Welcome to Issue 94 of Phaser World

Welcome to another monster packed issue! and I mean that quite literally given our featured game. We have a new release of Phaser CE, some great games,

and loads of news in the Dev Log.

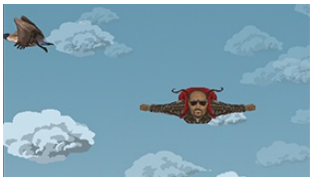
By the time you read this I'll be on a well-deserved holiday, which means there will be no newsletter next week. Issue 95 will arrive September 4th and along with it the next release of Phaser 3.

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

The Latest Games

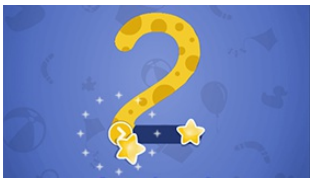
L.A. Rex >>> **Game of the Week**
It's your first trip to LA, so be a total tourist: hit the beach, see the sites... and eat tasty humans!

Pixel Peak >>> **Staff Pick**
A great little time trial skiing game in just 64x64 pixels.



Cloudfall

Catch the loops and gems as you fall from the sky and avoid everything else!



Drawing Numbers

A fun kids game that teaches them about drawing numbers.

Bounce to Dodge

Control the spikes so the ball doesn't hit them - a nice inverse of a popular mechanic.

What's New?

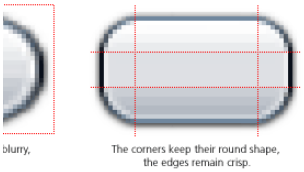
Phaser CE v2.8.4 Released

This new point release adds a new Ionic template and addresses some

TypeScript bugs.

Phaser UI Package

A set of UI components include progress bar, toast and a toggle slider.



Phaser Nine Slice Plugin

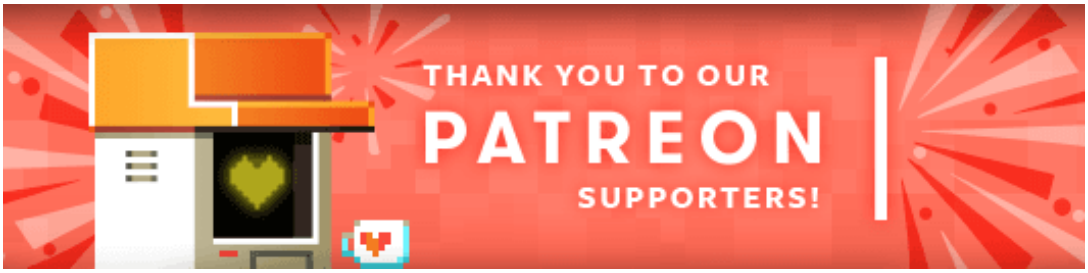
Easily add blazing fast nine slice scaling support into your Phaser games.

Creating an Endless Runner Part 2

The second part of the tutorial series on making your own endless runner game.

Creating an Endless Runner Part 3

The third part of the endless runner tutorial series sorts out the jumping.



Welcome and a massive thank you to the new Phaser Patreons who joined us this week: **Moritz Rebbert** and **Mick Chesterman**.

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 #94

We have an absolute tonne of things to cover in this Dev Log, if you're reading this in Gmail then it will almost certainly cut off part-way through, so be sure to view the full version to get all the content. Also please note there will be no Dev Log next week as we're on vacation, so we'll return with #95 in 2 weeks time.

Phaser CE v2.8.4

First up there's a brand new release of Phaser CE. This is v2.8.4 and contains lots of new features and tidying up. Gone are out-dated Pixi TypeScript defs and filters, the npm dependencies have been updated and there are enhancements to Arcade Physics, animations, tile sprites and lots more. Another superb effort from the community!

Yet More Lights

Felipe continued with his quest to optimize the hell out of the v3 lighting system. Deferred lighting is now running at quite a speed and we've tests with 200 lights blasting around as the camera zooms and rotates at a smooth 60fps. There isn't much left to do with them so he can move onto the shadows system next. I spent an evening creating a demo showing how lights can be used to create a suitably creepy scene:

Spooky!

The effect is a simple combination of tweens and 3 lights. I created an ellipse geometry object and placed it over the top of the candle animations. Then based on a timer the light takes a random position from within the ellipse and moves there. By keeping it the right size it causes it to look like the candle is casting a flickering light onto the tombs around it. Leave the example running long enough and you'll see clouds and fog rolling in the background (again simple tweens and nice assets).

Physics Ahoy!

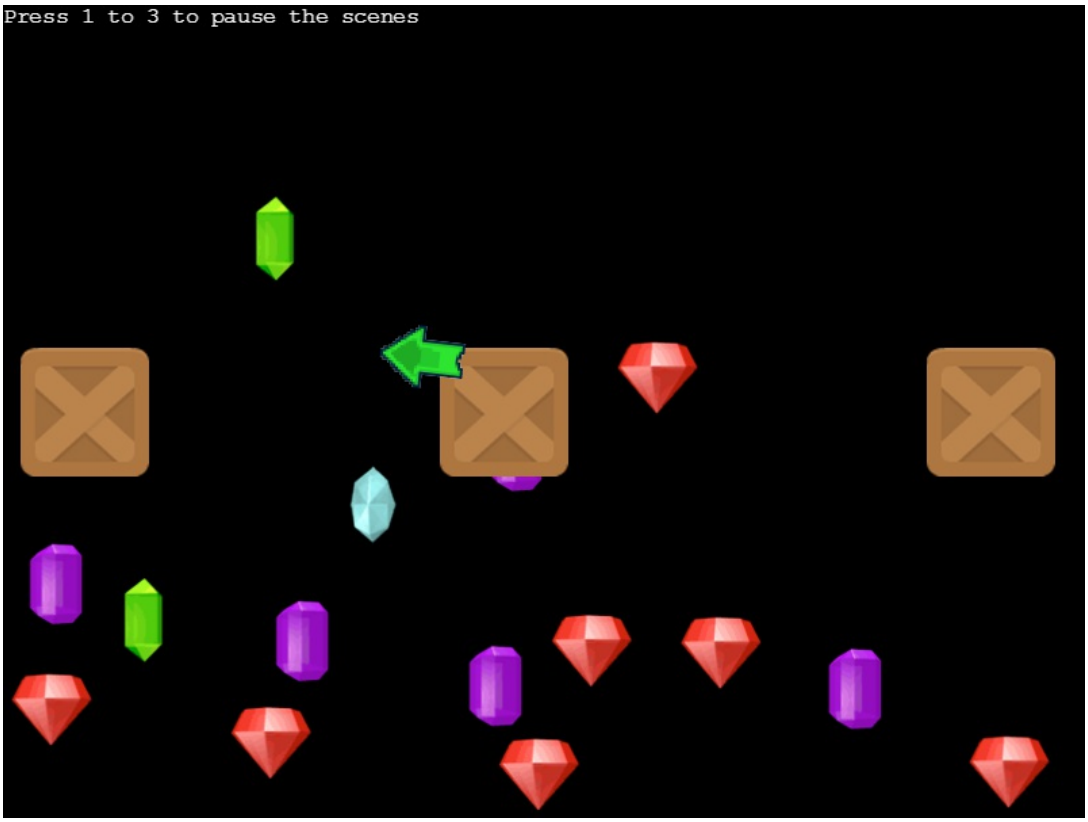
Some of the feedback I got about the v3 Alpha was that it was pretty tricky to create an actual game because none of the physics systems were working in it. This was a fair point. Although it's perfectly possible to handle this all yourself it's also one of the key reasons Phaser is so popular in the first place because it makes stuff easy for you, and the easier I can make using v3, the more people will be willing to help test it.

With that in mind, I shifted around our development plan a little and accelerated finishing off one of the physics systems for v3. I had already done quite a bit of work on it as part of the Tilemap import and thankfully it only took a few more days to get it into a state I was happy with. What's more, the process and structure will pave the way for Arcade Physics and others as they are added

later.

Scene based Physics System

In v2 you had to start the physics system you wanted by calling it from your code. One State could have several different physics systems running if it wanted to. In v3 this has been changed. A Scene can only have 1 physics system running at once, never more than this. The difference is that in v3 you can have multiple Scenes, and they each have their own instance of a physics system (if required). This is better explained in the example below:



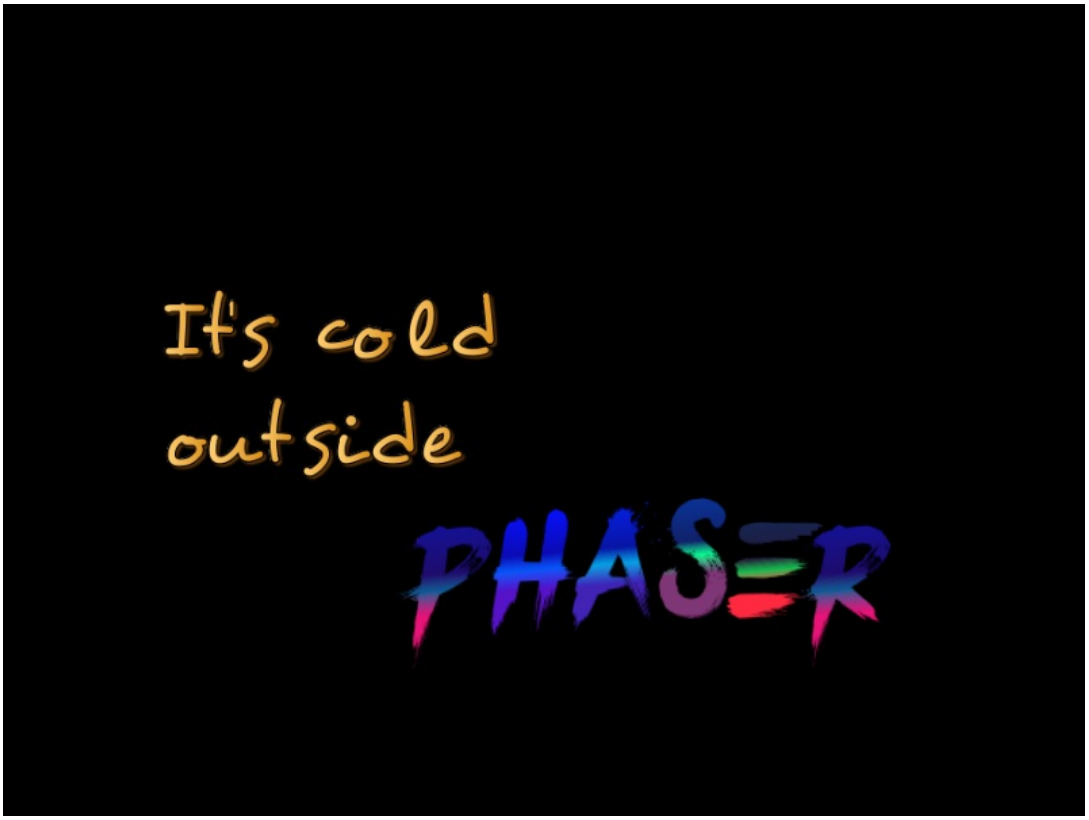
Multiple Scenes each with their own physics system

In the example above there are 3 Scenes running in parallel. Each one has its own instance of the Impact Physics Manager. This means that the gems in Scene A are being managed by a completely different system to the crates in Scene B. Press the keys to pause and resume the Scenes to see the effect it has. Another benefit of this is that because each Scene is fenced off they can have their own configs too. Again you can see this in the example above. Scene A (the gems) has a gravity value of 100 set. While Scene B (the crates) has gravity disabled. This is all controlled via the Scene config object, although you do have the option to set it at run-time via your code too.

Independent Bodies

Physics Bodies in v3 are entirely self-contained, existing within the physics system itself. Although they can be linked to Game Objects they don't have to be. This gives you much greater freedom in how your bodies interact with your Game Objects, allowing you to easily set-up custom rules or conditions. For example, maybe you want one physics body to be responsible for a whole bunch of Sprites - this is now possible by taking the body and applying it to your own custom classes.

It's safe to say though that most of the time you'll want your physics body attached to something that actually renders. You can add any Game Object to a body, and indeed you can swap the Game Object that a body is using at any point. This means you can do some cool things, like bounce around Dynamic Bitmap Text:

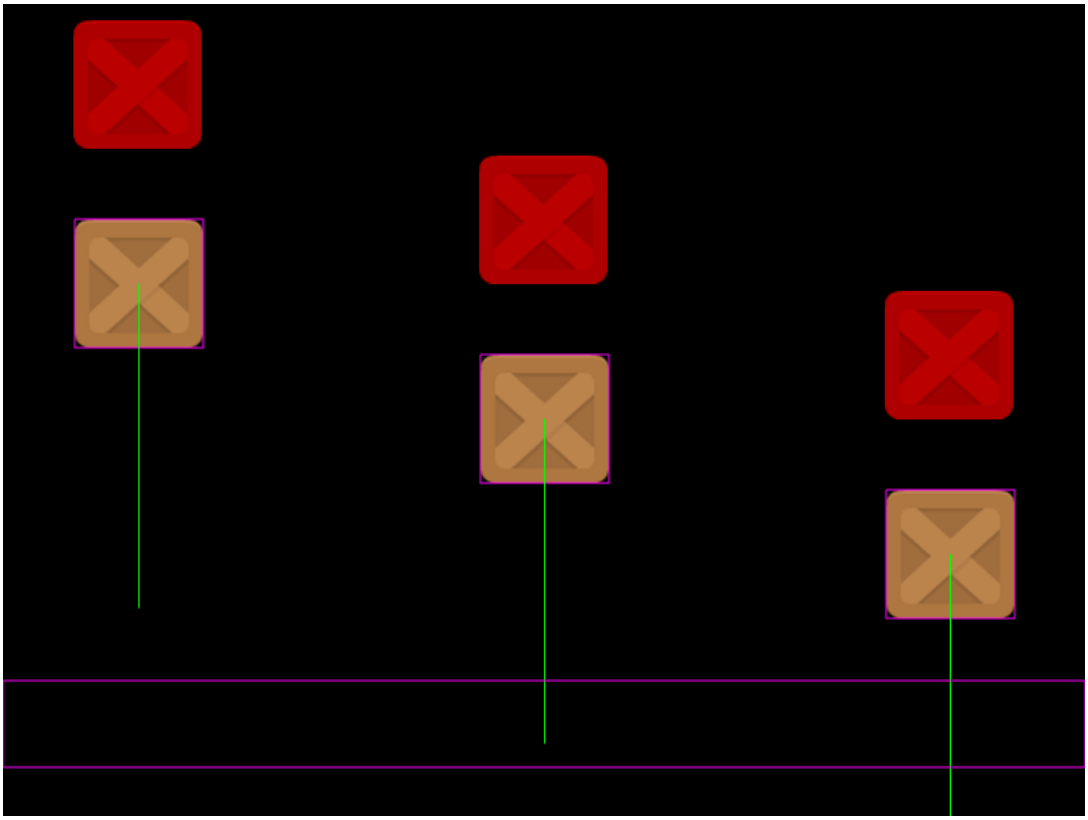


Hook any Game Object to a physics body

The size of the body can be set to anything and there is an offset value as well, allowing you fine-grained control over the position of it in relation to the Game Object.

Built-in Debug Graphics

It's always useful to be able to visually see your physics bodies and velocities, especially as bodies no longer have to be paired with Game Objects. Because the v3 Graphics object is so much more powerful than v2 it was a natural choice to use it to render the optional debug layer to:



Optional Debug Graphics

You can turn on the debug layer by simply defining it in your Scene or Game config:

```
physics: {
    impact: {
        debug: true,
        debugShowBody: true,
        debugShowVelocity: true,
        debugBodyColor: 0xffff00,
        debugVelocityColor: 0xff00ff
    }
},
```

If you don't like the default colors v3 uses then you can set your own using the color properties, as shown above. You can also toggle the display of the bodies and velocity markers. The values above are the defaults that the physics body

will inherit when it is created, but you can override them too. This means you can target a specific body to only show its debug body, or change your player body color to be something else that stands out more. It's pretty flexible!

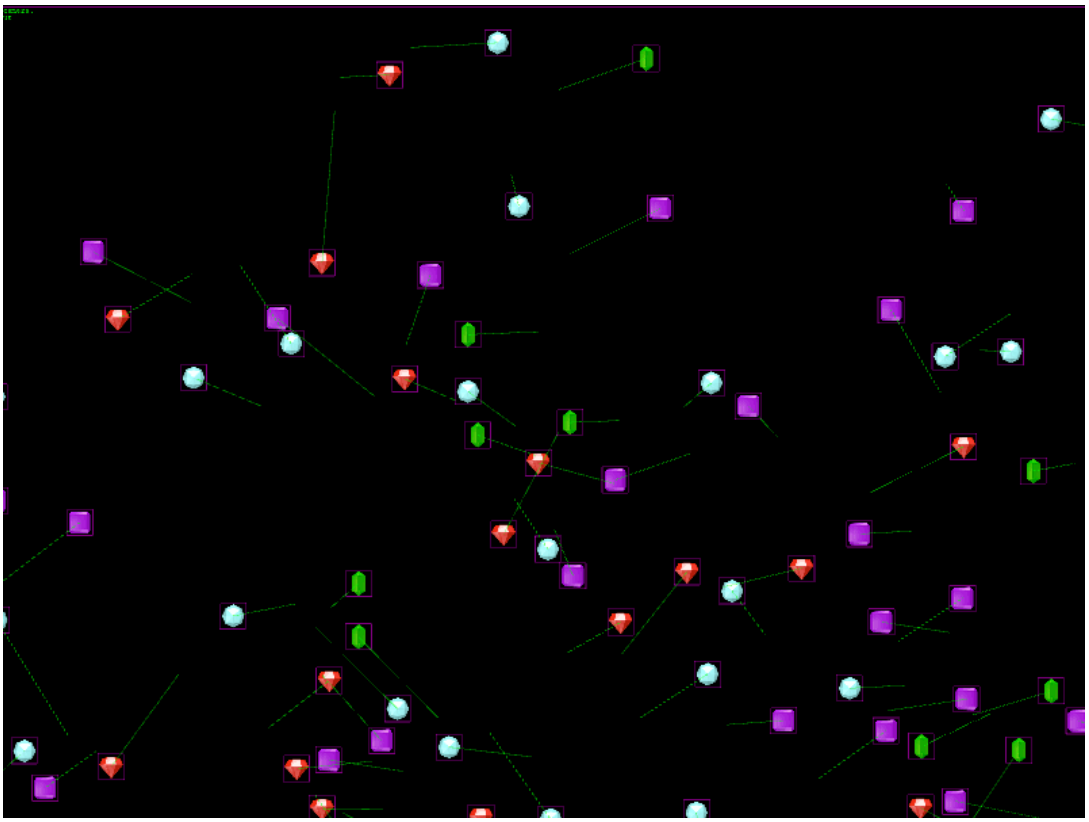
Callbacks and Events

The Physics world will emit a COLLIDE_EVENT when two bodies hit each other. The event contains the 2 bodies involved in the collision, the axis on which they collided and references to their respective Game Objects (if they have any). But events aren't always the best way to do things, for example as the event is fired from the world itself, you then have to perform your own logic to work out if it involves a body you're interested in.

Which is why I also added collision callbacks. These are specific to a single body and are invoked when it collides with any other body. Again you are sent the body and the axis as arguments to the callback. This gives you much more fine-grained control over what happens and when in your game.

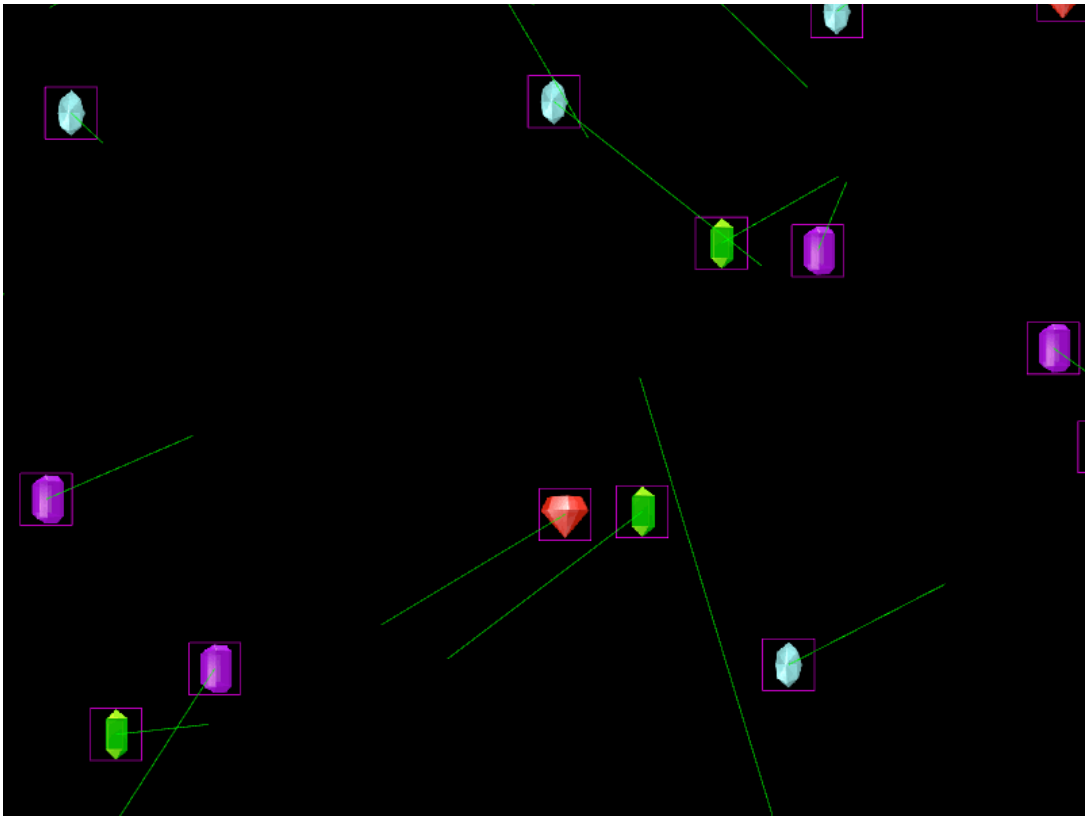
Push It, Push It Real Good!

Part of the process when building any part of v3 is for us to push it as hard as we can in terms of performance testing. In short, we don't want to be making anything that's too slow. You saw this last week in the Tweens performance test and you've seen it before in the renderer tests. So it was time to see what would happen with the Impact physics implementation. To test this I created 3 different examples. Each example was set in an ever larger world to accommodate the sheer volume of bodies and I added in camera controls too. Here is the 100 bodies test:



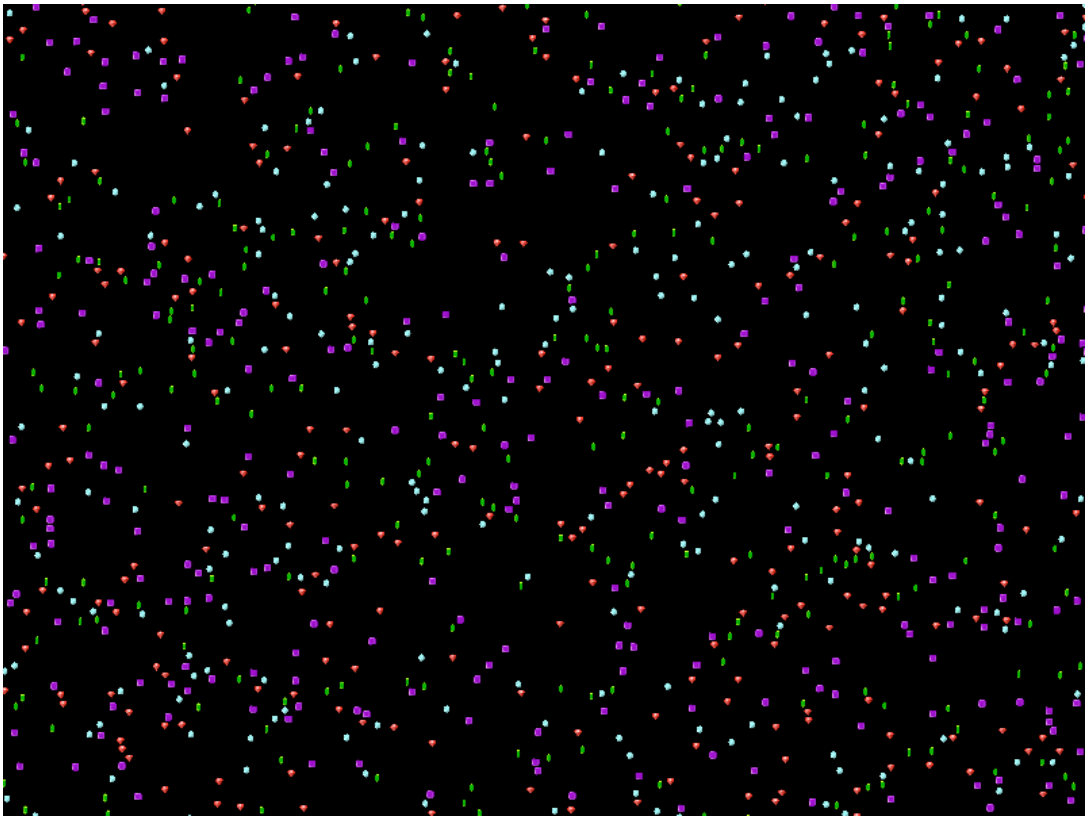
100 Physics Bodies

You'll notice that the debug layer is left on (which of course adds its own overhead to the example) and all of the Sprites are animated. Use the cursors to move around and the Q and E keys to zoom the camera in and out. The above example performed quite happily on all the current crop of browsers, but we wanted to push it even more. So here's a test with 1000 bodies:



1000 bodies!

Again, we found this ran without any issues in nearly everything we threw it at. But out of sheer curiosity, I pushed it even further again, this example is running with no less than 10,000 physics bodies:



That's a lot of bodies!

I had to disable the debug layer because it added too much overhead, but with that turned off this example still manages 60 fps on my current laptop, although it definitely struggles if you pan the camera too fast. To be fair though I can't really blame it. Without wanting to sound too much like Bill Gates, 10,000 ought to be more than enough simultaneous physics bodies for anyone :)

Now clearly it isn't going to run like this on all devices or desktops, but as an upper limit of what's possible in v3 while retaining 60fps, it's a nice showcase to have and fills me with confidence that this is going to easily handle most things you may wish to throw at it.

What's Next?

Next week, while I'm away, Felipe will hopefully manage to complete the Canvas Tilemap rendering functions and work on Shadows too. When I get back I will be tackling the API side of Tilemaps, of which there isn't a huge amount left thankfully, and preparing for the first Beta release. Based on our schedule I'm due to start the new Sound system after this, but I think it will make more sense to sort out how we're going to handle documentation and spend a week filling in JSDocs instead, as they're going to be crucial for user adoption as we move through the Beta releases.

If you feel like playing with the new features then please do so! You can get the latest build from the repo and all the examples can be found in the examples repo too. There are actually 28 different physics examples alone, I just only had room to feature a few in this issue, so please do take advantage of them to learn how things work!

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. The group is for anyone who wishes to discuss what the Phaser 3 API will contain.

Gaming beer?! Alrighty! Play Brew creates craft beer with a nostalgic twist, inspired by the culture of the 80's and 90's.

An interesting blog post about what it means to be a successful indie game developer.

It might be nearly a year old, but the 2016 Classic Tetris World Championships final was breathtaking!

Further Reading ...

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

Phaser Releases

The current version of Phaser CE is 2.8.4 released on August 15th 2017.

The current version of Phaser is 3.0.0 Alpha released on July 31st 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.