Welcome to Issue 106 of Phaser World

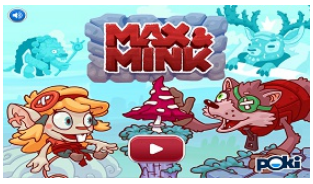
Another week, another newsletter! This is quite a massive issue with a lot of animated gifs, so apologies for that, but it's the best way to show off what just

landed in Phaser 3. Don't forget you can always read the issues on our website, along with all back issues, should this not format too well in your mail client.

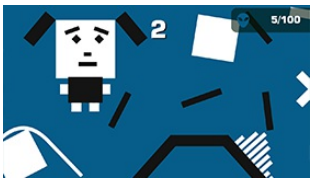
So, until the next issue, keep on coding. Drop me a line if you've got any news you'd like featured. You can reply to this email or grab me on the Phaser Slack channel.

Every year Zenva do some great Black Friday deals and this year is no exception. They've just released a new bundle pack called Beyond Hello World containing a staggering 27 complete courses! You'll learn everything from Phaser, Unity, Python, VR, game dev and web dev to WordPress plugins and loads more. A huge collection of resources for $25.

The Latest Games



Game of the Week
Max and Mink
Guide two eccentric animals through a creative, fun, and colorful world.



Staff Pick

Wire

Have you got what it takes to guide your wire through this frustrating yet highly addictive maze?

Feed the Bob

Eat a balanced diet to stay alive in this simple and fun game.

Soccer for Dummies

Crack up with Soccer for Dummies, the soccer physics game that makes no sense and can be controlled with just one finger.

BBC Gold Rush Game Maker

The intrepid 19th century gold seekers braved icy mountains, bears, wolves and worse. Create your own gold rush adventures and share them with the world.

What's New?

Phaser i18next

A plugin allowing you to use i18next in your games for easy translation management.

3W Academy

A 3 month long French Games Developer course. 400 hours of lessons and access to a job that recruits at the end.

English Duniya

Every child can now learn English on a learning pathway made just for them in this hybrid app.

Cocoon Tutorial

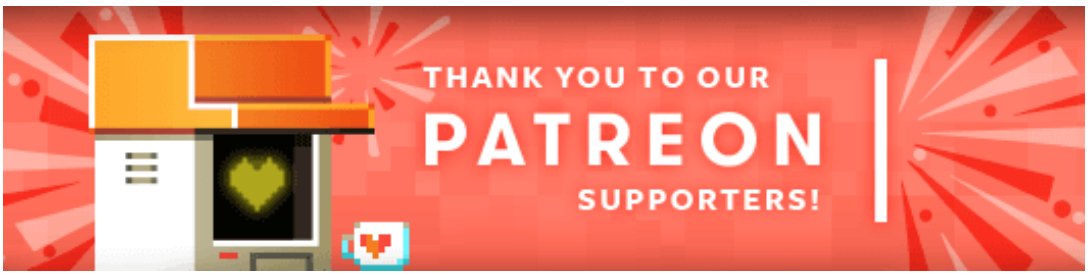
A step by step guide on packaging your game up for mobile using Cocoon.

Making Missiles in Phaser

A lesson from phasergames.com on how missiles can be made in shot using Phaser.

PlayFab SDK now supports Phaser

Get access to leaderboards, analytics, a/b testing and more from your games with native Phaser support direct from PlayFab.



Welcome and a massive thank you to the new Phaser Patron who joined us this week: **Nick Roach** and also to **Alberto Ini** for his kind 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 #106

Three exciting new additions landed in V3 last week: Loads of progress on the Tilemap API, sound now playing from the all-new Sound Manager and superb new masking support. You can try all of them in Phaser 3 Beta 11. Grab it from GitHub pre-built, or from npm using the beta tag.

Tilemap API Updates

Michael is working on the updates to the Tilemap API. I asked him to write a short summary of what he's been doing over the past week:

Most of the v3 Tilemap API is in place now. The last remaining large piece is the physics / collision integration. The API mirrors v2 with a few upgrades, so here's a quick overview. (Or you can just skip to the examples and screenshots!)

The new Tilemap class is your first stop when working with maps. You'll get an instance when you use **this.make.tilemap** or **this.add.tilemap**. This isn't a display object, rather, it holds data about the map and allows you to add tilesets and tilemap layers to it.

A map can have one or more layers, which are the display objects that actually render tiles. They come in two flavors: StaticTilemapLayer and DynamicTilemapLayer. A StaticTilemapLayer is super fast, but the tiles in that layer can't be modified. A DynamicTilemapLayer trades some speed for the flexibility and power of manipulating individual tiles. Below is a code snippet showing how to add one of each to a map:

```
var map = this.make.tilemap({ key: "map" });
var tileset = map.addTilesetImage("tiles");

// Parameters: name/index of layer, Tileset instance, x, y
var staticLayer = map.createStaticLayer('Tile Layer 1', tileset, 0, 0);
var dynamicLayer = map.createDynamicLayer('Tile Layer 2', tileset, 0, 0);
```

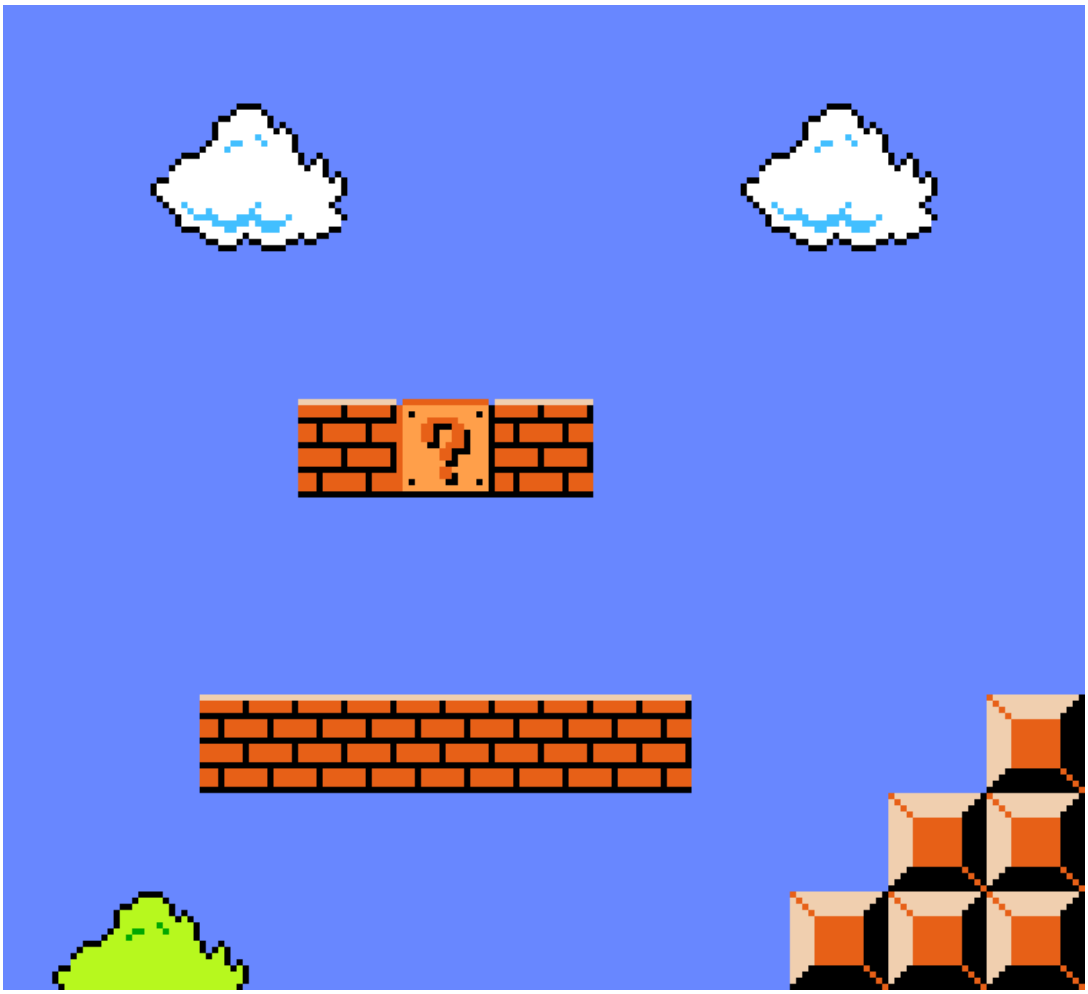
As mentioned previously, you can load up map data from a CSV file or a Tiled JSON file (click the code to load the example)

```
function preload ()
{
    this.load.image('tiles', 'assets/tilemaps/tiles/catastrophi_tiles_16.png');
    this.load.tilemapCSV('map', 'assets/tilemaps/csv/catastrophi_level2.csv');
}
```

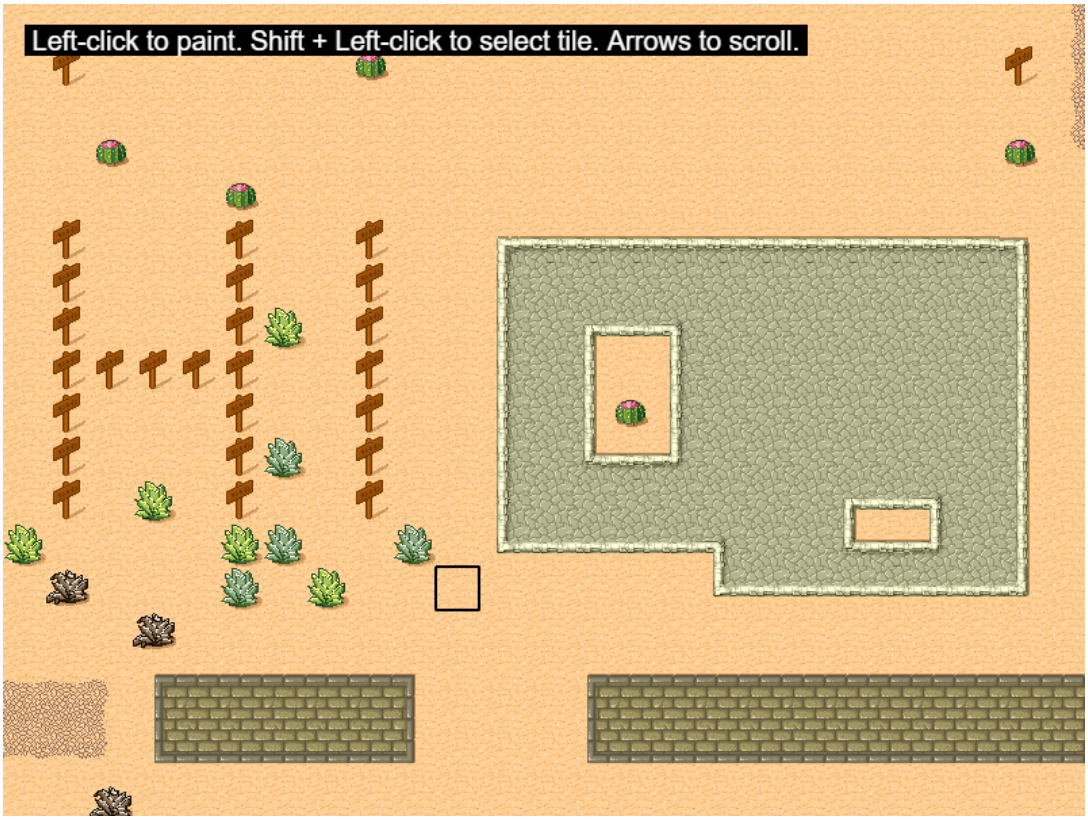
```
function preload ()
{
    this.load.image('tiles', 'assets/tilemaps/tiles/dangerous-kiss-x2.png');
    this.load.tilemapJSON('map', 'assets/tilemaps/maps/dangerous-kiss.json');
}
```

Sometimes you just want to load in something directly from an array, so now that's part of the API:

```
// Load a map from a 2D array of tile indices
var level = [
  [  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0 ],
  [  0,  1,  2,  3,  0,  0,  0,  1,  2,  3,  0,  0 ],
  [  0,  5,  6,  7,  0,  0,  0,  5,  6,  7,  0,  0 ],
  [  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0 ],
  [  0,  0,  0, 14, 13, 14,  0,  0,  0,  0,  0,  0 ],
  [  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0 ],
  [  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0 ],
  [  0,  0, 14, 14, 14, 14, 14,  0,  0,  0, 15,  0 ],
  [  0,  0,  0,  0,  0,  0,  0,  0,  0, 15, 15,  0 ],
  [ 35, 36, 37,  0,  0,  0,  0,  0, 15, 15, 15,  0 ],
  [ 39, 39, 39, 39, 39, 39, 39, 39, 39, 39, 39, 39 ]
];

// When loading from an array, make sure to specify the tileWidth and tileHeight
var map = this.make.tilemap({ data: level, tileWidth: 16, tileHeight: 16 });
var tiles = map.addTilesetImage('mario-tiles');
var layer = map.createStaticLayer(0, tiles, 0, 0);
```



Last but not least, a dynamic tilemap example that lets you paint tiles. In v3, the StaticTilemapLayer, DynamicTilemapLayer and Tilemap share much of the same API. You can use getTileAt, hasTileAt, forEachTile, etc. on any of them (of course, you'll won't be able to use a mutating method like removeTileAt, fill, shuffle, etc. on a StaticTilemapLayer.)




```
var pointerTileX = map.worldToTileX(this.input.x);
var pointerTileY = map.worldToTileY(this.input.y);

marker.x = pointerTileX * map.tileWidth;
marker.y = pointerTileY * map.tileHeight;

if (this.input.manager.activePointer.isDown)
{
    if (shiftKey.isDown)
    {
        selectedTile = map.getTileAt(pointerTileX, pointerTileY);
    }
    else
    {
        map.putTileAt(selectedTile, pointerTileX, pointerTileY);
    }
}
```

Sound Manager Updates

Pavle is in charge of coding the new Sound Manager for V3. He too has written an update on what was done last week:

This week for the first time you will be able to play sounds in v3 which is pretty exciting. What's more exciting is all the functionality that will be included in the v3 Sound API like dynamic effects, audio spatialization (3D audio) and audio sprite support just to name a few.

For now, you are able to load audio files pretty much the same way you did in v2

and perform basic operations like play, pause, resume and stop on individual sounds.

To match the rest of the Phaser v3 API you can now pass a config object to almost any method call when creating and playing individual sounds to set its properties to desired values but you will also be able to change them directly on the sound object like in v2.

You can set mute, volume, rate and detune settings to each individual sound as well as setting it globally directly on the sound manager to affect all sounds at once.

Mute and volume settings are pretty standard but rate and detune settings are new to v3.

Rate setting defines the speed at which the sound will be played. Value of 1.0 plays the sound at full speed, 0.5 plays the sound at half speed and 2.0 doubles the sounds playback speed. It could be used to
make some cool effects like slowing the playback down to enhance the slow-motion effect or speeding-up playback gradually to match increasing gameplay difficulty.

Detune is a setting representing detuning of playback in cents. This value is combined with the playback rate to determine the speed at which the sound is played to give you even more control over the playback.

Next on the list will be adding support for looping and seeking sounds. These options can come in pretty handy if, for instance, you decide to make your own music player using Phaser but there will most certainly be an example added in the future to demonstrate how to actually do that.

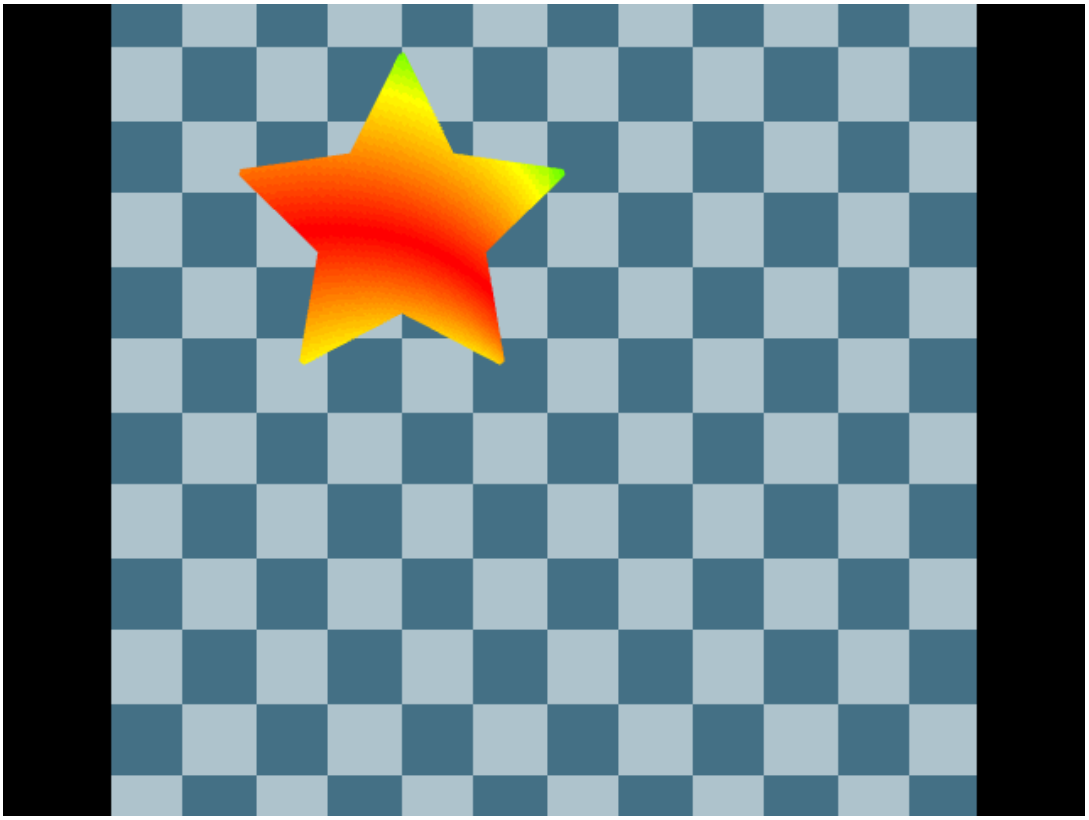
Feel free to check existing example below and play around and tweak all the values and come up with some cool effects even with only this much of the functionality exposed to you:

```
this.game.sound.add('boden', {
    mute: false,
    volume: 0.5,
    rate: 2,
    detune: -100
}).play();

var boden = this.game.sound.add('boden');
boden.mute = true;
boden.volume = 1;
boden.play({
    rate: 1.5,
    detune: 0
});

this.game.sound.mute = false;
this.game.sound.volume = 0.8;
this.game.sound.rate = 0.5;
this.game.sound.detune = 200;
```

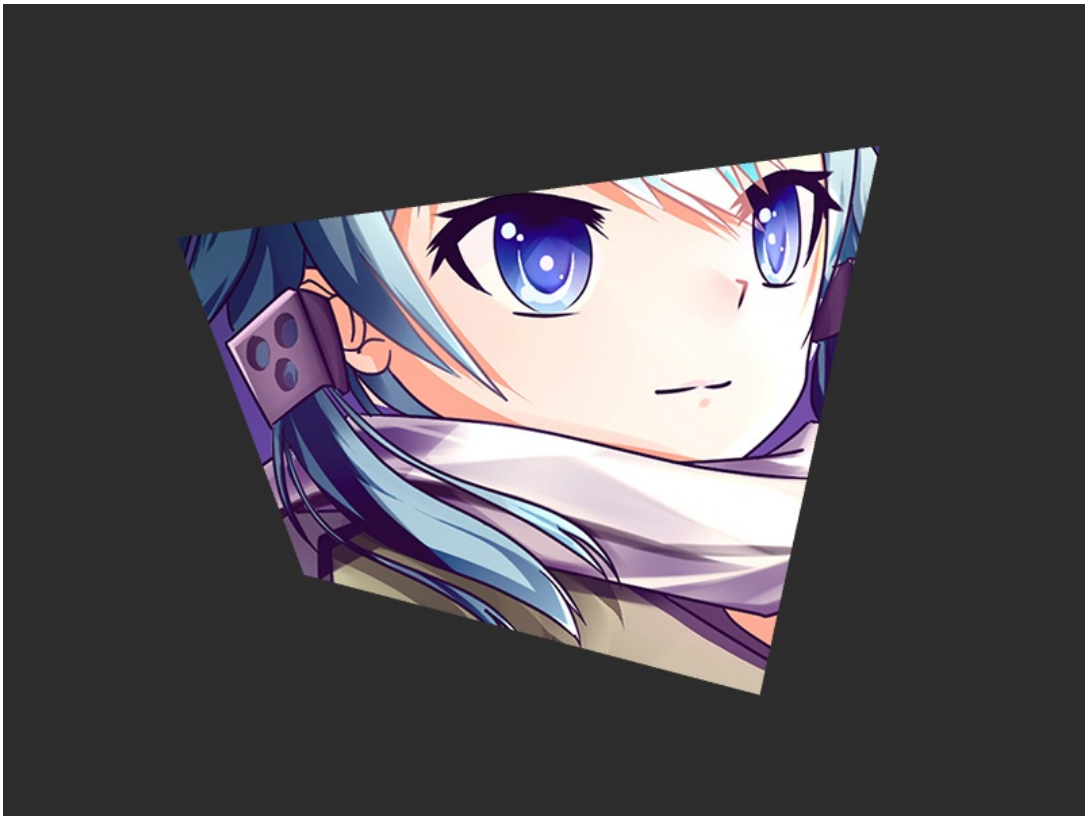
Bitmap and Geometry Masks

The big new feature to land in Beta 11 is full support for applying masks to Game Objects. Masks in v2 were somewhat limited. You could only use geometry for your masks because of the way in which it tried to maintain parity between Canvas and WebGL. In v3 that restriction has been removed and we're offering the ability to use Geometry based masks, which work on both Canvas and WebGL, and Bitmap based masks which are WebGL only.

What does a mask allow you to do? In short, it lets you define which parts of a Game Object can be seen. In the following example move the star around with the mouse. The star is a geometry object that is a mask for a big picture of a colorful swirl pattern. As you move the star masks you reveal part of the swirl image below it:



You're not just limited to static geometry. Here is an example of using a Quad as a mask for an image, with the quad being updated in real-time:

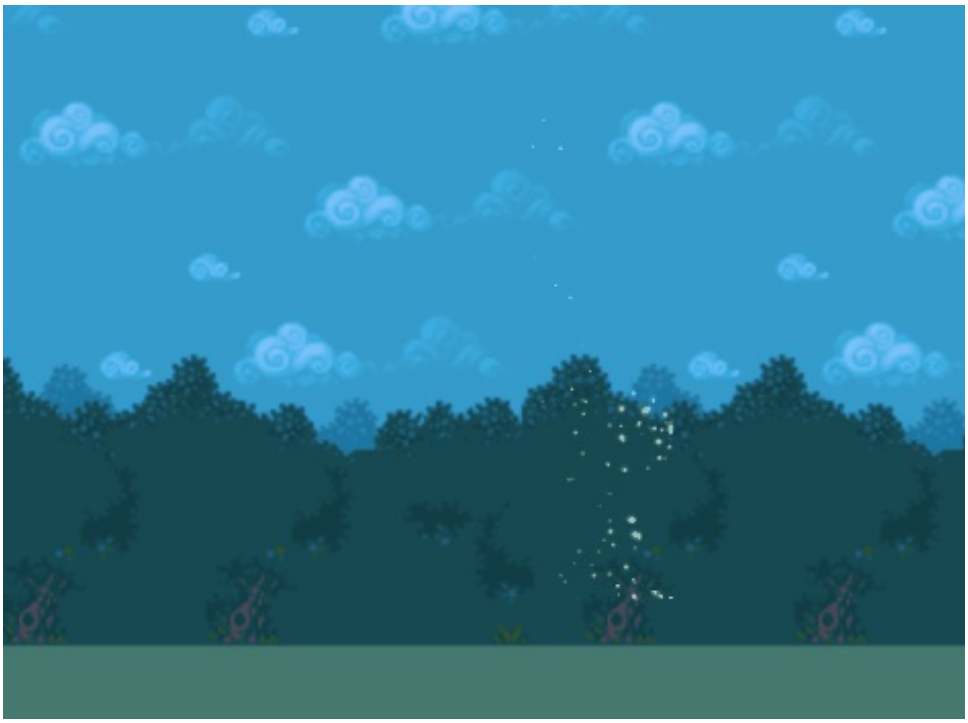


Here's a quick magnify glass demo, where the mask is the circular part of the

magnifying glass, which masks out the picture below it. An easy but effective illusion:

You're not restricted to geometry though. You can also use any bitmap as a mask, and the bitmap can be a frame from a texture atlas too. This allows you to create some incredible effects. In the example below we're using a texture as the mask for the large bunny sprite:

Masks can be used for all kinds of things. From progress bars and health meters to hiding parts of sprites as they vanish behind an object. You can even mask particles:



Or use the particles as the mask itself!

Mask an Effect Layer or Render Pass

As well as using masks on standard display objects you can also use them on Effect Layers and Render Pass objects too. Here's an Effect Layer in action, nestled between a bunch of Sprites, and with a mask applied to it (click to change the shader being used in the effect layer):

As you can see, they're powerful things! They are also essential in modern games so I'm really pleased to see them featured so well in V3. By way of a celebration here's a little demo I put together showing masks in all their glory:

There is a large bitmap mask applied to the planet in the background, so the planet is only revealed as the mask is tweened across it. There's a mask applied to the logo in the middle. Watch it for a bit and you'll see it start to have a rainbow effect passed through it. This is actually done using a TileSprite, showing how they can be used for masks too.

Finally, there is a particle emitter mask at the bottom, revealing the image with little shards of glass (the image is from the fantastic anime series Re:Zero, in case you were wondering!). The example may be a little over the top but all it took was a few pieces of stock art and a little code and you've got yourself what could be a really nice title screen for a shoot-em-up or similar (as you will see shortly when Matter.js support lands :))

It's really great to get these heavy-weight features in. We get ever closer to being feature complete and thank you to everyone who is working so hard on V3 now, including those submitting issues and PRs, as well as the core team of course.

Phaser 3 Labs

Phaser 3 Beta 11 is out and ready for testing.

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!

Firefox Quantum is out and it's genuinely impressive! We've been testing it for V3 development and it's nice to see such a big speed increase. Well worth checking out.

Western Digital are developing 40TB hard drives that use microwave technology. Yes, really.

I love OCRemix and this vocal Tetris soundtrack is outstanding - each block has its own singer and it's just great :)

Phaser Releases

Phaser CE 2.9.2 released November 10th 2017.
Phaser 3 Beta 11 released November 20th 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.