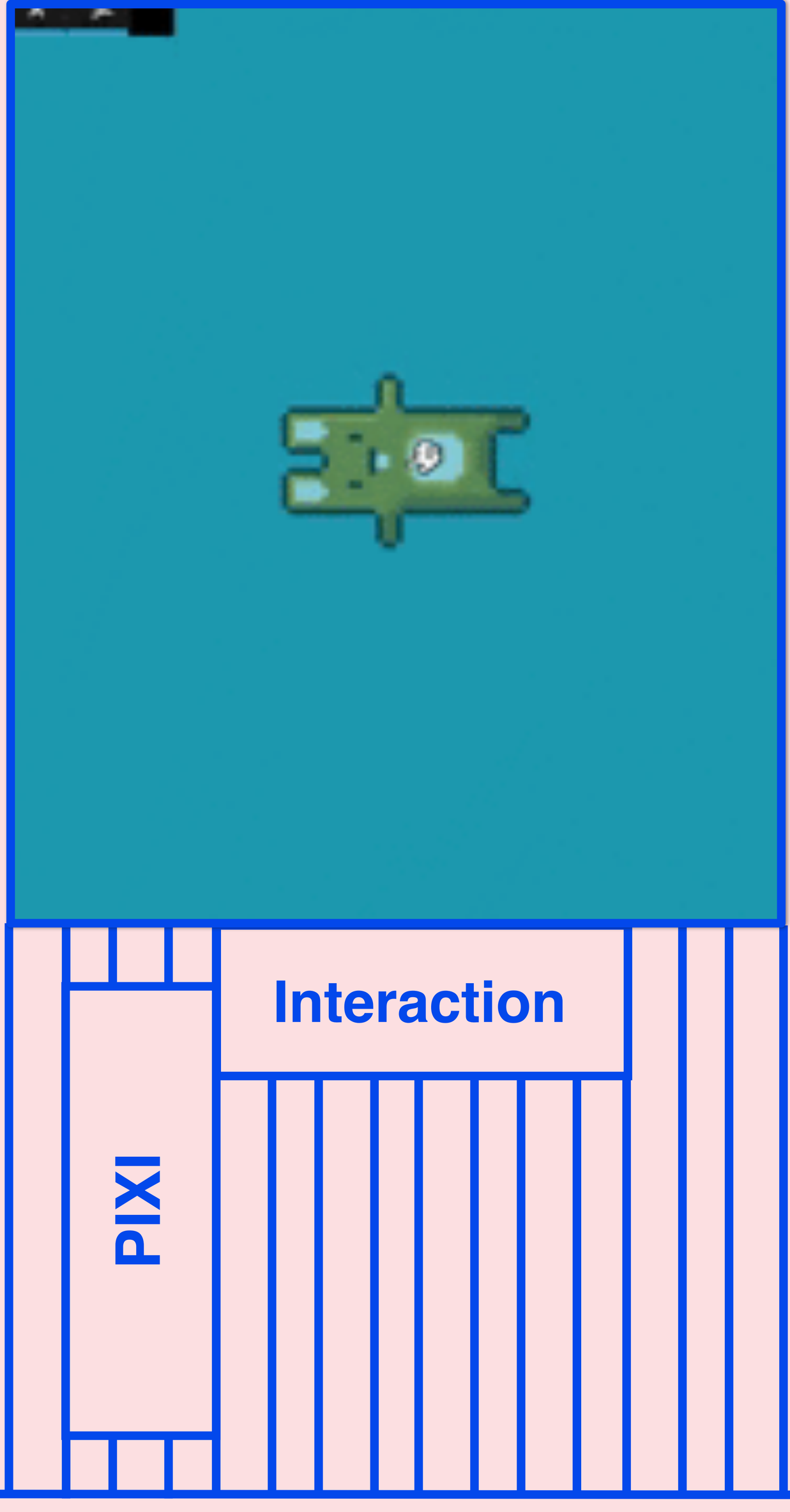
Week 3

PIXI: Interaction and MVC


Interaction
PIXI

Pointer Callbacks

Entities in Pixi have several event callbacks based on pointer actions (mouse on desktop or touch on mobile)

```
const button = new PIXI.Graphics();

// make the button interactive...
button.interactive = true;
button.buttonMode = true;
button
    .on('pointerdown', onButtonDown)
    .on('pointerup', onButtonUp)
    .on('pointerupoutside', onButtonUp)
    .on('pointerover', onButtonOver)
    .on('pointerout', onButtonOut);
```

Pointer Down

Event for the pointer being pressed or clicked on the entity

```
button.on('pointerdown', onButtonDown)

function onButtonDown() {
    this.isdown = true;
    button.clear()
....
    button.beginFill(0xff00ff, 0.5)
...
}
```

Pointer Up

Event for the pointer being pressed or clicked on the entity

```
button.on('pointerup', onButtonUP'p)

function onButtonUp() {
    this.isdown = false;
    button.clear()
....
    button.beginFill(0xff00ff, 1)
...
}
```

Pointer Over

Event for the pointer being pressed or clicked on the entity

```
button.on('pointerover', onButtonOver)

function onButtonOver() {
    this.isdown = false;
    button.clear()
....
    button.beginFill(0xff00ff, 1)
...
}
```

Pointer Out

Event for the pointer being pressed or clicked on the entity

```
button.on('pointerout', onButtonOut)

function onButtonOut() {
    this.isdown = false;
    button.clear()
....
    button.beginFill(0xff00ff, 1)
...
}
```

PIXI

dat.GUI

message dat.gui

maxSize 5.59

growthSpeed 0.2

speed 0.4

noiseStrength 10

displayOutline

Explode!

Close Controls

dat.GUI

dat.GUI enables us to make debug UI for real-time prototyping without having to change variables or refresh the browser

```
> npm install —-save dat.gui
```

```
// Creating a GUI and a subfolder.

import * as dat from 'dat.gui';

let gui = new dat.GUI();

let folder1 = gui.addFolder('My folder');
```

Adding parameters

Once we have a gui object, we can start adding properties of objects to our panel

```
// Add a string controller.
let person = {name: 'Sam'};
gui.add(person, 'name');

// Add a number controller slider.
let car = {speed: 45};
gui.add(person, 'age', 0, 100);
```

Adding parameters

We can also add colors to our gui

```
let palette = {
  color1: '#FF0000', // CSS string
  color2: [ 0, 128, 255 ], // RGB array
  color3: [ 0, 128, 255, 0.3 ], // RGB with alpha
   color4: { h: 350, s: 0.9, v: 0.3 } // Hue, saturation, value
};
gui.addColor(palette, 'color1');
gui.addColor(palette, 'color2');
gui.addColor(palette, 'color3');
gui.addColor(palette, 'color4');
```

Callbacks

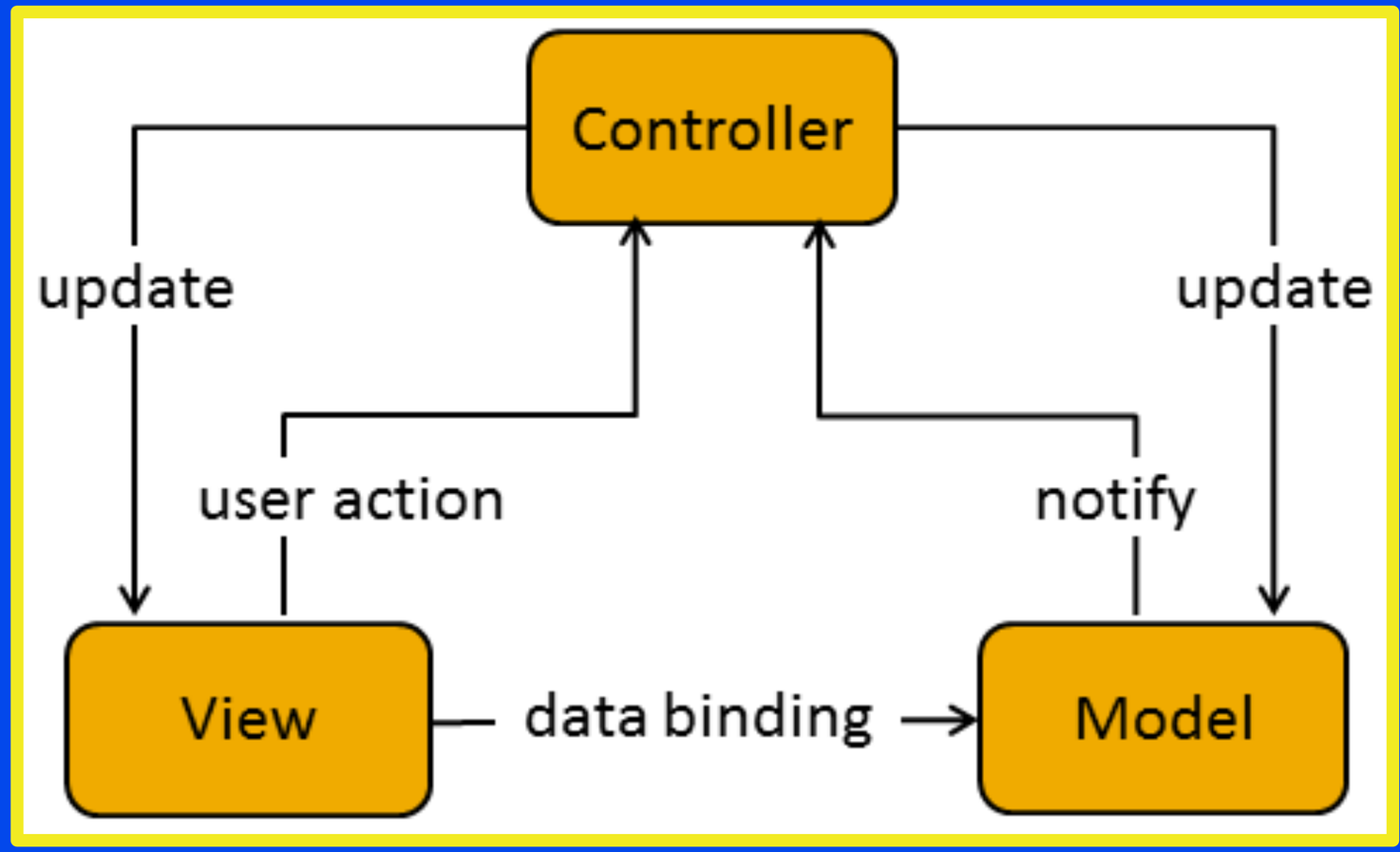
If we want to tie in some other logic to respond to our controllers, we can also add callbacks for onChange and onFinishChange

```
gui.addColor(palette, 'color1')
   .onChange( changeCallback )
   .onFinishChange( finishCallback )

function changeCallback() {. . .}
function finishCallback() {. . .}
```

Model–View–Control

Model–View–Control, or MVC, is a design pattern for managing larger apps or software projects

- Model: the data that informs the application
- View: how the data is drawn to the screen
- Control: mediates changes between the view and model



Model

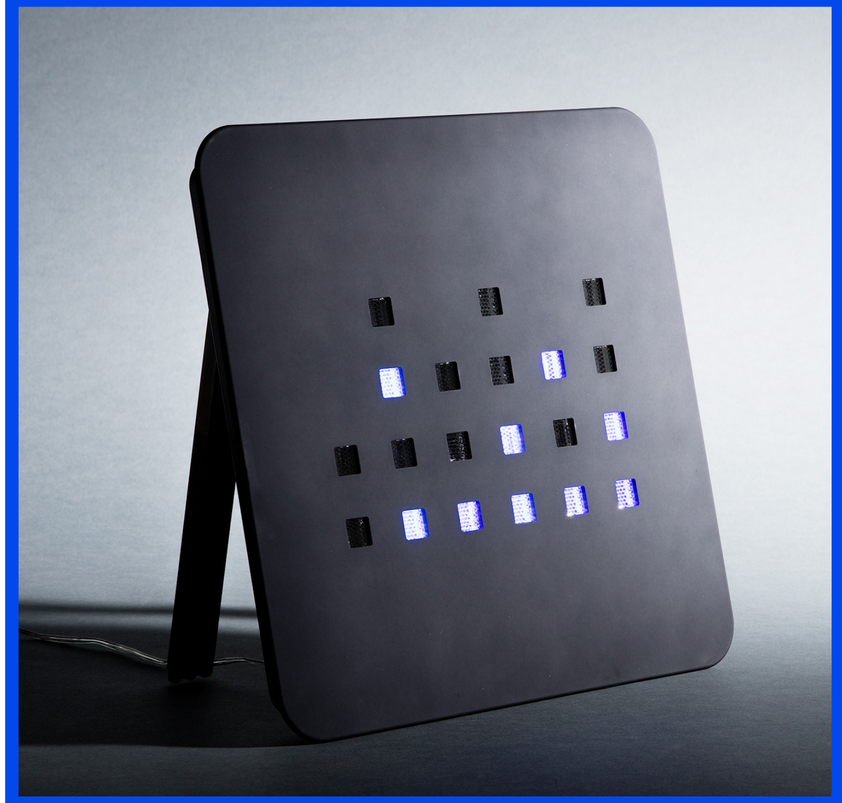
We can make ourselves a class which has all the global data that we need to inform our app. Most models conform to the Singleton pattern, which makes sure that there’s only one of in in our app

```
class Model{
  private static instance: Model;

  private constructor(){
    if(Model.instance) {
      Return Model.instance
    }
    Model.instance = this
  }

  public static get Instance()
  {
    return this._instance || (this._instance = new this());
  }
}
```

View

In Pixi, we can make a class that acts as a Container for it’s child objects and has access to our model

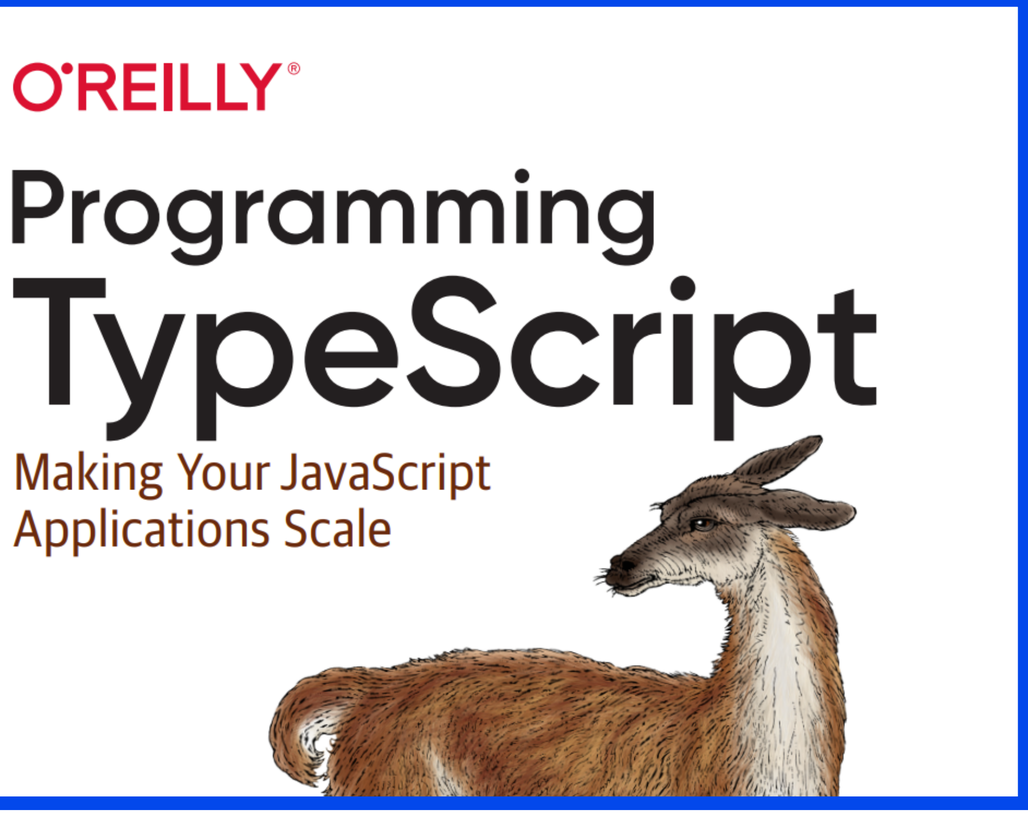
```
Class Scene {
  private model: Model;
  public container: Container;
  constructor(model:Model) {…}
  update() {…}
}
```



Homework for next week

A Novel Clock

Design a 'visual clock' that displays a novel or unconventional notion of time. Your clock should appear different at all times of the day, and it should repeat its appearance every 24 hours (or other relevant cycle, if desired). **Challenge yourself to convey time without numerals.**



Homework for next week

Readings

Read Chapter 5 of Programming Typescript