Phaser HTML5 Game Tutorial: Build A Pong Game 1: Project Setup

June 12, 2015 By 2 Comments

On November 29, 1972, Atari launched it's first game called Pong. By today's standards it was a simple two dimensional game that simulates table tennis.

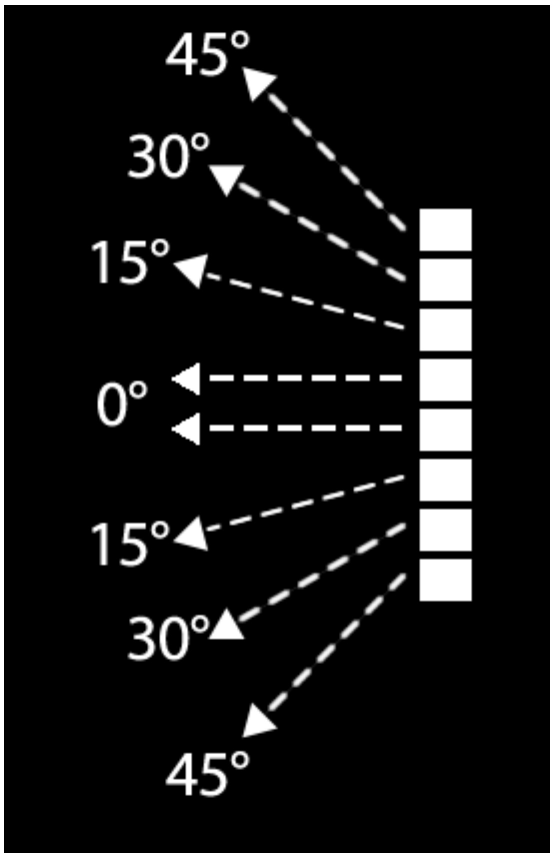
Designed by Allan Alcorn, Pong was originally a two-player game where one player controlled a paddle on the left and the other player controlled another paddle on the right. Players could only move their paddles vertically to hit a ball back and forth. The goal is to score eleven points before the opponent does and points were awarded to the opponent when a player failed to return the ball.

So here's what we are going to make:

Here's a breakdown of what's happening:

1. Before a match begins, we see the ball bouncing around the screen with the scores of the previous match still being shown.
2. When the match begins, scores for both players are reset to zero and the ball appears somewhere randomly along the middle dividing line then moves towards one of the players.
3. Notice the angle at which the ball travels when it appears. It is approximately at a 60-degree angle.
4. Both players move their paddles vertically to hit the ball and return it to the opponent.
5. When a player fails to return the ball, a point is awarded to the opponent. The winner is the player that reaches a score of 11 points.

A few additional things that are not immediately obvious:



1. In the illustration above, the paddle is made up of eight segments. Hitting the centre two segments returns the ball at a straight 90-degree angle from the paddle. Hitting the outer edges of the paddle returns the ball at a 45-degree angle. The other segments return the ball at a 15-degree and 30-degree angle such as shown in the above.
2. After hitting the ball a few times, it would actually increase its movement speed. Missing the ball would reset its speed.
3. The paddles cannot reach the very top of the screen due to a defect in the original circuit hardware. It was eventually left as it is and became one of the features that good players use to score points.

For this tutorial, we'll be using the Phaser HTML5 game framework. As I'm writing this article, the current version is 2.3.0.

Project files

Here's an empty template for you to download[1] for use in this tutorial. You will find the following files in the zip file:

`index.html`, the game will be displayed here

`/js/phaser.min.js`, version 2.3.0 of the Phaser framework

`/js/game.js`, the file where all our codes will go

`/assets/graphics/ball.png`

`/assets/graphics/paddle.png`

`/assets/sounds/ballHit.m4a`

`/assets/sounds/ballHit.ogg`

`/assets/sounds/ballBounce.m4a`

`/assets/sounds/ballBounce.ogg`

`/assets/sounds/ballMiss.m4a`

`/assets/sounds/ballMiss.ogg`

You may be wondering why there are 2 different file formats for each sound. At this time, there is no standard audio format across all browsers. Internet Explorer and Safari use the MPEG-4 Advanced Audio Coding (.m4a) format while other browsers such as Firefox, Chrome and Opera support the Ogg Vorbis format (.ogg).

Also, if you do not already have a code editor, I would suggest downloading Brackets here[2]. Brackets has a live preview feature which will enable you to test your game without installing a web server. There are also other code editors such as Sublime Text, Notepad++ and Net Beans that you can also use to make HTML5 games.

Project setup

We'll start by looking at the index.html file. This file will contain and display your HTML5 game in your browser. Here is the code you will need to add:

```
<!doctype html>

<html>

<head>

<title></title>
```

```
<meta charset"UTF-8"

<meta "viewport"content"width=device-width, initial-scale=1.0"

    <script "js/phaser.min.js"</script>

    <script "js/game.js"</script>

</head>

<body>

<div "gameDiv"</div>

</body>

</html>
```

Line 7 loads the Phaser game framework:

```
<script "js/phaser.min.js"</script>
```

Line 8 loads our game code:

```
<script "js/game.js"</script>
```

Line 11 is where our game will be displayed:

```
<div "gameDiv"</div>
```

The `<div>` tag defines a division or a section in an HTML document. An element `id` called `gameDiv` is given to identify where to insert the canvas element that Phaser creates.

Next, open up the `game.js` file where we will create an empty Phaser project. Enter the code below.

```
gameProperties

screenWidth

screenHeight

mainStatefunction

mainStateprototype

preloadfunction

createfunction

updatefunction

PhasergamePropertiesscreenWidthgamePropertiesscreenHeightPhaser'gameDiv'

state'main'mainState

statestart'main'
```

At line 1, we declare our `gameProperties` object that contains two variables: the HTML5 canvas width and height in pixels.

```
gameProperties

screenWidth

screenHeight
```

At line 6, we declare our `mainState` object which will contain all our code and logic necessary to make our game work.

```
mainStatefunction

mainStateprototype

preloadfunction

createfunction

updatefunction
```

The `prototype` keyword at line 7 is used to extend or add on an existing object. Notice there are three functions within the `mainState` object: The `preload`, `create` and `update` functions.

- The `preload` function (line 8) is used to load all of our game assets that could include images, sounds, level data, etc.
- The `create` function (line 12) is the first function to run after all our game assets have been loaded.
- The `update` function (line 16) is our game loop. As the Phaser framework runs at 60 frames per second, any code within the update function is repeatedly run 60 times each second.

Finally, we create a new Phaser game object. At line 21, we create a new `Phaser.Game` object that is assigned to the game variable.

```
PhasergamePropertiesscreenWidthgamePropertiesscreenHeightPhaser'gameDiv'
```

Notice the game object requires four arguments.

- The first two arguments are the width and the height of the canvas element. In this case 640 x 480 pixels. You can resize this in the `gameProperties` object above.
- The third argument is the renderer that will be used. Here, `Phaser.AUTO` is used to automatically detect whether to use the WebGL or Canvas renderer.
- The fourth argument is the element id we will use, in this case it is `gameDiv`, which is the element id we used above in the index.html file where the canvas element is inserted.

After a new `Phaser.Game` object is created, we need to assign game states to it. We do this at line 22.

```
state'main'mainState
```

Adding a state to the game object requires two arguments.

- The first argument is the state name that will is used when switching between states.
- The second argument is the declared object that will used when a state name is called.

The last step is to load our `mainState` at line 23.

```
statestart'main'
```

There is only one argument required when choosing which state to run. In this case, we will use the `'main'` state name as the argument to load our new state.

Okay, now that we have our empty project set up, you can test it out to see if it works. You should see an empty canvas with a black background similar to the screen shot below.

In the next part[3] of this tutorial, we’ll load our game assets and our sprites.

1. https://github.com/zekechan/phaser-html5-tutorial-pong/releases/download/1.0/1-Empty_project.zip
2. http://brackets.io/
3. http://zekechan.net/getting-started-html5-game-development-pong2/