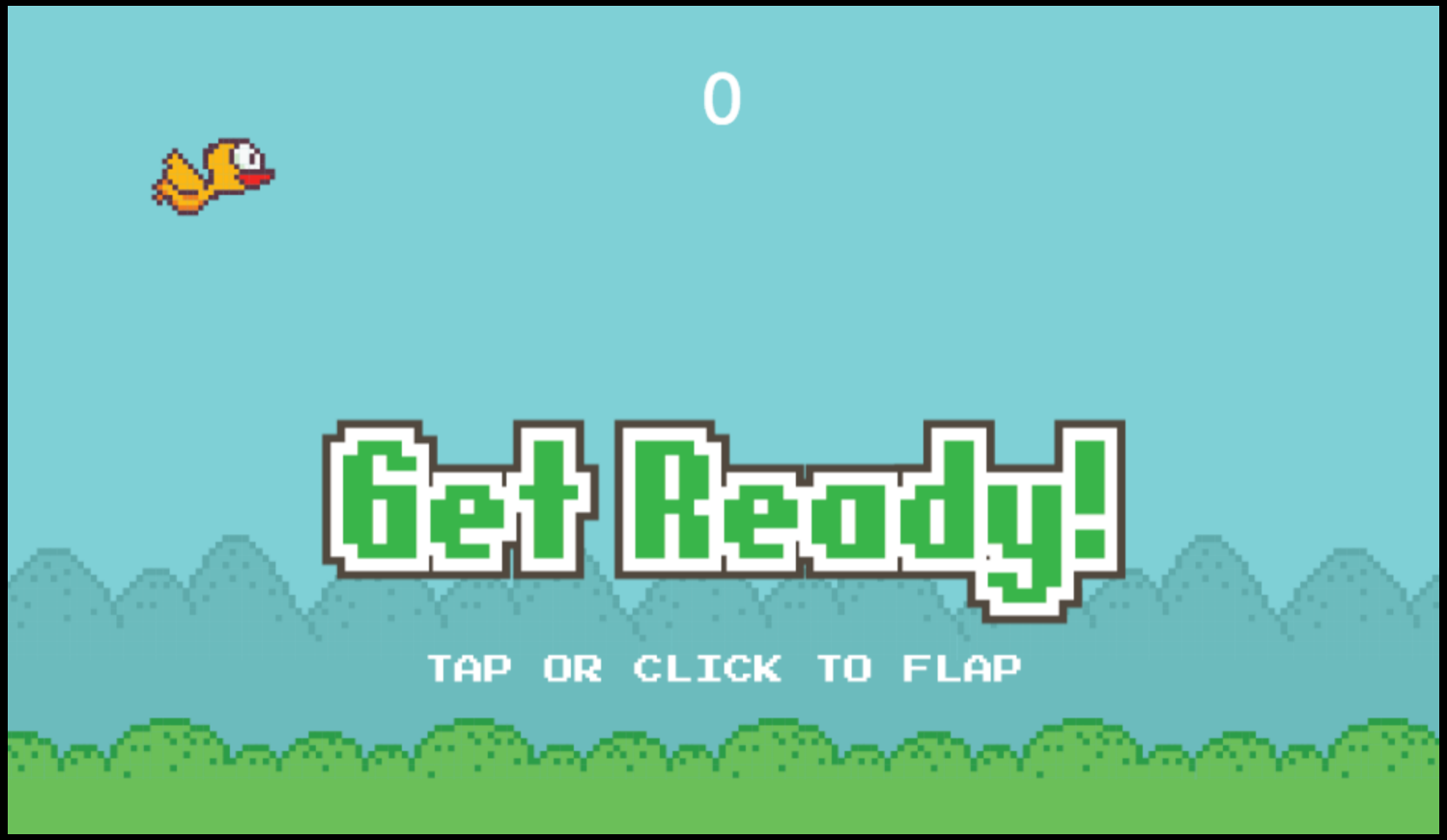
Flappy Flap

An iPad game written in javascript


0
Get Ready!
TAP OR CLICK TO FLAP

Things we need to build this game

A Code Editor - Sublime Text

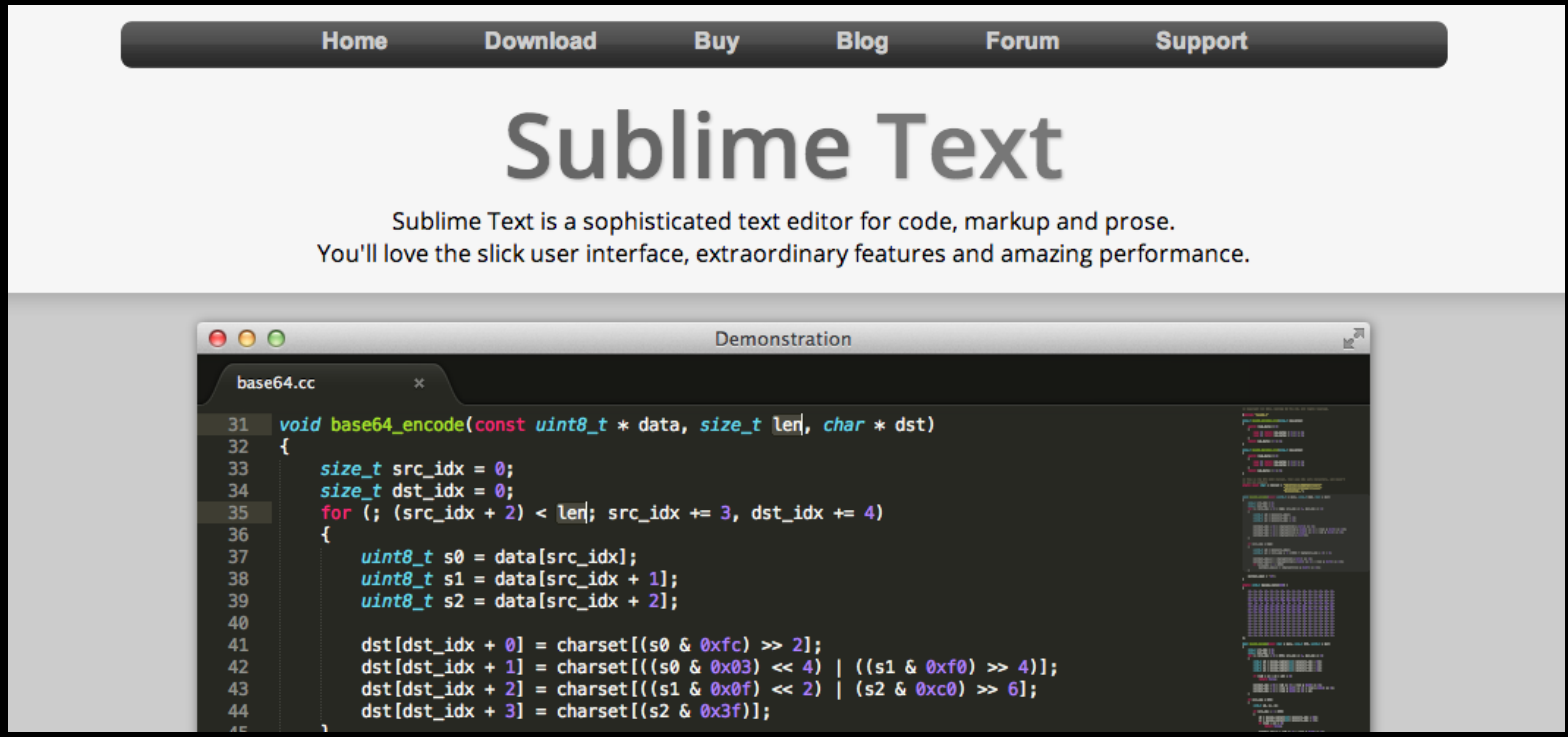
A Test environment - Web Browser

A Game engine - Enchant.js

Assets - Images are included with this project

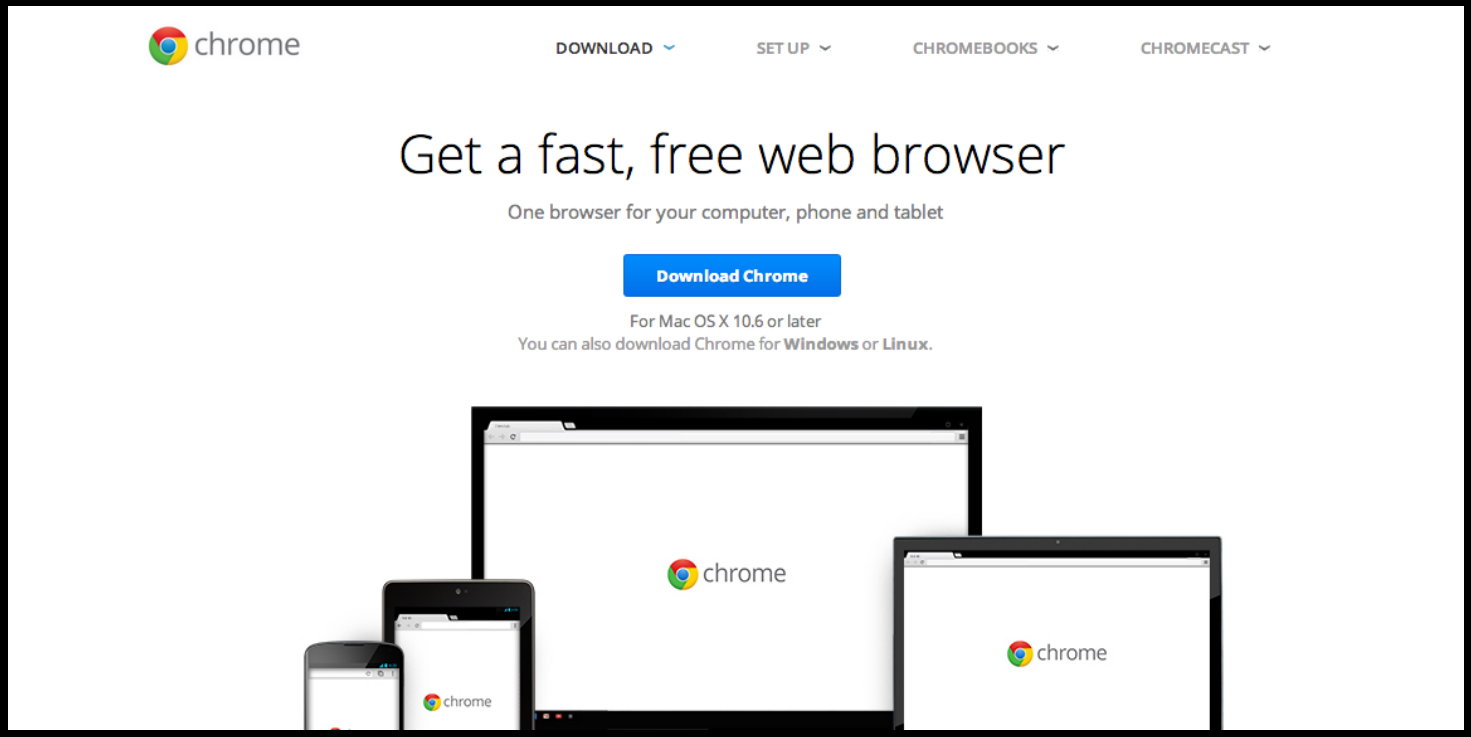
Sublime Text.app

sublimetext.com



Test Environment

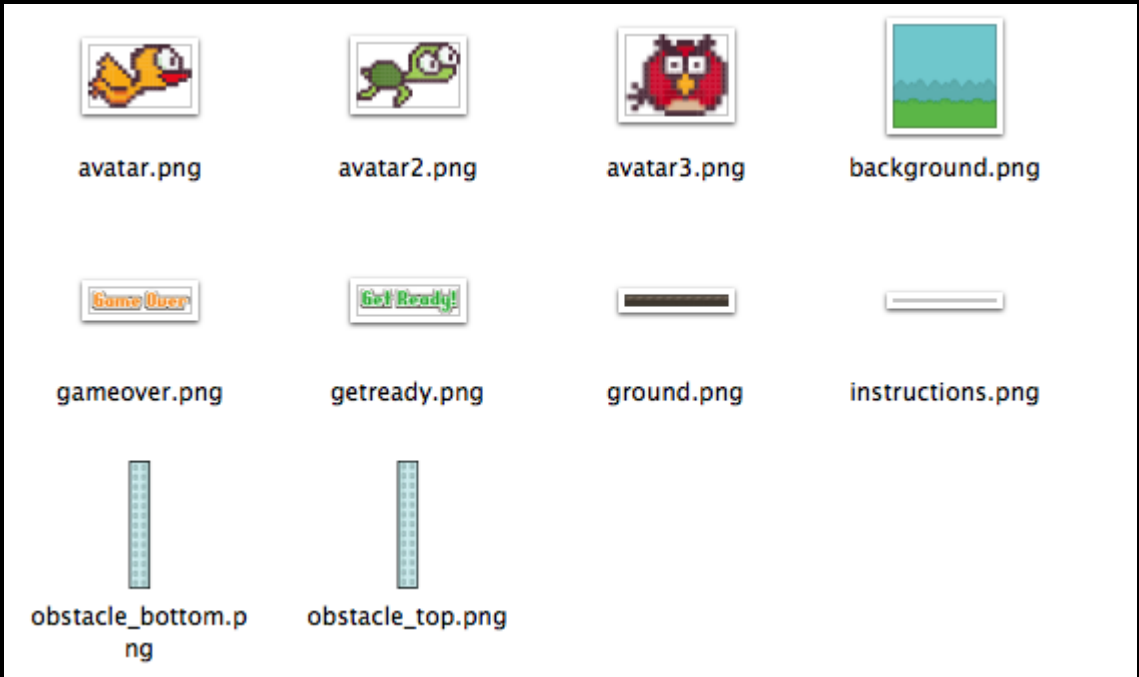
Use Google Chrome!



Enchant.js

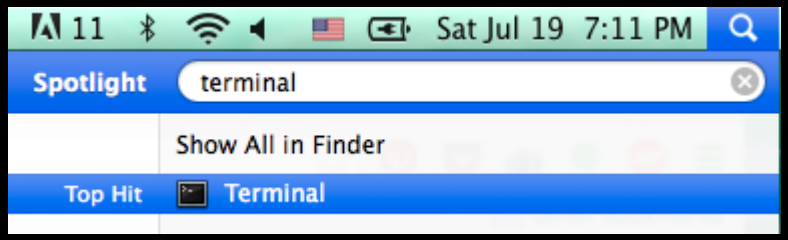
enchantjs.com

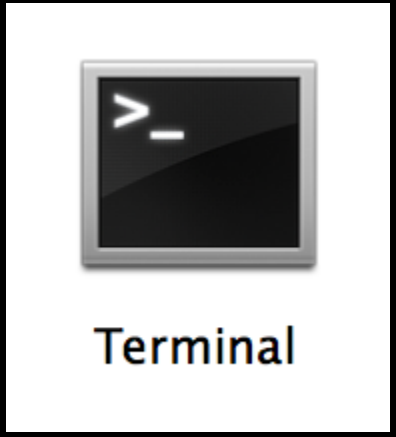
Assets


avatar.png
avatar2.png
avatar3.png
background.png
Game Over
gameover.png
Get Ready!
getready.png
ground.png
instructions.png
obstacle_bottom.png
obstacle_top.png

Open Terminal.app

Use Spotlight to find and open the **Terminal** app.





Type a command in Terminal.app

```
bash <(curl -s http://summer2014.gomagames.com/flappy/setup)
```

Then press Return.

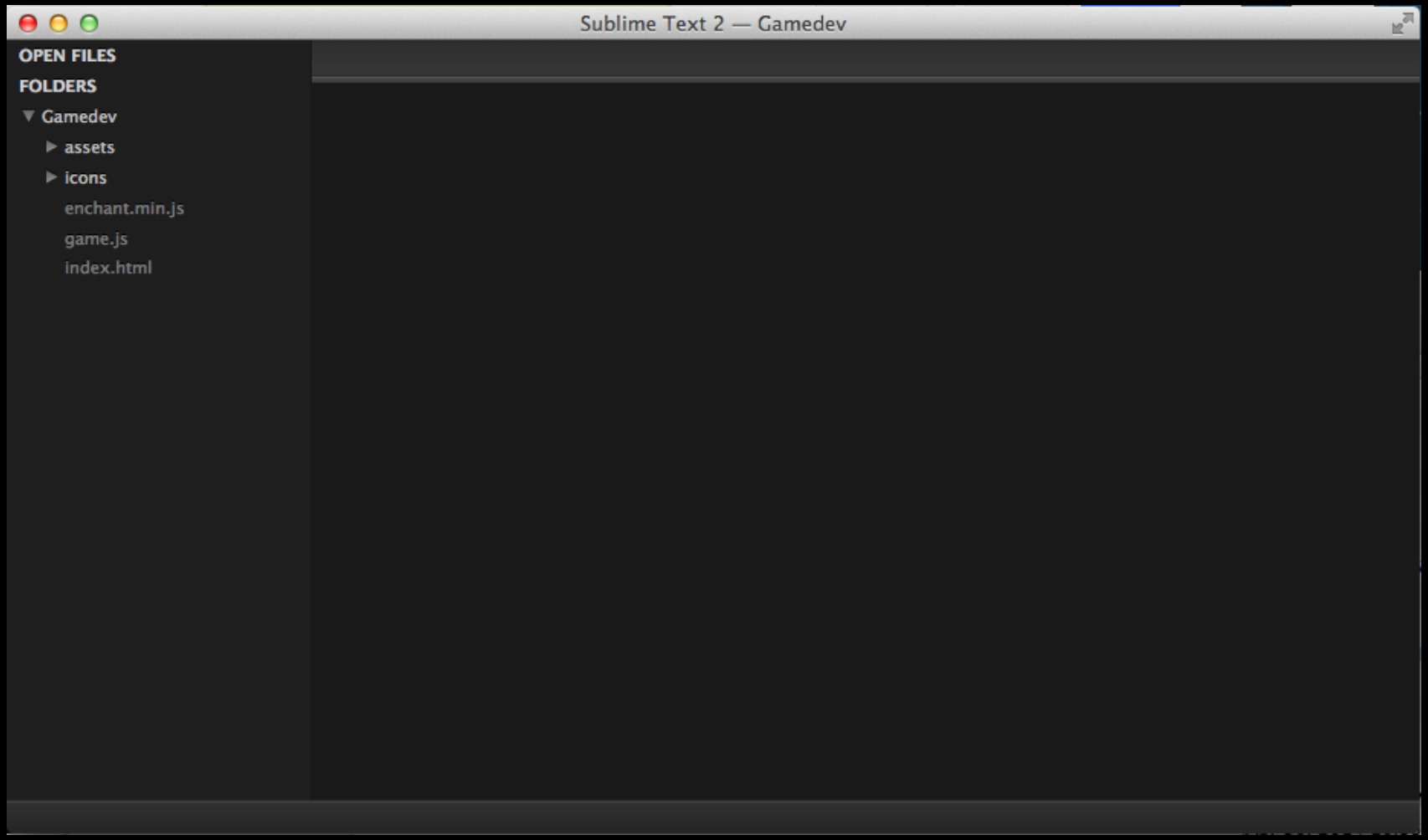
Close the terminal window when it says `all done`

Open Project

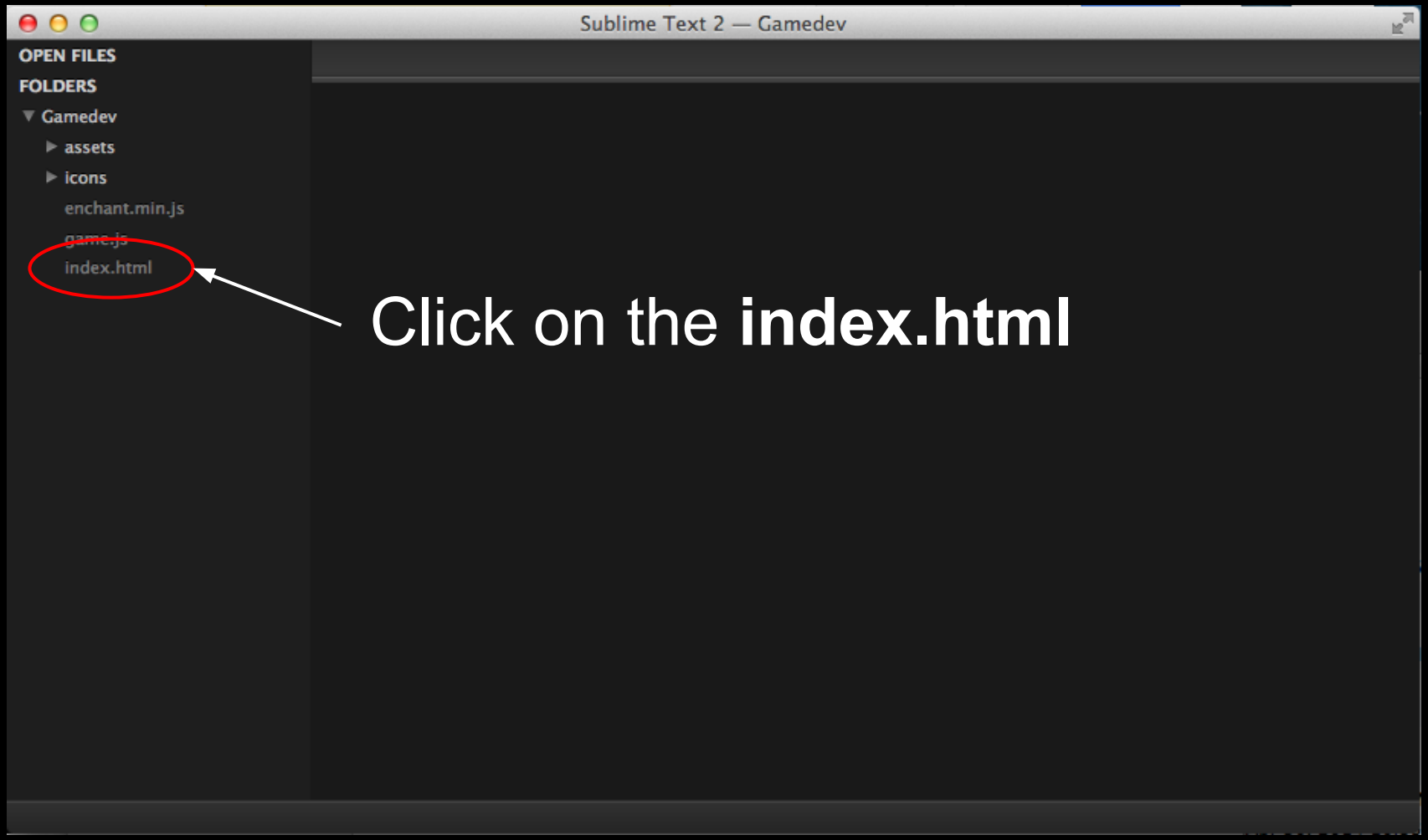
- Find Your FlappyFlap folder on the Desktop

- Drag the FlappyFlap folder onto the SublimeText icon in your Dock.

Setup: Sublime Text



Step 1 - Code Author's name



Step 1 - Code Author's name

Change YOURNAME to your first name:

```
<!-- AUTHOR: YOURNAME -->
<!DOCTYPE html>
<html>
<head>
```

Step 2 - Change the game name

Change GAME TITLE to whatever you want to call your game:

```
<title>GAME TITLE</title>
```

Step 3: Include the game engine

Add the game engine library
in the body of the index.html page.

```
<body>
  <script src="enchant.min.js"></script>
</body>
```

Step 4: Include your game code

After the game engine, add the game.js file:

```
<body>
  <script src="enchant.min.js"></script>
  <script src="game.js"></script>
</body>
```

Test Cycle

Right-Click on the **index.html** file.

Select **Open in Browser.**

You should see a **Blank Page**.

From now on, whenever you finish a step:

1. Edit Code in the file you are working on.
2. Save your File (**Command-S**).
3. Refresh Chrome (**Command-R**).

Click on **game.js** to open it.

What does this do?

```
enchant(); // initialize
var game = new Core(1000, 730); // game stage

// Initialize game
game.onload = function(){


} // end game.onload #initialize game
```

What does this do?

```
// Game loop
game.onenterframe = function(){



} // end game.onenterframe #game loop
```

What does this do?

```
game.start();
```

Open a tab for Instructions

Open another tab in Chrome and go to **stem.gomagames.com**

Click on the link to **FlappyFlap** to see the instructions up-close. You may need to refer to this page later.

Do you want to follow along with the class or follow the self-guided tutorial?

Decide NOW!!!

Go back to the **game.js** file.
This is the only file you need to edit.

Step 1:

Find the code that initializes the game

```
// initialize game
game.onload = function(){
  // add the background
  game.bg = new Sprite(1000,730);
  game.bg.image = game.assets['assets/background.png'];

  // #1 add game.bg to rootScene
```

Step 1:

Add game.bg to the rootScene

```
// initialize game
game.onload = function(){
  // add the background
  game.bg = new Sprite(1000,730);
  game.bg.image = game.assets['assets/background.png'];

  // #1 add game.bg to rootScene
  game.rootScene.addChild(game.bg);
```

Step 1:

Refresh Chrome! (Command-R)

Step 2:

Find the comment for adding “Get Ready” to the screen.

```
// #2 add game.getready to rootScene
```

Step 2:

Add the getready text to the rootScene.

```
// #2 add game.getready to rootScene
game.rootScene.addChild(game.getready);
```

Step 2:

Refresh Chrome! (Command-R)

Step 3:

Find the comment that adds the ground to the scene.

```
// #3 add game.ground to rootScene
```

Step 3:

Add the ground to the scene.

```
// #3 add game.ground to rootScene
game.rootScene.addChild(game.ground);
```

Step 3:

Refresh Chrome! (Command-R)

Step 4:

Find the comment for adding the game avatar.

```
// add the main character
game.avatar = new Sprite(87,55);
game.avatar.image = game.assets['assets/avatar.
png'];
game.avatar.x = 100;
game.avatar.y = 100;
game.avatar.ySpeed = 0;

// #4 add game.avatar to rootScene
```

Step 4:

Add the main character to the scene.

```
// add the main character
game.avatar = new Sprite(87,55);
game.avatar.image = game.assets['assets/avatar.
png'];
game.avatar.x = 100;
game.avatar.y = 100;
game.avatar.ySpeed = 0;

// #4 add game.avatar to rootScene
game.rootScene.addChild(game.avatar);
```

Step 4:

Refresh Chrome! (Command-R)

Step 5:

Add the game instructions to the scene.

```
// add the instructions
game.instructions = new Sprite(420,22);
game.instructions.image = game.assets['assets/instru
game.instructions.x = (game.width/2) - (game.instruc
game.instructions.y = 460;

// #5 add game.instructions to rootScene
game.rootScene.addChild(game.instructions);
```

Step 5:

Refresh Chrome! (Command-R)

Step 6: Find the code to start the game when touched.

```
function game_touched(){
  if(game.started){

    // #7 flap

  }else{

    // #6 start game

    // #6 remove getready and instructions

  }
}
```

Step 6:

Add the code to start the game
and remove the getready and instructions text.

```
// #6 start game
game.started = true;
// #6 remove getready and instructions
game.rootScene.removeChild(game.getready);
game.rootScene.removeChild(game.instructions);
```

Step 6:

Refresh Chrome! (Command-R)
Click the screen to see what happens.

Step 7:

Find the comment for flapping when the game is touched.

```
function game_touched(){
  if(game.started){

    // #7 flap


  }else{
```

Step 7:

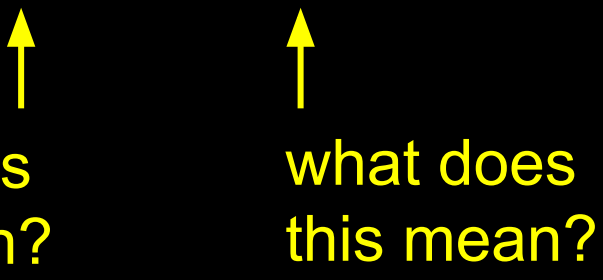
Add the code to make the bird “flap” if the screen is touched and the game is already started.

```
function game_touched(){
  if(game.started){

    // #7 flap
    game.avatar.ySpeed = -game.flap_strength;

  }else{
```

Step 7:

What is happening here?

```
game.avatar.ySpeed = -game.flap_strength;
```



Step 7:

Refresh Chrome! (Command-R)
Click the screen to see what happens.

Step 8: Find the code for adding the obstacles.

```
// #8 add top obstacle to obstacles group


var bottom = new Sprite(85, 545);
bottom.image = game.assets['assets/obstacle_bottom.p
bottom.x = -obstacles.x + game.width;
bottom.y = pos;

// #8 add bottom obstacle to obstacles group
```

Step 8: Add the obstacles.

```
// #8 add top obstacle to obstacles group
obstacles.addChild(top);

var bottom = new Sprite(85, 545);
bottom.image = game.assets['assets/obstacle_bottom.p
bottom.x = -obstacles.x + game.width;
bottom.y = pos;

// #8 add bottom obstacle to obstacles group
obstacles.addChild(bottom);
```

Step 8:

Refresh Chrome! (Command-R)
Click the screen to see what happens.

Nothing new loads!
What is wrong?

Step 8:

Refresh Chrome! (Command-R)
Click the screen to see what happens.

Nothing new loads!
What is wrong?

Remember when we added things to the rootScene?
We have to do that for every new graphic we add.

Step 9: Scroll up to the upper part of the document.
Find the comment for adding the obstacles to the rootScene

```
// #1 add game.bg to rootScene
game.rootScene.addChild(game.bg);

// #9 add obstacles to rootScene


// #16 add scoreBoard to rootScene
```

Step 9: Add the obstacles to the rootScene.

```
// #1 add game.bg to rootScene
game.rootScene.addChild(game.bg);

// #9 add obstacles to rootScene
game.rootScene.addChild(obstacles);

// #16 add scoreBoard to rootScene
```

Step 9:

Refresh Chrome! (Command-R)
Click the screen to see what happens…

Still, nothing new :(
Why?

Step 9:

Refresh Chrome! (Command-R)
Click the screen to see what happens…

Still, nothing new :(
Why?

The obstacles are loaded off-screen.
We have to move them onto the canvas.

Step 10: In the game loop, find the comment for moving obstacles to the left.

```
}

// #10 move obstacles to the left



// track flying progress
game.distance += game.fly_speed;

// check if we need to spawn obstacle
if(game.distance % game.obstacle_frequency == 0){
```

Step 10: Make the obstacles move onto the screen.

```
}

// #10 move obstacles to the left
obstacles.x -= game.fly_speed;

// track flying progress
game.distance += game.fly_speed;

// check if we need to spawn obstacle
if(game.distance % game.obstacle_frequency == 0){
```

Step 10:

What is happening here?

what does
this mean?

```
obstacles.x -= game.fly_speed;
```

what does
this mean?

Step 10:

Refresh Chrome! (Command-R)
Click the screen to see what happens…

Still, nothing new :(
Why?

Step 10:

Refresh Chrome! (Command-R)
Click the screen to see what happens…

Still, nothing new :(
Why?

We programmed a lot of settings for the obstacles, though we never *called the function* to spawn the obstacles. :(

Step 11: Find the comment to spawn obstacle.

```
// check if we need to spawn obstacle
    if(game.distance % game.obstacle_frequency == 0)

      // #11 spawn obstacle



      // clean up old obstacles
      for (var i = 0; i < obstacles.childNodes.lengt
        if(obstacles.childNodes[i].x + obstacles.x +
          obstacles.removeChild(obstacles.childNodes
        }
      }
    } // end if spawn obstacle
```

Step 11: Write the function call to spawn an obstacle.

```
// check if we need to spawn obstacle
    if(game.distance % game.obstacle_frequency == 0)

      // #11 spawn obstacle
      spawnObstacle();

      // clean up old obstacles
      for (var i = 0; i < obstacles.childNodes.lengt
        if(obstacles.childNodes[i].x + obstacles.x +
          obstacles.removeChild(obstacles.childNodes
        }
      }
    } // end if spawn obstacle
```

Step 11:

Refresh Chrome! (Command-R)
Click the screen to see what happens.

Step 12:

Refresh Chrome! (Command-R)
Click the screen to see what happens.

Yay! Finally. :)
You should see some buildings flying by.

Step 12: Find the comment to end the game if the bird hits a building.

```
// collision detection
for (var i = 0; i < obstacles.childNodes.length; i++
  if(game.avatar.intersect(obstacles.childNodes[i]))

    // #12 end the game if player hits obstacle


  }else if( // check for score
    !obstacles.childNodes[i].scored &&
    obstacles.childNodes[i].y < 0 &&
    obstacles.childNodes[i].x +
    obstacles.childNodes[i].width -
```

Step 12: Find the comment to end the game if the bird hits a building.

```
// collision detection
for (var i = 0; i < obstacles.childNodes.length; i++
  if(game.avatar.intersect(obstacles.childNodes[i]))

    // #12 end the game if player hits obstacle
    gameover();

  }else if( // check for score
    !obstacles.childNodes[i].scored &&
    obstacles.childNodes[i].y < 0 &&
    obstacles.childNodes[i].x +
    obstacles.childNodes[i].width -
```

Step 12:

Refresh Chrome! (Command-R)

Click the screen to see what happens...

Step 12:

Refresh Chrome! (Command-R)
Click the screen to see what happens…

Something seems wrong…
The game sort of ends, because if you hit a building, the next click restarts the game, though it doesn't really seem like the game is over.

Step 12:

Refresh Chrome! (Command-R)
Click the screen to see what happens…

Something seems wrong…
The game sort of ends, because if you hit a building, the next click restarts the game, though it doesn't really seem like the game is over.

We need to add some visual feedback so it looks like the game is over.

Step 13: Find the comment to end the game
if the bird falls to the floor or smashes into the ceiling.

```
// check if bird touches floor, or ceiling
if(
   game.avatar.y+game.avatar.height > game.ground.y
   game.avatar.y < 0
){

  // #13 end the game if avatar touches floor
  or ceiling



}
```

Step 13: Add the function call to end the game if the bird falls to the floor or smashes into the ceiling.

```
// check if bird touches floor, or ceiling
if(
    game.avatar.y+game.avatar.height > game.ground.y
    game.avatar.y < 0
){

  // #13 end the game if avatar touches floor
  or ceiling
  gameover();

}
```

Step 13:

Refresh Chrome! (Command-R)
Click the screen to see what happens…

The game still sort of ends when you hit the floor, ceiling, or a building (because the next click after you hit something restarts the game).

However, it still doesn't *feel* like we crashed into something.

Step 14: Find the comment to show a Game Over graphic if you hit something.

```
function gameover(){
  // add the instructions
  game.gameover = new Sprite(602,163);
  game.gameover.image = game.assets['assets/gameover
  game.gameover.x = (game.width/2) - (game.gameover.
  game.gameover.y = 120;

  // #14 show gameover graphic


  game.rootScene.addEventListener(enchant.Event.TOUC
    window.location.reload();
```

Step 14:

Load the gameover graphic to the rootScene.

```
function gameover(){
  // add the instructions
  game.gameover = new Sprite(602,163);
  game.gameover.image = game.assets['assets/gameover
  game.gameover.x = (game.width/2) - (game.gameover.
  game.gameover.y = 120;

  // #14 show gameover graphic
  game.rootScene.addChild(game.gameover);

  game.rootScene.addEventListener(enchant.Event.TOUC
    window.location.reload();
```

Step 14:

Refresh Chrome! (Command-R)
Click the screen to see what happens.

Step 14:

Refresh Chrome! (Command-R)
Click the screen to see what happens.

Better!
A Game Over sign appears if we hit something.

Step 15: Find the comment to stop the game loop.

```
    game.rootScene.addEventListener(enchant.Event.TO
    window.location.reload();
  });

  // #15 stop the game loop


}
```

Step 15: Stop the game loop.

```
    game.rootScene.addEventListener(enchant.Event.TO
    window.location.reload();
  });

  // #15 stop the game loop
  game.onenterframe = null;

}
```

Step 15:

Refresh Chrome! (Command-R)
Click the screen to see what happens.

Step 15:

Refresh Chrome! (Command-R)
Click the screen to see what happens.

Better!
Now things stop moving when we hit the buildings, floor, or ceiling.

Step 16: Find the comment to add a scoreboard.

```
// #1 add game.bg to rootScene
game.rootScene.addChild(game.bg);


// #9 add obstacles to rootScene
game.rootScene.addChild(obstacles);


// #16 add scoreBoard to rootScene
```

Step 16: Add the scoreboard to the rootScene

```
// #1 add game.bg to rootScene
game.rootScene.addChild(game.bg);


// #9 add obstacles to rootScene
game.rootScene.addChild(obstacles);


// #16 add scoreBoard to rootScene
game.rootScene.addChild(scoreBoard);
```

Step 16

Refresh Chrome! (Command-R)
Click the screen to see what happens.

Step 17:

Find the comment to increment score by 1

```
    obstacles.childNodes[i].x +
    obstacles.childNodes[i].width -
    game.distance < game.avatar.x
){
    obstacles.childNodes[i].scored = true;

    // #17 increment score by 1


}
```

Step 17:

Increment the score by 1 when the bird passes a building.

```
    obstacles.childNodes[i].x +
    obstacles.childNodes[i].width -
    game.distance < game.avatar.x
){
    obstacles.childNodes[i].scored = true;

    // #17 increment score by 1
    scoreBoard.text++;

}
```

Step 17

Refresh Chrome! (Command-R)
Click the screen to see what happens.

DONE!!!

You finished the game!

Experiment

Experiment with changing the position, speed, etc of the bird.

Game Graphics Mods

Use an original Avatar Graphic

Hint: You must put the graphic in the /assets folder. Then, you must reference the correct image in the game.js file. There are **TWO** places that you must swap out the image name.

Use an original Background Graphic

Hint: You must put the graphic in the /assets folder. Then, you must reference the correct image in the game.js file. There are **TWO** places that you must swap out the image name.

Modify the app icons

Hint: update all icons in the icons folder. You must use the correct image size!! Best method: Open the icon in Photoshop and copy your graphic on top of the file.

Run the deploy.game.command again when you finish your mods.

Deploying your game to iPad

In the FlappyFlap folder on your Desktop, double-click the `deploy.game.command`

The terminal app will ask:

```
Are you sure you want to continue connecting (yes/no)?
```

Type **yes** and hit Enter.

When it says `You can close this terminal now`

you can close the terminal window.

Play the game on your iPad

Visit **summer2014.gomagames.com**
using almost any mobile device. It works on iPad, it might not work optimally on other devices.

Click on your **name** to access your game.

Click the **bookmark icon** to add the game to your homescreen.

Exit the browser.

On the homescreen, click on the icon to your game.

Play!