Gerenciador de Cenas

Flávio Roberto Dias Silva

Encontro 10 - PDF 8
Desenvolvimento de Jogos Eletrônicos com Unity 3D.

Universidade Estadual do Oeste do Paraná

Cenas

Organização em cenas

Dividir o jogo em cenas é imprescindível para a organização do projeto e bom desenvolvimento do mesmo.

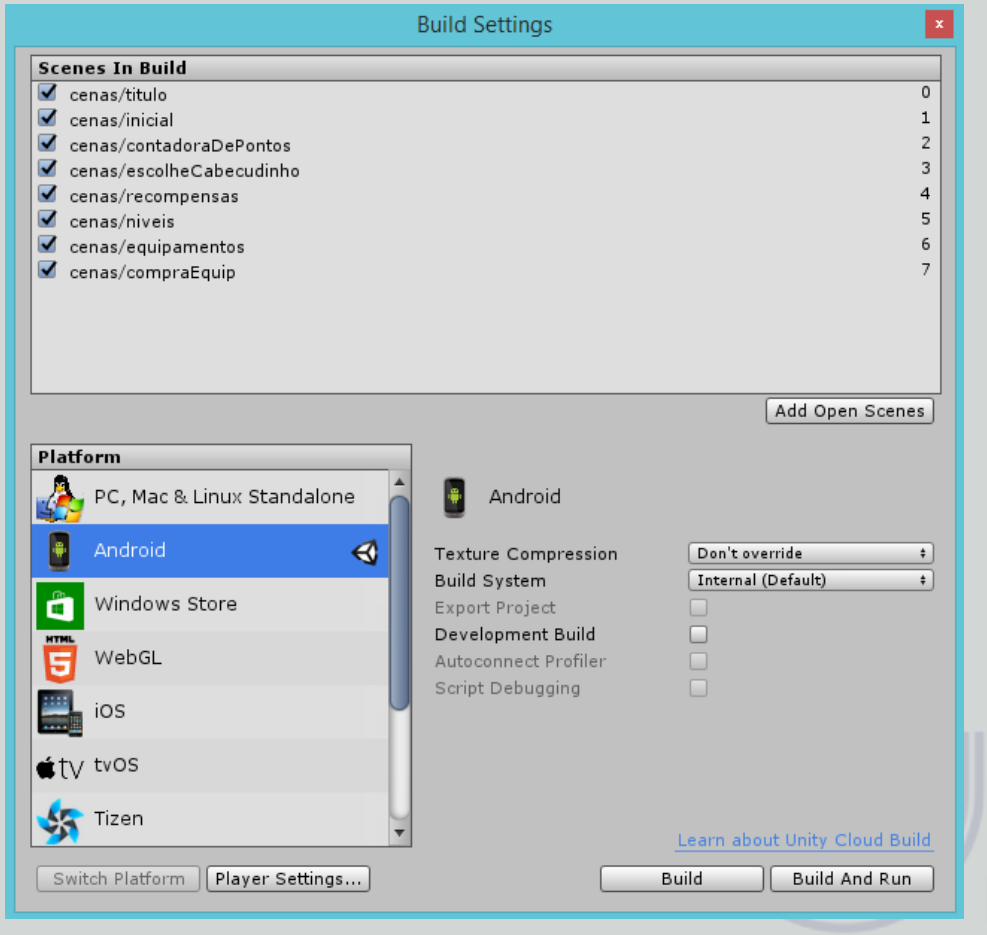
Até o momento neste curso montamos algumas cenas de teste, algumas com movimentação, algumas com testes de partículas outras com testes 2D. Nos concentraremos agora em alternar entre cenas e tratar os dados do nosso jogo na transição de cenas.

Application vs SceneManagement

Até as primeiras subversões do Unity 5 o carregamento de cenas era feito pela classe Application, sendo essa substituída pela classe SceneManagement.SceneManager. Esse encontro será focado na classe SceneManager com leves menções a classe Application.

Para correto funcionamento do carregamento de cenas, as cenas que serão carregadas em tempo de jogo devem ser incluídas na janela *Build Settings* que pode ser encontrada no menu *File* ou pressionando *CTRL + B*.

O icone de cena que consta na pasta Assets pode ser arrastado para a aba de Build Settings, uma outra maneira é clicar no botão “Add Open Scene”(adicionar a cena aberta) para adicionar a cena que está sendo editada ao BuildSettings.



Com as cenas carregáveis no BuildSettings podemos começar a prepara-las para o carregamento, para isso nosso script tem que utilizar a biblioteca `UnityEngine.SceneManagement`.

Essa biblioteca contém as funções dedicadas ao gerenciamento de cenas.

Entre as principais funções dessa biblioteca estão:

```
SceneManager.LoadScene(“titulo”);
Scene S = SceneManager.GetActiveScene();
```

LoadScene carrega a cena que tem como nome a string do parametro, essa função também pode ser chamada com um parametro do tipo **int**, esse se referindo ao indice que a cena tem no *BuildSettings*.

GetActiveScene retorna uma variável do tipo `Scene`, essa variável guarda valores relacionados com a cena carregada atualmente como seu nome, seu indice do Build, se ela está totalmente carregada etc..

Exercício:A partir de uma cena, construa um trigger que quando tocado carregue outra cena.

Tratamento de dados entre cenas

Quando carregamos uma nova cena os dados da anterior são totalmente perdidos se não tomarmos os devidos cuidados.

Ao programar um jogo precisamos o tempo todo transitar dados de uma cena para outra.

Por exemplo, no menu de seleção de personagens escolhemos nosso personagem. A informação com a escolha deve sair da cena de seleção e chegar até a cena de jogo.

Um outro exemplo, na tela de carregamento de jogo(que é uma cena) acessamos um arquivo de *saveGame* que consta num banco de dados, nesse arquivo está registrado em qual fase do jogo paramos e portanto devemos retornar ao jogo a partir dela. Devemos carregar a cena correspondente a tal fase levando em consideração todos os dados significativos: posição no mapa, pontos de vida, equipamento e etc.

Para levar dados de uma cena para a outra utilizamos a função

```
GameObject.DontDestroyOnLoad();
```

Como o próprio nome diz, essa função faz com que o **GameObject** não seja destruído no carregamento da cena.

A consequência imediata disso é que os scripts anexados a ele também não sejam destruídos e portanto podem transportar dados de uma cena para a outra.

Exercício: Nas duas cenas do primeiro exercício, insira vários trigger que levam para a outra cena. Construa um script que guarde qual trigger foi usado para transportar para a outra cena e posicione o personagem de acordo com esse trigger. Isso é trigger's diferentes devem carregar o personagem em posições diferentes.

[1] Barnes, D. J., Kölling, M.(2009), Programação Orientada a Objetos com Java. Uma introdução prática usando BLUEJ, 4ªed., Pearson Prentice Hall.

[Bttaiola,2015] Battaiola, A. L. (2000). Jogos por Computador ? Histórico, Relevância Tecnológica e Mercadológica, Tendências e Técnicas de Implementação In: XIX Jornada de Atualização em Informática. Curitiba: SBC, Julho/2000, v. 2. pp. 83 - 122

[2] Battaiola, A. L.; Elias, N. C.; Domingues, R.G. et al (2002). Desenvolvimento de um Software Educacional com Base em Conceitos de Jogos de Computador In: XIII Simpósio Brasileiro de Informática na Educação. São Leopoldo: SBC, 2002, pp. 282-290.

[3] Crua, E. W. G.; Bittencourte, J. R.(2005) Desenvolvimento de Jogos 3D: Concepção, Design e Programação. Anais da XXIV Jornada de Atualização em Informática do Congresso da Sociedade Brasileira de Computação, pp.1313-1356, São Leopoldo, Brasil, Julho de 2005. Deitel, H. M., Deitel, P. J.(2010), Java: Como programar, 8ªed., Pearson Prentice Hall. Rio de Janeiro IMPA.

[4] Freeman, E., Freeman, E.(2007), Use a Cabeça Padrões de Projetos, 2ªed., Rio de Janeiro Altabooks.

[5] Sintes, A.(2002), Aprenda Programação Orientada a Objetos em 21 dias, São Paulo Makron Books.

[6] Stellman, A.; Greene, J.(2011), Use a Cabeça! *C#*, Rio de Janeiro, AltaBooks.

[7] Unity Tecnologies (2016)(A). Unity 3D User Manual [online]. Disponivel em: [http://docs.unity3d.com/Manual/index.html][Acesso em 18/04/2016]

[8] Unity Tecnologies (2016)(B). Unity 3D Community Forum [online]. Disponivel em: [http://forum.unity3d.com/][Acesso em 18/04/2016]

[9] Unity Tecnologies (2016)(C). Unity 3D Online Tutorials [online]. Disponivel em: [https://unity3d.com/pt/learn/tutorials][Acesso em 18/04/2016]

[10] Unity Tecnologies (2016)(D). Unity 3D Community Wiki [online]. Disponivel em:

[http://wiki.unity3d.com/index.php/Main_Page][Acesso em 18/04/2016]