Deffence

Bullet のプレハブ

RigidBody オン（Gravity　オフ）
Tag を EnemyShell

```
//https://xn--
sckyeodz49lj8c.com/unity%E3%81%A7%E3%83%9B%E3%83%BC%E3%83%9F%E3%83%B3%E3%
82%B0%E5%BC%BE%E3%82%92%E4%BD%9C%E3%81%A3%E3%81%A6%E3%81%BF%E3%82%8B/

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

public class BulletController : MonoBehaviour
{
    // 弾が追跡する対象の Transform
    private Transform target;

    // 弾の生存時間
    public float time = 1;

    // 加速度の制限をするかどうか
    public bool limitAcceleration = false;

    // 加速度の最大値
    public float maxAcceleration = 100;

    // 初期速度の最小値と最大値
    public Vector3 minInitVelocity;
    public Vector3 maxInitVelocity;

    // 弾の位置、速度、加速度、自身の Transform
    private Vector3 position;
```

```
    private Vector3 velocity;
    private Vector3 acceleration;
    private Transform thisTransform;

    // 初期化処理
    void Start()
    {
        // Player という名前のゲームオブジェクトを検索し、その Transform を target に設定
        target = GameObject.Find("Player").transform;

        // 自身の Transform を thisTransform に設定
        thisTransform = transform;

        // 弾の初期位置を取得
        position = thisTransform.position;

        // 初期速度をランダムに設定
        velocity = new Vector3(Random.Range(minInitVelocity.x, maxInitVelocity.x), Random.Range(minInitVelocity.y, maxInitVelocity.y), Random.Range(minInitVelocity.z, maxInitVelocity.z));
    }

    // 更新処理
    void Update()
    {
        // もし対象が存在しない場合は処理を終了
        if (target == null)
        {
            return;
        }

        // 弾の加速度を計算
        acceleration = 2f / (time * time) * (target.position - position - time * velocity);
```

```
        // 加速度の制限が有効で、加速度の大きさが最大値を超えている場合は、加速度を
最大値に制限する
        if (limitAcceleration && acceleration.sqrMagnitude >
maxAcceleration * maxAcceleration)
        {
            acceleration = acceleration.normalized * maxAcceleration;
        }

        // 残り生存時間を減少させる
        time -= Time.deltaTime;

        // 残り生存時間が 0 以下ならば処理を終了
        if (time < 0f)
        {
            return;
        }

        // 速度に加速度を加算し、位置を更新する
        velocity += acceleration * Time.deltaTime;
        position += velocity * Time.deltaTime;

        // 弾の位置を更新
        thisTransform.position = position;

        // 弾の向きを速度に合わせて更新
        thisTransform.rotation = Quaternion.LookRotation(velocity);
    }
}
```

Player

Is　Trigger　オン

```
using System.Collections;
```

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

public class PlayerHPManager : MonoBehaviour
{
    private int PlayerHP = 10;

    private void OnTriggerEnter(Collider other)
    {
        if(other.CompareTag("EnemyShell"))
        {
            Destroy(other.gameObject);

            PlayerHP -= 1;
            Debug.Log(PlayerHP);
            if(PlayerHP < 1)
            {
                SceneManager.LoadScene("Gameover");;
            }
        }
    }
}
```

Enemy

子に空のオブジェクト　EnemyShot を作る
EnemyShot に BulletSpawner をアタッチ

```
//https://xn--
sckyeodz49lj8c.com/unity%E3%81%A7%E3%83%9B%E3%83%BC%E3%83%9F%E3%83%B3%E3%
82%B0%E5%BC%BE%E3%82%92%E4%BD%9C%E3%81%A3%E3%81%A6%E3%81%BF%E3%82%8B/

using System.Collections;
```

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

public class BulletSpawner : MonoBehaviour
{
    // 弾のプレハブ
    public GameObject Bullet;

    // 弾を生成する回数
    public int iterationCount;

    // 弾を生成する間隔
    public float interval = 0.1f;

    // 弾を生成する間隔を待つためのオブジェクト
    private WaitForSeconds intervalWait;

    // 弾が生成されてから消滅するまでの時間
    private float destroyTime = 0;

    // 初期化処理
    void Start()
    {
        // 弾の生成間隔を設定
        intervalWait = new WaitForSeconds(interval);

        // 弾の生成を開始する
        StartCoroutine("BulletSpawn");
    }

    // 更新処理
    void Update()
    {
        // 消滅までの時間を更新
        destroyTime += Time.deltaTime;
```

```
        // 一定時間が経過したら自身を破棄する
        if (destroyTime > 6.0f)
        {
            //Destroy(this.gameObject);
        }
    }

    // 弾の生成コルーチン
    IEnumerator BulletSpawn()
    {
        // 指定された回数だけ弾を生成する
        for (int i = 0; i < iterationCount; i++)
        {
            // 指定された間隔だけ待機する
            yield return intervalWait;

            // 弾を生成し、生成位置と回転を設定する
            Instantiate(Bullet, transform.position,
Quaternion.identity).GetComponent<BulletController>();
        }
    }
}
```