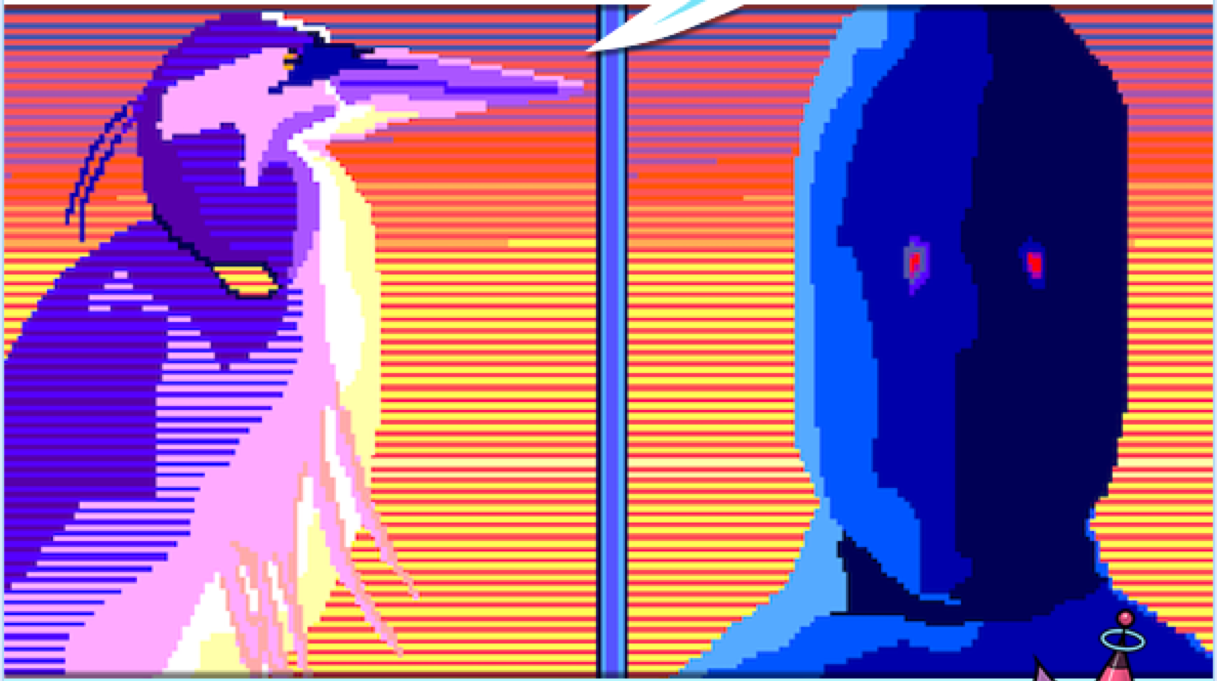
PHASER WORLD

MAY 2017

ISSUE 81



CONTENTS

NORCO: FARAWAY LIGHTS

FLEXCALE PLUGIN

NATURE BASKETBALL

ZHED PROTOTYPE TUTORIAL

FLOOD ESCAPE

MARIO AND LINK COMPONENTS

Welcome to Issue 81 of Phaser World

After two issues of the newsletter last week, I'm glad we're back to just one :) It has meant we've had the whole week to focus on working on Phaser 3, and there have been loads of great updates pushed to the repo this week. You can read about the new Mesh Game Objects and the snapshot feature this issue.

We've also a bunch of new games and tutorials. Norco was a delight to play and I eagerly await the full version. Given that it's all the work of a single person it's amazing what can be achieved on your own sometimes.

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.

Games made with Phaser

Norco: Faraway Lights

Game of the Week

A cyberpunk adventure from the deep south, set in a rich pixelated universe of bizarre conspiracies, driven by synth magic.

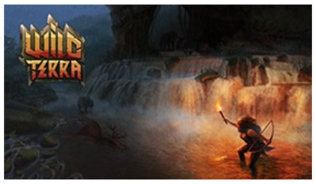
Nature Basketball

Staff Pick

Take aim and release! See how many you can sink in this addictive basketball flinger.

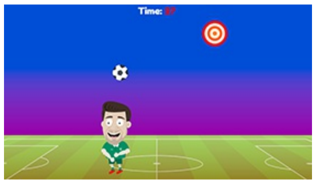
Flood Escape

Avoid the constantly rising waves by building up the tower. Be quick, but don't make too many mistakes!



Wild Terra Corrupted Lands

A trailer for the new Corrupted Lands expansion for Wild Terra.



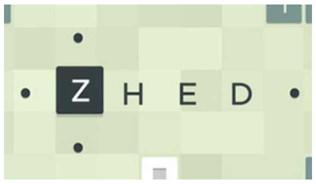
Long Shots

Help Shane Long practice for the big game. See how many targets you can hit in 30 seconds.

Phaser News & Tutorials

Phaser Mario Component

A player controlled Game Object with built-in movement, jumping and health handling.



Zhed Prototype Tutorial

Create a 10 level prototype based on the hit iOS game Zhed.

Flexcale Plugin

This Phaser plugin helps make your game responsive and free of black bars.

Creating Games with Phaser Part 3

In Part 3 of this new video based tutorial course learn about collision detection and user input.

Phaser Link Component

A player controlled Game Object with movement controls based on Link from the Zelda games.

Patreon Updates

Thank you and welcome to the following awesome people who joined the Phaser Patreon this week: **Mario Macari**. Also thank you to **Matthieu Desprez** for the donation.

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.

Development Progress

Last week I told you how I was waiting to hear back from Mozilla about my funding application for Phaser 3. Well, an hour after sending the newsletter and I got a reply. The reply was basically asking for a detailed technical schedule and told me they would be making a decision at the next funding meeting in a months time. So, that's good news in a way. It means it hasn't been rejected, and I now also know roughly when to expect to hear. I'm going to remain positive and carry on working on v3, and cross the Mozilla shaped bridge when we come to it.

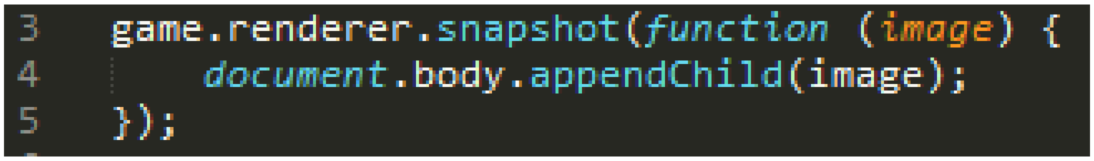
Phaser 3 Updates

This week both Felipe and I have managed to get a complete week of work in on v3. No public holidays, no DDoS attacks, just pure coding. I've been working on the Tween Manager, and Felipe has been creating a new Game Objected called a Mesh. As we're limited in space in the newsletter I'm going to cover the Mesh this week, and Tweens next week.

Make it Snappy

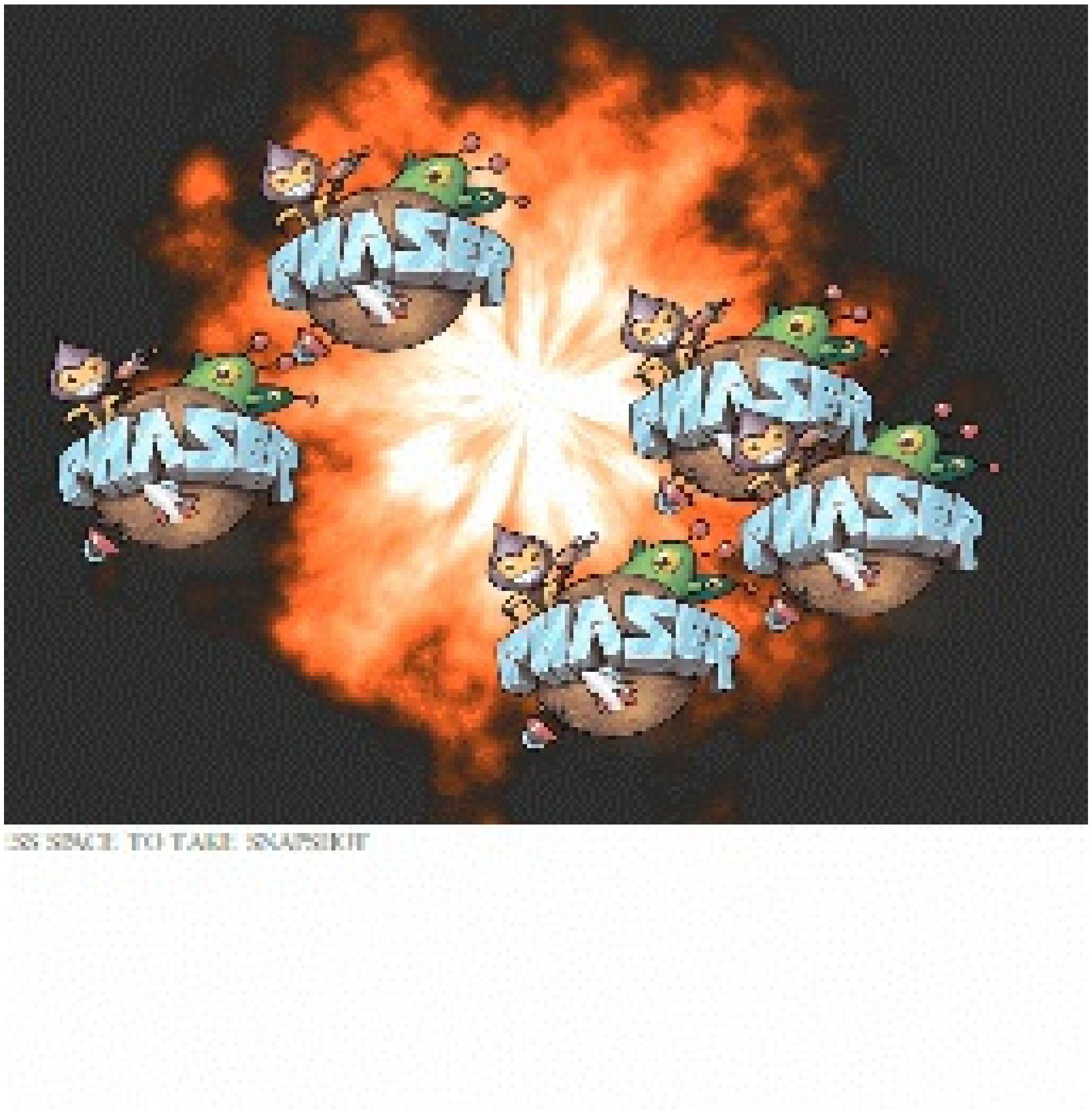
We've built into the renderers the ability for you to take a snapshot of the current canvas, and save it to an image object. No matter what is being rendered: Shaders, Graphics, sprites, text - it will happily grab it. You can then use the resulting image either within your game or even save it to the filesystem. For example, if you were creating an art package it would allow you to grab the image that has been drawn.

Taking a snapshot is very simple, it's just a single call that requires a callback function, which is invoked once the frame has finished rendering:



```
game.renderer.snapshot(function (image) {
    document.body.appendChild(image);
});
```

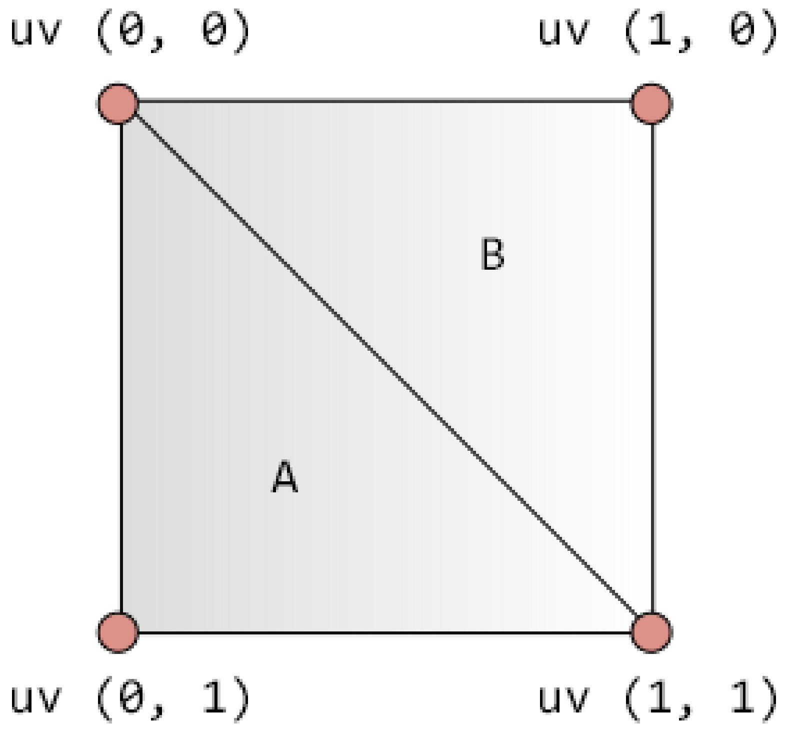
In the code above we take a snapshot of the canvas, then append it to the document. As always, click the screen shots to run the demos in browser:



The main reason we added the snapshot feature was so we can use it internally to provide you with smooth transitions from one State to another. But, we exposed it in the API so you can use it in other ways too. Always great to kill two birds with one stone!

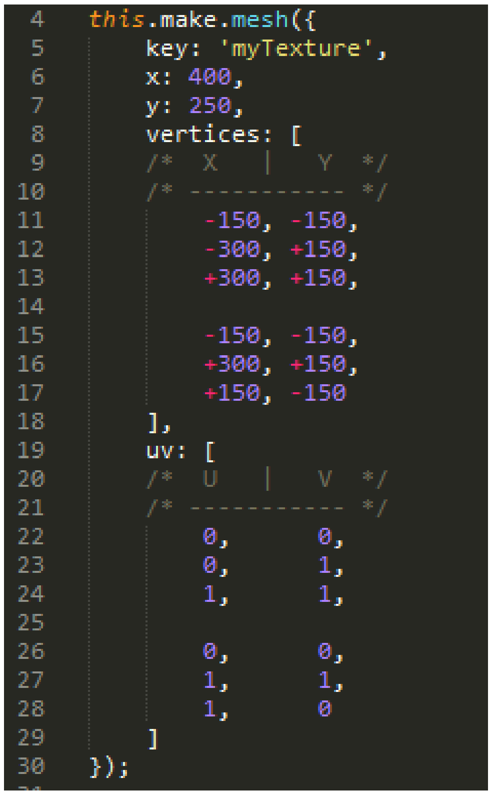
New Mesh Game Object

The biggest update this week is the addition of the new Mesh Game Object. This is a WebGL only feature and an extremely powerful one. When you render a standard Sprite in WebGL it consists of 2 triangles drawn as a quad:



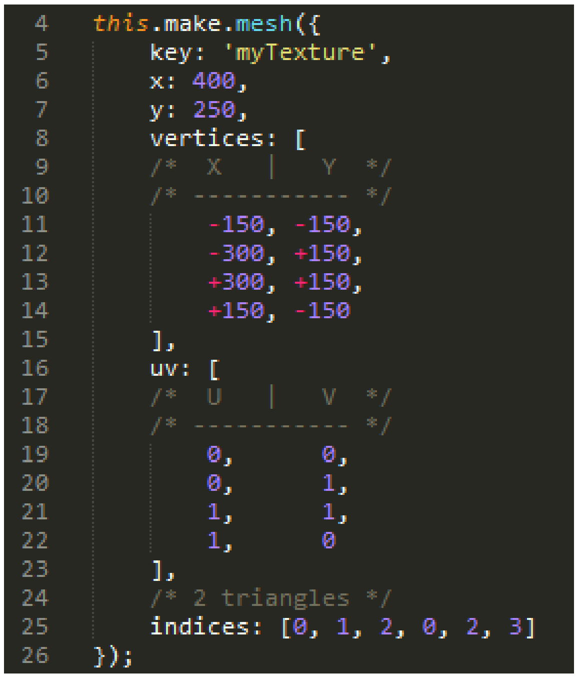
The texture is mapped using the UV coordinates, assigning half of the texture to each triangle. Because they are joined together the image appears as a whole. When you use Sprites in Phaser it manages all of this for you, so you just move a sprite around the screen and all the triangle vertices and UV mapping is handled internally.

What a Mesh allows you to do is break these rules. When creating a mesh you need to pass the vertex positions and texture mapping coordinates as two arrays. Each Mesh has its own transform, so you can position, rotate or scale it without the need to update every vertex yourself. Here is the code for creating a simple quad using the Mesh:



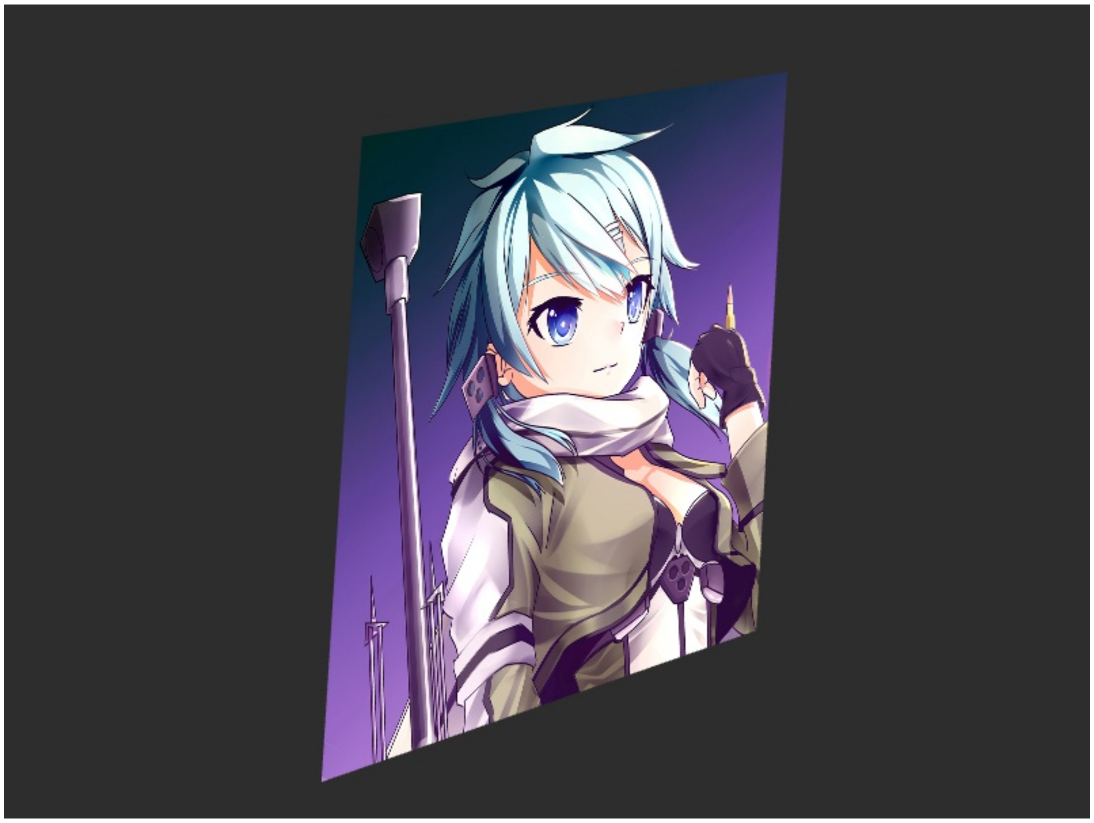
```
this.make.mesh({
    key: 'myTexture',
    x: 400,
    y: 250,
    vertices: [
    /*  X    |    Y  */
    /* ----------- */
        -150, -150,
        -300, +150,
        +300, +150,

        -150, -150,
        +300, +150,
        +150, -150
    ],
    uv: [
    /*  U    |    V  */
    /* ----------- */
        0,       0,
        0,       1,
        1,       1,

        0,       0,
        1,       1,
        1,       0
    ]
});
```

Writing out each vertex by hand can be tedious, especially when dealing with big meshes. That is why one of the properties of a Mesh are the indices. It allows developers to describe the triangles using indices that map to the vertex array. The code above can now be presented like this:



```
this.make.mesh({
    key: 'myTexture',
    x: 400,
    y: 250,
    vertices: [
    /*  X    |    Y  */
    /* ----------- */
        -150, -150,
        -300, +150,
        +300, +150,
        +150, -150
    ],
    uv: [
    /*  U    |    V  */
    /* ----------- */
        0,       0,
        0,       1,
        1,       1,
        1,       0
    ],
    /* 2 triangles */
    indices: [0, 1, 2, 0, 2, 3]
});
```

This will help you remove redundant vertex definitions. Here is a demo where each vertice is updated using a tween, creating a nice 'push / pull' effect on the image:



In the coming day's we will be adding a new Quad Game Object, which extends the base Mesh class and offers you an easier path in, with more friendly access to the vertices. This will be handy for effects such as a scrolling Space Harrier style floor, or a perhaps just adding a bit of perspective flair onto your game logo.

It's important to understand that although the example and code above are based around a single quad, you can actually do so much more than that. They're not just about manipulating quads. There are some really advanced things you can do with them. For example, a pseudo-3D environment, or a 2D terrain renderer are now entirely possible.

A Mesh can have up to 8000 vertices and you have complete control over the texture mapping. This means you could create an effect where an image 'shatters' into pieces, like a pane of glass exploding, all using a single mesh. They are also batched. A collection of consecutive Mesh objects on the display list, if sharing the same texture, will be a single draw call.

We look forward to bringing you some more Mesh examples soon, including texture scrolling, and going way beyond a simple quad. Keep your eyes peeled on the newsletter for more!

Phaser 3 Mailing List and Developers Guide

If you're interested in helping evolve the shape of Phaser 3, then please join the Phaser 3 Google Group. The group is for anyone who wishes to discuss what the Phaser 3 API will contain.

The Phaser 3 Developers Guide is available. Essential reading for anyone who'd like to help *build* Phaser 3.

Geek Links

R-Type Encounter is a short animation based on the legendary Irem game. Created by just one person it was released as a teaser a while ago and now here is the full thing. Only 3 minutes long, but it's a glorious 3 minutes!

Connor Krukosky is a college student who collects vintage computers. One day, he decided to buy an IBM z890. Here's what happens when an 18-year-old buys a mainframe.

Finally, a photo retoucher that transforms real-life animals into cubed creatures from Minecraft

---

Phaser Releases

The current version of Phaser CE is 2.7.9 released on May 9th 2017.

Phaser 3.0.0 is in active development in the GitHub v3 folder.

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.