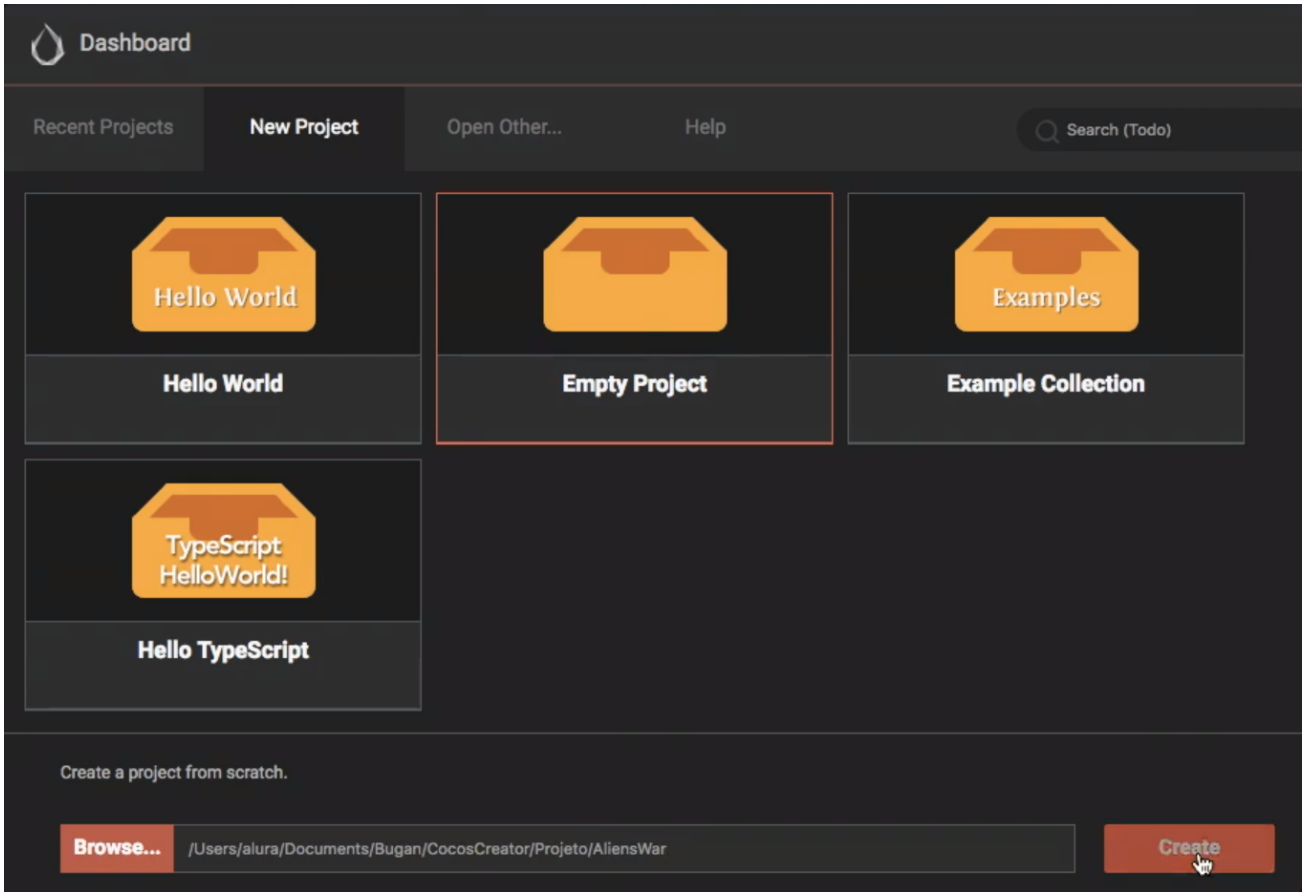
03

# Criando o projeto e adicionando o movimento

### Transcrição

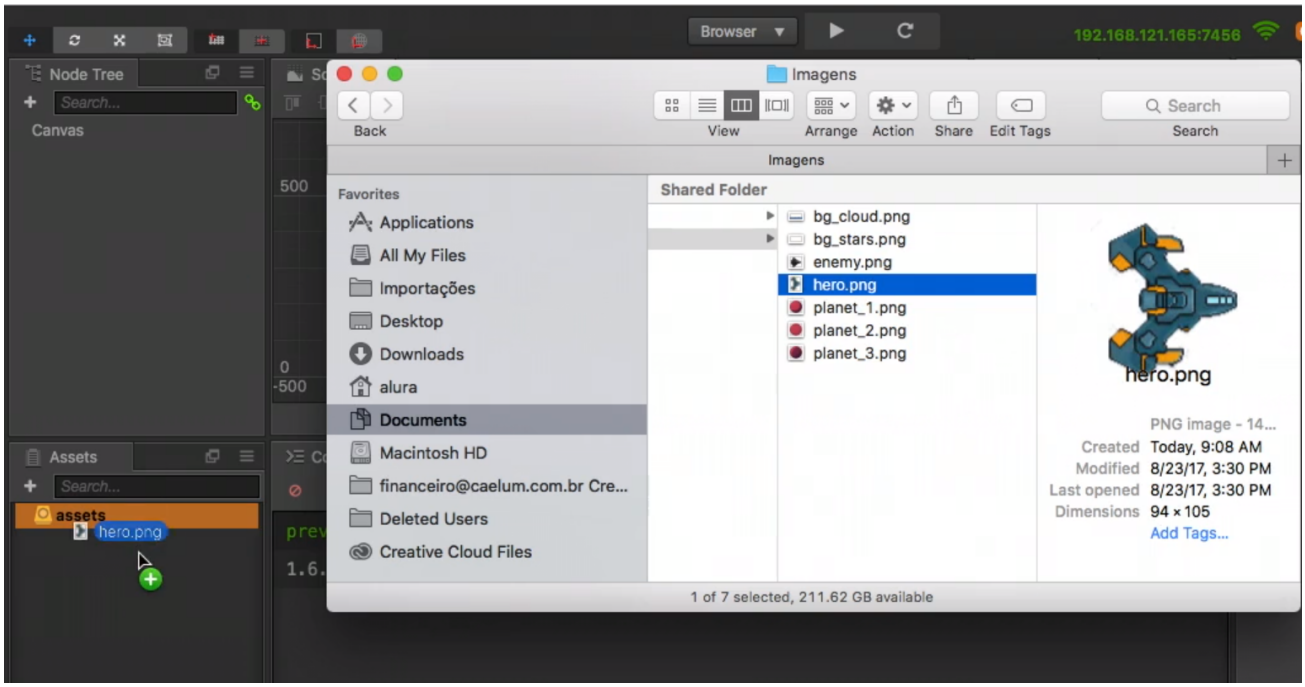
Para darmos início ao desenvolvimento do nosso jogo, abriremos a *Cocos Creator* e criaremos um projeto clicando nas opções "New Project". Como modelo de projeto, selecionaremos a opção `Empty Project` para que seja criado um projeto do zero e na barra `Browser` escolheremos o nome do projeto precedido do diretório onde o mesmo será criado. O nome do projeto será o mesmo do jogo, que como vimos, se chama `AliensWar` .




## Importando os Assets

Um dos primeiros passos que precisamos dar na criação do jogo é a importação dos **assets**. Eles são os arquivos que serão utilizados dentro do jogo, mas que são criados externamente. Isso inclui imagens, vídeos, sons, etc.

Tudo que precisamos fazer para importar um *asset* dentro da *Cocos* é selecionar o arquivo no sistema de arquivos do sistema operacional e arrastar para a área *Assets* dentro da *Cocos*.




Os arquivos a serem importados por enquanto são as imagens `hero.png`, que será nosso personagem principal, e o `enemy.png`, que será o inimigo.

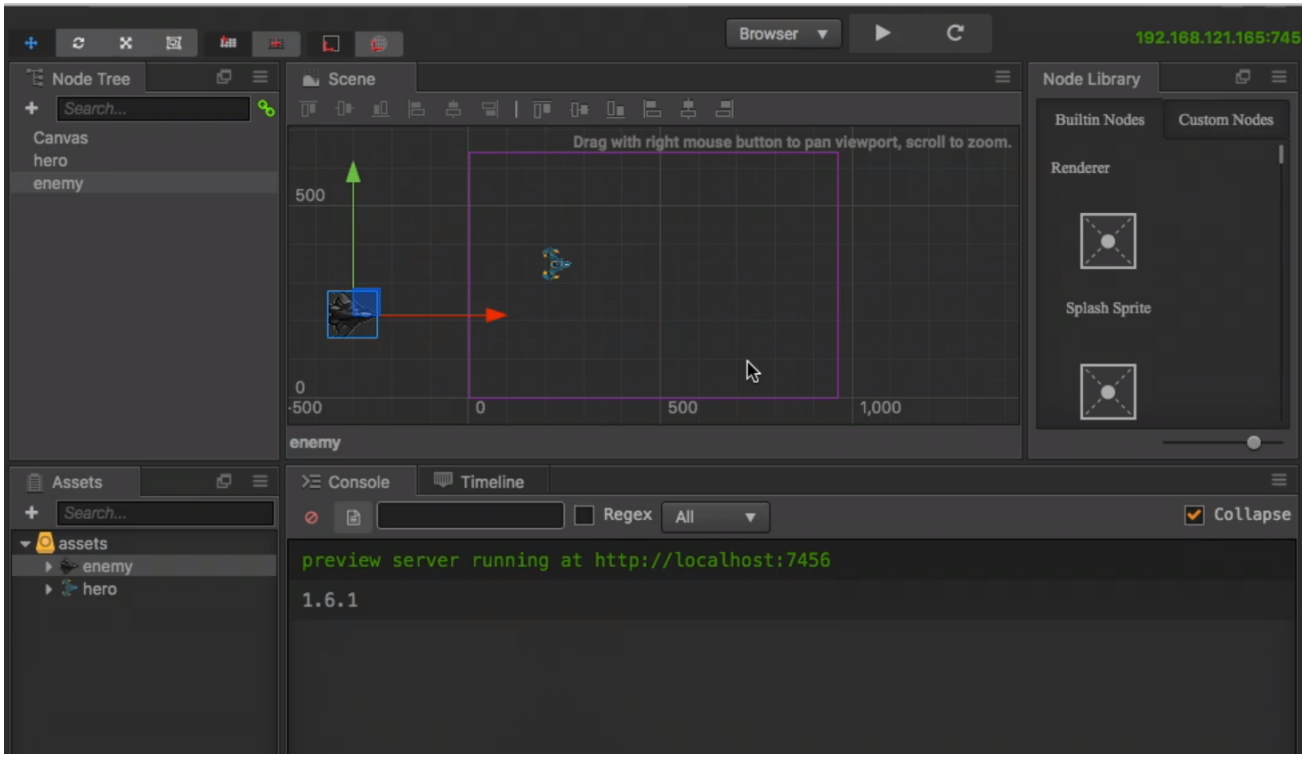
> Você pode fazer o download dos [dois arquivos (https://github.com/alura-cursos/cocos/tree/69c40dfcadf90901f7801f296954f84b98d84ad6/assets)](https://github.com/alura-cursos/cocos/tree/69c40dfcadf90901f7801f296954f84b98d84ad6/assets).

## Adicionando personagens à cena

Temos os dois personagens em nosso repositório de "assets", podemos incluí-los no jogo. Faremos o mesmo processo de importação, selecionaremos a imagem na aba "Assets*", clicar e mover a mesma para a aba "Scene".

Precisamos considerar um detalhe. Note que na aba "Scene", temos um quadro roxo. Ele é o que chamamos de câmera. Os itens que estão dentro da câmera, aparecem no jogo, os que estão fora, não aparecem.

Podemos comparar a cena com o "*universo do jogo*", tudo que o jogo precisa para funcionar, deve estar dentro dela. Assim adicionamos tanto o `hero` quanto o `enemy` à cena, visto que precisamos dos dois no jogo.




Note que a *Cocos* cria um quadro azul no canto direito superior de cada imagem, ele receberá o nome de "Gismo". Com ele, podemos arrastar os objetos pela cena e posicioná-lo como quisermos. Com os personagens na área da câmera, podemos clicar no botão de *play* para vermos o jogo funcionando.

> Dica: Lembre-se de fazer isso a cada passo, assim é verificado o status do desenvolvimento do seu jogo e problemas são detectados com antecedência.

## Movendo o jogador

Estamos com os personagens na cena mas eles ainda não fazem nada. Como fazemos o jogador ( `hero` ) se mover? Com JavaScript!

Na aba "assets", clicaremos com o botão direito e usaremos as opções `Create -> JavaScript` . Renomearemos o arquivo para `Jogador` e o abriremos em um editor de código.

O código inicial do arquivo `Jogador.js` é o seguinte:

```
cc.Class({
    extends: cc.Component,

    properties: {
        // foo: {
        //    default: null,      // The default value will be used only when the component att
        //                           to a node for the first time
        //    url: cc.Texture2D,  // optional, default is typeof default
        //    serializable: true, // optional, default is true
        //    visible: true,      // optional, default is true
        //    displayName: 'Foo', // optional
        //    readonly: false,    // optional, default is false
        // },
        // ...
    },
```

```
    // use this for initialization
    onLoad: function () {

    },

    // called every frame, uncomment this function to activate update callback
    // update: function (dt) {

    // },
});
```

A ideia é que movamos o `hero` para a direita, ou seja, no eixo X. Para isso, primeiro, removeremos os comentários do método `update` e adicionaremos a ele o seguinte código:

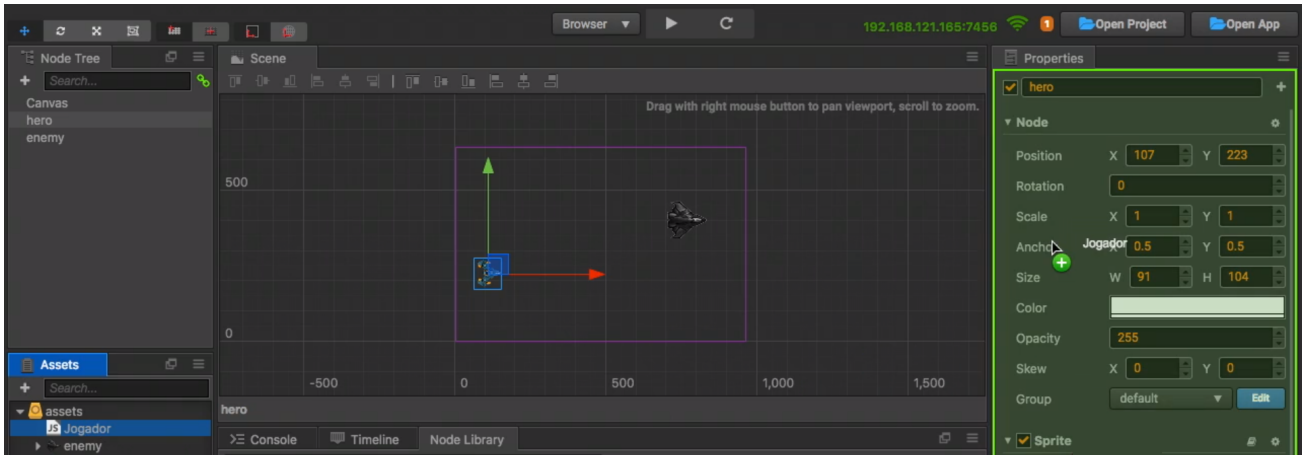
```
this.node.x += 1;
```

O código acima adiciona `1px` na posição X do objeto. O `this` referencia o próprio objeto ( `hero` ), o `node` é uma propriedade herdada da classe `Component` e o `x` , provavelmente você já deve imaginar: é a posição X do objeto.

Podemos clicar no *play* para testar o jogo novamente, mas veremos que nada acontecerá. O jogador não se move no eixo X.

O que precisamos fazer é anexar o *script* `Jogador.js` ao `hero` . Da forma que está, a *Cocos* não sabe como estes elementos se relacionam. Para isso vamos clicar sobre o *script* e arrastá-lo para a aba `Properties` do objeto `hero` .

Ou seja, primeiro clicaremos no jogador, depois clicamos e arrastamos o *script* para a aba indicada.




Podemos dar *play* novamente e agora sim, veremos nosso jogador se movendo para a direita.

## Por baixo dos panos

Nosso personagem se move e tudo está bem, mas por curiosidade, o que está de fato acontecendo? Quem está executando o método `update` ?

O que acontece por baixo dos panos é que a *Cocos* quando faz a leitura da cena, varre todos os objetos presentes nela executando o método `update` de cada objeto a cada **Frame**. O que é *Frame*? Trata-se de um intervalo de tempo.

Ou seja, o nosso personagem se move porque a *Cocos* varre os objetos da cena de tempos em tempos e chama o método `update`. Este método no script `Jogador.js`, adiciona o valor `1` ao eixo X do objeto. Como isso acontece de forma contínua, temos a impressão de movimento.

Com este passo pronto, podemos salvar. Mas temos uma observação: o que estamos salvando de fato ao usar o `CTRL + S` é a cena. Essa primeira chamaremos apenas de `Jogo`. Será nela que faremos toda a parte de interação no jogo.