01

Criando disparos

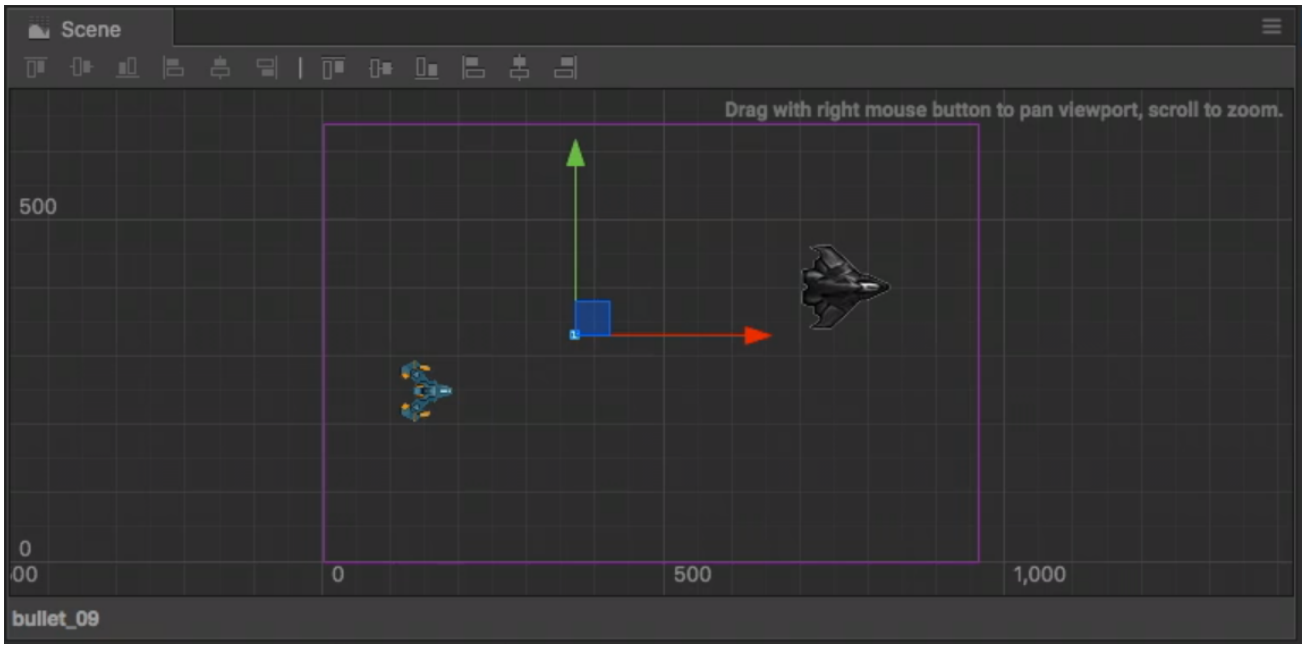
Transcrição

Até o momento trabalhamos na movimentação da nossa nave, mas o jogo não é composto só por isso. Ainda temos que trabalhar com os inimigos e com os tiros das naves. Vamos começar primeiramente com o tiro da nave do jogador. Primeiro, importamos a imagem que representa o tiro da nave, a `bullet_09.png` disponível neste link (https://github.com/alura-cursos/cocos/raw/master/assets/bullet_09.png).

Lembre-se, importamos movendo a imagem para a aba `Assets` e logo depois, movemos de `Assets` para a aba `Scene`, assim ela estará disponível em nosso universo. O tiro precisa de um comportamento atrelado a ele, o de mover-se na direção do mouse, isso por que o clique do mouse será o que vai fazer a nave atirar. Já sabemos como movimentar objetos na cena certo? Para lembrar: Subtração e soma de vetores seguida de uma normalização.

Movimentando o tiro

Posicionamos a imagem do tiro no meio da cena e logo em seguida, na aba `Assets` criamos um novo *script* chamado `Tiro.js` com o `clique direito -> Create -> JavaScript`.



Nas propriedades do `Tiro.js`, já vamos deixar criada a propriedade `direcao` que será o vetor resultante dos demais cálculos. Ele será somado ao vetor posição do objeto. Ainda não temos os cálculos prontos, mas vamos deixar o que sabemos que precisa ser feito, pré-codificado.

```
properties: {
    // foo: {
    //    default: null,      // The default value will be used only when the component attachi
    //                           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
```

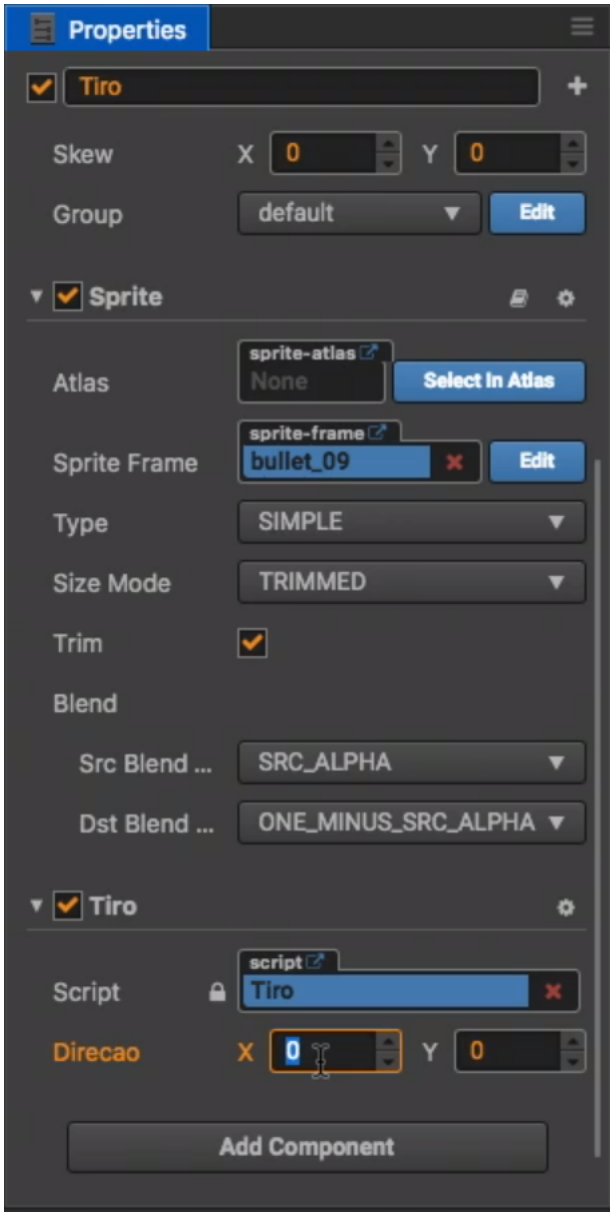
```
    //    readonly: false,    // optional, default is false
    // },
    // ...
    direcao: cc.Vec2,
},
```

E no método `update` já vamos deixar pronta a soma do vetor `direcao` com o vetor `position`.

```
update: function (dt) {
    this.node.position = this.node.position.add(this.direcao);
},
```

Lembre-se de mover o *asset* `Tiro.js` para a aba propriedades do objeto `bullet_09`, mas antes disso, renomeie o nome do objeto na aba `Node Tree` de `bullet_09` para `Tiro`. Fica mais claro identificar o objeto dessa forma. Depois de renomear e com o objeto `Tiro` selecionado, mova o *script* `Tiro.js` para a aba propriedades do mesmo.

Note que como criamos uma propriedade no código do *script*, a Cocos já a detecta e exibe uma nova sessão dentro da aba propriedades.



Agora precisamos pensar em uma característica do tiro: Sua velocidade. A velocidade do tiro não varia, entretanto, ela também é muito maior do que a velocidade da nave, então, não podemos depender da normalização do vetor direção. Isso quer dizer que precisaremos de uma nova propriedade para controlar isso. Chamaremos esta de `velocidade` e inicialmente diremos que seu valor é `10`.

```
properties: {
    // foo: {
    //    default: null,      // The default value will be used only when the component attachi
    //                           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
    // },
    // ...
    direcao: cc.Vec2,
    velocidade: 10,
},
```

Até aqui tudo bem, temos a direção e a velocidade, mas a direção precisa ser normalizada independente de usarmos seu tamanho ou não. Lembrando: A movimentação do tiro será realizada pela propriedade `velocidade` mas nós ainda precisamos da direção para saber para onde o tiro deve ir.

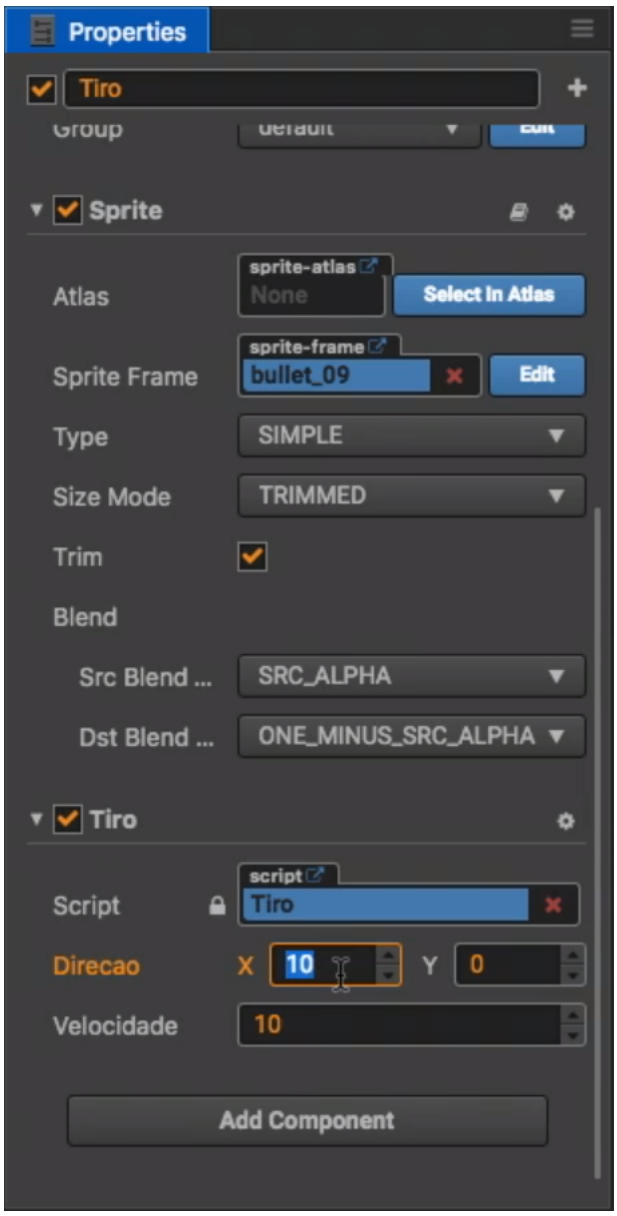
Outra observação que precisamos fazer é que a direção está como uma propriedade pública. O motivo para isso é que a direção depende da posição da nave. A nave é quem informa a direção para onde o tiro é disparado. Normalizamos a direção no método `onLoad`.

```
// use this for initialization
onLoad: function () {
    this.direcao = this.direcao.normalize();
},
```

Agora precisamos fazer com que no método `update` a velocidade do tiro seja considerada. Para isso, criaremos um novo vetor chamado `deslocamento` que será resultado da multiplicação dos vetores `direcao` e `velocidade`. E este vetor `deslocamento` será somado ao vetor `position` do objeto `Tiro`.

```
update: function (dt) {
    let deslocamento = this.direcao.mul(this.velocidade);
    this.node.position = this.node.position.add(deslocamento);
},
```

Note que deixamos também a propriedade `velocidade` como pública pois, dessa forma, podemos controlar seu valor diretamente pelo editor da Cocos.



Podemos testar e verificar que o tiro é movimentado de forma rápida do jogo. Nosso próximo passo é fazer com que seja possível disparar várias vezes, ou seja, criar várias instâncias do tiro.