Welcome to Issue 97 of Phaser World

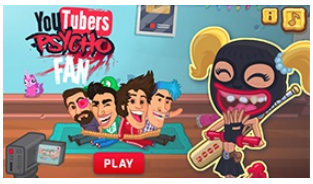
And we're back with another packed issue! New releases of both Phaser CE and Phaser 3 land this issue, as well as lots of details in the Dev Log. There are some

great games, even more tutorials and a massive sale on our books and plugins. Dig in and get creating!

So, until the next issue, keep on coding. Drop me a line if you've got any news you'd like featured. You can reply to this email or grab me on the Phaser Slack or Discord channels.

All of our own plugins and books are now 50% off or more For example, the Phaser Mega Bundle pack is now just $50, Interphase is now just $15 and Particle Storm is now just $20. Please visit the shop page for details.
Note that this excludes our affiliate products.

The Latest Games



Game of the Week
YouTuber's Psycho Fan

How good are you at laying into a bunch of YouTubers? as you knock money and subscribers out of them.

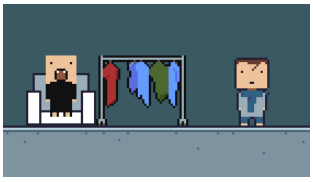
Staff Pick

Moorhuhn Shooter

See how many crazy chickens you can shoot in this remake of the popular 2000s game.

Mini Brain Doctor

Minions are crazy about experimenting in the lab, but now one of them needs your doctor skills for a brain surgery!



Let's go to the Mall

An endless runner in which you have to see how many shopping bags you can carry!

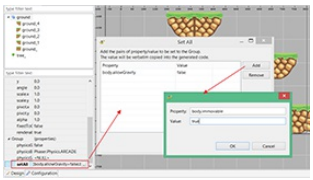
The Floorsland Reborn

A story driven cross between an RPG and a platformer.

What's New?

Phaser CE v2.8.7 Released

The community edition of Phaser CE includes fixes for TypeScript defs and particle emitter input signals.

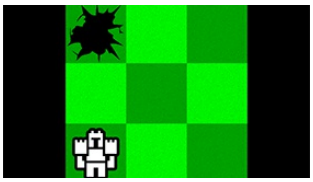


Phaser Editor v1.4.2 Released

The ultimate Phaser IDE gets a new update including support for HiDPI monitors, new Scene Editor features and JavaScript 6.

Multiplayer Game Tutorial Part 1

How to create a multiplayer game with Phaser, Node, Socket.io and Express.

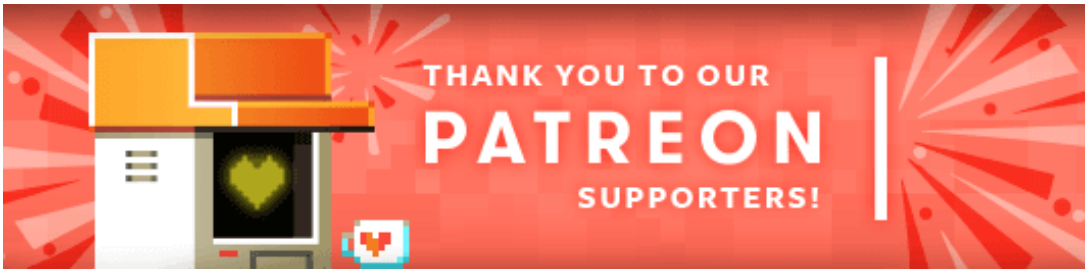


Flipping Legend Prototype Tutorial 2

In the second part of the tutorial series deadly holes are added into play.

Phaser CE v2.8.6 Released

The community edition of Phaser CE includes fixes across the input classes and new documentation updates.



Welcome and a massive thank you to the new Phaser Patreons who joined us this week: **Daniel Snd**, **Danny Page** and **Levi Meahan**.

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 #97

Welcome to another Dev Log. First of all in Phaser CE land we've got the 2.8.7 release. There were a few issues with TypeScript defs and Signals in 2.8.6 which the community quickly resolved, hence the new version. You can find the full

details in the ChangeLog like usual.

Phaser 3 Beta 2 + Weekly Releases

Things are changing fast around here. We're adding new features, tweaking the API and fixing bugs literally every day now. So rather than wait for a 'large' release I'm going to push new Beta versions out every week. To that end, Phaser 3 Beta 2 is now available.

Yes, there are still several large sections missing from the API, because we're not at that point yet in the schedule, but I still feel these are valid beta versions in that all the features that are present are worth testing in earnest, as they won't be going through any drastic revisions. We're still making lots of internal changes but the public API hasn't altered that much in a while now.

Documentation Update

I spent the start of the week testing every documentation generation I could get my hands on. JSDoc3, documentation.js, Sphinx, you name it, I tried it. And none of them worked as required. This is due to a combination of me needing some specific output option support and also due to the way we have internally structured v3. For example, I will often break large classes down so that the methods sit inside their own files in an 'inc' folder. This is to primarily to keep things tidy and also allows for us to swap out methods easily without changing whole classes. It also means that if you wanted to tweak a class you could literally make your changes to a single file and only that method is impacted, rather than the entire thing.

However, none of the doc generators out there like this one bit. I was determined not to have to write my own either though, so instead, I've gone for the middle ground. I'm now using a couple of npm packages: extract-comments, to suck all the jsdocs out of the v3 source, and doctrine, to parse the docblocks and get an AST back with all of the tags in. Doctrine nicely handles everything I've thrown at it so far and is actually the parser used internally by documentation.js. What this means is that the jsdoc scraping, extracting and tree building is now working nicely. Leaving me to have to write the generator that will take this data and spit it out into the formats we require (which are HTML, markdown, and JSON).

Because I'm going to write this generator by hand it means I can use my own custom jsdoc tags for things like extended tutorials and most importantly TypeScript defs. In lots of cases we can infer the types from the tags, but not all, and this will allow me to mark-up the more troublesome objects so we can export TS defs directly and not have to manually maintain them. So there's still a bit of

work to be done here in terms of creating the output - but at least I didn't try to reinvent the wheel with the most complex part, namely the extraction and parsing, and am just using tried and tested tools for that.

New Build Flags

From the start, we've been careful to ensure that all of the rendering functions are split out into their own files. Game Objects, for example, have a file that handles their Canvas rendering and one for WebGL. The thought being that this would allow for specific builds of v3 with just one renderer inside and literally none of the un-needed functions would be present.

I've now added this feature to v3. Taking advantage of webpack definitions we're able to signal to webpack that a block of code should only be included if a specific flag, or combination of flags, are set. This means you can now do a 'canvas only' build of v3 and literally none of the WebGL functions are present, at any level. This saves masses of space in the build file as you can appreciate. For example in a Canvas only build we don't include Game Objects such as the Effect Layer, Render Pass or Light Layer because they are WebGL only.

We can also use build flags to gate-off features that are in testing and shouldn't make it into production yet. Another change is that the entry point file now controls almost everything that is added to the library. If you don't require any of the extra math functions then they can now be easily excluded allowing for real 'minimum' build sizes. The same goes for Game Objects too. There's no fear of accidentally breaking a class by excluding something it may need either because classes all require the specific functions they need directly and never assume them to be available within the Phaser namespace. It's a great feeling to have this in place and working and will truly allow you create whatever package you want.

Dynamic Factory

As well as the build flags I also worked heavily on refactoring the Game Object Factory. This is the class you interact with whenever you call 'this.add' from within your Scene. Although all of the Game Objects have their own factory functions, the GOF used to basically be a giant list of functions pointing to each Game Object, i.e. 'add.sprite', 'add.graphics', 'add.tilesprite', etc. But this didn't fit in with the new build system - after all, if you exclude the Graphics Game Object then you should no longer be able to call 'add.graphics'. So I reworked this area and now the Game Objects register themselves with the Game Object Factory, adding their own unique 'shortcut' in. So if you exclude Graphics, 'add.graphics'

will no longer even be available to you, which is how it should be.

Another nice side-effect is that you can now dynamically load Game Objects at *run-time* too. This is great because I think it'll open up a whole new "market" alongside plugin authors. Now you can download and try out brand new Game Objects without ever having to build Phaser for yourself or change the source in any way.

Vector Math

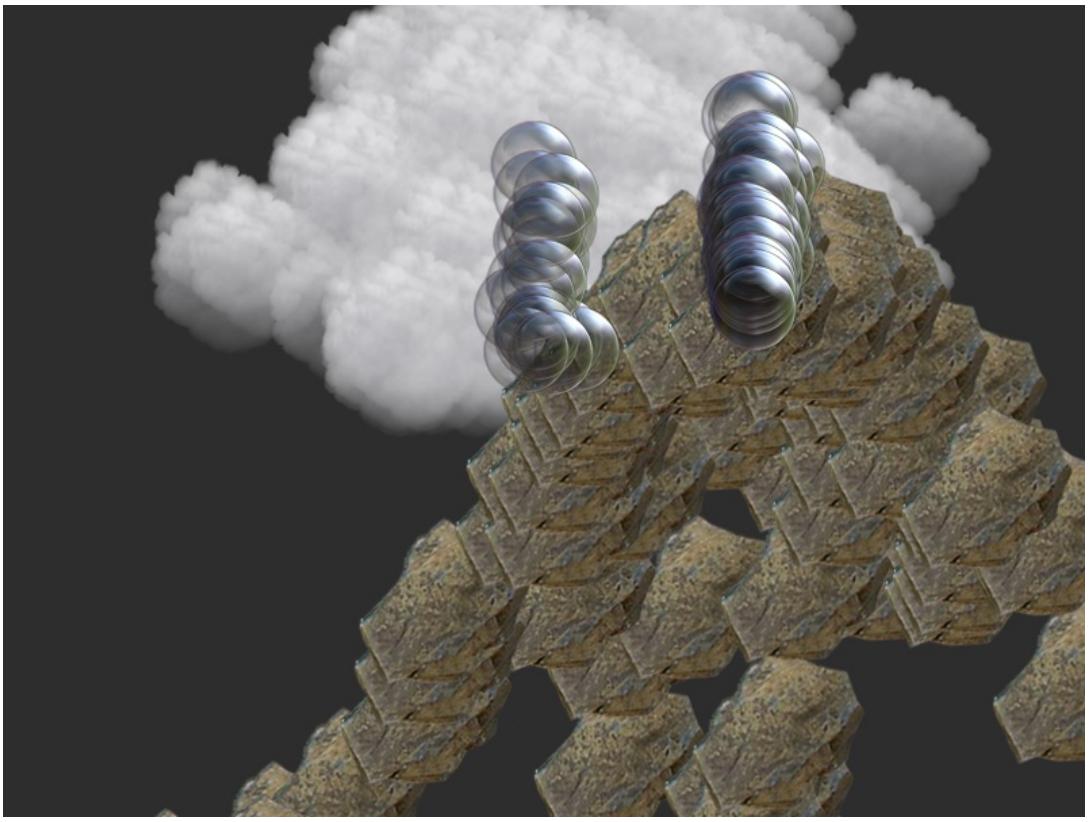
I also spent a day finishing off the Math functions which included adding in all of the Vector classes. Phaser 2 didn't have any support for Vector data types, instead, we used and abused the poor Point class in ways a Point really never should have been. In V3 we now have native Vector2, Vector3, Vector4, Matrix3, Matrix4 and Quaternion classes. These closely follow the best practices laid down in the gl-matrix library, such as unrolled loops and a careful consideration of performance. It feels great to have them in place and we can make good use of them across the revised Arcade Physics API, the Particle system and elsewhere.

Looking towards a 3D future

While Phaser 3 will be entirely 2D in nature I do not wish to cut myself off from going 3D in the future. When we added depth sorting support to the renderer we used the 'z' property, so you change a Sprites z property and it would shift position in the display list. However, x, y, z and w are the 4 common positioning components required for 3D space. So we made the decision to rename 'z' to 'depth'. You can now set the depth of a Game Object by changing its depth property or calling setDepth directly. We have also added in the z and w properties, reserved for future use in a 3D environment and even useful now for things like embedding particle position data.

Particle Emitter

Felipe has been working on the new Particle Emitter for v3. It's still early days but the core Game Object is in Beta 2 already and there's an example to play with too. It has basic particle rendering, particle movement and a life cycle and pooling. This week we'll add interpolation effects so you can tint, alpha, rotate and scale the particles, allowing for nice effects like fire, bubbles, stars, etc very easily. For now, have a play with the demo below. Sorry that it's the only eye candy this issue, but I have something neat lined-up for issue 98 :)



Drag the mouse around

Phaser 3 Labs

Visit the Phaser 3 Labs to view the new 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!

This is an incredible read all about the Complete history of Nintendo's arcade games. In-depth and packed full of information you probably never knew!

Check out the new trailer for the Early Man film - created by my old workmates at Aardman and looking great :)

Read about the 6 best new features in npm 5.

Further Reading ...

Phaser Facebook Group
GameDev.js Weekly Newsletter
HTML5 Game Development
Lostcast

Phaser Releases

Phaser CE 2.8.7 released September 14th 2017.
Phaser 3 Beta 2 released September 18th 2017.

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.