Web Version Unsubscribe

Welcome to Issue 128 of Phaser World

This issue is a big one, with lots of new tutorials and long Dev Log. So if you're reading inside Gmail, or any other mail client that dumbly strips the content, be sure to hit that 'Read on Web' link!

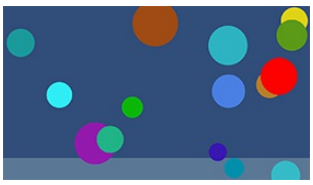
We skipped an issue as I was too entrenched in the Phaser 3.12 release but I feel like this one makes up for it :) There are some superb games to play and be inspired by and extra news to make up for being a little late.

Until the next issue keep on coding. Drop me a line if you've got any news you'd like featured by simply replying to this email, messaging me on Slack, Discord or Twitter.

The Latest Games

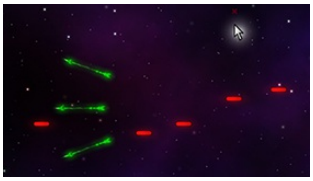
Game of the Week
Bob the Robber 5: Temple Adventure
Steal the temple treasure without being caught by the roaming guards in this addictive and beautifully presented action game.

Staff Pick
Maggot Diorama
Protect the Dwarf against nasty maggots. This game is about clicking so prepare your mouse to burn.



Spot the Spot

Click the red spot and avoid the others in this simple game that hides a comprehensive upgrade system.



MouseWars.io

Grab your cursor and do battle in this real-time multiplayer war game! Fight until only one cursor remains.

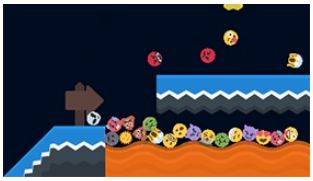
Throw Apple

A fun knife throwing game where you have to slice the apples in two.

What's New?

Phaser 3.12.0 Released

After 6 weeks work it's finally here! Sporting a brand new graphics pipeline, fully pimped-out Render Textures, Matter.js updates, SVG resizing and loads, loads more.

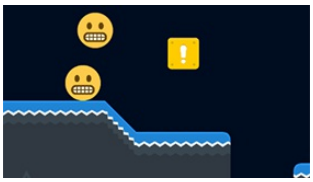


Modular Game Worlds in Phaser 3 Part 4

Learn how to add Matter.js physics into your game in part four of one of the most comprehensive, well written and illustrated tutorials on tilemaps.

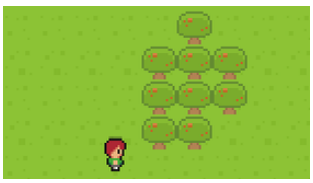
Free Phaser 101 Course

Try out 4 lessons for free from the Zenva Phaser 3 Game Development Course and learn about getting up your game and working with Sprites.



Phaser Matter Collision Plugin

A helpful plugin that wraps up the Matter collision logic in a friendly and more modular way.

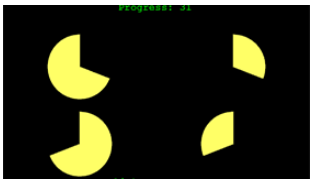


How to create a turn-based RPG in Phaser 3

Learn how to make a turn-based RPG game. including world building, player movement and bad guys.

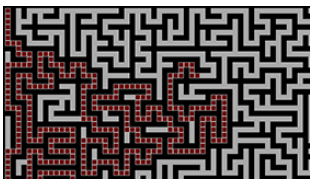
Retro Gaming Workshop

All of the classroom resources and code from the Seattle Coder Dojo Phaser game making session.



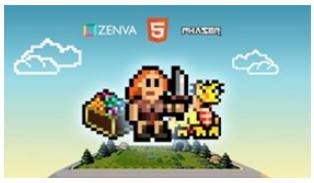
Circle Pie Meter Class

A quick and easy to use custom Game Object class for display a pie-chart progress meter.



A Star Maze Solving Tutorial

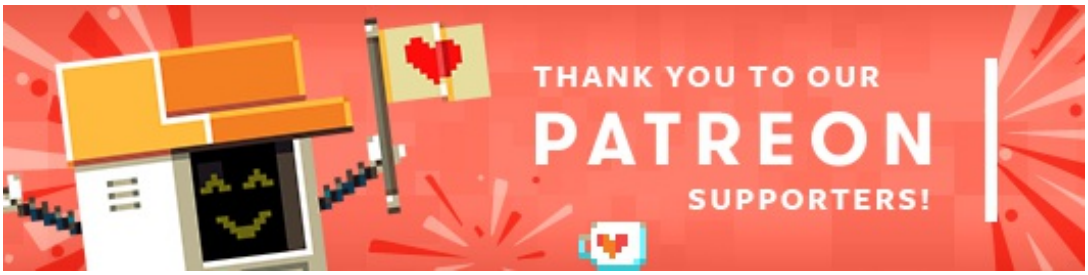
How to solve a maze using the A* algorithm with Phaser 3.



Phaser 3 Game Development Course

A complete Phaser 3 and JavaScript Game Development package. 9 courses, 119 lessons and over 15 hours of video content. Learn to code and create a huge portfolio of cross platform games.

Try this 4 course sample for free!



Thank you to these awesome Phaser Patrons who joined us recently:

Kristian Koivisto-Kokko
Wooden Plank Studios

Also, thank you to **@16patsle** and **Trevor Lock** for the donations :)

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 #128

Welcome to Dev Log 128. You may remember that last issue I was on the cusp of releasing Phaser 3.12. There were a couple of final bugs I needed to resolve before I was happy to publish it, and I spent the day after publishing the newsletter doing just that. On September 4th, Phaser 3.12 had its beta tag snipped and was kicked out into the wild.

It's a huge release. The culmination of weeks of hard work from myself and the wider community. As well as fundamentally changing the core renderer it bought

stacks of improvements along with it. Please download it now and drop it into your games. There's a comprehensive Change Log, loads of new examples and both the API Docs and TypeScripts definitions were updated for it.

With 3.12 released I wanted to cover the roadmap for the next few versions. As always, this is subject to change, but it represents my current plan as of today. I'm going to make sure that the next few versions are smaller in scope than 3.12 was, so I can release them sooner. I feel the gap between 3.11 and 3.12 was too large and it's time to bring that down again.

Phaser 3.13

This version will contain two important new features. First, the Facebook Instant Games Plugin. I've been working on this over the past couple of months, in conjunction with Facebook, and it's very nearly ready for release. It encapsulations the entire Instant Games API, allowing you to access every single feature the API offers directly from within Phaser.

This includes easy displaying of adverts, in-game purchases, leaderboards, user profiles, synced load events, session data, context switching, product catalogs and more. There will be a special build of Phaser that has this plugin fully enabled by default, along with a custom template you can use to get started with, and full documentation. Over time I will release tutorials showing how to get the best out of it as well. Instant Games is an exciting new platform to be on. I've already seen a number of Phaser games have great success on there, and hopefully this plugin will make taking your games onto IG that much easier for you.

The second feature in 3.13 is the new Shapes Game Object. I talk about this in more depth later this issue. I expect release of 3.13 to happen during the week starting September 17th.

Phaser 3.14

The 3.12 release had a bunch of features that were pushed under the Experimental flag, because they weren't properly tested or completed. 3.14 will see those finished off and documented. This includes the new DOM Game Objects and DOM Camera, which I covered in detail back in Dev Log 125, and also the new Scale Manager. This is the final system to be added to Phaser 3, after which it will be 100% feature complete, as based on the original plan from a few years ago. I expect release of 3.14 during the first weeks of October.

Phaser 3.15

The main focus of this version will be overhauling the way Containers work, and

adding in the ability to `addChild` to any Game Object. In order to do this I'll need to replace the way Transforms are managed and create a much smaller and more light-weight Container, as well as then integrating all of these changes across the Camera and Input systems. This is a significant piece of work and will fundamentally change the entire API. Yet, it can't be avoided.

It's frustrating, because I'm really pleased with the current speed of the display list and the ease with which you can z-order anything. Yet, at the same time, developers are so used to object hierarchies and nesting children, that we're going to have to trade some speed and just make it happen. It feels like a bit of a backward step but in hindsight some of the more advanced features I want to add to Phaser 3, like Spine and Spriter support, will be much easier to implement after doing it. I've no idea of the time this will take, but I'm expecting around 1 month, so a release around the end of October.

Phaser 3.16

One feature, which I've had on my to-do list for ages, and will be added this year, is for you to easily load Game Objects and entire Scene Layouts from JSON files. This would allow you to load a special data file that, when parsed by the Scene, will create all of the objects within it, setting up their properties and features as needed. For example, you could define a whole bunch of Sprites, set all of their physics and input properties, and then transform and place them within the Scene. The idea is that all of this can be handled in a single JSON data structure.

This will be really useful for 3rd party tools. You could easily create a Level Editor from this feature. And it will cut down on the amount of code needed in your Scenes too.

And beyond ...

There are literally hundreds of features I'd like to add to Phaser. Some of these can exist as plugins, others as part of the core, others as system replacements. Each day I see more and more games created with v3 and they're getting bigger in scope too. Thanks to the support from Patreon and donations I'm just about able to carry on working on Phaser full-time.

I'd very much like to overhaul the Phaser web site. It still doesn't even have a v3 sandbox, examples or API docs on there! But every time I think about what I'd like to do with it, I realize it's months of work, and doing that will divert much attention away from developing Phaser. Right now, I feel that Phaser itself should be the one getting my full attention. Perhaps if Patreon backing increases enough I could hire someone to work on the site for me.

I'd also like to take time out to write some Phaser 3 books and tutorials. As you can imagine, sales of Phaser 2 related products are really low now. This isn't surprising. I predicted it would happen over a year ago, because most people getting on-board with Phaser are using v3 now. Yet the money the v2 product sales brings in is really important, so I absolutely have to find time to address this soon. I'm thinking that once 3.15 is out I should take some time to focus on this as it will help keep things flowing in 2019.

If I could clone myself, Rick and Morty style, there would be no problems :) Right now, though, it is what it is. I'll continue to put 100% of my time into v3 development (as well as writing the newsletter and helping on Slack / Discord) and perhaps in a few months when the frantic release-schedule dust has settled I'll be able to split my time up more. For now, though, the source is where it's at.

The Shape Game Object

At the start of this Dev Log I said that 3.13 was going to include support for a new Game Object called a `Shape`, so it's worth explaining what those are and how to use them. At the time of writing, you can find Shape support in the master branch where 3.13 is being built.

You may remember in Dev Log 124 I wrote about the changes I'd made to the WebGL pipelines, merging the Flat Tint Pipeline (the one responsible for drawing Graphics) into the Texture Tint Pipeline, and causing significant speed-ups as a result. At the time I alluded to the fact that creating a Shape Game Object would now be easy. The reason I wanted to do this was because I regularly see developers new to Phaser trying to do this:

```
var rectangle = new Phaser.Geom.Rectangle(x, y, w, h);

this.add.existing(rectangle);
```

They create a Rectangle, using the geometry object, then add it to the display list, expecting to see it rendering in their game and getting highly confused when it doesn't work. I mean, it's just a rectangle, right? Surely Phaser can cope with rendering a rectangle?

This has always bemused me slightly. Both the approach and the assumption it would even do anything anyway. Yet, at its heart, it's a really sensible idea. Until the 3.12 release it was, however, highly impractical. Internally Phaser would have

had to create a new Graphics object for every shape you wanted on the display list, which would cause huge performance issues in WebGL, with constant batch flushing and shader swapping. In short, it would have been a Really Bad Idea to do.

Yet I knew when I did the work consolidating the Flat Tint Pipeline that this performance reason was now a thing of the past. There is, technically, absolutely no reason now why you can't mix and match Rectangles, Circles and any other Shape you like, on the display list, happily mingled in with your Sprites, Tilemaps and whatever else you've got. It's all the same shader, it's all the same batch, so it won't impact rendering performance at all. In fact, for some Shapes, it's actually faster than rendering a Sprite!

Armed with this knowledge clawing away at the back of my mind, and needing a more creative outlet after finishing a huge round of mentally draining bug fixing for the 3.12 release, I set aside a few days for this and got to work.

Phaser 3.13 has a new Game Object called `Shape`, which by itself isn't much use because it's a base class. However, extending that class are 11 different types of Shape (with more to come). And this is where it starts to get fun :) You add a Shape to the display list in the same way you add any other object:

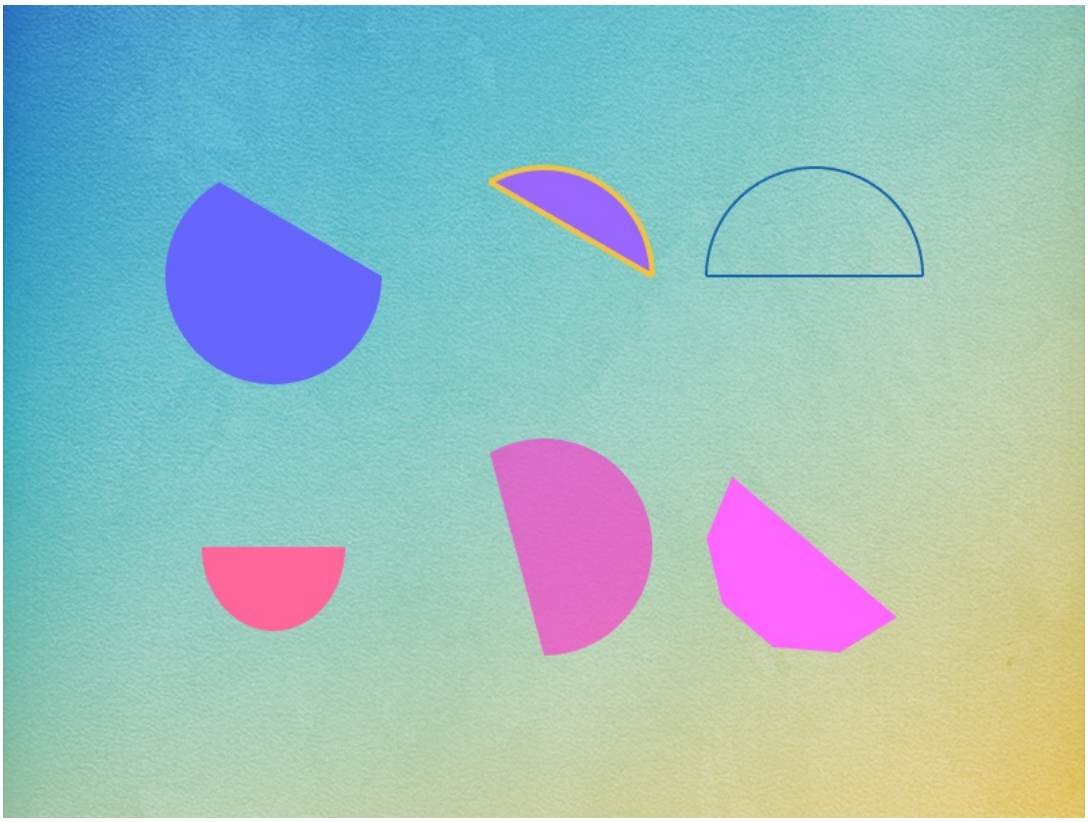
```
var shape = this.add.rectangle(400, 300, 500, 120, 0x00ff00);
```

Here we're creating a new Rectangle shape. It's positioned at 400 x 300 in the Scene and has a size of 500 x 120 pixels. The final value is the fill color.

The thing to remember is that you can treat this Shape just like you'd treat any other Game Object. You can scale it, rotate it, alpha it, blend mode it, change its origin, give it a Camera scroll factor, put it inside a Container or Group, give it input abilities or even give it a physics body. It is, to all intents and purposes, a normal Game Object. The only difference is that when rendering it uses its own special bit of display code.

So, what Shapes are available to you? This is the current list:

Arc



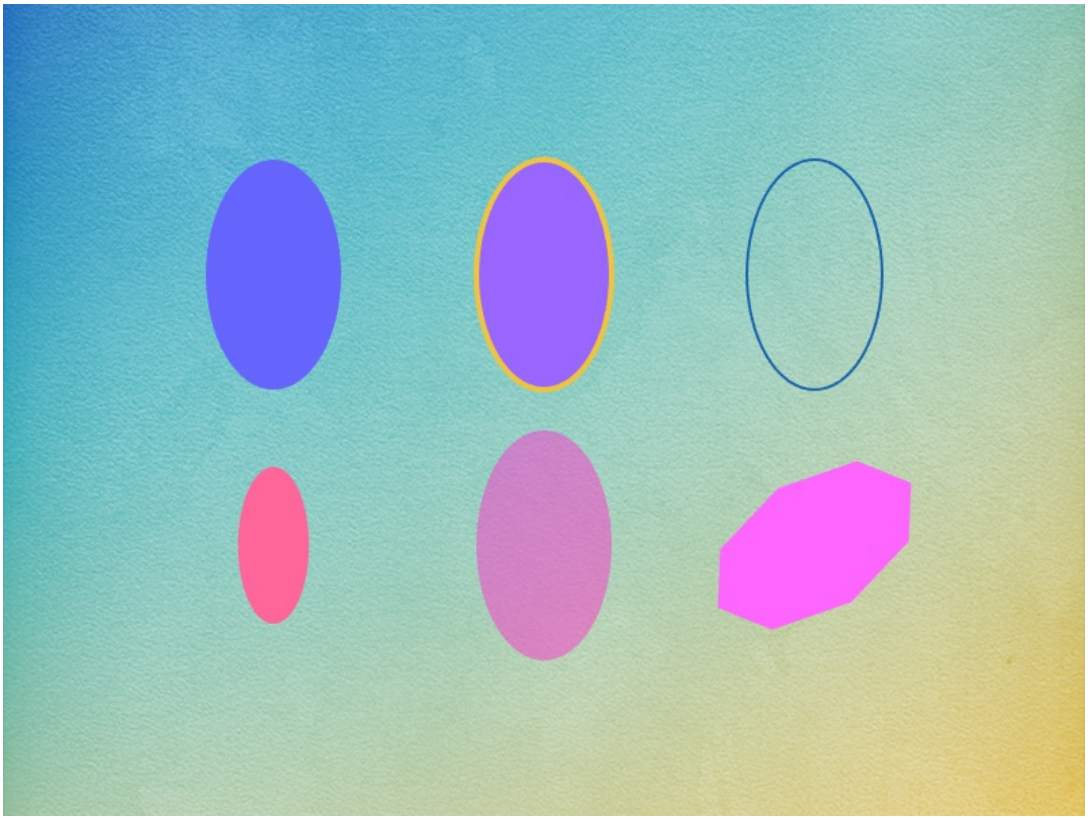
The arc allows you to draw either a circle, or part of a circle. You can set the start and end angle, if the rotation is clockwise or not, and even set the number of iterations the arc will use during rendering. In the screen shot above, you can see that the bottom-right arc has its iterations value reduced right down, creating half a decagon.

Curve

```
var startPoint = new Phaser.Math.Vector2(0, 300);
var controlPoint1 = new Phaser.Math.Vector2(100, 100);
var controlPoint2 = new Phaser.Math.Vector2(200, 100);
var endPoint = new Phaser.Math.Vector2(300, 300);

var curve = new Phaser.Curves.CubicBezier(startPoint, controlPoint1, controlPoint2, endPoint);

this.add.curve(400, 300, curve, 0x00aa00);
```

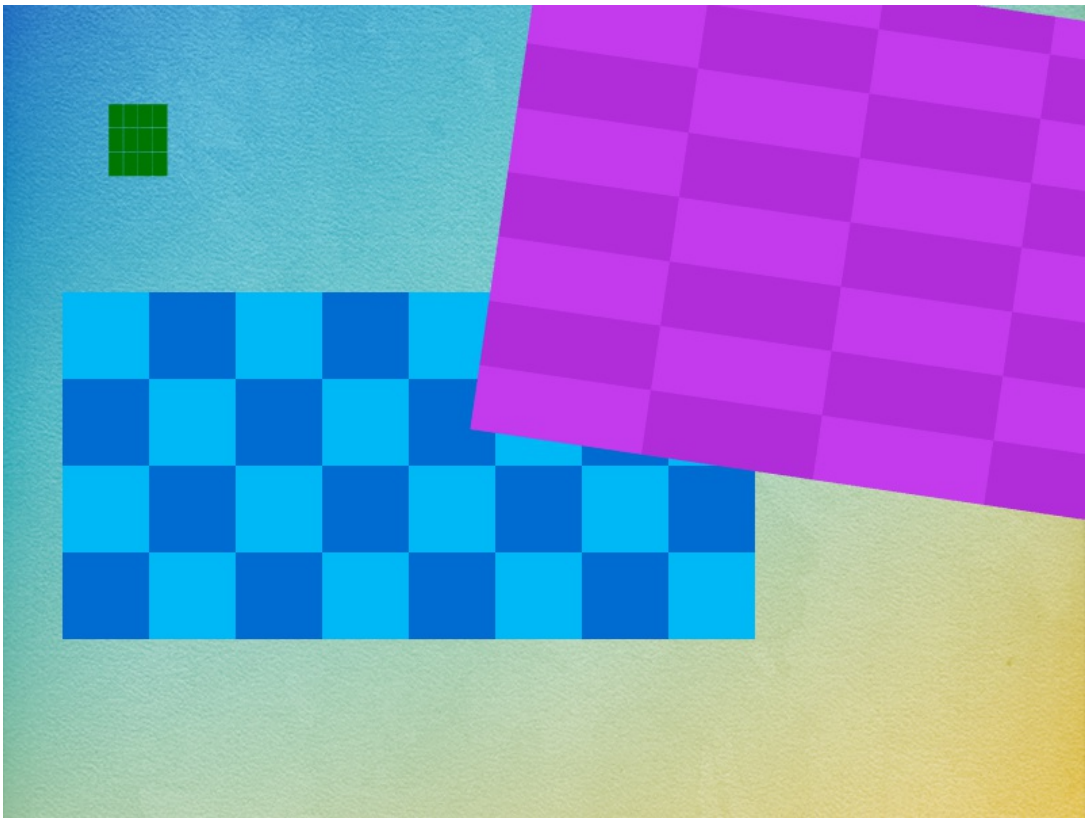
Phaser 3 has had support for different types of curve since release. You can easily create bezier curves, splines and other types. Using the Curve Shape you can now display them in-game, without needing a Graphics object. It's actually faster than using a Graphics object, too, because when you create a Curve Shape it will pre-calculate all of the polygon data needed to draw it just once. It then uses this cached data during rendering. Where-as for a Graphics object, because its command list is stateless, it generates this polygon data for all of its objects every frame.

Ellipse



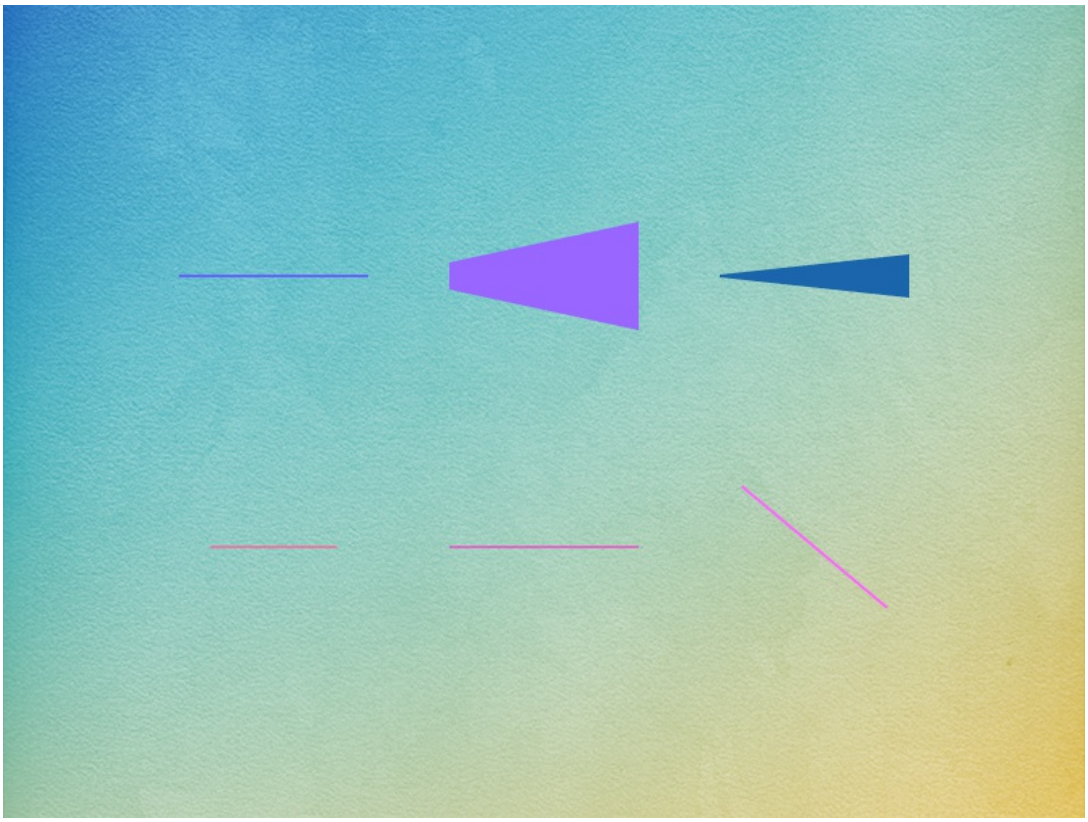
You can easily create an ellipse shape, which is essentially a circle with a differing width and height. It can be filled or stroked (or both!) and as with the arc you can set the 'smoothness' of it, allowing you to decrease the number of points used when creating its polygon data. You can see this in the bottom-right ellipse in the image above.

Grid



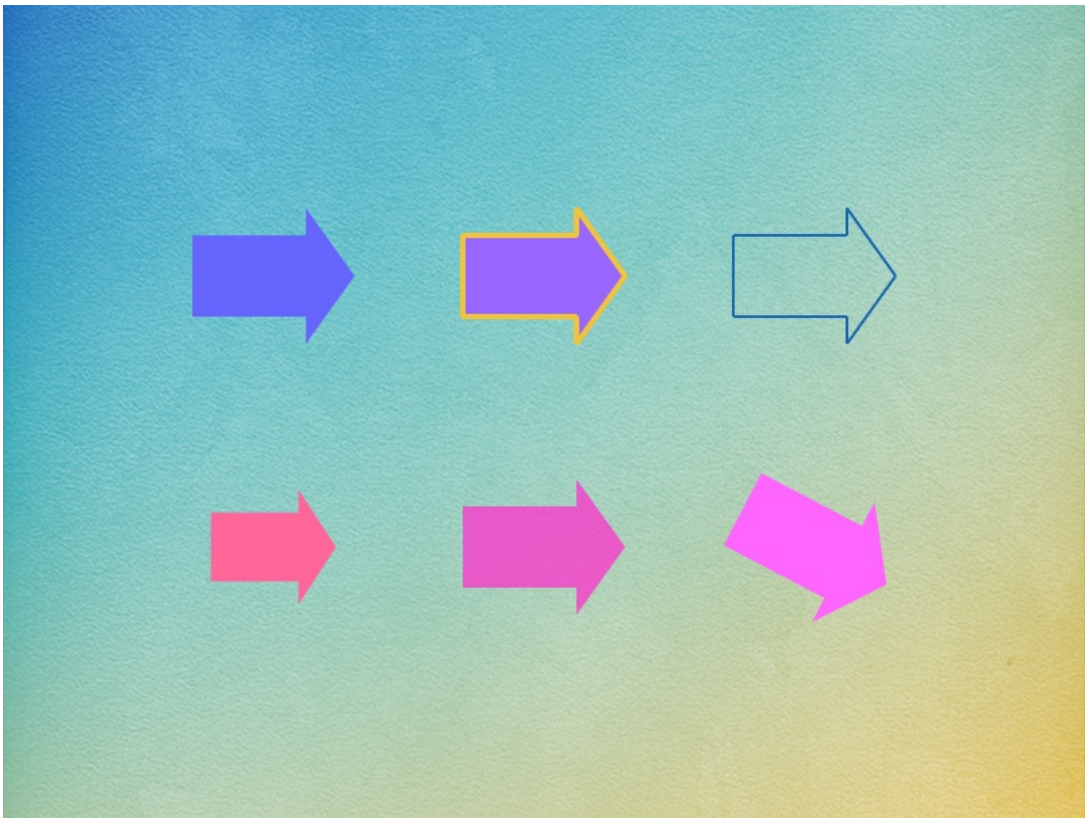
I don't know about you, but I'm always in need of grids in my games! They're just super-handy. From debug alignment tools to icon selection strips to backgrounds. The Grid Shape object allows you to generate them. You can set the width and height of the grid itself, as well as for the grid cells. The grid can either have a single color, or alternating cell colors and even have outline spacing between the cells, or not. It's pretty flexible!

Line



There's not a whole lot to say about Lines. They are, effectively, just lines :) You can draw them between any two points and give them a color and thickness. In WebGL you can actually specify a different thickness for the start and end of the line, allowing you to make some really interesting shapes (see the 2 lines in the top-right of the image)

Polygon

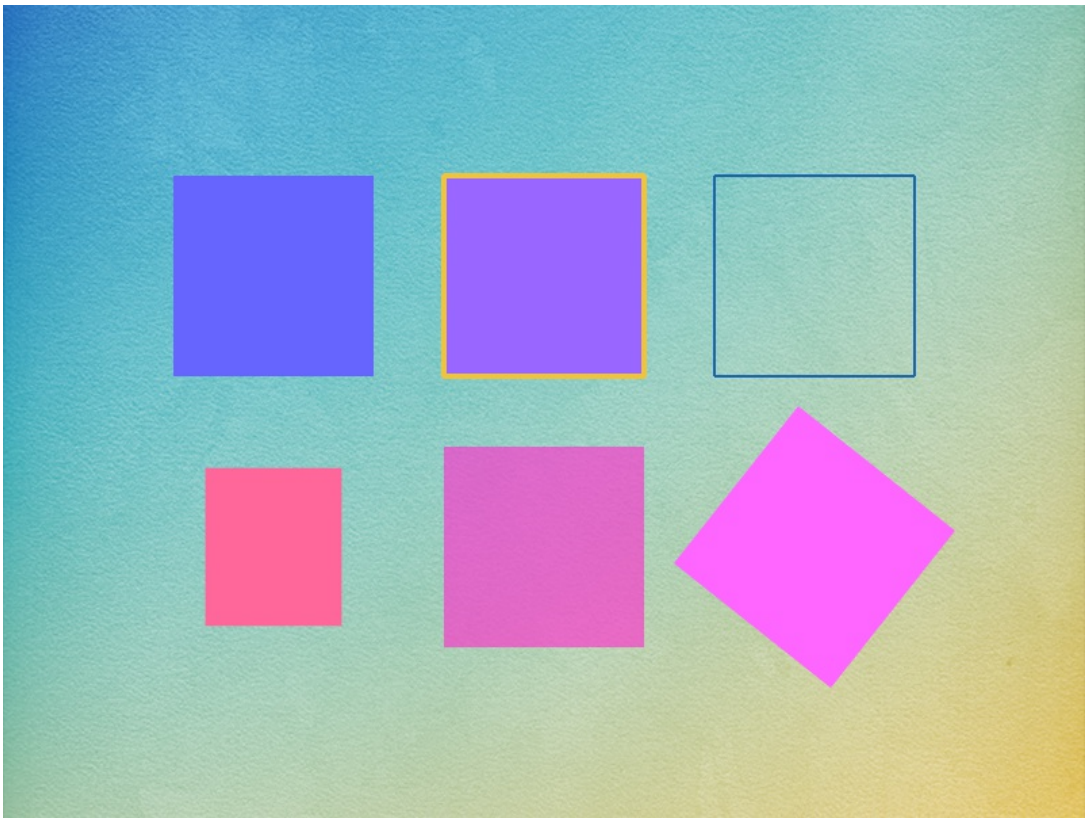


A Polygon is effectively just a list of points. The points can be provided in a number of different ways (as Vec2 objects, as an array, etc) and then you can either fill or stroke the resulting shape, or both. You have to be careful with the coordinates of the points when creating a polygon, as their relation to the x/y of the Game Object depends on their values.

Polygon Shapes also have an extra method called `smooth` which will automatically smooth the point data of the Shape. This can create some really lovely effects from originally quite 'jagged' data.

Ultimately they're really powerful and you can create some impressive paths really easily using them.

Rectangle

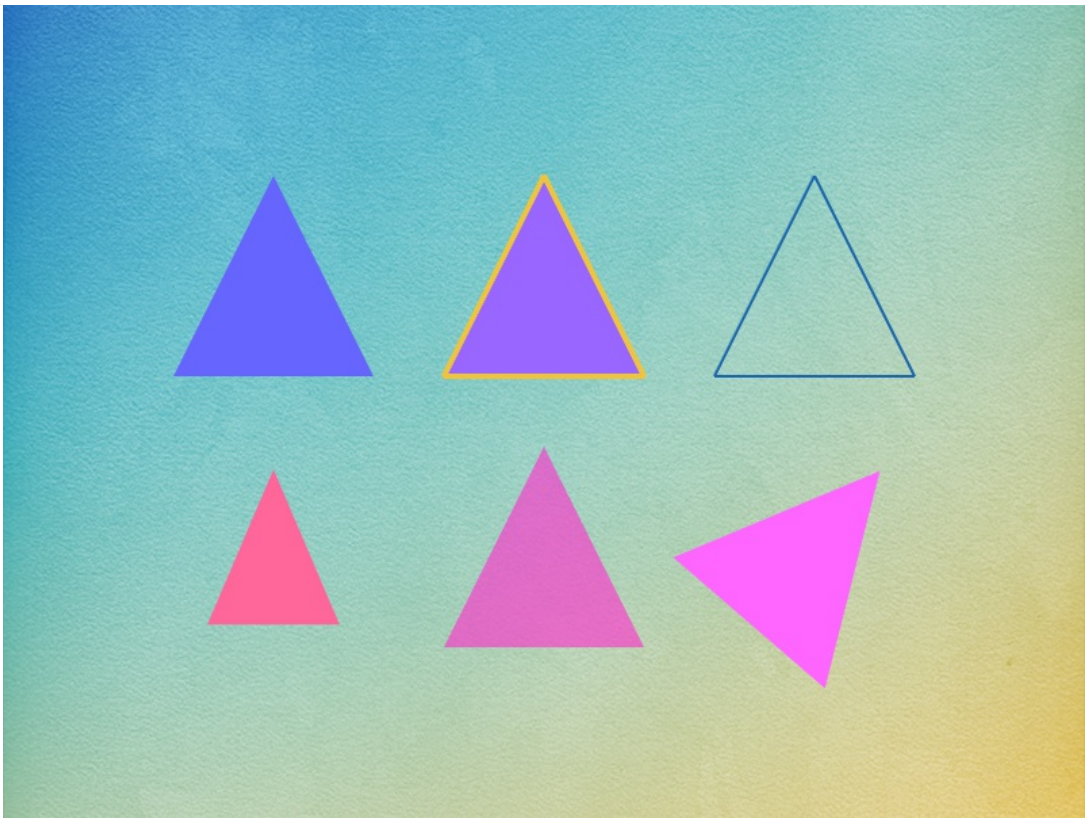


The Shape that started this all off in the first place! The humble rectangle. Endlessly useful, from backdrops, to abstract art, to bullets. They use a similar rendering path to Sprites, just without a texture, which makes them a little faster to process.

Star

The Star shape does as its name suggests: it displays a star. You can control the number of points in the star as well as the inner and outer radius of it. Super handy for simple particles.

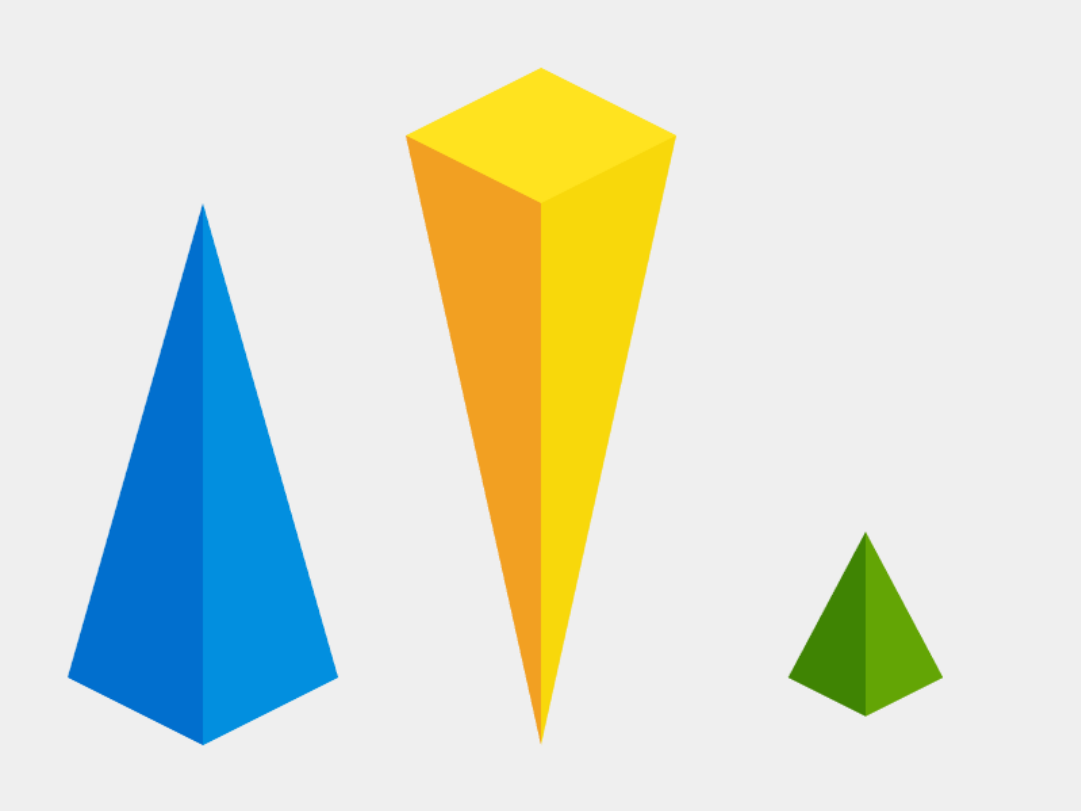
Triangle



The final standard geometric shape on offer is the Triangle. You have full control over the points used to make it and its fill and stroke. Internally it uses the `batchFillTriangle` method in WebGL, making it actually faster to draw than a Quad! Use them happily for bullets or abstract space ships, or anything else you feel like.

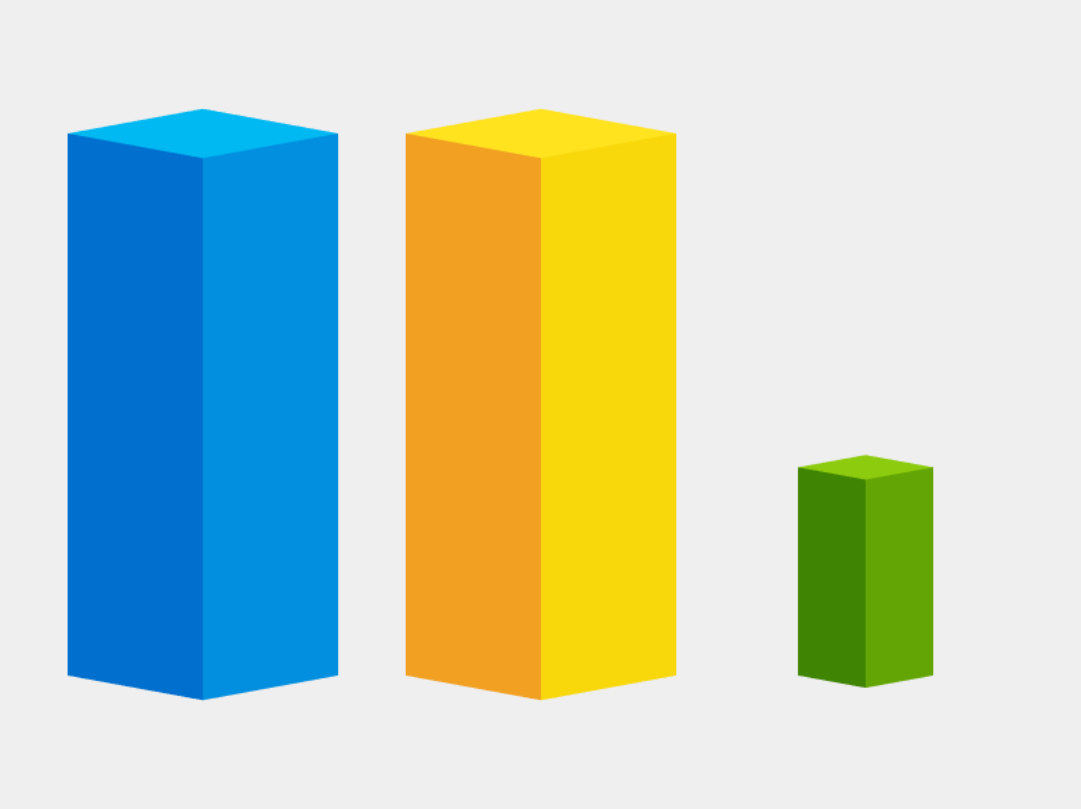
Iso Triangle

There are two more Shapes that are a little bit special but I couldn't resist adding them in. The first is the Iso Triangle.



This draws an isometric triangle, like a pyramid. You can control the colors of each face, if the pyramid is upside down or not and the width and height of it. It's a fun shape to include with the next one, the Iso Box.

Iso Box



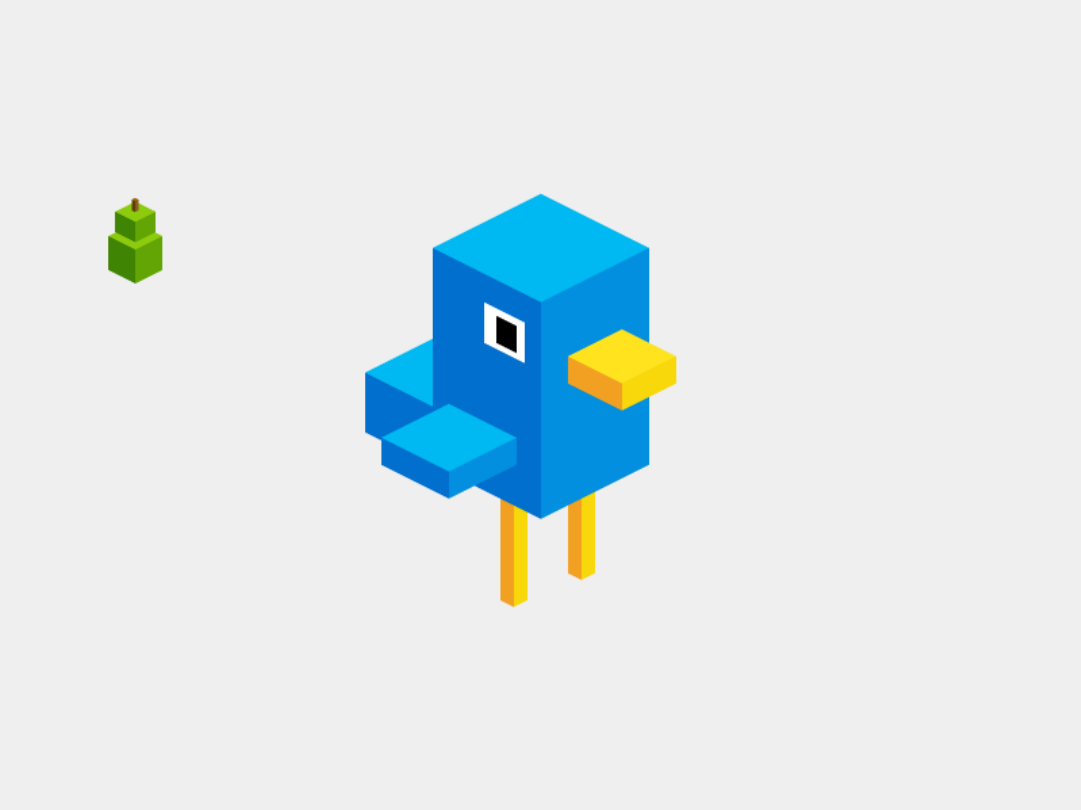
Similar to the Iso Triangle, this draws an isometric box. You can set the colors for each face of the box, as well as the projection angle and also which of the 3 faces are drawn.

The code is really simple:

```
var t1 = this.add.isobox(150, 500, 200, 400, 0x00b9f2, 0x016fce, 0x028fdf);

var t2 = this.add.isobox(400, 500, 200, 400, 0xffe31f, 0xf2a022, 0xf8d80b);

var t3 = this.add.isobox(640, 500, 100, 100, 0x8dcb0e, 0x3f8403, 0x63a505);

this.tweens.add({
    targets: t3,
    height: 300,
    ease: 'Sine.easeInOut',
    yoyo: true,
    repeat: -1
});

this.tweens.add({
    targets: [ t1, t2, t3 ],
    projection: 30,
    ease: 'Sine.easeInOut',
    yoyo: true,
    repeat: -1
});
```

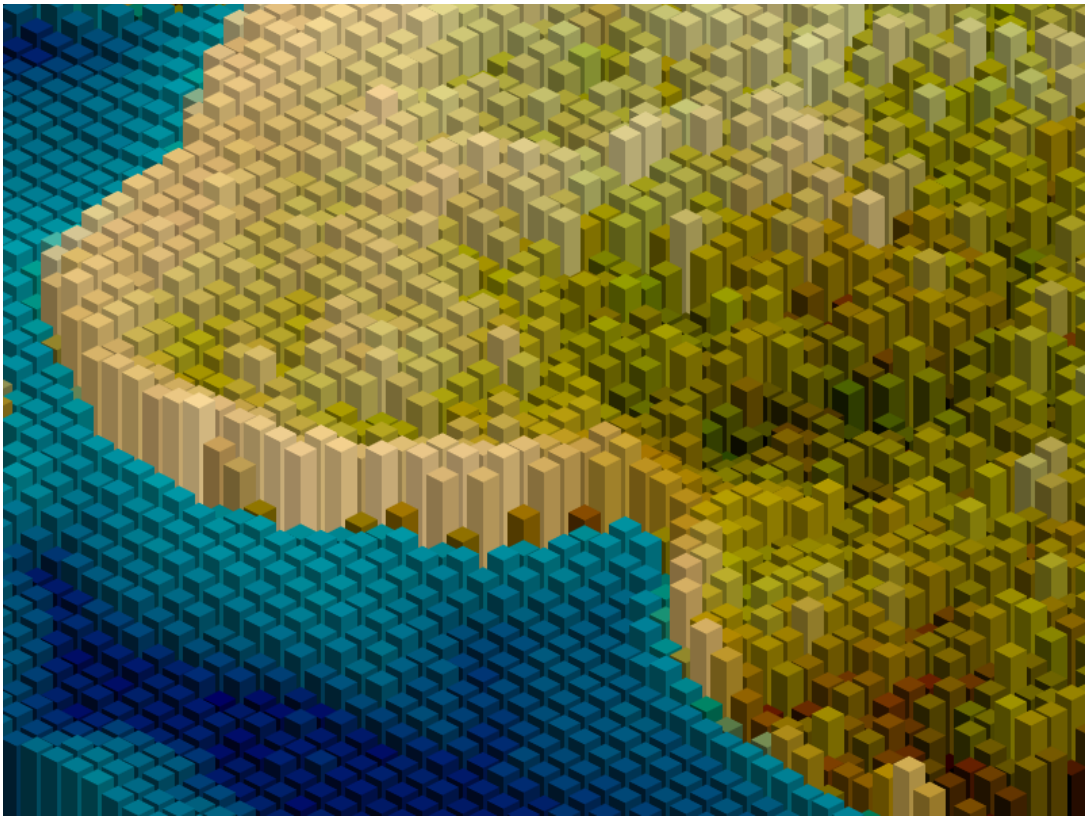
With some careful placement of the boxes, you can create some fun shapes.

Here's a little bird I created based on Crossy Road:



In the above example the various Iso Boxes are added into a Container and the Container is tweened to make the bird bounce. The little pear in the top-left corner was just an experiment in stacking a few boxes together. Perhaps you could extend this to create a game where you have to flap around and collect pears?

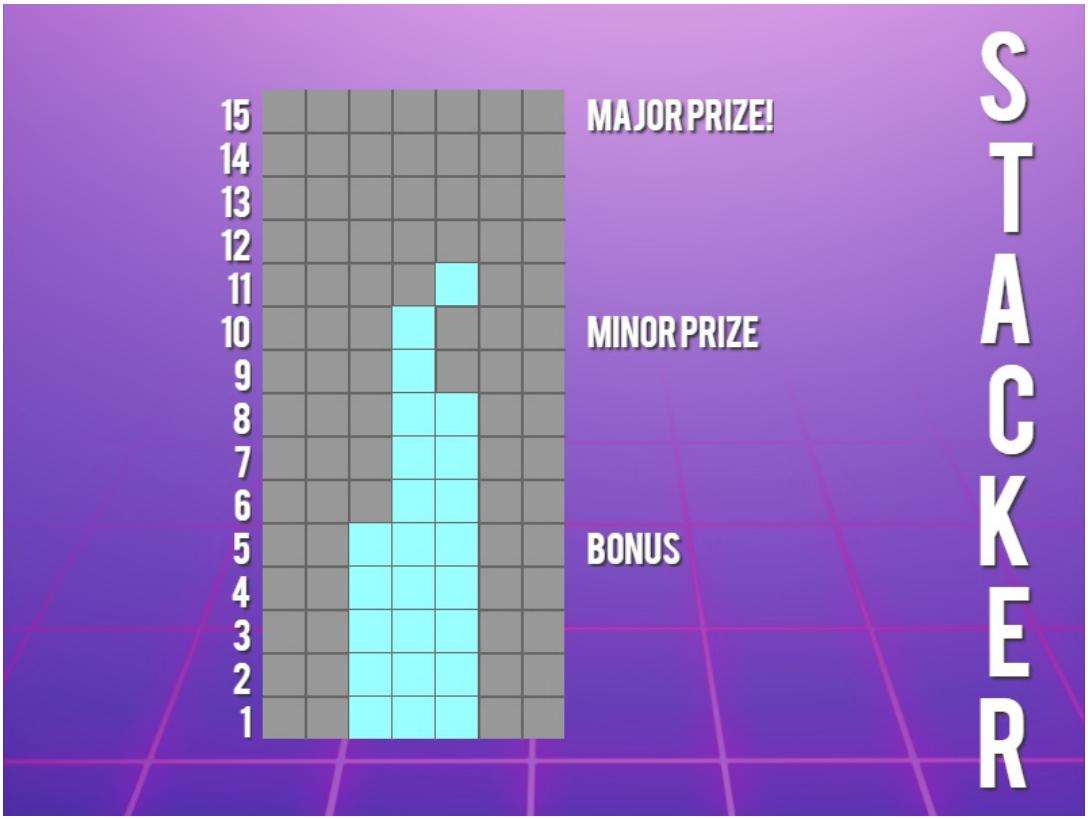
After creating the Iso Box I wanted to make a demo where you could fly over an isometric style landscape. Which is exactly what I did :)



Use the cursor keys to fly around. The code is extremely simple. All it does is read the pixels from a seamless PNG image I created and then update the Iso Boxes on-screen so that the height of them is based on the HSV value of that pixel. Hold down a cursor key and it just pans across the image. Simple but effective :) You can actually swap the image for anything you like in the code. Certain images create pretty crazy scenes!

Stacker Game

After finishing the addition of Shapes into 3.13 I felt like creating a little game that used them. So I built a variation on the Stacker concept. It's very easy: just click, or press the space bar, to drop the moving row down. Try and line it up perfectly with the one before to get the stack as high as you can. It gets faster, and the row gets smaller, the higher you get - see which prizes you ultimately win :)



The game uses the Grid Shape for the background and Rectangle Shapes for the blocks. The rest is just a PNG for the background and a font.

Hopefully, the code is easy enough to follow. If you've any questions then feel free to drop me a line on Slack / Discord or by email. If you'd like to start using Shapes now then grab the latest 3.13 build from the GitHub master branch. It will be released fully by the end of this week :) I've also completed all JSDocs for the new Shapes and there are lots of examples in the labs, so have fun and let me know what you make!

WWWBasic is a new project by Google that aims to bring the BASIC language to the web!

10 PRINT "PHASER RULES"
20 GOTO 10

A really great read: Learning BASIC Like it's 1983! In which the author talks about how he missed out on coding BASIC because he isn't old enough, and why it matters.

Finally, this is really important: How developers can defend open source from the EU copyright proposal

---

Phaser Releases

Phaser 3.12.0 released September 4th 2018.
Phaser CE 2.11.0 released June 26th 2018.

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.