CMPM 120

Your Final Project

Objectives

By the end of today you should be able to...

1. Recap
   a. Understand the citation requirements for your projects
2. Code review
   a. Explain what code review is
3. Your Final Project
   a. Understand the requirements for the prototype
4. Camera
   a. Demonstrate how to manipulate the camera with Phaser
   b. Practice using git

Citation

Remember to cite your sources!

If you copied or adapted it from somewhere else, you should cite it!

Don't simply reuse code! You can incorporate the ideas of others but the game should ultimately be your own approach. *It's fine to borrow a few functions or a library but not the entire game loop.*

Respect the License! Open Source has rules.

One guideline to citing code: https://integrity.mit.edu/handbook/writing-code
(It doesn't entirely apply, but it's close enough to start.)

For The Final Project

The majority of the code and art should be made by your team

Some outside assets are allowed (e.g. using a piece of music with an appropriately permissive license) but the overall look and feel of the game should be yours.

Game Jam Rules

For fun, the rules for Ludum Dare:

http://ludumdare.com/compo/rules/

(has some neat examples explaining derivative work)

And the rules for the Global Game Jam:

https://globalgamejam.org/news/be-cool-rules-global-game-jam-2019

Revising Past Assignments

Since the point of the exercises is to measure your understanding of the material, I will allow you to submit a revision of your past assignment as long as:

- ➔ Your updated submission demonstrates your understanding of the material
  - ◆ Include lots of comments, explaining why you chose to implement your solution in that way
  - ◆ If I can't understand why you made your decisions, you don't get the points
- ➔ Late penalties still apply, but from the point of your original turn-in
  - ◆ I want to encourage you to turn stuff in on time
  - ◆ Turning stuff in late makes extra work for both of us
- ➔ Revision grading will happen at a time of my discretion
- ➔ No revisions will be accepted past the end of August 23rd
- ➔ **Does not apply to the final project:** the final project milestones are **hard deadlines**

Mandatory

Code Review

Code review is...

- ➔ A process of improving the quality of your code by systemically having other people look at it.
- ➔ A common industry practice.
- ➔ One of the most effective methods of quality assurance.
- ➔ Required as part of your final projects.
    - ◆ You must show what you are working on either during office hours or in the tutoring session
    - ◆ Counts towards your final project grade
- ➔ I recommend also doing code reviews within your team

Final Project Code Review

1. Go to office hours (or the tutoring session).
2. Show the code, art asset, or other production task you have been working on.
3. Do this at least once before the end of **August 23rd**.

This does not have to be done in person: online office hours or scheduling another time is possible

Remember: the small group tutoring must be registered at least 24 hours in advance!

Code review checklist

What is the code doing?

Is there unused code here?

Is there code duplication?

Do the comments explain **why** the code does what it does?

Is the code understandable? (style, variable names, structure, no magic numbers, etc.)

Has the code been tested?

Working on a game is

More than just programming

Final Project: Prototype

First Prototype

Due August 12th **You must turn something in by 11:59 PM.**

(If you have accommodations and need to turn it in later, send it to me by email.)

Organization (2 points)

→ Logically organized code (0.5)
→ Clearly commented code (0.5)
→ Runs from localhost w/ no major programming errors (game bugs are OK) (0.5)
→ Include a comment in the header of your source with your team name and all team members (0.5)

First Prototype

Structure (5 points)

- ➔ A "core loop," i.e., you have defined states (yes, plural) and some means to switch between them (1)
- ➔ Player interaction. - The player can interact with the game. You should have at least one primary mechanic operational. If you make a platformer, perhaps movement and jumping are implemented. If you make a narrative game, perhaps the dialog boxes are implemented. If you're making a hidden object game, perhaps the basic point/click verbs are implemented. It all depends on your game. (1)

First Prototype

Structure (6 points) (continued)

- ➔ Temporary graphical assets. What good are core loops and interactions if there is nothing to look at? These should be assets *you* create, even if they are geometric primitives, quick doodles, or magazine cutouts. (1)
- ➔ Temporary sound assets. Don't leave decisions about sound until the end of the design process. Sound is integral to making a game feel "real." Make sure your game makes some noise. (1)
- ➔ **Version control.** You must have your git workflow established. GitHub is the standard we are using. Make sure you have a repository setup. **Add a comment in your source header with a link to your GitHub repository.** (2)

Think of the prototype like...

Presenting your game to your publisher

Postmortem for your

Endless Runner

Topics in the second half of the class

Potential Topics

- Cameras
- Particles
- P2 Physics
- Time & Timers
- Advanced Git
- State Machines
- Text and Fonts
- Animation and Tweens
- CSS (& other web dev stuff)
- Audio
- Scaling
- ...something you want to know about!

Cameras

Download the repository

```
git clone http://github.com/ikarth/camera
```

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