Phaser, Part III

Polishing your Phaser game

Today we'll learn about:

- Creating environments using tile maps
- Collision with tiles
- Creating text dialogs
- Adding sound to your game

Follow along!

Get the code for today's example game on GitHub!

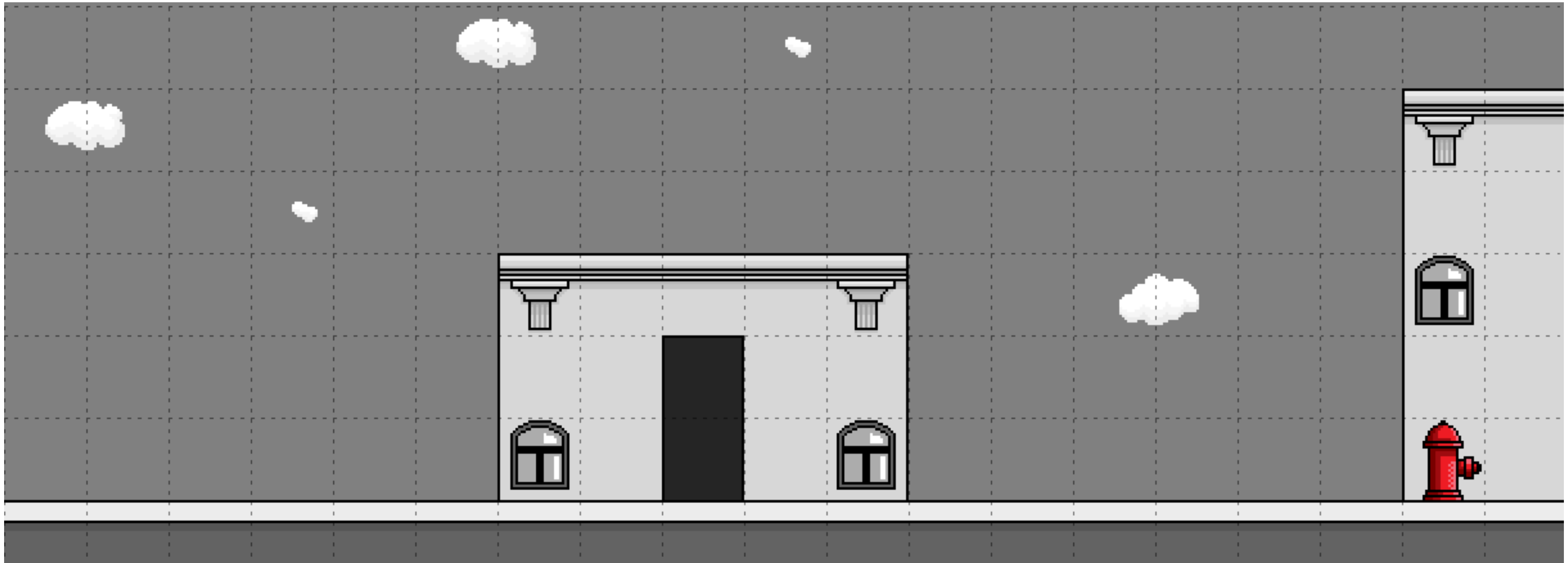
The URL: github.com/cattsmall/Phaser-game

Switch to the **8-2014-game** branch, then click the **Download ZIP** button.

Tile maps

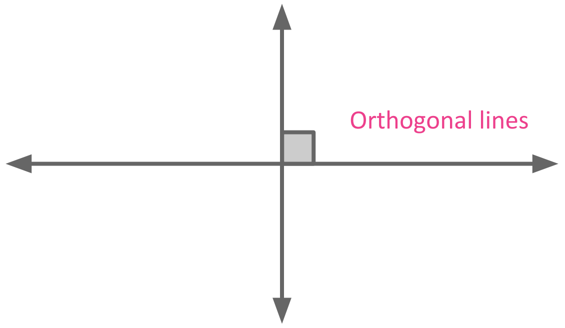
Tile maps

You can design levels using blocks of art called tiles.



Tile maps

Phaser allows you to use tile maps for creating levels. Tile maps must be **orthogonal** in order to be used in Phaser.



Tile map editors

There are many free editors you can use:

- **Tiled** (Mac, Linux, PC) - mapeditor.org
- **DAME** (Mac, Linux, PC) - dambots.com/dame-editor
- **tIDE** (PC) - tide.codeplex.com
- **Online Tile Map Editor** (Browser) elias-schuett.de/git/Online-Tile-Map-Editor

Tile map editors

Phaser allows you to use two types of files when you export tile maps:

- **JSON** - JavaScript Object Notation
- **CSV** - comma-separated values

Preloading tiled maps

Use the **load.tilemap** function to preload maps. You can either use a file or JavaScript object.

```
game.load.tilemap('map', 'map.json', null, Phaser.Tilemap.TILED_JSON);
```

$$\texttt{game.load.tilemap(}\underbrace{\texttt{'map'}}_{\text{Map name}}\texttt{, }\underbrace{\texttt{'map.json'}}_{\text{File URL}}\texttt{, }\underbrace{\texttt{null}}_{\text{JSON Object (if no file)}}\texttt{, }\underbrace{\texttt{Phaser.Tilemap.TILED\_JSON}}_{\text{File format (}\textbf{TILED\_JSON}\text{ or }\textbf{CSV}\text{)}}\texttt{);}$$

Preloading a tileset

Your tileset should be preloaded as an **image** – the tile map defines where the image will get cut up.

$$\texttt{game.load.image(}\underbrace{\texttt{'tileset'}}_{\text{Name}}\texttt{, }\underbrace{\texttt{'assets/tiles.png'}}_{\text{File URL}}\texttt{);}$$

Using tiled maps

Use the **add.tilemap()** function in **create()** to add your map to the game. Declare your tilemap as a variable so that you can reference it throughout your game.

$$\texttt{myGame.map = game.add.tilemap}\underbrace{\texttt{('map1')}}_{\text{Map name}}\texttt{;}$$

Using tiled maps

Use the **addTilesetImage()** function in **create()** to attach the tileset to your tile map.

```
myGame.map.addTilesetImage('tileset');
```

Image name

Tiled map layers

Tiled maps always have at least one layer. Layer names must match those in your map file/code.

```
myGame.BG = myGame.map.createLayer('BG');
```

$\underbrace{\texttt{'BG'}}_{\text{Layer name in JSON}}$

Tiled map layers

Most tile map editors allow you to use more than one layer. This allows you to separate objects and control collision better.

```
myGame.BG = myGame.map.createLayer('BG');
myGame.Floor = myGame.map.createLayer('Floor');
```

Tiled map layers

The **resizeWorld()** function sets the game world size to match the size of a layer. You only need to use this function on one layer.

```
    myGame.Floor.resizeWorld();
```

To do!

Set up a tiled map.

1. Make or download a map.
2. Preload the map JSON using **load.tilemap()**.
3. Preload the tileset using **load.image()**.
4. Add the map to the game using **add.tilemap()**.
5. Add the map's layers and resize the game world.

Collision with tiles

Checking for collision with map tiles

Before you can check when an object collides with a map tile, you must use the **setCollision()** function in **create()**.

```
myGame.map.setCollision(26, false, 'BG');
```

$\texttt{myGame.map.setCollision(}\underbrace{\texttt{26,}}_{\text{Tile number}}\ \underbrace{\texttt{false,}}_{\text{Solid?}}\ \underbrace{\texttt{'BG'}}_{\text{Layer name}}\texttt{);}$

Checking for collision with map tiles

Want to check collision for a range of tiles? Use the **setCollisionBetween()** function.

```
myGame.map.setCollisionBetween(0, 3, true, 'Floor');
```

From tile number, **to** tile number

Solid?

Layer name

Collision event listener

When collision is enabled between map tiles and an object, you can use the **setTileIndexCallback()** function to run code when collision is detected.

```
myGame.map.setTileIndexCallback(26, react, this, myGame.
BG);
```

Tile number

Function name

Function context

Layer object

Enabling collision between tiles & objects

Allow an object to collide with map tiles via layers.

As with all collision checking, this goes in **update()**.

$$\texttt{game.physics}\underbrace{\texttt{.arcade}}_{\text{Physics type}}\texttt{.collide(}\underbrace{\texttt{myGame.player}}_{\text{Object}}\texttt{, }\underbrace{\texttt{myGame.Floor}}_{\text{Layer object}}\texttt{);}$$

To do!

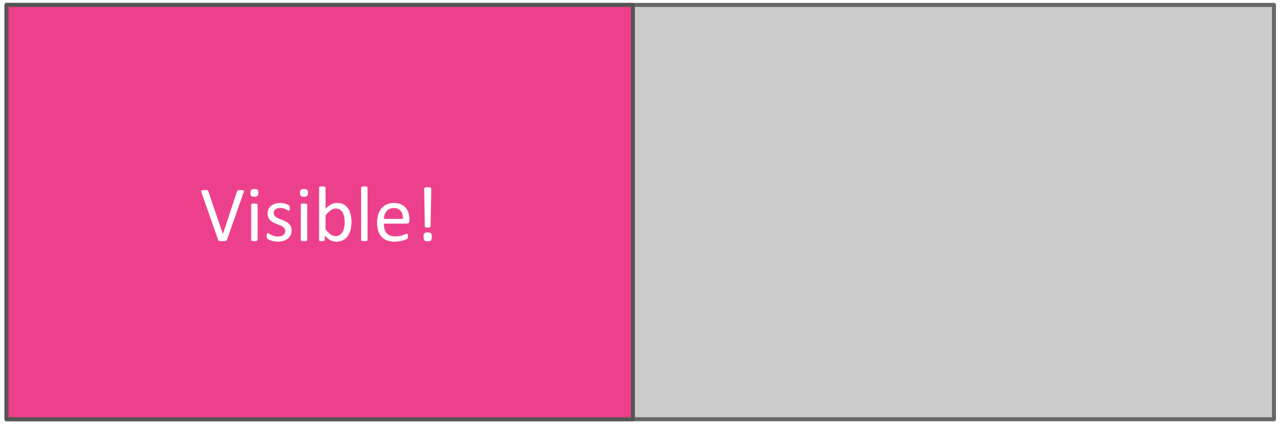
Create collision between an object and a tile.

1. Enable collision using **setCollision().**
2. Listen for collision with **setTileIndexCallback().**
3. Set the object to check for collision with using the **physics.arcade.collide()** function.

The Game Camera

The camera

In games, the camera refers to the visible portion of your game's map.



The camera

Set the camera to follow an object using the **camera.follow()** function.

```
game.camera.follow(myGame.player);
```

$\underbrace{\texttt{myGame.player}}_{\text{Object to follow}}$

The camera

You can also place objects on the screen in relation to the camera’s **x** & **y** properties.

```
myGame.object.x = game.camera.x;
myGame.object.y = game.camera.y;
```

Buttons

Buttons

You can put buttons in your game by loading their images as spritesheets and adding them to the screen using the **add.button()** function.

```
myGame.this.add.button(32, 400, 'startButton', functionName, this, 1, 0,
2);
```

X, Y

Name of sprite to use

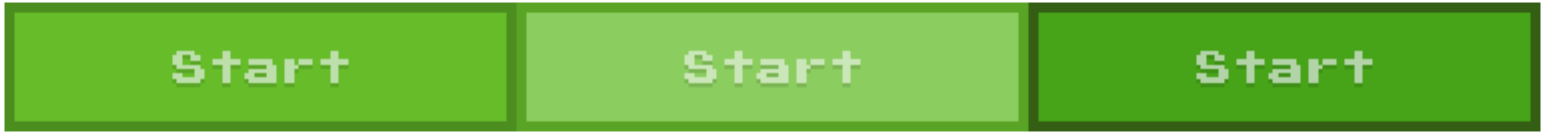
Function to run when pressed

Context for function

Buttons

The last 3 numbers of the button code determine which frames are used for which interaction.

```
myGame.this.add.button(32, 400, 'startButton', functionName, this, 1, 0, 2);
```

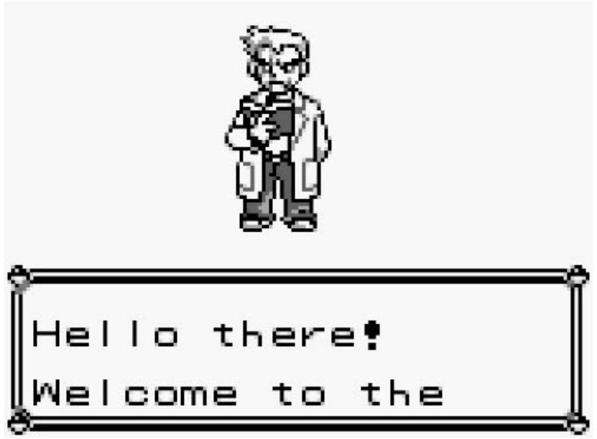


0: default 1: hover 2: pressed (active)

Creating dialogs

Dialogs

Sometimes you might want to show **dialogs**, or windows with text in them, during your game.

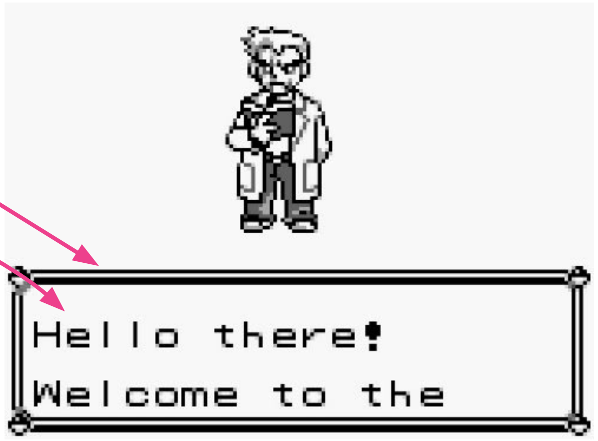


Hello there!
Welcome to the

Dialogs

There are several parts to creating dialogs:

- **A background image**
- **Text content**
- **Easy way to change text**



Loading an image

Preload your dialog’s background image using the **load.image()** function. Within the parentheses, name the image so it can be referenced later, then tell Phaser where to find it in your folder.

```
game.load.image('dialogWindow', 'img/ui/dialog.png');
```

Name of image

Location of image file

Drawing the image

You can draw the image onscreen using Phaser’s **add.sprite()** function.

```
myGame.dialogWindow = game.add.sprite(10, 30, 'dialogWindow');
```

X, Y

Name of image to use

Displaying text

To draw text on the screen, use the **add.text()** function. The text should be instantiated in **create()**.

X, Y

Text content

```
myGame.dialogText = game.add.text(90, 24, 'Hello!');
```

Updating text

In the **update()** function, you can change text using the **setText()** function.

$$\texttt{myGame.scoreText.setText}\underbrace{\texttt{("Score: }}_{\text{string}}\texttt{" + }\underbrace{\texttt{myGame.score}}_{\text{variable}}\texttt{);}$$

Setting visibility

Objects have the ability to be shown or hidden using the boolean **visibility** property.

```
myGame.dialogWindow.visible = true;
```

Functions

You can use **functions** to repeatedly run the same code with different information.

Declaration:

```
myGame.sayStuff = function(foo) {
    console.log(foo);
}
```

Usage:

```
myGame.sayStuff("hello!");
myGame.sayStuff("whazaaaaa!");
myGame.sayStuff("How are you?");
```

To do:

- Write a function that shows a dialog window and dynamically displays text.
- Make sure the dialog appears wherever the player is on the map.
- Run your function in the **update()** function.
- Write a function to hide your dialog.

Adding sound

Preloading audio

You can preload audio using **load.audio()**.

$$\texttt{this.load.audio(}\underbrace{\texttt{'meow'}}_{\text{Name}}\texttt{, }\underbrace{\texttt{'assets/audio/meow.mp3'}}_{\text{File location}}\texttt{);}$$

Preloading audio

To accommodate for browser compatibility issues with certain audio file types, Phaser allows you to use an array instead of a string & list multiple files.

```
this.load.audio('meow', ['assets/audio/meow.mp3', 'assets/audio/meow.
wav']);
```

Adding audio to your game

Use the **add.audio** function in **create()** to add sounds to your game. Declare a variable so you can reference the sound later.

```
myGame.sounds.meow = game.add.audio('meow');
```

Adding audio to your game

You can make a JavaScript object to organize things.

```
myGame.sounds = {};
myGame.sounds.button = game.add.audio('button');
myGame.sounds.pen = game.add.audio('pen');
myGame.sounds.meow = game.add.audio('meow');
myGame.sounds.dice = game.add.audio('dice');
```

Playing audio

Use the **play()** function to play a sound. This can be used inside of functions and conditional statements to create cool interactions.

```
myGame.sounds.meow.play();
```

Looping audio

Phaser also allows you to loop sounds and modify the volume settings in the **add.audio()** function.

```
myGame.sounds.meow = game.add.audio('meow', 1, true);
```

Loop?

Sound volume (0 - 1)

Looping audio

The **play()** function comes with additional settings that also allow you to modify or loop your sound.

$$\texttt{myGame.sounds.meow.play(}\underbrace{\texttt{''}}_{\text{Marker}}\texttt{, }\underbrace{\texttt{0}}_{\text{Starting position of sound}}\texttt{, }\underbrace{\texttt{1}}_{\text{Volume}}\texttt{, }\underbrace{\texttt{true}}_{\text{Loop?}}\texttt{);}$$

To do:

- Find and download an MP3.
- Preload the audio using **load.audio()**.
- Add the audio to your game using **add.audio()**.
- Play the audio using **play()**.

Finale!

You're about to get asked.

How would you:

- Ignore input when a dialog window is open?
- Create a win state for a game?
- Create a lose state?
- Make a game with many levels?
- Ensure that important information is passed throughout your game?

To do:

Make a small interactive game in Phaser. Put the game on GitHub when you finish and share it with us via email, Facebook, or Twitter!

Thanks! Questions?

@cattsmall
catt@codeliberation.org