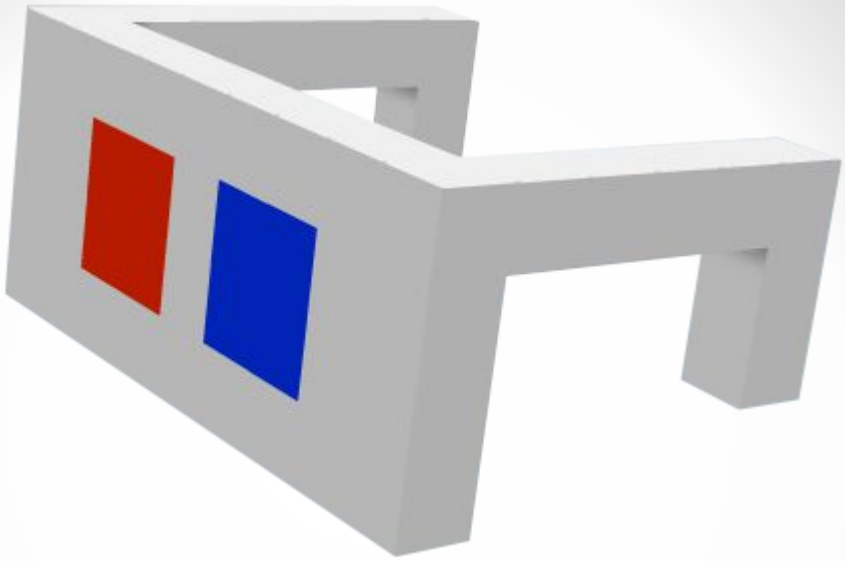


A BRIEF INTRODUCTION TO 3D

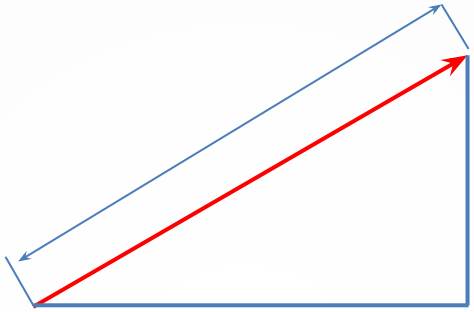
Selected topics with focus on Flash, Unity and WebGL


shader
WebGL
Texture
transformation
vertex
shadows
simple
Away3D
Frameworks
matrix
Profiling
color
Lighting
Fragment
Molehill
Vertex
Linear
Material
Space
Matrix
PixelBender3D
Unity
Unity3D
GPU
Plugin
shader
shading
mapping
Graphics
Cubemap
Flash
pixel
bump
stuff
language
common
availability
World
visible
Ambient
settings
shaders
textures
format
effects
SHADERS
memory
Point
Unitys
Vector3
level
Animations
Stack
properties
pipeline
API
Scale
Quality
Cross
new
ARB
Blending
Solutions
performance
Uses
Pix
player
Probably
Specular
objects
Node
transform
BSP
Skew
Geometry
view
map
light
ingredients
time
Enlighten
computed
Intel
Pixel
coordinate
Learning
Multiple
features
Outputs
Uniforms
Deferred
height
rgb
Open
OpenGL
examples
comments
texture
tool
normals
base
moving
Paralax-mapping
parts
code
collection
Model
Assembly
Product
Normal
magnification/minification
View/Camera
Rotation
Local/World
Shininess
lights
FSH
linear
animation
DirectX
Slerp
Shaders
image
Classes/Utils
yet
Quaternions
Buy
Two
Tree
far
vs
tile
get
Stages
intensive
2+0
classes
HTML5
interpolated
look
Apply
Built
diffuse
bullet
haXe
materials
final
Cg
think

MATH & ALGEBRA

VECTOR

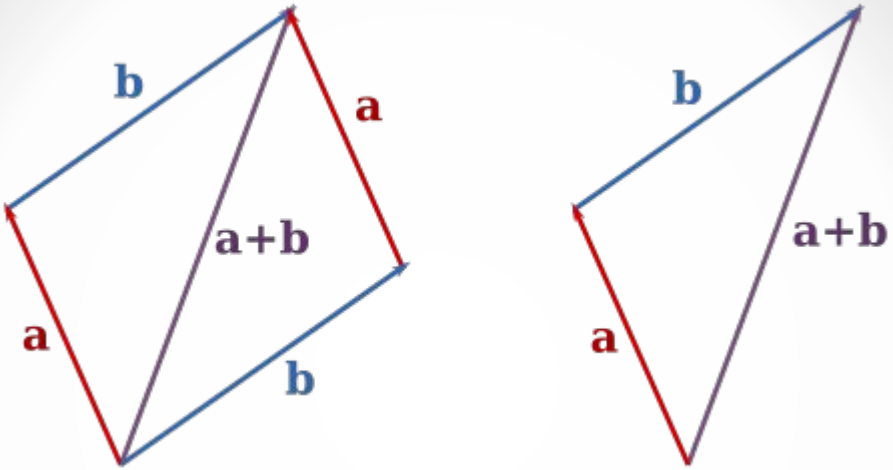
Length



Pythagorean Formula

$|V| = \text{sqrt}(x^2 + y^2)$

Addition

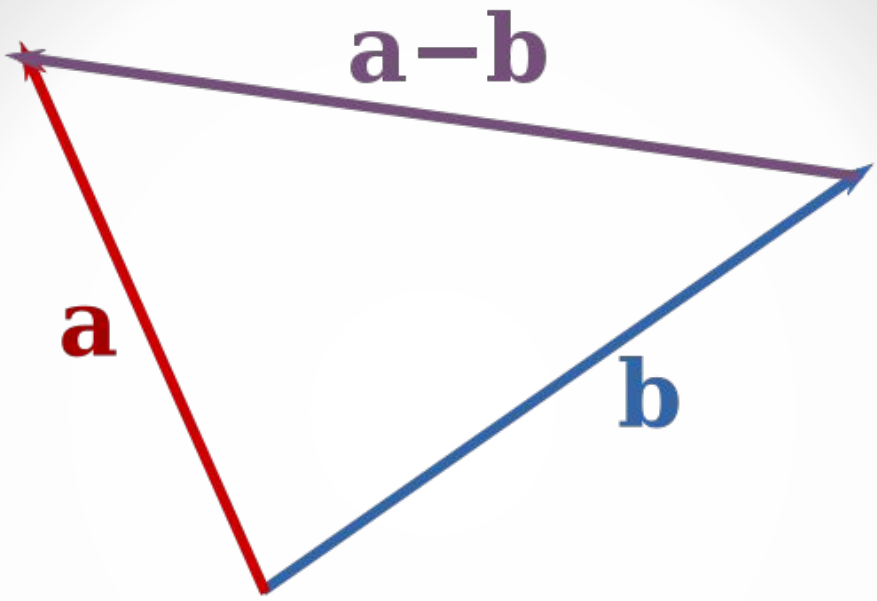


A = (1, 2)

B = (4, 0)

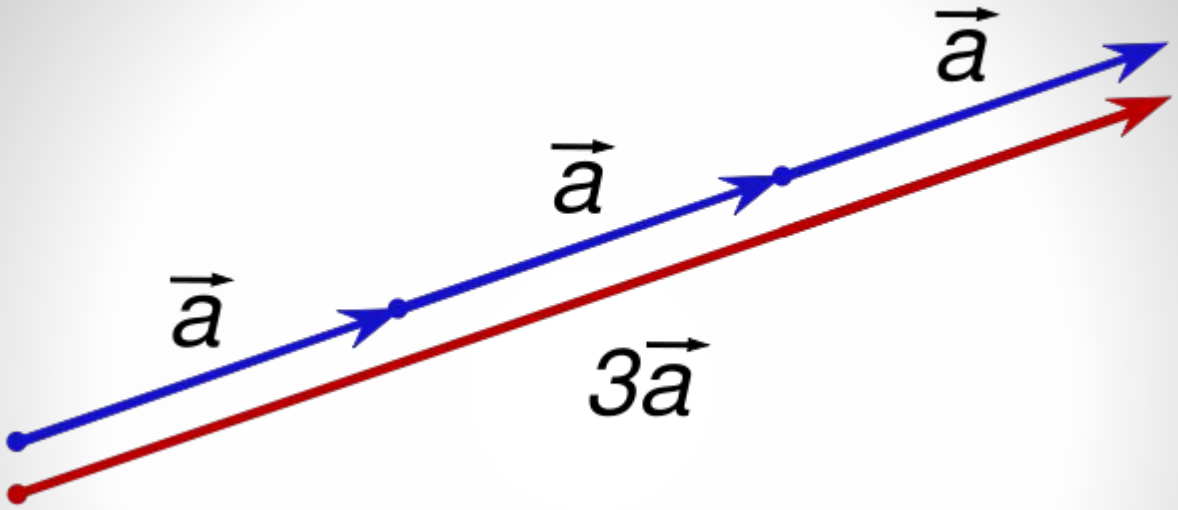
A + B = (1+4, 2+0) = (5, 2)

Subtraction



A = (1, 2)
B = (4, 0)
A - B = A + (-B)
A - B = (1-4, 2-0) = (-3, 2)

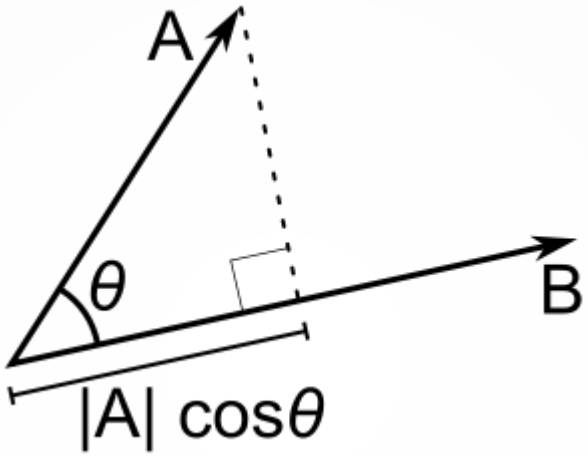
Scalar Multiplication



A*3 = (3*1, 3*2) = (3, 6)

(*unit* vector = divide the vector by it's length)

Dot Product



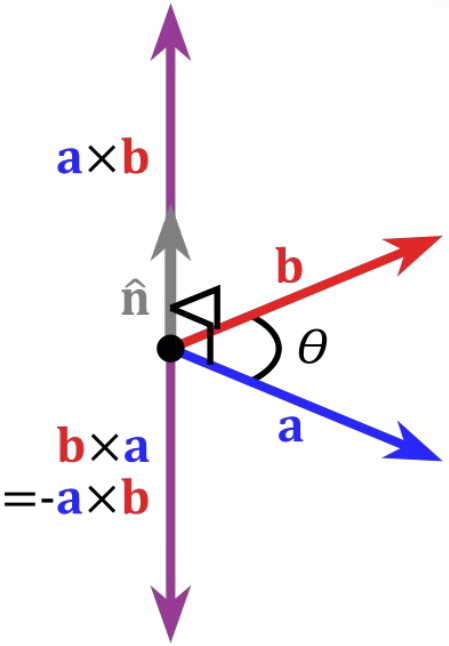
$A = (A_x, A_y, A_z)$

$B = (B_x, B_y, B_z)$

$A \cdot B = A_x B_x + A_y B_y + A_z B_z$

$A \cdot B = |A||B| cos\theta$

Cross Product



$$AxB = (A_yB_z - A_zB_y, A_zB_x - A_xB_z, A_xB_y - A_yB_x)$$

Real world examples

- In which direction should the missile be fired to hit the target?
- Is the enemy visible in the field of view?
- How far is the bullet from the window?

Solutions

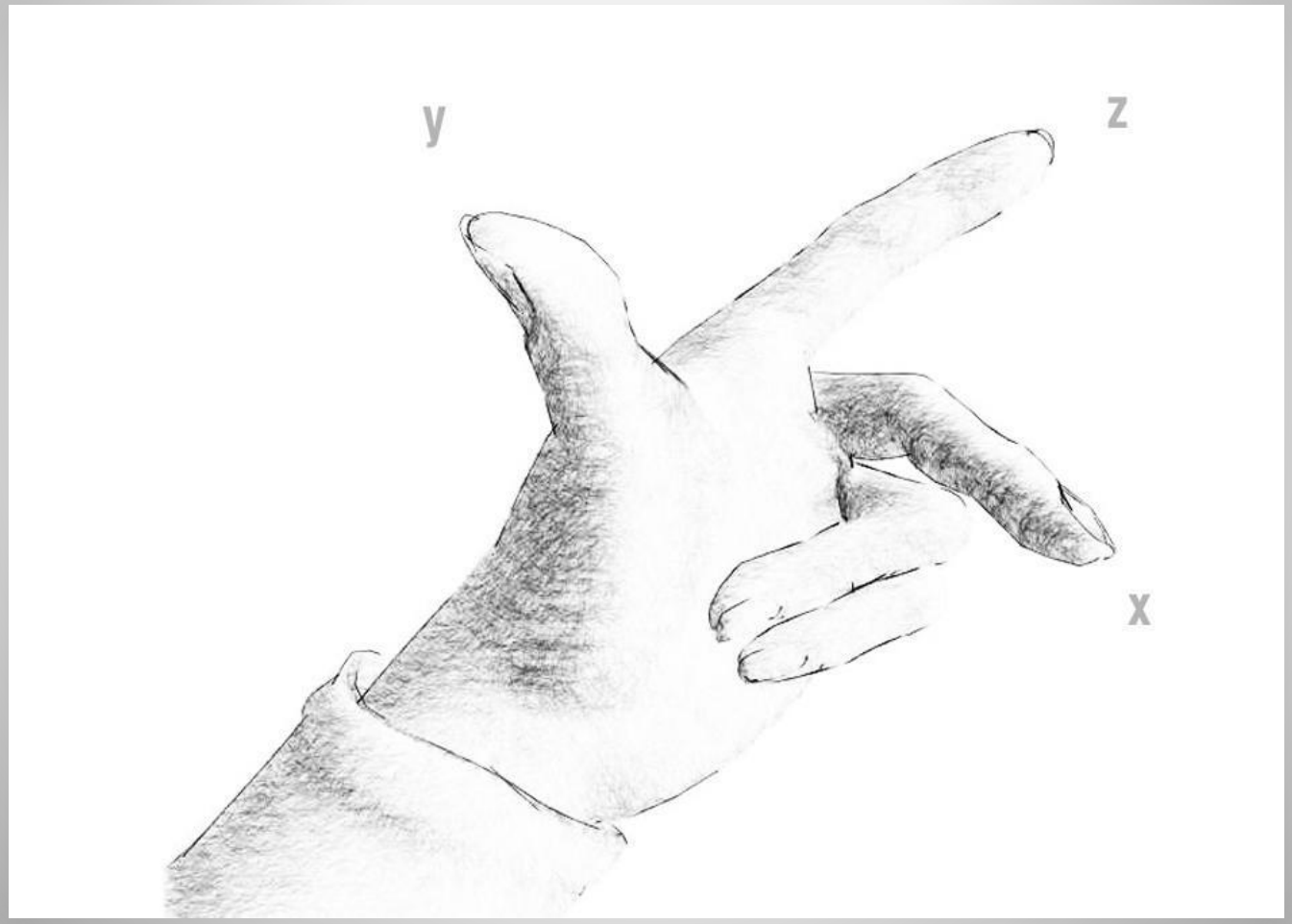
- Solutions have been done by many before.
- Know the basics to find them quicker.
- Use utils ar  classes like:
  - Vector3D 
  - Vector3DUtils 
  - Plane3D  Ray (4.0)
  - Vector3

SPACES

Spaces

- *Euclidean* space using *Cartesian* coordinates. (X, Y and Z)
- Local/Model Space
- World Space
- View/Camera Space (Point-of-view)
- Screen space (2D)

Left- and right-handed systems




ENTER THE MATRIX

Matrices

- Matrix = Transformation placeholder
- So again:
  - Local/Model matrix
  - World matrix
  - View/Camera matrix
- *WVP* = world * view * projection

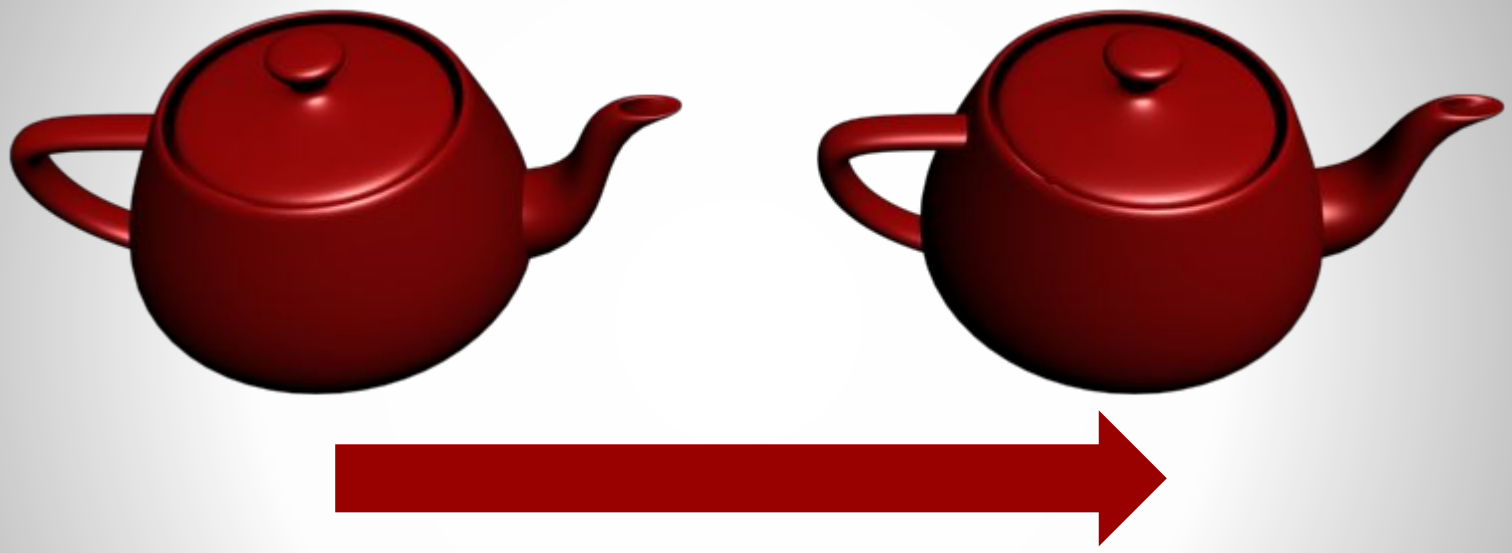
Classes/Utils

- Matrix3D 
- Matrix3DUtils 

- Matrix4x4 

TRANSFORMATIONS

Linear transformation



Translation

Linear transformation

Scale

Linear transformation

Skew

Linear transformation

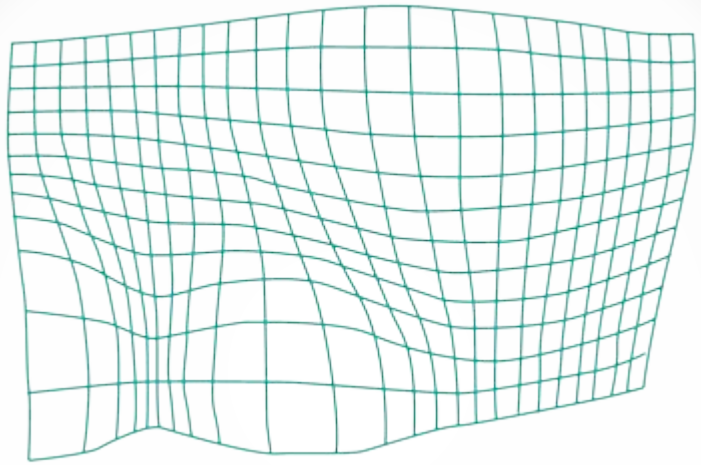
Rotation

- Eulers
- Quaternions
  - Avoids gimbal lock
  - Slerp (Smooth interpolated rotation)
- Matrix – memory intensive

Multi linear transformation

- Stack of matrices
- Apply all at once to an object
- The order is importent
- Identity matrix

Nonlinear transformations

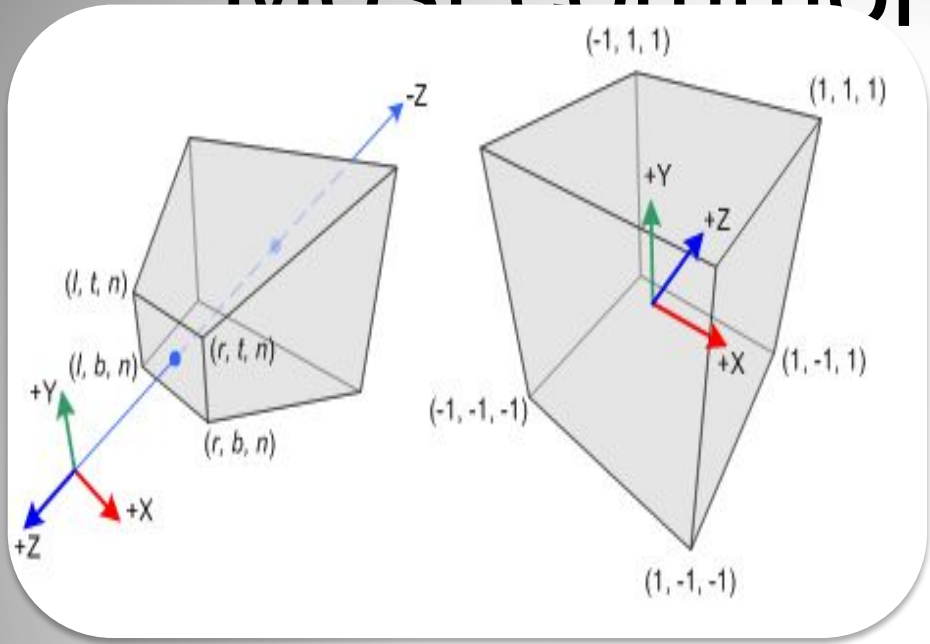


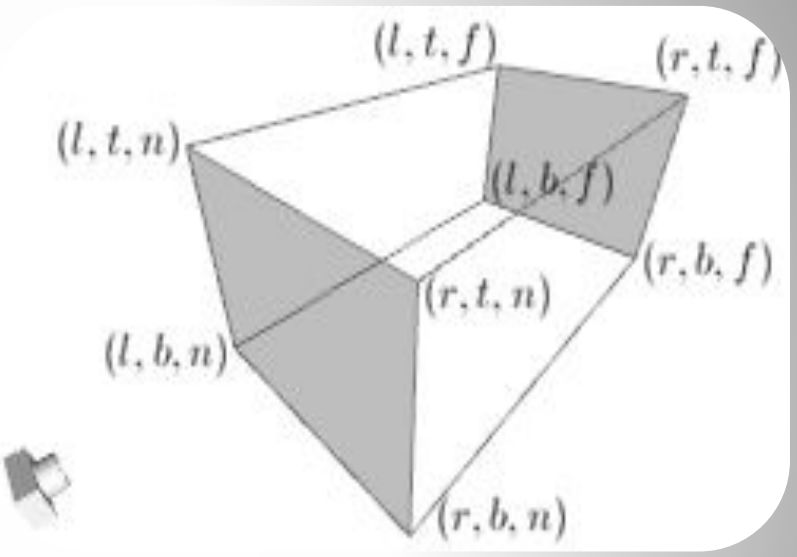
- Sin curve displacement
- Warp

PROJECTIONS

Converting a three-dimensional graphics object
or scene into two dimensions

Most common projections



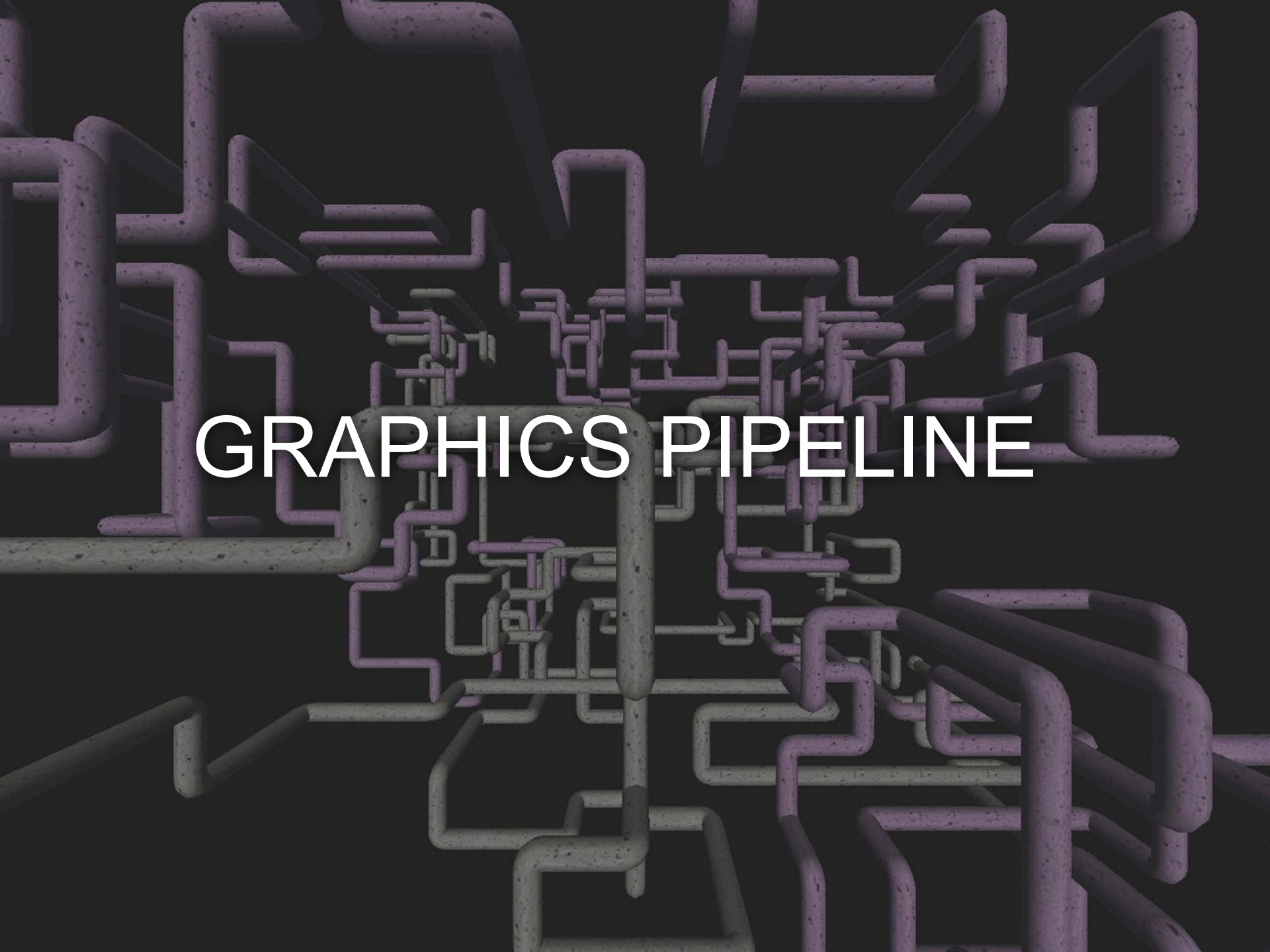


Perspective

- Objects is smaller further away
- Projection lines meet at center of projections (the vanish-point).
- Frustum

Ortographic

- No vanish-point.
- Parallell lines never meet
- Isometric , Architechtual blueprints


GRAPHICS PIPELINE

Programmable pipeline

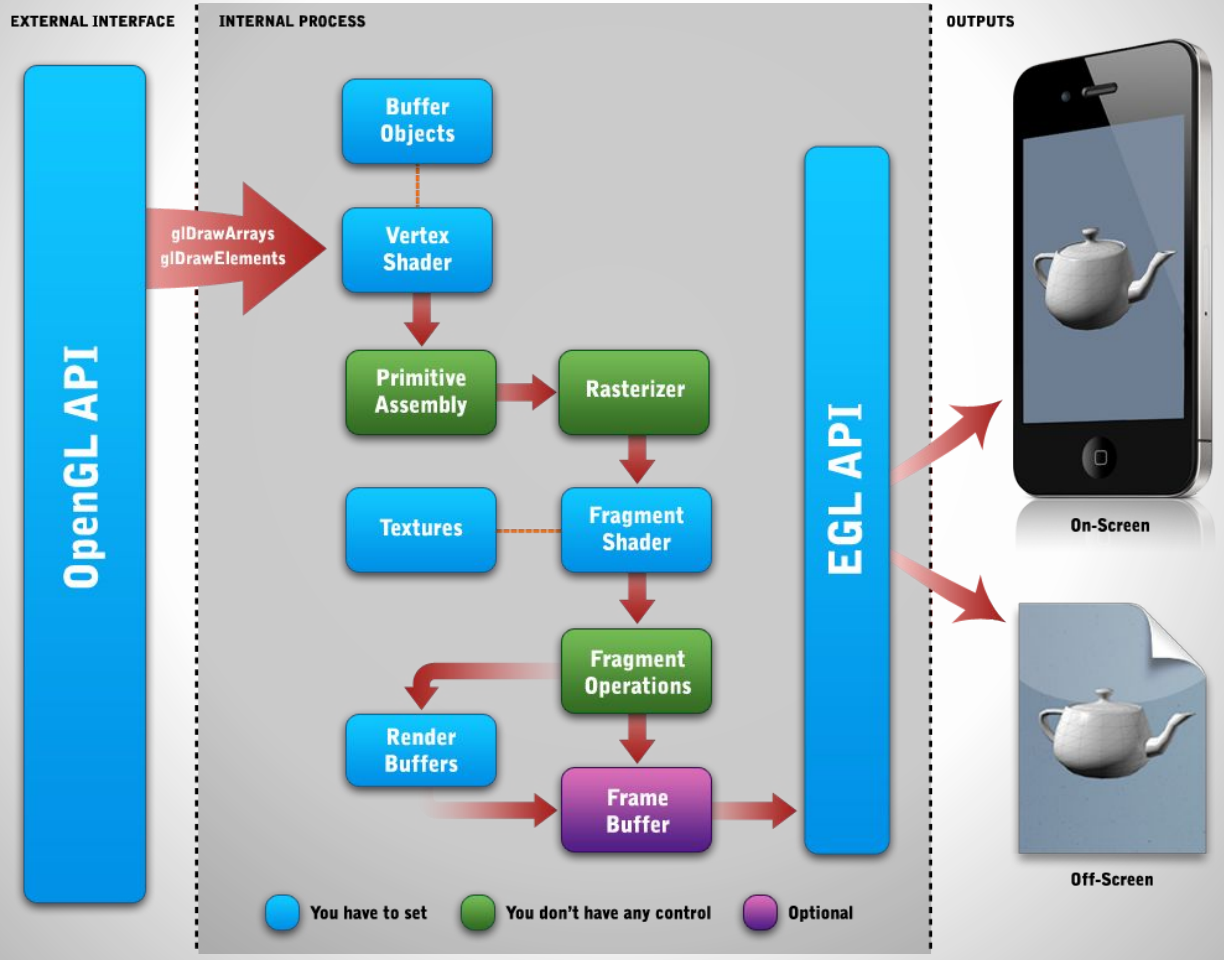
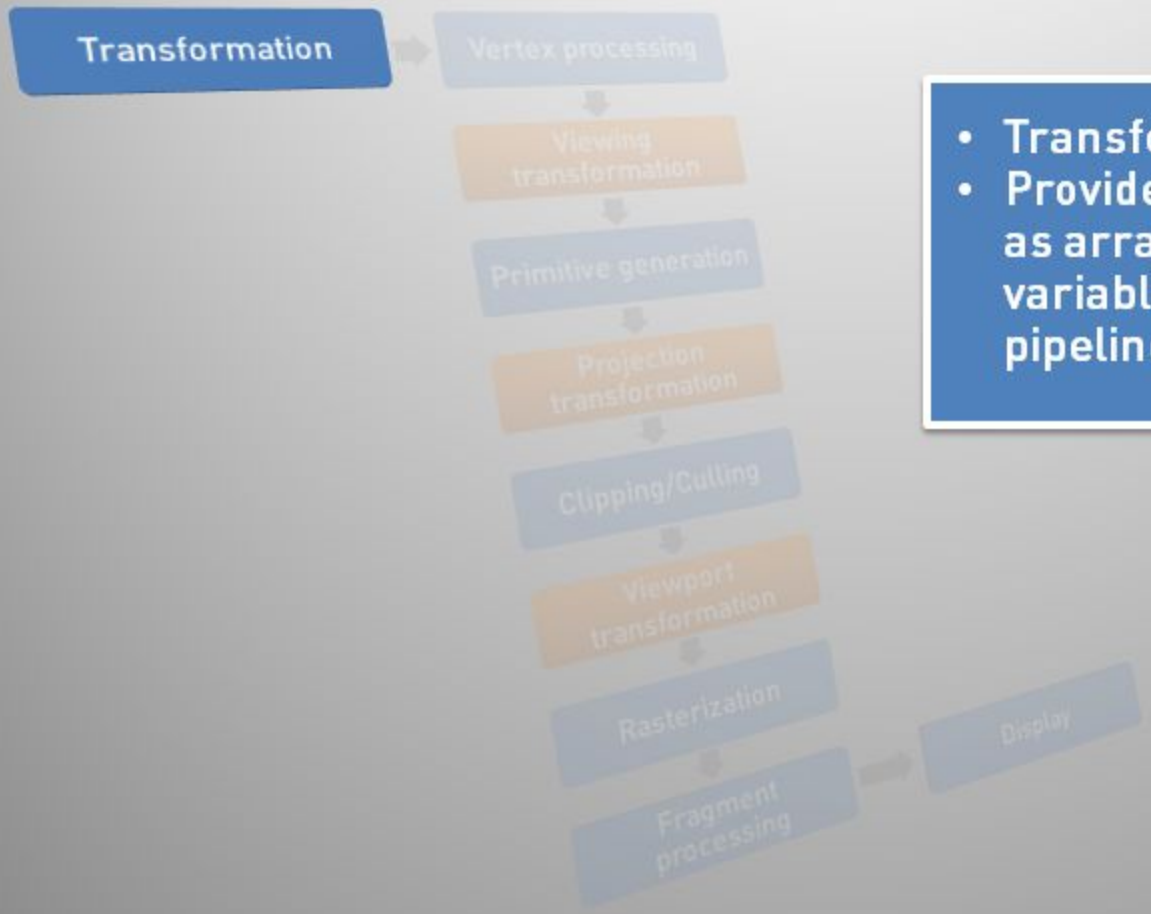


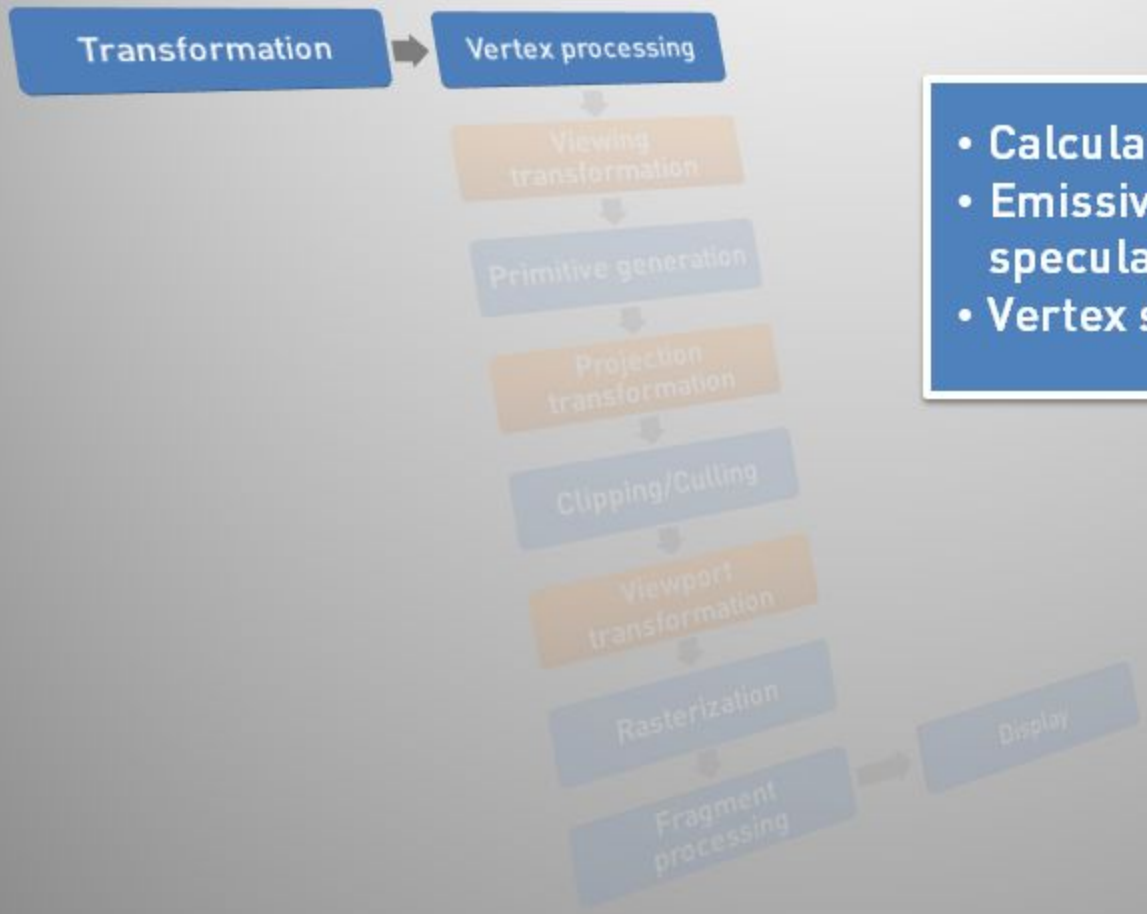
Illustration from db-in.com

Stages overview



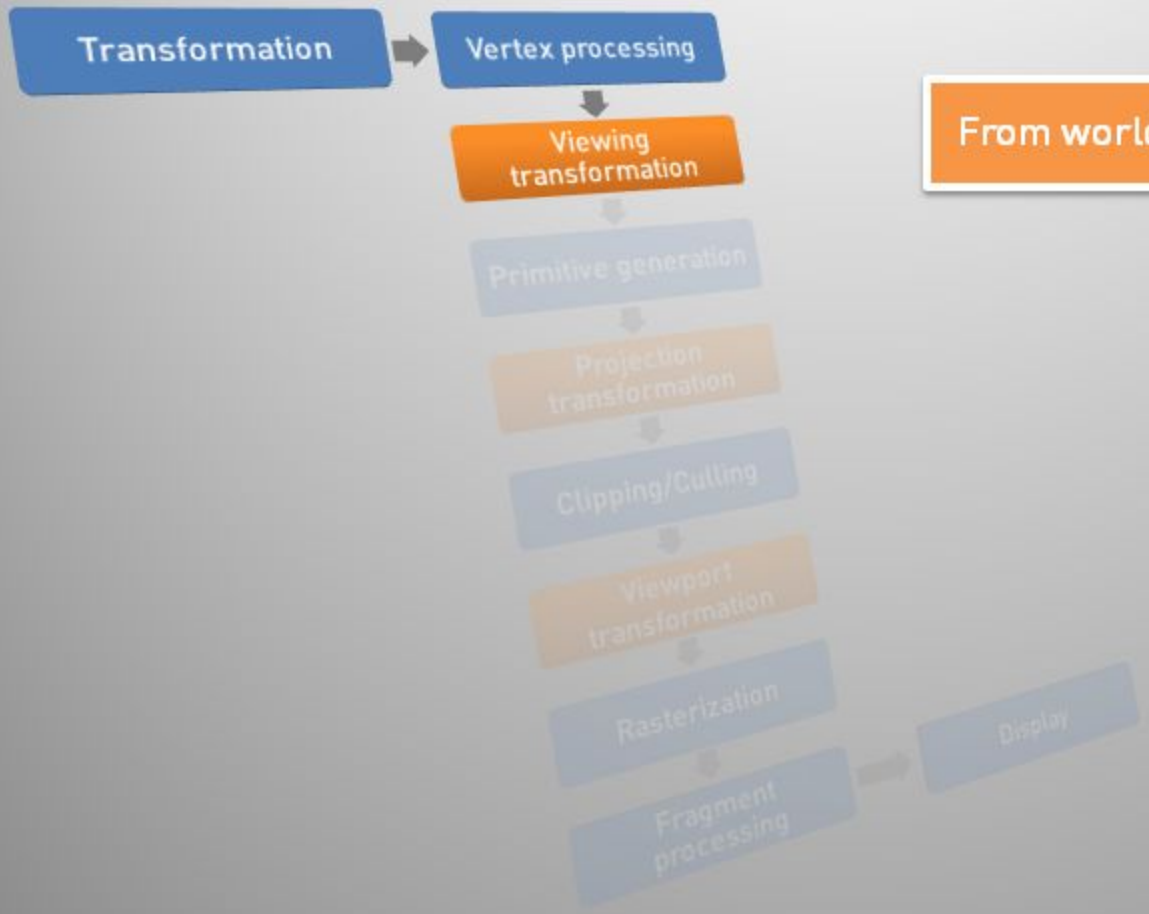
- Transformations
- Provide vertices and indicies as arrays and variables/constants to pipeline input.

Stages overview



- Calculate lighting on each vertex.
- Emissive + ambient + diffuse + specular → output vertex color
- Vertex shader

Stages overview

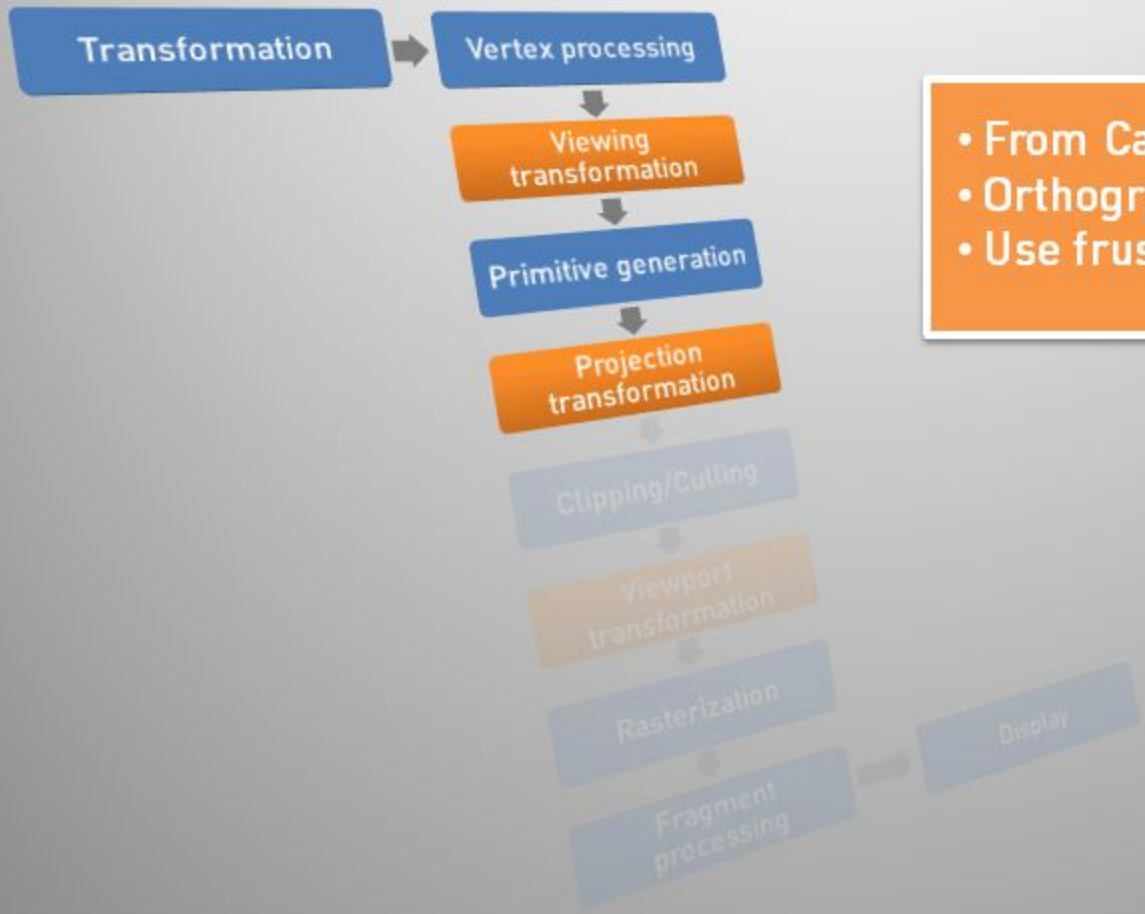


From world space to camera space

Stages overview

- Primitive Assembling.
- If geometry shader is available, new primitives can be generated.

Stages overview

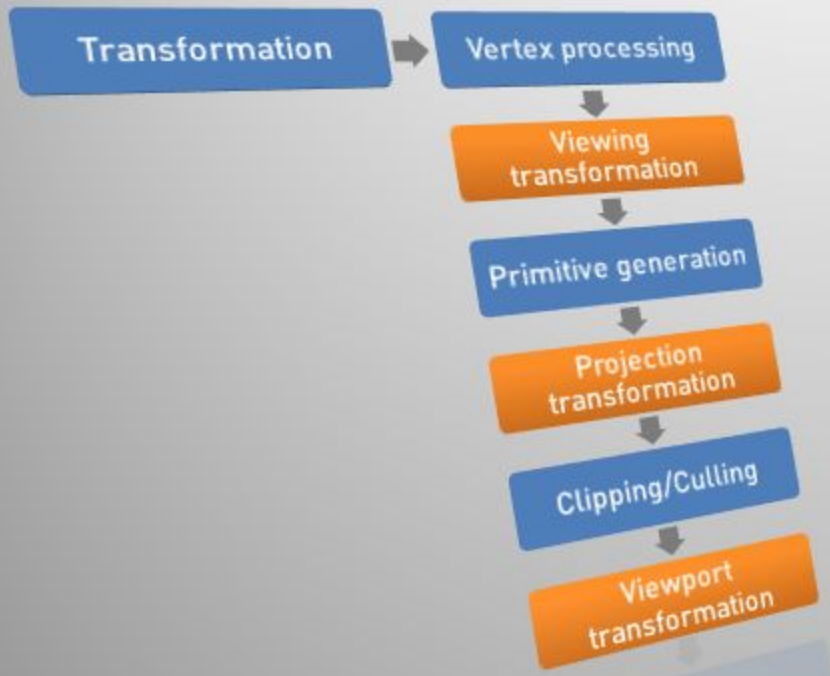


- From Camera Space to Clip Space
- Orthographic or Perspective
- Use frustum box

Stages overview

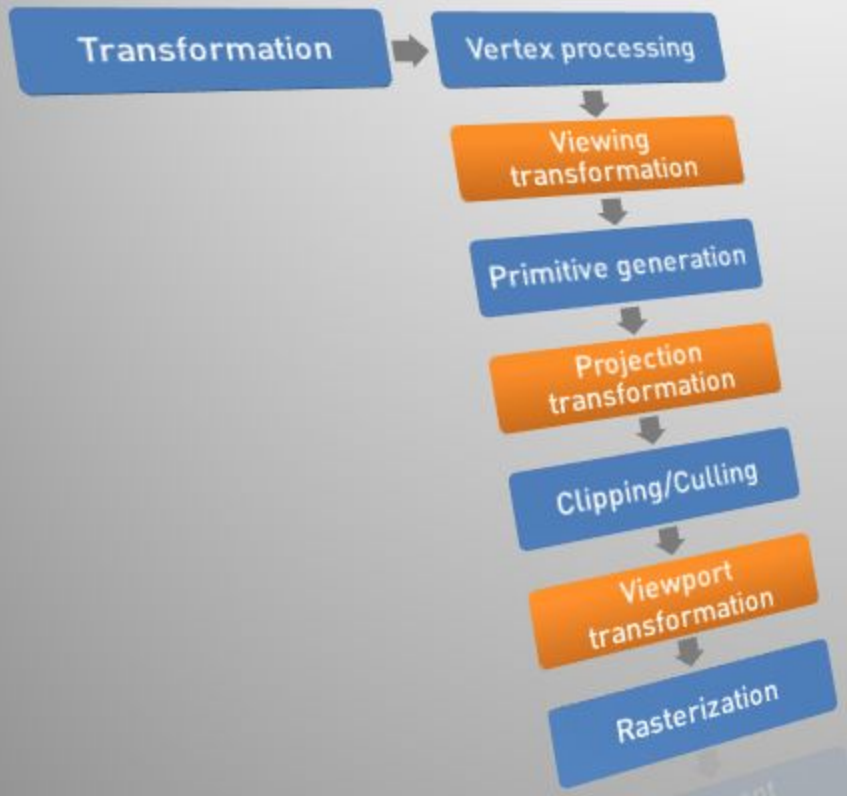
- Don't render what we can't see
- Clipping:
  Remove primitives outside of the camera's view frustum
- Back-face culling:
  Remove triangles facing away from camera

Stages overview



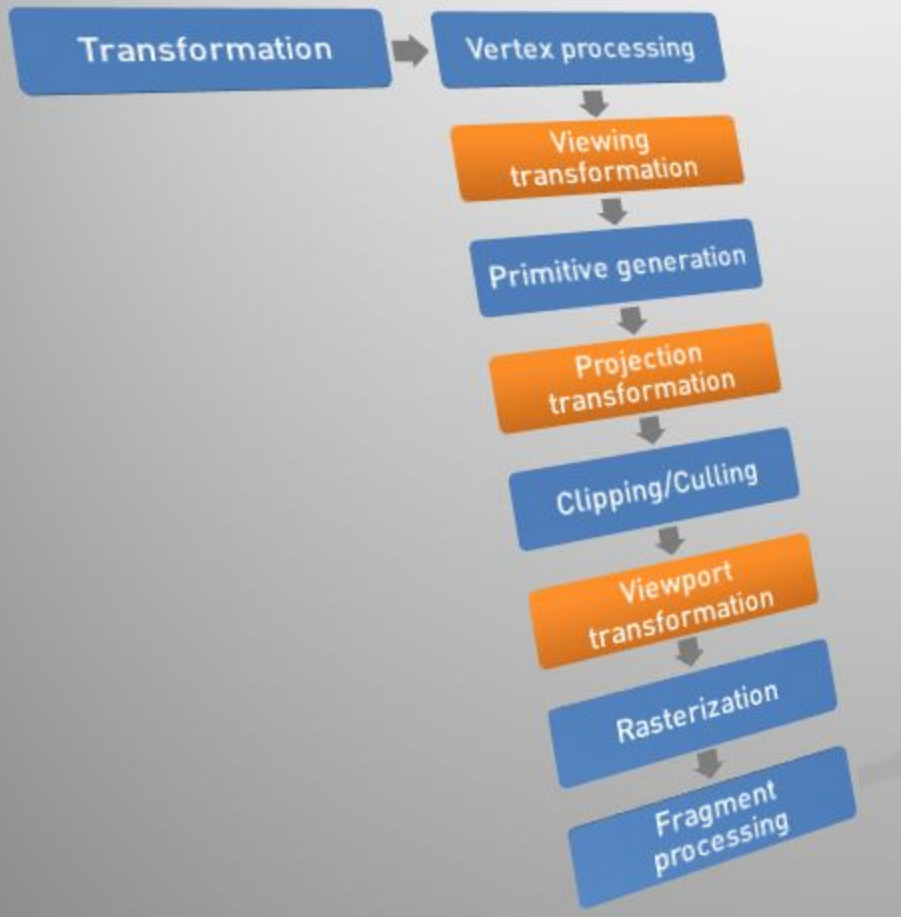
- From Clip Space to Window Space.
- e.g. [-1,1] → [0,640]
- Z-value retained for testing.

Stages overview



- Convert geometry into fragments
- (r,g,b,a), (x,y,z,w), (tx,ty)
- Interpolate vertex colors/texture coordinates over the fragment.
- Each fragment has RGB color and depth value (z-buffer)

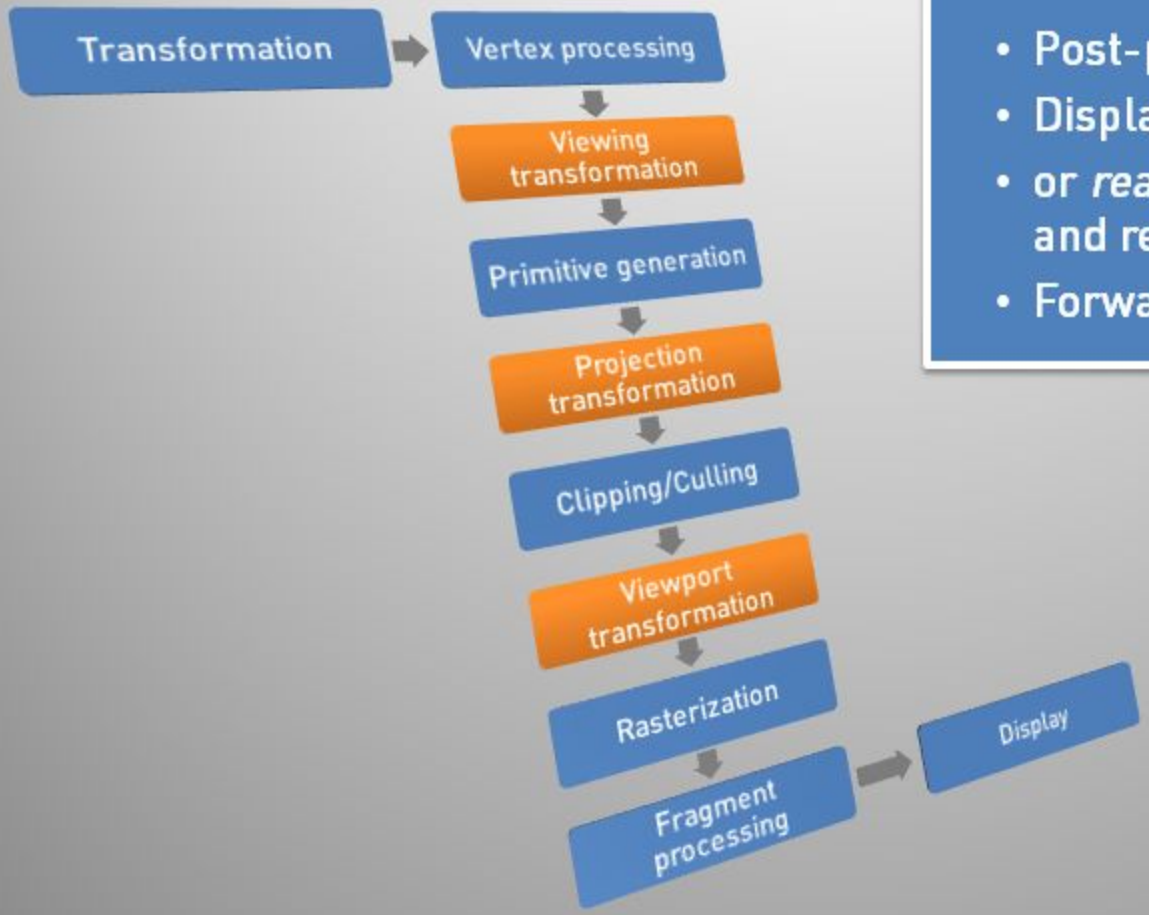
Stages overview



- Fragment shader
- Texturing
- Fog
- Series of tests with increasing complexity:

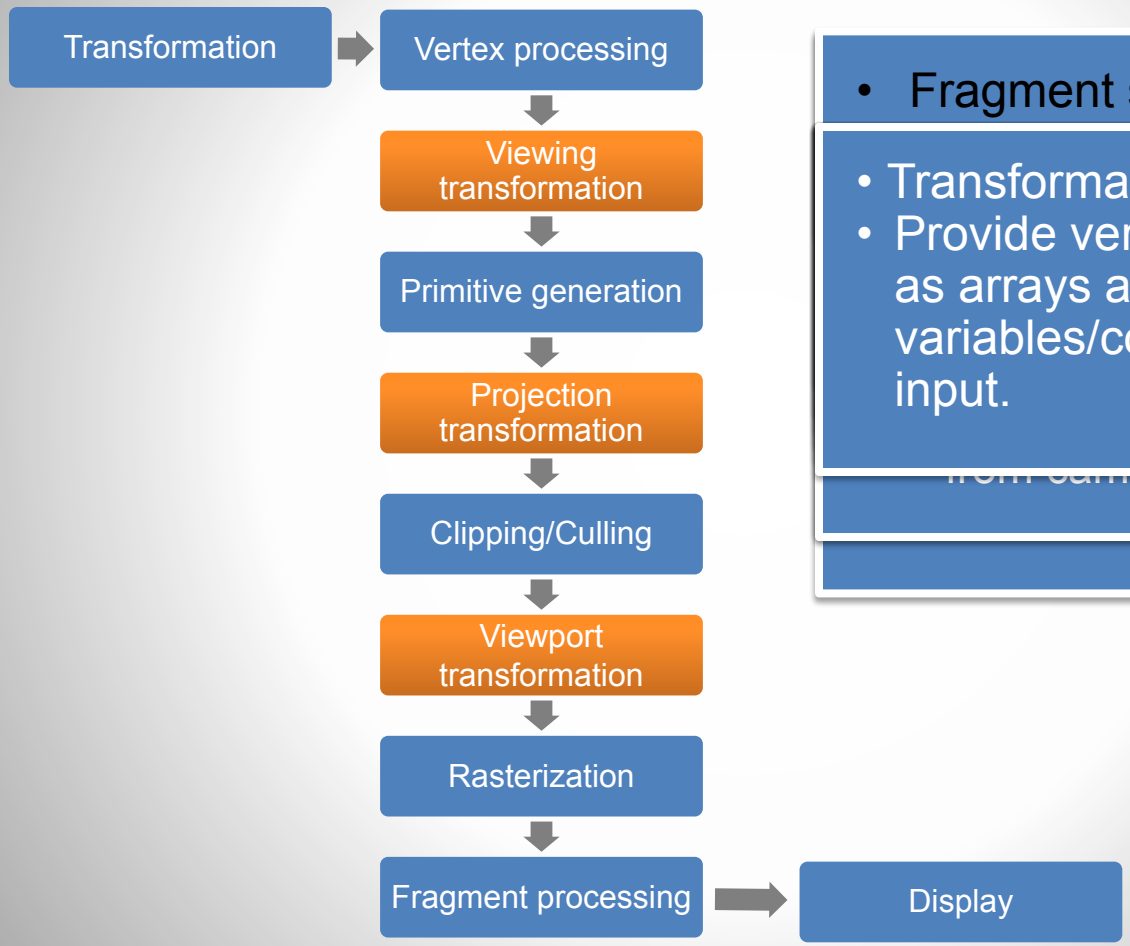
Scissor → Alpha → Stencil → Depth → Blend

Stages overview



- Post-processing
- Display on screen
- or *readback*: Render to buffer and retrieve values. Really slow!
- Forward / Deferred rendering

Stages overview

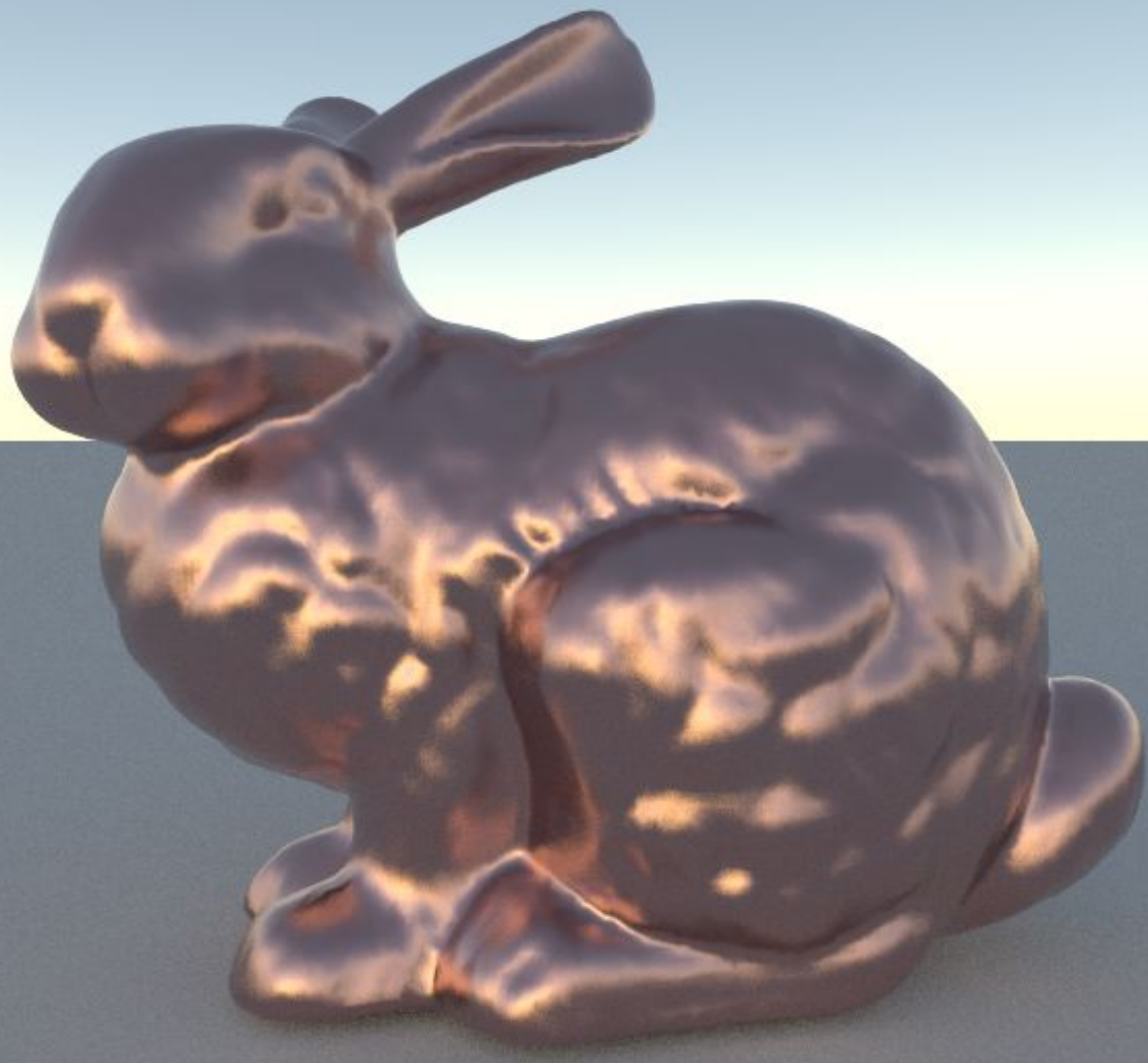


- Fragment shader

- Transformations
- Provide vertices and indicies as arrays and variables/constants to pipeline input.

SHADERS

The method to render an object.



About shaders

- Small programs that runs on the GPU.
- Most shader languages are the same.
- *Vertex* and *Fragment* shaders work in pairs.
- The pair is compiled into a *Program*
- Uniforms, Attributes, Varyings, Built in attributes

Low level shading language

- Assembly language
- ARB (GPU)

```
!!ARBfp1.0
TEMP color;
MUL color, fragment.texcoord[0].y, 2.0;
ADD color, 1.0, -color;
ABS color, color;
ADD result.color, 1.0, -color;
MOV result.color.a, 1.0;
```

- AGAL (Adobe Graphics Assembly Language)

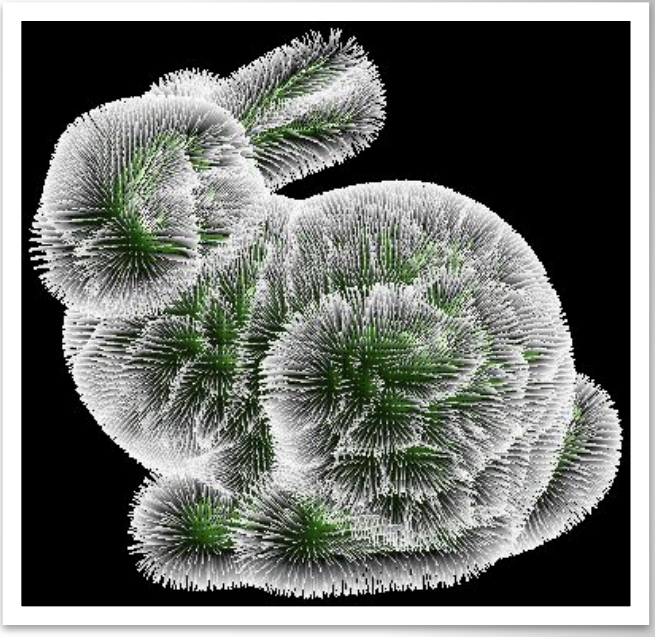
High level shading languages

- HLSL – DirectX API
- Cg – NVIDIA
- GLSL – OpenGL
- ShaderLab – Unity3D
- PixelBender3D – Molehill
- HxSL – haXe Shader

Vertex shader

- VS or VSH
- Executed at each vertex
- Transform between coordinate systems
- Lighting
- Defines the final position of that vertex
- Outputs some variables to the Fragment shader.

Geometry Shader



- Dynamic creation of geometry on the GPU
- Only Shader Model 4.0
- Direct3D 10, OpenGL 3.2
- Not available in OpenGL ES 2.0 (Molehill, webGL)

Fragment Shader

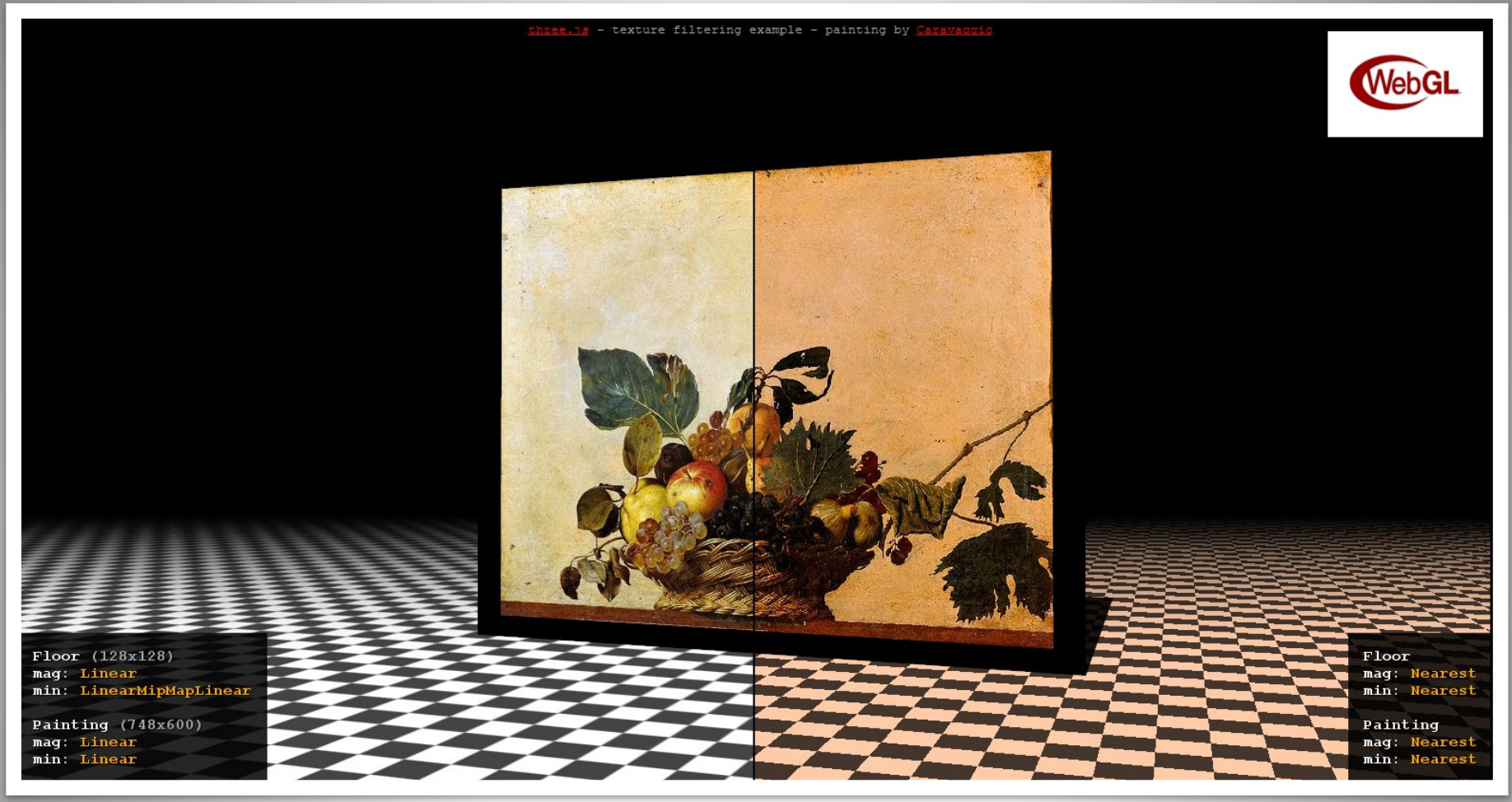
- FSH
- Processed at each visible fragment
- Fragment != Pixel
- Handles bump effects, shadows and lights, reflections, refractions, textures, ray casting and other effects.
- Output is a pixel color in the format RGBA

Texture objects

- Texels
- Power of Two (POT) 2, 4,…512, 1024 pixels
- Flipped pixel order (OpenGL)
- Integer/Floating-point

Texture Filtering

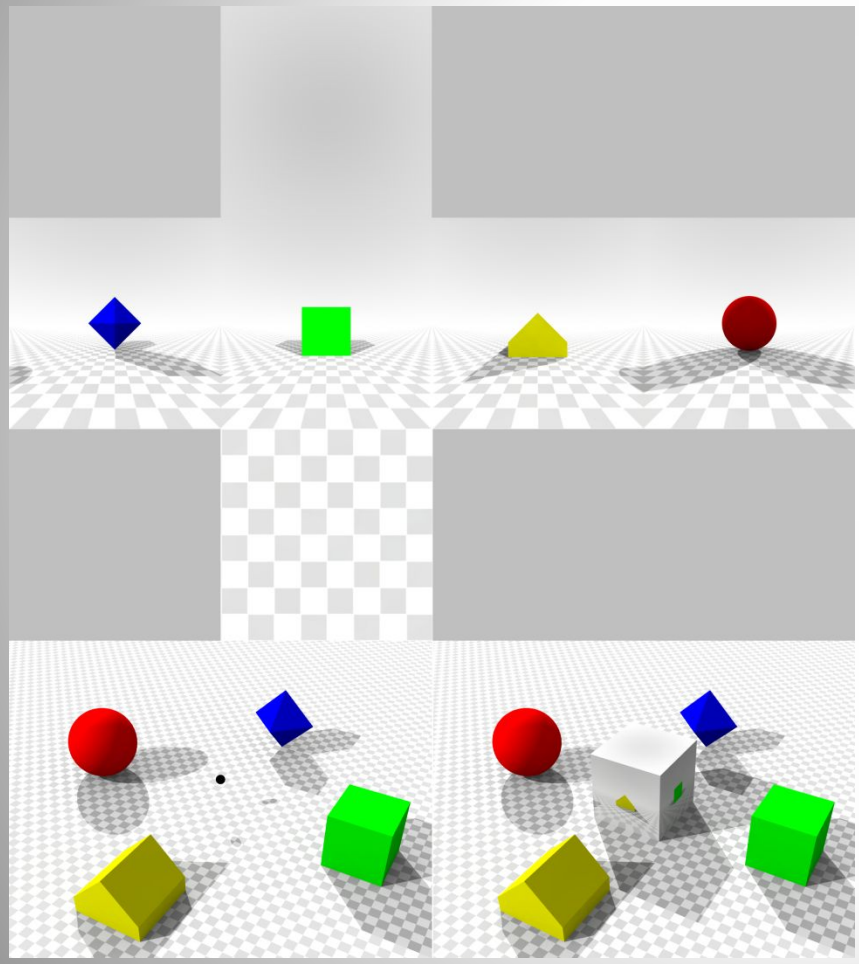
- Fixing artifacts
- Texture **mag**nification/**min**ification
- Mipmapping
- Different techniques:



Let’s have a look at the WegGL implementation (click on image)

three.js

Cubemap texture



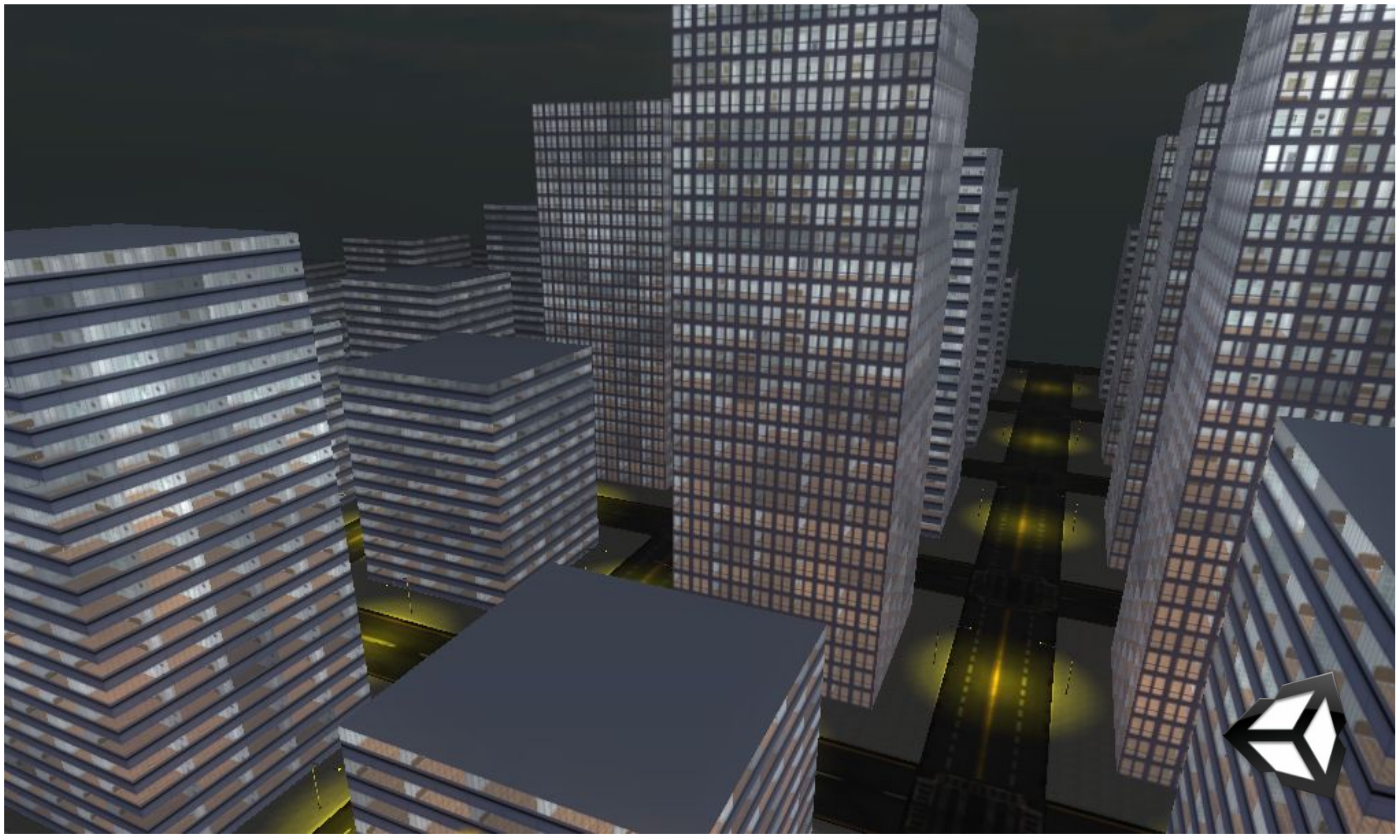
- 3D texture
- Skybox
- Reflections
- Environment map

Shader tool examples

- Shader Toy – WebGL
- MeShade – WebGL
- PixelBender3D – Molehill
- Node Based Shader Editor – Unity3D

DEMO

Interior mapping



Animations, Skin and Bones

- Tweens
- Animation controllers
  Blending
  Mixing/Additive
- Vertex animations in shader
- Procedurally animating

DEMO

Animations in Away3D Broomstick

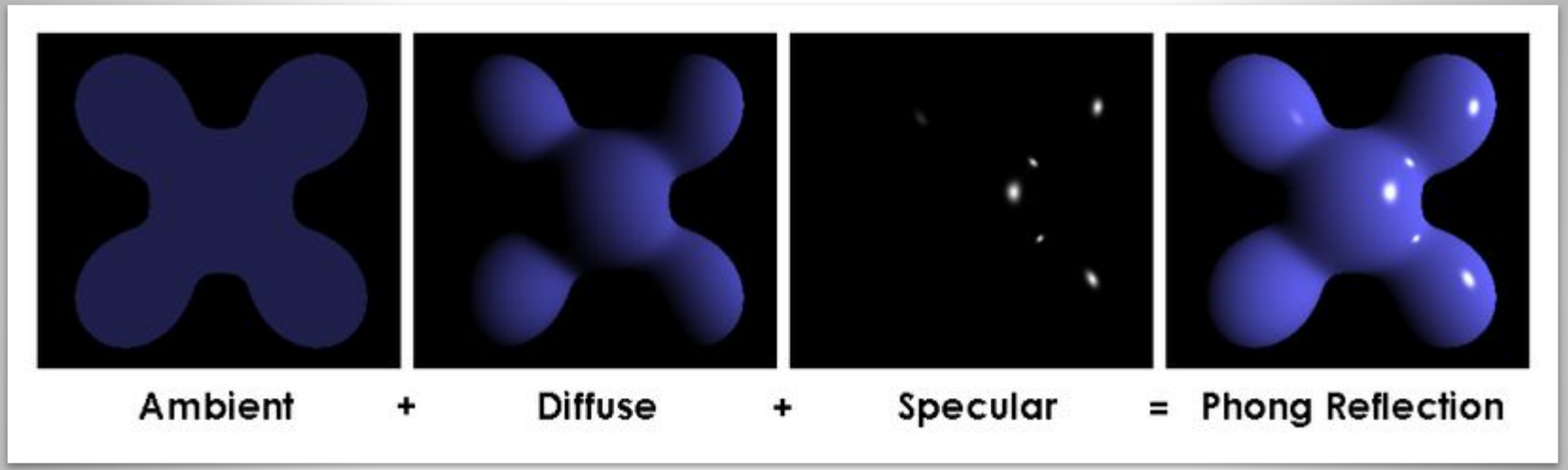
Materials

- Material is the collection of properties applied to an object.
- Shaders is the implemention. ”The code”
- In Unity, think that materials is a collection of exposed properties of the shader.

Some ingredients:

- Color
  - Diffuse: base color
  - Ambient: color of ambient light (shadowed parts). Mostly the same as diffuse.
  - Specular: Highlight color
  - Emissive: Glow. Overrides shadows.
  - Alpha: Transparency
- Texture (2D,Cubemap)
- Shininess: size of specular highlights (gloss)
- Reflection/Refraction
- Bump-mapping: height, grayscaled image
- Normal-mapping: Dot3 bump mapping, xyz->rgb
- Paralax-mapping: height + direction, graycaled+rgb

Example



Unitys Normal Shader Family

VertexLit

Diffuse

Normal mapped

Specular

Normal Mapped Specular

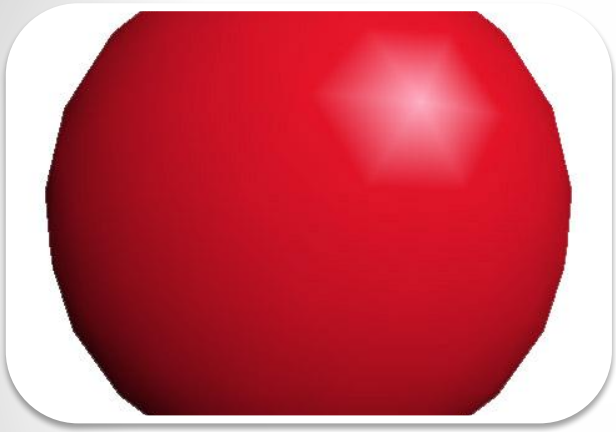
Parallax Normal mapped

Parallax Normal Mapped Specular

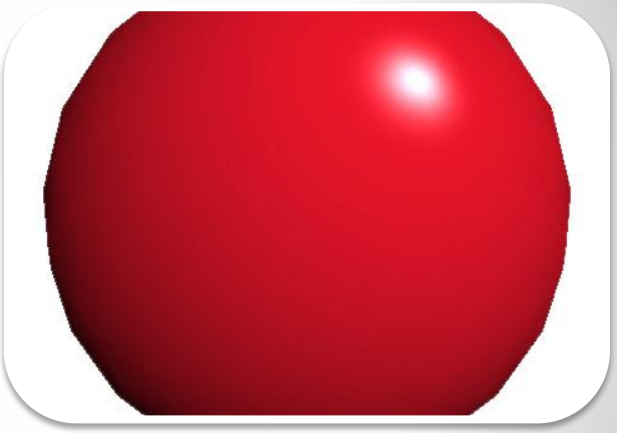
Lighting

- Uses normals
- Directional/point-lights
- Material settings to decide final color.
- Lighting is computed at each vertex.
- Light mapping (beast)
- Deferred shading

Lambert shading

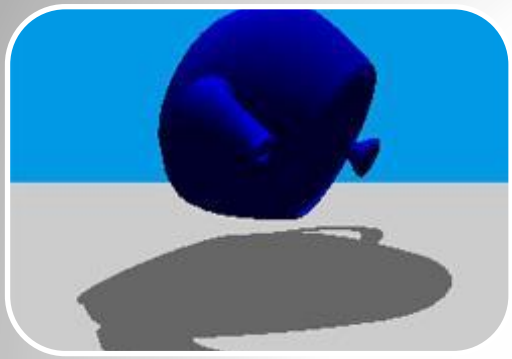


Gourau
d



Phong

Real-time shadows



PLANE PROJECTION SHADOWS

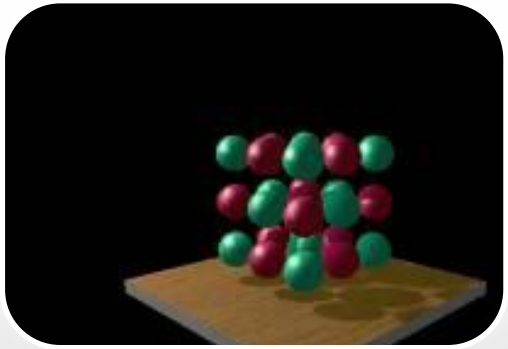
- Flattened objects/imposters on planar surfaces
- Fast but unrealistic
- No self-shadows

SHADOW VOLUMES

- Computationally heavy
- High detail
- Self-shadowing
- Using stencil buffer or texture

DEPTH SHADOW MAPPING

- Hardware
- Self shadowing
- Hard shadows: nearest map pixel
- Soft shadows: average map pixels



VERTEX PROJECTION

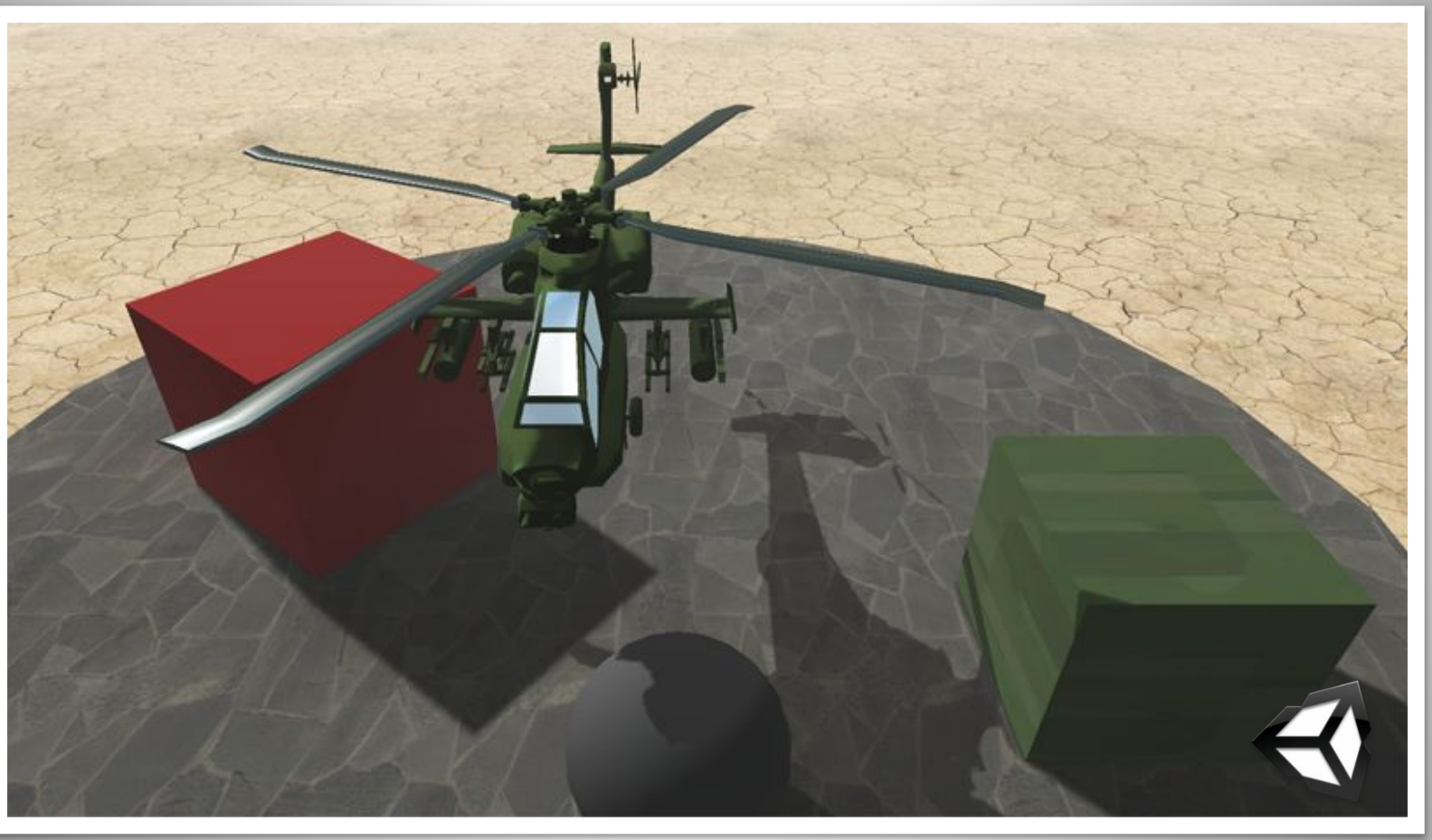
- Like plane projection but with calculated silluette.

Quality and performance

- Non realtime-shadows fastest!
- Shadow map resolution
- Number of lights

DEMO

Example in Unity



Special effects

- Effects
- Color correction
- Postprocessing stage / GPU
- LDR/HDR, Tone mapping

Bloom

Depth of field

Sun Shafts

SSAO

Blur

Noise

Physics

DEMO

Very simple physics demo

Frameworks

- Goal: Games, experimental, Vizualisation?
- Reach: Plugin? Multiple platforms/screens?
- Cost: Open source? Licenced?
- Support: Large community?

Unity3D

- Boo, C# and JavaScript
- Plugin
- Great and simple IDE
- Competent and mature framework
- Pro version to get all goodies
- Multiple screens/targets
- Future: Export to flash/molehill

Flash/Molehill

- Actionscript
- Plugin
- 3D content always under the DisplayList
- All the other stuff in the flash player.
- Molehill currently in alpha

Flash 3D Engines

Engine	Licence/Price	link
Away3D 4.0	Open source, free	»
Flare3D 2.0	Licence, price unknown	»
Aerys Minko	No licence, just consulting	»
Sophie 3D	Licence, 329£ (3000 kr)	»
CopperCube 2.5	Licence, 99£, professional 300£	»
Zest3D	Open source, free	»
Alternativa 8	Licence, price unknown	»
ND2D Molehill 2D Engine	Free	»
Mandreel	3000 £ (26000 kr)	»

Optimizing

- Profiling memory usage, cleanup/destroy
- Object Pooling! polygonal lab
- Take control of rendering pipeline
- Compression/Model to ByteArray
- AWD, Away3Ds own format (Prefab)
- Trends of resource-load in online 3D?
- Optimize opcodes in swf: http://www.buraks.com/azoth/


WebGL

WebGL

- Javascript
- No plugin
- Open / Royalty-free
- Not available in all browsers yet
- Frameworks in early states
- Probably available on iOS soon

WebGL Frameworks

GLGE

Canvas 3D JS Library

CopperLicht

EnergizeGL

O3D

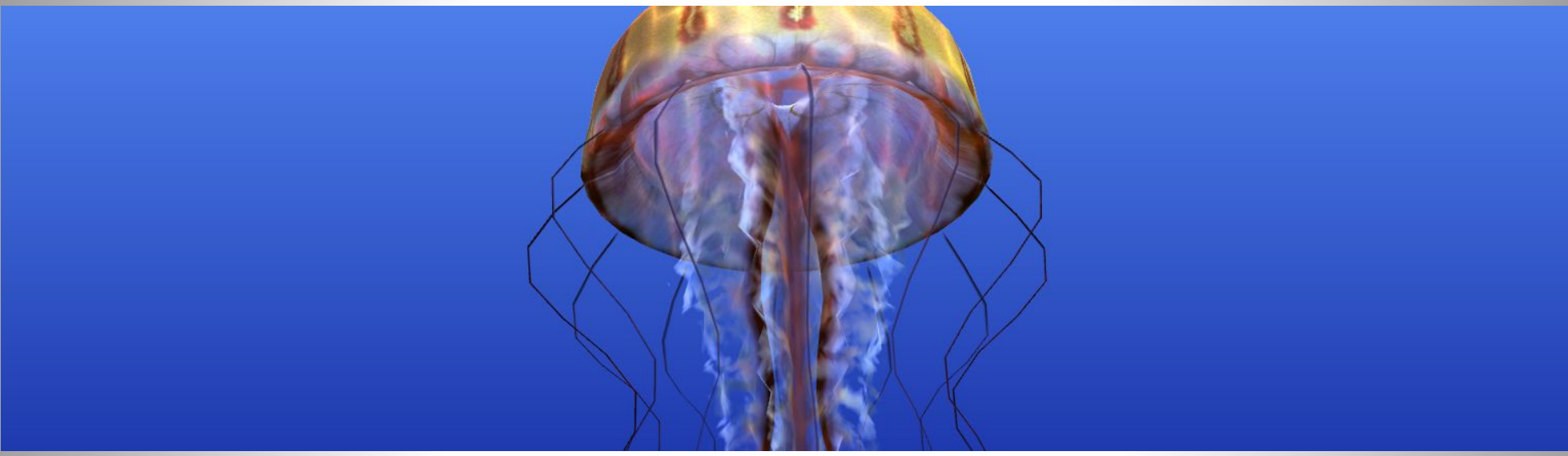
SpiderGL

Three.js

OSG.JS

DEMO

Jellyfish

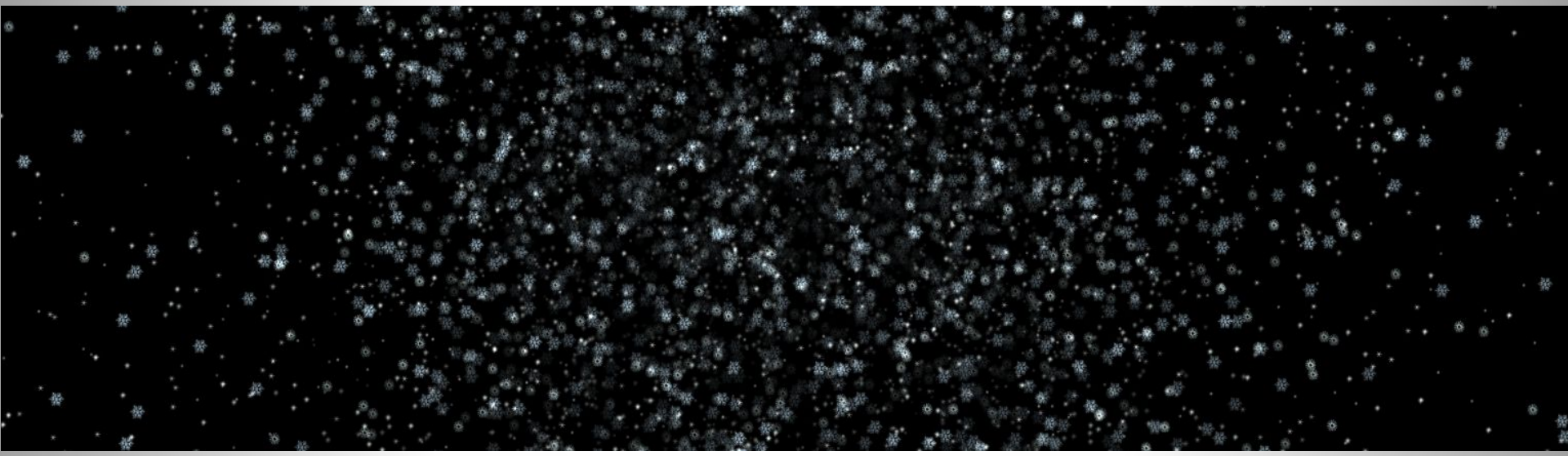


Aleksandar Rodic

WebGL

DEMO

Particles



alteredqualia.com

DEMO

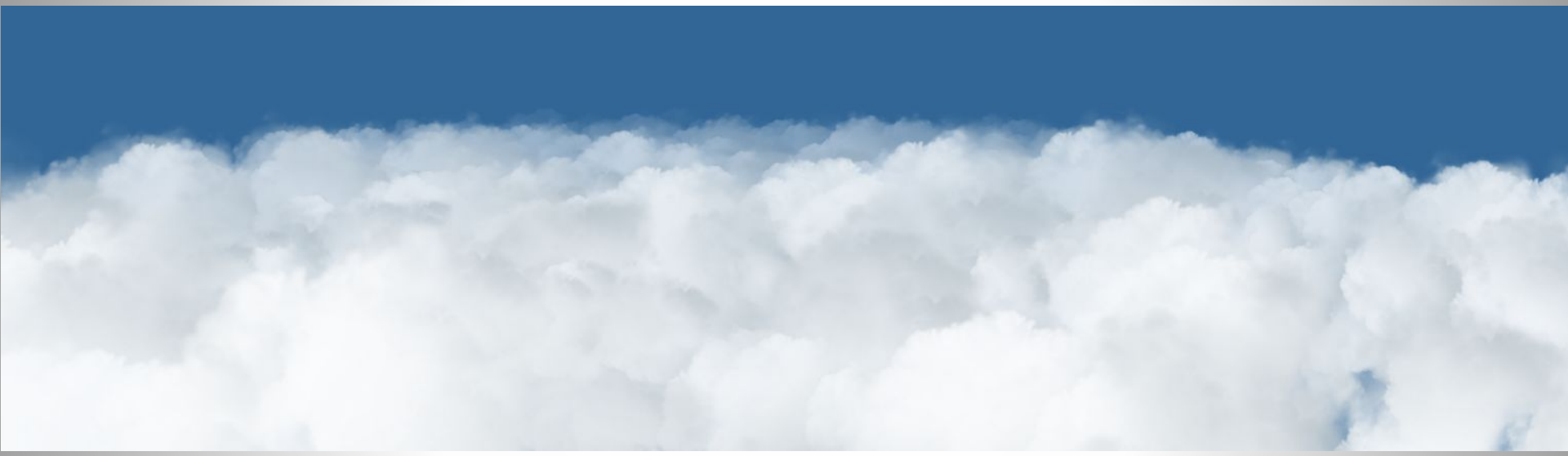
Hello Racer

HelloEnjoy™

WebGL

DEMO

Clouds



Mr Doob

WebGL vs. Molehill APIs

- HTML5 vs. Plugin.
- WebGL will probably run in iOS browser.
- Easy to port between them.
- Once it running on the GPU, performance is hardware related regardless of API.
- It is the high level frameworks that makes the difference.

Debugging

- Profiling CPU
  - FlashPreloadProfiler
- Profiling GPU
  - Pix for windows
  - Intel® Graphics Performance Analyzers (GPA)

3D Model filetypes

Format	Ext	Away3D	Unity3D	Dynamic
Actionscript	.AS	x		
Autodesk® FBX®	.FBX (MAX)		x	x
Wavefront	.OBJ	x	x	
Collada	.DAE	x	x	x
Quake 2	.MD2	x	x	x
Quake 3	.MD5	x	x	x
3ds Max object	.3DS	x	x	
Away 3D	.AWD	x		x

Learning tips

Try some tutorials with Molehill API or WebGL to get an understanding of the pipeline

Read, follow, blog, get interested!

Pay attention to techniques outside your own field. SIGGRAPH, GPU gems, Nvidia.

Get familiar with existing work. We'll get there eventually.

Port code from another language.

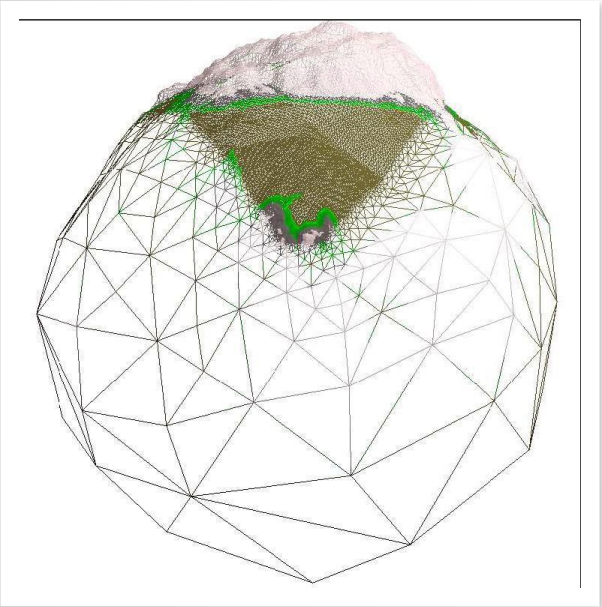
Tech is one thing, art is another. Good artwork is what makes it successful in the end.

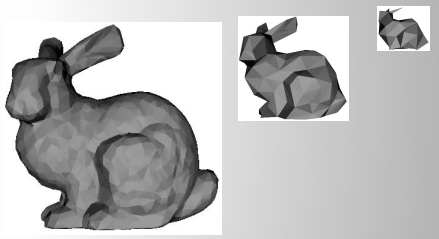
Stand on the shoulders of giants.

Random interesting topics

Random interesting topics

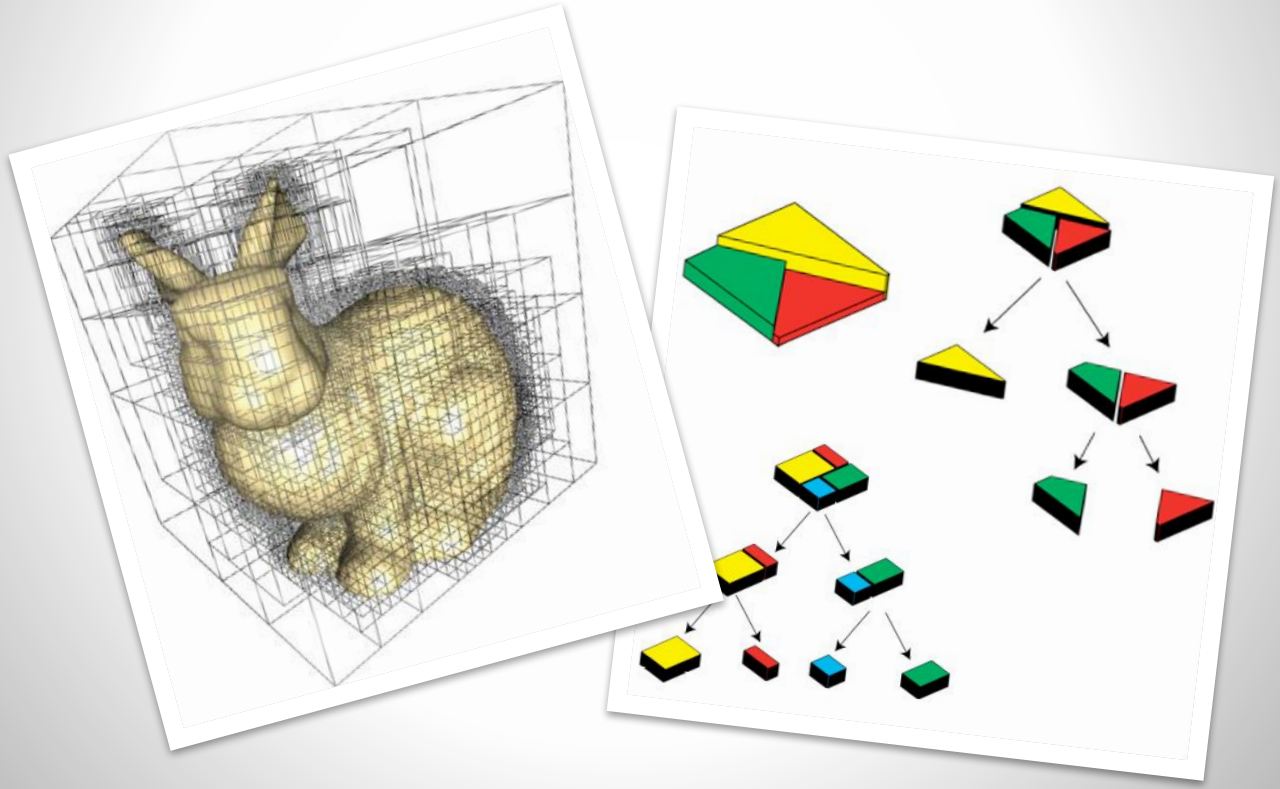
Level of detail





Random interesting topics

Octree, BSP Tree, Portals and Sectors

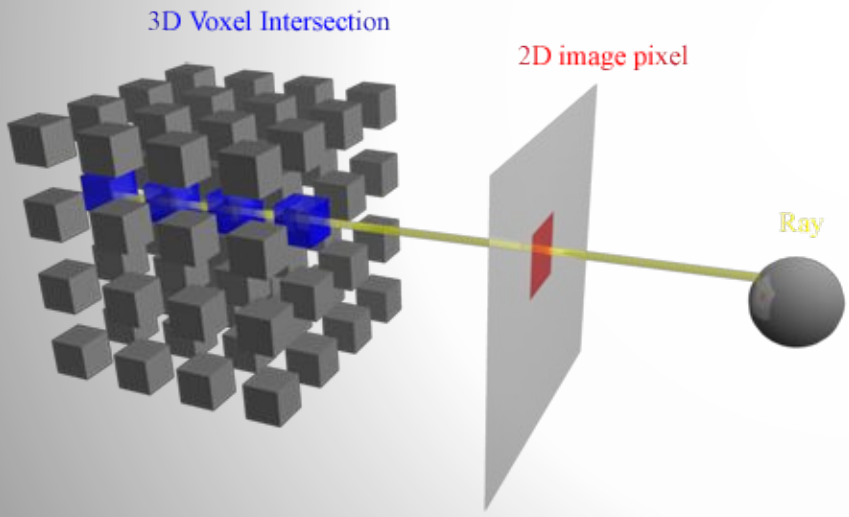


Random interesting topics

Global illumination / Ambient occlusion

Random interesting topics

Raytracing/Raycasting/Raymarching



Some useful resources

COMPUTER GRAPHICS

- SIGGRAPH papers: http://kesen.realtimerendering.com/
- GEEKS3D: http://www.geeks3d.com/
- Miles Macklins Blog: https://mmack.wordpress.com/
- GAMEDEV: http://www.gamedev.net/index
- Teaching machines: http://www.twodee.org/blog/

OpenGL / WebGL

- OpenGL resources: http://www.opengl.org/
- Game programming community: http://www.gamedev.net/
- OpenGl tutorial: http://db-in.com/blog/2011/01/all-about-opengl-es-2-x-part-13/
- ShaderToy WebGL http://www.iquilezles.org/apps/shadertoy/
- Fractal Lab: http://fractal.io/
- CG tutorial: http://http.developer.nvidia.com/CgTutorial/cg_tutorial_chapter01.html
- ModelViewMatrix explained: http://db-in.com/blog/2011/04/cameras-on-opengl-es-2-x/

FLASH

- Away3D 3.6 Tutorials: http://www.flashmagazine.com/Tutorials/category/3d/
- Creative coding podcast: http://creativecodingpodcast.com/

MOLEHILL

- 3d vs. flash tips: http://blog.bengarney.com/2010/11/01/tips-for-flash-developers-looking-at-hardware-3d/
- Molehill getting started: http://labs.jam3.ca/2011/03/molehill-getting-started/
- Digging into Molehill API: http://www.bytearray.org/?p=2555
- Molehill resources: http://www.uza.lt/2011/02/27/molehill-roundup/
- Molehill demos: http://tinyurl.com/molehilldemos
- Demystifying molehill: http://www.rictus.com/muchado/2011/02/28/demystifying-molehill-part-1/
- Slides about Zombie Tycoon: http://molehill.zombietycoon.com/FGSZombieNoVideos.pptx

TOOLS

- Pix GPU profiling: http://msdn.microsoft.com/en-us/library/ee417072(v=VS.85).aspx

UNITY

- Video tutorials: http://www.3dbuzz.com/vbforum/content.php?176

Books and papers

- Away3D 3.6 essentials
- Mathematics for Game Developer by Christopher Tremblay
- Mathematics for 3D Game Programming and Computer Graphics by Eric Lengyel
- Game Graphics Programming by Allen Sherrod
- Realtime shadows
- Raycasting in GPU shaders by Joost van Dongen

Thanks!

Wow! You made it all the way here! I hope you got inspired to continue your journey into the third dimension. Thanks for listening!

www.inear.se

twitter.com/inear