Andrei Inoue Hirata

Sistema de Recordes Genérico

Permitirá que você salve e carregue recordes

Este sistema permitirá que você salve e carregue recordes para diversas métricas em seus jogos, como "pontos_coletados," "obstaculos_pulados," "tempo_sobrevivencia," etc. Usaremos PlayerPrefs para salvar os dados localmente, o que é simples e eficaz para a maioria dos casos. Para dados mais complexos ou segurança online, outras abordagens como arquivos JSON ou bancos de dados seriam necessárias.

Sumário

Andrei Inoue Hirata

Desenvolvimento

Neste passo, vamos permitir que portões sejam abertos ao tocar em algum ativador. Isso adiciona uma mecânica cooperativa e interativa ao jogo.

Criando o Script

Iremos criar o ativador que será usado para ativar as portas para os elementos entrar.

1. Crie um script com o nome de S_ScoreManager
2. Copie o código

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

public static class S_ScoreManager
{
    [System.Serializable]
    public class ScoreEntry
    {
        public string scoreName; // Ex: "points_collected_level1", "jumps_flappy"
        public float scoreValue;

        public ScoreEntry(string name, float value)
        {
            scoreName = name;
            scoreValue = value;
        }
    }


    public static void SaveHighScore(string scoreType, float newValue, bool higherIsBetter =
true)
    {

        float currentHighScore = LoadHighScore(scoreType, higherIsBetter ? float.MinValue :
float.MaxValue);

        bool shouldSave = false;
        if (higherIsBetter)
        {
            if (newValue > currentHighScore)
            {
                shouldSave = true;
            }
        }
        else
        {
            if (newValue < currentHighScore)
            {
                shouldSave = true;
            }
        }

        if (shouldSave)
        {
            PlayerPrefs.SetFloat(GenerateScoreKey(scoreType), newValue);
            PlayerPrefs.Save();
            Debug.Log($"New high score saved for '{scoreType}': {newValue}");
        }
        else
        {
            Debug.Log($"The value {newValue} did not beat the current high score
({currentHighScore}) for '{scoreType}'.");
        }
```

```
    }


    public static float LoadHighScore(string scoreType, float defaultValue = 0f)
    {
        return PlayerPrefs.GetFloat(GenerateScoreKey(scoreType), defaultValue);
    }


    public static void ResetHighScore(string scoreType)
    {
        PlayerPrefs.DeleteKey(GenerateScoreKey(scoreType));
        PlayerPrefs.Save();
        Debug.Log($"High score for '{scoreType}' has been reset.");
    }

    public static void ResetAllTrackedHighScores()
    {


        Debug.LogWarning("ResetAllTrackedHighScores is a placeholder. " +
                         "To safely reset all scores managed by this system, you need to iterate
through your known scoreType keys and call ResetHighScore for each. " +
                         "Calling PlayerPrefs.DeleteAll() is dangerous as it clears ALL
PlayerPrefs for the project.");
         PlayerPrefs.DeleteAll();
         PlayerPrefs.Save();
    }

    private static string GenerateScoreKey(string scoreType)
    {
        return $"highscore_{scoreType}";
    }


    public static void SaveScoreToList(string listScoreType, float newScore, int maxListSize =
5, bool higherIsBetter = true)
    {
        List<float> scoreList = LoadScoreList(listScoreType, maxListSize);
        scoreList.Add(newScore);

        if (higherIsBetter)
        {
            scoreList.Sort((a, b) => b.CompareTo(a));
        }
        else
        {
            scoreList.Sort((a, b) => a.CompareTo(b));
        }


        if (scoreList.Count > maxListSize)
        {
            scoreList = scoreList.GetRange(0, maxListSize);
        }

        string scoresString = string.Join("|", scoreList);

        PlayerPrefs.SetString(GenerateScoreKey(listScoreType + "_list"), scoresString);
        PlayerPrefs.Save();
        Debug.Log($"Score {newScore} saved to list '{listScoreType}'. Current list:
{scoresString}");
    }


    public static List<float> LoadScoreList(string listScoreType, int maxListSize = 5)
    {
        string listKey = GenerateScoreKey(listScoreType + "_list");
        List<float> scoreList = new List<float>();

        if (PlayerPrefs.HasKey(listKey))
        {
            string scoresString = PlayerPrefs.GetString(listKey);
            string[] scoresArray = scoresString.Split('|');
```

```
            foreach (string s in scoresArray)
            {
                if (!string.IsNullOrEmpty(s) && float.TryParse(s,
System.Globalization.NumberStyles.Any, System.Globalization.CultureInfo.InvariantCulture, out
float score))
                {
                    scoreList.Add(score);
                }
            }
        }

        return scoreList;
    }

}
```

Salvar um Recorde:

Quando o jogador atingir uma pontuação que pode ser um recorde, chame a função SaveHighScore.

Como exemplo teremos 3 situações de jogos diferentes:

- Exemplo 1: Jogo de coletar itens (maior pontuação é melhor)

1. Escreva o código:

```
// Em algum script do seu jogo, após o jogador coletar itens:
int currentPoints = 150; // Pegue a pontuação atual do jogador
ScoreManager.SaveHighScore("total_collected_points", currentPoints);
```

- Exemplo 2: Jogo tipo Flappy Bird (maior número de obstáculos pulados é melhor)

2. Escreva o código:

```
// No script que controla os pulos:
int obstaclesJumped = 25;
ScoreManager.SaveHighScore("flappy_obstacles_jumped", obstaclesJumped);
```

- Exemplo 3: Recorde de tempo (menor tempo é melhor)

3. Escreva o código:

```
// Ao finalizar uma corrida/fase:
float finalTime = 45.7f; // em segundos
ScoreManager.SaveHighScore("race_track_X_time", finalTime, false); // false
indica que menor é melhor
```

Carregar um Recorde:

Para exibir o recorde em uma tela de menu, Game Over, etc.

1. Coloque esse código em um script que você já tenha para controlar a UI, ou crie um script novo e chame de S_GameUI

```
using UnityEngine.UI;

public Text textHighScoreCollect;
public Text textHighScoreJumps;
```

```
public Text textHighScoreTime;

void Start()
{
    // Carrega o recorde de coleta, se não existir, mostra 0
    float highScoreCollect = ScoreManager.LoadHighScore("total_collected_points");
    textHighScoreCollect.text = "High Score (Collect): " + highScoreCollect.ToString();

    // Carrega o recorde de saltos
    float highScoreJumps = ScoreManager.LoadHighScore("flappy_obstacles_jumped");
    textHighScoreJumps.text = "High Score (Jumps): " + highScoreJumps.ToString();

    // Carrega o recorde de tempo. Se não houver, pode mostrar um valor alto ou "N/A"
    // Para tempo (menor é melhor), o valor padrão se não existir deve ser float.MaxValue
    float highScoreTime = ScoreManager.LoadHighScore("race_track_X_time", float.MaxValue);
    if (highScoreTime == float.MaxValue)
    {
        textHighScoreTime.text = "Best Time: N/A";
    }
    else
    {
        textHighScoreTime.text = "Best Time: " + highScoreTime.ToString("F2") + "s"; // "F2"
formata para 2 casas decimais
    }

}
```

Outros Recursos se quiser colocar:

- Para resetar um Recorde Específico:

1. Coloque esse código

```
// Para dar ao jogador a opção de resetar um recorde específico

    ScoreManager.ResetHighScore("total_collected_points");
```

- Se você quiser manter uma lista dos X melhores scores em vez de apenas um único recorde.

1. Coloque esse código

```
float currentPlayerScore = 175;
ScoreManager.SaveScoreToList("top_adventure_scores", currentPlayerScore, 5,
true); // Salva na lista "top_adventure_scores", mantém top 5, maior é melhor
```

- Carregando e exibindo a lista de scores:

1. Coloque esse código

```
using System.Collections.Generic;
using System.Diagnostics;

List<float> bestScores = ScoreManager.LoadScoreList("top_adventure_scores", 5);
Debug.Log("Top Adventure Scores:");
for (int i = 0; i < bestScores.Count; i++)
{
    Debug.Log((i + 1) + ". " + bestScores[i]);
    // Aqui você atualizaria sua UI com esses scores

}
```

Segurança do PlayerPrefs: PlayerPrefs são armazenados de forma não segura e podem ser facilmente editados por usuários com conhecimento técnico. Para leaderboards online ou jogos competitivos onde a trapaça é uma preocupação, você precisará de soluções mais robustas (servidores, bancos de dados, ofuscação de dados).
Gerenciamento de Chaves: É importante que você escolha nomes de scoreType (as chaves) que sejam únicos e significativos. Evite usar nomes muito genéricos que possam colidir se você reutilizar o script em múltiplos contextos dentro do mesmo projeto sem querer. O método GenerateScoreKey ajuda a padronizar, adicionando um prefixo.
Resetar Todos os Recordes: A função ResetAllTrackedHighScores como está no exemplo é um *placeholder* para uma implementação segura. PlayerPrefs.DeleteAll() apaga *todos* os PlayerPrefs do seu jogo, não apenas os recordes. Para um reset seguro de *apenas* os recordes gerenciados por este sistema, você precisaria manter uma lista de todas as chaves de scoreType que seu jogo utiliza e chamar ResetHighScore para cada uma delas.