ポートフォリオ

総合学園ヒューマンアカデミー

横浜校

ゲームカレッジ　プログラマー専攻

オオサワ　コウセイ
大澤光生

プロフィール

名前	オオサワ　コウセイ 大澤　　光生	年齢:19	性別:男

好きなゲームジャンル：FPS、シミュレーションなど

希望職種：プログラマー

所持スキル	
言語	C#, C/C++, PHP, JavaScript, ActionScript3.0
開発環境	Visual Studio 2022, Unity, AdobeAnimate
開発支援ツール	GitHub, Adobe FhotoShop, XAMPP, Microsoft Office

自己PR

私は、作品制作において、新しい技術や独自で考えた数式を扱うことが多かったので、コメントの言葉の表現を気を付けて同じチームのプログラマーにわかりやすく伝わるように意識しました。

今後は、コードを書く際、より伝わるかつメモリーを気にするコードを書くようにこだわります。

目次

Unity

3D アクションゲーム

開発環境	Unity 2022.3.10f1
使用ツール	Microsoft Visual Stadio2022
使用言語	C#
動作環境	PC(キーボード&マウス/コントローラー)
制作期間	約3か月間(2023. 9月～12月頃)
制作人数	9人
	プログラマー　3人
	プランナー　2人
	CGデザイナー　4人
担当箇所	プログラマー：
	プレイヤーやカメラなどを担当

ゲーム概要

『雪だるまを転がしゴールを目指す3Dアクションゲーム』
雪だるまを転がし、ブレーキをうまく使い障害物を避け
タイムや雪だるまがなくなる前にゴールを目指す、
スピードアクションゲーム。

Unity 3D アクションゲーム

- カメラの自動遷移

ChinemachineのFreelookCameraを使い、段階的にカメラが遷移するというコードを書きました。

```
// カメラの状態変化
6 個の参照
public enum CameraState
{
    // 一番最初のカメラの引き、高さ
    firstCamera,
    // 二番目のカメラの引き、高さ
    secondCamera,
}
// 初期のカメラの引きや高さを決める
CameraState currentState = CameraState.firstCamera;

// PlayerScaleがtoChangeScaleより小さい時に扱います。
[System.Serializable]
2 個の参照
public struct FirstCamera
{
    // カメラの引き
    public float Ridius;
    // カメラの高さ
    public float Height;
    // 視野角
    public float Fov;
    // LookTagetの高さ(高いほどキャラクターが画面の下のほうに写る)
    public float LookTagetHeight;
}
[SerializeField]
private FirstCamera firstCamera = new FirstCamera();

// PlayerScaleがtoChangeScaleより小さい時に扱います。
[System.Serializable]
2 個の参照
private struct SecondCamera
{
    // カメラの引き
    public float Ridius;
    // カメラの高さ
    public float Height;
    // 視野角
    public float Fov;
    // LookTagetの高さ(高いほどキャラクターが画面の下のほうに写る)
    public float LookTagetHeight;
}
[SerializeField]
private SecondCamera secondCamera = new SecondCamera();
```

列挙型と2つの構造体をつくり、プレイヤーの大きさによって、Stateを管理しカメラの遷移に関する関数を呼び出してます。

```
// Update is called once per frame
Unity メッセージ|0 個の参照
void Update()
{
    switch (currentState)
    {
        case CameraState.firstCamera:
            UpdateForCamera(firstCamera.Ridius, firstCamera.Height, firstCamera.Fov, firstCamera.LookTagetHeight);
            break;
        case CameraState.secondCamera:
            UpdateForCamera(secondCamera.Ridius, secondCamera.Height, secondCamera.Fov, secondCamera.LookTagetHeight);
            break;
    }
    // Playerのscaleを参照する。
    Vector3 Scale = body.transform.localScale;

    // BodyのScaleによって、カメラの状態を変える
    if (Scale.x < toChangeScale)
    {
        currentState = CameraState.firstCamera;
    }
    else if (Scale.x >= toChangeScale)
    {
        currentState = CameraState.secondCamera;
    }
}
```

Unity 3D アクションゲーム

- カメラの自動遷移

右の画像が実際に遷移している様子です。

前ページの関数の中身は下の画像みたいに単純でこの部分をカメラの因子分繰り返し書いてるものです。

ただし、Fovだけ遷移スピードは変えてます。

```
// カメラの状態によって、それにあわせるようにする。
2 個の参照
private void UpdateForCamera(float ridius, float hight, float Fov, float TagetHight)
{
    // Scaleが一定以上になったときに、カメラの引きを上げる
    if (ridius > freeLookCamera.m_Orbits[0].m_Radius)
    {
        freeLookCamera.m_Orbits[0].m_Radius += changeSpeed * Time.deltaTime;
        freeLookCamera.m_Orbits[1].m_Radius += changeSpeed * Time.deltaTime;
        freeLookCamera.m_Orbits[2].m_Radius += changeSpeed * Time.deltaTime;

        // UpdateForCameraが無限に作動しないようカメラの引きを固定する
        if (ridius < freeLookCamera.m_Orbits[0].m_Radius)
        {
            freeLookCamera.m_Orbits[0].m_Radius = ridius;
            freeLookCamera.m_Orbits[1].m_Radius = ridius;
            freeLookCamera.m_Orbits[2].m_Radius = ridius;
        }
    }
    // Scaleが一定以下になったときに、カメラの引きを下げる
    else if (ridius < freeLookCamera.m_Orbits[0].m_Radius)
    {
        freeLookCamera.m_Orbits[0].m_Radius -= changeSpeed * Time.deltaTime;
        freeLookCamera.m_Orbits[1].m_Radius -= changeSpeed * Time.deltaTime;
        freeLookCamera.m_Orbits[2].m_Radius -= changeSpeed * Time.deltaTime;

        // UpdateForCameraが無限に作動しないようカメラの引きを固定する
        if (ridius > freeLookCamera.m_Orbits[0].m_Radius)
        {
            freeLookCamera.m_Orbits[0].m_Radius = ridius;
            freeLookCamera.m_Orbits[1].m_Radius = ridius;
            freeLookCamera.m_Orbits[2].m_Radius = ridius;
        }
    }
```

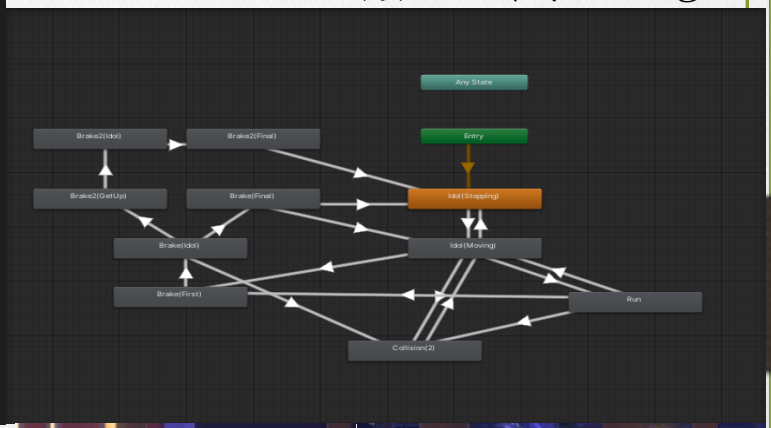
Unity 3D アクションゲーム

- アニメーション

Title、Stageのスタート、GameClear、プレイヤーのAnimatorを作り、プレイヤーに関してはCGからもらったデータを元にAnimationも作っています。

※プレイヤーのもの

```
case InputActionPhase.Canceled:
    if (headrigidbody.velocity.magnitude > 0)
    {
        // Brake(Final)アニメーションに切り替える
        animator.SetTrigger(BrakeEndId);
        if (currentState == PlayerState.Brake)
        {
            // 現在のプレイヤー状態変化
            currentState = PlayerState.Walk;
        }
    }
    else if (headrigidbody.velocity.magnitude == 0)
    {
        // Brake(Final)アニメーションに切り替える
        animator.SetTrigger(Brake2EndId);
        if (currentState == PlayerState.Brake)
        {
            // 現在のプレイヤー状態変化
            currentState = PlayerState.Idol;
        }
    }
    break;
```



また、マフラーが埋まってしまっていたので(左画像)、プレイヤーの最初と現在のスケールの違いによって(下画像参照)、最初のマフラーから比例の関係で現在のマフラーの角度を出しています。

```
1 個の参照
void SetScarf()
{
    // 最初の大きさから現在の大きさを引く
    var bodyScalediff = firstBody - body.transform.localScale;
    // 大きさの違いが0より下の場合のみ行う
    if (bodyScalediff.x < 0)
    {
        // LeftとRightはxの値が違うのそれぞれ行う
        var Scarfrotation_L = firstScarfrotation_L;
        var Scarfrotation_R = firstScarfrotation_R;
        // Bodyの大きさの違いによって、値を出す。
        Scarfrotation_L.z = Scarfrotation_L.z + (scarfRotationSpeed * bodyScalediff.x);
        Scarfrotation_R.z = Scarfrotation_L.z;
        if (Scarfrotation_L.z > 0)
        {
        // 代入
        scarfLeft.transform.localEulerAngles = Scarfrotation_L;
        scarfRight.transform.localEulerAngles = Scarfrotation_R;
        }
    }
}
```

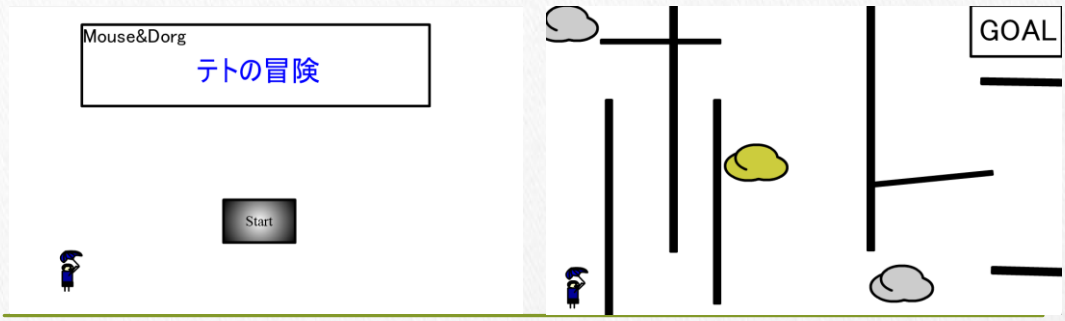
```
scarf 55, -12    67
body   1,   7     6
                 67 / 6 = 11.1666

scarf 55, 43.8333, 32.6667 角度

body   1,  2     ,  3      大きさ
```

Adobe Animate CC

2D イライラ棒ゲーム



開発環境	Adobe Animate CC
使用ツール	Adobe Illustrator
使用言語	Action Script 3.0
動作環境	PC(キーボード&マウス/コントローラー)
制作期間	約6日(2023. 7月頃)
制作人数	1人
担当箇所	すべて

ゲーム概要

『イライラ棒を通り抜けゴールを目指す2Dゲーム』

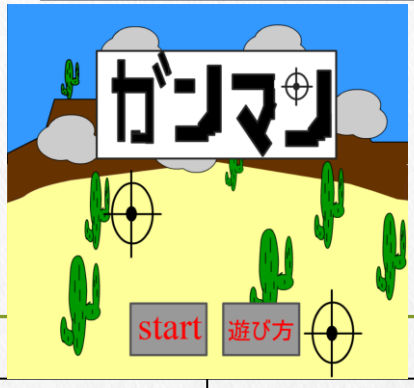
制作後記

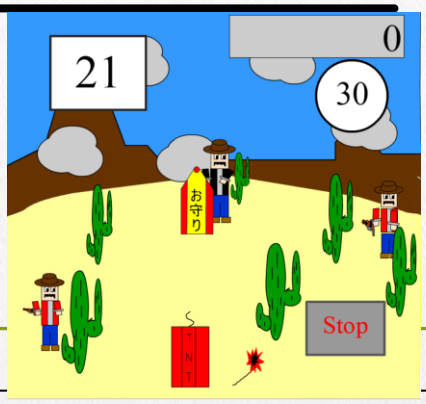
7月頃は、1ステージのみの制作方法しか教えられていなかったので、2ステージ以上の作成方法を思いついたので行動に移しました。(おそらく2ステージ以上作ったのは私のみ)

初めて作ったゲームというのもあり、バグに苦しんだが、ゲーム作りが楽しいと思えた一作。

Adobe Animate CC

2D ガンアクションゲーム





開発環境	Adobe Animate CC
使用ツール	Adobe Illustrator
使用言語	Action Script 3.0
動作環境	PC(キーボード&マウス/コントローラー)
制作期間	約3日(2024.1月頃)
制作人数	1人
担当箇所	すべて

ゲーム概要

『敵を撃つことによって点数を稼ぐスコアゲーム』

制作後記

事情によって取り組む時間が極端に少なくなってしまってたので、画像作成をしながら、構想を練って作りました。制作内訳は、画像作成2日残りの時間をプログラムに費やしました。