VR Graphics Programming for, well, you know

Some (relatively) easy steps to virtual reality

Tom Sgouros

Center for Computation and Visualization
Brown University
thomas_sgouros@brown.edu

Spring 2018

Basic 3D graphics

Coordinates Where things are in space.

Transformation matrices How one space relates to another.

Objects Shape, color, location.

Scene graph How one object relates to another.

Illumination Lights and lighting.

Textures Surface treatments.

Coordinates

- Four dimensions, “Homogeneous coordinates” $[x, y, z, w]$
- Vertex locations, normal directions.
- Also color (also 4D $[r, g, b, a]$), texture $[u.v]$, other?

Spaces and transformation matrices

- Move from one space to another.

- Model space – where an object is defined.

  $\Updownarrow$ model matrix
- World space – where an object is placed.

  $\Updownarrow$ view matrix
- Camera space – where an object appears to be placed (POV).

  $\Updownarrow$ projection matrix
- Screen – where it appears on the screen

Transformation matrices

$$\begin{bmatrix} a & b & c & j \\ d & e & f & k \\ g & h & i & l \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Translation Rotation Scale

$\begin{bmatrix} x & y & z & 1 \end{bmatrix}$ Point

$\begin{bmatrix} x & y & z & 0 \end{bmatrix}$ Direction

drawableObjData

A way to link some data, a name string, and two IDs (that OpenGL calls names).

```
template <class T>
class drawableObjData {
 private:
  std::vector<T> _data;

 public:
 drawableObjData(): name("") {
    _data.reserve(50); ID = 0; bufferID = 0;
  };
 drawableObjData(const std::string inName,
                 const std::vector<T> inData) :
  name(inName), _data(inData) {}
```

drawableObj

The data you need to draw an object.

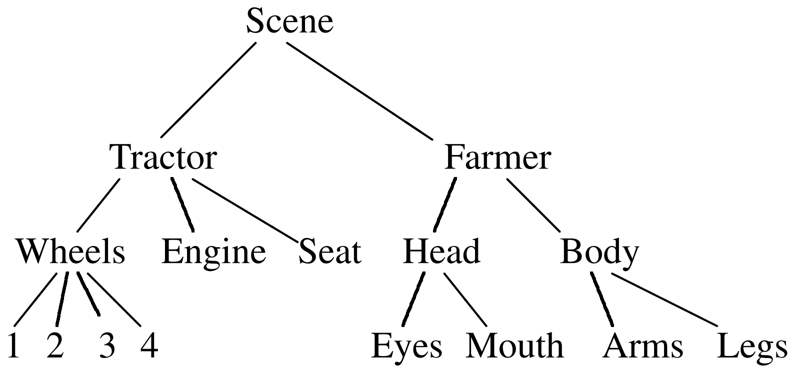
```
class drawableObj {
 protected:

  // Specifies whether this is a triangle, a triangle strip,
  // fan, quads, points, whatever.
  GLenum _drawType;
  GLsizei _count;

  drawableObjData<glm::vec4> _vertices;
  drawableObjData<glm::vec4> _colors;
  drawableObjData<glm::vec4> _normals;
  drawableObjData<glm::vec2> _uvs;
```

Scene graph

A way to group a collection of objects to be drawn into small groups that can be manipulated as a single object. Creates a hierarchy of model matrices with object models placed on the tree.



drawableMulti

A way to manage a position and location, and the model matrix system for a scene graph.

```
class drawableMulti {
 protected:

  drawableMulti* _parent;

  std::string _name;

  glm::vec3 _position;
  glm::vec3 _scale;
  glm::quat _orientation;

  glm::mat4 _modelMatrix;
```

drawableCompound

An object made up of pieces that you don't expect to move relative to one another.

```
class drawableCompound : public drawableMulti {
 protected:
  typedef std::list<bsgPtr<drawableObj> > DrawableObjList;
  DrawableObjList _objects;

  bsgPtr<shaderMgr> _pShader;

  glm::mat4 _totalModelMatrix;

  glm::mat4 _normalMatrix;

  std::string _modelMatrixName;
  GLuint _modelMatrixID;
```

bsgMenagerie

A fairly random collection of drawableCompound objects.

drawableRectangle	**drawableCylinder**
drawableSquare	**drawableAxes**
drawableCube	**drawableLine**
drawableSphere	**drawableSaggyLine**
drawableCone	**drawablePoints**
drawableCircle	**drawableObjModel**

drawableCollection

A collection of drawableCompound and drawableCollection objects. This is the scene graph.

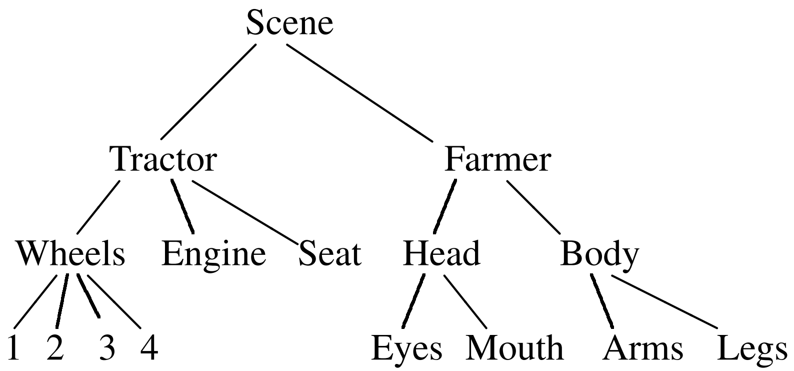
```
class drawableCollection : public drawableMulti {

  typedef std::map<std::string, bsgPtr<drawableMulti> >
                                            CollectionMap;
  CollectionMap _collection;

 public:
  std::string addObject(const std::string name,
               const bsgPtr<drawableMulti> &pMultiObject);
```

Scene graph redux

A look at the scene graph again. The drawableCollection objects are the branches on this tree and drawableCompound objects are the leaves. The drawableMulti is the parent class to both, but you won't instantiate any of them directly.



lightMgr

A list of lights and their colors.

```
class lightList {
 private:

  /// The positions of the lights in the list.
  drawableObjData<glm::vec4> _lightPositions;
  /// The colors of the lights in the list.
  drawableObjData<glm::vec4> _lightColors;
```

textureMgr

An easier way to load textures and make them available to a shaderhandle tex

```
class textureMgr {
 private:
  GLfloat _width, _height;

  GLuint _textureAttribID;
  std::string _textureAttribName;
  GLuint _textureBufferID;

  GLuint _loadPNG(const std::string imagePath);
  GLuint _loadCheckerBoard (const int size, int numFields);
  GLuint _loadTTF(const std::string ttfPath);
```