Week 2-1

PIXI: drawing, sprites, and scene-graph

What’s Pixi.js all about?

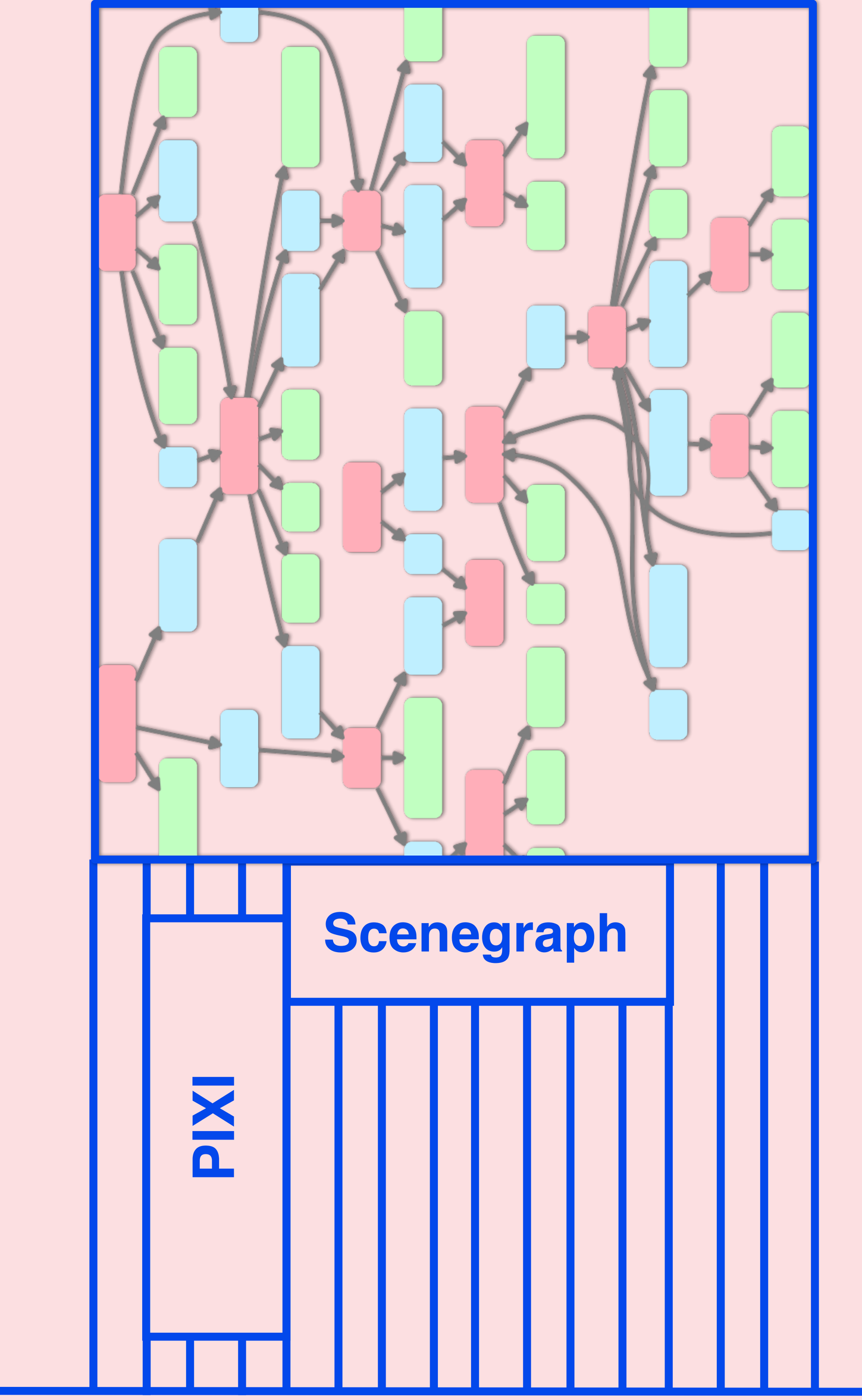
PIXI.js is a library built on top of webGL,

- a ‘Scenegraph’-style workflow for managing nested systems and entities, much like three.js, unity, and other contemporary
- A drawing api for creating procedural graphics
- Strong support for creating and manipulating sprites from images

Setting up our development environment

Clone the QuickStart repo at:

- git clone [repo url]
- cd [path/to/repo]
- npm install
- npm run dev
- in a separate terminal, npm run watch


Scenegraph
PIXI

Scenegraph

A Scenegraph is a way of organizing software where every object or ‘entity’ is a child or parent of another entity.

Stage

The stage of a Pixi app is the root of the Scenegraph and contains all the other entities

Parent/child heirarchy

In a Scenegraph, transforming or manipulating a parent object often carries down to it’s child entities

Containers

Pixi has objects which are just meant to act as parent to other objects called ‘containers’ this makes manipulating batches of objects much simpler

Sprites

PIXI

Loading a Texture

Before a texture can be applied to a sprite, it will need to be to be loaded into your app from a path or url (preferably a local path though)

```
const load = (app: PIXI.Application) => {
  return new Promise<void>((resolve) => {
      app.loader.add(
          'assets/hello-world.png'
          ).load(() => {
          resolve();
      });
  });
};
```

Assigning a texture

Once the texture is loaded, it can then be assigned to a sprite at instantion

```
// Load assets
  await load(app);
  let sprite = new PIXI.Sprite(
    app.loader.resources['assets/hello-world.png'].texture
  );
```

Sprite

A sprite is an entity made from an image. Pixi can use the image as a texture on a quad to display and manipulate in the scene. Textures are loaded into sprites as the app is starting up in the ‘load’ function of our QuickStart repo

Position

```
sprite.x = 100

sprite.y = 240
```

Scale

```
sprite.scale.set(  x, y);
```

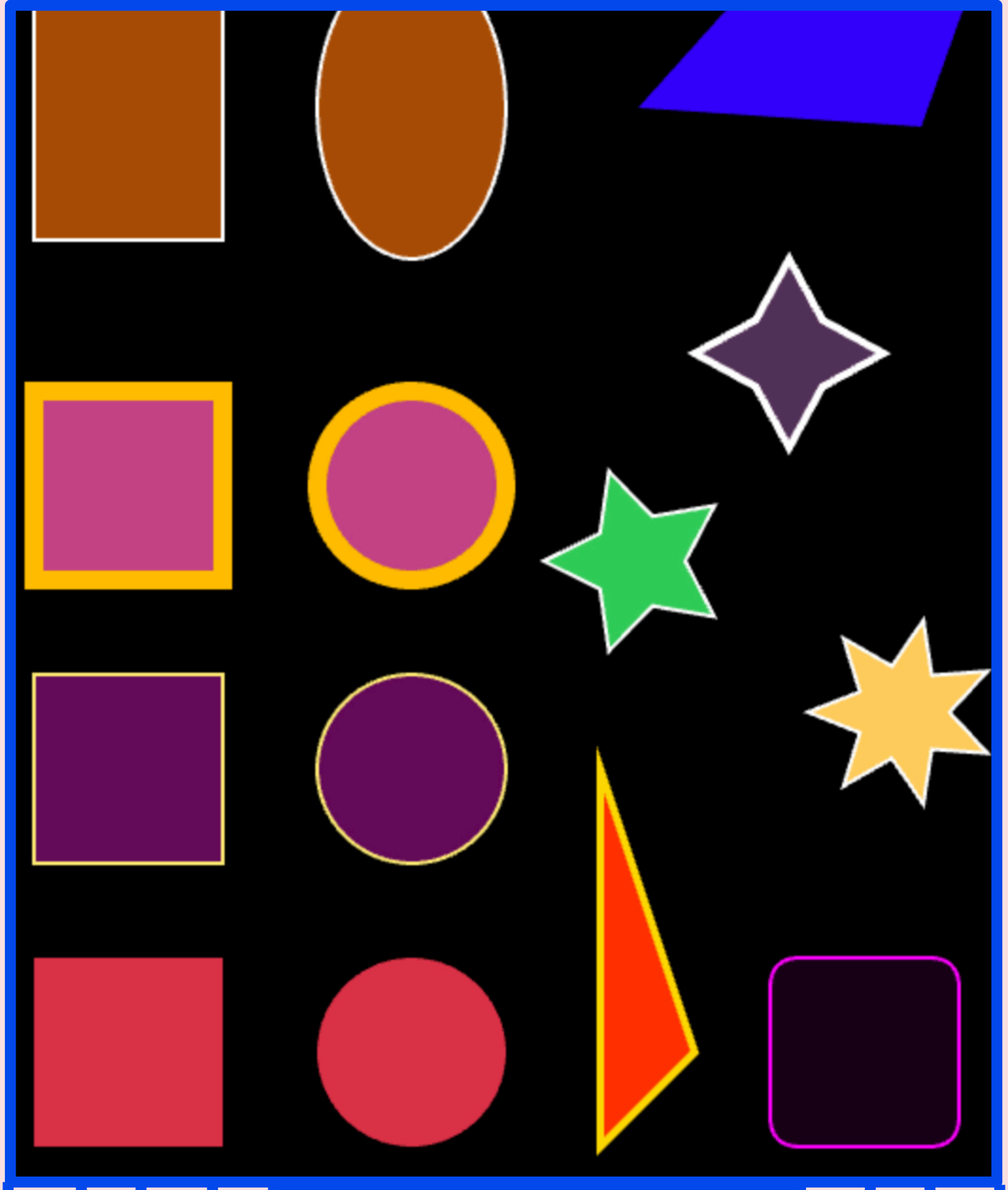
Rotation

The rotation in degrees of your sprite

```
sprite.angle = 90
```

Anchor

the axes around which the sprite scales and rotates



Graphics

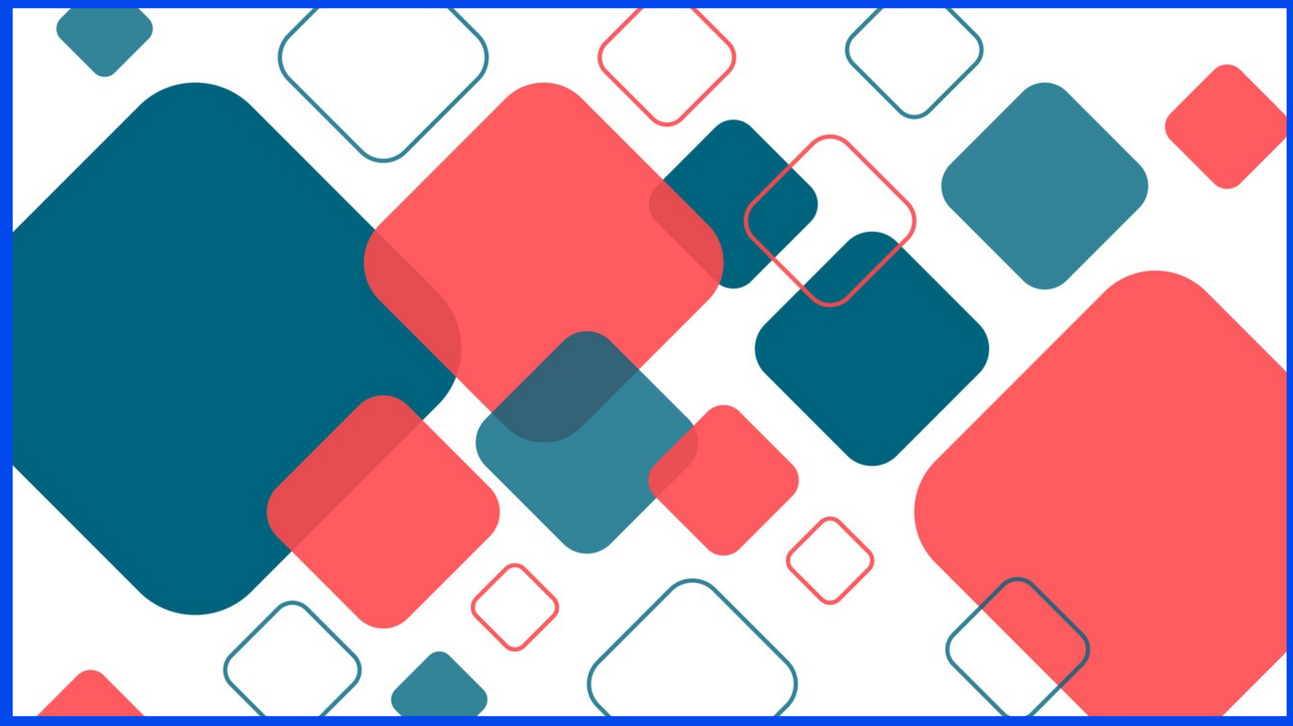
PIXI

Circles

```
let graphics = new PIXI.Graphics();
//like stroke() in p5, (weight, color, opacity)
graphics.lineStyle(2, 0xFEEB77, 1);
//like fill() in p5, (color, opacity)
graphics.beginFill(0x650a5a, 1);
graphics.drawCircle(250,250,50);
```

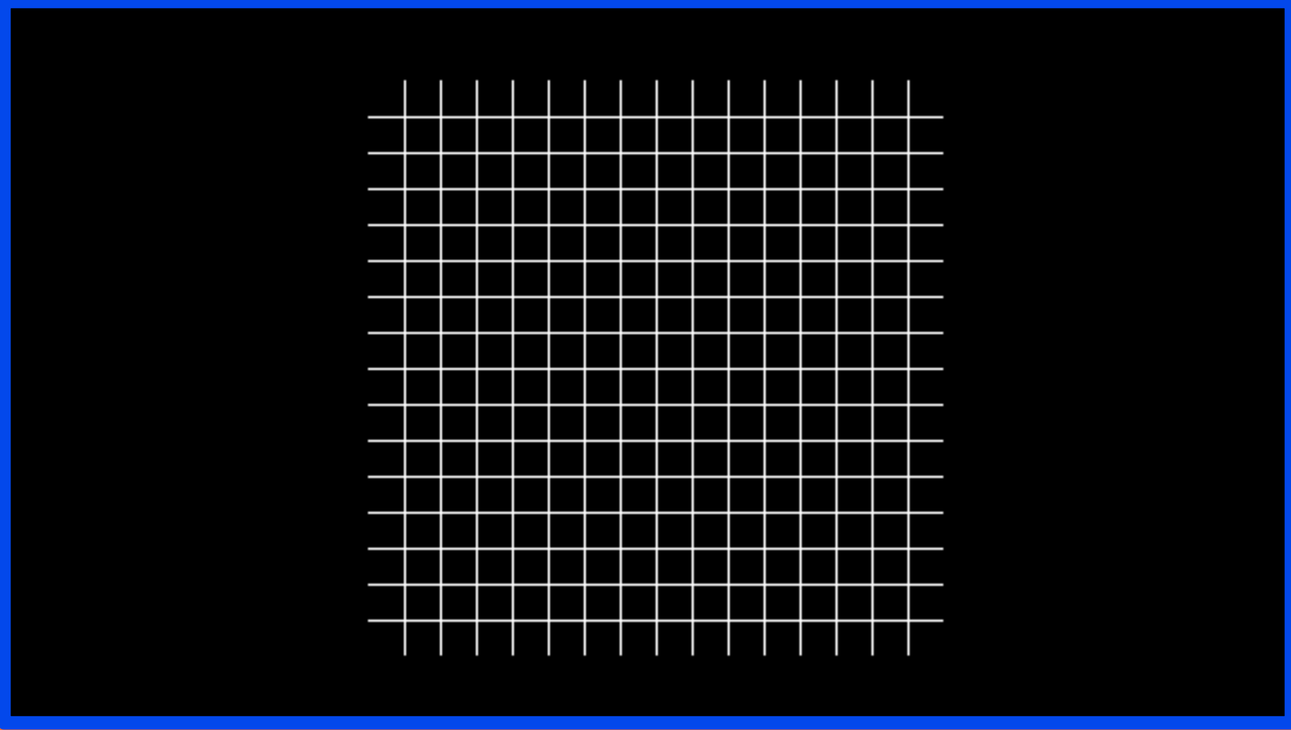
Rectangles

```
graphics.beginFill(0xFFFF00);

// set the line style to have a width of 5 and set the color
to red

graphics.lineStyle(5, 0xFF0000);

// draw a rectangle

graphics.drawRect(0, 0, 300, 200);

// draw a rounded rect

graphics.drawRoundedRect( x, y, width, height, degrees )
```



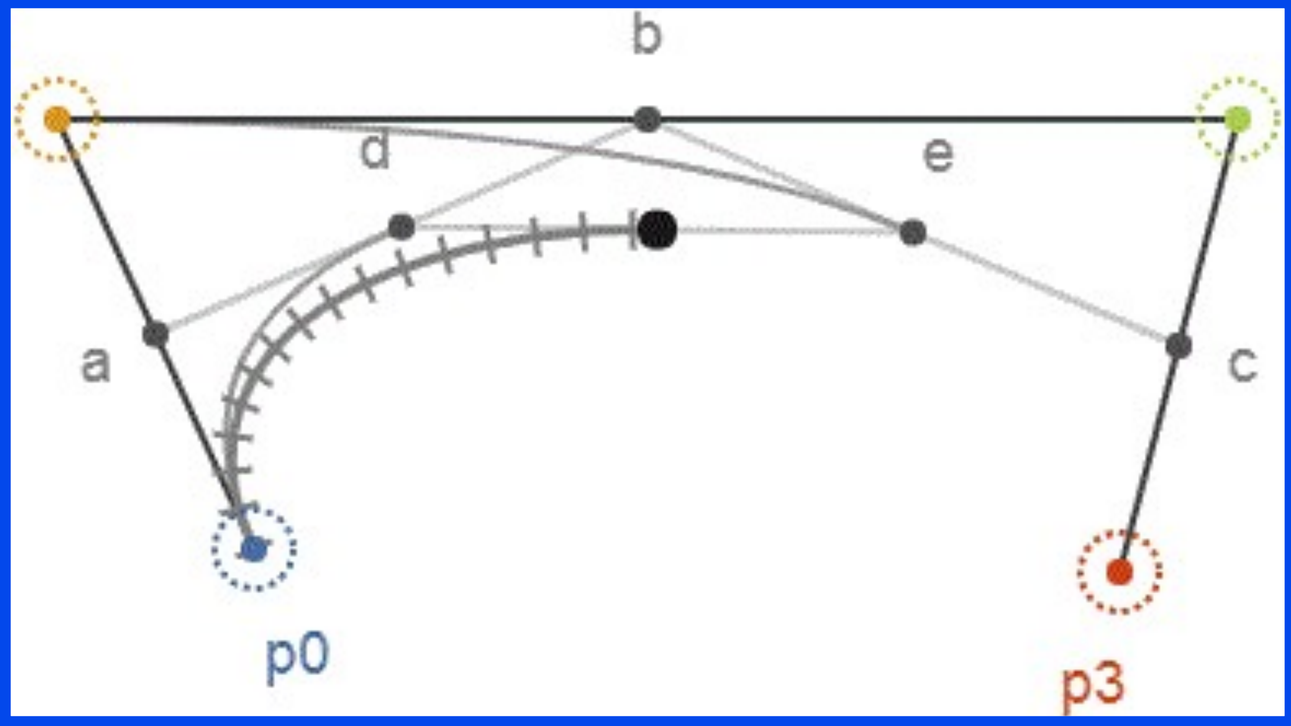
Lines

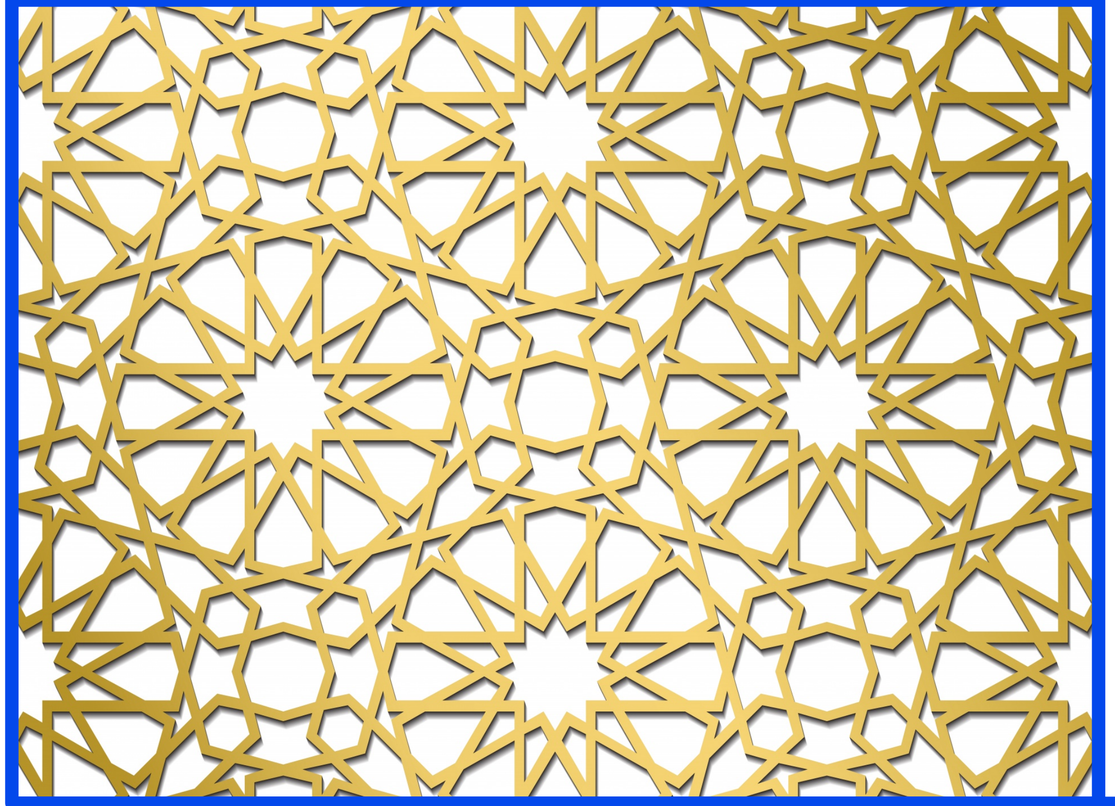
```
const realPath = new PIXI.Graphics();
realPath.lineStyle(2, 0xFFFFFF, 1);
realPath.moveTo(0, 0);
realPath.lineTo(100, 200);
realPath.lineTo(200, 200);
realPath.lineTo(240, 100);
realPath.position.x = 50;
realPath.position.y = 50;
```



Lines, Bezier

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

bezier.lineStyle(5, 0xAA0000, 1);

// starts at 0,0 by default

// add pairs of numbers for control
points

bezier.bezierCurveTo(100, 200, 200, 200,
240, 100);
```

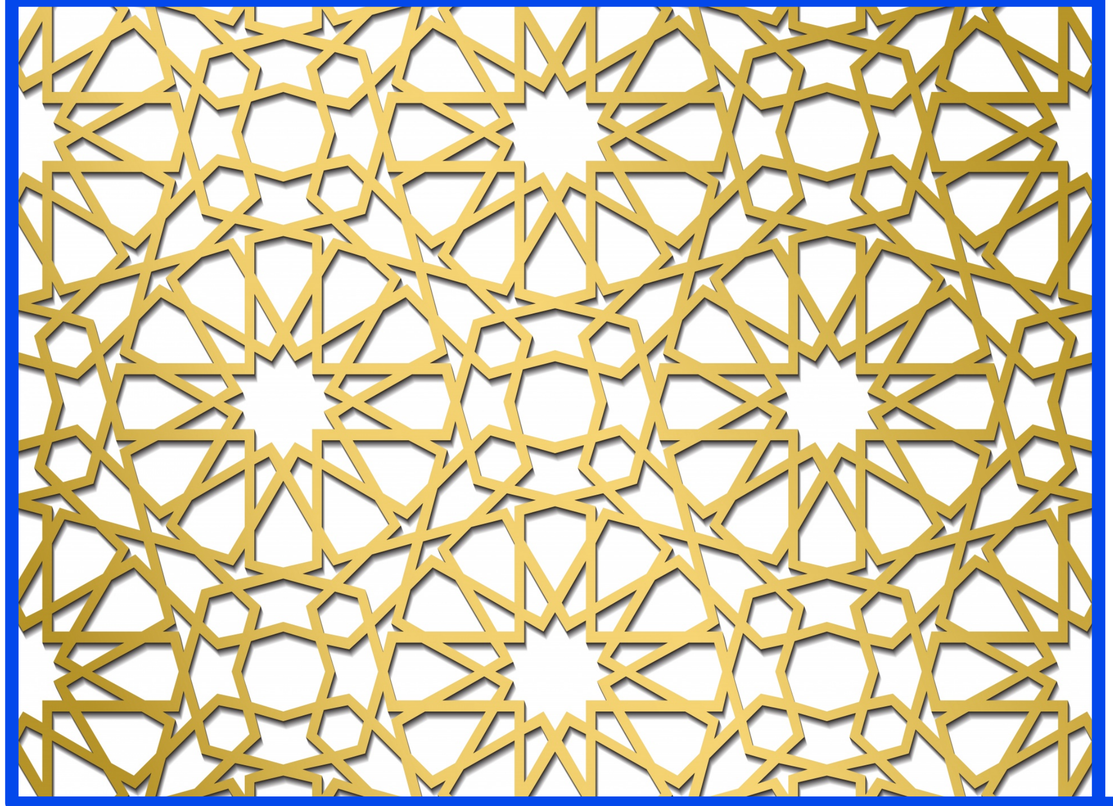




Homework for next week

Iterative Pattern

write code to generate a tiling pattern or textural composition. Give consideration to aesthetic issues like symmetry, rhythm, color; detail at multiple scales; precise control of shape; and balance between organic and geometric forms. your pattern should be designed so that it can be infinitely repeated or extended. It may be helpful to use structures like for of/in/each loops to take care of repetitions.



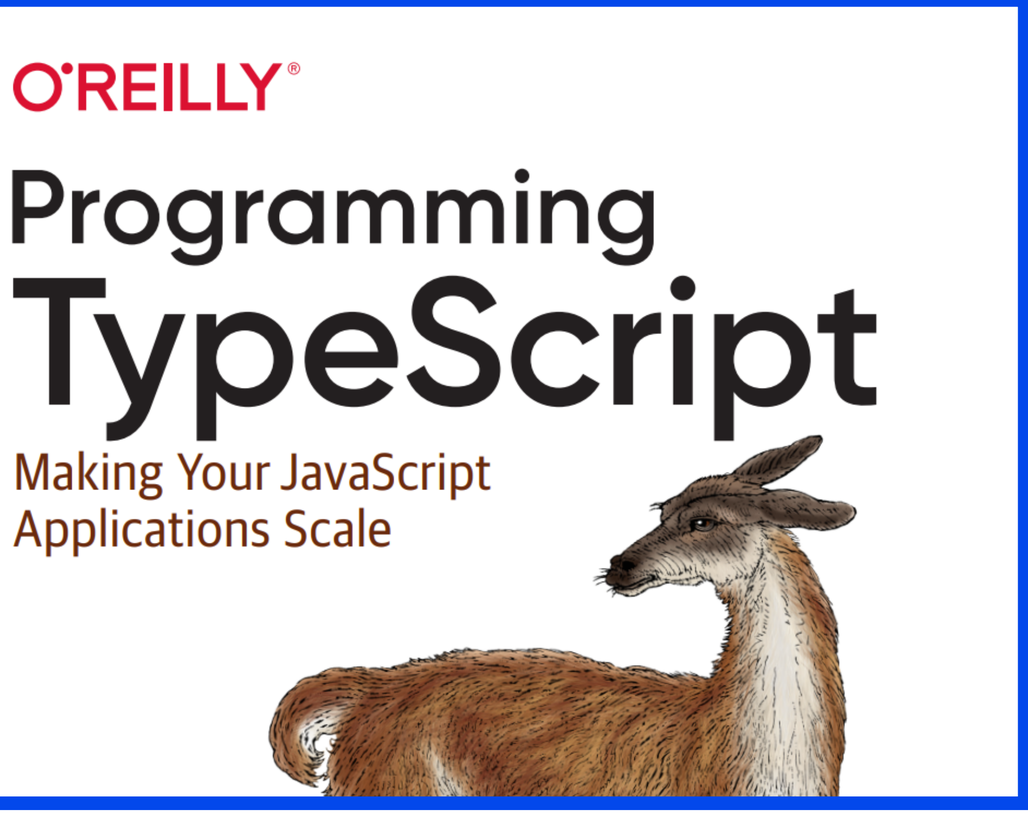
Homework for next week

github example: https://github.com/thatcotter/homework-repo-example

this repo is an example of how you can upload your work and host it live for

please provide a link to your repo and the live version of it for your weekly assignments on Brightspace and in the weekly homework slack threads

if you aren't familiar with how to host your work, GitHub Pages can be a quick way to get off the ground https://dev.to/github/how-to-use-github-pages-to-host-your-website-even-with-multiple-repos-27k2



Homework for next week

Readings

Read Chapter 4 of Programming Typescript

Optional readings (Links on Brightspace):

- The Book of Shaders, Chapters 0-9