Welcome to Issue 112 of Phaser World

It's been a huge week of updates for Phaser 3, which you can read all about in our Dev Log this issue. Lots of fun demos and great new features have landed.

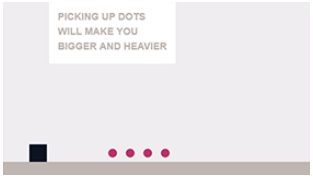
You'll also notice we've got some official Phaser t-shirts on sale too! Sadly only available in the US at the moment (we're working hard to find a European printer) but if you can order them they do look great :)

As is usual this is another long issue, which means lots of it will get cut off in Gmail, so please do "View the full message" if you need to to make sure you don't miss anything fun out!

So, 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 or Twitter.

The Latest Games

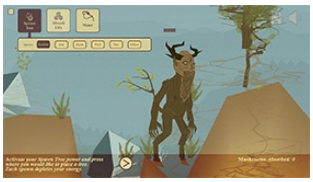
Game of the Week
Cybertank
Can you pilot your tank through the increasingly tricky levels to collect the cubes and escape in this superb action puzzle game.



Staff Pick
grooOW!
A great puzzle platformer where you have to carefully plan how to collect the dots. Because each one you eat makes you grow!

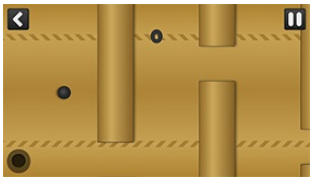
Plumber

Rotate the pipe pieces to complete the puzzle and let the water flow to the other side, in this charming puzzle game.



Arboreal Demigod

Absorb and create life in order to build a forest. Spawn trees, feed animals and manage forest fires in this unique nature themed game.



Tilter

Tilt your device to roll the ball and get it through as many incoming obstacles as you can.

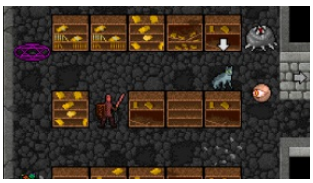
What's New?

Official Phaser T-Shirts

A new range of Phaser T-Shirts are now available from Amazon in the US. Available in 5 different colors with male and female fits, look the part at your next game jam or meet-up!

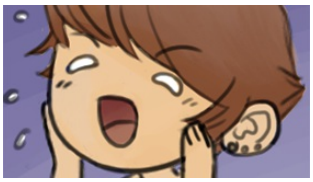
Solve Sudoku Techniques

Learn how to solve Sudoku puzzles in lightning fast times with this interactive app and web site.



Slashware 2017 Rewind

A look back at the games released by Slashware Interactive in 2017 including the Ultima 6 game engine OpenArthurianX6.



Phaser Starter

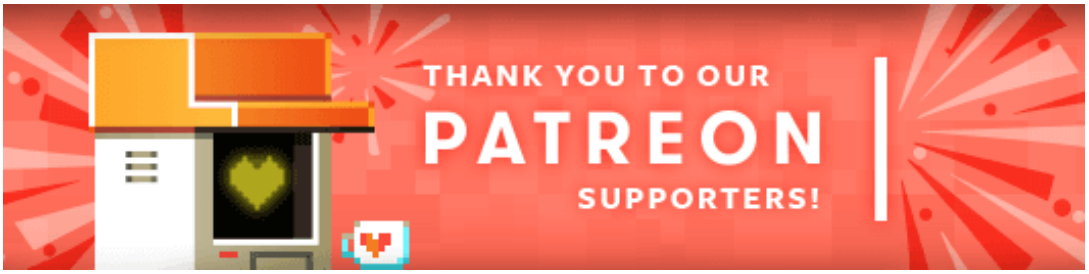
A starter Phaser project template that loads assets from JSON, served with Express and tested with Mocha.

5 Time Saving Phaser Classes

A set of five time saving classes to use in your own games to handle things like backgrounds, logos and thumbnails.

Tipsy Tower Prototype Part 2

In the second part of the series an effective camera zoom is added allowing you to stack the crates higher and higher.



Thank you to our awesome new Phaser Patron **Steven Landman**

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 #112

Welcome to Dev Log 112. It's been a very busy week and lots of changes have happened. First of all we just published Phaser 3.0 Beta 18 to npm and GitHub.

As usual, all examples this issue were built against it.

Structural Changes

In prep for the 3.0 launch we have re-structured the Phaser repository massively. This entailed removing the v2 folder and then moving all of the contents of the v3 folder into the root and generally tidying up lots of stuff. It resulted in a commit that looked like this:

Yup, +13 lines of code and minus *over 2 million*. Trust me, I have never been so scared to merge a commit in all my life :) It was a very strange feeling seeing years and years of work on v2 vanish before my eyes. I mean, I know it's still there - all the releases are there, you can browse the repo using any previous tag you like, and of course, Phaser CE exists too. Even so, it was quite a moment.

The new structure also fixes the problem with the npm packages where you had to reference Phaser from deep within the node_modules folder. Now you can simply import 'phaser' and you're done. Of course, this brings in the *whole* of Phaser, including every physics engine and extra little package, but it's trivial to create your own builds and I'll cover that in a future issue.

Class War

We've recently had a few issues opened asking why we're using our own custom Class and not ES6 classes. To be honest the more recent issues have been getting a little abusive and it's starting to grate. So I'll explain once and for all and be done with it.

Phaser 3 is written in ES5. All 56,000+ lines of it. We have every intention of migrating to ES6 later this year, but not now and certainly not this close to launch. In order to help both devs and ourselves, we use our own implementation of Class. It doesn't do much more than the old prototype method of creating classes that we used in v2, but it's tidier and gives us more structure internally. It's basically a bit of syntactic sugar that makes our lives easier. It's also exposed in the API so those of you who code in ES5 and want to use it can do so if you

wish.

In order to use ES6 classes instead, we would have to introduce babel into our workflow and build process. This is not something I'm willing to do when we're literally a few weeks out from release. The impact would not be insignificant for what is ultimately zero actual benefit.

There appears to be a misunderstanding that because we use our own Class format that means you can't then use ES6 classes in *your code*. This is completely untrue. If you want to use ES6 to make your games, go ahead, it works perfectly. You can use native classes that extend any Phaser objects without any problems at all. We know because we tested it months ago and it's still true today (because I re-tested it last week.)

When we swap to ES6 we will do it properly. I'm very much looking forward to using native classes, destructured objects, and default argument options. But there is a time and a place for doing this, and it absolutely isn't now.

Renderer Internal Cleanup

Felipe writes: Last week I spent it up cleaning the rendering side of V3. There was a lot of baggage still in there from the beginnings of the development. This cleanup is not just removing unused variables or code snippets it also about making it scalable and flexible enough to allow future extensions.

One of these changes is adding the concept of a pipeline. A pipeline in the Phaser renderer is how you draw something on the screen. This is not fixed for 2D it can even be 3D or for offloading computational tasks to the GPU. For example, the SpriteBatch and BlitterBatch now extend Pipeline. This means they both now share a common interface for interacting with WebGL. This will allow for easier implementation of future ways of rendering and possibly mixing 2D and 3D. You can follow the progress of this cleanup on the branch rendering-cleanup in GitHub.

Sound Manager Update

Pavle talks about what's new in the audio side of Phaser 3:

The final few bits are in place and Web Audio API implementation is done!

One of the last big features was adding support for various events related to sound. You can now listen for when one of the sound settings' value changes, a

playback controlling method gets called or when the sound has ended or looped. Here is an example that demonstrates basic playback options and uses events extensively so you can easily get familiar with how to use them:

Audio events

In v3 there is no more allowMultiple property on a sound object since each sound object controls playback of only one audio instance or a certain part of it.

But for making things more convenient when you only want to play sounds on a fly and not worry about keeping a reference to the sound object and disposing of it correctly after it has finished playing, I added two methods to the sound manager that allow you to do exactly that.

SoundManager#play method allows you to play sound with a specific key and also pass in a config or sound marker object to have more control over the playback.

SoundManager#playAudioSprite allows you to do the same only with audio sprite.

I've also updated two examples to make use of these methods:

Markers demo

Plus this example.

And finally I've started working on HTML5 Audio implementation of the Sound API and that will be my primary focus, along with writing up all the docs, these last few weeks before the v3 release.

One certain thing is that it won't be able to match timing accuracy of Web Audio API implementation but everything else will be on the same level and one special thing is that you will be able to play multiple instances of a sound, when using HTML5 Audio implementation, which you weren't able to do in v2!

It's a Big Event

You know how things go sometimes when you try to start on one task, which makes you realize that in order to complete it, you need to do something else, which in turn highlights there was another task reliant on that change and it all just merges into one giant update. That is what happened last week. I had been planning out how the Scene systems would work. The idea is that the Scene itself will emit events each frame, which the systems (and your own plugins) will respond to. Events like 'boot', 'preUpdate', 'preRender' and so on.

While getting ready to implement this I found some issues in the Event

Dispatcher we were using. Using events in the core game loop means that the event system has to be as absolutely fast as it can possibly be, or it slows everything down. I did a bit of research and came across EventEmitter3, which has recently had a nice update and looked perfect for the task at hand.

After some profiling tests we made the decision to move to using it. This meant two significant things: 1) I had to remove all of the Event classes through-out the codebase and swap to using strings and arguments instead, and 2) It would break all examples that used events (which is a lot.)

Still, the results were worth the change. Incidentally, I since found out that Pixi 4 uses a slightly older version of EventEmitter3 in all of its hot loops, dispatching 'render' events and so on, so if it's fast enough for them, it'll be perfectly fine for our needs too.

It took a lot of refactoring but the change was made and it had some really nice benefits. For a start, I was able to delete lots of Event class files, so our bundle size went down. It was no longer creating Event objects at run-time (as EE3 uses arguments instead) so it was causing less gc and we also made the decision to allow all Game Objects to extend it too, allowing them to emit and listen for events. I updated the Input system to reflect this, which means you can now do things like this:

```
var sprite = this.add.sprite(400, 300, 'eye').setInteractive();

sprite.on('pointerdown', function () {

  sprite.setTint(Math.random() * 16000000);

})
```

Or this ...

```
player.on('dead', this.goToDeathScene);

// stuff happens ...

player.emit('dead');
```

This will allow you to keep your game code nice and clean, with key Game Objects emitting events directly and your game responding to them.

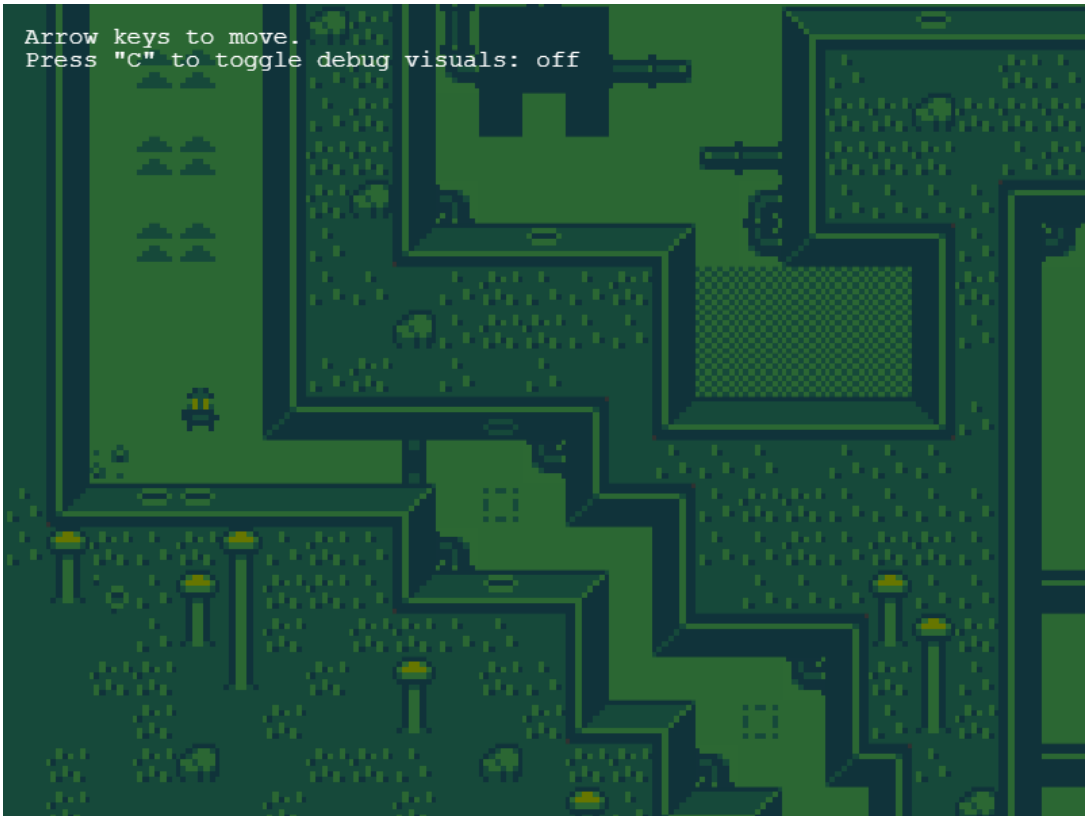
The Input Manager is hooked into Game Objects so it will emit events directly on them, as well as globally. We've also made the required changes to the Sound Manager, Physics and elsewhere in the API. As I said though, this broke lots of examples and there wasn't time to fix them before this newsletter went out, but we'll do so this week. Just be aware of this change if testing things out, please.

Tilemap Collision

Thankfully Michael had some extra time available last week and added a really important feature:

"This week, I've been integrating the Tilemap API with physics. Arcade Physics is hooked up and ready to be used in the latest beta. If you've been using v3 tilemaps, I'm betting this is something you've been waiting for.

The first example demos exploring a top-down world:



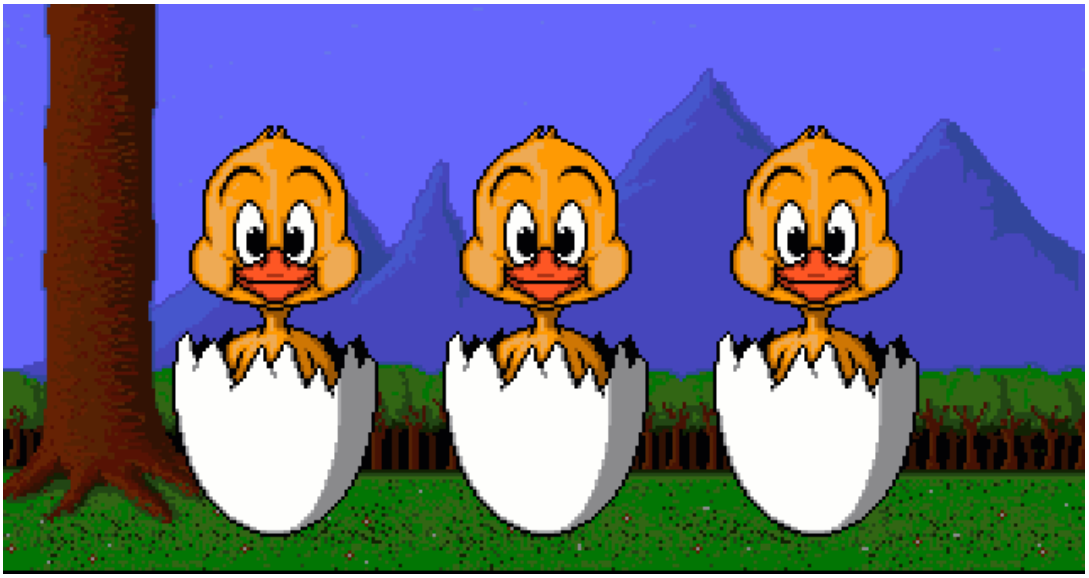
Arrows to move

The second example demos a platformer, using tile callbacks to let the player pick up coins. You can use map.setTileIndexCallback(…) and map.setTileLocationCallback(…) to set up collision callbacks for specific indices or locations within a map. (These methods match the v2 API.)

Platformer example

Thanks for this essential update Mike :) Hope you all have fun playing with it!

As you can see it was quite the week! Lots of changes, lots of improvements and things are getting really exciting now. I just wanted to end by sharing something I made. It's a re-creation of a scene from the classic Amiga Budbrains Mega Demo, which some of you may be old enough to remember. I've wanted to re-create this in Phaser for years now and using V3 I got the chance. The code is really simple, just a few tweens, an animation json file and Pavle's awesome sound API :) Take it away birdies ...



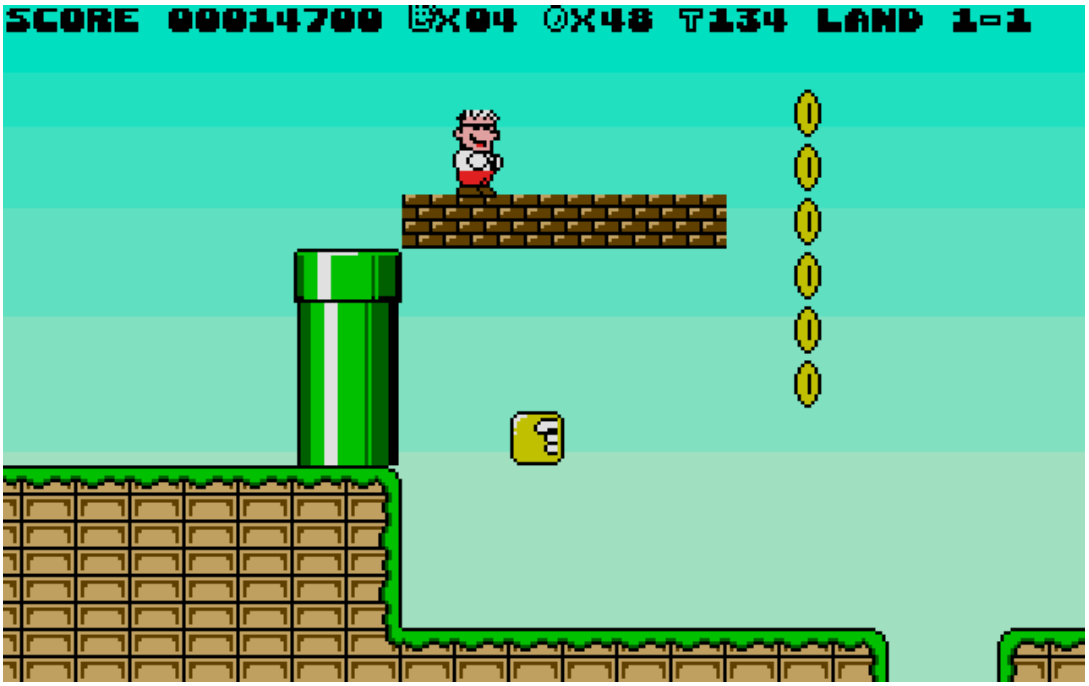
Until next week, have fun coding!

Phaser 3 Labs

Phaser 3 Beta 18 is out and ready for testing.

Visit the Phaser 3 Labs to view the API structure in depth, read the FAQ, previous Developer Logs and contribution guides.

You can also join the Phaser 3 Google Group or post to the Phaser 3 Forum - we'd love to hear from you!



Super Stario Land was a Mario clone for the Atari ST, which I did the graphics for back in 1995. Read my comments in the Facts section for details!

The topic every game dev is talking about behind closed doors: The cost of doing business.

WebKit has an excellent technical breakdown of what the Spectre and Meltdown flaws mean for webkit and browsers in general.

Phaser Releases

Phaser CE 2.9.4 released December 20th 2017.
Phaser 3 Beta 18 released January 15th 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.