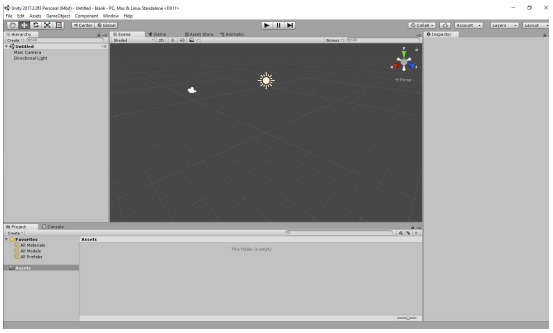
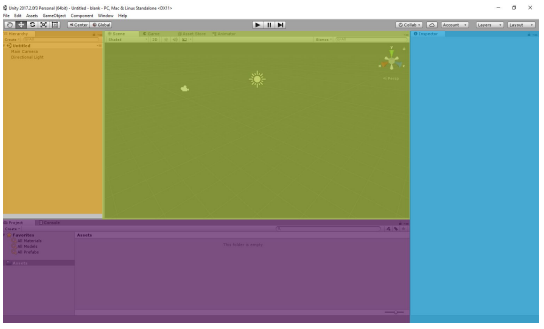
Conceitos básicos de Unity 3D

1. Entendendo a interface do editor

Ao abrirmos um projeto na Unity, podemos observar a interface gráfica do editor. Para facilitar a compreensão, a interface pode ser dividida em algumas áreas, ilustradas por diferentes cores na imagem a seguir. Uma descrição completa da interface do editor está disponível em [0].





- Na parte amarela, temos a **Hierarchy**, que é uma lista dos objetos inseridos na cena atual.
- Na parte azul, chamada de **Inspector**, podemos ver as propriedades e os componentes de um objeto. Aqui faremos a maior parte das configurações dos objetos, como o controle de colisão, entre outras coisas. Para que possamos visualizar as informações de um objeto, primeiro precisamos clicar sobre ele na listagem da Hierarchy.
- Na parte verde, temos algumas opções, organizadas em tabs, que permitem alternar entre os seguintes modos: **Scene** (montagem da cena), **Game** (renderização da cena), **Asset Store** (loja de acessórios da Unity) e **Animator** (controle de animações).
- Na parte roxa, temos duas possibilidades importantes, também organizadas em tabs. Na primeira delas, **Project**, temos a organização do nosso projeto em pastas. A segunda tab, **Console**, também tem grande importância pois exibirá prints, mensagens de erro e de compilação do projeto.

Como podemos observar, o projeto criado na Unity virá com dois objetos, listados na Hierarchy: Main Camera e Directional Light. Aqui, é importante saber que na Unity um objeto da cena é chamado de GameObject [0].

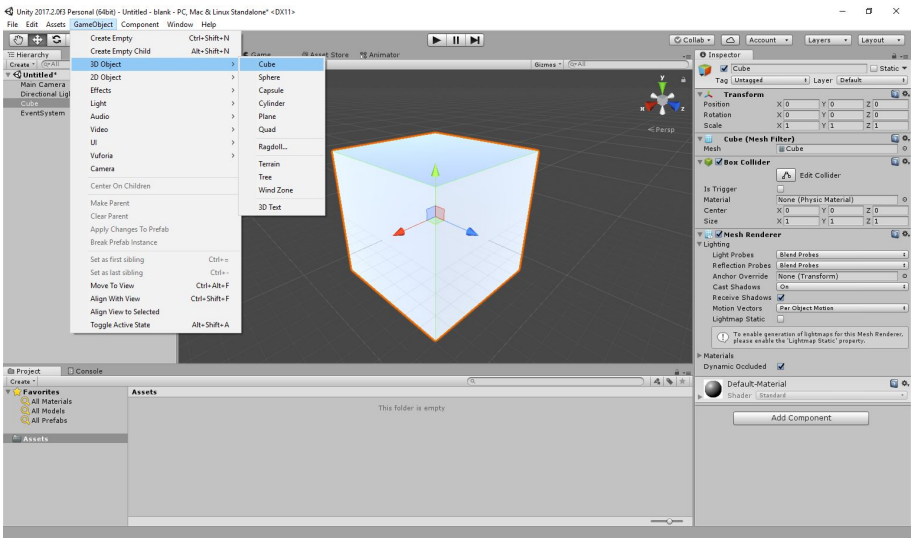
2. Criando outros objetos

A inserção de novos objetos na cena ou no projeto pode acontecer de duas formas:

1. Criando um novo GameObject pelo editor da Unity, como um cubo;
2. Importando um objeto já existente em um arquivo, como um avatar.

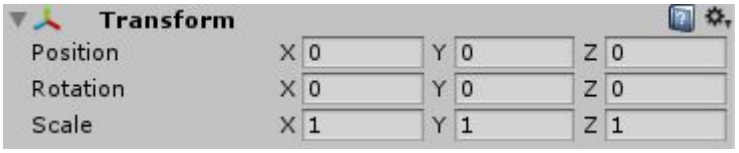
Para criar objetos do caso 1, pode-se clicar com o botão direito em Hierarchy e selecionar a opção “Create” e, então, escolher o tipo de objeto que deverá ser criado. Também é possível executar esta operação a partir do menu “GameObject”, na barra de ferramentas. Uma melhor compreensão sobre a criação de GameObject nativos está disponível em [2].

No caso 2, podemos usar o botão direito na área Project, mas selecionar a opção “Import new asset” ou, ainda, clicar em “Assets” na barra de ferramentas e escolher a opção “Import new asset”. É válido destacar que, no segundo caso, os objetos importados serão carregados no projeto - e não na cena. Para colocá-los à cena, basta arrastá-los até a lista de Hierarchy ou a Scene. Um tutorial completo sobre a importação de objetos está disponível em [3].



3. Translação, escala e rotação

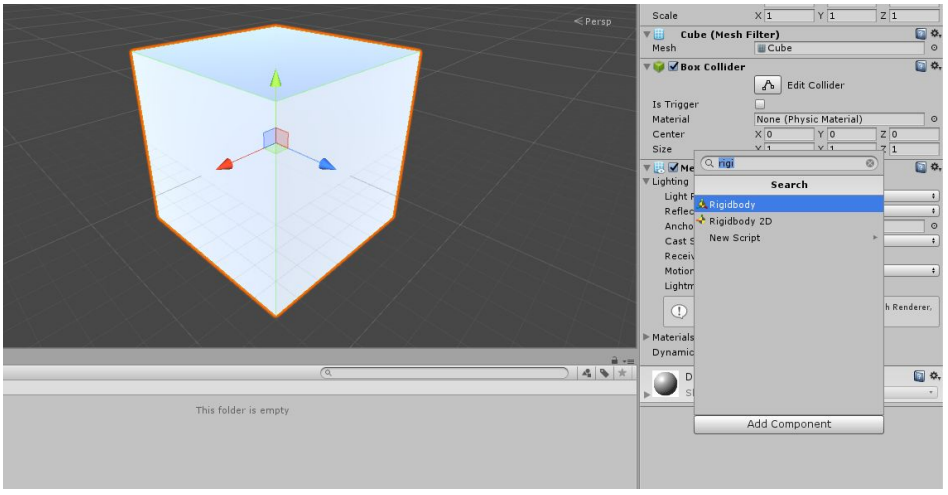
Conforme descrito em [4], os GameObjects possuem por padrão o componente Transform, que possibilita a aplicação das transformações de translação, escala e rotação. O componente Transform será exibido no Inspector do objeto, permitindo a alteração direta dos valores.



No modo Scene, também é possível alterar a posição, rotação e escala do objeto a partir dos seguintes controles, disponíveis no canto superior esquerdo da interface.

4. Inserindo componentes

Um componente atribui um comportamento ou função a um GameObject [5]. Atualmente, há diversos tipos de componentes disponíveis na Unity. No contexto da disciplina, os mais importantes são os componentes de física (RigidBody) [6], áreas de colisão (BoxCollider, SphereCollider ou MeshCollider) [7,8,9], controlador de animação (AnimatorController) [10], controlador de personagem (CharacterController) [11] e, finalmente, Script [12].



Para adicionar um componente a um objeto, basta selecionar o objeto na Hierarchy e, ao lado direito, clicar no botão “Add Component”, conforme ilustra a figura. Outras instruções acerca do uso dos componentes podem ser encontrados em [13].

A Unity fornece uma completa documentação sobre cada um dos componentes.

5. Colisão

A colisão é um aspecto muito importante em jogos e aplicações de Realidade Virtual e Realidade Aumentada. Sua aplicação permite simular o choque entre dois objetos (como um carro batendo num poste, por exemplo) ou simplesmente possibilitar que identifiquemos que o personagem está em uma área específica do jogo.

Em ambos os casos, o uso de um componente do tipo Collider será indispensável. Neste componente, podemos marcar ou desmarcar a opção “is trigger”. Quando estiver desativada, a colisão irá atuar como um objeto sólido (com o auxílio do componente RigidBody). Quando “Is Trigger” estiver ativado, a colisão permitirá que o objeto seja permeável.

Eventos decorrentes de uma colisão, como fazer o jogador ganhar ou perder pontos, devem ser implementados em script. A Unity têm funções que permite identificar:: i) se um objeto A entrou em colisão com um objeto B; ii) se um objeto A está em colisão com um objeto B; e iii) se um objeto A saiu da colisão com um objeto B.

Apesar de funcionarem de modo semelhante, as funções são diferentes para as colisões do tipo Trigger. As diferenças conceituais e exemplos são apresentados em [14] e [15].

Referências

[0] https://docs.unity3d.com/Manual/UsingTheEditor.html

[1] https://docs.unity3d.com/Manual/GameObjects.html

[2] https://docs.unity3d.com/Manual/class-GameObject.html

[3] https://docs.unity3d.com/Manual/ImportingAssets.html

[4] https://docs.unity3d.com/Manual/class-Transform.html

[5] https://docs.unity3d.com/Manual/Components.html

[6] https://docs.unity3d.com/Manual/class-Rigidbody.html

[7] https://docs.unity3d.com/Manual/class-BoxCollider.html

[8] https://docs.unity3d.com/Manual/class-SphereCollider.html

[9] https://docs.unity3d.com/Manual/class-MeshCollider.html

[10] https://docs.unity3d.com/Manual/class-AnimatorController.html

[11] https://docs.unity3d.com/Manual/class-CharacterController.html

[12] https://docs.unity3d.com/Manual/ScriptingSection.html

[13] https://docs.unity3d.com/Manual/UsingComponents.html

[14] https://docs.unity3d.com/ScriptReference/Collision.html

[15] https://docs.unity3d.com/ScriptReference/Collider.html