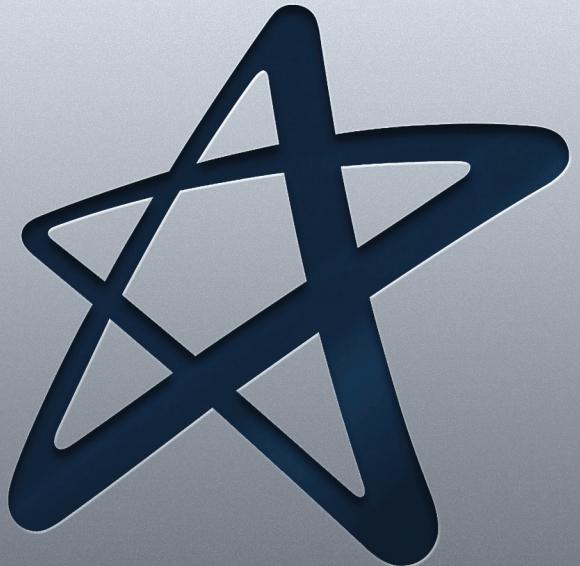


# Programação *Web* para Jogos



Cruzeiro do Sul Virtual  
Educação a distância



# Material Teórico



Desenvolvendo um Jogo 2D

**Responsável pelo Conteúdo:**

Prof. Me. Alcides Teixeira Barboza Junior

**Revisão Textual:**

Prof.<sup>a</sup> Dr.<sup>a</sup> Selma Aparecida Cesarin



# UNIDADE

## Desenvolvendo um Jogo 2D



- Ferramenta para Criar *Tileset/Tilemap*;
- Criando Cenários com Blocos (*Tileset* e Programação);
- Trabalhando com um Mapa Criado no *Tiled*;
- Mudando de Nível e Outros Recursos.



### OBJETIVO DE APRENDIZADO

- Apresentar os recursos do *Framework Phaser*;
- Desenvolver um Jogo 2D estilo Plataforma, utilizando o *Framework Phaser*.





# Orientações de estudo

Para que o conteúdo desta Disciplina seja bem aproveitado e haja maior aplicabilidade na sua formação acadêmica e atuação profissional, siga algumas recomendações básicas:

Determine um horário fixo para estudar.

Mantenha o foco! Evite se distrair com as redes sociais.

Procure manter contato com seus colegas e tutores para trocar ideias! Isso amplia a aprendizagem.

Seja original! Nunca plágie trabalhos.

Aproveite as indicações de Material Complementar.

Conserve seu material e local de estudos sempre organizados.

Não se esqueça de se alimentar e de se manter hidratado.

## Assim:

- ✓ Organize seus estudos de maneira que passem a fazer parte da sua rotina. Por exemplo, você poderá determinar um dia e horário fixos como seu “momento do estudo”;
- ✓ Procure se alimentar e se hidratar quando for estudar; lembre-se de que uma alimentação saudável pode proporcionar melhor aproveitamento do estudo;
- ✓ No material de cada Unidade, há leituras indicadas e, entre elas, artigos científicos, livros, vídeos e sites para aprofundar os conhecimentos adquiridos ao longo da Unidade. Além disso, você também encontrará sugestões de conteúdo extra no item **Material Complementar**, que ampliarão sua interpretação e auxiliarão no pleno entendimento dos temas abordados;
- ✓ Após o contato com o conteúdo proposto, participe dos debates mediados em fóruns de discussão, pois irão auxiliar a verificar o quanto você absorveu de conhecimento, além de propiciar o contato com seus colegas e tutores, o que se apresenta como rico espaço de troca de ideias e de aprendizagem.

# Contextualização

Você já deve ter jogado diversos Jogos de Plataforma. Entre eles, o mais conhecido é o *Super Marios Bros*.

Na Unidade anterior, fizemos um esboço desse gênero de Jogo para aprendermos os conceitos do *Phaser*. Chegou a hora de criarmos esse tipo de Jogo, mas um pouco mais avançado. Vamos seguir a mesma ideia da Unidade anterior, ou seja, ainda iremos trabalhar com um Jogo de Plataforma.

Como já vimos o básico, agora precisamos subir o nível.

Nesta Unidade, iremos conhecer como criar mapas e cenários a partir de *tilesets* usando ferramentas como *Tiled* e *Gimp* e programando tudo no *Phaser/JavaScript*.

Você irá aproveitar todos os conhecimentos adquiridos até este ponto e será capaz de desenvolver um Jogo de Plataforma 2D com mais de um nível, além de criar cenários em ferramentas que auxiliam o desenvolvimento de níveis.

**Vamos começar!**

# Introdução

Chegamos à nossa última Unidade.

Até este ponto, vimos como criar Jogos com *JavaScript* puro e *HTML* básico; com *JavaScript* puro e a *TagCanvas* e, por fim, como utilizamos um *framework* para nos auxiliar nessa tarefa de desenvolvimento de Jogos.

Nas Unidades anteriores, aprendemos o básico do *framework Phaser*; agora, chegou o momento de subirmos o nível do nosso conhecimento novamente, falando em nível, precisamos entender como criar um Jogo com diferentes níveis.

Temos mais de uma possibilidade para criar as cenas (telas) dos Jogos no *Phaser*. Na Unidade anterior, criamos um protótipo bem simples, no qual posicionamos os elementos do Jogo por meio da Programação. Na criação dessa cena, usamos as coordenadas do *Canvas* e determinamos os pixels de cada elemento.

Agora, imagine uma cena mais complexa do Jogo. Se seguirmos essa lógica, teríamos diversas linhas de código só para criar nossas cenas.

Então, você se pergunta, existe uma forma mais simples de criar as telas ou os níveis dos nossos Jogos no *Phaser*?

A resposta para sua pergunta é: sim, existe. E temos pelo menos duas formas diferentes de fazer isso fora a que já estudamos na Unidade anterior, sendo que nas duas formas que iremos estudar nesta unidade, iremos utilizar o conceito de *Tileset* e *Tilemap*.

Um *tilemap* é uma técnica que nos permite criar os mundos dos Jogos por meio de blocos. O princípio é dividir os elementos ou mundo do Jogo em blocos iguais na largura e na altura e deixar todos no mesmo arquivo de imagem. Com isso, ganhamos desempenho.

Ao iniciar nosso Jogo, carregamos esse mapa e informamos as dimensões de cada bloco. Um *tilemap* é criado por um conjunto de blocos que compõem um *tileset*.

Veja na Figura 1 um exemplo de *tileset* simples retirado do site OpenGameArt.org.

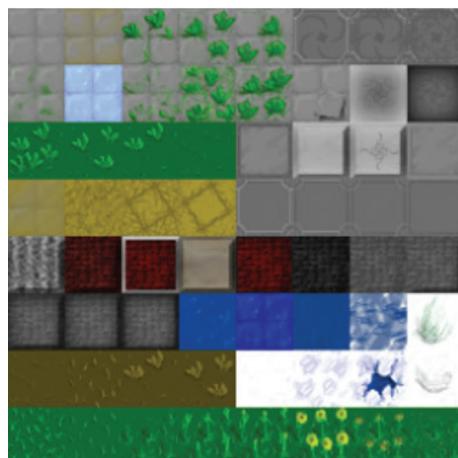


Figura 1 – *Tileset*

Fonte: opengameart.org

A Figura 1 apresenta diversos elementos para criar a cena de um Jogo e cada bloco tem dimensões de **32 x 32**. Com Programação, podemos percorrer cada um desses blocos para criar o Jogo.

Também é possível usar essa mesma técnica para criar o mundo ou níveis do Jogo.

Nas próximas seções, veremos como utilizar esses mapas.



- Noções de *tile* e *tilemaps* da *MDN web docs*. Disponível em: <https://goo.gl/vk4Bzf>
- Veja como utilizar mapas baseados em *tilesmaps* com *Canvas* e *JavaScript* no *MDN web docs*. Disponível em: <https://goo.gl/7KRPbr>
- Quando trabalhamos com mapas grandes, é necessário mover a câmera pelo *Canvas*. Veja um exemplo dessa implementação com *JavaScript* puro no *MDN web docs*. Disponível em: <https://goo.gl/fUbepF>

## Ferramenta para Criar Tileset/Tilemap

Se você procurar rapidamente no *Google* alguns *tilesets* ou *tilemaps*, irá encontrar diversas opções; contudo, em alguns sites os *tiles* são fornecidos separadamente. Quando você se deparar com isso, deverá criar seu próprio *tileset* para então montar seu cenário ou criar um *tilemap*.

Para realizar essa tarefa de criar um *tileset* com base em imagens separadas, utilize programas gráficos. Nossa sugestão gratuita seria o *software Gimp*. Como nosso foco não é esse programa, deixaremos um tutorial para você fazer, referente à criação de um *tileset* nessa ferramenta; é algo bem simples.

Seguindo o tutorial mencionado no final deste item, podemos criar, por exemplo, o arquivo de imagem, como exibido na Figura 2.

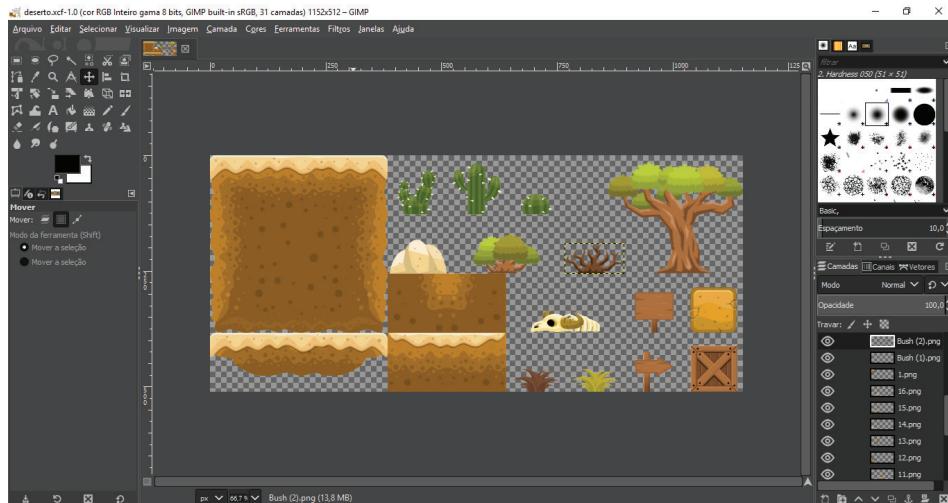


Figura 2 – Imagem com *Tiles* Juntos

Se pensarmos que cada componente da imagem exibida na Figura 2 tem a medida de **128 x 128**, temos uma malha que separa cada parte.

A Figura 3 exibe essa divisão.

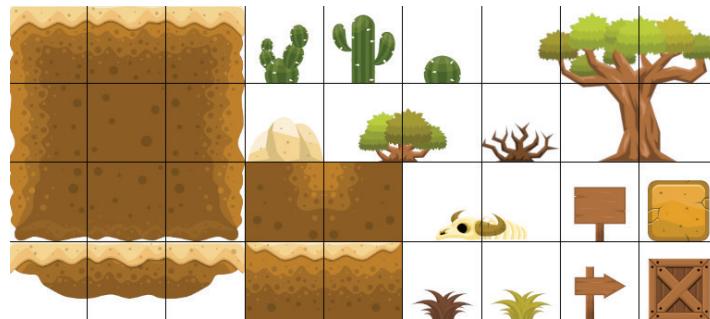


Figura 3 – Separação das imagens

Essas linhas que colocamos são somente para mostrar a disposição dos diversos objetos no arquivo .PNG. Quando trabalharmos com ele no *JavaScript*, essas linhas não existirão.

Com a imagem criada, agrupando todos os objetos que irão fazer parte do Jogo, podemos começar a criar nossos cenários, mas existem duas formas de fazer isso: a primeira seria, com base na imagem que possui os *tiles*, criar nossos cenários ou níveis com Programação e configurar cada posição dos elementos; a outra seria criar um *tilemap* e trabalhar com ele diretamente no nosso *script*.

Veremos a primeira opção no item 3 e a segunda no item 4.

Para criar nossos mapas para o item 4 deste Material, faremos uso da ferramenta chamada *Tiled*. Esse software é um editor de *tileset* e *telemap* e irá nos ajudar a criar nossos mapas ou níveis para o Jogo.

O *Tiled* permite trabalhar com camadas e, inclusive, adicionar colisões nos elementos do Jogo. Nós veremos como fazer isso.

Para criar os mapas do item 4, abra o *Tiled* e escolha a opção de Novo mapa. Aparecerá a janela para configurarmos nosso mapa (Figura 4).

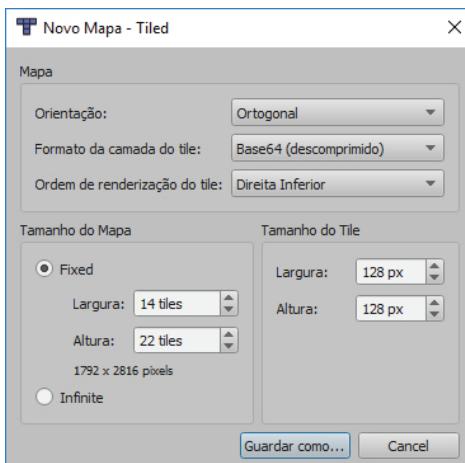


Figura 4 – Criando um mapa no *Tiled* (Parte1)

Nessa tela de configuração, podemos mudar a orientação do mapa, como, por exemplo, para isométrico, o formato da camada etc. Mantenha as configurações do Mapa padrões para esse exemplo, e iremos alterar as informações do tamanho do mapa e do tamanho do *tile*.

No Material Complementar fornecido, disponibilizamos algumas imagens. Se você as analisar, nossos *tileset* são compostos por quadrado de **128 x 128**; logo, iremos colocar essas medidas na caixa largura e altura do *tiled*.

Para o mapa, você pode colocar a medida que quiser; no geral, você informará a quantidade de *tiles* para a largura e para a altura, utilizando um exemplo simples, como um *tile* de medida **10 x 10** e uma mapa de **10 x 5**, por exemplo, em que teríamos uma imagem com dimensões de **100px x 50**.

Então, fique atento(a) sempre ao tamanho do seu arquivo, para não ficar impossível carregá-lo.

No nosso exemplo, como nossos tiles possuem **128 x 128** de dimensão, nosso mapa de **10 x 5** terá a medida em pixel de **1280 x 640**.

Após configurar e gravar o arquivo, você terá uma tela como a exibida na Figura 5.

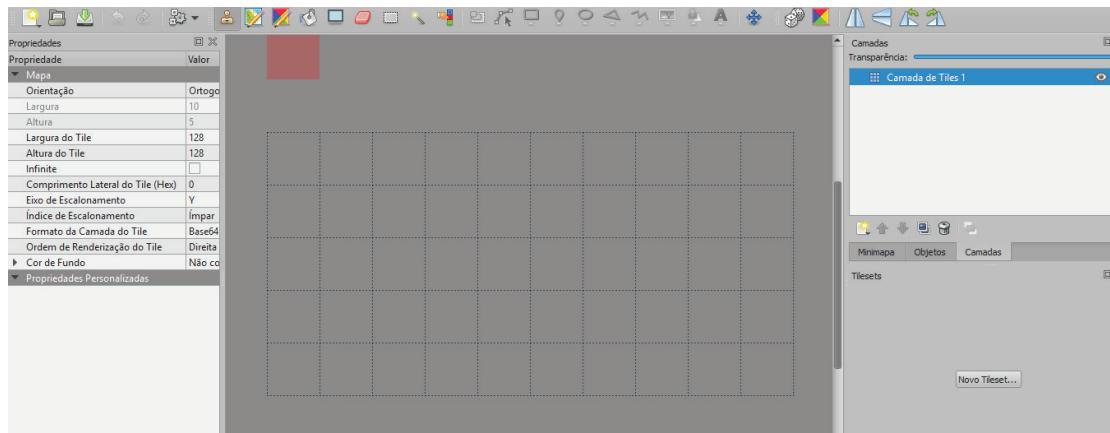


Figura 5 – Criando um mapa no *Tiled* (Parte 2)

Nosso novo mapa fará uso do *tileset* criado anteriormente; para isso, vamos clicar no botão Novo *Tileset*, no painel à direita. A janela de configuração do *tileset* irá aparecer (Figura 6).

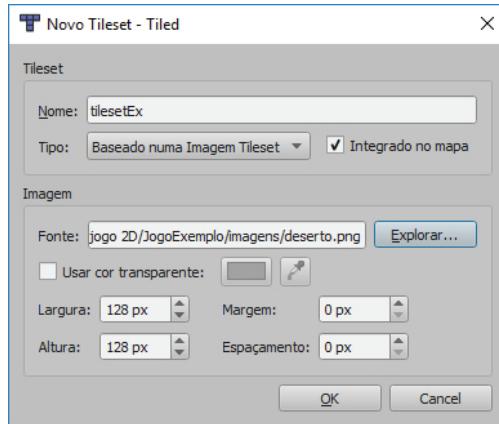


Figura 6 – Configurando o *Tileset* do mapa

Insira um nome para o *tileset*; deixe marcada a opção de Integrado ao mapa e selecione o arquivo criado anteriormente no *Gimp* (o arquivo .PNG, veja no material complementar o nome *deserto.png*).

É importante, na região da imagem, colocarmos o tamanho correto de cada *tile*. Nesse caso, mantenha o valor de **128 x 128**.

Após as configurações, você irá visualizar o *tileset* criado no Painel da Esquerda (Figura 7).

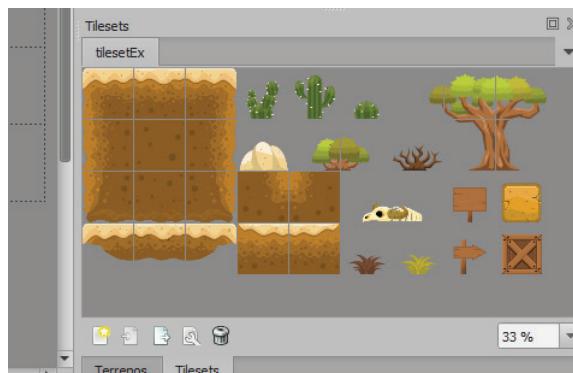


Figura 7 – *Tileset* criado com a Figura *deserto.png*

Fique atento ao nome dado a esse *tileset*. Ele será usado na Programação. Agora, é só criar o cenário conforme sua necessidade.

Para criar o cenário, clique no *tiled* e, na sequência, na malha do mapa. Para marcar mais de um *tile* em conjunto, pressione a tecla *Ctrl* enquanto selecione os *tiles*.

Antes de começar, vamos renomear a camada padrão para Plataformas e criar uma nova, com o nome de objetos (Figura 8).

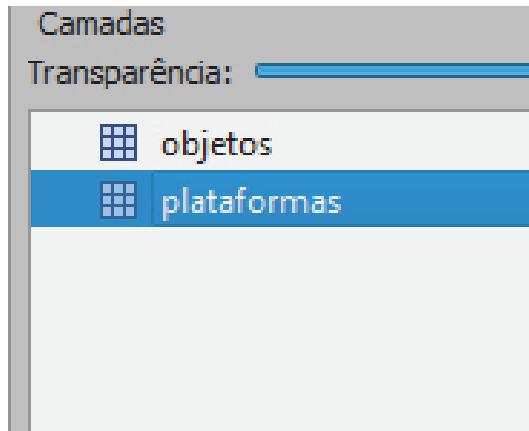


Figura 8 – Camadas do mapa

Agora iremos criar as plataformas, na camada “Plataformas”, e os elementos que irão fazer parte do cenário, na camada “objetos”. Esses nomes de camadas serão utilizados na Programação. **Fique atento(a)!**

Procure montar o cenário conforme a Figura 9.

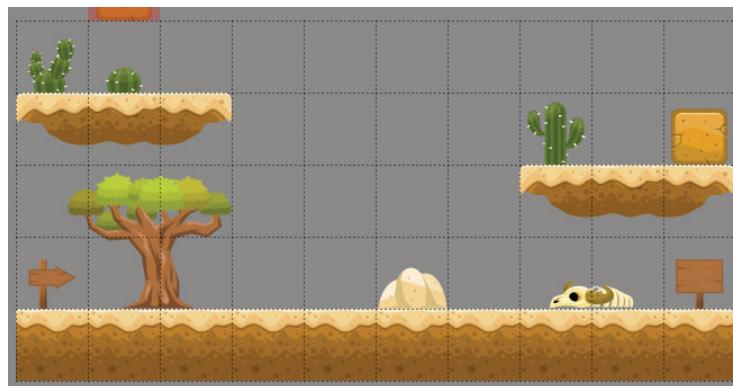


Figura 9 – Exemplo de *tilemap*

Cada objeto fica em um quadrado da malha; caso você precise colocar elementos sobrepostos, trabalhe com várias camadas.

**Estamos quase lá!**

Salve seu arquivo de mapa.

Para trabalharmos com esse mapa no *Phaser*, precisamos exportar o arquivo para o formato JSON. Esse formato de arquivo é uma forma que a ferramenta possui para criar toda a especificação do mapa. JSON significa *JavaScript Object Notation*, cujo link para consulta sobre o assunto está nos Materiais Complementares.

Para exportar o arquivo, salve primeiramente o mapa; em seguida, clique no menu Arquivo>>Exportar como. Dê um nome para o arquivo e escolha o tipo Arquivo de mapas JSON.

Já estamos com tudo preparado. Agora, vamos aprender como trabalhar com nosso Jogo no *Phaser*, usando os arquivos fornecidos nos Materiais Complementares, que trazem:

- Jogo de exemplo com níveis;
- Imagens dos objetos do deserto separadas;
- Imagens png de artefatos do Jogo;
- Imagem original do *Gimp* (arquivo xcf);
- Mapas criados na unidade e mapas dos níveis;
- Arquivos de *Tiles*.

Para encerrarmos o uso do *Tiled*, precisamos aprender a definir os objetos que terão colisão no Jogo.

No painel do *tileset*, clique no painel Editar *Tileset* (Figura 10).

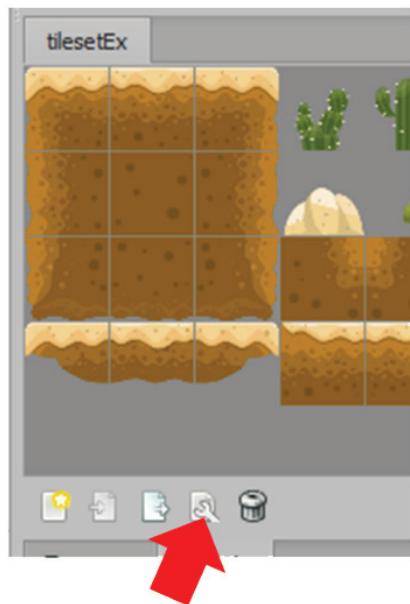


Figura 10 – Editor de *Tileset*

Agora, clique na ferramenta de Editor de Colisão de *Tiles* (Figura 11).

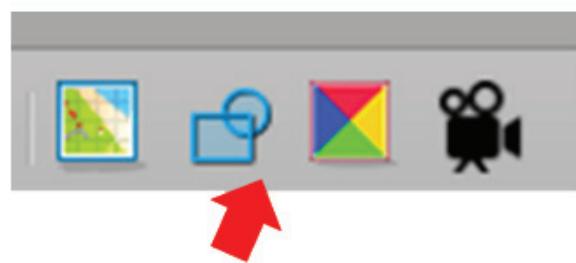


Figura 11 – Editor de Colisão de *Tiles*

Com a ferramenta ativada, marque todos os *tiles* na janela central. Precisamos inserir uma nova propriedade para esses *tiles*. Assim, com eles selecionados, clique no sinal de “+” do painel de propriedades e crie a propriedade *collider* do tipo *bool* (Figura 12).

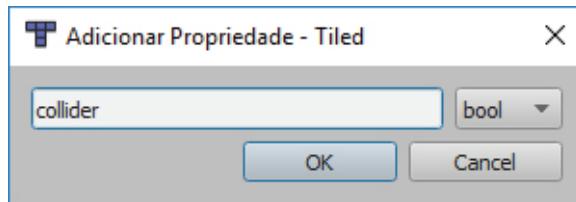


Figura 12 – Definindo uma nova propriedade para os *tiles*

Agora que todos os *tiles* possuem essa propriedade, você deverá selecionar somente os que deverão ter colisão no Jogo e marcar essa propriedade para que ela fique verdadeira.

Veja um exemplo na Figura 13.

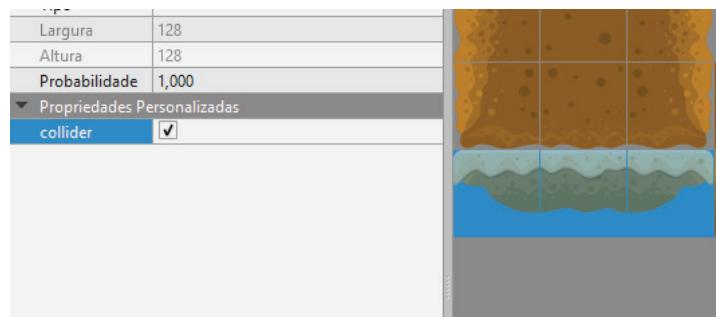


Figura 13 – Ativando a colisão para um *tile*

Salve as modificações no *tileset* e exporte novamente seu *timemap*. Agora podemos iniciar a programação.

Essa configuração será utilizada no código com o auxílio do *Phaser*.



- Acesse o Manual do *Tiled* no endereço: <https://goo.gl/8AWbVc>
- É possível criarmos nossos *tilesets* na ferramenta *Gimp*. Acesse: <https://goo.gl/fKGaxR>
- Acesse o site do *Gimp* para conhecer a ferramenta e para baixá-la. <https://goo.gl/3HaQCe>
- Acesse o site do *Tiled* para conhecer a ferramenta e para baixá-la: <https://goo.gl/ZDCVPv>
- Para obter as informações sobre o formato *JSON*, acesse: <https://goo.gl/yy31KV>

## Criando Cenários com Blocos (*Tileset* e Programação)

Vamos criar nosso primeiro cenário utilizando o arquivo *deserto.png*. Para ficar claro como os elementos do *tileset* serão utilizados, inserimos uma numeração nos quadrados que delimitam cada *tile* (Figura 14).

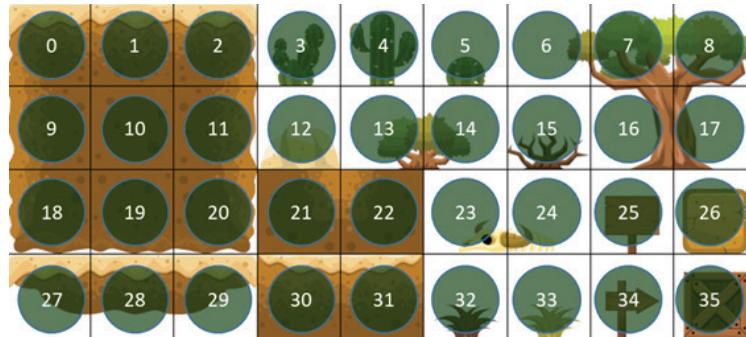


Figura 14 – Exemplo das posições nos *tiles*

O cenário do nosso Jogo irá utilizar somente alguns desses elementos. Quando definirmos a ideia, vamos utilizar esses números para compor a cena.

Vamos iniciar criando o arquivo *index.html* (no *Brackets*) com o código apresentado na Figura 15.

```

1  <!DOCTYPE html>
2 ▼ <html>
3 ▼   <head>
4       <meta charset="utf-8">
5       <title>Jogo PLataforma</title>
6   </head>
7 ▼ <body>
8     <script src="js/phaser.js"></script>
9     <script src="js/main.js"></script>
10 </body>
11 </html>
```

Figura 15 – Jogo com *tileset* e *Phaser* (Parte 1 – *HTML*)

Iremos montar um cenário conforme o modelo apresentado na Figura 16. Inserimos os números dos respectivos *tiles* para facilitar o entendimento.

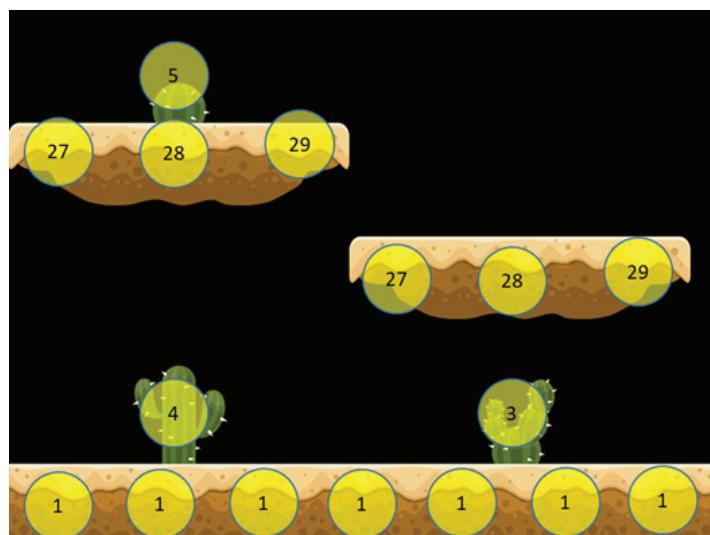


Figura 16 – Jogo com *tileset* e *Phaser* (Parte 2 – Esboço)

Agora que já definimos a ideia, vamos começar a programação com o *JavaScript/Phaser*. Copie o arquivo *deserto.png* para uma pasta chamada *imagens* e copie, também, o arquivo do *phaser.js* para uma pasta chamada *js*.

No *Brackets*, crie um arquivo chamado *main.js* e salve-o na pasta *js*; o primeiro passo, como já vimos em outras Unidades, será configurar o Jogo, inserindo o código apresentado na Figura 17.

```

1 ▼ const config = {
2     type: Phaser.AUTO,
3     width: 800,
4     height: 600,
5     parent: "game-container",
6     scene: [
7         preload,
8         create,
9         update
10    ]
11 };
12
13 const game = new Phaser.Game(config);
14

```

Figura 17 – Jogo com *tileset* e *Phaser* (parte 3)

Precisamos carregar a imagem *deserto* na função *preload*. Insira o código da Figura 18.

```

15 ▼ function preload() {
16     this.load.image("imgtiles", "../imagens/deserto.png");
17 }
18

```

Figura 18 – Jogo com *tileset* e *Phaser* (Parte 4)

Dentro da função *create*, vamos definir a estrutura da cena, especificando a numeração dos *tiles* dentro da imagem, por meio de uma matriz.

Em nosso exemplo, não teremos fundo e para ficar sem fundo, nesse caso, especificamos o valor -1, já que esse *tile* não existe (Figura 19).

```

const level = [
    [-1, 5, -1, -1, -1, -1, -1, -1],
    [27, 28, 29, -1, -1, -1, -1, -1],
    [-1, -1, -1, 27, 28, 29, -1],
    [-1, 4, -1, -1, 3, -1, -1],
    [1, 1, 1, 1, 1, 1, 1]
];

```

Figura 19 – Jogo com *tileset* e *Phaser* (Parte 5)

Para entender essa numeração, volte à imagem do esboço. Para que essa estrutura funcione, vamos implementar um *tilemap* no *Phaser* com base na imagem *deserto* carregada com o nome de *imgtiles*.

Insira o código apresentado na Figura 20.

```

19 ▼ function create() {
20
21 ▼   const level = [
22     [-1, 5, -1, -1, -1, -1, -1],
23     [27, 28, 29, -1, -1, -1, -1],
24     [-1, -1, -1, 27, 28, 29, -1],
25     [-1, 4, -1, -1, 3, -1, -1],
26     [ 1, 1, 1, 1, 1, 1, 1]
27   ];
28
29 ▼   const map = this.make.tilemap({
30     data: level,
31     tileWidth: 128,
32     tileHeight: 128
33   });
34   const tiles = map.addTilesetImage("imgtiles");
35   const layer = map.createStaticLayer(0, tiles, 0, 0);
36 }
37

```

Figura 20 – Jogo com *tileset* e *Phaser* (Parte 6)

A linha 29 cria um *timemap* com os dados definidos na constante *level* e especifica que cada *tile* terá a dimensão de 128 x 128. Na sequência, utilizamos a função *addTilesetImage* para associar a imagem carregada no *preload* e, por fim, definimos que nosso *timemap* terá uma camada estática (linha 35).

Para encerrar, coloque a função *update*, conforme apresentado na Figura 21.

```

38   function update(time, delta) {
39 }

```

Figura 21 – Jogo com *tileset* e *Phaser* (Parte 7)

Se você seguiu todos os passos, seu resultado deve ser igual ao apresentado na Figura 22.

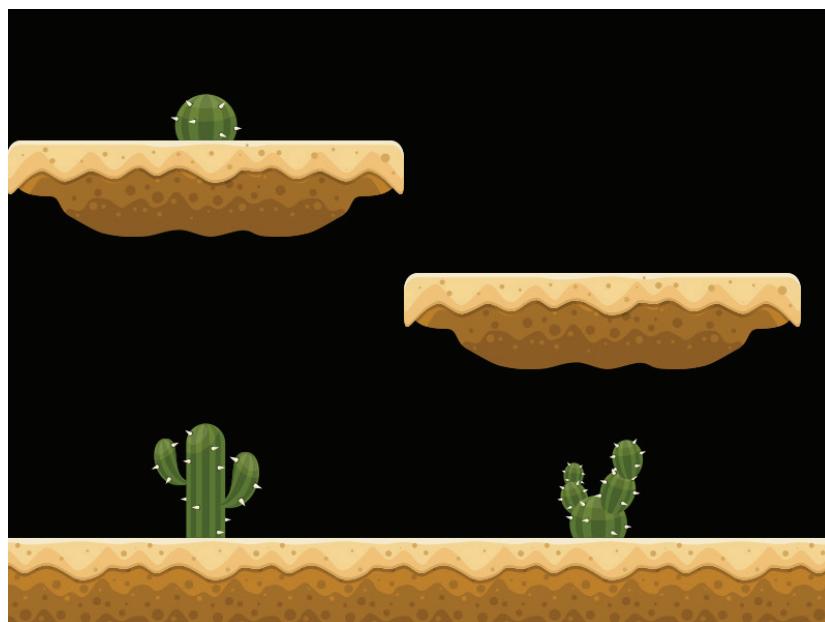


Figura 22 – Jogo com *tileset* e *Phaser* (Parte 8)

Procure fazer outros cenários com essa imagem, para praticar. Nas próximas seções, iremos ver como trabalhar com mapas externos e, posteriormente, como inserir o personagem nesse mapa.

## Trabalhando com um Mapa Criado no *Tiled*

Neste item, iremos trabalhar com um *tilemap* criado no software *Tiled*. Além de carregar o mapa, iremos inserir um personagem animado para que você veja como começar a criar os seus Jogos, juntando tudo o que vimos até o momento.

Crie uma pasta para esse novo exemplo. Dentro dela, você deverá criar uma subpasta *js* e, posteriormente, copiar o arquivo *phaser.js*. Em seguida, crie uma pasta *imagens* e copie os arquivos *deserto.png*, *MapaEx.json* e *ninja.png*.

Vamos inserir o código HTML apresentado na Figura 23 em um novo arquivo, chamado *index.html*.

```
1  <!DOCTYPE html>
2 ▼ <html>
3 ▼   <head>
4     <meta charset="utf-8">
5     <title>Jogo PLataforma</title>
6   </head>
7 ▼ <body>
8   <script src="js/phaser.js"></script>
9   <script src="js/mainMapa.js"></script>
10 </body>
11 </html>
```

Figura 23 – Jogo com *tilemap* e *Phaser* (Parte 1)

Nosso próximo passo, como você deve imaginar, será configurar o Jogo.

Assim, insira o código apresentado na Figura 24.

```

1 ▼ const config = {
2     type: Phaser.AUTO,
3     width: 800,
4     height: 600,
5     physics: {
6         default: "arcade",
7         arcade: {
8             gravity: {
9                 y: 500
10            }
11        }
12    },
13    scene: {
14        preload: preload,
15        create: create,
16        update: update
17    }
18};
19
20 const game = new Phaser.Game(config);
21 let cursors;
22 let player;

```

Figura 24 – Jogo com *tilemap* e *Phaser* (Parte 2)

Como iremos ter um personagem, precisamos definir o motor de física (linha 5 e 6) e também o valor aplicado na gravidade (linha 8 e 9).

A variável *cursors* servirá para analisarmos as setas direcionais e a variável *player* servirá para manipularmos nosso personagem.

Na função *preload*, precisamos carregar nossos recursos. Insira, agora, o código apresentado na Figura 25.

```

24 ▼ function preload() {
25     this.load.image("tiles", "../imagens/deserto.png");
26     this.load.tilemapTiledJSON("map", "../imagens/MapaEx.json");
27     this.load.spritesheet('ninja', '../imagens/ninja.png', {
28         frameWidth: 45,
29         frameHeight: 54
30     });
31 }
32

```

Figura 25 – Jogo com *Tilemap* e *Phaser* (Parte 3)

A imagem (linha 25) é nosso *tileset* e já foi discutida no item anterior. Assim como a imagem que cria o *sprite sheet* presente na linha 27, que foi trabalhada na Unidade anterior.

A linha que é nova em nosso código é a linha 26. É necessário utilizar a função *tilemapTileJSON* para carregar o arquivo *MapaEx.json*. Esse mapa está disponível na pasta de Materiais Complementares e foi criado no *Tiled*.

Veja sua estrutura na Figura 26.

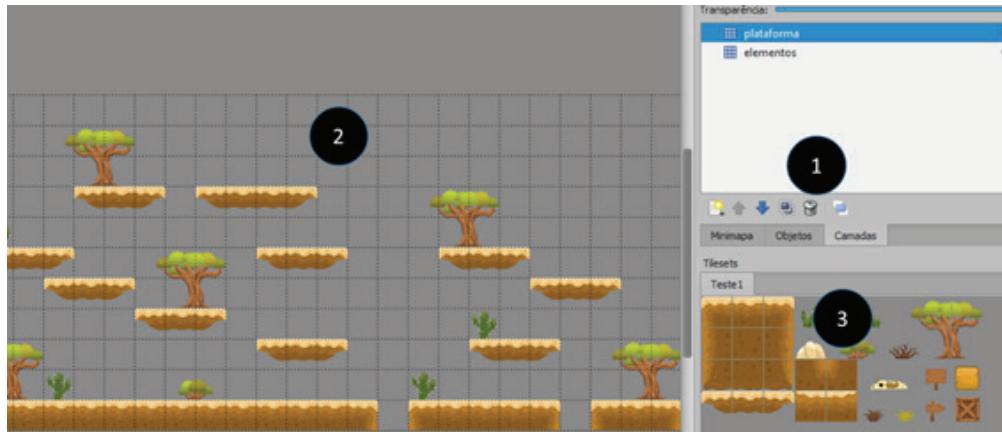


Figura 26 – *Tilemap* criado no *Tiled*

A região 2 é referente ao nosso mapa, no qual criamos uma camada para as Plataformas e uma camada para os elementos do Jogo (veja região 1). Por fim, utilizamos um *tileset* (região 3) chamado de Teste 1.

No *tileset*, como visto no item 2, configuramos tudo o que é chão para ter colisão (Figura 27).

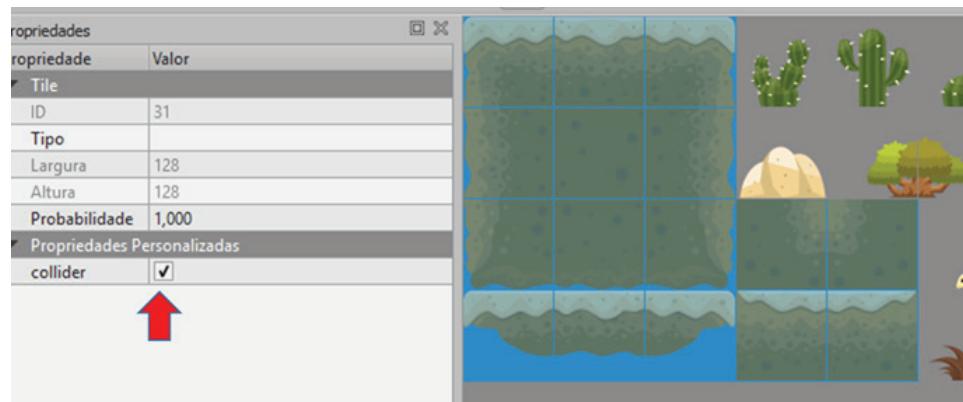


Figura 27 – Colisão no *tileset*

Agora que entendemos nosso mapa, vamos inserir o código da função *create* em partes.

Insira o código apresentado na Figura 28.

```

33 ▼ function create() {
34 ▼   const map = this.make.tilemap({
35     key: "map"
36   });
37   // devemos inserir o mesmo nome que utilizamos no tileset do Tiled
38   const tileset = map.addTilesetImage("Teste1", "tiles");
39
40   const elementos = map.createStaticLayer("elementos", tileset, 0, 0);
41   const plataformas = map.createStaticLayer("plataforma", tileset, 0, 0);
42
43 ▼   plataformas.setCollisionByProperty({
44     collider: true
45   });
46

```

Figura 28 – Jogo com *tilemap* e *Phaser* (Parte 4)

No bloco de linhas 34 a 36, definimos um objeto para nosso *tilemap* com a chave *map*. Para esse *map*, precisamos definir o *tileset* utilizado com as imagens.

A linha 38 insere o mesmo nome que definimos na criação do mapa no *Tiled* (confira na Figura 26, bolinha 3).

Para carregar as camadas corretamente, precisamos utilizar a função *createStaticLayer*. Note que nas linhas 40 e 41 carregamos as mesmas camadas que foram definidas na criação do mapa.

Como configuramos os *tiles* de chão para ter colisão, podemos inserir essa propriedade agora. O bloco de linhas 43 a 45 informa que a propriedade *collider*, quando verdadeira (*true*), deve ser carregada com colisão.

Para termos um Jogo realmente, iremos agora inserir o personagem com animação, igual foi feito na Unidade 4.

Insira o código do *player* e sua animação, apresentado na Figura 29.

```

47   player = this.physics.add.sprite(30, 0, 'ninja');
48
49 ▼   this.anims.create({
50     key: 'left',
51     frames: this.anims.generateFrameNumbers('ninja',
52       start: 21,
53       end: 31
54     ),
55     frameRate: 25,
56     repeat: -1
57   });
58
59 ▼   this.anims.create({
60     key: 'turn',
61     frames: this.anims.generateFrameNumbers('ninja',
62       start: 11,
63       end: 20
64     ),
65     frameRate: 10,
66     repeat: -1
67   });
68
69 ▼   this.anims.create({
70     key: 'right',
71     frames: this.anims.generateFrameNumbers('ninja', {
72       start: 0,
73       end: 10
74     }),
75     frameRate: 25,
76     repeat: -1
77   });
78
79   this.physics.add.collider(player, plataformas);

```

Figura 29 – Jogo com *tilemap* e *Phaser* (Parte 5)

O código apresentado na Figura 30 é o mesmo do exemplo feito na Unidade anterior. Aqui, somente vale destaque a linha 79, na qual configuramos a colisão do *player* com as Plataformas.

Como nosso cenário é grande, é necessário fazer com que a câmera siga o *player*.

Insira agora o código da câmera, para encerrarmos a função *create* (Figura 30).

```

81     const camera = this.cameras.main;
82     camera.startFollow(player);
83     camera.setBounds(0, 0, map.widthInPixels, map.heightInPixels);
84     cursors = this.input.keyboard.createCursorKeys();
85
86
87     // Exemplo de área para a HUD
88     this.add
89     .text(400, 16, 'Se movimento com as setas direcionais', {
90         font: "12px verdana",
91         fill: "#000000",
92         padding: { x: 20, y: 10 },
93         backgroundColor: "#ccc"
94     })
95     .setScrollFactor(0)
96     .setDepth(30);
97 }
```

Figura 30 – Jogo com *tilemap* e *Phaser* (Parte 6)

A linha 82 configura a câmera para seguir o *player*, que é controlado pelas setas direcionais (linha 84). O bloco de código compreendido entre as linhas 87 a 96 é um exemplo de elemento para a HUD do Jogo. Com esse exemplo, você poderá inserir outros elementos como vidas, pontos etc.

**Estamos quase acabando.**

Para finalizarmos o Jogo, precisamos inserir o código da função *update* (Figura 31).

```

99 ▼ function update(time, delta) {
100
101     const speed = 175;
102
103     if (cursors.up.isDown && player.body.velocity.y == 0) {
104         player.body.setVelocityY(-400);
105     }
106
107     if (cursors.left.isDown) {
108         player.body.setVelocityX(-speed);
109         player.anims.play('left', true);
110     } else if (cursors.right.isDown) {
111         player.setVelocityX(speed);
112         player.anims.play('right', true);
113     } else {
114         player.setVelocityX(0);
115         player.anims.play('turn', true);
116     }
117 }
```

Figura 31 – Jogo com *tilemap* e *Phaser* (Parte 7)

O código apresentado na Figura é semelhante ao que foi feito na Unidade anterior. Neste, inserimos uma variável para a velocidade do jogador (linha 101) e na condição referente ao pulo (linha 103) adicionamos uma verificação da velocidade Y do personagem.

A verificação garante que o personagem só pule quando estiver encostado no chão, ou seja, a velocidade no eixo Y deve ser igual a 0.

Com esse exemplo, você poderá criar Jogos com cenários bem extensos, como, por exemplo, o Jogo do *Super Mario Bros*.

Contudo, pode estar curioso(a) e se perguntando como podemos criar mais níveis no Jogo?

Essa resposta será dada no próximo item.

## Mudando de Nível e Outros Recursos

Neste item, vamos responder à pergunta sobre criar vários níveis. Na pasta de Materiais Complementares, deixamos um exemplo completo, com os mapas de dois níveis e o código do *Phaser*.

Para o exemplo mencionado acima, criamos dois mapas no *Tiled*, que usam o mesmo *tileset*

A Figura 32 apresenta o Mapa do Nível 1.

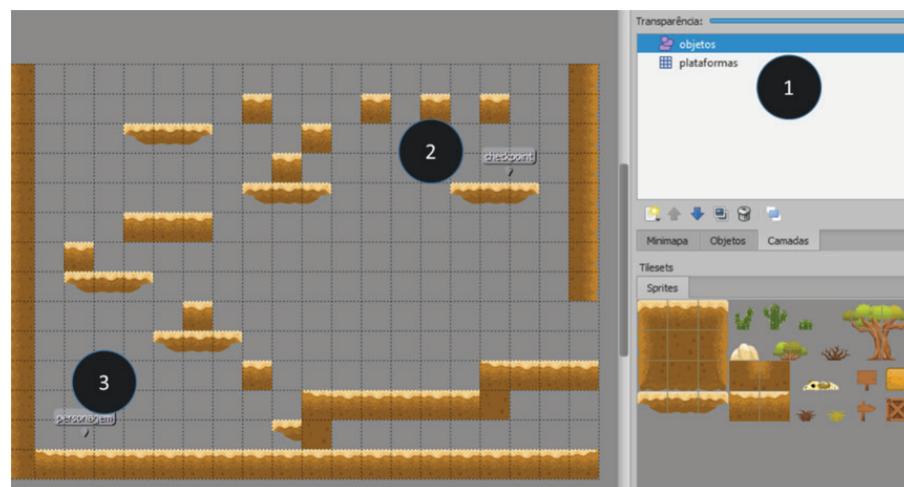


Figura 32 – Mapa Nível 1

Na criação do nível 1, criamos uma camada do tipo *object* (bolinha 1) para inserirmos dois pontos no mapa (bolinhas 2 e 3).

Esses pontos foram configurados com um nome específico, pois serão os locais nos quais poderemos inserir nossos objetos pelo código. A propriedade que foi configurada para ambos é apresentada na Figura 33.

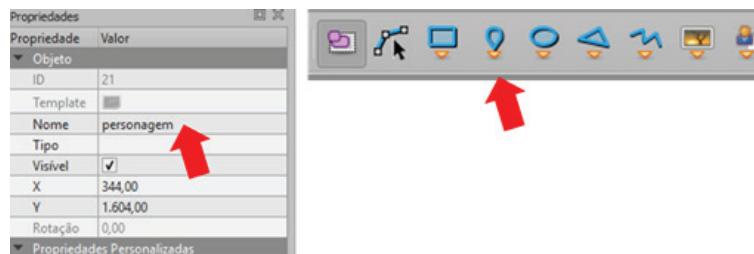


Figura 33 – Propriedade nome

Indicamos a ferramenta que foi utilizada para criar o ponto no qual os objetos serão inseridos. Veja que temos várias opções na barra de ferramentas.

Ao lado esquerdo na Figura 33, apresentamos a propriedade Nome, que foi configurada como personagem para o *player* e como *checkpoint* para inserir a chave de mudança de nível.

Em um exemplo simples, colocamos esses dois pontos lado a lado e, posteriormente, inserimos os objetos por meio do código.

Veja na Figura 34.

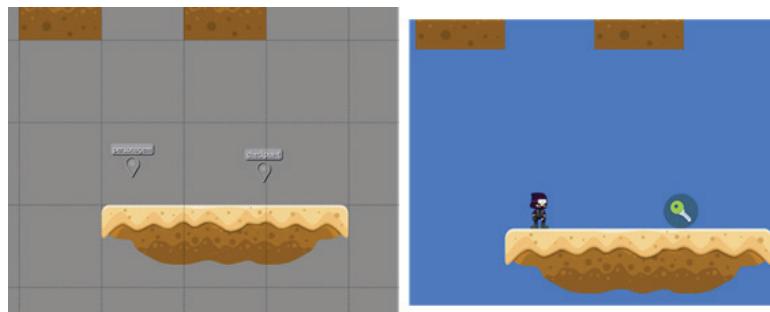


Figura 34 – Trabalhando com a camada objeto

Agora que já entendemos os conceitos do *Tiled*, vamos discutir sobre o código. Precisamos criar nosso código utilizando Classes e Cenas. No exemplo fornecido, criamos 2 cenas, para facilitar o entendimento.

A Figura 35 apresenta o código sem as instruções dentro das funções.

```

1 ▼ var Nivel1 = new Phaser.Class({
2   Extends: Phaser.Scene,
3   initialize: function Nivel1() {
4     Phaser.Scene.call(this, {
5       key: 'Nivel1'
6     });
7   },
8   preload: function () {
9   },
10  },
11  },
12  },
13  },
14  },
15  },
16  },
17  },
18  },
19  },
20  });

```

```

23 ▼ var Nivel2 = new Phaser.Class({
24   Extends: Phaser.Scene,
25   initialize: function Nivel2() {
26     Phaser.Scene.call(this, {
27       key: 'Nivel2'
28     });
29   },
30   preload: function () {
31   },
32   },
33   },
34   },
35   },
36   },
37   },
38   },
39   },
40   },
41   });
42 });

```

Figura 35 – Código de exemplo da criação de cenas

O nome das cenas é configurado na variável *config* (Figura 36).

```
var config = {
    type: Phaser.AUTO,
    width: 800,
    height: 600,
    backgroundColor: '#3681f8',
    physics: {
        default: "arcade",
        arcade: {
            gravity: {
                y: 500
            }
        },
        scene: [Nivel1, Nivel2]
    };
}
```

Figura 36 – Configurando as cenas no *config*

Perceba que as cenas criadas anteriormente possuem as três funções padrões: *preload*, *create* e *update* e que dentro de cada uma devemos criar a lógica necessária para cada nível.

Outro trecho de código que merece destaque é a parte que insere os objetos (*player* e *chave*) por meio da Programação nos pontos definidos no mapa.

A Figura 37 apresenta o código necessário para realizar essa tarefa.

```
31 const spawnPoint = nivel1.findObject("objetos", obj => obj.name === "personagem");
32 player = this.physics.add.sprite(spawnPoint.x, spawnPoint.y, "ninja");
33
34 const checkpoint = nivel1.findObject("objetos", obj => obj.name === "checkpoint");
35 chave = this.physics.add.sprite(checkpoint.x, checkpoint.y, "chave");
```

Figura 37 – Inserindo objetos em pontos específicos

No código da Figura 37, primeiro capturamos o nome da propriedade presente na camada objetos do *timemap* (linha 31 para o *player* e linha 34 para o *checkpoint*). A partir desta variável, temos acesso aos pontos **X** e **Y** desses pontos dentro do mapa e podemos utilizá-los como referência para colocarmos nossos *sprites* (linha 32 e 35).

Agora para mudar de nível, utilizamos a colisão: quando o personagem colide com a chave, é chamada outra cena (Figura 38).

```
70 ▼      this.physics.add.overlap(player, chave, function(){
71
72         this.scene.start("Nivel2");
73
74     }, null, this);
75
```

Figura 38 – Iniciando outra cena

Para que o personagem não empurre a chave, demos preferência a utilizar a função *overlap*; ao colidir o *player* com a chave, é utilizada a função *start*, que chamará a nova cena – ou nível, que criamos anteriormente (linha 72).

Os demais códigos desse exemplo fornecido no ambiente já foram estudados anteriormente.

Agora você pode praticar e criar seus Jogos estilo Plataforma.

Temos muito mais opções para criar Jogos de diferentes gêneros. Procure praticar mais e pesquisar outros materiais com tutoriais de Jogos como de naves e isométricos, entre outros.

Um bom local para ver mais exemplos do *Phaser* é no *PhaserLab*: procure abrir os materiais que eles fornecem para explorar outras ideias.



*Phaser Labs*. Disponível em: <https://goo.gl/QbVazn>

# Material Complementar

Indicações para saber mais sobre os assuntos abordados nesta Unidade:

## Sites

**W3Schools HTML**

*The Language for Building Pages.*

<https://goo.gl/gLZ9v>

**Solearn**

*Everyone can Code*

<https://goo.gl/zAeEe8>

**Codecademy**

<https://goo.gl/OTm7rG>

**Tutoriais MDN Mozilla**

<https://goo.gl/QKTk7>

**Phaser Framework**

<https://goo.gl/j8P8H>

## Referências

BUCHARD, E. **The Web Game Developer's Cookbook: Using Java Script and HTML5 to Develop Games.** USA: Addison-Wesley, 2013.

RAMTAL, Dev; DOBRE, Adrian. **Physics for JavaScript games, animation, and simulations: with HTML5 canvas.** New York: Apress, 2014.

SILVA, Mauricio Samy. **Html 5:** a linguagem de marcação que revolucionou a web. São Paulo: Novatec, 2011.

WORLD WIDE WEB CONSORTIUM W3C. Disponível em: <<https://www.w3.org/html/>>. Acesso em: 9 jun. 2018.





**Cruzeiro do Sul**  
Educacional