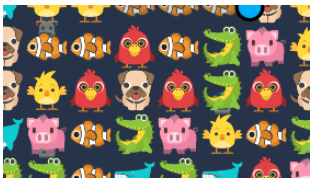
Welcome to Issue 89 of Phaser World

While the Internet simultaneously implodes from the combined weight of female Doctor Who manbaby rage and a new series of Game of Thrones I'm just

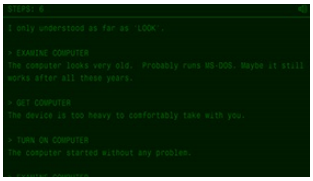
thankful that the awesome Phaser community just keeps on going! Cranking out game after game and tutorial after tutorial. We've some great titles this issue and another huge v3 update report. But, I must hit 'send' on this newsletter before daring to go back to my Massive Match browser tab, or it may never get released.

So, 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.

The Latest Games



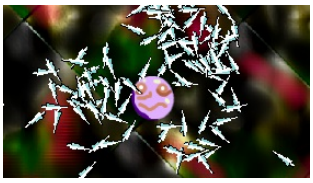
Massive Match >>> **Game of the Week**
A Massively Multiplayer Match 3 game that is extremely good fun!



Radiation >>> **Staff Pick**
A great text adventure set inside a nuclear reactor. You open the chest. Inside is a lamp. Get lamp. Taken.

Word Bath

Learn to spell with this fun game staring Monty the Dog.



Cavern of Spines

A two player game in which you must collect more spines than the other player and then hit them!

MTS Dunker

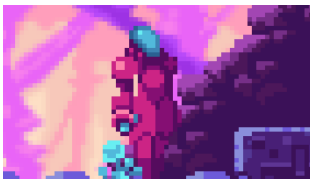
How many balls can you slam into the net in the time given?

What's New?

Phaser CE v2.8.2 Released

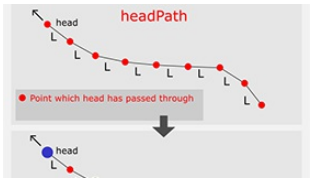
The Phaser community have done it again, with another fine update to Phaser

CE.



Warped Caves Pack

A beautiful pixel art pack and Phaser game source code.



Slither.io Tutorial Part 2

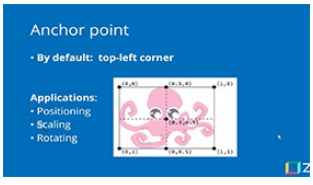
Part 2 of the tutorial series focuses on adding the snake into the game.

Slither.io Tutorial Part 1

Learn how to create the hit game Slither with Phaser in part 1 of this new series.

Audio Delay Tutorial

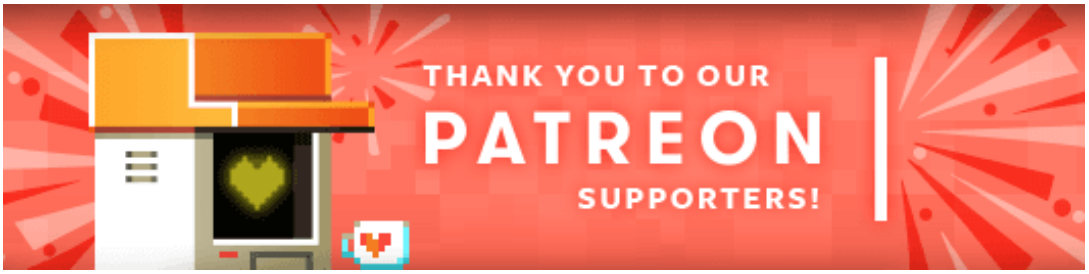
A tutorial on dealing with handling audio decoding in your games.



Live Teachers Training

A recording of a Phaser training session given at Clayfield College, Brisbane.

Phaser Hackathon is Live!
Join Zenva's #GameDev Compo – Make a Game and Win Prizes.



Welcome and a massive thank you to the new Phaser Patreons who joined us this week: **Doug Park** and **Jeremy**. Also, huge thanks to **Eric Wright** for his 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.

Dev Log #89

Before we dive into v3 this issue I just want to give a massive shout out to the community who pulled together to publish yet another awesome Phaser CE release. v2.8.2 is now up on npm, GitHub and all the usual channels and features

a great range of new features, updates and bug fixes. CE releases are coming thick and fast this year, with one every month or so.

From States to Scenes

Since starting Phaser 3 development in earnest I have been carefully evaluating *everything* that is going into it. I'm not just porting over v2 classes for the sake of it. In fact, the vast majority of the code in v3 is brand new, written entirely from scratch. However, the changes don't end at just the API - this is also the time to carefully reflect on internal choices as well, including the naming of things.

One such thing is the State Manager. Phaser has always used the term 'state' since day 1 because it was inherited from the Flixel project before it. Yet as a term it has confused a number of developers over the years. So I did a little research to see what terminology other frameworks used and the results were quite surprising. The most common term was 'Scene', used to represent a collection of Game Objects (or nodes in some frameworks). Lots of the more visual game engines use the term 'Level' instead, and others like Game Maker use the term 'Room', but none used State.

As a result, I have changed the use of the term 'State' within v3 to 'Scene'. There is now a SceneManager, all Game Objects have a property called 'scene' which indicates the scene responsible for them, and internally 'scene' is now used everywhere as well. It actually changes nothing with regard to features, but it does require a change in mindset. I've been typing in 'state' for so many years now that I'm still getting used to 'scene' instead :) But it's a more logical name and now was the time to change it. Game Objects will remain being called that however (just like in Unity) as I find the term 'node' too generic.

All About the Input

Last week was all about working on the Input Manager. We had a lot of work to do both in terms of figuring out how best to expose the new API and also how to keep it fast. In Phaser 2 you 'input enable' a Game Object and once that has happened you can access various properties, methods and signals that belong to its Input Handler. The downside is that every object has to carry around this heavy handler component with it, which is something I wanted to avoid in v3. Yet there is no denying the flexibility and ease of use it had. As with most things we've found there was a fine line to walk somewhere down the middle.

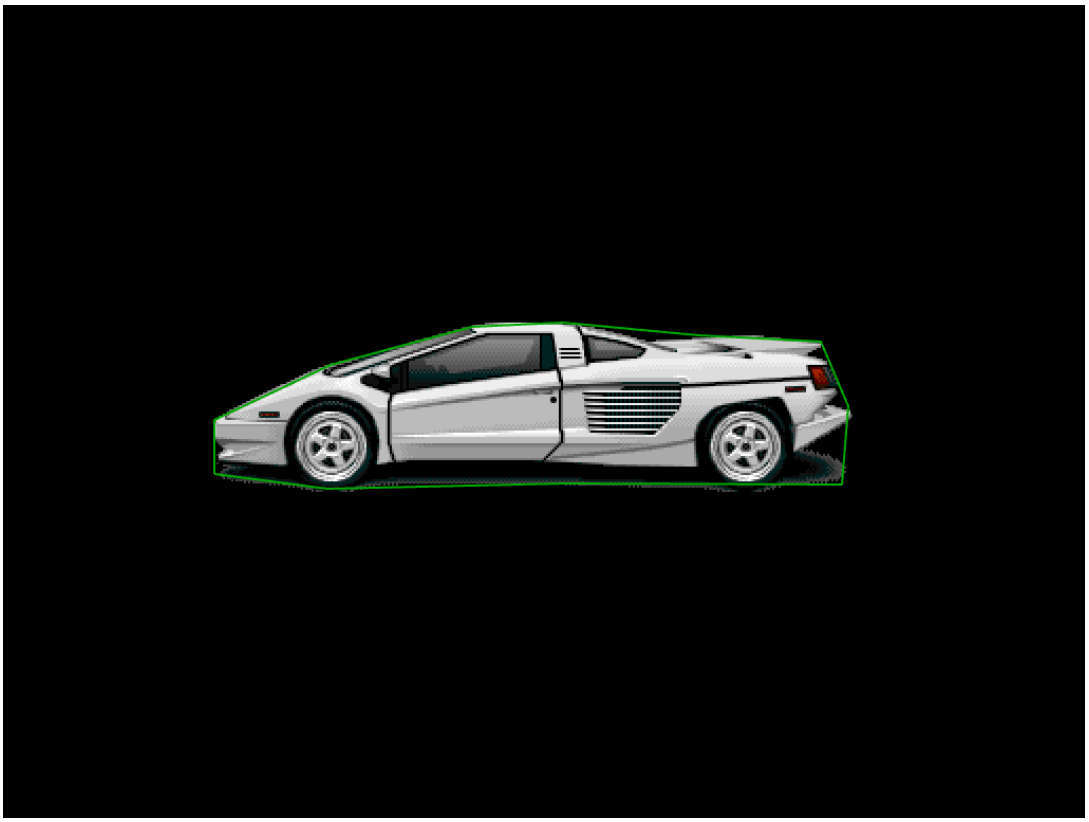
All Game Objects now have 'hitArea' and 'hitAreaCallback' properties. By default

these are null. You can either call 'setHitArea' directly on the Game Object, which will return a reference to the GO itself (allowing for method chaining), or you can call 'setHitArea' on the Input Manager which offers more flexibility - for example you can pass in an array of GOs to enable, or a Group of them.

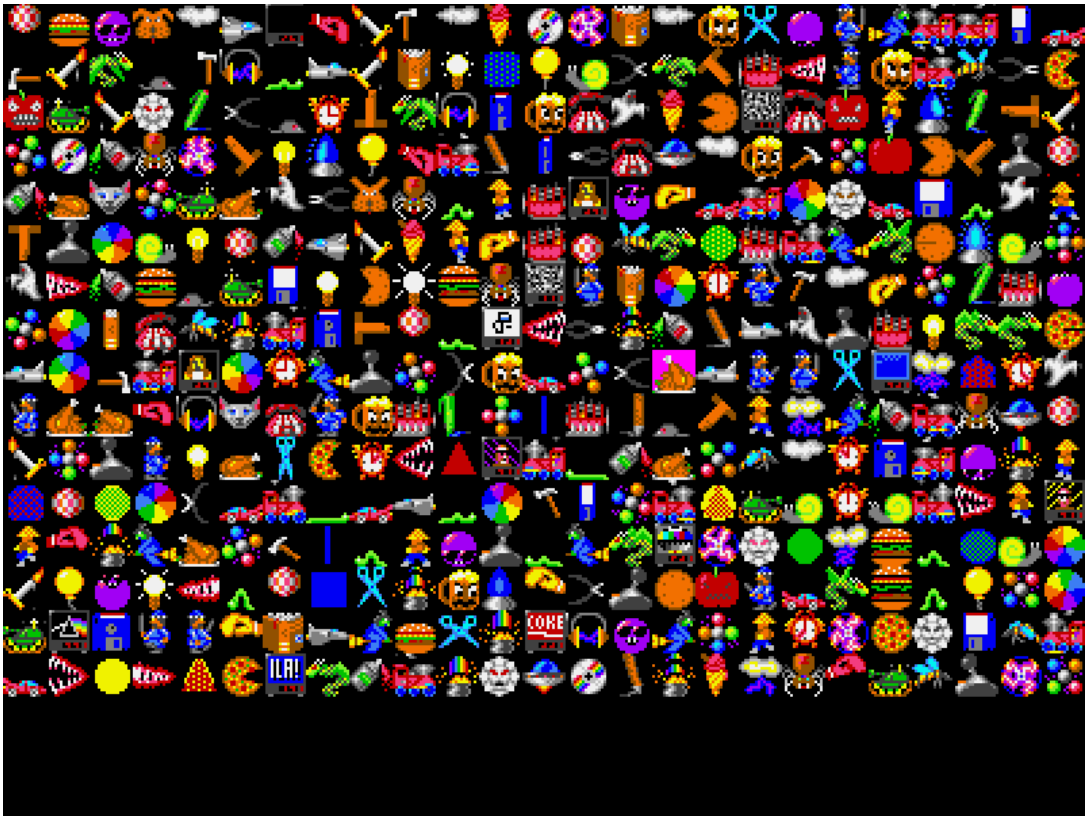
Currently, the setHitArea method takes two arguments: a shape object and a callback to invoke if the pointer goes over the shape. The shape can be any of the Phaser geometry objects. In the example below are 5 different sprites each with a unique geometry hit area:

Geometry based hit areas

There is a circle, rectangle, ellipse, triangle and polygon. All acting as hit areas for their respective sprites. Mouse over them and they should change tint. Here is an example showing the outline of the polygon rendered to a Graphics object so you can see the shape itself:



There are a couple of important points about using Shapes. First of all, more than one Game Object can share the same hit area shape. In the following example, there are 400 sprites aligned in a grid, but every one of them shares the *same* Rectangle instance. They don't each create their own rectangle, which helps cut down on memory use in your app (the less unique objects created, the better):



You can see the same Rectangle being defined in the code below:

```
var hitArea = new Phaser.Geom.Rectangle(-16, -16, 32, 32);
var hitAreaCallback = Phaser.Geom.Rectangle.Contains;

//  Create 400 sprites aligned in a grid
group = this.make.group({
    classType: Phaser.GameObjects.Image,
    key: 'bobs',
    frame: Phaser.Utils.Array.NumberArray(0, 399),
    randomFrame: true,
    hitArea: hitArea,
    hitAreaCallback: hitAreaCallback,
    gridAlign: {
        width: 25,
        height: 25,
        cellWidth: 32,
        cellHeight: 32
    }
});
```

Secondly, the hitAreaCallback can be any function you like. In the example above I'm passing in the native 'Contains' function that is part of the Rectangle class. This takes in a shape (the rectangle) and an x and y coordinate and returns a boolean. The Input Manager uses the result to determine if it was a hit

or not. You can use any callback you like though. As long as it takes the 3 arguments required (the hitArea, x and y) and returns a boolean it can technically do whatever you want internally. Maybe it doesn't use the shape at all and instead checks the color of the pixel the pointer is over? Maybe you need to define extra logic to control if a Game Object should be considered for input. Whatever the use-case you have the flexibility to control it yourself now.

Constant Polling

One of the new features of the v3 Input Manager is being able to set a poll rate. Remember that every Scene has its own Input Manager, so you can define this on a scene-by-scene basis. The poll rate defines a time period after which Game Objects should be checked for input.

You can use **setPollOnMove** to only refresh the Input Manager when the Pointer moves position. This is the most 'low intensity' setting, using the least amount of resources. Try this demo and notice that if you put the mouse pointer into the path of the moving Sprite and let go, it will not trigger the over and out events as the Sprite moves across the screen. This is because you're not moving the pointer so it doesn't know to update itself.

In contrast, you can also use **setPollAlways**. This will poll the input state of all Game Objects each frame. Look at this demo and again place your pointer into the path of the Sprite and don't move it. Unlike before, this time the Sprite will trigger its over and out events because it is being constantly checked. You can also set a time value for the poll rate in milliseconds, so it will check input both on movement and also after the given period of time.

Depending on your type of game depends on which option is best for you. For example, in a card game, the cards generally don't need to worry about if they are being interacted with until it actually happens and the player moves the pointer and clicks a card. So for this, you can use the lower intensity 'poll on move' system. For a faster physics based game (think Angry Birds) perhaps the other option is more suitable. At least now you have the choice.

Mass Updates

Internally Phaser 3 maintains its own list of 'input enabled' objects. As the API is developed over the course of this week more features will come in place. For example, at the moment if you scroll or scale the web page with an example on, the pointer coordinates get messed up. This is because the API isn't yet taking the canvas position or transform into account. All of these things will come, along

with the ability to drag sprites, pixel perfect detection and so on.

As it stands now though we put together a stress test, using the built-in camera culling and new shared hit area system and we managed to get a huge number of objects interacting with the pointer. In the example below there are 10,000 fully interactive Game Objects. Use the cursor keys to scroll the camera and the mouse to select an item:

There is also a different version of the stress test demo above that disables the camera culling. In some cases, this is more performant than having it enabled. The choice is entirely yours to make in your game, as only you will know what's best for your particular needs.

Having this sort of input selection power is actually new to Phaser and I hope will open up the opportunity for all kinds of games previously a bit out of its reach (without a lot of custom work). If I can find the time I'd love to put together a Command and Conquer style units demo together as I think it would be a great example to have. First, though it's time to get back to complete the API.

The Input Manager is the last 'big' manager required for us to hit Alpha release later this month. We're still on target but there is no denying that the volume of work to undertake in the coming weeks is massive. If you are one of our supporters then again I thank you *so much*. We're burning through our savings

right now getting this project completed, so it feels a bit like a race against time! Yet with every new feature and update we land it just gets more and more powerful. Damn exciting :)

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. The group is for anyone who wishes to discuss what the Phaser 3 API will contain.

Apparently, there's a new Atari console on the horizon. Called the Atari Box it's reported to be able to play both old and brand new titles. Will be interesting to see what that translates to.

Francois DIY is kind of like Mario Maker but written in JavaScript and running in the browser.

Muzzle is an app that silences pop-up notifications while you're screen sharing or broadcasting. Also, 10/10 for the hilarious landing page :)

Finally, can I just say well done to the BBC to finally picking a female lead for Doctor Who. The man babies across the internet crying about it is hilarious (and depressing) but I cannot wait to see the new series.

Further Reading ...

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

Phaser Releases

The current version of Phaser CE is 2.8.2 released on July 14th 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.