** で Alpha を 128 にすると図 28 のような発光感が生まれます。このように、Alpha50%のパーティクルを重ねて描画するだけで手軽に強い発光エフェクトをつくることができます。

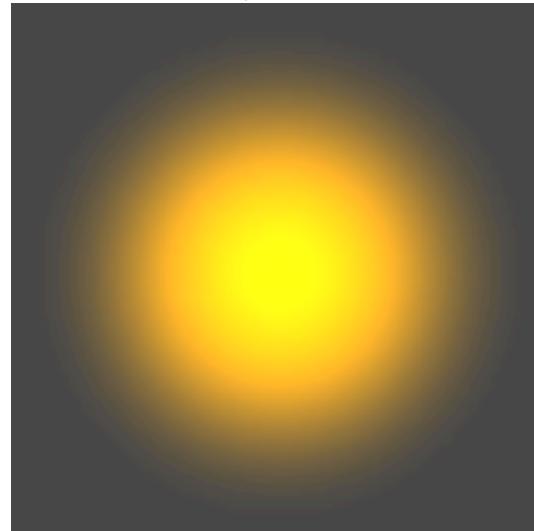


図 28:Alpha の値以外は図 27 と同じ

2. 多重パーティクルを移動させたい

ただしこの方法には 1 つ問題点があります。多重描画のために原点にパーティクルを固定している為、このパーティクルは動きません。**Initial** の **Start Speed** を 0 にする代わりに **Emission** の **Rate** を 0 にし、**Burst** で噴射をすると動く多重パーティクルすることができますが、一度に 1 つしか飛ばせず、螢のように沢山の飛び回る光を表現することは不可能です。

3. 多重パーティクルを自由に飛ばす

しかし、**Sub Emitters** を活用することにより、多重パーティクルを普通のパーティクルと同様に、一度に沢山に噴射させることができます。サンプルエフェクト

「Firefly」はこの手法で作成しています。まず、先ほどと同様に、多重パーティクルでの黄緑色に輝くエフェクトを作成します。螢らしさを出すために **Size over Lifetime** や **Color over Lifetime** も活用します。この時点で、動きは少ないですが「螢らしい見た目のエフェクト」ができます[図 29]。そして、これとは別に新しいシステムをつくります。**Shape** を **Box** にし、**Size** をある程度大きめに、さらに **Random Direction** を有効にすることで「動きは螢っぽいエフェクト」を作ることができます。そして、この「動きはそれっぽいエフェクト」の **Sub Emitters** の **Birth** に先ほど作った「見た目はそれっぽいエフェクト」を設定します。これで、多重パーティクルに動きの多様性を与えることができます。最後に「動きはそれっぽいエフェクト」の **Renderer** を無効にすることで、「自由に動く多重パーティクル」が完成します[図 30]。なお、パーティクルの速度の制御はサブシステムの **Start Speed** を操作する以外に、**Inherit Velocity** でメインシステムの速度を反映させる方法もあります。

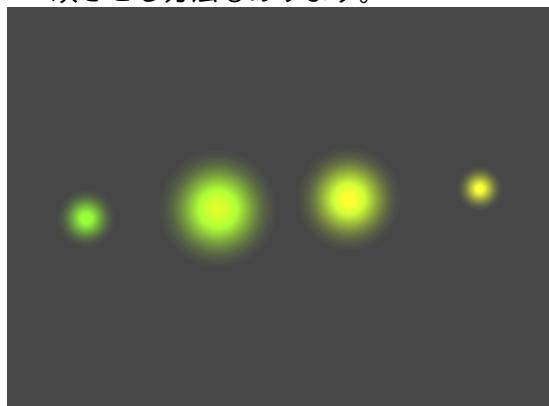
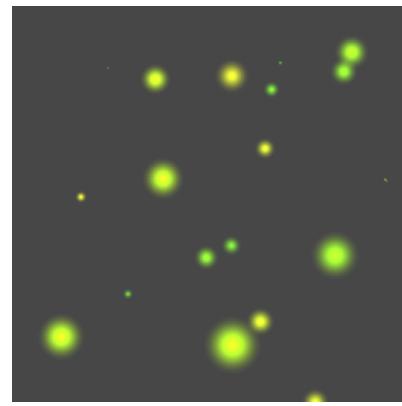


図 29: 螢っぽい見た目エフェクト

図 30:サンプルエフェクト
[Firefly]

✓ 応用アイデア

- パーティクルの色や重ねる数を変化させることにより爆発のようなエフェクトも作れます(サンプルエフェクト「Flare」)
- メインシステムの **Shape** や双方の速度制御系のモジュールを活用することにより、動きの幅を大きくつけることも可能です。

たき火・ロウソク

今度は炎の作り方です。パラメータの設定によって、猛火からロウソクの火まで、様々な種類の炎を作ることができます[図 31]。

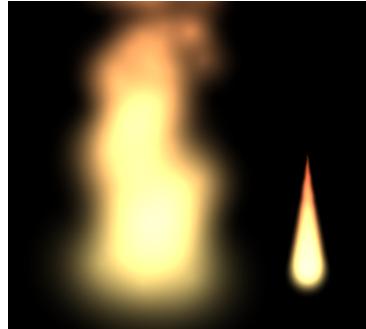


図 31: パラメータ次第で
様々な炎になります

1. パーティクルの塊をつくる

各モジュールを以下のように設定すると、パーティクルが連なって1つの塊に見えるようになります[図 32]。

- Initial : Start Speed=1, Start Lifetime=2
- Shape : Cone, Angle=0, Radius=0
- Emission : Rate=30

2. 先端を細くする

Size over Lifetime を経過時間に従ってパーティクルが小さくなるように[図 33]、パーティクル塊の先端を細くします[図 34]。

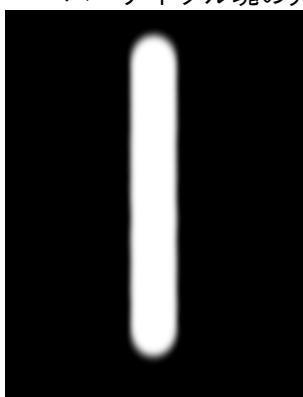


図 32
パーティクル塊

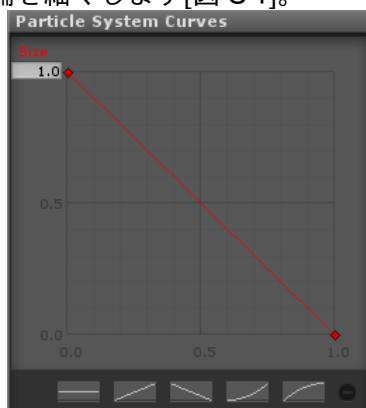


図 33
Size over Lifetime の設定

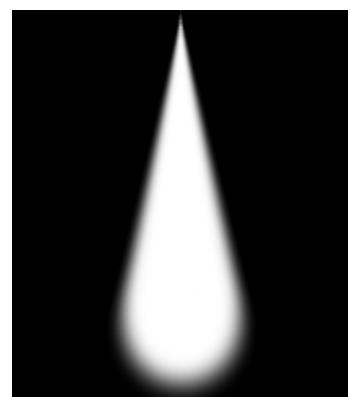


図 34
炎らしいシルエットに

3. 彩色のコツ

Start Color で赤色をつけてもいいですが図 36 のようにいまいち決まらない絵になります。ここは、**Start Color** を黄色にして、**Color over Lifetime** で徐々に赤くなるようにしてみましょう。よりそれらしくなります。ここでも、**Start Color** の Alpha を 50% にすることで炎の発光を演出することができます。

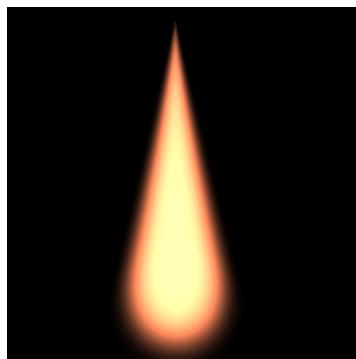


図 36:Start Color のみ

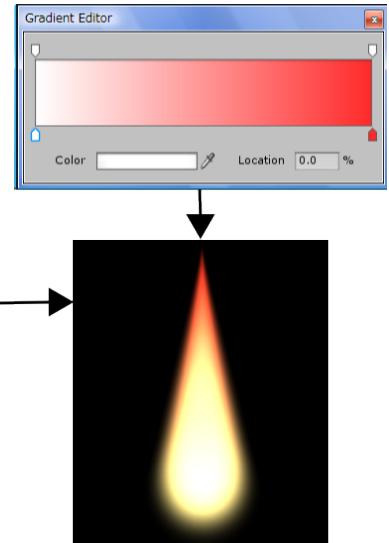
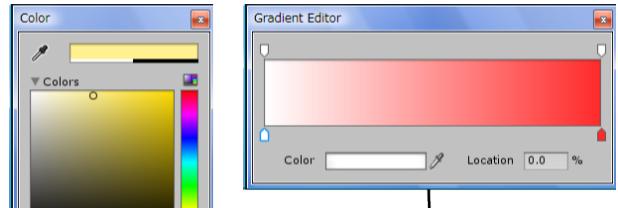


図 35

Color over Lifetime と組み合わせて調整

また、Color over Lifetime で Alpha を小さくすると、輪郭が変化します。

4. パラメータによるバリエーション

これで蝋燭の炎は完成しましたが、各プロパティのパラメータを変化させることにより様々な炎になります。

- **Start Speed** で炎の勢いを調整
- **Start Lifetime** で炎の長さを調整
- **Start Size** で炎の大きさを調整
- **Shape** の **Angle** で炎の安定感を、**Radius** で横方向の揺れを調整

一見すると複数のプロパティが同様の効果を持つように見えますが、炎が大きくなるほどプロパティによる効果の違いが顕著になります。これは自分で触りながら確認するとわかりやすいです。サンプルエフェクトとして「Candle」と「Fire」を用意しました。

5. 煙を出す

Sub Emitters の **Death** を利用すると、炎から煙が出来ます[図 37]。煙の作り方は別に説明します。

✓ 応用アイデア

Force over Lifetime で横向きの力(X,Z)を加えることによって、風に流される炎を演出することができます[図 38]。

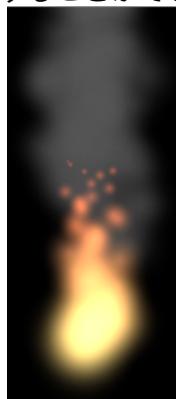


図 37

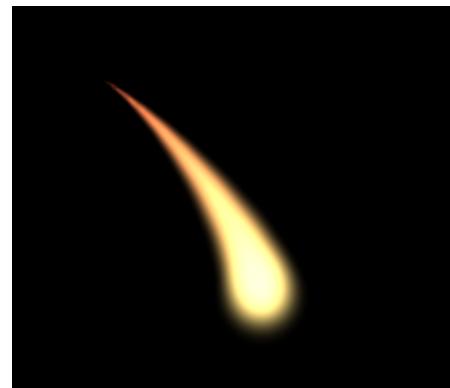


図 38:サンプルエフェクト
「Candle-Force」

煙

1. 炎とは逆で、大きくなるように

煙の作り方も基本的には炎と同じです。ただし、**Shape** は Box に、**Size over Lifetime** は大きくなるように設定したほうがそれらしい形になります[図 39]。また、ある程度パーティクル間の隙間にむらができるように **Start Size** や **Shape** の **X,Y,Z** を設定するとより煙らしく見えます[図 40]。サンプルエフェクトとして「Smoke」と「White Smoke」を用意しました。

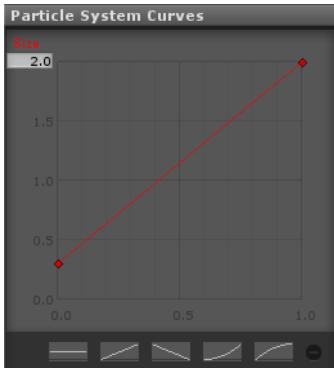


図 39

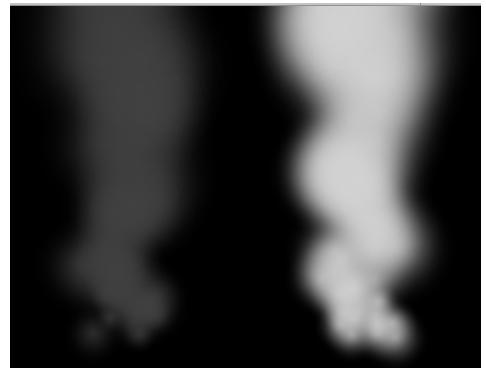


図 40

白くすれば湯気にも見える

2. 上昇のさせ方

Start Speed で上昇させるのも悪くはないですが、基本的に煙は上昇するので **Gravity Modifier** を負の値(サイズにもよりますが-0.05 ぐらい)にすることです。これで、システムの向きにかかわらず常に重力と反対の方向に加速するようになり、煙らしい挙動をするようになります。このとき、**Start Speed** は 0 にしましょう。

✓ 応用アイデア

- 色を変えることで、土埃にすることもできます。システムの正面を水平方向に向け、**Start Speed** の値を設定することによって、横に噴き出しつつ、徐々に上昇する煙になります[図 41]。また、**Emission** を **Distance** にすることによって、パーティクルシステムが移動したときのみ、移動量に応じて煙が出るようになります。キャラクターの足元などに使うときはこちらの方が便利かもしれません。
- **Sub Emitters** に設定するときは、パーティクル 1 つ 1 つからこの煙が発生することになるので、**Emission** の **Rate** を減らすと丁度良くなります。



図 41:サンプルエフェクト「Dash Smoke」

打ち上げ花火

打ち上げ花火は **Sub Emitters** を活用することによって簡単に作ることができます。基本的に **Death** にサブシステムを追加することで連鎖させていきますが、**Birth** も使うことで演出の幅がより広がります。色が途中で変わる花火のサンプルエフェクト「Firework1」

1. 打ち上げる玉を作る

まず、螢・爆発(多重描画による発光効果)[P.31]で紹介した方法で、多重パーティクルを上に飛ばします。一直線に飛ばすだけなので、多重パーティクルを飛ばすために別のシステムを作る必要はありません。この玉を打ち上げるシステムの名前を仮にシステム A とします。

2. 玉を破裂させる

システム A の **Sub Emitters** の **Death** に球状にパーティクルを飛ばすシステム B を設定します。実際に生成されるシステム B のパーティクルの数は、(システム A で設定した数)×(システム B で設定した数)になります。

3. さらに色を変える場合

劇的に色が変わる様な花火を再現する場合は、システム B で **Color over Lifetime** を使うよりも、**Sub Emitters** の **Death** に、同じ形状の違う色のパーティクルを飛ばす方がそれらしく見えます。この時、システム B の **Death** に設定するシステム C の **Shape** は無効にして、**Emission** で **Burst** の値を 1 にしておきます。こうすることで、前のパーティクルの色が変わったかだけのように見せかけることができます。

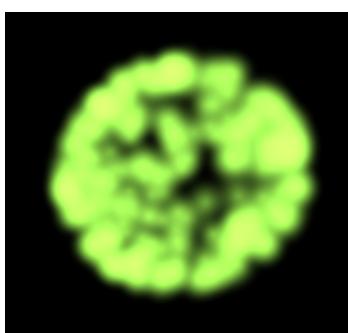


図 42
システム B による破裂

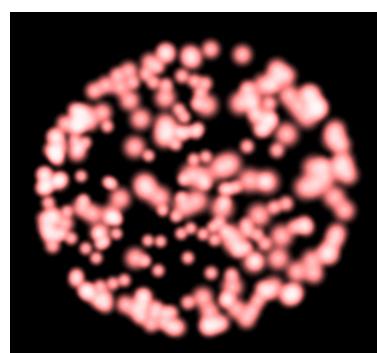


図 43
システム C による色変更

4. さらに破裂させる場合

システム C の **Shape** を **Sphere** にして、**Emission** で **Burst** の値を 1 よりも大きくすると、破裂した玉をさらに破裂させることができます。この時のコツとして、破裂後の玉は破裂前の玉よりも小さくしておくと自然に見えます。このパターンのサンプルは「Firework2」です[図 44]。

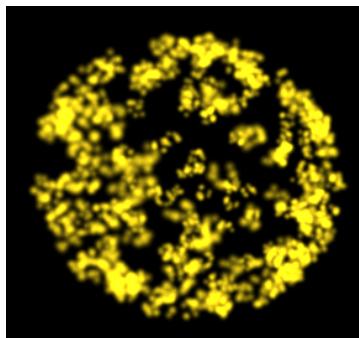


図 44:Firework2

5. 玉に尻尾をつけてみる

お好みでシステム A の **Sub Emitters** の **Birth** に煙を吐き出すシステムを設定するのも 1 つの手です[図 45]。この時、煙を吐き出すシステムの **Size over Lifetime** はパーティクルの大きさを徐々に小さくする設定にしておくと、それらしく見えます。また、同様にシステム C の **Birth** に煙をつけると、また趣の違った花火になります。これらのサンプルとして「Firework3」を用意しました[図 46]。

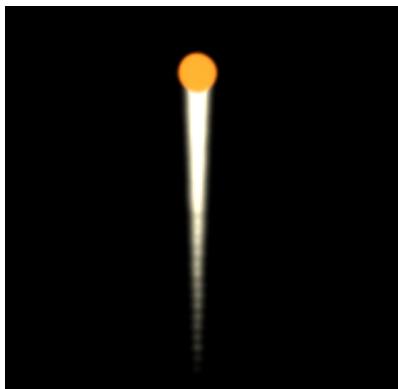


図 45:尾をひく球



図 46:Firework3

滝・噴水

Sub Emitters の **Collision** を利用することにより、水しぶきを再現することができます[図 47]。

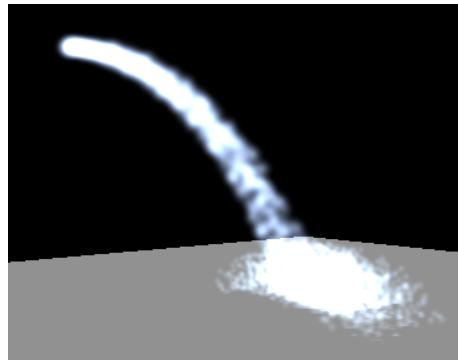


図 47
サンプルエフェクト「Fall」

1. 水流を作る

衝突するまでの水流は今までのエフェクトとほぼ同じように、半透明のパーティクルを連ねることで作れます。 **Emission** を **Rate** を多めに、**Start Size** を小さめにすると、水らしく見えます。 **Start Speed** によりますが、例えば **Start Speed** が 1 の時、**Rate** を 400、**Start Size** を 0.2 が目安です。 **Start Speed** はそのまま水流の勢いになるので、状況に合わせて調整してください。**Gravity Modifier** に正の値を設定すると、重力に従って水が落ちるようになります[図 48]。

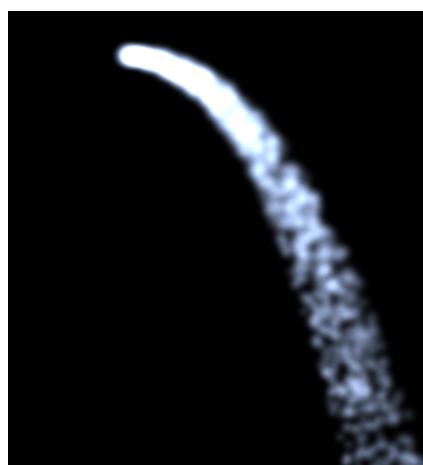


図 48

2. 衝突時に水流を消滅させる水しぶきを作る

Collision モジュールで衝突判定の平面を設定します。初期設定だとそのまま弾んでしまいますが、**Min Kill Speed** の値を設定することによって平面に衝突したパーティクルを全て消滅させます[図 49]。設定する値は他のパラメータに依存するので、実際に結果を確認しながら調整します。

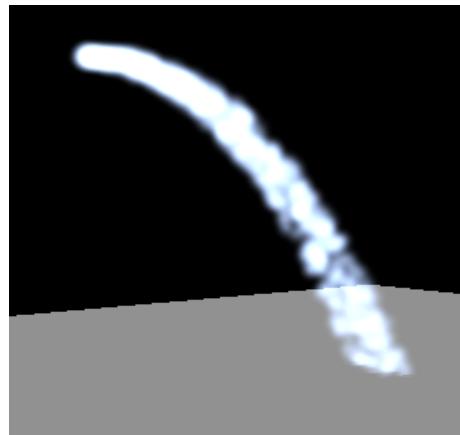


図 49
衝突によりパーティクルを消す

3. 水しぶきをつける

そして **Sub Emitters** の **Collision** にシステムを追加することで水しぶきを発生させます。水しぶきエフェクトの **Shape** は **Hemisphere** が最適だと思います。衝突した 1 つのパーティクルが複数に割れるイメージで、**Emission** の **Burst** は 10 ぐらいに、**Start Size** は元のパーティクルよりも小さくします。もちろん多ければ多いほど激しくなりますが、多すぎると不自然なので慎重に。さらに **Gravity Modifier** を、発生した後の水しぶきが落ちるように設定します。水しぶきは元のパーティクルより小さいので **Gravity Modifier** の値も小さく設定します。[図 50]

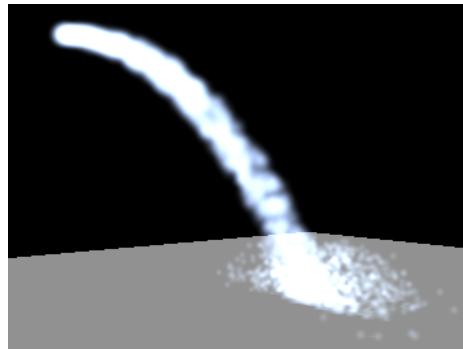


図 50
水しぶきがついてそれらしく

Sub Emitters を使用する際の共通の注意点ですが、サブシステム側の **Max Particles** の値を大きめに設定しておくことをお勧めします。サブシステムは条件を満たしたパーティクル全てから設定された量のパーティクルを噴射するため、想像以上のパーティクルを噴射します。このため **Max Particles** による制限の影響を受けやすく、パーティクルの噴射を意図せず止める原因になりやすいです。

4. 形状を変えて雰囲気を変える

このままで水流に見えますが、**Renderer** の **Render Mode** を **Stretched Billboard** にすると雰囲気の異なる水流ができます。今回の場合は **Speed Scale**、**Length Scale** はそれぞれ 0.3、0 ぐらいにすると良い感じになりました。同時に、**Shape** の **Angle** をパーティクル間に大きな隙間ができる程度に大きくすることによって、水のカーテンのような表現が生まれます[図 51]。このバリエーションを「Fall2」としてサンプルエフェクトに収録しています。

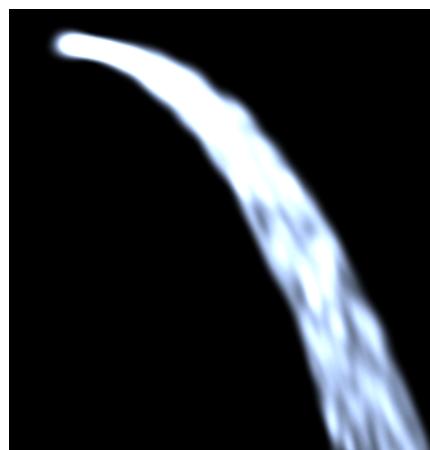


図 51
サンプルエフェクト「Fall2」

✓ 応用アイデア

- システムの角度を変更し、噴射の方向を真上に変えるだけで噴水としても使えます。
- **Start Speed** と **Emission** の **Rate** を曲線を用いて同時に変化させることで、噴射の勢いに緩急をつきます。噴水を表現する際に特に有効です。
(例としてサンプルエフェクト「Spring」を収録)

周回軌道、スパイラル(公転するエフェクト)

図 52 のように「何かの周りを公転するエフェクト」はよくゲームに使われるエフェクトの一つです。しかし、Particle System のみでこのようなエフェクトを作ることは困難です。本書の最後の実装例はこのような公転エフェクトです。

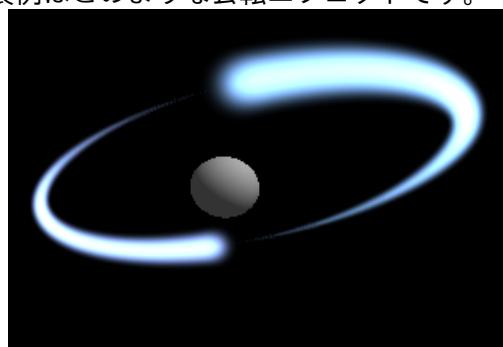


図 52:周回運動をする光

1. 公転の実現法

公転を実現させる為には、図 53 回転の中心となる Game Object(Center)を用意しその子供として Particle System(satellite)を Center から離れた位置に配置します。この状態で Center を自転させると、それに合わせて satellite が回ります。これで噴射口が Center の周りを公転します、あとは、今までのエフェクトを応用することによって、目的のエフェクトを作ることができます。

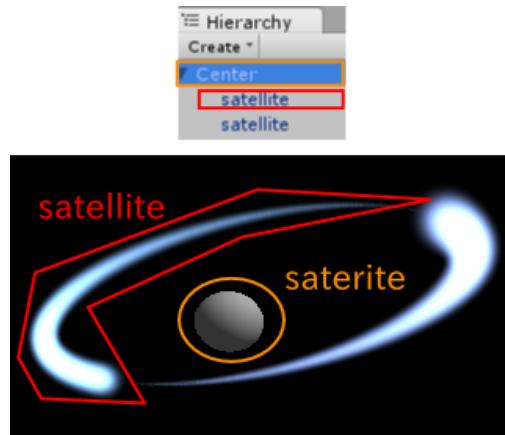


図 53:システムを子供にする

2. 自転のさせ方

ある Game Object を自転させる方法として以下の 2 つが考えられます。

- i. Animation コンポーネントで回転させる
- ii. スクリプトで回転させる

i. の方法はとりあえず回転させるだけなら簡単ですが、回転速度の制御が困難です。よって今回は ii. の方法を採用します。今回はエフェクトに特化した本にするため、スクリプトの解説はしませんが、自転させるためのスクリプト「Orbital Rotation」を用意しました[図 54]。使い方は簡単で、自転させたい Game Object にスクリプトを追加し、Angular Velocity に回転速度(deg/s)を設定するだけです。このスクリプトは Game Object の Y 軸を回転軸として自転するので、自転の軸の調整は Game Object の Rotation を調整することで行います。

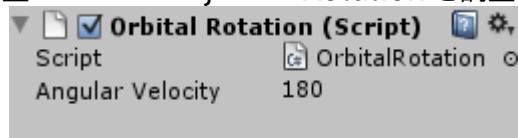


図 54:自転スクリプト

3. Particle System の設定のコツ

サンプルエフェクト「Orbit」を例に説明します。まず **Simulation Space** を World にします。Local にするとシステムの移動にパーティクルが付いてきてしまうため、尾を引くようなエフェクトにすることができません。また、**Emission** を Distance にすることもコツの一つです。Time でも同様のエフェクトは作れますが、Distance にすると「移動量当たりの噴射量」を設定できるため、エフェクトの回転速度と噴射量を独立に扱うことができ、調整が簡単になります。**Shape** を無効、Start Speed=0 にして、パーティクル間に隙間ができるないように

Emission の Rate を増やすと、円周上にパーティクルの帯ができます。最後に Size over Lifetime で尾を細らせることで完成です。

✓ 応用アイデア

- Gravity Modifier や Start Speed などで、公転の軸と平行な方向にパーティクルを移動させることにより、らせん状のエフェクトを作ることができます[図 55]。(サンプルエフェクト「Spiral」を参照)
- 今回の趣旨とは外れますが、システム自体を回転させても面白いエフェクトができます。サンプルエフェクト「Snow Ball」はその一例です。

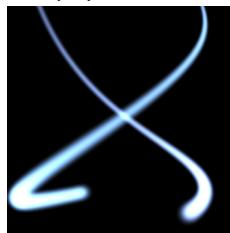


図 55

クレジット

発行:Tablet Frontierlet

執筆者:よもぎ

mail: yomogi@tabletfrontierlet.net

twitter: yomogi_ft