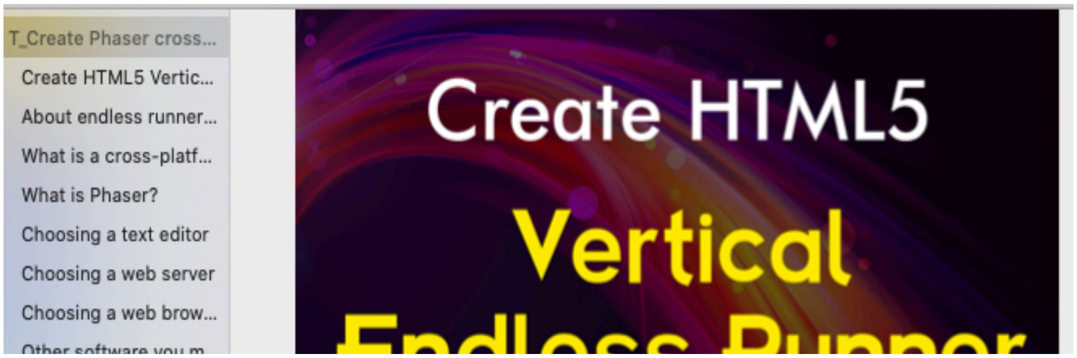
Week_27

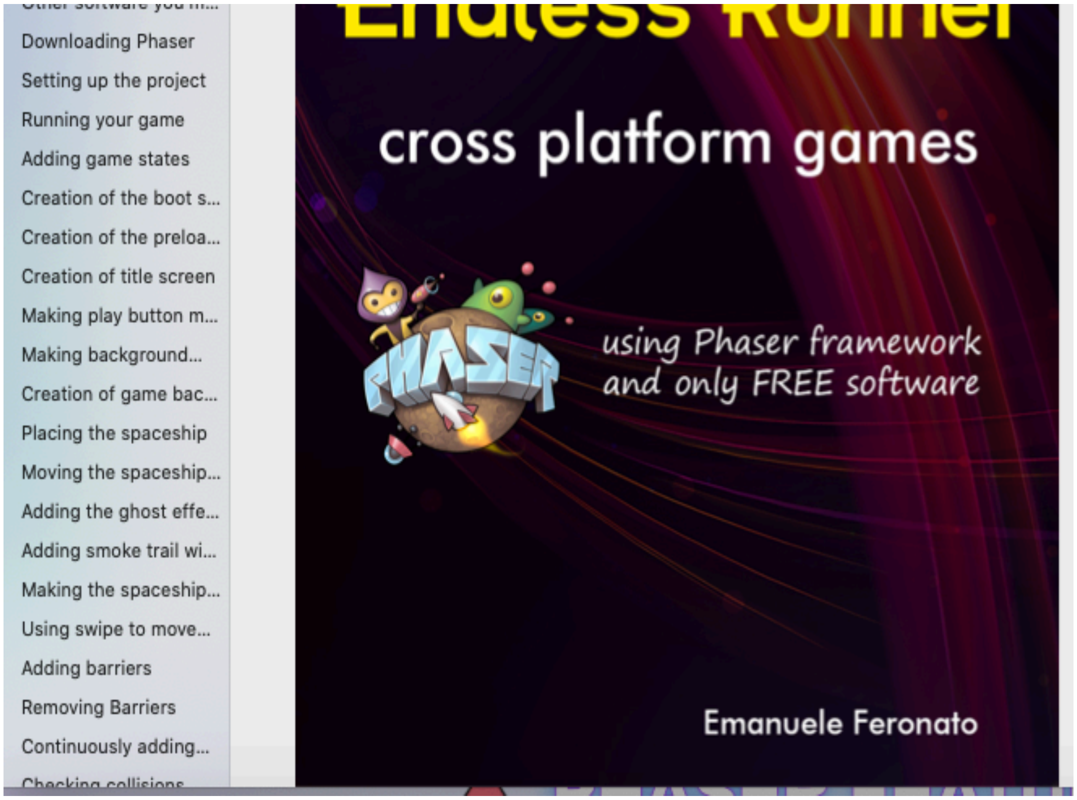
Trinity - intro to computer Programming
With Game Development

04/10/2023

Congratulations to all our students!
(you are learning more and growing in your skill set!)

Students we will begin a new tutorial for our
Next game development segment -
I want to credit the author:
(below)





Downloading Phaser
Setting up the project
Running your game
Adding game states
Creation of the boot s...
Creation of the preloa...
Creation of title screen
Making play button m...
Making background...
Creation of game bac...
Placing the spaceship
Moving the spaceship...
Adding the ghost effe...
Adding smoke trail wi...
Making the spaceship...
Using swipe to move...
Adding barriers
Removing Barriers
Continuously adding...

cross platform games

using Phaser framework and only FREE software

Emanuele Feronato

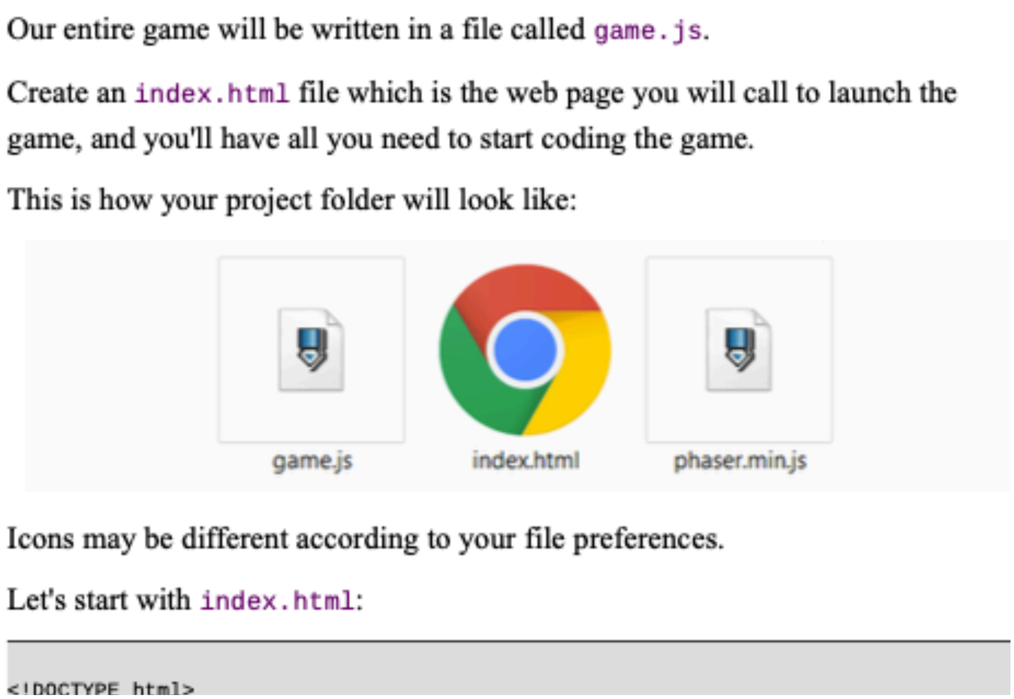
I want to credit the author: Emanuele Feronato

Today, We will have 4 Demonstrations…
Students focus on how the game is
Developed, what strategies the author is teaching,
And the code we are learning, lastly about the game
Itself…

We can briefly review the introduction, we already know about browsers,
I.D.E. Integrated Development Environment - Brackets,
Web Servers - Fenix (PC) or
(Mac) Python3 -m simple.http 80xx(Terminal)

Setting up the project

The whole project is basically a web page including Phaser framework and another JavaScript file with our game.
There are two important things you should consider before you start coding your game: first, size matters. When importing third party frameworks like Phaser, always choose minified versions if provided.
Talking about Phaser, inside `build` folder in the zipped package you just downloaded, you will find `phaser.min.js` file. That's the only Phaser file we will need during the development of our game.
Second, writing the entire code of a game in a single file is generally considered a malpractice. You should create one JavaScript file for the splash screen, another for game logic, and basically one file for each actor you will include in your game.   (Launch the index.html)
The problem is some sponsors need to have the entire game in one file. And since finding sponsors and selling game licenses can be a great income source, we have to code games with sponsors needs in mind.



Our entire game will be written in a file called `game.js`.

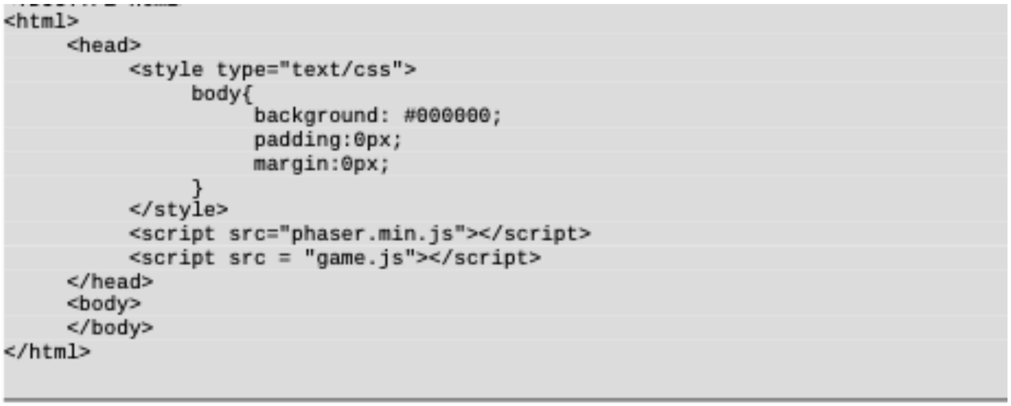
Create an `index.html` file which is the web page you will call to launch the game, and you'll have all you need to start coding the game.

This is how your project folder will look like:

Icons may be different according to your file preferences.

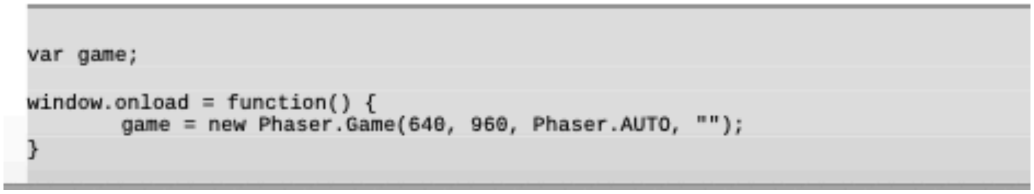
Let's start with `index.html`:

```
<!DOCTYPE html>
```



```
<html>
     <head>
          <style type="text/css">
               body{
                    background: #000000;
                    padding:0px;
                    margin:0px;
               }
          </style>
          <script src="phaser.min.js"></script>
          <script src = "game.js"></script>
     </head>
     <body>
     </body>
</html>
```

As you can see, it's just an empty web page with only a call to two JavaScript files: `phaser.min.js` is the file we just downloaded, and `game.js` will contain our game script. There are very few lines in `game.js` too, at the moment:

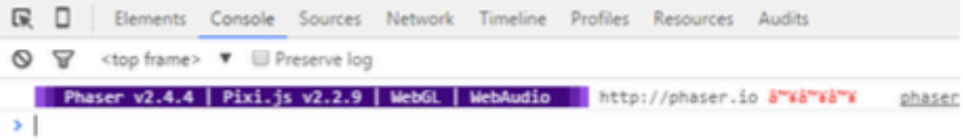


```
var game;

window.onload = function() {
       game = new Phaser.Game(640, 960, Phaser.AUTO, "");
}
```

Running your game

To run the game on your local server, simply point your browser to your game folder which in most cases will be http://localhost/yourgamefolder/ and this is what you should get if running it on your Google Chrome browser:



```
 Phaser v2.4.4 | Pixi.js v2.2.9 | WebGL | WebAudio   http://phaser.io â™¥â™¥â™¥
```

This is the default debug string Phaser prompts on the console window. You can generally access to your console window pressing F12 in your browser, anyway refer to your browser documentation.

Text output may vary a bit according to Phaser version, this book has been updated to **Phaser 2.4.7**.

Adding game states

Although managing Phaser states is an advanced feature, it's very important to learn how to use states from the beginning of your Phaser programming course, as they will allow you to write better code and have a better resource management.

Let's think about the game we are making. We still do not know to code it but we can easily imagine the game will all have at least a title screen, a screen with the game itself, and a game over screen.

Each "screen" can be developed as a Phaser state, which can be executed cleaning memory and resources before it starts, allowing us to easily switch through game "screens".

Now let's write this concept in a more detailed way, listing all the states we will actually use in our game:

Boot state: in the boot state we will make all adjustment to the game to be resized

Creation of title screen

We want the title screen to have game title displayed – obviously – and a "play" button. Also, to give the game a more modern feeling, the background color should change at each play.

While just choosing a random color would be easy, you understand not all colors are suitable as background colors. We want to have only a small selection of background colors and randomly choose among them.

To choose nice background colors, if you don't have ideas you can inspire yourself by googling something like "color schemes", you'll get a lot of ideas, satisfaction guaranteed.

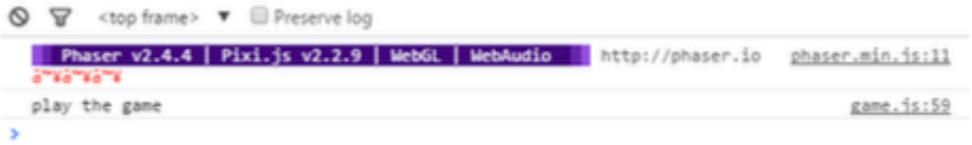
All background colors selected will be stored into `bgColors` array which will be declared as a global variable. Declaring all customizable variables as global variables is a good practice from a game developer point of view because it will allow sponsors to easily edit the most important parameters if they need to tune the gameplay.

So, here is `bgColors` array with its 10 colors to be randomly chosen:

```
var game;
var bgColors = [0xF16745, 0xFFC65D, 0x7BC8A4, 0x4CC3D9, 0x93648D, 0x7c786a,
0x588c73, 0x8c4646, 0x2a5b84, 0x73503c];
```

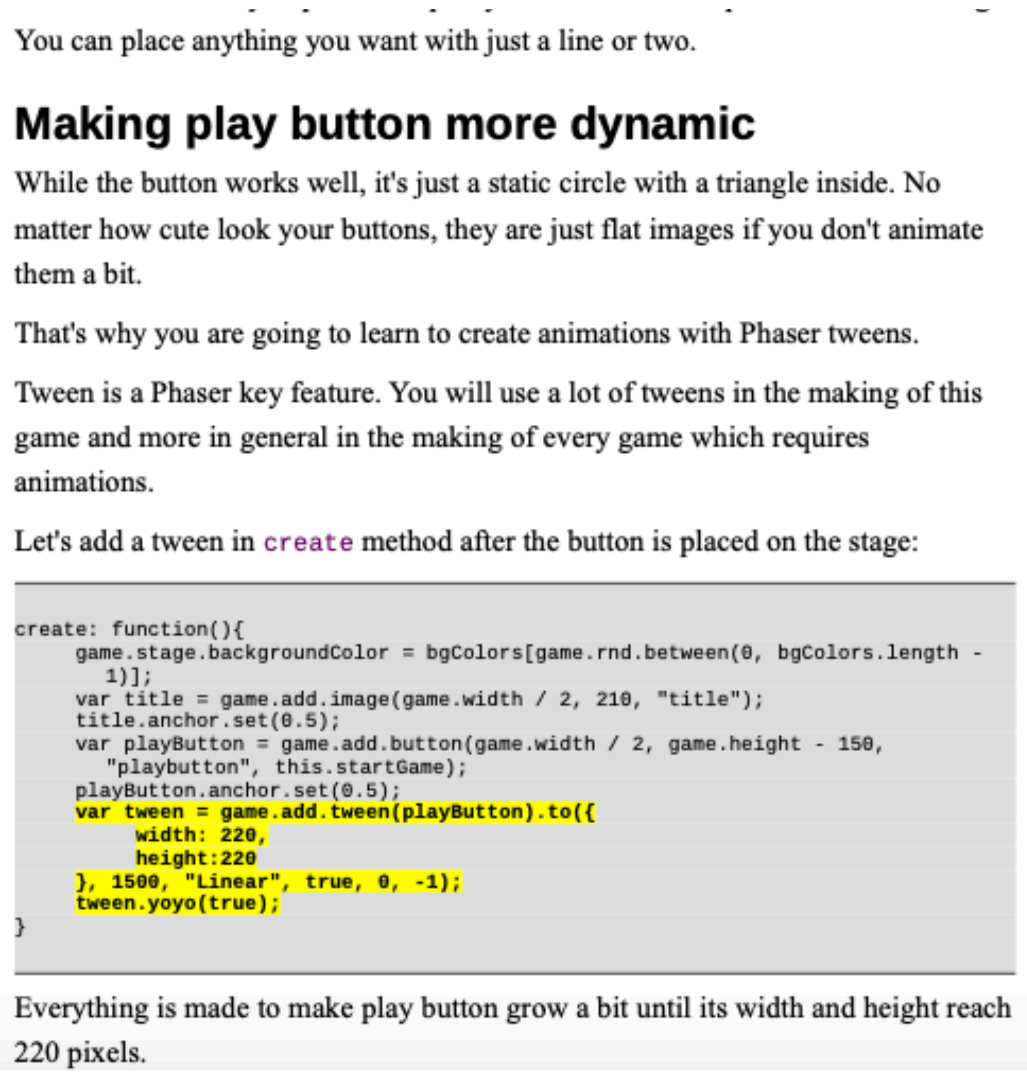
Now we have to choose one color, then place the title and the play button we

Click or tap on the button, and you will be redirected on a black screen – actually `PlayGame` state – seeing a debug message on your console.



```
Phaser v2.4.4 | Pixi.js v2.2.9 | WebGL | WebAudio   http://phaser.io   phaser.min.js:11
play the game                                                          game.js:59
>
```

This has been a very important step as you learned how to place stuff on the stage.



You can place anything you want with just a line or two.

Making play button more dynamic

While the button works well, it's just a static circle with a triangle inside. No matter how cute look your buttons, they are just flat images if you don't animate them a bit.

That's why you are going to learn to create animations with Phaser tweens.

Tween is a Phaser key feature. You will use a lot of tweens in the making of this game and more in general in the making of every game which requires animations.

Let's add a tween in `create` method after the button is placed on the stage:

```
create: function(){
     game.stage.backgroundColor = bgColors[game.rnd.between(0, bgColors.length -
        1)];
     var title = game.add.image(game.width / 2, 210, "title");
     title.anchor.set(0.5);
     var playButton = game.add.button(game.width / 2, game.height - 150,
        "playbutton", this.startGame);
     playButton.anchor.set(0.5);
     var tween = game.add.tween(playButton).to({
          width: 220,
          height:220
     }, 1500, "Linear", true, 0, -1);
     tween.yoyo(true);
}
```

Everything is made to make play button grow a bit until its width and height reach 220 pixels.

From our downloads lets open the pdf file:
T_Create Phaser cross platform games.pdf

Demo 1 <ship>   , moves with a click
Demo 2 <add emitter> visible
Demo 3 <add vertical lift>