Welcome to Issue 96 of Phaser World

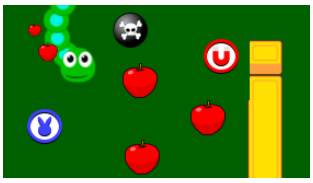
I'm glad to say this issue is back to normal. The normal day of the week and the normal sort of content. Fun games, new tutorials, and a proper Dev Log all about

Phaser 3 Beta 1.

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.

The Latest Games

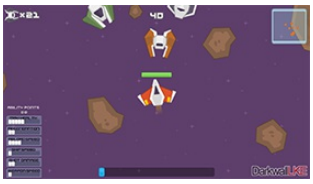
Game of the Week
Copter Rescue
A frustratingly brilliant pixel-perfect action game. Rescue the other copters, if you can.



Staff Pick
Impossible Snake
Slink your way around the tricky levels and munch all the apples in this one-button action puzzle game.

Color Stars

Meet the universe with this colorful puzzle game.



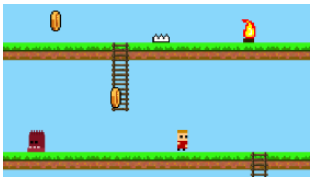
Space 2

Fly around the universe destroying enemy ships and asteroids, using the experience to upgrade.

Flappy Pong

A cute mash-up of Flappy Bird and Pong. See how long you can bounce the ball.

What's New?

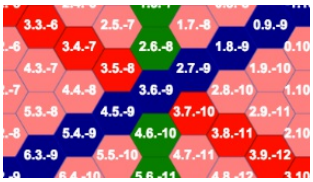


Ladderz

This new endless runner is fully documented and ready to adapt or re-skin for your own purposes.

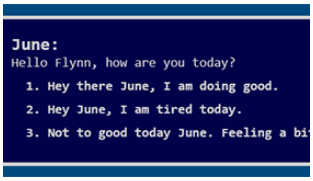
Pocket Debug Plugin

Pocket sized debug module that displays FPS and drawcalls in any DOM (text) element.



Axial Coordinates Tutorial

Introduction to Axial Coordinates for Hexagonal Tile-Based Games.

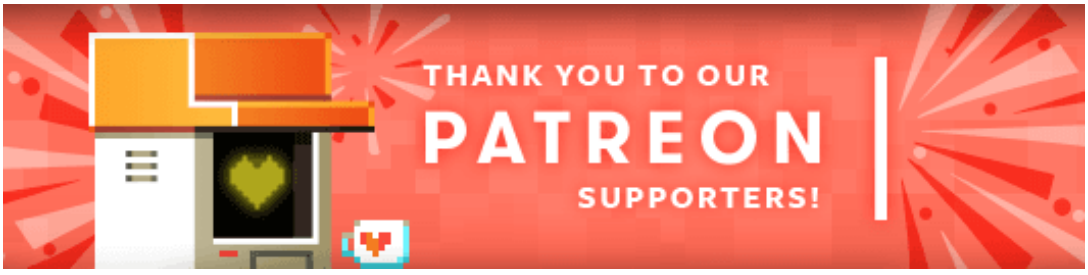


Dialog Manager Plugin

A plugin to easily create dialog trees with actors, choices, answers and more.

Slither.io Tutorial Part 7

In the final part of the tutorial series food is added for the snakes to eat.



Welcome and a massive thank you to the new Phaser Patreons who joined us

this week: **Hank Hsiao** and **Mark Bailey**.

Also, thank you to the following for their donations: **Bruce Ban Herm Chia** and **Paul Kreemer**.

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 #96

I'm pleased to announce that **Phaser 3 Beta 1** is now available. It's a little late due to the antics detailed in Dev Log 95, but it's finally here and ready for testing.

You can grab a build from GitHub or npm using the 'beta' tag. There is a Getting Started Guide available that covers the options, after which I would strongly recommend you hit the Phaser 3 Examples and start tweaking things for yourself.

I'd love your feedback, please. There's a forum thread you can use, or post a GitHub issue, or grab me on Slack.

Tween Timelines

Timelines give you the ability to run multiple tweens either together, or in order and control them all as if they were one single tween. Timelines can be played, paused or stopped, and have their own TimeStep value as well. It's a clean way for you to sequence lots of Tweens and create more complex motions as a result. Here is an example of creating a Timeline. As usual, if the images below are clickable then they link to the example running in the Labs:

```
var timeline = this.tweens.createTimeline();

timeline.add({
    targets: image,
    x: 600,
    ease: 'Power1',
    duration: 3000
});

timeline.add({
    targets: image,
    y: 500,
    ease: 'Power1',
    duration: 3000
});

timeline.add({
    targets: image,
    x: 100,
    ease: 'Power1',
    duration: 3000
});

timeline.add({
    targets: image,
    y: 100,
    ease: 'Power1',
    duration: 3000
});

timeline.play();
```

The method above creates a Timeline object and then uses its 'add' method to add tweens to it. Each tween plays in succession, so in the example above the target image will move in a rectangle motion around the screen before completing.

However, you can define the Timeline using a single config object, rather than the more verbose method above:

```
var timeline = this.tweens.timeline({

    targets: image,
    ease: 'Power1',
    duration: 3000,

    tweens: [{
        x: 600
    },
    {
        y: 500
    },
    {
        x: 100
    },
    {
        y: 100
    }]

});
```

As you can see, it's the exact same end result but with a lot less code. Properties defined in the core of the config object are then given to each 'tween' within the tweens array. Anything specified in the tweens array overrides the default properties, allowing for some very flexible configurations.

Timeline Offsets

Tweens running sequentially are useful but it's also handy to be able to control exactly when those tweens start. You can do this with the **offset** property. There are two ways to use offset: absolute and relative. Here is an example of absolute offsets:

```
var timeline = this.tweens.timeline({

    targets: image,
    ease: 'Linear',
    duration: 3000,

    tweens: [{
        x: 600
    },
    {
        y: 500,
        offset: 2000
    },
    {
        x: 100,
        offset: 4000
    },
    {
        y: 100,
        offset: 6000
    }]

});
```

The example above is the same motion as before except we've applied an *absolute offset* to when the tweens begin. The values given are the ms time the tween will start based on the beginning of the timeline. The duration of each tween is 3000ms, but the second tween in the list starts playing at 2000ms in, so before the first has completed. Run the example to see the difference this makes, the image no longer moves in a rectangle but has its corners sheered off.

You can also use relative offsets:

```
var timeline = this.tweens.timeline({

    targets: image,
    ease: 'Linear',
    duration: 3000,

    tweens: [{
        x: 600
    },
    {
        y: 500,
        offset: '-=500'
    },
    {
        x: 100,
        offset: '-=500'
    },
    {
        y: 100,
        offset: '-=500'
    }]

});
```

A relative offset of '-=500' will start the tween running 500ms before the *previous* tween ends. In the example above because the tween durations are 3000ms, the second tween will start at 3000 - 500, so 2500ms. The offset can be expressed as: +=value, -=value, *=value or /=value.

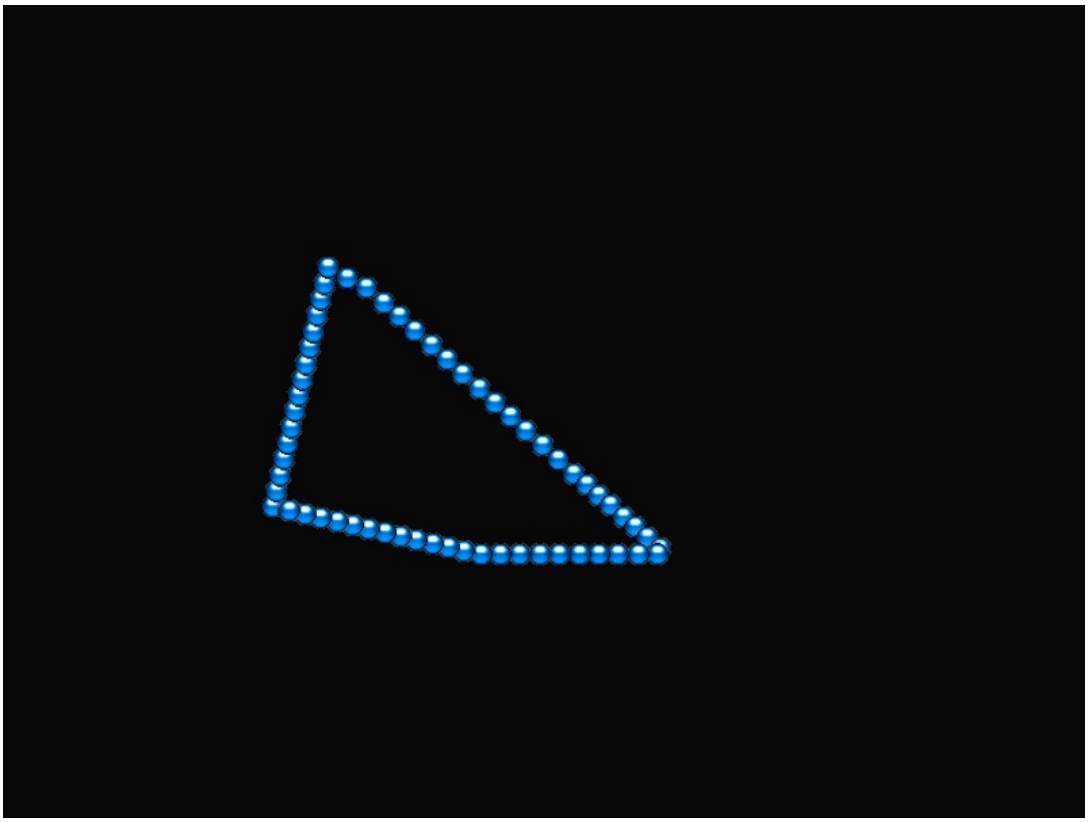
Timeline can also be looped and controlled like normal tweens, so they should give you plenty of scope to experiment and create some nice effects.

Dynamic Properties

Traditionally when you create a tween you define an end value, that is the value you wish the property to tween towards. Once the tween has started that's it, you can't really do anything except wait for it to finish. I was adamant that I would allow you to be able to set fully dynamic start and end values for a tween, which would by extension allow for a lot more creativity. These are now in Beta 1 and are defined like this:

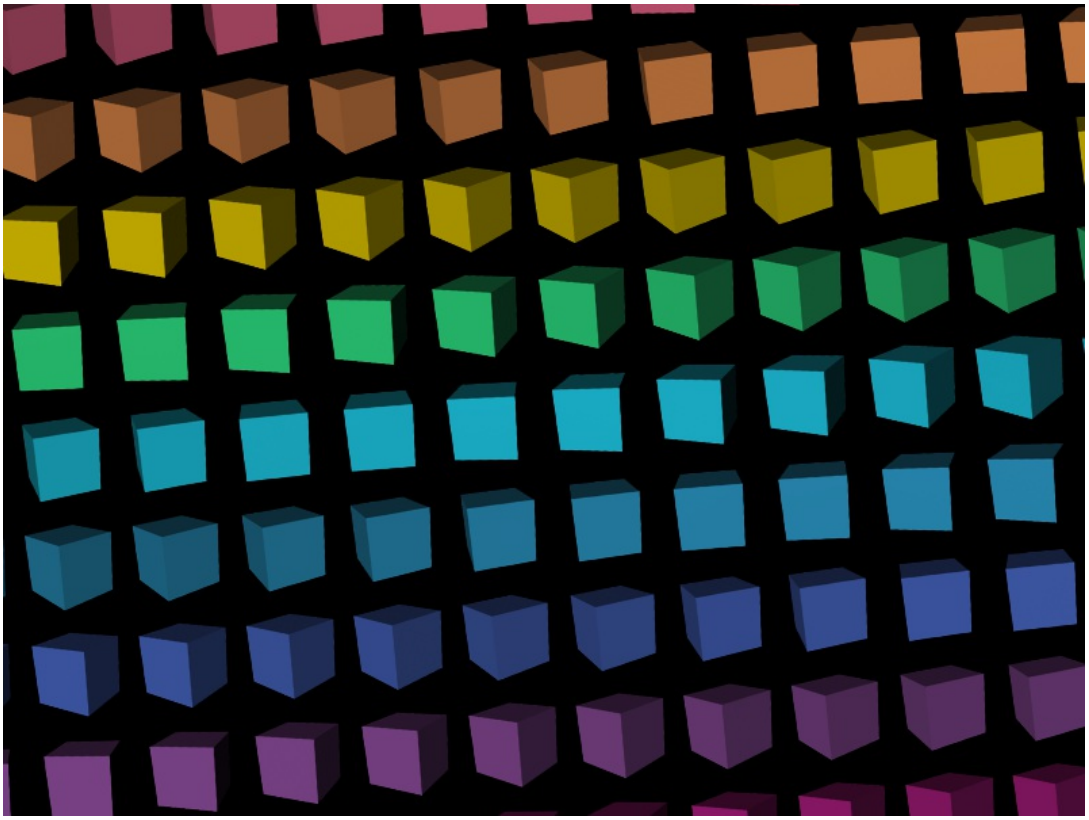
```
var destX = 700;

var tween = this.tweens.add({
    targets: image,
    duration: 500,
    yoyo: true,
    repeat: 8,
    ease: 'Sine.easeInOut',

    x: {

        getEnd: function (target, key, value)
        {
            destX -= 30;

            return destX;
        },

        getStart: function (target, key, value)
        {
            return value + 30;
        }

    }

});
```

If a property (x in this case) has a function 'getEnd' and/or 'getStart' it will use those functions to populate the end and/or starting values as the tween runs. In the example above you'll see the ball move horizontally and slowly get less and less as it settles down into the middle of the screen. This is because the distance it is tweening per repeat is being made smaller by the two functions. It gives an almost physics 'spring like' quality to the sprite without using any physics at all. Combine the above tween with a normal one on the y property and you can create effects like this. Or you can take it even further:

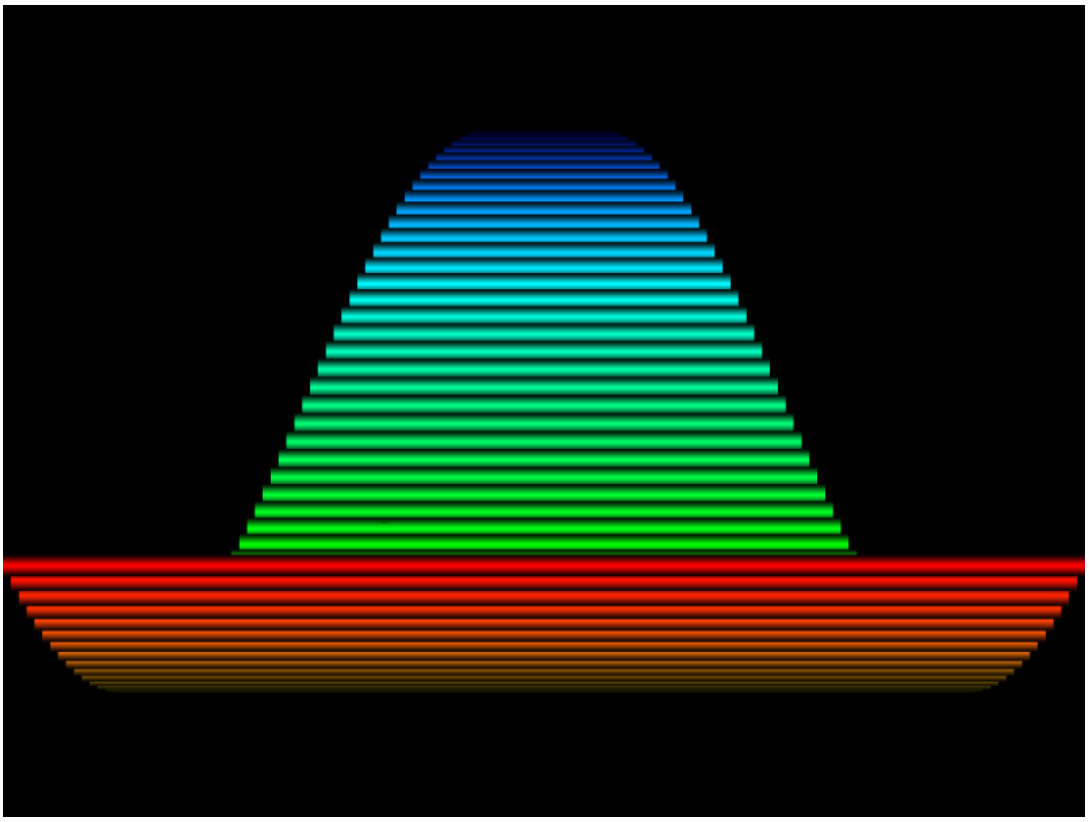


Mighty morphing power shapes!

The above example is just a collection of coordinates that are fed into the dynamic start and end tween values. The result is a fun little shape morphing sequence in just a single tween. Here's another one tweening the zoom and rotation properties of the camera:

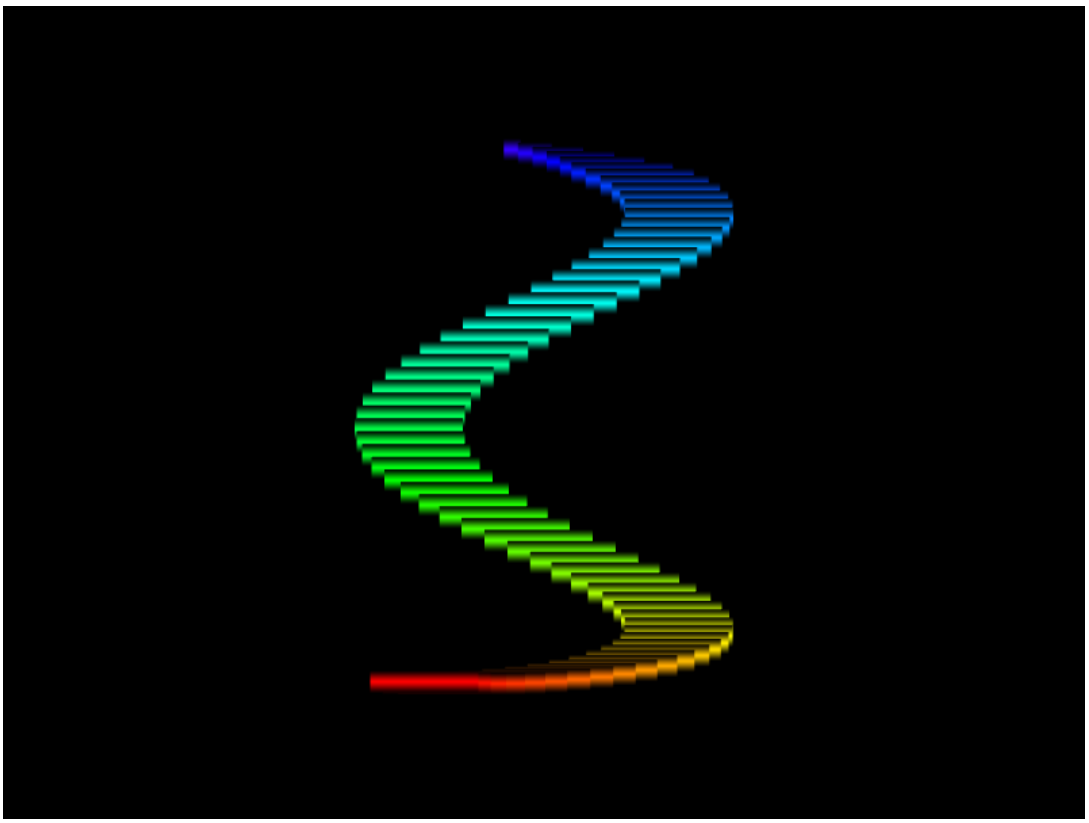


Or the position of a few tinted bars:



Amiiiiga

And modifying that tween ever so slightly creates even more interesting patterns:



Magic Carpet

I'm always happy when you can get a lot of mileage from one single feature and I honestly believe Phaser 3 Tweens now give you that. However, it hasn't been just tweens. I also implemented the **Gamepad API** which you can now find in Beta 1, along with examples on how to use it in the Input section.

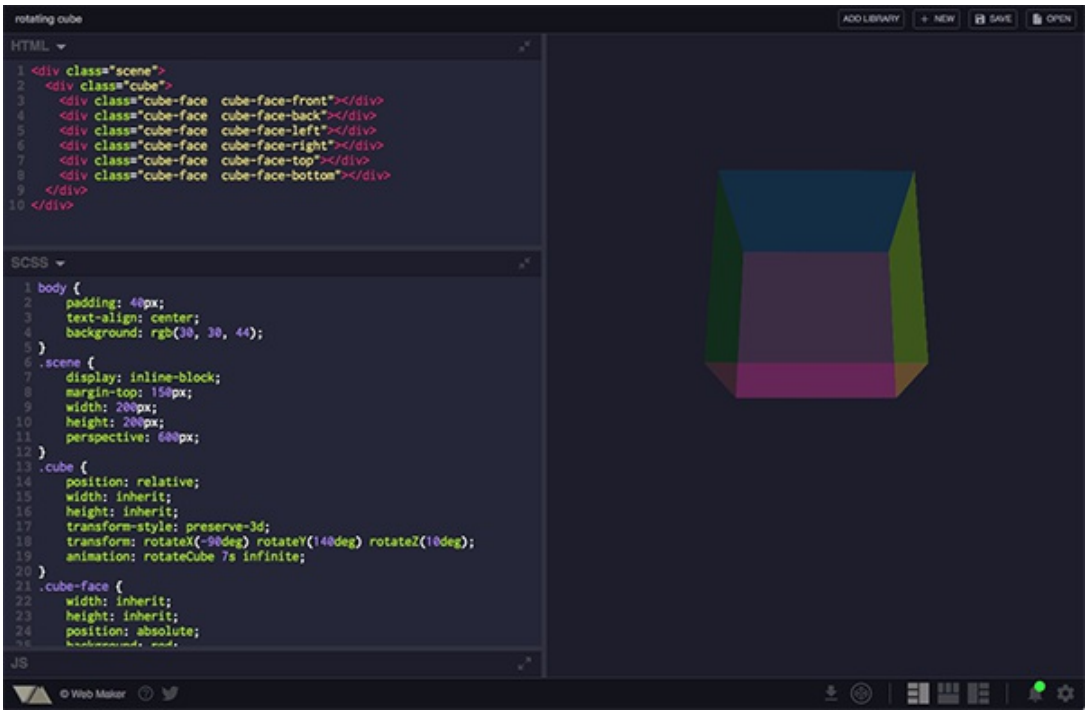
There are sample gamepad configurations include and you can easily add your own. It's also a lot more tolerant of the inconsistencies between browsers than v2 was. You can apply thresholds to analog buttons and axis controls and handle pads connecting and disconnecting mid-flow. It's another thing on our schedule we can tick off, which is a good feeling.

Felipe also managed to get the **Tilemap canvas renderer** implemented, finishing off all of the rendering sections of our Tilemap classes. I will move on to handling the data structures around tilemaps, so you can import Tiled and CSV data easily, but we're very close to done on this area too.

I still need to pick a direction to head in for the documentation and do more R&D with JSDocs. I underestimated how long it would take to do this and I'm still not 100% happy with the solutions I've found so far. It's essential we get starting documenting the API though and I want at least 15% of it finished in time for Beta 2 in a few weeks. However, it's a great feeling to be back on track with nothing else to worry about other than getting Phaser 3 done.

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 or post to the Phaser 3 Forum - we'd love to hear from you!



Web Maker is a "blazing fast & offline web playground in your browser". Kind of like Codepen, but running locally.

JS Party is a weekly podcast covering a range of different JS related subjects. The last episode (19) was all about the Web Audio API. A fun listen.

The WebUSB API is coming and already in the latest Chrome.

Further Reading ...

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

Phaser Releases

Phaser CE 2.8.6 released September 11th 2017.
Phaser 3 Beta 1 released September 11th 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.