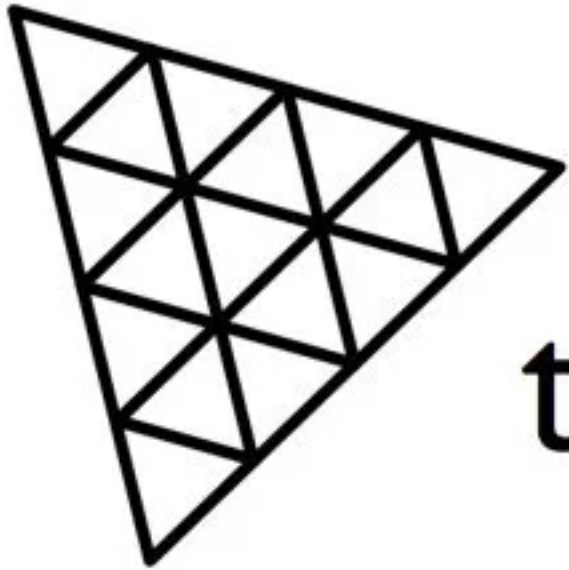


# Introduction to 3D Graphics with three.js

Adapted from slides by Crystal Hess

# Learning Goals

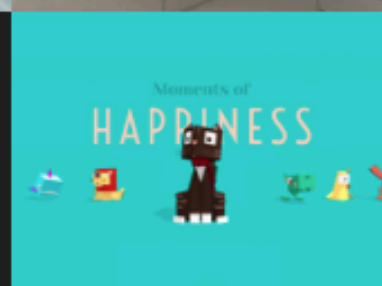
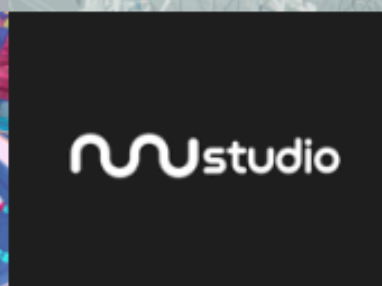
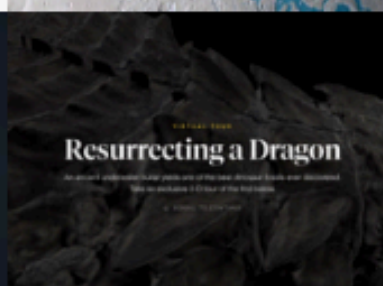
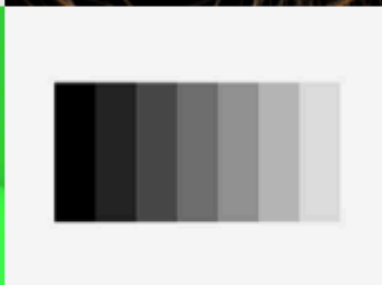
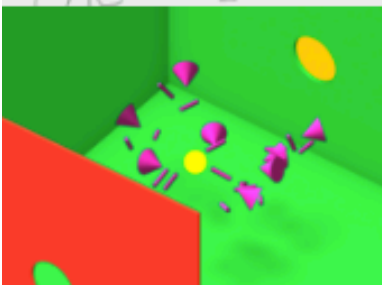
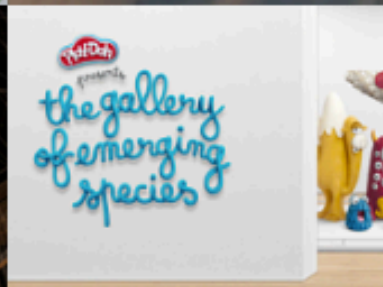
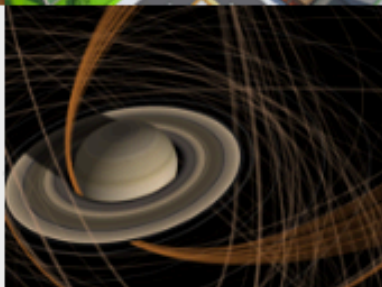
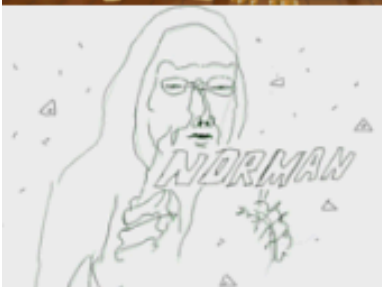
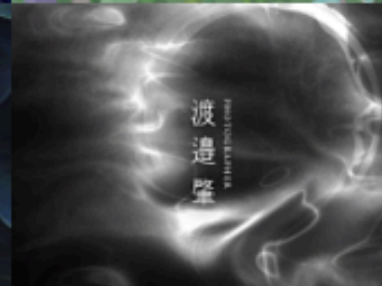
- What is three.js?
- What is “the documentation”?
- How does the coordinate system work in 3D?
- What can a for-loop be used for?
- Note similarities between programming languages used to code in three.js (JavaScript) and Arduino (c/c++)



three.js

# What is three.js?

- JavaScript language
- Runs in a web browser
- Uses WebGL to run graphics calculations off of the graphics card
- Relatively accessible and (mostly) platform-agnostic 3D graphics



# Basic Lingo

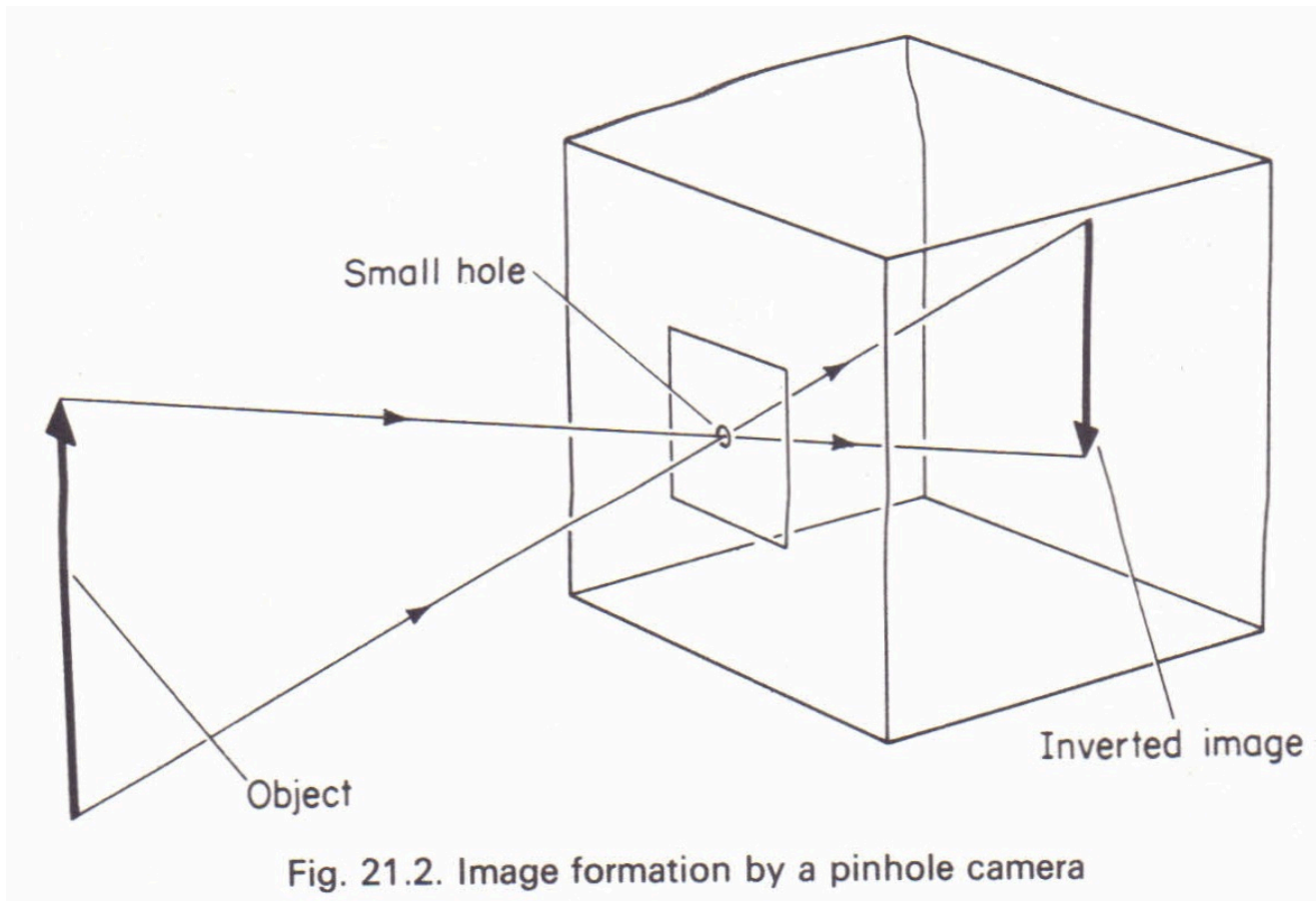
- Scene
  - Where the objects exist
- Camera
  - How we view the scene
- Lightning
  - How anything is visible
- Renderer
  - The technical details for how a virtual scene becomes something we see of the screen

# Example: Basic

- Download the web->three\_projects->00\_basic folder
- Open three.html with a browser
- Open project.js with a text editor
- **Play with the code!**
  - Try changing some camera numbers
  - What happens if you set the mesh material to wireframe: false?
- How does js look similar to the Arduino language?
- How does it look different?

# The Camera

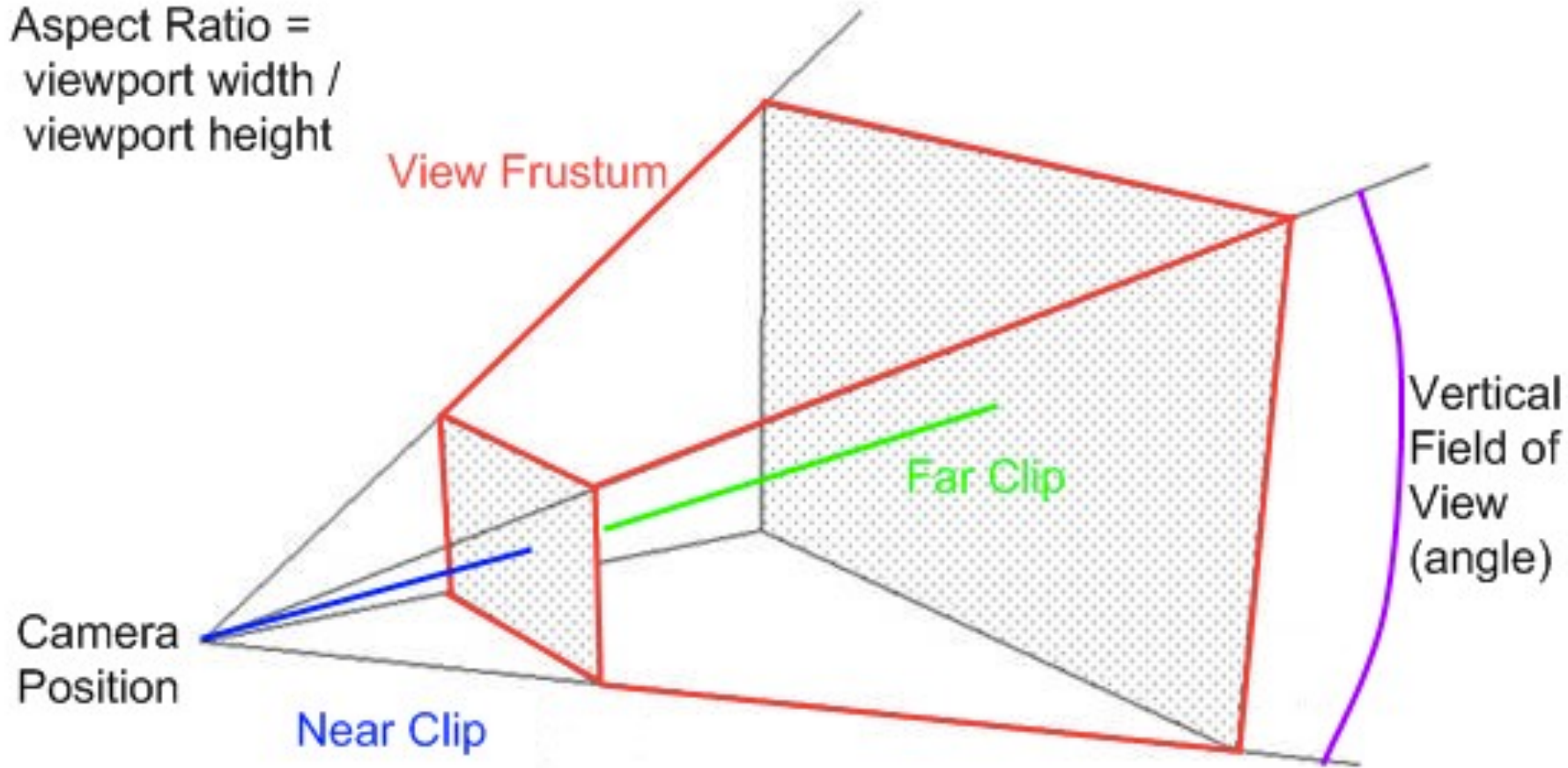
- Graphics cameras are modeled like old pinhole cameras





# The Synthetic Camera

Aspect Ratio =  
viewport width /  
viewport height





Type to filter

x

## Manual

### Getting Started

[Creating a scene](#)  
[Import via modules](#)  
[Browser support](#)  
[WebGL compatibility check](#)  
[How to run things locally](#)  
[Drawing Lines](#)  
[Creating Text](#)  
[Loading 3D Models](#)  
[Migration Guide](#)  
[Code Style Guide](#)  
[FAQ](#)  
[Useful links](#)

### Next Steps

[How to update things](#)  
[Matrix transformations](#)  
[Animation System](#)

### Build Tools

[Testing with NPM](#)

## Reference

# The three.js API

Remember the  
Documentation!

[https://threejs.org/  
docs/index.html](https://threejs.org/docs/index.html)

# Example: Lighting

- Now get web->three\_projects->01\_sphere\_with\_lighting folder
- What's different?
- Try commenting out the ambient light in the code
- Turn that back on and try it without the point light code
- Add a new point light
- Play with the sphere's metal-ness and rough-ness values (range 0.0-1.0)
  - What do you think they do?
  - Try it with the flat shading turned off
- **Added Fun**
  - Replace the point light with a directional light
  - Make the sphere use a different material than MeshStandardMaterial

# Light Types


- Ambient
  - That dim glow that's everywhere
- Point Light
  - Just like a lightbulb
- Directional Light
  - Light coming from one direction, everywhere
  - Looks like daylight
- Spotlight
  - Just like you would think
  - What would this be most similar to?

# Material Lighting Properties

- Diffuse (aka “roughness”)
  - Like a matte finish
  - Brighter when the light source is perpendicular to the surface
- Specular (aka “metalness”)
  - Makes a thing look shiny
  - Brighter when the light source is at the same angle to the surface as the camera
- Together with ambient lighting these are commonly known as the “Phong” model of graphics lighting
- Other lighting stuff (these mostly require ray-casting):
  - Reflection
  - Refraction
  - Caustics

# For Loops

```
for (var i = 0; i < 5; i++) {  
    addSphere(-4 + 2*i, 0, 0, 1, 2);  
}
```

- Get the 02 project, Loops and Events
- Consider the code here 
  1. Can you make the line of spheres longer?
  2. Can you make the spheres stack vertically instead of line up?
  3. Can you space the spheres farther apart?
  4. Can you make the spheres larger?
  5. Can you make each one larger than the last?
- **Added Fun**
  - Make each sphere be a different random color
  - Make each object be a random shape