Welcome to Issue 118 of Phaser World

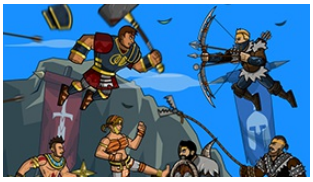
And we're back with another jam packed issue! On the games-front, we've a multiplayer hero battler, a superb game created by a developer for his 4-year old

son and a 3D pixel puzzler. Tutorials include dripping blood, animating tiles and parsing psds.

I've also finally made a decision about the Discord channel and have appointed a new admin to help run it. There's now a dedicated set of mods, new channels, and integrations to make it generally more useful for everyone. I am on there most days, so feel free to tag me if you like.

Until the next issue, keep on coding. Drop me a line if you've got any news you'd like featured by simply replying to this email, messaging me on Slack, Discord or Twitter.

The Latest Games



Game of the Week
On the Way to Nemroth
Ultra-fast action paced, competitive gaming. Pick a hero and get stuck in, taking advantage of your legendary gear and abilities.

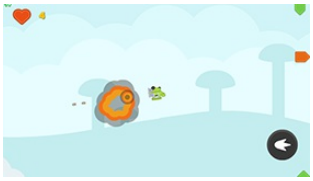
Staff Pick
Kazzle
Create the most impressive castle you can by stacking the pieces, one after the other, and discover hundreds of hidden shapes and creatures.

Math Pixel Puzzle

Study the pixel art for 3 seconds then watch it blow apart. Can you reassemble it in 3D space?

Gladiators of the Underworld Remastered

You are a Necromancer and must prove yourself in battle in this fast-paced, combat focused, arcade survival game.



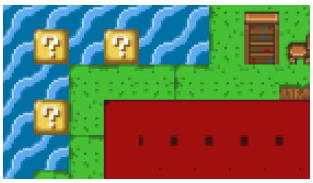
Planes Quick Battle

Take the the skies and blow away the other planes in this arcade battler.

What's New?

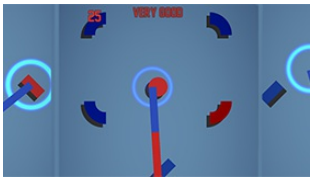
Bloody Drippy Scene

Create a gloriously gory sticky dripping blood effect for your games!



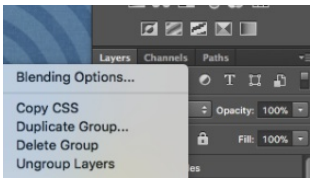
Phaser 3 Animated Tiles Plugin

A Plugin to support animated tiles in Phaser 3, as exported from the Tiled map editor.



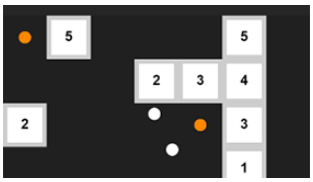
Multiplayer Game Tutorial Series

An on-going tutorial series covering creating a multiplayer game. From leaderboards and room logic to hosting.



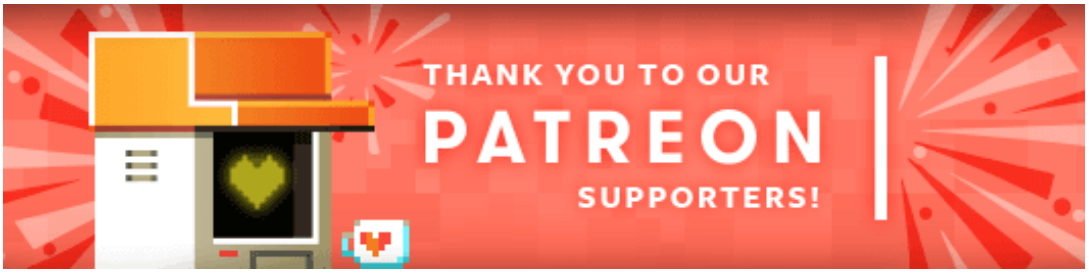
PSD Parse and TexturePacker Tutorial

Generate aPhaser Sprite Sheet from PSD layers using psdparse + TexturePacker.



Ballz Prototype Tutorial Part 3

In the third part of this tutorial series the multiball feature is implemented.



Thank you to our awesome new Phaser Patrons who joined us this week:

Tom B
Max Solomin
Tobias Graf

also thank you to **Robert Willie** for the donation.

We are currently **93%** of our way to the next funding goal: "We will publish an exclusive Phaser 3 code example once per month. Patrons get to vote on the topic they'd like to see covered in the example."

Patreon is a way to contribute towards the Phaser project on a monthly basis. This money is used *entirely* to fund development costs and is the only reason we're able to invest so much time into our work. You can also donate via PayPal.

Backers get forum badges, discounts on plugins and books, and are entitled to free monthly coding support via Slack or Skype.

Dev Log #118

This Dev Log is a little quieter than usual as both Felipe and I were off most of last week. Even so, that didn't stop us releasing Phaser 3.2.1! I've also written another installment of the Migration Guide, which this issue focuses on events and the Game object.

On March 12th we released Phaser 3.2.1. This was another point release, meaning non-invasive API updates and bug fixes only. Updates include new optional arguments for the setFrame method to control resizing of the base Game Object. The ability to restart a Scene by just calling start from its local ScenePlugin. Cameras following Game Objects will now factor in their zoom level in the follow offset. Render Textures now draw any frame, not just the base frame. And many more.

Grab the latest build from GitHub, npm or CDN and as usual see the full Change Log for details.

Phaser 3 Migration Guide - Part 2

Let's continue the major differences between versions 2 and 3 of Phaser. It's been interesting seeing forum posts and comments from people who are trying to do things in v3 the same way they did and v2 and hitting roadblocks, either because the approach they're using no longer exists or just because v2 in-grained so many traits in devs that the removal of them in v3 is proving problematic. Hopefully, part two of this guide will help ease you into both the API and the mindset behind Phaser 3.

Game Over

In v2 nearly everything was handled via the Phaser.Game object. Whenever you did ‘this.add’ within a State you were essentially just communicating with a game level system. The Game instance had a whole raft of systems hanging off of it, everything of any importance was there. All of the examples played upon this too and would often access properties and systems directly off the local game variable. So it’s no wonder that I’ve seen a lot of example code posted in the forum and on Slack that carried on this tradition with v3.

The thing is, you absolutely shouldn’t be doing it that way anymore.

In v3 the Phaser.Game object does indeed still have a few systems attached to it. These are the global managers that it made no sense as having instantiated on a per Scene basis. Examples include the core Input Manager, the Scene Manager, the Texture Manager and the Sound Manager. However, it’s important to understand that these are global systems and most of them are designed to be interfaced with via plugins and not directly. Here’s a good example:

The Input Manager is a global system responsible for the processing of native browser input events and handing them off to the various input systems to deal with. If you use 'game.input' this is the class you'd be talking to. However, there is also an InputPlugin which belongs to the Scene. The plugin is what you interface with whenever you do ‘this.input’ from within a Scene. The Input Manager has hardly any useful (to your game) methods or properties. The Input Plugin has lots. More importantly, it's the plugin that you listen to input events on: **this.input.on('pointerdown')** would simply not work if you did **game.input.on('pointerdown')**. It's important to understand that we created the Scene Plugins to be the systems you interface with, in nearly all cases and the global ones as being for plugins to talk to.

So if you ever find yourself reaching for the ‘game’ crutch, please slap your own wrists and realize it’s no longer how you should interface with those systems. Always go in via the Scene Plugin, as it can have an impact on the execution order of your calls as well as what parts of the API are exposed to you.

Event Horizon

Phaser 2 used what are known as Signals for most of its communication from systems to user-land code. If you’ve ever written code such as **onInputDown.add** then you’ve been interfacing with a Signal. Signals are like Events but have a different internal structure and approach. There’s an interesting article on the subject here and I used to really enjoy using Signals back in the Flash / AS3 days. However, for v3 we didn’t carry this through. There are a few

reasons for this but the primary one is that Phaser is a JavaScript based library, and on the most part, when in Rome you should do like the Romans do. For JavaScript, and the web in general, that means using events.

In v3 we are using the EventEmitter3 library, which is proven to be both insanely fast and flexible enough for our needs during development and beyond. We use it internally for systems to communicate with each other and also lots of systems, and indeed the core GameObject class, all extend from it.

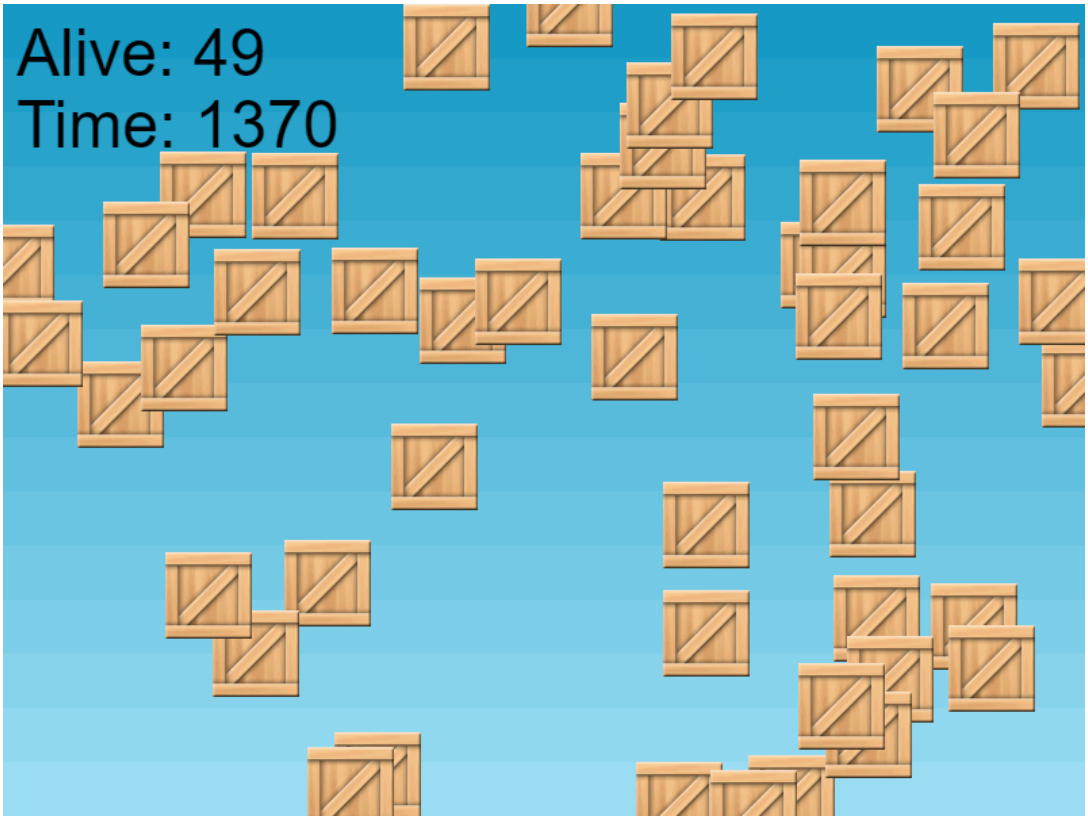
This means you can now write code like:

```
var player = this.add.sprite(x, y, texture);

player.on('hit', this.hitHandler);
player.on('killedMonster', this.killedHandler);

function hitHandler ()
{
  // spawn blood effect
  // play suitably squishy sound fx
}

function killedHandler (monster)
{
  deadDudes++;

  if (monster === Oxygene)
  {
    // add troll bonus
  }
}

// Within a baddie class:

if (player.isAttacking() && hitpoints <= 0)
{
  player.emit('killedMonster', this);
}
```

Which as you can appreciate can be insanely powerful and opens up all kinds of

opportunities for you. You no longer have to create masses of Signal instances and can instead always rely on the fact that a GameObject can both emit and receive events.

Here is an example of a simple 'clicker' game - how many crates can you remove in 10 seconds? It shows how to use both input events and Game Object events.



You can look at the other examples as I recently updated the Labs and removed all the previously broken ones and replaced them with working examples.

There are lots of internal events dispatched by the various systems. For example, the Loader will dispatch a ‘progress’ event which you can listen for in order to display your own progress bars. The Sound Manager dispatches all kinds of events too. We’ve got a lot of these covered in the jsdocs, but by no means all, so it’s an on-going mission to ensure they’re all referenced.

More importantly though, if you feel like we’re missing any events that you’d like to see dispatched, then please open an issue about it on GitHub. The ‘hotter’ an event, the more consideration it will require. By ‘hot’, we mean an event that is likely to be emitted either at a very high frequency or in large numbers (or both!) - those kinds of events take a lot more justification to add as the potential impact performance can be dramatic. Yet if the event is bound to a less frequent activity,

such as the instantiation or destruction of something, or the invocation of a method, then we're happy to consider making them part of the core API. We don't want to add things just for the sake of it, but we appreciate that we didn't consider all use-cases during development.

Coming Next

As mentioned in the previous Dev Log I'm still fighting a monstrous battle with the documentation. Felipe is working flat-out now on Containers, both of which we should be in a position to show you next issue. We will also stay on-top of PRs and issues in GitHub in a run-up to the 3.3.0 release. I won't put a firm date on it but I'd hope it will be out by the 26th of March, if not before.

If you're as massive a Neil Gaiman fan as I am, then you may already know about Sandman Universe. If not, prepare yourself for another trip into the wonderful realm of Dream and the Endless.

A really fascinating and detailed post from a dev team specializing in RPGs on

how the market reacted to their releases over the previous years.

Kartridge is a new PC gaming platform from Kongregate. Let's see if this platform gets enough steam behind it to out-do Steam.

Phaser Releases

Phaser 3.2.1 released March 12th 2018.
Phaser CE 2.10.1 released February 18th 2018.

Please help support Phaser development

Have some news you'd like published? Email support@phaser.io or tweet us.

Missed an issue? Check out the Back Issues page.