Welcome to Issue 105 of Phaser World

Greetings everyone! We've one monster issue here for you. Lots of new games and tutorials, some awesome new supporters on Patreon, and a huge Dev Log

on Phaser 3. That's what happens when you've 4 devs working on it at once! There's a new beta to try, lots of examples and even a complete game to play :)

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.

It's that time of the year again when Zenva run insane discounts on their popular Phaser courses. You can get the Complete Mobile Game Development Course (Platinum Edition) for just $39.

That's 17 modules, including the new PhoneGap module, 15 projects, loads of content, resources and tutor lead learning for a bargain price. I'm often asked which is the best video course to learn Phaser, and honestly, this is it. Grab it while you can!

The Latest Games

Game of the Week

Governor of Poker 2

One of the biggest poker games in the world comes to the browser. How will you rank in the every increasing stakes of Texas holdem?

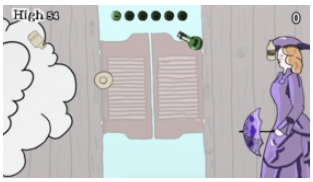
Staff Pick

Lasso Lassie

Shoot up and lasso the bad guys in this wild west themed arcade style game.

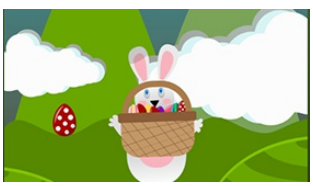
Space Monster Assault Force

A side scrolling space shooter - Collect power ups and health to make it through the level, beat the boss, and get a high score.



Riot at the Wyatt

A riot has broken out at the Wyatt Saloon, and poor Miss Dallas has only her parasol to protect herself from all the flying debris!



Funny Bunny

Catch the falling eggs in a colorful, cartoony environment.

What's New?

PlayFab SDK now supports Phaser

Get access to leaderboards, analytics, a/b testing and more from your games with native Phaser support direct from PlayFab.

Phaser Editor 1.4.3

A new edition of Phaser Editor, with loads of new and useful features, has been released!

Phaser CE v2.9.2 Released

The latest version of Phaser CE is out.

Phaser Tiled

An npm Package for optimizing large tiled maps.

Phaser Typescript Template

A Typescript template for writing HTML5 Phaser games with Typescript

Game Off

The 5th annual GitHub game jam starts is go! Get involved and check out our Phaser resources.

Welcome and a massive thank you to the new Phaser Patrons who joined us this week: **JR** and **Richard Herbert**.

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 #105

This week we've got a pretty huge update, so if this newsletter gets cut-off part-way through (<sarcasm>thanks, Gmail, that's a killer feature you've got there</sarcasm>) then please be sure to read the full thing or you'll miss out on lots of goodies.

First-up: Phaser 3 Beta 10 is now out. You can grab it from GitHub pre-built, or from npm using the beta tag. This build includes all work completed over the past week, in short, everything we discuss in this Dev Log.

You may notice that the unminified file size is getting pretty massive, currently at 5.2MB. Don't let this scare you. Over 64% of that is the embedded source map that we use for debugging! The remaining 1.9MB includes all of the jsdoc blocks too. Minify things and we're closer to 600KB, even before gzip, and this is the 'whole kitchen sink' build as well. The main Phaser entry point includes literally *everything* we've built so far. When you come to create your own bundles you can pick and choose.

Schedule Update

Those of you who pay attention to our schedule will know that we are not going to meet the self-imposed early November deadline. Quite simply, some areas have taken a lot longer or been built to a much higher fidelity than predicted. When we received the award from Mozilla it was linked to agreed milestones, which had to have dates attached. Our V3 release date *in the contract* was by the end of 2017 (on the schedule it was mid-November, but they allowed us some leverage for the paperwork).

I've been laser-focused on getting V3 done by the end of the year, to the detriment of all else really. There are only so many 4am coding sessions in a row that your body and soul can take. Last week I sat down and planned out more accurately what is needed for the final push and it was clear, that with all the interruptions that December brings to schedules, that there wasn't enough time

left.

I've since talked to Mozilla who have agreed we can extend this by one month. This means we need to release by the end of January 2018, with no further delays allowed beyond that. To be honest, we can't really afford any more delays anyway. While it's great that they've given us this extra time, it does, of course, mean delaying their payment to us by another month. Software development is always swings and roundabouts it would seem.

We now have 4 developers working on V3, two of which are full-time, so lots of key areas are now in build simultaneously. If you follow our progress on GitHub you'll have seen a huge influx of commits over the past week across some major areas. Not having to rush it out for Christmas is a significant weight lifted off my shoulders, so thank you to Mozilla for agreeing to this. If you're interested, there is an email thread in the Phaser 3 Group where I discuss what is expected re: the contract.

Tilemap API Updates

Michael is working on the updates to the Tilemap API. I asked him to write a short summary of what he's been doing over the past week:

"The Tilemap development so far has been focused mainly on the static & dynamic rendering pipelines, so this week I've been restructuring things to bring in familiar features from v2. Tilemaps can now be created from CSV files or Tiled JSON files via a modular parser system. Though it doesn't come with pretty examples (yet!), I have also been restructuring the overall API. Like v2, there are Tile, Tileset, TilemapLayer and Tilemap classes. In the new Tilemap, you can mix static layers and dynamic layers in the same map - so you can optimize your background layers with static rendering while getting the power of dynamic rendering on just the layers that need it."

Sound Manager Updates

Pavle is in charge of coding the new Sound Manager for V3. He too has written a short update on what was done last week:

"Loading of audio files (only Web Audio at the moment) and an example that demonstrates they are stored in cache after successful loading.

Drafting public API for handling sounds with focus on exposing all the neat functionalities of the Web Audio API to users as well as making the best possible

implementation of that API using fallback HTML5 Audio."

You can read more about updates to both of these APIs next week.

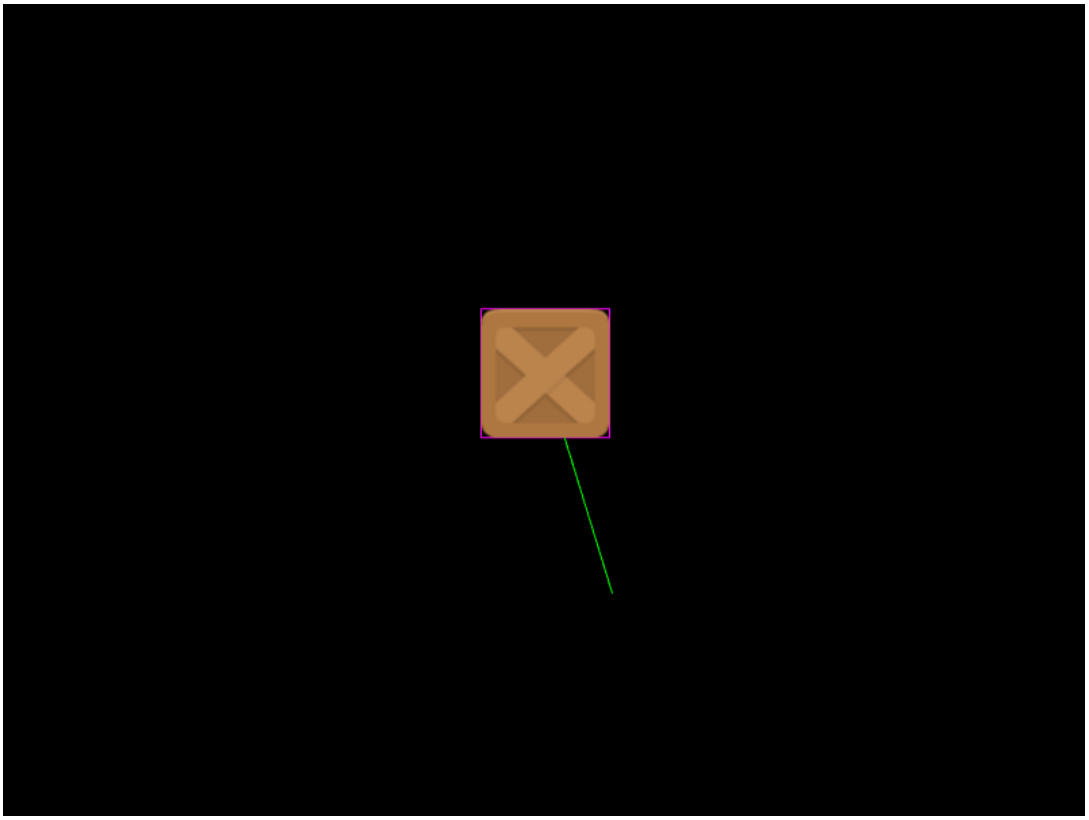
Arcade Physics Strikes Back

Last week I focused on adding Arcade Physics into V3. It was always on the roadmap, as despite limitations it's an extremely popular system for simpler games and beginner devs. After a lot of hard work, it's now complete and has even gained a stack of features and optimizations over and above V2.

When I went through the original code, including all the enhancements and fixes introduced by Phaser CE, I realized that fundamentally I wanted it to work differently internally. In V2 you use Arcade Physics (which I'll abbreviate to AP) as more of a collection of helper methods rather than an actual system. The AP World, for example, enables physics bodies on Sprites and owns the QuadTree they can use for collision, but beyond this, it's little more than a bag of helper functions. Important and vital functions, sure, but that's all. Bodies are only updated because every Game Object in V2 has its preUpdate method called and this is what causes the physics integration to happen.

In V3 this is all turned on its head. You create bodies in the World and it is the World that manages and looks after them. Game Objects no longer need to have pre or post update methods, they just need a valid AP Body and the World will handle the rest.

Fundamentally this doesn't change much from the API point of view, yet the internal separation is a lot cleaner and it means no more redundant update cycles for non-physics Game Objects. Here are a few examples and their code to show what I mean (as always, click the screen shots to run the demos)



A simple AP Body

Here we've a box bouncing around, colliding off the world bounds. You'll immediately notice the debug layer has been enabled. As with Impact Physics, this is controlled via the Game Config and you can toggle the body and velocity guides as well as change the colors, globally or for specific bodies.

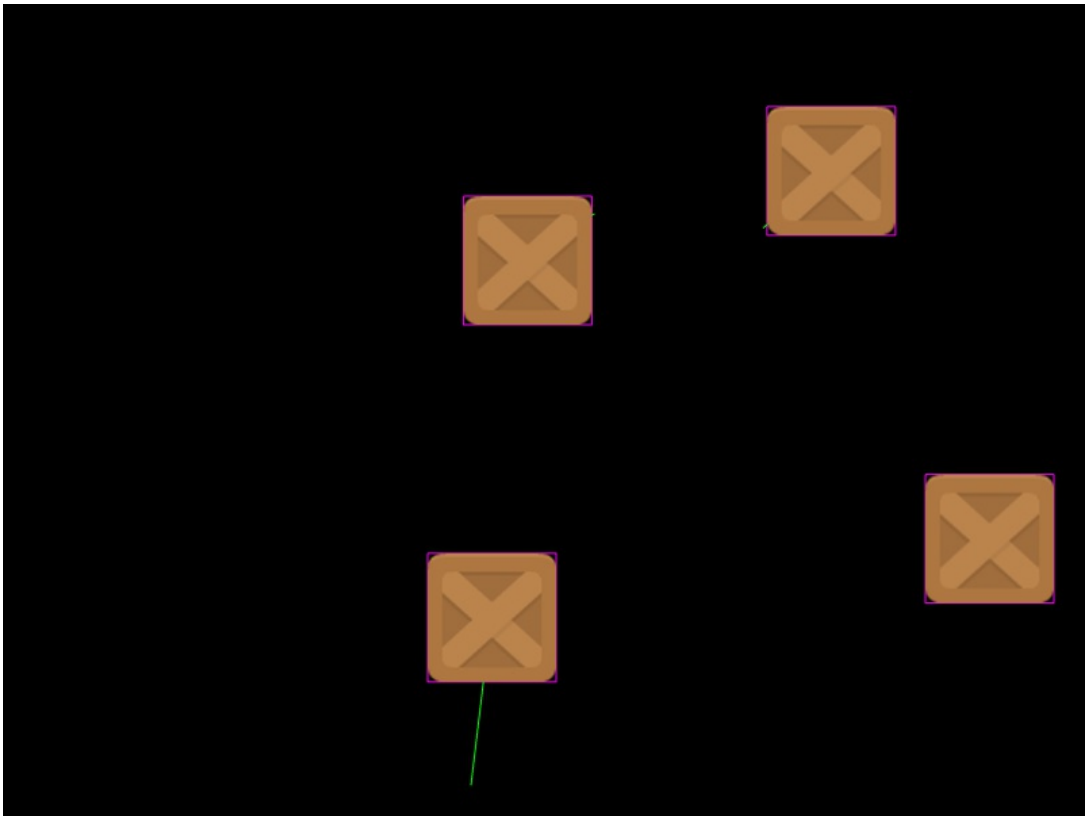
Here's the code for the above demo:

```
function create ()
{
    var block = this.physics.add.image(400, 100, 'block');

    block.setVelocity(100, 200);
    block.setBounce(1, 1);
    block.setCollideWorldBounds(true);
}
```

As you can see it's just 4 lines and that's only because we didn't chain the calls together in one! If you're familiar with AP in V2 you're probably wondering where these 'setVelocity' and related methods are coming from. They're new and exposed directly on the Body object, or any Arcade Physics Game Object. They're chainable too, so you could have set them all on one line if you'd liked.

In V2 to set the velocity you would have accessed the velocity property itself, which was a Point object, and set the values directly into the x and y properties of it. You can still do this. All of the important properties from V2 are present in V3 and have the same names, so old code will be a lot easier to port over. However, you can take advantage of the new style if you prefer, it's up to you.



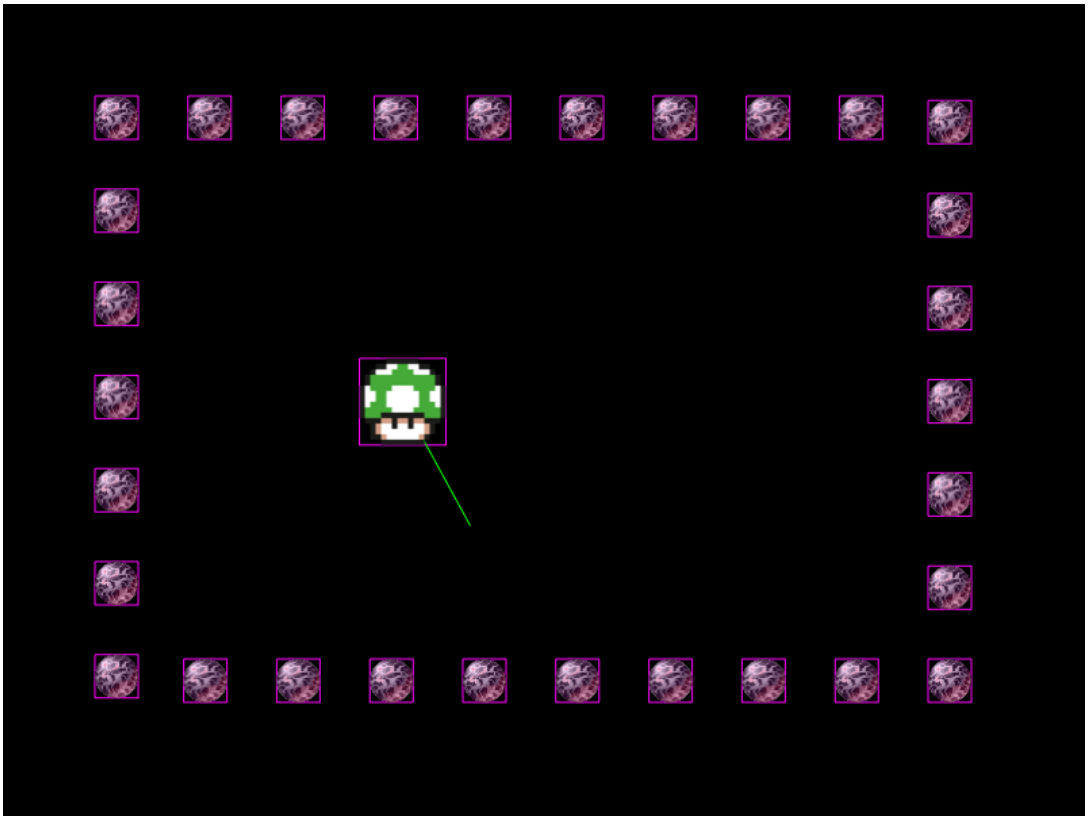
A Physics Group

In this example, we create a simple physics group. A Physics Group is much like a normal Group in V3, in fact, it extends from that class, but it adds in the ability to manage bodies and use a config object to set-up how they're all created as the following code shows:

```
function create ()
{
    var group = this.physics.add.group({
        bounceX: 1,
        bounceY: 1,
        collideWorldBounds: true
    });

    group.create(100, 200, 'block').setVelocity(100, 200);
    group.create(500, 200, 'block').setVelocity(-100, -100);
    group.create(300, 400, 'block').setVelocity(60, 100);
    group.create(600, 300, 'block').setVelocity(-30, -50);
}
```

In the above example, every Body the Group creates is instantly given bounce values and the ability to collide with the world. As each block is created they're manually given different positions and velocities. The end result is what you saw in the demo. Groups go further than this though, as well as being collections of physics bodies you can also influence all of the bodies at once and most importantly of all, they provide for easy collision as you can check for collision between a sprite and a group, or a group and a group, or a group an an array, or any combination you like:



Sprite vs. Group

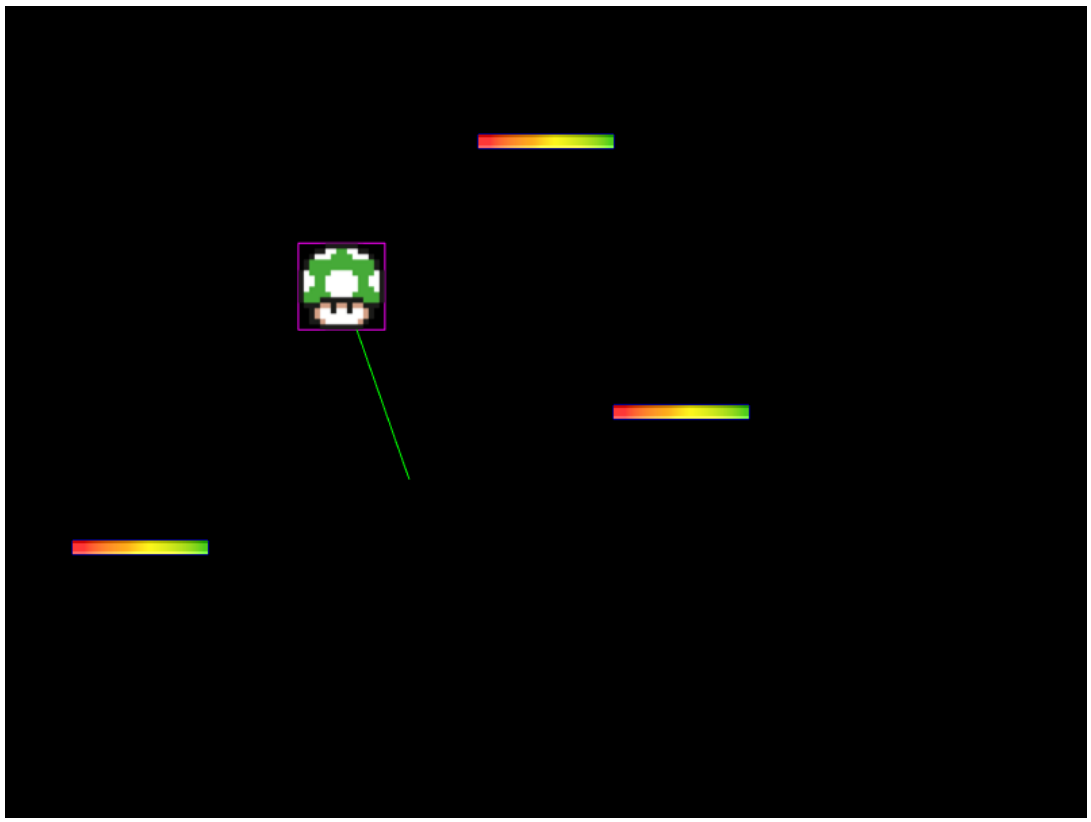
Colliders

New to Arcade Physics in V3 is the concept of Colliders. In V2 if you wanted to check for overlaps or collision between two objects (or group of objects) then you had to run these checks in your update function, every frame. Sometimes this is really useful because it allows you to specify the sequence of collisions, or perform them conditionally based on other events.

However, very often it meant you had an update loop full of collide checks and little more. To avoid this V3 introduces the concept of Colliders. These are bundles of objects and callbacks that are run automatically by the physics world for you. You can create a collider, tell it what to monitor and leave it be. It will then happily take control, firing callbacks as needed. You can, of course, pause and resume colliders, or remove them altogether, but they should prove helpful in reducing the volume of code in your games.

Static Bodies

V3 also brings the concept of Static and Dynamic Bodies to Arcade Physics. A Static Body is one that never moves and cannot be moved during a collision with other bodies. They are especially useful for game scenery like platforms or walls or objects that you want the player to collide with but not be affected by that:



Dynamic vs. Static Bodies

Internally V3 uses an RTree for its broadphase. This is a special type of data structure designed specifically for extremely fast spatial sorting. The downside is that it's designed for objects that don't move. If an object is moving then you need to remove it from the tree and re-insert it again. This is a costly operation, to the point where it's often faster to just clear the entire tree and re-populate it from scratch.

This is why we introduced Static Bodies. When you create a body with Arcade Physics it will now add it to either the static or dynamic tree. The static tree never needs to change until you add or remove a static body. This means it retains its phenomenal searching speed for the majority of its life. By comparison, the Dynamic Tree is cleared and rebuilt each frame.

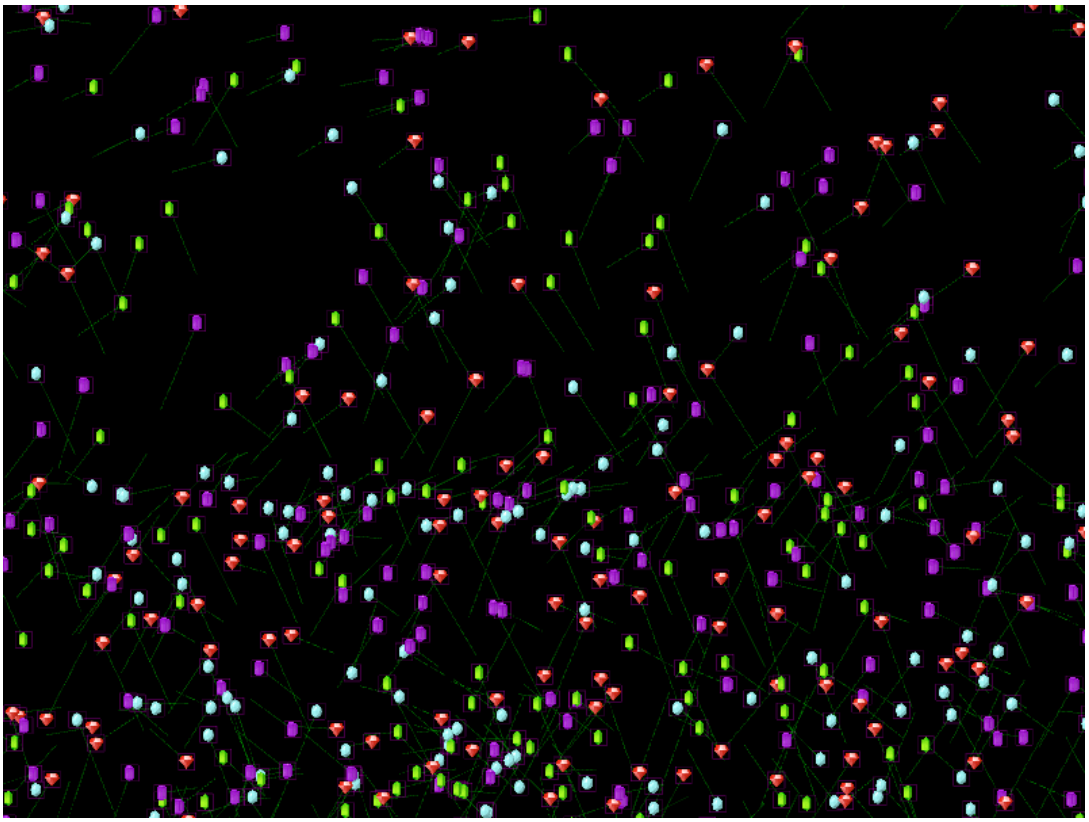
As a result of this change, you're now able to say when you create a body if it's going to be static or dynamic and V3 will handle the tree selection for you. It also handles it when you run overlap or collide checks and of course the Colliders know which tree to poll as well. It also allowed me to get rid of all the 'sort' code from V2, which was used to sort Groups based on game direction. This is now gone as the RTree makes it redundant.

Depending on the type of game you're making this could provide a significant speed boost. And remember, it's only the body that isn't allowed to move if it's static - the camera can move quite happily! Static bodies can also be animated or have tween effects applied to them (like alpha or tint), making them highly flexible.

Circular Bodies, Offsets, etc

I also ported over Circular bodies from v2, which are in and working properly (as either static or dynamic). I recoded how the offset works as well. Previously calling `setSize` on a body took quite a bit of trial and error. For example, if you made a body larger than the Sprite texture you'd have to manually work out its offset in order to position it around the sprite properly. This is now handled for you and makes it a lot easier to use. Of course, you can still tweak the offset directly if needed.

I decided to copy the stress test I created for Impact Physics for Arcade, so here's a 100 test and the one below is 1000 dynamic bodies:



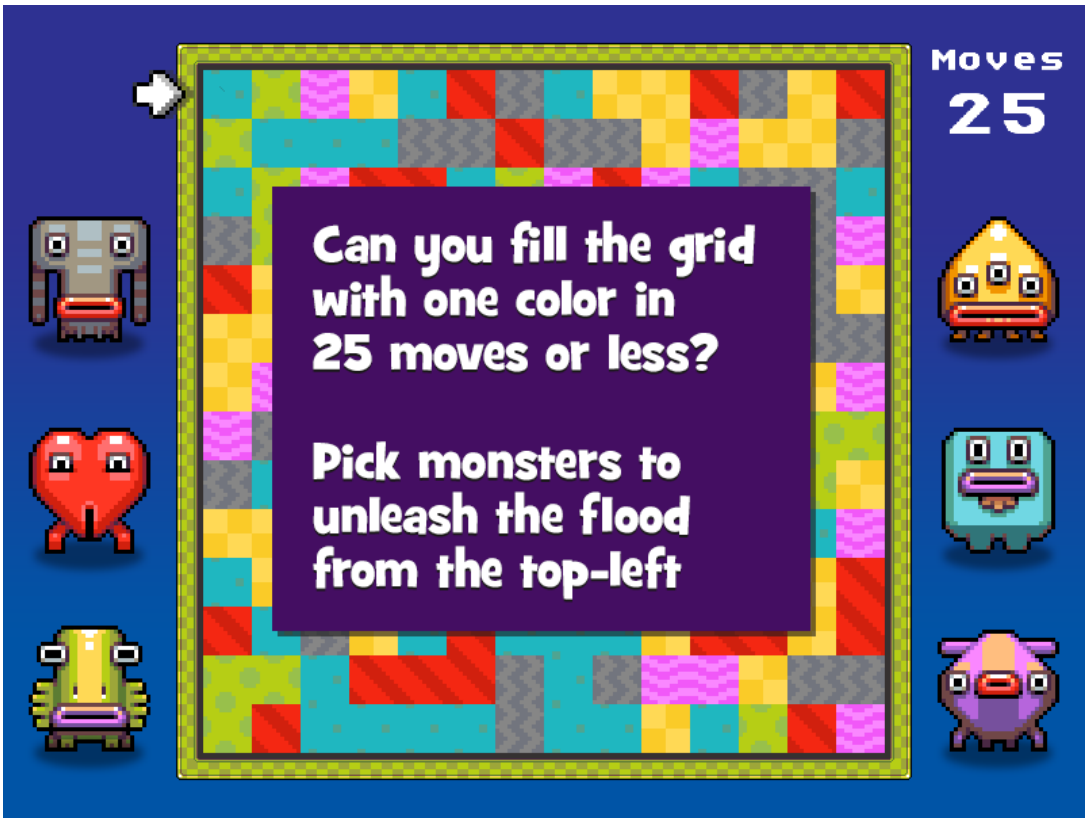
1000 bodies

I'm very pleased with how Arcade Physics has turned out in V3. We actually did a lot of R&D work, quite some time ago, into creating Arcade Physics 2 - and although this isn't present in V3 I feel like the new version is still a significant improvement over the old, while still retaining all of its ease-of-use, and even expanding that in some areas. Old limitations still apply, of course, i.e. it's all still just an AABB system at its heart, but it does it better than it ever has before.

Flood Fill Game

I really felt like making a game in Phaser 3. I knew I didn't have much time, it would have to literally be created in less than a day, so I had to keep it simple. I also didn't want to get wrapped up in graphics, so I decided to do a remake of an old game I remember playing years ago.

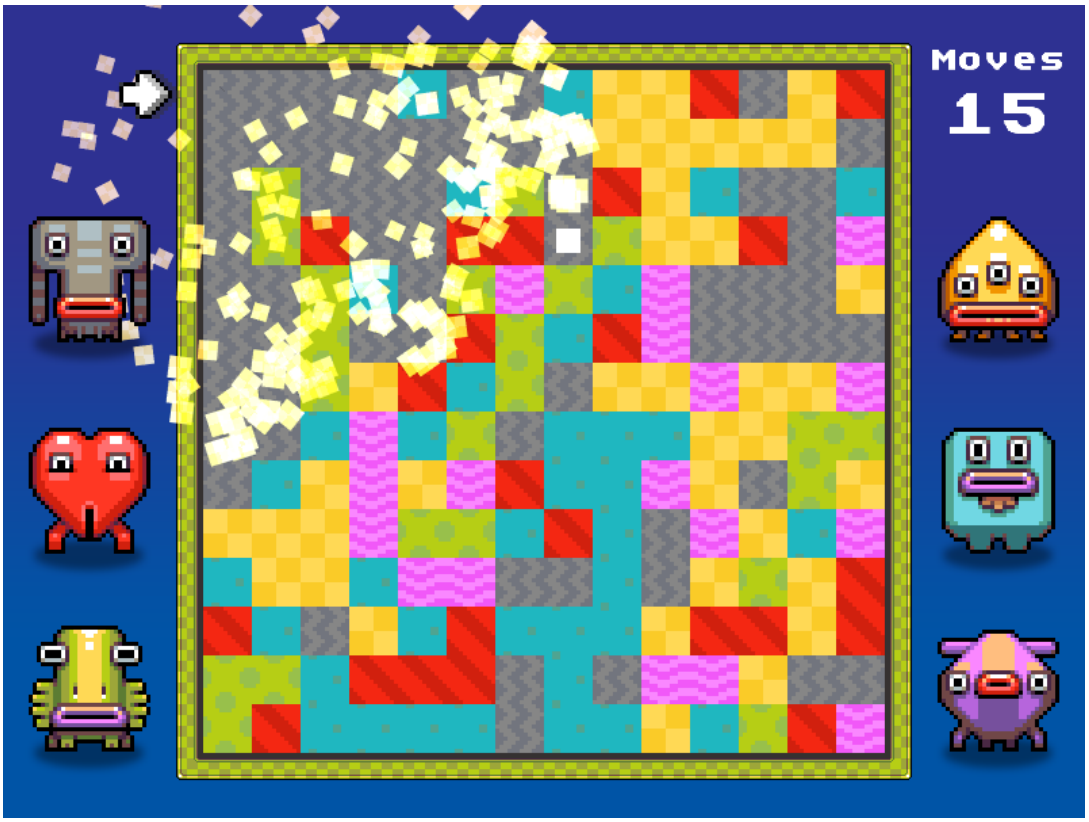
The concept is very simple. You're given a grid of 14x14 squares comprised of 6 different colors. Starting from the top-left square you have to see if you can fill the entire grid in 25 moves or less. The way filling works is via a 'flood fill', just like in an old-school art package. A new color is 'flooded' into the top-left tile and it'll change any other tile that it touches of the same color. Using this you have to see if you can work your way across the grid and fill them all. This version is desktop only but I'll modify it for mobile shortly:



Can you fill the grid?

It was really fun to work on the game and, actually, it helped find a couple of nicely hidden issues in V3. This doesn't surprise me. When I was first building Phaser I was using it daily in my client work and one fed into the other. But V3 has been built mostly in isolation so far. It's not until you start putting it through its paces with real games that things manifest, no matter how small that game.

So it's something I'm going to try and do more and more of. In the meantime enjoy playing Flood and see if you can get to the end sequence! If you need it, there is a cheat mode built-in (just press M to add extra moves, or X to take them away). It's not the most complex game in the world, but it's small and fun, with some neat particle and tween effects and Ilija's lovely graphics help bring it to life :)

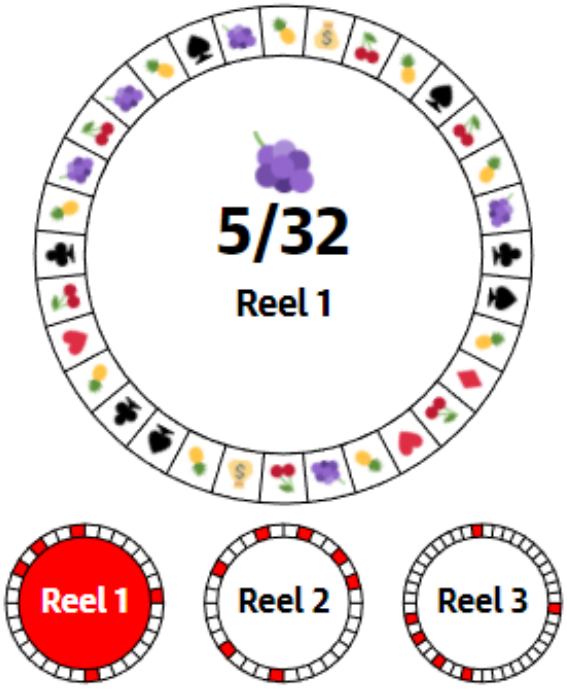


Phaser 3 Labs

Phaser 3 Beta 10 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 brilliant interactive article on how poker machines are designed to be addictive. Lots to learn for game devs (especially given this weeks cover game!)

A superb tutorial on using FlexBox and Grids for the ultimate in layout control!

Positech Games recently gave a presentation on "How not to go bankrupt making indie games" and has uploaded his slides online, which are worth a flick through.

Phaser Releases

Phaser CE 2.9.2 released November 10th 2017.
Phaser 3 Beta 10 released November 13th 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.