Web Version Unsubscribe

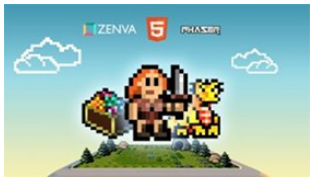
Welcome to Issue 122 of Phaser World

It's been a few months since the last issue of the Phaser World newsletter. So if you've forgotten you signed-up, or no longer wish to receive them, please use the unsubscribe link at the top of this email.

A lot has happened since issue 121. Heck of a lot. There have been no less than six new versions of Phaser 3 and four of Phaser CE, as well as multiple game and tutorial releases. That's a lot to catch-up with. Producing these newsletters takes an awful lot of work, so bear with me as things settle down again. It means some of the tutorials and games in the next few issues may be a bit older, but they've never been featured here before, so they should still be new for most of you.

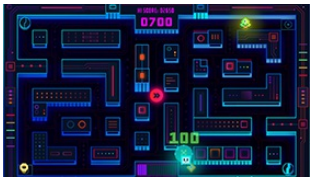
Until the next issue - and let's all hope it comes sooner than this one did! - keep on coding. Drop me a line if you've got any news you'd like featured by simply replying to this email, messaging me on Slack, Discord or Twitter.

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.

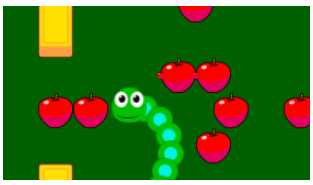
The Latest Games



Game of the Week

Bit Defendir

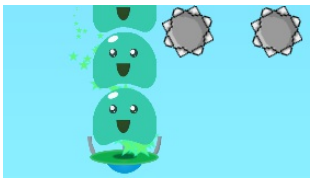
Created to celebrate 25 years of the Lenovo ThinkPad can you stop all the hackers stealing your encryption keys as you zoom around the maze?



Staff Pick

Impossible Snake

It's snake, but not as you know it! Rotate your way around the level, eating the applies and avoiding yourself and the walls.



Blob Catcher

Stack-up the blobs, avoid the mines and try to free them into the UFOs in this superb little action game.

Big Bubble Pop

You know the drill. Shoot those balls and match-up the colors to pop them all. Doesn't stop it being infuriatingly addictive though!

Space Mandala

Carefully choose from the ancient stones around the mandala to try and unlock those that are still sealed.

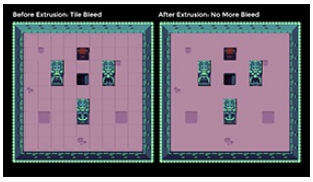
What's New?

Phaser 3.11.0 Released

Another massive update lands! With new features for cameras, lighting, texture cropping, tinting and many, many more.

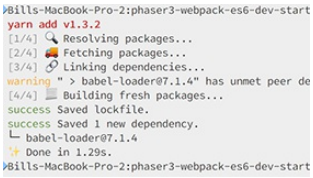
Phaser CE v2.10.6 Released

The latest version of Phaser CE is out with bug audio fixes and TypeScript updates.



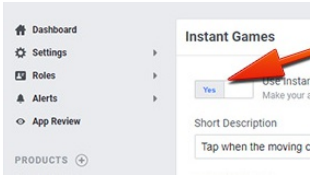
WebGL Tile Extruder

A tiny node app to extrude tilesets, avoiding the texture bleeding issues on WebGL.



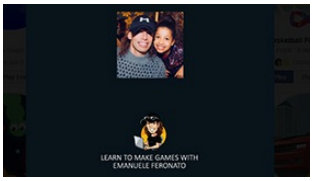
Modern Web Dev Set-up for Phaser 3

A guide to setting-up a modern web dev environment for Phaser 3 development.



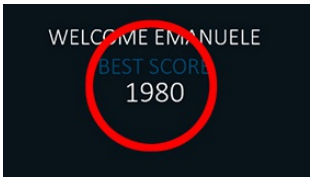
FB Instant Games in Phaser

A new tutorial on integrating Phaser with the Facebook Instant Games API.



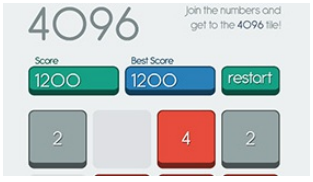
FB Instant Games Tutorial 2

Learn how to display the players name and profile picture in your game.



FB Instant Games Tutorial 3

Learn how to save player data in your Facebook Instant Games.



4096 Source Code Released

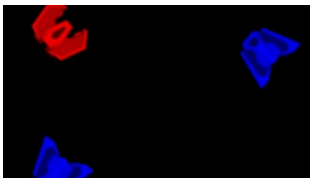
The full source code to the 2048 clone, written in Phaser 3.

Knife Hit Tutorial Part 1

Re-create the game Knife Hit using only tweens and trigonometry in Phaser 3.

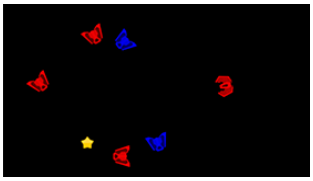
Knife Hit Tutorial Part 2

Re-create the game Knife Hit in Phaser 3. This time adding in hitting other knives.



Socket.io Phaser 3 Tutorial

Creating a basic multiplayer game with Socket.io and Phaser 3.

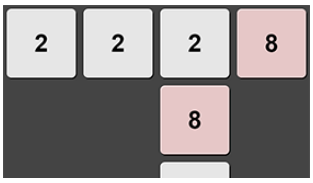


Socket.io Phaser 3 Tutorial Part 2

In part 2 of creating a multiplayer game with Socket.io and Phaser 3 it adds player removal, input and collectibles.

Phaser 3 Preloader Tutorial

How to create a Preloader Scene with progress bar in Phaser 3.



2048 Tutorial Part 1

A new tutorial series on re-creating the game 2048 in Phaser 3.

2048 Tutorial Part 2

The second part of tutorial series on re-creating the game 2048 in Phaser 3 adds swipe controls and sprite sheets.

Thank you to our awesome Phaser Patrons who joined us recently (well, I say recently, but this newsletter is 3 months old, so we have a lot to catch up on!):

Denis Zubo - **Dmitry Matveev** - **Matt Lintz** - **Joseph Bowman** - **Michael Carver** - **Nicholas Kramer** - **Aiken** - **Webellion** - **Oddvar** - **Tomas Frinta** - **Andrew Ozdion** - **Nicolas Roehm** - **Matt DeWolfe** - **Daniel Weipert** - **Robert Larnach** - **Thomas Boccinfuso** - **George Frick** - **Rob** - **Michael Roth** - **Scott McFarlane** - **James Sewell** - **Christin Morton** - **Richard Elms** - **Jon Harsem**

Thank you to the following for increasing their pledges:

Steven Landman
Johan Lindfors

also, thank you **DBA**, **Walter Heil**, **Douglas Lapsley** and **Ayuth Mangmesap** for the PayPal 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 #122

Welcome to Dev Log 122. It's been a good while since I wrote one of these, or indeed since there has been an issue of Phaser World. There are multiple contributing factors to this. The overwhelming one is that I've been really ill. It floored me physically. And when you spend so much time in pain and exhausted, it floors you mentally as well. It took several months and multiple tests to get over. Doctors are still getting to the bottom of it, but I'm thankfully now feeling better than I have in a very long time. Better than I have for years in fact. During it, it really took its toll though.

I've said before about how writing all the news on the website, and putting together Phaser World each week, is a monstrous drain on my time - and it absolutely is. I cannot stress that enough. I managed to do it for so long because I'd then work on Phaser into the small hours or had other people helping with development or Dev Logs, so the two things balanced out if a little precariously. Being ill threw that out the window. Some days I didn't have enough strength to get out of bed, let alone work. So rather than take on too much I made sure that Phaser development carried on, and put everything else on hold during this period. That is why releases of Phaser have been constant over the past few months because I used my few hours of work each day to work on that, but everything else has been silent.

Another change that happened is that Felipe has left to work for another company. Felipe was responsible for coding the renderer and various other parts over the years. His leaving was very abrupt (literally a couple of days notice) which gave me no time to plan for a replacement. Falling ill prevented this from

progressing and meant the renderer was effectively frozen. It didn't matter terribly, because what was there worked. Yet, there was still a lot I wanted to do with it but couldn't because other areas of the API needed my attention first.

In hindsight, it was the right move for Felipe to leave. You can tell when someone becomes far less invested in a project than before. Things take longer to happen, involvement is reduced. In short, the passion just isn't there anymore. I totally get that. This must be the most common reason for open source projects to go stale: running out of passion. It was frustrating at the time, especially as it occurred when I started to get more seriously ill. Yet, looking back it was the right thing to happen. I had felt it coming for a while and I'm glad he's now doing work that excites him again.

It's actually a brilliant thing to have happened for me as well. It forced me to really dive into the renderer code and unpick just what on earth was going on in there. I now understand every single line of it, which allowed me to recode large chunks of it for the 3.11 release. The reason I wanted to do this work is that it had grown from being a quite compact and flexible system into a huge monolith. With every new Game Object more and more functions had been added to the renderer. To the point where it had literally hundreds of lines of duplicate code and even the duplicated code was not always consistent. Vars in one function may mean something different in another, and requiring a change, such as adding Camera

alpha support, meant updating code in lots and lots of places. The WebGL side may have been called a pipeline by name, but it didn't operate like one - there was no input or output, no ability to pass Game Objects through multiple pipelines, it was to all intents and purposes a pretty standard batch system.

So I dug in and started following the threads. Unraveling what I found and making notes the deeper I got. It took weeks of work but the end result was more than worth it. As well as fixing numerous bugs reported on GitHub everything is now a lot more logical and extensible. The code is consistent. Game Objects are responsible for their rendering. Common functions are used all over the place and I documented lots of it too. There is more work to be done and plenty of features I'd like to implement, yet I'm more than happy to do those myself now. Especially as it's some of the most rewarding and exciting work in the API. Or, if another developer comes in, I'm happy it will be far less confusing for them to pick-up and work with.

Where does this leave Phaser? Thankfully, in one of the strongest positions, it has ever been in. The API becomes more clean, more documented and more powerful with every iteration. I'm now fully in control of every aspect of the framework. And I'm feeling healthier than I have in a *very* long time. I still need to find a way to ensure this newsletter returns to a weekly publication because realistically I feel like at the moment I can only manage an issue every 2 weeks. Ultimately regularity will come down to finding the right sort of people to help me. I've tried before, several times in fact, and those who started helping always vanished after a couple of weeks. Which I think goes to show just how demanding the work is. Yet, there must be a way. Perhaps it involves a more community-driven approach to writing the news? If I could get a core team of people who all they had to do was contribute 1 article a week, that would allow me to focus purely on writing the Dev Logs each issue, and not all the other content as well. I'm not really sure what the answer is right now. It's something I'll keep working on. Because I really want Phaser World to carry on being published, just not to the detriment of anything else.

New Phaser Releases

There have been six Phaser releases since the last Dev Log. It would take an awful lot of space to detail them all, so instead I'm going to list them below and cover their most significant changes. Use the links to read the Change Log entries for each release.

Phaser 3.7.0 (8th May)

Most of the 3.7 release focused on updates to the Loader. I overhauled most of it. Loader Packs were bought back in, allowing you to specify a json file full of assets to load. The Loader was also changed to be a lot more flexible - you could add files to it, even during a load, you could call it at any point and the process flow was turned around too, so files were processed as they were streamed in, instead of in one huge batch at the end. 3.7 also introduced the new CanvasTexture object, allowing for easy canvas creation and texture binding.

Phaser 3.8.0 (16th May)

This release saw me recoding the Plugin Manager from scratch. Phaser has always used plugins extensively internally but this release opened them up and buildt in a lot of new features making them easy for you to both create and digest. You can create global or scene level plugins, bring them in via your build system, game config or the Loader, and have a lot more control over how they are used. Other updates include the ability to pass in your own context for the renderer to use, allowing you to use WebGL2 contexts. There were also lots of new Game events added and, of course, bugs fixed.

Phaser 3.9.0 (24th May)

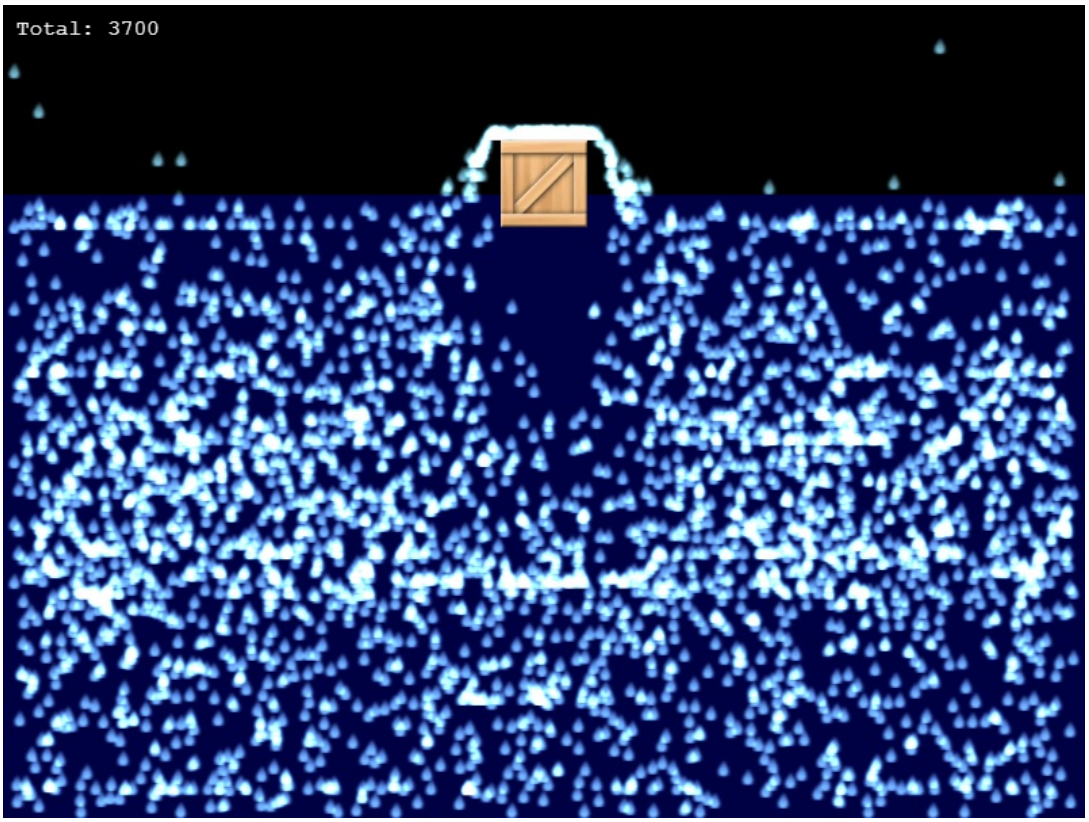
A smaller release this time that added a bunch of new features in. Including Camera lerp, Camera follower offsets, new Arcade Physics config properties,

Keyboard focus fixes and Container updates.

Phaser 3.10.0 and 3.10.1 (13th June)

3.10 was a truly massive release. I think easily the single largest release in Phaser 3's life (until 3.11 anyway!) - in short, I virtually rewrote the Input Manager and Input Plugin, so they natively supported multi-touch and multi pointers. I also added in custom cursor support, which was really quite fun to implement :) The input systems, like the Keyboard Manager, were removed and instead replaced with Scene-level plugins. This means that every Scene now has its own instance of the Keyboard Plugin. The benefits of this are two-fold: First, it allows you to now entirely exclude all of the keyboard classes from a custom build, saving a lot of space if not required. Secondly, it means that the Scenes themselves are now responsible for keyboard events, where-as before they were entirely global. This means a Scene can be paused and stop processing keyboard events, and stop having its Key objects updated, while another Scene can still carry on doing this. It also prevents key related callbacks in sleeping Scenes from being fired. I recoded Gamepad support too. It works as an input plugin too, like the Keyboard, and just handled gamepads so much better than before. I know they're a bit of an edge-case and lots of games don't use them, but I figured if you're going to support them, support them as well as you can.

Another huge update in 3.10 was that I changed how Arcade Physics works. It now uses a fixed time-step for all internal calculations, and you can control the fps rate of it easily, as well as manually advance it (great for multiplayer games). You could even time scale it. There was a new damping feature too. I also updated the Data Manager, allowing for much easier access to values. There are lots of other updates and fixes. 3.10 really was significant.



Phaser 3.11 (13th July)

Exactly one month after 3.10 I released 3.11. If 3.10 was a large release, this one eclipsed it by several magnitudes. I hadn't planned on it taking quite so long or being as large as it was, but it touched some key areas that in themselves meant it could be nothing less.

3.11 saw me recoding nearly all of the Texture Tint Pipeline, the key pipeline most Game Objects use to render. And you don't just touch a 'little bit' of the renderer. Once you start you really can't stop until it's all covered. Which is why it took a month to finish this release, but it was well worth it. I added in Texture Crop support and a brand new tint-mode to the renderer. Texture cropping allows you to crop a Game Object using a rectangle, which is a great way to make things like health or progress bars without the need for using a mask. The new tint mode introduced a full-color tint, rather than an additive one like the old one. Want to flash a sprite white when hit? Use the new tint mode! (something impossible with the old tint mode)

I also gave the Camera another boost in 3.11. I added in Camera alpha support, something much needed, allowing you to fade out a whole camera. I added in deadzone support for the camera follower, two brand new Camera effects: Pan and Zoom, which allow for easy panning and zooming around your game world. I also added in lots of helpful properties and methods like dirty, midPoint, centerOn and worldView. I fixed-up the Camera Manager to remove a lot of redundant

pooling and let you easily extend your own Camera classes. I consolidated how roundPixels and the pixelArt game config properties worked, which in turn lead to lots of fixes with Cameras following objects, entirely removing the previous 'stutter' that could be visible at high zoom levels.

BitmapText also received some TLC. I added in multi-line alignment support, merged in a great PR that unified the bounds calculations, greatly optimized the internal calculations (and how often they happen) and sorted out some callback issues. Another huge update was with how Dynamic Tilemap Layer culling was handled. Read more about that in the following mini-tutorial in this Dev Log :)

There were stacks of new features too, like rounded rectangles, the ability to scale TileSprites textures, pointer interpolation, Keyboard enhancements and yet more updates and bug fixes. Sometimes, Phaser releases can feel a bit like they are fire-fighting. I.e. mostly sorting out issues. And sometimes they feel like proper evolutionary steps for the framework, and 3.10 and 3.11 certainly fall into that camp. And 3.12 will too.

What's planned for 3.12?

Development of the 3.12 release has begun already. There are going to be two core elements to it: The introduction of the new Scale Manager, and a Facebook Instant Games plugin. I've been planning out how the Scale Manager should

work for a good while now. The overall concept is to be far less dominating than the one in v2. Looking back, the v2 Scale Manager had features and peculiarities in it that quite frankly even I struggle to understand.

It really doesn't need to be as complex for v3. I feel there are a few fundamental things it should offer:

- High DPI Resolutions. While there is no denying that high dpi drains mobile batteries faster than Mothra in Godzilla, it's really important for certain games. So part of the new Scale Manager will be ensuring that you can specify

an increased resolution and it'll be used supported across the API - in the Input system, the renderer and certain Game Objects like Text or TileSprites that use their own internal canvases.

- The ability to scale the canvas to fill the browser. This is a given

really and is what most people want. The plan is that you will be able to set a 'game size' and a 'canvas size'.

- Responsive support on a per Scene basis. A while ago I added in support for the Camera system to have a base

resolution. This worked on a per-Scene basis. It meant that you could have a background Scene that was a 1:1 pixel size, matching the canvas, and then a foreground Scene which was scaled to fit a particular aspect ratio.

This, I think, is the crux of what the Scale Manager in v3 will provide: The ability of each Scene to handle scaling differently. There will be a base canvas size of course. Once established though, each Scene can implement their own scaling on-top of this, in whatever way is most suitable for that Scene.

I'm also going to make sure that the Scale Manager doesn't get *too* involved in what's going on. Once it has handled the basics it will be up to you deal with more of the edge-cases. A lot of work involved in scaling for browsers actually takes place with the CSS and page layout, and I'm adamant that Phaser shouldn't be doing this for you. You should be understanding what's needed in the layout and handling it yourself. I can provide templates and guidance, but I don't want Phaser injecting any CSS into the DOM anymore.

Another major feature of 3.12 will be native support for Facebook Instant Games. I cannot stress enough how much support Facebook have given Phaser this year. It's early but exciting days in our partnership, and quite frankly Instant Games are a pretty hot area to support right now. Their API is clean and pretty powerful already, but there are definitely elements of it that will benefit from easier integration with Phaser. So I'll be building a Phaser plugin for Instant Games and ensuring it works smoothly with the new Scale Manager and all other existing systems.

I feel that 3.12 development will be quite rapid. In my mind, I don't think there will be a whole lot of coding required, but there will be *lots* of testing needed, especially across devices, which is always time-consuming! Ideally, I'll release in around a months time. But don't hold me to that. If it needs longer, I'll give it longer. I will, of course, stay on top of the issues in GitHub too. Don't under-estimate my commitment not only to supporting the new and shiny but to also making sure issues are resolved and more examples are created. My passion for Phaser remains as strong as ever. My resolve to make it the best it can be is unwavering. I trust that even in the absence of Phaser World and Dev Logs this has still been apparent.

Tilemap Mini-Tutorial

Before I cover how to handle Tilemap culling in 3.11 it's well worth explaining the difference between a Static and Dynamic Tilemap and how they differ internally.

The main difference is that with a **Static Tilemap Layer** once the layer data has been set, you cannot change it. The tile data is usually read in from a file, i.e. imported as a CSV or Tiled map, or it can be set at run-time. But once the map data is parsed the static layer constructed and remains fixed. Internally it creates its own array buffer which is populated during the parsing. This buffer contains the vertex data for every single visible tile in the layer.

When a Static Layer renders, instead of iterating the tiles or working out any culling it literally dumps the entire pre-calculated array buffer onto the renderer.

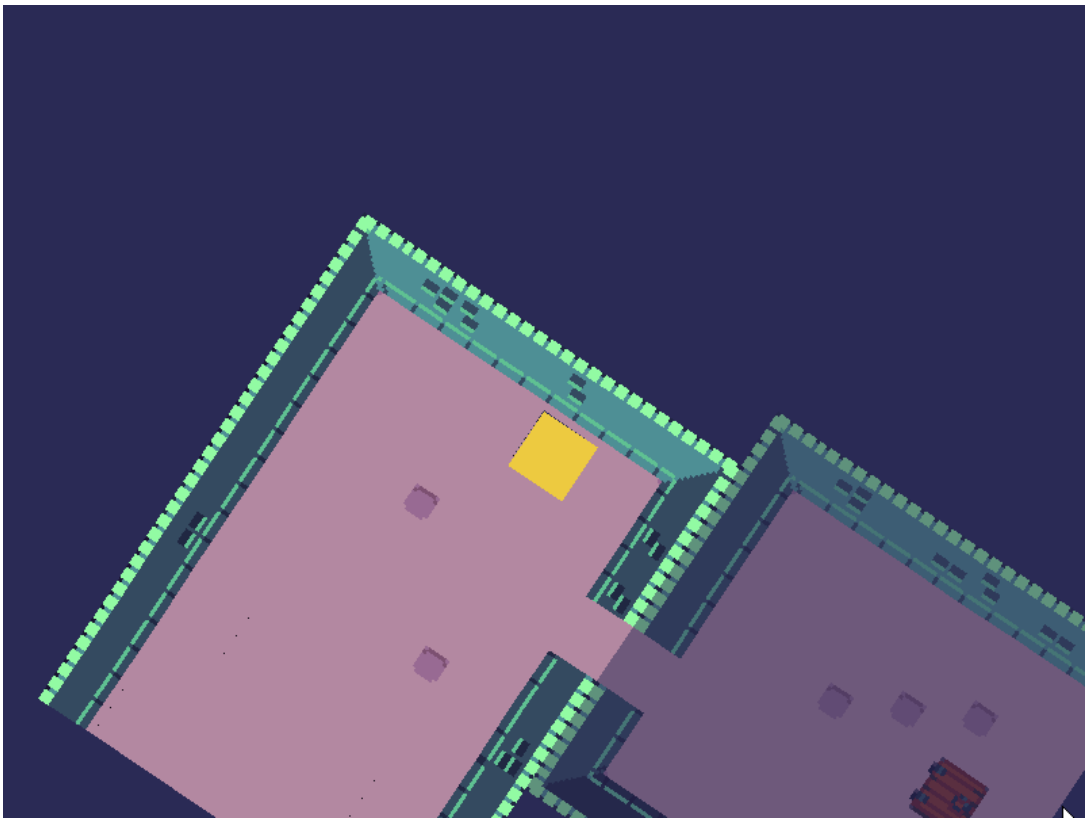
There are pros and cons to this approach. For a start, there is no tile iteration or vertex calculations per frame. It's literally a direct buffer transfer, which is about as fast as you can get when it comes to WebGL. The downside is that if you have a particularly large layer, in terms of quantity of tiles, not pixel dimensions, then that's a lot of data to dump on the GPU every frame, especially considering potentially the vast majority of it isn't even going to be visible after rendering (as it'll be off-screen). Because it dumps the whole buffer every frame there is no culling involved. So they're fast, but equally, a bit dumb. And should likely be avoided if you have got a large layer to deal with. For a single-screen layer though, it's fast. Very fast.

Dynamic Tilemap Layers are different in that you can change the contents of the layer as often as you like. They traditionally start out populated with data in the same way as a Static Layer does, however you can manipulate it as much as you wish. Adding tiles, changing tiles, even changing the layer size. Because they can be changed at run-time it means that it calculates which tiles are visible to the Camera/s *every frame*, and then uploads the vertex data for those visible tiles to the WebGL batch. To help avoid uploading too much data the tiles are culled first. This means it only includes those tiles which are visible to the Camera, based on the Camera worldView rectangle.

Tilemap Culling

Most of the time the default culling just works. But there are cases where you may need more tiles to be visible. For example, if you rotate the Camera (especially if you also zoom it), or are moving it really quickly, then you may notice 'clipping', which is when the next row of tiles haven't been drawn yet because the Camera has moved or rotated further than the culling calculations allowed for, or the tiles fell outside of the expected rectangle.

The animated gif below demonstrates the clipping effect:



To resolve this 3.11 added three new properties to the Dynamic Tilemap Layer. The first is the `skipCull` boolean. If you set this to `true` then the layer will not cull anything at all. Every single tile will be drawn. There are occasions where you may need this. It's very game specific though and generally, you should try and use culling where possible. But if your map size is quite small and you've got lots of Camera transforms going on or lots of Cameras, then it may be beneficially to disable culling entirely. Remember, tiles are re-culled for *every single* Camera that can view the map layer.

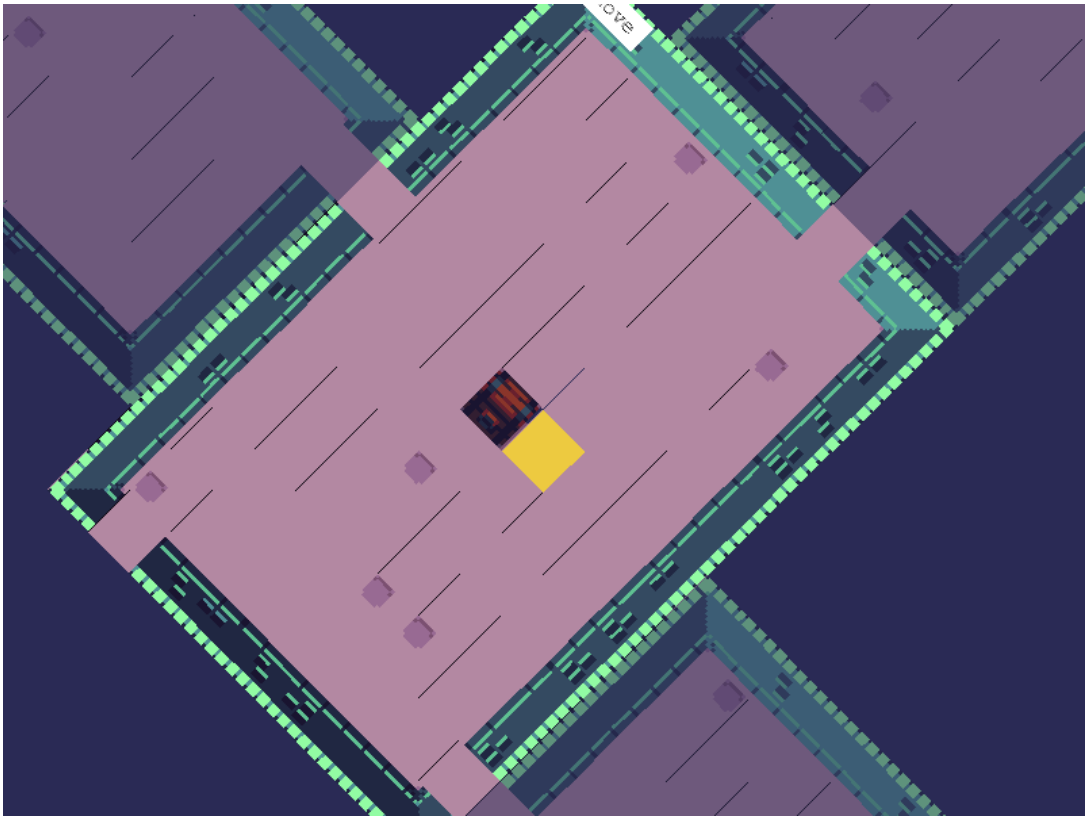
However, if you want to use culling, and you should in most cases, then it's worth learning about the two new properties: `cullPaddingX` and `cullPaddingY`. These are integers, set to the value 1 by default, and they control how many additional rows or columns of tiles are added to the culling calculations. If you find that you are seeing tiles get clipped off too early, perhaps if the Camera is rotated, then increasing these values will help massively. You can change them directly or use the method `setCullPadding`.

This example shows the difference setting cull padding can have. Rotate and zoom the camera using the controls and change the cull values.

Tileset Extrusion

If you are using tilemaps under WebGL then you will almost certainly run into the issue of artifacts appearing between tiles during rendering. This happens because the way WebGL calculates texture coordinates and is the result of the tile textures bleeding into each other. Texture Atlas software solves this for you by offering atlas borders, padding or extrusion around the frames. But lots of Tilesets are drawn with no such padding or extrusion at all, making them highly susceptible to texture bleeding. This isn't just an issue for Phaser. All WebGL based frameworks would exhibit it, as does Unity.

Here's a screenshot showing just one way in which it can manifest:



It's not the only way. You may also notice it if you have transparent areas of tiles and you start to see other tile textures fill those in. Or you may only notice it happen when scrolling at specific camera offsets, or on certain devices.

However it manifests though, it's important to get used to extruding your tilesets so it never happens at all. Thankfully, Michael Hadley has developed a node script to automate this process for you.

Using it is simple. You call tile-extruder and provide the path to your tileset and the dimensions of the tiles. It will then rebuild a new tileset with extrusion applied. You'll need to either import the tileset back into Tiled (if you're using it), or adjust your Phaser code so it knows about the new border and margin values. Both

approaches are detailed in the documentation. Pick whichever you prefer and then random lines and texture bleeding will be a thing of the past!

Whatever you do, make this part of your workflow.

A spilt paint fluid-dynamics shader! (warning: only really suitable for desktop-grade GPUs)

Tales from inside 90s Nintendo - from the man who made Mario's face.

Adobe Flash's Gaming Legacy - Thousands upon Thousands of Titles - and my efforts to save it.

Phaser Releases

Phaser 3.11.0 released July 13th 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.