Esqueleto Boquinhas [Documentação]

Todos os jogos que fizerem parte do projeto Boquinhas devem utilizar o **EsqueletoBoquinhas** como base para as funcionalidades do jogo. Todos os jogos compartilham diversas similaridades, entre menus de opções, opções de dificuldade, estruturas de telas, janelas de opções etc.

Dessa forma o esqueleto irá auxiliar e agilizar o processo de desenvolvimento e manutenção de um projeto Boquinhas.

Além disso, o **EsqueletoBoquinhas** já vem acompanhado da última versão do **DEVKIT** da PlayTable e utiliza diversas de suas funções, por isso, ler o **manual do DEVKIT** e estar familiarizado com o mesmo antes de prosseguir com esse manual é **ALTAMENTE RECOMENDADO**.

Pontos de atenção:

1. *Jamais altere um script do **EsqueletoBoquinhas** ou **DEVKIT** dentro do seu projeto. Esses arquivos serão substituídos após instalar qualquer atualização do **DEVKIT** ou do próprio **EsqueletoBoquinhas**.*

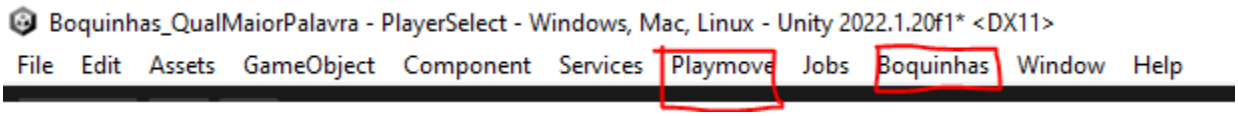
2. *Coloque os seus scripts customizados do jogo dentro da pasta **Assets/GameScripts/***
   *Dessa forma o seu código ficará organizado fora da estrutura do **DEVKIT** e **EsqueletoBoquinhas**.*

Se você já está familiarizado com o **DEVKIT** e leu os pontos de atenção acima, vamos começar a configurar o **EsqueletoBoquinhas** no seu projeto:

A primeira etapa é importar a versão mais recente do **EsqueletoBoquinhas.**

Antes de importar qualquer unitypackage em seu projeto, garanta que não exista qualquer erro no console da Unity.

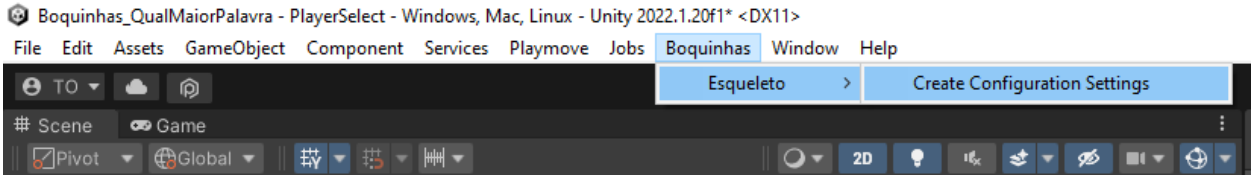
Se você realizou o import do package de maneira correta, duas novas opções (Playmove e Boquinhas) vão aparecer no menu superior do projeto na Unity. Isso indica que você importou o esqueleto com sucesso.



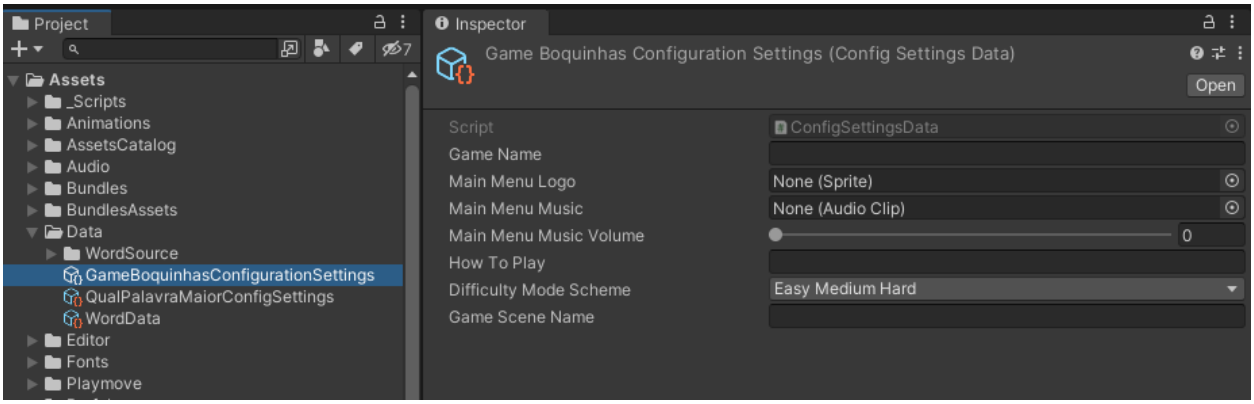
Após importar a versão mais recente do unitypackage do **EsqueletoBoquinhas** no seu projeto, você precisará criar um arquivo de configuração do Game Boquinhas, chamado de **Game Boquinhas Configuration Settings**.

*É importante ressaltar que esse arquivo não substitui o **Game Settings** do **DEVKIT** da PlayTable, que deve ser também configurado assim como é indicado no manual do **DEVKIT**. O arquivo **Game Boquinhas Configuration Settings** é um arquivo adicional, específico para os jogos do projeto Boquinhas.*

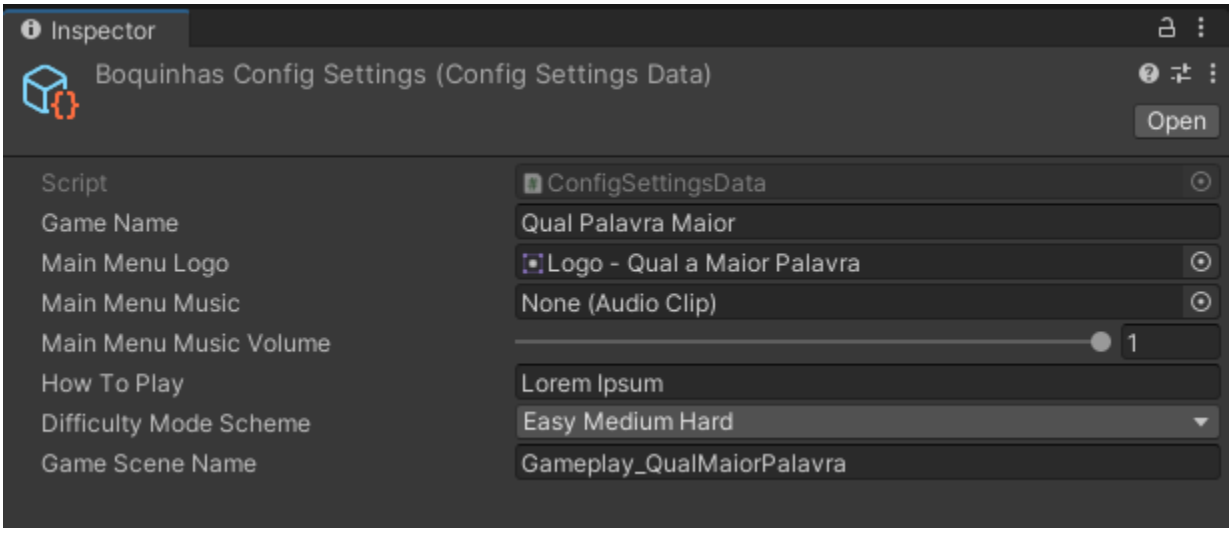
Para criar o arquivo, você deve ir ao menu superior do projeto em **Boquinhas>Esqueleto>Create Configuration Settings**:



Um arquivo com o nome **GameBoquinhasConfigurationSettings** será criado em *Assets\Data*

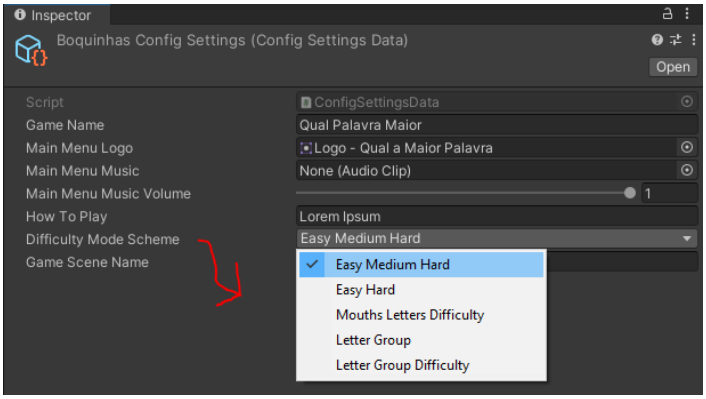


Game Boquinhas Configuration Settings



Nesse arquivo de configuração você deve colocar todas as informações referente ao jogo em que está trabalhando. Essas configurações alteram a estrutura do menu principal e as opções de dificuldade do jogo, além de alterar também o texto na janela “Como Jogar” e outras opções que estão detalhadas abaixo:

- Game Name: Indique aqui o nome do jogo;
- Main Menu Logo: Insira aqui a imagem que será utilizado como logo na tela de título/menu;
- Main Menu Music: Coloque aqui o arquivo de áudio que irá tocar em loop na tela de menu;
- Main Menu Music Volume: Configure aqui o volume da música acima;
- How To Play: Edite aqui o texto que aparece nas informações da janela “Como Jogar”;
- Difficulty Mode Scheme: Selecione a o modelo de “Opções de Dificuldade” que será usado no jogo. Esse modelo será utilizado no menu de seleção de dificuldades e poderá ser usado acessando o método **GameManager.Instance.GetDifficultyMode()**, que retorna uma das opções do enum **DifficultyModeScheme**.



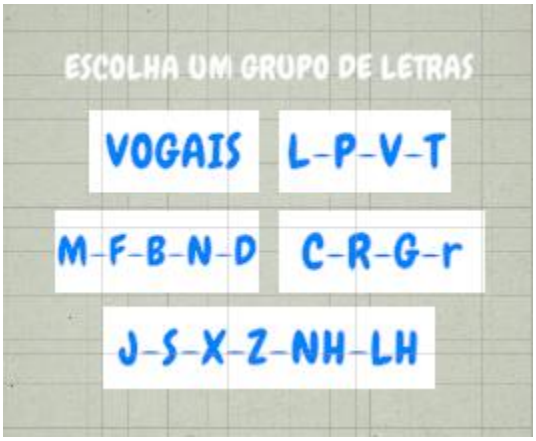
As opções estão detalhadas abaixo:

Easy Medium Hard:

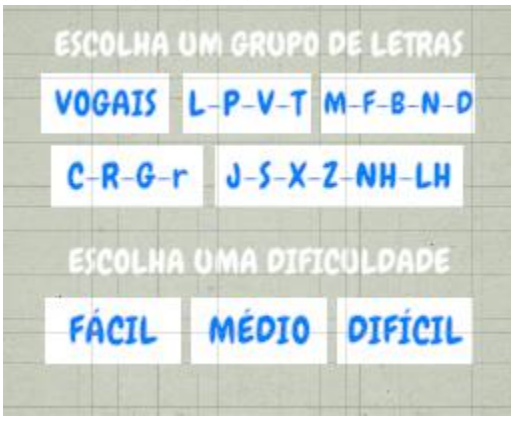
Easy Hard:

Mouths Letters Difficulty:

Letter Group:

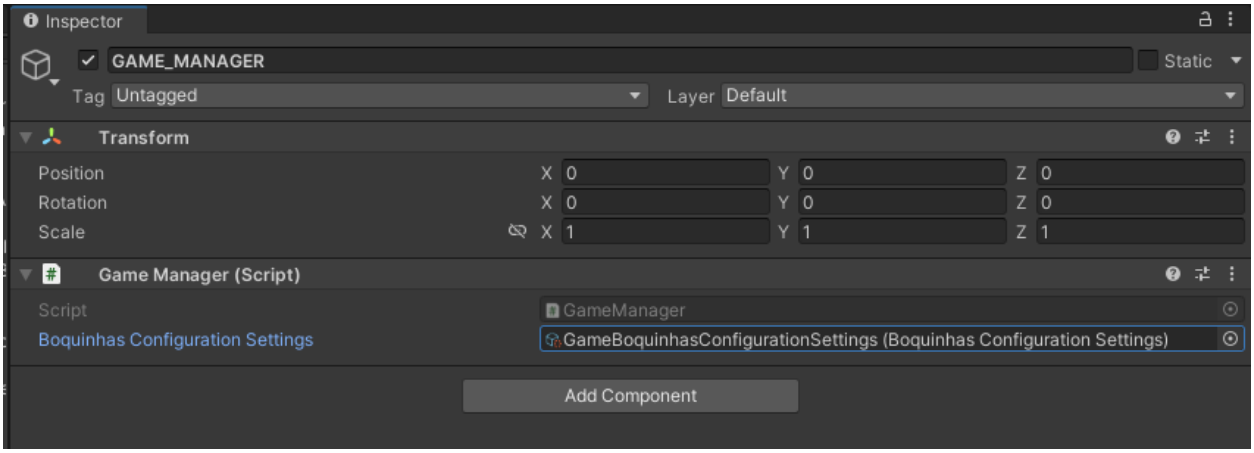


Letter Group Difficulty:



- Game Scene Name: Em “Game Scene Name” deve ser colocado o nome do arquivo scene onde fica o gameplay do jogo, a cena será chamada ao iniciar o jogo após o menu. **Não deve ser utilizada a cena de exemplo que acompanha o esqueleto, já que a mesma será atualizada em cada nova versão do esqueleto, sobrescrevendo qualquer modificação**;

Após finalizar as configurações no arquivo, vá até a cena **MainMenu** e no GameObject **GAME_MANAGER**, utilize o arquivo que você criou como referência para as configurações.



*Fique atento para não fazer suas configurações no arquivo “**ExampleBoquinhasConfigurationSettings**” já que o mesmo é apenas um exemplo de referência e sempre será substituído em qualquer atualização dos arquivos do **EsqueletoBoquinhas**.*

Turno/Rodada

A classe RoundManager gerencia os turnos do jogo, para saber de qual slot é a vez. Essa classe deve ser utilizada na lógica de controle de turno do jogo, dessa forma você pode utilizar as funcionalidades de Flip180 automático para os slots 2 e 3, além de garantir o envio facilitado de métricas de cada slot para o Class e até mesmo utilizar as informações para sua própria lógica de rodadas do seu jogo.

Existe um prefab chamado RoundManager que contém toda a lógica de rodadas de todos os jogos boquinhas. Para utilizar o prefab, basta arrastar ele para a cena. Esse prefab é responsável pelas animações de turno, rodada e do timer dos jogos.

No prefab existem eventos para auxiliar as chamadas customizadas que você precisar fazer no seu jogo.

- Game Start After Tutorial: Chamado assim que finaliza a exibição do tutorial;
- Gameplay Start: Chamado quando a lógica de gameplay do jogo é íniciada.
- Start Timer: O evento é chamado assim que o timer começa a rodar;
- Right Answer: Chamado após o usuário acertar uma resposta (Selo de correto exibido na tela);
- Wrong Answer: Chamado após o usuário errar uma resposta (Selo de errado exibido na tela);
- Continue After Right Answer: Chamado após o usuário clicar em continuar no selo de correto;
- Continue After Wrong Answer: Chamado após o usuário clicar em continuar no selo de errado;
- Try Again: O evento é chamado após o usuário clicar em “Tentar Novamente”;
- Game End: Chamado no encerramento da partida;

Para utilizar a classe RoundManager, você deve utilizar o **Namespace Core**. Adicionando **Using Boquinhas.Core;** no início de sua classe.

O **EsqueletoBoquinhas** utiliza diversos **Namespaces** diferentes para organizar suas classes e funcionalidades. Na sequência desse documento, você encontrará mais informações detalhadas sobre cada Namespace e suas funcionalidades.

Namespace Core [Boquinhas.Core]

Namespace responsável por gerenciar as classes principais do jogo.

- GameManager: Pode ser acessada sempre através da chamada **GameManager.Instance**;

[GameManager.Instace. GetDifficultyMode();]: Retorna o esquema de dificuldade utilizado no jogo;
[GameManager.Instace. GetCurrentDifficultyLevel();]: Retorna o nível de dificuldade selecionado, sendo 0 = Fácil, 1 = Médio, 2 = Difícil;
[GameManager.Instace. SetCurrentDifficultyLevel();]: Define o nível de dificuldade, sendo 0 = Fácil, 1 = Médio, 2 = Difícil;

Namespace Util [Boquinhas.Util]

Esse namespace contém algumas classes que podem ser úteis para acelerar o desenvolvimento de novos jogos.

- Shuffle: Pode ser chamado para embaralhar listas;
- ThreadSafeRandom: Um Random eficaz, performático e "thread safe", que deve ser usado na maioria dos casos;
- Timer: Um timer que pode ser utilizado para controlar diversas funções que envolvem contadores e marcadores de tempo;
- Flip 180: Essa classe pode ser utilizada para rotacionar GameObjects 180 graus na tela da playtable. Recurso que pode ser necessário para apresentar informações para os jogadores do Slot 2 e 3. Vale lembrar que objetos adicionados em **GameManager.Instance.AddGameObjectToFlip(*GAMEOBJECT*) ou no inspetor do objeto GamePlayManager [GameObjectsToFlip]** são rotacionados automaticamente, com apoio dessa funcionalidade, quando é a vez do slot 2 ou 3 no gerenciador de rodadas/turnos;

Namespace Interaction [Boquinhas.Interaction]

Nesse namespace você encontrará classes para agilizar algumas funcionalidades específicas de interações com o jogo através do touch da PlayTable.

- Hold
- Drag and Drop

Namespace Scriptable Objects [Boquinhas.ScriptableObjects]

Dentro desse namespace você irá encontrar classes para auxiliar no uso de ScriptableObjects para controlar os dados do jogo.

- ImportTXT
- ImportCSV

Audio

Para tocar sons nos jogos boquinhas você deve utilizar a classe AudioSystem.

Utilizando os métodos:

AudioSystem.Instance.PlayNarration();

AudioSystem.Instance.PlaySound();

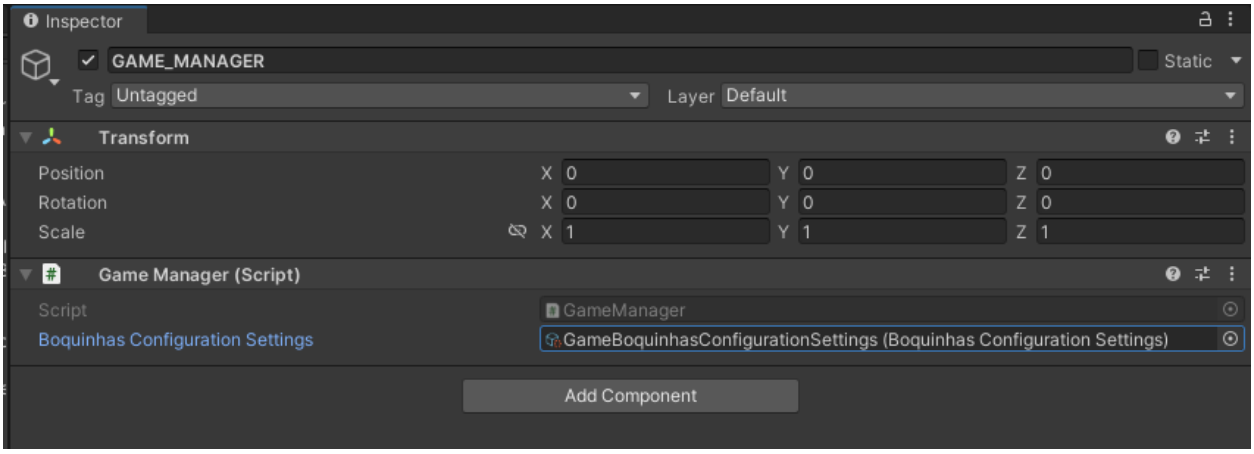
AudioSystem.Instance.PlayMusic();

Esses métodos irão obedecer às opções de configurações de som selecionadas no menu do jogo.

Caso queira tocar um som fora desse mixer controlado, utilize um AudioSource padrão da Unity.

Atualizando o EsqueletoBoquinhas

Sempre que atualizar o Package do **EsqueletoBoquinhas**, você precisa referenciar o **GameBoquinhasConfigurationSettings** na cena **MainMenu**, no GameObject **GAME_MANAGER**, utilizando o arquivo que você criou como referência para as configurações.



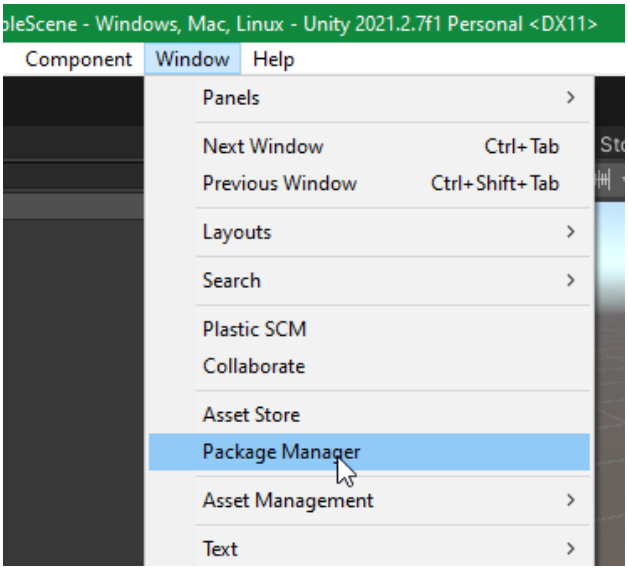
*Fique atento para não fazer suas configurações no arquivo "**ExampleBoquinhasConfigurationSettings**" já que o mesmo é apenas um exemplo de referência e sempre será substituído em qualquer atualização dos arquivos do **EsqueletoBoquinhas**.*

FAQ

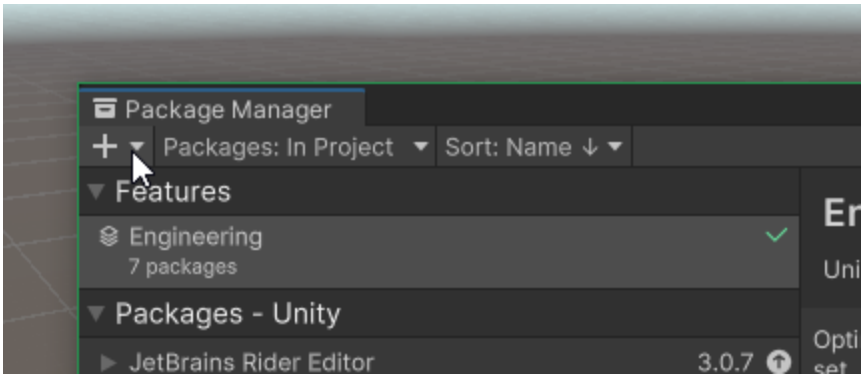
Meu projeto está apresentando problemas em relação a biblioteca newtonsoft-json ou em assuntos relacionados a json, o que fazer?

Siga os passos abaixo:

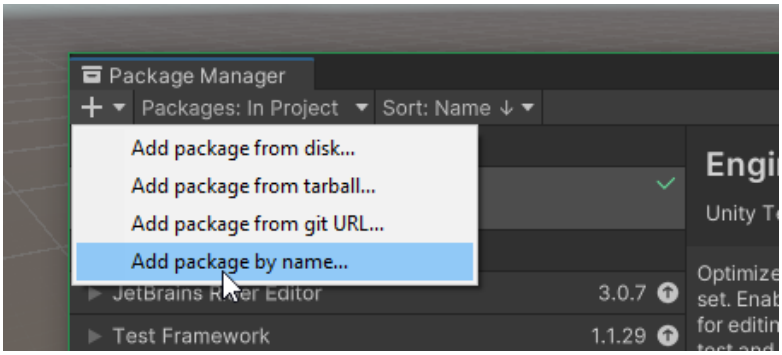
1. Abra o Package Manager



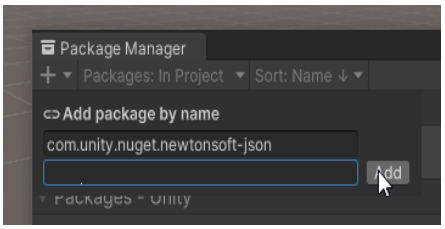
2. Clique em Add Package



3. Clique em "Add package by name"



4. Insira “com.unity.nuget.newtonsoft-json” no package name, sem as apas.



5. Clique em Add
6. Pronto!

Qual versão da Unity devo utilizar nos projetos do Boquinhas?

2022.3.0f1 LTS

Após atualizar a versão do esqueleto meu projeto está dando erro nos scripts EsqueletoVersion ou EsqueletoVersionEditor, o que fazer?

Esse erro acontece se você estiver atualizando um jogo com uma versão muito antiga do esqueleto, devido a um script que teve seu local alterado.

Para resolver, basta excluir o script abaixo do projeto: