ポートフォリオ

目次

Ｉ 自己紹介

profile

- 名前　　かみつる　しゅうた
　　上鶴　柊太
- 性　　男
- 年齢　　２２歳
- 出身地　　鹿児島

Career

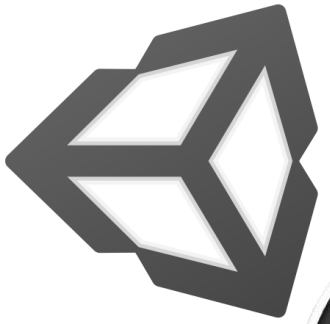
- 九州ポリテクカレッジ
　　(4年制)卒業
- ヒューマンアカデミー鹿児島校　プログラマ専攻

I 自己紹介

Skill

Game Engine

- Unity



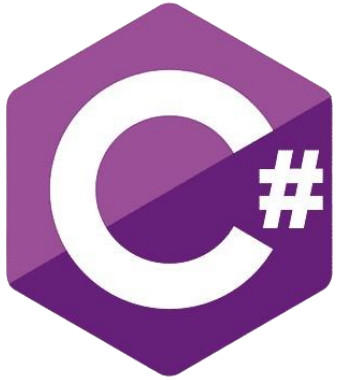
- UE5

Modeling

- Blender

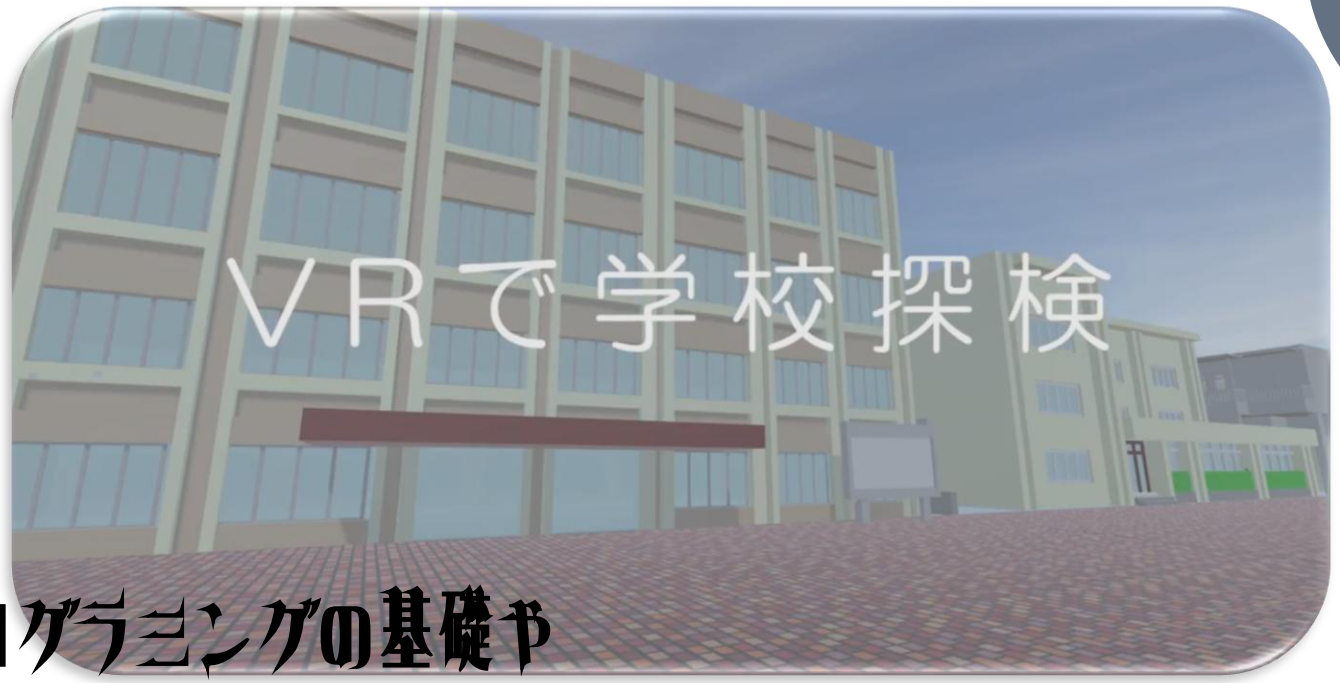
Language

- C C++ C# java
  python php VS...



Ⅰ 自己紹介

Achievements



昨年までの学校では主にプログラミングの基礎や
情報電子分野について幅広く学んでいました

その中でも卒業制作で15人程度のチームでのロボット
制作を経験し生産現場に近い開発を行ってきました

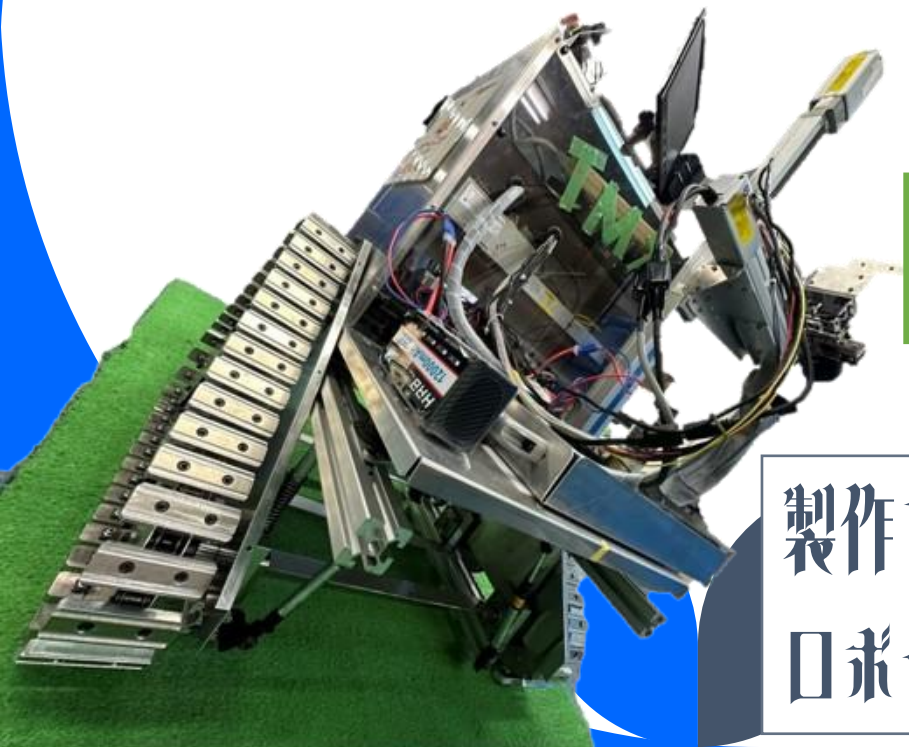
Ⅰ 自己紹介

Achievements

大会で提出した紹介動画

RKB毎日放送より

テレビによる紹介記事



製作したトマト収穫ロボットの外観図

Ⅱ 作品紹介

Ⅰ TrialsUnleashed

ジャンル	コマンドアクション
開発環境・言語	Unity6・C#
開発期間	1週間(制作中)
人数	1人

Ⅱ 作品紹介

Ⅰ TrialsUnleashed

・敵の攻撃を避けたりギミックを処理しながら様々なアクションを用いてモンスターの討伐を目指すゲーム

Ⅱ 作品紹介

Ⅰ TrialsUnleashed

Code

トーラス型判定の実装

- ✓ 外側・内側コライダーによる多重判定の分離設計。
- ✓ 安全地帯管理のための明確なメソッド分離。

```
void OnDisable()
{
    // 外側コライダーに接触しているが、内側コライダーに接触していないプレイヤーにダメージ
    foreach (var player in playersInRange)
    {
        if (player == null) continue;

        // 内側コライダーに接触していない場合のみダメージ
        if (!playersInSafeZone.Contains(player))
        {
            Debug.Log($"プレイヤーに{attackData.damage}ダメージ！（AoE消滅時/外側のみ接触）");
            var playerStatus = player.GetComponent<PlayerStatus>();
            if (playerStatus != null)
            {
                playerStatus.TakeDamage((int)attackData.damage);
            }
        }
        else
        {
            Debug.Log($"プレイヤー{player.name}は内側コライダーに接触しているためダメージ無効");
        }
    }

    // 多重実行防止
    playersInRange.Clear();
    playersInSafeZone.Clear();
}
```

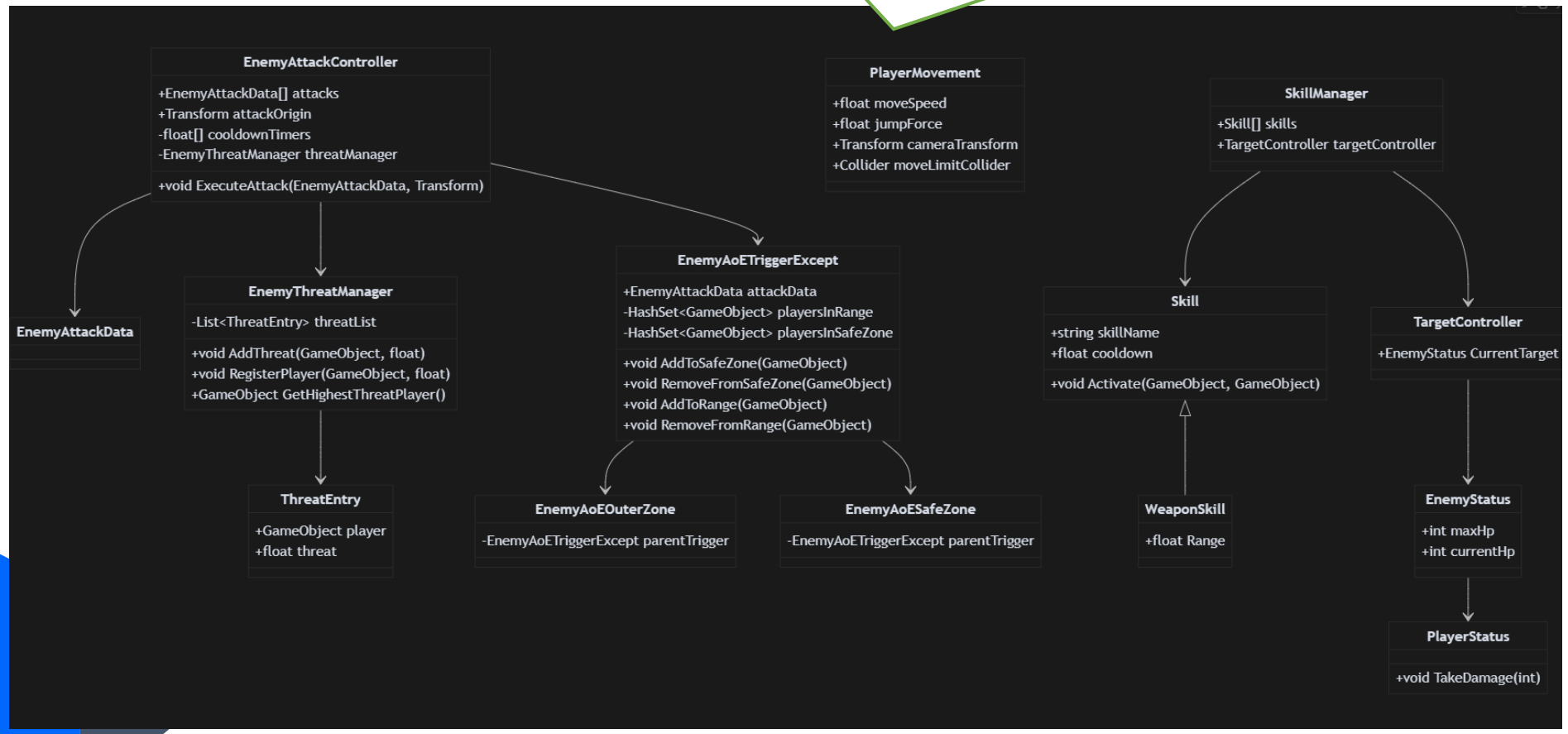
```
// 内側コライダーからの通知用メソッド
public void AddToSafeZone(GameObject player)
{
    playersInSafeZone.Add(player);
    Debug.Log($"内側コライダーに接触（安全地帯）: {player.name}");
}

public void RemoveFromSafeZone(GameObject player)
{
    playersInSafeZone.Remove(player);
    Debug.Log($"内側コライダーから離脱（安全地帯解除）: {player.name}");
}

// 外側コライダーからの通知用メソッド
public void AddToRange(GameObject player)
{
    playersInRange.Add(player);
    Debug.Log($"外側コライダーに接触: {player.name}");
}

public void RemoveFromRange(GameObject player)
{
    playersInRange.Remove(player);
    Debug.Log($"外側コライダーから離脱: {player.name}");
}
```

Ⅱ 作品紹介

Ⅰ TrialsUnleashed

- ✓ 左側は主にEnemyに関するクラス
- ✓ 右側は主にPlayerに関するクラス

Class Diagrams



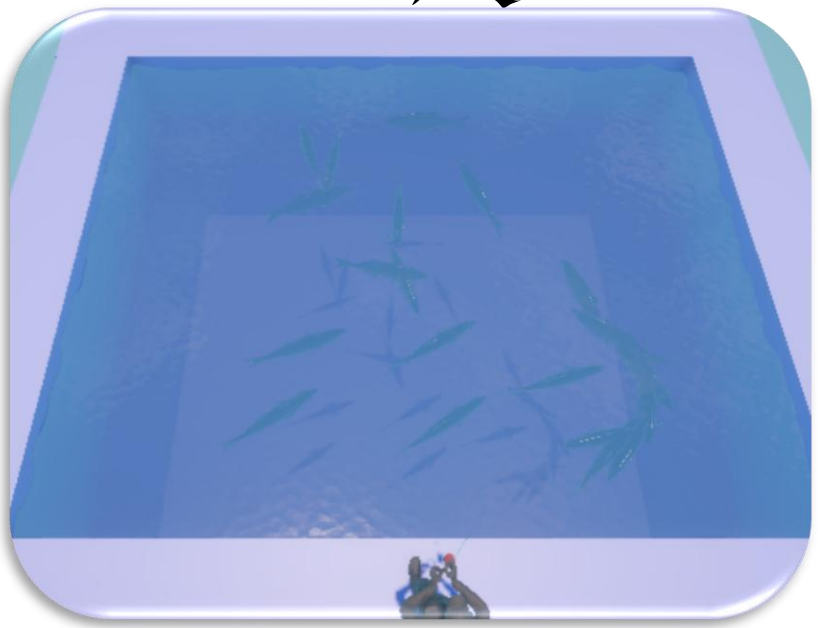
Ⅱ 作品紹介

Ⅱ FishingQTE

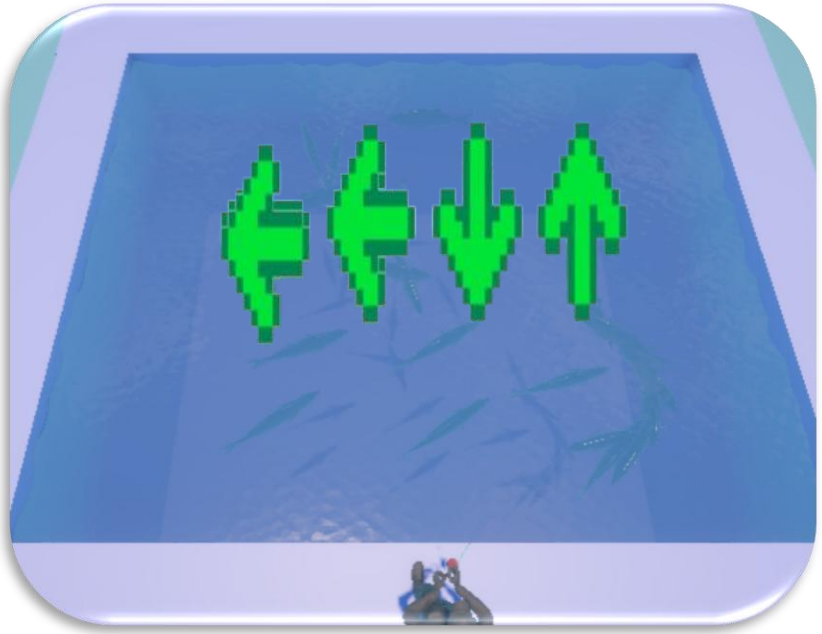
ジャンル	釣り+QTE
開発環境・言語	Unity6・C#
開発期間	2か月
人数	1人

Ⅱ 作品紹介

Ⅱ FishingQTE



一般的な釣りの動作



キャストからファイト
アクション(QTE)

Ⅱ 作品紹介

Ⅱ FishingQTE

Code

```
4   public class FishingGameControl
5   {
6       public enum FishingState
7       {
8           Idle,
9           Casting,
10          Waiting,
11          Hooked,
12          Reeling,
13          Finished
14      }
```

状態管理の実装

- ✓ FishingGameControllerでの状態管理がとても整理されている。
- ✓ 各状態の遷移が明確で、状態に応じた処理が適切に分離されている。

Ⅱ 作品紹介

Ⅱ FishingQTE

Code

```
4   public class FishMover : MonoBehaviour
5   {
6       public BoxCollider swimArea;
7       public float moveSpeed = 1f;
8
9       private Transform hook;
10      private Vector3 targetPosition;
11
12      private enum State { Swimming, ApproachingHook, Nibbling, Hooked }
13      private State currentState = State.Swimming;
14
15      private float nibbleTimer = 0f;
16      private int nibbleCount = 0;
17
18      private static FishMover currentHookedFish = null;
19      public static FishMover CurrentHookedFish => currentHookedFish;
```

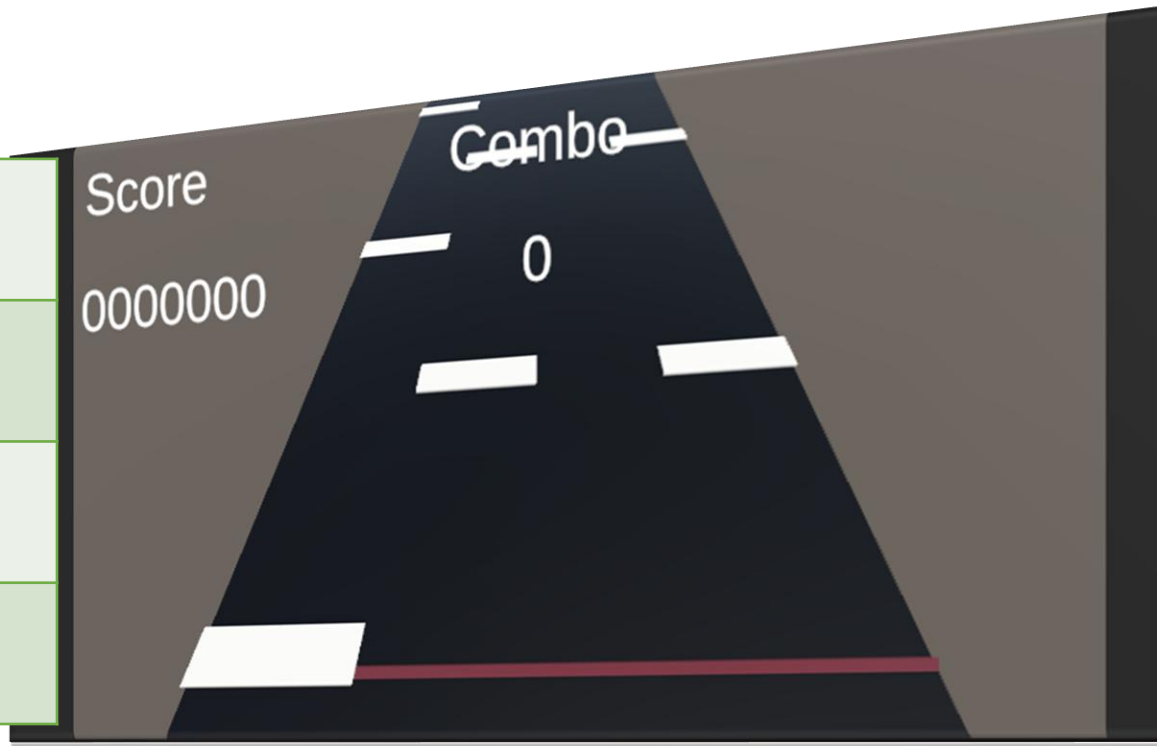
コンポーネント間の疎結合な設計

- ✓ シングルトンパターンを適切に使用し必要な情報の共有を実現。
- ✓ コンポーネント間の依存関係が明確で、保守性が高い。

Ⅱ 作品紹介

Ⅲ KeyRythm

ジャンル	リズムゲーム
開発環境・言語	Unity2020・C#
開発期間	1か月弱
人数	1人



Ⅱ 作品紹介

Ⅲ KeyRythm

Code

データ構造の設計

- ✓ DataクラスとNoteクラスを適切に定義し、JSONデータとの相互変換を容易にしている。
- ✓ ノートの情報をList<T>を使用して効率的に管理。
- ✓ BPM、LPB (Lines Per Beat) などの音楽理論に基づいた正確な時間計算を実装。

```
[Serializable]
public class Data
{
    public string name;
    public int maxBlock;
    public int BPM;
    public int offset;
    public Note[] notes;
}
[Serializable]
public class Note
{
    public int type;
    public int num;
    public int block;
    public int LPB;
}
```

```
    for (int i = 0; i < noteNum; i++)
    {
        float interval = 60 / (inputJson.BPM * (float)inputJson.notes[i].LPB);
        float beatSec = interval * (float)inputJson.notes[i].LPB;
        float time = (beatSec * inputJson.notes[i].num / (float)inputJson.notes[i].LPB) + inputJson.offset * 0.01f;
        NotesTime.Add(time);
        LaneNum.Add(inputJson.notes[i].block);
        NoteType.Add(inputJson.notes[i].type);

        float z = NotesTime[i] * NotesSpeed;
        GameObject note = Instantiate(noteObj, new Vector3(inputJson.notes[i].block -1.5f, 0.55f, z), Quaternion.identity);
        NotesObj.Add(note);
        Destroy(note, 8.0f + GManager.instance.StartTime);
    }
}
```

Ⅲ今後の展望

ゲームプログラマーとして、より高いクオリティの作品を追求し、技術と表現力を磨き続けたいと考えています。

プレイヤーにとって印象に残る、心を動かすようなゲーム体験を提供できるクリエイターになることが目標です。

そのために、ユーザー視点を大切にしながら、細部までこだわったゲーム作りを心がけていきます。

Ⅳ動画&PDFダウンロードリンク

作品投稿用YouTubeアカウント

PDFダウンロードQRコード

※随時更新予定

ご観覧ありがとうございました