06

Delta time

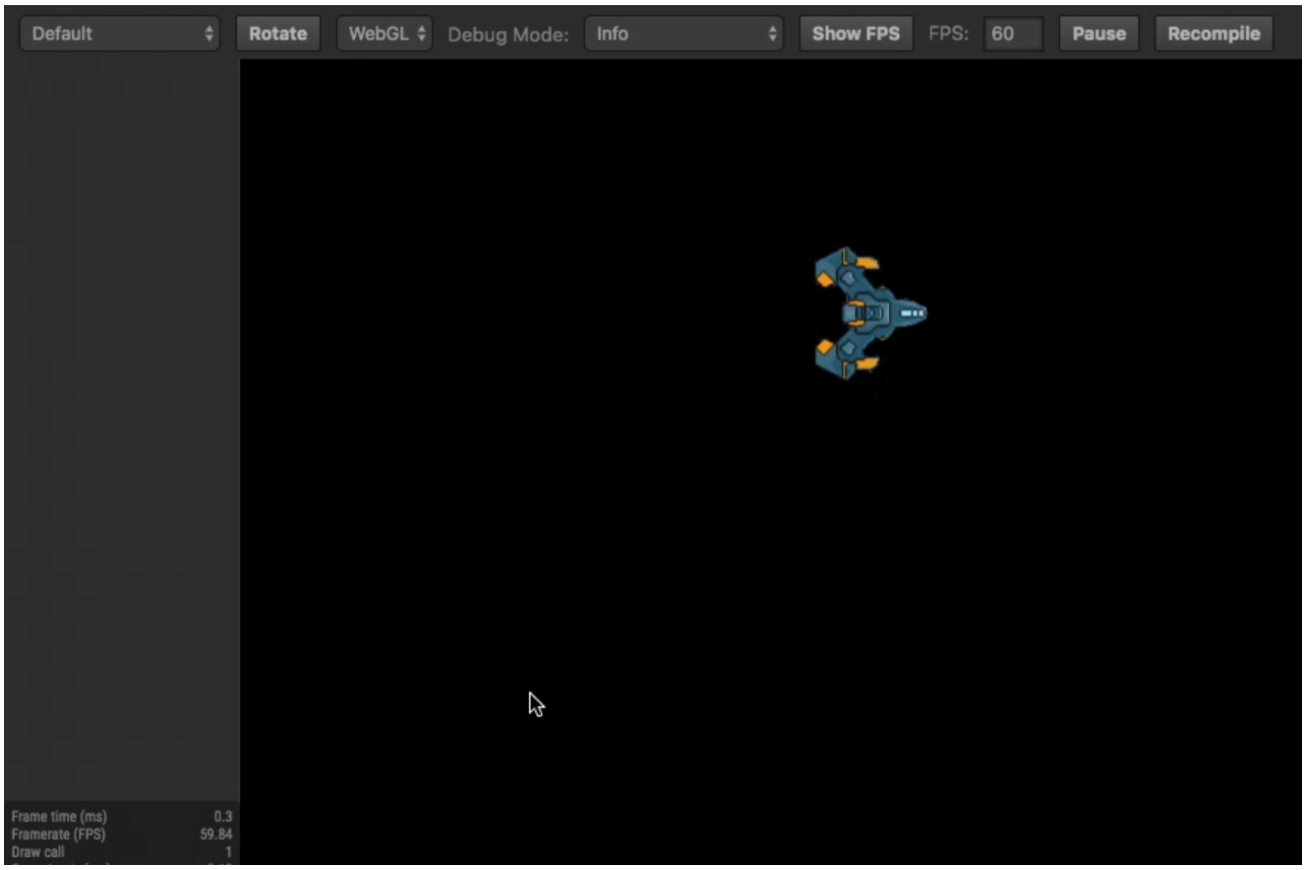
Transcrição

Estamos muito perto de ter tudo funcionando como apresentado no inicio do curso. Já podemos disparar tiros e destruir nosso inimigo. O problema que vamos atacar agora é a criação dos inimigos de forma dinâmica, afinal, não terá graça destruir apenas um inimigo que não atira nem faz nada.

Porém, antes de chegarmos a este ponto, precisamos estar atentos a um detalhe. A questão é: A velocidade do tiro apesar de estar bem rápida, depende de um outro fator. O **FPS** (*frames per second*), quadros por segundo no português. Se observar bem, quando estamos executando o jogo, na parte superior da tela temos algumas opções de configuração, entre elas o FPS que está com o valor de `60`, ou seja, 60 quadros por segundo.

Note que se diminuirmos esse valor para 20, por exemplo, tanto a velocidade do tiro, quando da nave serão reduzidos. Isso é um problema. Pode não parecer, mas no mundo ideal, parece que o jogo está sendo executando exatamente nesse ritmo de 60 quadros por segundo. Mas no mundo real, esse não é bem o cenário. As velocidades variam de acordo com o poder de processamento.

Podemos observar que, apesar de o jogo estar configurado para funcionar a 60 FPS, o valor real varia bastante, ele fica oscilando. É possível verificar isso no canto inferior esquerdo da tela enquanto o jogo está em execução.



Nosso jogo atualmente considera o FPS em todas as movimentações, isso quer dizer que, apesar de termos todo o cuidado de normalizar os vetores para termos velocidades de deslocamento contantes, isso na realidade não acontece de fato. Estamos sempre com variações. A ideia é que agora possamos pensar em uma forma de manter a velocidade dos movimentos mesmo com variações no FPS.

Vejamos o método `update` da classe no *script* `Jogador.js`.

```
// called every frame, uncomment this function to activate update callback
update: function (dt) {
    if(this._acelerando){
        this.node.position = this.node.position.add(this._direcao);
    }
},
```

O problema que explicamos acontece justamente por que o método `update` é chamado a cada *frame*, fazendo com que todos os demais movimentos dependam do *frame* e não do tempo que é contínuo. Lembre-se de que o tempo não varia, mas sim a quantidade de *frames* executados dentro do intervalo.

A solução apesar de simples, em código, parece complexa. Veja que o método `update` recebe um parâmetro chamado `dt`, abreviação de *delta time*, em português, diferença de tempo. Em um FPS de 60, o valor do DT será algo em torno de 0,016 (1 / 60 *um segundo para 60 quadros, ~16 milésimos de segundo para cada um*). O que faremos é: criar um vetor `deslocamento` que seja o resultado da multiplicação de uma `velocidade` por um valor de tempo (`dt`) que ainda será multiplicado pelo vetor `direção`.

```
// called every frame, uncomment this function to activate update callback
update: function (dt) {
    if(this._acelerando){
        let deslocamento = this._direcao.mul(this.velocidade * dt);
        this.node.position = this.node.position.add(deslocamento);
    }
},
```

Lembrando que precisamos declarar a propriedade `velocidade` com algum valor. Neste caso, deixaremos como `200`, para testarmos.

```
properties: {
    _acelerando: false,
    _direcao: cc.Vec2,
    tiroPrefab: cc.Prefab,
    velocidade: 200,
},
```

Isso quer dizer que a nave agora se moverá a `200px` por segundo, e não por *frame*. Dessa forma podemos ter variações no *frame*, mas a velocidade de movimentação da nave já não oscilará por este motivo.

Com o problema da nave resolvido. Aplicamos a mesma solução para o tiro. O método `update` do objeto `Tiro` se encontrava assim:

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

Ficará assim:

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

Atualizando um Prefab

Caso prefira atualizar o *prefab* do tiro, mesmo sem o tiro na cena, temos que adicionar o *prefab* na cena, realizar os ajustes necessários na aba `Properties`, salvar estas atualizações no botão `Save` da mesma aba, para assim, removê-lo da cena novamente.

O valor da velocidade do tiro precisa ser ajustado para `600`, assim, o tiro se moverá a 600 pixels por segundo, em vez de 5, que é o valor atual.

Considerações

Ao reduzir o FPS, pode-se perceber alguns travamentos na movimentação dos objetos. Pequenos teletransportes. Isso acontece por que o FPS é quem atualiza a tela, como essa atualização varia, podemos ter esse tipo de comportamento. Mas vale lembrar que o movimento dos objetos não terá variação. Eles considerarão o tempo.