



# JumpSlide

# Level Design

First, we are going to plan out the game we want to build. On graph paper, plan out a level you would want to build. Remember, the primary game components available are:

## **platforms**

- floating platforms
- floors
- tall walls
- pits
- tunnels
- low hanging walls

## **coins**

- when collected, these translate to your game score

## **goal flags**

- touching a flag ends the game.

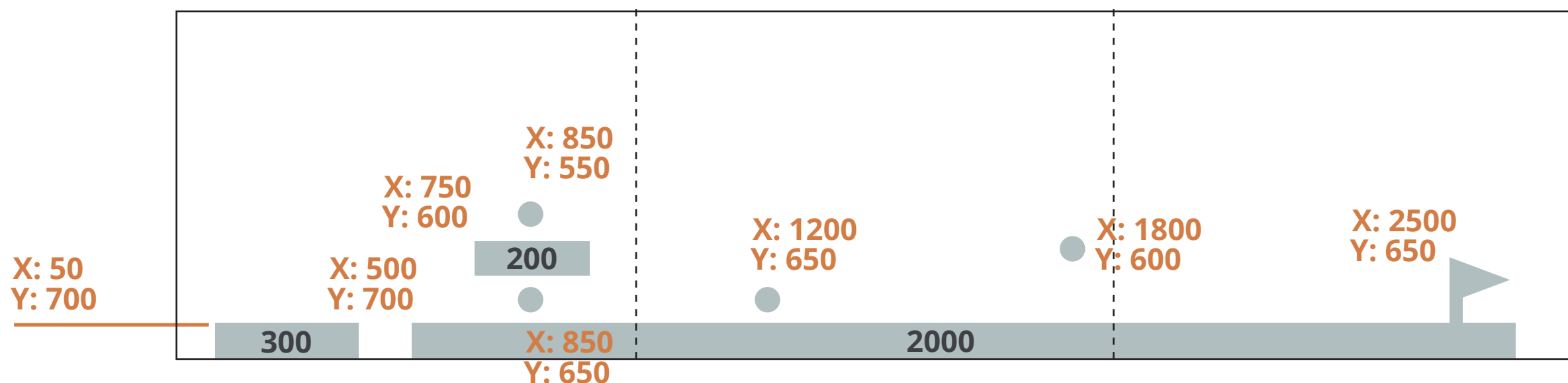
# Level Design

Here's a Demo Game Level Plan.



# Level Design

Here's a Demo Game Level Plan. Note that position is based on the upper left corner of the object. Yes, this requires **math**. Plan out some of the coordinates for objects in your game, however know that they are just estimates at this point.



Standard Floor Height: 68px  
Y-Coordinate Floor Position: 700px

Floating Platform Height: 45px  
Floating Platform Width: 200px

# Level Design to Code

The function calls for placing objects on the screen follow this format:

Placing a Platform:

```
JumpSlide.addPlatform(50, 700, 300, 68);
```

x                  y                  width      height

Placing a Coin:

```
JumpSlide.addCoin(50, 700);
```

x                  y

Placing a Goal:

```
JumpSlide.addGoal(50, 700);
```

x                  y

# Level Design to Code (Advanced)

If you want to re-use some values throughout your game level, you can create some settings for your objects. Inside the GAME.init block, at the TOP you can define your own variables, then throughout your game level you can reuse those variables.

```
// Settings
```

```
var platform_width = 200;
```

```
var platform_height = 50;
```

```
var floor_height = 68;
```

```
var floor_y = 700;
```

```
// Demo Game Level
```

```
JumpSlide.addPlatform(50, floor_y, 500, floor_height);
```

```
JumpSlide.addPlatform(650, floor_y, 500, floor_height);
```

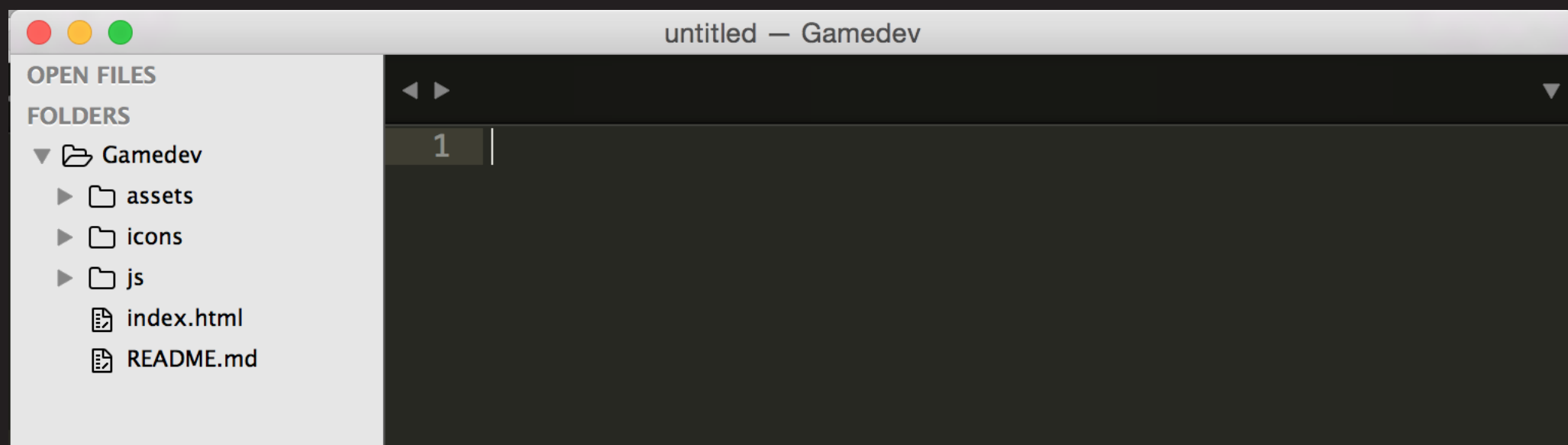
```
JumpSlide.addPlatform(1200, 600, platform_width, platform_height);
```

Test in Chrome!

# Workspace Setup

Open the GameDev directory in Sublime Text. Be sure to open the whole folder, not just a single file. To Open the entire folder...

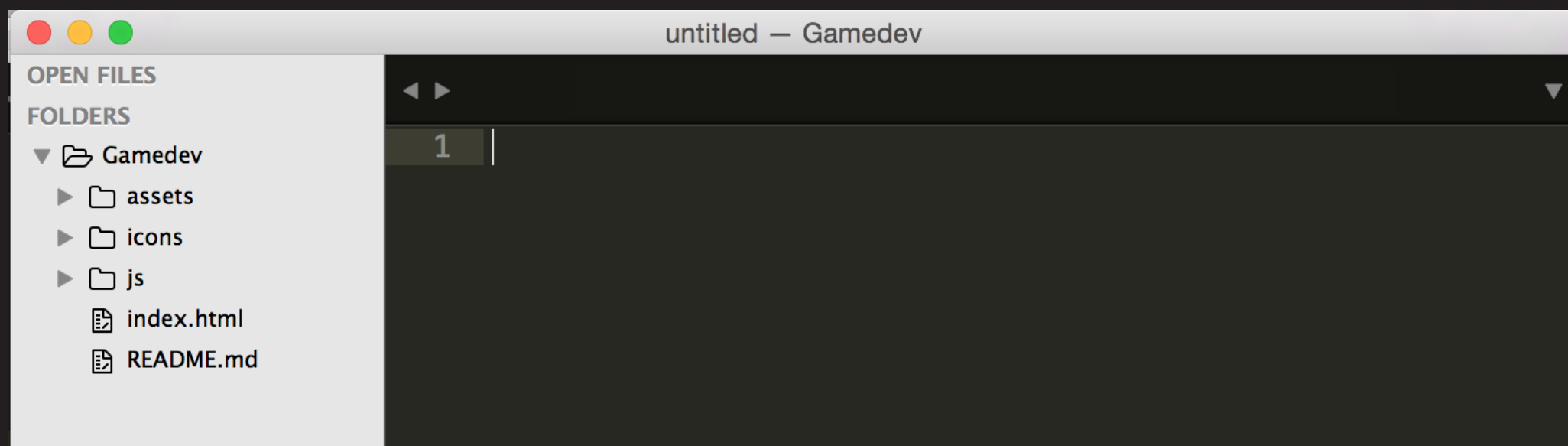
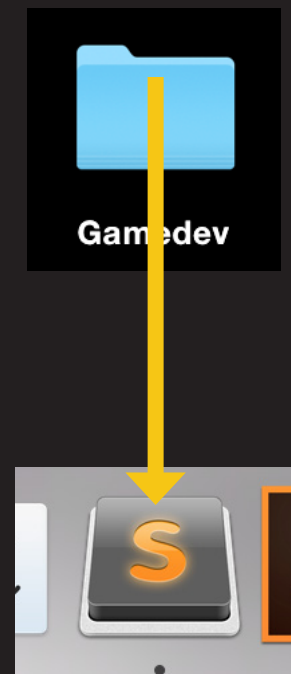
1. Open the **Sublime Text** application.
2. Drag your project folder onto the **Sublime Text** icon in the dock.
3. Your entire folder should open with a sidebar that shows all the project files. If you don't see a sidebar with files, go to **View > Sidebar > Show Sidebar**.



# Workspace Setup

Open the GameDev directory in Sublime Text. Be sure to open the whole folder, not just a single file. To Open the entire folder...

1. Open the **Sublime Text** application.
2. Drag your project folder onto the **Sublime Text** icon in the dock.
3. Your entire folder should open with a sidebar that shows all the project files. If you don't see a sidebar with files, go to **View > Sidebar > Show Sidebar**.





# /GameDev

**/GameDev** | This is your main project directory.

**index.html** | This is the file your game loads into.

**/js** | This directory holds all the javascript files.

**game.js** | This is the main file you will be editing.

**JumpSlide.js** | This is the where the game logic is stored.

**pixi.js** | This is the Game Engine. Leave it alone!

**howler.min.js** | This is the Sound Effects Engine. Leave it alone!

**/assets** | This directory holds all the graphics and sound effects.

# View the Game in Chrome

To view the game in Google Chrome:

1. Open a new **Google Chrome** browser.
2. Type the following url in the address bar:

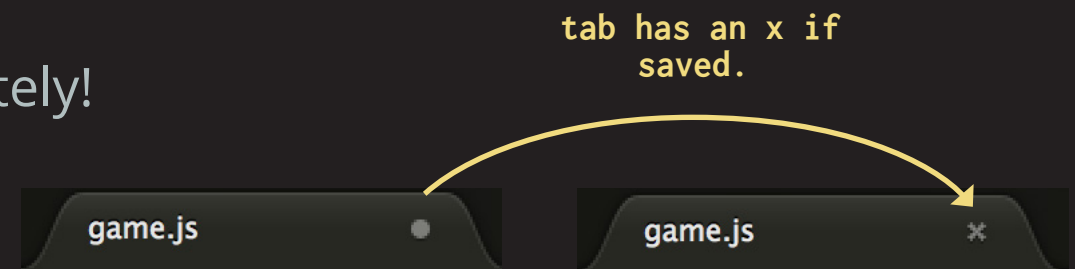
**`http://localhost:8000`**

3. You should see a black background

# General Programming Workflow

Whenever you add a chunk of code, test it immediately!

1. **Save the code you typed.**
2. **Switch to your Google Chrome browser that contains your game.**
3. **Refresh the page by hitting Command-R**



Is something broken? If something is broken, Inspect the Error.

1. **In Chrome, right-click (Control-Click) the game page and choose **Inspect Element** from the dropdown.**
2. **Click **Console** in the nav. If there any errors in your code it will show along with hints to the line number of the error.**

# Add Yourself as the Game Author

1. Click on **index.html**
2. At the top of the file, you will see:  
`<!-- AUTHOR:Your Name -->`
3. Change “Your Name” to your actual name. It should look something like this:  
`<!-- AUTHOR:Kelli Borgonia -->`
4. Save.

# Start Coding!

Follow the slides in this tutorial. After every slide, be sure to test if the code works!  
All coding is done in the **game.js** file.

To begin programming, click the **/js** directory and then the **game.js** file in Sublime Text.  
Study the structure of the file so you have a general idea of where you will be adding code.

# Start the Game!

In the Game.init block, add the following code that starts the game. Soon, we will be adding more code in this block. The **JumpSlide.start();** function call should always be at the END of the block. All new code in the Game.init block should be above this code.

```
GAME.init = function (JumpSlide) {  
  
    // Start the Game  
    JumpSlide.start();  
  
}
```

Test in Chrome!

# Create the Game Level

We will add some elements to the game roughly based on the coordinates on Level Design mockup. Later, you can modify this section to add your own Level. In the **Game.init** block, add some floors and platforms to the map. Always add visible game objects ABOVE the `JumpSlide.start();` code.

```
GAME.init = function (JumpSlide) {  
  
    /* Demo Level */  
  
    // Floors  
    JumpSlide.addPlatform(50, 700, 300, 68);  
    JumpSlide.addPlatform(480, 700, 2000, 68);  
  
    // Short Platform  
    JumpSlide.addPlatform(750, 580, 200, 45);  
  
    // Start the Game  
    JumpSlide.start();  
}
```

Test in Chrome!

# Add Some Coins

In the Game.init block, add some demo coins to the map:

```
// Floors
JumpSlide.addPlatform(50, 700, 300, 68);
JumpSlide.addPlatform(480, 700, 2000, 68);

// Short Platform
JumpSlide.addPlatform(750, 580, 200, 45);

// Coins
JumpSlide.addCoin(850, 517);
JumpSlide.addCoin(850, 660);
JumpSlide.addCoin(1200, 637);
JumpSlide.addCoin(1630, 517);

}
```

Test in Chrome!

You won't see all the coins yet because the game doesn't scroll yet!



# Add a Goal

In the Game.init block, add a Goal to the map:

```
JumpSlide.addCoin(850, 660);  
JumpSlide.addCoin(1200, 637);  
JumpSlide.addCoin(1630, 517);  
  
// Goal  
JumpSlide.addGoal(2100, 665);  
  
}
```

Test in Chrome!

You won't see the goal yet because the game doesn't scroll yet!

# Making the Player Run - Part 1 of 3

In the **Game.loop** block, add the following code to start making the player move.

To make it look like the player is running, we actually need to move all the other game elements to the **left** to simulate the player moving **right**. First, we move the platforms left:

```
GAME.loop = function (JumpSlide) {  
  
    // make the player run right  
    // by looping through each platform  
    JumpSlide.forEachPlatform(function(platform) {  
        // translate it's x position  
        platform.position.x -= JumpSlide.SETTINGS.run_speed;  
    });  
  
}
```

Test in Chrome!

# Making the Player Run - Part 2 of 3

Then, we move the Coins left:

```
// make the player run right
// by looping through each platform
JumpSlide.forEachPlatform(function(platform) {
  // translate it's x position
  platform.position.x -= JumpSlide.SETTINGS.run_speed;
});

// loop through each coin
JumpSlide.forEachCoin(function(coin) {
  // translate it's x position
  coin.position.x -= JumpSlide.SETTINGS.run_speed;
});
}
```

Test in Chrome!

# Making the Player Run - Part 3 of 3

Finally, we make the goal move left:

```
// make the player run right
// loop through each coin
JumpSlide.forEachCoin(function(coin) {
  // translate it's x position
  coin.position.x -= JumpSlide.SETTINGS.run_speed;
});

// loop through each goal
JumpSlide.forEachGoal(function(goal) {
  // translate it's x position
  goal.position.x -= JumpSlide.SETTINGS.run_speed;
});
}
```

Test in Chrome!

You actually cannot get to the goal yet since the character cannot jump and he runs into a platform.

# Making the Player Jump

Find the **GAME.tap** block. Add code to make the player jump:

```
GAME.tap = function ( point ) {  
  
    // check if player touches the top part of the screen  
    if( point.y < JumpSlide.SETTINGS.controls.up ){  
  
        // make player jump  
        JumpSlide.player.jump();  
    }  
}
```

Test in Chrome!

Now if you click above the center of the screen, the character jumps.

# Making the Player Slide

Find the **GAME.touch\_start** block. Add code to make the player slide:

```
GAME.touch_start = function ( point ) {  
  
    // check if player touches bottom part of screen  
    if( point.y > JumpSlide.SETTINGS.controls.down ){  
  
        // make player start sliding  
        JumpSlide.player.slide();  
    }  
}
```

Test in Chrome!

Now if you click below the center of the screen, the character crouches down.

# Making the Player Stop Sliding

Find the **GAME.touch\_end** block. Add code to make the player stop sliding when you release the mouse button after initiating a slide action.

```
GAME.touch_end = function ( point ) {  
  
    JumpSlide.player.stop_sliding();  
  
}
```

Test in Chrome!

When the character is sliding, he now stands up again when you release the mouse button.

# Add the Ability to Collect Coins

Back in the **GAME.loop** block, find the section starting with `// loop through each coin` and add the following code after the line ending with `.run_speed;`

```
// loop through each coin
JumpSlide.forEachCoin(function(coin) {
  // translate it's x position
  coin.position.x -= JumpSlide.SETTINGS.run_speed;

  // check if player is touching a coin
  if(JumpSlide.player.check_collision(coin)){

    // collect the coin to score
    JumpSlide.collectCoin(coin);
  }
});
```

Test in Chrome!



# Add Collect Coin SFX

After the `JumpSlide.collectCoin(coin);` line, add the following code to add the coin sfx:

```
// check if player is touching a coin
if(JumpSlide.player.check_collision(coin)){

    // collect the coin to score
    JumpSlide.collectCoin(coin);

    // play coin sound effects
    JumpSlide.sfx.coin.play();
}
});
```

Test in Chrome!

# Add the Victory Condition

In the **GAME.loop()** block, find the block starting with `// loop through each goal` and add the following code after the line ending with `.run_speed;`

```
// loop through each goal
JumpSlide.forEachGoal(function(goal) {

    // translate it's x position
    goal.position.x -= JumpSlide.SETTINGS.run_speed;

    // victory condition
    if(JumpSlide.player.check_collision(goal)){
        JumpSlide.game_win();
    }

});
```

Test in Chrome!


# Add the Lose Condition

In the **GAME.loop()** block, add code after the victory condition:

```
goal.position.x -= JumpSlide.SETTINGS.run_speed;
```

```
// victory condition  
if(JumpSlide.player.check_collision(goal)){  
    JumpSlide.game_win();  
}
```

don't hit enter.  
this is a  
continuous line



```
// check if player falls, then lose game  
if( JumpSlide.player.position.y >= JumpSlide.SETTINGS.  
ipad_dimensions.height ){  
    JumpSlide.game_lose();  
}
```

```
});
```

Test in Chrome!

You might not see anything change, however this adds logic that causes the player to lose if the character falls off the screen.

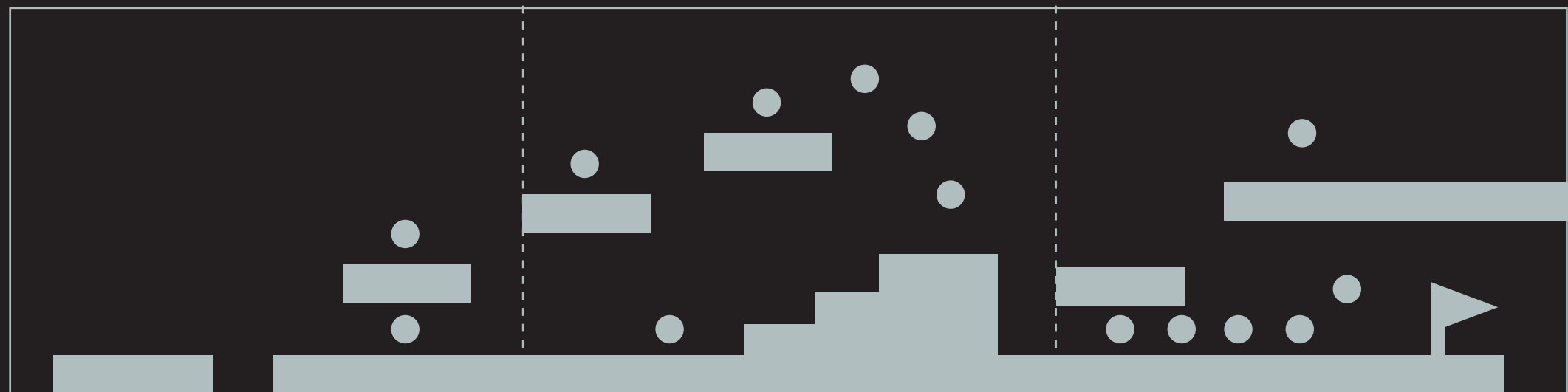
# Build a Longer Game Level

You're done with the basic game logic! Go back to the **GAME.init** block and add more walls, floors, and coins to make a full game level. Use Graph Paper to plan out the level. It is much easier to be creative and plan it out on paper than while you are programming.

Refer to the API guide for how to use all the game settings and functions:

<https://github.com/GomaGames/JumpSlide>

Be sure to adjust the speed settings **early in your planning stages**. If you adjust them later you will find that speed greatly affects how far apart objects are to the character that is jumping and sliding.



# Experimentation

If you want to be experimental, you can swap out the graphics & sound. The entire set of game graphics and Adobe Illustrator mockups can be downloaded at <http://stem.gomagames.com>. All the game art and sfx were found on [opengameart.org](http://opengameart.org).

If you want to be really adventurous, you can modify the code in JumpSlide.js. Careful! Be sure you know what you are doing in here and make a lot of backups...

When you done building your Game Level, move on to **Game Deployment** ----->

# Deployment

When you are ready to Publish your game:

1. Double click the **deploy.game.command** script on your desktop.
2. If it asks, type **yes** and hit **Enter**.
3. Navigate to **<http://stem2015.gomagames.com/>** in your browser or on an iPad.  
Your game is now published online.
4. To play the game on iPad, open the game in Safari, click the **Bookmark Icon** and **Save to your Homescreen**.

**BUG!** Unfortunately, the sound doesn't work on iPad when you Save to Homescreen :(

Note that authentic iOS app deployment is a much longer process. This quick deployment method simulates publishing a mobile app without the lengthy deployment process.