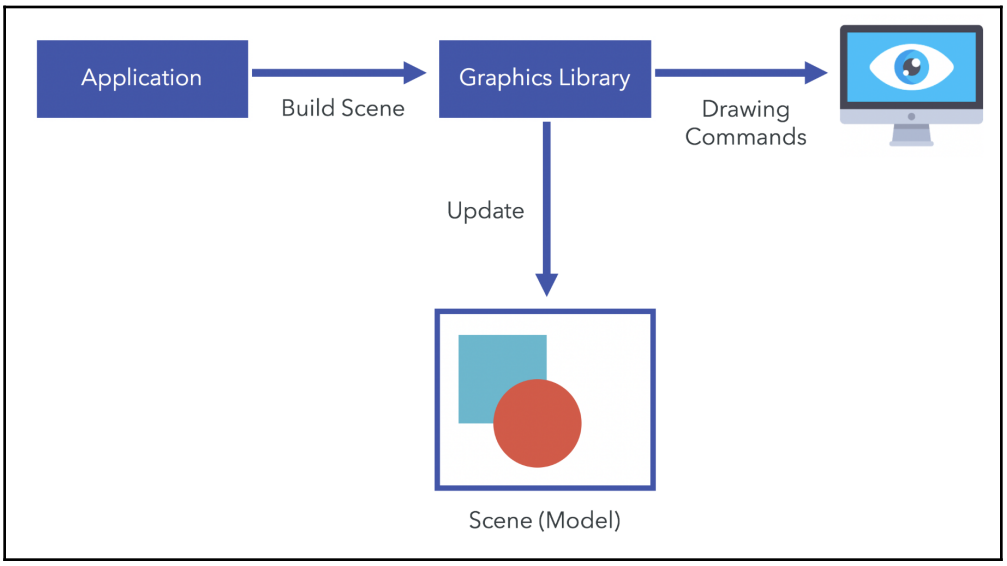
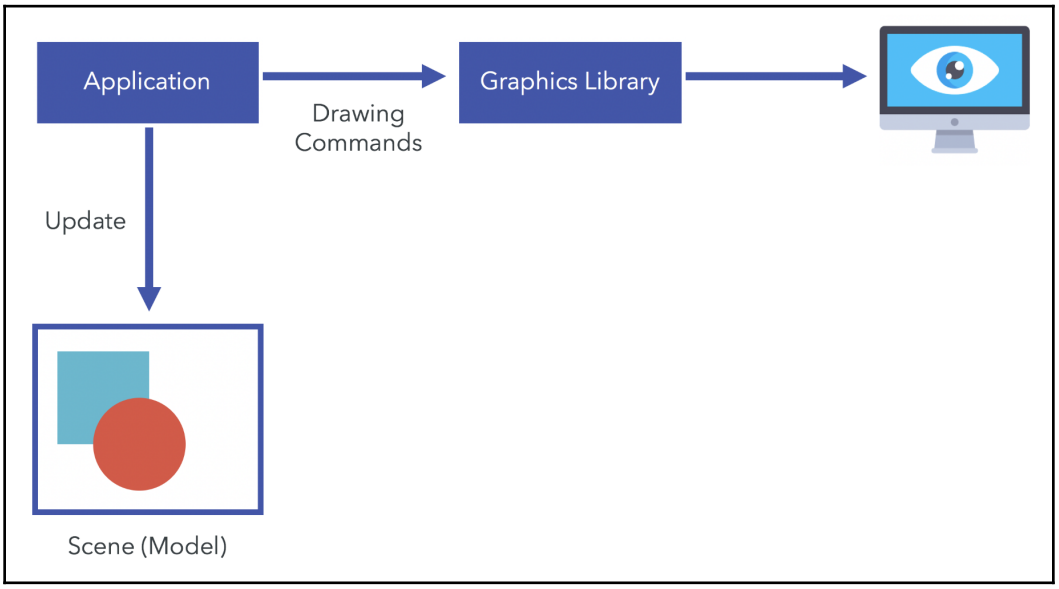
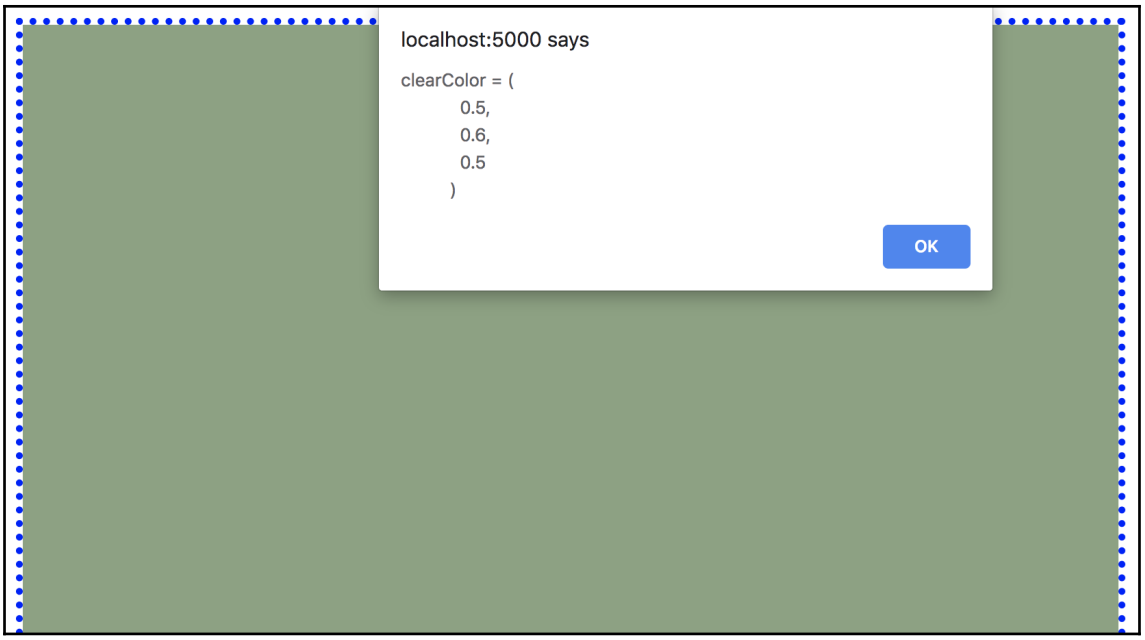
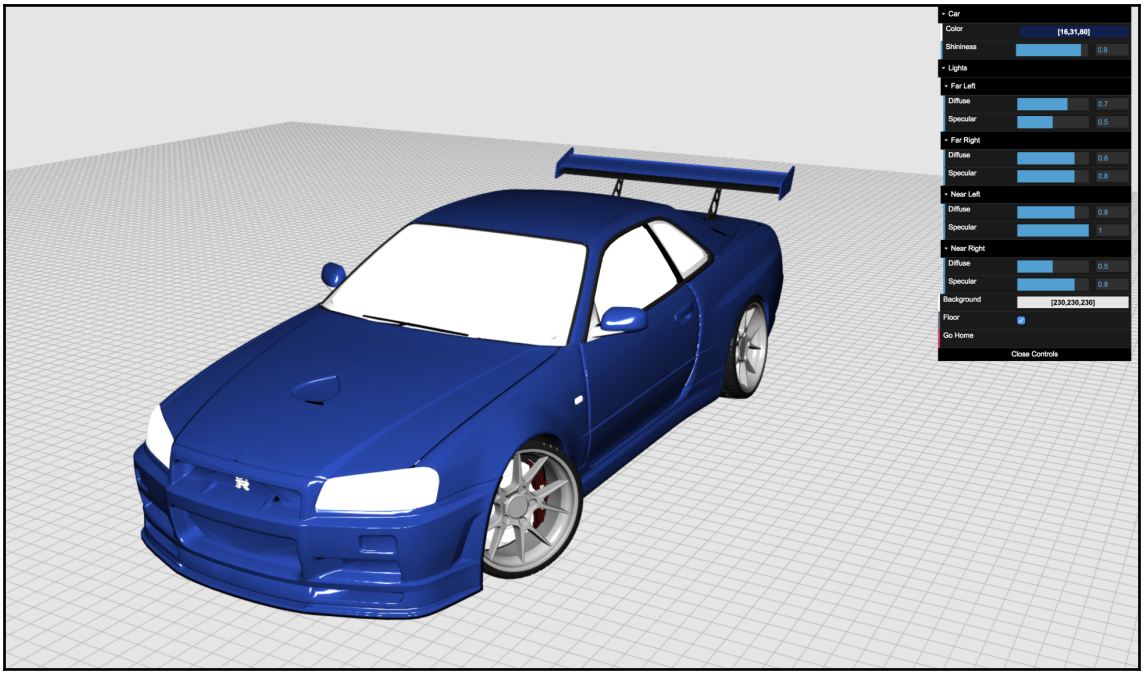
Chapter 1: Getting Started



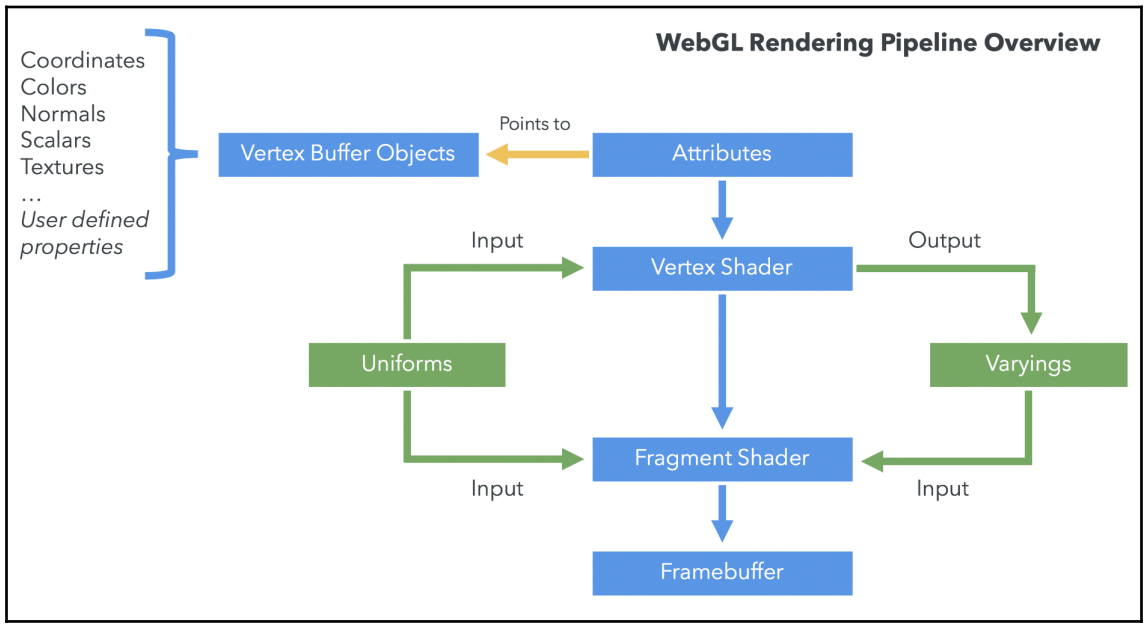


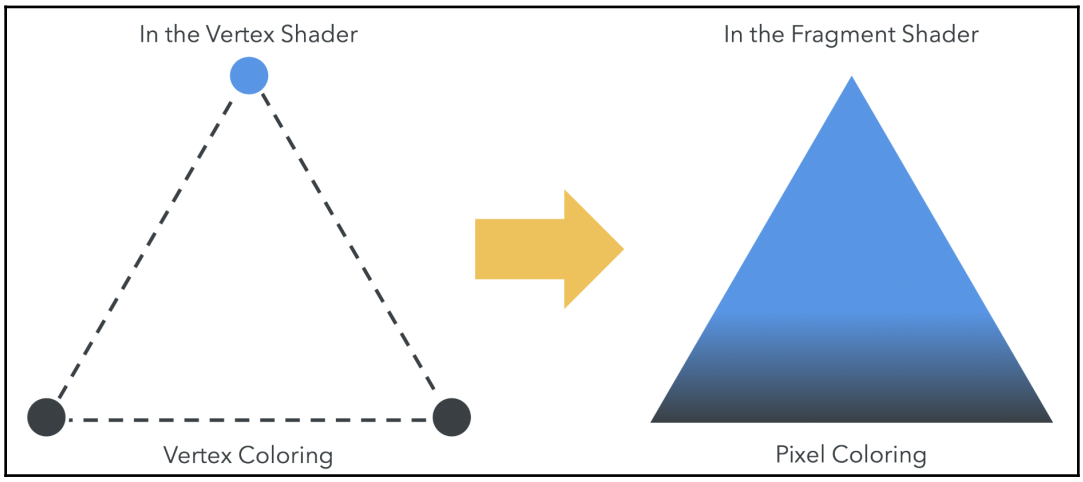

localhost:5000 says
Hooray! You got a WebGL2 context
OK


localhost:5000 says
clearColor = (
0.5,
0.6,
0.5
)
OK

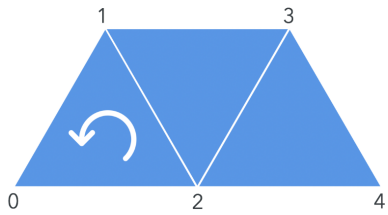

Car
Color
[16,31,80]
Shininess
0.9
Lights
Far Left
Diffuse
0.7
Specular
0.5
Far Right
Diffuse
0.8
Specular
0.8
Near Left
Diffuse
0.8
Specular
1
Near Right
Diffuse
0.5
Specular
0.8
Background
[230,230,230]
Floor
Go Home
Close Controls

Chapter 2: Rendering





Vertices and Indices



Index	Vertex Coordinates
0	(0, 0)
1	(10, 10)
2	(20, 0)
3	(30, 10)
4	(40, 0)

coordinates

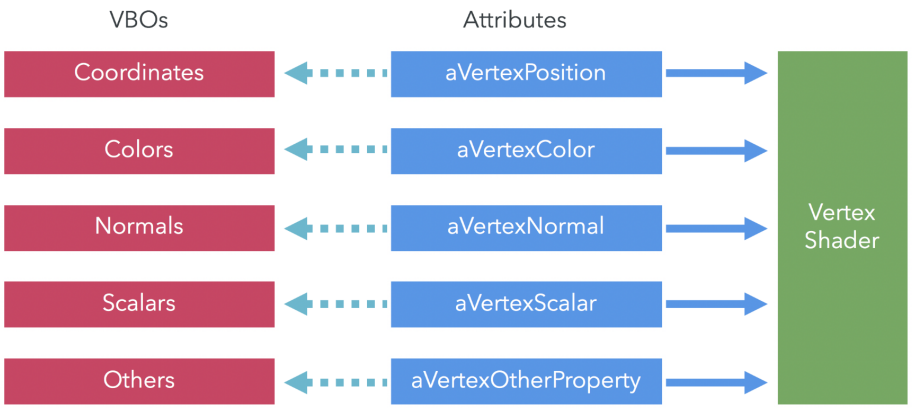
Vertex array = `[0, 0, 10, 10, 20, 0, 30, 10, 40, 0]` → Vertex Buffer

Index array = `[0, 2, 1, 1, 2, 3, 2, 4, 3]` → Index Buffer

triangles

Triangles in the index array are *usually*, but not necessarily, defined in counter-clockwise order.

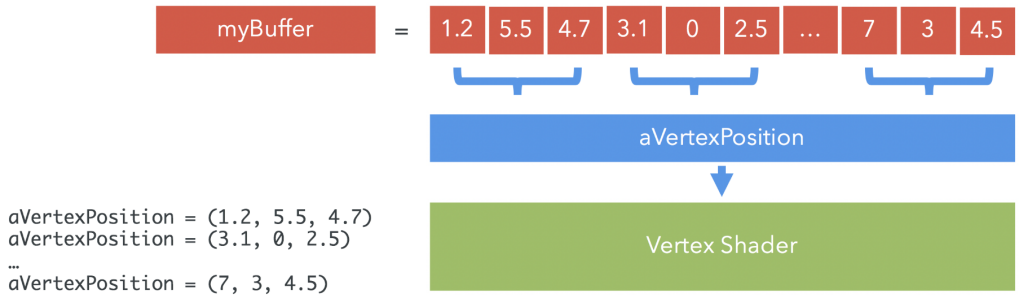
Associating Attributes to VBOs



Each attribute points to *one* WebGL buffer. From that buffer, the attribute extracts a value to pass to the Vertex Shader.

Pointing an Attribute to the Currently Bound VBO

```
1. gl.bindBuffer(gl.ARRAY_BUFFER, myBuffer);
```

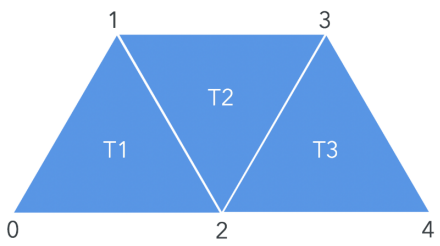



```
2. gl.vertexAttribPointer(aVertexPosition, 3, gl.FLOAT, false, 0, 0);

3. gl.enableVertexArrayAttrib(aVertexPosition);

4. gl.bindBuffer(gl.ARRAY_BUFFER, null);
```

Using drawArrays



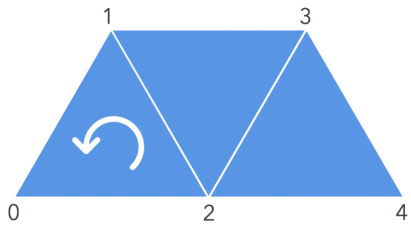
Index	Vertex Coordinates
0	(0, 0)
1	(10, 10)
2	(20, 0)
3	(30, 10)
4	(40, 0)

Triangle 1 Triangle 2 Triangle 3

Vertex array = `[0, 0, 20, 0, 10, 10, 10, 10, 20, 0, 30, 10, 20, 0, 40, 0, 30, 10]`

`drawArrays` uses the vertex data in the order they are defined in the vertex array.

Vertices and Indices



Index	Vertex Coordinates
0	(0, 0)
1	(10, 10)
2	(20, 0)
3	(30, 10)
4	(40, 0)

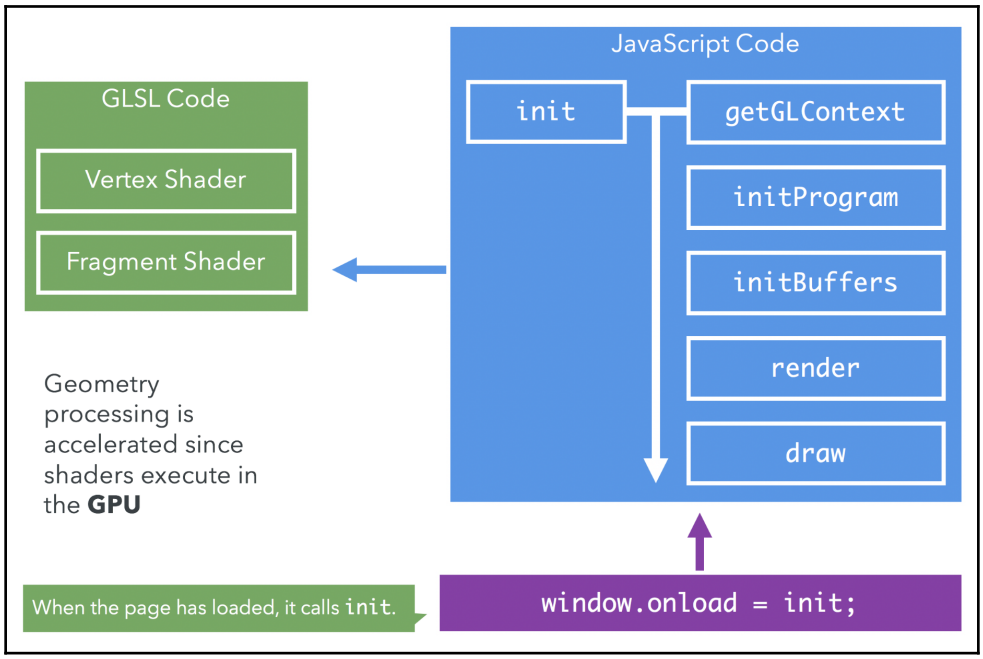
coordinates

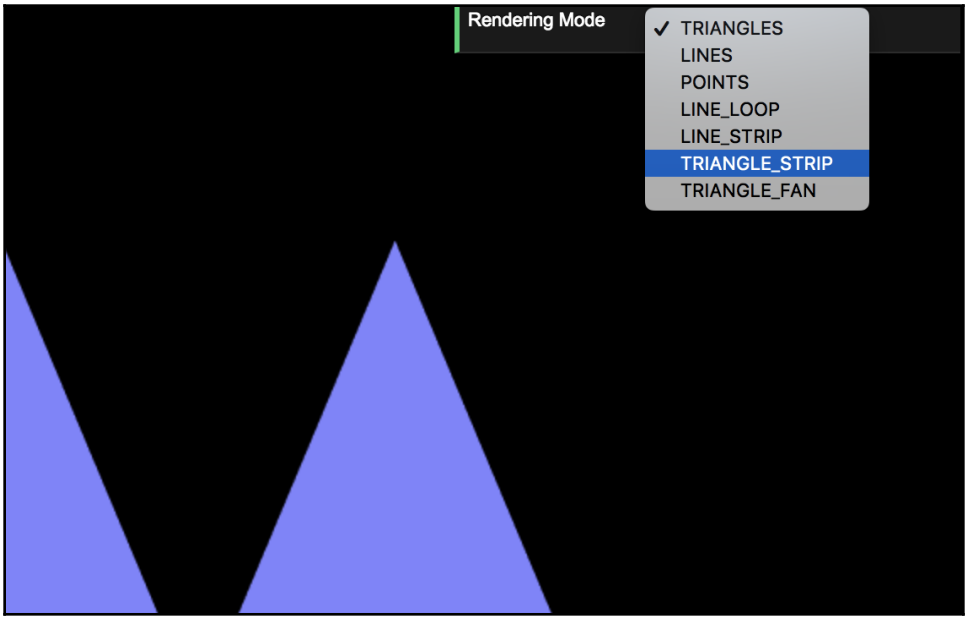
Vertex array = `[0, 0, 10, 10, 20, 0, 30, 10, 40, 0]` → Vertex Buffer

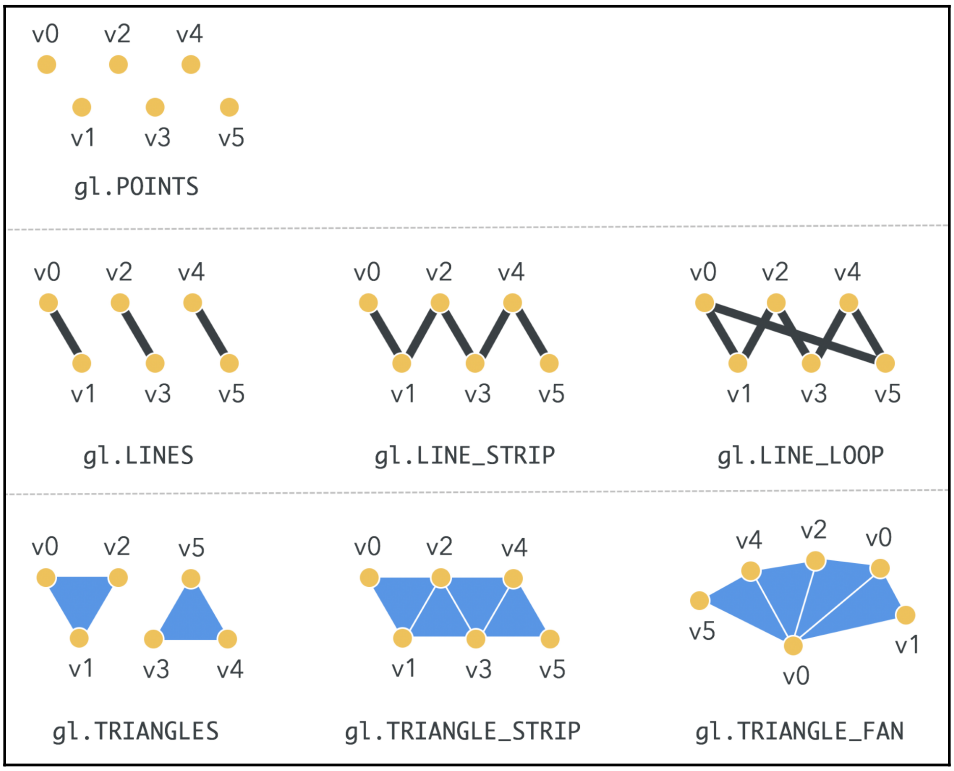
Index array = `[0, 2, 1, 1, 2, 3, 2, 4, 3]` → Index Buffer

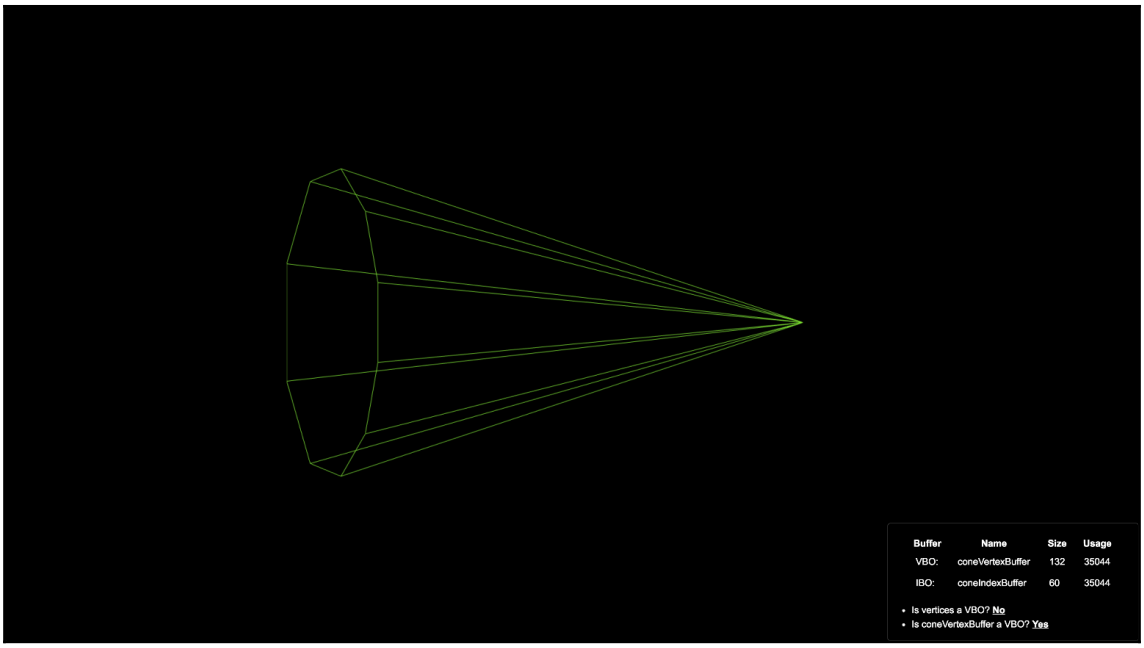
triangles

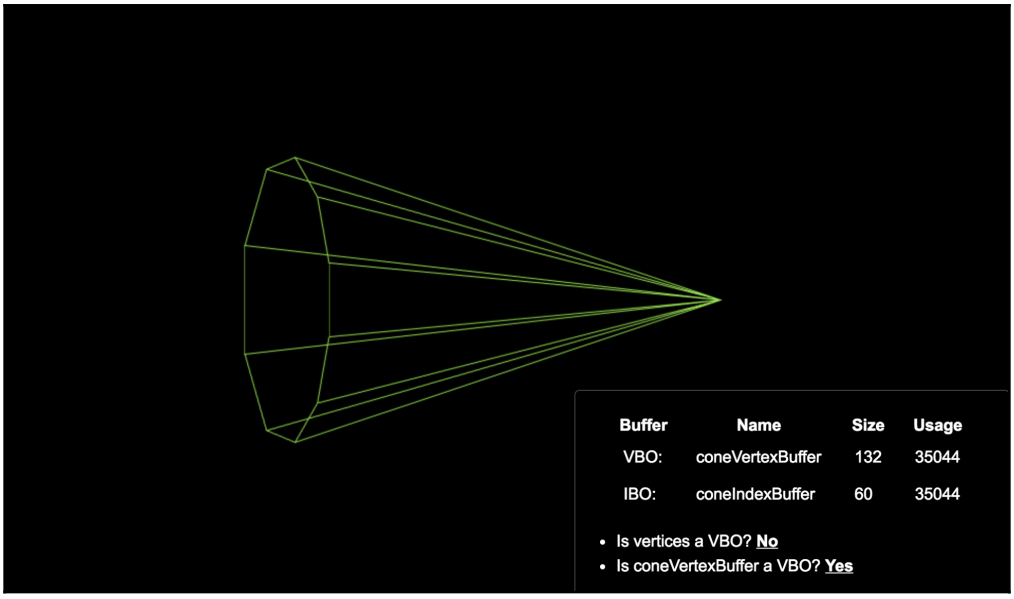
Triangles in the index array are *usually*, but not necessarily, defined in counter-clockwise order.

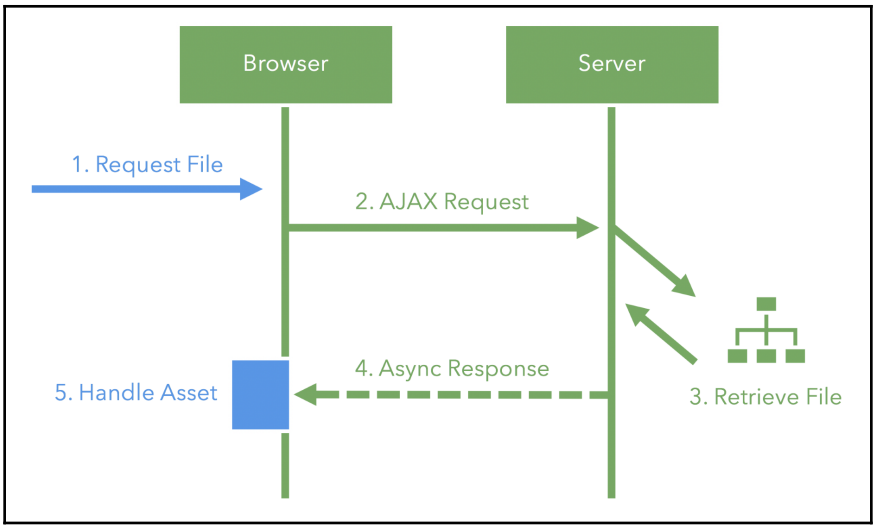



Rendering Mode
TRIANGLES
LINES
POINTS
LINE_LOOP
LINE_STRIP
TRIANGLE_STRIP
TRIANGLE_FAN


v0
v2
v4
v1
v3
v5
gl.POINTS
gl.LINES
gl.LINE_STRIP
gl.LINE_LOOP
gl.TRIANGLES
gl.TRIANGLE_STRIP
gl.TRIANGLE_FAN

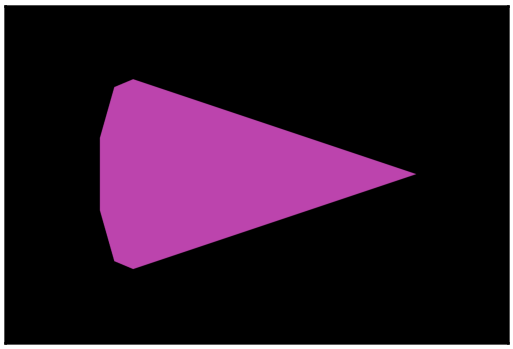

Buffer Name Size Usage
VBO: coneVertexBuffer 132 35044
IBO: coneIndexBuffer 60 35044
Is vertices a VBO? No
Is coneVertexBuffer a VBO? Yes

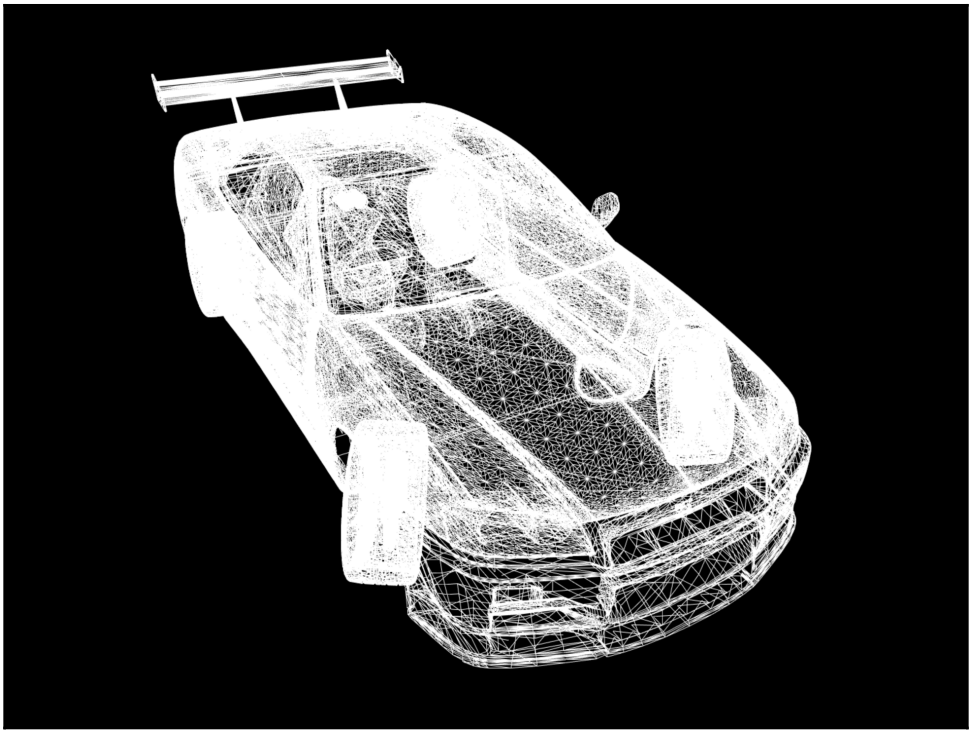

Buffer Name Size Usage
VBO: coneVertexBuffer 132 35044
IBO: coneIndexBuffer 60 35044
Is vertices a VBO? No
Is coneVertexBuffer a VBO? Yes



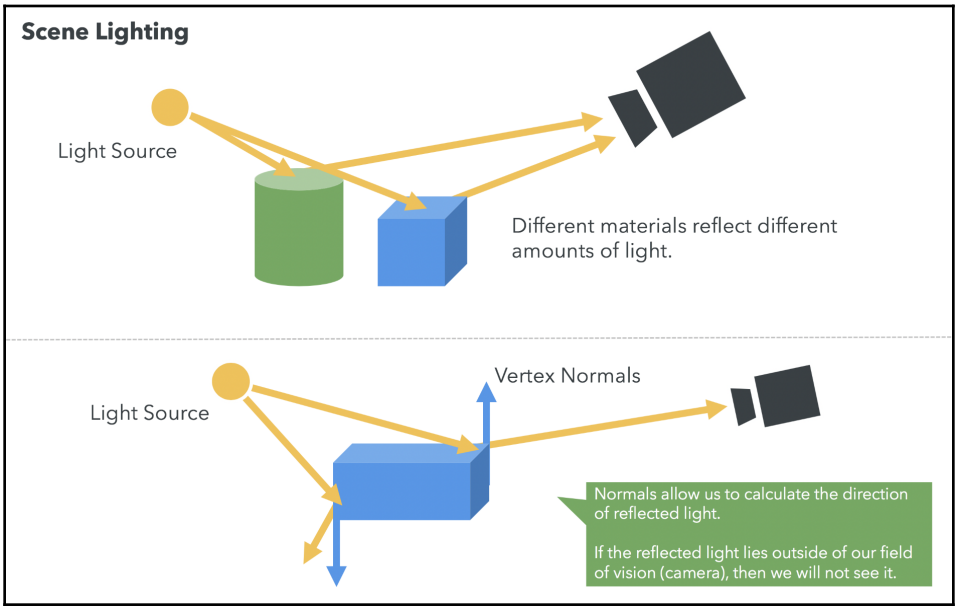
Index of **webgl2-book/ch02/**

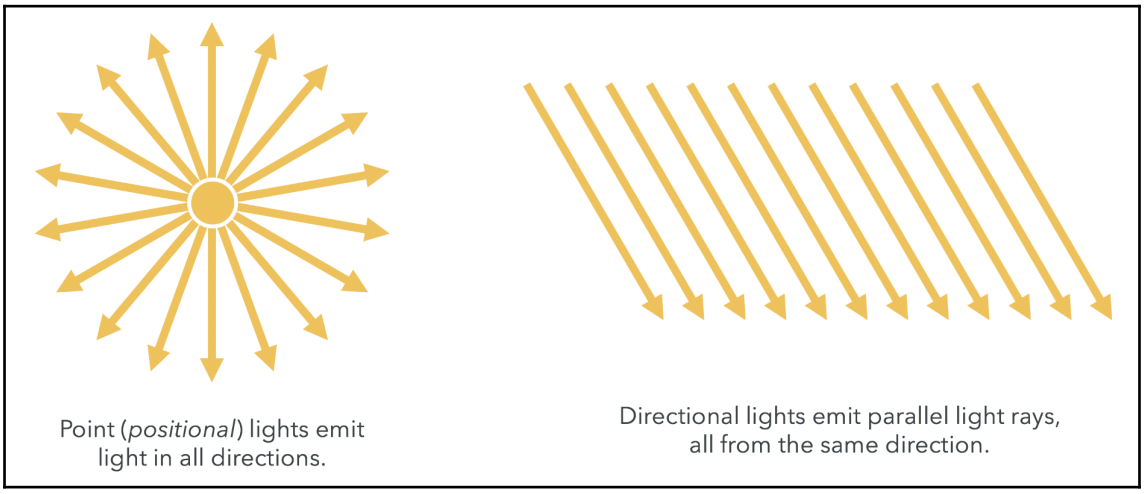
..	ch02_01_square.html 6 KB	ch02_02_square-arr… 5 KB
ch02_03_square-vao… 6 KB	ch02_04_rendering-… 7 KB	ch02_05_state-mach… 8 KB
ch02_06_cone.html 5 KB	ch02_07_ajax-cone.… 6 KB	ch02_08_ajax-car.h… 6 KB
ch02_09_ajax-car-f… 5 KB		

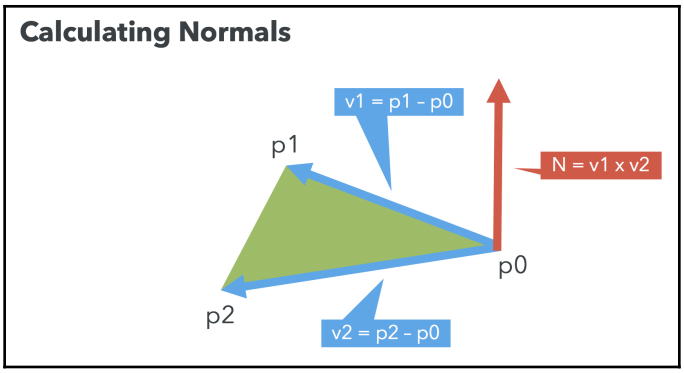


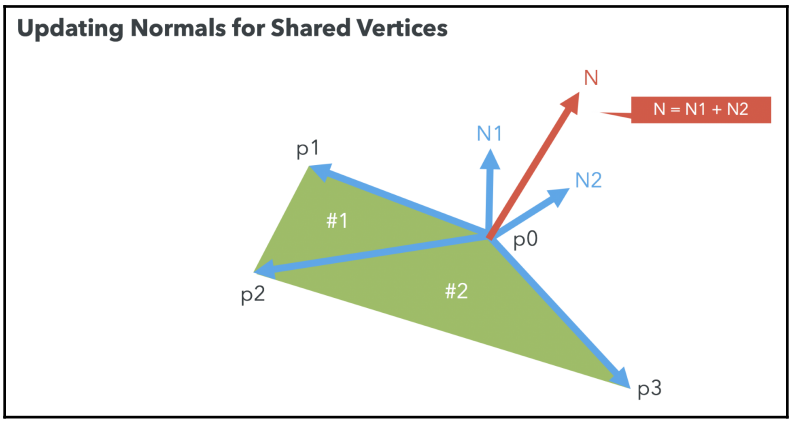


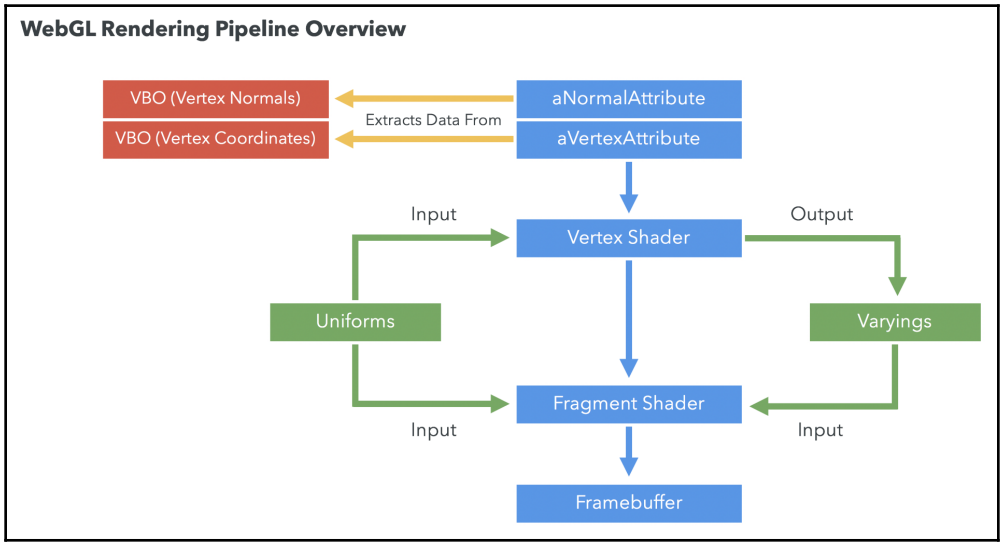
Chapter 3: Lights

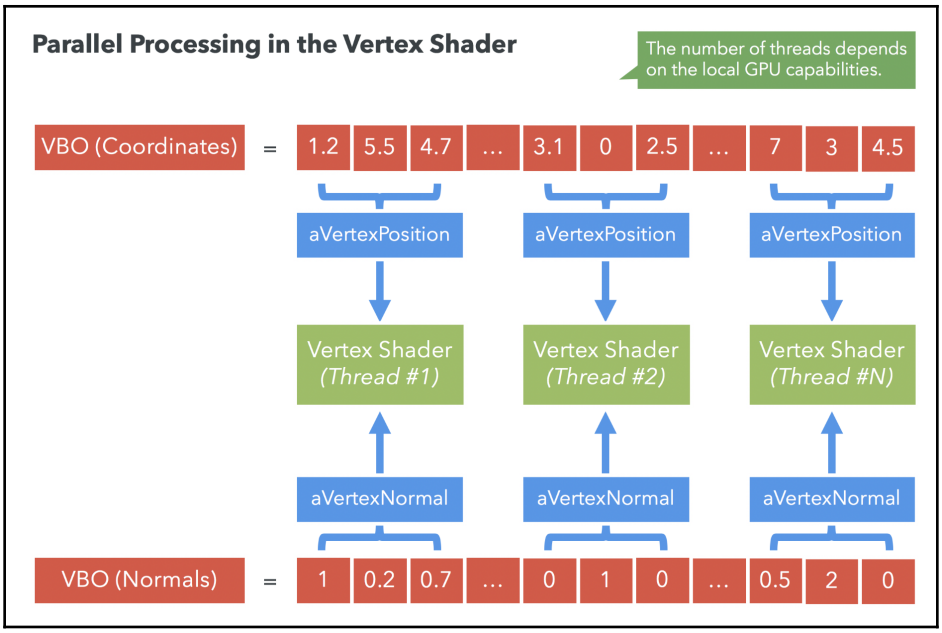




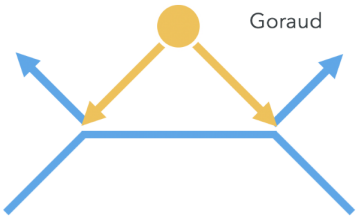

Calculating Normals
v1 = p1 - p0
N = v1 x v2
p1
p0
p2
v2 = p2 - p0


Updating Normals for Shared Vertices
N
N = N1 + N2
N1
N2
p1
#1
p0
p2
#2
p3


WebGL Rendering Pipeline Overview
VBO (Vertex Normals)
VBO (Vertex Coordinates)
Extracts Data From
aNormalAttribute
aVertexAttribute
Input
Vertex Shader
Output
Uniforms
Varyings
Fragment Shader
Input
Input
Framebuffer



Shading/Interpolation Methods



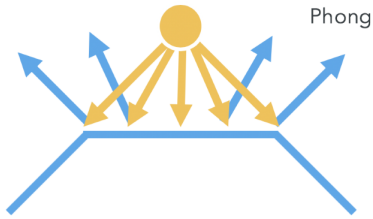
Vertex Shader

Computes the final color for the vertex using the vertex normal. Then, it passes the calculated color to the fragment shader in a varying variables.

Varying Color

Fragment Shader

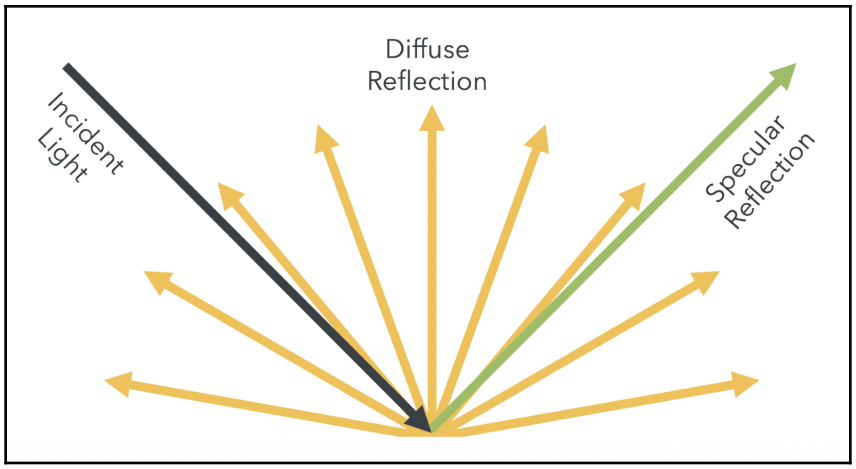
Assigns the color for the fragment using the interpolated varying color.

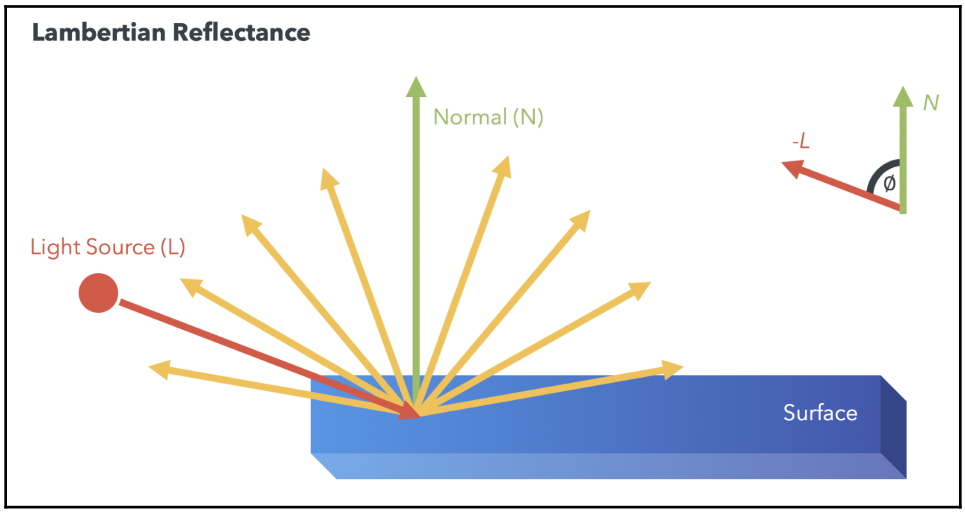


Passes the vertex normal to the fragment shader in a varying variable.

Varying Normal

Computes the final color for the fragment using the respective interpolated normal.





$$\boxed{F = C_l C_m (-L \bullet N)}$$

$$\boxed{-L \bullet N = |-L||N| cos\varnothing}$$

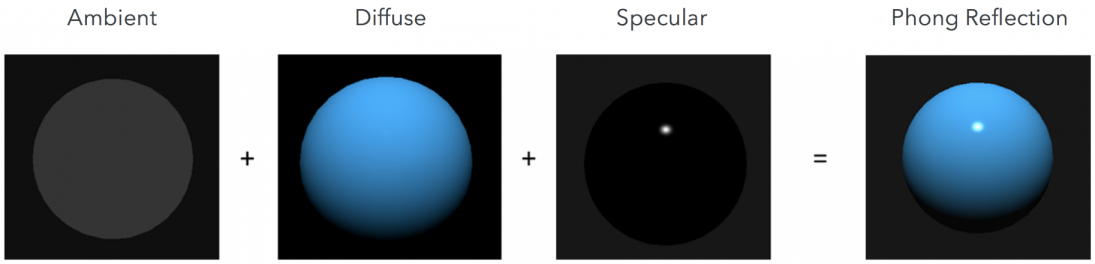
$$\boxed{-L \bullet N = cos\varnothing}$$

$$\boxed{-L \bullet N = cos\varnothing}$$

$$\boxed{F = C_l C_m cos\varnothing}$$

Phong Reflection Model

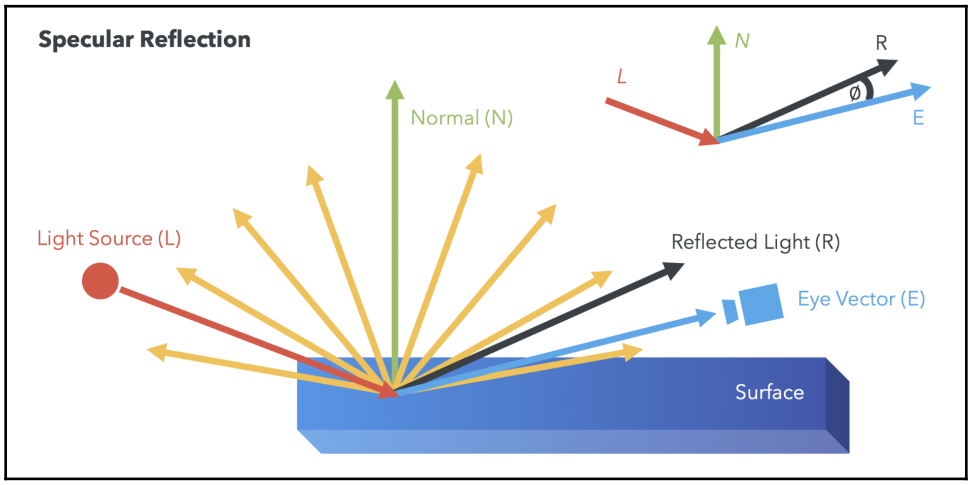
Reflected color is the result of combining three types of light-object interactions:



Amount of light present everywhere in the scene, independent from any light source.

The incident light is reflection in many directions. It can be modeled by a Lambertian surface.

Mirror-like reflection. The direction of the incoming light and the direction of the reflected outgoing light make the same angle with respect to the surface normal.

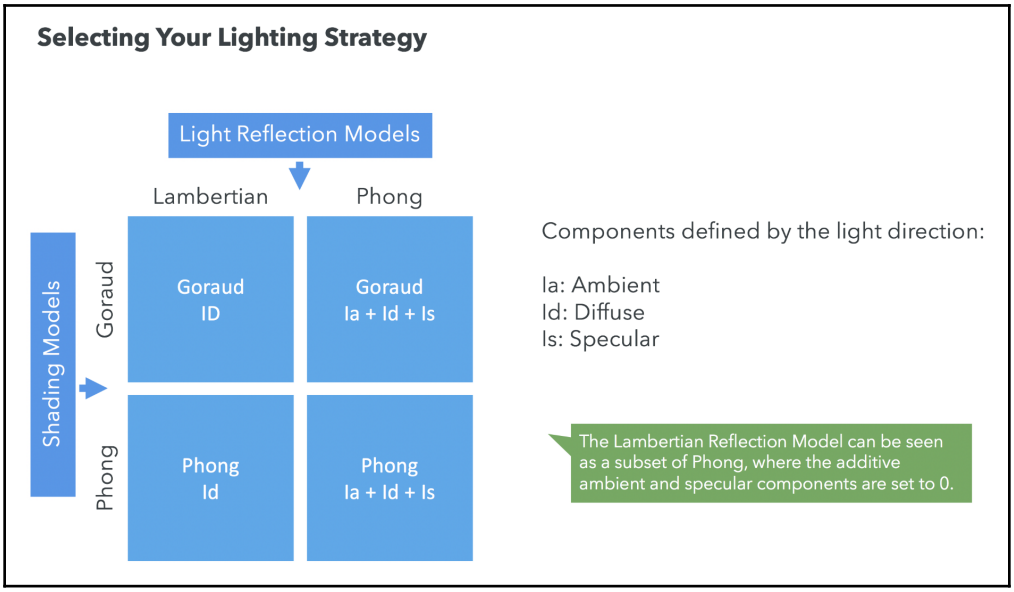


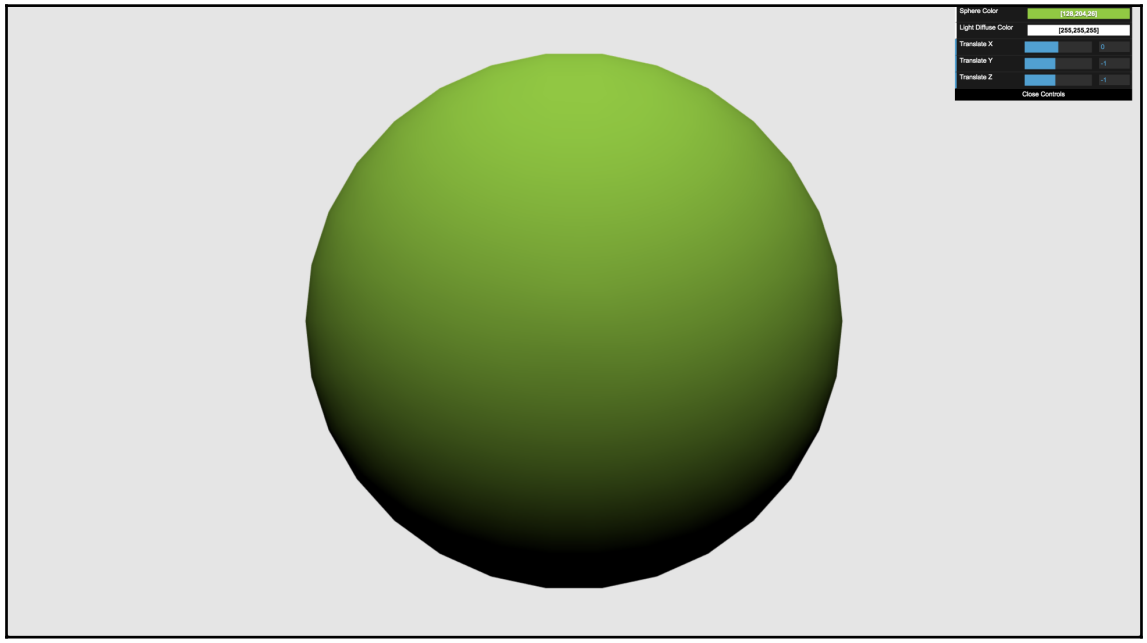
$$F_s = C_l C_m (R \bullet E)^n$$

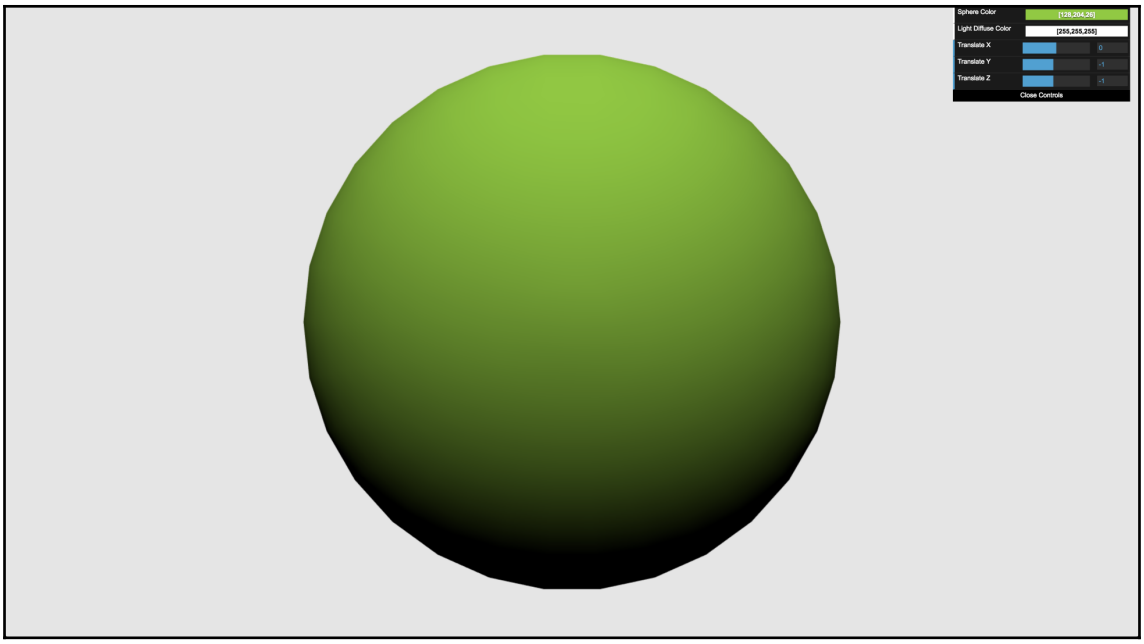
$$R \bullet E = |R||E| cos\varnothing$$

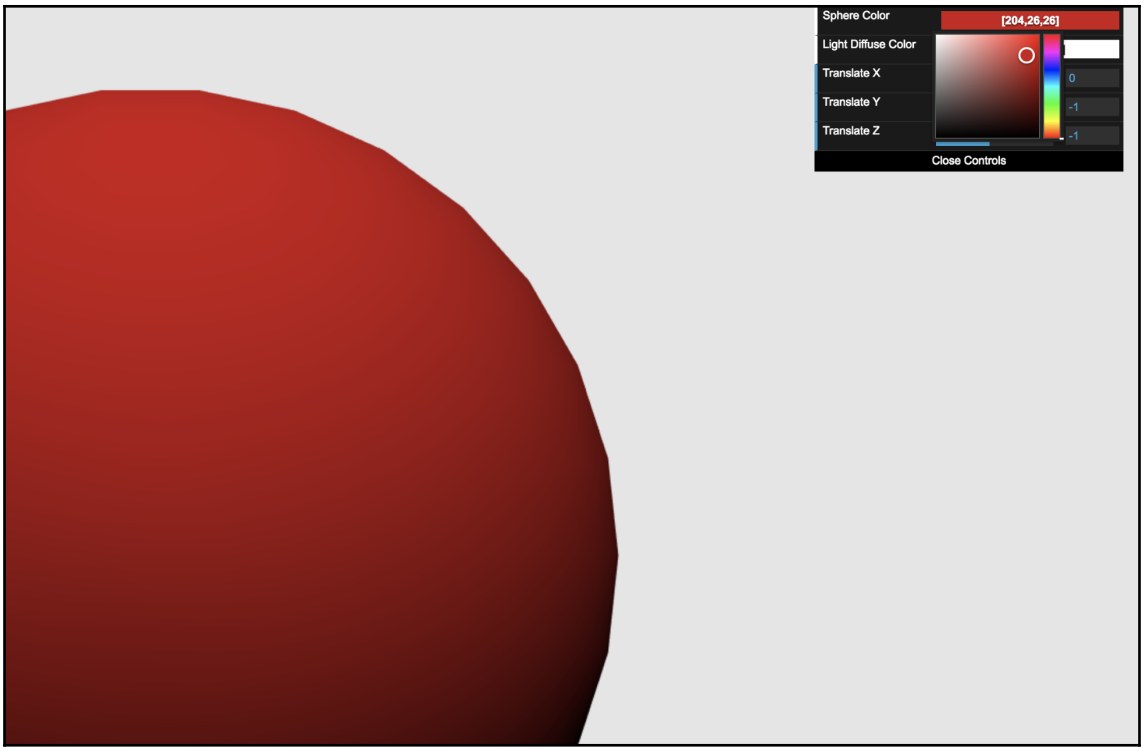
$$R \bullet E = cos\varnothing$$

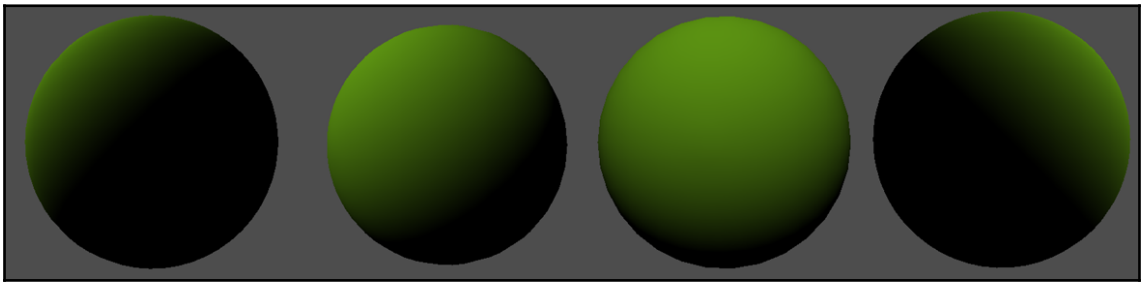
$$\boxed{F = C_l C_m cos^n \varnothing}$$

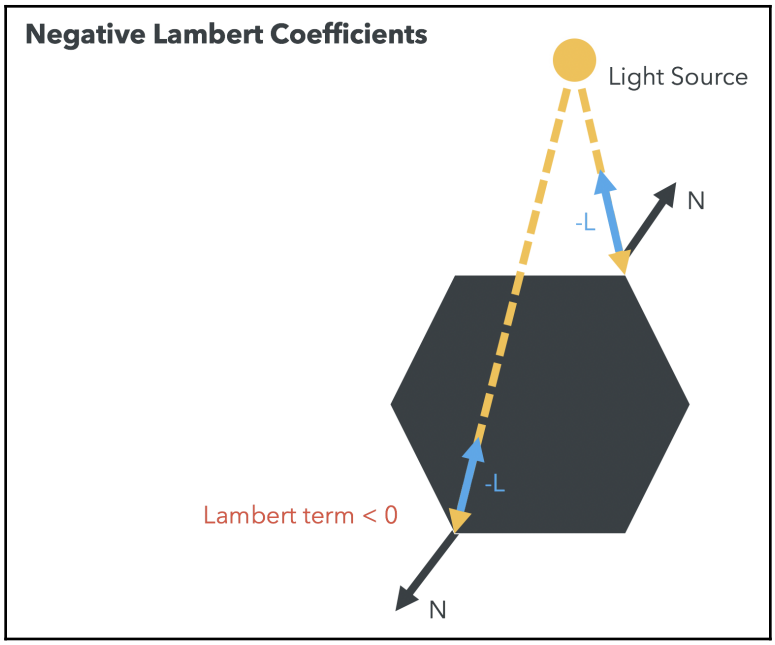


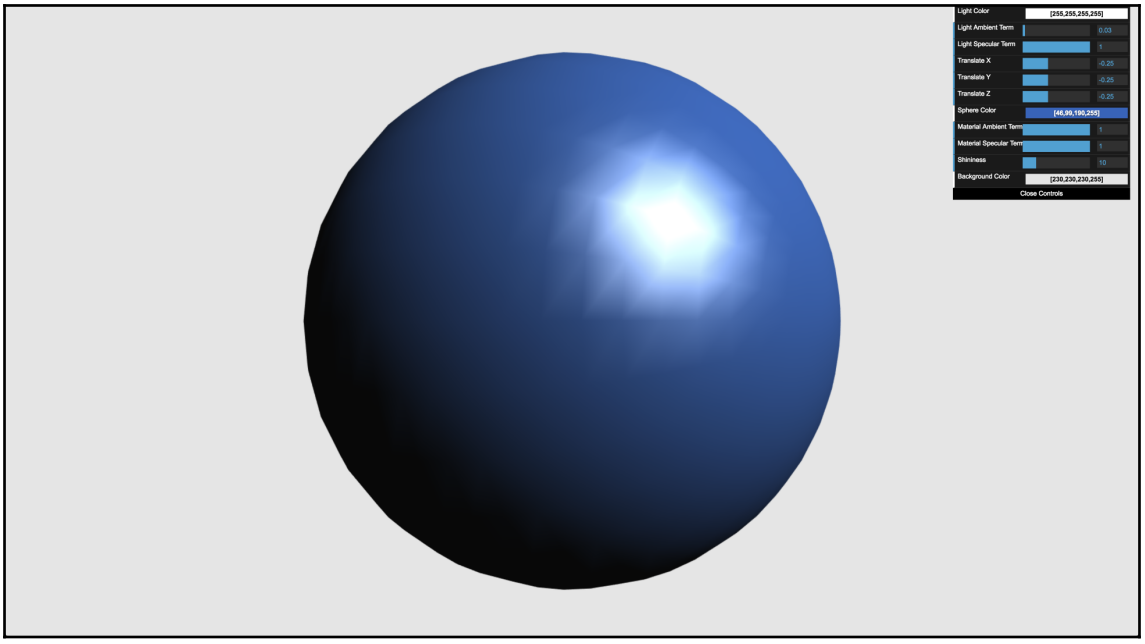



Sphere Color
[128,204,26]
Light Diffuse Color
[255,255,255]
Translate X
0
Translate Y
-1
Translate Z
-1
Close Controls

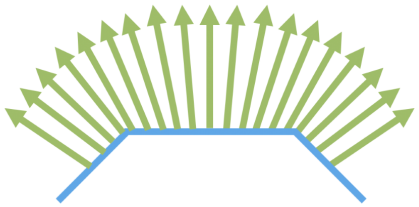

Sphere Color
[204,26,26]
Light Diffuse Color
Translate X
0
Translate Y
-1
Translate Z
-1
Close Controls



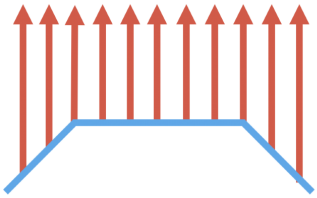

Negative Lambert Coefficients
Light Source
N
-L
-L
Lambert term < 0
N


Light Color
[255,255,255,255]
Light Ambient Term
0.03
Light Specular Term
1
Translate X
-0.25
Translate Y
-0.25
Translate Z
-0.25
Sphere Color
[46,99,190,255]
Material Ambient Term
1
Material Specular Term
1
Shininess
10
Background Color
[230,230,230,255]
Close Controls

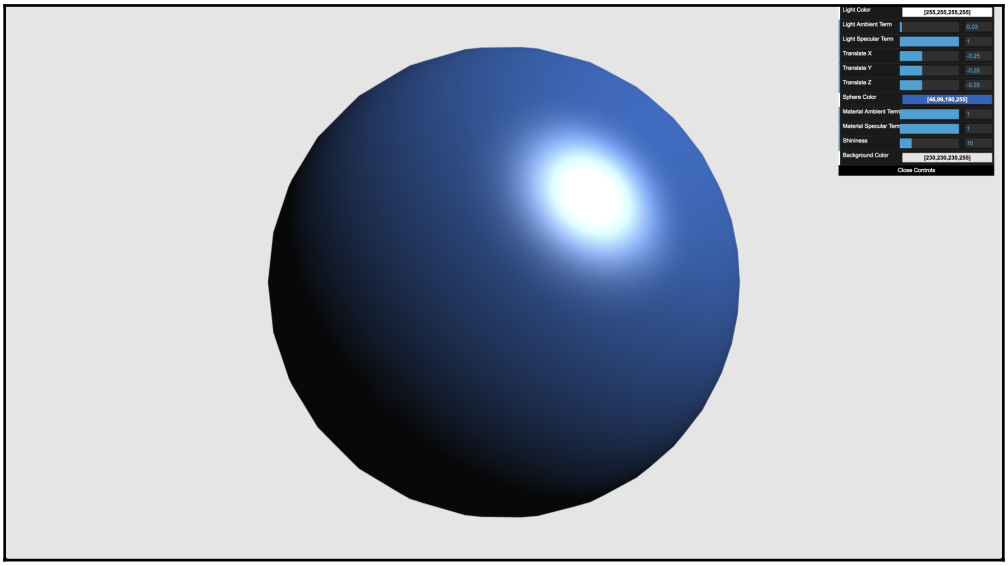
Per-Pixel Coloring

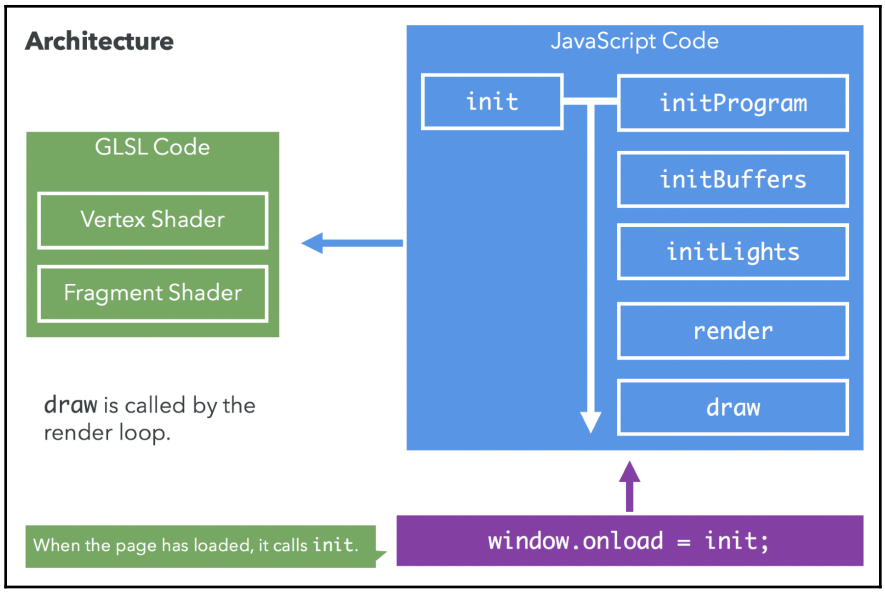


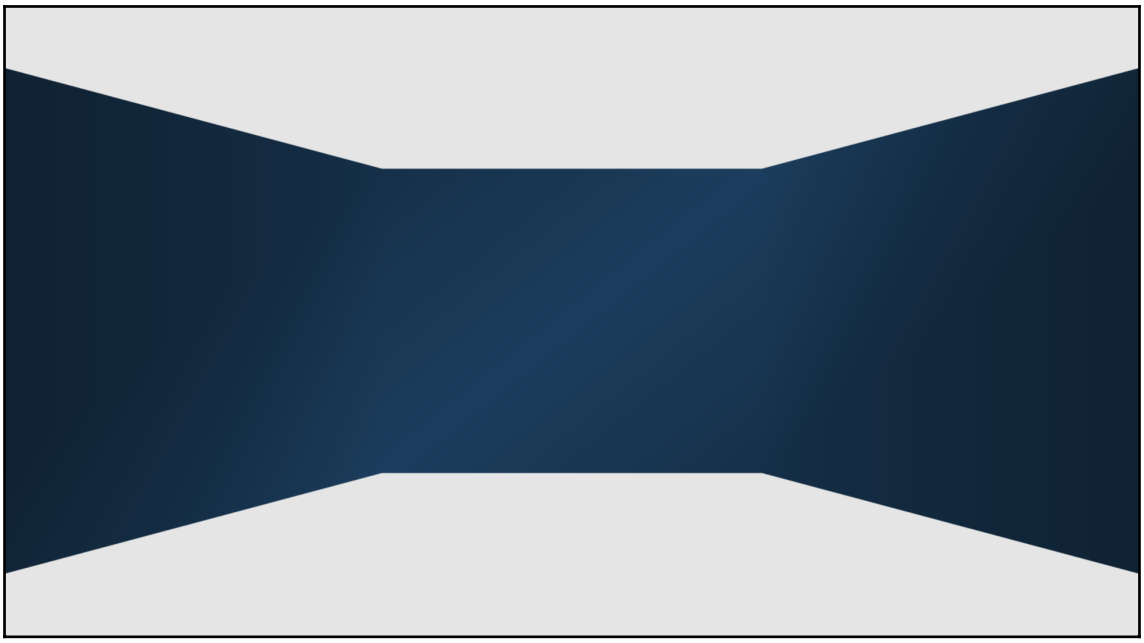
Interpolated Normals

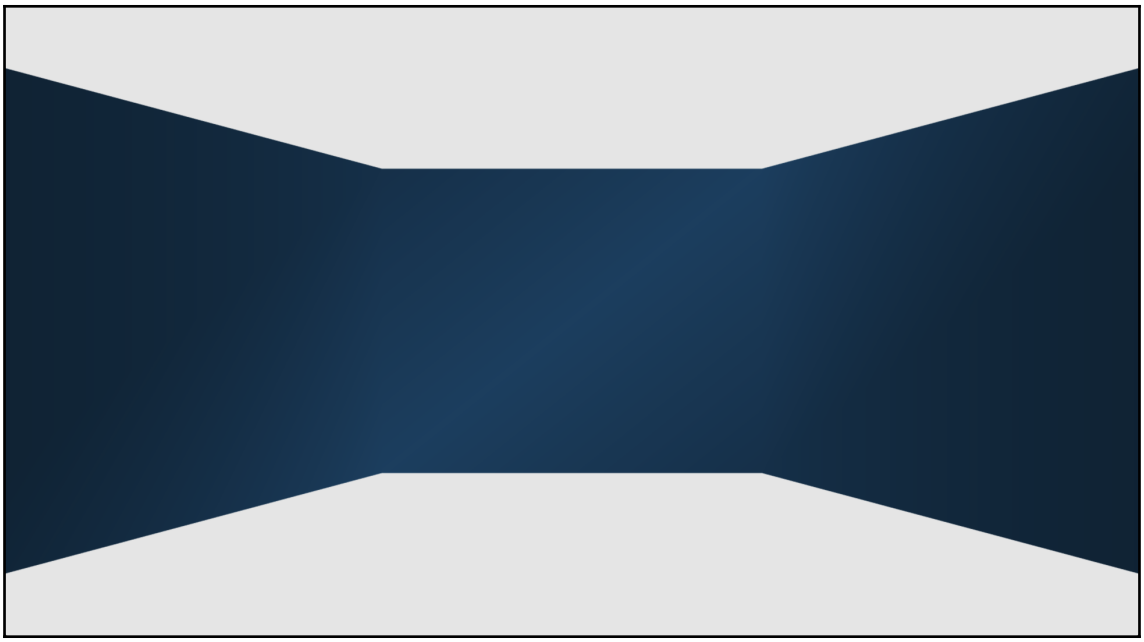


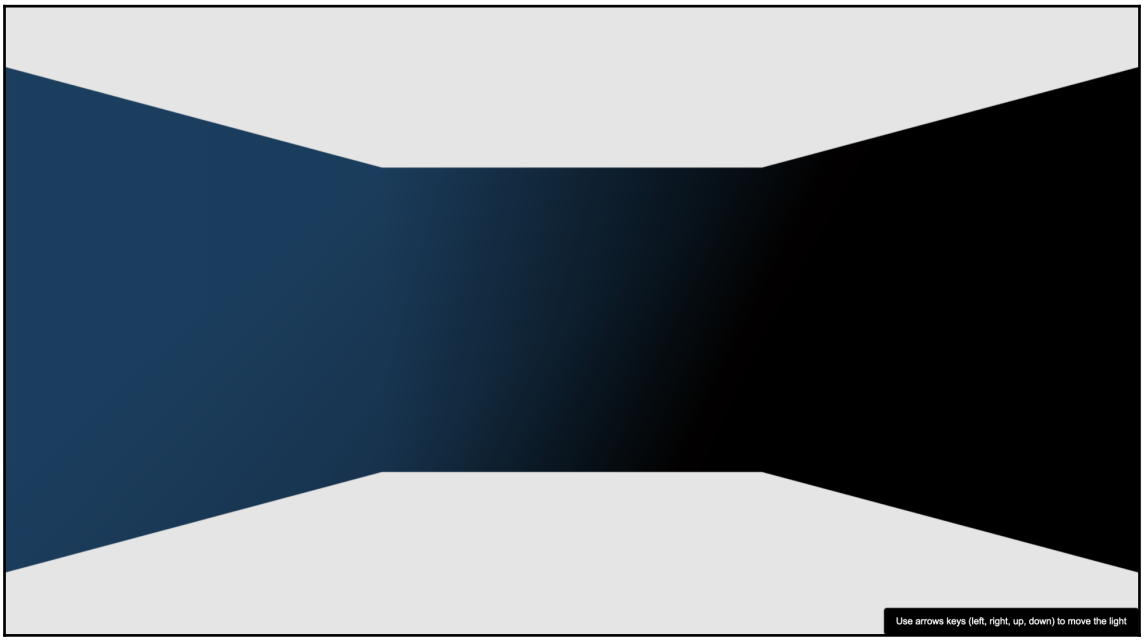
Inverted Light Direction

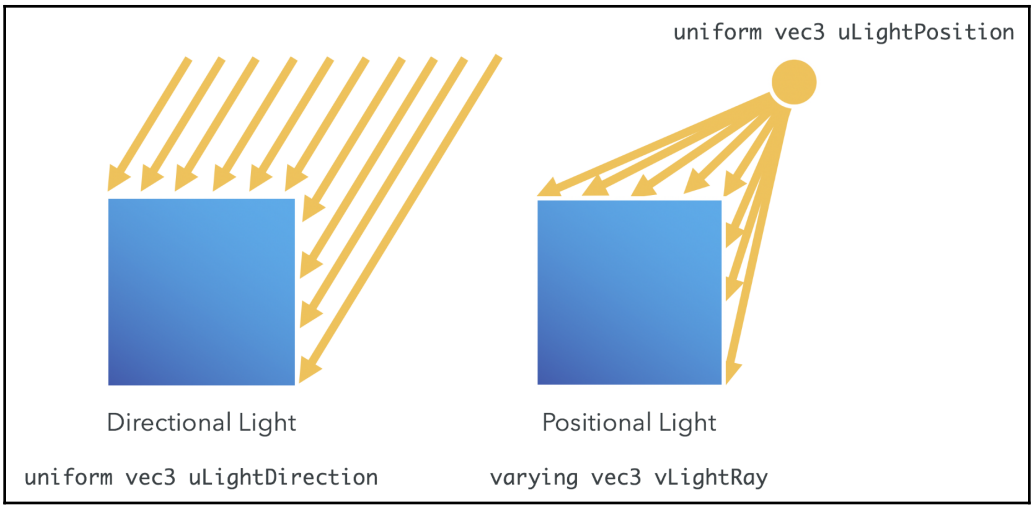


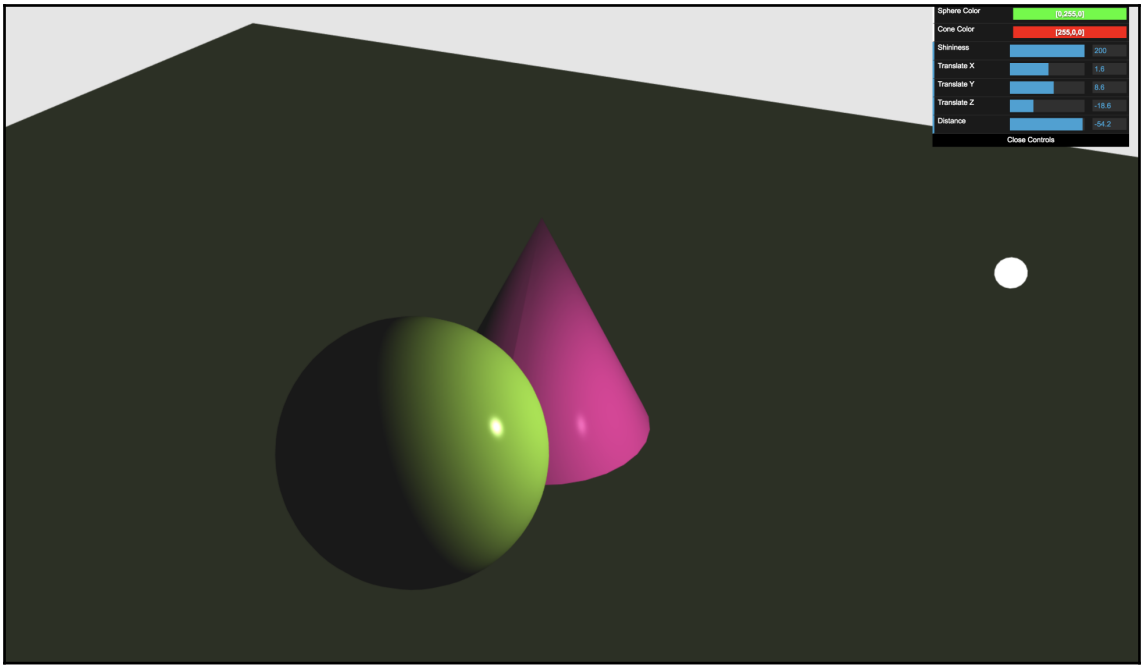

Architecture
JavaScript Code
init
initProgram
initBuffers
initLights
render
draw
GLSL Code
Vertex Shader
Fragment Shader
draw is called by the render loop.
When the page has loaded, it calls init.
window.onload = init;

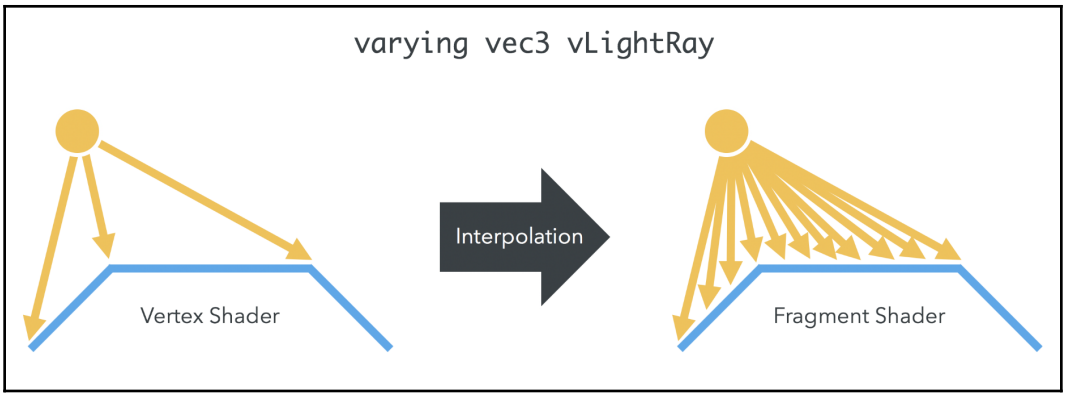


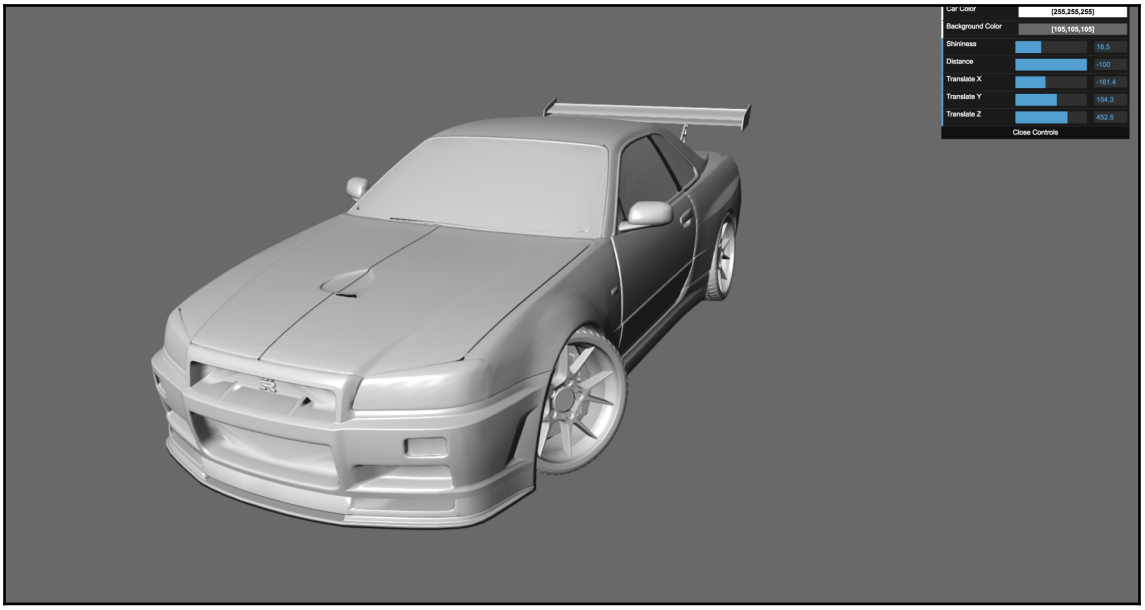



Use arrows keys (left, right, up, down) to move the light

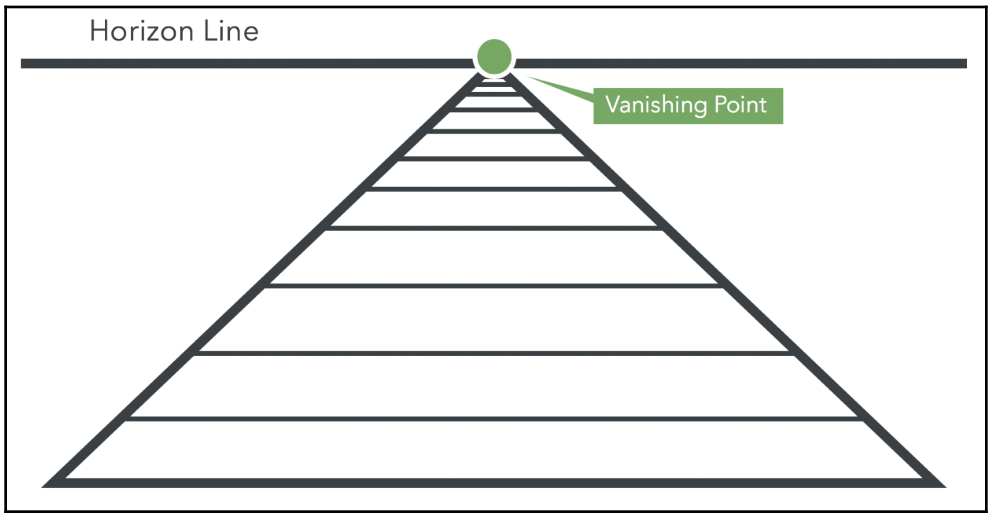

uniform vec3 uLightPosition
Directional Light
Positional Light
uniform vec3 uLightDirection
varying vec3 vLightRay


Sphere Color
[0,255,0]
Cone Color
[255,0,0]
Shininess
200
Translate X
1.6
Translate Y
8.8
Translate Z
-18.6
Distance
-54.2
Close Controls


varying vec3 vLightRay
Vertex Shader
Interpolation
Fragment Shader


Car Color
[255,255,255]
Background Color
[105,105,105]
Shininess
18.5
Distance
-100
Translate X
-161.4
Translate Y
154.3
Translate Z
452.6
Close Controls

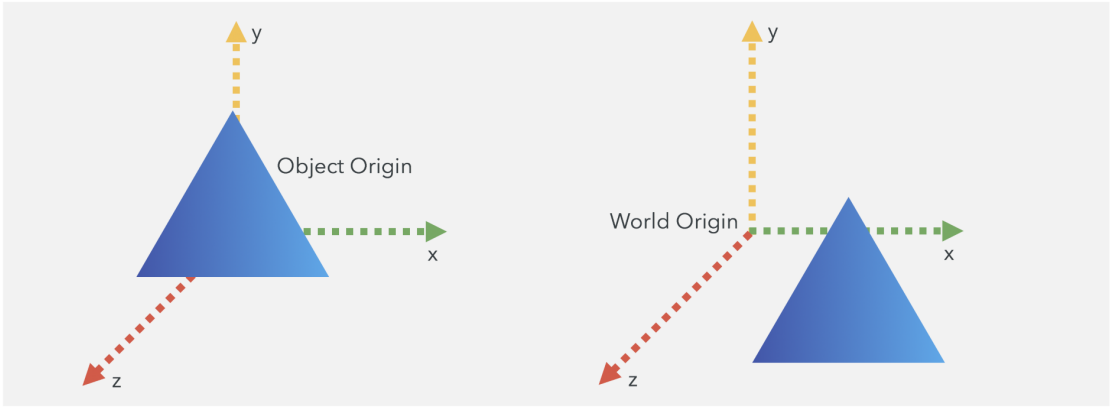
Chapter 4: Cameras



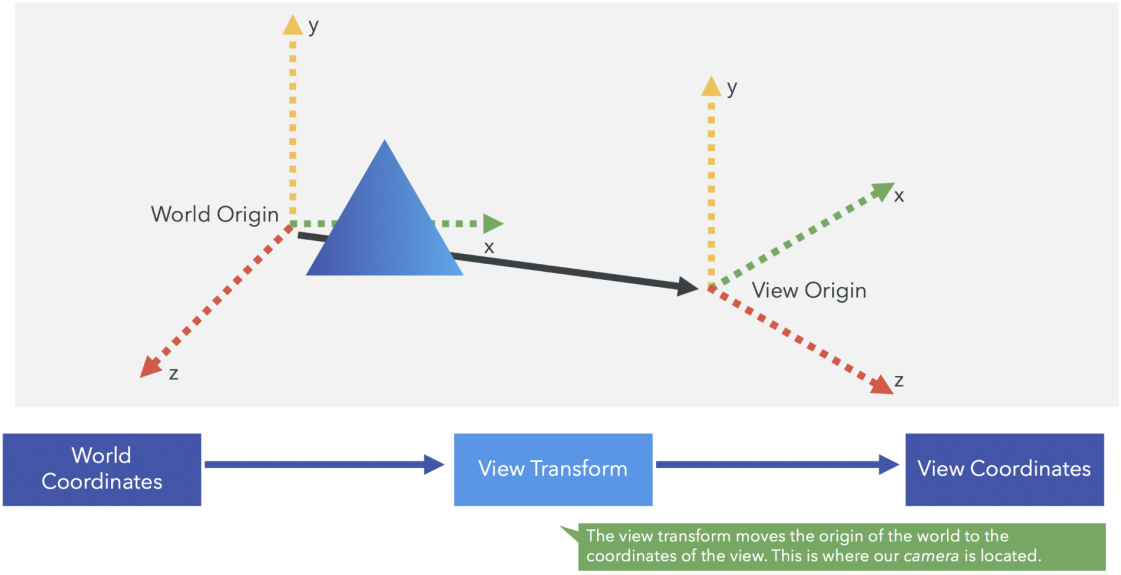
$$\boxed{h(x, y, z, w) = v(x/w, y/w, z/w)}$$

$$\boxed{v(x, y, z) = h(x, y, z, 1)}$$

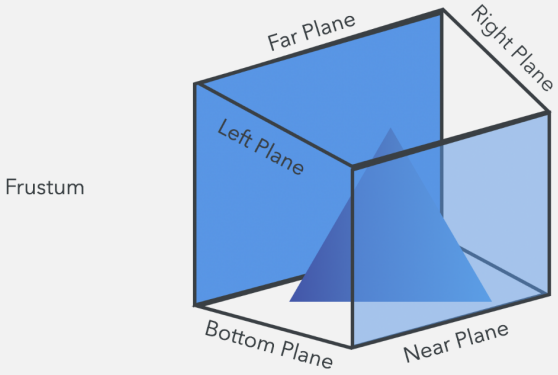
Model Transform



View Transform



Projection Transform

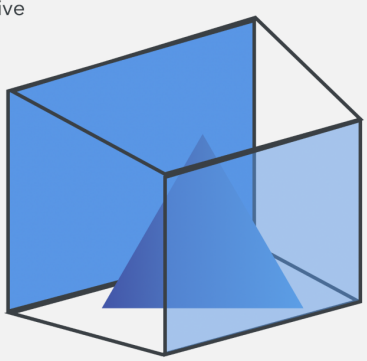


View Coordinates → Projection Transform → Clip Coordinates

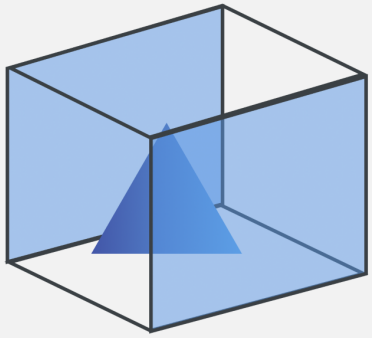
The frustum determines which objects or portion of objects will be *clipped out* and discarded.

Frustum Shape

Perspective

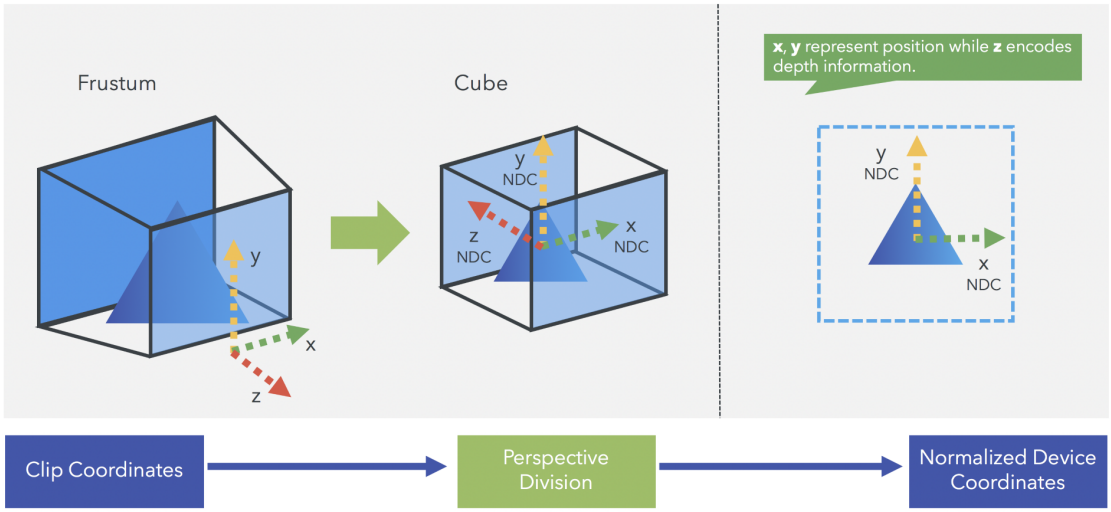


Orthographic

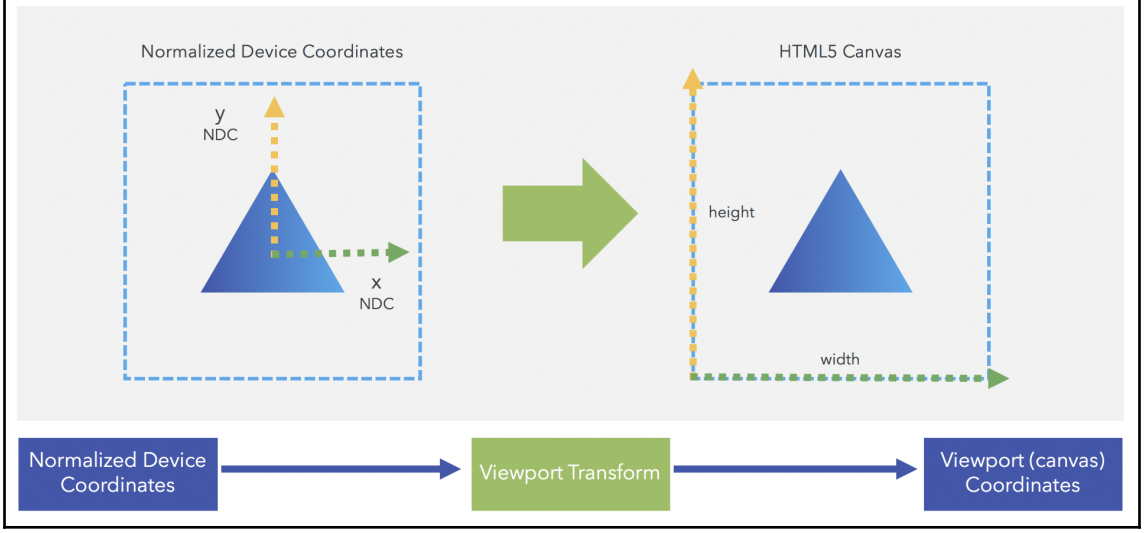


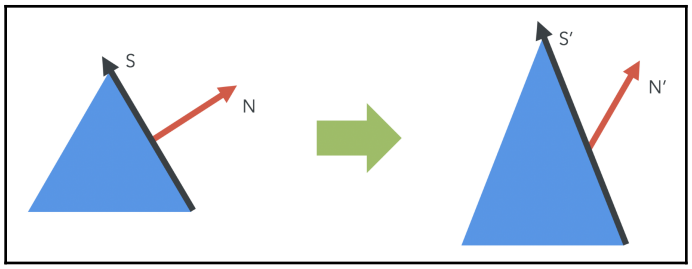
The extend and shape of the frustum determines how much of the 3D view space is visible and the type of projection mode.

Normalized Device Coordinates



Viewport Transform





$$\boxed{N \bullet S = 0}$$

$$\boxed{S' = M \bullet S}$$

$$\boxed{N' = K \bullet N}$$

$$\boxed{N' \bullet S' = 0}$$

$$\boxed{(KN) \bullet (MS) = 0}$$

$$\boxed{(KN)^T (MS) = 0}$$

$$\boxed{N^T K^T MS = 0}$$

$$\boxed{N^T (K^T M) S = 0}$$

$$\boxed{N \bullet S = 0}$$

$$\boxed{N^T S = 0}$$

$$\boxed{K^T M}$$

$$\boxed{K^T M = I}$$

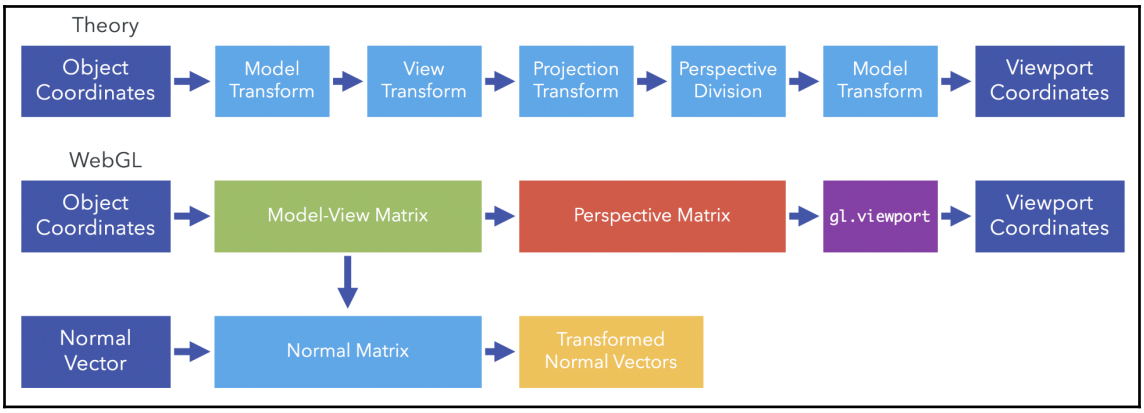
$$\boxed{K^T M M^{-1} = I M^{-1} = M^{-1}}$$

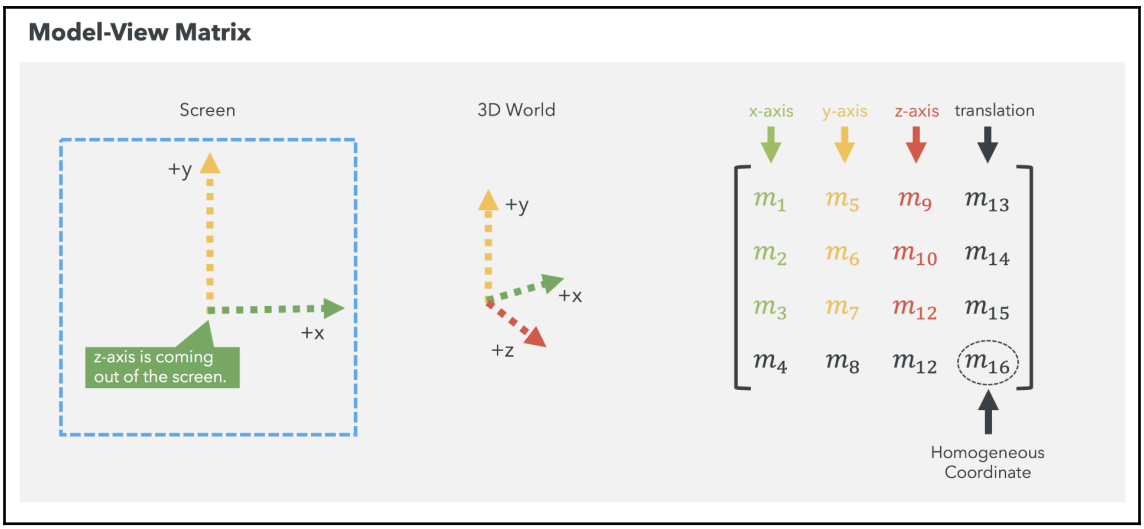
$$\boxed{K^T I = M^{-1}}$$

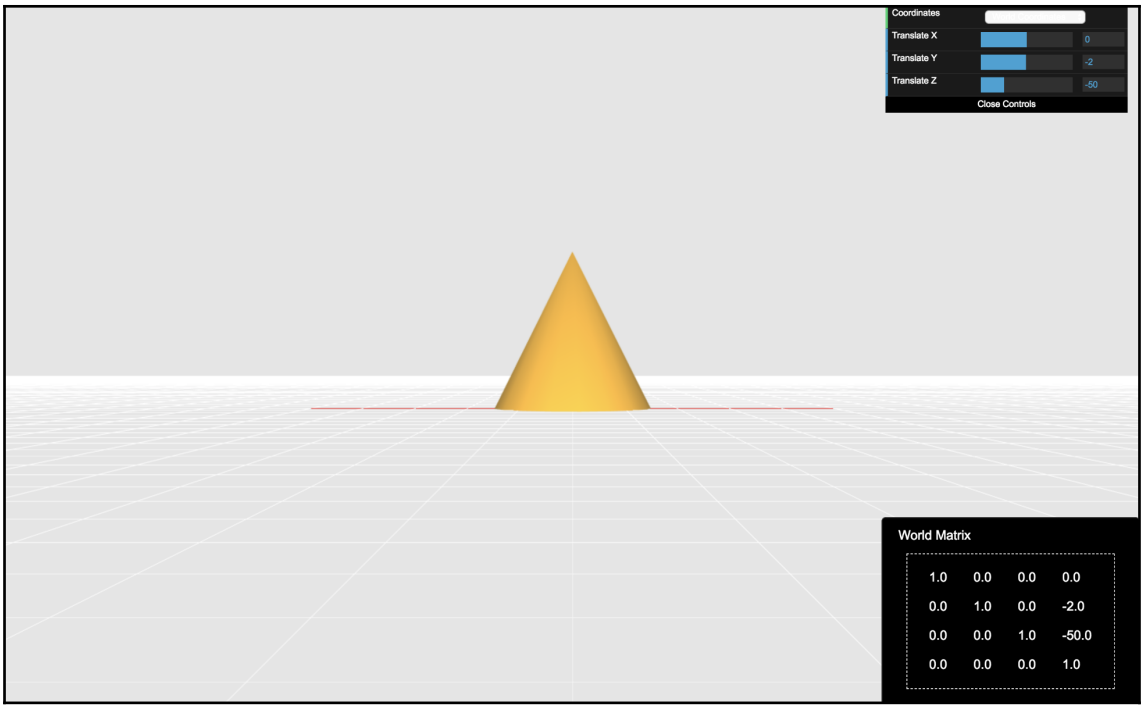
$$\boxed{M M^{-1} = I}$$

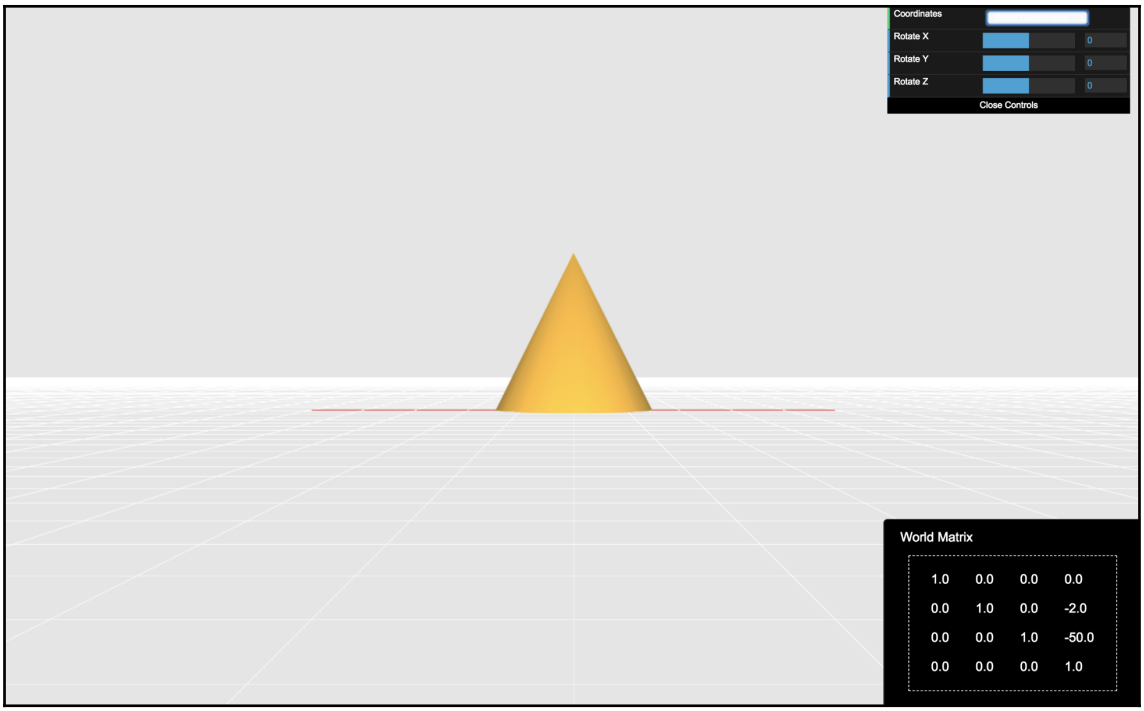
$$\boxed{(K^T)^T = (M^{-1})^T}$$

$$\boxed{K = (M^{-1})^T}$$

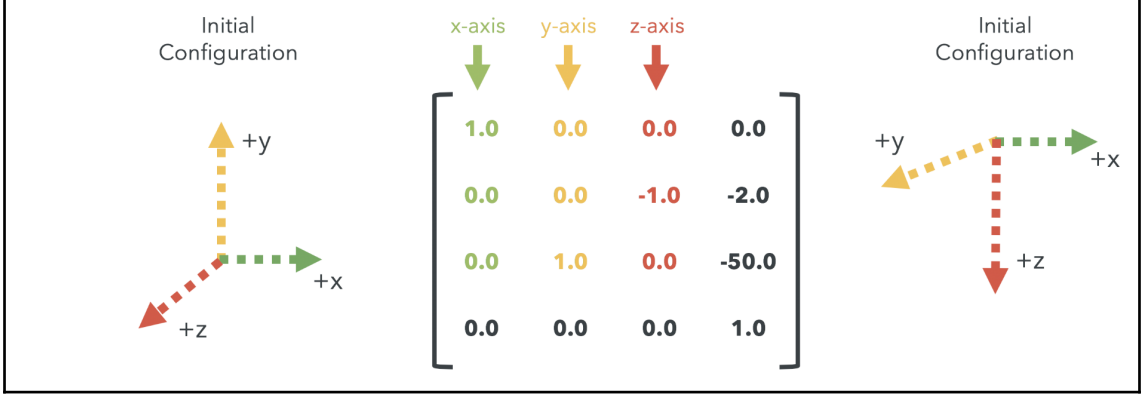

Theory
Object Coordinates
Model Transform
View Transform
Projection Transform
Perspective Division
Model Transform
Viewport Coordinates
WebGL
Object Coordinates
Model-View Matrix
Perspective Matrix
gl.viewport
Viewport Coordinates
Normal Vector
Normal Matrix
Transformed Normal Vectors


Model-View Matrix
Screen
+y
+x
z-axis is coming out of the screen.
3D World
+y
+x
+z
x-axis
y-axis
z-axis
translation
m_1 m_5 m_9 m_{13}
m_2 m_6 m_{10} m_{14}
m_3 m_7 m_{12} m_{15}
m_4 m_8 m_{12} m_{16}
Homogeneous Coordinate

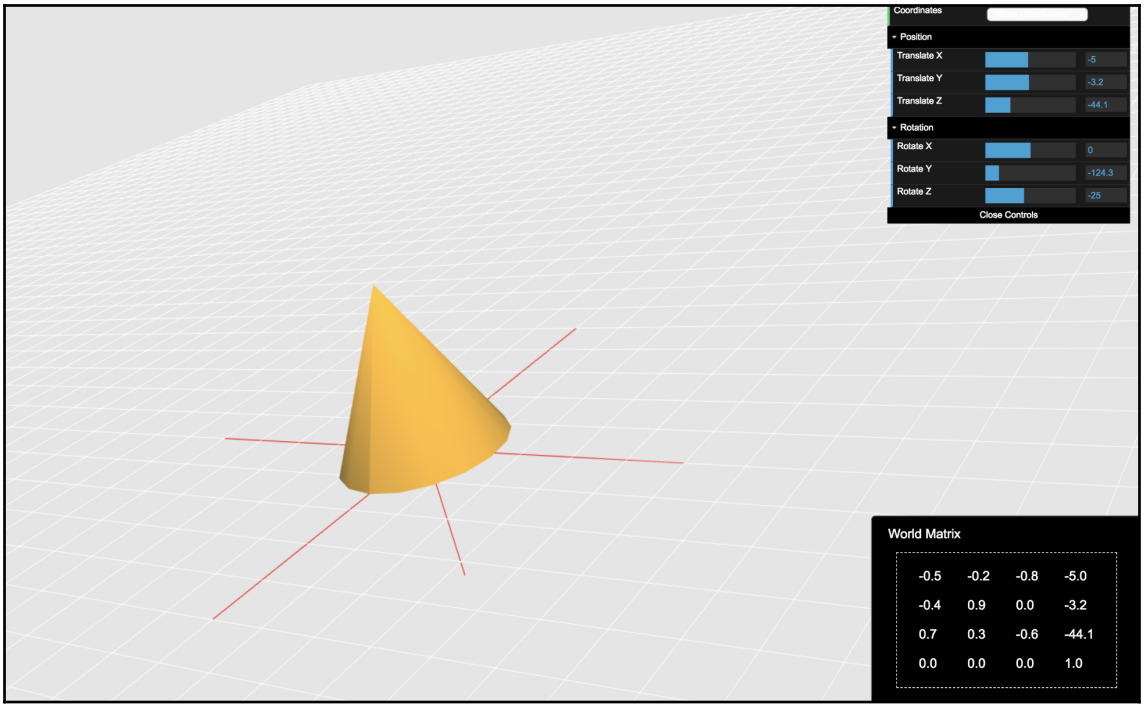

Coordinates
Translate X
0
Translate Y
-2
Translate Z
-50
Close Controls
World Matrix
1.0 0.0 0.0 0.0
0.0 1.0 0.0 -2.0
0.0 0.0 1.0 -50.0
0.0 0.0 0.0 1.0



Interpreting Rotations Using the Model-View Matrix



$$\begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & -1.0 & -2.0 \\ 0.0 & 1.0 & 0.0 & -50.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$



$$\boxed{MC = I}$$

$$\boxed{M^{-1}MC = M^{-1}}$$

$$\boxed{C = M^{-1}}$$

$$\boxed{RTv}$$

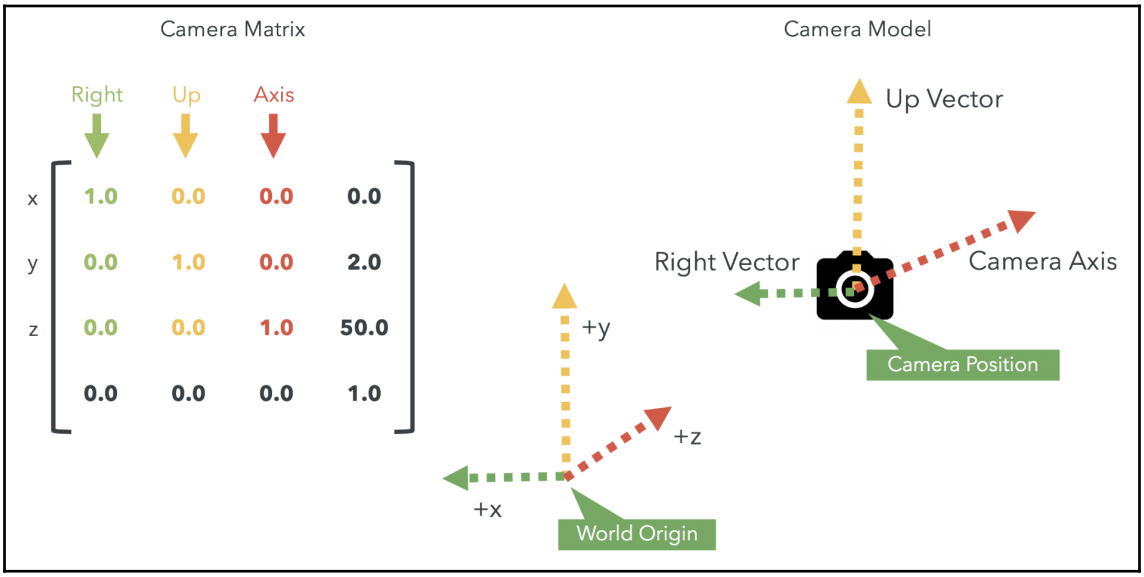
$$\boxed{M = RT}$$

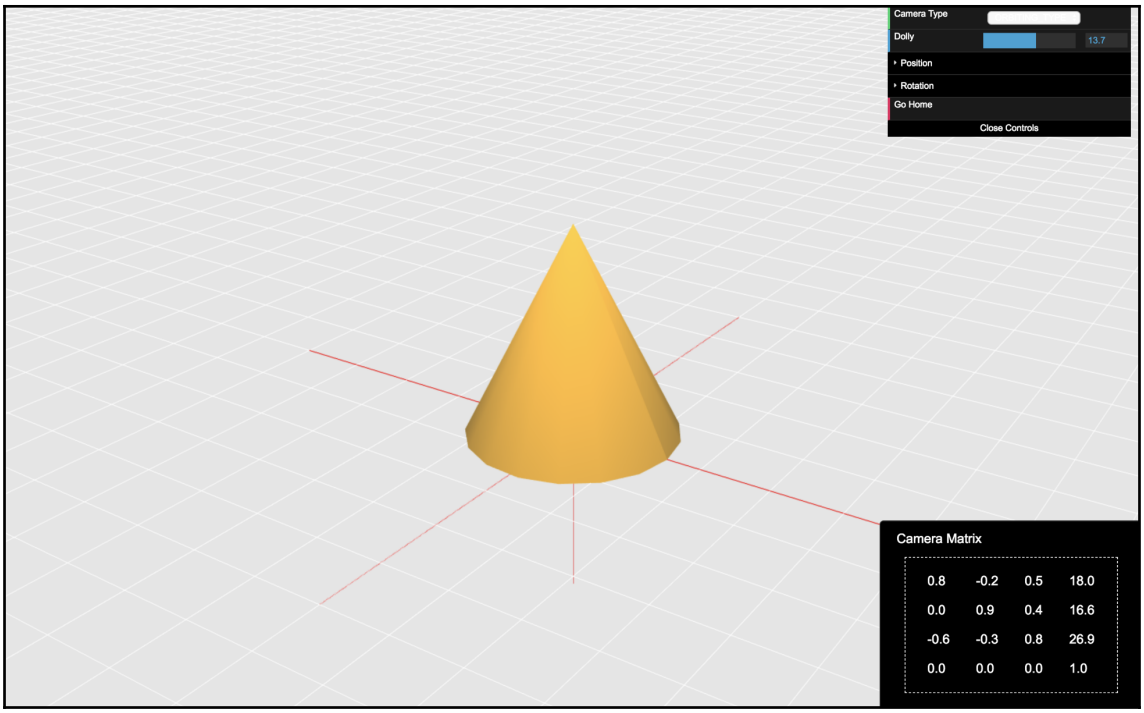
$$\boxed{C = M^{-1}}$$

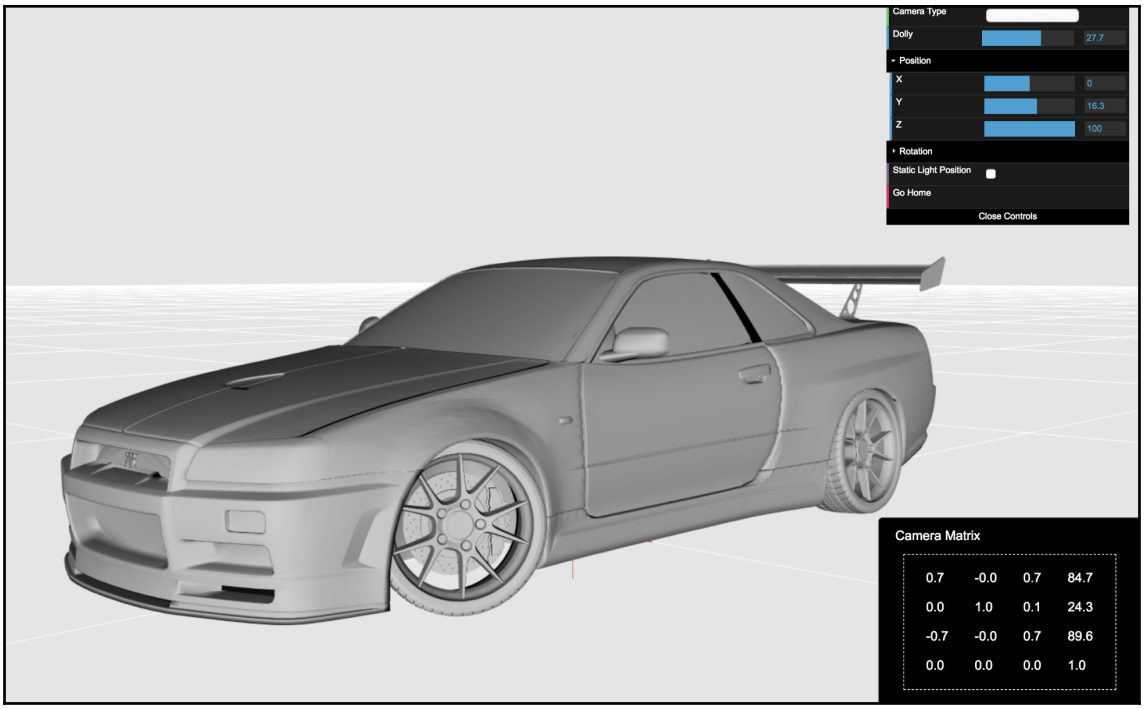
$$\boxed{C = (RT)^{-1}}$$

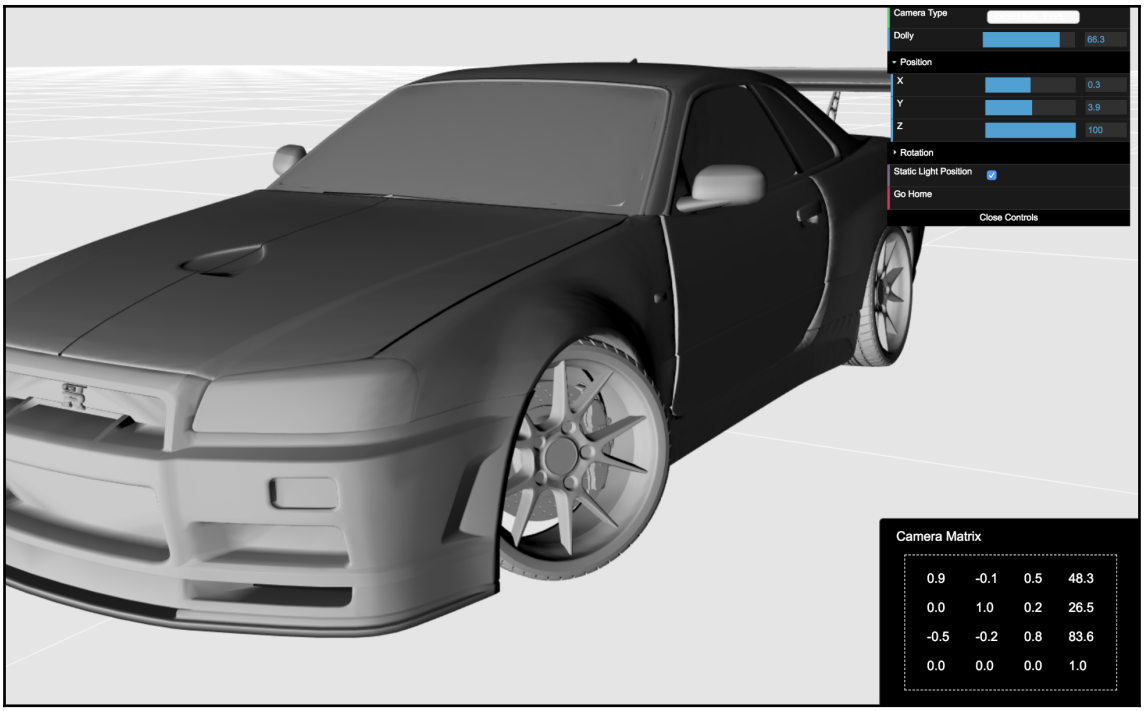
$$\boxed{C = T^{-1}R^{-1}}$$

$$\boxed{RT \neq TR}$$

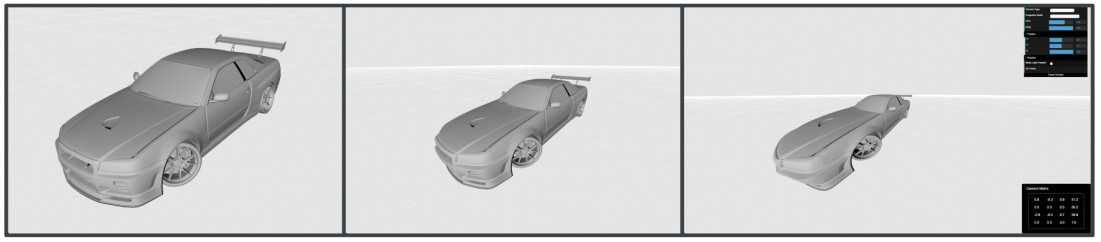



Camera Type
Dolly
13.7
Position
Rotation
Go Home
Close Controls
Camera Matrix
0.8 -0.2 0.5 18.0
0.0 0.9 0.4 16.6
-0.6 -0.3 0.8 26.9
0.0 0.0 0.0 1.0

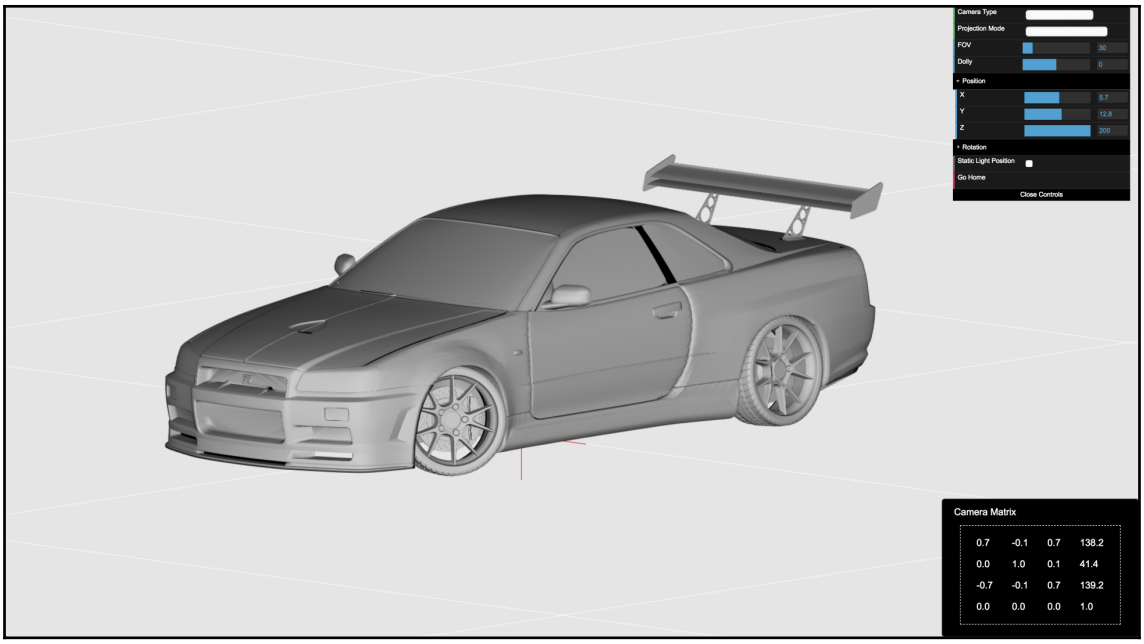

Camera Type
Dolly
27.7
Position
X
0
Y
16.3
Z
100
Rotation
Static Light Position
Go Home
Close Controls
Camera Matrix
0.7 -0.0 0.7 84.7
0.0 1.0 0.1 24.3
-0.7 -0.0 0.7 89.6
0.0 0.0 0.0 1.0

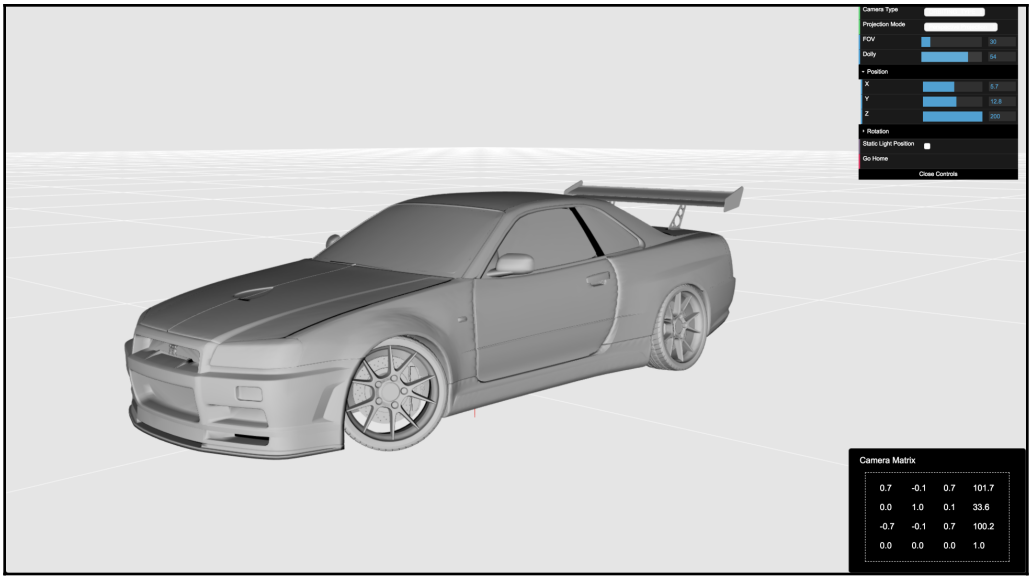


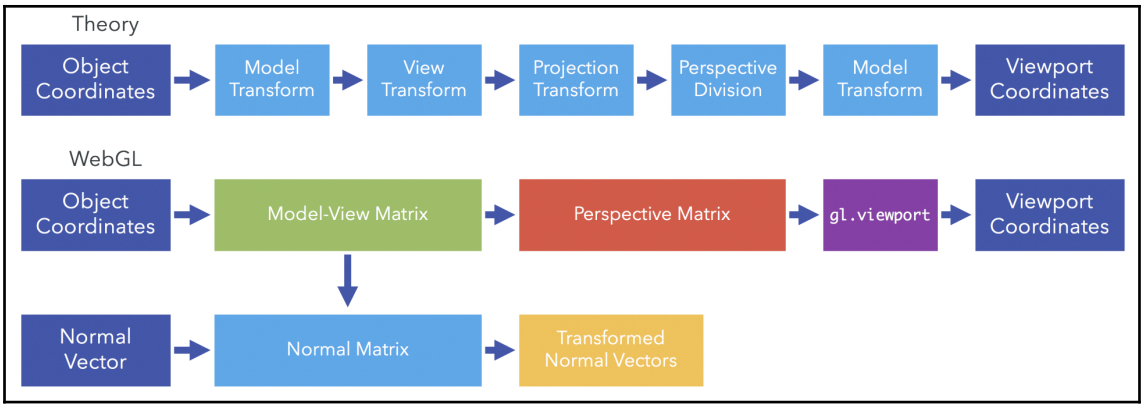
Field of View



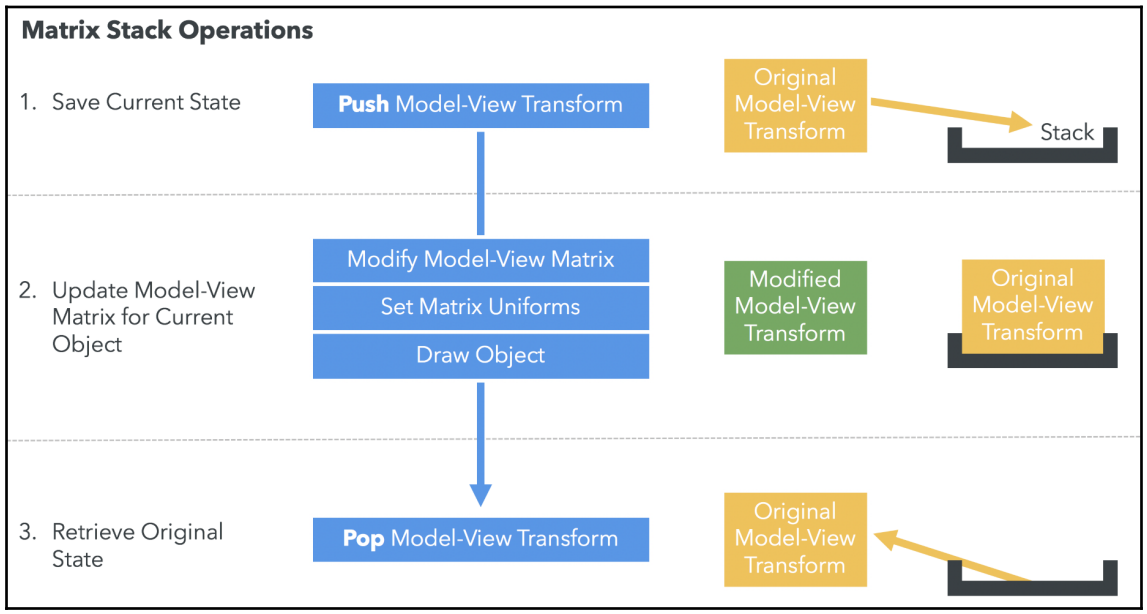
45 95 130

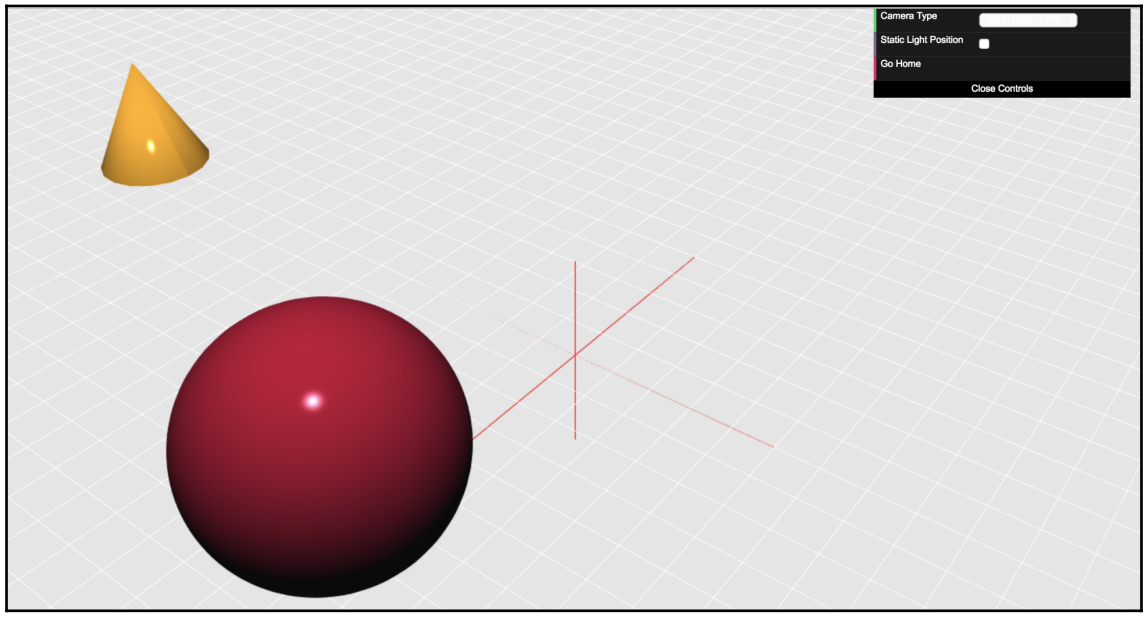

Camera Type
Projection Mode
FOV
30
Dolly
0
Position
X
5.7
Y
12.8
Z
200
Rotation
Static Light Position
Go Home
Close Controls
Camera Matrix
0.7 -0.1 0.7 138.2
0.0 1.0 0.1 41.4
-0.7 -0.1 0.7 139.2
0.0 0.0 0.0 1.0

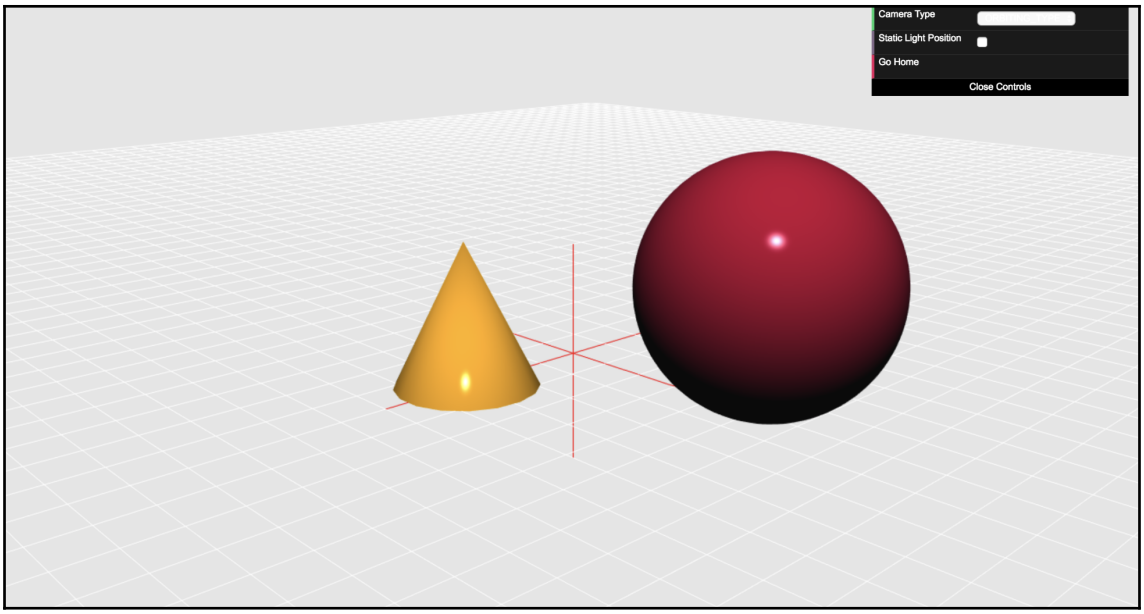

Camera Type
Projection Mode
FOV
30
Dolly
64
Position
X
5.7
Y
12.8
Z
200
Rotation
Static Light Position
Go Home
Close Controls
Camera Matrix
0.7 -0.1 0.7 101.7
0.0 1.0 0.1 33.6
-0.7 -0.1 0.7 100.2
0.0 0.0 0.0 1.0


Theory
Object Coordinates
Model Transform
View Transform
Projection Transform
Perspective Division
Model Transform
Viewport Coordinates
WebGL
Object Coordinates
Model-View Matrix
Perspective Matrix
gl.viewport
Viewport Coordinates
Normal Vector
Normal Matrix
Transformed Normal Vectors

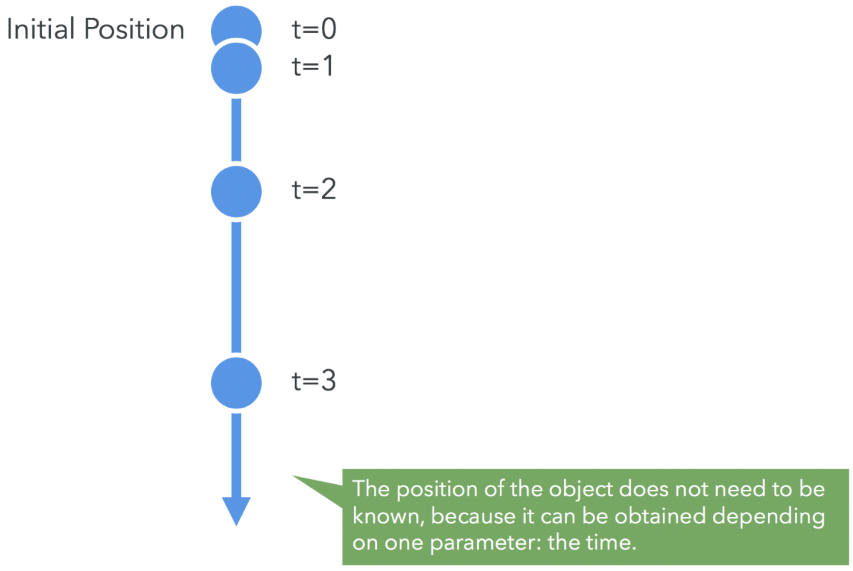
Chapter 5: Animations





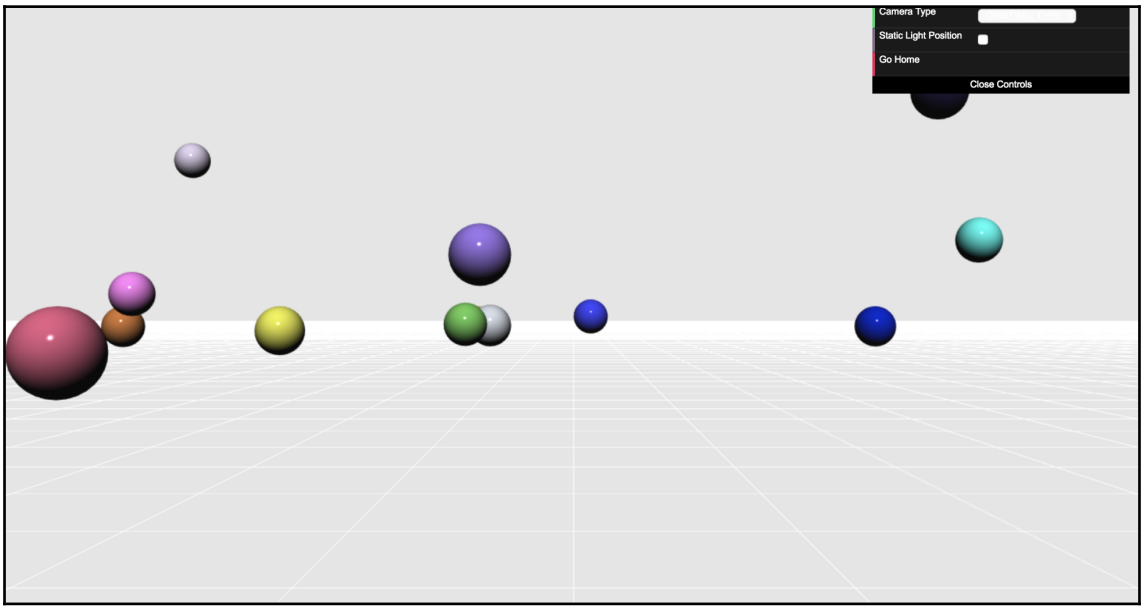


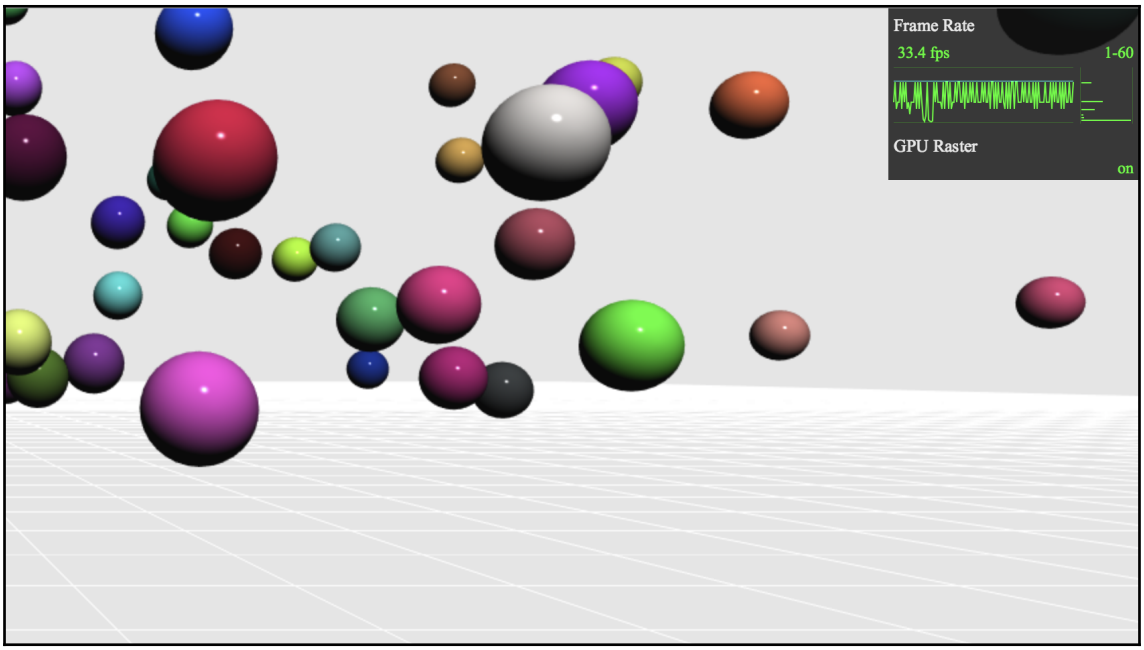
Parametric Curves: Free Fall



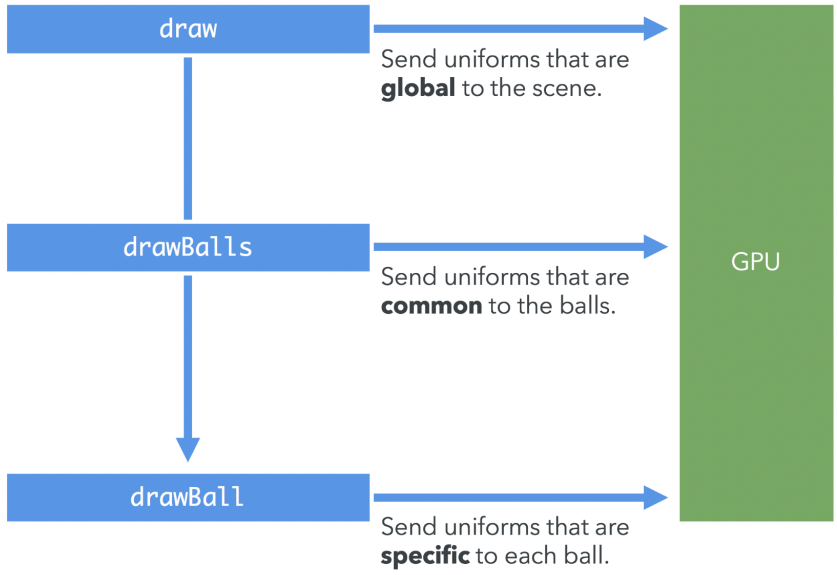
$$\boxed{h = H_0 + V_0 t - \frac{1}{2} g t^2}$$

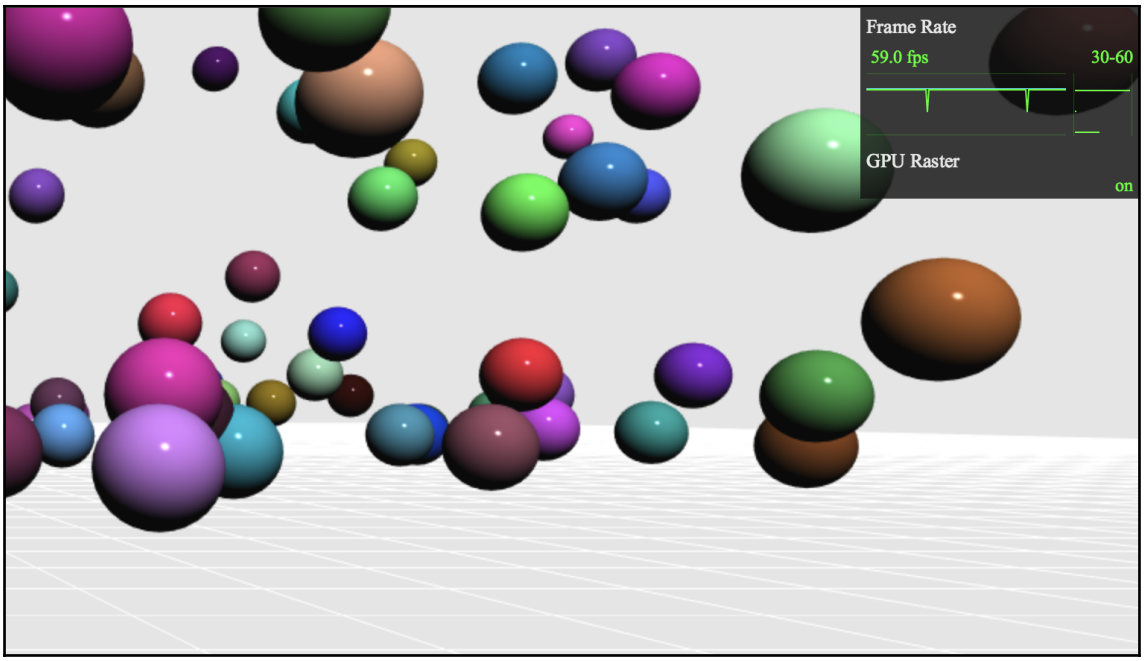
$$\boxed{9.8 m/s^2}$$

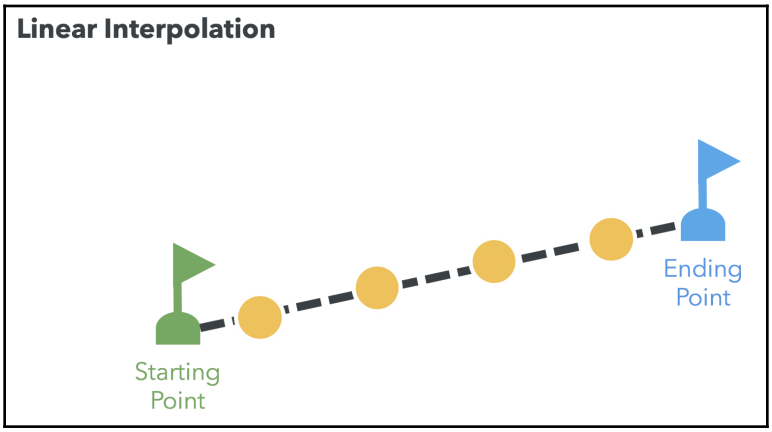


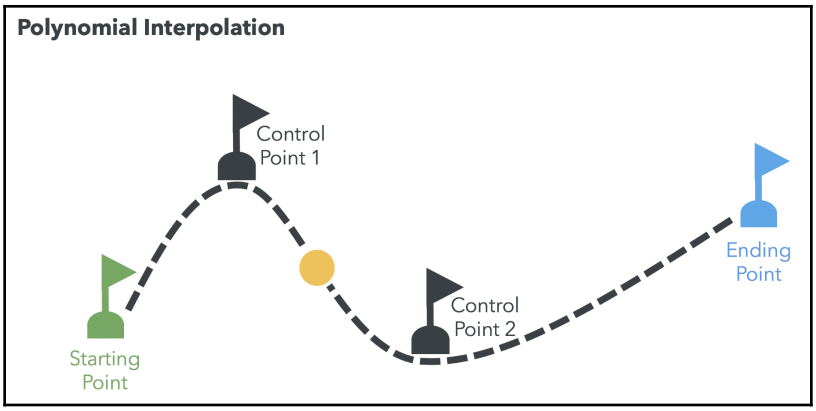


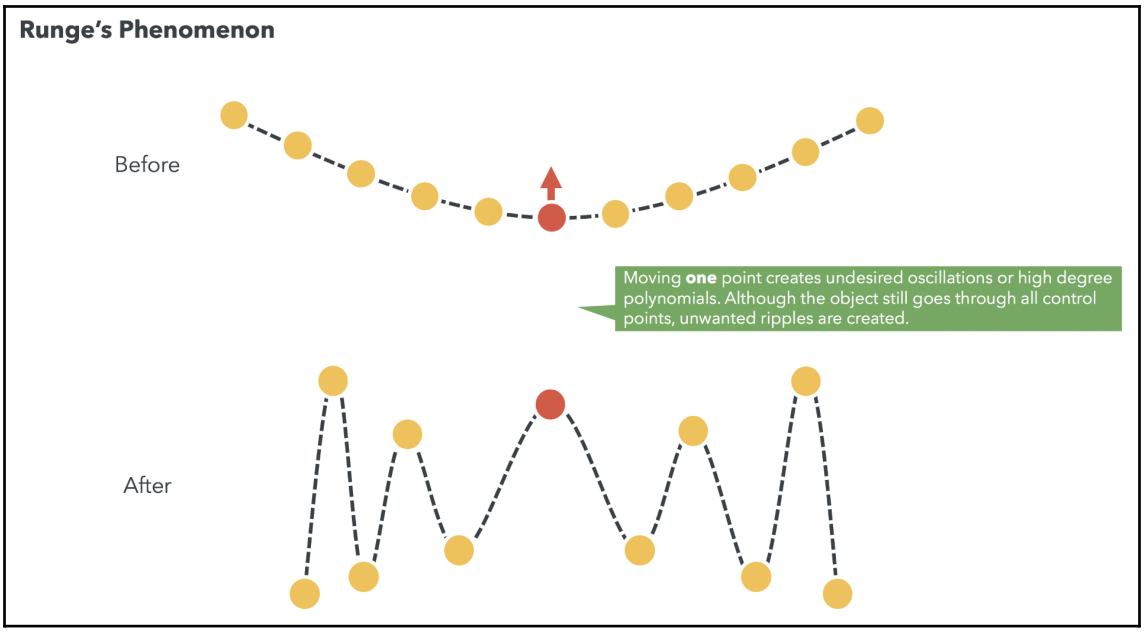
Optimizing Batch Performance

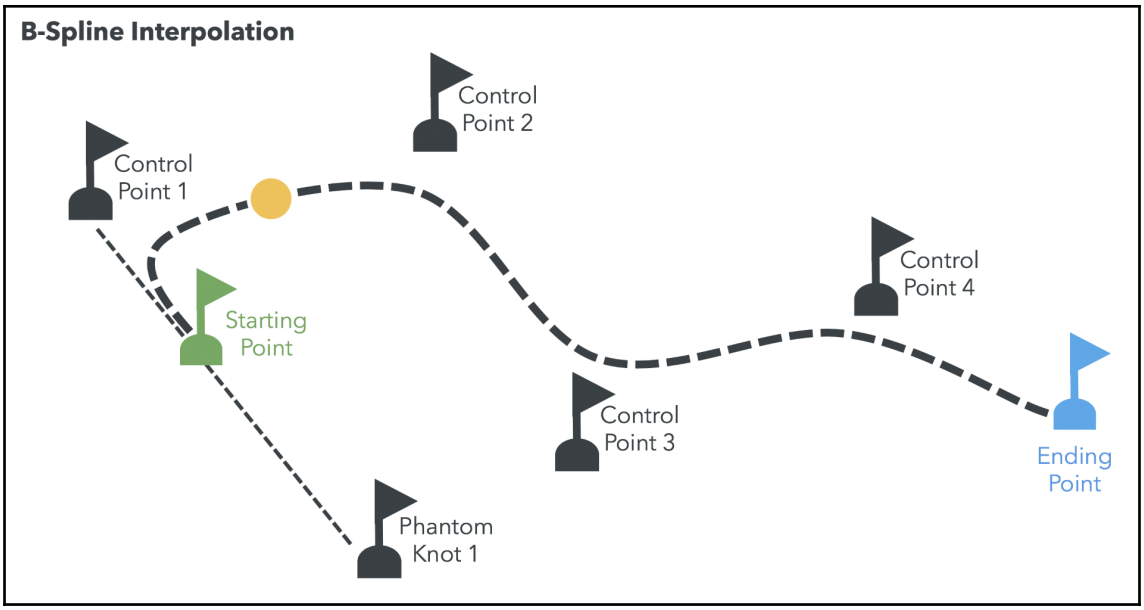


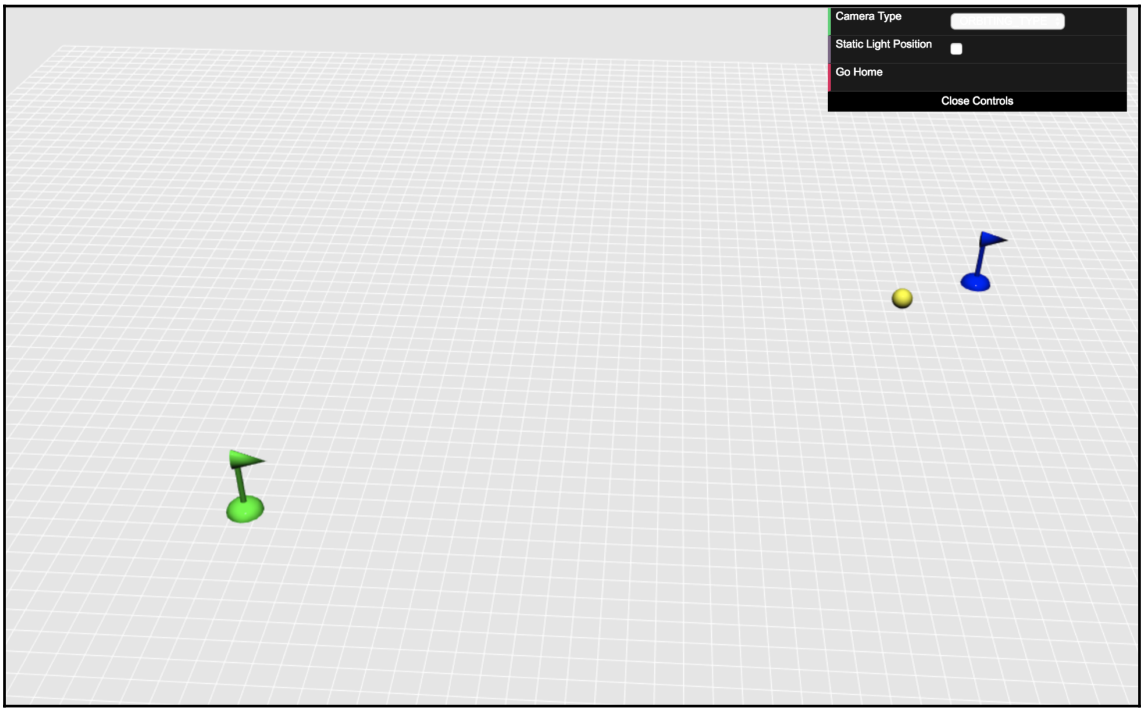

Frame Rate
59.0 fps
30-60
GPU Raster
on

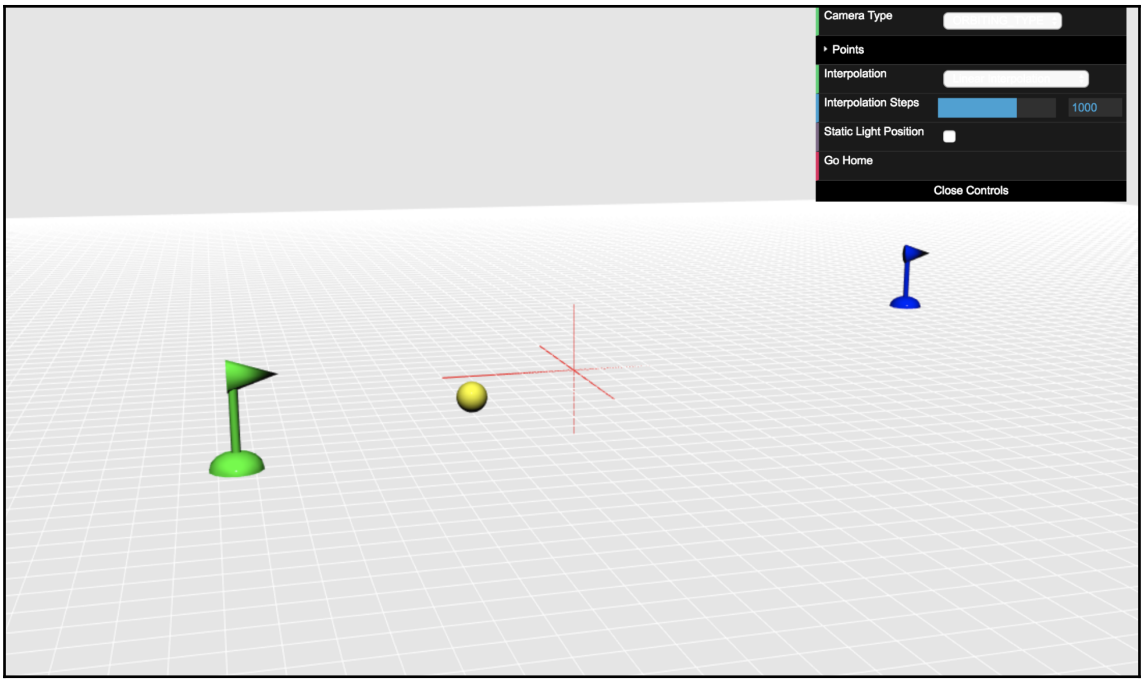

Linear Interpolation
Starting Point
Ending Point

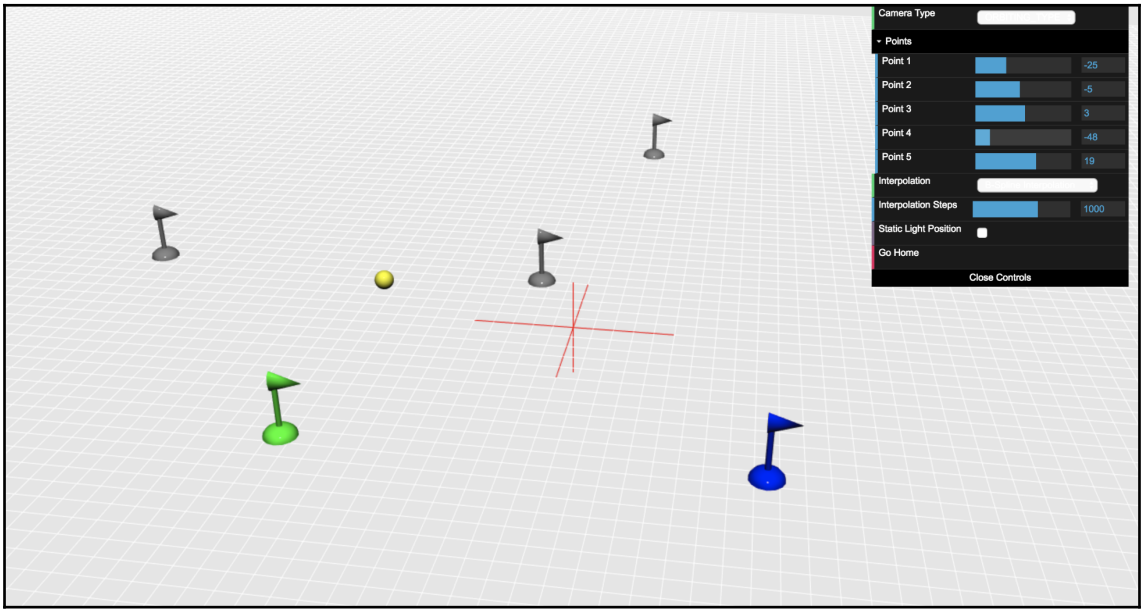

Polynomial Interpolation
Control Point 1
Control Point 2
Starting Point
Ending Point


Runge's Phenomenon
Before
Moving one point creates undesired oscillations or high degree polynomials. Although the object still goes through all control points, unwanted ripples are created.
After

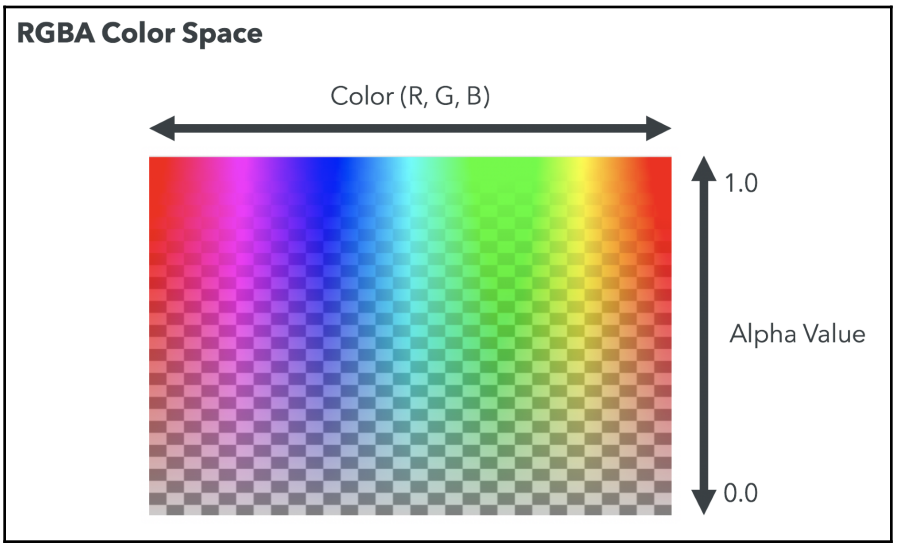

B-Spline Interpolation
Control Point 1
Control Point 2
Starting Point
Control Point 3
Control Point 4
Ending Point
Phantom Knot 1

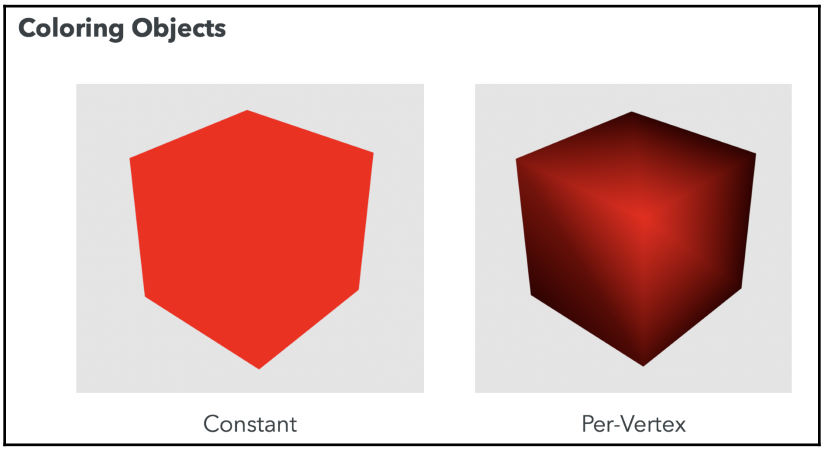

Camera Type
Static Light Position
Go Home
Close Controls


Camera Type
Points
Interpolation
Interpolation Steps
1000
Static Light Position
Go Home
Close Controls

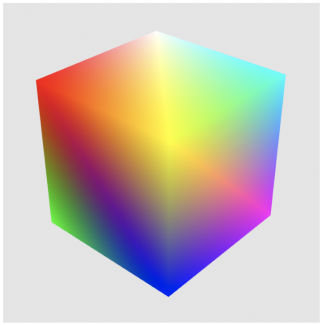

Camera Type
Points
Point 1
-25
Point 2
-5
Point 3
3
Point 4
-48
Point 5
19
Interpolation
Interpolation Steps
1000
Static Light Position
Go Home
Close Controls

Chapter 6:
Colors, Depth Testing, and Alpha Blending

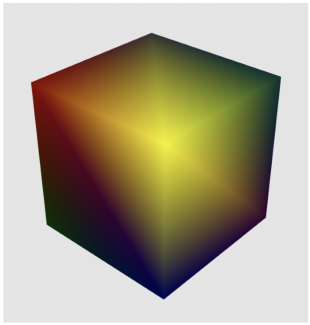




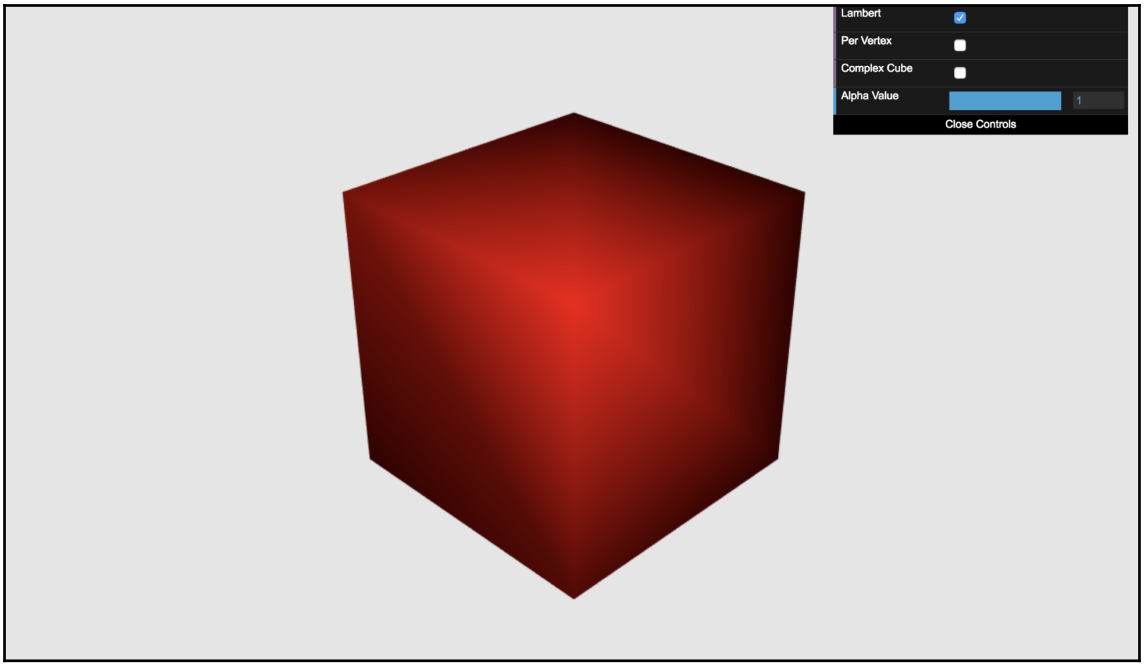
Per-Vertex Colors



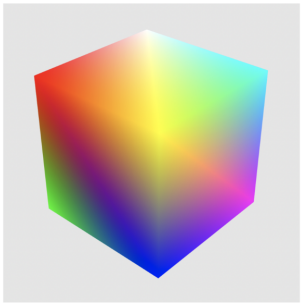
Not Using the
Lambertian Term



Using the
Lambertian Term

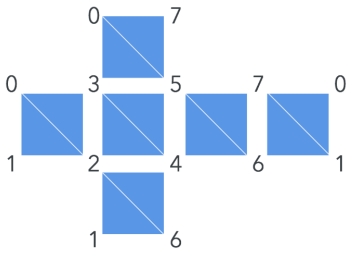


Per-Vertex Coloring: Color Interpolation



Simple Cube: Vertices are defined *once*.

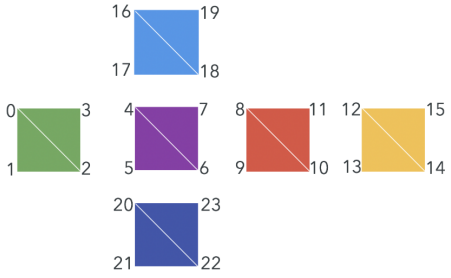
Result: Each vertex has one color and colors are interpolated in all directions on the surface of the cube.

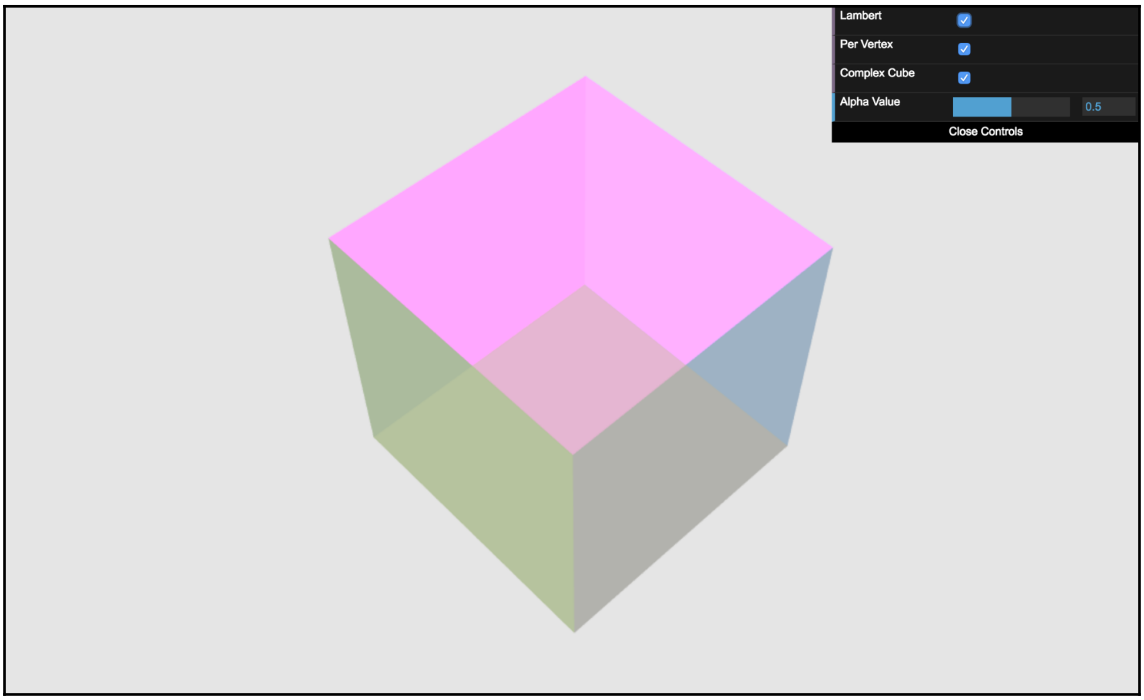


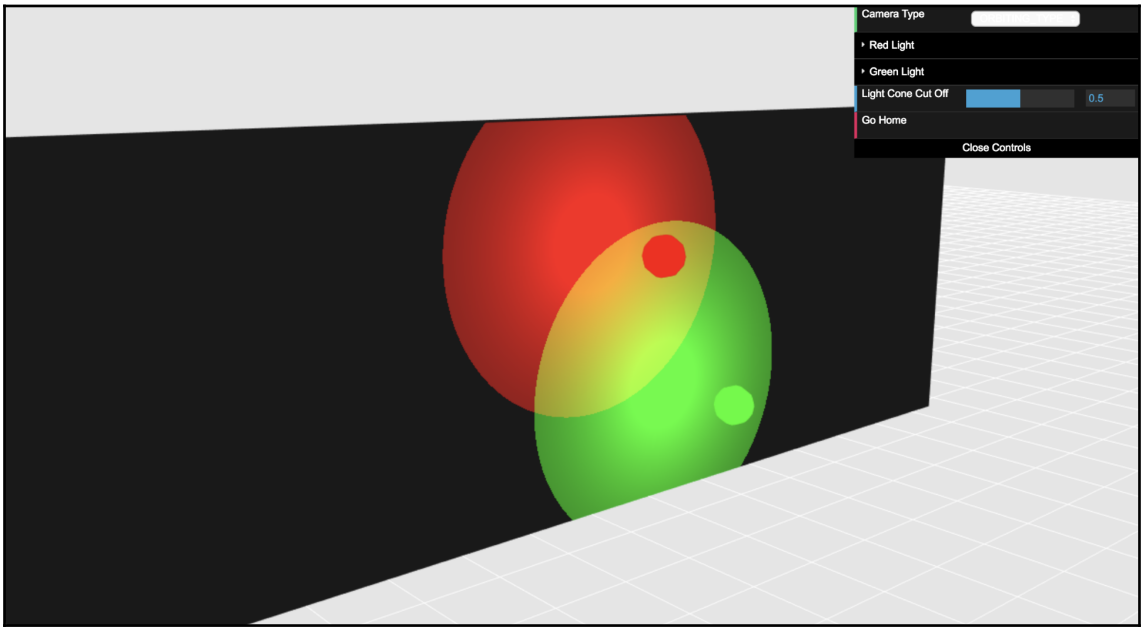
Per-Vertex Coloring: Color Interpolation

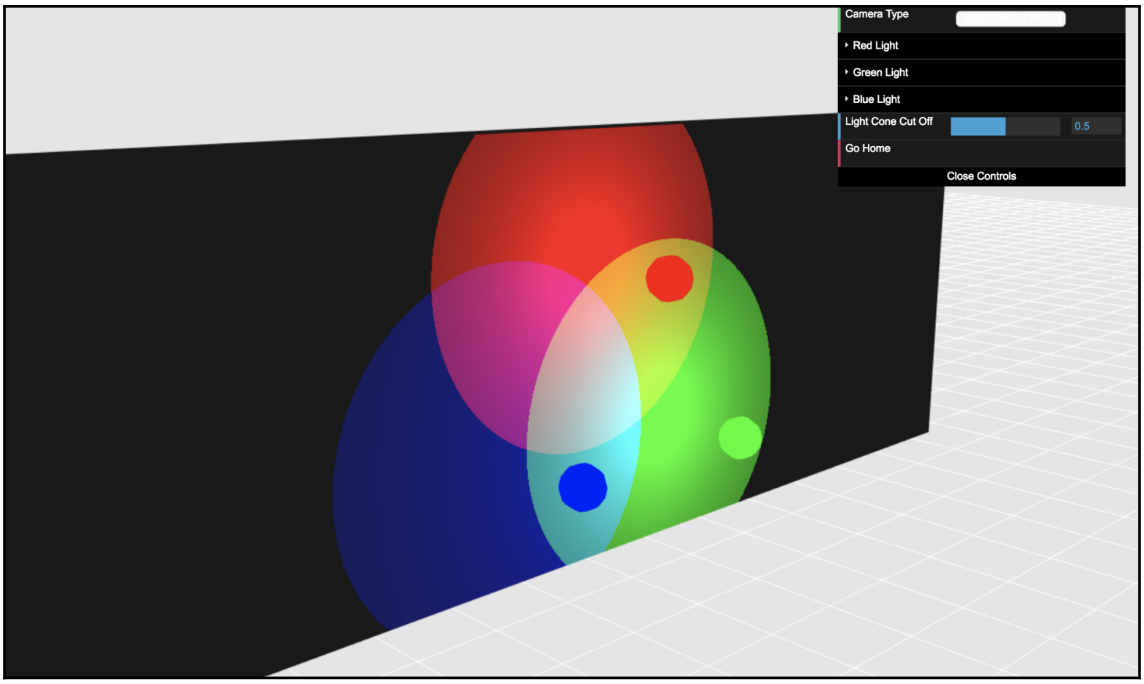
Complex Cube: Each vertex is repeated *three* times in the vertex buffer object.

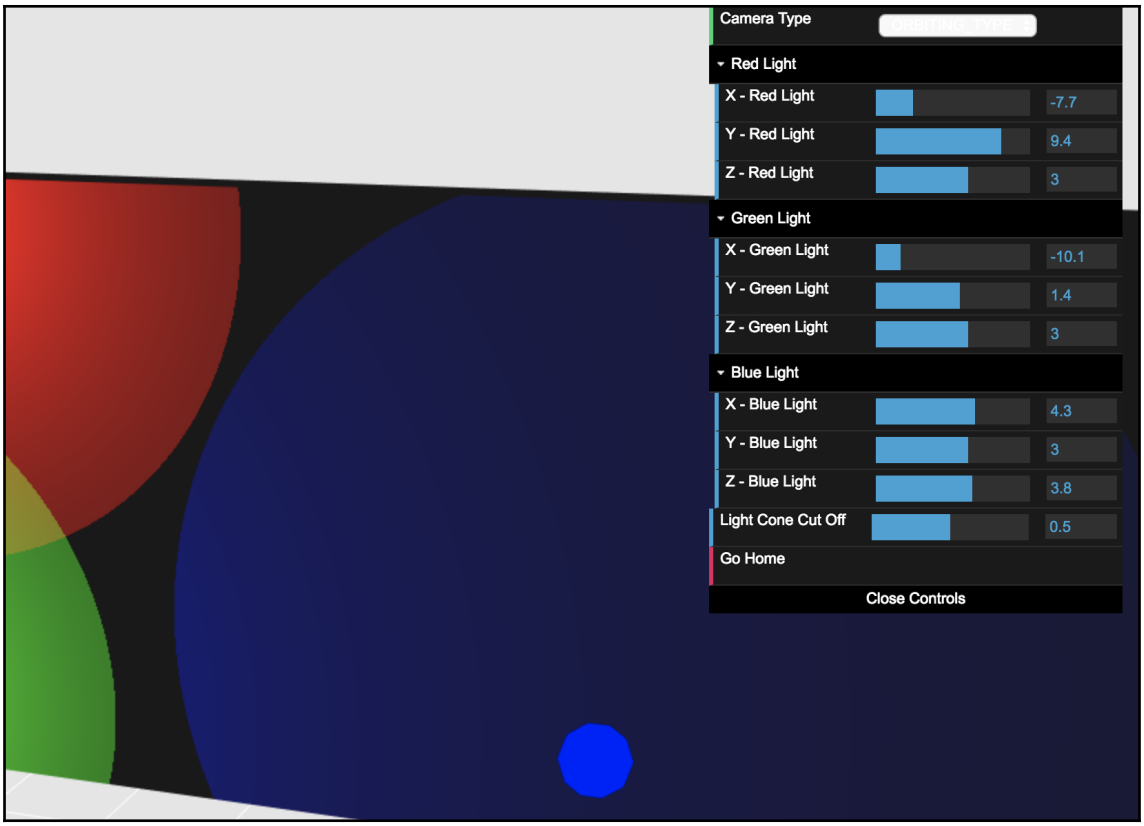
Result: Each vertex has three colors and each face has its own color.

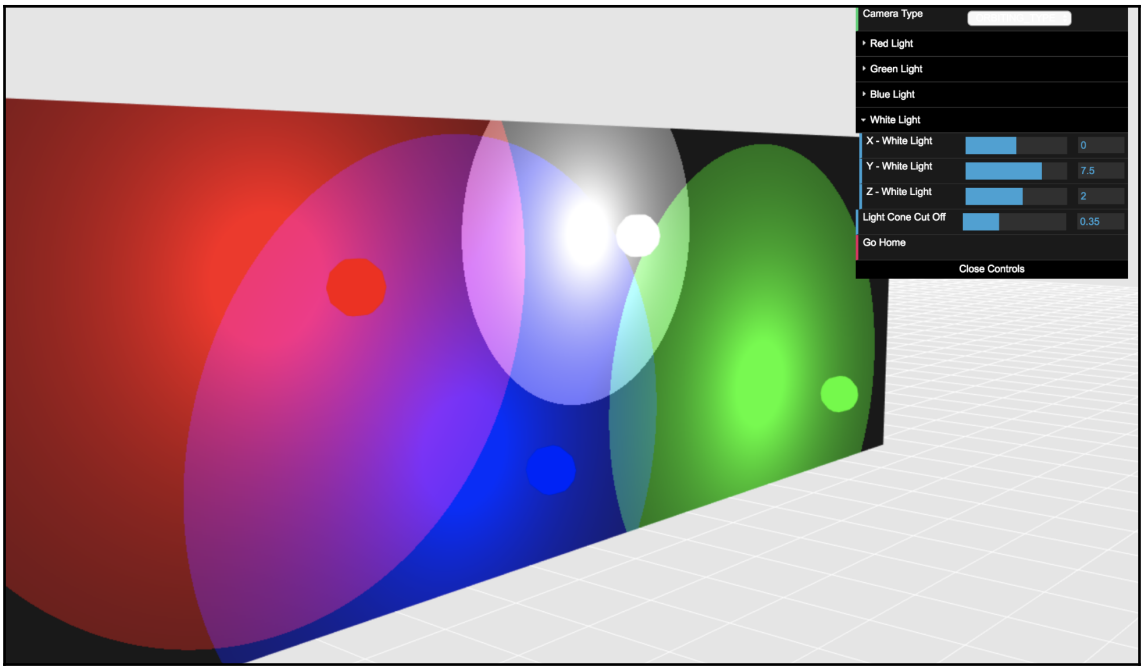


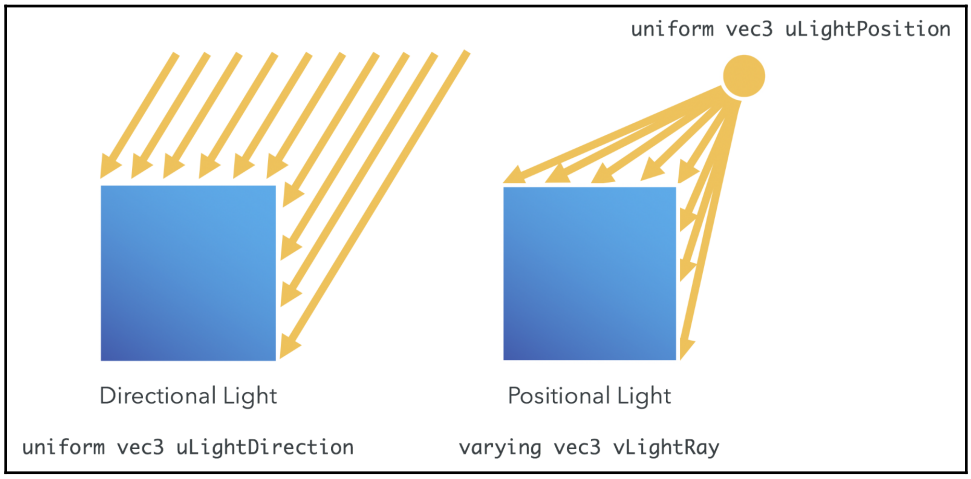

Lambert
Per Vertex
Complex Cube
Alpha Value
0.5
Close Controls


Camera Type
Red Light
Green Light
Light Cone Cut Off
0.5
Go Home
Close Controls


Camera Type
Red Light
Green Light
Blue Light
Light Cone Cut Off
0.5
Go Home
Close Controls


Camera Type
Red Light
X - Red Light
-7.7
Y - Red Light
9.4
Z - Red Light
3
Green Light
X - Green Light
-10.1
Y - Green Light
1.4
Z - Green Light
3
Blue Light
X - Blue Light
4.3
Y - Blue Light
3
Z - Blue Light
3.8
Light Cone Cut Off
0.5
Go Home
Close Controls


Camera Type
Red Light
Green Light
Blue Light
White Light
X - White Light
0
Y - White Light
7.5
Z - White Light
2
Light Cone Cut Off
0.35
Go Home
Close Controls


uniform vec3 uLightPosition
Directional Light
Positional Light
uniform vec3 uLightDirection
varying vec3 vLightRay

Directional Point Lights

Implicit Light Direction

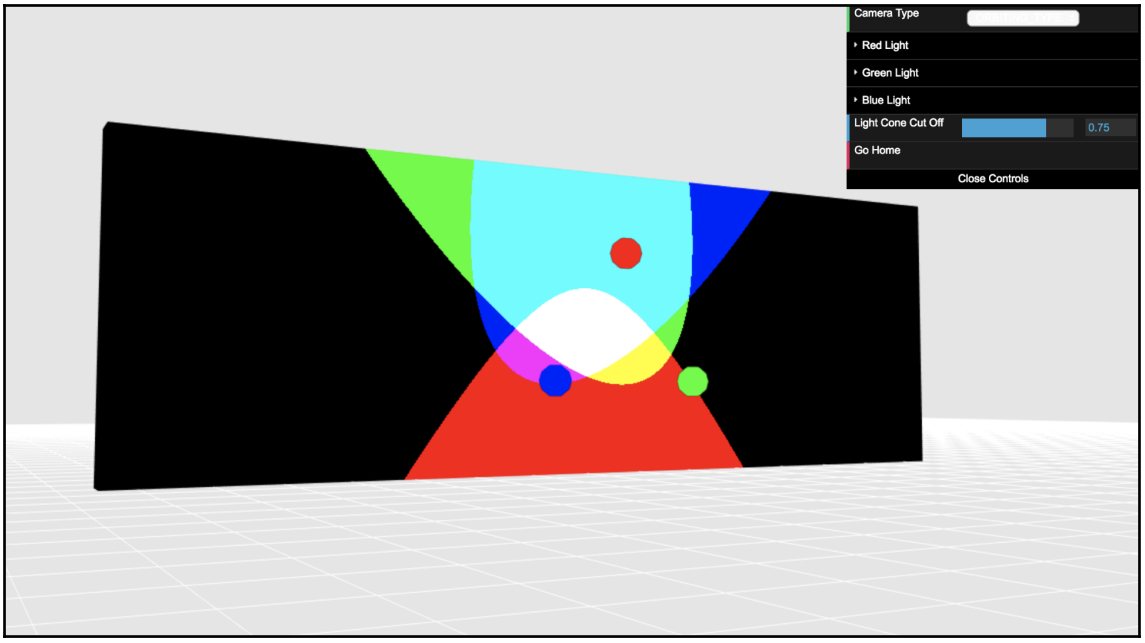
Explicit Light Direction

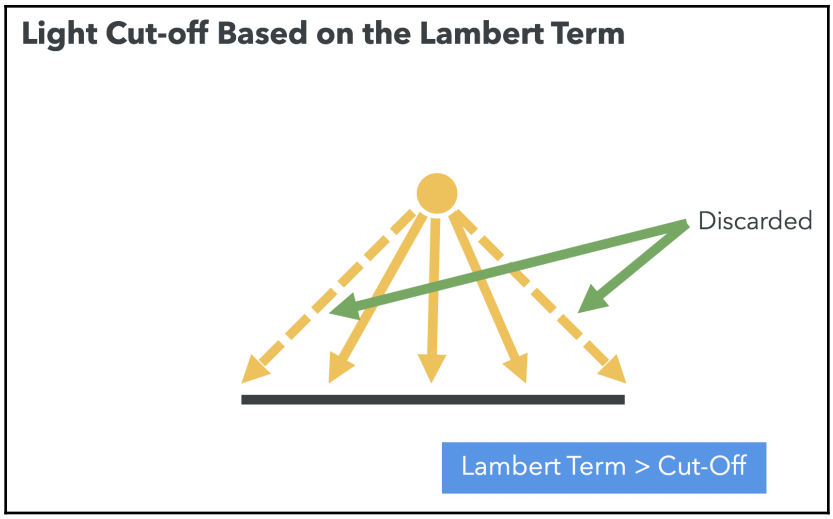
The trick is to subtract the light direction from the normal.

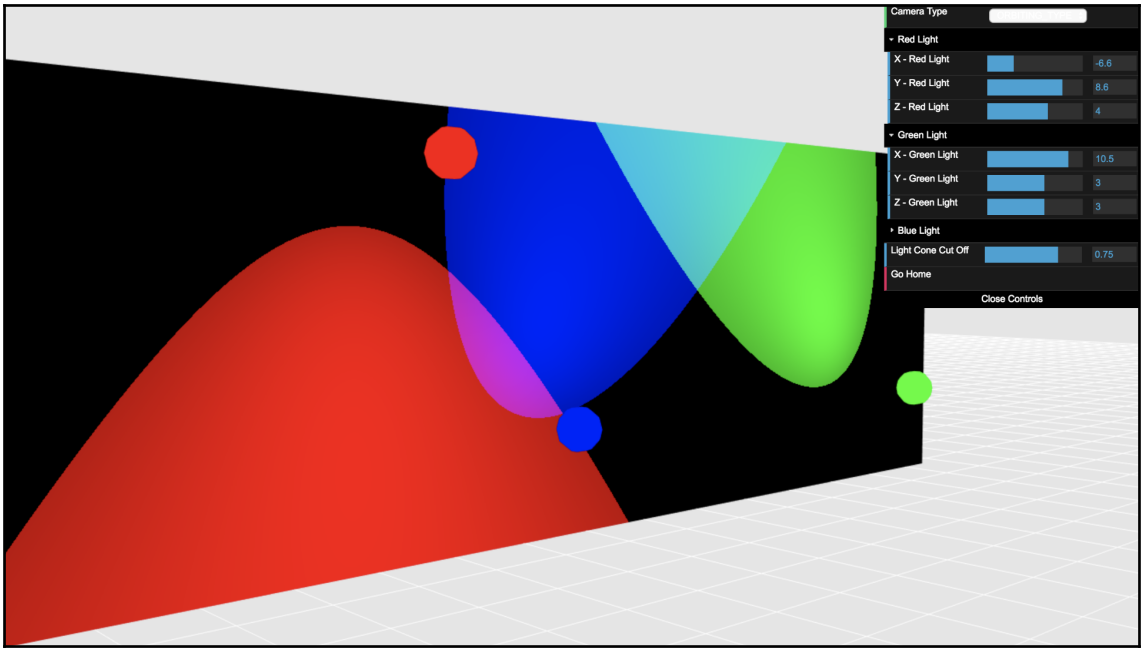
Light Ray

Normal

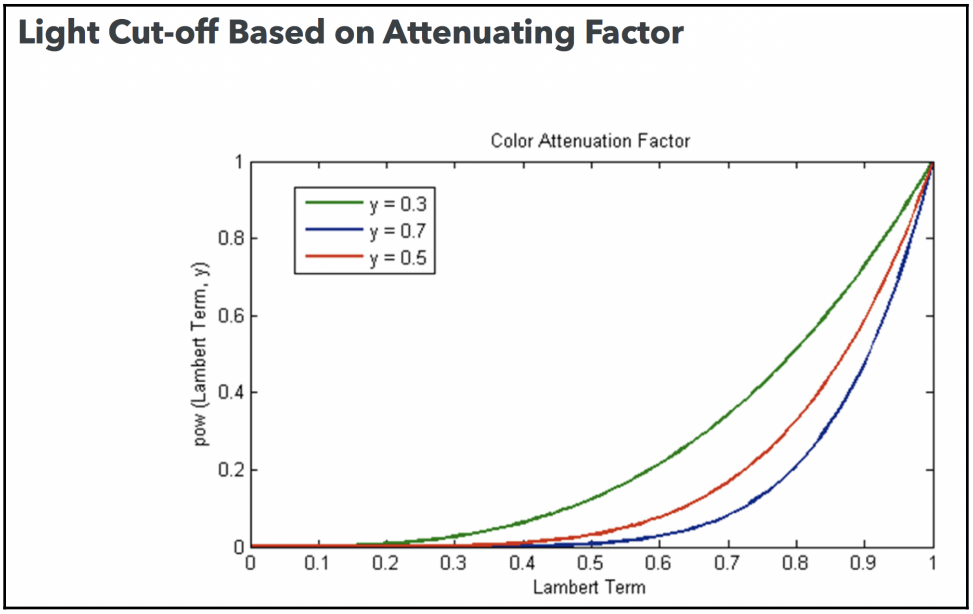
Light Direction

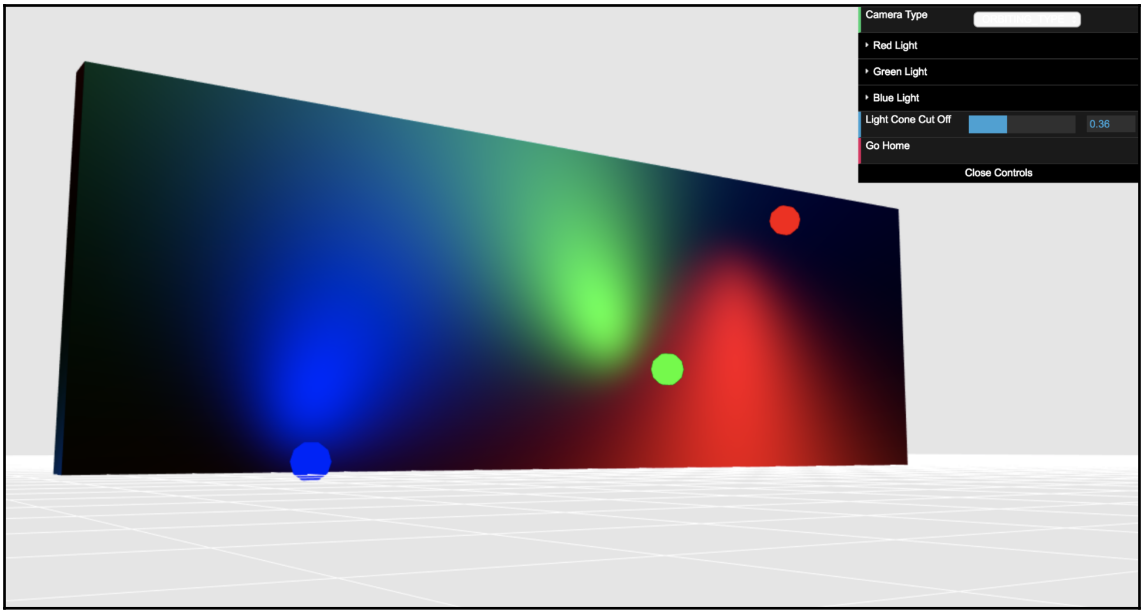


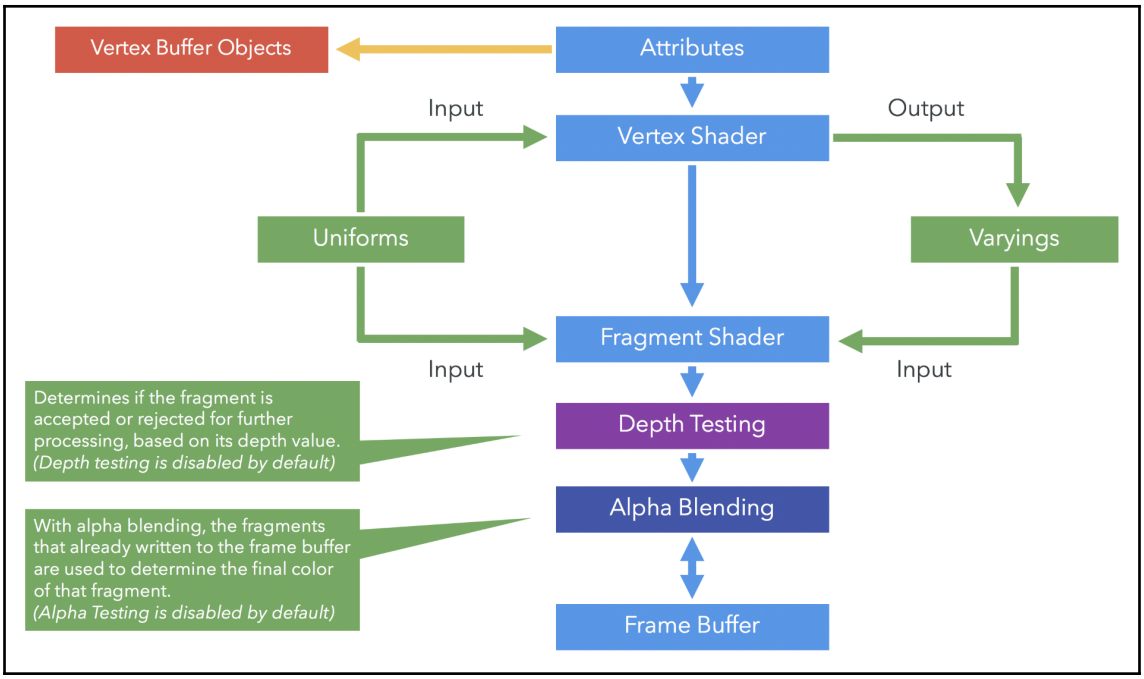

Light Cut-off Based on the Lambert Term
Discarded
Lambert Term > Cut-Off


Camera Type
Red Light
X - Red Light
-6.6
Y - Red Light
8.6
Z - Red Light
4
Green Light
X - Green Light
10.5
Y - Green Light
3
Z - Green Light
3
Blue Light
Light Cone Cut Off
0.75
Go Home
Close Controls

Light Cut-off Based on Attenuating Factor



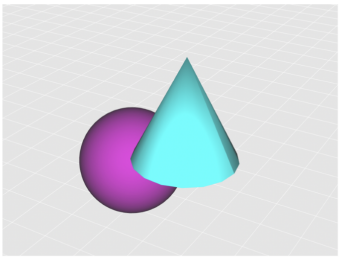




Depth Testing in Action

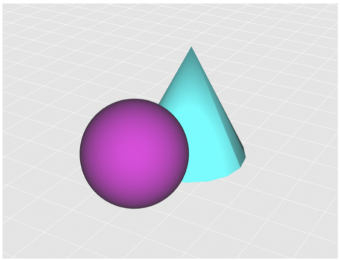
```
gl.disable(gl.DEPTH_TEST)
```

This is incorrect, as the sphere should be in front of the cone.



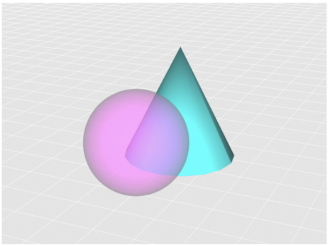
```
gl.enable(gl.DEPTH_TEST)
```

Depth testing produces consist object occlusion where there are overlapping objects. In this scene, the sphere has been rendered first.

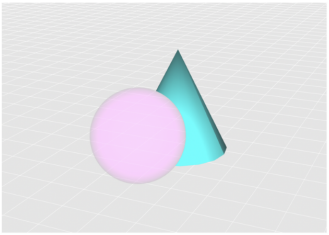


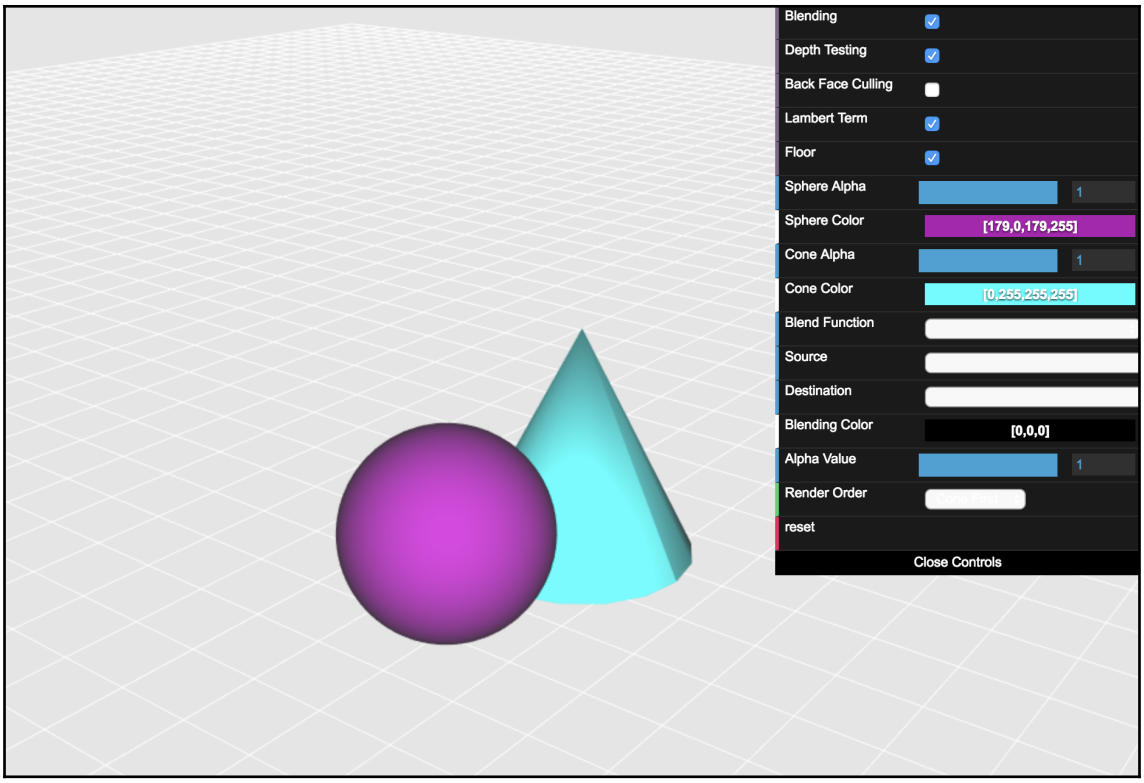
Rendering Order in Blending Operations

Back to Front Order
The cone is rendered first.
The overlapping sphere fragments pass the depth test are available for blending.



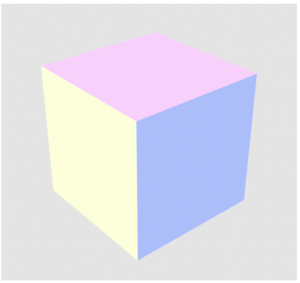
Front to Back Order
The sphere is rendered first.
The overlapping cone fragments do not pass the depth test. Blending is not possible.




Blending
Depth Testing
Back Face Culling
Lambert Term
Floor
Sphere Alpha
1
Sphere Color
[179,0,179,255]
Cone Alpha
1
Cone Color
[0,255,255,255]
Blend Function
Source
Destination
Blending Color
[0,0,0]
Alpha Value
1
Render Order
reset
Close Controls

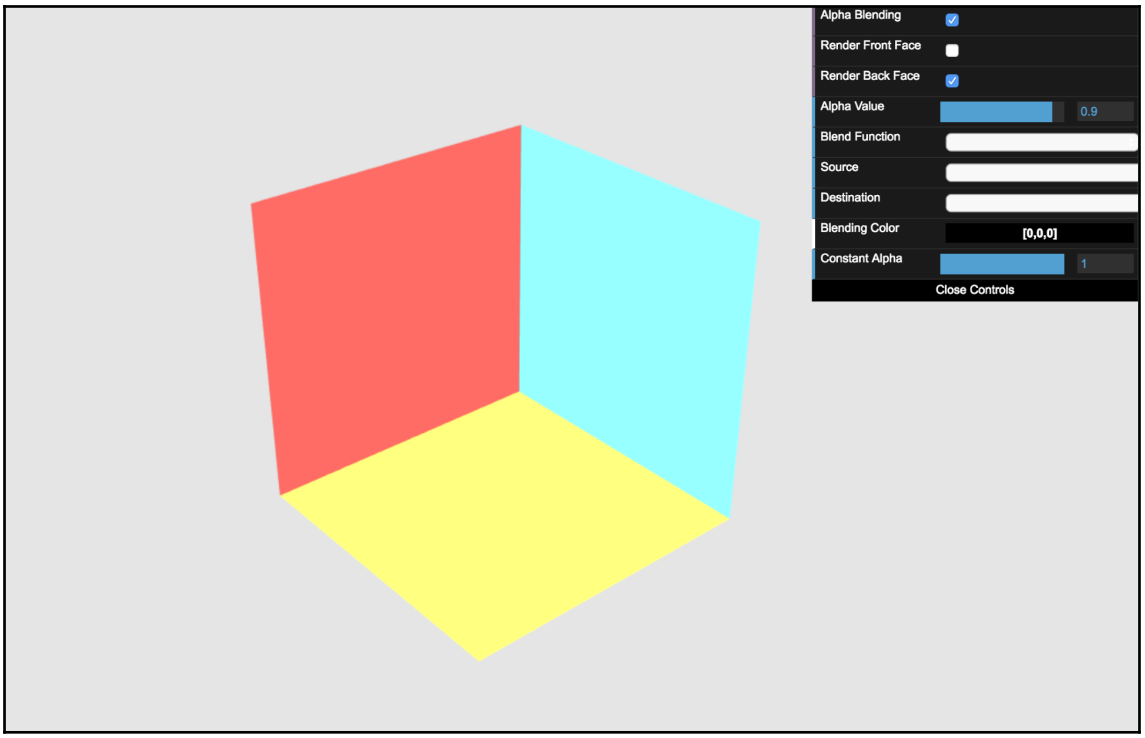
Alpha Blending in Action

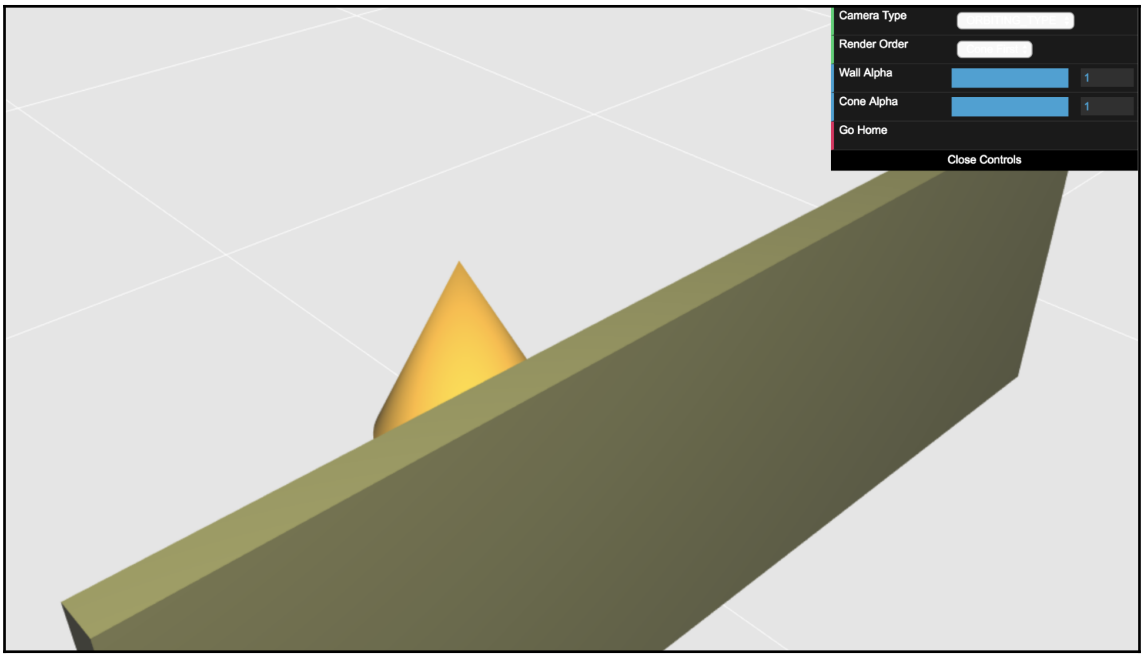
Back Face
These fragments are already present in the frame buffer.

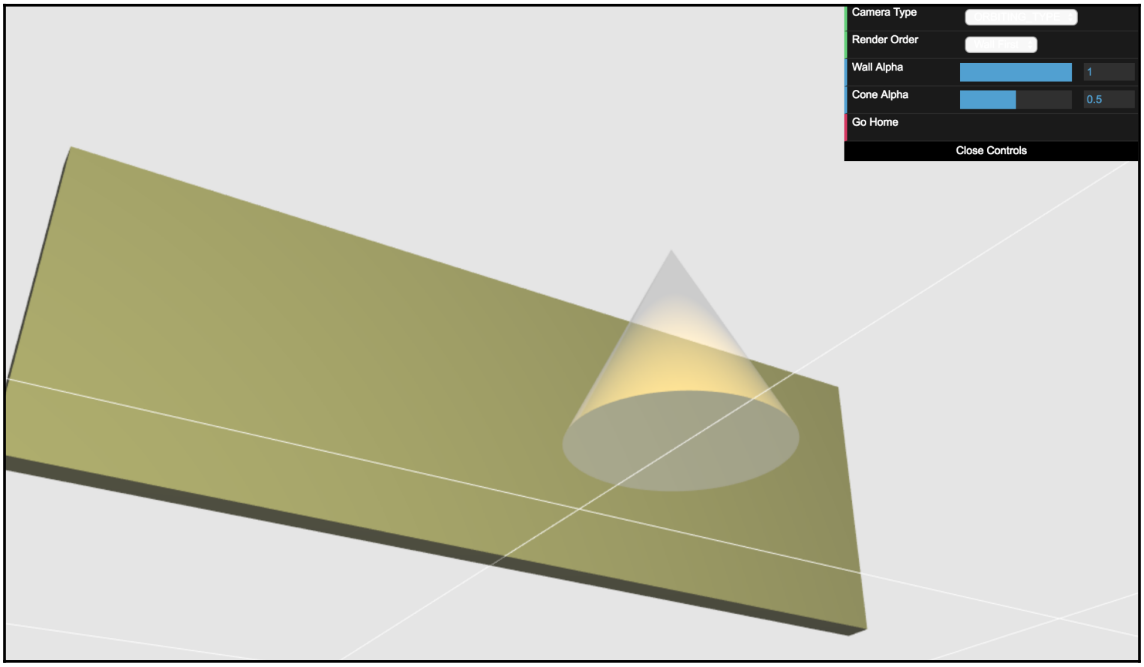


Front Face
These newer fragments are going to be blended with the fragments already present in the frame buffer.

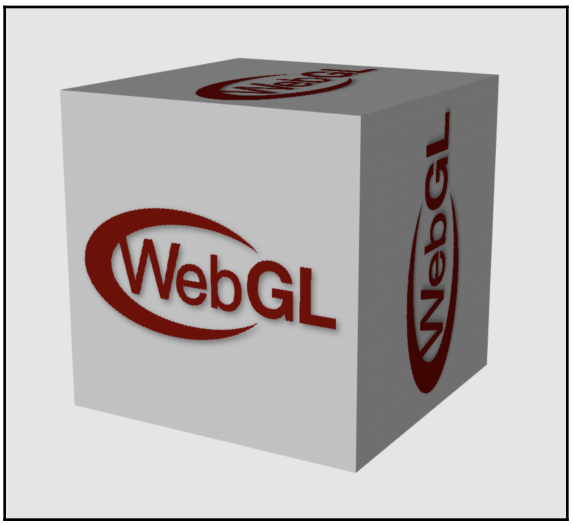
Blending Result
Using an alpha value of 0.5.

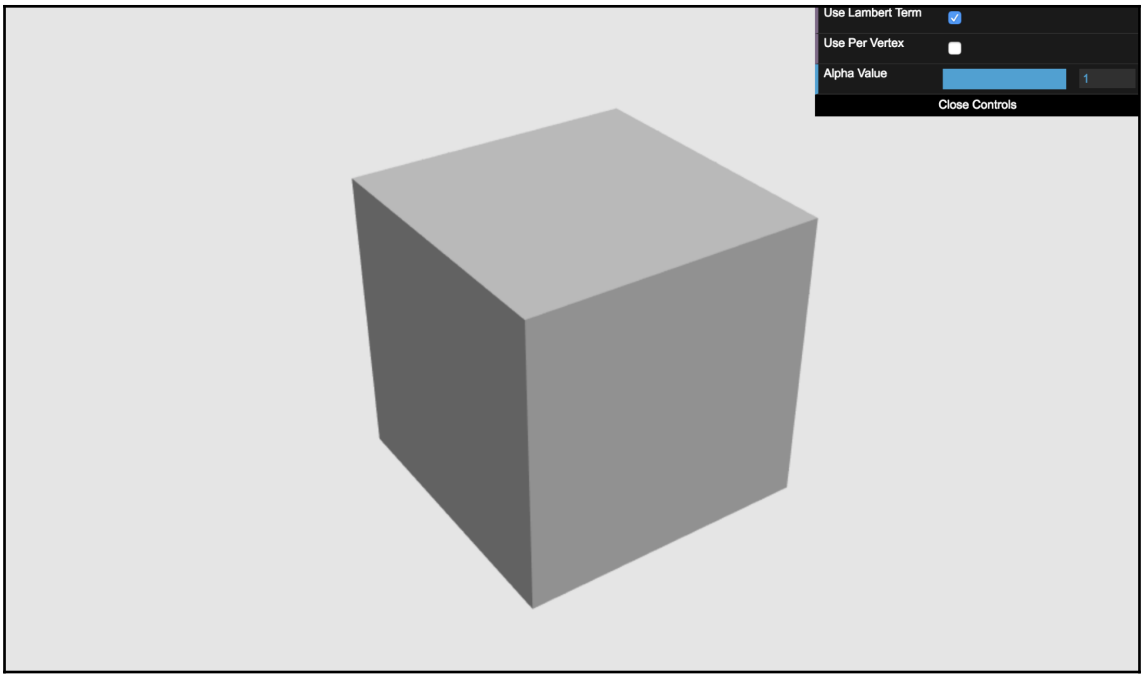



Camera Type
Render Order
Wall Alpha
1
Cone Alpha
1
Go Home
Close Controls


Camera Type
Render Order
Wall Alpha
1
Cone Alpha
0.5
Go Home
Close Controls

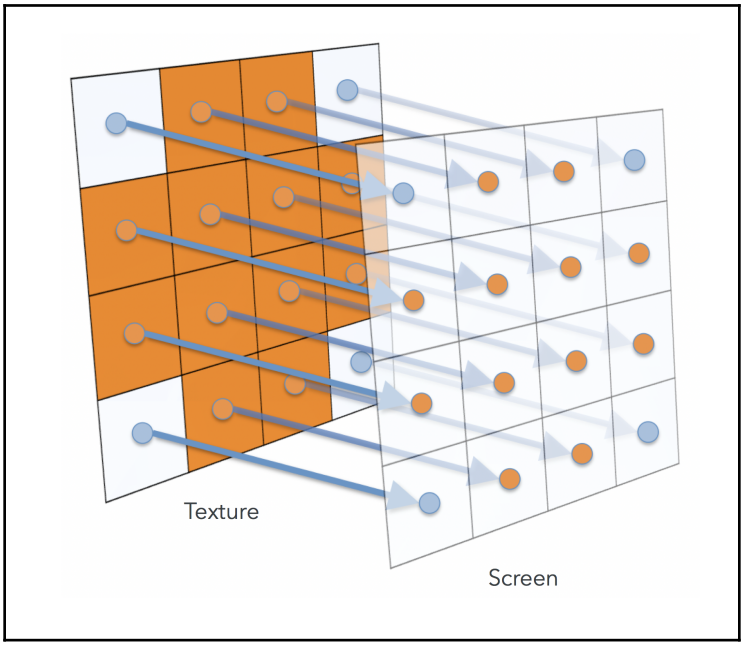
Chapter 7: Textures

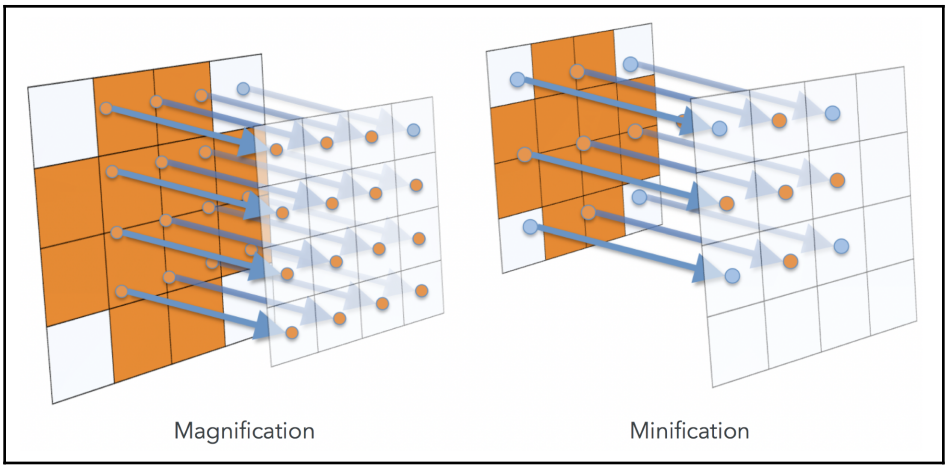


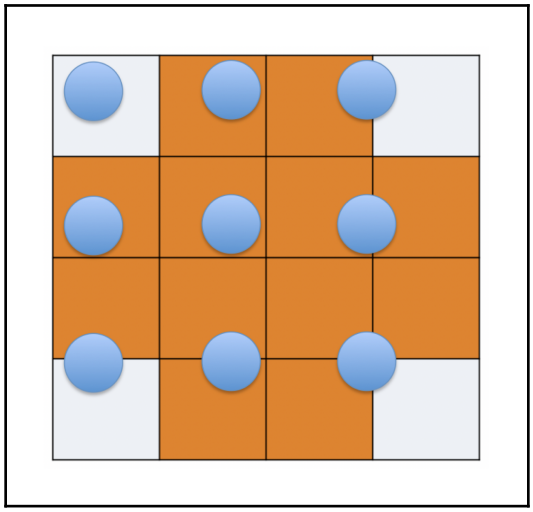

Use Lambert Term
Use Per Vertex
Alpha Value
1
Close Controls

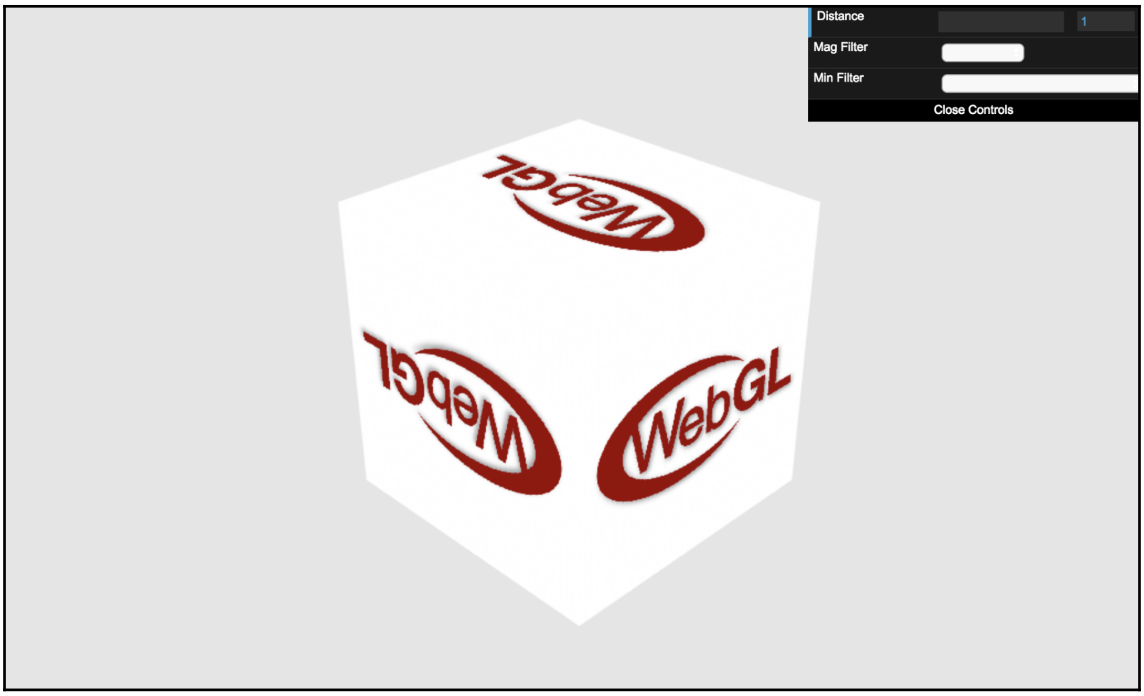

Use Lambert Term
Use Per Vertex
Alpha Value
1
Close Controls
WebGL

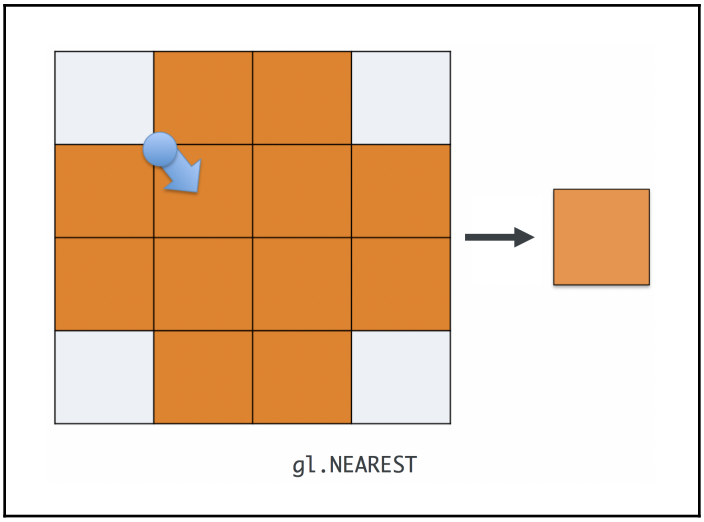

Use Lambert Term
Use Per Vertex
Alpha Value
1
Close Controls

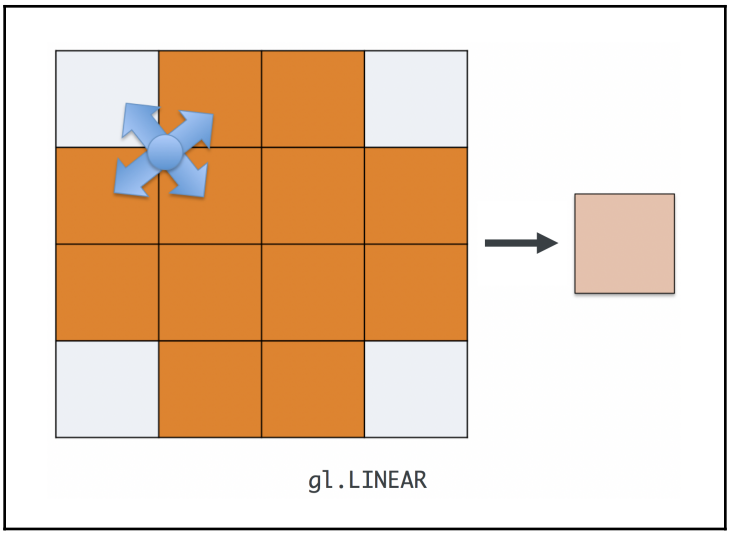

Texture
Screen


Magnification
Minification



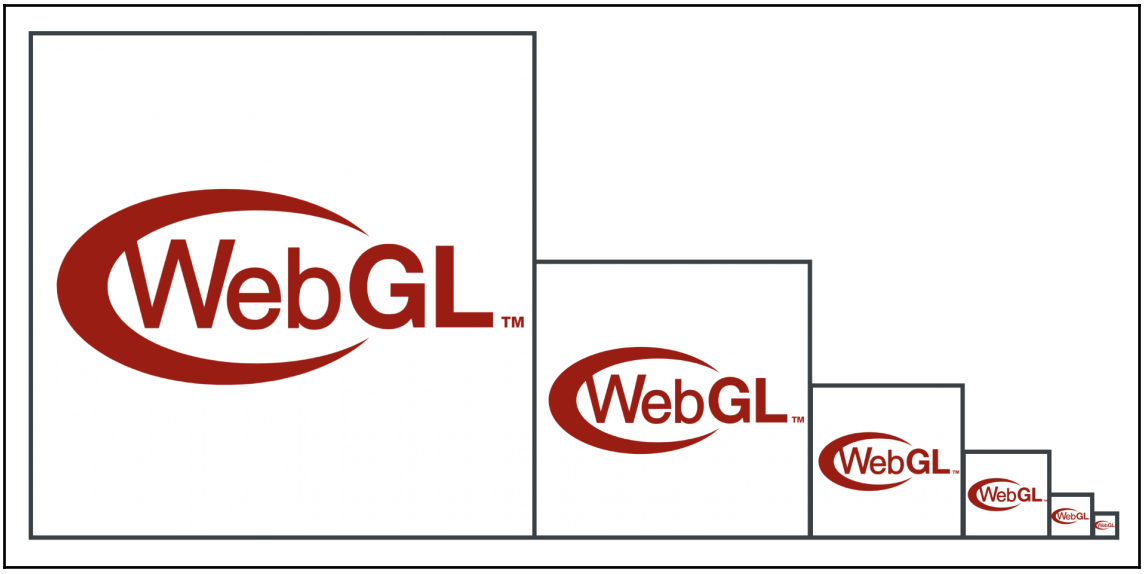

Distance
1
Mag Filter
Min Filter
Close Controls
WebGL
WebGL
WebGL

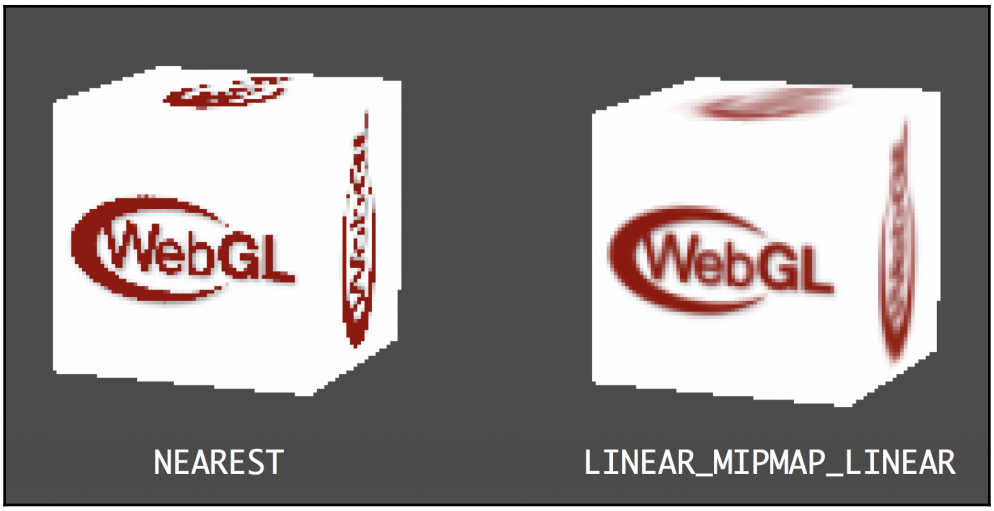

gl.NEAREST


gl.LINEAR

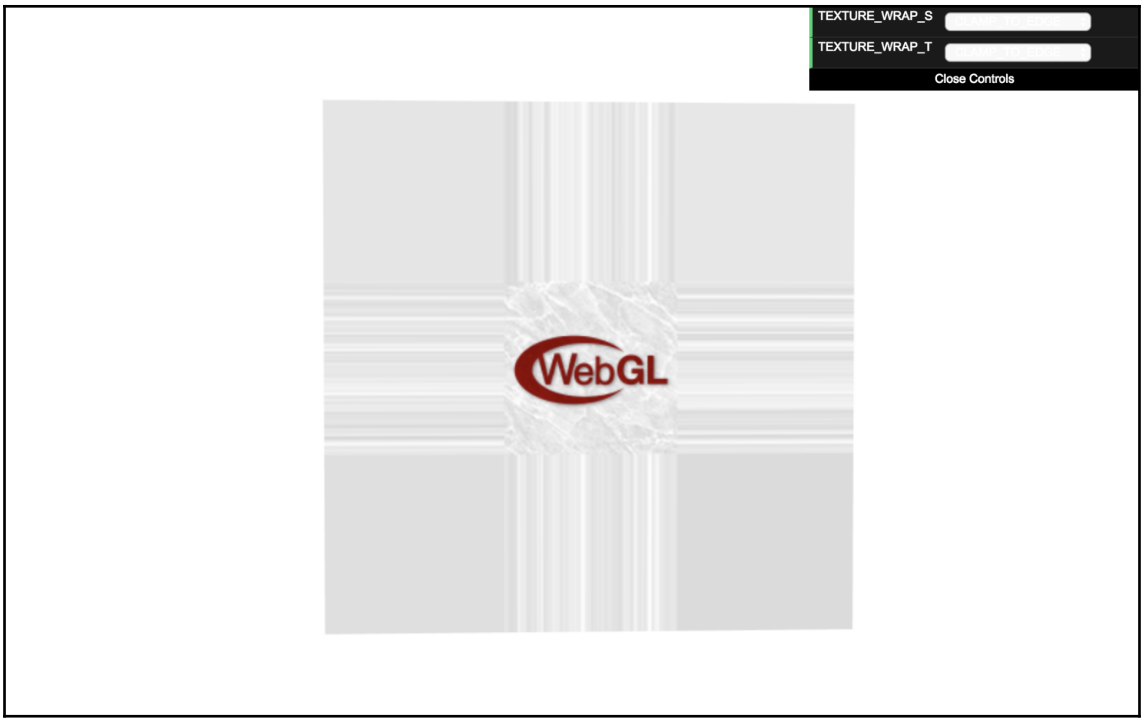

Distance
1
Mag Filter
Min Filter
Close Controls


WebGL
WebGL
Scaled up 400%


WebGL™
WebGL™
WebGL™
WebGL
WebGL

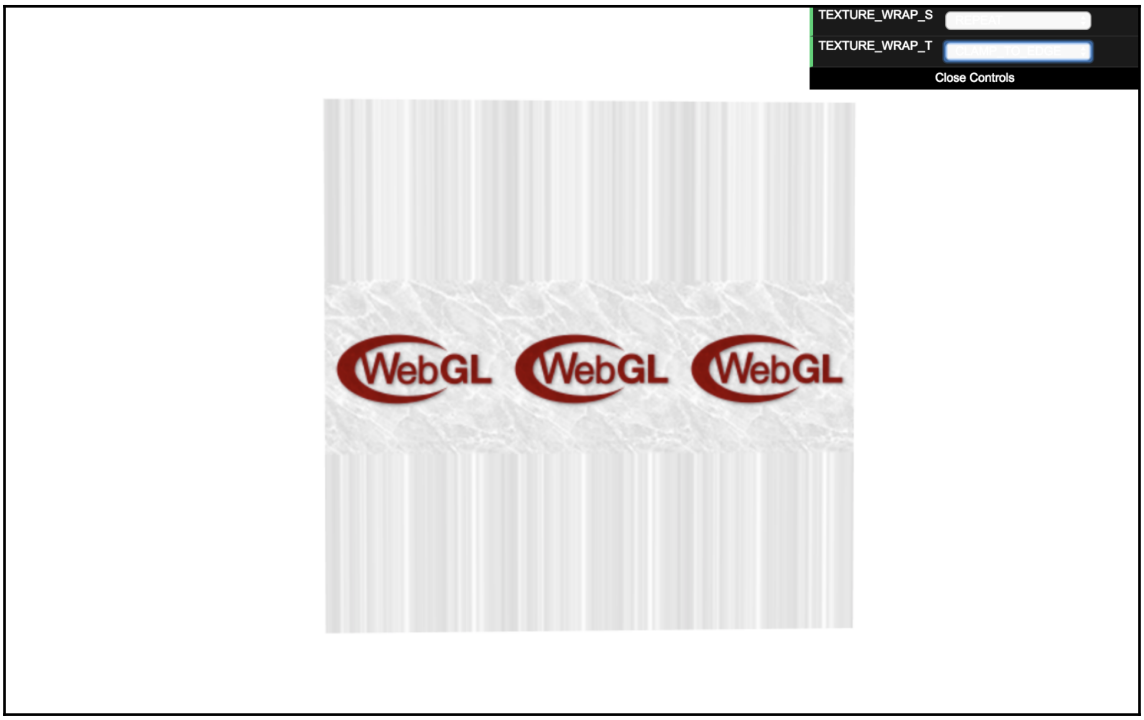

NEAREST
LINEAR_MIPMAP_LINEAR

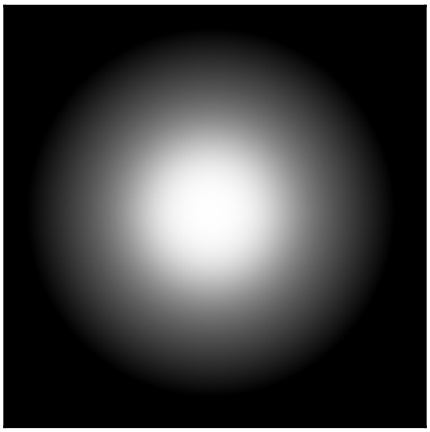

TEXTURE_WRAP_S
TEXTURE_WRAP_T
Close Controls

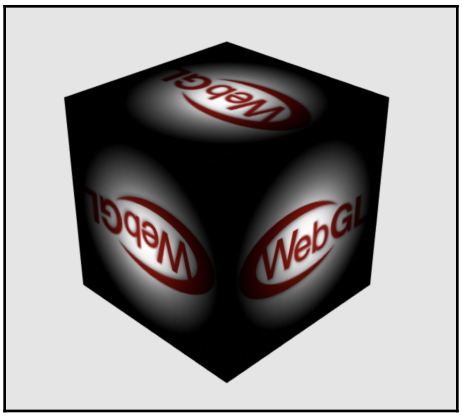

TEXTURE_WRAP_S
TEXTURE_WRAP_T
Close Controls
WebGL

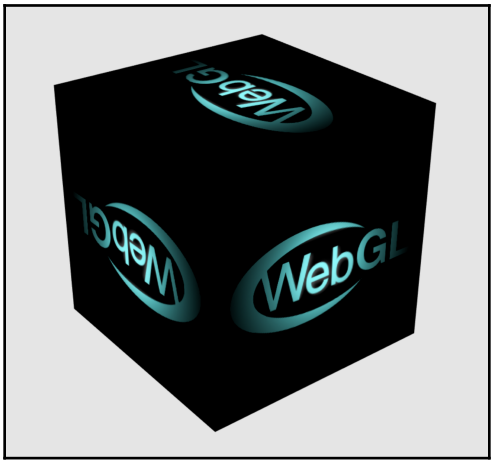

TEXTURE_WRAP_S
TEXTURE_WRAP_T
Close Controls
WebGL
WebGL
WebGL
WebGL
WebGL
WebGL
WebGL
WebGL
WebGL

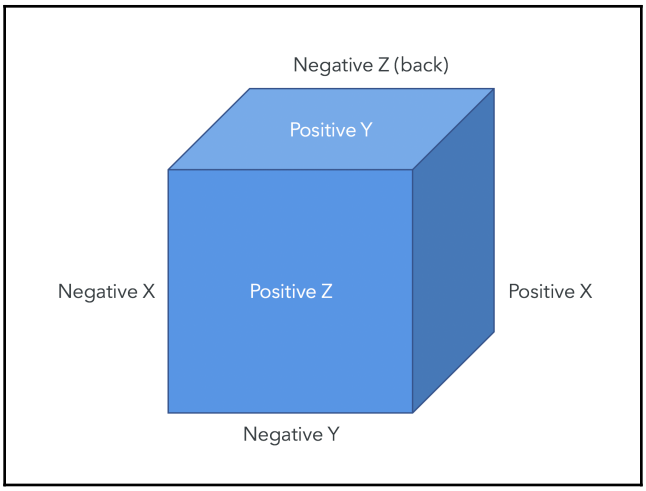

TEXTURE_WRAP_S
TEXTURE_WRAP_T
Close Controls
WebGL

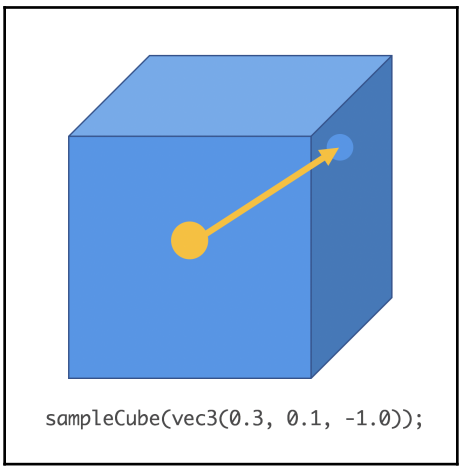

TEXTURE_WRAP_S
TEXTURE_WRAP_T
Close Controls
WebGL
WebGL
WebGL




WebGL
WebGL
WebGL


WebGL
WebGL
WebGL

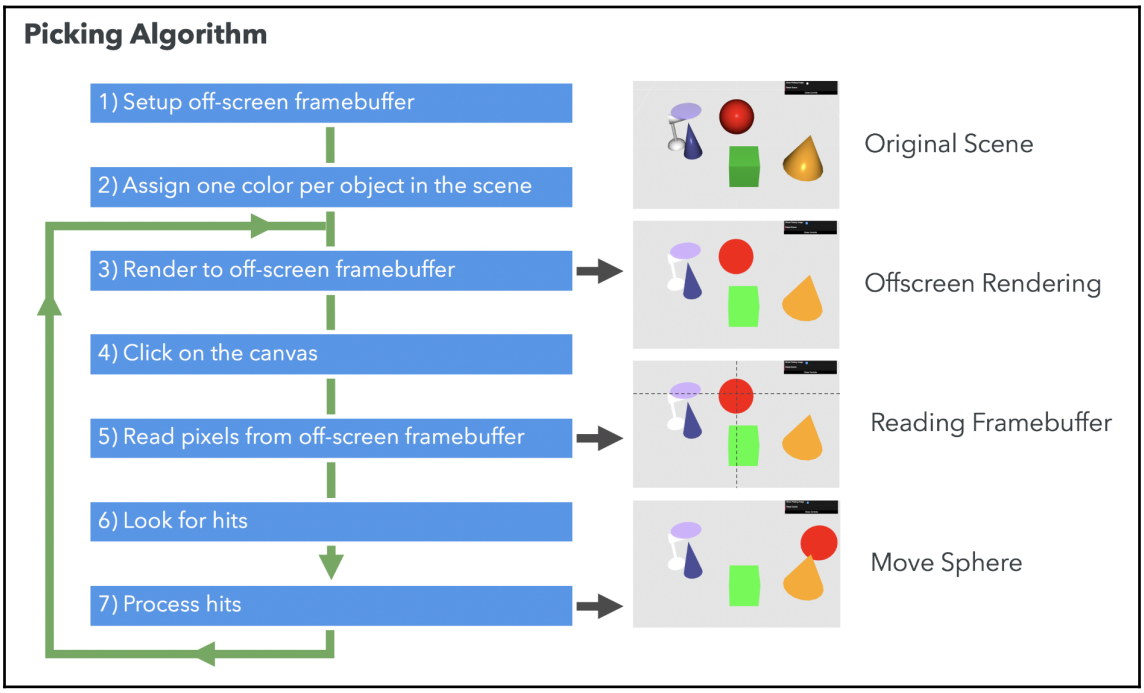




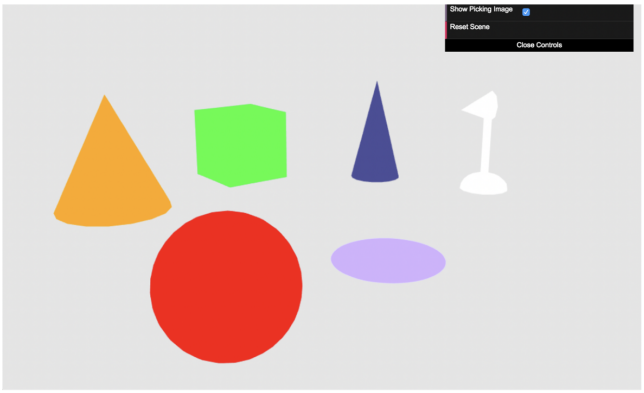
```
sampleCube(vec3(0.3, 0.1, -1.0));
```

NVIDIA

Chapter 8: Picking

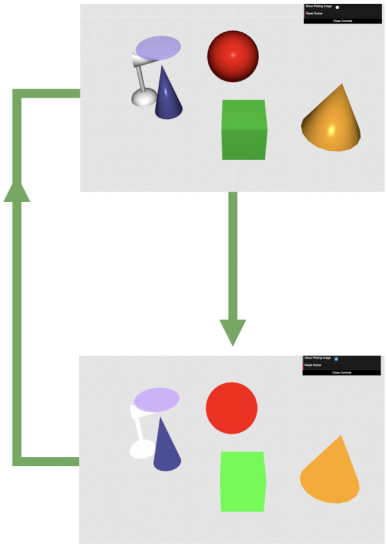


Off-Screen Framebuffer: Identifying Objects with Colors



A unique color is used to identify each object.

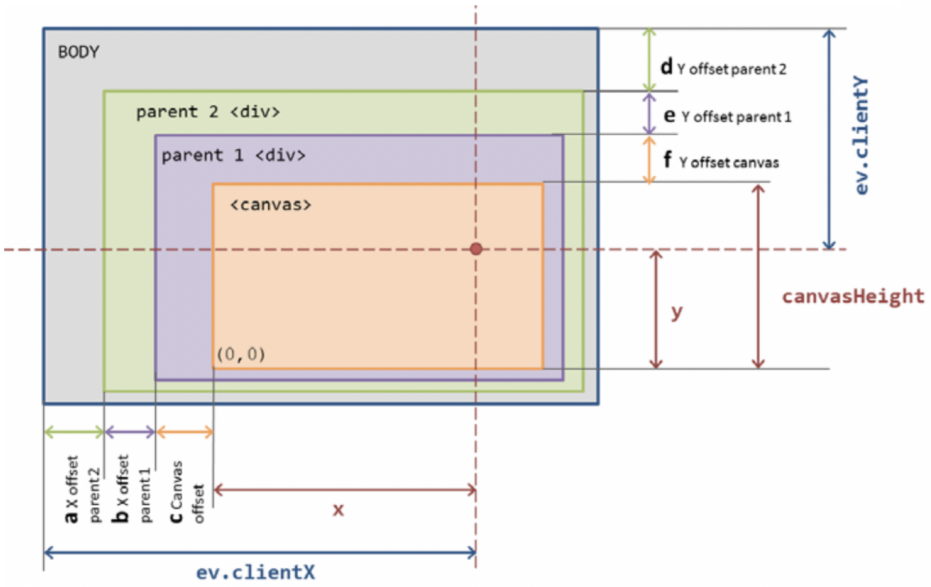
Rendering Cycle



Off-Screen Framebuffer
Constant colors are used.
Lights are disabled.
The material diffuse property can be used as long as it is unique for every object in the scene. Otherwise, a unique color/label needs to be assigned to each object.

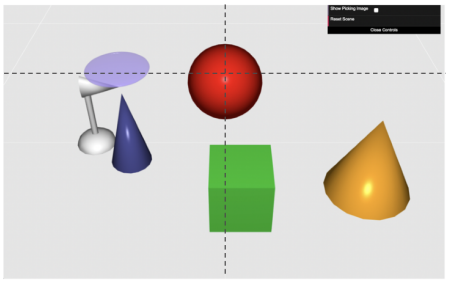
On-Screen Framebuffer
Textures are enabled.
Light and material properties are enabled (i.e. specular, diffuse, specular, etc.).

Calculating Clicking Coordinates into Canvas Space

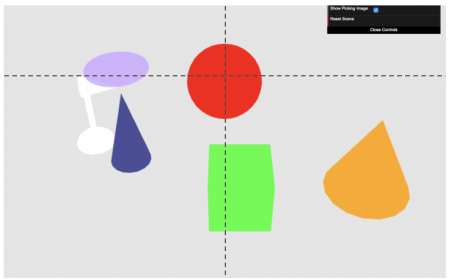


Clicking on the canvas selects the same coordinates in the off-screen framebuffer.

Canvas



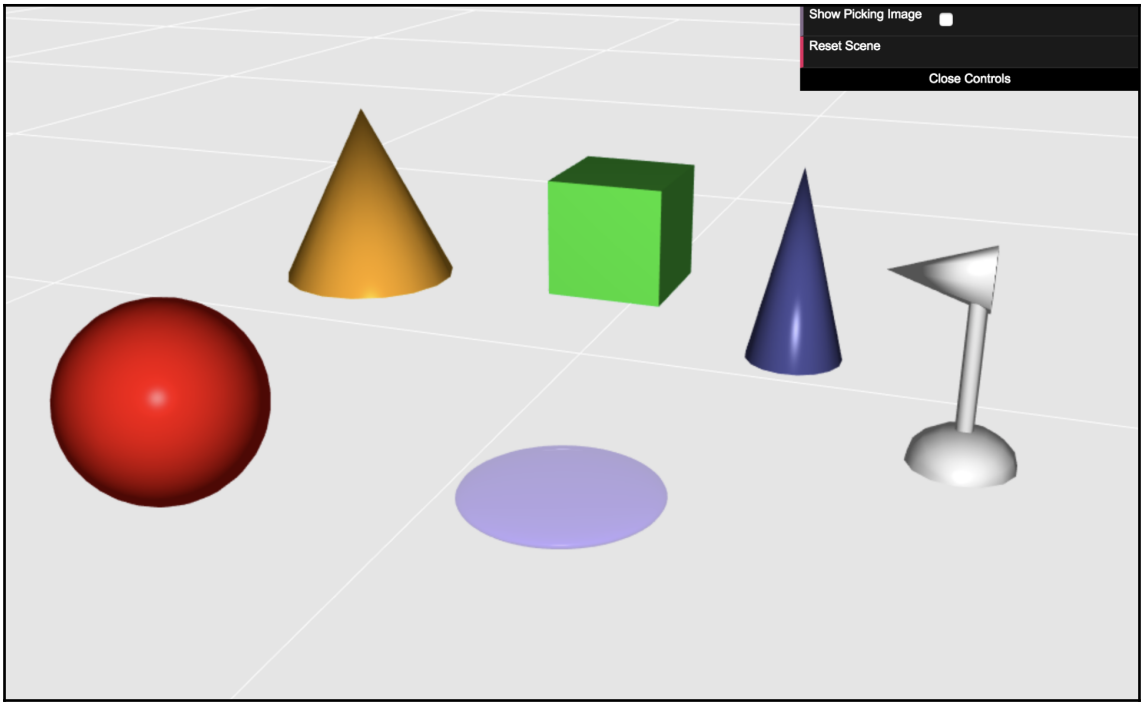
(0, 0)

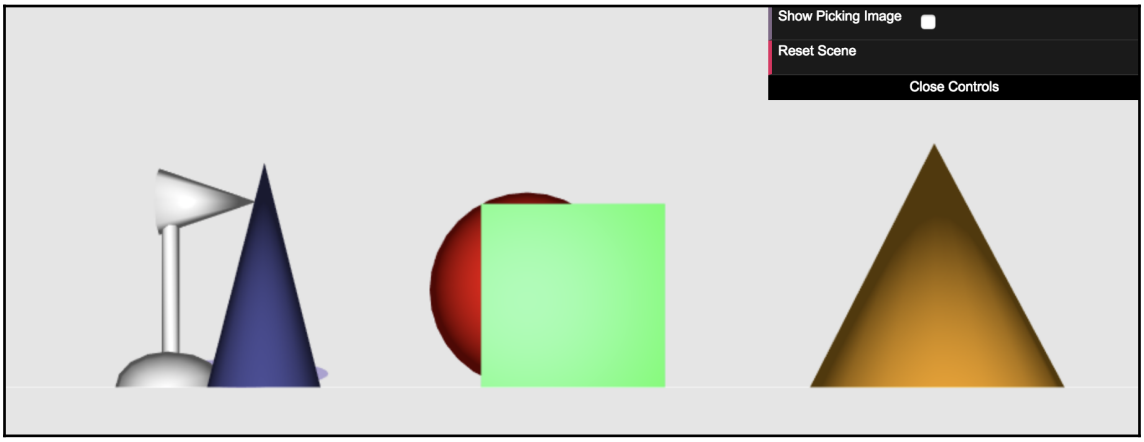


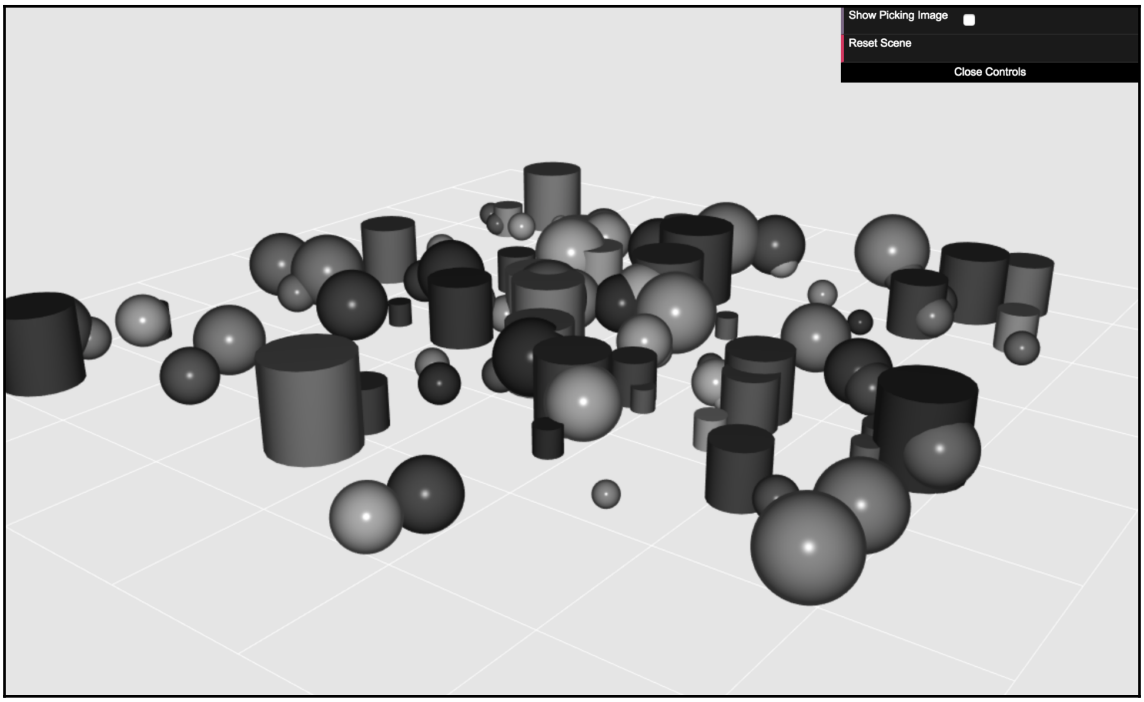
(0, 0)

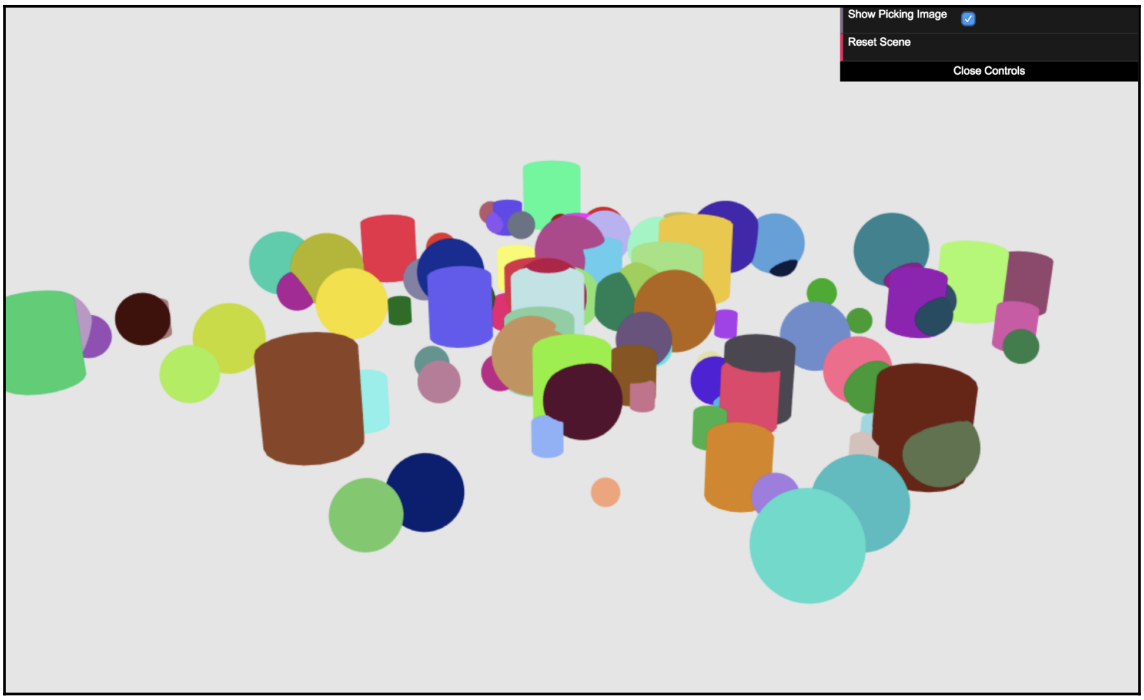
Off-Screen Framebuffer

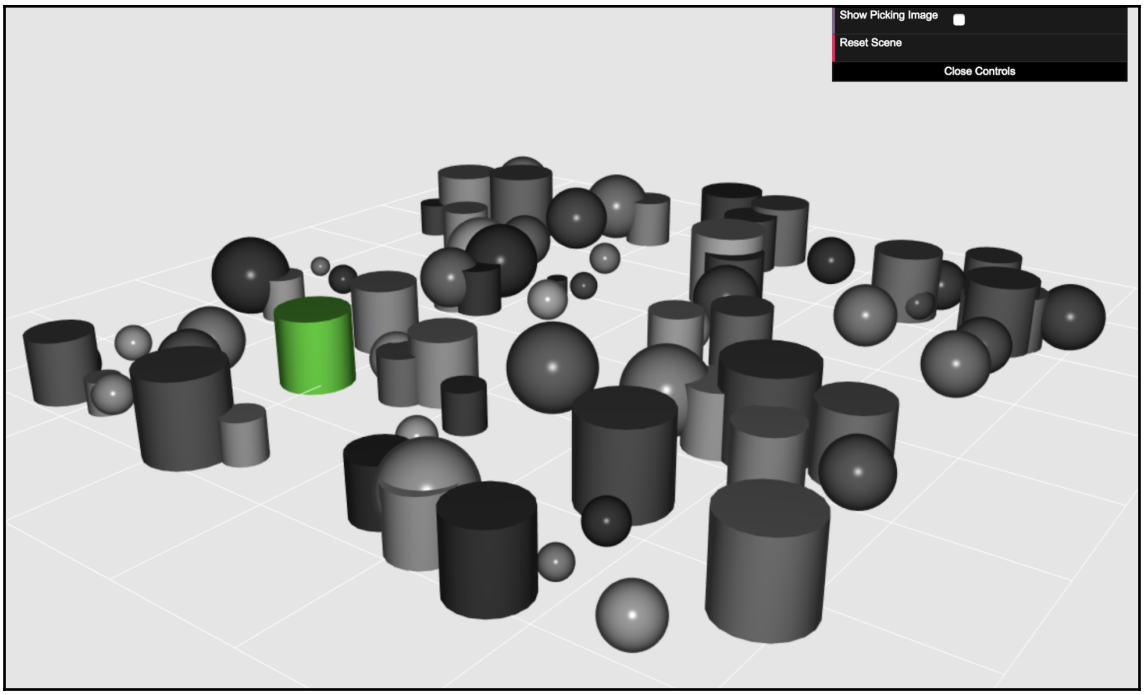
Canvas and framebuffer coordinates have their origin in the lower-left corner.

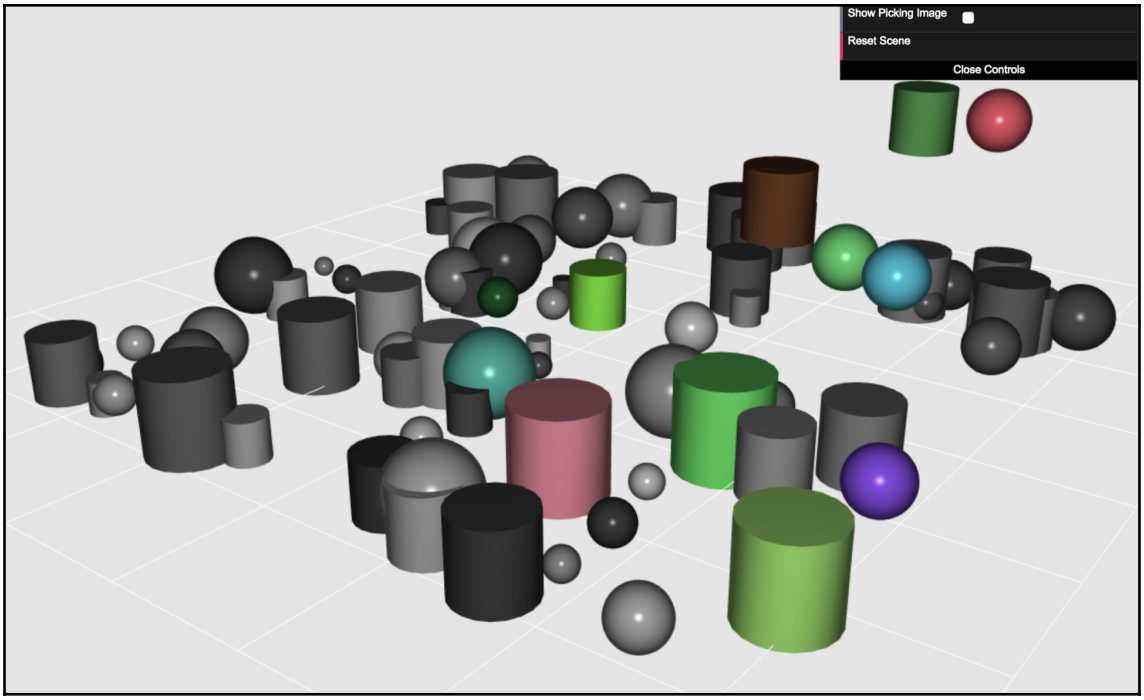

Show Picking Image
Reset Scene
Close Controls


Show Picking Image
Reset Scene
Close Controls

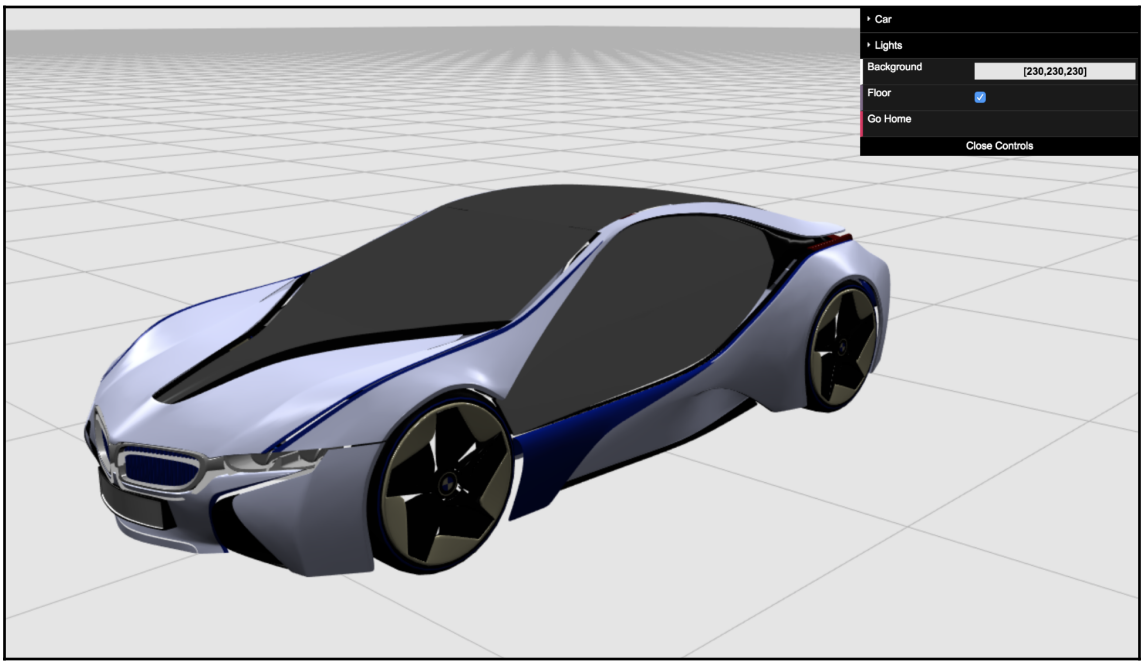

Show Picking Image
Reset Scene
Close Controls

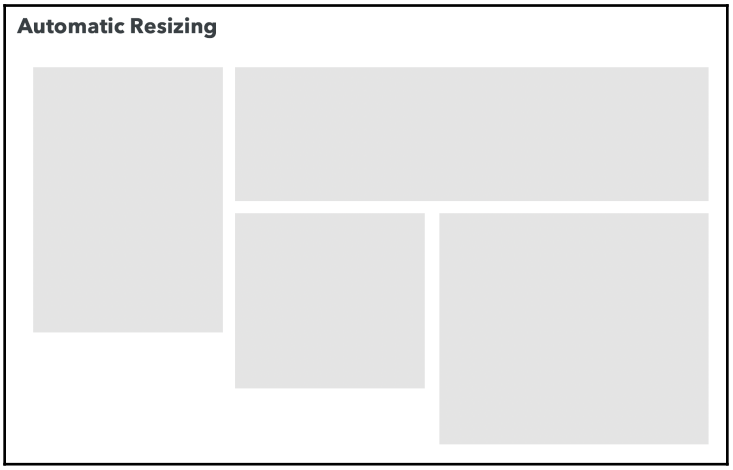

Show Picking Image
Reset Scene
Close Controls


Show Picking Image
Reset Scene
Close Controls

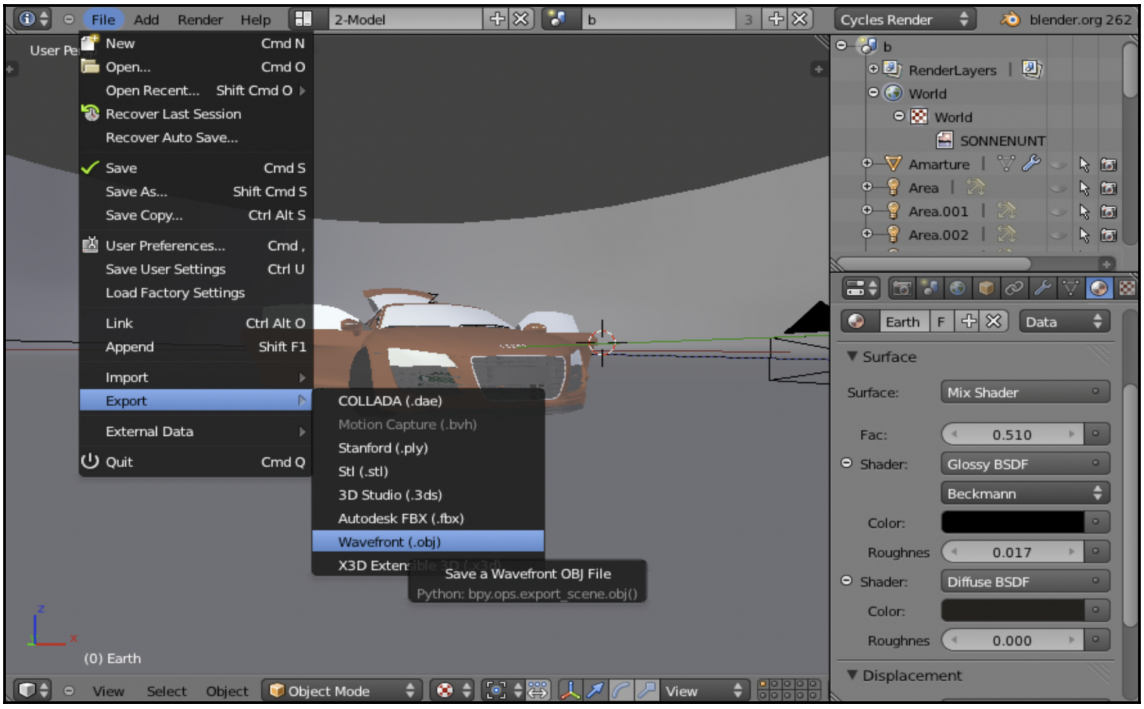

Show Picking Image
Reset Scene
Close Controls

Chapter 9: Putting It All Together

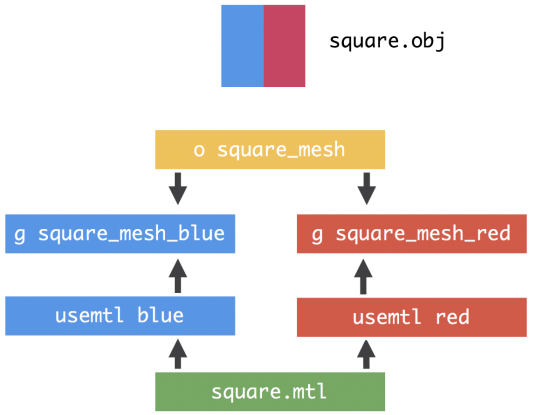




Setting up the Lights

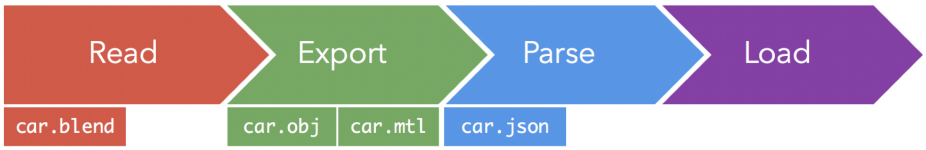


Understanding the Wavefront OBJ Format




```
# Blender v3.62 (sub 0) OBJ
File: 'square.blend'
# www.blender.org
mtllib square.mtl
o square_mesh
v  1.0  0.0 -2.0
v  1.0  0.0  0.0
v -1.0  0.0  0.0
v -1.0  0.0 -2.0
v  0.0  0.0  0.0
v  0.0  0.0 -2.0
vn 0.0  1.0  0.0
g square_mesh_blue
usemtl blue
s off
f 6//1 4//1 3//1
f 6//1 3//1 5//1
g square_mesh_red
usemtl red
f 1//1 6//1 5//1
f 1//1 5//1 2//1
```

Using Blender Models in a WebGL Scene



Read the original model in blender

Use the export option to generate the OBJ and MTL files

Use the python parser to create the JSON file

Load the JSON file

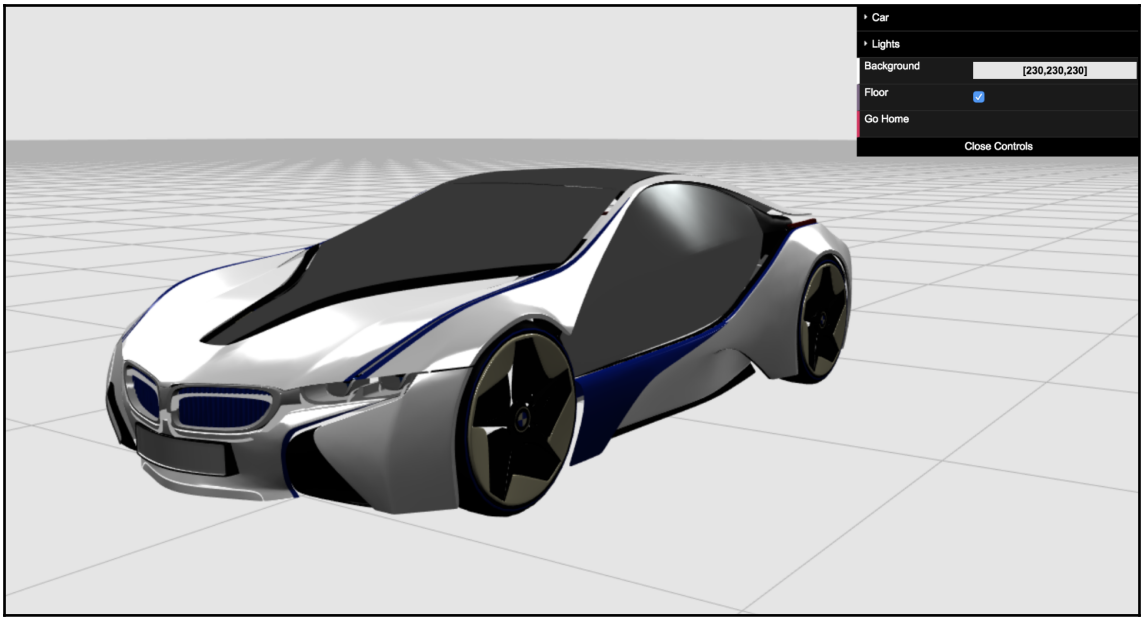
This process retrieves:

- Vertices
- Indices
- Texture coordinates
- Material properties

Rendering Call Stack

Whenever the timer goes off…

- Updates viewport Size
- Clear `canvas`
- Update `camera` perspective
- For each object in the `scene`
  - Calculate local transformations
  - Pass uniforms
  - Pass attributes
  - Execute draw call




Car
Lights
Background
[230,230,230]
Floor
Go Home
Close Controls
Model
Color
[255,0,0]
Shininess
50

Chapter 10: Advanced Techniques

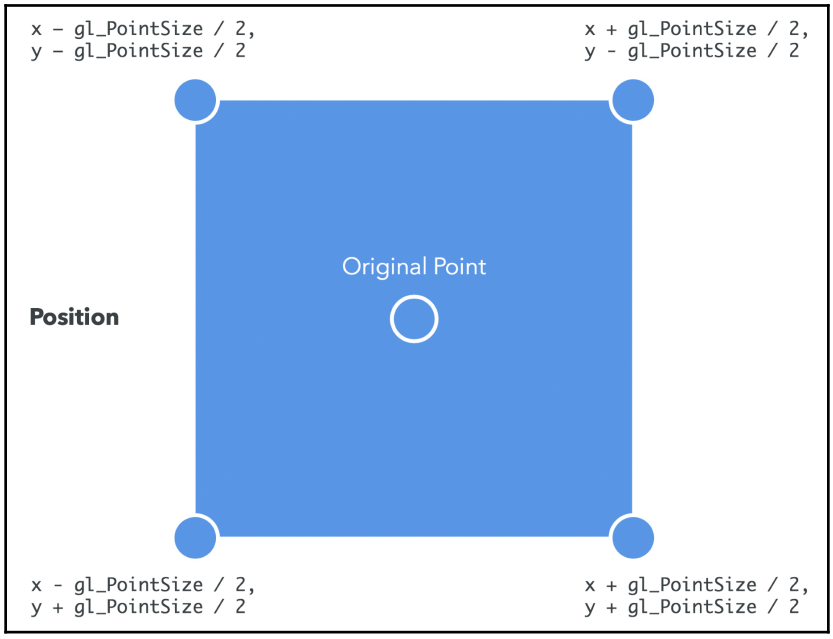

Filters
Close Controls
WebGL
WebGL
WebGL

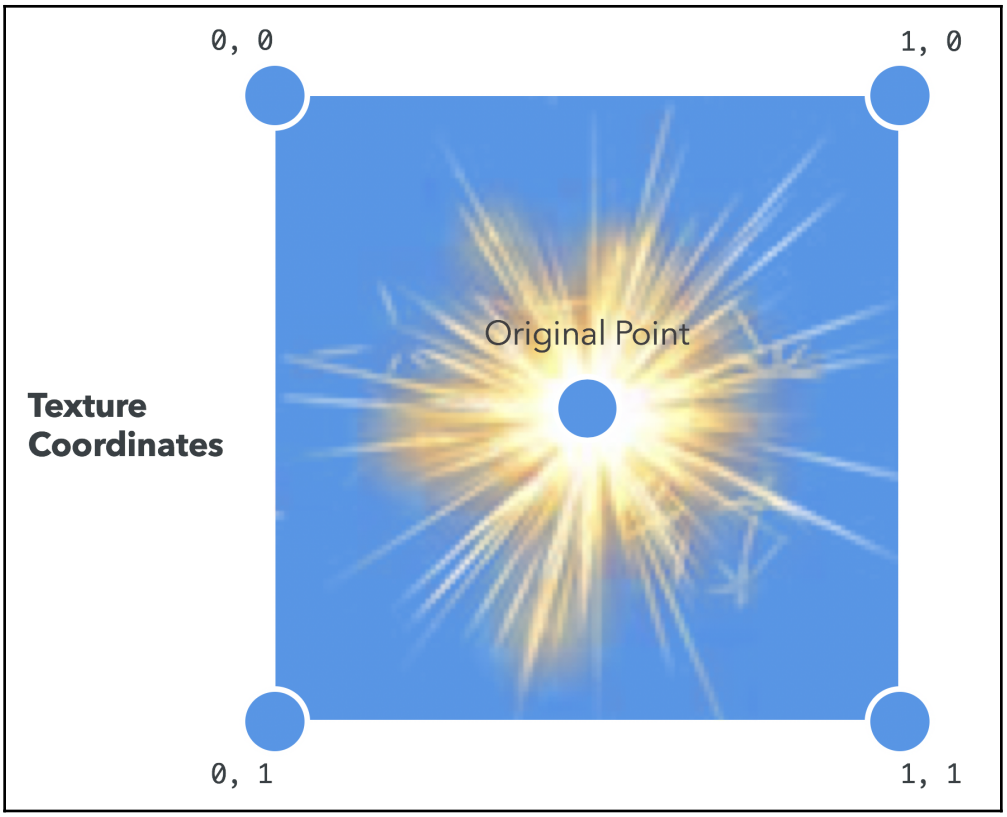

Filters
Close Controls
WebGL
WebGL
WebGL

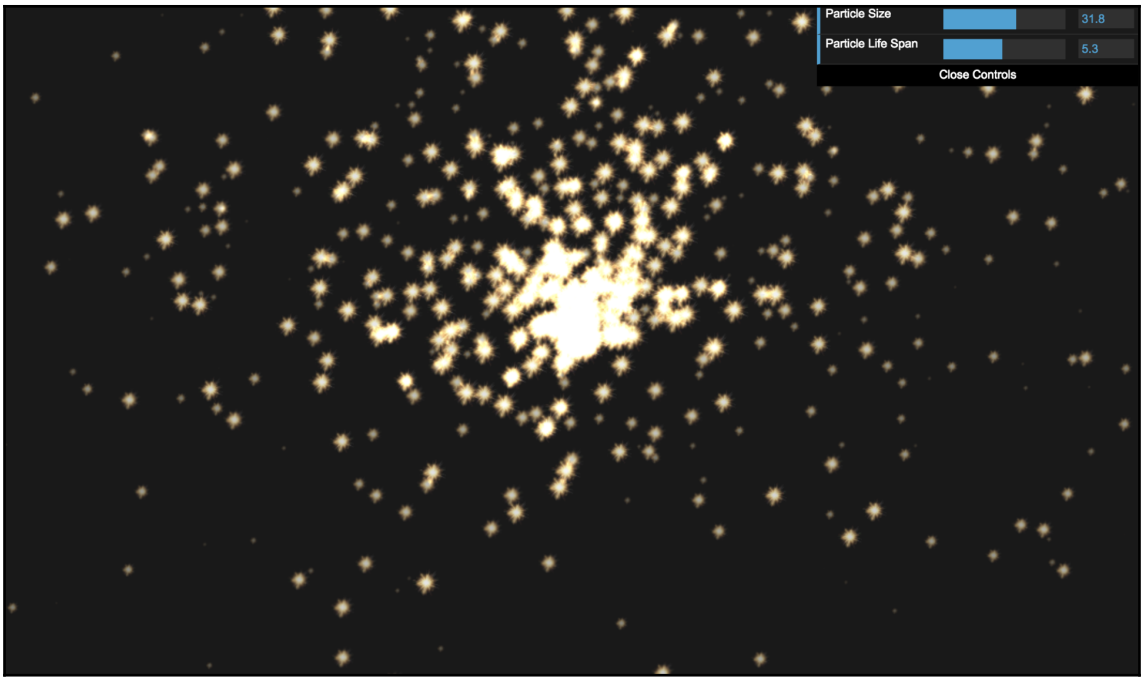

Filters
Close Controls
WebGL
WebGL
WebGL

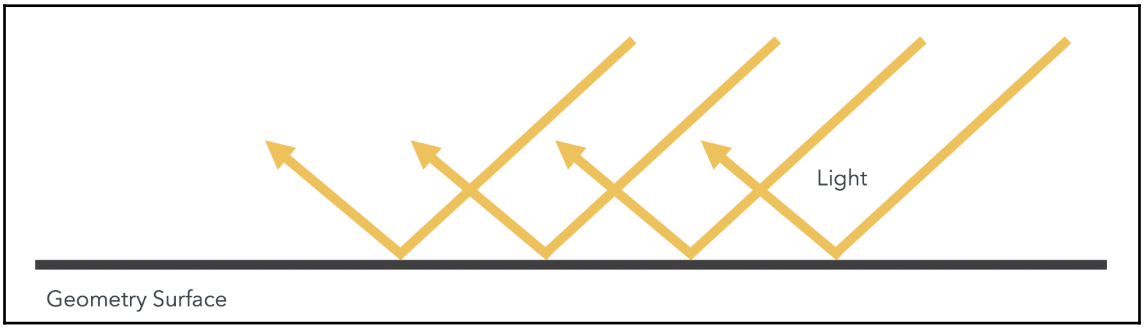

Filters
Close Controls
WebGL
WebGL
WebGL

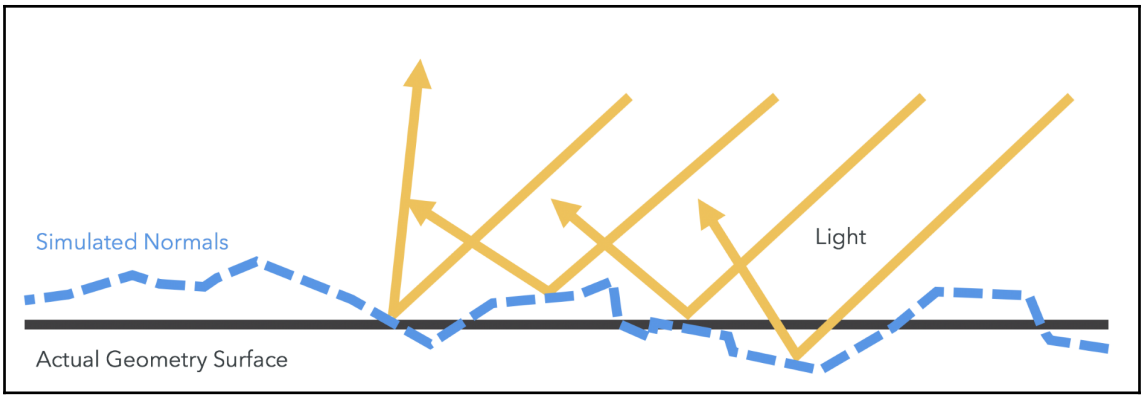

Filters
Close Controls

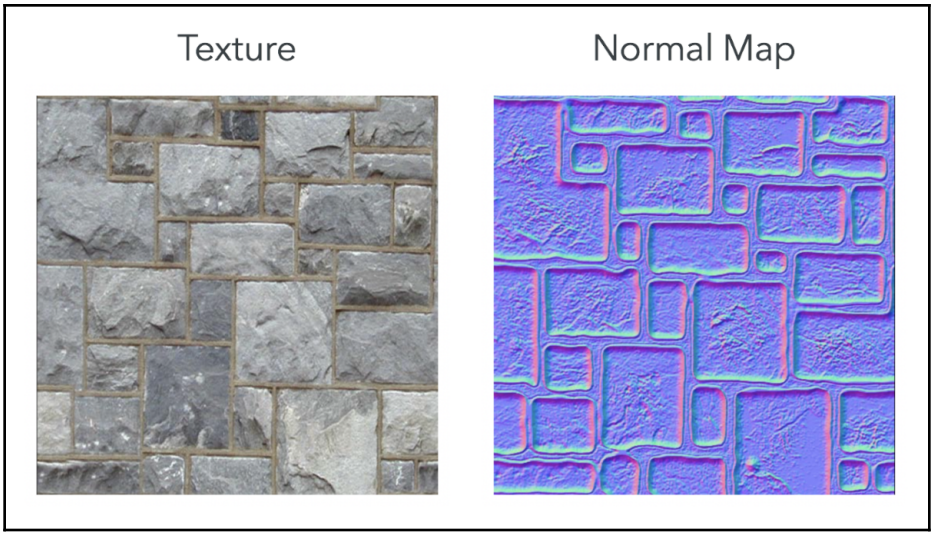

x – gl_PointSize / 2,
y – gl_PointSize / 2
x + gl_PointSize / 2,
y - gl_PointSize / 2
Original Point
Position
x - gl_PointSize / 2,
y + gl_PointSize / 2
x + gl_PointSize / 2,
y + gl_PointSize / 2

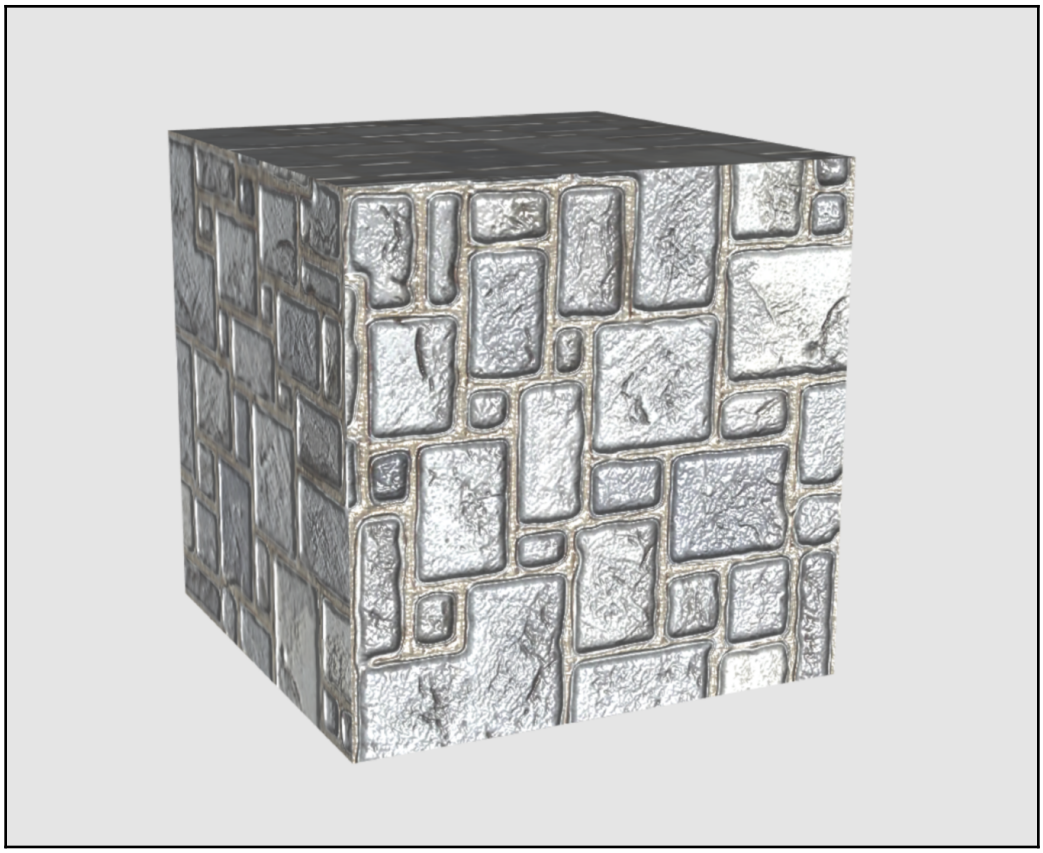

0, 0
1, 0
Original Point
Texture
Coordinates
0, 1
1, 1

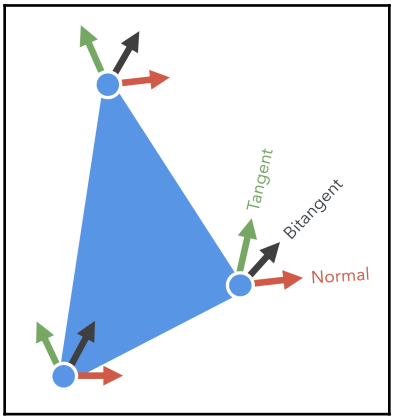

Particle Size
31.8
Particle Life Span
5.3
Close Controls

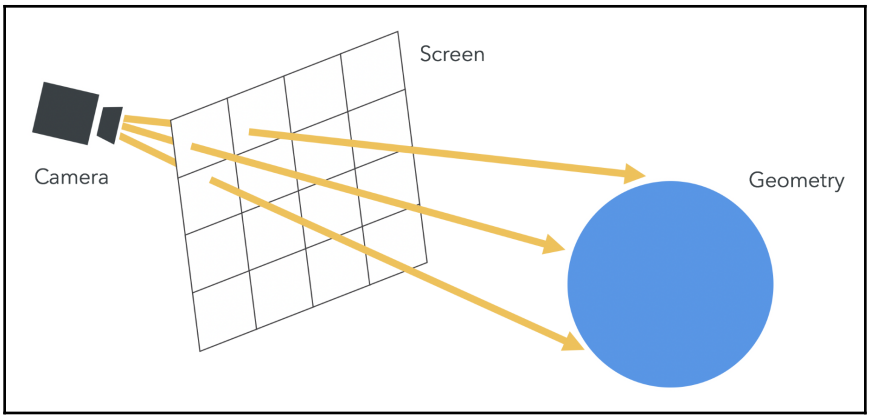

Light
Geometry Surface

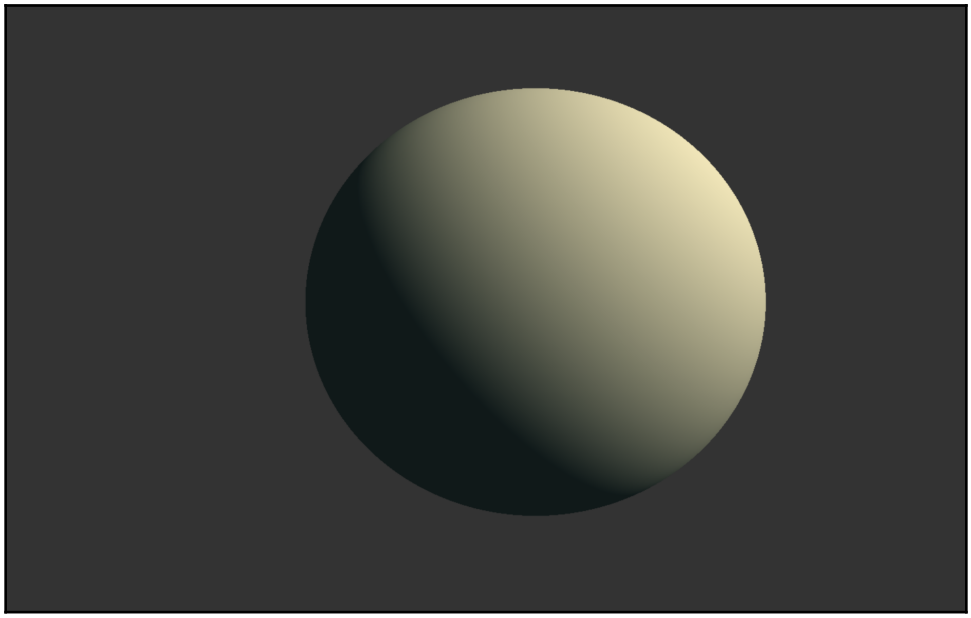

Simulated Normals
Light
Actual Geometry Surface


Texture
Normal Map

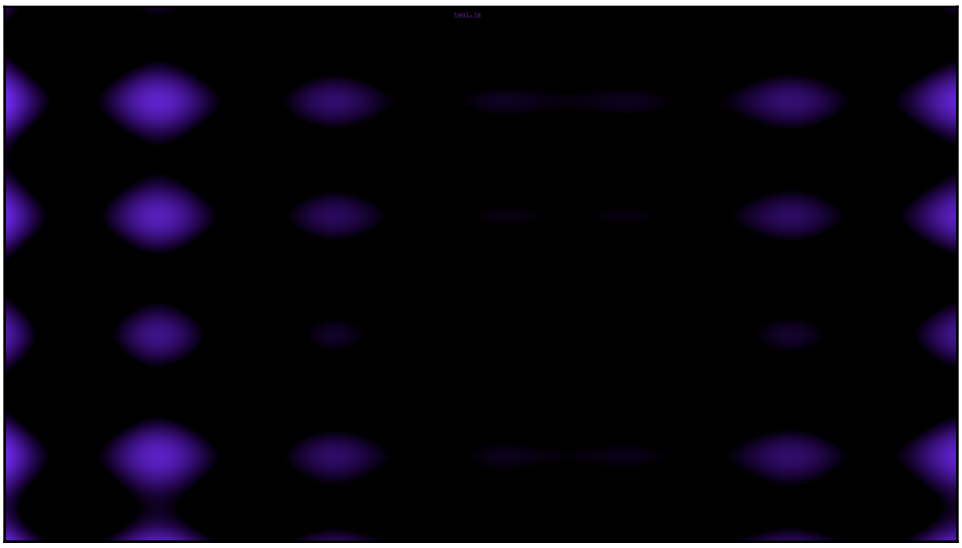


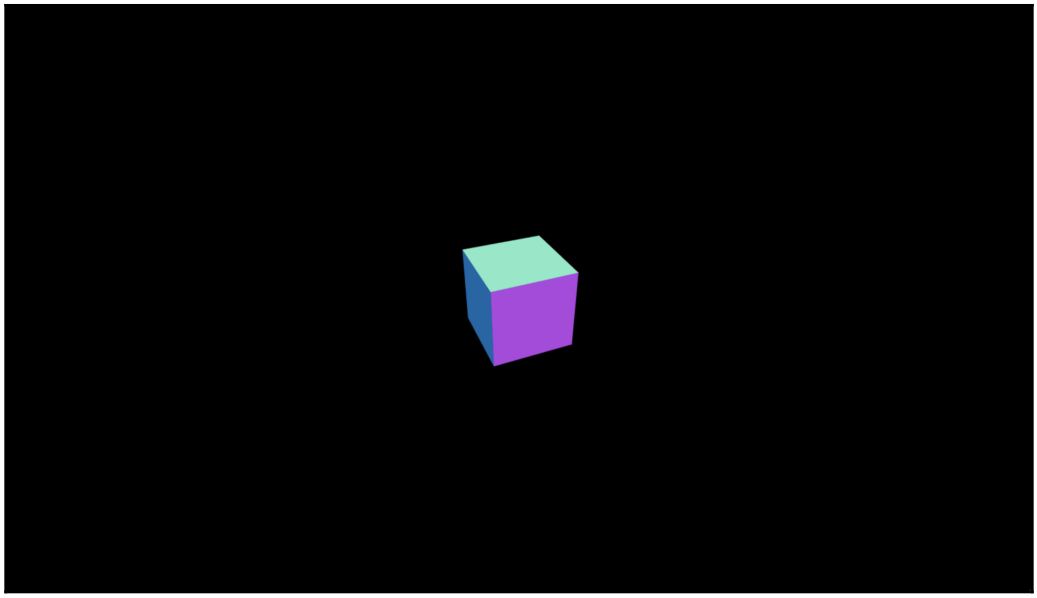

Tangent
Bitangent
Normal

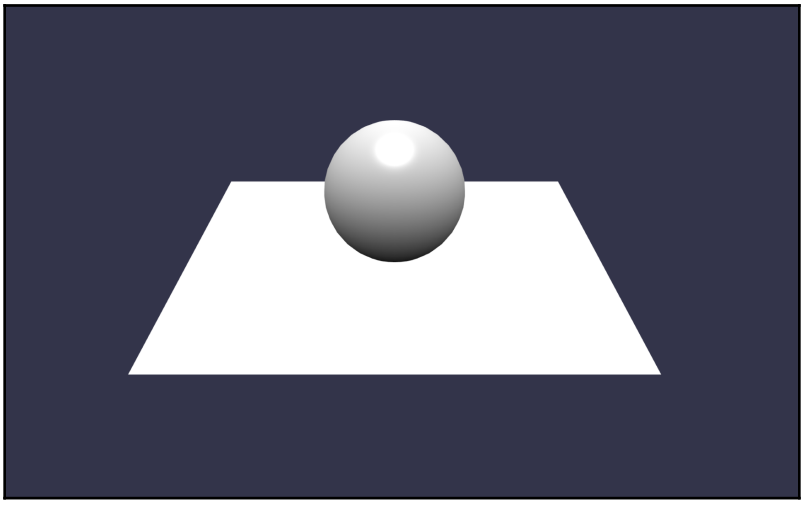

Screen
Camera
Geometry

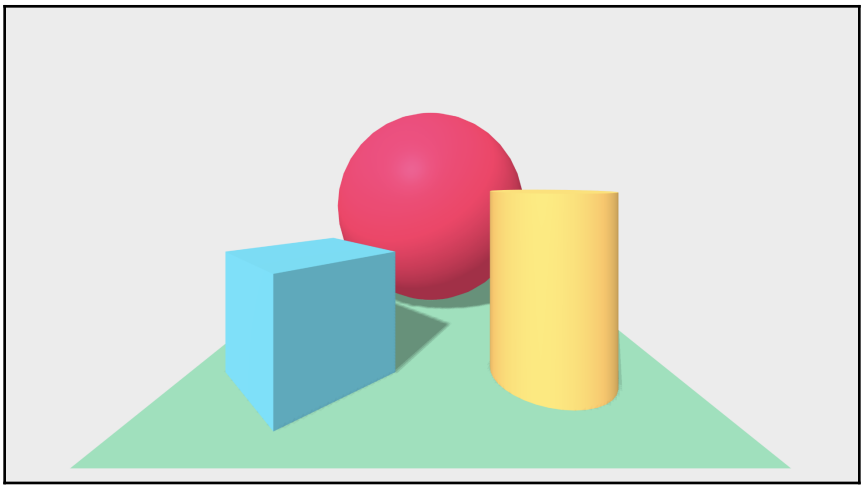


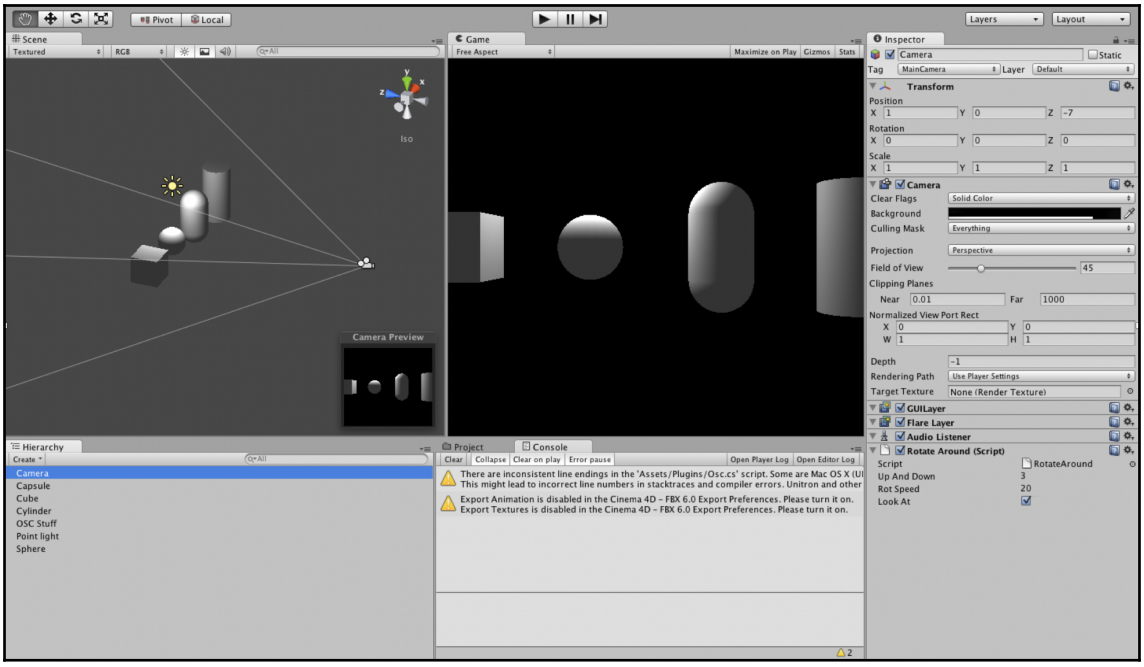
Chapter 12: Journey Ahead



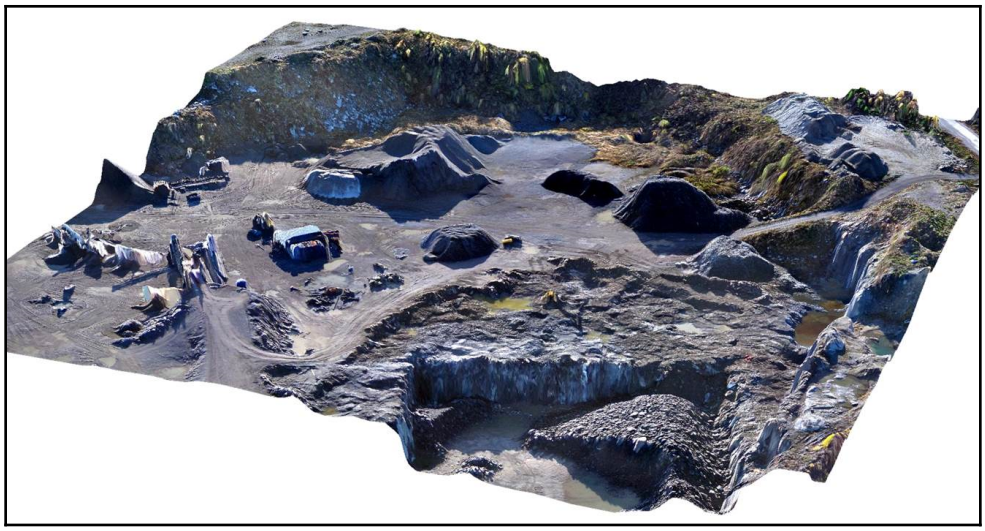


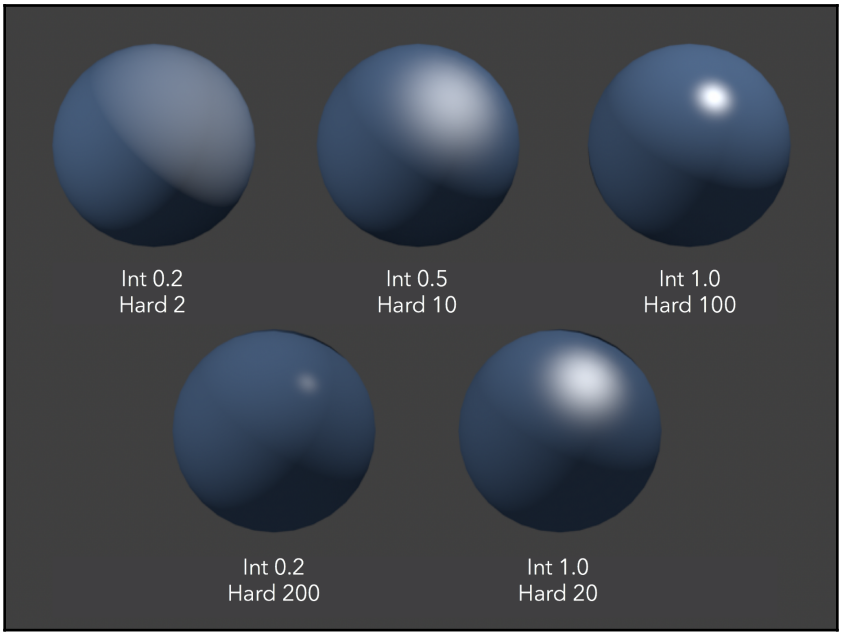


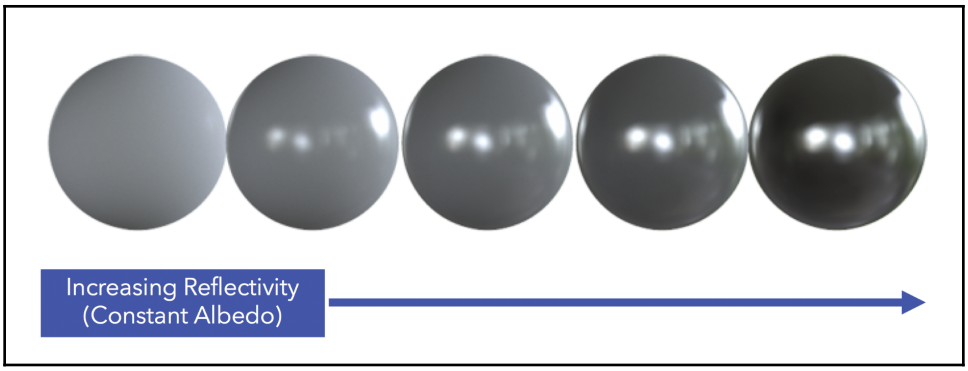



Pivot
Local
Layers
Layout
Scene
Textured
RGB
Iso
Camera Preview
Game
Free Aspect
Maximize on Play
Gizmos
Stats
Inspector
Camera
Static
Tag
MainCamera
Layer
Default
Transform
Position
Rotation
Scale
Camera
Clear Flags
Solid Color
Background
Culling Mask
Everything
Projection
Perspective
Field of View
45
Clipping Planes
Near
0.01
Far
1000
Normalized View Port Rect
Depth
-1
Rendering Path
Use Player Settings
Target Texture
None (Render Texture)
GUILayer
Flare Layer
Audio Listener
Rotate Around (Script)
Script
RotateAround
Up And Down
3
Rot Speed
20
Look At
Hierarchy
Create
Camera
Capsule
Cube
Cylinder
OSC Stuff
Point light
Sphere
Project
Console
Clear
Collapse
Clear on play
Error pause
Open Player Log
Open Editor Log
There are inconsistent line endings in the 'Assets/Plugins/Osc.cs' script. Some are Mac OS X (UI
This might lead to incorrect line numbers in stacktraces and compiler errors. Unitron and other
Export Animation is disabled in the Cinema 4D – FBX 6.0 Export Preferences. Please turn it on.
Export Textures is disabled in the Cinema 4D – FBX 6.0 Export Preferences. Please turn it on.
2

Article Title	Article Title	Article Title
		
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi eleifend sollicitudin varius. Mauris urna nibh, bibendum non nunc id, mollis lacinia leo. Nam egestas auctor feugiat.	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi eleifend sollicitudin varius. Mauris urna nibh, bibendum non nunc id, mollis lacinia leo. Nam egestas auctor feugiat.	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi eleifend sollicitudin varius. Mauris urna nibh, bibendum non nunc id, mollis lacinia leo. Nam egestas auctor feugiat.
Baseline	Change	Diff




Int 0.2
Hard 2
Int 0.5
Hard 10
Int 1.0
Hard 100
Int 0.2
Hard 200
Int 1.0
Hard 20


Increasing Reflectivity
(Constant Albedo)