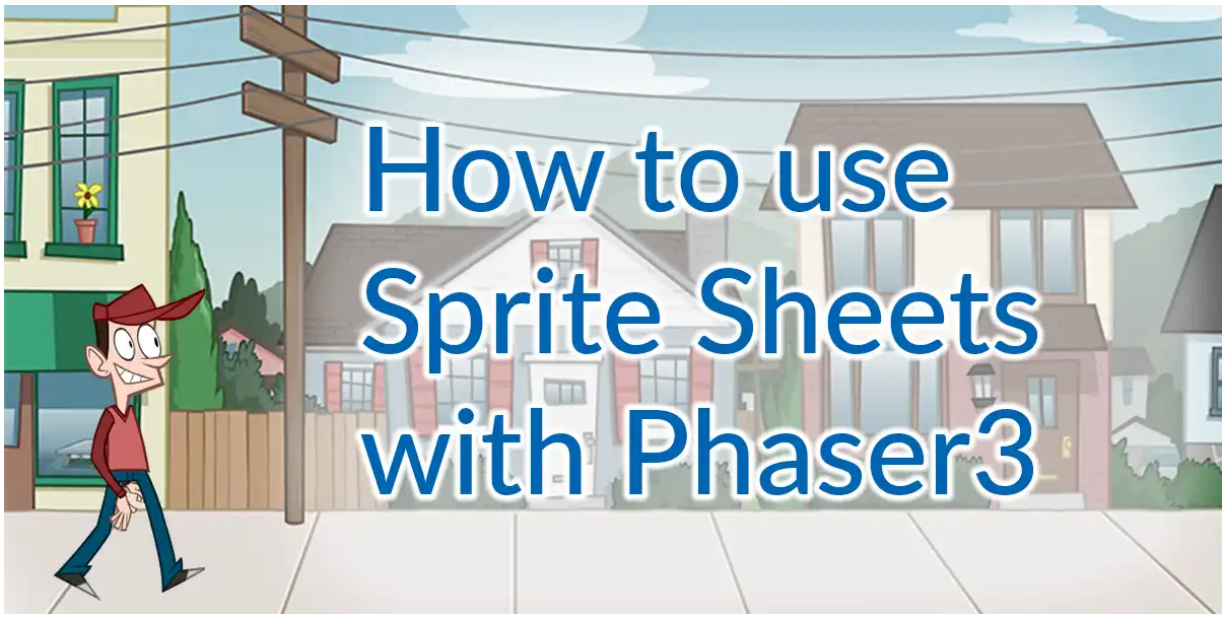
How to create sprite sheets for Phaser 3 with TexturePacker

Joachim Grill, Andreas Löw

texturepacker, phaser, tutorial



What you are going to learn

- Creating sprite sheets with TexturePacker
- Loading sprite sheets in Phaser 3
- Setting pivot points with TexturePacker
- Playing animations from the sprite sheet
- Optimizing start up time and reducing download size
- Complete code is on GitHub

Why should I use a sprite sheet?

The next thing you are going to do is creating a sprite sheet. Using sprite sheets with Phaser has two main reasons:

Speed up loading of the game

Instead of loading tons of single images from a web server all graphics can now be loaded at once. This speeds up loading time of your game.

Improve frame rate

Using a sprite atlas also improves performance of the game. With WebGL textures have only to be set once for rendering.

> *""Texture Packer is an essential part of our daily workflow. Every bit of GPU memory helps when dealing with mobile html5 games, so intelligent packing of assets is a must. And Texture Packer has all the features we need to effortlessly create atlases for our games."*

> *Richard Davey (@photonstorm) - Creator of Phaser"*

Setup a new Phaser project

Creating a new phaser project is quite simple: just clone the **Phaser 3 Webpack Project Template** from GitHub, fetch the node modules, and run the **start** script:

```
git clone git@github.com:photonstorm/phaser3-project-template.git MyPro
cd MyProject
npm install
npm start
```

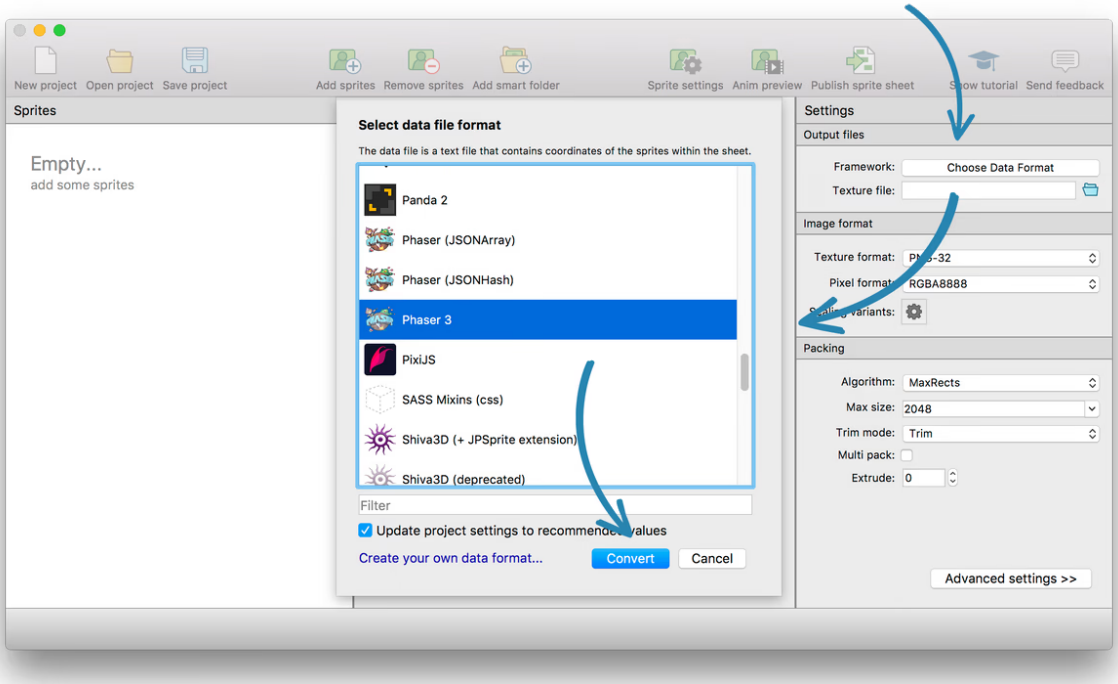
The **start** script bundles your project sources using webpack and serves your app on **localhost:8080**.

The complete demo project we're going to create in this tutorial is also available on GitHub **here**.

Creating sprite sheets - the easy way

The easiest way to create your sprite sheets is using TexturePacker. Please download TexturePacker from here:

When starting the application choose **Try TexturePacker Pro**. In the main window use the **Choose Data Format** button and select **Phaser 3** from the list. You can use the filter to find it faster.

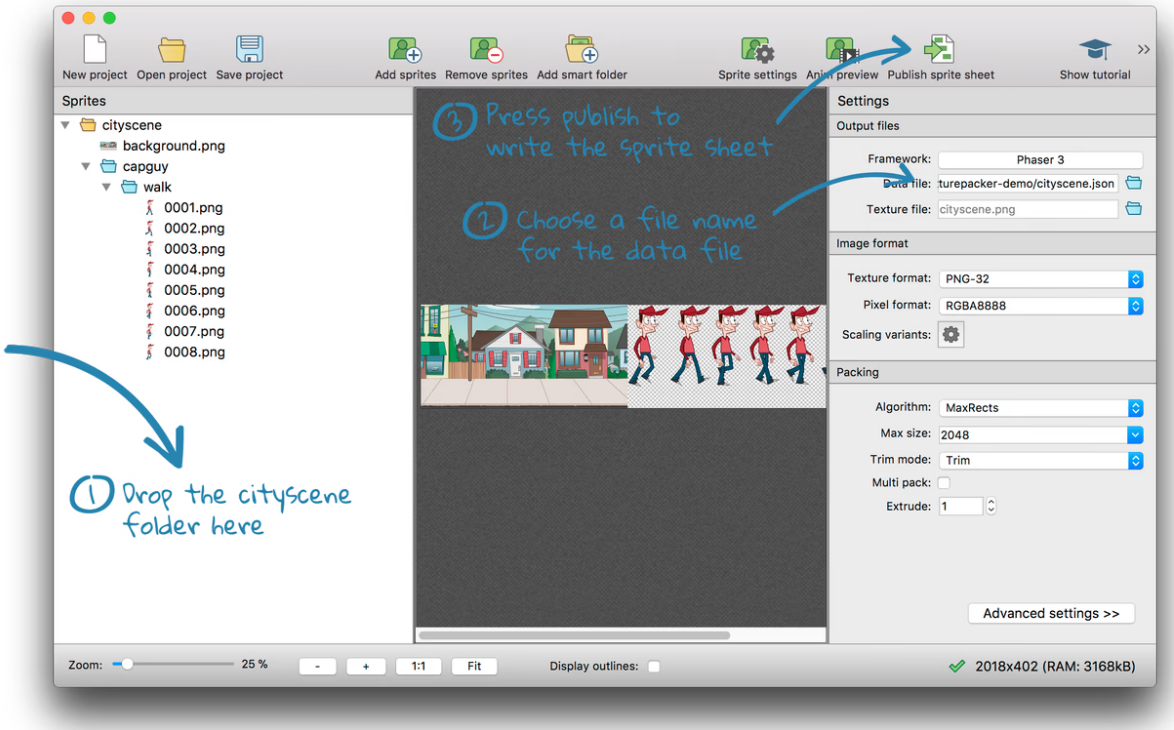


Be careful to select the **Phaser 3** format, only this one supports pivot point editing, multi-pack with one single json file and normal map packing. The other two Phaser data file formats can be used with older Phaser versions. In this case please have a look on the **previous version of this tutorial**

The **demo project** of this tutorial already contains some artwork in the **assets**-folder. Simply drag and drop the **cityscene** folder into TexturePacker.

Adding folders has 2 main advantages over adding single sprites

- Adding or removing sprites in the folder also adds or removes them from the sprite sheet.
- Sub folder names become part of the sprite names — e.g. **`capguy/walk/0001.png`** and not just **`0001.png`**



> TexturePacker currently enables **Allow rotation** by default. The packing algorithm can rotate sprites if this creates smaller textures. Phaser unfortunately only supports rotates sprites with the WebGL renderer. You have to disable the rotation feature if you want to make sure your game runs on all devices (Canvas render as fallback for browsers which don't support WebGL).
>
> Please disable the feature on the **Advanced settings** page. It's in the **Layout** section, which you have to expand to see it.

After that use the file selection button to enter a Data file. Name it **cityscene.json** and place it in the **assets** folder of your project. By default TexturePacker will save the texture image as **cityscene.png** in the same folder.

Finally press **Publish sprite sheet** to create and save the sprite sheet. We are now finished in TexturePacker. That's all it takes to create a sprite sheet.

Loading the sprite sheet in Phaser

The project template contains a **src/index.js** file which displays a bouncing phaser

logo. For our demo app we remove the **`preload()`** and **`create()`** functions and reuse the configuration as starting point:

```
import 'phaser';

var config = {
    type: Phaser.AUTO,
    parent: 'phaser-example',
    width: 800,
    height: 600,
    scene: {
        preload: preload,
        create: create
    }
};

var game = new Phaser.Game(config);
```

The **`preload()`** function is used to load assets. Let's add our own one, which loads the sprite sheet:

```
function preload()
{
    this.load.multiatlas('cityscene', 'assets/cityscene.json', 'assets'
}
```

The first parameter **`'cityscene'`** specifies the key that can be used to access the atlas after it has been loaded. The second parameter **`'assets/cityscene.json'`** is the atlas definition to load, the last parameter **`'assets'`** is the name of the folder in which the image files are stored.

The **`create()`** function is used to setup our game scene. Let's add the background sprite to the scene. Within the sheet the sprite is referenced by its filename (enable "Trim sprite names" in TexturePacker if you prefer sprite names without **`.png`** extension):

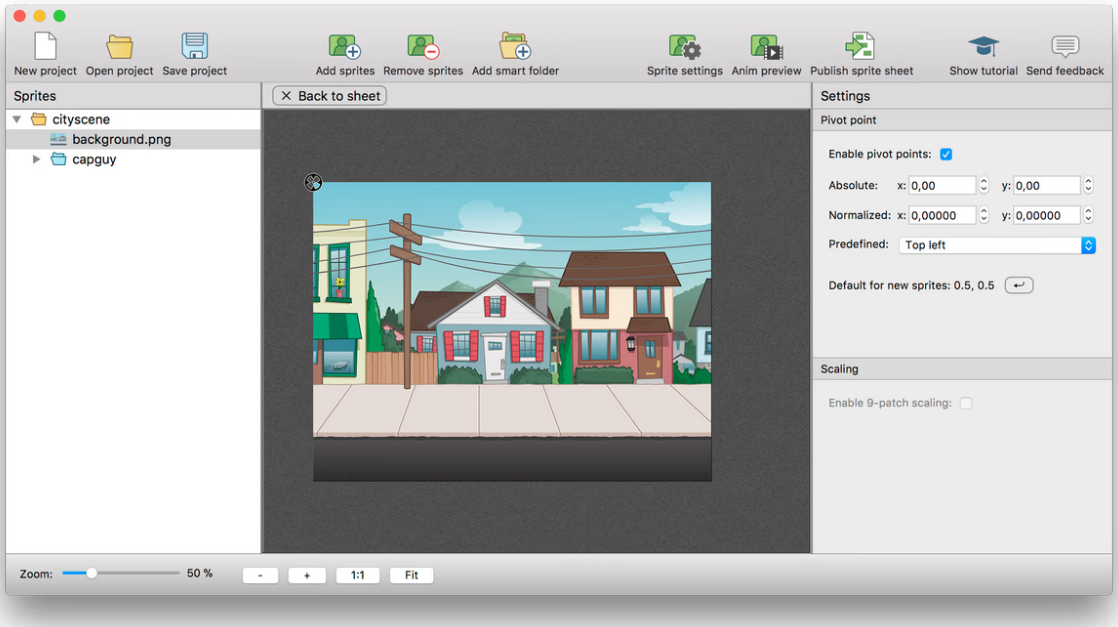
```
function create ()
{
    var background = this.add.sprite(0, 0, 'cityscene', 'background.png
}
```

After saving the **index.js** file the **npm start** script will automatically rebuild the game and refresh the browser window. You will see only a quarter of our background image in the top-left corner:

By default, the pivot point of a sprite is in the center of the image, so the call **`this.add.sprite(0, 0, ...)`** places the center of our background sprite at position (0,0). In the next section we will learn how to fix this.

Setting pivot points with TexturePacker

Pivot points can easily set for each sprite in TexturePacker. Select **background.png** on the left side of the TexturePacker window, and then click on the **Sprite Settings** button in the toolbar. On the left side you can now enable and configure the pivot point of the selected sprite:



Select **Top left** in the **Predefined** input field, and re-publish the sprite sheet. Now reload the browser window displaying your phaser game (changed asset files are not detected automatically). The background image should be perfectly placed now:

If the sprite isn't displayed as expected it's always a good idea to check the Javascript console in your browser. Maybe the sprite sheet wasn't found, or you're using a wrong sprite name? There shouldn't be any warning or error in the console!

Adding an animation

To add a walking character to the scene we have to create an animated sprite and

move it across the screen. First we'll have to create the sprite and store it in a variable declared outside **`create()`**:

```
var capguy;

function create ()
{
    var background = this.add.sprite(0, 0, 'cityscene', 'background.png

    capguy = this.add.sprite(0, 400, 'cityscene', 'capguy/walk/0001.png
    capguy.setScale(0.5, 0.5);
```

This creates a sprite with the first frame of the animation: **`capguy/walk/0001.png`** and places it at position (0,180). The next line scales the sprite down by 50% because it would otherwise be a bit too big for your scene.

The function **`generateFrameNames()`** creates a bunch of frame names by creating zero-padded numbers between *start* and *end*, surrounded by *prefix* and *suffix*. 1 is the start index, 8 the end index, and the 4 is the number of digits to use:

```
var frameNames = this.anims.generateFrameNames('cityscene', {
                    start: 1, end: 8, zeroPad: 4,
                    prefix: 'capguy/walk/', suffix: '.png'
                });
```

The resulting names are:

- **`{ key:'cityscene', frame:'capguy/walk/0001.png' }`**
- **`{ key:'cityscene', frame:'capguy/walk/0002.png' }`**
- ...
- **`{ key:'cityscene', frame:'capguy/walk/0008.png' }`**

Now we can create an animation called *walk* and add it to the *capguy* sprite:

```
this.anims.create({ key: 'walk', frames: frameNames, frameRate: 10, rep
    capguy.anims.play('walk');
}
```

The result is Capguy walking on a spot at the left border:

Moving the sprite

There are several ways to move a sprite in Phaser. You are going to do a simple animation - just pushing the sprite along and resetting it after Capguy left the scene.

Extend the configuration object of your game to now also call a function called update:

```
var config = {
    ...
    scene: {
        preload: preload,
        create: create,
        update: update
    }
};
~~~`

and add the update function like this:

~~~js
function update(time, delta)
{
    capguy.x += delta/8;
    if(capguy.x > 800)
    {
        capguy.x = -50;
```

```
        }
    }
```

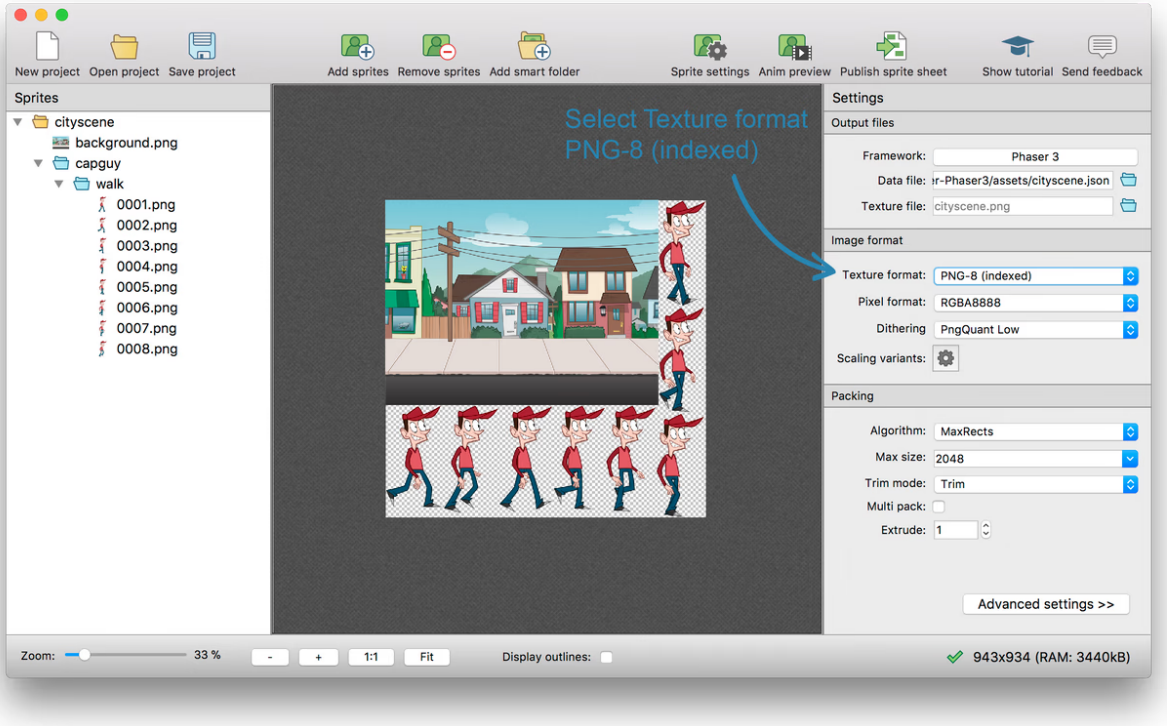
Not much to say about that: The time (in milliseconds) since the previous update call is passed as second parameter. To increase Capguy's position 125 pixels per second we divide delta by 8. After reaching the right border the sprite position is reset to the left border.

Optimizing your game

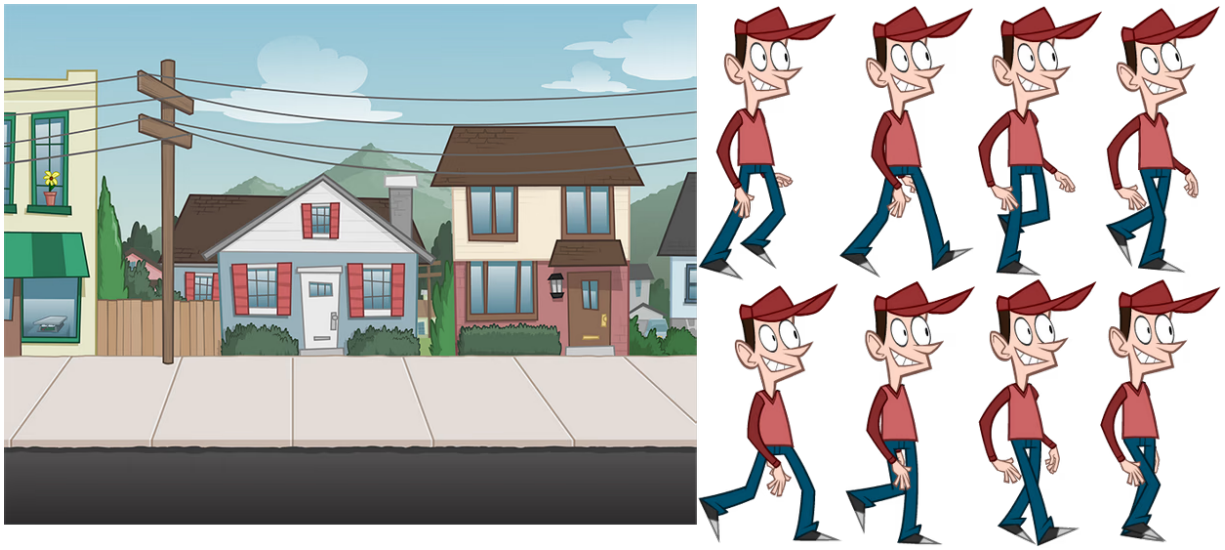
The resulting **cityscene.png** has a size of around 400kb — Not much for a fast internet connection but for a mobile device it might be a good idea to optimize the loading time.

TexturePacker allows you to dramatically reduce the amount of memory used by your sprite sheets — while keeping the sprite quality almost like the original.

To enable this optimization select Texture format **PNG-8 (indexed)**:



Here's the original sheet: **410kb**



Here's the optimized sheet: **115kb - that is 72% less!**