

AngryBirds



HIGHSCORE: 47600
SCORE: 0



作成スタート

1-24 of 54 results for **2d Background**

Sort by

Popularity

View Results

24



2d Background

Free Assets

clear filters



DENVZLA ESTUDIO
2D Background

(not enough ratings) | ❤️ (171)

FREE

Add to My Assets

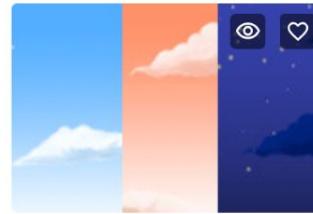


CPASTEGAME
Free 2D Cartoon Parallax ...

★★★★★ (7) | ❤️ (432)

FREE

Add to My Assets



SUPER BRUTAL ASSETS
Free 2D Adventure Beach ...

★★★★★ (41) | ❤️ (2625)

FREE

Add to My Assets



FEON
2D Background Builder - R...

(not enough ratings) | ❤️ (18)

FREE

Add to My Assets

Refine by

Hide Purchased Assets

All Categories

3D (1)

2D (49)

Characters (7)

Environments (26)

Textures & Materials (14)

Audio (1)

Tools (3)

Level Design (1)

Particles & Effects (1)

Sprite Management (1)

Pricing

Free Assets (54)



NEW RELEASE

DIGITAL MOONS
Pixel Skies DEMO Backgro...
(not enough ratings) | ❤️ (35)

FREE

Add to My Assets



BRETT GREGORY
2D Space Backgrounds
★★★★★ (9) | ❤️ (380)

FREE

Add to My Assets



BRACKEYS
Free 2D Mega Pack
★★★★★ (47) | ❤️ (3806)

FREE

Add to My Assets



DINV STUDIO
Dynamic Space Backgrou...
★★★★★ (75) | ❤️ (2730)

FREE

Add to My Assets



Painted HQ 2D Forest Medieval Background



Super Brutal Assets

★★★★★ (22) | ❤️ (1987)

FREE

⌚ 214 views in the past week

[Open in Unity](#)



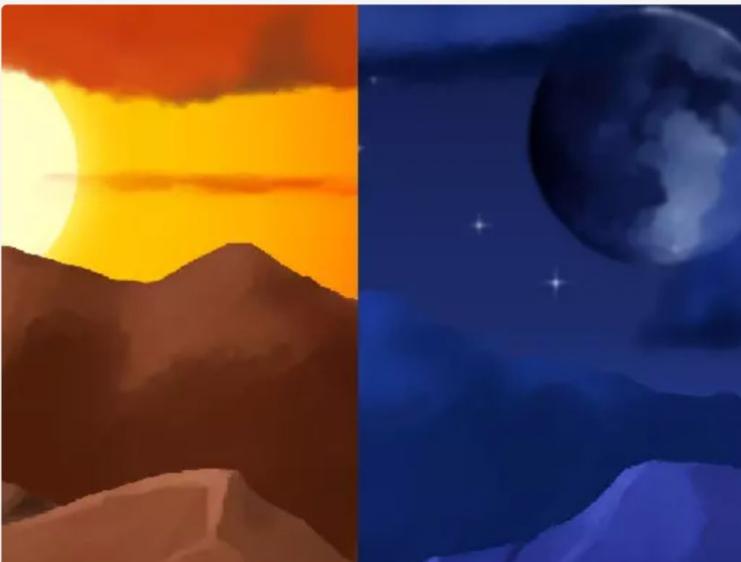
nabeelasad2007

★★★★★ 4 months ago

Good Work!

Really Good for UI based games and for testing purposes.

[Read more reviews](#)



1/5



License agreement

[Standard Unity Asset Store EULA](#)

License type

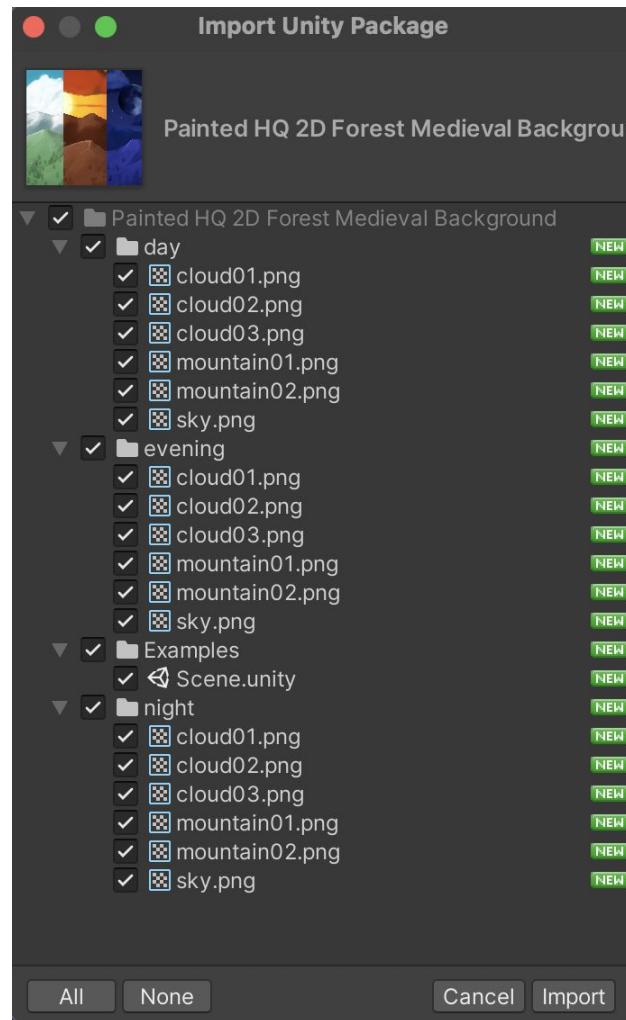
Extension Asset

File size

10.2 MB

Latest version

1.0





[pixel art](#) [Free Assets](#)



SVEN THOLE
Hero Knight - Pixel Art
★★★★★ (46) | ❤ (2138)
FREE [Add to My Assets](#)



CAINOS
Pixel Art Platformer - Villa...
★★★★★ (47) | ❤ (4653)
FREE [Add to My Assets](#)



SVEN THOLE
Bandits - Pixel Art
★★★★★ (45) | ❤ (3801)
FREE [Add to My Assets](#)



PURCHASED
CAINOS
Pixel Art Top Down - Basic
★★★★★ (38) | ❤ (2538)
FREE



EDER
Free Pixel Art Forest
★★★★★ (14) | ❤ (2409)
FREE [Add to My Assets](#)



SVEN THOLE
Prototype Hero Demo - Pi...
★★★★★ (9) | ❤ (736)
FREE [Add to My Assets](#)



BLACKSPIRE
Medieval pixel art asset F...
★★★★☆ (22) | ❤ (3334)
FREE [Add to My Assets](#)



POLYMESHORLD
FREE Pixel Art Kit
(not enough ratings) | ❤ (248)
FREE [Add to My Assets](#)

Refine by

Hide Purchased Assets

All Categories

- 3D (3)
- 2D (139)
 - Characters (55)
 - Environments (47)
 - Fonts (3)
 - GUI (11)
 - Textures & Materials (18)
- Audio (2)
- Templates (4)
- Tools (9)
 - Camera (2)
 - GUI (1)
 - Modeling (1)
 - Painting (3)
 - Sprite Management (2)
- VFX (1)





•



Pixel Art Platformer - Village Props



Cainos

★★★★★ (47) | ❤ (4653)

FREE

🕒 1584 views in the past

week

[Open in Unity](#)



ktorres_65

★★★★★ 15 days ago

Excellent Selection!

Create Empty

2D Object >

3D Object >

Effects >

Light >

Audio >

Video >

UI >

UI Toolkit >

Camera

Sprites >

Physics >

Tilemap >

Sprite Shape >

Pixel Perfect Camera

Sprite Mask

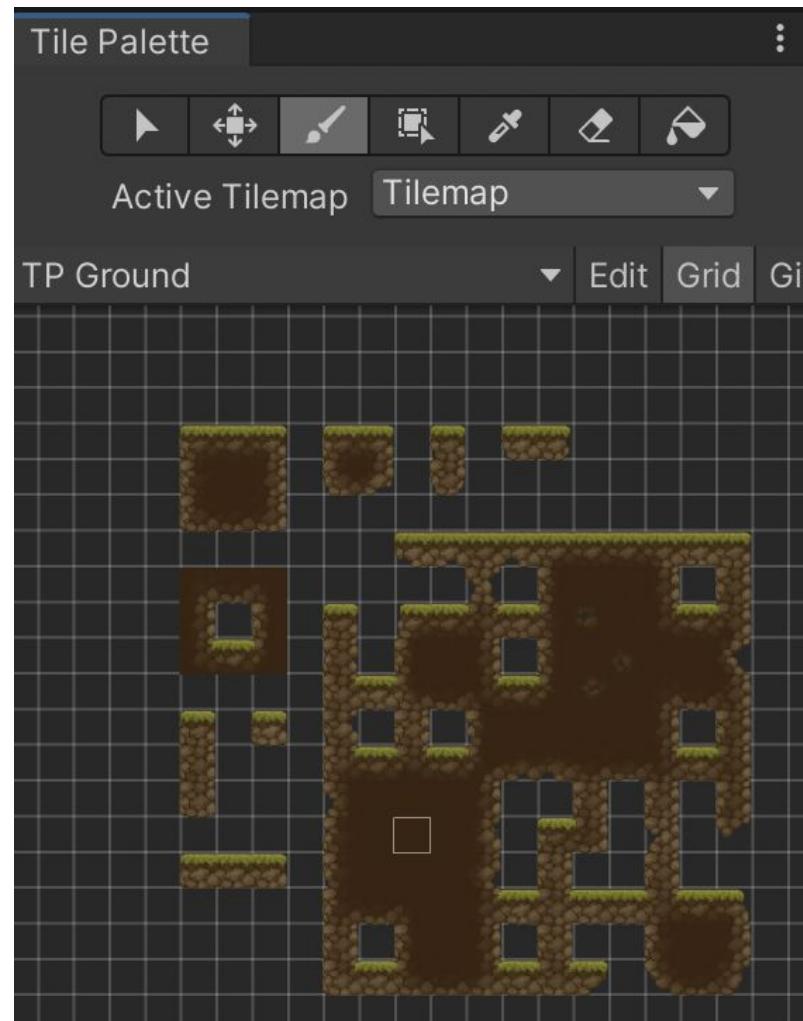
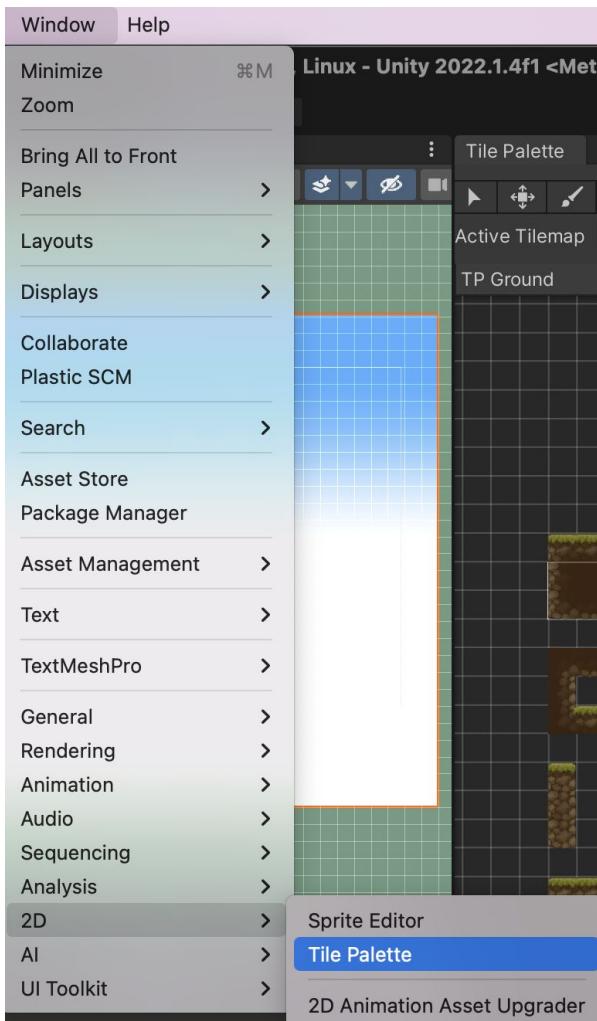
Rectangular

Hexagonal - Pointed-Top

Hexagonal - Flat-Top

Isometric

Isometric Z as Y

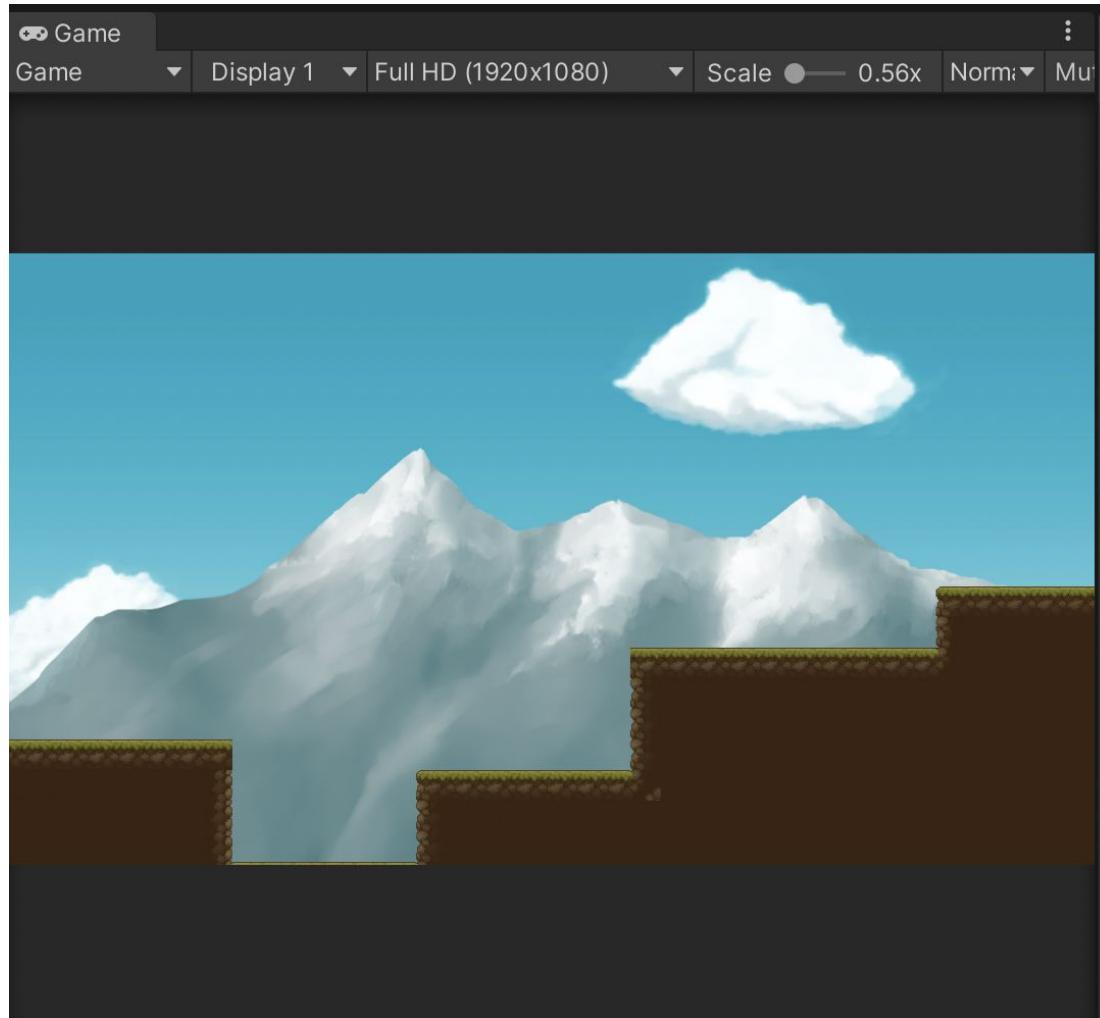


Assets > FromAssetStore > Cainos > Pixel Art Platformer - Village Props > **Tileset Palette**

 TP Ground

 TP Ground

 Palette Settings



coll

Search

Box Collider 2D

Capsule Collider

Capsule Collider 2D

Circle Collider 2D

Composite Collider 2D

Custom Collider 2D

Edge Collider 2D

Mesh Collider

Polygon Collider 2D

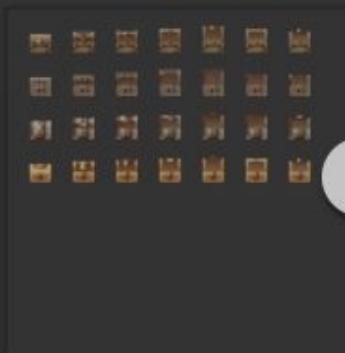
Sphere Collider

Terrain Collider

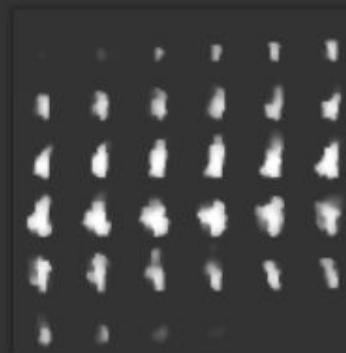
Tilemap Collider 2D



TX Background Gr...



TX Chest Animati...



TX Particle Flame



TX Tileset Ground



TX Village Props

Game

Game

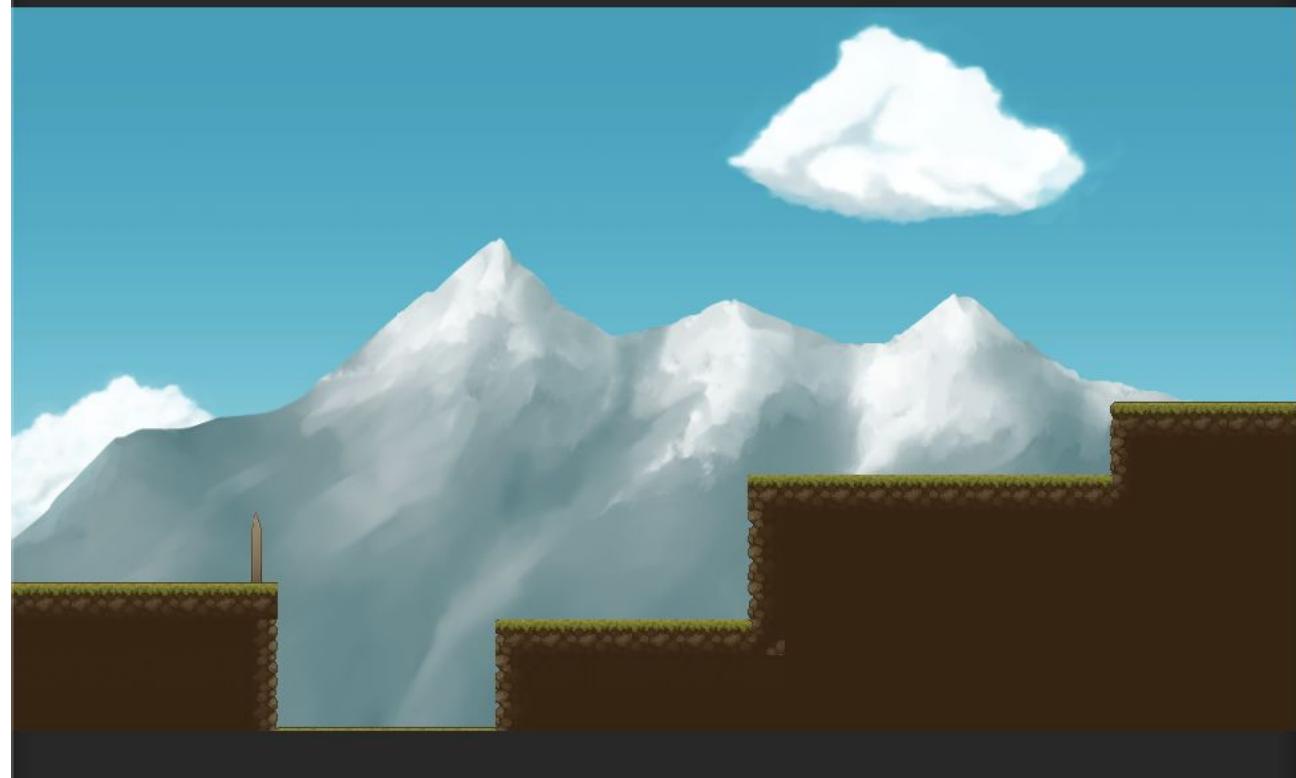
Display 1

16:9 Aspect

Scale 1x

Norm

TX Village Props ...





Animal Cube (Cat Series) - 2D Asset



Hisa Games

(not enough ratings)

18

FREE

29 views in the past week

Add to My Assets



License
agreement

Standard Unity Asset Store
EULA

License type

Extension Asset

File size

3.6 MB

Latest version

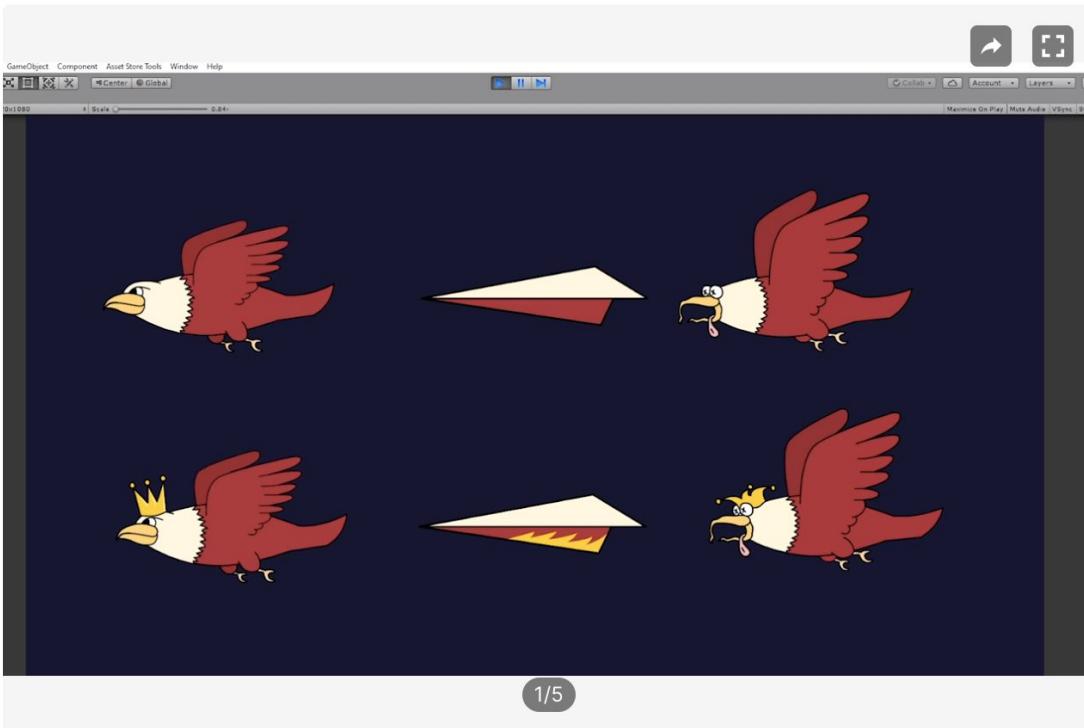
1.0

Latest release date

Dec 23, 2021

Supported Unity
versions

2020.3.14 or
higher



Cartoon Eagle 2D

 SR Studios Kerala

(not enough ratings) |  (70)

FREE

 **77 views** in the past week

[Add to My Assets](#)



License agreement [Standard Unity Asset Store EULA](#)

License type [Extension Asset](#)

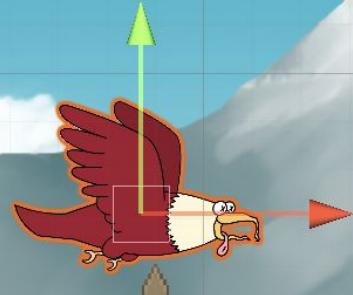
File size 1.5 MB

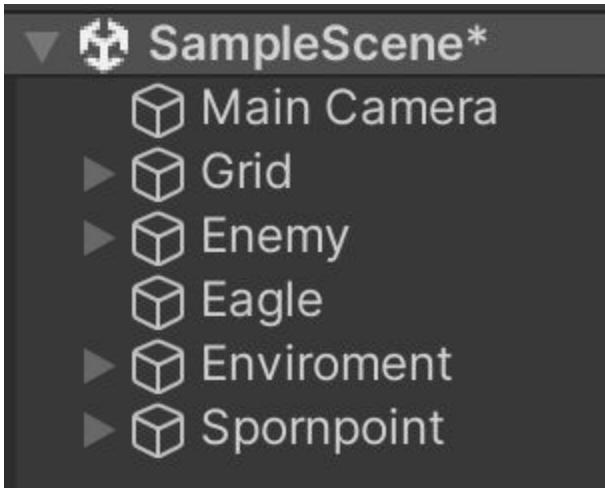
Latest version 1.0

Latest release date Jun 15, 2021

Supported Unity versions 2019.2.21 or higher

Support [Visit site](#)





Eagle Static

Tag: Untagged Layer: Default

Transform

Position	X: -11.44	Y: -3.04	Z: 0
Rotation	X: 0	Y: 0	Z: 0
Scale	X: -0.6	Y: 0.6	Z: 0.6

Sprite Renderer

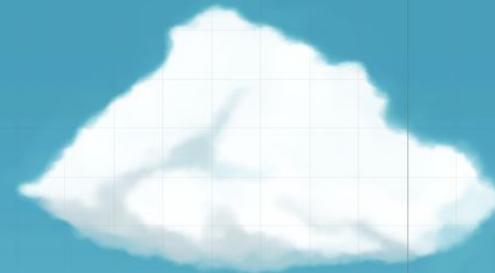
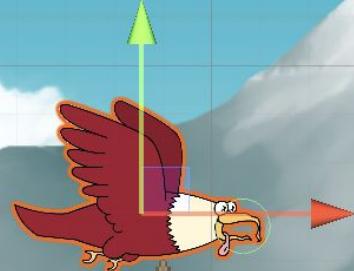
Sprite	Eagle_Died
Color	[Color swatch]
Flip	X: [checkbox], Y: [checkbox]
Draw Mode	Simple
Mask Interaction	None
Sprite Sort Point	Center
Material	Sprites-Default

Additional Settings

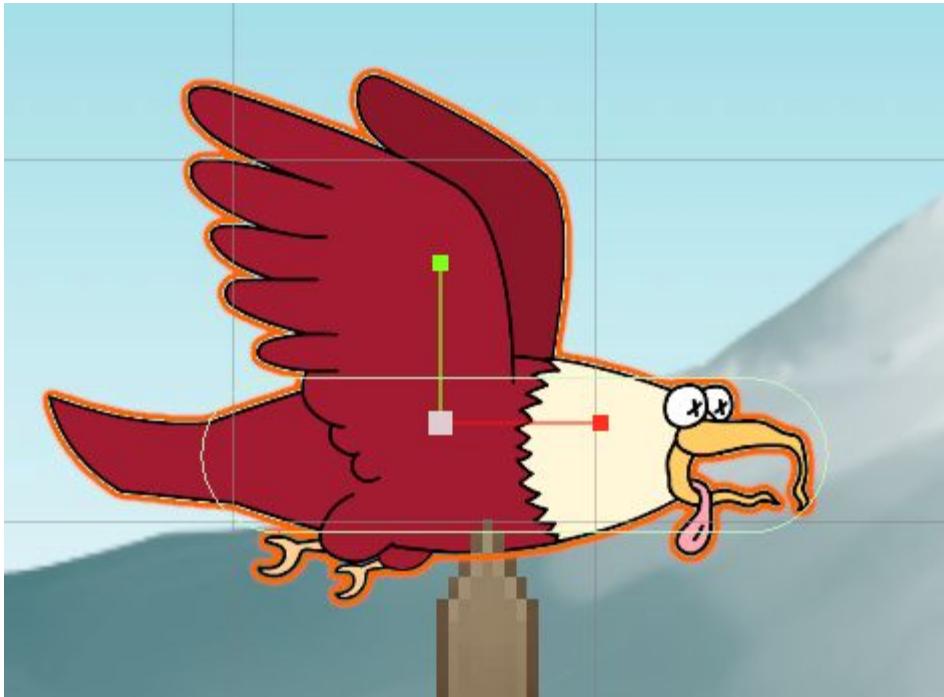
Sorting Layer	Default
Order in Layer	0

Capsule Collider 2D

Edit Collider	[Edit Collider button]
Material	None (Physics Material 2D)
Is Trigger	[checkbox]
Used By Effector	[checkbox]
Offset	X: -3.45 Y: -0.42
Size	X: 1.59 Y: 1.85



キャラを操作するスクリプト



Eagle Static Tag Untagged Layer Default

Transform

Position	X -11.44	Y -3.04	Z 0
Rotation	X 0	Y 0	Z 0
Scale	X -0.6	Y 0.6	Z 0.6

Sprite Renderer

Capsule Collider 2D

Edit Collider

Material	None (Physics Material 2D)	
Is Trigger	<input type="checkbox"/>	
Used By Effector	<input type="checkbox"/>	
Offset	X -3.45	Y -0.42
Size	X 1.59	Y 1.85
Direction	Horizontal	

Info

Attached Body	None (Rigidbody 2D)
Friction	0.4
Bounciness	0
Shape Count	1
Bounds	<input type="text" value="ari"/> Search Z 0

Rigidbody **Rigidbody 2D**

Contacts

The image shows the Unity interface with the Project panel on the left and the Inspector panel on the right.

Project Panel:

- Folder
- C# Script
- 2D
- Visual Scripting
- Shader
- Shader Variant Collection
- Testing
- Playables
- Assembly Definition
- Assembly Definition Reference
- TextMeshPro
- Text

Inspector Panel:

A Physics Material 2D asset named "Eagle (Physics Material 2D)" is selected. The Inspector shows the following settings:

	Eagle (Physics Material 2D)
Friction	0.5
Bounciness	0.5

キャラの位置をどう決める？

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

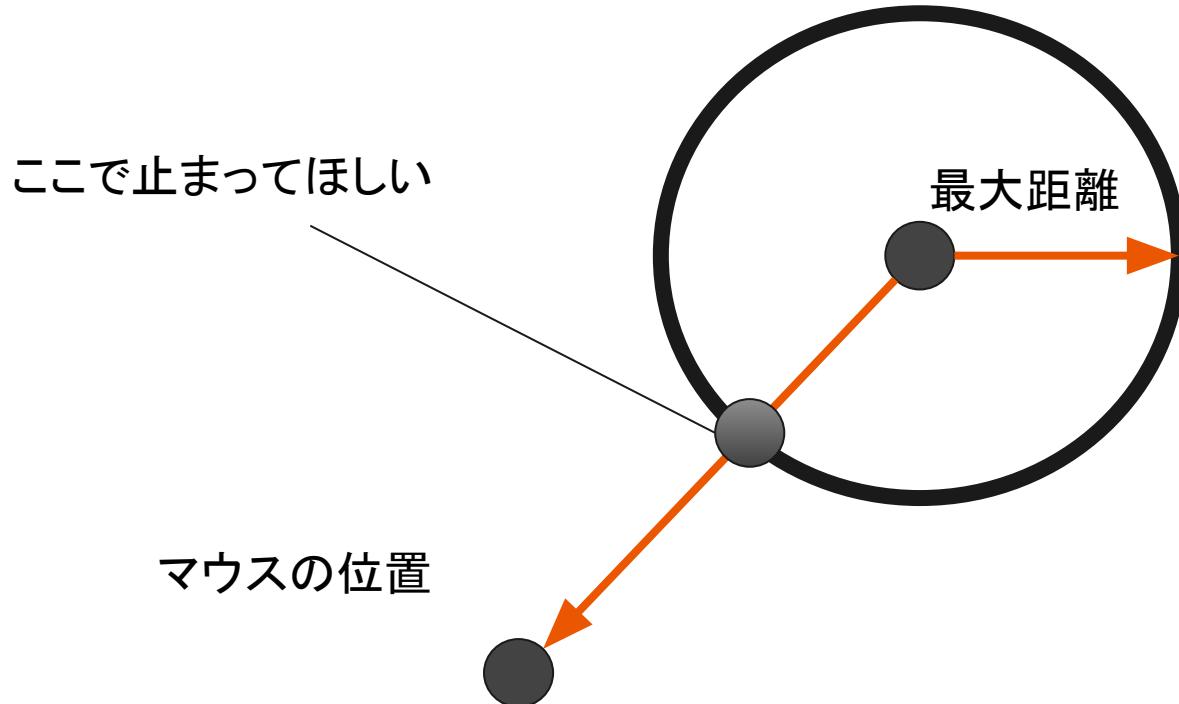
public class EagleCtrl: MonoBehaviour
{
    Rigidbody2D _rb2d;

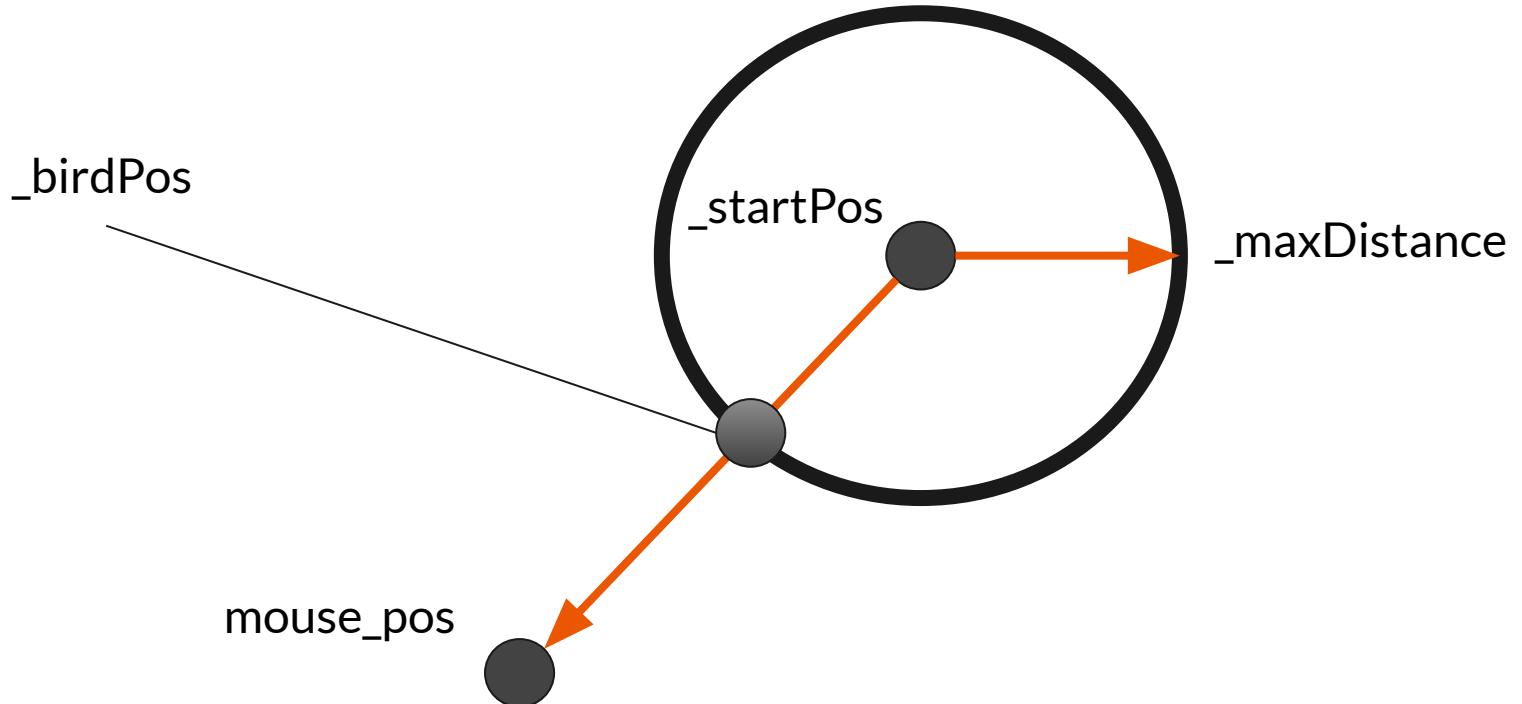
    void Start()
    {
        TryGetComponent( out _rb2d);
        _rb2d.isKinematic = true; //物理演算を切っておく
    }

    private void OnMouseDown()
    {
        Vector2 mouse_pos = Camera.main.ScreenToWorldPoint(Input.mousePosition); //マウスの位置
        transform.position = mouse_pos; //キャラの位置
    }
}
```

実行、操作キャラがマウスに追従することを確認

操作キャラに制限を加える





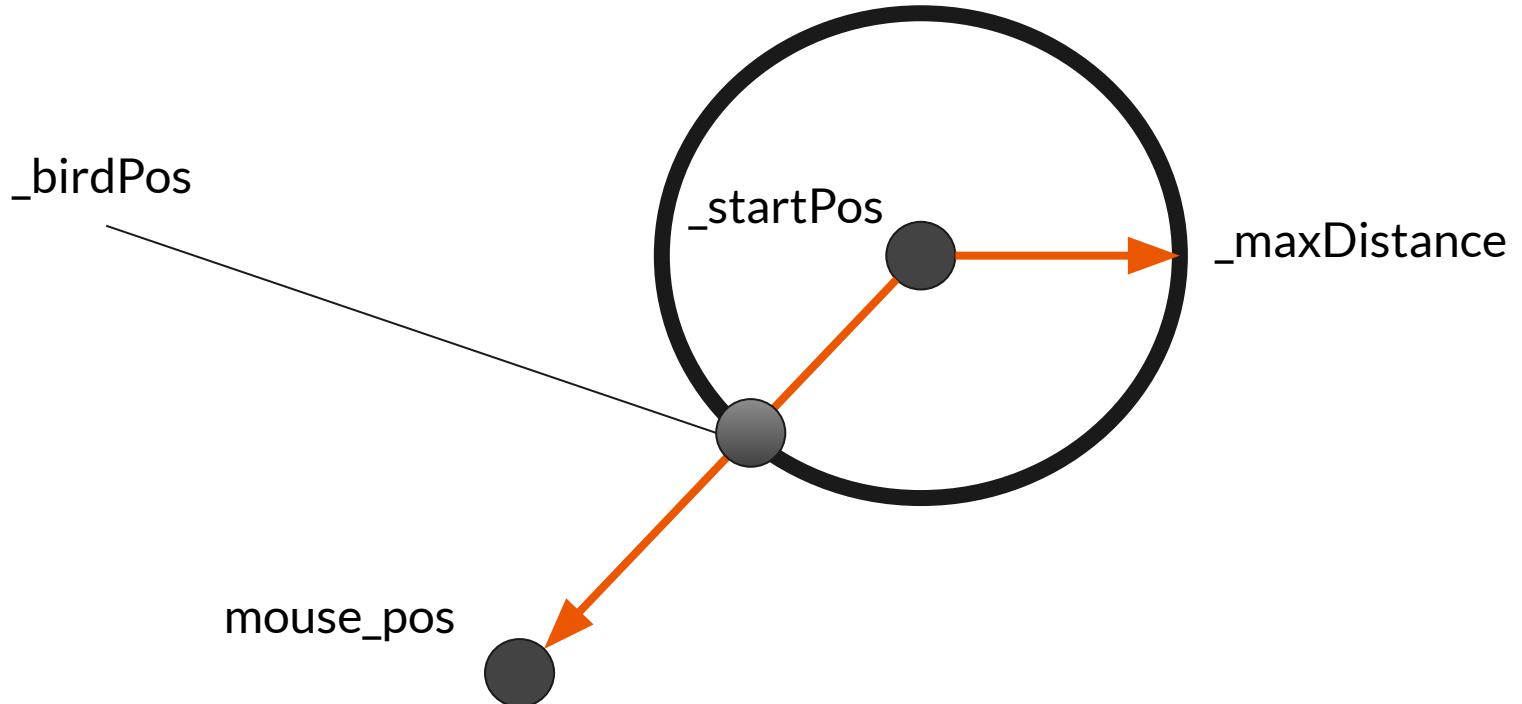
```
public class EagleCtrl: MonoBehaviour
{
    Rigidbody2D _rb2d;
    Vector2 _startPos, _birdPos;

    void Start()
    {
        _startPos = Vector2.zero; //初期化
        _birdPos = Vector2.zero; //初期化

        TryGetComponent( out _rb2d);
        _rb2d.isKinematic = true; //物理演算を切っておく

        _startPos = transform.position; //開始位置
    }

    private void OnMouseDrag()
    {
        Vector2 mouse_pos = Camera.main.ScreenToWorldPoint(Input.mousePosition); //マウスの位置
        _birdPos = mouse_pos;
        Vector2 flyDirection = _startPos - mouse_pos; //マウスの位置から見た開始位置のベクトル
        float sToM_Distance = flyDirection.magnitude; //マウスとキャラの距離
    }
}
```



```
public class EagleCtrl: MonoBehaviour
{
    Rigidbody2D _rb2d;
    Vector2 _startPos, _birdPos;
    float _maxDistance = 3f; //引張る最大距離
    void Start()
    {
        _startPos = Vector2.zero; //初期化
        _birdPos = Vector2.zero; //初期化
        TryGetComponent( out _rb2d);
        _rb2d.isKinematic = true; //物理演算を切っておく
        _startPos = transform.position; //開始位置
    }

    private void OnMouseDown()
    {
        Vector2 mouse_pos = Camera.main.ScreenToWorldPoint(Input.mousePosition); //マウスの位置
        _birdPos = mouse_pos;
        Vector2 flyDirection = _startPos - mouse_pos; //マウスの位置から見た開始位置のベクトル
        float sToM_Distance = flyDirection.magnitude; //マウスとキャラの距離
        if (sToM_Distance > _maxDistance)
        {
            Vector2 _flyDirectNorm = (-1) * flyDirection.normalized; //正規化して方向だけ求めて
            _birdPos = _startPos + _maxDistance * _flyDirectNorm; //最大距離まで伸ばして、開始地点に足す
        }
        transform.position = _birdPos;
    }
}
```

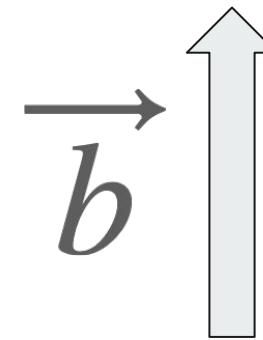
実行、操作キャラが最大距離までしか移動しないことを確認

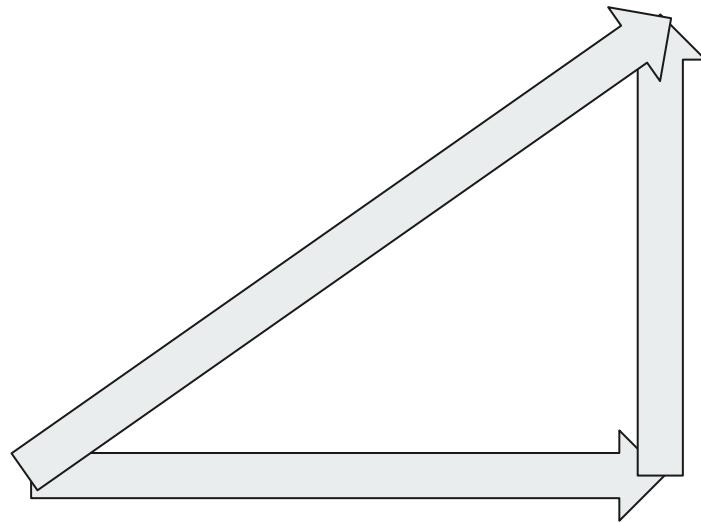
ベクトル

UFOキャッチャー



開始位置





三平方の定理

ルート

の存在は知ってるが

ルート19

を

3ルート1

と言ってしまうレベル

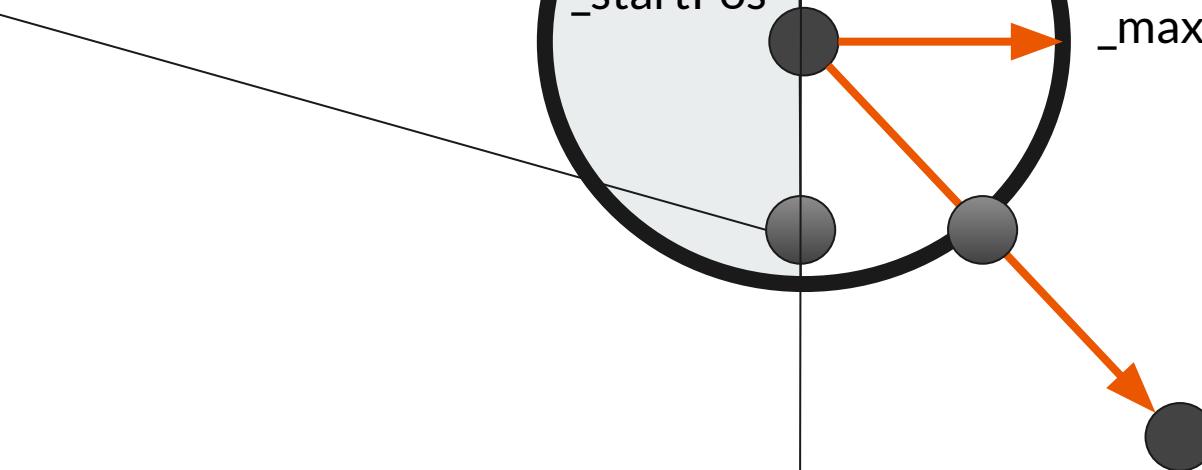
この線より左でのみ動いてほしい

_birdPos

_startPos

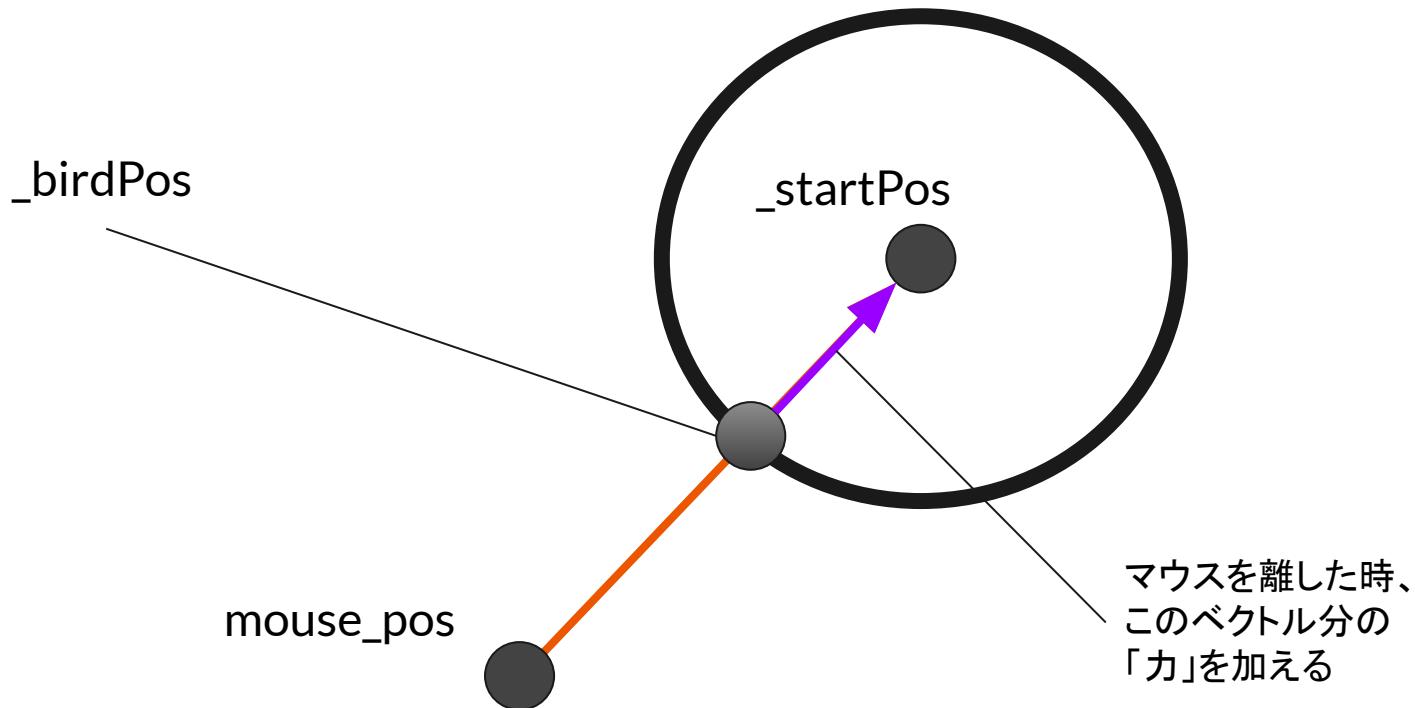
_maxDistance

mouse_pos



```
private void OnMouseDrag()
{
    Vector2 mouse_pos = Camera.main.ScreenToWorldPoint(Input.mousePosition); //マウスの位置
    _birdPos = mouse_pos;
    Vector2 flyDirection = _startPos - mouse_pos; //マウスの位置から見た開始位置のベクトル
    float sToM_Distance = flyDirection.magnitude; //マウスとキャラの距離
    if (sToM_Distance > _maxDistance)
    {
        Vector2 _flyDirectNorm = (-1) * flyDirection.normalized; //正規化して方向だけ求めて
        _birdPos = _startPos + _maxDistance * _flyDirectNorm; //最大距離まで伸ばして、開始地点に足す
    }
    if (_startPos.x < _birdPos.x) _birdPos.x = _startPos.x; //開始位置より右に行かないようにする
    transform.position = _birdPos;
}
```

操作キャラを飛ばす



Vector2 _flyDirection;//飛ばす方向

```
private void OnMouseDown()//マウスをドラッグしてる時
{
    Vector2 mouse_pos = Camera.main.ScreenToWorldPoint(Input.mousePosition); //マウスの位置
    _birdPos = mouse_pos;
    _flyDirection = _startPos - mouse_pos; //マウスの位置から見た開始位置のベクトル
    float sToM_Distance = _flyDirection.magnitude; //マウスとキャラの距離
    if (sToM_Distance > _maxDistance)
    {
        Vector2 _flyDirectNorm = (-1) * _flyDirection.normalized; //正規化して方向だけ求めて
        _birdPos = _startPos + _maxDistance * _flyDirectNorm; //最大距離まで伸ばして、開始地点に足す
    }
    if (_startPos.x < _birdPos.x) _birdPos.x = _startPos.x; //開始位置より右に行かないようにする
    transform.position = _birdPos;
}
```

```
float _flyForce = 5f;
private void OnMouseUp()
{
    _rb2d.isKinematic = false; //物理演算を行う
    _rb2d.AddForce(_flyForce * _flyDirection, ForceMode2D.Impulse); //飛ばした瞬間だけ外力を加える
}
```

実行、操作キャラが飛ばせることを確認

飛ばした直後から当たり判定が起こり、キャラが止まってしまう

解決策例：

飛ばす前にColliderの設定を「isTrigger」をtrueにしておき、
手を離した時かつ、開始地点より右にいった時に「isTrigger」をfalseにする

```
Rigidbody2D _rb2d;  
Collider2D _c2d;  
Vector2 _startPos, _birdPos; // 開始位置、鳥の位置  
Vector2 _flyDirection; // 飛ばす方向  
float _maxDistance = 3f; // 引っ張る最大距離  
bool _isFlited; // 飛ばしたらtrue
```

```
void Start()
{
    _startPos = Vector2.zero; //初期化
    _birdPos = Vector2.zero; //初期化
    TryGetComponent( out _rb2d);
    TryGetComponent(out _c2d);
    _rb2d.isKinematic = true; //物理演算を切っておく
    _c2d.isTrigger = true; //トリガー判定をオンにしておく
    _startPos = transform.position; //開始位置
    _isFlited = false; //飛ばす前
}
```

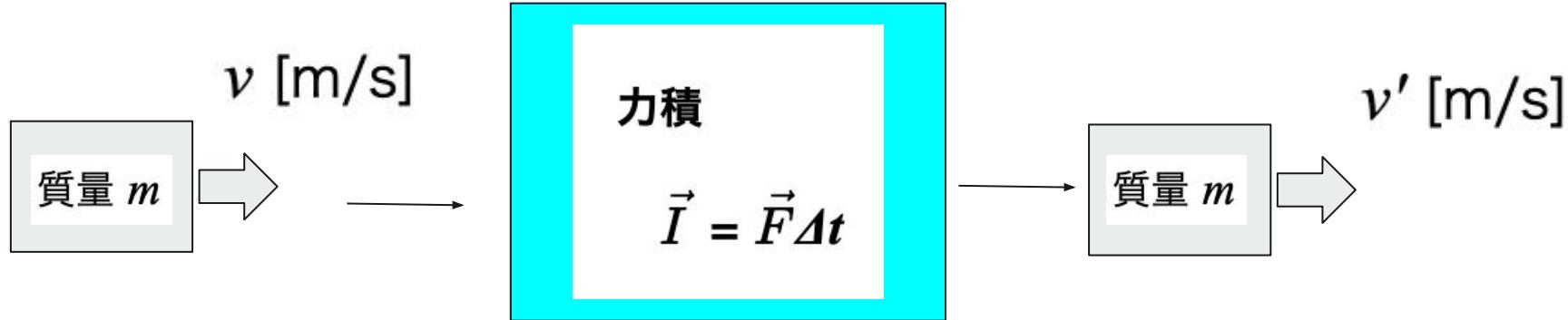
```
private void OnMouseDown()//マウスをクリックした時
{
    if (_isFlied) return;//飛ばしたら 操作できないようにする
    Vector2 mouse_pos = Camera.main.ScreenToWorldPoint(Input.mousePosition); //マウスの位置
    _birdPos = mouse_pos;
    _flyDirection = _startPos - mouse_pos; //マウスの位置から見た開始位置のベクトル
    float sToM_Distance = _flyDirection.magnitude; //マウスとキャラの距離
    if (sToM_Distance > _maxDistance)
    {
        Vector2 _flyDirectNorm = (-1) * _flyDirection.normalized; //正規化して方向だけ求めて
        _birdPos = _startPos + _maxDistance * _flyDirectNorm; //最大距離まで伸ばして、開始地点に足す
    }
    if (_startPos.x < _birdPos.x) _birdPos.x = _startPos.x; //開始位置より右に行かないようにする
    transform.position = _birdPos;
}
```

```
float _flyForce = 5f;
private void OnMouseUp()
{
    if (_isFlid) return;//飛ばしたなら 操作できないようにする
    _isFlid = true;//「飛ばした」状態にする
    _rb2d.isKinematic = false;//物理演算を行つ
    _rb2d.AddForce(_flyForce * _flyDirection, ForceMode2D.Impulse);//飛ばした瞬間だけ外力を加える
}

private void Update()
{
    if (_startPos.x < transform.position.x && _isFlid)//現在地が開始地点より右に行き、かつ「飛ばした」状態なら
    {
        _c2d.isTrigger = false;
    }
}
```

ちなみに

「ForceMode Impulse」モードとは「力積を加える計算のこと」

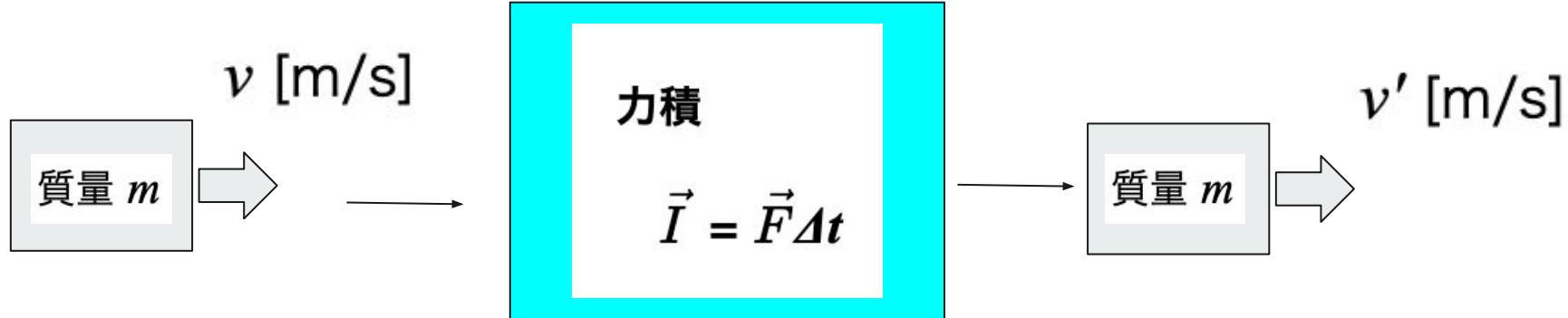


物体の運動量の変化は、その間に受けた力積に等しい

$$m\vec{v}' - m\vec{v} = \vec{F}\Delta t$$

ちなみに

「ForceMode Impulse」モードとは「力積を加える計算のこと」



物体の運動量の変化は、その間に受けた力積に等しい

$$m\vec{v}' - m\vec{v} = \vec{F}\Delta t$$

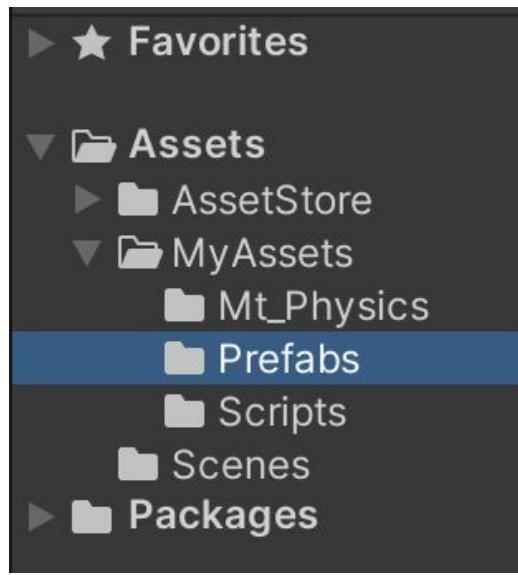
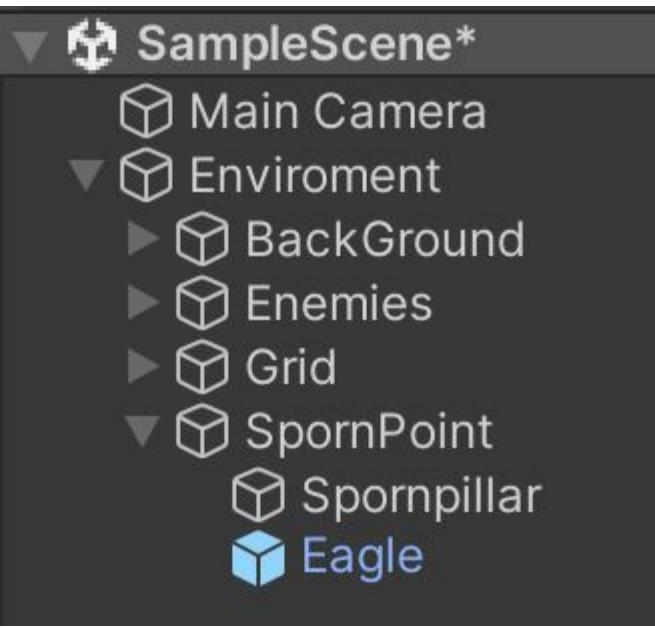
m : `_rigidbody.mass`

v : `AddForce`実行前の`_rigidbody.velocity`

v' : `AddForce`実行後の`_rigidbody.velocity`

$|$: `Vector3`の内容

何度も飛ばせるように
Prefabを用意する
(どのシーンからでも
取得できるよう「シングルトン」を使用)



Assets > MyAssets > Prefabs



Eagle

```
// 「同一のゲームオブジェクトは常に1つだけにする」
using UnityEngine;
using System;

public abstract class SingletonMonoBehaviour<T> : MonoBehaviour where T : MonoBehaviour
{
    private static T instance;
    public static T Instance
    {
        get
        {
            if (instance == null)//インスタンスがまだ作られていない
            {
                Type t = typeof(T);
                instance = (T)FindObjectOfType(t); //全オブジェクトを探索、名前が一致するクラスがあった場合は取得する
                if (instance == null)
                {
                    Debug.LogError(t + " をアタッチしているGameObjectはありません");
                }
            }
            return instance;
        }
    }
}
```

```
virtual protected void Awake()
{
    CheckInstance(); // 他のゲームオブジェクトにスクリプトがアタッチされているか調べる
    DontDestroyOnLoad(this.gameObject);
}

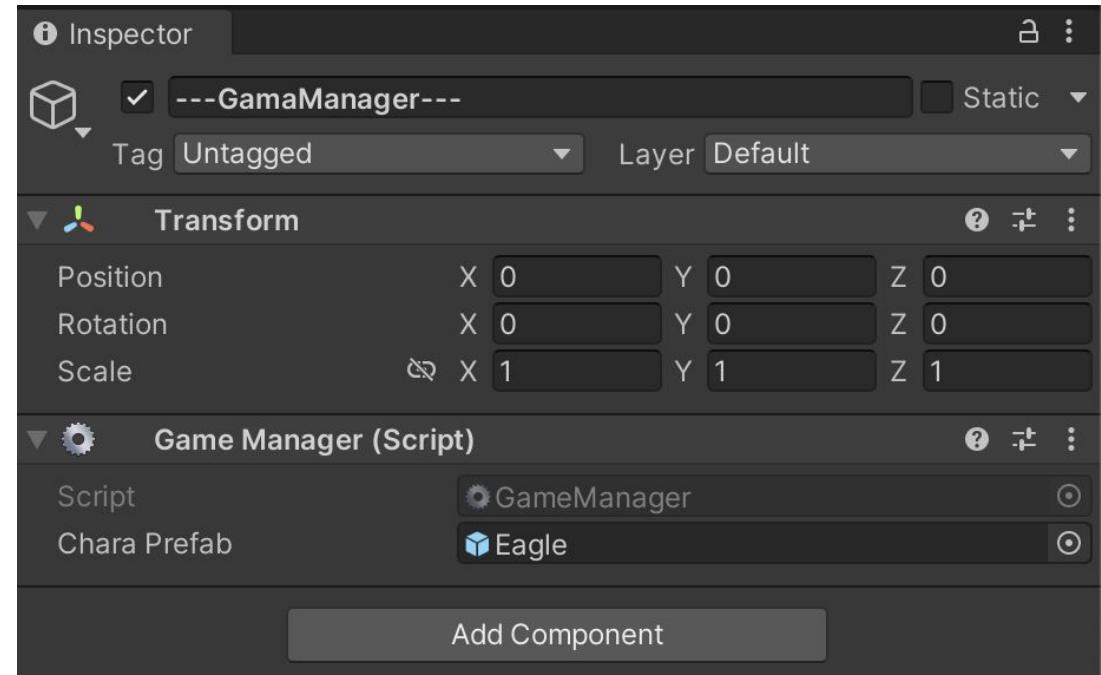
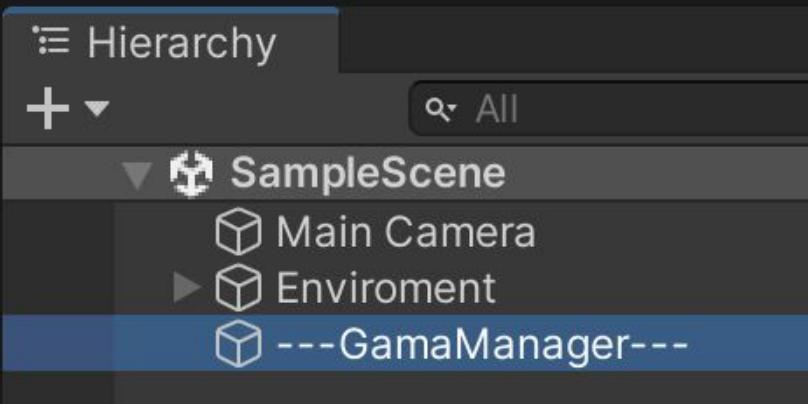
protected bool CheckInstance()
{
    if (instance == null) // インスタンスがまだ作られていない
    {
        instance = this as T;
        return true;
    }
    else if (Instance == this)
    {
        return true;
    }
    Destroy(this.gameObject); // 既に同一名のクラスが存在していた場合は破棄する。
    return false;
}
```

```
using System.Collections;
using System.Collections.Generic;
using UnityEditor.Timeline.Actions;
using UnityEngine;

//SingletonMonoBehaviourをGameManagerクラスにして継承
//GameManagerクラスをstaticに持つ
public class GameManager : SingletonMonoBehaviour<GameManager>
{
    [SerializeField] GameObject _charaPrefab;
    public GameObject CharaPrefab => _charaPrefab;

    Vector2 _startPosition = new Vector2(-5.43f, -0.73f);

    public void CharaCreate() {
        Debug.Log("キャラ作れ");
        Instantiate(_charaPrefab, _startPosition, Quaternion.identity);
    }
}
```



UniTaskを使用

Window Help

Minimize ⌘ M

Zoom

Bring All to Front

Panels >

Layouts >

Displays >

UniTask Tracker

Collaborate

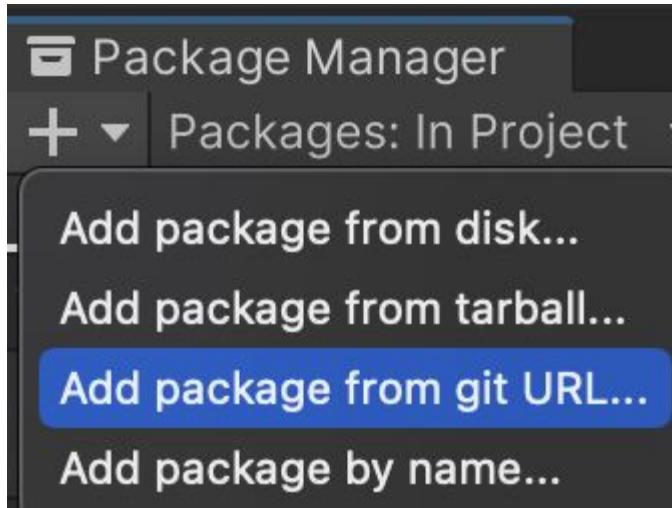
Plastic SCM

Search >

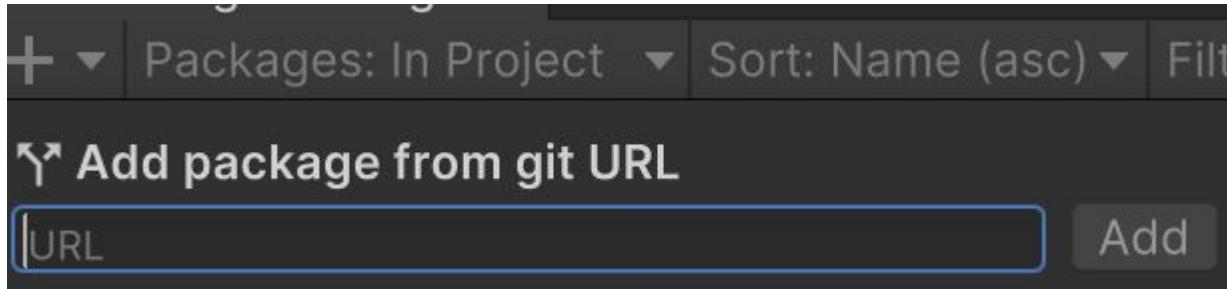
Asset Store

Package Manager

Asset Management >



<https://github.com/Cysharp/UniTask.git?path=src/UniTask/Assets/Plugins/UniTask>



EagleCtrl.cs

```
using UnityEngine;
using Cysharp.Threading.Tasks;
using System;
```

EagleCtrl.cs

```
private async UniTask NewCharacterCreate()
{
    float _createTime = 2f;
    await WaitSec(_createTime);
    Debug.Log("キャラ作成");
    GameManager.Instance.CharaCreate();
}

// ?秒待機するUniTask
async UniTask WaitSec(float sec)
{
    await UniTask.Delay(TimeSpan.FromSeconds(sec));
}
```

EagleCtrl.cs

```
float _flyForce = 5f;
private async void OnMouseUp()
{
    if (_isFlited) return; // 飛ばしたなら 操作できないようにする
    _isFlited = true; // 「飛ばした」状態にする
    _rb2d.isKinematic = false; // 物理演算を行う
    _rb2d.AddForce(_flyForce * _flyDirection, ForceMode2D.Impulse); // 飛ばした瞬間だけ外力を加える
    await NewCaracterCreate();
}
```

EagleCtrl.cs

```
private void OnCollisionEnter2D(Collision2D collision)
{
    if (!collision.gameObject.CompareTag("Player")){
        float _destroyTime = 1f;
        Destroy(gameObject, _destroyTime);
    }
}
```

飛ばす軌跡を作成

2D Object >

3D Object >

Effects >

Light >

Audio >

Video >

UI >

Sprites >

Physics >

Tilemap >

Sprite Shape >

Pixel Perfect Camera

Sprite Mask

Square

Circle

Capsule

Isometric Diamond

Hexagon Flat-Top

Hexagon Pointed-Top

9-Sliced

Assets > MyAssets > Prefabs



Eagle



OrbitCircle

▼ SampleScene*

Main Camera

Grid

Enemy

Enviroment

Spornpoint

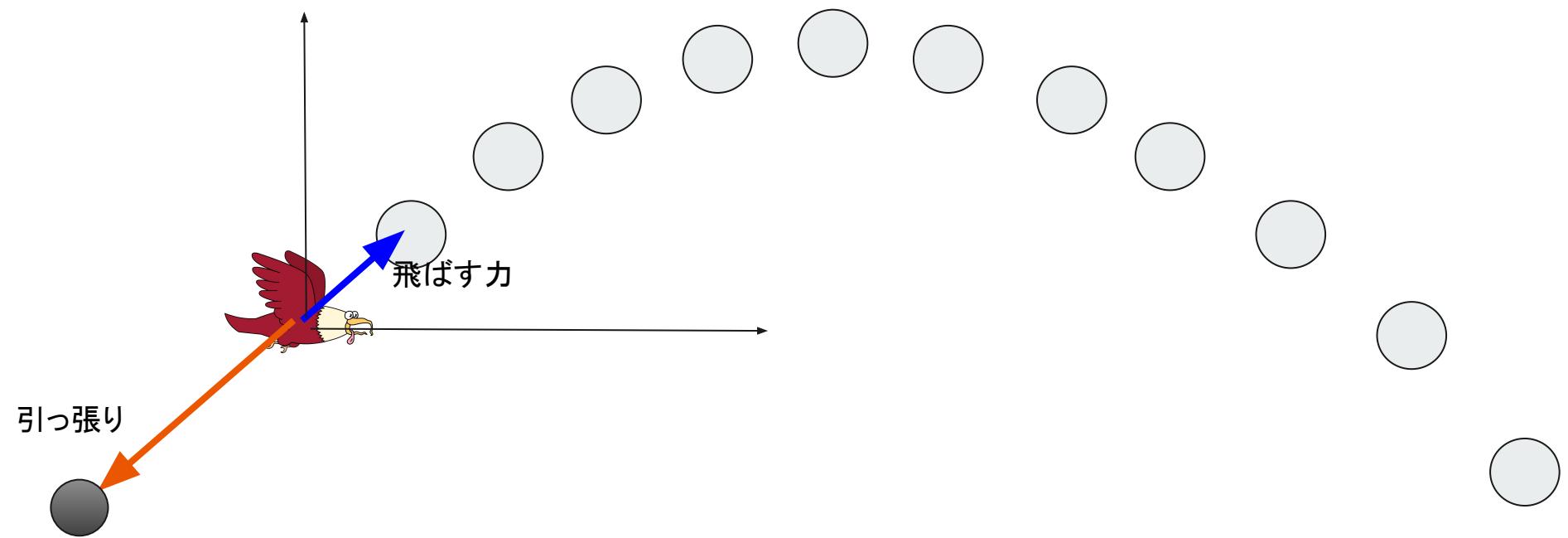
Eagle

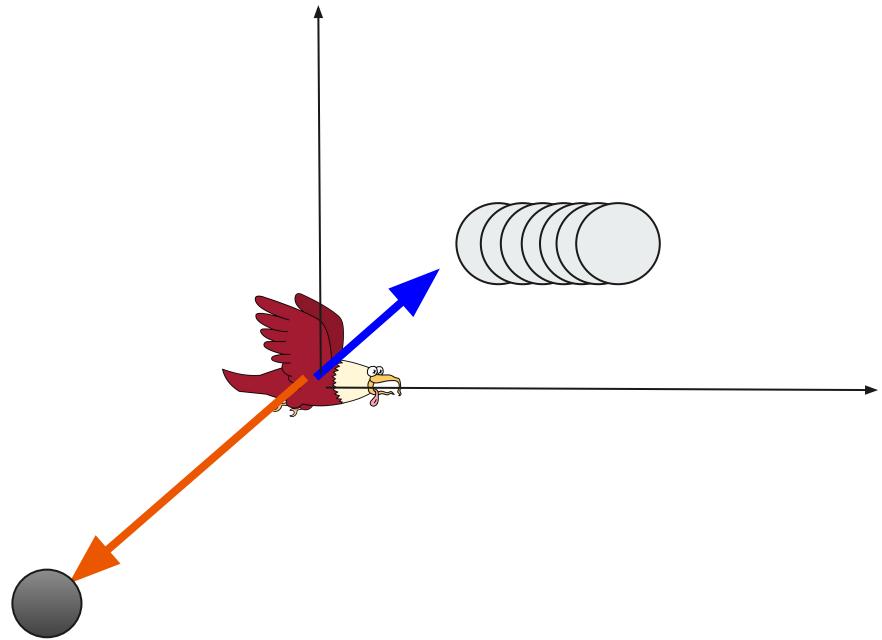
OrbitLine

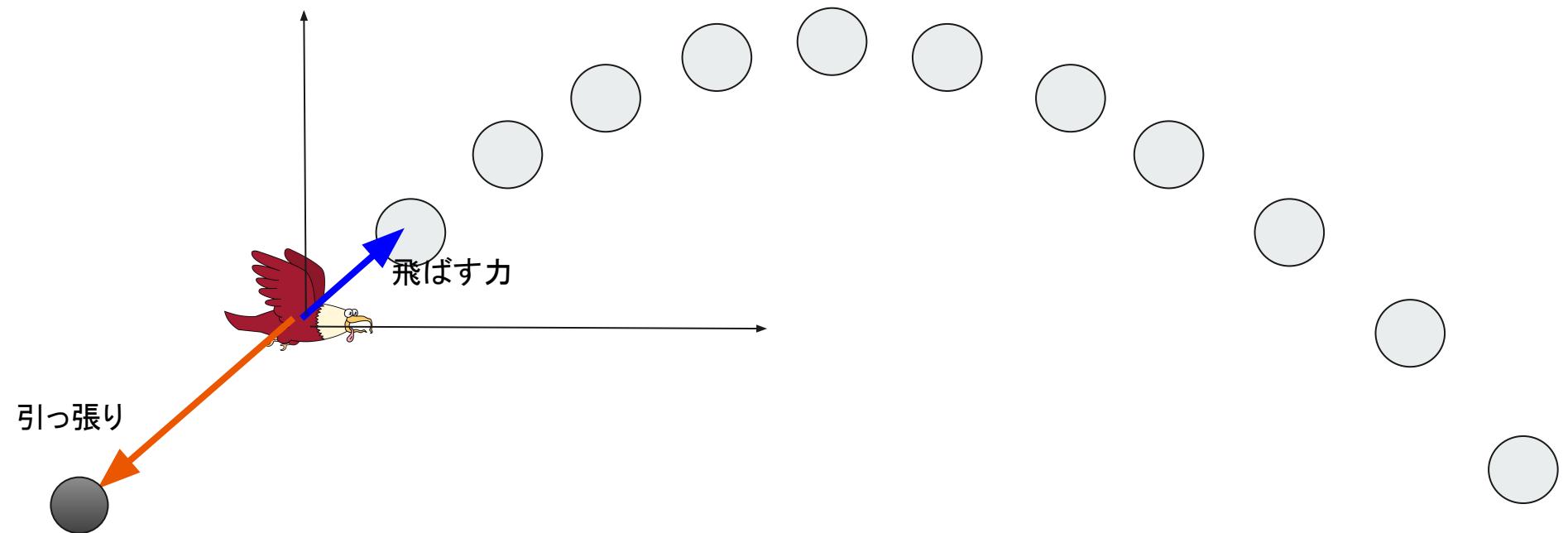
SporneRod

---GameManager---

OrbitCircle

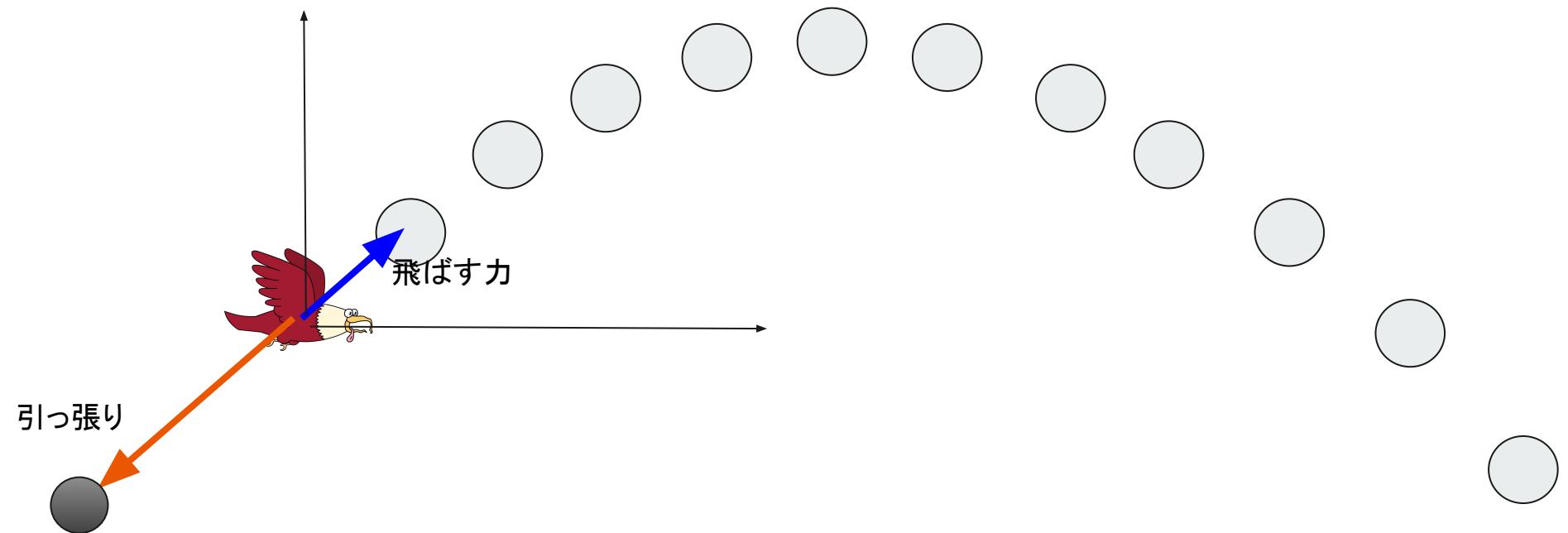




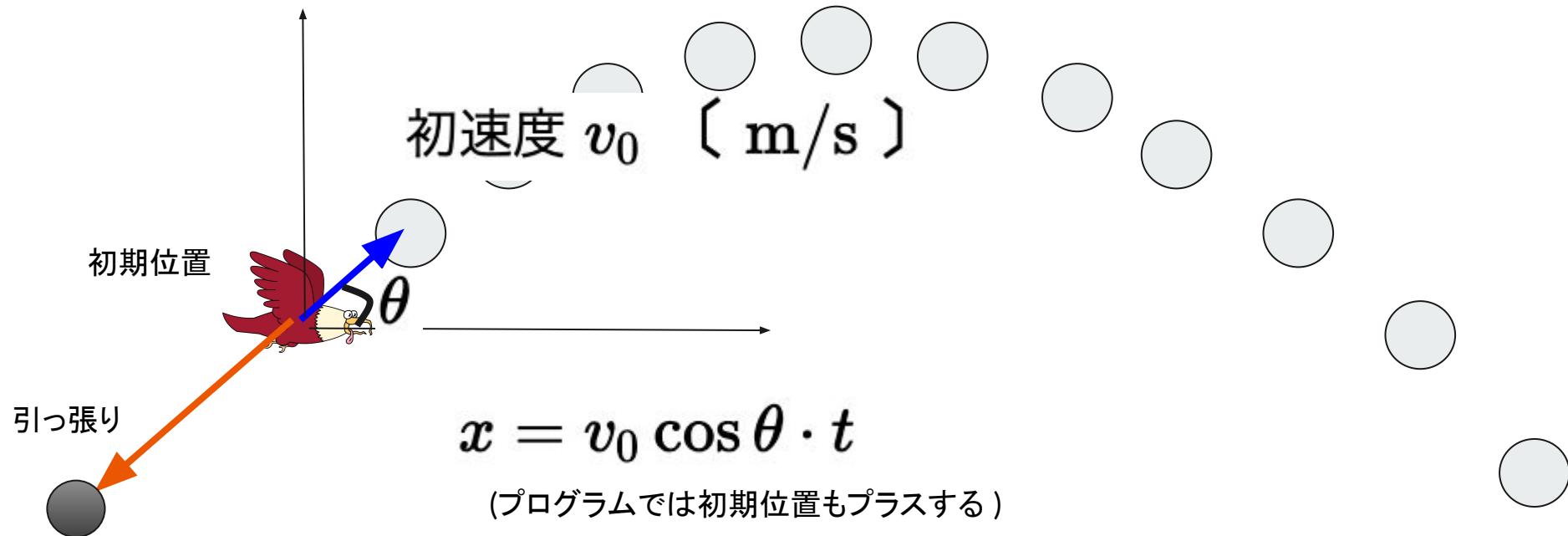


```
/*軌跡*/
[SerializeField] GameObject _orbitCircle; //描画する円
GameObject[] _orbitLineArray = new GameObject[20];
void Start()
{
    _startPos = Vector2.zero; //初期化
    _birdPos = Vector2.zero; //初期化
    TryGetComponent( out _rb2d);
    TryGetComponent(out _c2d);
    _rb2d.isKinematic = true; //物理演算を切っておく
    _c2d.isTrigger = true; //トリガー判定をオンにしておく
    _startPos = transform.position; //開始位置
    _isFlited = false; //飛ばす前

    for (int i=0;i<_orbitLineArray.Length;i++) {
        _orbitLineArray[i] = Instantiate(_orbitCircle);
    }
}
```



$$y = v_0 \sin \theta \cdot t - \frac{1}{2} g t^2 \quad (\text{プログラムでは初期位置もプラスする})$$



```
/*軌跡*/
[SerializeField] GameObject _orbitCircle; //描画する円
GameObject[] _orbitLineArray = new GameObject[20]; //軌道描画時に使用する配列
[SerializeField] float _renderInterval = 0.25f; //描画間隔
```

```
//軌道の描画
void DrawOrbitLine()
{
    Vector2 _forceVec = _flyForce * _flyDirection;//飛ばす速度ベクトル
    var _xAccele = _forceVec.x ;//x方向加速度
    var _yAccele = _forceVec.y ;//y方向加速度

    var currentTime = _renderInterval;

    for (int i = 0; i < _orbitLineArray.Length; i++)
    {
        var Position = new Vector2();
        Position.x = transform.position.x + _xAccele * currentTime;
        Position.y = transform.position.y + _yAccele * currentTime
            - Physics2D.gravity.magnitude * Mathf.Pow(currentTime, 2.0f) / 2;
        _orbitLineArray[i].transform.position = Position;
        currentTime += _renderInterval;
    }
}
```

```
private void OnMouseDown()//マウスをドラッグしている時
{
    if (_isFlited) return;//飛ばしたなら 操作できないようにする
    Vector2 mouse_pos = Camera.main.ScreenToWorldPoint(Input.mousePosition); //マウスの位置
    _birdPos = mouse_pos;
    _flyDirection = _startPos - mouse_pos; //マウスの位置から見た開始位置のベクトル
    float sToM_Distance = _flyDirection.magnitude; //マウスとキャラの距離
    if (sToM_Distance > _maxDistance)
    {
        Vector2 _flyDirectNorm = (-1) * _flyDirection.normalized; //正規化して方向だけ求めて
        _birdPos = _startPos + _maxDistance * _flyDirectNorm; //最大距離まで伸ばして、開始地点に足す
    }
    if (_startPos.x < _birdPos.x) _birdPos.x = _startPos.x; //開始位置より右に行かないようする
    transform.position = _birdPos;

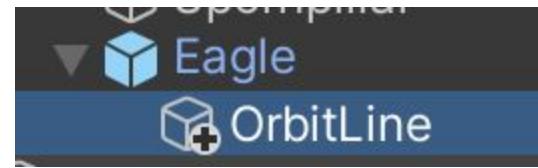
    DrawOrbitLine();
}
```

実行すると、

軌跡はよさそうなのであとは細かいところを直していく

軌跡が操作キャラと一緒に消えるようにする

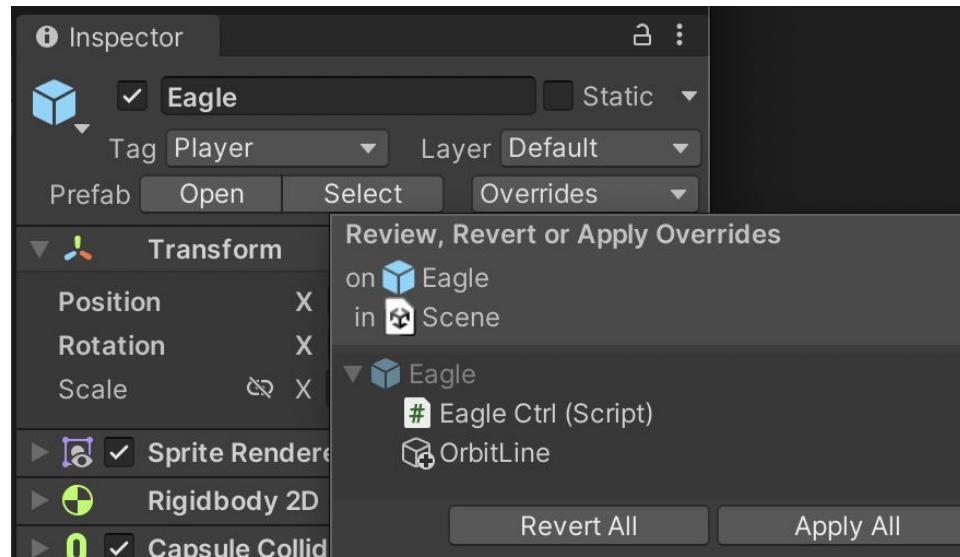
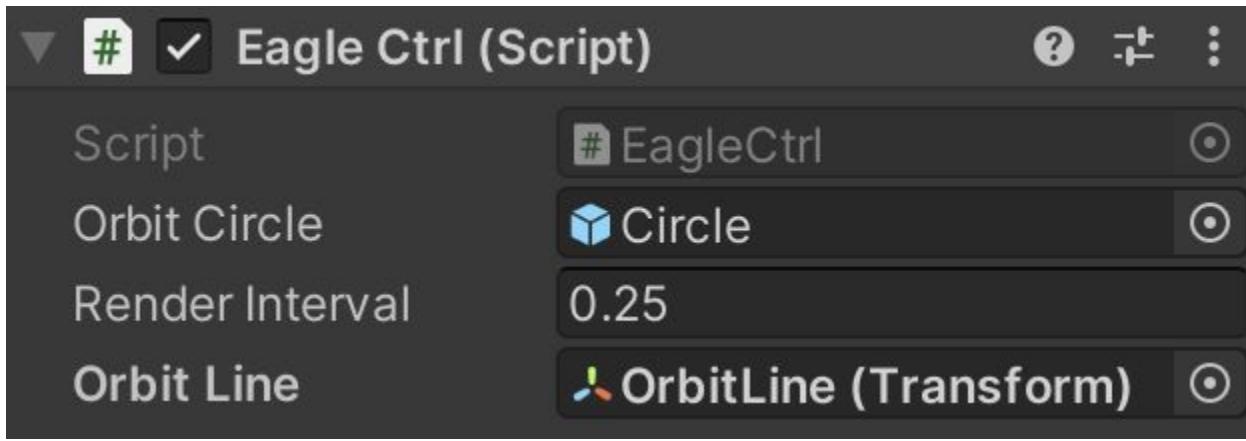
軌跡を操作キャラの子オブジェクトにする



```
[SerializeField] Transform _orbitLine; //この子階層に軌跡のオブジェクトを生成

void Start()
{
    _startPos = Vector2.zero; //初期化
    _birdPos = Vector2.zero; //初期化
    TryGetComponent( out _rb2d);
    TryGetComponent(out _c2d);
    _rb2d.isKinematic = true; //物理演算を切っておく
    _c2d.isTrigger = true; //トリガー判定をオンにしておく
    _startPos = transform.position; //開始位置
    _isFlited = false; //飛ばす前

    for (int i=0 ; i < _orbitLineArray.Length ; i++) {
        _orbitLineArray[i] = Instantiate(_orbitCircle);
        _orbitLineArray[i].transform.parent = _orbitLine.transform;
    }
}
```



実行すると、

軌跡が手を離した後もキャラと一緒に移動してる
あと、話す前から軌跡の円が見える

```
void Start()
{
    _startPos = Vector2.zero; //初期化
    _birdPos = Vector2.zero; //初期化
    TryGetComponent( out _rb2d);
    TryGetComponent(out _c2d);
    _rb2d.isKinematic = true; //物理演算を切っておく
    _c2d.isTrigger = true; //トリガー判定をオンにしておく
    _startPos = transform.position; //開始位置
    _isFlied = false; //飛ばす前

    for (int i=0 ; i < _orbitLineArray.Length ; i++) {
        _orbitLineArray[i] = Instantiate(_orbitCircle);
        _orbitLineArray[i].transform.parent = _orbitLine.transform;
        _orbitLineArray[i].SetActive(false);
    }
}
```

```
//軌道の描画
void DrawOrbitLine()
{
    Vector2 _forceVec = _flyForce * _flyDirection; //飛ばす速度ベクトル
    var _xAccele = _forceVec.x; //x方向加速度
    var _yAccele = _forceVec.y; //y方向加速度

    var currentTime = _renderInterval;

    for (int i = 0; i < _orbitLineArray.Length; i++)
    {
        _orbitLineArray[i].SetActive(true);
        var Position = new Vector2();
        Position.x = transform.position.x + _xAccele * currentTime;
        Position.y = transform.position.y + _yAccele * currentTime
            - Physics2D.gravity.magnitude * Mathf.Pow(currentTime, 2.0f) / 2;
        _orbitLineArray[i].transform.position = Position;
        currentTime += _renderInterval;
    }
}
```

```
float _flyForce = 5f;
private async void OnMouseUp()
{
    if (_isFlited) return;//飛ばしたなら 操作できないようにする
    _isFlited = true;//「飛ばした」状態にする
    _rb2d.isKinematic = false;//物理演算を行う
    _rb2d.AddForce(_flyForce * _flyDirection, ForceMode2D.Impulse);//飛ばした瞬間だけ外力を加える
    for (int i = 0; i < _orbitLineArray.Length; i++){//軌跡に描画する円の配列
        _orbitLineArray[i].SetActive(false); //離したら描画を止める
    }
    await NewCaracterCreate();
}
```

実行、
あとは、軌跡の大きさを緩やかに小さくするなど

```
float _magnification = 0.045f;//縮小度

void Start()
{
    _startPos = Vector2.zero;//初期化
    _birdPos = Vector2.zero;//初期化
    TryGetComponent( out _rb2d);
    TryGetComponent(out _c2d);
    _rb2d.isKinematic = true;//物理演算を切っておく
    _c2d.isTrigger = true;//トリガー判定をオンにしておく
    _startPos = transform.position;//開始位置
    _isFlited = false;//飛ばす前

    for (int i=0 ; i < _orbitLineArray.Length ; i++) {
        _orbitLineArray[i] = Instantiate(_orbitCircle);
        _orbitLineArray[i].transform.parent = _orbitLine.transform;// _orbitLineに設定した子オブジェクトにする
        //徐々に小さくする マイナスになると拡大してしまう
        _orbitLineArray[i].transform.localScale = _orbitLineArray[i].transform.localScale * (1 - _magnification * i);
        _orbitLineArray[i].SetActive(false);
    }
}
```

敵との衝突処理

Add Component



bo



Search



Box Collider



Box Collider 2D



Articulation Body



Billboard Renderer



Rigidbody



Rigidbody 2D



Skybox

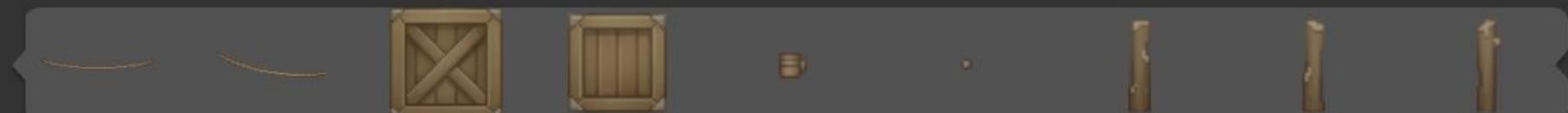
New script



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

public class Enemy : MonoBehaviour
{
    private void OnCollisionEnter2D(Collision2D collision)
    {
        Destroy(gameObject);
    }
}
```

Assets > AssetStore > Cainos > Pixel Art Platformer - Village Props > Texture



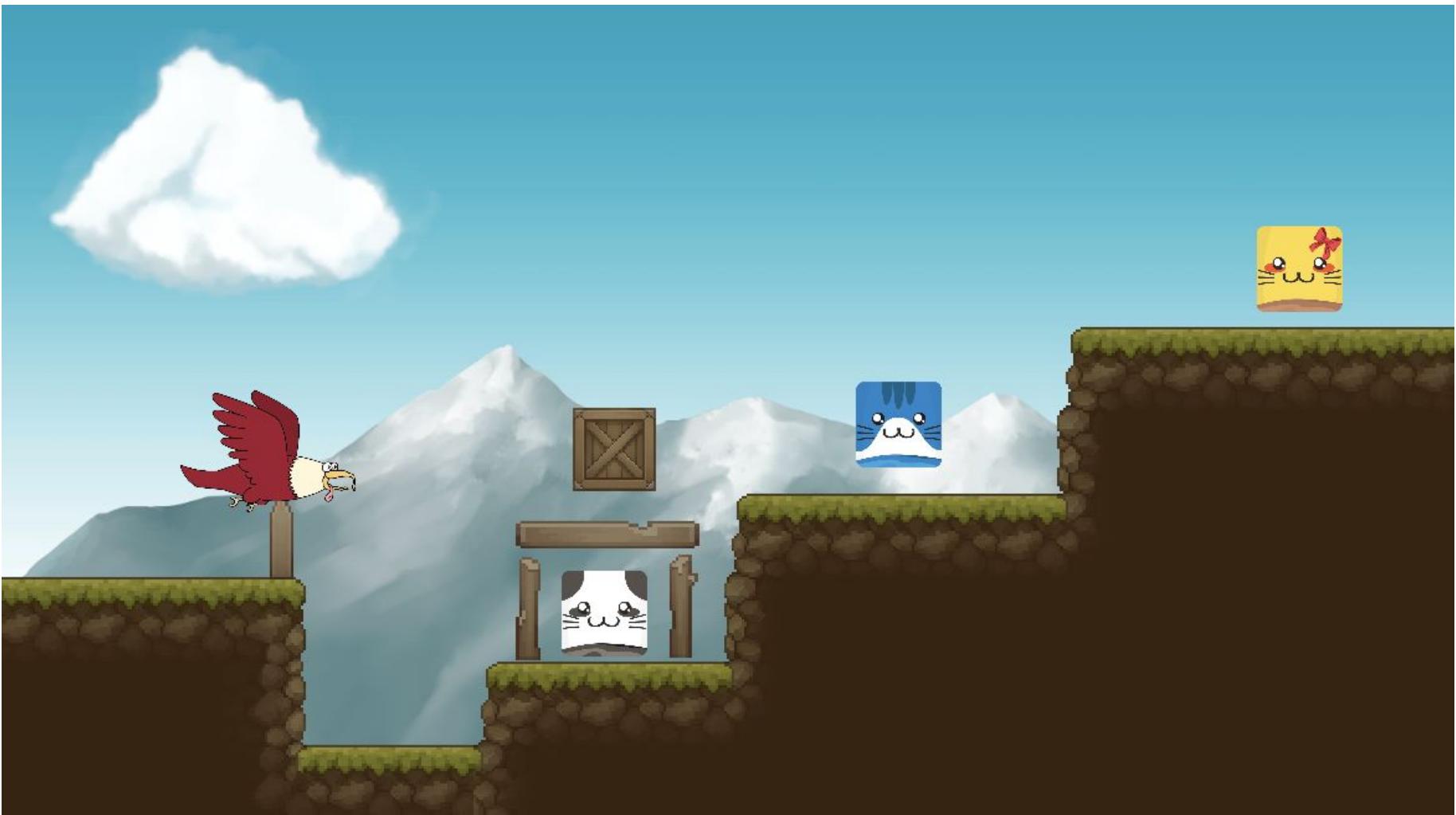
TX Villa... TX Villa...



TX Villa... TX Villa...



TX Villa... TX Villa...



▶	 <input checked="" type="checkbox"/>	Box Collider 2D	  
▶	 <input type="checkbox"/>	Rigidbody 2D	  

実行すると
敵キャラが何と当たっても消えてしまうので、速さ依存で消えるようにする

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

public class Enemy : MonoBehaviour
{
    private void OnCollisionEnter2D(Collision2D collision)
    {
        Debug.Log(collision.relativeVelocity.sqrMagnitude);
        //Destroy(gameObject);
    }
}
```

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

public class Enemy : MonoBehaviour
{
    float _destroyVelocity = 40f; //この値以上ならオブジェクトを破壊する
    private void OnCollisionEnter2D(Collision2D collision)
    {
        if (collision.relativeVelocity.sqrMagnitude > _destroyVelocity)
        {
            Destroy(gameObject);
        }
    }
}
```

実行

敵キャラの数が0になったことをどうやって判断するか

敵キャラが出現した時に、staticなGameManagerに、
敵キャラの数をカウントしてくれるプログラムにしてみよう

GameManager.cs

```
int enemyCount
```



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

public class Enemy : MonoBehaviour
{
    float _destroyVelocity = 40f; //この値以上ならオブジェクトを破壊する
    private void OnCollisionEnter2D(Collision2D collision)
    {
        if (collision.relativeVelocity.sqrMagnitude > _destroyVelocity)
        {
            Destroy(gameObject);
        }
    }

    private void Start()
    {
        //生成されたときに敵の数を増やす
        GameManager.Instance.EnemyCountAdd();
    }
}
```

GameManager.cs

```
//敵の数  
int _enemyCount;  
public int EnemyCount => _enemyCount;
```

```
///<summary>  
/// 初期敵の数  
///</summary>  
public void EnemyCountAdd()  
{  
    _enemyCount++;  
}
```

```
///<summary>
/// 敵の数を減らす
///</summary>
public void KillEnemy()
{
    _enemyCount--;
    if (_enemyCount == 0)
    {
        Debug.Log("クリア");
    }
}
```

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

public class Enemy : MonoBehaviour
{
    float _destroyVelocity = 40f; //この値以上ならオブジェクトを破壊する
    private void OnCollisionEnter2D(Collision2D collision)
    {
        if (collision.relativeVelocity.sqrMagnitude > _destroyVelocity)
        {
            Destroy(gameObject);
            GameManager.Instance.KillEnemy();
        }
    }
    private void Start()
    {
        //生成されたときに敵の数を増やす
        GameManager.Instance.EnemyCountAdd();
    }
}
```

実行

クリア画面までは作る

あとは、

- ・GameOver
 - ・残機を設定、表示
 - ・ステージを増やしてシーン遷移、タイトル
 - ・カメラの挙動を変える
 - ・BGM、SE
 - ・エフェクト
- などなど

Cut

Copy

Paste

Paste As Child

Rename

Duplicate

Delete

Select Children

Set as Default Parent

Create Empty

2D Object >

3D Object >

Effects >

Light >

Audio >

Video >

UI >

Canvas

Render Mode Screen

Pixel Perfect

Render Camera Main Camera

Plane Distance 100

Resize Canvas Enabled

Sorting Layer Default

Order in Layer 0

Image

Text - TextMeshPro

Raw Image

Panel

Toggle

Slider

Scrollbar

ScrollView

Button - TextMeshPro

Dropdown - TextMeshPro

Input Field - TextMeshPro

▼   **Canvas**   

Render Mode	Screen Space - Camera
Pixel Perfect	<input type="checkbox"/>
Render Camera	 Main Camera (Camera) 
Plane Distance	100
Resize Canvas	Enabled
Sorting Layer	Default
Order in Layer	0
Additional Shader Chanr	Nothing

▼  **Rect Transform**   

center	Pos X	Pos Y	Pos Z
middle	0	0	0
	Width	Height	
	1200	900	<input type="button" value="[]"/> R
▶ Anchors			
Pivot	X 0.5	Y 0.5	
Rotation	X 0	Y 0	Z 0
Scale	X 1	Y 1	Z 1



Game

Game

Display 1

16:9 Aspect

▼

Scale

1x

Norm: ▼

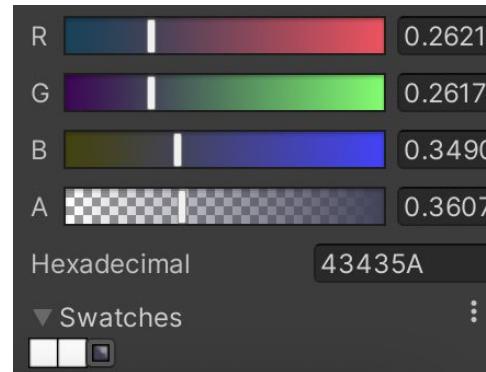
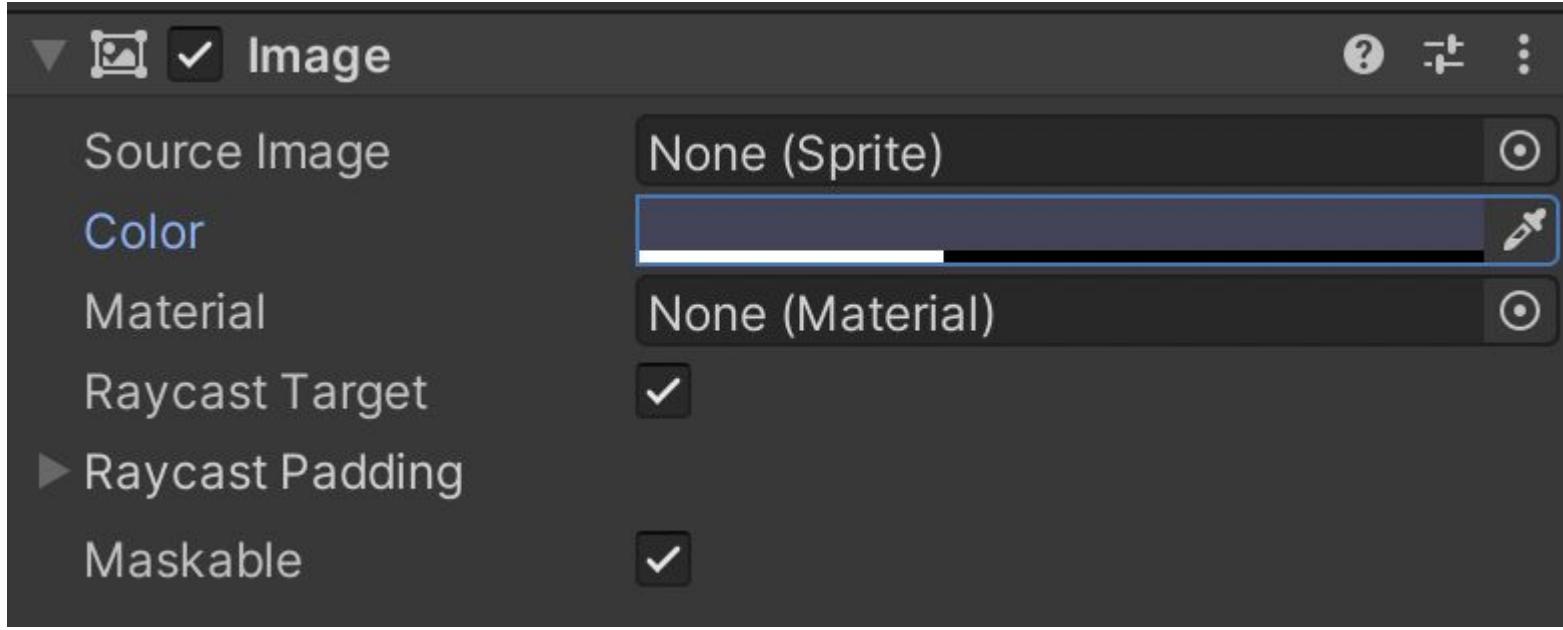
Mute Audio

Stats

⋮



! キャラ作成



Game

Game

Display 1

16:9 Aspect



Scale



1x

Norm:



Mute Audio

Stats



- Cut
- Copy
- Paste
- Paste As Child
-
- Rename
- Duplicate
- Delete
-
- Select Children
-
- Set as Default Parent
-
- Create Empty
- Create Empty Parent
-
- 2D Object >
- 3D Object >
- Effects >
- Light >
- Audio >
- Video >
- UI >

center


middle

POS X 0

Width 1200

▶ Anchors

Pivot X 0.5

Rotation X 0

Scale X 1

▼  Canvas Renderer

Cull Transparent Mesh

▼  Image

Source Image None

Color 

⋮

23

Image

Text - TextMeshPro

Raw Image

Panel

Toggle

Slider

Scrollbar



TMP Importer

TMP Importer

⋮

TMP Essentials

This appears to be the first time you access TextMesh Pro, as such we need to add resources to your project that are essential for using TextMesh Pro. These new resources will be placed at the root of your project in the "TextMesh Pro" folder.

[Import TMP Essentials](#)

TMP Examples & Extras

The Examples & Extras package contains addition resources and examples that will make discovering and learning about TextMesh Pro's powerful features easier. These additional resources will be placed in the same folder as the TMP essential resources.

[Import TMP Examples & Extras](#)

Alignment



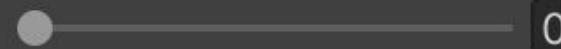
Face

- Click to collapse -

Color

HDR

Softness



Dilate



Outline

- Click to collapse -

Color

HDR

Thickness



Underlay

- Click to expand -

TextMeshPro(日本語Font)



「フリーフォント」で検索かけて何かしらの日本語フォントを取ってくる

FONT TEST

これは「こどもれゴシック」Fontです。

これは「マメロン3 Regular」Fontです。

これは「マメロン5 Regular」Fontです。

これは「源ノ角ゴシック Regular」Fontです。

これは「源ノ角ゴシック Heavy」Fontです。

Instantly share code, notes, and snippets.



kgsi / [japanese_full.txt](#)

Last active 6 days ago

Star 135

Fork 18

[Code](#)

[-o-Revisions 2](#)

[Stars 135](#)

[Forks 18](#)

Embed ▾

<script src="https://



[Download ZIP](#)

日本語文字コード範囲指定 (ascii・ひらがな・カタカナ・第一水準および第二水準(JIS-X0208-1997)に含まれる漢字)

[japanese_full.txt](#)

Raw

This file contains bidirectional Unicode text that may be interpreted or compiled differently than what appears below. To review, open the file in an editor that reveals hidden Unicode characters. [Learn more about bidirectional Unicode characters](#)

[Hide revealed characters](#)

1 !#\$%&'()*+,-./0123456789:@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~ abcdefghijklmnopqrstuvwxyzABCDEGHIJKLMNOPQRSTUVWXYZ

<https://gist.github.com/kgsi/ed2f1c5696a2211c1fd1e1e198c96ee4?h=1>

<https://gist.github.com/kgsi/ed2f1c5696a2211c1fd1e1e198c96ee4?h=1>

▼  kodomo



KodomoRounded-Light.otf



KodomoRounded.otf

Assets > Font

Aa^o

KodomoRo...

s

Window

Help

Minimize

⌘M

Zoom

Bring All to Front

Panels

>

Layouts

>

Collaborate

Asset Store

Package Manager

Asset Management

>

TextMeshPro

>

General

>

01 - PC, Mac & Linux Standalone - Unity

zmos



Shaded

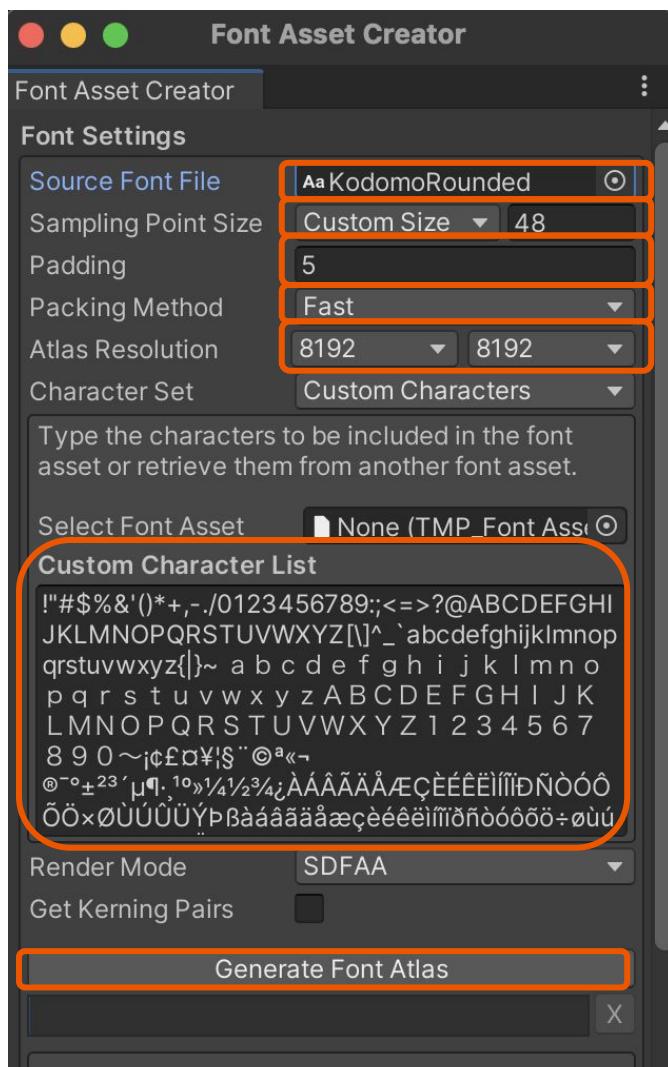


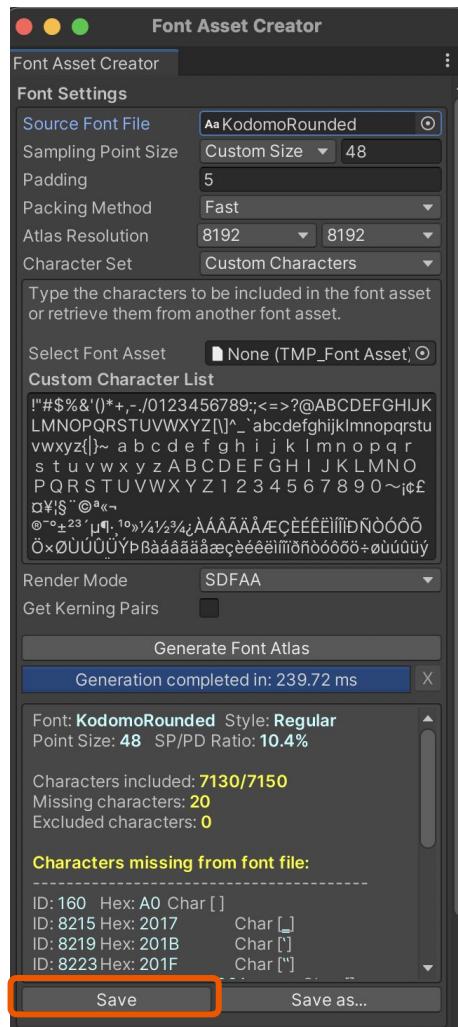
2D



Font Asset Creator

Sprite Importer





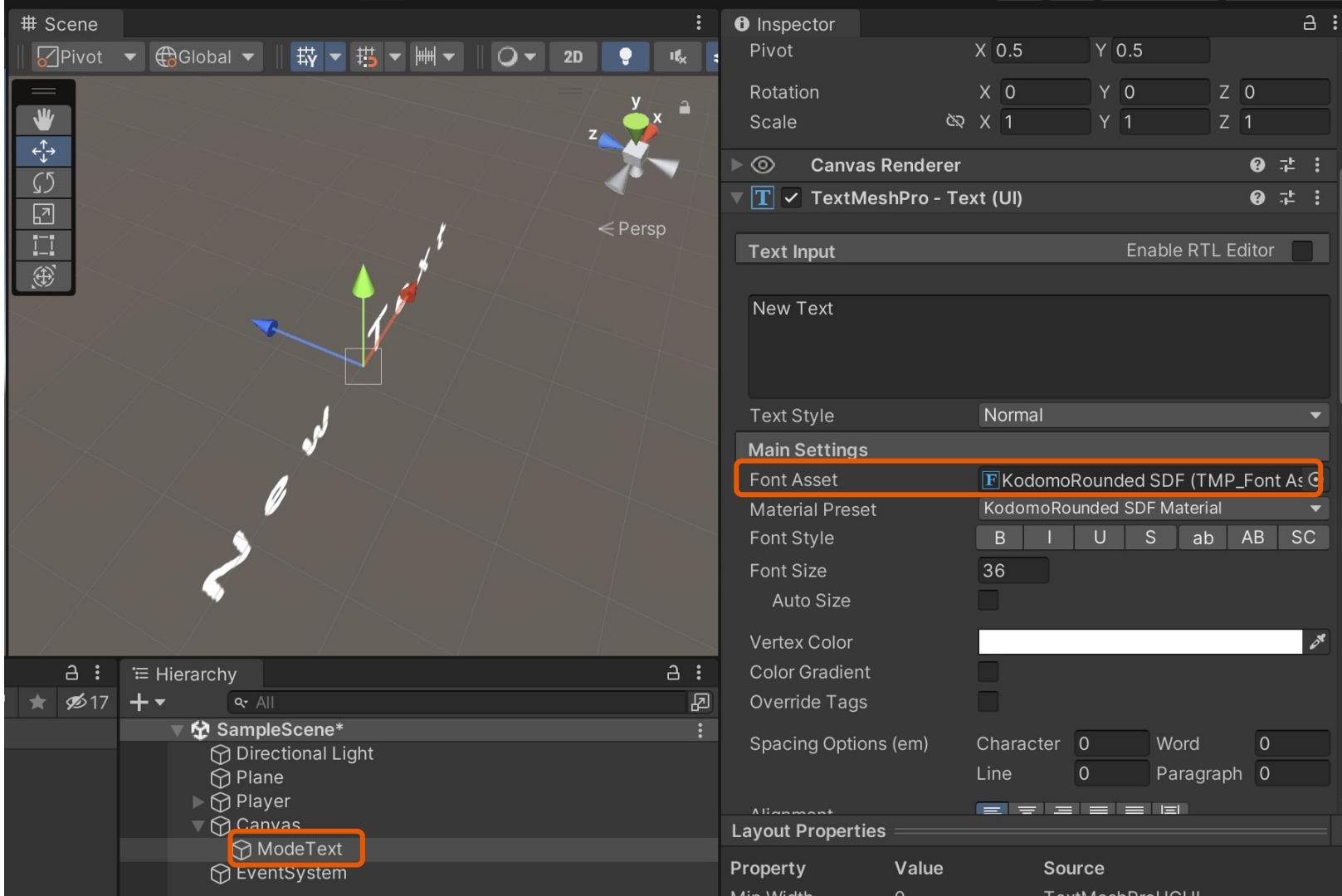
Assets > Font

Aa F

KodomoRo...

KodomoRo...





Game

Game

Display 1

16:9 Aspect



Scale



1x

Norm:



Mute Audio

Stats

Gizm



Game

⋮

Game

▼

Display 1

▼

16:9 Aspect

▼

Scale 1x

Norm:

▼

Mute Audio

Stats

Gizmo



Hierarchy

Font Asset Creator



All

SampleScene

Main Camera

Enviroment

BackGround

Enemies

DownLimit

Grid

SpornPoint

Spornpillar

Eagle

OrbitLine

Obstacles

---GamaManager---

Canvas

Clear

Image

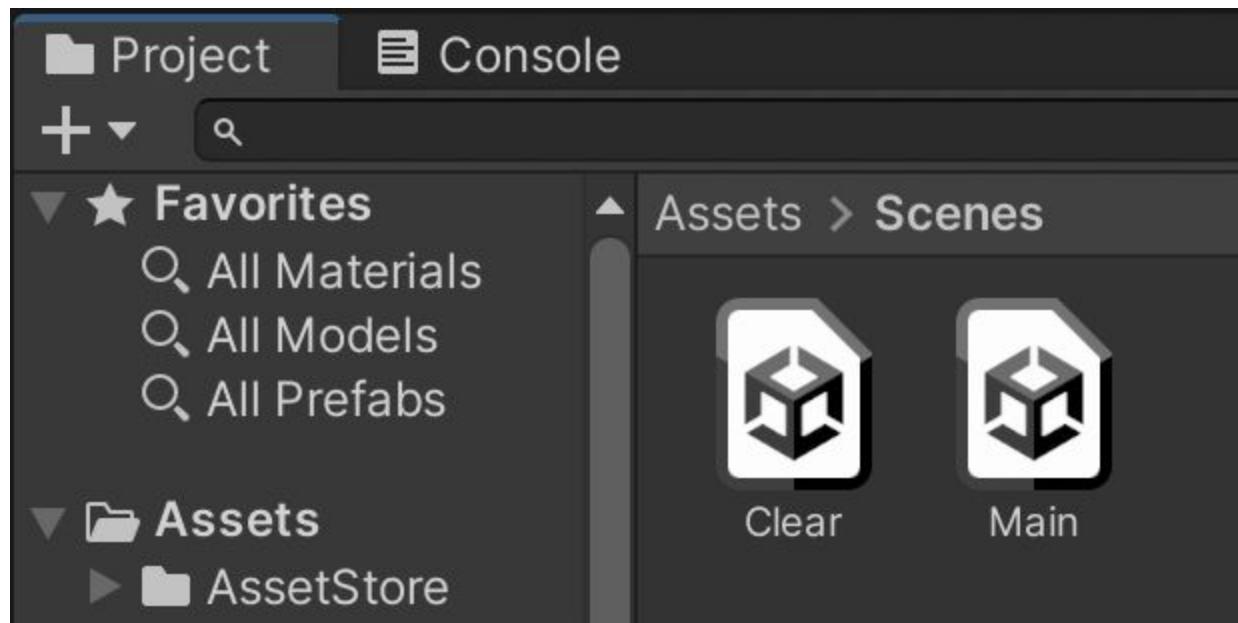
Text (TMP)

Button

EventSystem

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

public class Retry : MonoBehaviour
{
    private void Start()
    {
        GetComponent<Button>().onClick.AddListener(() =>
            { SceneManager.LoadScene(SceneManager.GetActiveScene().name); });
    }
}
```



Hierarchy

Font Asset Creator



All



▼ Main

- Main Camera
- Enviroment
- GamaManager---
- Canvas
- EventSystem

▼ Clear

- Main Camera
- Canvas
- EventSystem

Game

⋮

Game

▼

Display 1

▼

16:9 Aspect

▼

Scale 1x

Norm:

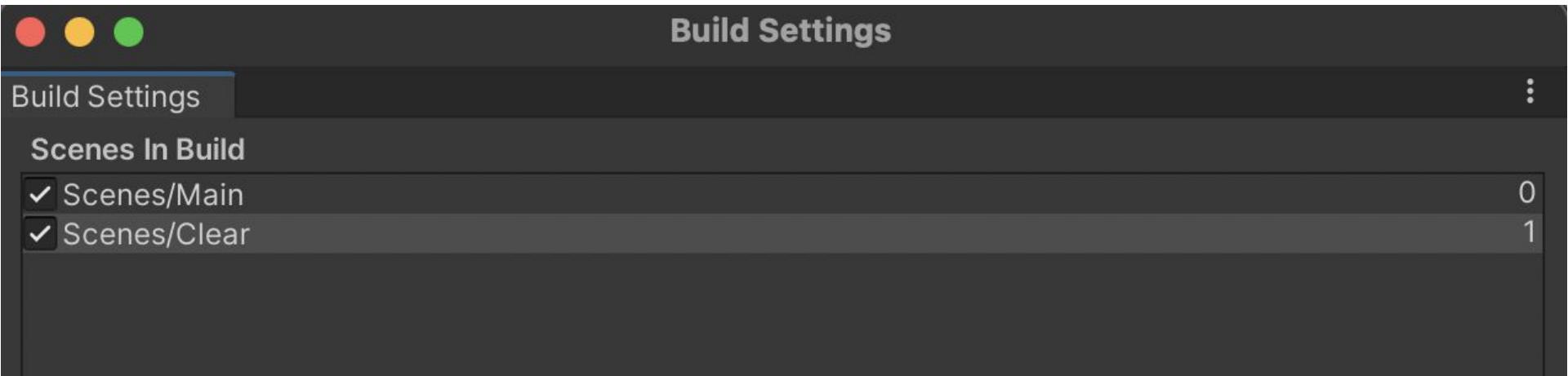
▼

Mute Audio

Stats

Gizmo





GameManager.cs

```
using UnityEngine.SceneManagement;
```

```
///<summary>
/// 敵の数を減らす
///</summary>
public void KillEnemy()
{
    _enemyCount--;
    Debug.Log(_enemyCount);
    if (_enemyCount == 0)
    {
        Debug.Log("クリア");
        SceneManager.LoadSceneAsync("Clear", LoadSceneMode.Additive);
    }
}
```

Hierarchy

Font Asset Creator



All

▼ Main

- Main Camera
- Enviroment
- GamaManager---
- Canvas
- EventSystem

Clear

- Main Camera
- Canvas
- EventSystem

Set Active Scene

Save Scene

Save Scene As

Save All

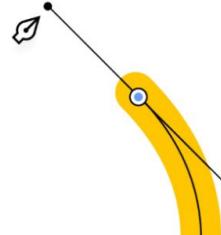
Unload Scene

Remove Scene

[サインアップ](#)

一人ではすごいものは 完成できない。

Figmaでは、チームがよりよい製品をより早く提供できるように、全員がデザインプロセスに参加できるようにします。

[Figmaを無料で体験する](#)

実行してみる

あとは、

- ・残機を設定、表示
 - ・ステージを増やしてシーン遷移、タイトル
 - ・BGM、SE
- などなど
- ・GameOver処理
 - ・カメラの挙動を変える
 - ・エフェクト

