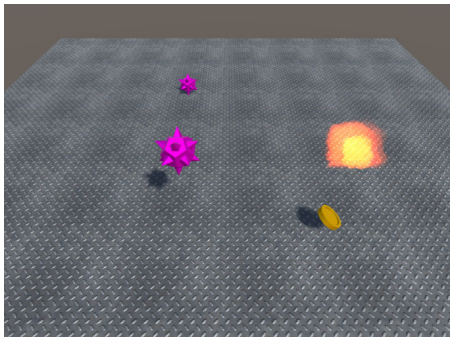
CoinSpike (コインとスパイク)

SOUND

あらかじめ定義されたプログラムやデータ構造、設計された(クラス化された)ものなどを、メインメモリ上に展開して処理・実行できる(見える)状態にしたものをインスタンスといいます。
ゲームプログラミングでは、非常に重要な概念です。

Unity に於けるインスタンス化とは、プレハブ化(登録)してあるものをシーン(ヒエラルキー)内に運び込んで登場させることを指します。プロジェクト欄のプレハブをヒエラルキー欄にマウスで運んでもインスタンス化ですが、ここではゲーム実行中に発動される命令文で運び込む(発生させる)方法を用います。

ゲームでの具体例ではミサイルや爆発などが相当します。最初から存在しなかったものがゲーム中に登場し、途端にサウンドを鳴動し、自発的に飛行(移動)をし、命中(接触)が起これば自分自身を撤去(デストロイ)します。撤去直前には、その座標位置に爆煙や閃光などのプレハブを設置(インスタンス化)で視覚演出を行っています。



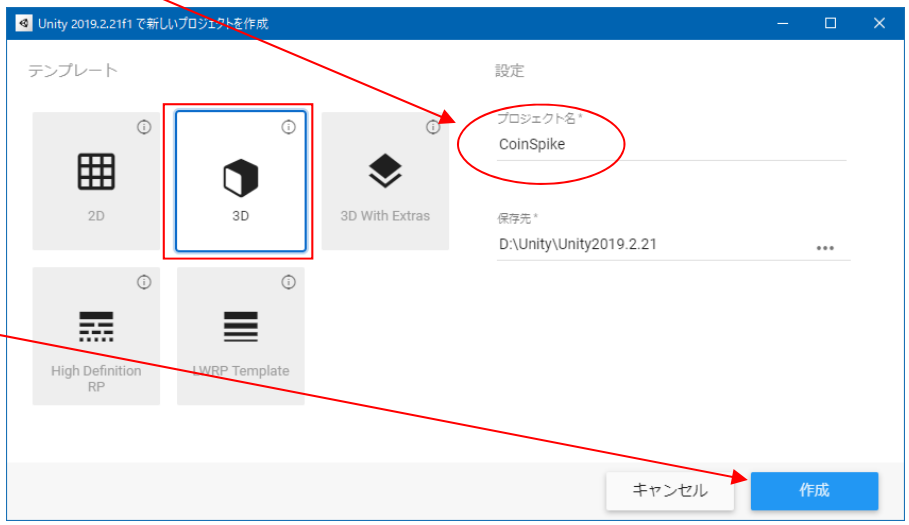
今回取り組むコンテンツの内容は、ゲームフィールドにプレハブを自動で登場(インスタンス)させる手法を理解します。
プレハブは2種類とし、獲得すべきもの(コイン)と避けるべきもの(スパイク)となります。それぞれ寿命があり、数秒後には撤去(デストロイ)されます。

応用として、爆発のプレハブを自身で用意します。爆発は一般的にパーティクル(粒子)システムを用い、Unity にもシュリケン・パーティクルの名で実装されています。スパイクが消える際に爆発パーティクルを登場させ、インスタンス理解の反復演習とします。生成時と撤去時には「サウンド鳴動」を行い、サウンド素材の扱い方についても理解を深めることとします。

制作環境の整備

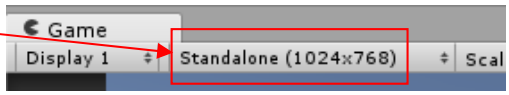
【STEP1】 プロジェクトの準備

● Unity を起動し、新規プロジェクト CoinSpike を 3D モードで作成します。

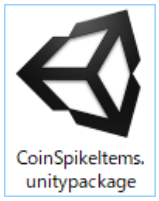


ボタン作成を押下します。

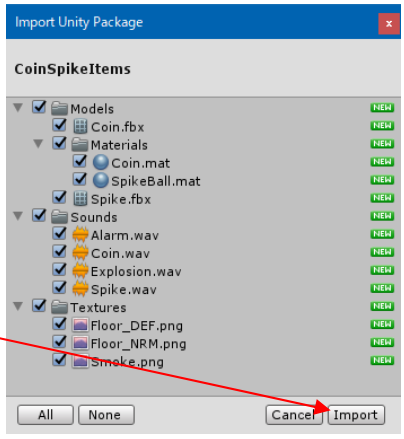
● Game パネルで、画面を Standalone(1024x768)にします。



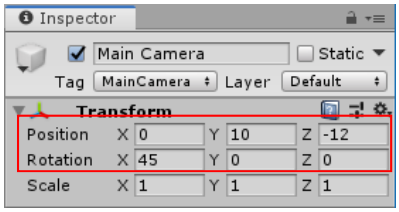
● メニューAssets > Import Package > Custom Package…と進み、配布した CoinSpikeItems.unitypackage を選択します。



内容が表示されたら、Import ボタンを押下します。



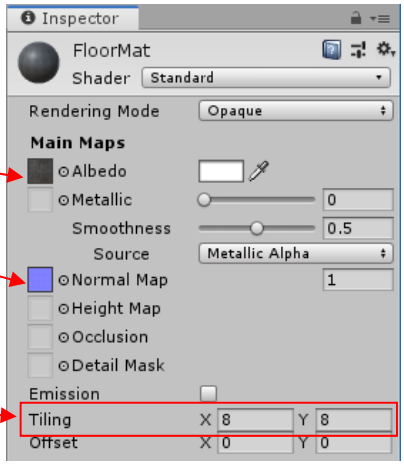
● ヒエラルキー欄から Main Camera を選び、インスペクタで調整します。



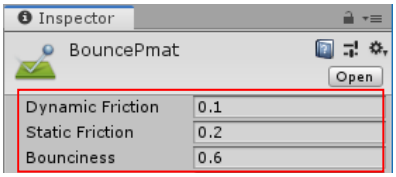
【STEP2】 ゲームフィールドの設定

● プロジェクト欄の Create から Material（マテリアル）を選んで作成し、名称を FloorMat とします。
インスペクタでパラメータを調整します。
アルベドマップ：Floor_DEF
ノーマルマップ：Floor_NRM

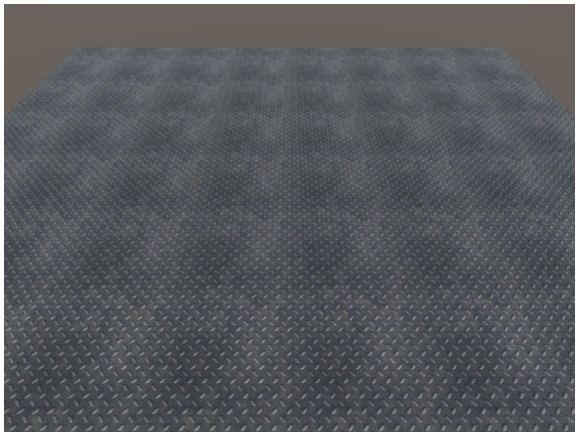
タイリング回数を8にします。

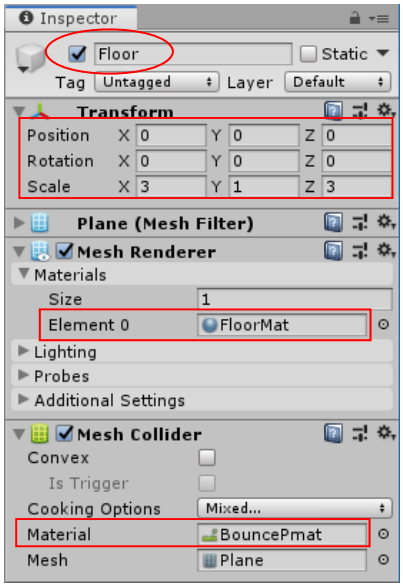


● プロジェクト欄の Create から Physic Material（物理マテリアル）を選択して作成し、名称を BouncePmat とします。
インスペクタでパラメータを調整します。



● ヒエラルキー欄の Create から 3D Object ＞ Plane を選択して作成し、名称を Floor とします。
インスペクタでパラメータを設定します。

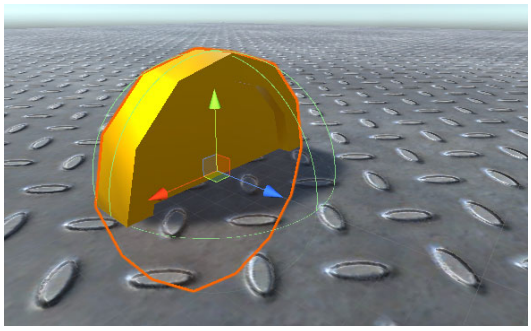


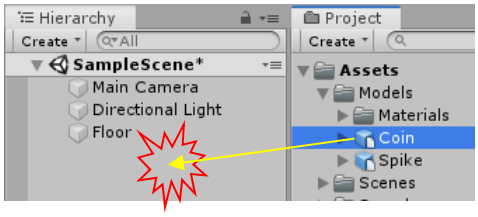


コインの運営

【STEP3】 コインのプレハブ化

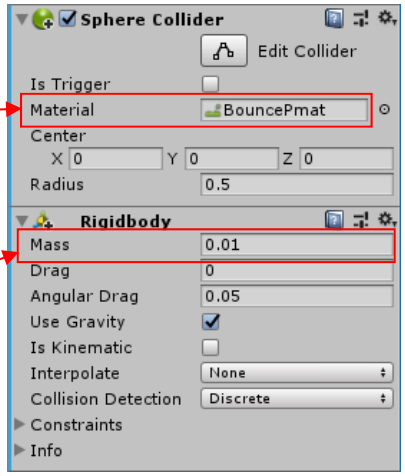
● プロジェクト欄の Models ＞ Coin をヒエラルキー欄へドラッグ &ドロップします。





● ヒエラルキー欄の Coin を選んでいる状態で、インスペクタの Add Component から Physics ＞ SphereCollider（球体コライダ）を追加します。物理マテリアルを設定します。
（ん？なぜ、12 角柱のコライダーではないのでしょうか？）

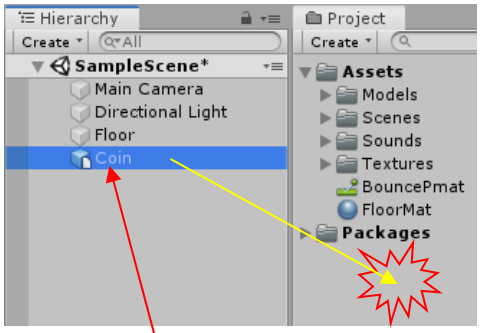
● 同様に Add Component から Physics ＞ Rigidbody（物理演算の仲間入り指示）を追加します。
Mass（重さ）に 0.01 を設定します。（10g の意味となります。）



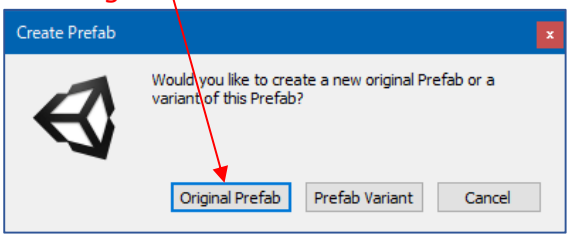
● 同様に Add Component から Audio ＞ AudioSource（音源）を追加します。

項目 AudioClip に Coin を指定します。

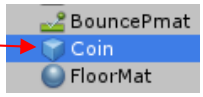
● これで完成です。ヒエラルキー欄の Coin をプロジェクト欄に運んで
プレハブとして登録します。



このウィンドウが出たら、Original Prefab を選びます。



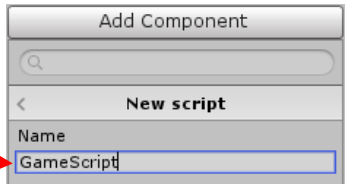
プレハブ化されました。



● 今後は、このプレハブをインスタンス生成していきます。ヒエラルキー欄に残った Coin は削除しておきます。

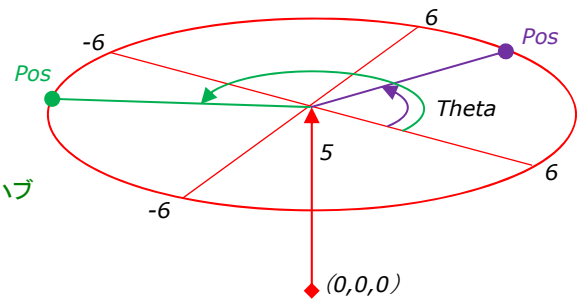
【STEP4】 ジェネレーター（生成装置）の作成

● ヒエラルキー欄の MainCamera を選択し、インスペクタの Add Component から New script を選択します。
スクリプトの名称を GameScript とします。



● スクリプト GameScript を次のように記述します。

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class GameScript : MonoBehaviour {

    float Elapsed = 0.0f; // 経過時間
    float Theta; // 角度シータ(θ)
    public float Interval = 1.5f; // 発生間隔
    public GameObject CoinPrefab; // コインのプレハブ

    void Start() {

    }

    void Update () {
        Elapsed += Time.deltaTime;
        if (Elapsed >= Interval) {
            Elapsed = 0.0f;
            //ランダムの角度シータ(θ)
            Theta = Random.Range(0.0f, Mathf.PI * 2.0f);
            //高さ5で直径12の円周のどこか
            Vector3 pos = new Vector3(Mathf.Cos(Theta) * 6.0f, 5.0f,Mathf.Sin(Theta) * 6.0f);
            Instantiate(CoinPrefab, pos, Quaternion.identity);    //インスタンス生成
        }
    }
}
```



乱数の使い方1
Random.Range(num1,num2)

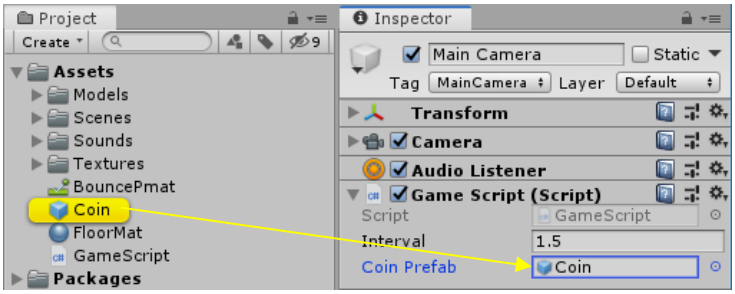
【学習項目】 インスタンス命令
基本構文
Instantiate(生成するゲームオブジェクト名);
ゲームオブジェクトはパブリックにしてインスペクタ欄に表示し、マウスで与えてあげることが多い。
応用構文①
Instantiate(生成するゲームオブジェクト , 生成位置 , 回転状況);
実はゲーム制作ではこちらの使い方が多い。生成位置はVector3型で指定、回転状況はQuaternion型で指定。
応用構文②
一時的なゲームオブジェクト名 = Instantiate(生成するゲームオブジェクト名) as GameObject;
シーンに置くだけでなく、まだ何か処理を与えたい時に一時的な名前で生成し、続く命令文で処理を行う。

インスタンスを生成する際に、3つ目の指定項目である回転の状況では、プレハブが設計された通りの回転向きで生成される Quaternion.Identity がよく使われます。ですが、ランダムな回転向きで生成された方が良い場合もあります。その場合は、Random.rotation を指定します。

乱数の使い方2
Random.rotation

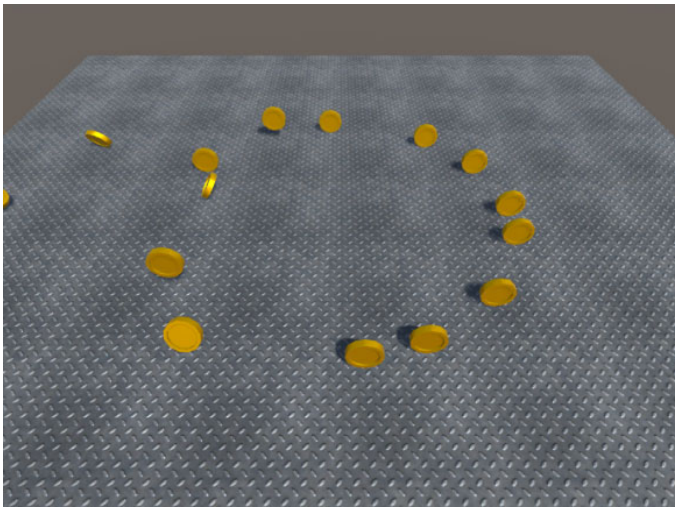
● ヒエラルキーの MainCamera を選択し、インスペクタの GameScript が見えるので、CoinPrefab にプレハブの Coin をドラッグ＆ドロップします。
プレハブの Coin ですよ。プレハブの。

この時に間違えて、よく Models ＞ Coin を渡してしまうミスがあります。

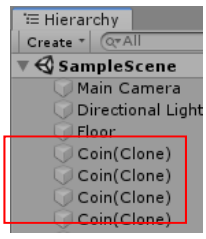


● プレイボタンを押下して、動作確認します。

コインが生成され、サウンドが鳴動されます。

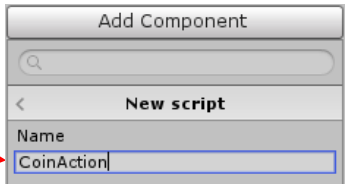


三角関数を用いて生成位置を決めたので、円状に落下してくることと、ヒエラルキー欄にオブジェクトが生成されている状況と名称にも注意します。



【STEP5】 コインの破棄（デストロイ）

● プロジェクト欄のプレハブの Coin を選択し、インスペクタの Add Component から New script を選択します。
スクリプトの名称を CoinAction と命名します。



● スクリプト CoinAction を次のように編集します。

【学習項目】 デストロイ：ゲームオブジェクトを撤去する命令

構文 ： Destroy (ゲームオブジェクト名 , 撤去までの遅延秒数);

小文字で始まる gameObject の場合、このスクリプトが取り付けられている自分自身を指します。

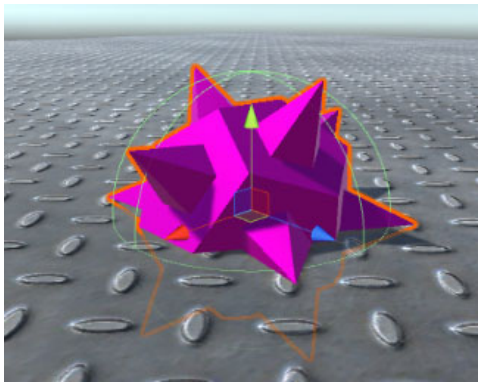
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CoinAction : MonoBehaviour {

    void Start () {
        Destroy (gameObject,3.0f);
    }

    void Update () {

    }
}
```

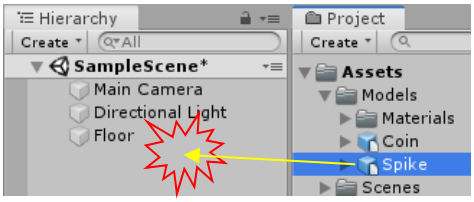
● プレイボタンを押下します。
寿命が3秒（3秒で撤去）になっていることを確認します。

スパイクの運営

【STEP6】 スパイクのプレハブ化

● プロジェクト欄の Models > Spike をヒエラルキー欄へドラッグ &ドロップします。

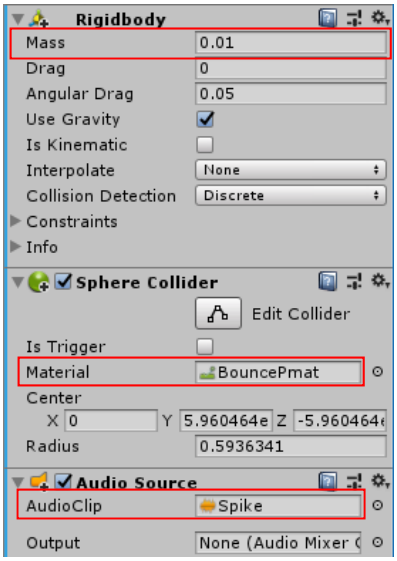




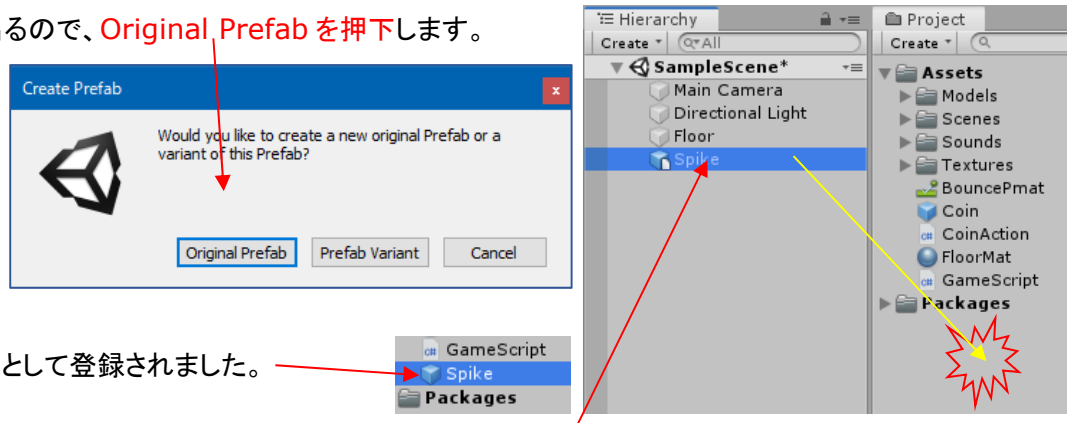
● インスペクタの AddComponent から Physics > Rigidbody（物理演算の仲間入り）を追加し、Mass（重さ）に 0.01 を設定します。（10g の意味となります。）

● 同様に AddComponent から Physics > SphereCollider（球体コライダ）を追加します。
インスペクタでパラメータを設定します。

● 同様に AddComponent から Audio > Audio Source を選択し、音源としての機能を持たせます。
インスペクタでパラメータを設定します。



● 完成したヒエラルキー欄の Spike をプロジェクト欄に運んでプレハブとして登録します。
このメッセージが出るので、Original Prefab を押下します。



このようにプレハブとして登録されました。

● 登録後は、ヒエラルキー欄に残った Spike は不要になったので削除しておきます。

【STEP7】 スパイクのインスタンス生成

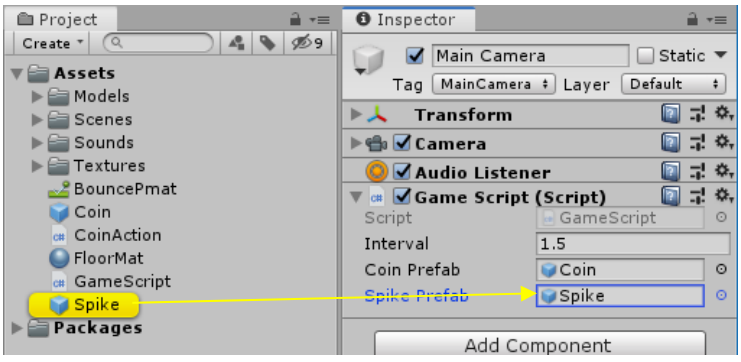
● スクリプト GameScript を改造します。

```
//～前略～
    public GameObject SpikePrefab; //スパイクのプレハブ

    void Start() {

    }

    void Update () {
        Elapsed += Time.deltaTime;
        if (Elapsed >= Interval) {
            Elapsed = 0.0f;
            //ランダムの角度シータ(θ)
            Theta = Random.Range(0.0f, Mathf.PI * 2.0f);
            //高さ5で直径12の円周のどこか
            Vector3 pos = new Vector3(Mathf.Cos(Theta) * 6.0f, 5.0f,Mathf.Sin(Theta) * 6.0f);
            GameObject Prefab = ( Random.value < 0.3f ) ? CoinPrefab : SpikePrefab;
            Instantiate( Prefab , pos, Quaternion.identity);    //インスタンス生成
        }
    }
}
```

```
GameObject Prefab;
if (Random.value < 0.3f) {
    Prefab = CoinPrefab;
} else {
    Prefab = SpikePrefab;
}
```

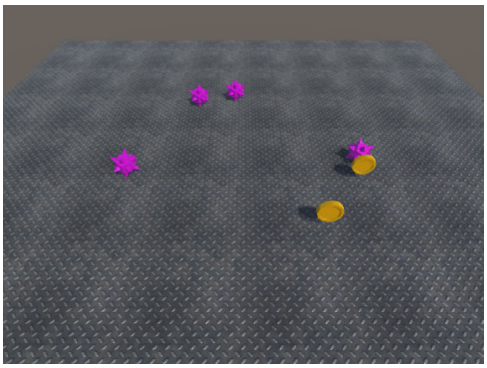
乱数の使い方3
Random.value

● ヒエラルキー欄の MainCamera を選択し、インスペクタ欄のスクリプトに登場した Spike Prefab にプレハブ Spike をドラッグ &ドロップします。



● プレイボタンを押下します。

生成音と同時にスパイクとコインが一定の確率でランダムに生成されます。

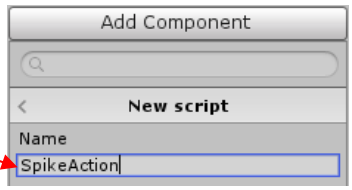


【STEP8】 スパイクの破棄

スパイクを6秒後に破棄する直前に、消滅することを警告する1秒間を設けます。その1秒間では、警告音を鳴動し、色を点滅させる効果を与えます。破棄の方法がコインとは異なり、破棄までの時間を Elapsed で測っています。

どちらで実現しても構いませんが、こちらの方が手動で測って破棄しているので、警告音を鳴動するなどの複雑な処理の実装に耐えらるわけです。

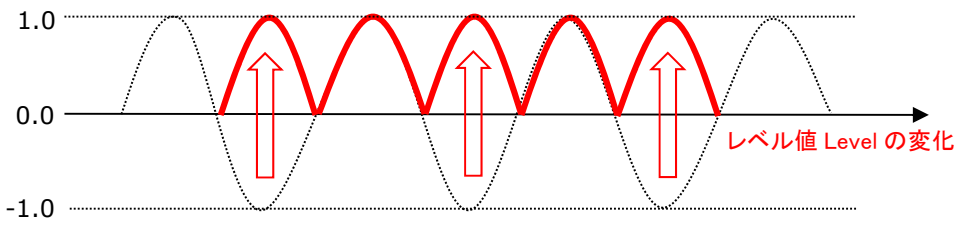
● プロジェクト欄のプレハブ Spike を選択し、インスペクタの Add Component から New script を選択します。名称を SpikeAction とします。



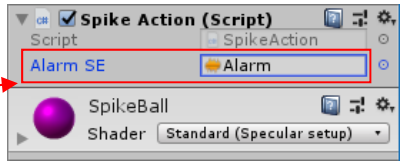
● スクリプト SpikeAction を次のように編集します。

Mathf.Abs ： 絶対値

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class SpikeAction : MonoBehaviour {

    float Elapsed = 0.0f; //経過時間
    float Level; //点滅レベル
    bool isAlarm; //アラーム中か？
    public AudioClip alarmSE;    //アラーム音

    void Update () {
        Elapsed += Time.deltaTime;
        if (isAlarm) {   //サイン関数で指定色に変化させ点滅を演出
            Level = Mathf.Abs(Mathf.Sin(40.0f * Time.time));
            GetComponent<Renderer>().material.color = Color.red * Level;
            if (Elapsed > 6.0f) {
                Destroy(gameObject, 0.0f); //自身Spikeを即刻撤去
            }
        }
        else if (Elapsed > 5.0f) {
            GetComponent<AudioSource>().PlayOneShot(alarmSE, 1.0f);
            isAlarm = true;
        }
    }
}
```

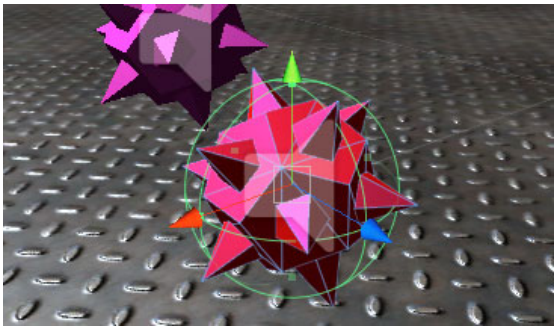


● プロジェクト欄のプレハブ Spike を選択し、インスペクタに表示されるスクリプトのコンポーネントに、新しくパラメータ AlarmSE が登場しています。
サウンドファイル Alarm を設定します。



● プレイボタンを押下します。

5秒が経過すると、スパイクはアラームを鳴らして点滅を開始し、1秒後に消えることを確認します。

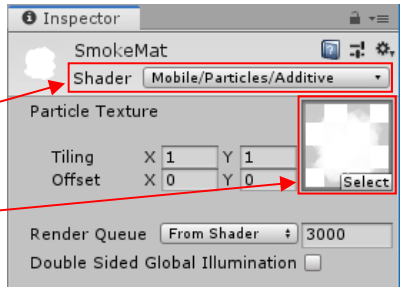


爆発エフェクトの運営

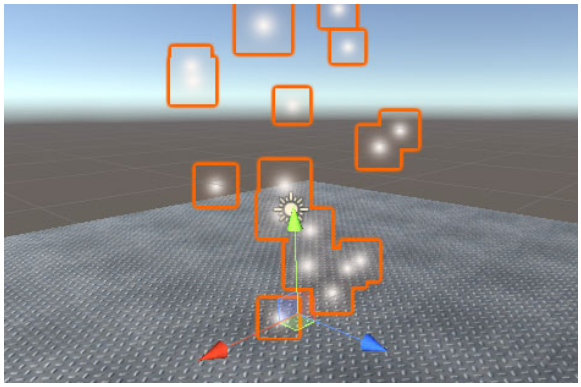
【STEP9】 パーティクルシステムを扱う

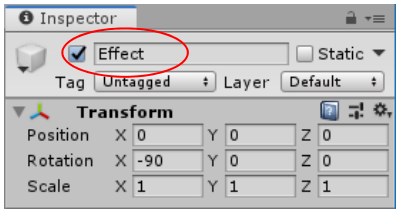
● プロジェクト欄の Create からマテリアルを作成し、名称を SmokeMat とします。
描画方法の Shader を Mobile > Particles > Additive とします。

テクスチャはボタン Select を押下し、Smoke を指定します。



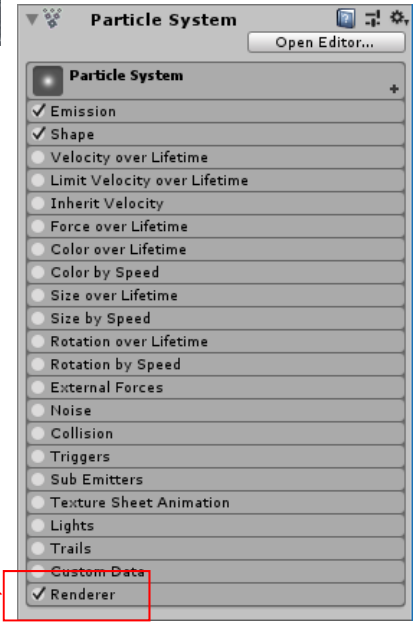
● ヒエラルキー欄の Create から Effects > Particle System を選択し、名称を Effect とします。





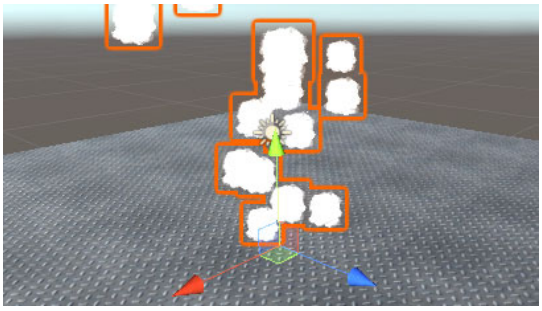
パーティクルシステムは、非常に数多くのカテゴリで表現され、その中に幾つものパラメータを有しています。
設置時の状況では3個のカテゴリだけが設定されていて、幾つかは追加しながら、詳しい設定を加えていきます。

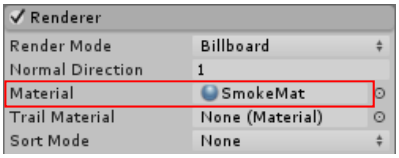
この時に、比較的、鉄則ともいえるアプローチがあり、必ず最後のカテゴリである Renderer（描画方法）から設定を行う事です。



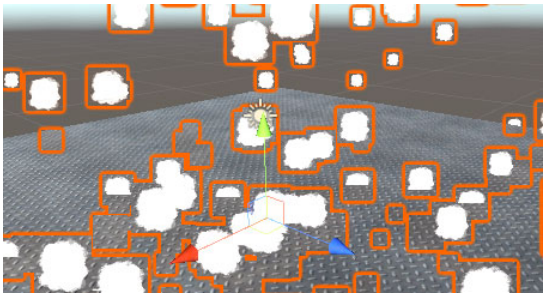
見え方に関する情報が最初に決まらないと、他の設定値を決定することが出来ないからです。

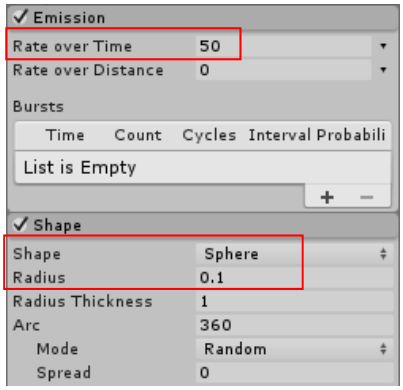
● カテゴリ Renderer を開きます。
パラメータを設定します。



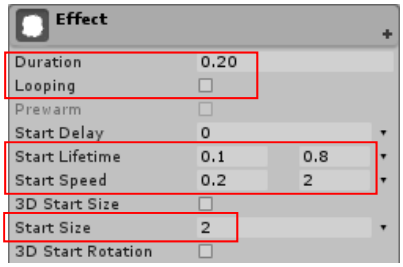


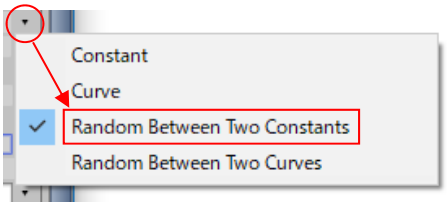
● カテゴリ Emission、Shape を開きます。
パラメータを設定します。



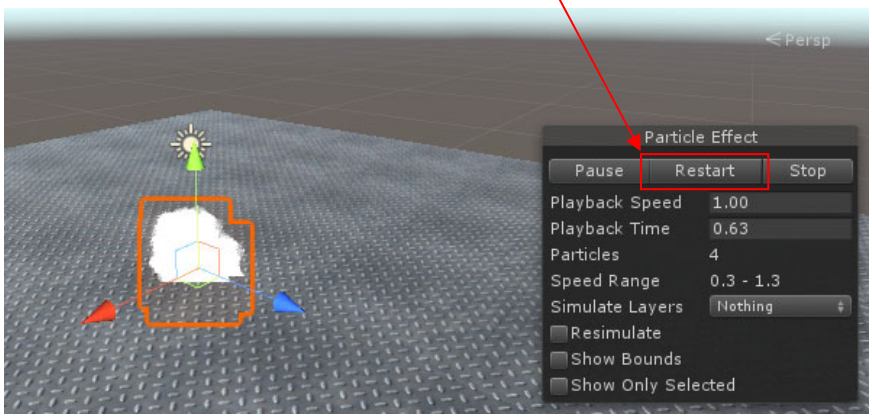


● このパーティクル本体である、最初の欄 Effect を開きます。
パラメータを設定します。
値を2つ入力するパラメータは、▼マークを押下して、Random Between Two Constants を選びます。

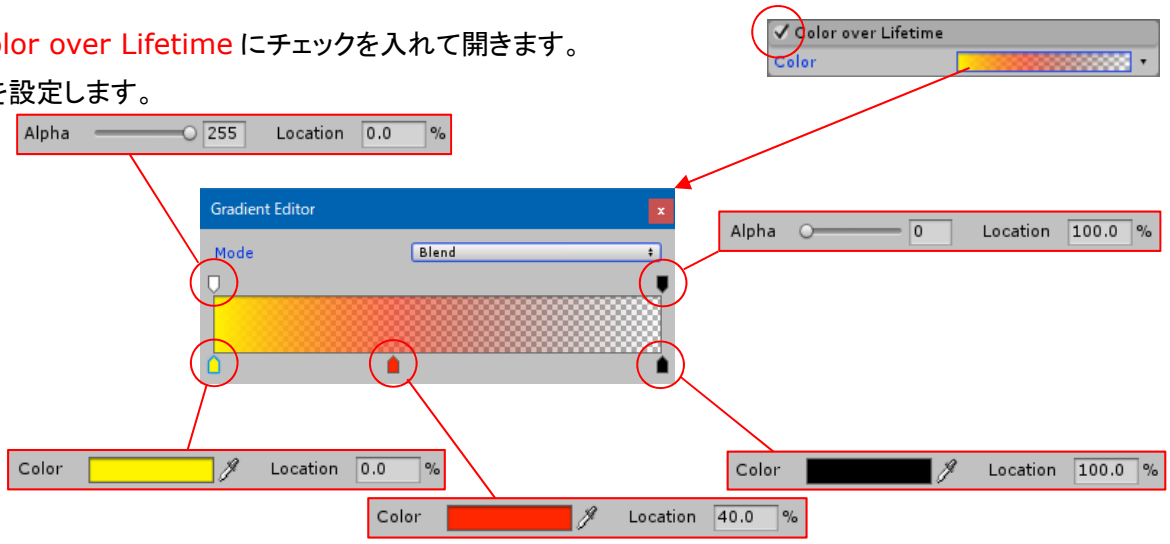




項目 Looping がオフになるので、シーンで確認が出来なくなります。この場合、Restart を何度も押して確認することになります。



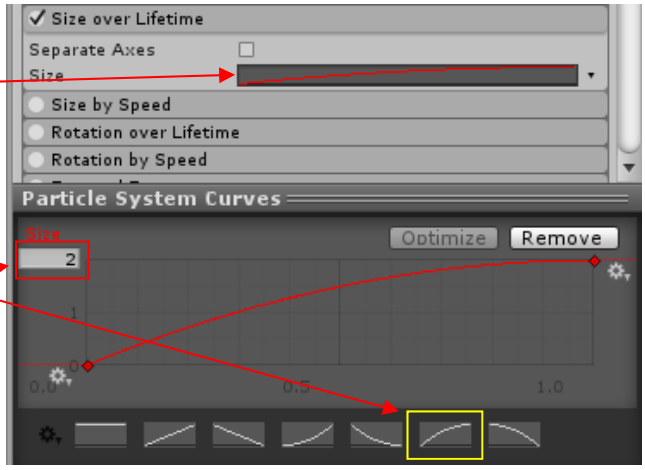
● カテゴリ Color over Lifetime にチェックを入れて開きます。
パラメータを設定します。



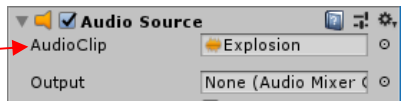
● カテゴリ Size over Lifetime を開きます。
Size のグラフをクリックすると、カーブが現れます。

インスペクタの幅を広げると、カーブのプリセットが表示されるので、クリックします。

最大サイズを2にします。

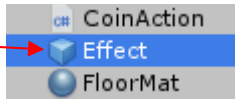


● この Effect を選択している状態で、インスペクタの Add Component から Audio ＞ Audio Source を選択します。
AudioClip に Explosion を設定します。

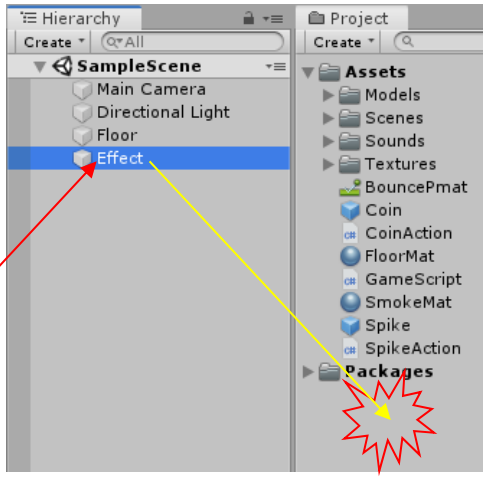


● これでエフェクトは完成したので、プロジェクト欄にドラッグ &ドロップしてプレハブ化します。

プレハブ化されました。



ヒエラルキー欄に残った Effect は不要になったので削除します。



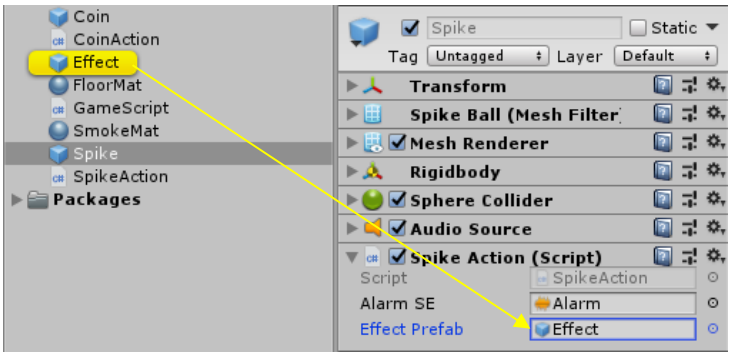
【STEP10】 爆発エフェクトをインスタンスする

● スクリプト SpikeAction を次のように編集します。

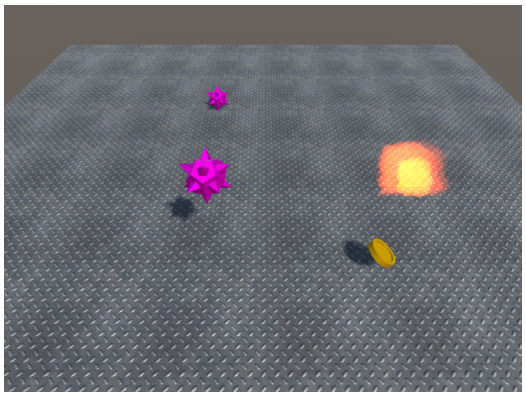
爆発プレハブをインスタンス生成し、2秒後に撤去する指示も与えています。

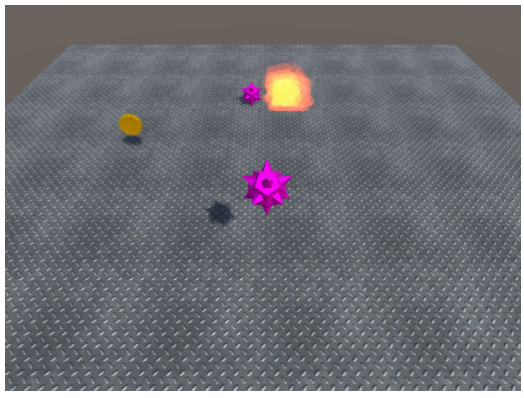
```
//～前略～
    public GameObject EffectPrefab; //エフェクトプレハブ

    void Update() {
        Elapsed += Time.deltaTime;
        if (isAlarm) {   //サイン関数で指定色に変化させ点滅を演出
            Level = Mathf.Abs(Mathf.Sin(40.0f * Time.time));
            GetComponent<Renderer>().material.color = Color.red * Level;
            if (Elapsed > 6.0f) {
                //エフェクトをインスタンス生成
                GameObject E = Instantiate( EffectPrefab, transform.position,
                    Quaternion.identity ) as GameObject;
                Destroy( E, 2.0f ); //エフェクトを2秒後に撤去
                Destroy(gameObject, 0.0f); // 即刻破棄
            }
        } else if (Elapsed > 5.0f) {
            GetComponent<AudioSource>().PlayOneShot(alarmSE, 1.0f);
            isAlarm = true;
        }
    }
}
```

● プロジェクト欄のプレハブ Spike を選択し、インスペクタに登場した項目に、プレハブ Effect をドラッグ＆ドロップします。



● プレイボタンを押下して動作確認をします。





以上