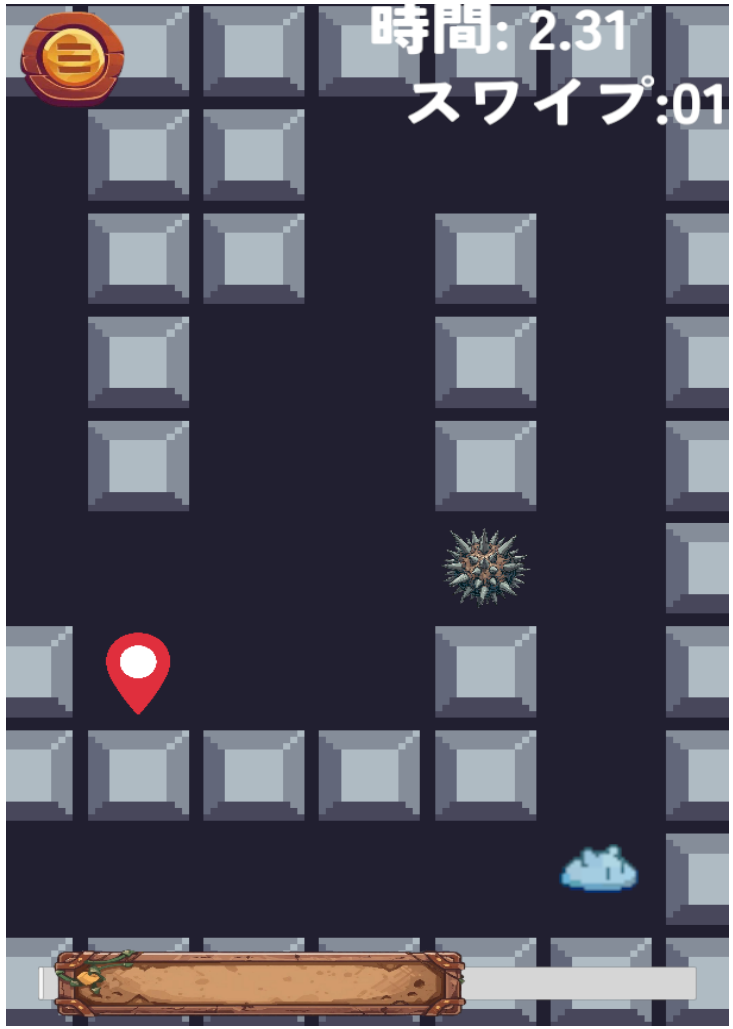
プログラム説明
作品名「SlideDungeon」

横浜デジタルアーツ専門学校 ゲーム科
2年　新井 桜大

SLIDE DUNGEON



SlideDungeon

GitQR：

ジャンル: パズルゲーム
動作環境: Android、IOS
開発環境: C# / Unity
使用ツール: Visual Studio
開発期間: 2025年8月29日～9月25日 約60時間

紹介動画:
GitURL :https://github.com/araioudai/SlideDungeon.git

作品概要:

2年次の個人制作作品で、モバイルゲームとして開発。Jsonを利用したオフラインランキング、PlayFabを利用したオンラインランキングを実装。csvを用いたステージロードとScriptableObjectを用いたステージ基準管理で無限に拡張可能。

作品概要としては、壁にぶつかるまで止まらないスライムをスライドで操作し、早いクリアタイムを目指すパズルカジュアルゲーム。

SlideDungeon

アピールポイント・こだわった点

- マネージャー設計
- ステージ基準管理
- ステージごとのマップロード
- ランキング機能
- プレイヤー体験を意識した最適化
- UI設計

マネージャー設計

シングルトンによる統一管理：シーンを跨いで一貫したアクセスを実現
責務分割：サウンド、進行管理などを分離し保守性を向上
高い再利用性：可読性と安定性を両立した設計

```
#region シングルトン（他のスクリプトからInstanceでアクセスできるようにする）
public static StageIndex Instance { get; private set; }
#endregion

void Awake()
{
    //シングルトン管理
    if (Instance != null && Instance != this)
    {
        Destroy(gameObject); //既にInstanceがあれば自分を破棄
        return;
    }
    Instance = this;
    DontDestroyOnLoad(gameObject);
}
```

・StageIndex：ステージ番号管理（Set / Get）
・RankingManager / OffLineRankingManager / DebugMode：ランキング処理、ランキング切り替え

・シーンを跨いで必要な情報を保持
ランキングやステージ番号などゲーム進行に必須なデータは DontDestroyOnLoad で保持。

```
#region シングルトン（他のスクリプトからInstanceでアクセスできるようにする）
public static GameManager Instance { get; private set; }
#endregion

void Awake()
{
    //シングルトン管理
    if (Instance != null && Instance != this)
    {
        Destroy(gameObject); //既にInstanceがあれば自分を破棄
        return;
    }
    Instance = this;
}
```

・GameManager / TitleManager：ゲーム全体の進行制御
SoundManager / CameraManager：サウンド・演出制御（責務を分離）

・不要データの適切な破棄
シーンごとで限定的に使う情報はロード時に破棄し、メモリ効率を確保。
・安定性と効率性の両立
無駄なデータを残さない設計でパフォーマンス向上。

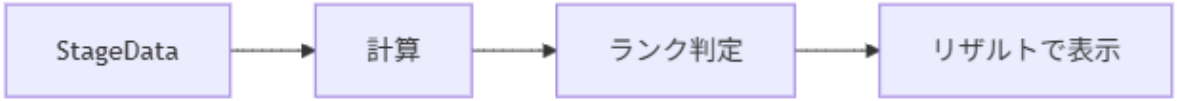
ステージ基準管理

SlideDungeon

データ駆動設計：基準値を ScriptableObject で管理し、コード編集不要で調整可能
柔軟性：非エンジニアでも調整でき、ステージ追加・拡張も容易
効率化：開発効率向上＋バランス調整コスト削減

```
using UnityEngine;

[CreateAssetMenu(menuName = "Stage/StageData")]
public class StageData : ScriptableObject
{
    public float baseTime;
    public int baseSlide;
}
```

・ステージ基準管理フロー図



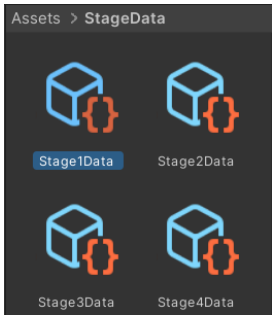
・計算方法

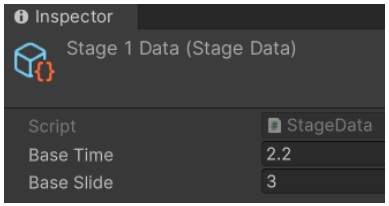
各項目を基準値と除算で正規化。各項目合計して、基準達成の2.0以内ならSrank

```
//現在のステージ番号を取得
stageNumber = StageIndex.Instance.GetIndex();
//該当ステージの基準データを取得
StageData data = stageDatas[stageNumber - 1];
//Sランク基準タイム
baseTime = data.baseTime;
//Sランク基準スライド数
baseSlide = data.baseSlide;

//スコア計算
//各項目を基準値で正規化する：1.0が基準達成、1.0未満なら基準より良い
float timeNorm = clearTime / baseTime;                //タイムの基準達成度
float slideNorm = (float)slideCount / baseSlide;      //スライド数の基準達成度
float score = timeNorm + slideNorm;                   //合計スコア（小さいほど高成績）2.0がピッタリ
```

・各ステージの基準値

Inspectorで設定可能。





ステージごとのマップロード

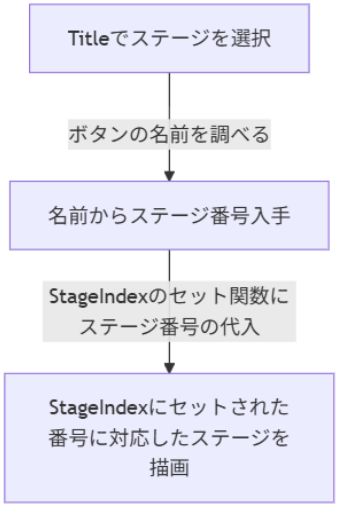
SlideDungeon

CSV管理：Excel等で非エンジニアも編集可能
スケーラブル：シーンを増やさず複数ステージを扱える
自動配置：CSVから座標を計算しワールドに正しくマップ配置

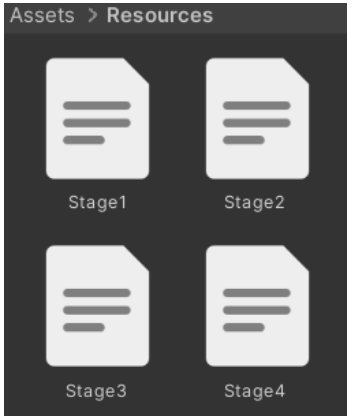
・**CSVマップロード**

```
for (int y = 0; y < lines.Length; y++)
{
    string[] values = lines[y].Trim().Split(',');

    for (int x = 0; x < values.Length; x++)
    {
        if (int.TryParse(values[x], out int value))
        {
            //タイルの描画場所
            Vector3Int cellPos = new Vector3Int(topLeftCell.x+x, topLeftCell.y-y, 0);

            switch (value)
            {
                case 0:
                    floorTilemap.SetTile(cellPos, floorBackTile);      //床タイル
                    break;
            .
            .
            .
```

・CSVでステージを柔軟に追加可能
・非エンジニアもデータ編集だけでマップ制作できる



・**CSVファイル(ステージデータ)**



ランキング機能

オンライン対応：PlayFab を利用したグローバルランキング
オフライン対応：Jsonによるローカル保存で展示会環境にも対応
拡張性：通信環境に左右されない柔軟な実装

・JSONを用いた永続化
Application.persistentDataPath を使い、端末ごとに安全に保存
JsonUtility.ToJson / FromJson を活用し、人間が読める形で保存・読み込み

```
//保存
private void SaveRanking()
{
    //trueで整形出力
    string json = JsonUtility.ToJson(rankingData, true);
    File.WriteAllText(filePath, json);
}
```

```
//読み込み
private void LoadRanking()
{
    if (File.Exists(filePath))
    {
        string json = File.ReadAllText(filePath);
        rankingData = JsonUtility.FromJson<RankingData>(json);
    }
    else
    {
        rankingData = new RankingData();
    }
}
```

4. 安全設計
•**EnsureStageExists関数**：ステージ数不足時にリストを拡張 → 例外を防止
•null チェックを適切に行い、安全にリストを返す。エラーを防ぐ。
•ResetAllRanking でデータをリセットできる → **テストや再プレイに便利**

```
//ランキングリセット
rankingData = new RankingData(); //全ステージのランキングデータを空にする
SaveRanking();                   //空データで上書き保存
```

```
//NullReferenceExceptionを防ぐ
var stageRanking = rankingData.stageRankings[stageIndex - 1];
if (stageRanking == null || stageRanking.scores == null)
{
    return new List<ScoreEntry>(); //空のリストを返す
}
```

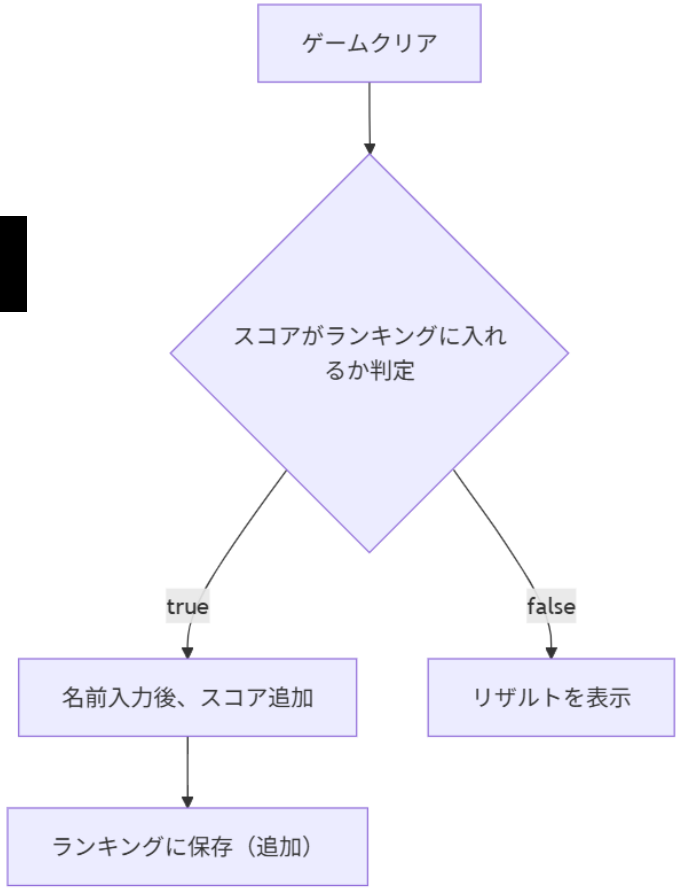
ランキング機能

オンライン対応：PlayFab を利用したグローバルランキング
オフライン対応：Jsonによるローカル保存で展示会環境にも対応
拡張性：通信環境に左右されない柔軟な実装

・ランキング処理の基本要件を満たす
上位9件だけ保持（容量を無駄にしない）
クリアタイムが早い順にソート（成績順が直感的に理解できる）
9件未満なら必ず登録可能 → 初期プレイヤーでも達成感がある

```
var stageRanking = rankingData.stageRankings[stageIndex - 1];
if (stageRanking.scores.Count < 9) return true; //9位未満なら必ず入れる
```

・オンライン／オフライン ランキングの切り替え
DebugMode.cs のフラグをinspector上で
変更すればランキングの切り替えが可能



プレイヤー体験を意識した最適化

SlideDungeon

操作フィードバックの強化：移動時にバイブレーションを加え、直感的な応答を実現
テンポを損なわない操作性：移動中のスライド操作で「先行入力」を受付、快適なプレイを維持

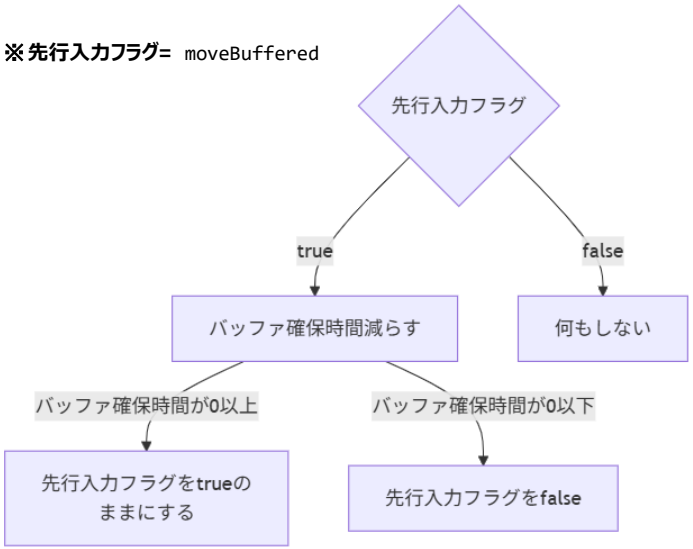
・移動時のバイブレーション

モバイル端末に備わっている Unity標準のバイブレーション機能を利用し、移動操作時にフィードバックを与える。
これにより、プレイヤーは入力に対して直感的で即時性のある反応を感じられる。

```
VibrationHelper.WeakVibrate(30, 30);
```

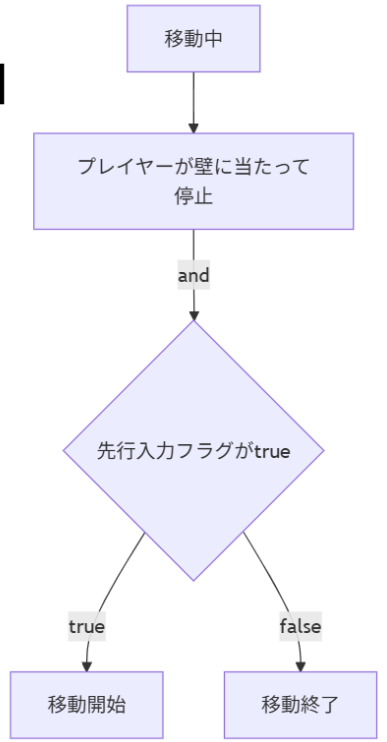
・先行入力有効時間

UpdateBuffer関数では、先行入力の有効時間を管理しています。




```
void UpdateBuffer()
{
    //バッファ時間の更新
    if (moveBuffered)
    {
        //バッファ確保時間減らす
        moveBufferTimer -= Time.deltaTime;
        if (moveBufferTimer <= 0f)
        {
            //時間切れで無効化
            moveBuffered = false;
        }
    }
}
```

・先行入力処理フロー図

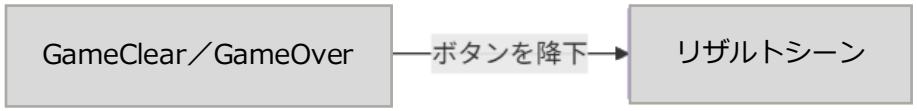


UI設計

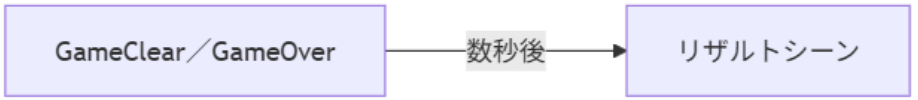
ユーザー目線のUI改善：無駄な入力やストレスを排除
マルチデバイス対応：タブレット展示用とモバイル用の両方に最適化
運用体制の信頼性：「タブレット展示用」と「モバイル用」GitHub＋SourceTreeによる確実なバージョン管理

・入力削減によるUI改善

修正前



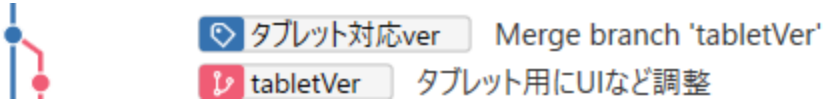
修正後



改善例：
結果画面への自動遷移を導入し、不要な入力を削減。これによりユーザー体験の改善と操作ストレスの低減を実現。特に繰り返しプレイ時のテンポ改善に効果を発揮。

・運用体制の信頼性

「タブレット展示用」と「モバイル用」を明確に区分し、GitHub＋SourceTreeでバージョン管理を実施。開発・更新時の混乱を防ぎ、安定した運用を保証。→両デバイス環境での欠損や不具合を防止。



SlideDungeon

- **apkファイル**

バイブレーション機能を追加し、より直感的な操作感を実現しました。
よろしければ、アプリケーション上で実際にお試しください。
https://drive.google.com/drive/folders/1cuFKtno9hMKwVpuzTVYWwcu9OGwUW3SL?usp=sharing

- **オンラインランキング**

現状は修正中の段階ですが、オンラインランキングの機能自体は既に実装済みです。必要であれば、UnityRoom 上で動作をご確認いただけます。
https://unityroom.com/games/testslide

SlideDungeon

- **UnityRoom**

UnityRoom上ではバイブレーション機能は手元で確認できませんが、最新版を公開しました。
よろしければ UnityRoom 上でプレイしてみてください。PCの場合は、
フルスクリーンで遊ぶと、画面サイズも適切に表示されます。
https://unityroom.com/games/slidedungeon