Introdução ao Unity

Criação de Jogos
A criação de jogos não é uma tarefa simples. Ela envolve o conhecimento e uso de várias ferramentas e conhecimentos, como design, conteúdo, programação, softwares de imagens e de modelagem 3D, etc. Ainda bem que apareceram ferramentas que facilitam este trabalho, já colaborando e facilitando várias áreas.

Engine de Games
Uma engine para games facilita a criação de games pois abstrai de programar diretamente para Direct X ou OpenGL e já traz integrados vários recursos como um motor gráfico para renderizar os gráficos, um motor de física para simular física nos jogos, suporte a animações, a sons, a redes, uma ou mais linguagens de scripts, etc.

Unity
O Unity é uma engine que facilita esta tarefa de criação de jogos, pois tem uma interface amigável, sem contar que com ela criamos um jogo sem nos preocupar com a arquitetura que irá usá-lo e ao final apenas configuramos para que seja usado por uma entre diversas arquiteturas. Também oferece uma versão free inteiramente funcional e realmente free.

Os indie games têm inovado mais que os games das grandes companhias. Veja alguns títulos premiados ultimamente: Minecraft, Limbo e Super Meat Boy.

Para iniciar na criação de jogos a versão free do Unity é suficiente.

Importação
O Unity importa de uma grande quantidade de formatos, todos os principais, tanto gráficos quanto de áudio.

GameObject
GameObjects são os objetos ativos atualmente na cena aberta. Um **GameObject** praticamente não faz nada sozinho, para dar vida a ele precisamos adicionar **componentes** e em especial o componente script. Todo GameObject possui pelo menos o componente Transform, que é obrigatório e requerido, não pode ser removido.

Componentes
Eles podem adicionar comportamento, mudar aparência e influenciar em outros aspectos dos objetos. Eles dão vida aos objetos, especialmente os scripts.

Static
Podemos tornar um objeto estático, que não poderá se mover durante o jogo, desativando em Static à direita do nome do componente.

Scripts
São criados para dar interatividade aos objetos, e são considerados componentes pelo Unity. Conhecidos no Unity como behavior/comportamento. Permite tornar os objetos interativos.
Os scripts podem ser em UnitScript (similar ao JavaScript ou ActionScript), CSharp e Boo (variante do Python). Pense nos scripts como algo que ampliam e modificam as funcionalidades do Unity ou cria comportamento. Eles são partes essenciais da produção de jogos.
A linguagem Boo não é suportada pelos dispositivos móveis.

Editor de Scripts Monodevelop
Recomendado pelo Unity, auto-completa o código ao digitar e é nativamente desenvolvido e atualizado pela equipe do Unity.

Prefab

É uma forma de armazenar objetos eom configurações e componentes como assets para serem reusados em diferentes partes do jogo e então ser instanciados (clonados) a qualquer momento.
Um prefab é como um template ou uma referência, criada a partir de um GameObject, que quando criamos várias cópias dele e mudamos uma característica em uma cópia ou nele, ele altera também dos demais.

Criando um Prefab

Assets – clicar com o botão direito (de preferência na pasta Prefabs) – Create – Prefab
Observe que quando criado ele tem a cor cinza, mostrando que está desabilitado.
Ele não é como os demais objetos, que são criados na cena. Ele é criadoe fica na aba Project, junto aos demais assets.
Ao clicar sobre ele e observar o Inspector, nenhuma informação aparece.
Para que seja ativado precisamos arrastar um GameObject para ele.
Preparamos um objeto com todas as características desejadas e então arrastamos e soltamos sobre o Prefab criado.
Então ele fica de cor azul claro, mostrando que agora ele tem vida.
Agora, ao selecionar o prefab e observar o Inspector veremos todas as informações do objeto que serviu de modelo para o prefab, inclusive um preview no Inspector.
Realmente temos uma cópia do objeto, tanto que podemos até excluir o objeto original e passar a usar somente o prefab.
Tomar cuidado ao criar frebab e manter o objeto original, pois são independentes e alteraçãoes em um não refletem no outro.

Duplicando o Prefab
Quando duplicamos o prefab e arrastamos uma ou mais cópias para a cena. Observe que agora as cópias têm no Inspector os botões Apply e Reverty.
Agora, após fazer qualquer alteração em uma cópia ou no prepaf original precisamos selecionar a cópia alterada e clicar em Apply para que sincronize as alteraçaões nos demais, inclusive no prefab.

Algumas alterações como move e rotate não são sincronizadas. Estas alterações surtem efeito no prefab mas nas demais cópias, não.

Para evitar problemas, somente duplique um prefab após realizar todas as alteraçãoes no mesmo.
Caso tenha que alterar após duplicar, idealmente remova as cópias e duplique novamente o prefab.

Lembrando que nos scripts, no código, um prefab é um gameobject como outro qualquer.

Order in Layer
Observar as camadas dos objetos na cena. Cada sprite tem o componente Sprite Renderer com a propriedade Order in Layer

Material
Ajuda a mudar a aparência de modelos 3D, desde apenas uma cor básica até com imagens de superfície (materiais).

Herança
Um objeto filho herdará movimento, rotação e escala do seu pai. Quando arrastamos um objeto na hierarquia e soltamos sobre outro, este que arrastamos transforma-se em filho do outro.
Objetos filhos ficam “dentro” dos objetos pais, à sua direita e abaixo na hierarquia.

Padronização
Uma boa forma de trabalhar e que nos ajuda é adicionar um sufixo do tipo ao nome dos objetos:

Objetos – apenas com seu nome: player, terrain, sky, etc
Scripts: playerScript, enemyScript, etc
Prefabs: playerPrefab, enemyPrefab, etc
E assim por diante.

Compilação/Publicação
Podemos publicar o projeto em Windows, OS X e na web com o Unit Web Player, que pode ser baixado daqui:
http://unity3d.com/webplayer

Também pode publicar em:
iOS, Android, Wii, XBOX 360, Playstation 3 e uma versão Flash via Web Player.

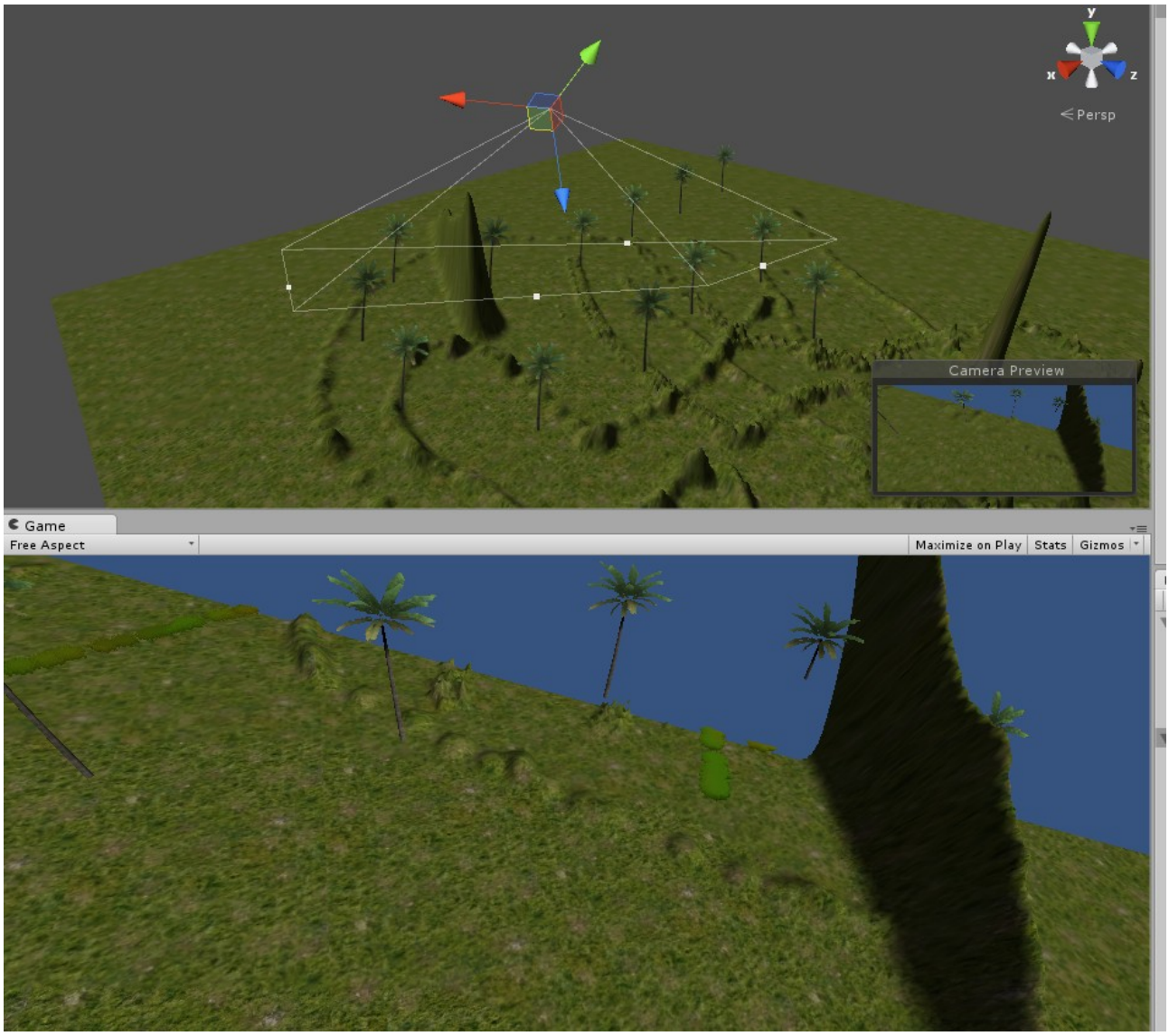
Para publicar pelo Web Player:
File – Build & Run
Selecione Web Player na lista
Clique em Build and Run

Câmeras

A main camera é um instrumento do jogo que delimita a área da cena que aparecerá para o jogador. Somente aparecerá o que a main camera tiver definido.

Câmeras – servem para representar o comportamento do campo de visão de quem vê o jogo em sistemas em 3 dimensões. O campo de visão é uma pirâmide invertida.

A região abarcada pela Main Camera acima será mostrada no painel Game abaixo e somente esta região:

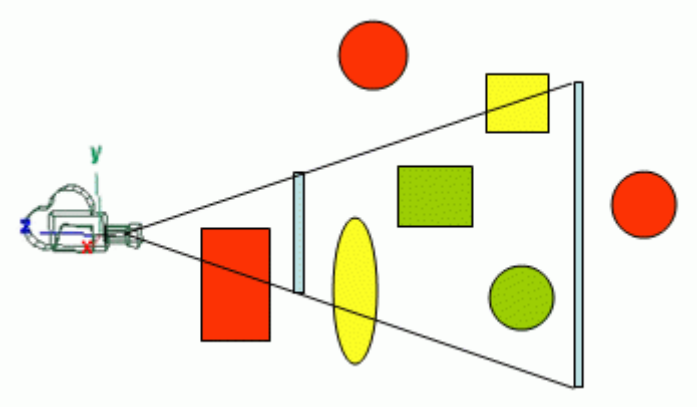


Main Camera – sempre que criamos um jogo novo o Unity cria por default uma Main Camera.

Câmeras – elas servem para limitar o campo de visão do jogo e de como o jogo aparecerá para o jogador. Precisamos regular sua posição e ângulo para uma exibição adequada do jogo. Dependendo de sua posição parte ou todo o jogo ficará fora da área de visão do jogador.

Câmeras em jogos servem para representar o comportamento dos olhos em um sistema em três dimensões. Através dela são determinados quais objetos devem aparecer na tela e como eles devem ser mostrados. Elas podem ser posicionadas em qualquer lugar no espaço, possuem uma direção e um campo

de visão, ou *Field of Vision* (FoV). Esse campo de visão pode ser interpretado como uma pirâmide invertida com origem no ponto de origem da câmera e com direção determinada pela direção da câmera, assim, os pontos mostrados na tela são os que estão dentro dessa pirâmide.
Quando limitamos a distância do campo de visão, estamos especificando uma distância máxima que um objeto deve estar do ponto de origem da câmera para que ele seja mostrado na tela, desde que ele esteja dentro do campo de visão da câmera. Também é possível especificar uma distância mínima que os objetos deverão estar da câmera para que eles sejam mostrados na tela.



Os objetos vermelhos não serão renderizados, não serão vistos pelo jogador.

Cliping Planes (região da cena)
Com a Main Camera selecionada, no Inspector, componente Camera, em Clipping Planes reduzir para por exemplo 20, reduz a área de atuação da câmera para uma área mais adequada para o jogo. Depois disso o ponto (-6,5 z) fica no canto superior esquerdo da tela. Especifica uma posição inicial para um jogador.

Modo de Execução/Play
Podemos fazer alterações neste modo. Elas podem ser muito úteis mas elas são temporárias. Logo que saimos deste modo as alterações se perdem.

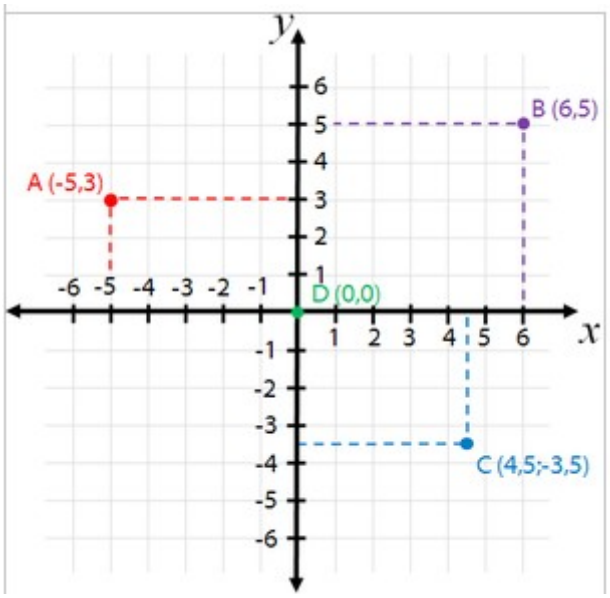
Reset
A qualquer momento podemos fazer um componente voltar a seu estado original. Para isso clicamos na pequena roldana à direita do nome do componente e clicamos em Reset.

Plano Cartesiano

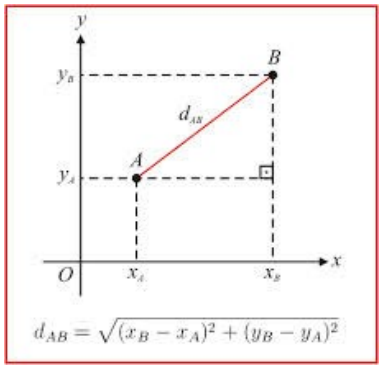
O plano cartesiano é uma base para representar pontos, planos e sólidos no espaço.

2D

Temos o plano cartesiano no plano 2D, representado por dois eixos, que chamamos x e y.

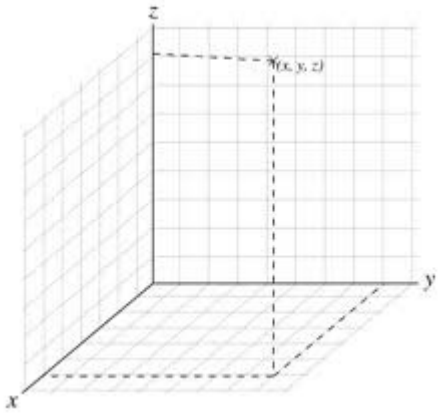


Com ele representamos pontos, retas e gráficos de funções.

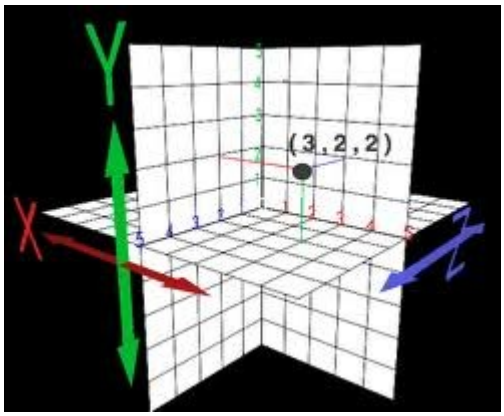


3D

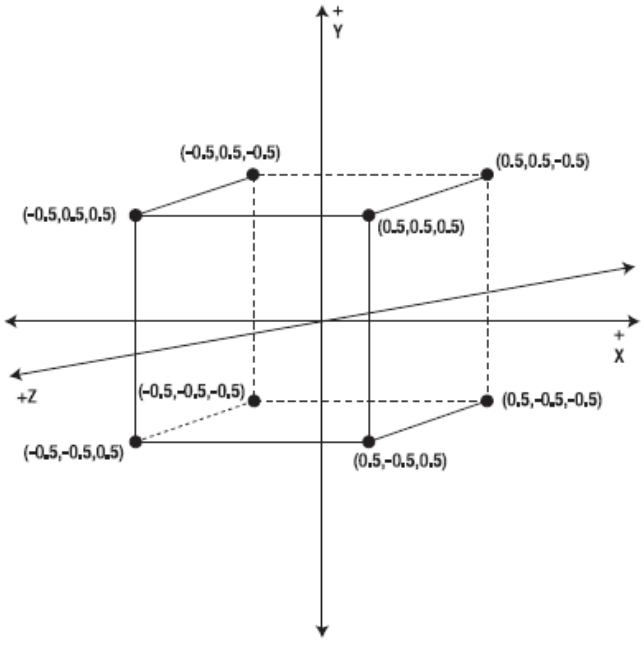
E temos o plano cartesiano para o 3D, que é representado pelo eixos x, y e z.



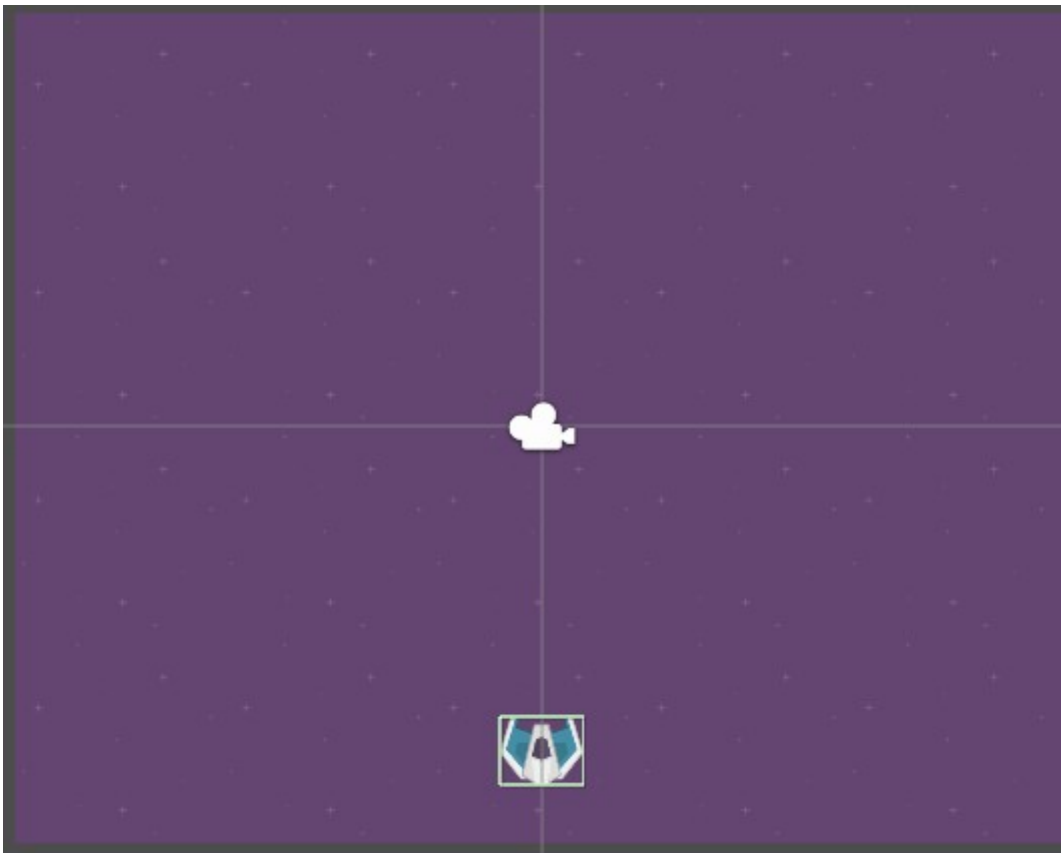
Acima vemos a representação de um ponto no plano 3D.



Origem (encontro dos 3 eixos)

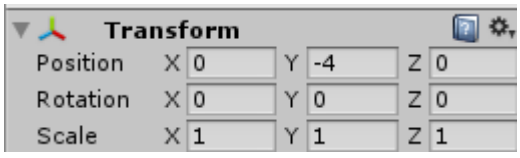


O encontro dos eixos é chamado de origem e está no ponto (0,0) para o 2D e (0,0,0) para o 3D.
Cena



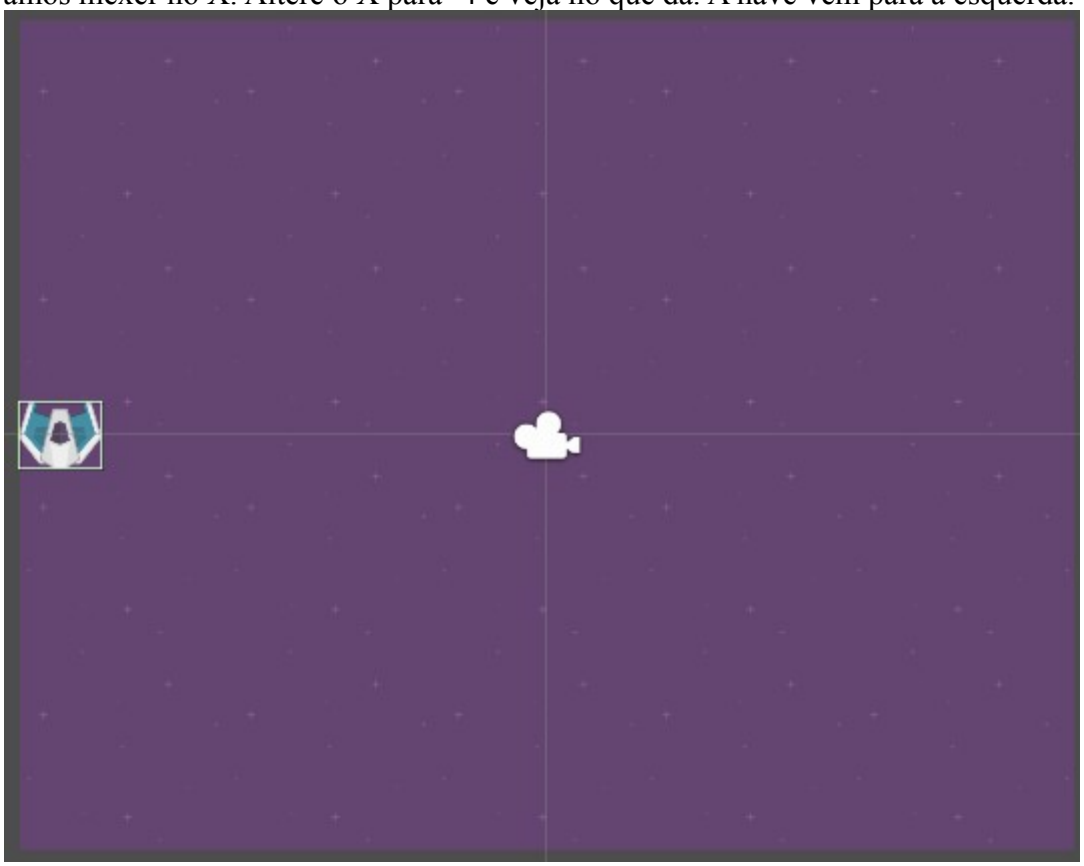
O traço branco representa o plano cartesiano. A origem do plano cartesiano, representada pelo ponto 0,0, é onde está a câmera. Como nossa nave está abaixo, então o Y é negativo.

Inspector

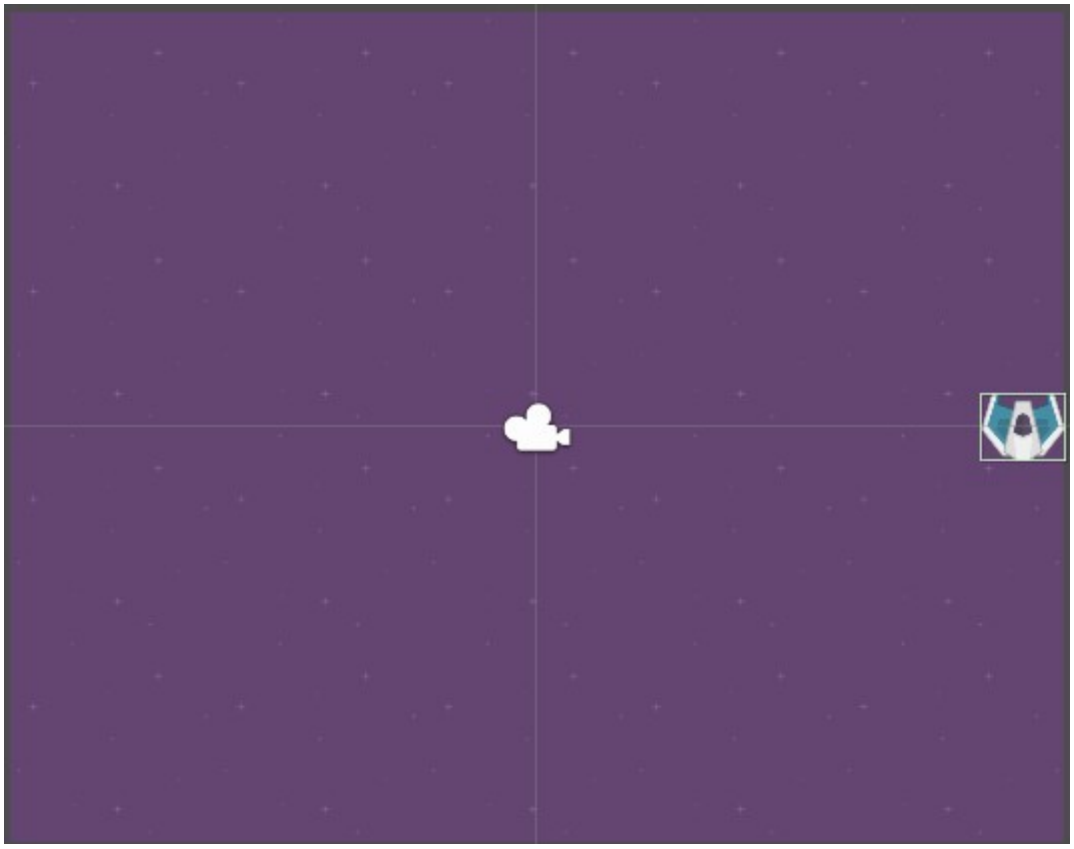


Mude no Inspector o Y da nave para 4 e veja o que acontece.
A nave vai para cima.

Agora vamos mexer no X. Altere o X para -4 e veja no que dá. A nave vem para a esquerda. (-6,0,0)



Agora mude X para 4. A nave vai para a direita. (6,0,0)



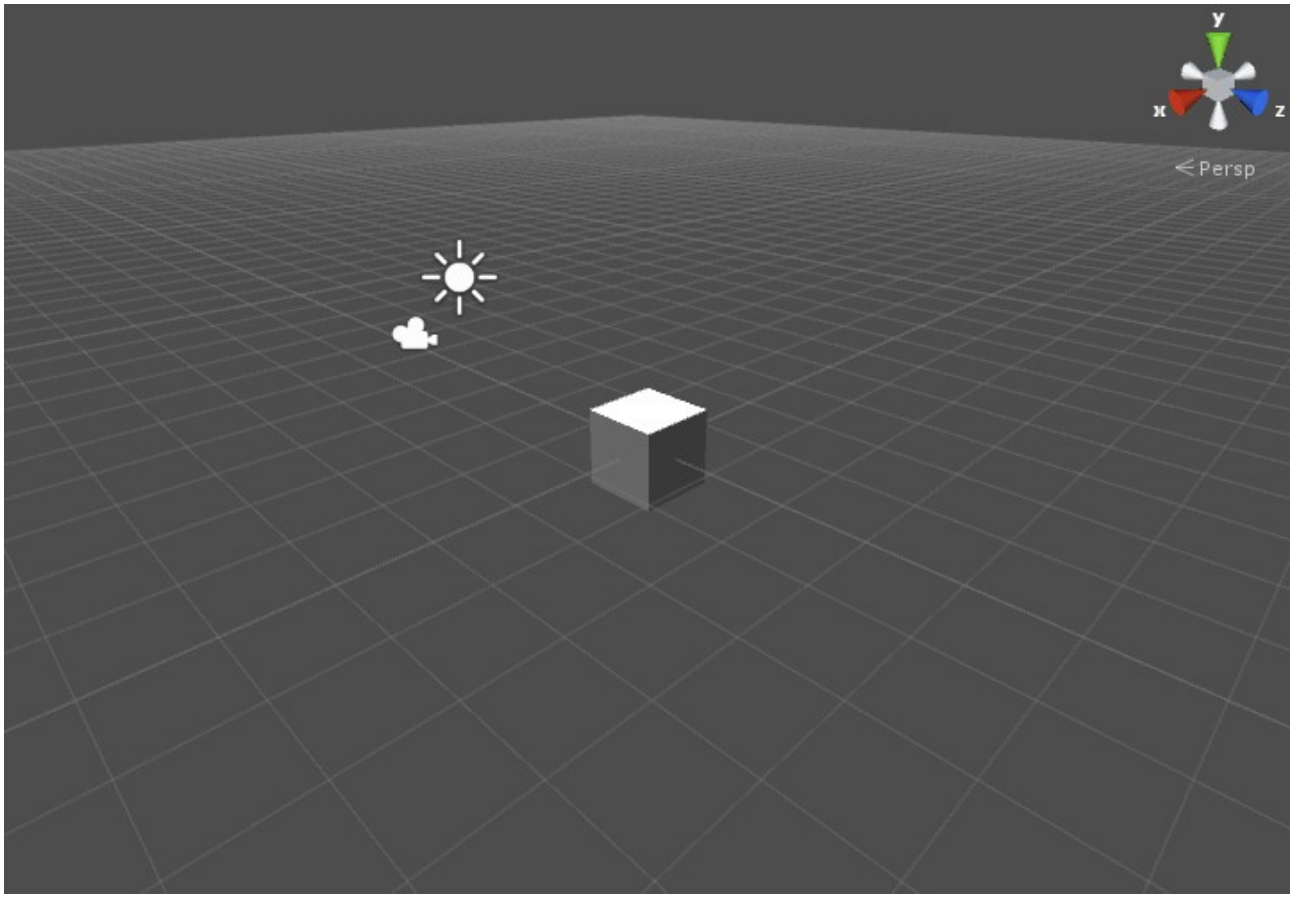
Finalmente mude para (0,0,0)

A nave ficará no centro da tela.

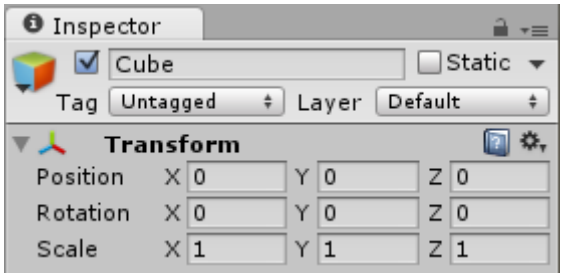
Brinque bastante com estes valores para se acostumar.
Lembre que com 2D o Z praticamente não conta.

Vamos agora abrir o Unity num jogo 3D para analisar o plano cartesiano.

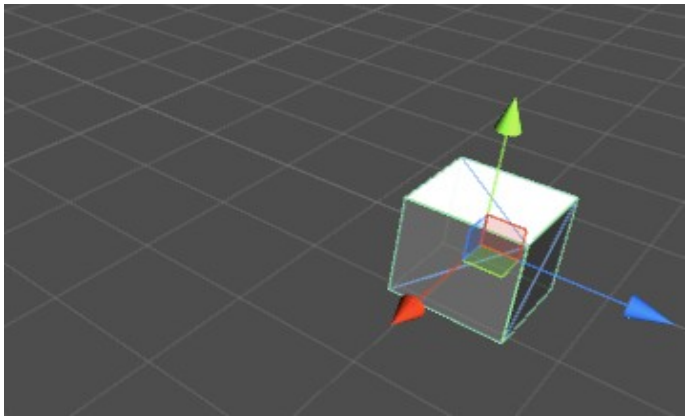
Veja a cena do Unity para 3D



Um cubo adicionado.
Inspector



Os eixos seguem os guizmos coloridos: x vermelho, y verde e z azul.



Vamos alterar as coordenadas da posição do Cubo para ver como se comporta. Lembrando que a posição é formada de
(X,Y,Z), x e z na horizontal e y na vertical

(5,0,0) – Assim o cubo vem para a esquerda e aumenta de tamanho. Quando mais próximo maior.

(-5,0,0) – Vai para o fundo e reduz seu tamanho, visto que estamos usando uma projeção em Perspectiva.

(0,4,0) – Sobe

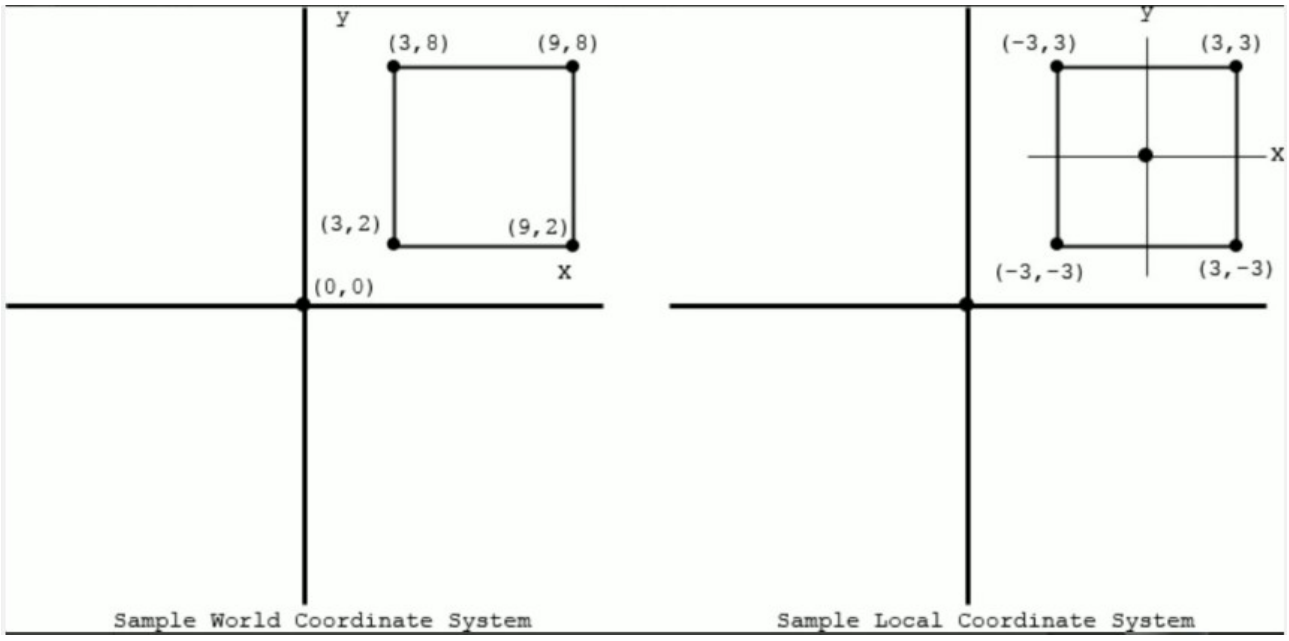
(0,-4,0) - Desce

(0,0,4) – Eixo Z para a frente

(0,0,-4) - Eixo Z para traz

Coordenadas Local/Global

Local - coordenadas relativas ao GameObject
Global - coordenadas relativas ao fundo, à cena



Global Local

Vetor – é um segmento de reta que sai da origem até um certo ponto. No Unity temos o Vector2 e o Vector3. Vetores 3D são linhas desenhadas num mundo 3D que tem uma direção e um comprimento.

Posição de Objeto no Espaço e Quaternion
Para representar a posição de um objeto no espaço precisamos de sua posição e seu ângulo no espaço.

Materiais – são como materiais do mundo real que usamos para melhorar a aparência dos objetos. São aplicados sobre os GameObjects. Eles recebem cores, texturas, etc..

Aplicando Material a Objeto

Crie um cubo
Em Assets crie um material – Botão direito – Create – Material
Altere a cor e outras propriedades
Arraste o material criado e solte sobre o objeto na cena.

Com o cubo selecionado podemos alterar as propriedades no Inspector em Materials.
Pivot – é o centro de gravidade de um ou mais objetos 3D (quando todos selecionados). Cada objeto também tem seu ponto Pivot, que é usado para fazer as alterações do Transform.

Física
São simuladores das leis da física que podem ser aplicados aos objetos como o rigdbody (corpo rígido, com massa, gravidade, rotação, velocidade), são os de colisão, etc.

Um **Collider** define uma região ou superfície que será sensível e acusará o contato com outro objeto. Um bom exemplo é adicionar ao jogador para que colida com o chão. Ao aplicarmos um collider a um objeto estamos adicionando uma rede/malha ao redor do mesmo, cuja finalidade é detectar a colisão com outros objetos.

No componente **RigdBody** ao marcar **Is Kinematic** faz com que o objeto não seja controlado pelo controlador da física mas sim pelos mecanismos da animação ou pelo código do script. Existem outras propriedades no Rigidbody, como Mass, Gravity, etc.

Is Trigger – informa se a colisão de um objeto será ativada por um gatilho ou não.

Estes componentes da física contam com eventos respectivos como OnCollisionEnter, OnTriggerEnter, etc. Estes eventos são funções que executam seu conteúdo de acordo com a configuração do componente.

Botão Local/Global – as coordenadas do Local são relativas ao objeto e as Globais são relativas ao plano base, à cena.

Primitivas Geométricas

O Unity vem com algumas primitivas geométricas:
GameObject – Create Other:
Cube, Sphere, Capsula, Cilynder, Plane, Quad

Importando Pacotes/Packages
Assets – botão direito – Import Packages:
Custom Package, Character Controller, Terrain, Skybox, etc.

Esses pacotes já vem com muita coisa pronta para usar: sprites, texturas, materiais, scripts, prefabs, etc.

Adicionar o Character Controller (First Person)

Precisamos adicionar, mover ele de forma que fique nivelado com o terrano ou a base da fase, deve ficar sobre o plano de fundo.

Ao clicar em Play teremos como andar pela cena com o mouse e as setas do teclado.

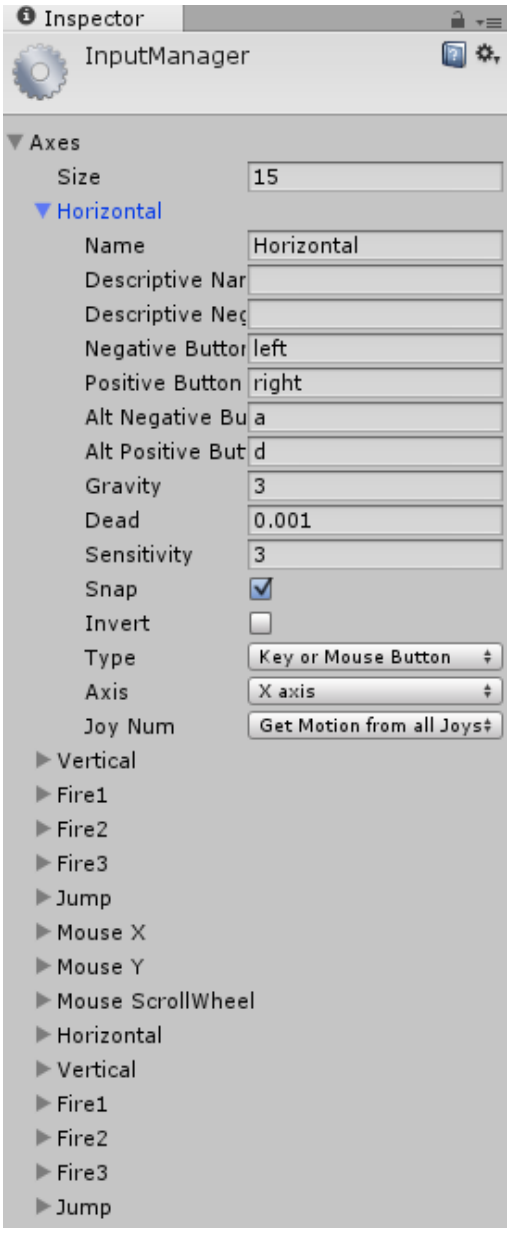
Podemos excluir a Main Camera existente, pois o controle traz uma.

Quando importamos objetos, materiais, precisamos ativar o mesh collider para que colida com o personagem.

Input Manager
Onde podemos configurar as entradas dos usuários:
Edit – Project Settings – Input – Axes

Detalhes:
http://docs.unity3d.com/Manual/class-InputManager.html

As entradas do usuário através de teclado, mouse, toque, etc são definidas no Inpu Manager, oude também podemos criar outras ou alterar as existentes. Use com cautela.
Edit – Project Settings – Input



Game Object Empty – uma dica é criar um game object Empty apenas para agrupar outros objetos, organizando assim os objetos do jogo na Hierarquia.
Mudando a posição do objeto pai (mais externo) também altera a posição dos objetos filhos.
Objetos vazios são puramente lógicos.

Animação Simples
-Selecione 2 ou mais sprites e arraste para a cena
-Salvar e dar um nome para a mesma.
-Serão criados dois arquivos, um AnimatorController e um AnimationClip, que serão gravados no mesmo diretório.

Editando Região de Colliders
Aumentar o zoom até que perceba o traço do collider na cor verde.
Para ajustar suas dimensões arrastando segure o Shift. Então aparecem pequenos pontos que poderá arrastar para redimensionar.

Para remover traços segure a tecla Ctrl e clique no traço que aparece em vermelho e deseja remover.

Importando um projeto completo da Loja do Unity/Asset Store

- Crie um novo projeto do tipo 2D e dê um nome parecido com o que irá importar
- Window - Asset Store
- Em Categories - Complete Projects – Tutorials
Unity Projects: 2D Platform
Download
Faça login, caso seja a primeira vez
Aguarde o download
Clique em Import e aguarde
Novamente clique em Import à esquerda e aguarde
Após concluir feche a janela do Asset Store
Clique abaixo em Scenes e à direita em Level
Clique na seta acima para executar o jogo.
Salve o projeto.