Welcome to Issue 103 of Phaser World

Happy (almost) Halloween! We've a particularly suitable Game of the Week this issue, details of the new GitHub Game Off jam, a beautiful mahjong puzzle game and loads more. The Dev Log is back to usual and packed full of interesting updates to Phaser 3 and some direction for where we're heading.

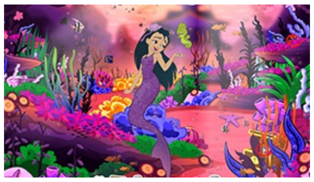
Thank you to everyone who responded to my Dev Log last week about helping with the Phaser News and dev work. I'll be contacting everyone this week! The response was massive, so I'm sorry if you don't get a chance to work with us this time around. I really appreciate you offering all the same.

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.

The Latest Games

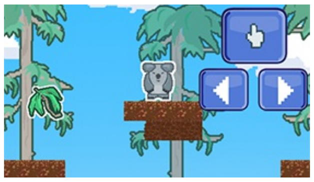
Game of the Week
Frog Grog
Moth wings, eyeballs and hair of a witch, bottles and flasks could make you rich! in this stunning slots game.

Staff Pick
Animal Connection
Test your pattern detection skills over 30 stages and 3 game modes in this cute animal matching puzzler.



Mermaidens Hidden Object
Come and meet the Fin Fun Royal Mermaiden Princesses in their seadoms! Can you help find the Mermaidens' lost items?

Dark Souls Casual Jigsaw
A nice full-browser jigsaw game with random images and various game modes.

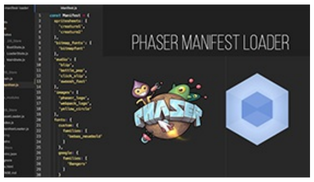


Koala Rumble
You have 15 seconds to gather all of the eucalyptus leaves!

What's New?

Game Off
The 5th annual GitHub game jam starts in a few days time - get involved and

check out our Phaser resources.

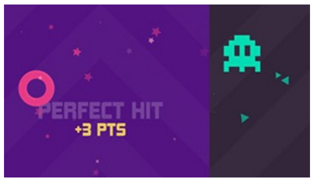


Phaser Manifest Loader

A new update to this essential game asset loader for webpack.

Message Box Tutorial

A tutorial on adding a configurable pop-up message box into your game.



Boom Dots Source Code

Emanuele takes his old Boom Dots game and updates it for Phaser CE 2.9.1 and releases the full commented code.

Creating an Endless Runner Part 5

The fifth part of the endless runner tutorial series adds a baddie to hit and a Game Over screen and logic.

Welcome and a massive thank you to the new Phaser Patreon who joined us this week: **Jonas Luz Jr.**.

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 #103

First things first, Phaser 3 Beta 8 is now released. You can grab it from GitHub pre-built, or from npm using the beta tag. It includes all of the work with jsdocs from a few issues ago and all of the more recent work enhancing the Particle system, which I'll cover in more depth this issue.

Secondly, *thank you* to everyone who emailed me offering to help with clearing the v3 tasks backlog! I've had an overwhelming number of responses. Now that Felipe is back from holiday we can go through them all towards the end of this week and make some decisions.

Particle Manager Updates

A few issues ago I demonstrated the new particle renderer we've added into v3. Over the past few weeks, I've steadily been working on making it feature complete. The renderer was always solid but there was a lot more I wanted it to be able to do. In this dev log I'll run through some of the changes that have been made and what you can now do as a result.

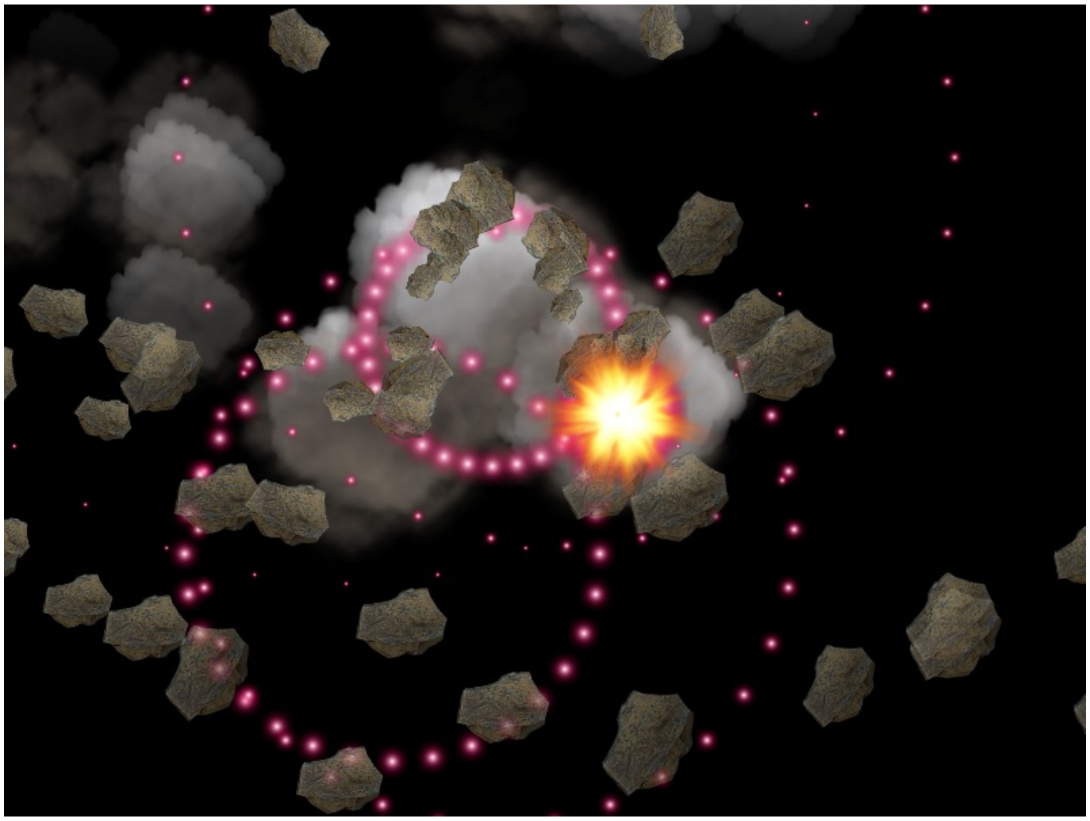
Batched Emitters

In the previous version, the renderer would flush the batch after each emitter, even if it was using the same texture as the previous emitter. You were also limited to using just one frame from a texture atlas. I rewrote this part of the renderer and it's now a lot more versatile. Emitters can use *any* frame found in the given texture. So if an emitter is using a texture atlas it can emit particles using any frame within the atlas. It will also now group together the batches from consecutive emitters. So if you've 5 emitters in a row that all use the same texture it won't flush the batch until the 5th one (or if you exceed the particle count). This means that what would have taken 5 draw calls (in this instance) now only takes 1.

One Manager to rule them all

Previously the API exposed the ability to create a single emitter. The emitter itself sat on the display list and managed its own properties and batch of particles. This is great for simple games but when you need to do more complex effects it meant you end up having to deal with a mass of emitters. This got even more complex should you need to do an effect that relied on multiple emitters all running in sync. It had to be made easier.

I restructured things internally so that there is now a Particle Manager which can create any number of emitters. Each emitter still maintains its own set of particles and related properties. However, you can sequence the emitters within the manager as needed. They can be ordered as in a display list and you can emit from the manager itself, meaning it will invoke an emit across all active emitters at the same time. This is hard to explain, easier to demonstrate:

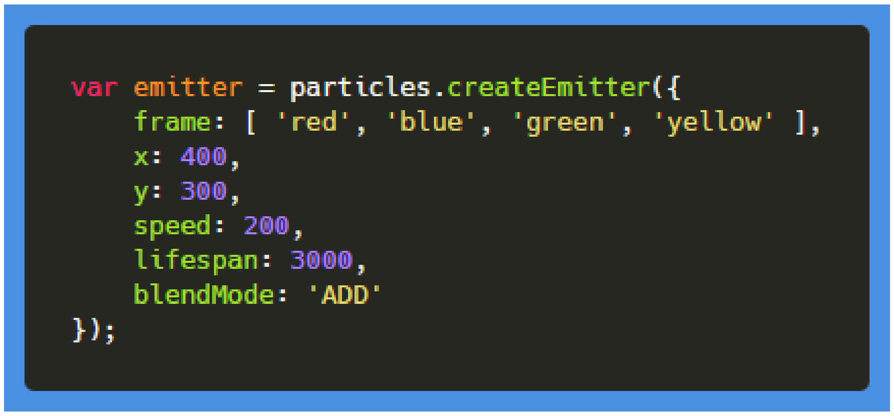

Click to explode

When the example loads just click anywhere. You'll notice it creates an explosion effect. This consists of 4 separate emitters: the smoke puffs, the red circle burst, the stone debris and the fire flash. By calling 'emitAt' on the manager itself it triggers all of these emitters at the same time, causing the result you see on screen. Each emitter is responsible for looking after its own type of particle and their respective timelines.

This new structure also means you will have less top-level display objects to manage. You could now create a particle manager specifically for 'alien bullets' that handles all of the bullets for every alien in the scene, rather than one emitter per alien.

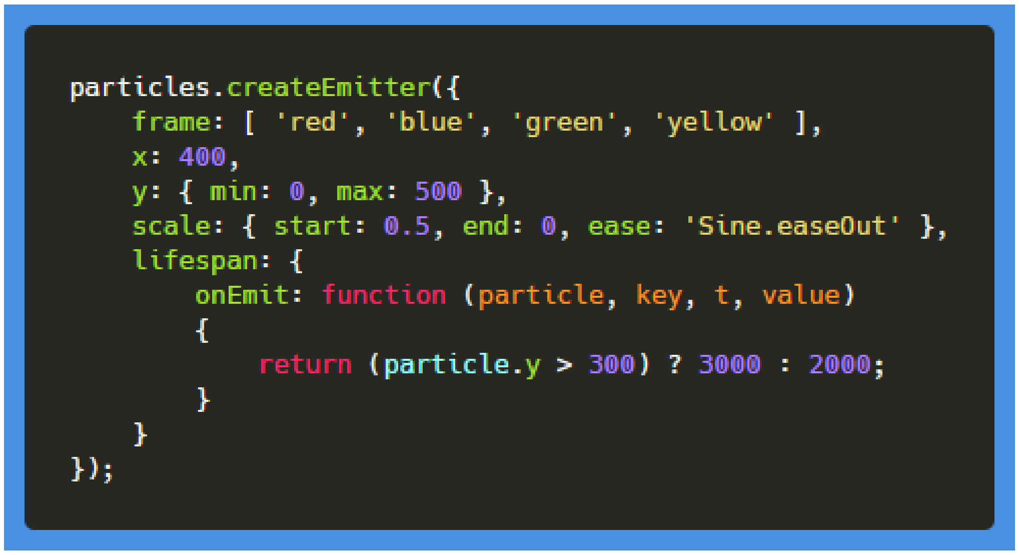
Emitter Config Objects

Previously you could only change the properties of an emitter by calling setter functions directly on an emitter instance. While this works great (and still does) I wanted to bring particles in-line with the rest of v3, meaning you could configure them from config / JSON objects. This is now complete and working and you'll see most of the examples use this method. Here's an example of a nice and simple config:



```
var emitter = particles.createEmitter({
    frame: [ 'red', 'blue', 'green', 'yellow' ],
    x: 400,
    y: 300,
    speed: 200,
    lifespan: 3000,
    blendMode: 'ADD'
});
```

Config objects also allow us to support different types of special values for the properties:



```
particles.createEmitter({
    frame: [ 'red', 'blue', 'green', 'yellow' ],
    x: 400,
    y: { min: 0, max: 500 },
    scale: { start: 0.5, end: 0, ease: 'Sine.easeOut' },
    lifespan: {
        onEmit: function (particle, key, t, value)
        {
            return (particle.y > 300) ? 3000 : 2000;
        }
    }
});
```

In the above config, there are several different types of property being used. First the 'frame' property with the array. Using an array in this way means "randomly pick one element from this array and use that value". So it will randomly pick one of the four different frames when the particle is emitted.

The x property has a direct numeric value (400), so this is used as-is.

The y property has a min-max pair. This means "pick a random number between min and max".

The scale property is using an eased range. Scale has a value at the start of the particles life but it can also ease over the duration of the lifespan. Here the particle scale will start at 0.5 and then using the Sine out ease it'll reduce down to a scale of zero during its life.

Finally, the lifespan property is using a custom onEmit function. The value this function returns is used as the property, so in this case, the lifespan will be either 3000 or 2000 ms depending if the particle is above or below 300px on the screen.

Using a variety of config settings like this allows you to be far more expressive in the creation of your emitters. And we're not done yet ...

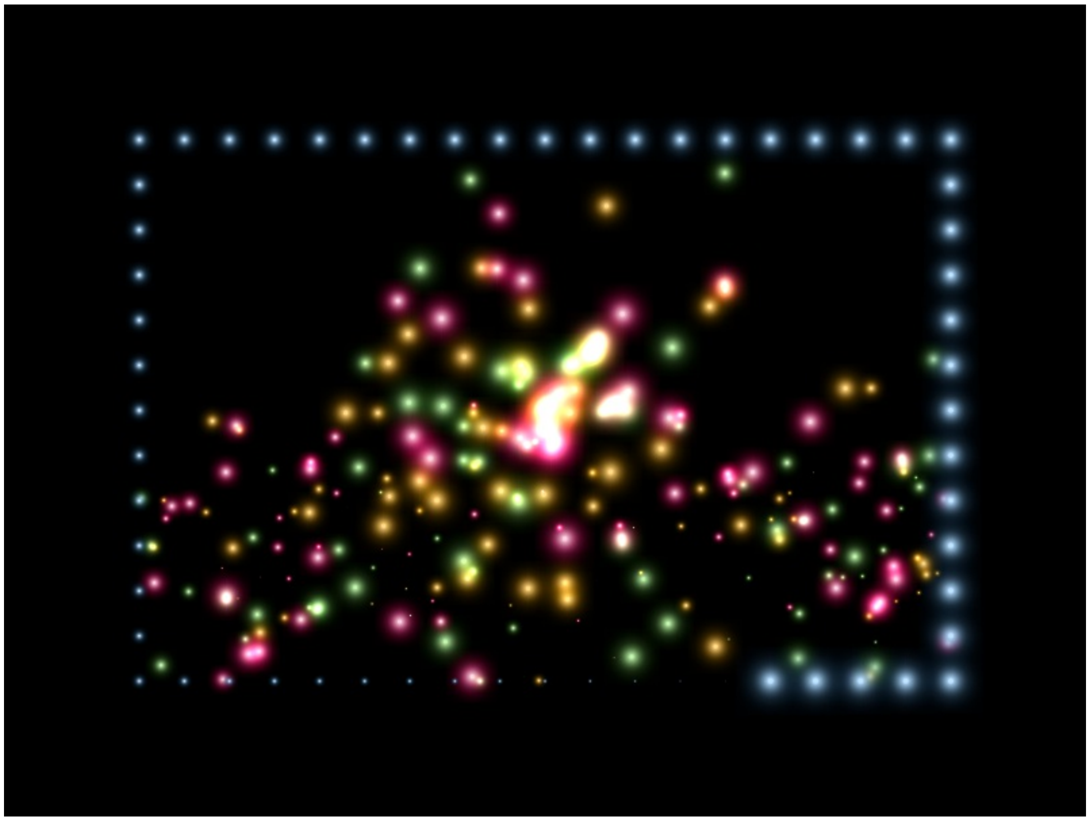
New Features: Acceleration and Bounce

You've always been able to set the velocity of the particles, but sometimes you need a different kind of effect where the speed ramps up over time. So we've implemented acceleration into the emitter. You can now set the accelerationX/Y properties in the config and it allows your particles to build-up speed over time:

It's a particle accelerator!

Because acceleration is steadily applied to the velocity we needed a way to cap it, so we also added maxVelocityX/Y properties to the emitter as well. These are set by default but can be overridden to ensure your effects don't get out of hand.

One benefit that particles in Phaser 2 had was that each particle was an Arcade Physics body. This, of course, meant they were hideously expensive in terms of processing. It also meant they could do cool things like bounce! It's an effect that is often needed with particles and I didn't want to lose it entirely, so I implemented an optional 'bounds' for the emitter and the ability for particles to bounce around within it:

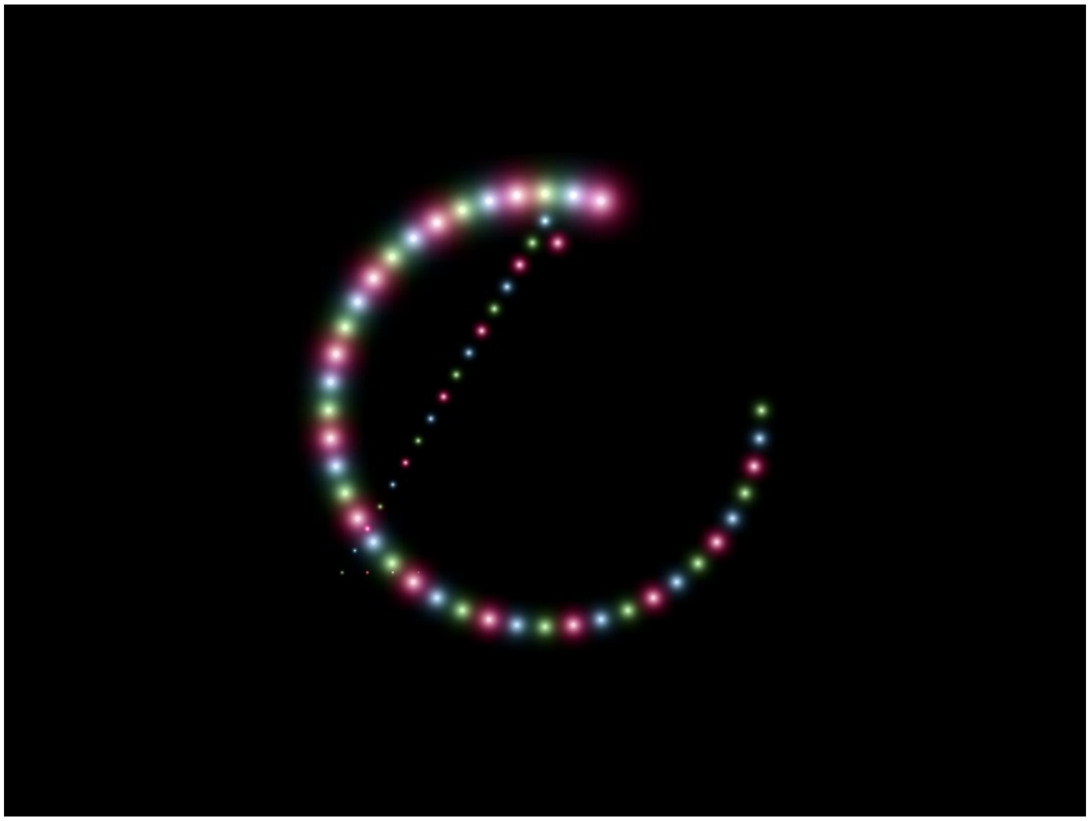

Weeeee!

The emitter bounds is a rectangle object and can be positioned anywhere in the game world. There is a new 'bounce' property in the config that controls the amount of rebound a particle has if it hits the bounds, and that's pretty much it. Using acceleration, maxVelocity, bounce and gravity allows you to re-create pretty much any effect you could previously do with Phaser 2, and a load more that weren't previously possible. You can even define which sides of the bounds should collide, allowing you to make just 'floors' for particles to land on.

New Feature: Emit Zones and Death Zones

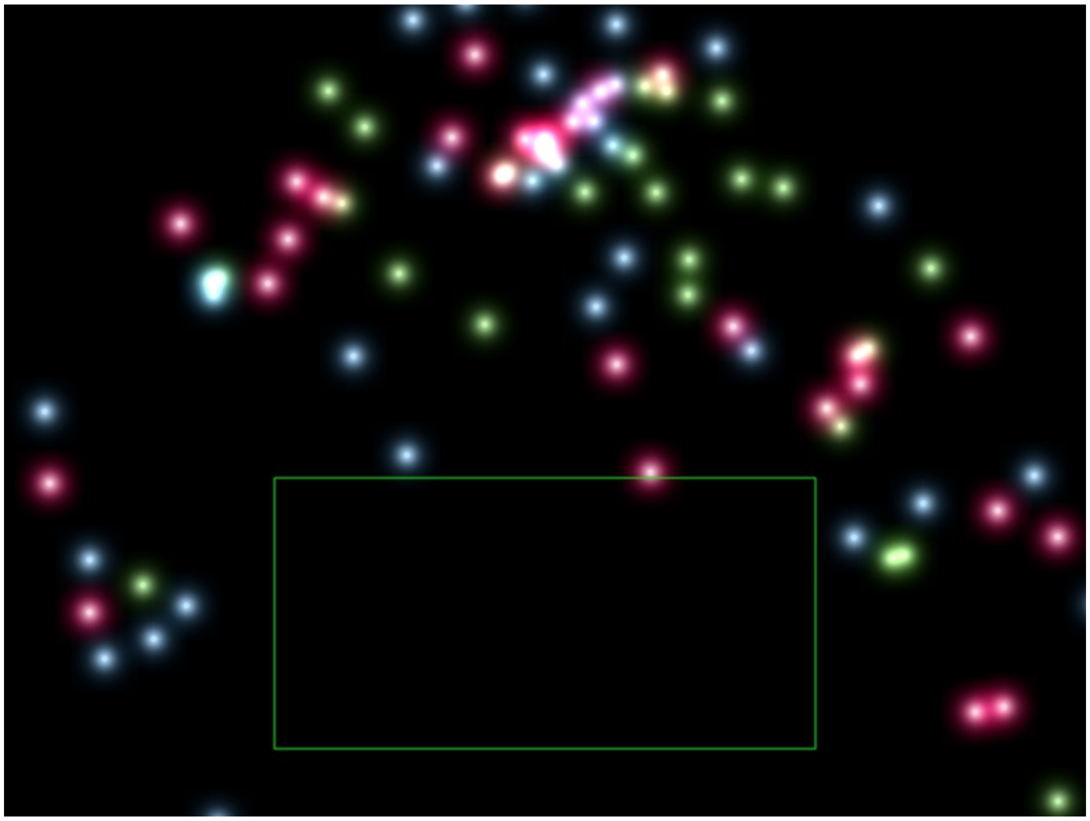
Zones were in the previous beta of V3 but they only worked in one way: You'd define a geometry shape and then it would pick a random point within the shape to emit the particle from.

Zones are now much more capable and easy for you to extend with your own logic. We've still got a RandomZone in there but I also added an EdgeZone and a DeathZone. An EdgeZone works by taking a source object, which can be geometry, such as a Circle or Ellipse, or a Curve or Path, or your own custom class, and then allows you to emit particles from around the edge of it. You can also control how many 'steps' there are around the edges too. Again, it's one of those things easier to show than describe:

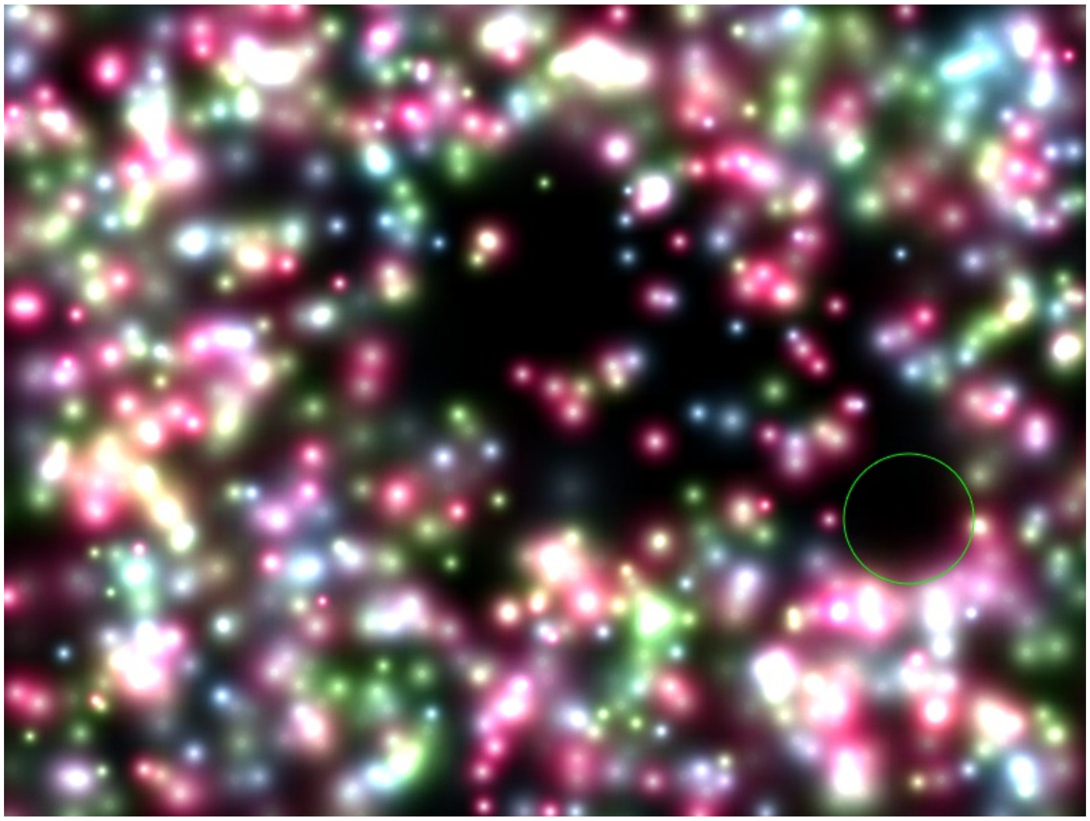

Click to change shape

Notice how the particles are aligned around the edges of the geometry shapes? If you click you'll see the shape change through 5 different types, to get an idea of what's possible. You can even set the 'yoyo' property to true in the emitZone object, which means the particles will emit in one direction, then the other, which creates some wonderful effects.

A Death Zone is the opposite of an Emit Zone. Instead of controlling where a particle is born, it controls where the particle will die. Death Zones again can be geometry objects, or your own classes, as long as they implement a function called 'willKill' that takes a particle as its argument and returns a boolean then you can use them. Death Zones come in two flavors: 1) Kill the particle as soon as it enters the zone or 2) Kill the particle as soon as it leaves the zone. Here is a rectangle acting as a death zone. Any particles unfortunate enough to stray into it are instantly killed:


Dead on arrival

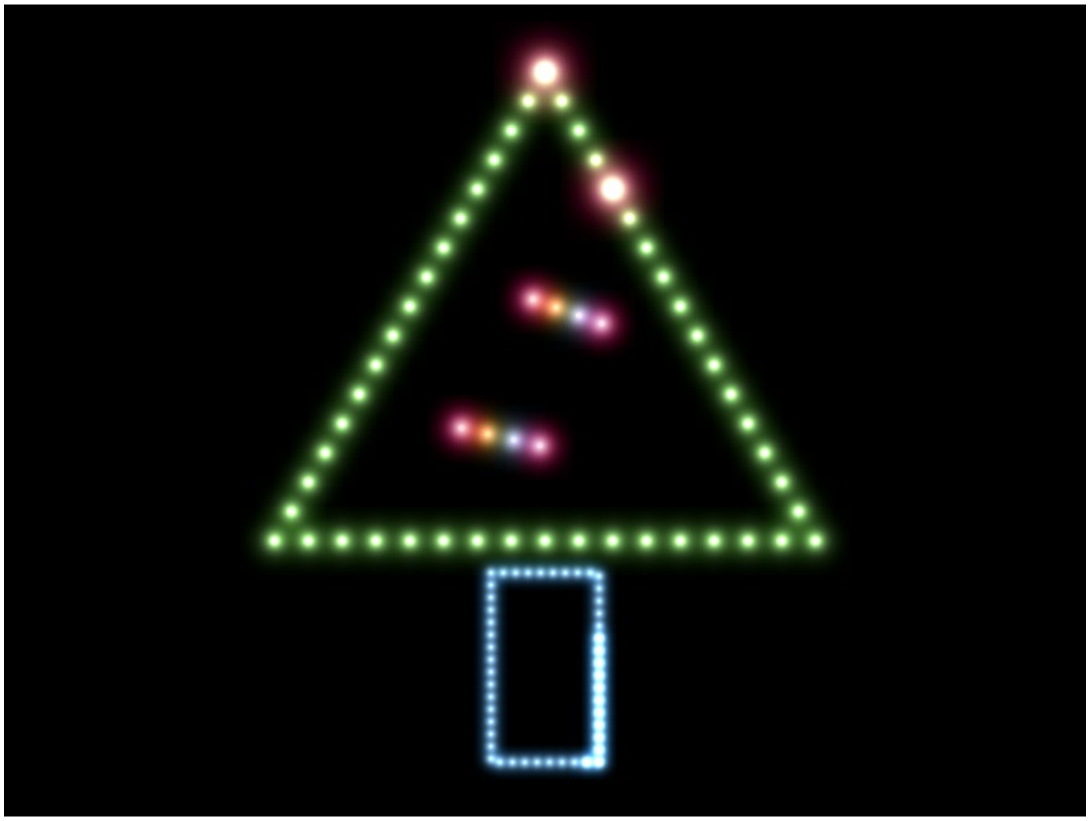
Because the zone is just an object, if you move it, it changes things immediately. Here's an example where you can move the mouse around and try your best to wipe away all the particles!



Gotta nuke 'em all

As you can appreciate the opposite of this is a Death Zone where the particles die if they leave the zone. I won't screenshot it, but here's an example of exactly that.

Zones are powerful, easy to extend and create your own, and can add a lot to your emitters. To end this part of the dev log here's a fun example of some stepped edge zones creating an almost neon effect:



It's never too early for neon

There are a couple of things left to do with the particles, one of which includes the ability to change their tint over time, and the other is a special edge-case pixel emitter. With these in particles in Phaser 3 are now on-par with the Particle Storm plugin, which is a fantastic place to be!

Over the next few days, I'll be returning to JSDocs land. Felipe starts again on Wednesday and will kick off by going through a few issues that you've kindly been raising on the repo, such as the fact the tilemap renderer doesn't work with zoomed cameras and by the end of the week we'll have made our choices on which extra devs we'll be hiring to help get this all wrapped up. I'm well aware we're behind schedule and that's not ideal, but I'm still calmly confident we'll get V3 out this year. And honestly, I can't wait.

Phaser 3 Labs

Phaser 3 Beta 8 is out and ready for testing.

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!

A superb article about writing a retro 3D FPS engine from scratch. The game is made on the TIC-80 Virtual Console although you could rework it into JavaScript and Canvas with a little effort.

Lucky enough to get your hands on a SNES Mini? This guide will show you how to hack it to fit loads more ROMs on :)

This article is an interesting study in how eBay handle font loading on their website. Not as simple as you'd imagine!

Phaser Releases

Phaser CE 2.9.1 released October 10th 2017.
Phaser 3 Beta 8 released October 30th 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.