9 月 11 日 課題

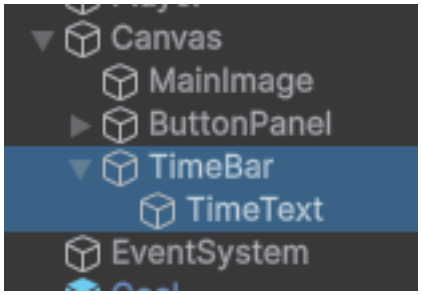
9 月 10 日 のおさらい

各プログラムの全容については github をご参考ください
https://github.com/Dyna-Rise/JewelryHunter_Unity6

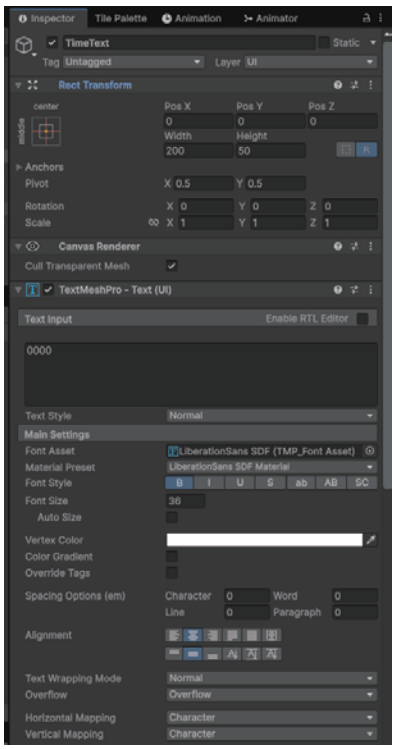
9 月 10 日 のおさらい①

Canvas にアタッチしている TimeController.cs で計算しているカウント時間を
UI「TimeText」オブジェクトに反映させる

ヒエラルキーにある
TimeBar オブジェクトと TimeText オブジェクト



TimeText オブジェクトのインスペクター



・どうやって反映させるのか？
同じく Canvas についている
「UIController.cs」にて
・TimeText オブジェクト
を認識し、
・TimeController の変数 displayTime
の内容を代入する
（切り上げの上、文字列型に変換が必要）

UIController.cs

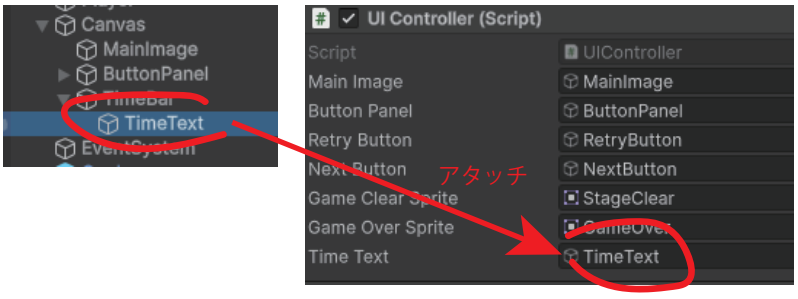
まずは TMPro の呼び出し

```
using TMPro;
```

TimeText と TimeController を扱う変数

```
TimeController timeCnt; //TimeController.csの参照
public GameObject timeText;  //ゲームオブジェクトであるTimeText
```

public である変数 timeText は Unity 上で初期化



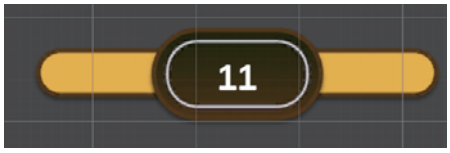
Start メソッドの中で TimeController 情報の取得

```
void Start()
{
    timeCnt = GetComponent<TimeController>();
```

Update の条件分岐の中に、
ステータスが「playing」だったら TimeController の変数 displayTime を TimeText オブジェクトに反映させるコードを追加

TimeText オブジェクトの TextMeshProUGUI コンポーネントの変数 text に
切り上げ（Mathf.Ceil) して文字列化 (ToString) した時間を代入

```
else if(GameManager.gameState == "playing")
{
    //いったんdisplayTimeの数字を変数timesに渡す
    float times = timeCnt.displayTime;
    timeText.GetComponent<TextMeshProUGUI>().text = Mathf.Ceil(times).ToString();
}
```



プレイ中に反映されれば OK

9 月 10 日 のおさらい②

Item オブジェクトを作成、プレハブ化し
Player が Item に触れたら GameManager のステージスコアに加算する仕組みをつくる

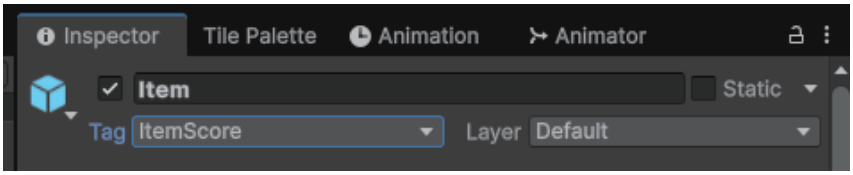
GameManager.cs

GameManager.cs に 2 つの static 変数
totalScore と stageScore を用意しておく

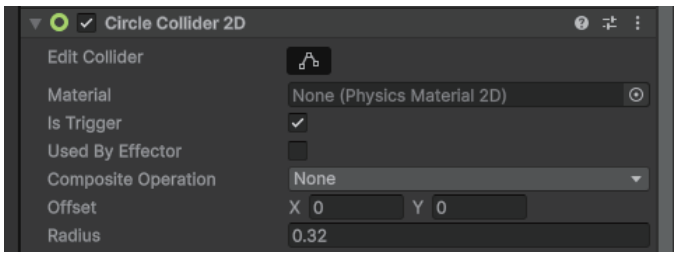
```
using UnityEngine;

public class GameManager : MonoBehaviour
{
    public static string gameState;

    public static int totalScore; //ゲーム全般を通してのスコア
    public static int stageScore; //そのステージに獲得したスコア

    // Start is called once before the first execution of Update a
    void Awake()
```

Item オブジェクトを準備

ItemScore タグをつくりオブジェクトにつけておく
※授業では ScoreItem と作ってしまいました、失礼いたしました



CircleCollider2D を追加して「isTrigger」にチェック



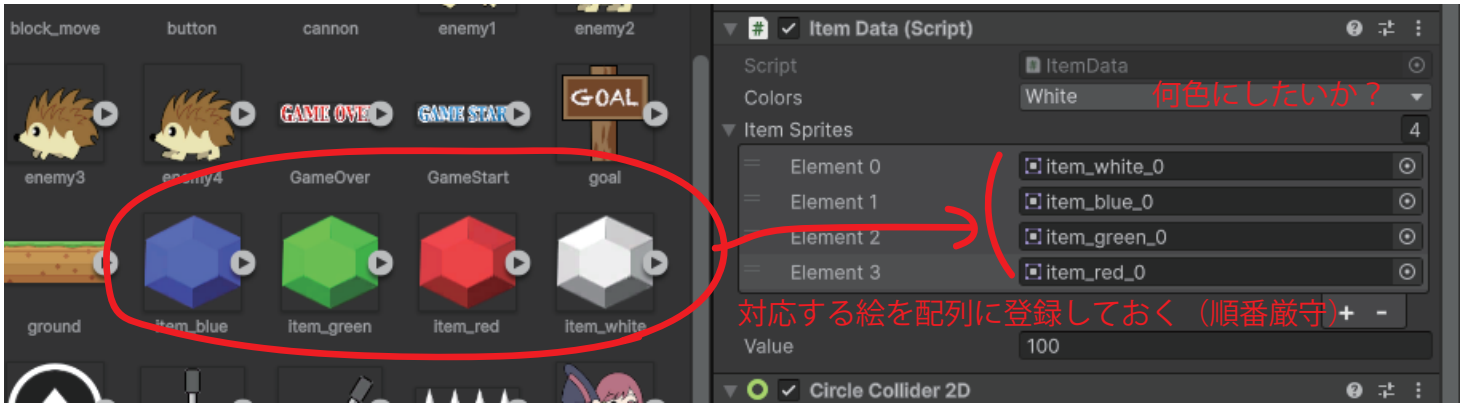
ItemData.cs（Item オブジェクトにアタッチ）

ItemColor という名前で
「White」、「Blue」、「Green」、「Red」の 4 つの値をもっている自作の列挙型をつくる

```
using UnityEngine;

public enum ItemColor
{
    White,
    Blue,
    Green,
    Red
}
```

自作した ItemColor を扱う変数「colors」、
絵の情報（Sprite）を複数扱う配列「itemSprites」を用意し、
Unity エディター上で初期化する

```
public class ItemData : MonoBehaviour
{
    public ItemColor colors = ItemColor.White;
    public Sprite[] itemSprites;

    public int value = 0;          // 整数値を設定できる
```




```
void Start()
{
    SpriteRenderer spriteRenderer = GetComponent<SpriteRenderer>();

    switch (colors)
    {
        case ItemColor.White:
            spriteRenderer.sprite = itemSprites[0];
            break;
        case ItemColor.Blue:
            spriteRenderer.sprite = itemSprites[1];
            break;
        case ItemColor.Green:
            spriteRenderer.sprite = itemSprites[2];
            break;
        case ItemColor.Red:
            spriteRenderer.sprite = itemSprites[3];
            break;
    }
}
```

Start メソッドにて
SpriteRenderer コンポーネントの
情報を変数に代入した後、
switch 文で
変数 colors の指定した
見た目になるように条件分岐

PlayerController.cs

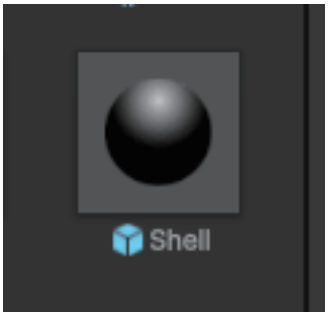
衝突イベント OnTriggerEnter2D メソッドの中で
「ItemScore」タグとぶつかったら GameManager の stageScore 変数が加算されるよう追加

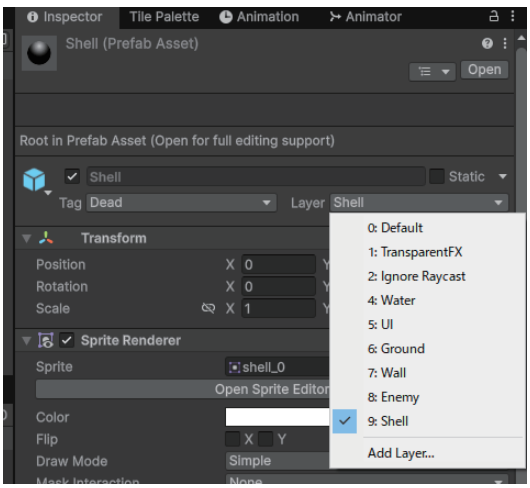
```
//isTrigger特性をもっているColliderとぶつかったら処理される
private void OnTriggerEnter2D(Collider2D collision)
{
    //ぶつかった相手が"Gaol"タグを持っていたら
    //if (collision.gameObject.tag == "Goal")
    if (collision.gameObject.CompareTag("Goal"))
    {
        GameManager.gameState = "gameclear";
        Debug.Log("ゴールに接触した！");
        Goal();
    }

    //ぶつかった相手が"Dead"タグを持っていたら
    if (collision.gameObject.CompareTag("Dead"))
    {
        GameManager.gameState = "gameover";
        Debug.Log("ゲームオーバー！");
        GameOver();
    }

    //アイテムに触れたらステージスコアに加算
    if (collision.gameObject.CompareTag("ItemScore"))
    {
        GameManager.stageScore += collision.gameObject.GetComponent<ItemData>().value;
        Destroy(collision.gameObject);
    }
}
```

9 月 10 日 のおさらい③　※データインポートの後で

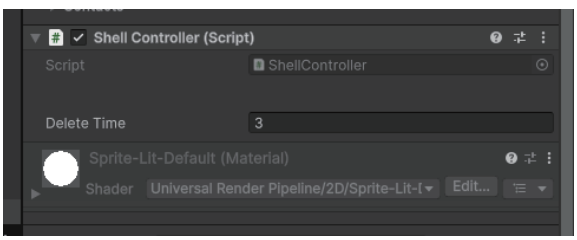
Shell プレハブに ShellController.cs
Cannon プレハブに CannonController.cs　を用意し
砲弾が飛ぶように準備する

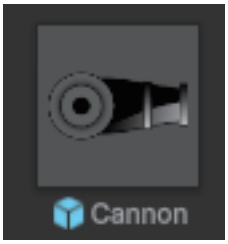
Shell プレハブのレイヤーを 9 番「Shell」にする
※8 番「Enemy」もつくっておく

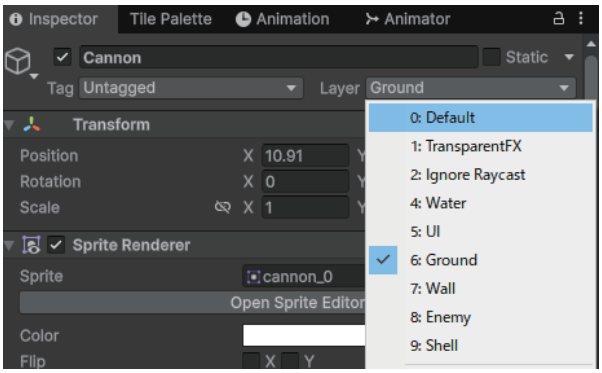




ShellController.cs をコーディングし、
Shell プレハブにアタッチしておく







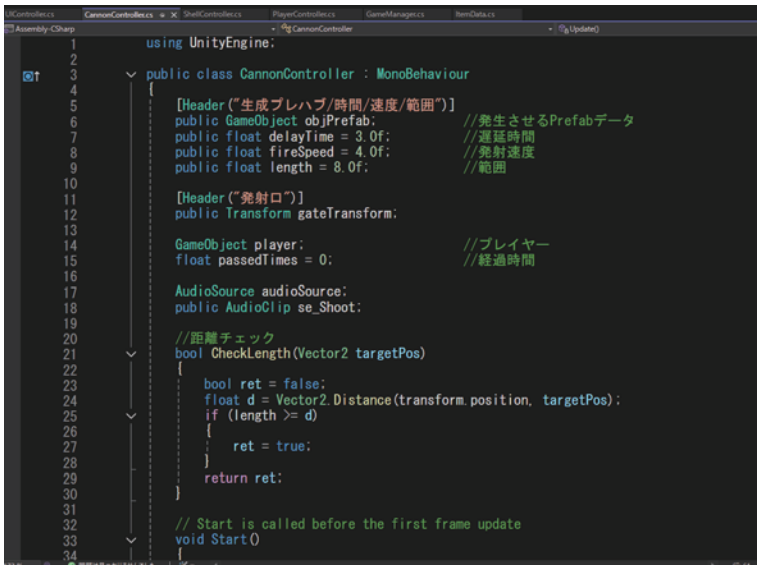
Cannon プレハブのレイヤーを
6 番「Ground」にする

CannonController.cs をコーディングする
※次の部分は今はいらない
変数宣言
・AudioSource 型の audioSource
・public AudioClip se_Shoot

Start メソッド
・audioSource = GetComponent<AudioSource>()

Shell 生成後の
・audioSouce.PlayOneShot(se_shoot);

・void OnDrawGizmosSelected メソッドそのもの



```
using UnityEngine;

public class CannonController : MonoBehaviour
{
    [Header("生成プレハブ/時間/速度/範囲")]
    public GameObject objPrefab;          //発生させるPrefabデータ
    public float delayTime = 3.0f;        //遅延時間
    public float fireSpeed = 4.0f;        //発射速度
    public float length = 8.0f;           //範囲

    [Header("発射口")]
    public Transform gateTransform;

    GameObject player;                    //プレイヤー
    float passedTimes = 0;                //経過時間

    AudioSource audioSource;
    public AudioClip se_Shoot;

    //距離チェック
    bool CheckLength(Vector2 targetPos)
    {
        bool ret = false;
        float d = Vector2.Distance(transform.position, targetPos);
        if (length >= d)
        {
            ret = true;
        }
        return ret;
    }

    // Start is called before the first frame update
    void Start()
    {
```

ここまでがおさらい

9 月 11 日課題につづく