Welcome to Issue 146 of Phaser World
There's not one, but two big releases this issue. The long-awaited Phaser 3.17 is out, which you can read about in this issue's Dev Log. And hot on its heels is Phaser CE 2.13. So there's plenty to get stuck into, regardless of which version you use.

Like lots of you, I went to see Avengers Endgame recently. As expected, it was a fantastic film. Which makes it even more fun that not one, but two, Avengers-themed Marvel games were made with Phaser and recently released. One of them an advergame built using no code at all and the other was shown prior to the film in cinemas across America. It's always fun to see Phaser cropping up in new locations, and pre-film cinema games is certainly a new one :)

As usual, be sure to read this newsletter on the web so you don't miss anything.

Got a game or tutorial you'd like featured? Simply reply to this email, or message me on Slack, Discord or Twitter. Until the next issue, keep on coding!

The Build Your Own Phaser 3 Arcade Games Pack is a brand new release and
available to buy today. It contains 5 complete training courses that teach you how to remake 5 classic retro games in full ES6 with Phaser 3. Learn how to make Pong, Breakout, Asteroids, Nibbler and Space Invaders with the full tutorial and code per game. If you create a game based on one of these tutorials, please let us know!

The Latest Games

Game of the Week
Noovie Trivia Shuffle
Test your Movie IQ in this Marvel Superhero themed trivia game, with lots of fun rounds and slick presentation.

Staff Pick
Avengers Endgame
Test your reflexes by blasting away the holograms with the correct hero in this official Avengers Endgame title.

Jewel Bubbles 3
Break all the blocks, create huge explosions, combine chains for bonuses and bring the marbles down in this match 3 game.

Galaxian
You know the drill - move left and right and waste those aliens! It's a retro remake of the classic.

Bubble Shooter World Cup
It's a bubble shooter, but with World Cup themed footballs instead of bubbles!

Chopstick Cooking
Catch the food as it flies past with your chopsticks and then assemble it into the burgers the customers want!
**Phaser 3.17.0 Released**
The latest version of Phaser 3 is now available, including the new Shader and DOM Game Objects, Camera Masks and much, much, more!

**Phaser CE v2.13.0 Released**
The latest version of Phaser CE is now available with some important changes to Mouse and MS Pointer input handling

**Destructible Terrain Code**
A helper library for creating destructible 2D terrain complete with collision checks and examples.

**Match 3 Tutorial Part 2**
In the next part of this series a new class is created that can generically handle any match 3 grid system.
2D Skeletal Animation Tutorial
A new tutorial on creating 2D skeletal animations using Creature and Phaser 3.

How to Build a Simple Game in TypeScript
Start with setting up a dev environment and moving to coding a game in TypeScript in this beginners tutorial.

Block It Tutorial
Build a game like Block It in less than 100 lines of code in this new tutorial.

Phaser 3 Game Development Course
A complete Phaser 3 and JavaScript Game Development package. 9 courses, 119 lessons and over 15 hours of video content. Learn to code and create a huge portfolio of cross platform games.

Help support Phaser
Because Phaser is an open source project, we cannot charge for it in the same way as traditional retail software. What's more, we don't ever want to. After all, it's built on, and was born from, open web standards. The core framework will
always be free, even if you use it commercially.

You may not realize it, but because of this, we rely 100% on community backing to fund development.

Your support helps secure a constant cycle of updates, fixes, new features and planning for the future. There are other benefits to backing Phaser, too:

I use Patreon to manage the backing and you can support Phaser from $1 per month. The amount you pledge is entirely up to you and can be changed as often as you like.

Please help support Phaser on Patreon
Thank you to these awesome Patrons who recently joined and make continued development of Phaser possible:

Adam C Moore  
Arne Keller  
Camilo Baquero  
Daniel  
Francois Lionet  
Greg Findon  
Jace Clark  
Jim  
Kevin Clark  
Kilian Braquin-Mitel  
Murthy  
Simon Farshid  
Taylor Nelson  
Thomas Mikalsen  
Zoran Knezevic

Dev Log #146

It's been a busy few weeks since the last issue to say the least! And the headline announcements are, of course, the release of Phaser 3.17 and Phaser CE 2.13. Let's dive in ...
Phaser CE 2.13

Phaser CE 2.13.0 is an API changing release, hence the bump in version number. Most of the updates are to do with the way input events are handled, including changes to MS Pointer, the mouse wheel event names and the cursor leaving the canvas.

There are two new loader methods: 'tilemapCSV' and 'tilemapTiledJSON' for easy loading of CSV and Tiled map data respectively. Finally, there are fixes for animation speed and sound timer syncing. It's another great community-driven update and if you use CE it's well worth upgrading.

Read more
Phaser 3.17.0

It's finally here :) There's no denying that 3.17 was an awful lot of work. And I didn't even get to include everything that was worked on in this release, either. The headline items include:

- DOM Elements are no longer experimental
- The new Shader Game Object is included
- Vastly improved Bitmap and Geometry mask performance
- The ability to add masks to Cameras
- Lots of new Geometry intersection functions
- New File Loaders for CSS Files,
Multiple JS files and GLSL Bundles

- New Arcade Physics features including 'get all bodies in a rectangle' or 'overlap a body with just these tiles in a tilemap'

- New Tilemap functions including removing tiles or specific layers

- New Key getDuration method and key handling updates

- You
can now exclude the whole sound API using a new webpack flag

There are lots more than this, but those are some juicy tidbits that should get you interested in trying out the upgrade. You can read about the headline items in this news post.

Download Phaser 3.17.0 from the GitHub releases page, or pull it in via the npm package. There are also the pre-built files available on the jsDeliver CDN.

The brand new 3.17 documentation has been updated and published. As before, you can read it online, or check out the repo to have a local copy.

The TypeScript definitions have been improved further still and are now available in the main Phaser repo. So, there's no need to clone another repo when updating Phaser any longer.

As always, please check out the Change Log for comprehensive details about what recent versions contain.
Phaser 3.18

There's no rest for the wicked. Or game framework developers it would appear. The paint may still be dripping wet on 3.17 but I've already started planning what 3.18 will focus on.

First of all, I'm going to finish off the Spine Plugin. There are a couple of lingering tasks left to do, and some mask tests to perform, but otherwise, it's working as expected. One of the most significant changes I need to investigate is batching together Spine game objects. Currently, when Phaser encounters a Spine object it shuts down the renderer, flushes out the batch, sets-up all of the new shaders and gets the pipeline ready for Spine to do its thing. When it's finished, this is all restored again ready for the next Game Object. If that next Game Object is another Spine object, it repeats all of the same tasks yet again.

As you can appreciate, this has an impact on performance, as shaders and textures are swapped in and out. Part of my work on finishing Spine will be seeing if there is a way the Spine runtimes can be batched to avoid this. The core Phaser render loop will also need updating so it has knowledge not only of what it's currently rendering, but what is next in the stack too. I'm quite looking forward to doing this work as it's scratching a satisfyingly geeky itch.

I'm also going to give the GitHub issues list a bit of a sort. There are lots of issues in there unrelated to the API, such as reporting broken examples, mixed in with lots of genuine issues. This is definitely less fun work :) Yet sadly unavoidable. It's like an unwritten contract between myself and those who support Phaser: that issues will eventually get looked at (no matter how impractical some of them are).

One of the things I really want to do is get Arcade Physics 2 finished. I spent weeks working on it and it was so much better that it genuinely hurt to have to remove it from the 3.17 release. But it wasn't quite right and I couldn't include it as it was. In hindsight, this was the right decision. Although the API is mostly the same it's different enough that it could potentially break a lot of in-production titles. So I'm going to think carefully about how it's released, possibly putting it under a different namespace, or at least offering some way of swapping back to AP1 if needed. I'll give it some more thought and ask the community before doing anything drastic.

There is also all of the work I did on the new Phaser docs site that needs finishing. Some days, it really does feel like juggling spinning plates. Then, of course, I need to tackle the update to Containers across the core API. This is a
large piece of work, and Spine takes priority for now, so I don't anticipate starting it until 3.19, but it will definitely be started soon. For now, I hope you enjoy updating to 3.17 and creating great games with it :)  

The Shader Game Object

For those of you familiar with Phaser 2, it lets you set a 'filter' on a Display Object, which causes it to be rendered with your own custom shader, rather than the default one. In Phaser 3 we offered the ability to create your own custom WebGL pipeline, which was extremely powerful, but equally quite daunting. And also overkill, if all you want to do was splash a shader on-screen for a background-effect or similar.

As I was working through updating the masks in 3.17 I needed to change how shaders were handled and it made a lot of sense to just expose that in a simple Game Object. Enter, the Shader Game Object. At its heart, it's just a basic quad that uses its own shader to render with, rather than the current pipeline. However, it also forces the WebGL viewport to the dimensions of the Game Object, which means you can do things you've never easily been able to do before (in v2 or with custom pipelines), such as rotate, scale or position shaders anywhere in the game world. This may sound trivial, but if you recall how it worked previously you would need to pump a whole raft of uniforms into your shader to handle the projection matrix required for this, where-as 3.17 does it for you.

The Shader Game Object is backed by a new BaseShader object. This is a light-weight object containing just the fragment and vertex source for the shader, along with an object defining any uniforms it may have. These BaseShaders are now created automatically by the GLSL File Loader whenever you load a shader into Phaser, and they're stored in the Shader Cache. Let's take a look at the process. First, we'll create a basic fragment shader. The following GLSL shader code is saved to a file called marble.glsl.js (you can find all of these in the Phaser 3 Examples repo):
precision medium float;
uniform vec2 resolution;
uniform float time;

varying vec2 fragCoord;

void main()
{
    vec2 p = (2.0 * fragCoord.xy - resolution) / max(resolution.x, resolution.y);

    for (int i = 3; i < 30; i++)
    {
        vec2 newp = p + time * 0.01;
        newp.x = 0.6 / float(i) * sin(float(i) * p.y + time / 50.0 + 0.3 * float(i)) + 0.5;
        newp.y = 0.6 / float(i) * sin(float(i) * p.x + time / 25.0 + 0.3 * float(i + 30)) - 0.5;
        p = newp;
    }

    vec3 col = vec3(0.8 * sin(3.0 * p.x) + 0.8, 0.8 * sin(3.0 * p.y) + 0.8, 0.8 * sin(p.x + p.y) + 0.8);

    gl_FragColor = vec4(col, 1.0);
}

Now let’s load this into a Phaser game and display it. The following code should be simple enough to understand, as it uses a very similar structure to every other Game Object Phaser has:

var config = {
    type: Phaser.WEBGL,
    parent: 'phaser-example',
    width: 800,
    height: 600,
    scene: {
        preload: preload,
        create: create
    }
};

var game = new Phaser.Game(config);

function preload ()
{
    this.load.glsl('marble', 'assets/shaders/marble.glsl.js');
}

function create ()
{
    var shader = this.add.shader('marble', 400, 300, 800, 600);
}

This adds a new Shader object into the display list. As with all other Game Objects, it's origin is the center by default. The 800, 600 values are the size of it. In this case, it'll fill the complete game when run. Let's take a look at our shader in action:
Click the screenshot to see it running in all its pretty glory :) If you look at the GLSL code again you'll notice it has a few uniforms defined in it. This shader uses 'time' and 'resolution' along with the varying 'fragcoord'. These are fed into the shader automatically by the default vertex shader code that Phaser uses, allow you to take advantage of them right away. If you want you can, of course, provide your own vertex and fragment source and set-up your own uniforms, but for the sake of this example, we'll use the defaults.

Now, let's do something to the shader. Add in the code `shader.setAngle(20)` and run it again. As you may expect, you'll now see this:
The area being drawn to by the shader has rotated and the fragment code has picked up these new coordinates and is using them. We can scale it, too. Here's a more complete demo. Click the screenshot to run it and then click anywhere to start the tweens running:
Woohoo! A fully transforming shader. The bird is just to demonstrate the interaction of a shader on the display list along with a standard Image Game Object. Plus, it looks cute.

Shaders don't have to be loaded from files. You can also create BaseShader objects directly using strings and then create Shader Game Objects in the same way as before, using the shader cache key as its reference. You can also bundle multiple shaders together in a single GLSL file using a block of frontmatter to separate them. Here is an example

```glsl
---
name: Rainbow
type: fragment
---
precision mediump float;
uniform float time;
uniform vec2 resolution;
uniform vec2 mouse;
varying vec2 fragCoord;
void main(void)
{
  vec2 uv = fragCoord.xy / resolution.xy;
  vec3 col = uv.xyz;
  gl_FragColor = vec4(col, 1.0);
}
---
name: Tunnel
type: fragment
uniform alpha: { "type": "if", "value": 1.0 }
uniform origin: { "type": "if", "value": 2.0 }
uniform iChannel0: { "type": "sampler2D", "value": null, "textureData": { "repeat": true } }
---
precision mediump float;
uniform float time;
uniform vec2 resolution;
uniform sampler2D iChannel0;
uniform float alpha;
uniform float origin;
varying vec2 fragCoord;
#define S 0.79577471545 // Precalculated 2.5 / PI
#define E 0.0001
void main(void)
{
  vec2 p = (origin + fragCoord.xy) / resolution.xy - 1.0) * vec2(resolution.x / resolution.y, 1.0);
  vec2 t = vec2(S * atan(p.x, p.y), 1.0 / max(length(p), E));
  vec3 c = texture2D(iChannel0, t + vec2(time * 0.1, time)).xyz;
  gl_FragColor = vec4(c / (t.y + 0.5), alpha);
}
```

You can find example shader bundle files in the Phaser 3 Examples repo (in the assets/shaders folder). They're just plain text files, with a simple block of frontmatter dividing the shaders up. In the example shown above, you can hopefully make-out a shader called 'Rainbow', followed by another called 'Tunnel' which has its own set of uniforms for the alpha, origin and a sampler2D, which
means it uses an external texture as part of the shader code. It's pretty powerful stuff and I'll cover it in more depth in a future dev log.
If you support **Phaser on Patreon** then in May's Code Bundle you will already
have 14 different examples of using the new Shader system, including shaders under mouse control, using them as masks and more. If you support Phaser today, you can download the pack instantly, along with all previously released packs too. I will, of course, release these examples onto the public site at some point, but for now, backers get them first because it's their awesome support that made this release even possible.

I hope you enjoy playing with 3.17. Please let me know how you get on. If you find bugs, open an issue with a test case and I'll look into it. And when you create something beautiful, send it to me so I can feature it here.

This is brilliant :) Ben Wiley has recreated the Windows 95 Media Player purely in the web!

I love Custom PC magazine and have been reading it for years. They recently changed publisher and the new look is fantastic. Plus, they've thousands of copies to give away for free!

Microsoft has open-sourced the new and original Windows Terminal apps.

Phaser Releases
Phaser 3.17.0 released 10th May 2019.  
Phaser CE 2.13.0 released 11th May 2019.  

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.