Mestrado em
Engenharia Informática

VI-RT Scene Loading

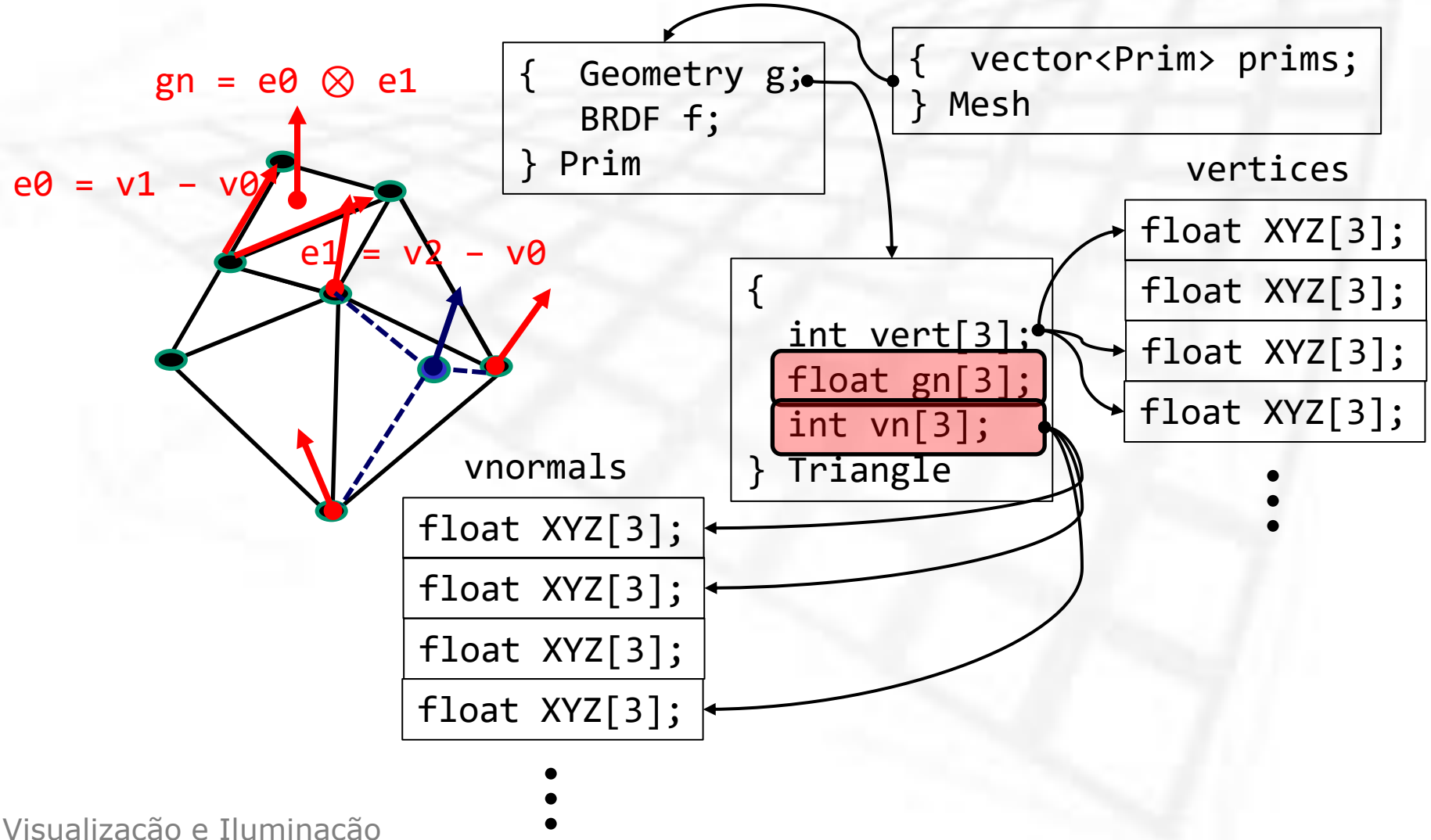
Visualização e Iluminação

Luís Paulo Peixoto dos Santos

SCENE LOADING

[Vecteezy.com]

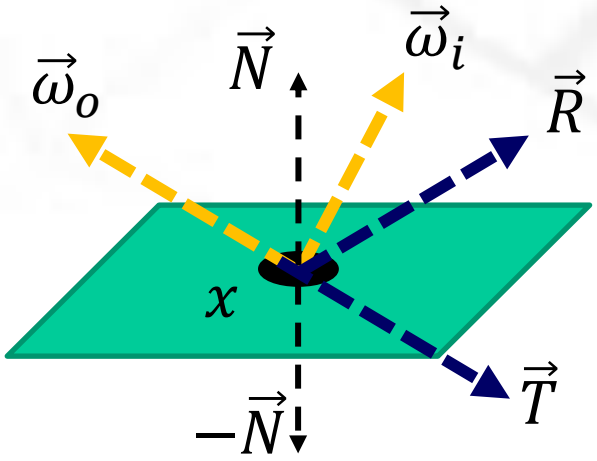
Mesh Representation



X-Phong Representation

$$L(\vec{\omega}_o \leftarrow x \leftarrow \vec{\omega}_i) = k_a * L_a +$$

$$+ L(x \leftarrow \vec{\omega}_i) * (k_d * (\vec{\omega}_i \cdot \vec{N}) + k_s * (\vec{\omega}_i \cdot \vec{R})^{N_s} + k_t * (\vec{\omega}_i \cdot \vec{T})^{N_s})$$




```
class Phong:: public BRDF {
    RGB Ka;
    RGB Kd;
    RGB Ks;
    float Ns;
    RGB Kt;
}
```

Wavefront .obj file

```
# List of geometric vertices, with x, y, z coordinates
v 0.123 0.234 0.345
v ... ...
# List of texture coordinates:(u, [v, w]) in [0 ... 1]
vt 0.500 [1 [0]]
vt ... ... #
List of vertex normals in (x,y,z) form
vn 0.707 0.000 0.707
vn ... ...
```

https://en.wikipedia.org/wiki/Wavefront_.obj_file

Wavefront .obj file

```
# which materials definition file to use
mtllib [external .mtl file name]
# which material to use in the subsequent objects
usemtl [material name]
# group subsequent faces onto an object
o [object name]
# Polygonal face element: f v1/vt1/vn1 v2/vt2/vn2 v3/vt3/vn3 ...
# arguments are the vertices' indices according to their order in the file
f 1 2 3
f 3/1 4/2 5/3
f 6/4/1 3/5/3 7/6/5
f 6//3 2//1 7//2
```

https://en.wikipedia.org/wiki/Wavefront_.obj_file

Wavefront .mtl file

```
newmtl my_mtl
       Ka 0.0435 0.0435 0.0435
       Kd 0.1086 0.1086 0.1086
       Ks 0.0000 0.0000 0.0000
       Tf 0.9885 0.9885 0.9885
       Ns 10.0000
```

http://paulbourke.net/dataformats/mtl/

TINY OBJ LOADER

```
#define TINYOBJLOADER_IMPLEMENTATION
#include "tiny_obj_loader.h"
using namespace tinyobj;

ObjReader myObj;
// this loader triangulates the faces
if (!myObj.ParseFromFile(fname)) return false;

// materials
const vector<material_t> materials = myObj.GetMaterials();
```

https://github.com/tinyobjloader/tinyobjloader

TINY OBJ LOADER

```
// ... continued

// access the vertices
const tinyobj::attrib_t attrib = myObj.GetAttrib();
float *vertices = attrib.vertices;

// access the shapes
const vector<shape_t> shps = myObj.GetShapes();
```

https://github.com/tinyobjloader/tinyobjloader

TINY OBJ LOADER

```
// iterate over shapes
for (auto shp = shps.begin() ; shp != shps.end() ; shp++) {
 // iterate over this shape's vertices
 auto indices = shp->mesh.indices;
 for (auto vertex = indices.begin() ; vertex != indices.end() ; ) {
   // each 3 consecutives vertices form a face (triangle)
   XYZ myVertex[3];
   for (int v = 0 ; v<3 ; v++ , vertex++) {
     // get each vertex XYZ
     const int objNdx = vertex->vertex_index;
     myVertex[v].X = vertices[objNdx*3];
     myVertex[v].Y = vertices[objNdx*3+1];
     myVertex[v].Z = vertices[objNdx*3+2];
} } }
```

https://github.com/tinyobjloader/tinyobjloader