

Desenvolvimento com Motores de Jogos I

Aula 01 - Introdução à Interface do Motor de Jogos

Apresentação

Olá, desenvolvedores! Sejam bem-vindos à primeira aula da nossa disciplina de Desenvolvimento com Motores de Jogos II! Essa aula marca o começo de uma jornada muito interessante em um novo universo do mundo dos games: o de desenvolvedor! Já temos alguma experiência com jogos na perspectiva dos jogadores, mas você já parou para pensar como seria desenvolver seus próprios jogos? E que outras pessoas também poderiam utilizá-lo e se divertirem jogando? Começaremos a conhecer essas técnicas na aula de hoje! Estão prontos?

Na disciplina de Motores I, desenvolveremos apenas jogos em 2D! Já na disciplina de Motores II, teremos os conceitos estendidos ao 3D, incluindo algumas técnicas que são exclusivas desse formato. Como os conceitos serão, em grande parte, reaproveitados e expandidos, fique ligado no que veremos aqui!

Conhecemos as diversas vantagens de usarmos um motor em relação ao desenvolvimento de jogos utilizando bibliotecas e uma linguagem de programação qualquer. Veremos também a interface de um motor de jogos e os principais componentes que lá estão para que possamos ter o desenvolvimento de nosso jogo facilitado. Ainda nesta aula, aprenderemos a criar novos projetos e a utilizar a loja do próprio motor para conseguirmos alguns exemplos. Ao fim da aula, já teremos o nosso primeiro jogo funcionando! Dá para acreditar?! Vamos lá!

Use o código de Konami para descobrir o segredo do controle.

UP, UP, DOWN, DOWN, LEFT, RIGHT, LEFT, RIGHT, B, A, START!

Objetivos

Ao final desta aula, você deverá ser capaz de:

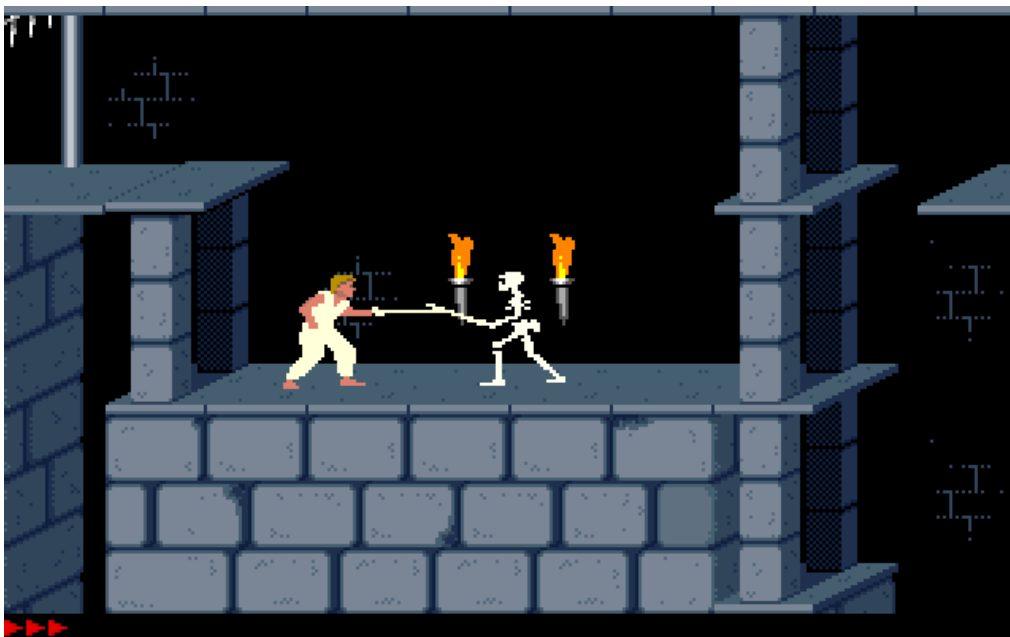
- Entender o que é e como funciona o motor Unity;
- Entender os conceitos básicos do desenvolvimento de jogos com Unity;
- Conhecer a Interface do Unity;
- Criar novos projetos e conhecer a Asset Store;
- Criar um Hello World no Unity.

A Utilização de Motores de Jogos

Há pouco tempo, em uma galáxia não tão distante, ocorreu o surgimento dos primeiros jogos digitais. Esses jogos, que conhecemos com mais detalhes na disciplina de Introdução aos Jogos Digitais, eram simples, feitos com gráficos de baixa qualidade e recursos limitadíssimos. Você imagina como eram desenvolvidos esses jogos? Não? Pois eu vou lhe contar: os jogos eram desenvolvidos em [Assembly](#)! :-)

Agora, pare um pouco e pense em um jogo (articulação complexa de elementos gráficos, animações, sons, cenários, etc.) desenvolvido em linguagem de baixo nível! Haja trabalho! E muito! Sério mesmo! Quer ter uma ideia de quanto? Dá uma olhada no código fonte oficial do primeiro Prince of Persia (1989), lançado para a Apple II:

<https://github.com/jmechner/Prince-of-Persia-Apple-II/tree/master/01%20POP%20Source/Source>



Prince of Persia (1989).

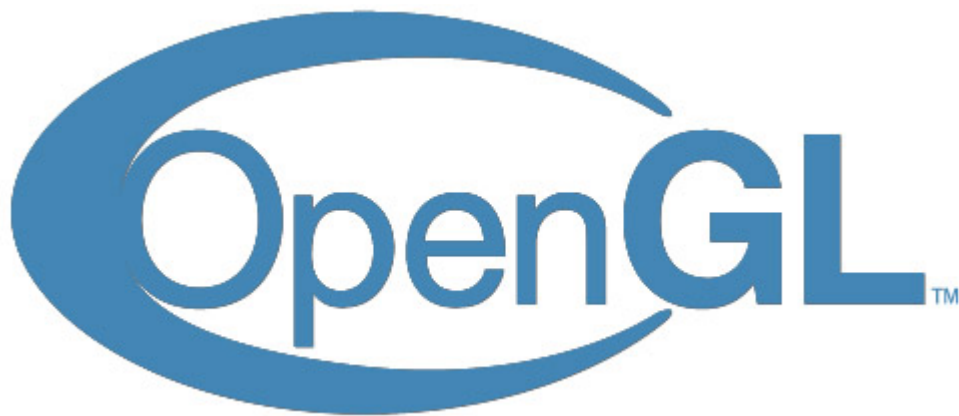
Fonte: <http://poucopixel.com/pc/prince-of-persia/>. Acesso em 24 de jan de 2017.

Não entendeu nada? Tudo bem... Eu também não. Mas fica aí o valor histórico da informação. E também o alívio de não precisar mais trabalhar desse jeito para desenvolver os jogos atualmente! Ufa :-)

Após essa fase complicada, com a evolução dos computadores e dos consoles de jogos, começaram a surgir diversas maneiras de facilitar o desenvolvimento de jogos através de novas técnicas e aplicações da área. Bibliotecas gráficas, por exemplo, surgiram a fim de facilitar a integração de software e hardware e definir padrões para que fosse possível desenvolver em 2D e 3D com mais facilidade, como a IRIS GL, que se tornou o [OpenGL](#), uma [API](#) gráfica muito bem-vista e aceita por toda a comunidade de computação gráfica.

API é um conjunto de rotinas e padrões de programação para acesso a um aplicativo de software ou plataforma baseado na Web. A sigla API refere-se ao termo em inglês "Application Programming Interface" que significa em tradução para o português "Interface de Programação de Aplicativos".

Fonte: Canaltech. **O que é API?**. Disponível em: <https://canaltech.com.br/o-que-e/software/o-que-e-api/>. Acesso em 24 de jan de 2017.



Fonte: <https://commons.wikimedia.org/wiki/File:Opengl-logo.svg>.
Acesso: 24 jan. 2017

Em 1995, no entanto, a Microsoft lançou o Direct3D, o qual surgiu objetivando ser um competidor ao padrão OpenGL. Depois alguns esforços insatisfatórios para unificar os projetos, as duas APIs seguiram caminhos diferentes e até hoje são assim. Temos o DirectX e o OpenGL trazendo funcionalidades diferenciadas e atraindo a atenção dos desenvolvedores para ambos os lados.

Explorando Recursos de APIs

Apesar da facilidade que ambas as APIs trouxeram aos desenvolvedores gráficos, ainda mostrava ser proveitosa a existência de novas camadas acima dessas APIs, contribuindo para que os desenvolvedores pudessem atingir seus objetivos de maneira mais simples e rápida. Nesse novo nível de abstração, estão as bibliotecas que encapsulam o OpenGL ou o DirectX e permitem ao programador realizar as tarefas de um modo ainda mais simples. A fim de dar um exemplo mais prático, vejamos um trecho de código para desenhar um cubo em OpenGL, como na

Listagem 01:

```
1 // FRENTE
2 glBegin(GL_POLYGON);
3     glVertex3f( 0.5, -0.5, -0.5 );
4     glVertex3f( 0.5, 0.5, -0.5 );
5     glVertex3f( -0.5, 0.5, -0.5 );
6     glVertex3f( -0.5, -0.5, -0.5 );
7 glEnd();
8
9 // TRASEIRA
10 glBegin(GL_POLYGON);
11     glVertex3f( 0.5, -0.5, 0.5 );
12     glVertex3f( 0.5, 0.5, 0.5 );
13     glVertex3f( -0.5, 0.5, 0.5 );
14     glVertex3f( -0.5, -0.5, 0.5 );
15 glEnd();
16
17 // DIREITA
18 glBegin(GL_POLYGON);
19     glVertex3f( 0.5, -0.5, -0.5 );
20     glVertex3f( 0.5, 0.5, -0.5 );
21     glVertex3f( 0.5, 0.5, 0.5 );
22     glVertex3f( 0.5, -0.5, 0.5 );
23 glEnd();
24
25 // ESQUERDA
26 glBegin(GL_POLYGON);
27     glVertex3f( -0.5, -0.5, 0.5 );
28     glVertex3f( -0.5, 0.5, 0.5 );
29     glVertex3f( -0.5, 0.5, -0.5 );
30     glVertex3f( -0.5, -0.5, -0.5 );
31 glEnd();
32
33 // TOPO
34 glBegin(GL_POLYGON);
35     glVertex3f( 0.5, 0.5, 0.5 );
36     glVertex3f( 0.5, 0.5, -0.5 );
37     glVertex3f( -0.5, 0.5, -0.5 );
38     glVertex3f( -0.5, 0.5, 0.5 );
39 glEnd();
40
41 // BASE
42 glBegin(GL_POLYGON);
43     glVertex3f( 0.5, -0.5, -0.5 );
44     glVertex3f( 0.5, -0.5, 0.5 );
45     glVertex3f( -0.5, -0.5, 0.5 );
46     glVertex3f( -0.5, -0.5, -0.5 );
47 glEnd();
48
49 glFlush();
50 glutSwapBuffers();
```

Listagem 1 - Código para criar um Cubo em OpenGL.

Fonte: WikiHow

Wow! E isso é só a parte de desenho em si. Esse cubo não tem uma textura, não tem cores, não tem nada. E ainda mais, não estamos mostrando a parte de criação da janela, de inicialização da câmera, de posicionamento da câmera, blá-blá-blá... Ah! Estamos falando apenas de todos os detalhes gráficos da coisa! Imagina que também temos a parte de áudio, de física, de inteligência artificial... Complicado, né?! Porém é bem mais fácil que aquele Assembly, você concorda?

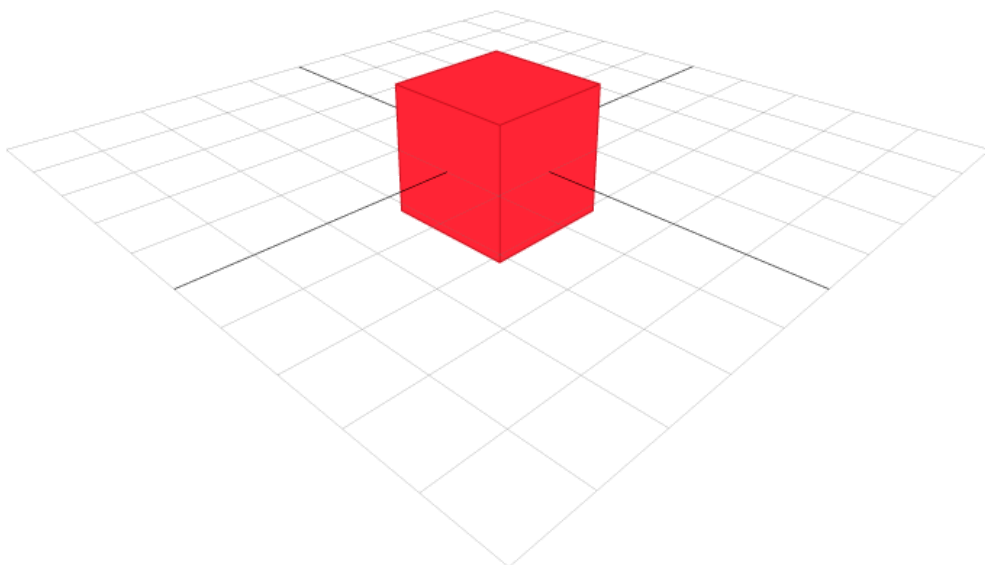
Que tal discutir com seus colegas no fórum suas impressões sobre ambos?

Poxa, professor! Mas eu queria que fosse ainda mais fácil... Não dá? Dá sim, senhor(a)! Veremos agora como podemos fazer a criação de um cubo em uma biblioteca gráfica um pouco mais alto nível, pois esta encapsula justamente o código OpenGL para que você não tenha o trabalho de escrevê-lo. Um cubo na [Raylib](#), na

Listagem 02:

```
1 Camera camera;  
2 Vector3 cubePosition = { 0.0f, 0.0f, 0.0f };  
3 Begin3dMode(camera);  
4 DrawCube(cubePosition, 2.0f, 2.0f, 2.0f, RED);  
5 End3dMode();
```

Listagem 2 - Criação de um Cubo na Raylib



Cubo gerado. Captura de tela da Raylib.

Fonte: Adaptada de Raylib (2017).

Bem melhor, não acha? Nessa nova proposta, especificamos a câmera encapsulada em um objeto Camera, criamos um Vector3 explicitando a posição do cubo e, com a chamada de uma única função, desenhamos o cubo em tal posição. Facilita muito! Mesmo assim, é complicado para um jogo mais complexo como os que temos hoje em dia, principalmente por não ter nada de física ou de áudio, por exemplo. Ainda precisamos nos preocupar com a criação da janela, com a atualização, com a maneira que redesenharemos a tela para fazer a animação acontecer e tudo o mais. E aí surgem os Motores de Jogos: para facilitar ainda mais as nossas vidas! ;)

A existência de motores de jogos não é uma novidade. Temos motores potentes e abertos já há algumas décadas. A novidade no momento é que motores os quais antes custavam milhares, ou até milhões de dólares, estão abrindo seus usos aos desenvolvedores *indies* para que os jogos criados possam ter uma qualidade tão boa quanto desejarem! E foi assim que a [Unity 3D](#) ganhou a sua popularidade.

Trata-se de um motor de jogos muito potente e completo, com funcionalidades que vão desde as funções mais básicas, de criação de janelas, câmeras e primitivas, até as mais complexas, como som 3D, criação de caminhos, renderização avançada, física 2D e 3D, entre outras. Tudo isso, claro, com uma interface limpa, de fácil entendimento, e um ótimo suporte através de documentação, tutoriais, fóruns e etc.

Depois de ter aberto as suas portas ao público, a Unity 3D lançou também um novo pacote de funcionalidades que cobrem muito bem a parte de desenvolvimento 2D, visto que, para muitos desenvolvedores, até por questão de recursos, esse é o único formato em que se é possível desenvolver para começar. Com essa liberação, aliado ao crescimento do mercado de dispositivos móveis e a subsequente liberação da compilação de seus jogos para essa plataforma por parte da Unity, houve um grande *boom* no número de usuários desse poderoso motor.

Vamos pensar um pouco. Um motor de jogos completo, que é capaz de desenvolver em 2D com facilidade, grátis (discutiremos isso em breve), e permite lançar os jogos que estou desenvolvendo para Computador, Android, iOS e outras plataformas ao mesmo tempo sem nenhum custo adicional. Ok. Quero.

E foi assim que muitos desenvolvedores e empresas de todos os níveis pensaram! Jogos *Indie* poderiam ser desenvolvidos com facilidade por empresas de menor porte, jogos grandes poderiam ser desenvolvidos por empresas grandes com um custo menor. Por quê não?

Pensando nisso tudo, escolhemos o Unity como motor a ser estudado nesta disciplina. Vocês poderão utilizar o conteúdo aprendido para desenvolver seus próprios jogos ou também se envolver no mercado de trabalho com mais facilidade, visto que diversas empresas da área estão utilizando esse motor para o desenvolvimento de seus jogos. Isso depende de como vocês pretendem aplicar os conhecimentos adquiridos no decorrer do curso e das motivações para aprender sempre mais! Não se esqueçam, no entanto, da história que estudamos aqui: Assembly, OpenGL e bibliotecas, pois o conhecimento é importante quando buscamos ser um profissional completo! Quem sabe alguns de vocês adquiram um interesse diferenciado e queiram não apenas usar o motor, mas desenvolver um! Algumas empresas pensam assim e vocês também podem! Só não abordaremos isso por aqui, ok?

Ah... E o tal do cubo? Como faz no Unity? Veremos! Mas é só clicar em GameObject -> 3D Object -> Cube. Em seguida, clica e arrasta para posicionar! ;-)

A Utilização do Unity

Como discutido anteriormente, a utilização de motores de jogos facilita o desenvolvimento de jogos ao trazer diversas simplificações ao processo. Discutimos como é importante haver simplificações para o código gráfico, para a física, para o áudio, enfim, para todos os aspectos que fazem um jogo. Além disso, concluímos que uma ótima opção, a qual traz todos esses aspectos mencionados, é o motor Unity 3D. Então, veremos mais detalhes sobre o que faz esse “bicho”!

O primeiro aspecto importante a ser tratado quando se fala da utilização de qualquer motor, ou qualquer ferramenta, na verdade, é discutirmos a licença que essa ferramenta possui para o seu uso. O Unity tem uma licença um pouco complicada (e quase fez com que vocês não tivessem essa disciplina :/), mas que favorece (e muito!) os desenvolvedores pequenos! Após uma reformulação recente, o Unity criou algumas categorias grátis e pagas, para atender a públicos diversos.

Funciona assim: se a empresa na qual você está envolvido ou você, enquanto desenvolvedor, tem um lucro anual menor que \$100.000 (cem mil dólares), é seu direito utilizar o Unity Personal! Essa versão possui todos os recursos do motor em si, compila para todas as plataformas, mas não tem acesso ao suporte premium, ao material de certificação (mas vocês têm essa disciplina!), entre algumas outras funcionalidades adicionais que, para o desenvolvimento de um jogo, não farão falta. Essa versão é completamente gratuita, com atualizações inclusas!

Já para empresas com rendimento maior que esse, independentemente da fonte do rendimento (e por essa razão abrangem diversas instituições de ensino, inclusive a nossa), é necessário que haja uma contratação do Unity Pro. Essa versão traz alguns suportes premium, além de acesso a alguns materiais a mais, mas isso apresenta um custo alto, em torno de R\$ 400 (quatrocentos reais) mensais, por máquina. Pois é! Ainda bem que, recentemente, o Unity resolveu fazer parcerias com instituições de ensino para montagem grátis de laboratórios! Em troca disso, as instituições devem formar alunos capacitados para utilização dessas ferramentas e, assim, preencher os espaços no mercado de trabalho no intuito de manter a tecnologia em uso! Façam isso, por favor, ok? Formem-se!

Passada a parte da licença, o segundo aspecto necessário quando escolhemos uma tecnologia para trabalhar é saber se ela é capaz de suprir as nossas necessidades enquanto desenvolvedores. Esse segundo aspecto é muito importante para evitar que fiquemos presos no meio do desenvolvimento porque simplesmente chegamos em uma parte da caminhada na qual não conseguimos mais avançar com o equipamento disponível. Para evitar isso, o melhor caminho é se informar de antemão e saber de tudo o que é preciso levar. Quem nunca jogou um RPG e precisou voltar à cidade para pegar aquele equipamento que esqueceu, né? Tempo perdido!

Nesse ponto, a Unity simplesmente destrói. De verdade! Como já conversamos, nesta disciplina buscaremos abordar todos os aspectos necessários para o desenvolvimento de um jogo 2D do começo ao fim. Veremos como adicionar os elementos gráficos, como colocar física no jogo, como criar a interface com o usuário, incluindo menus e HUDs. Veremos também a parte de áudio, animações em 2D, um pouco de criação de inimigos e algumas coisas mais! E a Unity é capaz de resolver tudo isso! À medida que avançarmos nas aulas, vocês verão como tudo isso pode ser criado. Ansiosos? Eu também! ;)

Atividade 01

1. Faça uma pesquisa na Internet e encontre quais os motores de jogos mais utilizados atualmente pelos desenvolvedores mundo afora.
2. Quais os principais componentes de um motor de jogos?

Desenvolvendo Jogos no Unity

Agora que já discutimos bastante a razão de utilizarmos um motor de jogos e também o motivo de utilizarmos o Unity, especificamente, então vamos falar sobre os aspectos desse motor relacionados ao desenvolvimento. O Unity permite que desenvolvamos os scripts necessários para os nossos jogos em duas linguagens principais: JavaScript e C#. E aí você pensa: "Oh, God! Mas eu não sei nenhuma das duas, o curso só ensinou Java, socorro!". E eu lhe digo: - Calma, jovem!

Calma, jovem!



Nós sabemos que nenhuma dessas linguagens foi abordada durante as suas aulas do Módulo Intermediário, mas também sabemos que as disciplinas, apesar de terem sido em Java, ensinaram a vocês todos os conceitos importantes, além da própria linguagem em si. Portanto, devem saber o que é uma condicional, uma estrutura de repetição, uma função, uma classe, um objeto, uma herança, pois tudo isso já foi visto nas disciplinas de programação! Ok, você está certo, foi visto em Java, mas o conceito também foi passado! E é somente com esses conceitos que

trabalharemos. Ao longo de nossos exemplos com códigos, sempre mostraremos a vocês umas dicas sobre a linguagem e os detalhes importantes a serem incorporados relacionados à linguagem em cada parte.

Ah! Existe muito material na Internet sobre o assunto e os fóruns da turma estão disponíveis para esclarecer dúvidas. Temos os tutores e também o professor! Enfim, recursos de ajuda não faltam!

Dito isso, o Unity estima que mais de 80% de seus projetos utilizam C# como linguagem oficial. Por esse motivo, iremos pelo mesmo caminho, então utilizaremos C# em nossos exemplos. Aos que não conhecem a linguagem, saibam que é um misto, criado pela Microsoft, entre C++ e Java (JAVA!), considerada uma linguagem multiparadigma, ou seja, pode ser utilizada também com conceitos de orientação a objetos.

Para os que estão iniciando os seus estudos nessa nova linguagem ou que desejem aprender um pouco mais sobre ela, é importante que utilizem sempre os fóruns para o esclarecimento de dúvidas, assim como os momentos com os tutores nas aulas presenciais, a fim de que aprendam a desenvolver seus jogos com segurança, utilizando as ferramentas mais interessantes para as suas propostas! Estaremos sempre juntos, combinado?

Instalação do Motor de Jogos

Vamos agora dar uma olhadinha rápida no passo a passo para a instalação do Unity. Não tem nada de complicado ou muito diferente na instalação, então vamos apenas mostrar alguns pontos principais e discuti-los rapidamente para ajudar apenas nas decisões importantes que precisarem ser tomadas ao longo do caminho.

O primeiro passo é acessar o site do Unity (<https://unity3d.com/pt/>) e entrar na parte de download do produto através do menu “Obtenha o Unity”. Ao fazer isso, você será direcionado à página de escolha do produto que mais se adequa às suas necessidades. Lembre-se que a escolha está também diretamente ligada ao faturamento anual de sua empresa, como dito anteriormente. Veja o site com as opções a seguir, na **Figura 01**.

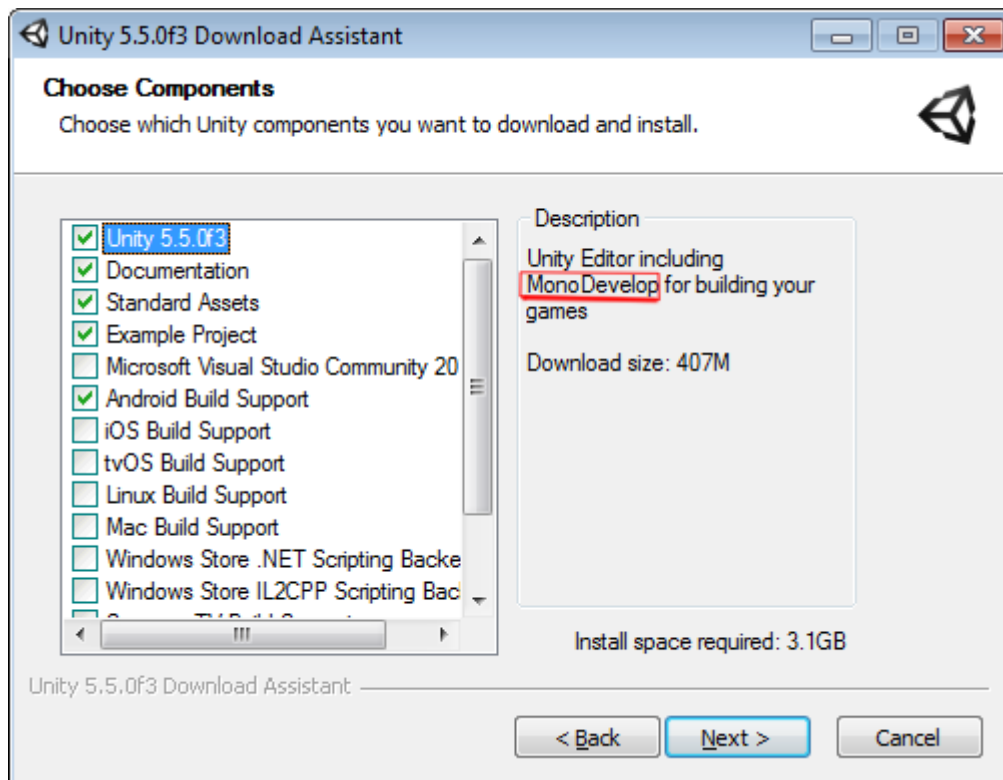
Figura 01 - Site de download do Unity com as licenças disponíveis.



Fonte: Captura de tela do Unity – Game Engine. Disponível em: <https://unity3d.com/pt/> Acesso em: 24 de jan de 2017.

Selecionada a versão adequada, o próximo passo é baixar e executar o instalador. Avance na instalação, leia a licença e aceite, caso deseje prosseguir, concordando com os termos, e então você será levado à primeira escolha a ser feita. O Unity perguntará qual arquitetura será utilizada para a instalação. Caso esteja em um sistema 64 bits, escolha essa opção para instalação, uma vez que o desenvolvimento para consoles depende dessa arquitetura para funcionar. Avançando mais uma vez, chegaremos à tela de escolha de componentes que serão instalados. Nessa tela, temos algumas opções. Vejamos na **Figura 02**.

Figura 02 - Componentes a serem instalados pelo Unity.



Fonte: Captura de tela do Unity. Disponível em: <https://unity3d.com/pt/>

O primeiro componente, marcado na **Figura 02**, é o próprio editor do Unity. Perceba, pela descrição contida, no entanto, que ele também inclui o MonoDevelop, um editor simples e leve para os scripts que forem incluídos no seu jogo. Com isso, podemos não instalar o Microsoft Visual Studio Community, caso não queiramos lidar diretamente com essa interface, que é um pouco mais completa e muito mais pesada que o MonoDevelop. Ao longo do nosso curso não a utilizaremos, mas fique à vontade para instalar ou não de acordo com o seu gosto. Essa é a primeira opção não marcada da imagem.

Abaixo do editor em si, temos a opção de *Documentation*. Essa opção baixa e instala toda a documentação do Unity em seu computador, o que facilita o entendimento do funcionamento de algumas funções e pode servir como material de consulta offline, caso seja necessário. **Recomendamos a instalação.**

Na sequência, conheceremos dois componentes interessantes. Os Standard Assets são um conjunto de Assets, ou Recursos (estudaremos isso mais em detalhes na **Aula 02** desta disciplina), padrão do Unity que incluem alguns sons, alguns modelos, alguns sprites, entre outros recursos que são utilizados nos projetos de

exemplo do Unity. Ao incluir esses recursos em sua instalação, você ganha acesso a eles para utilizar em seus projetos como teste até obter os seus recursos, ou mesmo como referência para desenvolver os seus. É um pacote interessante que também vale a pena ser instalado.

O outro pacote, o *Example Project*, contém os projetos exemplos que são construídos utilizando os recursos padrões citados anteriormente, baixados no pacote acima desse. Com isso, é possível você já obter alguns exemplos *plug and play*, precisando somente abrir no Unity, e eles já estarão funcionando. É uma estratégia interessante para conhecer um pouco mais do motor sem precisar desenvolver nada!

Em seguida, após o Visual Studio que desmarcamos nesse caso, temos os pacotes de Build Support. Esses pacotes são responsáveis por fazer exatamente o que o nome diz: dar suporte à compilação para as plataformas que estão atrelados! Recomendo sempre marcar o Android Build Support, uma vez que o Android é uma plataforma amplamente difundida e pode ser interessante desenvolver jogos voltados a ela. É simples de testar, por ser uma plataforma bem aberta (diferente do iOS, da Apple) e pode trazer um *plus* aos nossos jogos desenvolvidos: funcionar em dispositivos móveis!

Além disso, há outras plataformas que você pode incluir o suporte, se quiser. Basta pesquisar um pouco sobre a plataforma desejada e, caso tenha interesse desenvolver para ela, é só marcar essa opção e fazer o download. Caso não queira tomar essa decisão agora, não há problema, uma vez que é possível baixar esses suportes de dentro do Unity mesmo, após ele ter sido instalado.

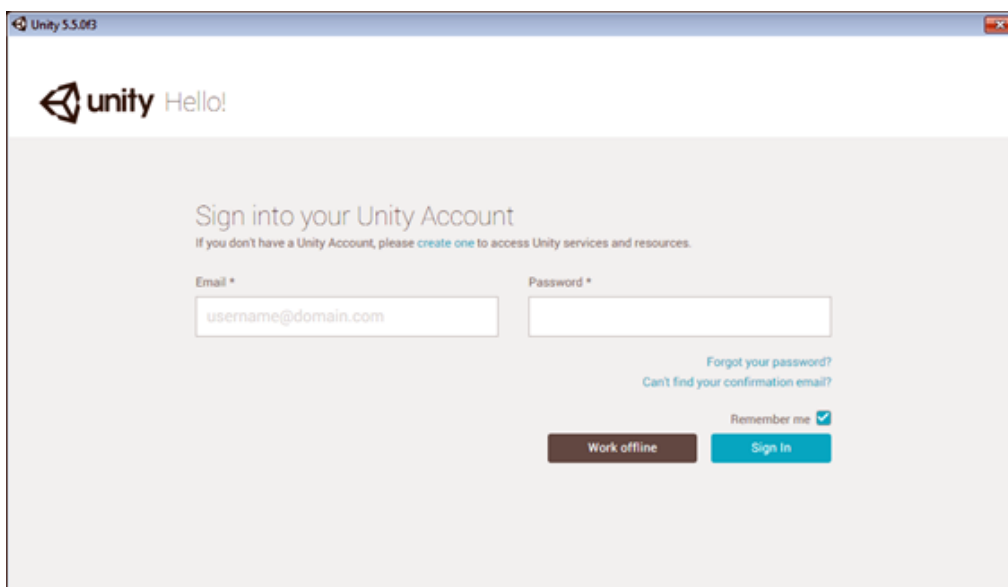
Escolhidos os pacotes, avance pela instalação, escolha o diretório no qual o Unity será instalado e simplesmente aguarde a mágica acontecer. Ao concluir todos os downloads e instalações (pode demorar um pouco, principalmente se a Internet estiver lenta), o instalador se concluirá com uma opção de iniciar o Unity instalado. Inicie-o e vamos adiante!

Inicialização do Motor de Jogos e Criação de Projetos

Legal! Até aqui já conversamos bastante, abordamos um pouco de história, falamos acerca do motor que utilizaremos e da linguagem, porém ainda não fizemos nada! Tudo bem, afinal essa é a primeira aula, ainda temos muito o que acertar e mostrar! Vamos agora nos aproximar um pouco mais do que serão as próximas aulas. Mão na massa!

Inicialmente, vejamos as telas iniciais que aparecem ao utilizarmos o Unity pela primeira vez! Percebam que se vocês forem seguir esses passos no laboratório, ou em algum lugar em que o Unity já tenha sido executado, não é necessário repeti-los, apenas adaptá-los. O primeiro passo, no entanto, é muito importante! Vejamos a primeira tela que o Unity exibe ao ser inicializado pela primeira vez, na **Figura 03**.

Figura 03 - Tela de criação de conta no Unity.



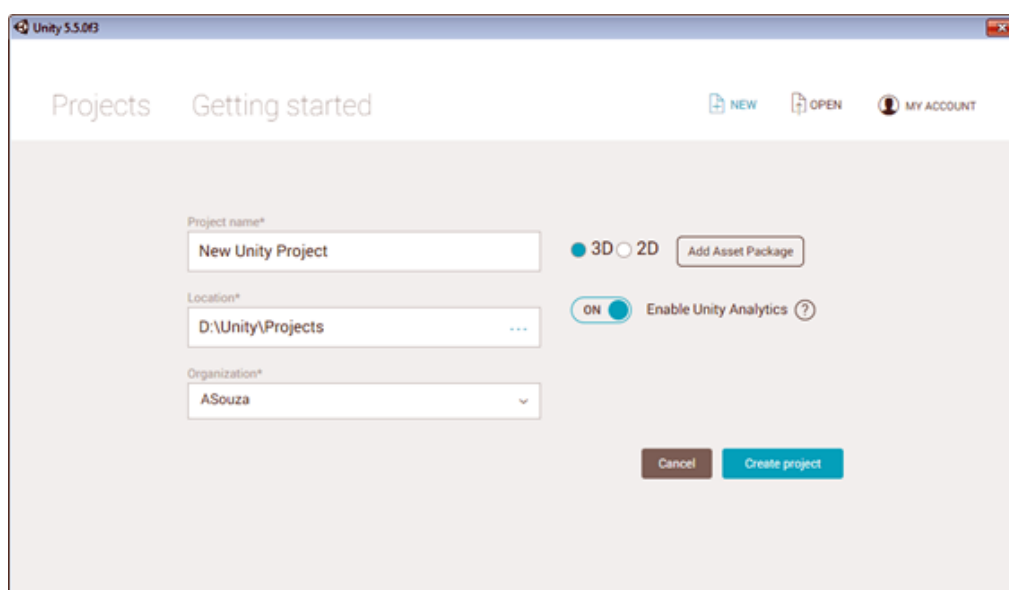
Fonte: Captura de tela do Unity. Disponível em: <https://unity3d.com/pt/>

Como vocês podem ver, a primeira tela do Unity é a de login! O Unity possui um sistema de contas (é grátis!) o qual permite que você tenha uma identificação única ao utilizar o programa. Com essa conta você pode manter as suas organizações, gerenciar a sua verba de Ads (propagandas), entre diversas outras coisas. Por esse

motivo, é interessante ter sua própria conta e utilizá-la sempre que possível. Sabendo disso, vai lá e faz a sua! Há também a opção de trabalhar offline, clicando no botão Work offline. Se necessário, utilize-a.

Após passar pela tela inicial, o Unity te levará a uma tela de criação de novos projetos, (**Figura 04**) se executado pela primeira vez, ou para a tela de projetos, se algum já existir. Caso tenha ido direto à tela de projetos, basta clicar em New e então você será direcionado à tela da **Figura 04**.

Figura 04 - Criação de um novo projeto no Unity.

A imagem mostra a interface de criação de um novo projeto no Unity 5.5.0f3. No topo, há uma barra de navegação com 'Projects' e 'Getting started'. À direita, há links para 'NEW', 'OPEN' e 'MY ACCOUNT'. O formulário principal contém: 'Project name*' com o valor 'New Unity Project'; 'Location*' com o valor 'D:\Unity\Projects'; 'Organization*' com o valor 'ASouza'. À direita do formulário, há uma seleção entre '3D' (ativo) e '2D', um botão 'Add Asset Package', uma opção 'ON' para 'Enable Unity Analytics' e um ícone de ajuda. No rodapé, há botões 'Cancel' e 'Create project'.

Fonte: Captura de tela do Unity. Disponível em: <https://unity3d.com/pt/>

Nessa tela, temos alguns pontos importantes a serem discutidos. O primeiro deles é o *Project Name*. Nesse campo, você dará um nome ao seu projeto Unity. Esse nome servirá apenas para identificação dentro do próprio Unity e não influenciará o produto que será gerado ao final (o jogo que você desenvolverá). Escolha um nome que o ajude a identificar o projeto, pois é fácil se perder caso não haja organização.

O segundo ponto é o *Location*. Esse campo indica onde o seu projeto será salvo no disco. Usualmente o Unity cria uma pasta com o nome do projeto dentro da pasta que está descrita em *Location*. Escolha uma localização conhecida, pois precisaremos lidar com os arquivos do projeto ao longo do desenvolvimento, além disso é necessário que você navegue até a pasta escolhida. Note, no entanto, que é possível acessar essa pasta diretamente do Unity, como veremos depois.

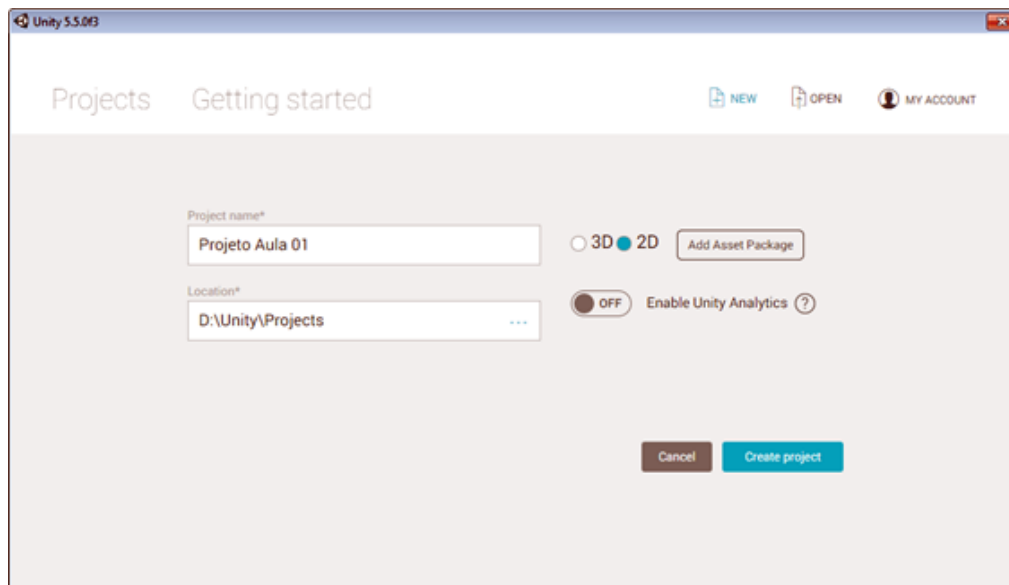
Seguindo adiante temos o campo *Organization*, que está diretamente ligado ao campo *Enable Unity Analytics*. O Unity Analytics gera algumas estatísticas sobre o seu projeto, que são coletadas durante o desenvolvimento. Itens como métricas, [benchmarks](#) em comparação a outros projetos similares, entre outras informações, podem ser colhidas utilizando essa funcionalidade. Isso tudo ficará atrelado à sua organização e por esse motivo é importante criar uma e escolhê-la ao selecionar essa opção. Mais informações sobre o Analytics e também sobre organizações podem ser encontradas na própria documentação do Unity, cujo link para acesso está indicado na Leitura Complementar. Por enquanto, não utilizaremos essa opção.

Benchmark, no universo da computação, é a execução de programas de computador para comparar performance e desempenho de um objeto.

Fonte: Canaltech. Disponível em: <https://m.canaltech.com.br/o-que-e/o-que-e/O-que-e-benchmark--26350/>.

Os dois últimos pontos referem-se à escolha entre 2D e 3D e o botão de adicionar pacotes de recursos (*Add Assets Packages*). A escolha entre 2D e 3D demandará algumas pré-configurações no seu projeto, facilitando o desenvolvimento na opção escolhida. Como lidaremos com 2D em nossos projetos, recomendo que escolha essa opção. Já o botão de adicionar recursos permite que você escolha algum dos pacotes de recursos do Unity para adicionar ao seu projeto desde a criação. Isso facilita o desenvolvimento caso o objetivo seja de fato trabalhar com algum desses recursos ou rodar algum dos exemplos que instalamos anteriormente. Por enquanto, não vamos utilizar nada e criaremos o nosso primeiro cenário do zero! Dito isso, as configurações para o nosso primeiro projeto poderão ser visualizadas na **Figura 05**, a seguir.

Figura 05 - Tela de criação de novos projetos já configurada.



Fonte: Captura de tela do Unity. Disponível em: <https://unity3d.com/pt/>

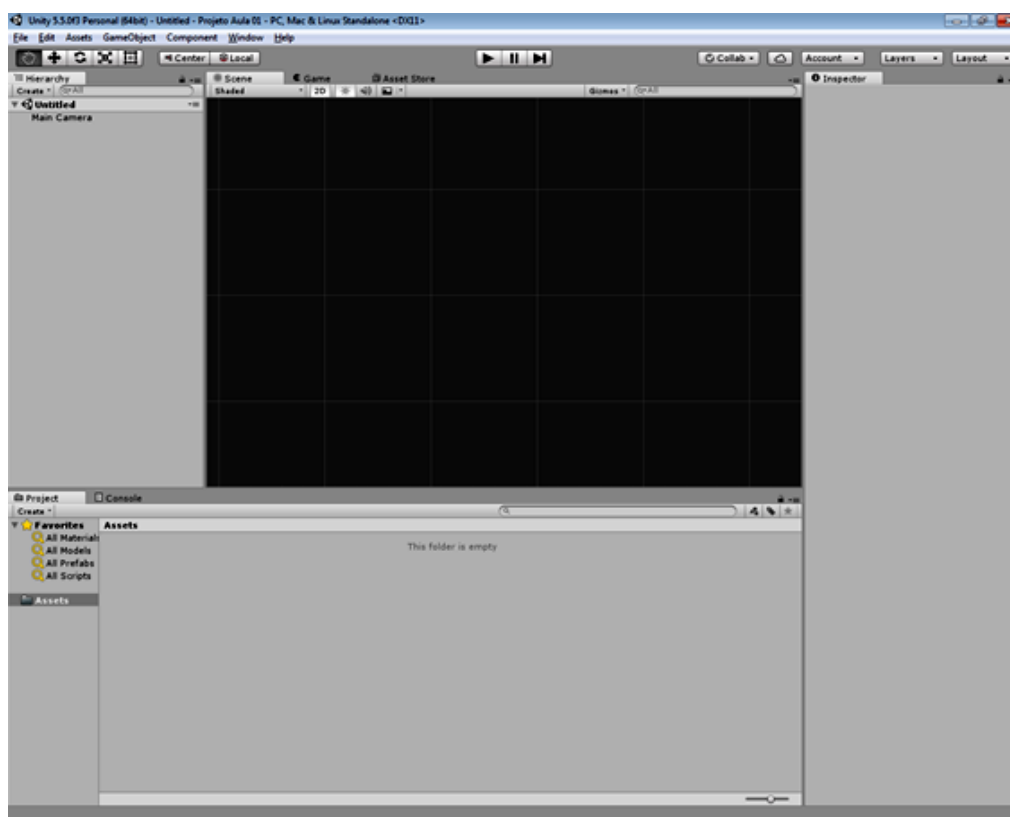
Perceba que o caminho escolhido e o nome do projeto ficam a seu critério. Estamos utilizando esses simplesmente para facilitar a referência se precisarmos falar sobre isso adiante. Após toda a configuração feita, cliquemos em Create Project e então o projeto será criado!

Antes de encerrarmos essa seção, percebam apenas que há, no canto superior esquerdo, uma aba chamada Projects. É por essa aba que podemos visualizar os projetos abertos recentemente e voltar a eles, caso necessário. Há também, no canto superior direito, um botão de Open, que cumpre função similar.

Interface do Motor de Jogos

Agora que temos o nosso primeiro projeto criado, podemos observar a interface do nosso motor de jogos pela primeira vez, conforme mostra a imagem a seguir. A **Figura 06** será a base para que possamos discutir cada uma das abas que aparecem, visando um melhor entendimento do motor. Vamos conhecê-las!

Figura 06 - Interface do Motor Gráfico.



Fonte: Captura de tela do Unity. Disponível em: <https://unity3d.com/pt/>

O que vemos na **Figura 06** é a interface inicial e padrão do Unity. Ela traz as principais abas já abertas e organizadas de uma maneira que faz sentido para o desenvolvimento. Na barra de menu há algumas opções, como a *Window*, a partir da qual podemos mudar a interface ou abrir outras janelas que não estão visíveis. Dentro desse menu há uma opção chamada *Layout*, que permite também trocar o modo como essas abas se posicionam na tela. Esse que estamos mostrando é o *layout Default*. Explore um pouco esse menu para conhecer melhor as opções.

Falando agora da interface, passaremos pelas abas da esquerda para a direita, de cima para baixo. A primeira delas que você vê, abaixo dos botões de manipulação de objetos, é a *Hierarchy*. Essa aba conterá sua **cena**, nome que o Unity dá a cada nível que é criado e todos os elementos que nele estão contidos, além dos objetos presentes nela. Perceba que ao criar um novo projeto, a cena não vem automaticamente salva, é então chamada de **Untitled** e contém apenas um objeto chamado de **Main Camera**. Esse objeto representa a nossa câmera e será discutido em detalhes em uma aula mais adiante. Todos os outros objetos que adicionarmos em nossa cena serão incluídos nessa aba e poderão ser selecionados para manipulação através dela.

Em seguida, à direita de *Hierarchy*, temos um agrupamento de três abas de acordo com o layout padrão. São elas: *Scene*, *Game* e *Asset Store*. A primeira delas, *Scene*, é responsável por mostrar a cena e como ela está organizada. É através dessa tela que podemos posicionar elementos, alterar suas propriedades visualmente e montar o nosso mundo da maneira que desejarmos. Notem que a cena apresentada nessa tela não é o que vemos no jogo em si, pois o que será posicionado para a nossa câmera e renderizado de fato está na aba seguinte: a aba *Game*. Outro detalhe interessante dessa aba são os botões que estão na barra logo abaixo dela. Entretanto, merece destaque aqui o botão 2D, que já deve vir marcado após a criação de um projeto 2D. Ele simplesmente faz sumir uma dimensão do mundo para que possamos trabalhar em 2D (apesar de que o projeto ainda é, e sempre será, 3D, efetivamente).

Passando para a segunda aba desse agrupamento, temos a aba *Game*. Nessa aba vemos o que de fato será o nosso jogo, já através da câmera e com todos os efeitos de mundo renderizados. O que vemos nessa aba, o nosso jogador verá no jogo. Por esse motivo, ao testarmos o jogo apertando o botão de Play (|>) que fica acima dessas abas, o Unity trará automaticamente essa aba à frente para que possamos jogar o nosso jogo, testando-o.

Por fim, a terceira aba é a aba da *Asset Store*. Como o nome diz, essa aba nos leva à loja de recursos do Unity. Essa prática é comum entre os motores abertos e dá aos desenvolvedores mais uma maneira de obter algum lucro, ajudando outras pessoas com os seus talentos. Nessa loja é possível encontrar os diversos recursos que formam um jogo e adquiri-los gratuitamente ou com valores elevados. Isso é importante, no entanto, por permitir que desenvolvedores os quais estejam com algum problema no desenvolvimento ou mesmo que desejem acelerar o processo possam adquirir soluções prontas nas diversas frentes de um jogo. É possível encontrar scripts que sirvam como ponto de partida na programação, é possível encontrar elementos gráficos 2D e 3D, é possível encontrar sons... Tudo isso com seu preço, claro! Vejamos na **Figura 07** a interface da *Asset Store*, com alguns produtos já sendo exibidos na tela inicial.

Figura 07 - Asset Store do Unity.



Fonte: Captura de tela do Unity. Disponível em: <https://unity3d.com/pt/>

Percebam que a *Asset Store* funciona para os dois lados! Se você está precisando de um recurso, pode ir lá e conseguir o que quer. Porém, caso seja um bom artista/programador/músico/criador de conteúdo, também pode aplicar o seu produto à loja e ganhar, com isso, algum dinheiro a mais, de acordo com as vendas que forem efetuadas! Pense sobre isso e explore um pouco a loja, para ver tudo que é possível encontrar lá.

Sério! Para de ler um pouco e abra a loja. Vou até colocar uma atividade com isso para você olhar mesmo! Quem sabe aquele recurso especial, cuja existência lhe devolveria as noites de sono perdidas, não está lá, esperando por você?

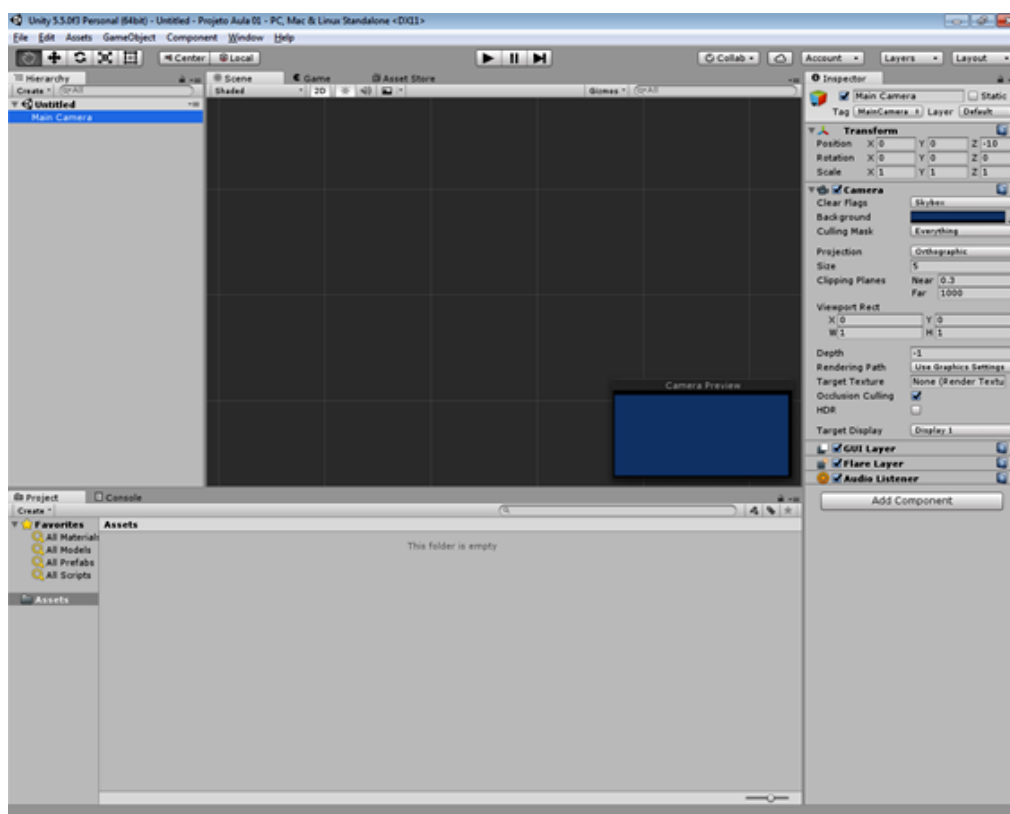


Atividade 02

1. Explore a Asset Store, selecione alguns elementos que você achou interessante e compartilhe nos fóruns com seus colegas.

Continuando a exploração da interface de acordo com a **Figura 06** vista anteriormente, a próxima aba, à direita, é a aba do *Inspector*. É nessa aba que vemos todas as propriedades e componentes de um objeto que tenhamos adicionado à nossa cena. Também através dessa aba podemos editar todas essas propriedades por meio de uma interface bem simples e facilitada. Para entender melhor essa parte de componentes e propriedades, selecionaremos a câmera da nossa cena e observaremos suas propriedades, como visto na **Figura 08**

Figura 08 - Propriedades da câmera exibidas no *Inspector* após selecionar o objeto câmera na aba *Hierarchy*.



Fonte: Captura de tela do Unity. Disponível em: <https://unity3d.com/pt/>

Percebam que, agora, após selecionar a câmera, o *Inspector* mudou para conter as propriedades relacionadas ao objeto selecionado. A primeira dessas propriedades é o nome do objeto, no caso, Main Camera. A checkbox ao lado do nome indica se ele está ativo ou não. Apenas objetos ativos são mostrados no seu jogo. Objetos inativos são adicionados, mas não se tornam parte do jogo até serem ativados (via script, por exemplo).

Em seguida, dentro do *Inspector*, vemos os componentes que fazem parte do objeto. O primeiro deles, o componente ***Transform***, é comum a todos os objetos e contém informações de posicionamento, rotação e escala de cada um deles. Todos os objetos devem possuí-lo para que o motor possa saber onde posicionar o objeto em questão, além da rotação e da escala a serem utilizadas para tanto.

Perceba que os objetos possuem um ou mais componentes, nomes dados a esses conjuntos de propriedades que estão editáveis e mostrados no *Inspector*. Cada componente possui uma função única e está relacionado a um aspecto diferente do objeto. Veremos adiante como adicionar novos componentes, mas, por enquanto, guarde bem esse nome, pois falaremos deles por todo o curso!

Os outros componentes contidos nesse objeto não serão discutidos nesta aula. Deixaremos para falar um pouco mais sobre eles e todas as suas propriedades na próxima aula, quando abordarmos melhor sobre a câmera!

Para finalizar a nossa visualização da interface padrão do Unity, vejamos as duas abas que estão na parte de baixo: *Project* e *Console*.

A aba *Project* é o navegador de pastas que temos para gerenciar os recursos de nossos projetos. É nessa aba que poderemos criar novas pastas, arrastar os recursos para uma ou para outra, abri-los para edição, adicioná-los à cena (arrastando da aba *projects* para a aba *Scene*) ou mesmo criar *Prefabs*, recursos pré-fabricados que o Unity permite que sejam criados para que possamos replicar no futuro sem a necessidade de redefinir as mesmas propriedades. Esses elementos são bem importantes e serão revisitados em aulas futuras!

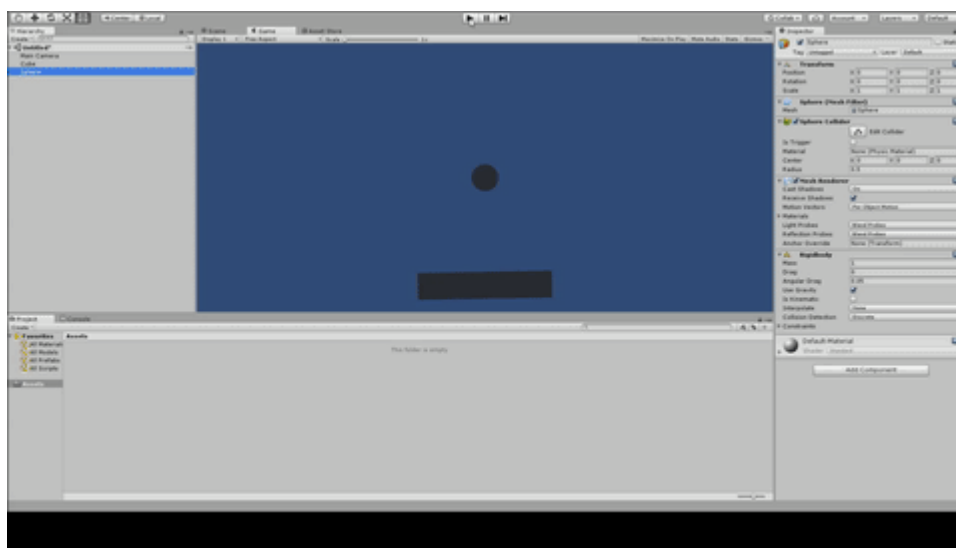
A última aba de nossa interface é a aba Console. Ela funciona exatamente como um console normal, utilizado em outros ambientes, como na programação Java, por exemplo. É nessa aba que veremos erros de compilação em nossos scripts, warnings indicando problemas em nossos códigos, mensagens escritas durante o debug, enfim, aspectos de programação em geral!

Fiquem sempre de olho nessa aba e utilizem também, sempre que conveniente, uma propriedade importante que pode ser acessada ao clicar nesta aba: *Clear on Play*. Ao marcar essa opção no console, todas as vezes que o seu jogo for reiniciado, o console será limpo antes da execução. Isso ajuda a separar erros ou mensagens gerados NESTA execução do jogo e não em alguma que você não tem nem mais ideia de qual versão do script estava rodando. Use!

Hello, World! (ou The EPIC Falling Ball Game)

Agora que já conhecemos a interface do nosso motor, temos o nosso projeto criado, entendemos o que o motor nos proporciona e sabemos o que é um componente de um objeto no Unity, estamos prontos! É chegado o grande momento: criaremos, juntos e do zero, sem ajuda externa, sem nada, um incrível jogo em que uma bolinha cai numa plataforma e escorre para fora dela! WOW \o/ Vejamos na INCRÍVEL **Figura 09**!!

Figura 09 - Bolinha caindo na plataforma e escorrendo para fora.



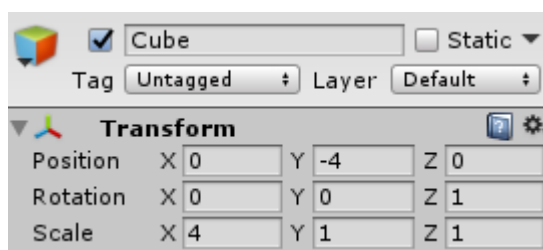
Fonte: Captura de tela do Unity. Disponível em: <https://unity3d.com/pt/>

Ok. Talvez não seja tão emocionante assim, mas todos precisam começar de algum lugar, correto? E com esse exemplo já é possível aprender muitas coisas interessantes! Então, vamos a ele!

Criando o Hello World

O primeiro passo para que tenhamos o nosso jogo funcionando é adicionar os elementos necessários. Para começar, adicione uma plataforma, equivalente a um cubo esticado. Para fazer esse exemplo, utilize objetos 3D. Apesar disso, a animação desenvolvida parecerá 2D, pois é assim que a câmera está configurada. Para adicionar um cubo à cena, utilize o menu GameObject -> 3D Object -> Cube. Isso criará um cubo no centro da tela. Utilizando a Hierarchy, selecione o cubo para que ele surja no Inspector. Ao fazer o cubo aparecer no Inspector, altere suas propriedades de acordo com a **Figura 10**.

Figura 10 - Transform do cubo que será a nossa plataforma.

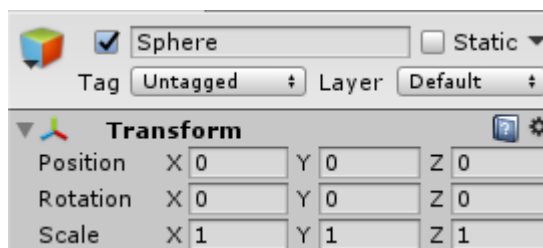


Fonte: Captura de tela do Unity. Disponível em: <https://unity3d.com/pt/>

Perceba que alteramos valores nos três componentes do Transform. Para a posição, baixamos a plataforma até o Y = -4. Como o Unity **posiciona os eixos sendo X na horizontal, Y na vertical e Z na profundidade**, ao colocar um Y negativo, estamos abaixando o objeto na vertical, como visto no posicionamento da **Figura 09**. Em seguida, alteramos a rotação em torno do eixo Z (o eixo que está perpendicular à tela) em 1 grau. Isso faz com que a plataforma se transforme em uma rampa de inclinação baixíssima, fazendo com que a nossa bola escorregue por ela lentamente, uma vez que a tenhamos adicionado. Por fim, modificamos a escala para X = 4. Com essa modificação, o cubo será aumentado de tamanho em 4 vezes na direção X, ou seja, horizontalmente. Com isso, ele parecerá uma pequena plataforma.

OK! Temos uma plataforma. Agora só precisamos da nossa esfera caindo para concluir o nosso primeiro exemplo. Para adicionar a esfera na cena, utilizamos um caminho similar: GameObject -> 3D Object -> Sphere. Modifique o Transform da esfera para a origem, ou seja, position: 0, 0, 0, como visto na **Figura 11**.

Figura 11 - Transform da esfera.



Fonte: Captura de tela do Unity. Disponível em: <https://unity3d.com/pt/>

Feito isso, já temos a nossa esfera posicionada acima da plataforma, e a plataforma posicionada para fazê-la escorregar. Tudo certo! Clique no botão Play, que fica centralizado no topo da interface e observe a mágica acontecer!

Não?

Nada acontece? Ok...

Esperado! É **muito importante** entender que o universo dos jogos que criamos não obedece, naturalmente, às leis da física, como é no mundo real. São apenas pixels na tela! Para que o seu jogo tenha alguma física (como uma bola que está voando possa cair), é necessário que você adicione alguma física a ele! Vamos então fazer alguns cálculos sobre quais as forças que atuam na bola e pensar um pouco sobre a lei de gravitação universal. Vamos relembrar a fórmula?

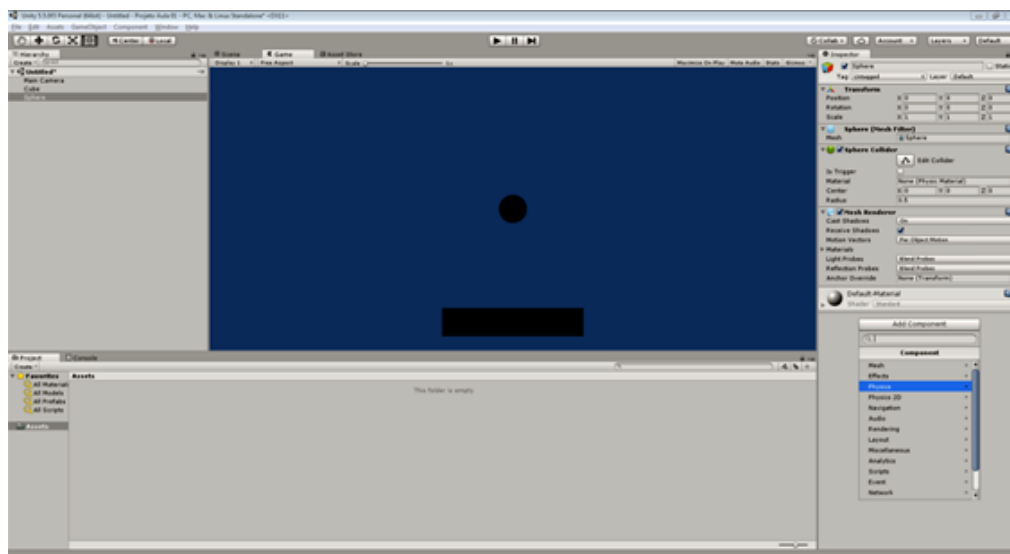


BRINCADEIRA! A Unity já tem um motor de física extremamente potente embutido e implementado para que você possa simplesmente configurar os objetos da maneira adequada e, então, eles passarão a obedecer aquelas leis criadas. Mais uma vez, repito: os objetos não obedecem naturalmente a uma lei mágica que rege o universo, como é a gravitação para nós. Mas eles podem passar a obedecer assim que forem configurados!

Você pode perguntar: - Como fazemos isso? A resposta é simples e está marcada como **IMPORTANTE** há algumas páginas: - Através de componentes! Os componentes farão tudo que você precisar que um objeto faça dentro de um jogo, e adicionar alguma física a eles é parte disso. Vamos, então, adicionar um componente físico ao nosso objeto esfera, de modo que ele passe a se comportar como um corpo rígido, obedecendo à lei da gravidade, atrito, etc.

Para adicionar um novo componente, selecione a esfera e, no inspector, clique no botão Add Component -> Physics -> Rigidbody, como vemos em parte na **Figura 12**. É importante escolher **Physics** e não **Physics 2D**, pois há uma diferença de comportamento entre os dois motores físicos que fazem com que as coisas não funcionem bem se forem misturadas. Quando começarmos, a partir da próxima aula, com objetos 2D de fato, passaremos a usar Physics 2D.

Figura 12 - Adição de componente ao objeto esfera.



Fonte: Captura de tela do Unity. Disponível em: <https://unity3d.com/pt/>

Ao adicionar esse novo componente, sua esfera passará a se comportar como um corpo rígido. Se você observar o novo componente inserido, verá que ele possui várias propriedades possíveis de ser configuradas, como a maneira que ele reage à gravidade, a massa que possui, entre outras questões. Não entraremos em detalhes nessas propriedades agora, mas saiba que elas existem!

Agora, sim! Temos um corpo rígido posicionado acima de um outro objeto que responde a colisões, então, podemos apertar play e ver a mágica acontecer de verdade!

YEAH! Aqui funcionou! Muito bom \o/ Já consigo ver como isso poderá ser o próximo Dark Souls.



Com esse interessante exemplo encerramos a nossa primeira aula. Caso tenham percebido algum problema no decorrer da aula, fiquem à vontade para usar o fórum e expor suas dúvidas! Estaremos sempre de olho para discutir cada ponto levantado e melhorar o entendimento de vocês e, consequentemente, o material!

Espero que tenha sido um começo agradável nesta disciplina. Vemo-nos em breve, guerreiros (ou magos, paladinos, amazonas, feiticeiras, o que for... sem preconceito!): Até lá!

Leitura Complementar

Para complementar o que estudamos nesta aula, vale a pena dar uma olhada na documentação oficial do Unity. Ela será nossa amiga e, ao longo das aulas, servirá como grande referência para nossos estudos.

Documentação Oficial do Unity (em Inglês ou Espanhol):

<https://docs.unity3d.com/Manual/index.html>

Resumo

Em nossa primeira aula, começamos entendendo um pouco da história da criação dos jogos e a evolução destes até chegar aos potentes motores de jogos que temos hoje. Em seguida, nos convencemos (espero que sim!) de como é uma ótima ideia utilizarmos esses avanços conseguidos, e escolhemos, para tanto, o Unity. Conhecemos também, durante a aula, a Asset Store, loja de recursos do Unity que contém diversos produtos à venda e gratuitos, para facilitar/ajudar o desenvolvimento de jogos no motor.

Após essa escolha, conhecemos melhor a interface do Unity, aprendemos a criar novos projetos, conhecemos os componentes dos objetos Unity e, para fechar com chave de ouro, implementamos o nosso primeiro projeto, utilizando primitivas 3D e o motor de física do Unity a fim de fazer uma bolinha cair sobre uma plataforma.

YEAH, BABY!



Autoavaliação

1. Indique a importância de se utilizar um motor de jogos em comparação a outras maneiras de se desenvolver um jogo.
2. Crie um projeto no Unity em 2D e adicione a ele primitivas 3D de maneira que elas interajam entre si.
3. O que é a Asset Store? Quais as vantagens que ela pode trazer aos desenvolvedores de jogos? E aos desenvolvedores de Assets?

Referências

Documentação oficial do Unity. Disponível em: <https://docs.unity3d.com/Manual/index.html>. Acesso em 24 de jan de 2017.

Tutoriais oficiais do Unity. Disponível em: <https://unity3d.com/pt/learn/tutorials>. Acesso em 24 de jan de 2017.

RABIN, Steve. **Introdução ao Desenvolvimento de Games**, Vol 2. CENGAGE.

Site Oficial do Unity. Disponível em: <https://unity3d.com/pt/>. Acesso em: 24 de jan de 2017.