CompSci 372 – Tutorial

Part 3

OpenGL Programming

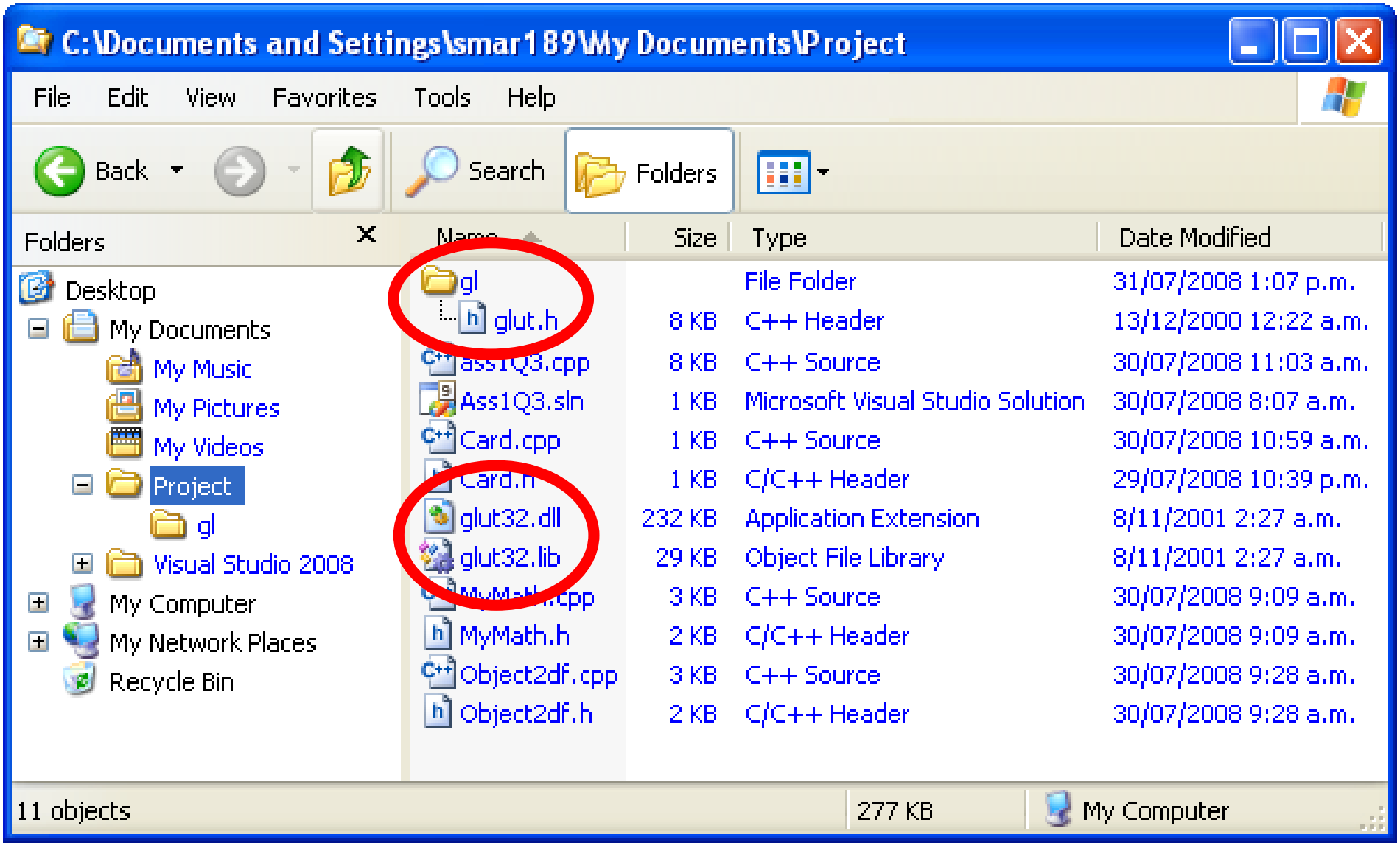
OpenGL Programming

- Lab computers:
  - Visual Studio 2008 √
  - OpenGL √
  - GLU √
  - GLUT –

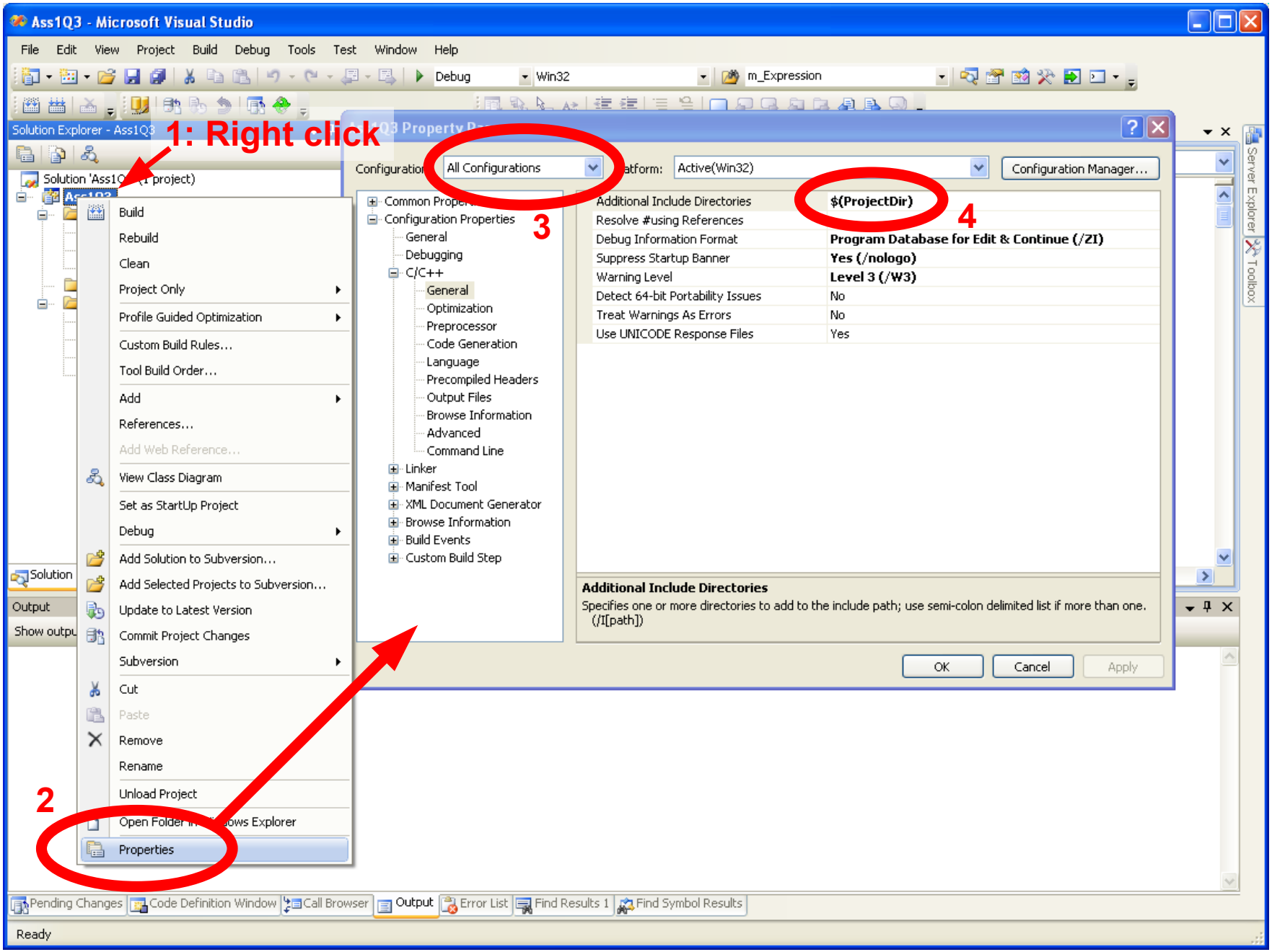
OpenGL Programming

- Lab machines: No access to system files
- → Install GLUT into your project:
  - Download .zip file from http://www.xmission.com/~nate/glut.html
  - Copy
    - glut.dll and glut.lib into your project directory
    - glut.h into a “gl”-subdirectory of your project directory
  - Add `$(ProjectDir)` to the include search path for the C++-Compiler

OpenGL Programming



OpenGL Programming



OpenGL Program

- Header files

```
#include <windows.h>
#include <gl/gl.h>
#include <gl/glu.h>
#include <gl/glut.h>
```

- Main program

```
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(windowWidth, windowHeight);
    glutInitWindowPosition(100, 100);
    glutCreateWindow("My first OpenGL program");

    init(); // initialise view

    glutDisplayFunc(display); // draw scene
    glutMainLoop();

    return 0;
}
```

OpenGL Program

- Init method

```
const int windowWidth=300;
const int windowHeight=400;

...

void init(void)
{
    // select clearing color (for glClear)
    glClearColor (1.0, 1.0, 1.0, 0.0);

    // initialize view (simple orthographic projection)
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    GLdouble halfWidth=(GLdouble) windowWidth/2.0;
    GLdouble halfHeight=(GLdouble) windowHeight/2.0;
    gluOrtho2D(-halfWidth, halfWidth, -halfHeight, halfHeight);
}
```

OpenGL Program

- Draw method

```
void display(void)
{
    // clear all pixels in frame buffer
    glClear(GL_COLOR_BUFFER_BIT);

    // draw something
    // ...

    // start processing buffered OpenGL routines
    glFlush();
}
```

OpenGL Program

- Flexible window

```
void reshape(int width, int height)
{
    // Set the viewport to be the entire window
    glViewport(0, 0, width, height);

    // Project so that (0,0) is always in the middle
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    GLdouble left   = (GLdouble) ( -width / 2.0);
    GLdouble right  = (GLdouble) (  width / 2.0);
    GLdouble bottom = (GLdouble) (-height / 2.0);
    GLdouble top    = (GLdouble) ( height / 2.0);
    gluOrtho2D(left, right, bottom, top);
}
```

```
int main(int argc, char** argv)
{
    ...
    glutReshapeFunc(reshape); // reshape function
    ...
}
```

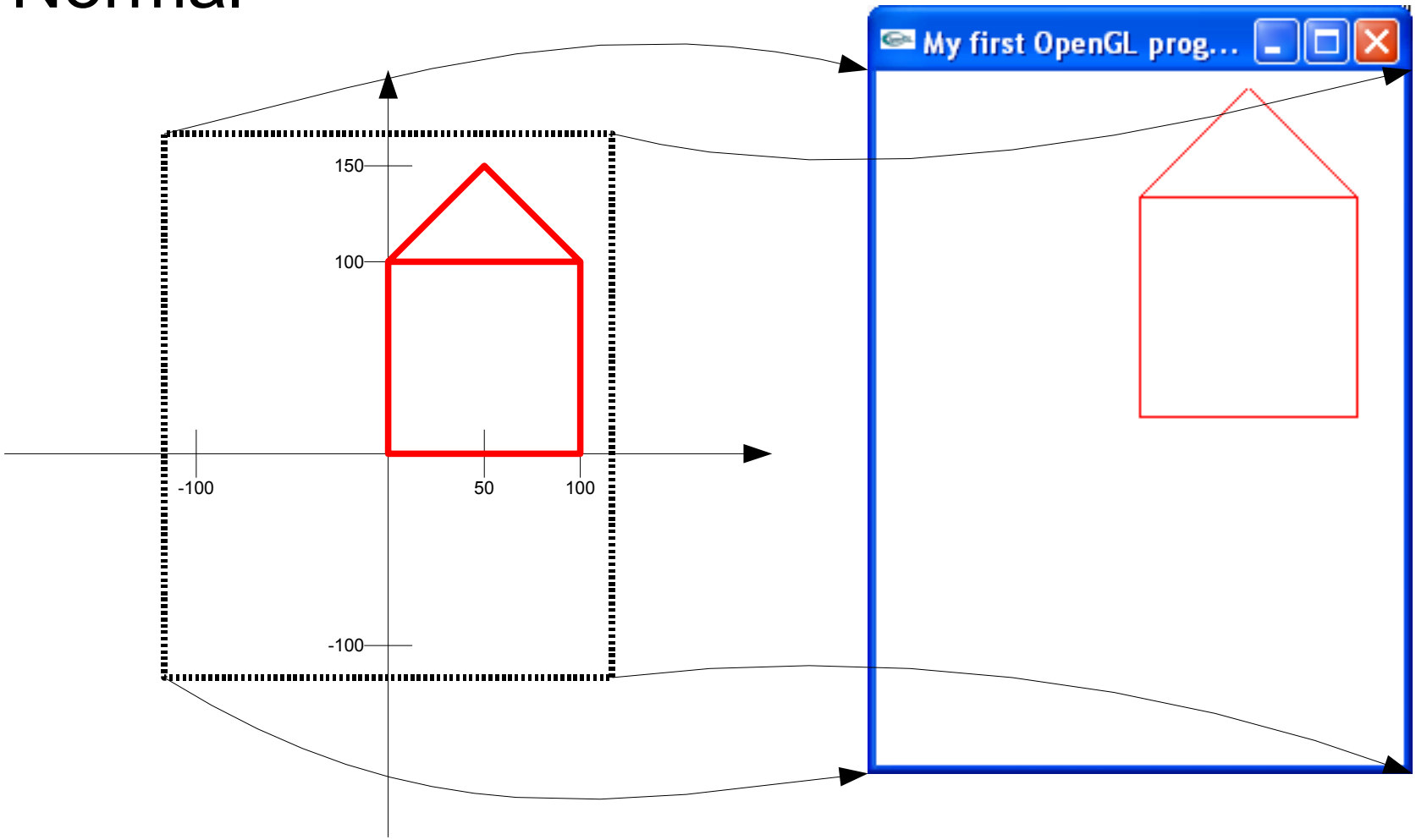
OpenGL Program

- Flexible window → Init method

```
void init(void)
{
    // select clearing color (for glClear)
    glClearColor (1.0, 1.0, 1.0, 0.0);

    // initialize view (simple orthographic projection)
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    GLdouble halfWidth=(GLdouble) windowWidth/2.0;
    GLdouble halfHeight=(GLdouble) windowHeight/2.0;
    gluOrtho2D(-halfWidth, halfWidth, -halfHeight, halfHeight);
}
```

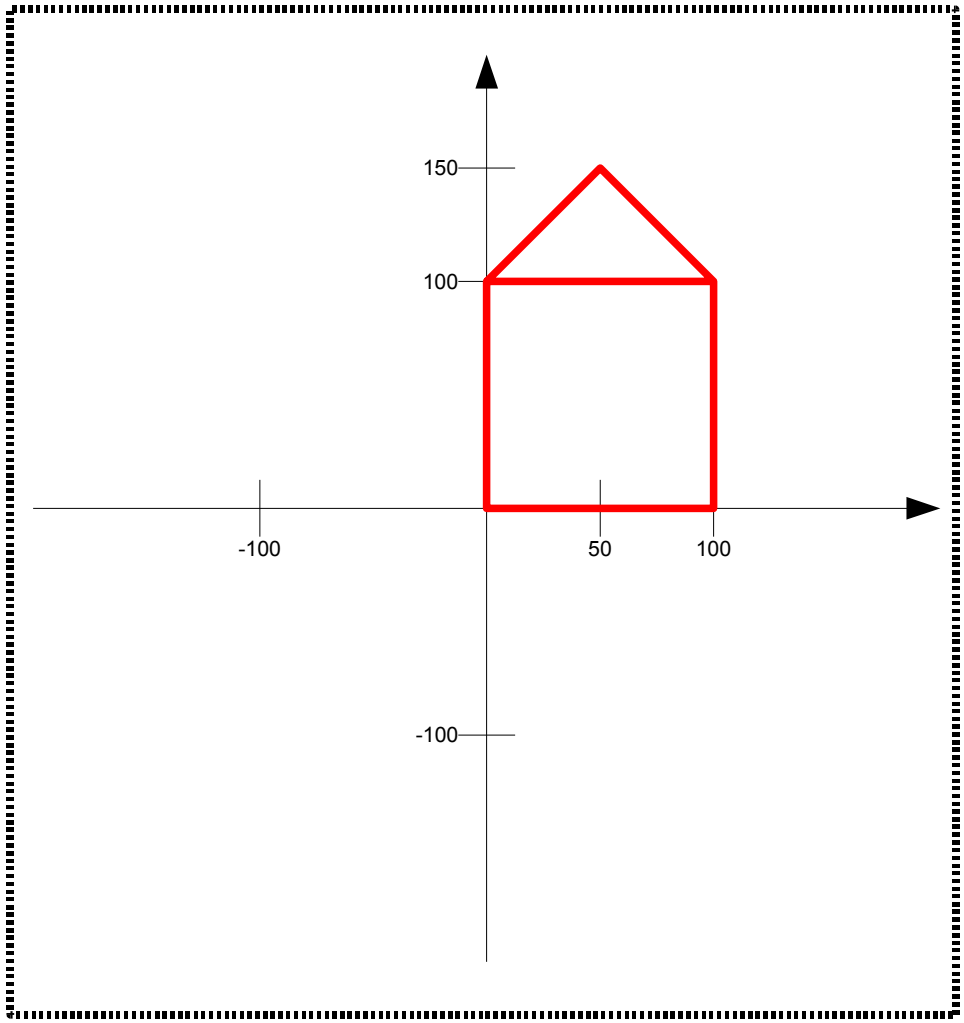
Viewport

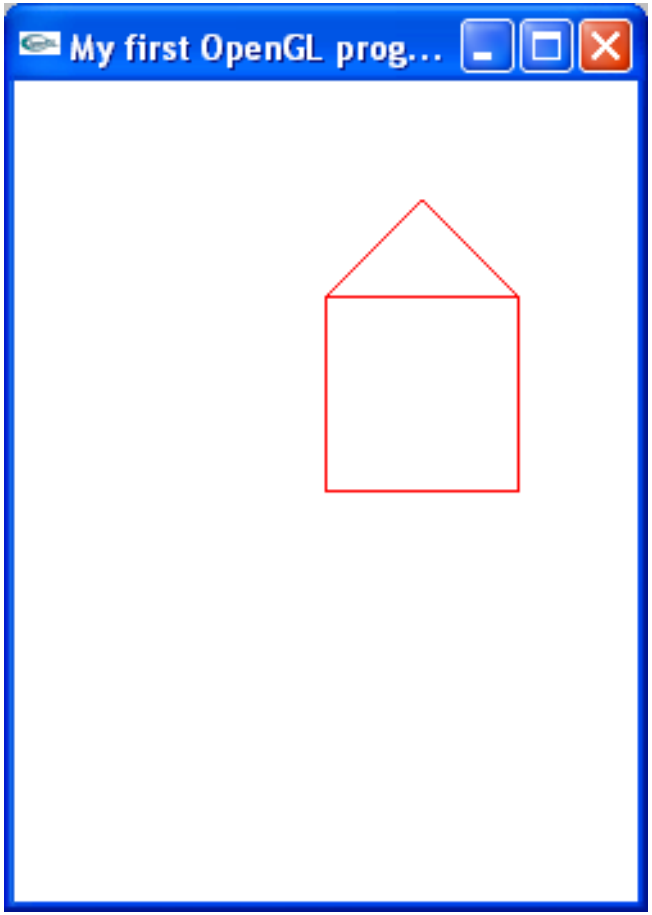
- Normal



Zoom in

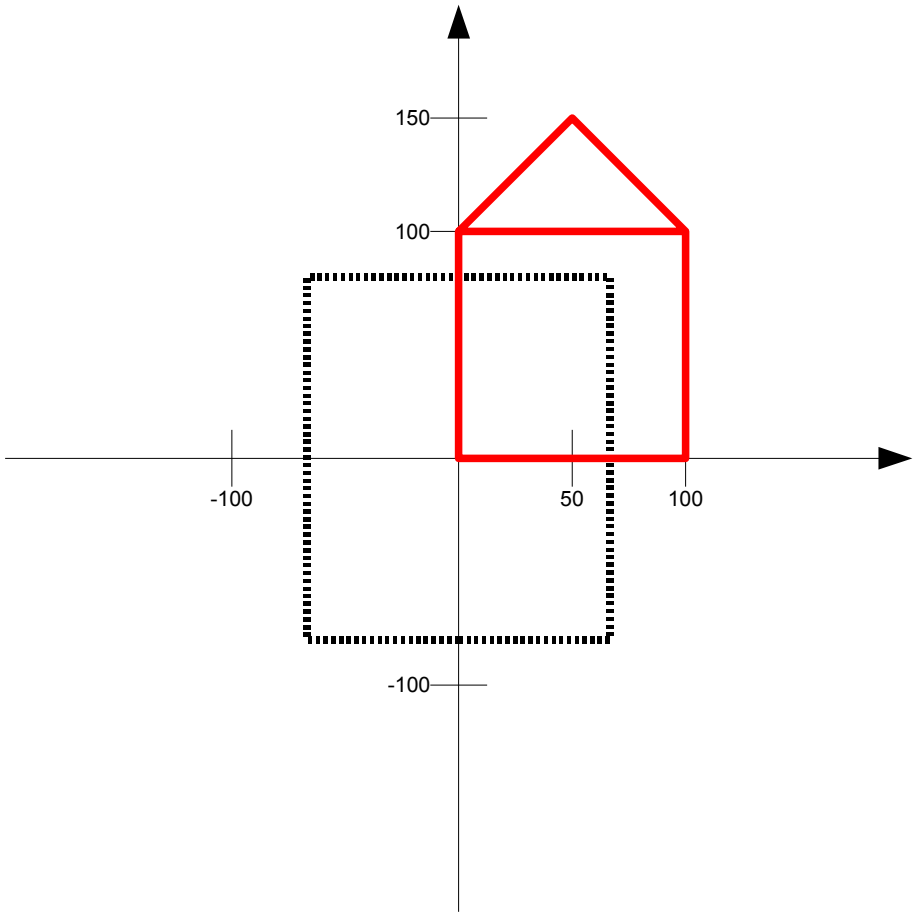
- Larger world-window

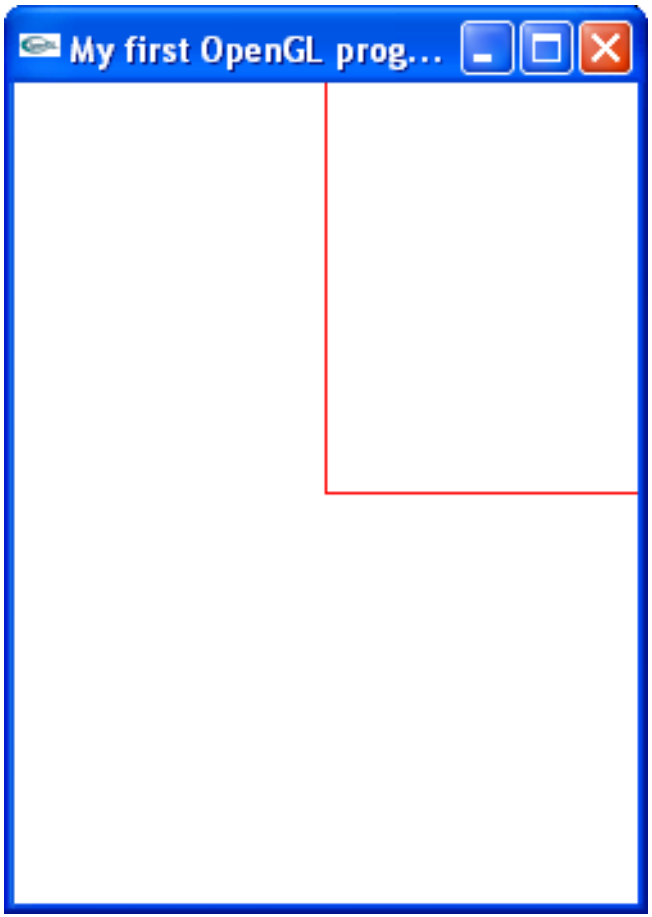




Zoom in

- Smaller world-window

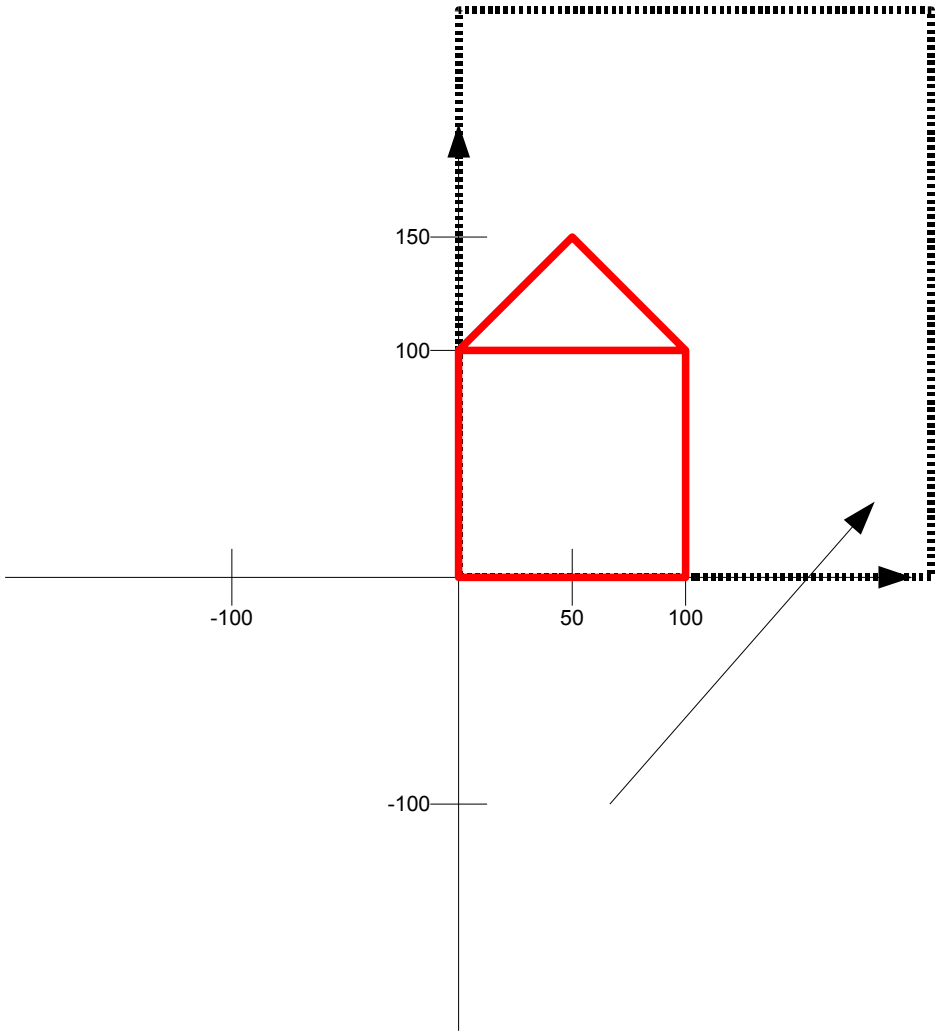


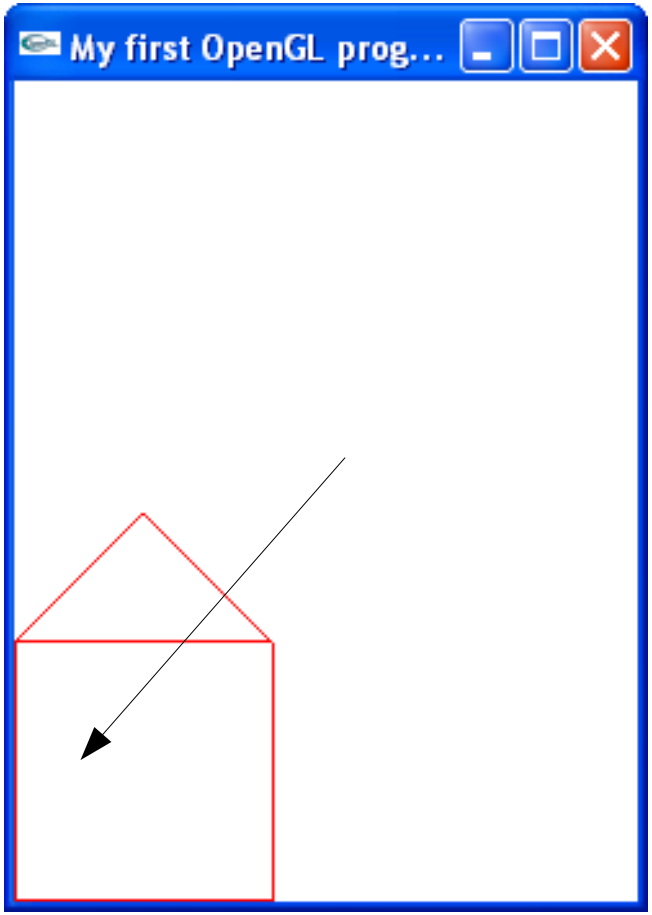


Zooming

```
float zoom = 1.0;

...

void reshape(int width, int height)
{
    // Set the viewport to be the entire window
    glViewport(0, 0, width, height);

    // Project so that (0,0) is always in the middle
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    GLdouble left   = (GLdouble) (( -width / 2.0) / zoom);
    GLdouble right  = (GLdouble) ((  width / 2.0) / zoom);
    GLdouble bottom = (GLdouble) ((-height / 2.0) / zoom);
    GLdouble top    = (GLdouble) (( height / 2.0) / zoom);
    gluOrtho2D(left, right, bottom, top);
}
```

Moving origin





Moving origin

```
void reshape(int width, int height)
{
    // Set the viewport to be the entire window
    glViewport(0, 0, width, height);

    // Project so that (0,0) is always in the middle
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    GLdouble left   = (GLdouble) 0.0;
    GLdouble right  = (GLdouble) width;
    GLdouble bottom = (GLdouble) 0.0;
    GLdouble top    = (GLdouble) height;
    gluOrtho2D(left, right, bottom, top);
}
```

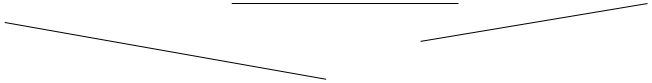
Viewport

- The World-Window → Viewport transformation does **only** allow for
  - Translation
  - Scaling (uniform/non-uniform)

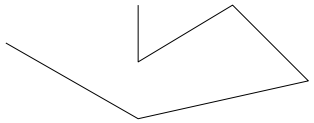
glBegin(...)

- **GL_POINTS**

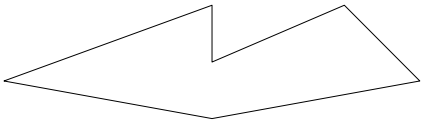
- **GL_LINES**



- **GL_LINE_STRIP**

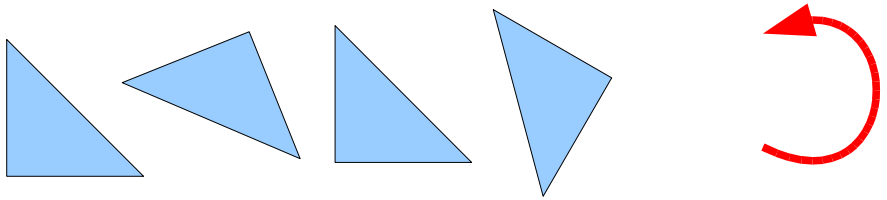


- **GL_LINE_LOOP**

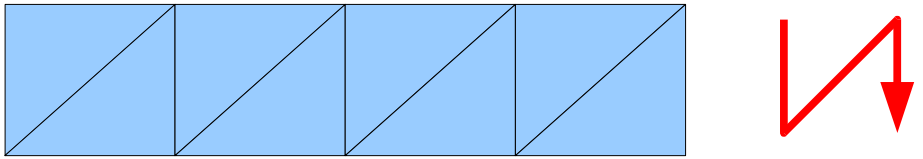


glBegin(...)

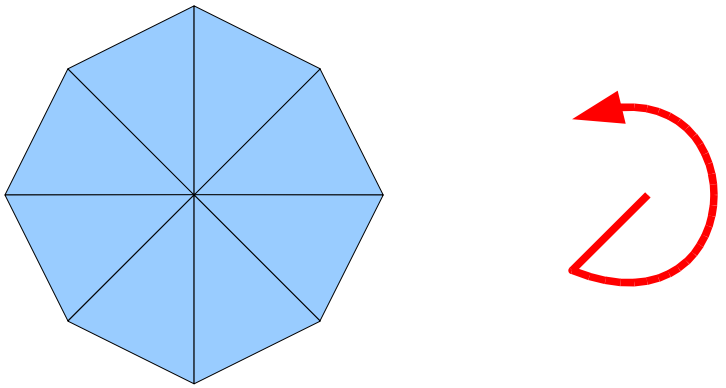
- **GL_TRIANGLES**



- **GL_TRIANGLE_STRIP**

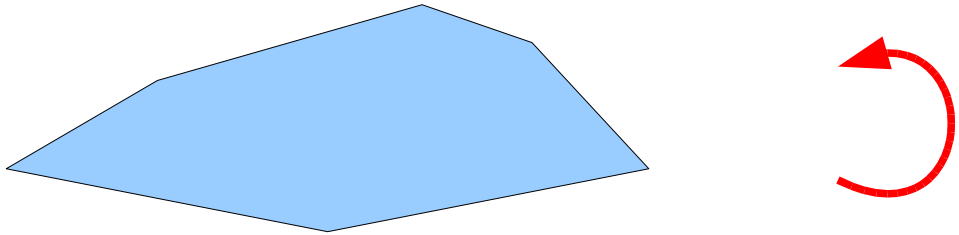


- **GL_TRIANGLE_FAN**



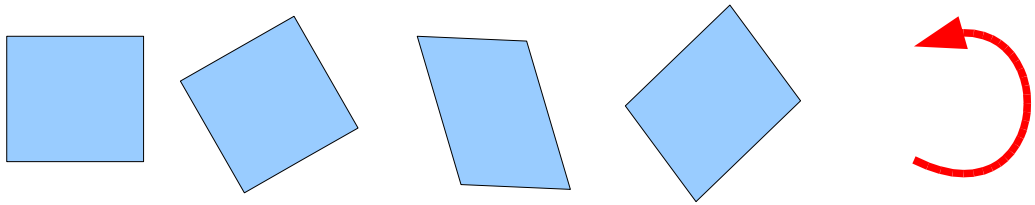
glBegin(...)

- **GL_POLYGON**



Polygon must be convex!
If not, the result is undefined.

- **GL_QUADS**



- **GL_QUAD_STRIP**

Tips

- Stick to clockwise/counter-clockwise when defining vertices!
  - This will define front/back of 3D surfaces (later)
- Think of how you can define a geometric form with as little **`glVertex...()`** calls as possible

glBegin(...)

- Draw with
  - **GL_TRIANGLES**
  - **GL_TRIANGLE_FAN**
  - **GL_TRIANGLE_STRIP**
  - **GL_QUADS**
  - **GL_QUAD_STRIP**

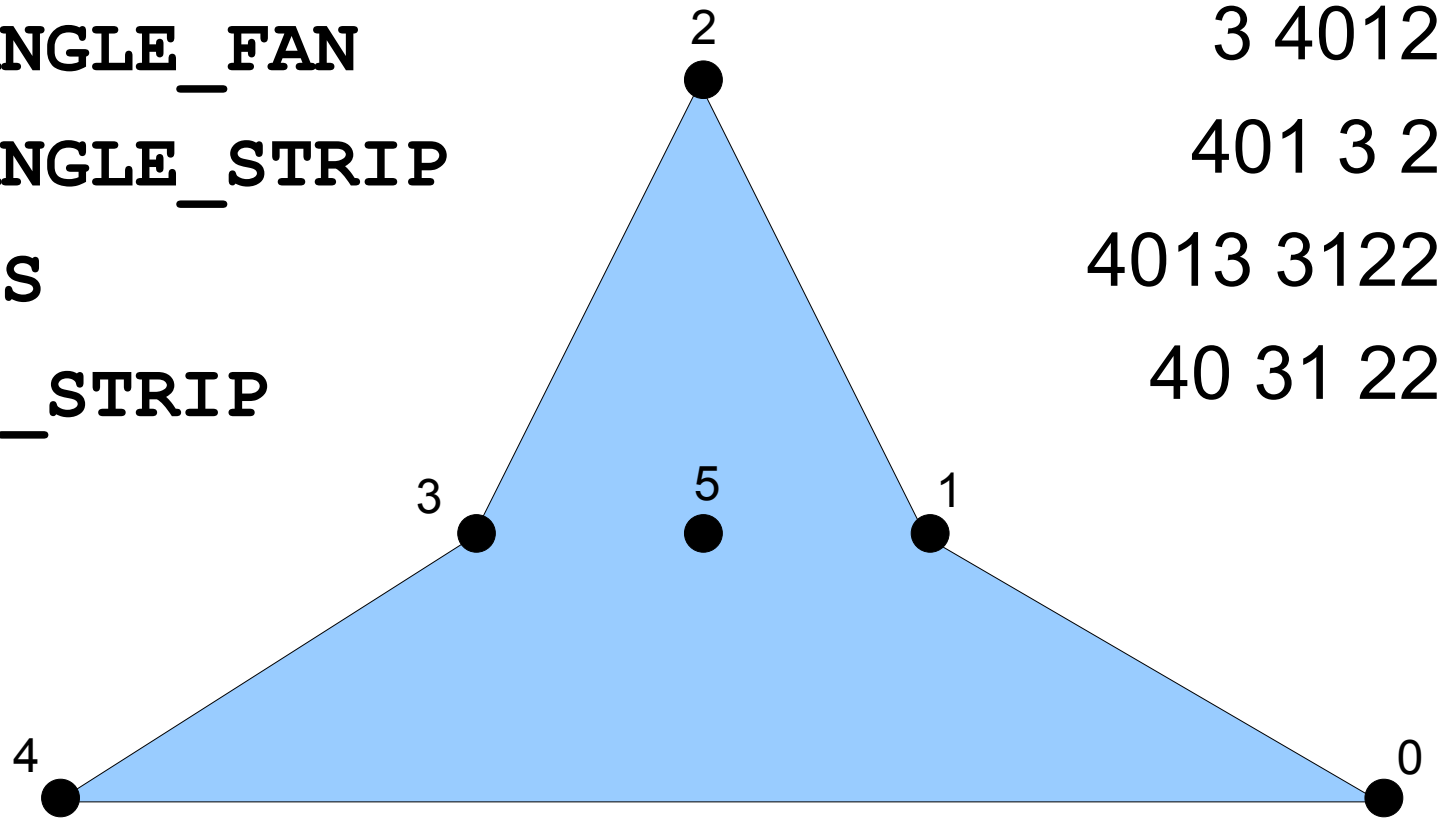
401 134 123

3 4012

401 3 2

4013 3122

40 31 22



Non-convex `GL_POLYGON`

- Draw with
  - **`GL_POLYGON`** → **`Result undefined`**