CS55.11 JavaScript

Spring 2017 ~ Ethan Wilde

Week 12

SANTA ROSA JUNIOR COLLEGE

Phaser.js

Game Development Library

Desktop and Mobile HTML5 game framework

A fast, free and fun open source framework for Canvas and WebGL powered browser games.

https://phaser.io/

Reading This Week

ONLINE

- **Making Your First Phaser Game:** https://phaser.io/tutorials/making-your-first-phaser-game
- **ANY other tutorial of your choice from those available at:** https://phaser.io/learn/official-tutorials

Working with Phaser
Objects

1. **JavaScript Objects**

```
var car = {
    type:"Fiat",
    model:"500",
    color:"white"
};
```

Working with Phaser
Objects

2. **Object Properties**

```
var car = {
    type:"Fiat",  ←  name and value pair
    model:"500",
    color:"white"
};
```

Object properties are name-value pairs.

Working with D3
Objects

3. **Object Methods**

```
var car = {
    type:"Fiat",
    model:"500",
    color:”white",
    getName:function() {
        return this.color + ' ' + this.type + ' ' + this.model;
    }
};
```

objects can contain methods (anonymous functions)

JavaScript this object refers to current instance of object.

Working with D3
Game Object

Phaser.Game() method returns a game object

4. **The Phaser game object**

```
var game = new Phaser.Game(
    800, 600, Phaser.AUTO, '', {
        preload: game_preload,
        create: game_create,
        update: game_update
    }
);
```

Working with D3
Required Methods for Game

5. **Required methods for Phaser game object**

```
var game = new Phaser.Game(
    800, 600, Phaser.AUTO, '', {
        preload: game_preload,  ←——  preload
        create: game_create,         create
        update: game_update          update
    }
);
```

Preload: run before game initialized
Create: run to initialize game
Update: runs repeatedly every 1/60 sec.

Working with D3
Game World + Physics

6. The game object includes many properties that provide access to library features like physics and world environment properties.

```
game.physics.startSystem(
  Phaser.Physics.ARCADE);
```

Working with D3
Assets + Sprites

7. Individual media elements - images, sounds, etc. are stored in external files known as assets. Live game play objects are known as sprites and appear in the game world.

```
game.load.image('sky', ‘assets/sky.png’);
game.add.sprite(0, 0, 'sky');
```

Working with D3
Sprite Physics

8. Sprite are given physics properties including gravity and bounce via their *body* property. Collisions and overlap between sprites can be detected and code executed in response.

```
var player = game.add.sprite(
    32, game.world.height - 150, 'dude');
game.physics.arcade.enable(player);
player.body.bounce.y = 0.2;
player.body.gravity.y = 300;
player.body.collideWorldBounds = true;

var hitPlatform = game.physics.arcade.collide(
    player, platforms);
```

Working with D3
Groups

9. Sprites can be associated in a collection known as a group. Groups allow easy manipulation and testing of sprite interactions like collisions.

```
platforms = game.add.group();
var ledge = platforms.create(400, 400, 'ground');
ledge.body.immovable = true;
```

Working with D3
Input

10. **The *game.input* property provides access to user input detection, event triggering and code execution in response.**

```
cursors = game.input.keyboard.createCursorKeys();
if (cursors.left.isDown) {
    player.body.velocity.x = -150;
    player.animations.play('left');
}
```

Code Examples

“A First Game: Flappy Bird”

A tour through a basic game example.

Code Examples

“A more complex game”

A basic example showing groups with physics including collision detection and scoring.