Week_24

Trinity - Introduction to Computer Programming:
Game Development…

Today's Class is has 2 goals:
(1) Explore the phaser3-example.html
** Powerful Var's
** Particle Emitter

```
var particles = this.add.particles('red');
```

** Learn about debugging.

(2) Begin a new game from scratch…

Debugging, what is it?
How can debugging help?

How can debugging help?

Debugging is the process of finding and resolving defects or problems within a computer program that prevent correct operation of computer software or a system.

Debugging tactics can involve interactive debugging, control flow analysis, unit testing, integration testing, log file analysis, monitoring at the application or system level, memory dumps, and profiling.

Sometime the code is running right, but there problem with the supplementary data (assets

Phaser3-Example.html

e is a
s).

(open this up in Brackets).
We will attempt to

Run this file with Firefox:

What do you observe?

Why is the "emitter" only on the top left of the page?

Answer (hint), a comment in JavaScript is //

Answer (hint), a comment in JavaScript is "//" (slash slash) - this is often used for testing or changing configurations, or giving information to yourself or other programmers…

Next, What is wrong with this example…

(1) _____
(2) _____
(3) _____
(4) _____

Debug - the program…

** Hint - look at the .\assets (dir)

The config object is how you configure your Phaser Game. There are lots of options that can be placed in this object and as you expand on your Phaser knowledge you'll encounter more of them. But we're just going to set the renderer, dimensions and a default Scene.

An instance of a Phaser.Game object is assigned to a local variable called game and the configuration object is passed to it. This will start the process of bringing Phaser to life.

In Phaser 2 the game object acted as the gateway to nearly all internal systems and was often accessed from a global variable. In Phaser 3 this is no longer the case and it's no longer useful to store the game

instance in a global variable.

The type property can be either Phaser.CANVAS, Phaser.WEBGL, or Phaser.AUTO. This is the rendering context that you want to use for your game. The recommended value is Phaser.AUTO which automatically tries to use WebGL, but if the browser or device doesn't support it it'll fall back to Canvas. The canvas element that Phaser creates will be simply be appended to the document at the point the script was called, but you can also specify a parent container in the game config should you wish.

The width and height properties set the size of the canvas element that Phaser will create. In this case 800 x 600 pixels. Your game world can be any size you like, but this is the resolution the game will display in.

The scene property of the configuration object will be covered in more detail further in next class.

Open Game_.html

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8" />
    <title> (Your Nmae)  Making A Phaser Game </title>
    <script src="phaser.min.js"></script>
    <style type="text/css">
        body {
            margin: 0;
        }
    </style>
</head>
<body>

<script type="text/javascript">

    var config = {
        type: Phaser.AUTO,
        width: 800,
        height: 600,
        scene: {
            preload: preload,
            create: create,
            update: update
        }
    };

    var game = new Phaser.Game(config);

    function preload ()
    {
```

```
    {
    }

    function create ()
    {
    }

    function update ()
    {
    }

</script>

</body>
</html>
```