Week 9

PCG and Models

Part 1 - PCG

- In computing, **procedural generation** (sometimes shortened as **proc-gen**) is a method of creating data algorithmically as opposed to manually, typically through a combination of human-generated content and algorithms coupled with computer-generated randomness and processing power.
- In computer graphics, it is commonly used to create textures and 3D models.
- In video games, it is used to automatically create large amounts of content in a game. Depending on the implementation, advantages of procedural generation can include smaller file sizes, larger amounts of content, and randomness for less predictable gameplay.
- Procedural generation is a branch of media synthesis.

https://en.wikipedia.org/wiki/Procedural_generation

Procedural Content Generation (PCG)

- Algorithmic Generation of Content

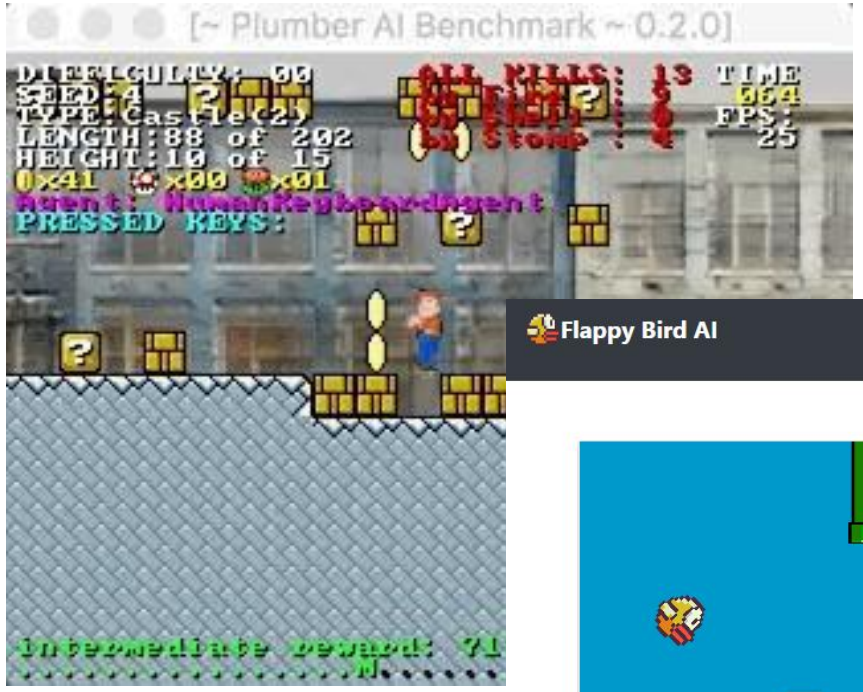
Program / Software

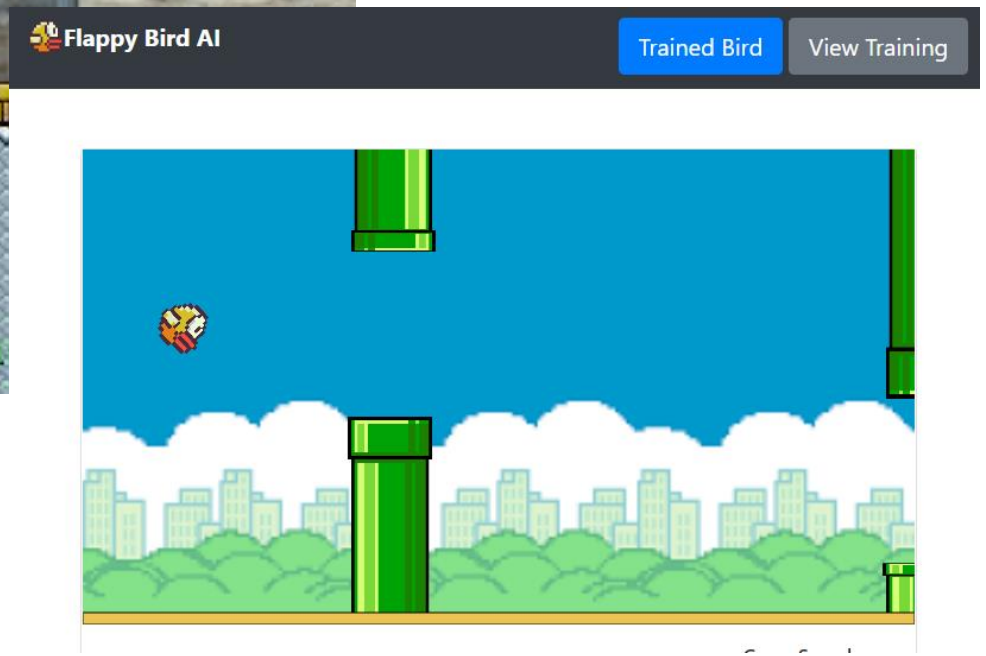
Created online/offline

Content:
Music, Audio, Visuals, Game-Objects, Texts/Story

Procedural Content Generation (PCG)

- Algorithmic Generation of Content
  - Games





https://flappybird-ai.netlify.app/

https://code.google.com/archive/p/marioai/

[redacted]

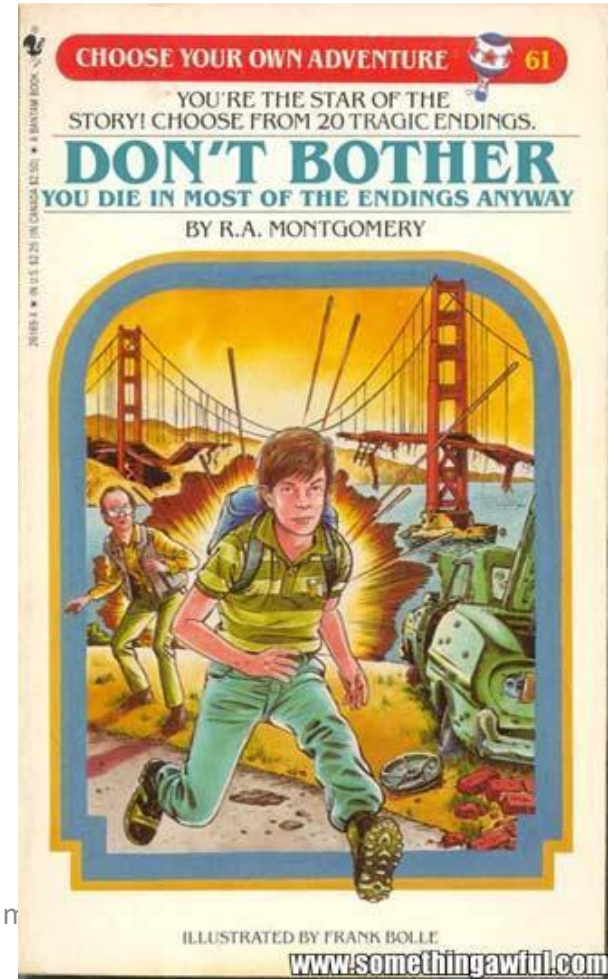
Procedural Content Generation (PCG)

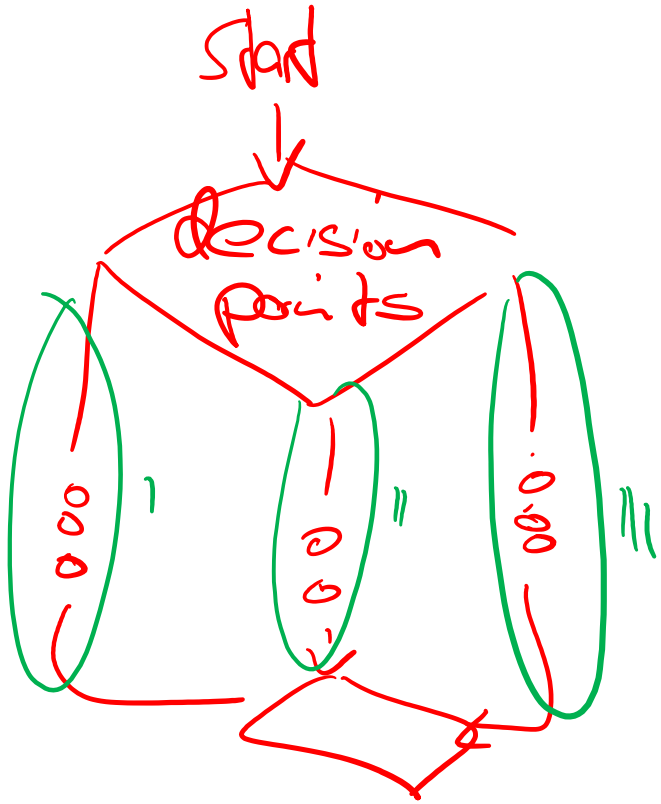
- Algorithmic Generation of Content
  - Games
  - Music
  - Books/Stories

https://www.youtube.com/watch?v=0dOn-EvSPUs

Procedural Content Generation (PCG)

- Algorithmic Generation of Content
  - Games
  - Music
  - Books/Stories





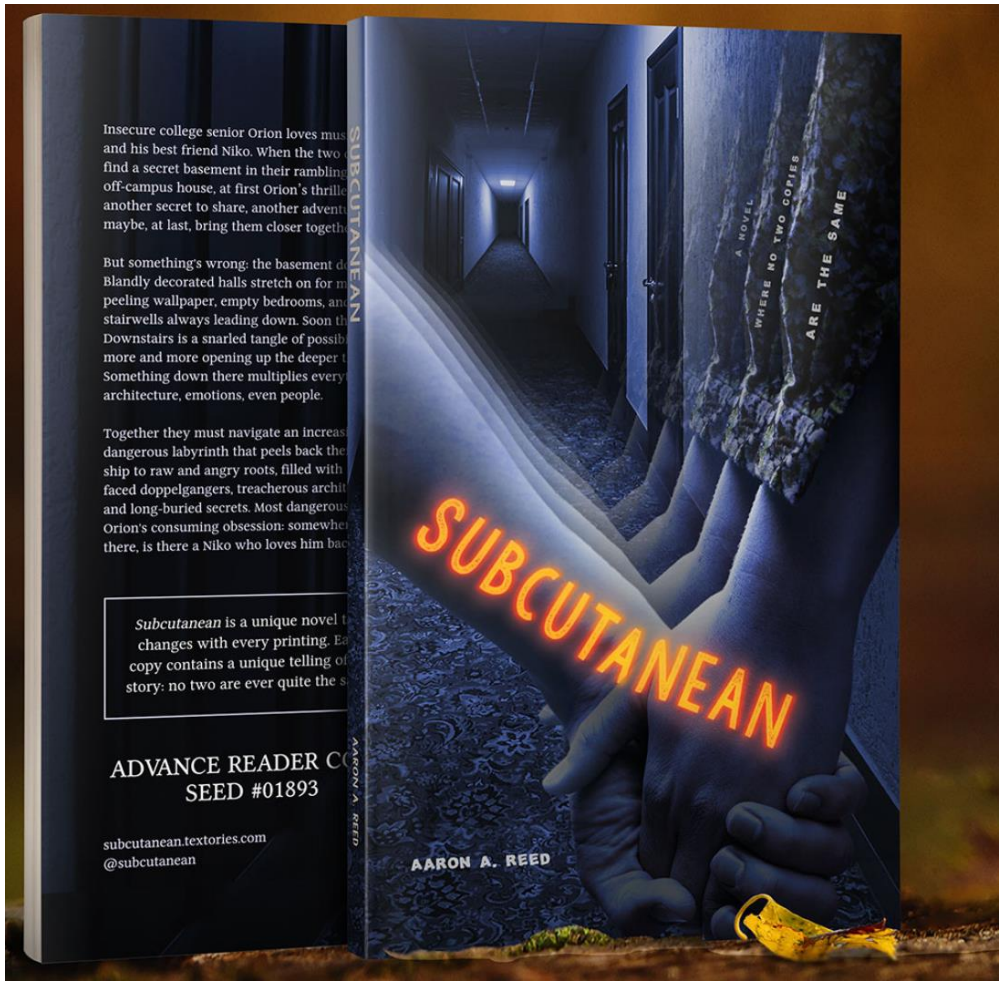
Procedural Content Generation (PCG)

- Algorithmic Generation of Content
  - Games
  - Music
  - Books/Stories

http://ice-bound.com/

Procedural Content Generation (PCG)

- Algorithmic Generation of Content
  - Games
  - Music
  - Books/Stories



Subcutanean

Procedural Content Generation (PCG)

- Algorithmic Generation of Content
  - Games
  - Music
  - Books/Stories
- Creation/Modification of Game Assets
  - Unity/OpenGL
  - Instancing (covered in next optimization lecture)

Procedural Content Generation (PCG)

- C++ Libraries
  - Noise (https://github.com/Auburn/FastNoise, http://libnoise.sourceforge.net/)
  - Genetic Operations (https://www.geeksforgeeks.org/genetic-algorithms/)
  - Logic Rules (http://clipsmm.sourceforge.net/)

- Inspirations:
  - https://github.com/JamesRandall/SimpleVoxelEngine

Procedural Content Generation (PCG)

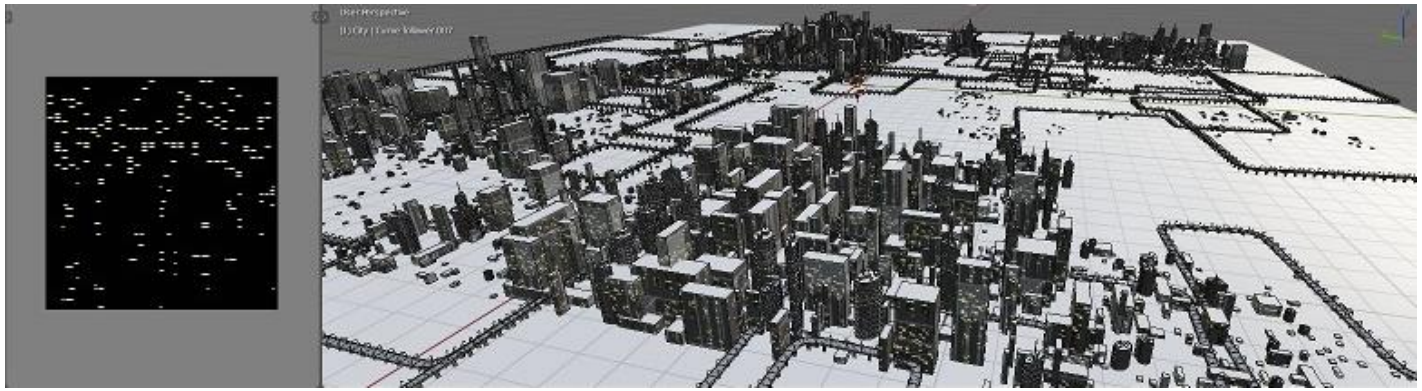
Perlin Noise

More details on Perlin algo: https://www.youtube.com/watch?v=kCIaHqb60Cw&ab_channel=Zipped

Other PCG techniques

- How to approach PCG:
  - have an idea that requires similar but continuous content
    - world generation
    - level generation
    - story generation
  - start with a small set of building blocks

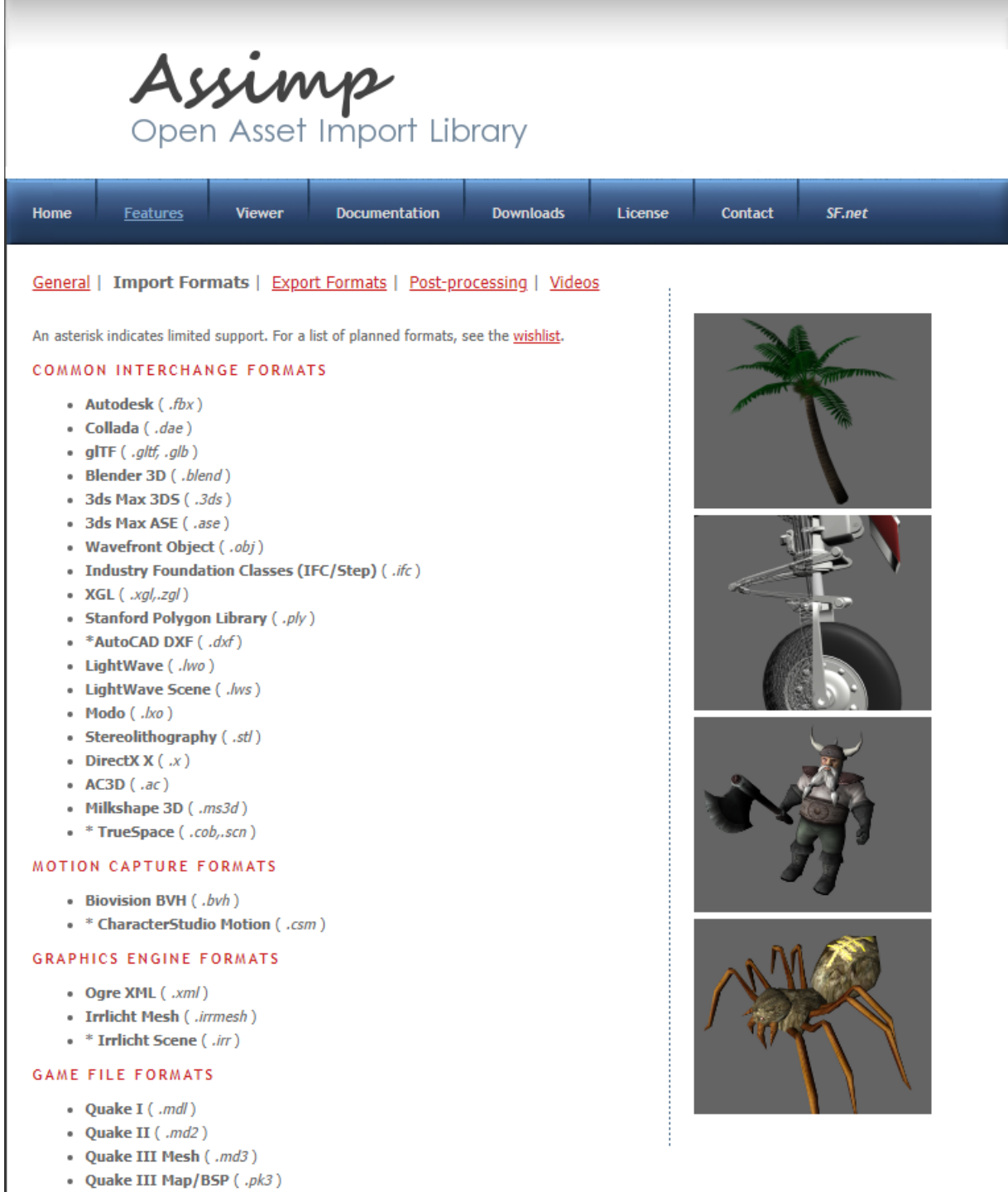
- design a process (steps) on paper
- code basic version creating minimal proof of concept
  - use:
    - evolution approaches
    - search approaches
    - random



Part 2 - Models

- ASSIMP library

- Open Asset Import Library is a library to load various 3d file formats into a shared, in-memory format. It supports more than **40 file formats** for import and a growing selection of file formats for export.

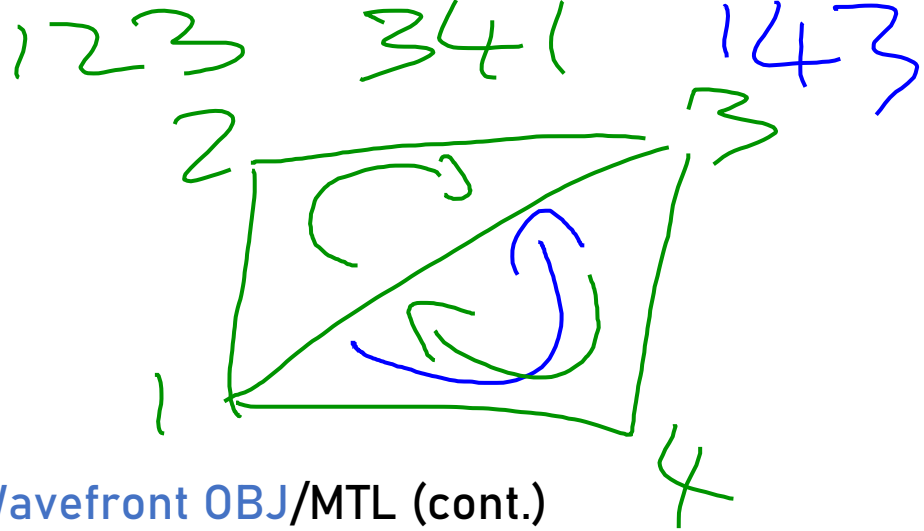


FileFormats

- Polygon PLY
- Wavefront OBJ/MTL
- Collada DAE
- Autodesk FBX (proprietary)
- Autodesk 3DS (proprietary)
- Khronos glTF/glB

- Further Reading (https://all3dp.com/2/most-common-3d-file-formats-model/)

FileFormats *(Example)*



- Wavefront OBJ/MTL (link)

vertex

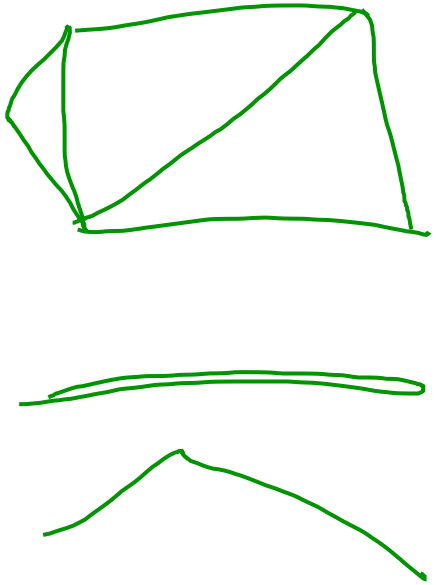
```
mtllib creeper.mtl
o Box_Mesh_345357280
v -0.500000 -0.500000 -0.500000
v -0.500000 0.500000 0.500000
...
vt 0.593540 -0.000303
vt 0.593540 0.249793
...
vn -0.0000 -1.0000 -0.0000
vn 0.0000 1.0000 0.0000
...
```

normal

texture

- Wavefront OBJ/MTL (cont.)

```
g Box_Mesh_345357280_Material.001
usemtl Material.001
s 1
f 3/1/1 7/2/1 8/3/1 1/4/1
f 5/5/2 2/6/2 6/7/2 4/8/2
f 3/9/3 1/10/3 5/11/3 4/12/3
f 2/13/4 5/14/4 1/15/4 8/16/4
f 7/17/5 3/18/5 4/19/5 6/20/5
f 8/21/6 7/22/6 6/23/6 2/24/6
```



FileFormats

- Wavefront OBJ/MTL (link)

newmtl Material.004

Ns 4.499998

Ka 1.000000 1.000000 1.000000

Kd 1.000000 1.000000 1.000000

Ks 2.000000 2.000000 2.000000

Ke 0.0 0.0 0.0

Ni 1.450000d 1.000000

illum 2

map_Kd Texture.png

map_d Texture.png

- Wavefront OBJ/MTL (std)

newmtl my_red

Material color & illumination statements

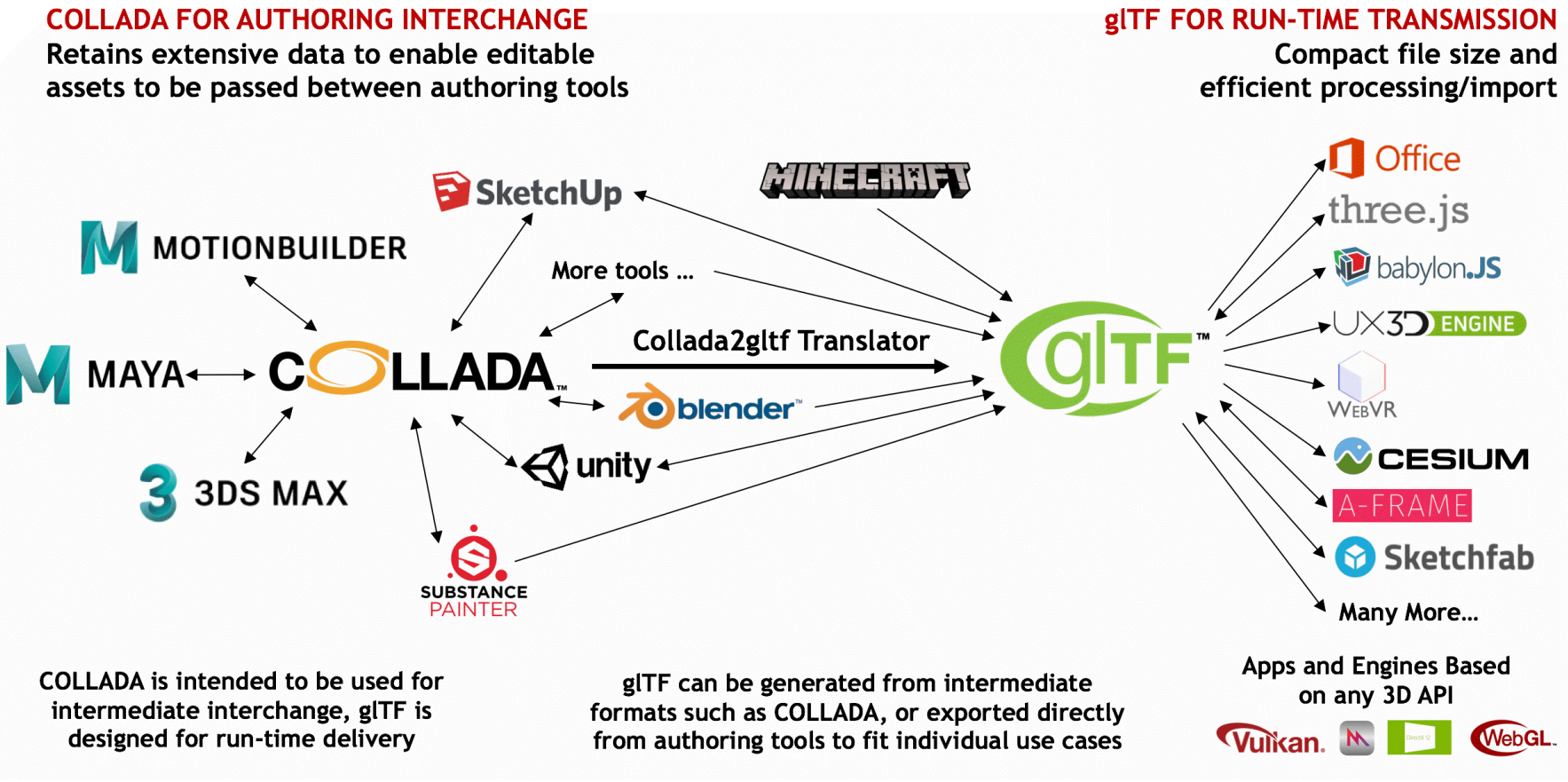
- Ka (ambient reflectivity) r g b
- Kd (diffuse reflectivity) r g b
- Ks (specular reflectivity) r g b
- Ke (emmisive reflectivity) r g b
- Ni (optical density) [0:100]
- d (dissolve)

texture map statements

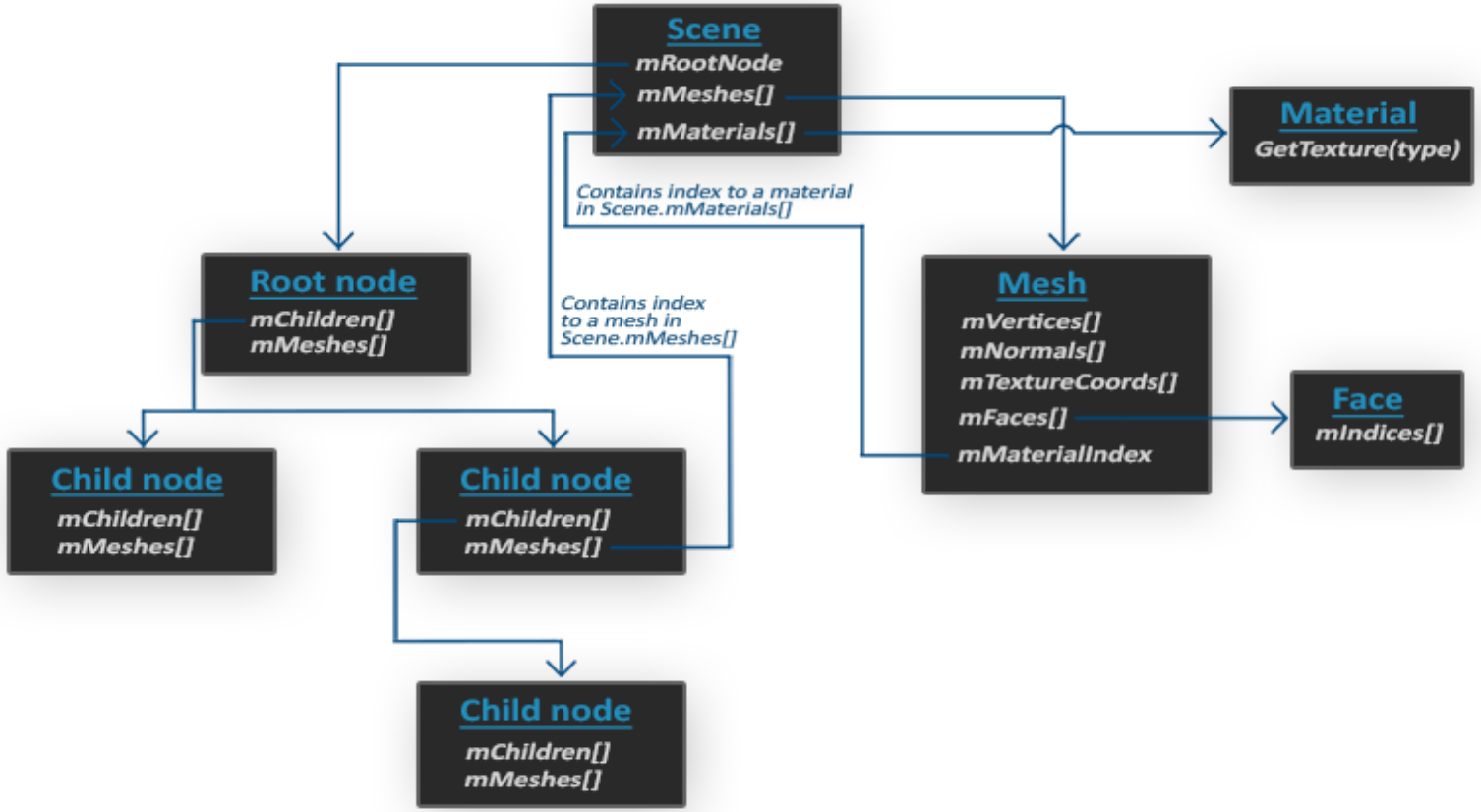
reflection map statement

FileFormats

- Collada DAE



ASSIMP


Scene
mRootNode
mMeshes[]
mMaterials[]
Material
GetTexture(type)
Contains index to a material in Scene.mMaterials[]
Root node
mChildren[]
mMeshes[]
Contains index to a mesh in Scene.mMeshes[]
Mesh
mVertices[]
mNormals[]
mTextureCoords[]
mFaces[]
mMaterialIndex
Face
mIndices[]
Child node
mChildren[]
mMeshes[]
Child node
mChildren[]
mMeshes[]
Child node
mChildren[]
mMeshes[]

Meshes

- With Assimp we can load many different models into the application, but once loaded they're all stored in Assimp's data structures.
- What we eventually want is to transform that data to a format that OpenGL understands so that we can render the objects.
- https://learnopengl.com/code_viewer_gh.php?code=includes/learnopengl/mesh.h
- The Mesh class just defined is an abstraction for the early chapter topics.
- For more details: https://learnopengl.com/Model-Loading/Mesh

Model Loading

- start creating the actual loading and translation code.
- The goal is to create another class that represents a model in its entirety, that is, a model that contains multiple meshes, possibly with multiple textures.
- The Model class contains a vector of Mesh objects and requires us to give it a file location in its constructor.
- It then loads the file right away via the loadModel function that is called in the constructor.
- The private functions are all designed to process a part of Assimp's import routine.
- We also store the directory of the file path that we'll later need when loading textures.
- More details: https://learnopengl.com/code_viewer_gh.php?code=includes/learnopengl/model.h
- Using Assimp you can load tons of models found over the internet.
- There are quite a few resource websites that offer free 3D models for you to download in several file formats like Turbosquid and Sketchfab.
- Do note that some models still won't load properly, have texture paths that won't work, or are simply exported in a format even Assimp can't read.

Model Loading – Another Solution

Video uploaded separately on DLE: https://www.youtube.com/watch?v=sP_kiODC25Q&t=425s&ab_channel=OGLDEV