Levantamentos sobre Unity 2D

Prof. Dr. Flávio Roberto Dias Silva

Unity 2D
Introdução a programação de jogos com Unity

Universidade Estadual do Oeste do Paraná

Introdução ao Unity 2D

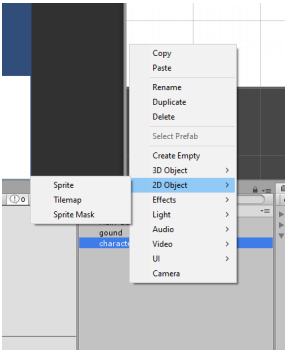
Muitos dos elementos já estudados para Unity 3D estão presentes em Unity 2D.

Para criar jogos em 2D você conta com `Rigidbody2d`, `Collider2d`, `OnCollisionEnter2D` entre outras funções com versões 2D.

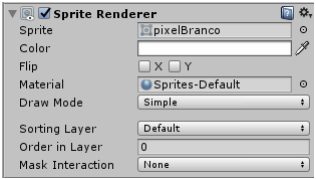
A maioria das funções tradicionais continua funcionando da mesma maneira no 2D, inclusive a programação do jogo do 2D é feita também no ambiente 3D tendo a particularidade do eixo z estar fixo.

O elemento primordial para a programação de jogos 2D na Unity é o Sprite.

Uma das formas de adicionar um sprite na cena clicando com o botão direito na aba hierarchy e usando a opção adicionar Sprite.



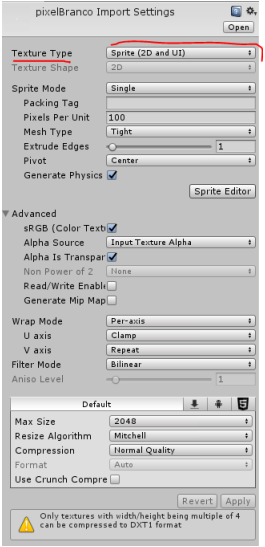
Com um gameObject adicionado por meio da opção Sprite temos anexado a ele um componente chamado **SpriteRendere**. Neste componente temos uma propriedade Sprite. O componente é responsável por desenhar em cena o sprite que está nesse parametro.



Note que esse componente entre outras propriedades, tem a propriedade **flip**. Em jogos 2D essa propriedade é bastante usada, ela tem o papel de espelhar o Sprite em relação aos eixos X e Y, praticamente ela muda a direção que o sprite está virado.

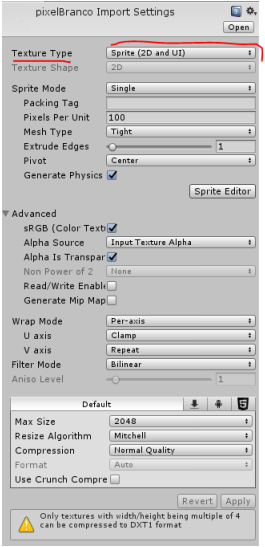
Quando adicionamos uma imagem na Unity, em geral, esta não vem configurada para ser usada como Sprite. Como a unity inicialmente foi projetada para jogos 3D todas as imagens inseridas vem configuradas para serem usadas como texturas.

Para modificar a configuração da imagem devemos seleciona-la na aba project. Com a imagem selecionada na aba project temos acesso ao menu de configurações da imagem. Inicialmente devemos modificar a opção **texture type** para **Sprite**.



Com essa configuração podemos usar nossa imagem como Sprite na Unity.

Outra configuração interessante de ser usada, principalmente em jogos de pixel art, é o **Filter Mode**, na maioria dos jogos 2D é interessante colocar a opção **Point (no filter)** para evitar que a imagem fique com um aspecto borrado.



Tilemaps

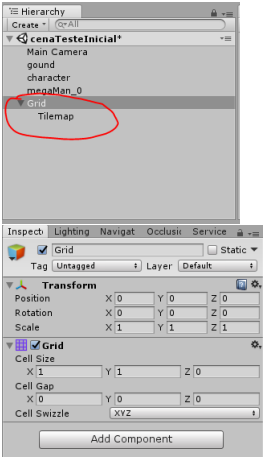
Apesar de possivel fazer o cenário todo no jogo 2D juntando sprites simples com colliders aplicados a unity, a partir da versão 2017, conta com suporte a grid de tilemaps.

Com o grid de tilemaps é possivel fazer um cenário praticamente pintando blocos de tamanhos pré-definidos na tela.

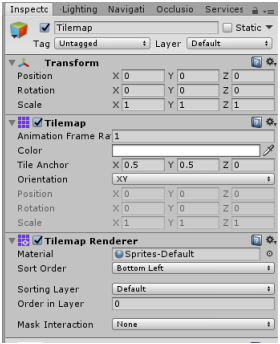
No que segue faremos um breve tutorial de como montar um cenário com tilemaps na Unity.

Após clicado na opção tilemap, na hierarchy, será criado um objeto chamado grid que tem um objeto filho na hierarchia chamado tilemap.

Ao lado damos destaque ao inspector do objeto grid que tem um componente com o mesmo nome. Nesse componente temos a opção de escolher o tamanho do retangulo do grid. No caso, este está configurado para ter 1x1 em unidades de medida da Unity. Logo vamos falar da imagem que utilizaremos com o tilemap e esta deve ter uma configuração para coincidir com essa unidade de medida.



Já o inspector do objeto **tilemap** tem dois componentes anexados a ele, o de mesmo nome e o de nome **TilemapRenderer**



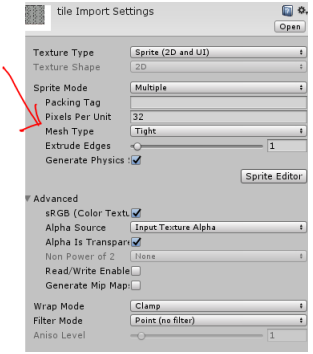
No componente tilemap, damos destaque a propriedade **color**, que pode influenciar a cor do tile utilizado. entre muitas utilidades, a propriedade color fica bem interessante quando temos tiles monocromáticos e queremos faze-los aparecer em diferentes cores no jogo.

No componente tilemaprenderer, a propriedade de mais destaque é a **Order in Layer**, que diz pra gente em qual camada de renderização esse tilemap aparecerá. Essa propriedade está presente em qualquer Sprite, então ela pode ser usada para dizer se essa camada aparece a frente do personagem ou atrás, também podemos criar varios tilemaps num grid e organiza-los por layers.

O próximo passo para fazer nosso cenário em tilemap é a imagem que será usada. Importando para a unity a imagem que será usada no tilemap, devemos configura-la como Sprite e além disso, precisamos configurar a opção **Pixels per Unit.**

A opção pixels per unit trabalha em conjunto com a propriedade Cell size do componente Grid. No componente grid, dizemos quantas unidades da Unity cada retangulo do tilemap ocupa, e na opção pixels per unit, dizemos quantos pixels da nossa imagem cabem em cada unidade da unity.

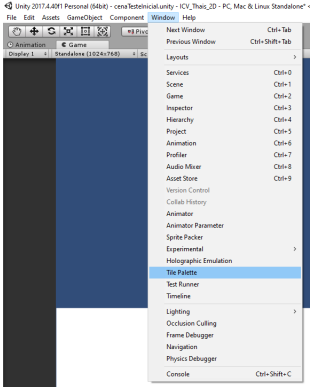
Essa opção deve ser preenchida com a quantidade de pixel de cada retangulo do seu tilemap.



Com a imagem do tilemap de prontidão precisamos configurar uma paleta para o tilemap.

Vamos agora usar o menu suspenso da Unity, na opção window, clicar em Tile Pallet.

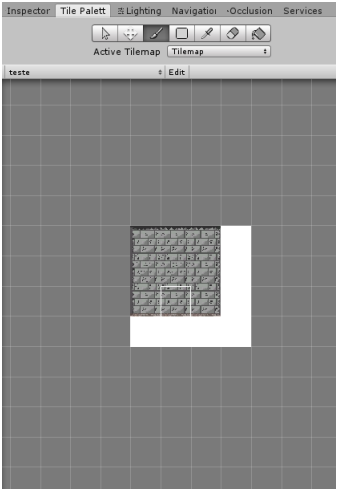
Observação: Nas versões mais novas da Unity, os componentes 2D não vem pré instalados. Estes podem ser procurados e baixados a partir do package manager constante nas versões mais novas.



Clicado na opção tilepalette, um painel se abrirá e este pode ser anexado a um grupo de paineis da Unity.

Com o painel aberto podemos arrastar imagens para ele, no caso, devemos arrastar as imagens que farão parte do nosso tilemap. Ao arrastar imagens para o painel ele pedirá para salvar esse conteudo em algum lugar da pasta project, faça uma pasta para organizar seus tilemaps.

Com as imagens adicionadas a paleta, podemos agora usar as ferramentas para pintar o cenário



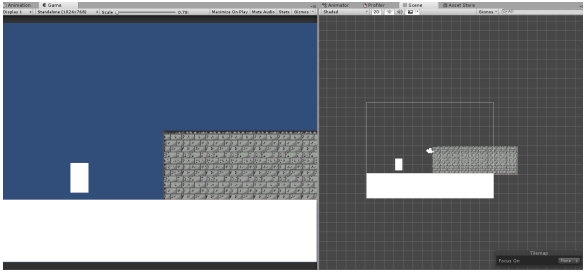
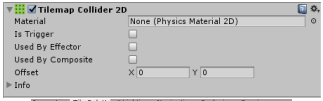
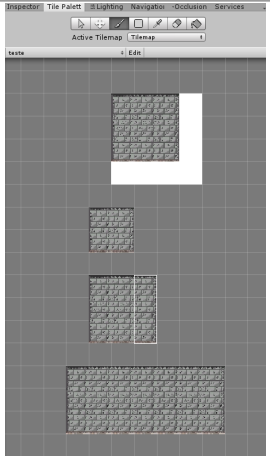


Figura: O inicio de um desenho de cenário com tilemap.

É interessante observar que não precisamos redesenhar varias vezes elementos que se repetem no cenário, como plataformas,montanhas ou outros desenhos especificos que possamos fazer com nosso tilemap, podemos na paleta usar os tiles que ja existem para fazer esses desenhos repetitivos nela usando a opção edit.





Vale mencionar que os Tilemaps tem um componente de colisão especifico para ele, este é **TilemapCollider2D**, ele tenta desenhar a colisão baseado nos pixels que não são transparentes na sua imagem, ou seja, ele tenta se adaptar às transparências.

Isso nos da total liberdade para fazer partes do cenário que tem ou não colisão. Usar tiles para fazer backgrounds, paredes falsas e etc..

Animação em Unity para 2D

Um elemento indispensável para qualquer jogo é a animação. A unity conta com poderosas ferramentas de animação com suporte a diferentes tipos de tecnicas que podem ser aplicadas a jogos 2D.

Trataremos aqui de dois tipos principais: a animação por fatias(camadas) e a animação por tabelas de sprites.

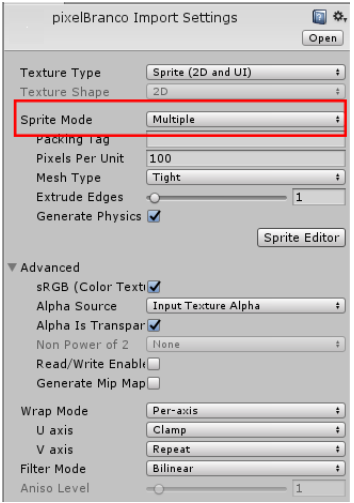
Animação por tabelas de sprite

Para exemplificar usaremos uma tabela de sprites pronta como pode ser visualizada ao lado

É importante observar que na mesma imagem temos um conjunto de sprites que devem aparecer separadamente no jogo, como é uma animação, deve aparecer numa sequencia que de a impressão dessa animação.

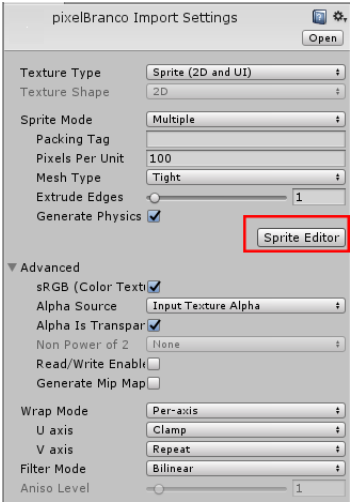
Um primeiro passo é configurar a imagem como um sprite multiplo. Isso pode ser feito no menu de configurações de imagem.

Devemos mudar a opção **Sprite mode** para **multiply**.



A ultima opção diz ao unity que o Sprite é multiplo mas ainda não diz quantos são os sprites ou como o Unity deve separa-los.

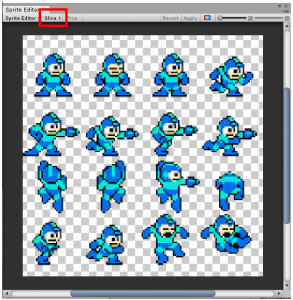
Para dizer ao Unity como separar os Sprites devemos clicar na opção Sprite Editor.

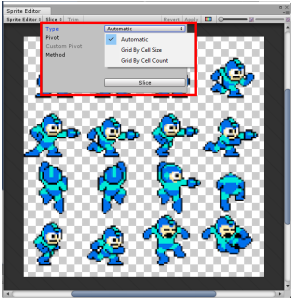


Com Sprite Editor aberto devemos clicar em Slice(fatiar).

Isso abrirá as opções de divisão onde podemos selecionar os tipos Automatico, pelo numero de celulas ou pelo tamanho das celulas.

Clicando no botão Slice teremos divido nosso Sprite.

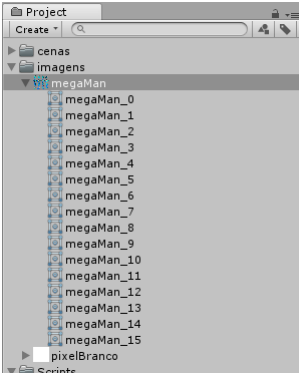




Após dividido nosso Sprite, as partes dividas aparecem na pasta project como hierarquia da imagem original.

Podemos agora fazer o arquivo de animação escolhendo os quadros que representam a nossa animação e arrastando para a cena. Fazendo isso o Unity pedirá para salvar o arquivo de animação.

Após salvar o arquivo de animação, o objeto será criado na aba hirarchy, este terá anexado a ele um componente chamado **Animator**, este ultimo é o componente responsável por gerenciar as animações e falaremos mais dele adiante.



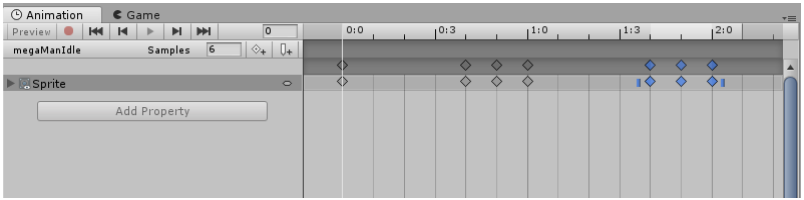


Figura: Painel Animation com os quadros chaves da animação

Voltaremos nossas atenções agora para o painel **Animation** (não confundir com **Animator**), nesse painel podemos fazer algumas alterações na animação que acabamos de criar, como aumentar e diminuir o tempo da animação ou mesmo da troca de quadros.

Animação por Sprite fatiado

Anteriormente vimos como fazer animações com uma tabela de Sprites, que em resumo, consiste em colocar uma sequencia de imagens sendo substituidas após um certo tempo.

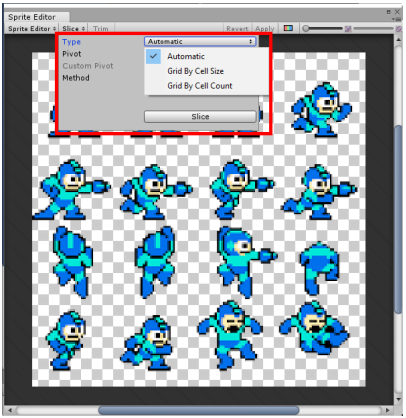
A segunda técnica de animação que veremos agora consiste em ter um personagem com elementos separados na imagem, como por exemplo, braços, pernas, tronco e cabeça como imagens independentese com o sprite cumprindo esses pré requisitos animar cada parte separadamente nas faixas de animação do painel animator.

Ao lado temos o exemplo de um personagem fatiado. As partes são colocadas na imagem de maneira separadas para terem animações independentes.

Inicialmente vamos configura-la para ser usada com o seu propósito. Para isso vamos novamente no inspector da imagem, configura-la como Sprite multiplo e entrar no Sprite editor.

Nas opções de Slice podemos agora usar o automatico e clicar em Slice. Caso o slice automatico não tenha sido satisfatório ainda podemos fazer ajustes manualmente.

Após aplicado o slice, da mesma forma como mencionado anteriormente, as partes aparecem na hierarchia da imagem na aba project.



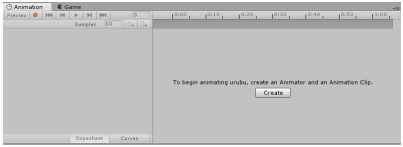
Nesse momento vamos montar nosso personagem na aba scene. É interessante criar um objeto vazio para ser a raiz do personagem e colocar cada parte de acordo com sua prioridade no parentesco da hierarchia. No caso da nossa imagem, o corpo terá como filhos o pescoço as pernas e o rabo. o pescoço será pai da cabeça, a cabeça pai do chapéu e assim por diante.

Os posicionamentos de parentescos são importantes para que haja influencia da peça pai na animação da pela filho.

Outra informação importante que podemos configurar é a camada de renderização de cada parte. Em alguns momentos, podemos querer que uma imagem apareça acima ou abaixo de uma outra, e podemos configurar isso modificando a unidade em order in layer nos componentes sprites das imagens envolvidas.

Chegamos no momento de animar o personagem. Para isso vamos na aba **Animation** (se ela não estiver aparecendo na interface do unity a procure no menu Window)

Com o personagem selecionado na aba hierarchy, na aba Animation aparece a opção de criar uma animação para ele. Vamos clicar nessa opção

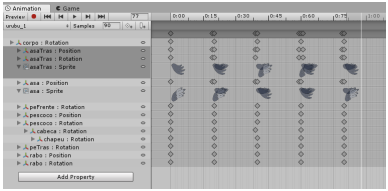


Após clicar no botão criar da aba animation é pedido para salvar o arquivo de animação que estamos prestes a editar.

Observe que após criar a animação, no nosso personagem, é anexado um componente chamado Animator.

Vamos agora clicar no botão de gravar que está na aba Animation. Após clicado, alguns elementos da janela aparecem em vermelho.

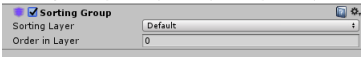
Podemos agora montar nossas animações, posicionando a linha branca do painel animation no quadro chave que queremos editar e fazendo alterações no nosso personagem, geralmente, essas alterações são em termos de rotação, posição e escala, mas podem abranger outros elementos.



Outro elemento comum de ser envolvido na animação é substituir o sprite de alguma parte. No caso da nossa imagem exemplo, podemos substiruir o sprite da asa para dar a impressão do bater de asas durante um voo.

Uma última observação interessante nesse momento é sobre as camadas(layers) de animação. Como no personagem fatiado precisamos configurar muitas layers isso em algum momento pode entrar em conflito com outros elementos do jogo, como outros personagens ou inimigos ou mesmo o cenário.

Para diminuir possíveis problemas com as layers do personagem fatiado é interessante anexar a ele um componente chamado shorting group. Com esse componente podemos reidentificar o personagem todo como pertencente a uma camada, e as camadas das fatias ficam com a configuração de subcamadas.



Exercicio

1. Faça uma cena de jogo 2d. Insira movimentação, pulo e cenário.

No próximo encontro falaremos de manipulação de animação, mas se quiser se adiantar, tente colocar animação na sua cena 2D.

[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 (2022)(A). Unity 3D User Manual [online]. Disponivel em: [http://docs.unity3d.com/Manual/index.html][Acesso em 18/04/2022]

[8] Unity Tecnologies (2022)(B). Unity 3D Community Forum [online]. Disponivel em: [http://forum.unity3d.com/][Acesso em 18/04/2022]

[9] Unity Tecnologies (2022)(C). Unity 3D Online Tutorials [online]. Disponivel em: [https://unity3d.com/pt/learn/tutorials][Acesso em 18/04/2022]

[10] Unity Tecnologies (2022)(D). Unity 3D Community Wiki [online]. Disponivel em: [http://wiki.unity3d.com/index.php/Main_Page][Acesso em 18/04/2022]