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

石森 颯

目次

プロフィール

カナ	: イシモリ タツキ
氏名	: 石森 颯
年齢	: 18歳
希望職種	: プログラマー

対応可能言語

C++ / C# / Microsoft JavaScript
ActionScript / HTML

使用開発環境

Unity / Microsoft Visual Studio
Git(Git for Windows, Git)
Adobe Animate
Adobe Photoshop / Adobe Illustrator

情報共有ツール

Git(Git for Windows, Git)
Google Drive / Slack / Discord

自己PR

私は常に与えられた役割を徹底的にこなすこと、そしてより良い結果を求めることを意識しています。
チーム制作では主にUI動作やシーンの制御などを担当しました。
現在は、Direct3D 11についての学習を進めています。
今後は、様々なネイティブアプリケーション作品の作成に挑戦したいと考えています。

3Dアクションゲーム

作品名	: Rolling Snow
使用言語	: C#
動作環境	: コントローラー
制作人数	: プランナー2人 / プログラマー3人 CGデザイナー4人
担当場所	: 主にUI
開発期間	: 3ヶ月
使用ツール	: Microsoft Visual Studio 2022 Unity ver.2022.3.11f1 / GitHub

ゲーム概要

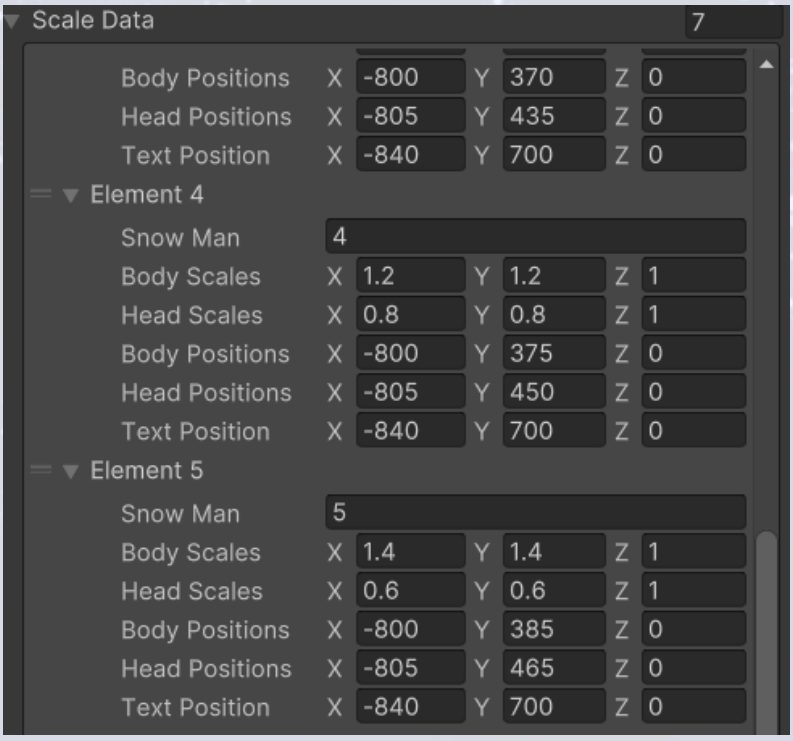
3D空間で時間以内にゴールを目指すスピードアクションです。
プレイヤーは雪だるまの精霊となり、朝を迎えて溶けてしまう前に
子供たちの待つ家にたどり着かなければなりません。
しかし、周りは家や木々が生い茂るすこし危険な道。
果たして雪だるまは無事にたどり着けるのでしょうか？

常に変化するUI

```
private void Update()
{
    if (snowmanObject == null || scaleData == null || scaleData.Length == 0)
    {
        return;
    }

    float snowmanScale = Mathf.Max(0f, snowmanObject.transform.localScale.x);

    // スケールが2以下かつ0.8以上の場合はデフォルトの値を適用
    if (snowmanScale <= 2f && snowmanScale >= 0.8f)
    {
        SetUIProperties(defaultBodyScale, defaultHeadScale, defaultBodyPosition, defaultHeadPosition);

        return;
    }
    //Debug.Log(snowmanScale);
    int index = GetScaleIndex(snowmanScale);

    if (index < 0 || index >= scaleData.Length)
    {
        return;
    }

    // スケールが指定された数値と同じになったら、その段階の値を適用
    SetUIProperties(scaleData[index].bodyScales, scaleData[index].headScales, scaleData[index].bodyPositions, scaleData[index].headPositions);
    maxText.transform.localPosition = scaleData[index].textPosition;
}
```

雪だるまの大きさに応じてUIが変化するように。
まず、「頭」と「胴体」に分かれた雪だるまのUIを用意した。
雪だるまのサイズに応じてUIのスケールを変化させるためのメソッドを作成した。

```
[System.Serializable]
1 個の参照
public class ScaleData
{
    public float snowMan;
    public Vector3 bodyScales;
    public Vector3 headScales;
    public Vector3 bodyPositions;
    public Vector3 headPositions;
    public Vector3 textPosition;
}
```



ScaleDataクラスで指定したい数値をまとめることで
Unity側でまとめてスケールやポジションを変更できるように。
雪だるまの体がどこまで大きくなればUIが変化するのか、と言った条件も変更可能。

リザルト画面の評価UI


```
void Start()
{
    // クリアタイムの獲得
    playerTime = 300.0f - PlayerPrefs.GetFloat("PlayerTime", 0.0f);

    // Image達の初期化
    SetImageStates(true, true, true,
                   false, false, false);
```

ゴールをした時点でPlayerPrefsにクリアタイムが保存される。
スクリプト側で保存されたタイムを呼び出し、クリアタイムが評価の条件を満たしているか否かで星の量やメーターの長さが変わる。

```
private void SetImageStates(params bool[] states)
{
    if (states.Length != 6)
    {
        return;
    }

    RankStar0.enabled = states[0];
    RankStar1.enabled = states[1];
    RankStar2.enabled = states[2];
    OnRankStar0.enabled = states[3];
    OnRankStar1.enabled = states[4];
    OnRankStar2.enabled = states[5];

}
```

```
public void SetStar()
{
    // 1分以上
    if (StarControl == 1 && playerTime >= 90.0f)
    {
        SetImageStates(false, true, true, true, false, false);
    }
    // 2分以上
    else if (StarControl == 2 && playerTime >= 150.0f)
    {
        SetImageStates(false, false, true, true, true, false);
    }
    // 3分以上
    else if (StarControl == 3 && playerTime >= 210.0f)
    {
        SetImageStates(false, false, false, true, true, true);
    }
}
#endregion
```

評価に使用される星はそれぞれ番号を割り振られている。
SetStarにて評価の条件によってどの星をfalseにするかtrueにするかが変わっている。
この時、番号を予め割り振っているおかげで、どれがどの星なのかが分かりやすくなっている。

アニメーションの停止防止


```
// ゲームを一時停止します。
1 個の参照
public void Pause()
{
    if (gameState == SceneState.Play)
    {
        if (!IsPaused)
        {
            IsPaused = true;
            Time.timeScale = 0;
            animator.updateMode = AnimatorUpdateMode.UnscaledTime;
            pauseUI.Show();
        }
        else
        {
            Time.timeScale = 1;
            animator.updateMode = AnimatorUpdateMode.Normal;
            IsPaused = false;
            pauseUI.Hide();
        }
    }
}
```

ポーズ画面が出現するとき、ゲーム自体の全ての動きが停止する。
この時、アニメーションの動きまで止まってしまう。
そこで、UnscaledTimeを使うことでアニメーションは動作するようになっている。

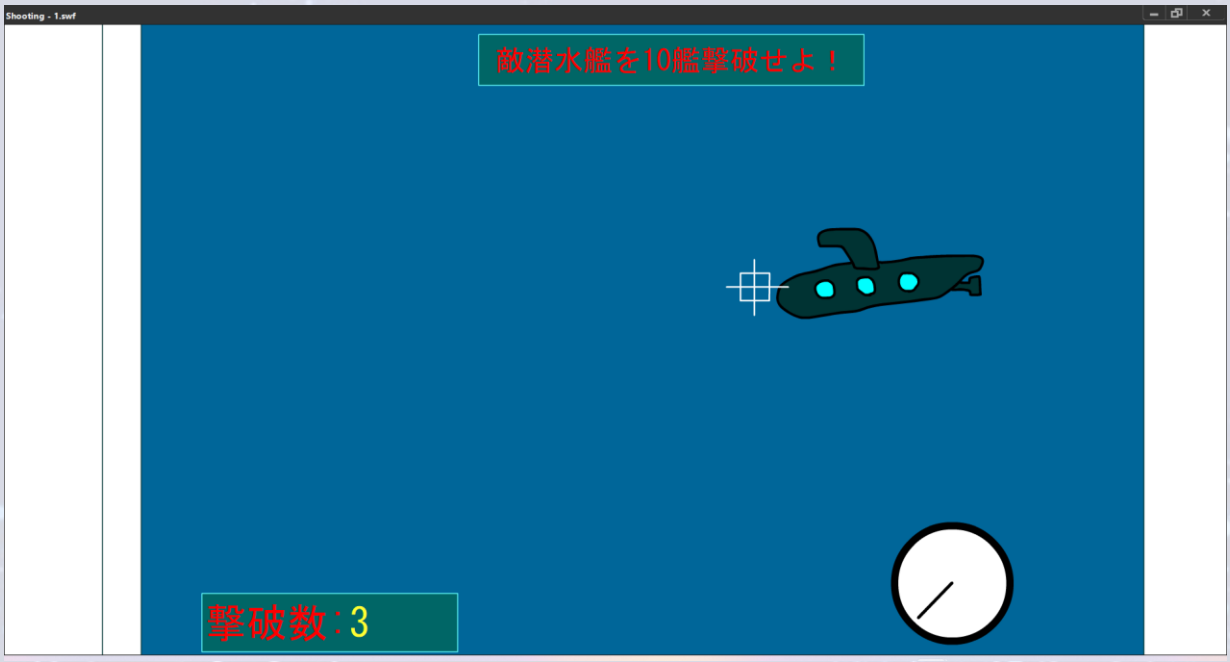
Animate作品

作品名　　：MarineMission
使用言語　　：ActionScript 3.0
動作環境　　：マウス
開発期間　　：2日
使用したツール　：AdobeAnimate 2023

ゲーム概要

プレイヤーは戦艦となり、続々と現れる敵艦隊倒さなければなりません。
敵艦隊は合計10艦存在しており、無事にすべて撃破することが出来ればゲームクリアです。
制限時間は10秒です。もしも10秒を超えてしまった場合はゲームオーバーとなります。

遊び方



操作は簡単かつ直感的で、敵がいる位置にカーソルを合わせてクリックで撃つだけです。
右下にある時計は残り時間で、針が一周するとゲームオーバー画面に移行します。

このゲームの特徴の一つとして、普段パソコンを使用しているときは矢印の形をしているカーソルがゲーム中は戦艦の照準へと変化します。
この仕様により、ゲームの臨場感が増しています。