glTF 2.0 Launch

Web3D Conference, June 2017

Neil Trevett | Khronos President
NVIDIA Vice President Developer Ecosystem
ntrevett@nvidia.com | @neilt3d

glTF - Efficient Runtime 3D Asset Delivery

Audio	Video	Images	3D
MP3	H.264	JPEG	glTF™
napster	YouTube™	facebook	New market opportunities for 3D content creation and deployment!

model/gltf+json MIME type Approved by IANA

Portable 3D Graphics Transmission Format
A vital building block for the Metaverse
Widely shared avatars, objects, information
Avoids siloed, per-app, per-platform content formats

Compact to Transmit 

Fast to Load 

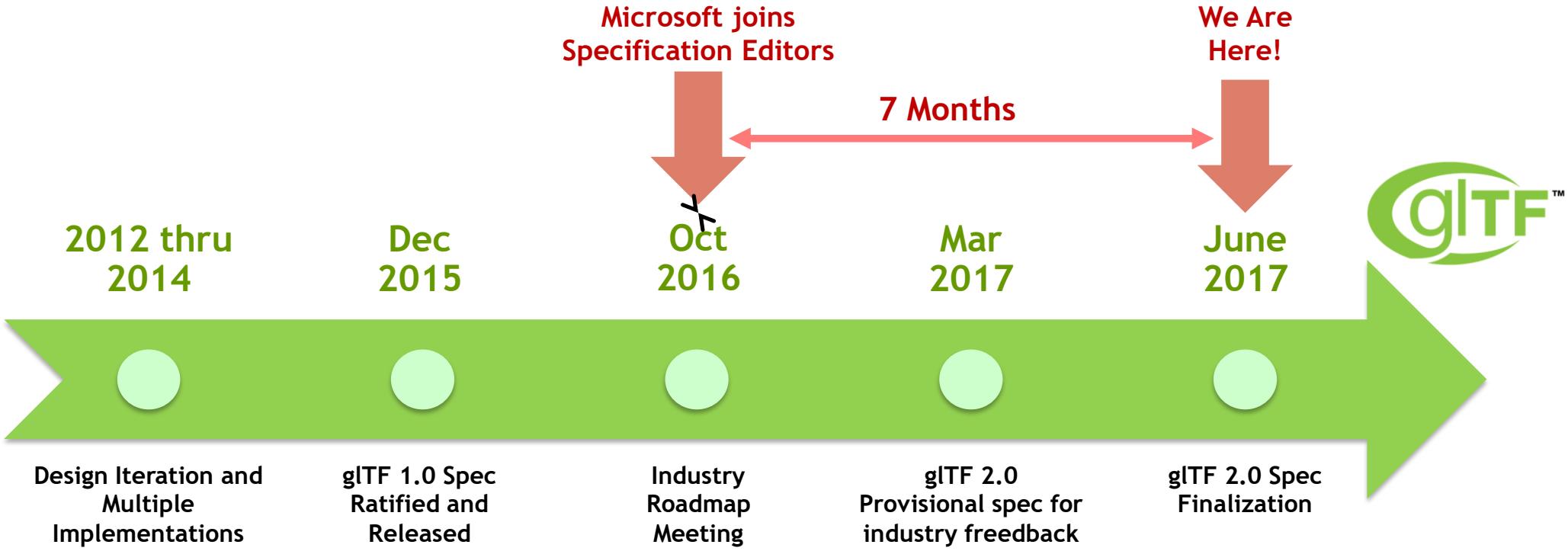
Describes Full Scenes 

Runtime Neutral 

Open and Extensible 

KHRONOS GROUP

glTF Milestones

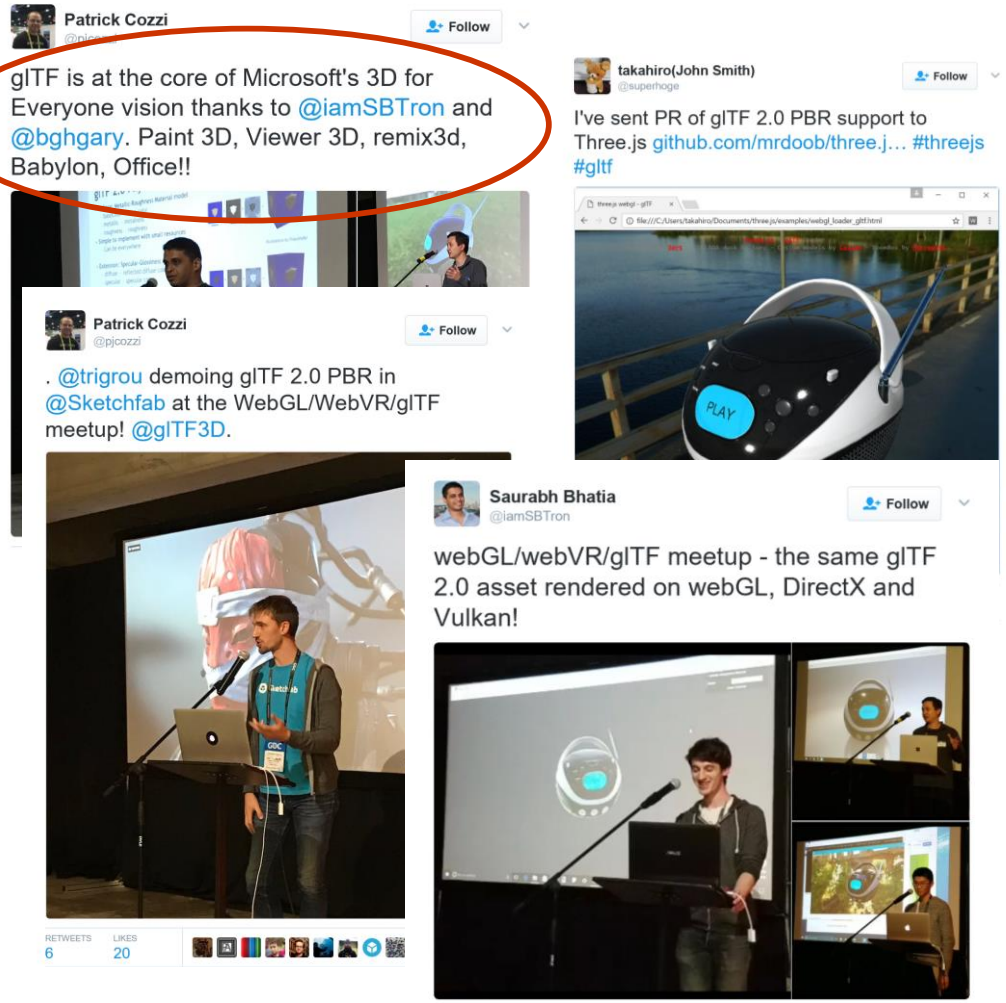


Original motivation: standardized way to deliver 3D into WebGL applications

Significant Industry Adoption

Physically Based Rendering for higher-quality materials and rendering API independence

Strong glTF Momentum



oculus
web 3D CONSORTIUM
MPEG
Adobe
UX3D
blender
OGC
box
Microsoft
otoy
Fraunhofer
AREA Augmented Reality for Enterprise Alliance
NVIDIA
unity
Sketchfab
CESIUM
AUTODESK
Google

Publicly Stated Support for glTF

glTF Ecosystem

All glTF spec development on open GitHub:
https://github.com/KhronosGroup/glTF

Visual Studio
glTF in Visual Studio Code
glTF-VSCode extension
https://marketplace.visualstudio.com/items?itemName=cesium.gltf-vscode

blender
Blender DIRECT exporter in progress

unity
Unity DIRECT Exporter in Progress

Tools
Export

KHRONOS GROUP
glTF open source Validator

Translators
Convert | Optimize

glTF

Validator
Validate

AUTODESK.
Drag and Drop FBX -> glTF
(coming soon)
http://gltf.autodesk.io/

COLLADA
Drag and drop COLLADA -> glTF
http://cesiumjs.org/convertmodel.html

Other Translators
AssImp
OBJ2GLTF
glTF Pipeline
COLLADA2GLTF
Cesium converter

Apps & Engines
Import

A-FRAME
Sketchfab
NVIDIA
nvpro-pipeline
babylon.js
CESIUM
three.js
PEX
xeoEngine

KHRONOS GROUP

What's New in glTF 2.0

- **Physically Based Rendering (PBR) material definitions**
  - Material information stored in textures
- **Graphics API neutral**
  - Proven by engine implementations using WebGL, Vulkan and Direct3D
  - GLSL materials moved to extension for existing content and specialized use cases
- **Morph Targets**
  - Enhanced animation system
- **Improvements**
  - Binary glTF in core
  - Dozens of refinements for enhanced performance and a tighter, clearer specification

Sketchfab User: theblueturtle
https://sketchfab.com/models/b81008d513954189a063ff901f7abfe4

glTF 2.0 Scene Description Structure

.gltf (JSON)

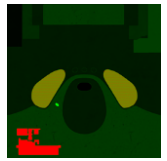
Node hierarchy, PBR material textures, cameras

.bin

Geometry: vertices and indices
Animation: key-frames
Skins: inverse-bind matrices

.png
.jpg
...
Textures

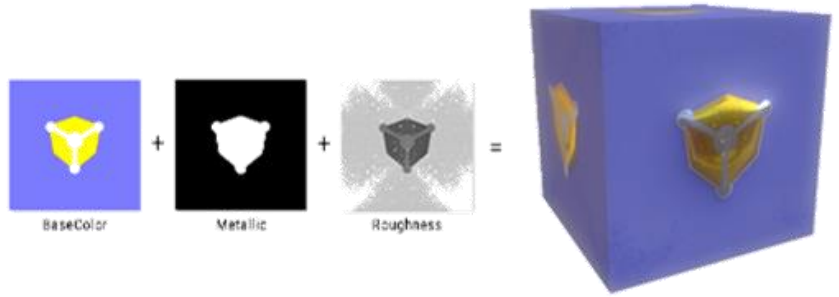
Geometry



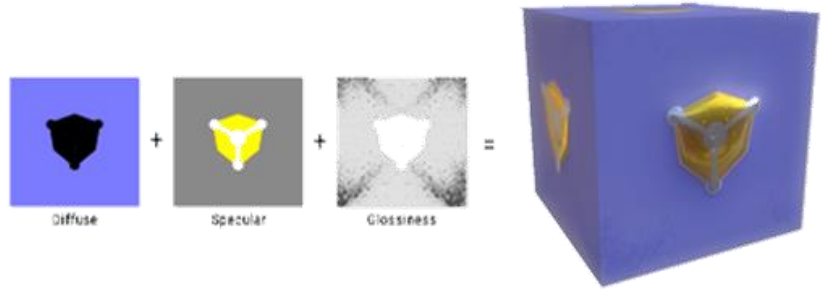
Texture based
PBR materials

glTF 2.0 Scalable, Portable PBR

- **glTF 2.0 PBR Requirements**
  - Simple and inexpensive to implement
    - So can be everywhere – even mobile devices
  - Scalable
    - Two combinable models – but materials continue to work even if just core supported
- **Metallic-Roughness Material model**
  - baseColor — base color
  - metallic — metalness
  - roughness — roughness
  - Simple to implement - mandated in core
- **Specular-Glossiness Material model**
  - diffuse — reflected diffuse color
  - specular — specular color
  - glossiness — glossiness
  - Slightly more resources - optional extension



Illustrations by Fraunhofer



glTF 2.0 PBR – Consistency Across Engines

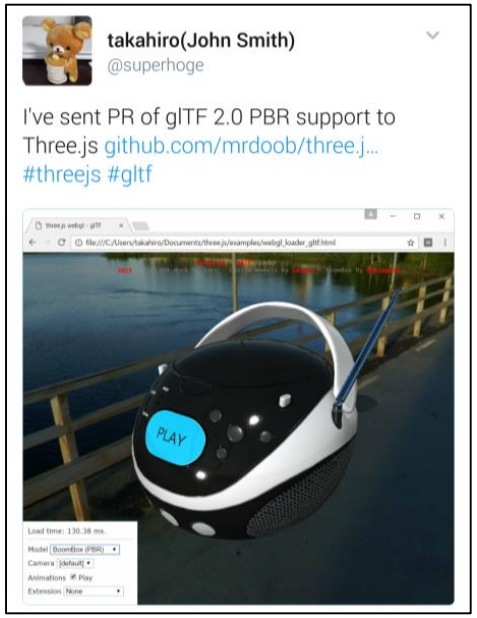
WebGL reference implementation
http://github.khronos.org/glTF-WebGL-PBR/

Laugh Engine running on Vulkan
https://github.com/jian-ru/laugh_engine

Industry Transitioning to glTF 2.0

- **Breaking changes from 1.0 to 2.0 - but processing is streamlined and simplified**
  - NOT significant work and great benefits by upgrading to 2.0
- **Industry moving quickly to glTF 2.0 — lots of early adopters**
  - BabylonJS, three.js, Cesium, xeogl, instant3Dhub
- **gltf-pipeline integrating glTF 2.0 updates — has glTF 1.0 to glTF 2.0 translator**
  - Open source - use this to support both glTF 1.0 and 2.0 or move your users to 2.0
- **Converters/Translators/Validators glTF 2.0 updates nearly ready**
  - COLLADA2GLTF and obj2gltf translators
  - Khronos Validator and Gltf-test
- **Samples and Tutorials**
  - glTF 2.0 sample models with PBR
  - Extensive glTF tutorial series in draft

Move your pipeline to glTF 2.0
Its time!



glTF Roadmap Discussions

- **Incrementally ship new functionality as extensions**
  - For testing out new features, or for long-term optional functionality
  - glTF baseline needs to remain easy to process and deploy
- **Mesh Compression**
  - Google Draco team
- **Progressive Geometry Streaming**
  - Fraunhofer SRC
- **Basis unified compressed texture format for transmission from Binomial**
  - Optimized transmission format with efficient local expansion to any GPU format
- **Enhanced PBR**
  - E.g. NVIDIA MDL
- **Point Clouds**
  - Generated by geometry capture
- **Lighting Extension**
  - Enhanced lighting control
- **Extensions for API and language specifics**
  - Optional hooks for enhanced perf/functionality
  - Vulkan, DX12, Metal, GLSL, HLSL, SPIR-V, Metal C++

glTF 2.0 PBR Rendering - Image courtesy instant3Dhub / instantUV - Max Limper

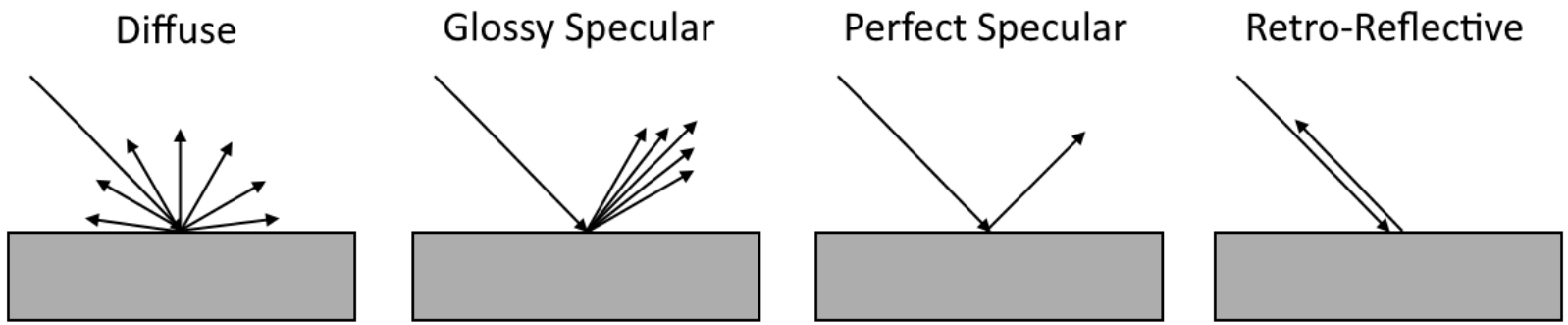
Industry Calls to Action

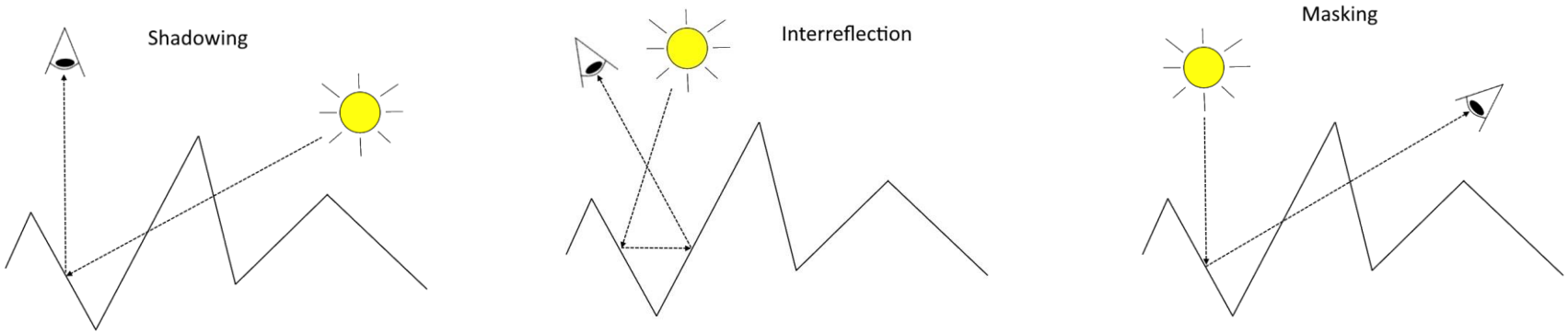
- **Implement glTF 2.0 – specification finalized!**
  - https://github.com/KhronosGroup/glTF/tree/2.0/specification/2.0
- **Primary glTF Online Resources**
  - Github Page https://github.com/KhronosGroup/glTF
  - Resource Hub: https://www.khronos.org/gltf/
- **Share and coordinate your open source glTF projects**
  - https://github.com/KhronosGroup/glTF/issues/867
- **glTF 2.0 Blender Exporter project complete in a few months – input and help welcome!**
  - https://github.com/KhronosGroup/glTF-Blender-Exporter
- **Share your roadmap priorities with us!**
  - https://github.com/KhronosGroup/glTF
- **Join Khronos!**
  - Get directly involved in the glTF Working Group

glTF 2.0 PBR Rendering - Image courtesy Fraunhofer

Introduction to PBR

Intro to PBR





https://github.com/moneimne/glTF-Tutorials/tree/master/PBR

BRDF Lighting Equation

$$f(l,v,h) = \underbrace{Diff(l,n)}_{\text{BRDF Diffuse}} + \underbrace{\frac{F(l,h)\,G(l,v,h)\,D\,(h)}{4(n*l)(n\,*v)}}_{\text{BRDF Specular}}$$

- l is the light direction
- v is the view direction
- h is the half vector
- n is the normal

The metallic-roughness material model is defined by the following properties:

- baseColor - The base color of the material
- metallic - The metalness of the material
- roughness - The roughness of the material

BRDF Diffuse

$$Diff(l,n) = (1 - F(v * h))\frac{C_{diff}}{\pi}$$

Lambertian with energy conservation

C_{diff} is the diffuse reflected color. To conserve energy, the Fresnel term from specular component is subtracted from diffuse component.

```
const dielectricSpecular = rgb(0.04, 0.04, 0.04)
const black = rgb(0, 0, 0)
```

C_{diff} = `lerp(baseColor.rgb * (1 - dielectricSpecular.r), black, metallic)`

BRDF Specular

$$f(l,v,h) = Diff(l,n) + \frac{F(l,h)\ G(l,v,h)\ D\ (h)}{4(n*l)(n\ *v)}$$

BRDF Specular from Cook-Torrance

BRDF Specular : F

$$\frac{\boldsymbol{F(l,h)}\ G(l,v,h)\ D\ (h)}{4(n*l)(n\ *v)}$$

F is the Fresnel function used to simulate the way light interacts with a surface at different viewing angles.

$$F(l,h) = F_0 + (1 - F_0) * (1 - v * h)^5$$

Schlick Fernel model

F_0 is the specular reflectance at normal incidence

```
const dielectricSpecular = rgb(0.04, 0.04, 0.04)
```

F_0 = `lerp(dieletricSpecular, baseColor.rgb, metallic)`

BRDF Specular : G

$$\frac{F(l,h)\,\boldsymbol{G(l,v,h)}\,D\,(h)}{4(n*l)(n\,*v)}$$

G is the geometric occlusion derived from a normal distribution function like Smith's function

$$G(l,v,h) = G_1(n,l)G_1(n,v)$$

$$G_1(n,v) = \frac{2\,(n*v)}{(n*v) + \sqrt{\alpha^2 + (1-\alpha^2)(n*v)^2}}$$

$$\alpha = (roughness)^2$$

BRDF Specular : D

$$\frac{F(l,h)\ G(l,v,h)\ \boldsymbol{D}\ (\boldsymbol{h})}{4(n*l)(n\ *v)}$$

D is the normal distribution function like GGX that defines the statistical distribution of microfacets.

$$\mathrm{D}(h) = \frac{\alpha^2}{\pi\ ((n*h)^2(\alpha-1)+1)^2}$$

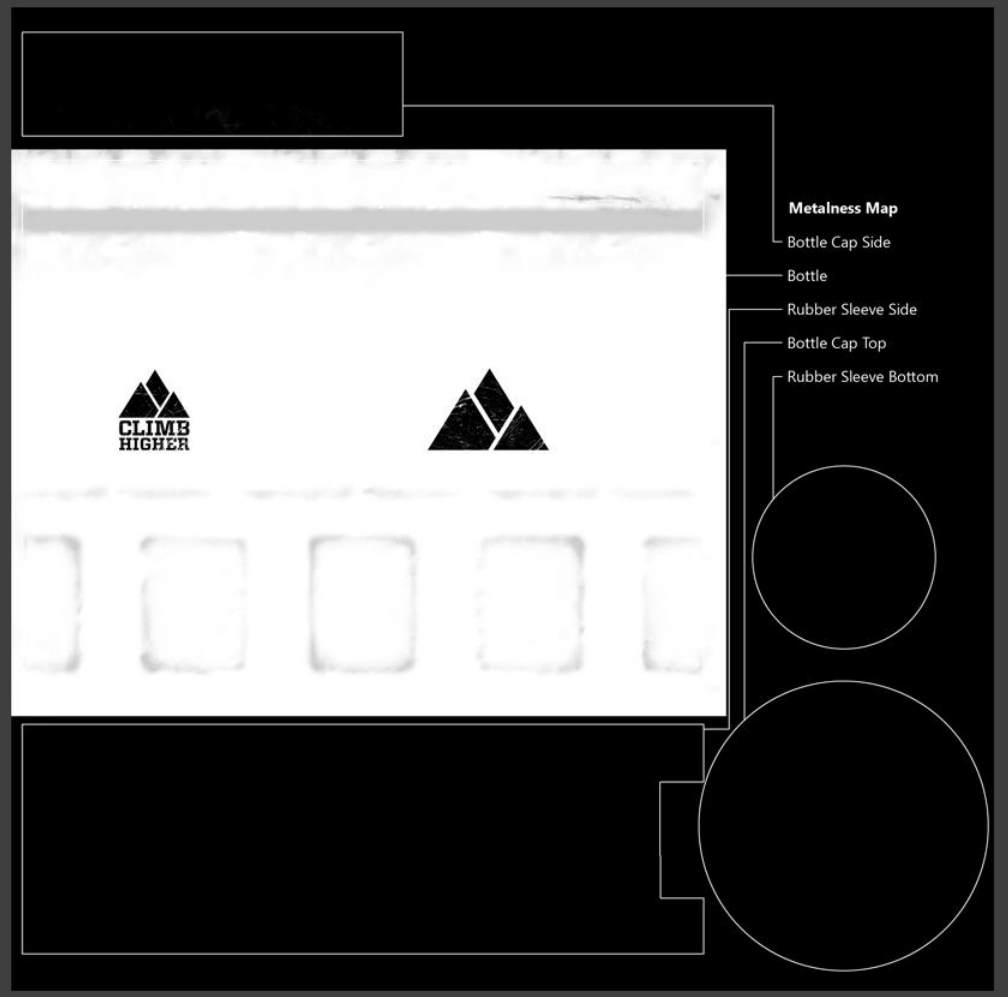
$$\alpha = (roughness)^2$$

Metallic Roughness

PBR Materials

Metallic

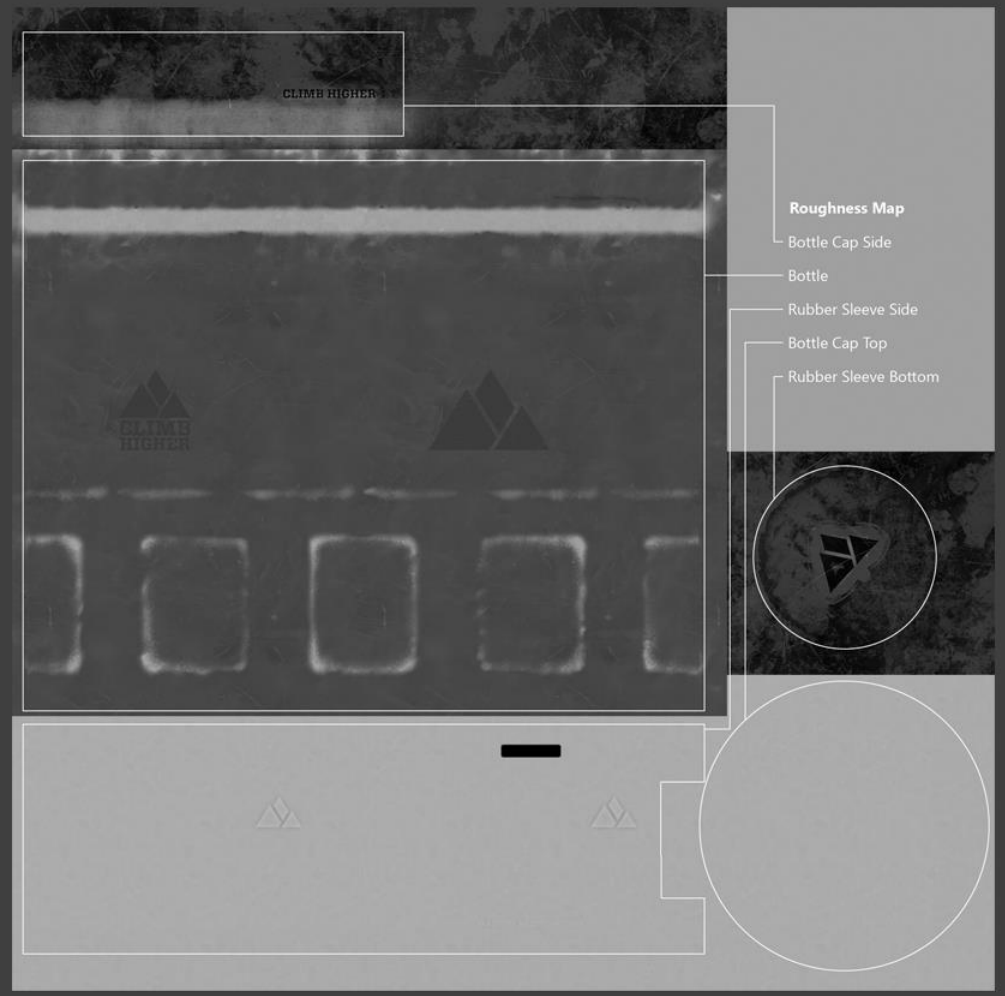
Non-Metallic



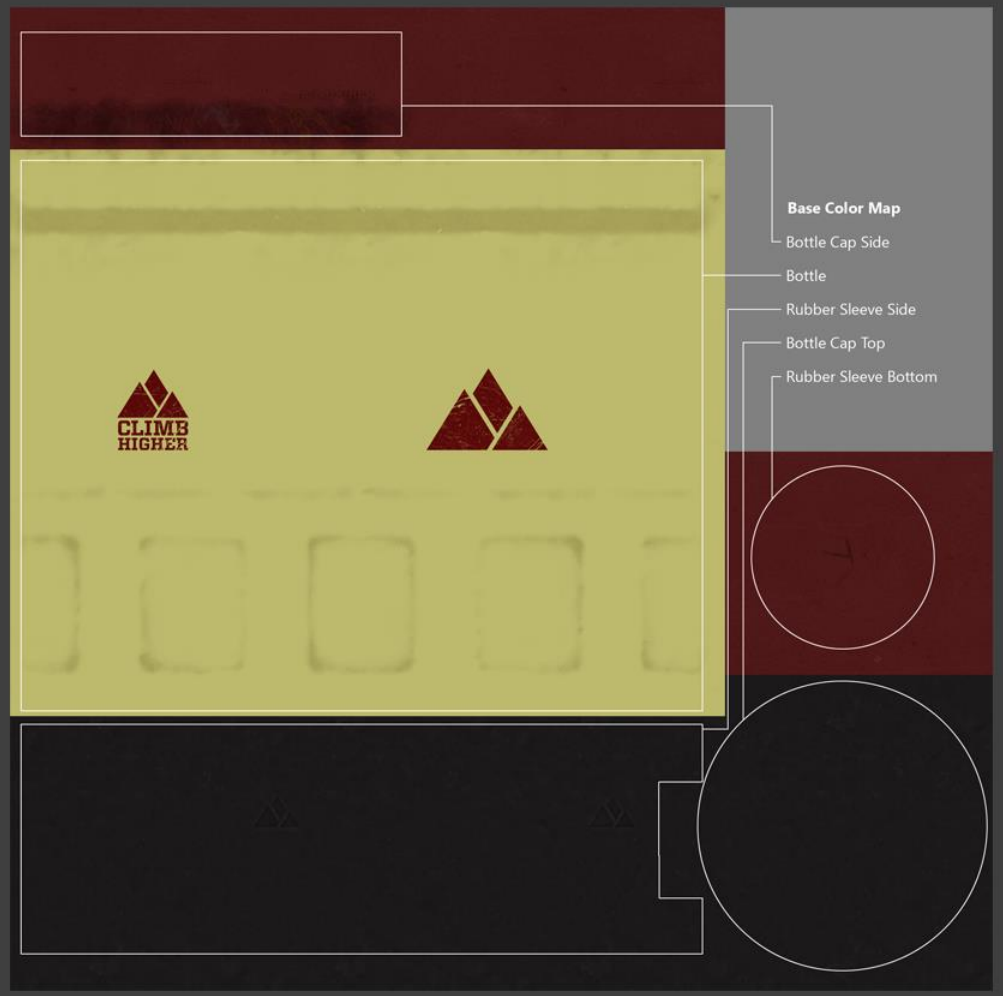
Metalness Map

Rough

Smooth

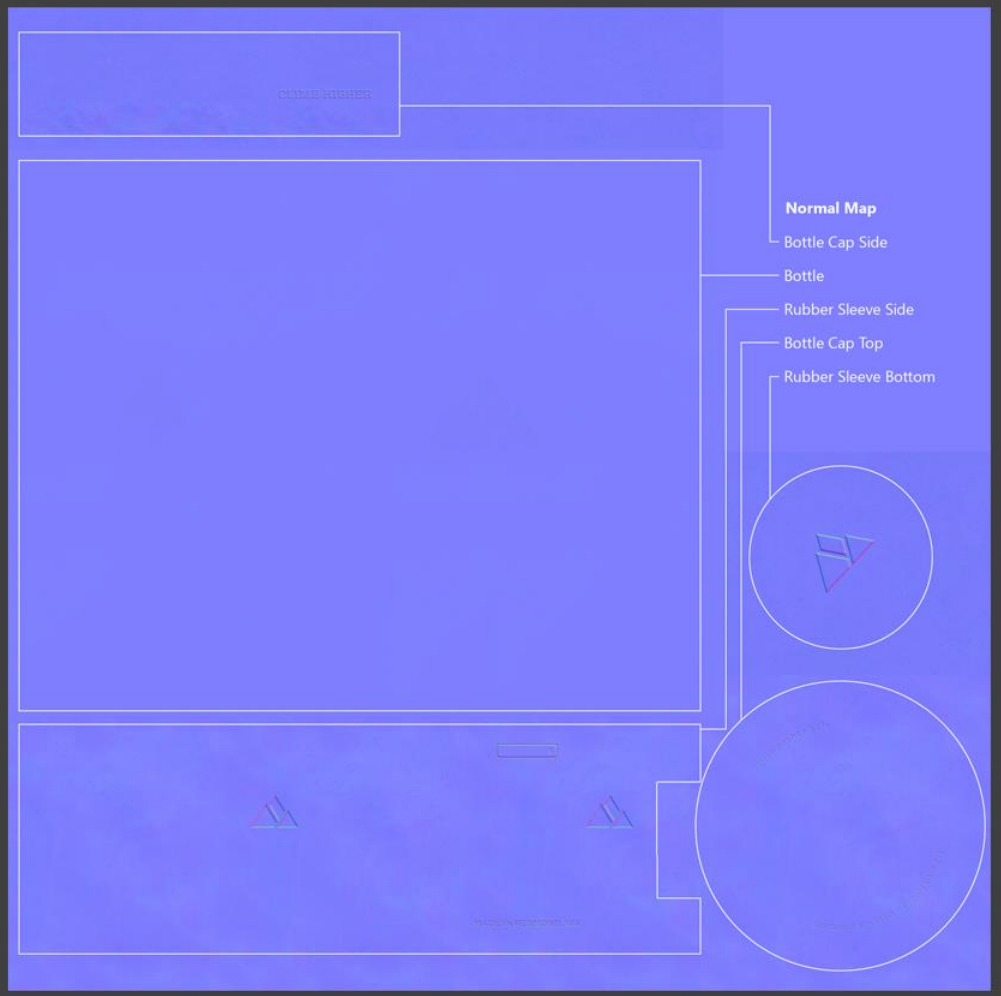


Roughness Map

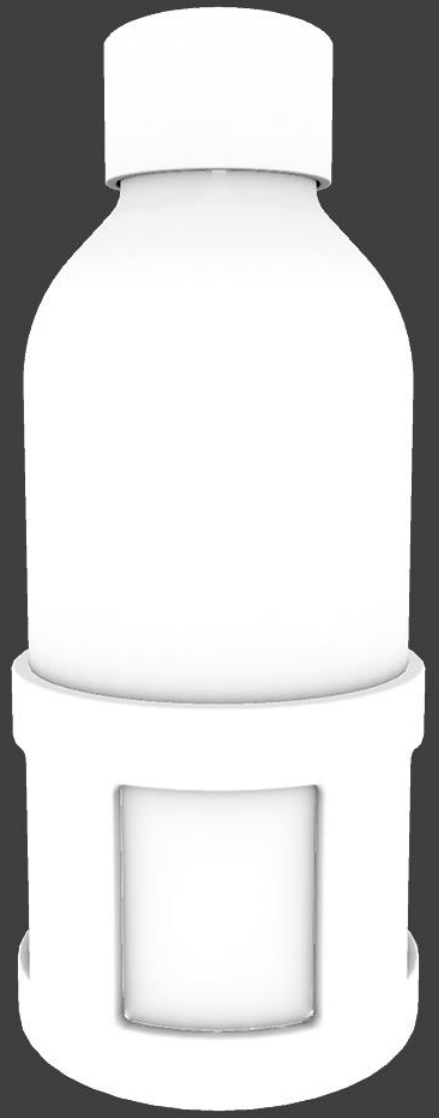


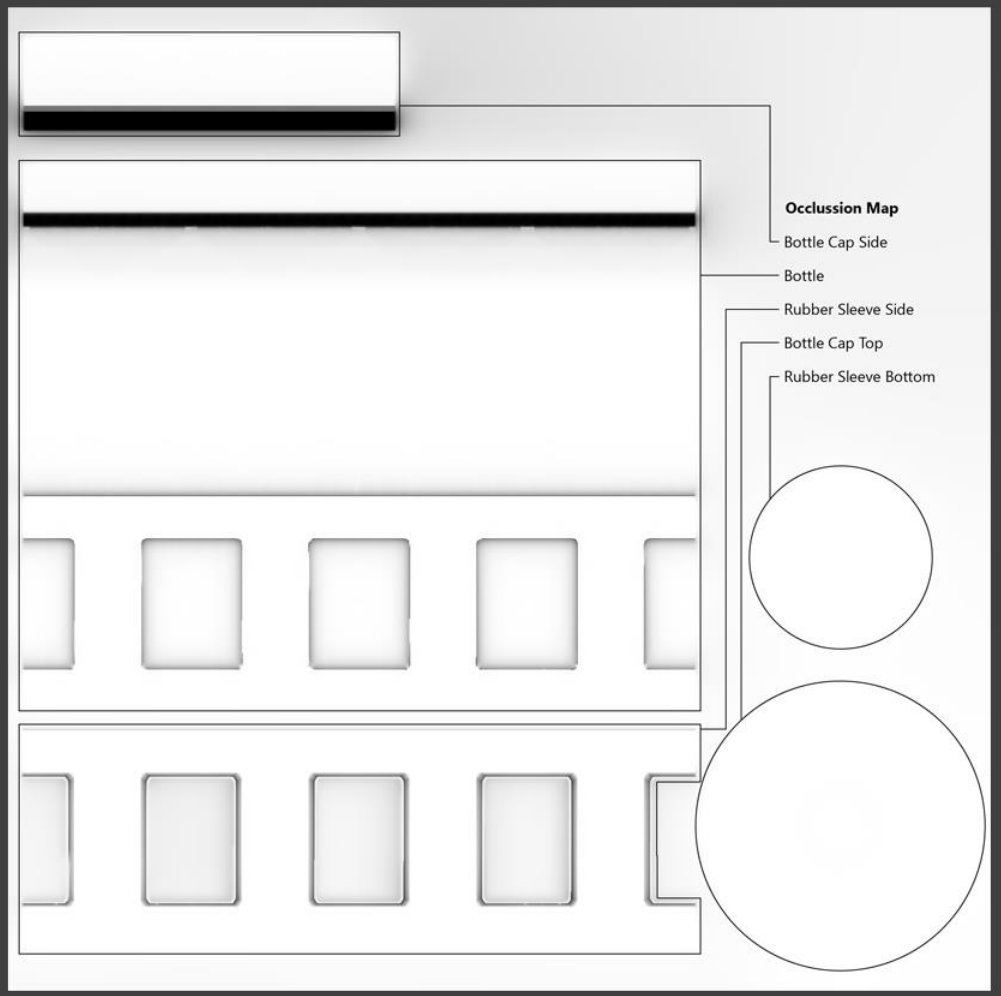
Base Color Map


CLIMB HIGHER
MADE IN REDMOND, WA

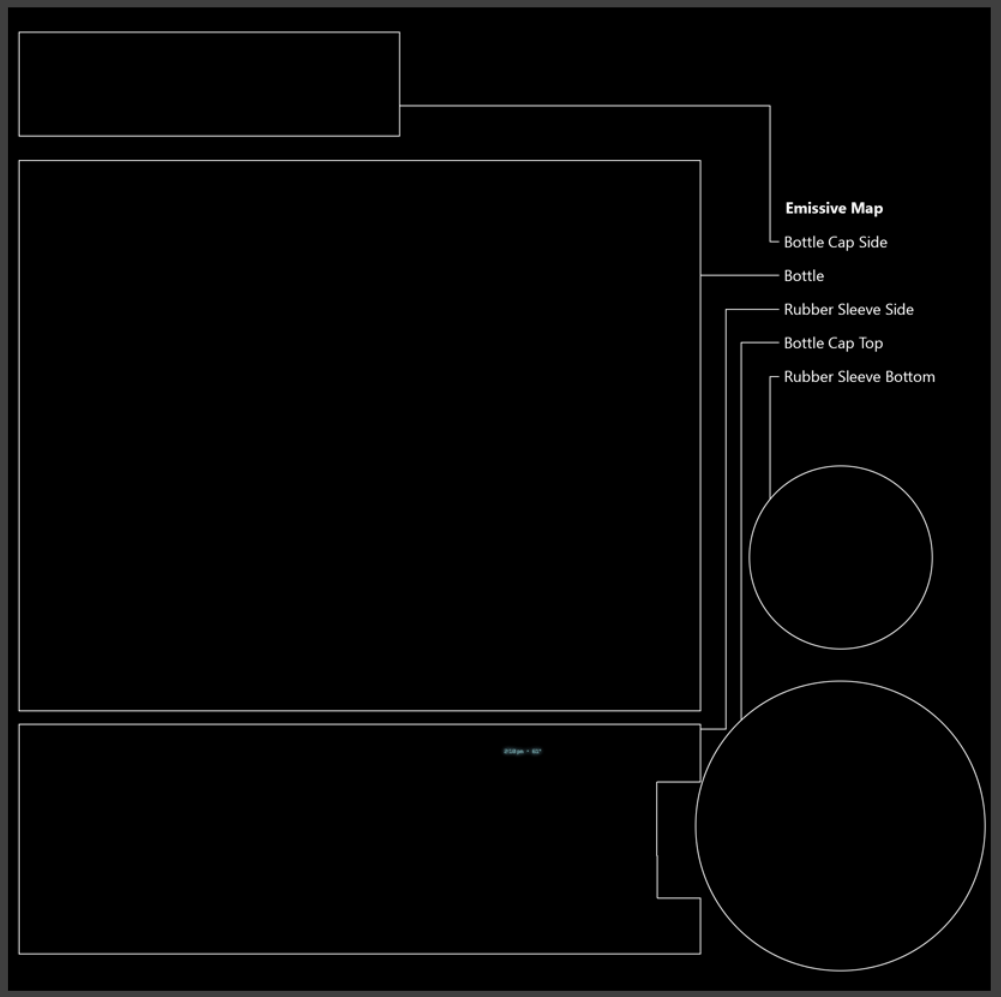

Normal Map
Bottle Cap Side
Bottle
Rubber Sleeve Side
Bottle Cap Top
Rubber Sleeve Bottom

Normal Map



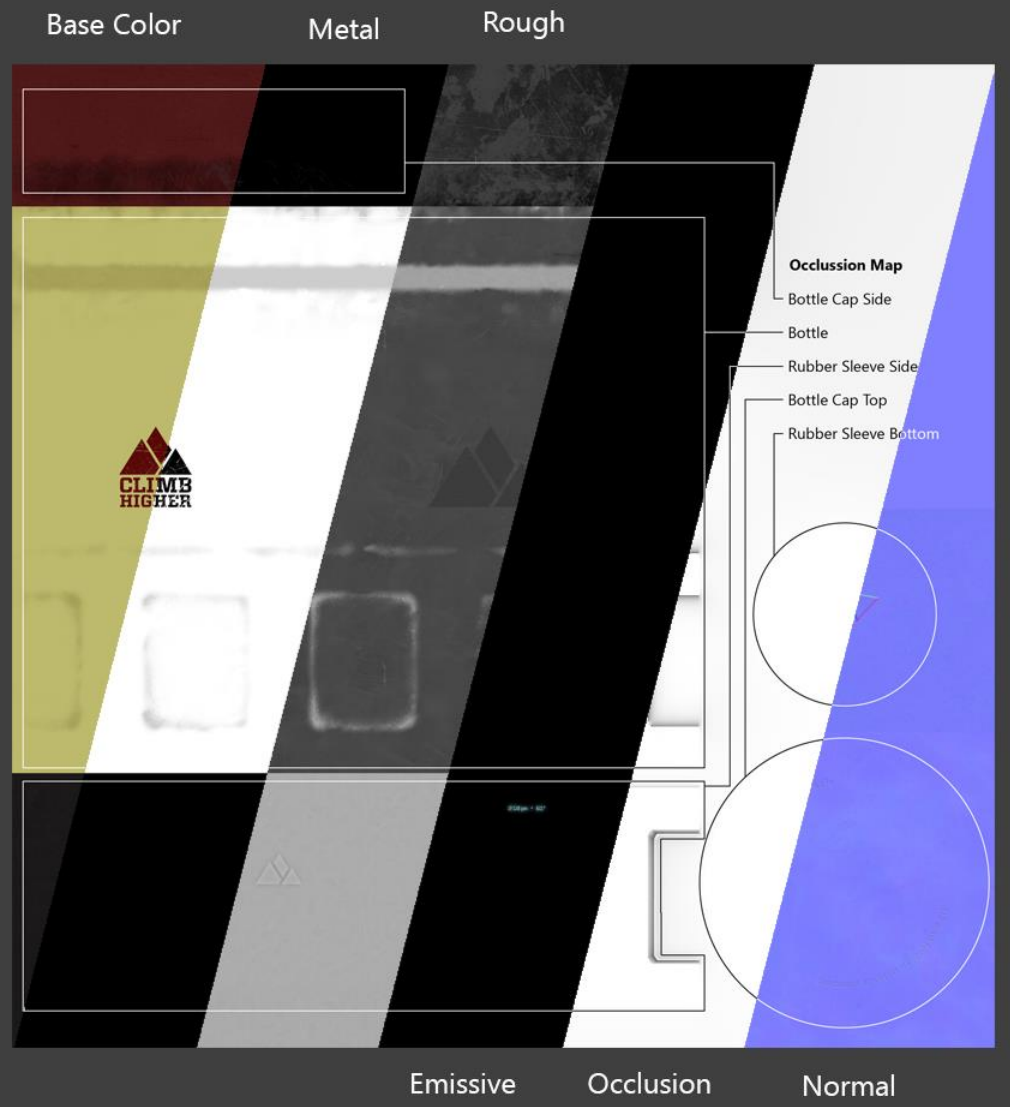


Occlusion Map



Emissive Map


CLIMB HIGHER
2:18 pm • 61°


Base Color
Metal
Rough
Occlussion Map
Bottle Cap Side
Bottle
Rubber Sleeve Side
Bottle Cap Top
Rubber Sleeve Bottom
CLIMB HIGHER
Emissive
Occlusion
Normal

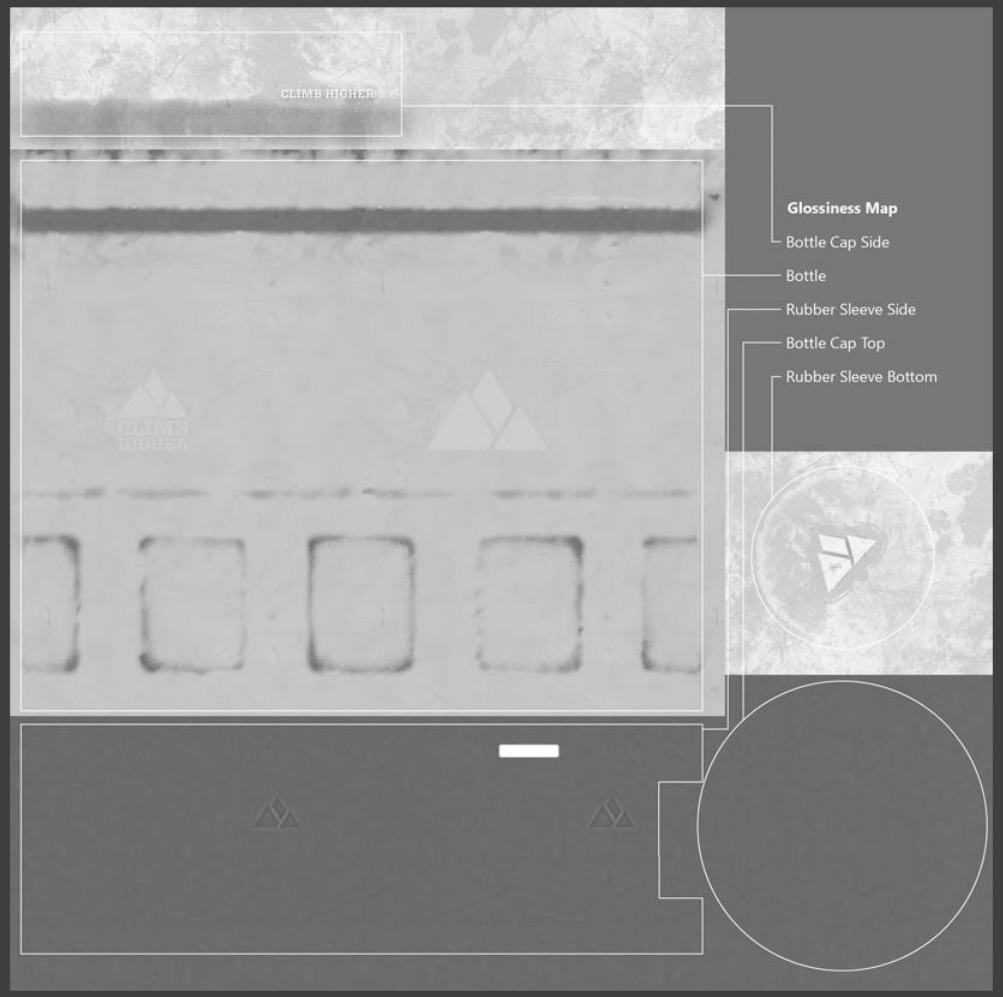

2:18pm • 61°

Specular Glossiness

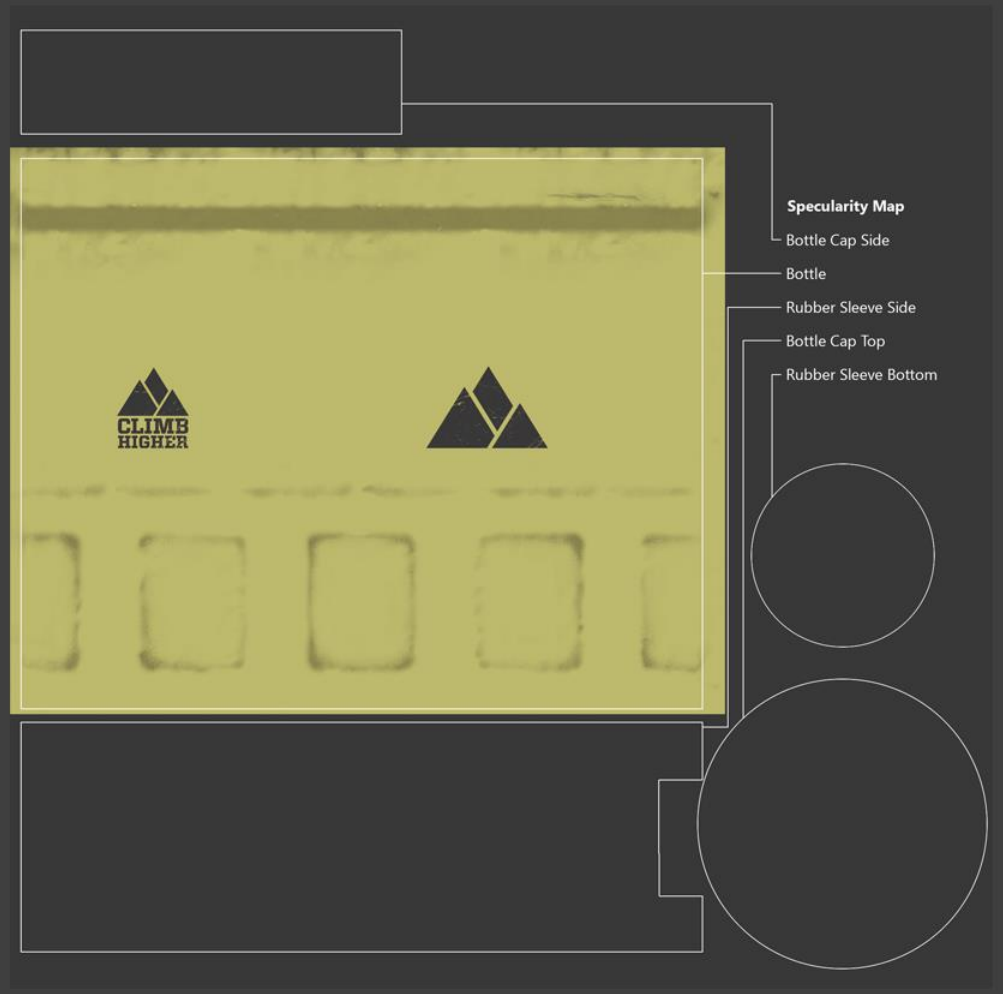
PBR Materials

Glossy

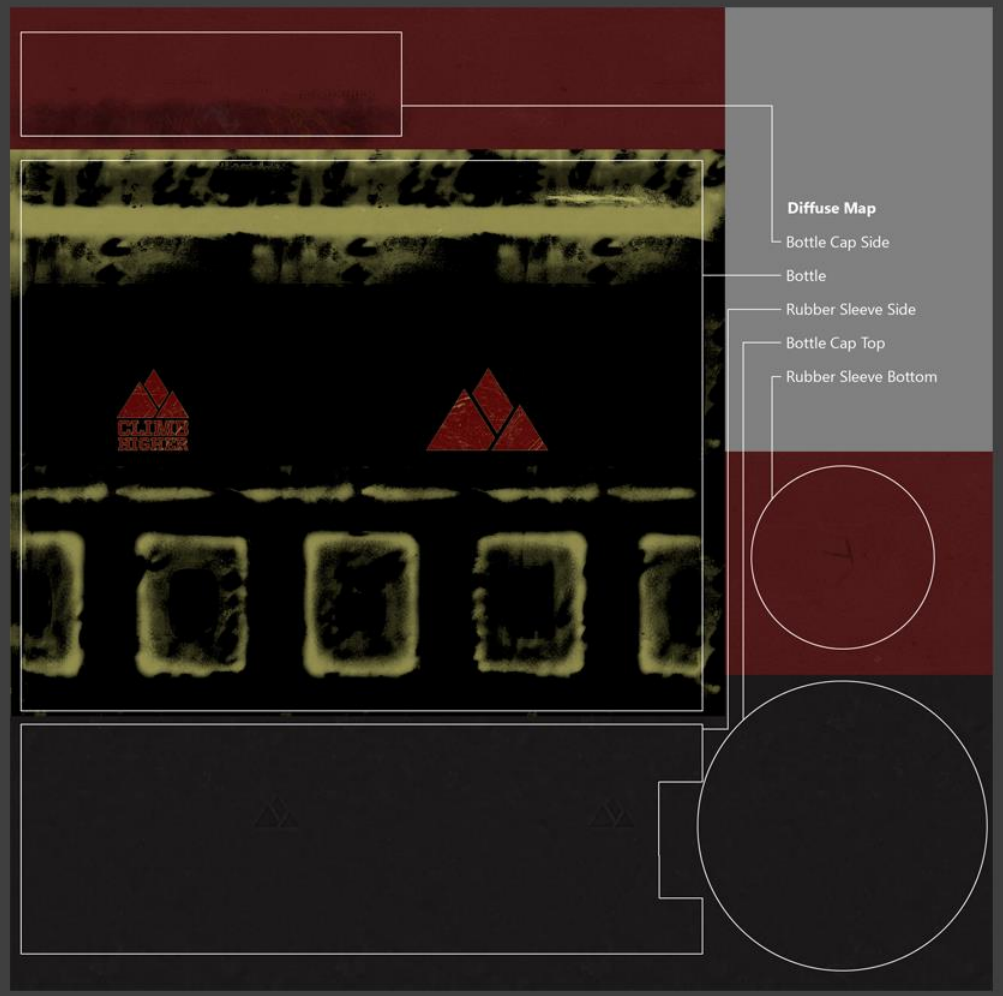
Non-Glossy



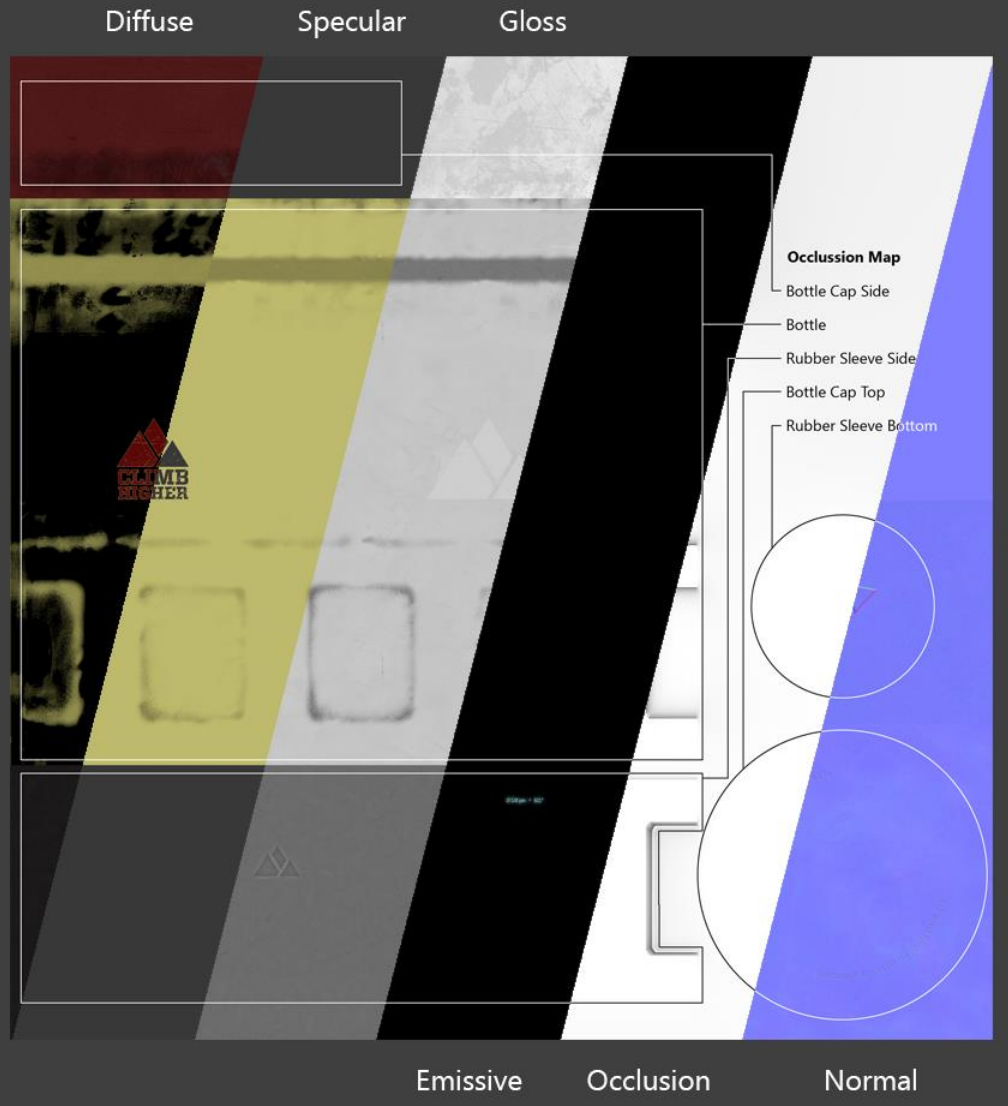
Glossiness Map



Specularity Map


Diffuse Map
Bottle Cap Side
Bottle
Rubber Sleeve Side
Bottle Cap Top
Rubber Sleeve Bottom
CLIMB
HIGHER

Diffuse Map


Diffuse
Specular
Gloss
Occlussion Map
Bottle Cap Side
Bottle
Rubber Sleeve Side
Bottle Cap Top
Rubber Sleeve Bottom
Emissive
Occlusion
Normal

PBR Resources

- **Demos**
  - WebGL-PBR implementation
    - http://github.khronos.org/glTF-WebGL-PBR/
  - glTF 2.0 Sample Models
    - https://github.com/KhronosGroup/glTF-Sample-Models
- **Articles**
  - glTF PBR Tutorial:
    - https://github.com/moneimne/glTF-Tutorials/tree/master/PBR
  - Substance PBR-guide:
    - https://www.allegorithmic.com/pbr-guide
  - Moving Frostbite to PBR:
    - http://www.frostbite.com/2014/11/moving-frostbite-to-pbr/
  - Good example values:
    - https://seblagarde.wordpress.com/2014/04/14/dontnod-physically-based-rendering-chart-for-unreal-engine-4/