# Assignment 1

**CEG477/CEG677**

**Computer Graphics II**

**Assignment 1:**

Please implement a computer program that is capable of rendering a scene that – for now – consists entirely of polygons using OpenGL. Your software should be able to generate an image of the scene based on a PLY file that was given to your software program as a command line option.

A description of the PLY file format can be found, for example, at the following web page:

http://astronomy.swin.edu.au/~pbourke/dataformats/ply/

Implement a routine that reads polygons from a PLY file and converts it into your own data structure. Keep your data structure flexible, since it may be extended later on. Render the scene using the following OpenGL methods:

- Rendering using display lists
- Rendering using vertex arrays
- Rendering using vertex buffer objects

Implement a mechanism that allows your software to measure the time how long it takes to render an image, i.e. determine the time between two consecutive calls of your drawing routine. Compare the two different results on several sample scenes. You can find data sets for testing on the course's web page.

Your software should allow a user to rotate, zoom, and pan within the scene. You can either use your own viewing manipulation code or use the sample code from this web page:

http://www.nigels.com/glt/gltzpr/

Keep in mind that this assignment will be the basis for the next ones, so your code should be somewhat readable and extendable.

# Assignment 1

## The PLY file format

The PLY file format uses a header containing a description of the format of the following data. Most commonly, the header defines a list of vertices and then a list of faces (which can be triangles, quads, or any other polygonal structure). After the header, the data entries follow immediately one item after the other.

So, usually the structure looks like this:

*Header*
*Vertex List*
*Face List*
*(lists of other elements)*

# Assignment 1

## PLY header

First, a PLY file has an entry to identify itself as such and then defines its format (ASCII, binary). We will use the ASCII format so that the first two lines of our PLY files look like this:

*ply*
*format ascii 1.0*

Then, the following data items will be declared:

*element vertex*
*property float x*
*property float y*
*property float z*

Values can be of type float, float32, uchar, int, or even vertex_indices (a list of vertex indices)

# Assignment 1

## Example (cube)

*ply*
*format ascii 1.0*
*comment created by platoply*
*element vertex 8*
*property float32 x*
*property float32 y*
*property float32 z*
*element face 6*
*property list uint8 int32 vertex_indices*
*end_header*

# Assignment 1

**Example (data entries for cube)**

*-1 -1 -1*
*1 -1 -1*
*1 1 -1*
*-1 1 -1*
*-1 -1 1*
*1 -1 1*
*1 1 1*
*-1 1 1*
*4 0 1 2 3*
*4 5 4 7 6*
*4 6 2 1 5*
*4 3 7 4 0*
*4 7 3 2 6*
*4 5 1 0 4*

# Assignment 1

## Adding more information

You can add more properties to the data entries. For example, color information could be added to the vertices:

*ply*
*format ascii 1.0*
*comment created by platoply*
*element vertex 8*
*property float32 x*
*property float32 y*
*property float32 z*
*property red*
*property green*
*property blue*

# Assignment 1

## Grading

The assignment will be graded using the following criteria:

- Reading PLY file (10%)
- Rendering using display lists (25%)
- Rendering using vertex arrays (25%)
- Rendering using VBOs (25%)
- Timing comparison (10%)
- Code structure/documentation (5%)

# Assignment 1

## Further information about using VBOs

The web site *Neon Helium* describes several lessons for learning OpenGL. Generally, it gives very nice examples with downloadable source code that teach you how to use OpenGL. In particular, there is a specific lesson dealing with VBOs:

http://nehe.gamedev.net/data/lessons/lesson.asp?lesson=45

# Assignment 1

Measuring time:

**Linux**

*double now;*

*static double old = 0.0;*

*struct timeval tp;*

*gettimeofday (&tp, 0);*

*now = (double)tp.tv_sec + (1.e-6)*(double)tp.tv_usec;*

Framerate: *1.0 / (now - old)*

**Windows**

*__int64 now, freq;*

*static __int64 old = 0.0;*

*QueryPerformanceFrequency((LARGE_INTEGER*)&freq);*

*QueryPerformanceCounter ((LARGE_INTEGER*)&now);*

Framerate: *1000.0 / (double)((now - old) * 1000 / freq)*