Mathfクラス

目次

１．Mathfクラスとは

UnityにはMathfクラスというクラスがあります。

スクリプトリファレンス

Unityが提供する数学関数の集まり(クラス)です。ゲーム開発でよく使う数値計算（例：三角関数、補間、絶対値、丸め処理など）を簡単に行えるようになっています。

このMathfで使用頻度が高いメソッドについて知見を付けたり、どのようにしてUnityに活用出来るかを紹介していきます。

１．Mathfクラスとは

＜Mathfの基本情報＞

名前空間：UnityEngine

クラス名：Mathf

特徴：静的クラスなのでインスタンス化せずに使える(Debug.Logと一緒)

（例：Mathf.Sin(x)）

用途：ゲーム内の計算処理に便利

（動き、回転、物理、UIアニメーションなど）

２．メソッド紹介と処理内容

Mathfの中でも使用頻度が高いメソッドを紹介していこうと思います。

関数	説明	備考
Mathf.Abs(x)	絶対値を返す	
Mathf.Clamp(x,min,max)	値をmin～maxの範囲に制限	
Mathf.Sqrt(x)	平方根	
Mathf.Floor(x)	切り捨て	
Mathf.Ceil(x)	切り上げ	
Mathf.Round(x)	四捨五入	偶数丸めであることに注意！
Mathf.Lerp(a,b,t)	線形補間(aからbへtの割合で移動)	めっちゃ使える！
Mathf.Sin(x)	サイン波を返す(ラジアン)	別でやりましょう！
Mathf.Cos(x)	コサイン波を返す(ラジアン)	別でやりましょう！
Mathf.Rad2Deg	弧度法から度数法に変換	別でやりましょう！

２．メソッド紹介と処理内容

＜問題＞

Input.GetAxisで動かない場合は0、左右どちらにでも動いたときは1となるような処理を作成してください。

２．メソッド紹介と処理内容

Mathf.Clamp

public static int **Clamp** (int **value**, int **min**, int **max**);

パラメーター

価値	最小値から最大値の範囲内に制限する整数ポイント値。
分	比較する最小の整数ポイント値。
最大	比較する最大の整数ポイント値。

戻り値

int 最小値と最大値の間の int 結果。

説明

指定された値を、指定された最小整数値と最大整数値で定義された範囲内にクランプします。指定された値が最小値と最大値の範囲内にある場合は、指定された値を返します。

指定された値が最小値より小さい場合は最小値を返します。指定された値が最大値より大きい場合は最大値を返します。最小値と最大値のパラメータは両端を含みます。例えば、Clamp(10, 0, 5) は最大値として4ではなく5を返します。

＜問題＞

int型の変数hpを定義し、100を設定してください。その後ランダムで0～200の範囲でhpを減算してください。その際にhpを100～0の範囲に収めてください。

２．メソッド紹介と処理内容

Mathf.Sqrt

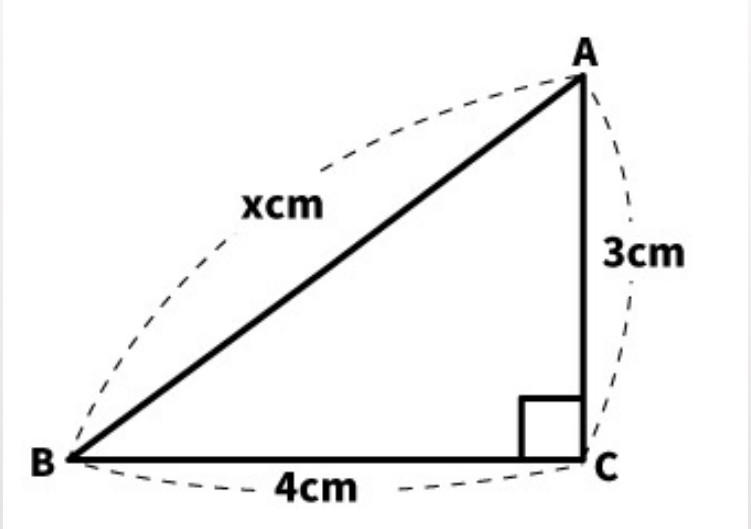
public static float **Sqrt** (float **f**);

説明

f の平方根を返します。

＜問題＞

斜辺xを求めてください。



２．メソッド紹介と処理内容

＜問題＞

上記の関数で9.0f、9.4f、9.5f、9.6fでどのような値になるか確認してみてください。

２．メソッド紹介と処理内容

＜問題＞

上記の関数で9.0f、9.4f、9.5f、9.6fでどのような値になるか確認してみてください。

２．メソッド紹介と処理内容

Mathf.Round

public static float **Round** (float **f**);

説明

f に最も近い整数を返します。

数が .5 で終わる場合はふたつの整数（偶数と奇数）の中間に位置していますが、偶数が返されます。

＜問題＞

上記の関数で0.5f、1.5f、2.5fを引数とした場合、どのような値となるでしょうか。また、それはなぜでしょうか？

２．メソッド紹介と処理内容

Mathf.Lerp

public static float **Lerp** (float **a**, float **b**, float **t**);

パラメーター

a	開始値
b	終了値
t	2つのfloatの補間値

戻り値

float 2つのfloat値の間の補間されたfloat

説明

`a` と `b` の間で `t` による線形補間します。

パラメーター `t` は [0, 1] の範囲で制限されます。

When `t` = 0 returns `a`.
When `t` = 1 return `b`.
When `t` = 0.5 returns the midpoint of `a` and `b`.

```
public class LerpLesson1 : MonoBehaviour
{
    private float lerpTime;

    // Start is called before the first frame update
    void Start()
    {
        //積算時間初期値
        lerpTime = 0;
    }

    // Update is called once per frame
    void Update()
    {
        //0から1を1秒かけて実行する
        Debug.Log(Mathf.Lerp(0,1,lerpTime));
        lerpTime += Time.deltaTime;
    }
}
```

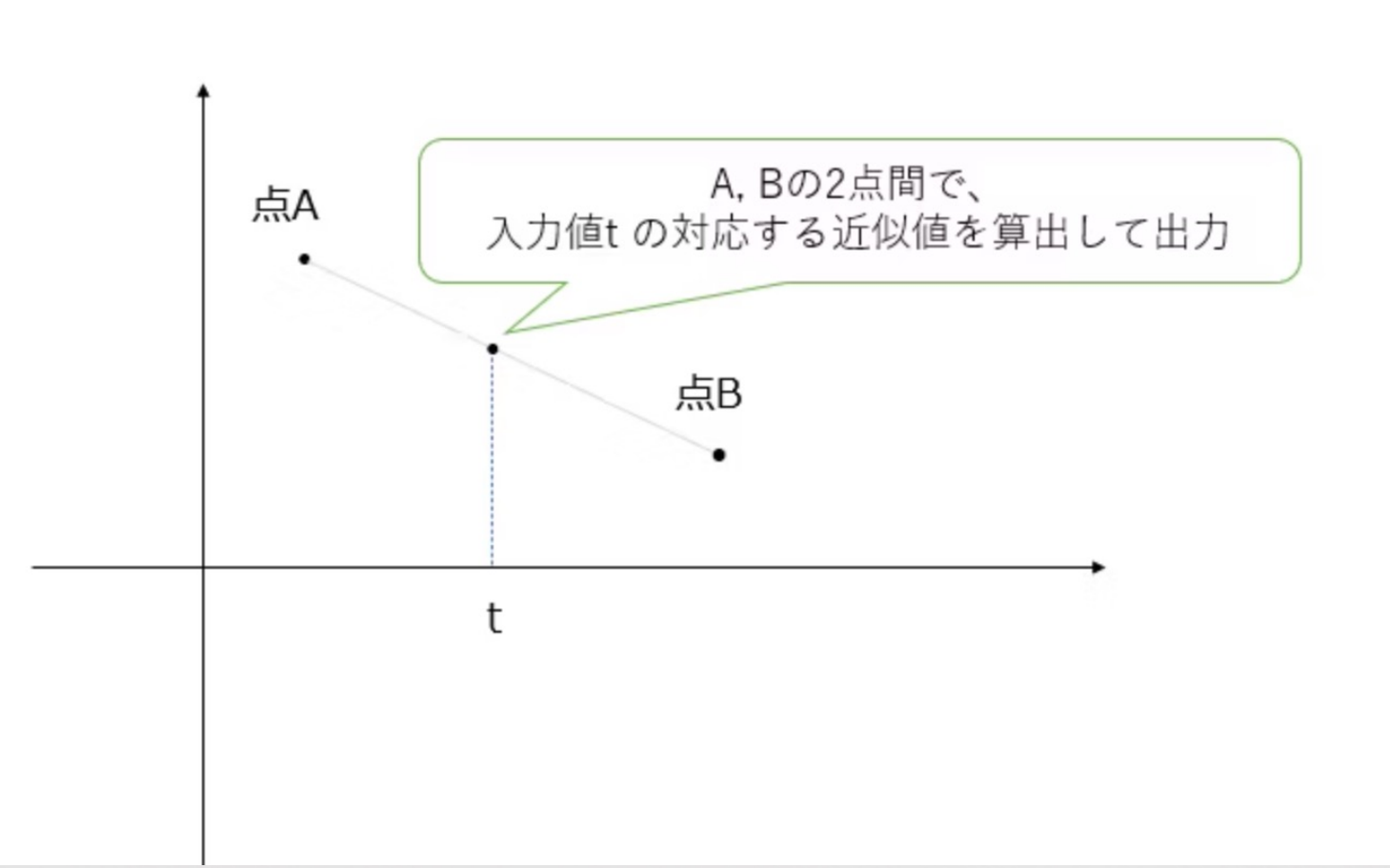
＜練習＞

線形補間のことです！！(次のページを参照)

LerpLesson1.csを作成して、どのようなことが出来るか確認してみましょう。

３．活用例(線形補間)

始点と終点等の2点が与えられた場合に、その2点の間の数値を近似的に算出する(補完する)ことを指す。



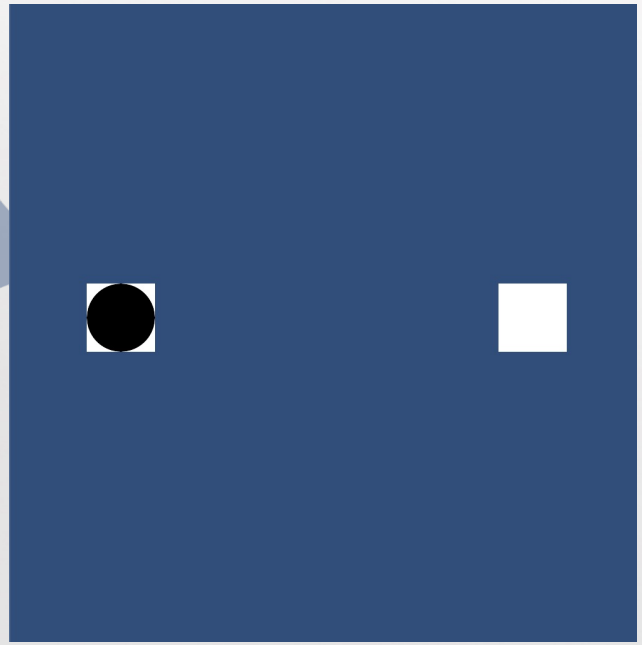
３．活用例(線形補間)

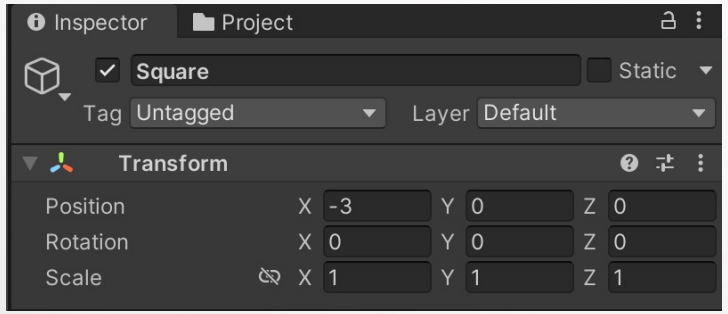
＜問題＞

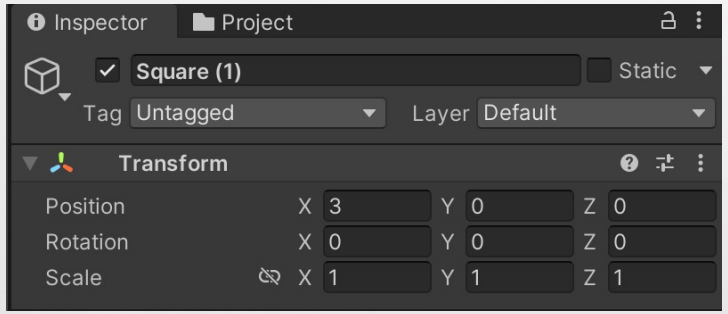
以下のようにSquareを2つ、Circleを1つ準備してください。

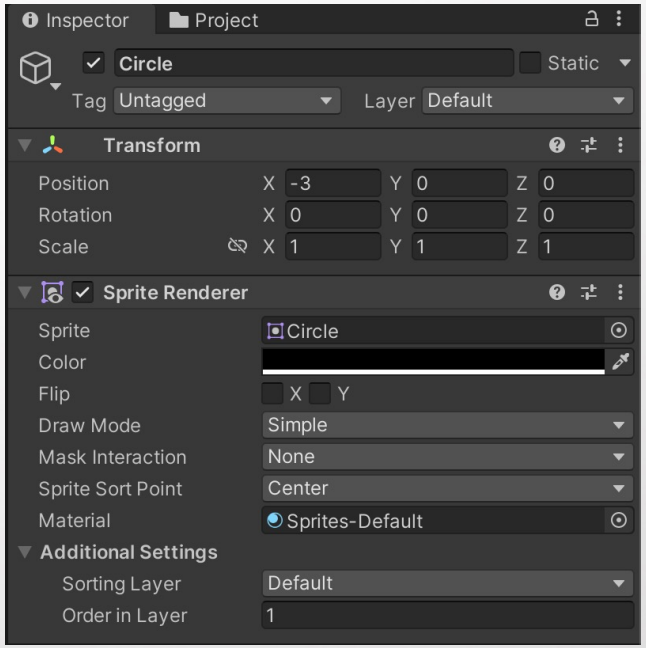
その後LerpTest1.csを作成し、黒のCieclをもう1つのSquareまで

2秒かけて移動させてみてください。









3．活用例(線形補間)

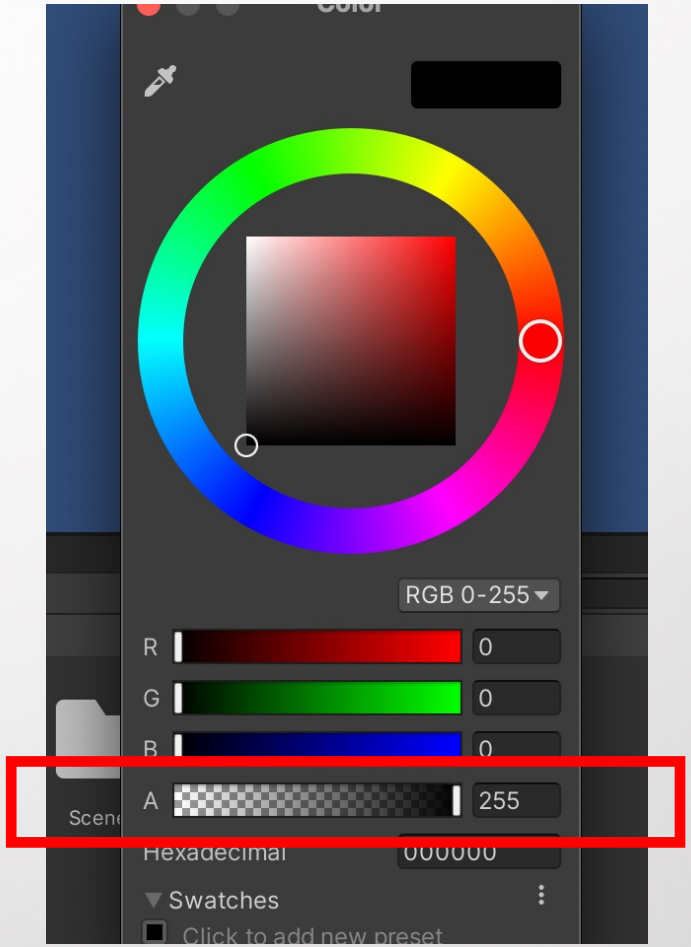
```
public class LerpTest1 : MonoBehaviour
{
    [SerializeField] private GameObject gameObj1;
    [SerializeField] private GameObject gameObj2;

    private float lerpTime;

    // Start is called before the first frame update
    void Start()
    {
        //積算時間初期値
        lerpTime = 0;
    }

    // Update is called once per frame
    void Update()
    {
        //2秒かけて移動する
        transform.position =
            new Vector2(Mathf.Lerp(gameObj1.transform.position.x, gameObj2.transform.position.x, lerpTime / 2), 0);
        lerpTime += Time.deltaTime;
    }
}
```

３．活用例(線形補間)

この線形補完は移動だけではなく色々なところで使えたりします。
例として、フェードアウトを実施していきたいと思います。

やることは、A(アルファ)値をLerpを使って切り替えていきます。
(プログラムで指定する場合は255-0ではなくて、
1-0を指定します)

LerpLesson2.csを作成し、Squareに
アタッチしてみてください。



3．活用例(線形補間)

```
public class LerpLesson2 : MonoBehaviour
{
    private SpriteRenderer spriteRender;
    private float lerpTime;
    private Color colorSprite;

    // Start is called before the first frame update
    void Start()
    {
        //積算時間初期値
        lerpTime = 0;

        //画像取得
        spriteRender = GetComponent<SpriteRenderer>();
        colorSprite = spriteRender.color;
    }

    // Update is called once per frame
    void Update()
    {
        //1秒かけて画像を透明にしていく
        colorSprite.a = Mathf.Lerp(1, 0, lerpTime);
        spriteRender.color = colorSprite;
        lerpTime += Time.deltaTime;
    }
}
```

３．活用例(線形補間)

＜問題＞

シーン開始時にかっこいいフェードインカットを作ってください。

・下準備のヒント

①UI⇨Imageを選択

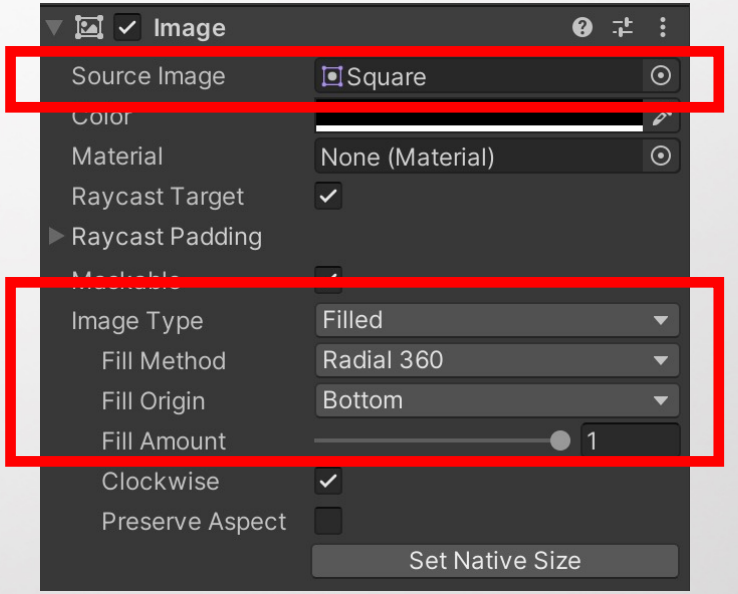
②Projectファイルで右クリック⇨Create⇨2D⇨Sprite⇨Square

③SquareをImageのSouce Imageにドラッグ

④Imageのコンポーネントを右記のように設定

⑤FillAmountを1から0にすることで実装可能

⑥LerpはMathf.Lerpを使用してください。



４．ロジックを考えよう

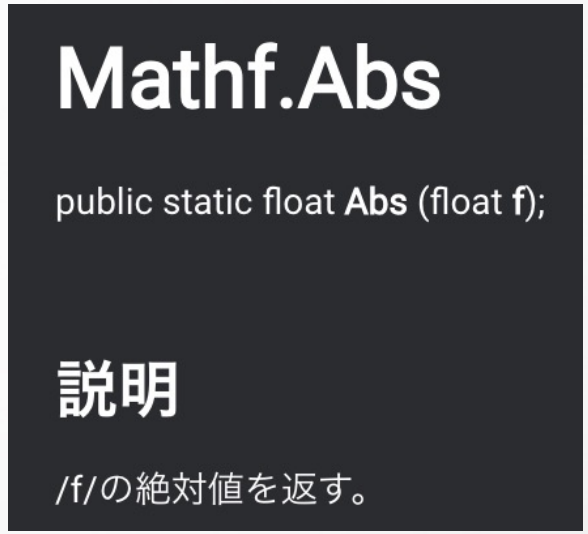
Mathfの理解を深めるため、クラスの各関数のリファレンスを読み解き、
同じ引数を受け取り、同様の結果となる戻り値を返すような、
オリジナルのMathf関数を作成してみようと思います！

MathOriginal.csのスクリプトファイルを作成してください。

４．ロジックを考えよう

一緒にMathf.Abs関数を作成していきましょう！

MathOriginalクラスにMathf.Absと同様、
public float Absで関数を作成してください。
(staticは今回は不要とします)

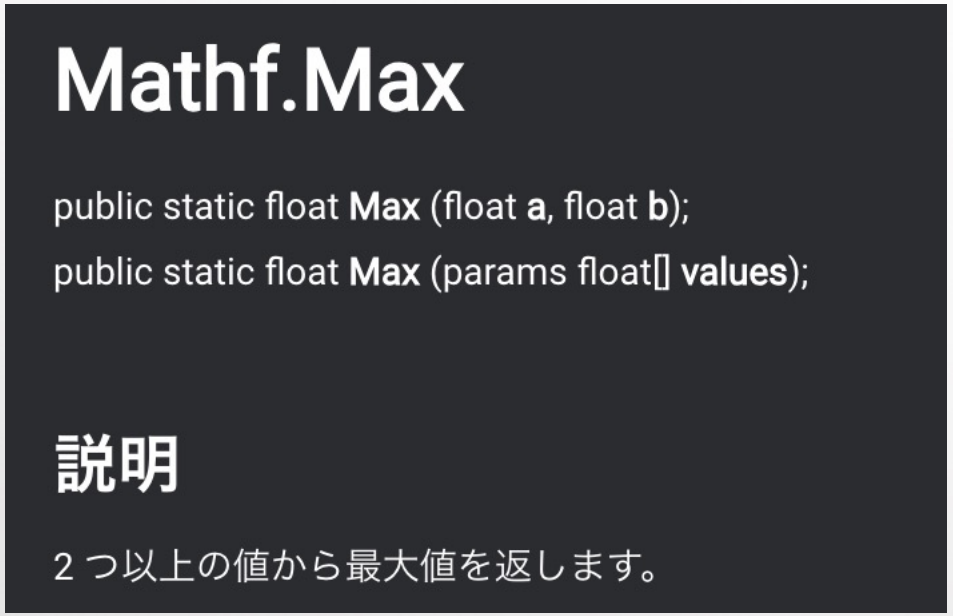


この関数は引数で受け取ったfloat fを絶対値で返すという関数になっています。
　絶対値：原点(0)からの距離を表す。例えば5の場合は5となる。
　　-5の場合でも原点(0)からは5の距離となるため、答えは5となる。
　　⇨正の数字の場合はそのまま、負の数字の場合は+側に直す
結果を戻り値で返すこと。

４．ロジックを考えよう

＜注意＞

オーバーロードという、同じ名前でパラメータの数や型が異なるメソッドが複数宣言されている箇所については、すべてのパラメータで実施すること。



この場合、float a, float bで引数を受け取った場合と、float[] valuesで受け取った場合の2パターンで実施すること。(paramsは調べてみて！)

４．ロジックを考えよう

以下のMathfの関数を実装してみてください。

(01) Mathf.Sign
(02) Mathf.Clamp
(03) Mathf.Clamp01
(04) Mathf.Max
(05) Mathf.Min
(06) Mathf.Ceil
(07) Mathf.Floor
(08) Mathf.Lerp
(09) Mathf.InverseLerp