Programming in Phaser

TypeScript

JavaScript, but with static typing.

https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes-oop.html

IDE

- Visual Studio Code
- …?

Setup

- Install **node**: https://nodejs.org/en/download (this will also install the **npm** package manager)

- Install **ionic** CLI:

```
npm install -g @ionic/cli
```

Getting started

Create a project

- Create a project using **Angular** framework, **blank** template and **standalone** components:

```
ionic start
```

- Go into created directory

- Install **phaser**

```
npm install phaser
```

Tweak build settings

- Edit **tsconfig.json**
- Add **scriphost** to **lib**
- Set **allowSyntheticDefaultImports** to **true**

```
"compilerOptions": {
    …
    "lib": [
        …
        "scripthost"
    ],
    "allowSyntheticDefaultImports": true
}
```

Component

- Generate Angular component:

```
ionic generate component game
```

game.component.html

```
<div id="phaser-game-parent"></div>
```

game.component.ts

```
import { Component, OnInit } from '@angular/core';
import Phaser from 'phaser';

@Component({
  selector: 'app-game',
  templateUrl: './game.component.html',
  styleUrls: ['./game.component.scss'],
  standalone: true
})
export class GameComponent  implements OnInit {
  game: Phaser.Game | undefined;
  config: Phaser.Types.Core.GameConfig;

  constructor() {
      this.config = {};
  }

  ngOnInit() {
      this.game = new Phaser.Game(this.config);
  }
}
```

Config

```
this.config = {
  type: Phaser.AUTO,
  scale: {
    parent: 'phaser-game-parent',
    width: '100%',
    height: '100%',
  },
  backgroundColor: '#2d2d2d',
  parent: 'phaser-game-parent',
    physics: {
    default: 'arcade',
    arcade: {
      debug: window.location.hostname.includes('localhost')
    }
  },
  scene: []
};
```

home.page.html

```
<ion-content [fullscreen]="true">
  <div id="container">
    <app-game></app-game>
  </div>
</ion-content>
```

home.page.ts

```
import { Component } from '@angular/core';
import { IonContent } from '@ionic/angular/standalone';
import { GameComponent } from '../test-game/game.component';

@Component({
  selector: 'app-home',
  templateUrl: 'home.page.html',
  styleUrls: ['home.page.scss'],
  standalone: true,
  imports: [IonContent, GameComponent],
})
export class HomePage {
  constructor() {}
}
```

Or… use a template!

https://github.com/blindflugstudios/ionic-phaser-starter

Running and building

- Run the app locally (with auto-reload):

```
ionic serve
```

- Create a distributable build:

```
ionic build
```

Let’s create something

Define a box

```
export default class Box extends Phaser.Physics.Arcade.Sprite {
  constructor(scene: Phaser.Scene, x: any, y: any) {
        super(scene, x, y, 'box');
        this.scene.add.existing(this);
        this.scene.physics.add.existing(this);
        this.body!.setSize(50, 50);
  }
}
```

Draw a box!

```
import Box from "../box";
…
  box!: Box;
…
  preload() {
        this.load.image('box', 'assets/game/images/box.png');
  }
…
  create() {
        this.box = new Box(this, 200, 200);
  }
```

Move the box!

```
  keyLeft: Phaser.Input.Keyboard.Key | undefined;
  keyRight: Phaser.Input.Keyboard.Key | undefined;
…
  init() {
        this.keyLeft = this.input.keyboard?.addKey(Phaser.Input.Keyboard.KeyCodes.A);
        this.keyRight = this.input.keyboard?.addKey(Phaser.Input.Keyboard.KeyCodes.D);
  }

  override update(time: number, delta: number) {
    if (this.keyLeft?.isDown) {
      this.box.x -= delta * 2;
    } else if (this.keyRight?.isDown) {
      this.box.x += delta * 2;
    }
  }
```

Collide the box!

```
  create() {
    const obstacles = this.add.group({ runChildUpdate: true });
    obstacles.add(new Box(this, 400, 200));

    this.box = new Box(this, 200, 200);

    this.physics.world.addOverlap(this.box, obstacles, (a, b) =>
console.log('BONK!'));
  }
```

Timers

Timer

```
const timer = this.scene.time.addEvent({
  delay: 10000,
  callback: () => console.log('Hey!'),
  loop: true,
})
```

Delayed call

```
this.scene.time.delayedCall(2000, () => 'Hey from the future!');
```

UI

Define a button

```
export default class Button extends Phaser.GameObjects.Text {
  constructor(scene: Phaser.Scene, x: any, y: any, text: string, callback: Function, fontSize = 16, width = 0) {
    super(scene, x, y, text, {
      backgroundColor: '#000000',
      padding: { x: 20, y: 20 },
      align: 'center',
      fontSize: fontSize,
      fixedWidth: width,
    });

    this.alpha = 0.1;
    this.scene.add.existing(this);
    this.setInteractive({ cursor: 'pointer' });

    this.on('pointerdown', (pointer: PointerEvent, localX: number, localY: number, event: InputEvent) => {
      callback();
      event.stopPropagation();
    });
  }
}
```

Create a button

```
  create() {
    const button = new Button(this, 400, 400, 'Click me', () => console.log('Hello
there'));
  }
```

Audio

Play sound on key press

```
  keyShoot: Phaser.Input.Keyboard.Key | undefined;
…
  init() {
    this.keyShoot = this.input.keyboard?.addKey(Phaser.Input.Keyboard.KeyCodes.SPACE);
  }


  preload() {
    this.load.audio('shoot', 'assets/game/audio/shoot.wav');
  }

  create() {
    const shootSound = this.sound.add('shoot');
    this.keyShoot?.on('down', () => shootSound.play());
  }
```

Where to look further?

- Docs: https://newdocs.phaser.io/
- Examples: https://labs.phaser.io/
- Godzilla! https://github.com/blindflugstudios/Swisscom-Godzilla

Questions?