# OpenGL in C

A short summary of OpenGL programming in the C language on Linux

Einar Baumann

September 27, 2013

# 1 General program structure

## 1.1 The old way

The old structure for drawing with OpenGL is something like the following:

```
glBegin(GL_QUADS);
    glColor3f(1.0f, 1.0f, 1.0f);    // set red, green, blue
    glVertex3f(-0.8f, -0.8f, 0.0f);
    glVertex3f(0.8f, -0.8f, 0.0f);
    glVertex3f(0.8f, 0.8f, 0.0f);
    glVertex3f(-0.8f, 0.8f, 0.0f);
glEnd();
```

## 1.2 The new way

```
glClear(GL_COLOR_BUFFER_BIT);
const float coords[] = {
    -0.8f, -0.8f, 0.5f,
     0.8f, -0.8f, 0.5f,
     0.8f,  0.8f, 0.5f,
    -0.8f,  0.8f, 0.5f,
};
const unsigned char indeces[] = { 0, 1, 2, 0, 2, 3 };

glEnableClientState(GL_VERTEX_ARRAY);
glVertexPointer(3, GL_FLOAT, 0, coords);
glColor3f(1.0f, 1.0f, 1.0f);    // set red, green, blue
glDrawElements(GL_TRIANGLES, 6, GL_UNSIGNED_BYTE, indeces);
```

# 2 Libraries to include in Linux

```
#include<stdio.h>
#include<stdlib.h>
#include<X11/X.h>        // X window system
#include<X11/Xlib.h>     // Libraries for X window system
#include<GL/gl.h>        // Header file for OpenGL32 Library
#include<GL/glx.h>       // OpenGL Extension for X window system
#include<GL/glu.h>       // Header File For The GLu32 Library
```

Table 1: Possible datatypes for OpenGL commands.

| | |
|---|---|
| b | Byte |
| ub | Unsigned byte |
| s | Short |
| us | Unsigned short |
| i | Integer |
| ui | Unsigned integer |
| f | Float |
| d | Double |

Table 2: Possible values for number of components in OpenGL commands

| | |
|---|---|
| 2 | (x, y) |
| 2 | (x, y, z) |
| 4 | (x, y, z, w) |

## 3 OpenGL command formats

Lets look at an example function: `glVertex3fv( v )`. There are a number of interesting points here:

- `gl` is a prefix for all OpenGL functions.
- `Vertex` means that we're drawing a vertex.
- `3` means that the vertex has three components.
- `f` indicates that the components will have the datatype `float`.
- `v` indicates that the components will be given as a vector argument. The `v` may be omitted to use the scalar for instead, i.e. `glVertex3f( x, y, z )`.

The possible values for "number of components" and "datatype" are given in, respectively, tables ?? and ??.

## 4 OpenGL functions

`glVertexPointer(size, type, stride, pointer)` Allows OpenGL to extract positional data from varous array and memory constructs. Has to be initialized using `glEnableClientState(GL_VERTEX_POINTER)`. The four parameters are: