CMPM 120

Tilemaps

Activate Transcription

Questions

https://forms.gle/WLpCdXGDzU9CAqfW6

Schedule Overview*

6/22	Introduction
6/24	Programming Our First Phaser Game
6/29	Version Control & Debugging
7/1	Scenes, Loops, Physics
7/6	Input, Cameras, State Machines
7/8	Assets, Collisions, Transitions
7/13	JSON, Tilemaps, Map Editors
7/15	Tweens & Particles
7/20	Special Topics
7/22	Final Presentations

*This will inevitably change a bit

Schedule Overview*

6/22	Introduction		
6/24	Programming Our First Phaser Game	Rocket Patrol Tutorial Due	6/26
6/29	Version Control & Debugging	Rocket Patrol Mods Due	6/29
7/1	Scenes, Loops, Physics		
7/6	Input, Cameras, State Machines	Endless Runner Due	7/6
7/8	Assets, Collisions, Transitions		
7/13	JSON, Tilemaps, Map Editors	Final Game: First Build	7/13
7/15	Tweens & Particles		
7/20	Special Topics		
7/22	Final Presentations	Final Game Due	7/22

*This will inevitably change a bit

The Week Ahead

- Tuesday, July 13, 2021
  - Final Game: First Playable Build [~15-30 hours] due by 9am
  - Game Cameras due by 9am
- No new assignments after this point
- Thursday, July 22, 2021
  - Final Game Due (9am)
  - Final Presentation

Now is the time to work on your final game. If you have previous assignments that you haven't done (or want to update) now is the time to do it.

Your Final Game Teams

Add yourself to your team:

https://canvas.ucsc.edu/courses/44176/groups#tab-6003

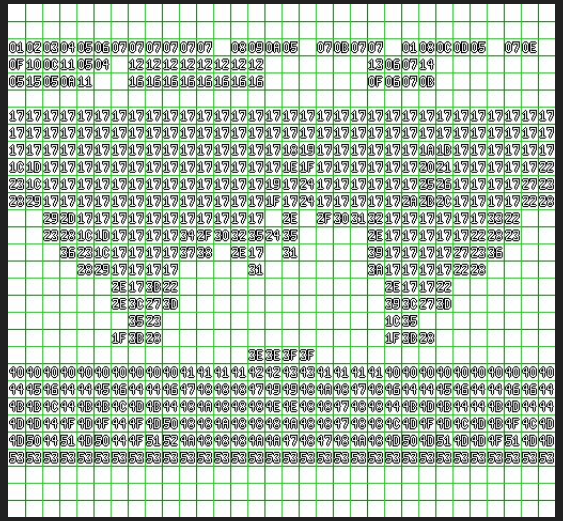
During lecture, which do you feel like you learn the most from? (You may select more than one if desired.)

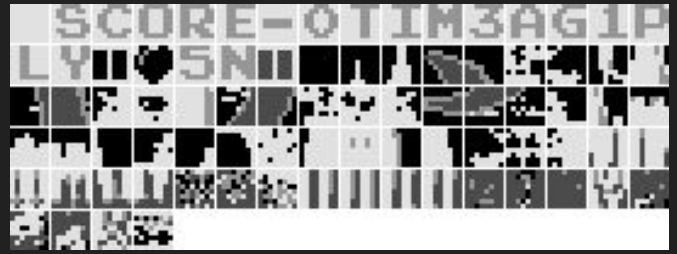
Talking through slides with conceptual explanations	10 respondents	32 %
Reviewing pre-made code examples in VSCode (no debugging; but lots of unfamiliar code)	19 respondents	61 %
Creating fresh code examples from scratch (shows debugging processes but might waste time doing so; but all code is created in front of you)	25 respondents	81 %
Asking questions about your own code (e.g. via screen sharing)	6 respondents	19 %

Tilemaps

Historical motivation

Earlier game platforms only supported complex full-screen graphics by way of a hardware-supported tileset and nametable data structure.





http://www.dustmop.io/blog/2015/04/28/nes-graphics-part-1/

Present-day motivations

- Establish a consistent visual language to communicate with the player
- Define the game's rules over a convenient abstraction, without working about pixel-level details
- Allow designers to manipulate their world designs without directly modifying the game's code (and waiting for the game to rebuild/restart every time)
- ...

Infiniminer → *Minecraft*, not a hardware limitation

By Grabthar's Hammer, What a Savings: Making the Most of Texture Memory with Sprite Dicing

By Jon Manning

https://www.gdcvault.com/play/1025419/By-Grabthar-s-Hammer-What

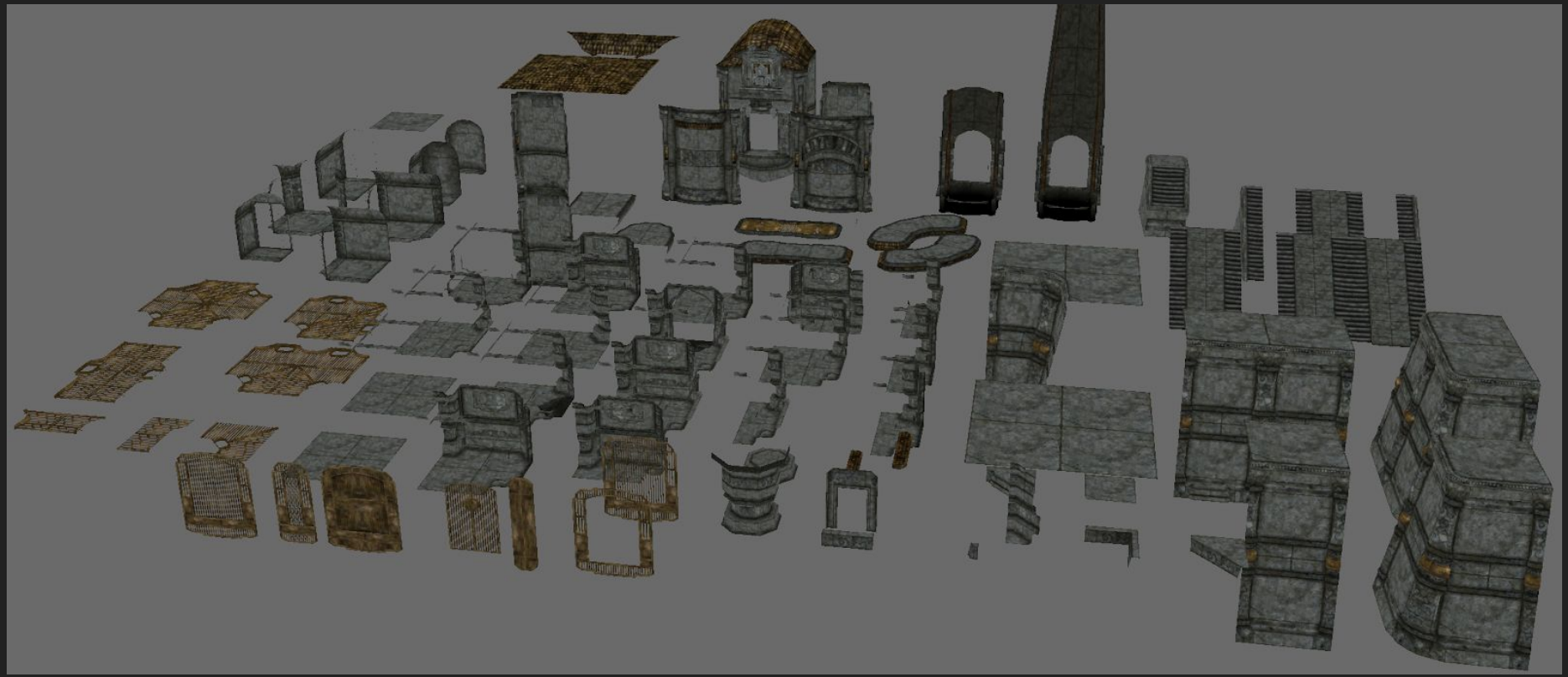


Image by Joel Burgess
http://blog.joelburgess.com/2013/04/skyrims-modular-level-design-gdc-2013.html

Modular Level Kits

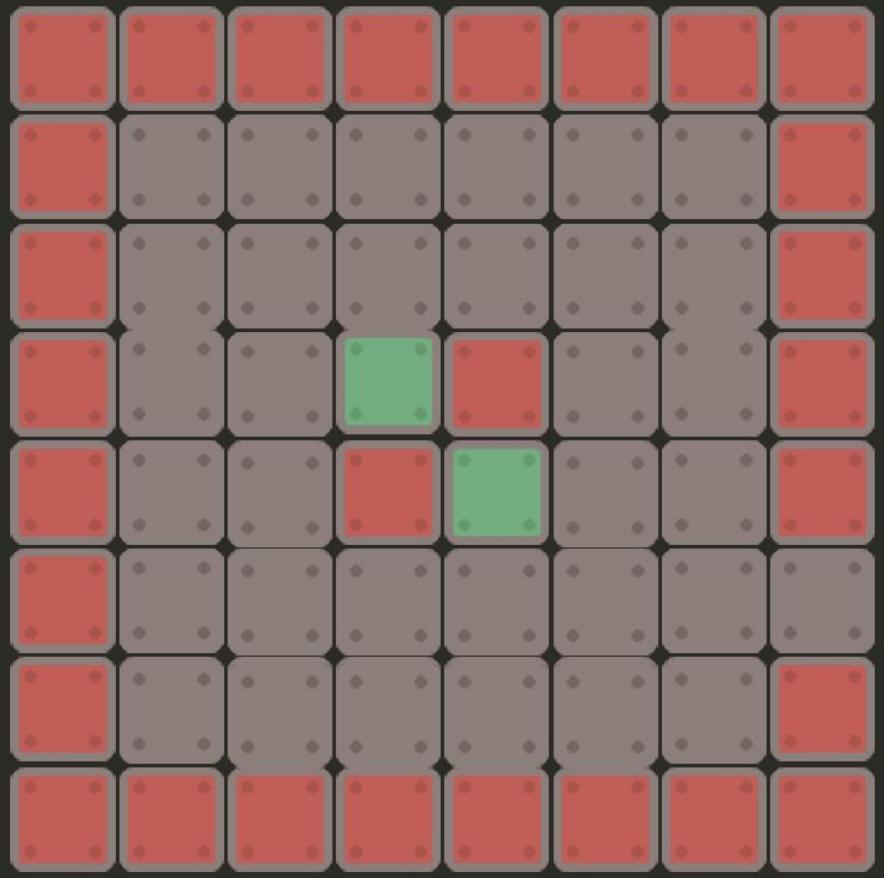
http://twvideo01.ubm-us.net/o1/vault/gdc2016/Presentations/Burgess_Joel_Modular%20Level%20Design.pdf

https://www.youtube.com/watch?v=QBAM27YbKZg

https://youtu.be/JKn7u09mR8M

Tiles are efficient

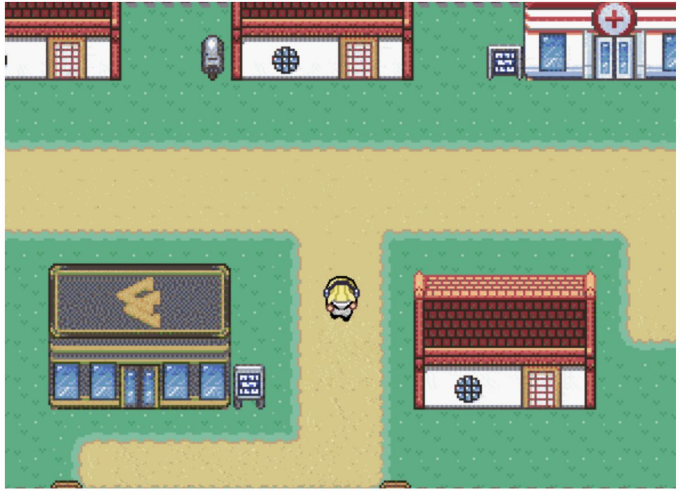
```
var map = [
    [1, 1, 1, 1, 1, 1, 1, 1],
    [1, 0, 0, 0, 0, 0, 0, 1],
    [1, 0, 0, 0, 0, 0, 0, 1],
    [1, 0, 0, 2, 1, 0, 0, 1],
    [1, 0, 0, 1, 2, 0, 0, 1],
    [1, 0, 0, 0, 0, 0, 0, 0],
    [1, 0, 0, 0, 0, 0, 0, 1],
    [1, 1, 1, 1, 1, 1, 1, 1]
];
```



Modular Game Worlds in Phaser 3 (Tilemaps #1) — Static Maps

Michael Hadley Jul 4, 2018 · 10 min read

This is a series of blog posts about creating modular worlds with tilemaps in the Phaser 3 game engine. In this first post, we'll go from zero to creating a Pokemon-style top down game world that a player can explore:



Final example we'll create — graphics from Tuxemon.

Excellent reference guide!

https://medium.com/@michaelwesthadley/modular-game-worlds-in-phaser-3-tilemaps-1-958fc7e6bbd6 (from the creator of Phaser's tilemap system).

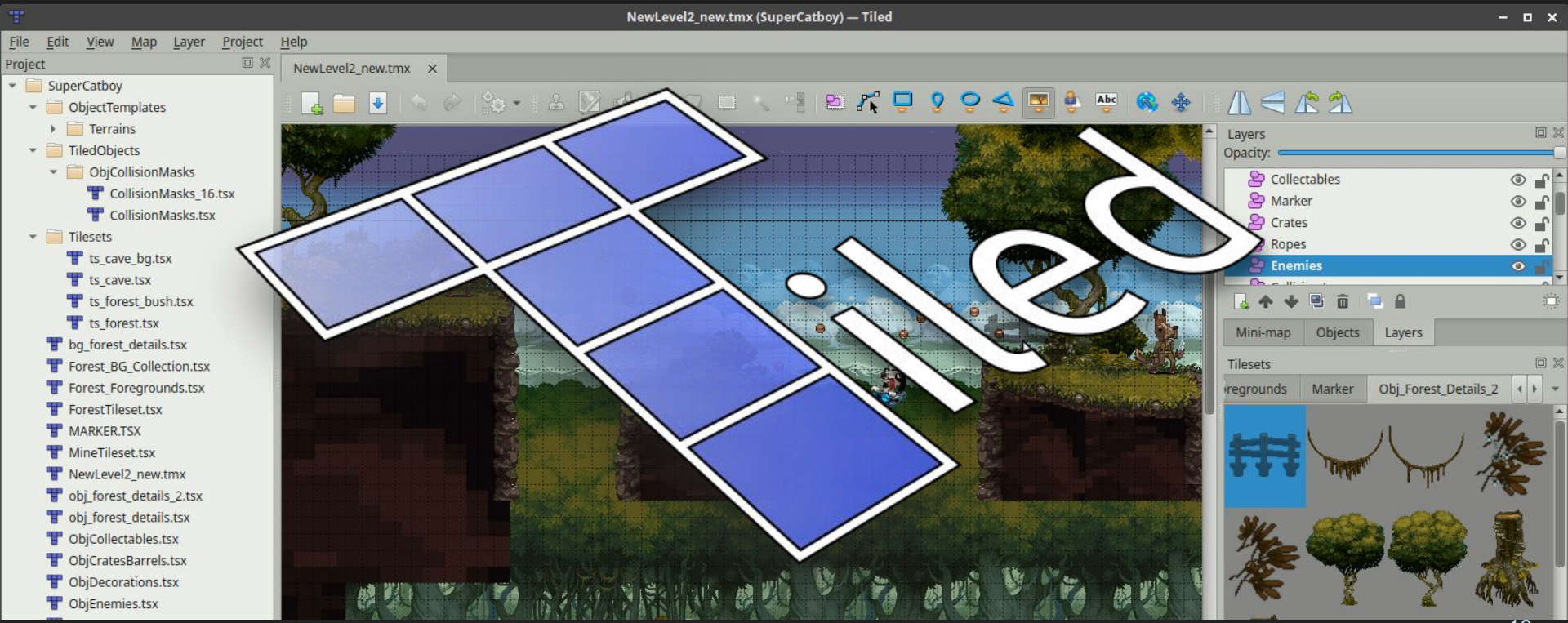
Receipts:
https://github.com/photonstorm/phaser/blame/5c8ecbcf999e6f328d21884e877c9e5935d2d350/src/tilemaps/Tilemap.js

Nathan's Mappy example project

- Several ways of working with tilemap data in Phaser!
- Let's see how we can hard-code data into our scenes.
- Then let's see how to avoid hard-coding it.

Tiled (a multipurpose game map editor)

https://www.mapeditor.org/



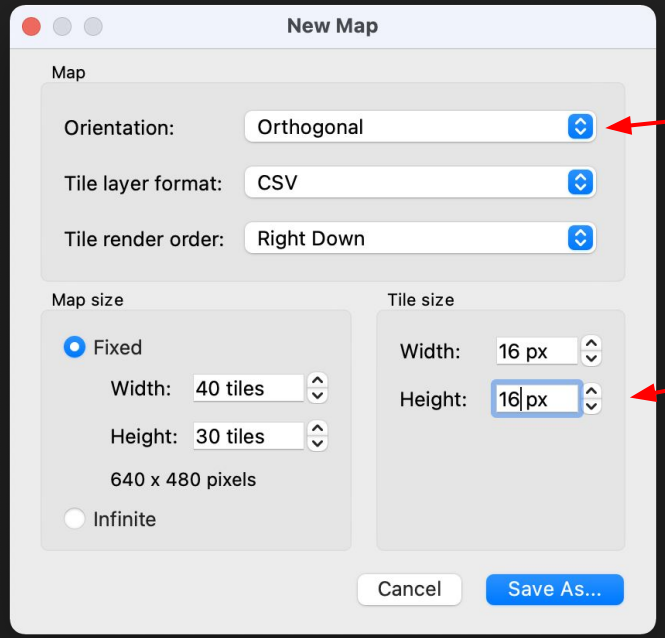
But first, we need a tileset

Here's one: https://www.kenney.nl/assets/bit-pack

Options:

- Find one already licensed for reuse while fitting your style (good luck!)
- Make your own by painting over someone else's (💪💪💪)

Creating a blank tilemap (.tmx file)

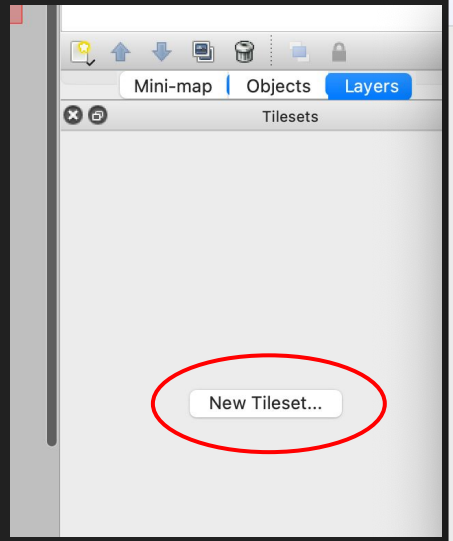


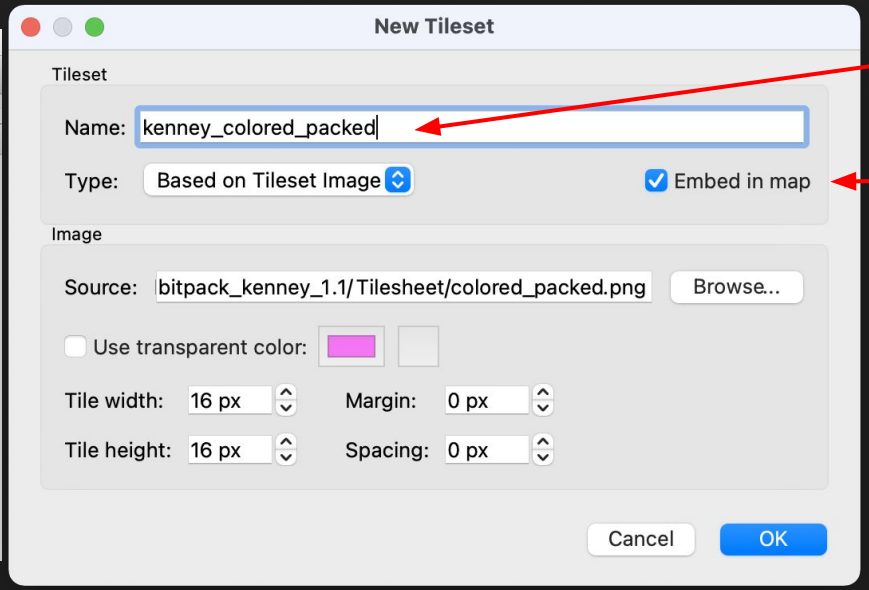
Orthogonal means grid of squares (Isometric and Hexagonal alternatives available)

Coordinate tile sizes with asset creators (powers of 2 are common)

You can control the on-screen size of tiles separately in code.

Importing a tileset to your tilemap





This name needs to match one of your assets in Phaser

Check this box!

Painting tools

To show interactively:

- Place one tile
- Fill with one tile
- Stamp several tiles together
- Fill with randomized selection
- ...

File Formats

Phaser

.png

Tilemap image

.json

Map data

Tiled

.tmx

Tiled map file

.tsx

Tiled tileset file

File formats

.tmx: Tiled's native tilename format, open this in Tiled

.json: JavaScript's native data format, load this in your Phaser code

Use the "File > Export As ..." menu to get your JSON file.

Loading Tiled tilemaps in Phaser

`preload():`

1. Load the Tiled JSON file **and** tilesheet (matching names)

`create():`

2. Make a Tilemap object (from JSON)
3. Attach a tileset image (from image) to the tilemap object
4. Create one or more tilemap layers
5. Optionally set your layers to use collision
6. Optionally add object--map physics colliders

`update():`

7. Optionally check object--map collisions

⚠️ COLLISION WEIRDNESS ⚠️

If you are having collision problems between objects (e.g., your player and map tiles), be sure you're using **physics** to propel objects rather than direct x-/y-coordinate control.

For some reason, `body.touching` does not work for sprite/tilemap collisions, but **`body.blocked`** does. *I do not know why.*

If you're having collision tunneling issues because your objects move at high speeds, increase the `TILE_BIAS` to a value higher than the default 16 px.

Tilemaps in Depth

Essence of tilemaps: a 2D array of integers

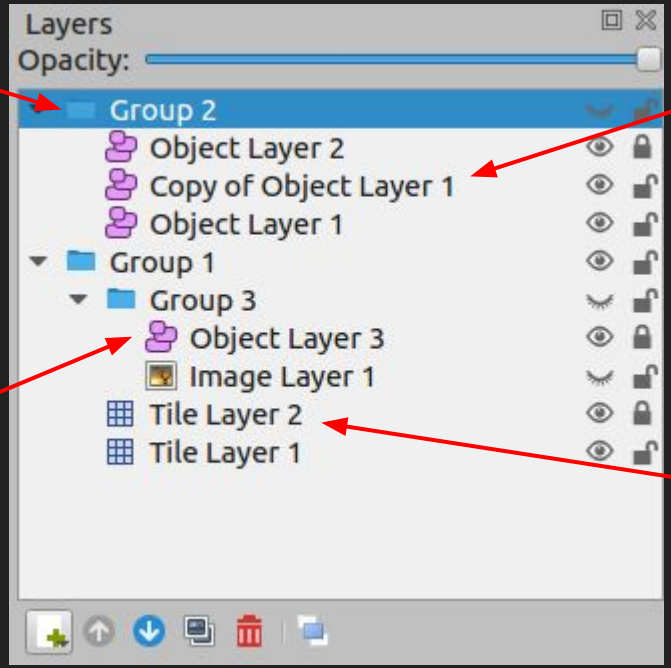
Where can you get them:

- Hand-type them into your source code
  - Demo: revisit example from Mappy project seen in previous lecture
- Edit them in a dedicated grid editor (like Tiled or a spreadsheet program??)
  - Demo: Let's design our level together in Google Sheets! (on Adam's laptop only)
- Generate them procedurally with other code
  - Demo: https://observablehq.com/@makio135/zelda-wfc

Beyond grids of integers in Tiled

Groups for organization

Background images for reference



Freely-movable and uniquely configurable game object descriptions

Grids of integers (and associated properties encodable as integers)

Tiled is cool, but have you tried LDtk?

https://ldtk.io/

New tool, super polished, pretty sophisticated.

Note “auto-rendering” feature (we have an assignment in CMPM 147 dedicated to implementing this rather tedious feature)

Can I use tile-based map editors in games without tiles?

Suppose you were making a location-based game (like *Pokemon Go* but specific to the UCSC campus). You would need a way to place special points, special encounter regions, forbidden zones. You'd want to specify these in reference to satellite maps (rather than hand-typing latitude/longitude values).

Let's make a map like this in Tiled!

Use: points, polygons/polylines, tiles with rotation/scale.

What are tilemaps in Phaser 3 specifically?

This page is your official reference material for everything you can do with a Tilemap:

https://photonstorm.github.io/phaser3-docs/Phaser.Tilemaps.Tilemap.html

Some useful methods:

- **addTilesetImage**: connect the tilemap to an image that you've loaded.
- **createLayer**: create a player-visible image from the tilemap
- **getTileAt** or **getTileAtWorldXY**: find on which tile is at a given position (by tile index or world position)
- **setCollision...**: mark a subset of tile locations as collidable
- **createFromObjects**: (maybe should be read as "create objects from"): create independent game objects from special tiles
- **getObjectLayer**: access non-tile object layers from a Tiled map file

Revisiting Mappy in detail

Questions for live exploration:

- How do we know where to place the player initially?
- How do we specify which tiles you can stand on?
- How do we bring certain tiles to life with interactive game objects (coins)?
- How do we get different layers to scroll by different amounts?
- How can we find out of the player is on a special square (e.g. a jump-boosting zone)?

The Threshold Problem

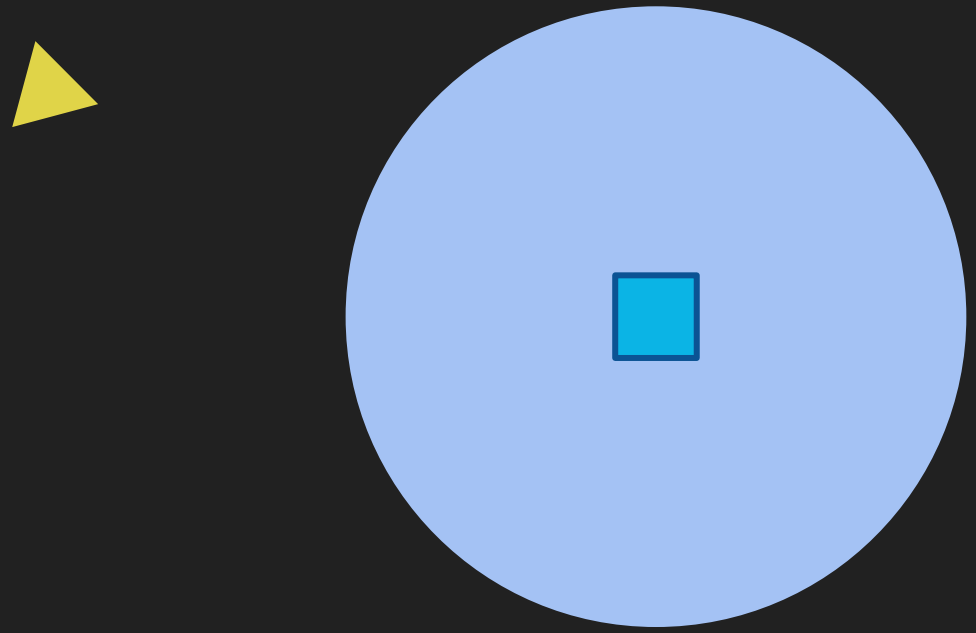
Reverse toggles too quickly

This kind of threshold problem is one that you'll encounter a lot

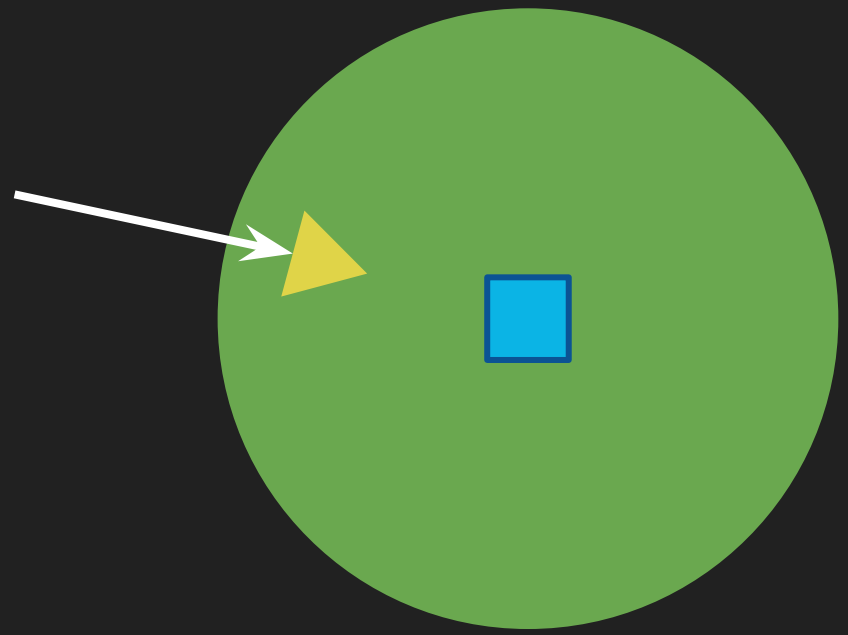
```
// reverse horizontal by pressing the 'R' key
if(game.input.keyboard.isDown(Phaser.Keyboard.R)) {
    this.body.velocity.x = -this.body.velocity.x;
}
```

problem

Threshold Problem

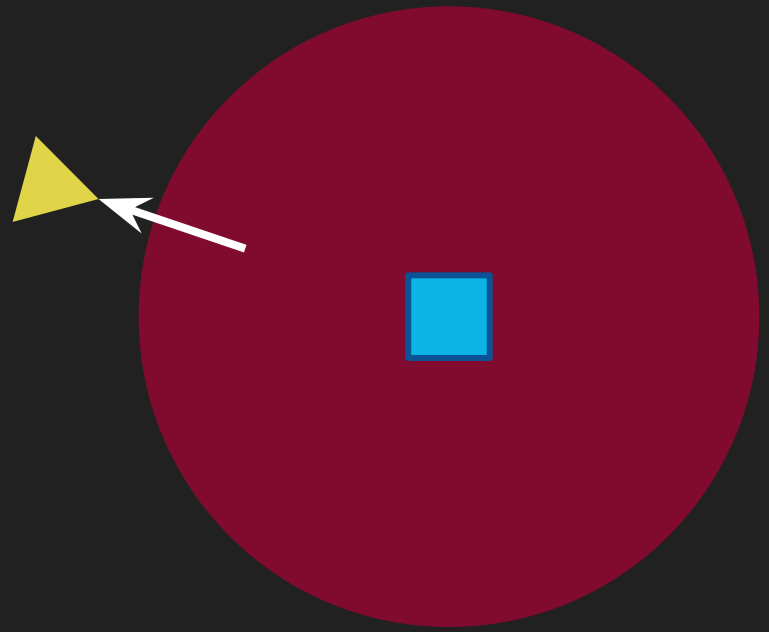


Threshold Problem



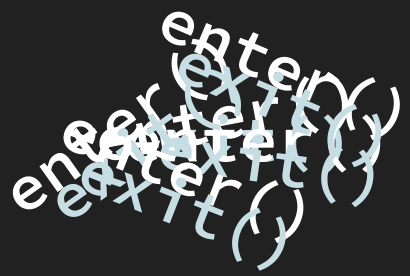
```
enter()
```

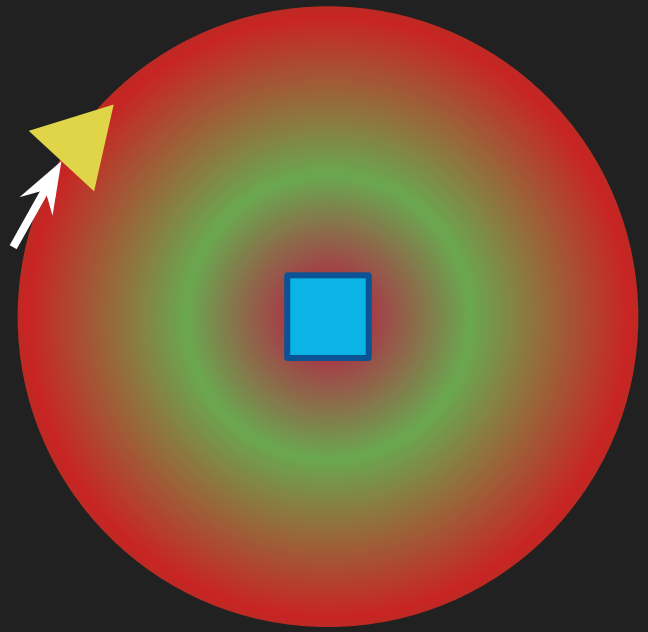
Threshold Problem



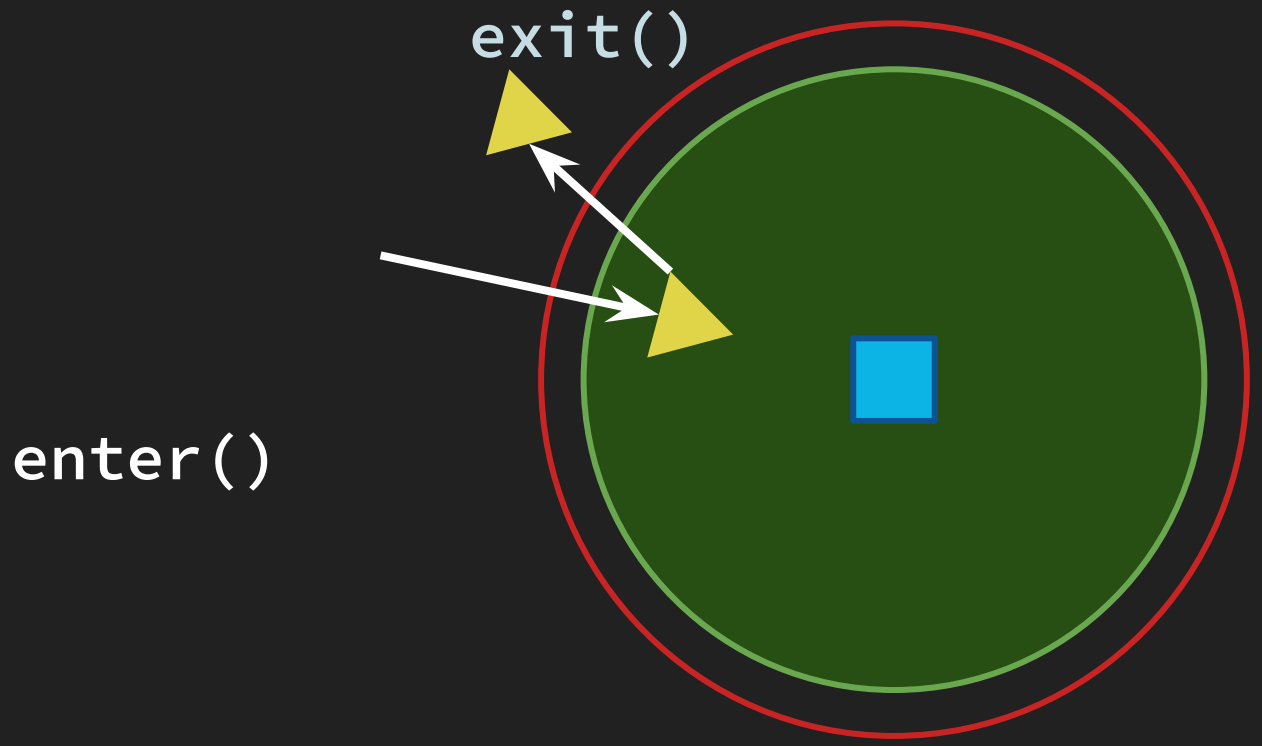
```
exit()
```

Race condition!





Add a threshold



Reverse toggles too quickly

This kind of threshold problem is one that you'll encounter a lot

```
// reverse horizontal by pressing the 'R' key
if(game.input.keyboard.isDown(Phaser.Keyboard.R)) {
    this.body.velocity.x = -this.body.velocity.x;
}
```

problem

```
    if(this.reverseKey.justPressed() ){
        this.body.velocity.x = this.body.velocity.x * -1;
    }
```

or

```
// reverses the horizonal velocity
if (game.input.keyboard.isDown(Phaser.KeyCode.R) && this.canPressR) {
    this.xVelocity = -this.xVelocity;
    this.canPressR = false;
}
if (!game.input.keyboard.isDown(Phaser.KeyCode.R) && !this.canPressR) {
    this.canPressR = true;
}
```

More Debugging Tips

Useful random debugging advice

1. When you find a problem, change something so that same problem can't happen again
   a. assert()
   b. Keep a debugging notebook
2. Make debug tools
   a. Quicker feedback is better
   b. Display values live if possible
3. Only make one change at a time and then test it
4. Just because you paused the game doesn't mean it's paused
   a. And stopping one update doesn't mean you stopped all of them
5. console.log() is slow
   a. Faster to print an array as a string than to individually print the contents

Useful random debugging advice

Walk through your code step by step, explaining to yourself what is supposed to happen

Useful random debugging advice

1. When you find a problem, change something so that same problem can't happen again
   a. assert()
   b. Keep a debugging notebook
2. Make debug tools
   a. Quicker feedback is better
   b. Display values live if possible
3. Only make one change at a time and then test it
4. Just because you paused the game doesn't mean it's paused
   a. And stopping one update doesn't mean you stopped all of them
5. console.log() is slow
   a. Faster to print an array as a string than to individually print the contents

Useful random debugging advice

1. When you find a problem, change something so that same problem can't happen again
   a. assert()
   b. Keep a debugging notebook
2. Make debug tools
   a. Quicker feedback is better
   b. Display values live if possible
3. Only make one change at a time and then test it
4. Just because you paused the game doesn't mean it's paused
   a. And stopping one update doesn't mean you stopped all of them
5. console.log() is slow
   a. Faster to print an array as a string than to individually print the contents

Useful random debugging advice

1. When you find a problem, change something so that same problem can't happen again
   a. assert()
   b. Keep a debugging notebook
2. Make debug tools
   a. Quicker feedback is better
   b. Display values live if possible
3. Only make one change at a time and then test it
4. Just because you paused the game doesn't mean it's paused
   a. And stopping one update doesn't mean you stopped all of them
5. console.log() is slow
   a. Faster to print an array as a string than to individually print the contents

Useful random debugging advice

1. When you find a problem, change something so that same problem can't happen again
   a. assert()
   b. Keep a debugging notebook
2. Make debug tools
   a. Quicker feedback is better
   b. Display values live if possible
3. Only make one change at a time and then test it
4. Just because you paused the game doesn't mean it's paused
   a. And stopping one update doesn't mean you stopped all of them
5. console.log() is slow
   a. Faster to print an array as a string than to individually print the contents

AABB characters and slopes

An example of a real-world physics-and-debugging problem in a game with 2D physics like yours

https://twitter.com/eevee/status/1133248372624613376

Exit Slip

https://forms.gle/rZJBZTRrgAZUhJnWz

Bonus Slides

Alternatives

https://www.patreon.com/posts/adobe-software-26834357

If you don't have a subscription to Adobe's software (Photoshop, etc.) then it's helpful to know about all of the free and inexpensive alternatives.