Performance

Vertex Buffer Objects

Data Preparation

- Allocate and fill arrays with vertices and indices (optional)

```
// array for vertices
float *vertexB;
// fill arrays with vertex values

// array for indices
unsigned int *indices;
// fill arrays with indices
```

VBOs - Initialization

- Enable Buffer Functionality

  - Do once during initialization of the app

```
glEnableClientState(GL_VERTEX_ARRAY);
```

Buffer Initialization

- **Generate Vertex Buffer Objects**

```
// buffers is a global variable
// n is the number of buffers - one buffer per array
GLuint buffers[n];
...
glGenBuffers(n, buffers);
```

- **For Each Buffer**
  - Set buffer active

```
glBindBuffer(GL_ARRAY_BUFFER,buffers[0]);
```

  - Fill buffer

```
glBufferData(GL_ARRAY_BUFFER,arraySize, vertexB, GL_STATIC_DRAW);
```

in bytes

Index Buffer Initialization

- **Generate Index Buffer Object**

  ```
  // indexes is a global variable
  GLuint indexes[1];
  ...
  glGenBuffers(n, indexes);
  ```

- **Set buffer active**

  ```
  glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, indexes);
  ```

- **Fill buffer**

  ```
  glBufferData(GL_ELEMENT_ARRAY_BUFFER,arraySize,
                    indices, GL_STATIC_DRAW);
  ```

VBOs - Drawing

- Step 1: Set buffer active and define the semantics

```
glBindBuffer(GL_ARRAY_BUFFER,buffers[0]);
glVertexPointer(3,GL_FLOAT,0,0);
```

- Step 2 : Draw VBOs

  - With index list

```
glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, indicesBuff);
glDrawElements(GL_TRIANGLES, count ,GL_UNSIGNED_INT, NULL);
```

  - Without index list

```
glDrawArrays(GL_TRIANGLES, first, count);
```

  `first` – the starting index

  `count` – the number of vertices (not triangles) to draw

Frames per Second

```
int time;
time = glutGet(GLUT_ELAPSED_TIME);
```

- Returns the number of milliseconds since GLUT has been initialized

…

```
frame++;
time=glutGet(GLUT_ELAPSED_TIME);
if (time - timebase > 1000) {
    fps = frame*1000.0/(time-timebase));
    timebase = time;
    frame = 0;
}
```
…

- Use function `glutSetWindowTitle(char *s)` to display the `fps` counter (`sprintf`)

Practical Assignment

- Define vertex buffers for the cylinder (without indices)

- Initialization:

  - Create the arrays with the suitable dimension for the vertices of the cylinder
    - Number of vertices = sides x 3 + sides x 6 + sides x 3
      -         top   body   bottom
    - Each vertex takes three floats

  - Fill the vertex array with the appropriate values to draw the cylinder

  - Generate and enable the VBOs
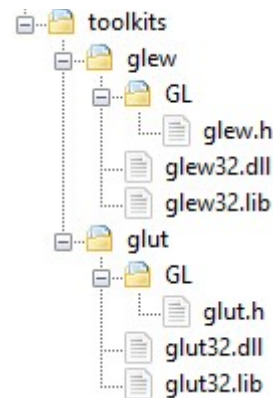
Practical Assignment

- Render:
  - Bind the array
  - Define the semantic for the vertex buffer
  - Use `glDrawArrays` to draw the cylinder

- Compute the FPS values obtained with and without VBOs for several cylinders and fill the grid below:

Mode\sides	4096	16384	65536	262144
Immediate mode				
VBO				

Installing GLEW (Win)

- Add the content of "glew.zip" to the toolkits folder such that the final configuration looks as follows:

```
toolkits
    glew
        GL
            glew.h
        glew32.dll
        glew32.lib
    glut
        GL
            glut.h
        glut32.dll
        glut32.lib
```

- Run Cmake and fill the TOOLKITS_FOLDER field with the correct path

Installing GLEW (Linux)

- Install GLEW
  - `sudo apt-get install libglew-dev`

OpenGL > 1.1 (Win & Linux)

- GLEW – library that facilitates access to OpenGL functionality post version 1.1 (Win)

```
#include <GL/glew.h> // before including glut.h
```

- In the main function (after GLUT's callback registry):

```
glewInit(); // before calling any OpenGL function
```