(Extreme) Game Development with Javascript

Let's start with the basics

? gamedev... anyone? ?

Game loop

Game loop #2

Game loop #3

Time management

We need a game engine

Let's use Phaser!

Why Phaser?

- Easy to learn
- Open source
- It's Javascript
- It's fun!

Basic concepts

- Scenes
- Game Objects & Factory
- Loader
- Physics

Scenes

A scene is a container for all the game objects. It can be a menu, a level, a game over screen, etc.

Example:

Super Mario Bros has a scene for the title screen, a scene for each level, and a scene for the game over screen.

Scenes

Scenes registration

Game Object

A game object is an entity that can be placed in a scene. It can be a sprite, a text, a group, an audio, etc.

They are implemented as classes inheriting from `Phaser.GameObject` and can be created using the **factory**.

Factory

Inside a scene, you can create game objects using the factory, invoking `this.add`.

Loader

To be able to use assets in your game, you need to load them first. You can do this in the `preload` method of a scene.

Inputs #1

Inputs #2

Physics

Arcade vs MatterJS

Arcade Physics

Collisions

Workshop time!

Let's make PONG!

PONG Rules

- Two players (WASD and Arrow keys)
- Each player has a paddle
- A ball bounces between the paddles
- If the ball hits the wall behind a player, the other player scores
- First player to reach 10 points wins

Useful links

- This Repo
  https://github.com/ProGM/phaser-xp
- Phaser Cheatsheet
  https://progm.github.io/phaser-xp/cheatsheet/
- Phaser Debugger
  https://chromewebstore.google.com/detail/phaser-debugger/aigiefhkiaiihlploginlonehdafjljd
- Phaser Examples
  https://labs.phaser.io/

PONG Rules

- Two players (WASD and Arrow keys)
- Each player has a paddle
- A ball bounces between the paddles
- If the ball hits the wall behind a player, the other player scores
- First player to reach 10 points wins

XP and Video Game Industry...

A sad story.

Cultural problems

- Competition & management pressing for results
- Non-technical people involved
- (Historically) No interest for long-time maintenance

Technical problems

- A lot of proprietary / licensed / obscure software
- No standardization (lack of tooling)
- Hard to test player experience
- You're testing the engine, not the game

But... it's possible!

Separation of concerns

- Game state
- Presentation
- Input handling

Let's test Game State

Other approaches

- MVC (Model-View-Controller)
- MVP (Model-View-Presenter)
- ECS (Entity-Component-System)