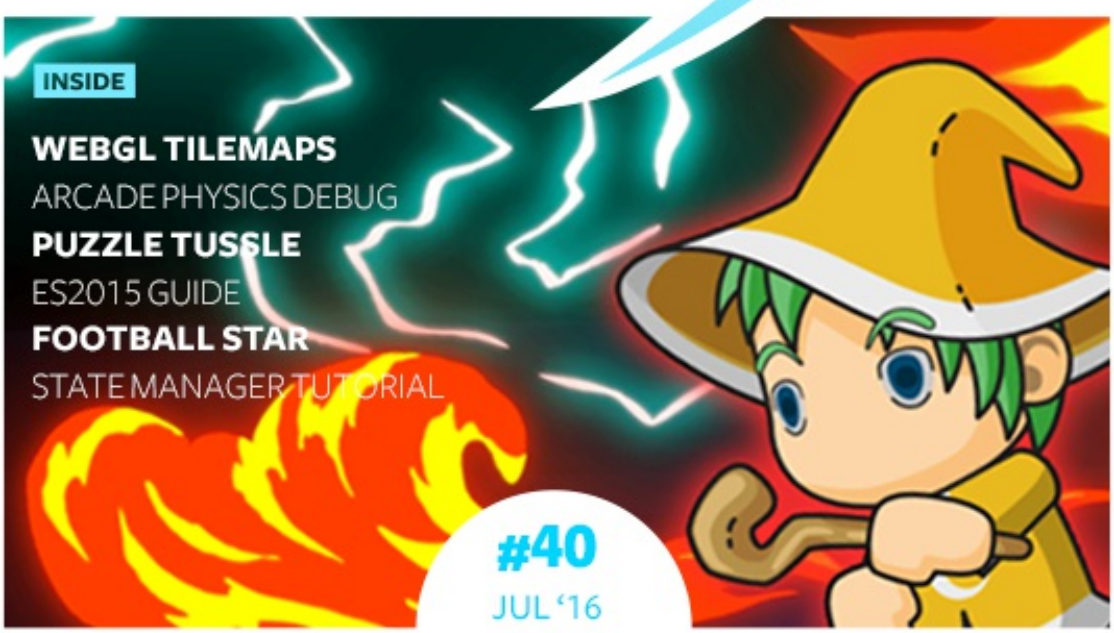
PHASER WORLD



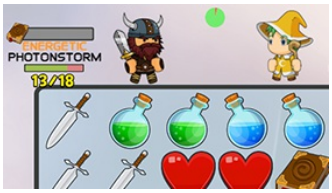
Welcome to Issue 40 of Phaser World

Woohoo, issue 40! That's a nice milestone to have reached :) and it's a pretty sweet issue this week too! On the games front *Puzzle Tussle* is a ridiculously fun match 3 RPG, that I was absorbed in for far too long. And *Football Star* just goes to show what you can achieve with a single button and a great concept.

The biggest news this week though is all happening in the Development Progress section. There's a brand new version of Phaser for you all to test, lots of great new features, and a rough outline of what's on the horizon. Dig in, read, play and let me know what you think. Next issue I'll be outling a new roadmap for Phaser too.

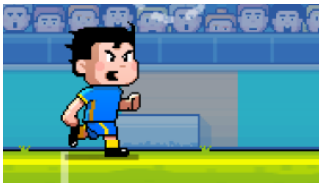
Until the next issue, keep on coding. Drop me a line if you've got any news you'd like featured (you can just reply to this email) or grab me on the Phaser Slack channel.

Games made with Phaser



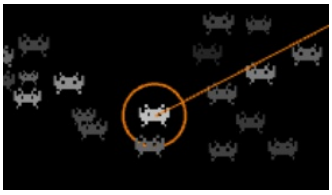
Puzzle Tussle

Game of the Week
A match 3 RPG, with deep strategy, upgrades, skills and real-time PVP.



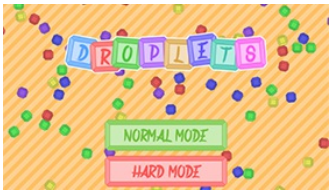
Football Star

Staff Pick
A fun one-button football high-score challenge game.



CiTIUS Invaders

An old-style arcade game to learn evolutionary algorithms and genetic programming.



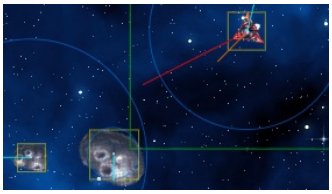
Droplets

Avoid being crushed by remembering the colors from the previous waves.

Pirates Bomb Rush

Rush back and forth, collecting grog and avoiding bombs!

Phaser Tutorials

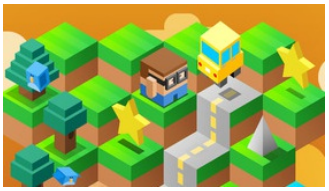


Debug Arcade Physics Plugin

Visually debug all kinds of Arcade Physics properties with this awesome plugin.

ES2015 Getting Started Tutorial

Getting Started with Phaser and ES2015.



[Down The Mountain Tutorial]

Adding in smooth jumping motion using bezier curves.

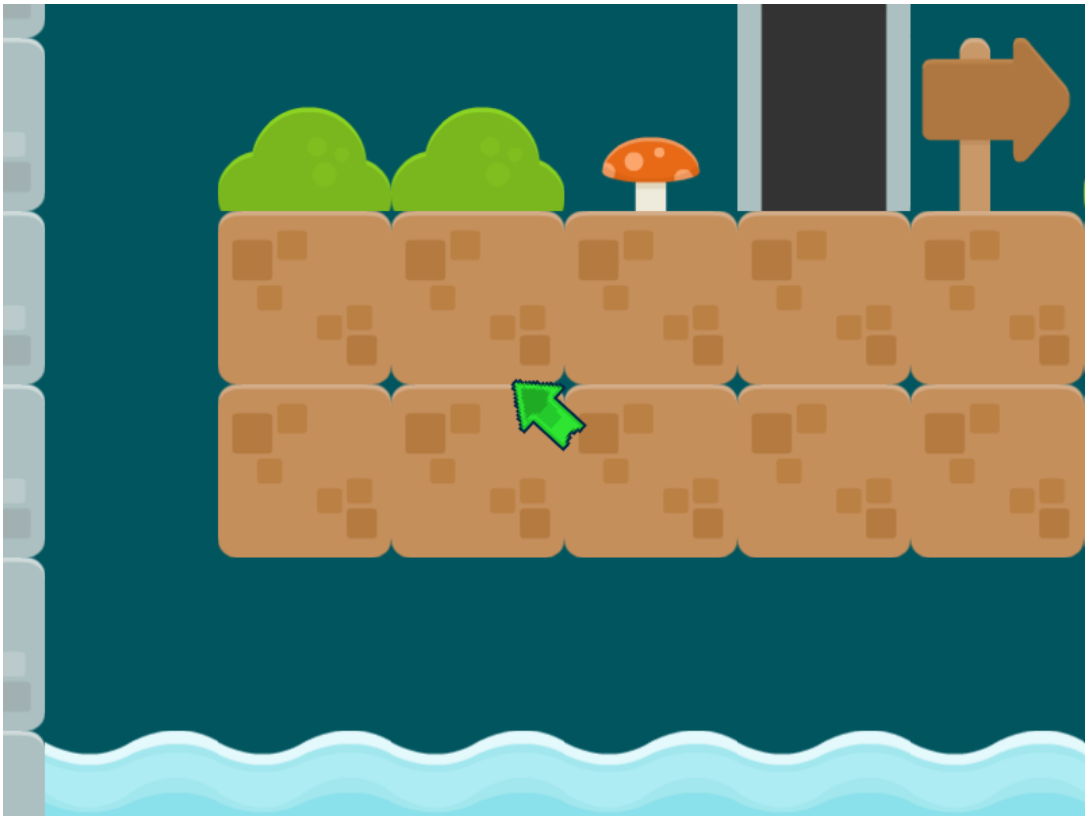
[Getting Started with Phaser]

A new video and tutorial series about app development with Phaser.

[State Manager Tutorial]

In this TechnoTip video tutorial they cover the State Manager.

Development Progress



It's been one heck of a week in the Phaser HQ, and killer new features are flying off the production line at quite a rate. Here is what's new:

WebGL Tilemap Renderer - Beta 2 Ready

At the start of the week we released Phaser 2.7.0 Beta 1. This had the brand new WebGL Tilemap renderer built into it. Pete worked really hard to ensure the shader was powerful, and stupidly fast, and his hard work has paid off.

I took his work and then spent a long time sorting out the implementation, so it was as friendly as possible, and also making some much needed tweaks along the way. And as a result Beta 2 is now ready to download and test.

You can download pre-built versions of Phaser 2.7.0 from the GitHub dev branch. This is a drop-in replacement for any game using Tilemaps under WebGL. Note that 2.7.0 is an upgraded version of Phaser 2.6.1, so if your game doesn't yet run under 2.6.1 you'll need to remedy that, before you can test out the new tilemaps.

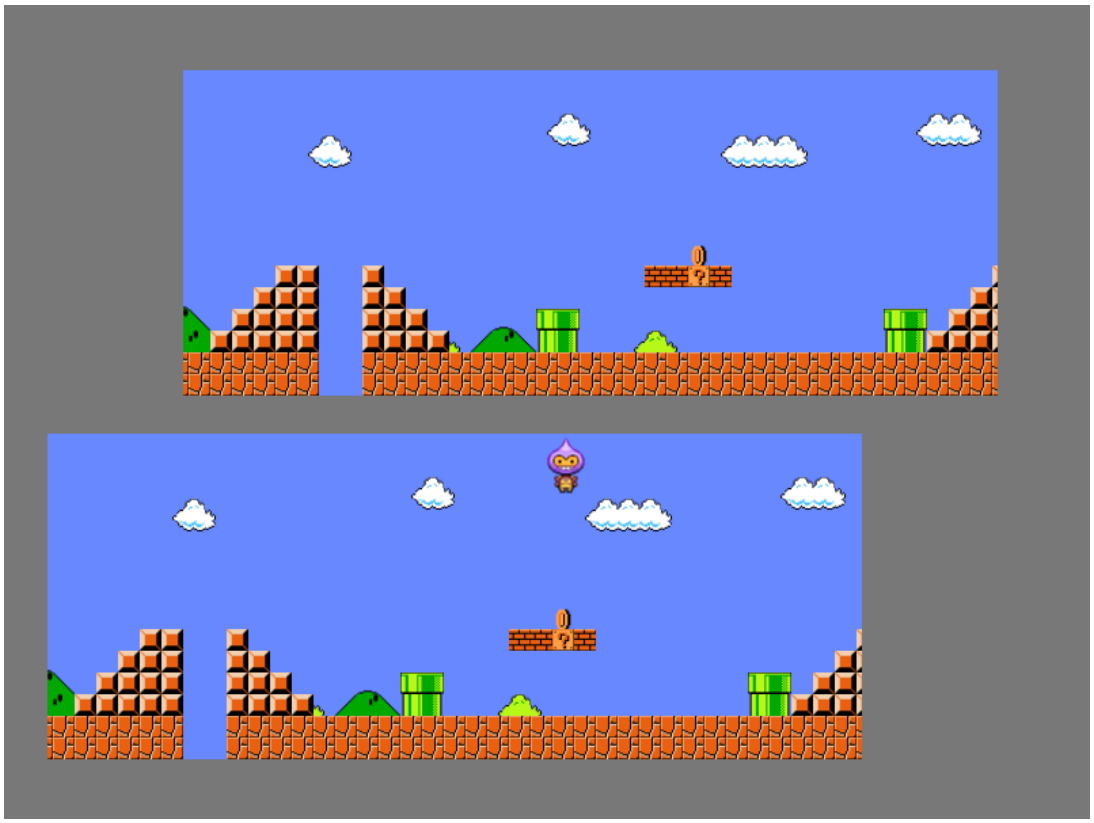
Download Phaser 2.7.0 Beta 2

Once you've had a chance to test it, please report your findings in this Phaser Forum thread, or open a GitHub Issue. We're especially interested to know what performance you get on very low-end GPUs or mobile hardware. But it's safe to say we're seeing silky smooth 60fps rendering on a vast range of desktops now.

New Tilemap Features

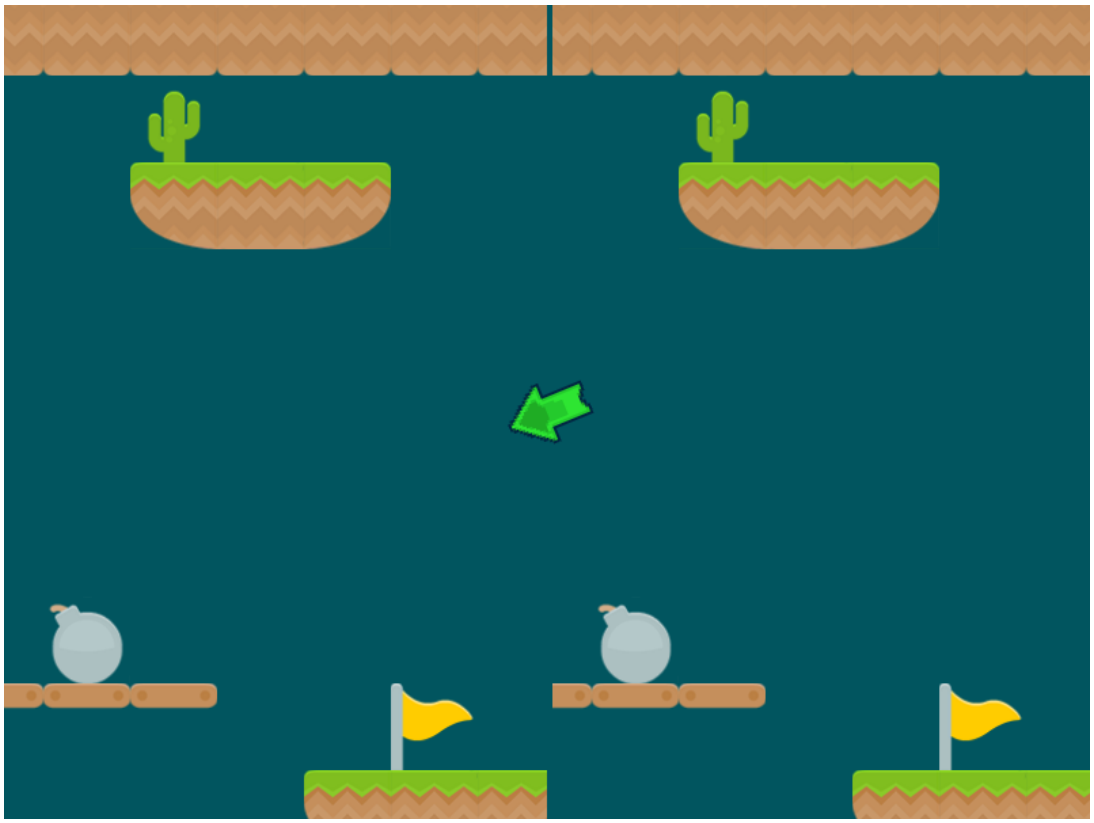
Aside from using WebGL shaders for the tilemap, what else can it now do?

You can now set the display dimensions of a Tilemap Layer, and you can use the x and y properties to place the map anywhere you like on the screen. This means you can create a smaller 'view' into a tilemap, and yet it will still respond to collision events like normal. The following example demonstates this:



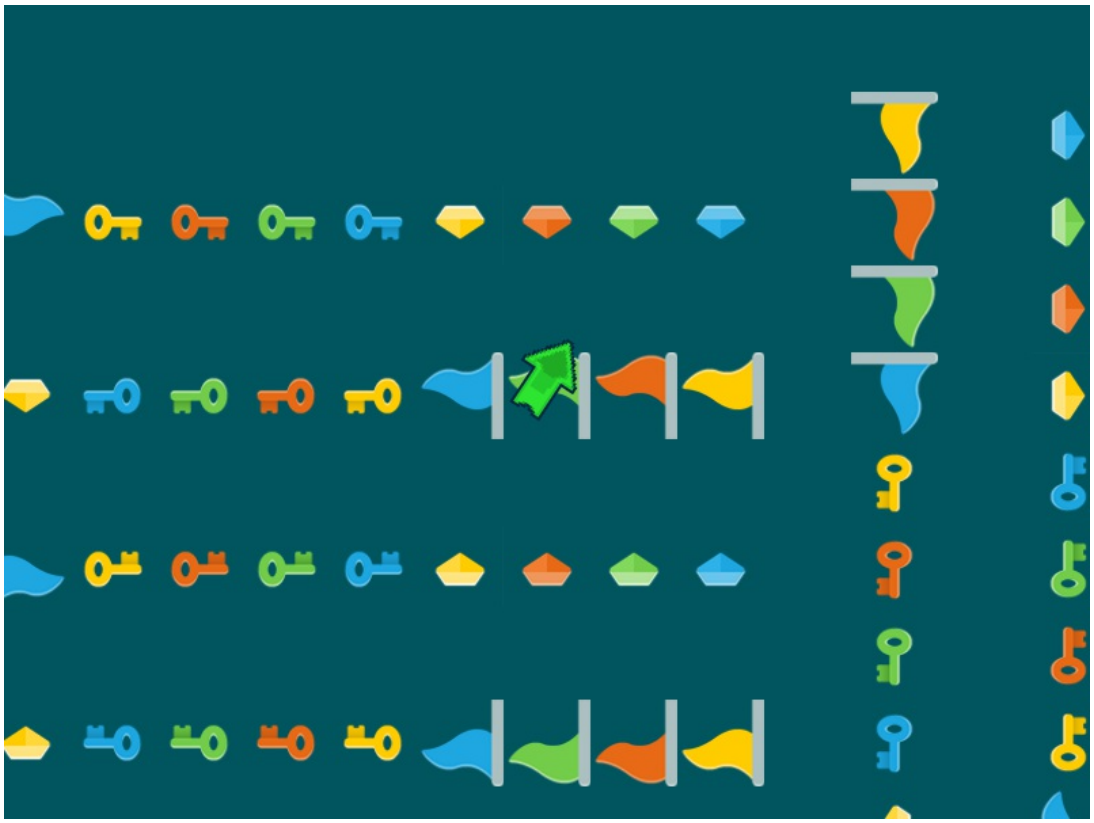
In the example above you can run and jump around the map, dropping happily from the view at the top into the view at the bottom, and bouncing between them, even though they're offset. What you're seeing here are two views into the exact same Tilemap Layer, with each view under your control in terms of size and position.

This allows you to do things like dual play fields:



Being a WebGL shader it smartly batches draw calls, even when using a map with tiles from split tilesets. So it's capable of rendering the entire map in a single draw call per tileset, no matter how interleaved those tiles are. This means you can have really big playfields, with no performance hit any more. Yay!

The new renderer of course supports all of the various flipped tile options you can set in Tiled. The following example uses just one tileset, but with various states of mirroring and flipping on the tiles. It also supports the new Camera Shake too - click the example to see :)



What's Next?

Please help test out Beta 2. It should be a drop-in replacement for pretty much all games, although you will need to use layer.x and layer.y to position a layer now. See the examples above for code.

There are a few more things I'm going to add before this hits release. First is the ability to show the debug tiles in WebGL mode (currently it only works in Canvas). And secondly I want to add the the ability to scroll a tilemap fully independant of the camera. For now, have a play and enjoy the speed :)

Multi-Batch Support

I talked about this feature last issue, and Felipe has been hard at work, re-coding all of the Pixi shaders and WebGL renderer to support it. Here is his update:

This new feature will allow you to batch multiple textures on a single draw call. Why is this good? The old way every time you switched textures it meant flushing the current batch before binding the new texture, this of course means submitting vertex and index buffers, and doing a draw call. This could become a huge overhead, especially when dealing with a lot of textures.

Our new implementation detects how many textures your GPU can support at once, and adjust the shaders to use that that.

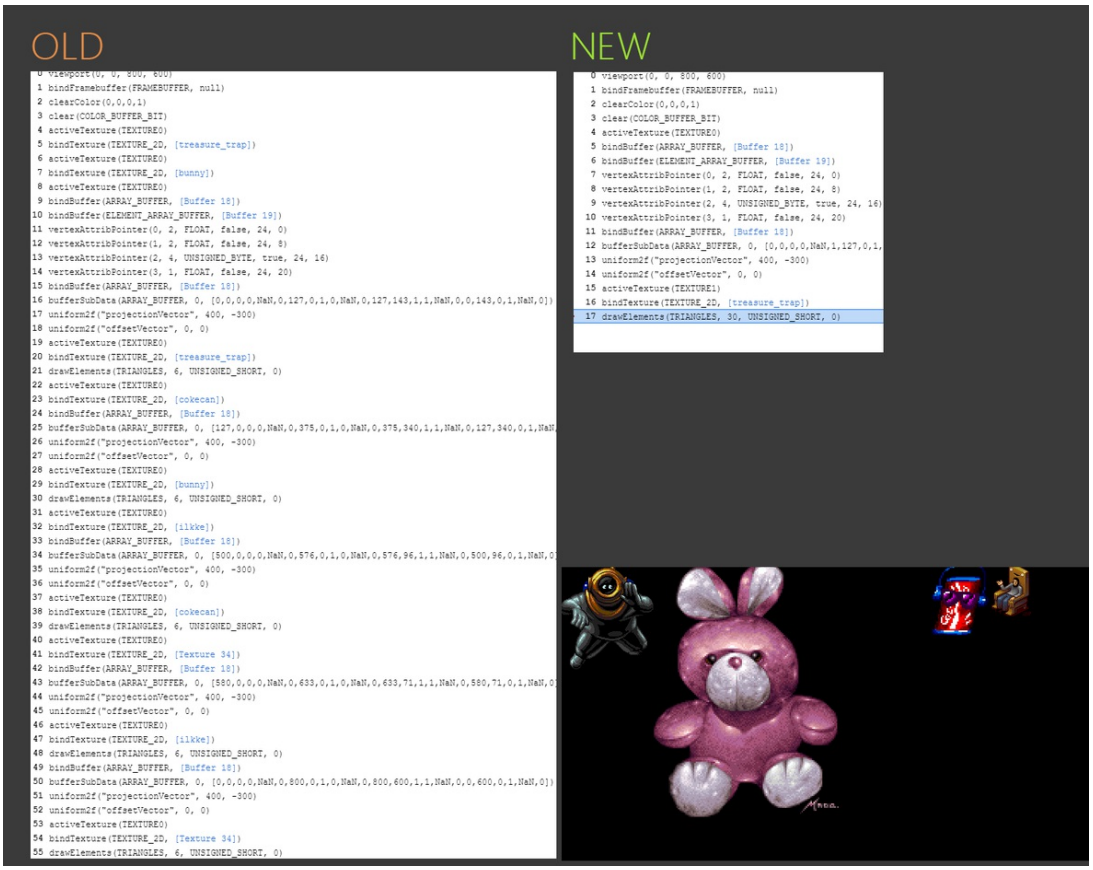
How can you take advantage of this new feature? There are two ways. One is using the new function **WebGLRenderer.setTexturePriority** and the other is by setting the property **textureIndex** on the base texture to the texture unit you want. The second option gives you more power and control, but it's up to you to be aware of the limits of the platform. If you define a texture index bigger than the system can support the shader will by default draw a pink rectangle. The first approach however is fully guarded against that.

You can see the benefit here very clearly. If you look at the following code:

```
function preload() {
    game.load.image('treasure_trap', 'assets/sprites/treasure_trap.png');
    game.load.image('bunny', 'assets/sprites/bunny.png');
    game.load.image('cokecan', 'assets/sprites/cokecan.png');
    game.load.image('ilkke', 'assets/sprites/ilkke.png');
}

function create() {
    // Use this helper function to set texture priority
    game.renderer.setTexturePriority(['treasure_trap', 'bunny', 'cokecan', 'ilkke']);

    var spriteA = window.spriteA = game.add.sprite(0, 0, 'treasure_trap');
    var spriteB = window.spriteB = game.add.sprite(127, 0, 'bunny');
    var spriteC = window.spriteB = game.add.sprite(500, 0, 'cokecan');
    var spriteD = window.spriteB = game.add.sprite(580, 0, 'ilkke');

    // You can also set it manually by accessing the base texture
    spriteA.texture.baseTexture.textureIndex = 5;
}
```

Here you can see just 4 sprites added to your game, but each one using a different texture.

During the course of one single frame look how many draw calls are done on the old version, compared to how few are done with this new feature:



Now imagine the difference this will make when you use texture atlases! It will be entirely possible to render your entire game in a single draw call with effective use of this new feature and atlases.

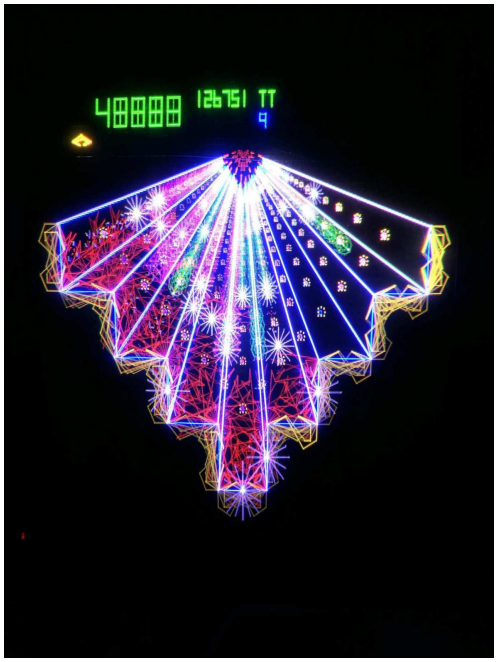
You can track development of Multi Texture support in its GitHub branch, but we will be rolling out a public beta for it next week.

Geeky Links

Look at this beautiful time-lapse photos of Atari's classic arcade game Tempest.

Web MSX is a superb web-based MSX 8-bit emulator. I used to love the MSX, so many classic Konami games on it. Give it a whirl :)

Finally check out this video of Mega Processor. Words don't do it justice, best to just watch and marvel!



Phaser Releases

The current version of Phaser is 2.6.1 released on July 11th 2016.

Phaser 2.7.0 is in development in the GitHub dev branch.

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.