EKON 12

Die Entwickler-Konferenz für Delphi, C++, PHP & MORE

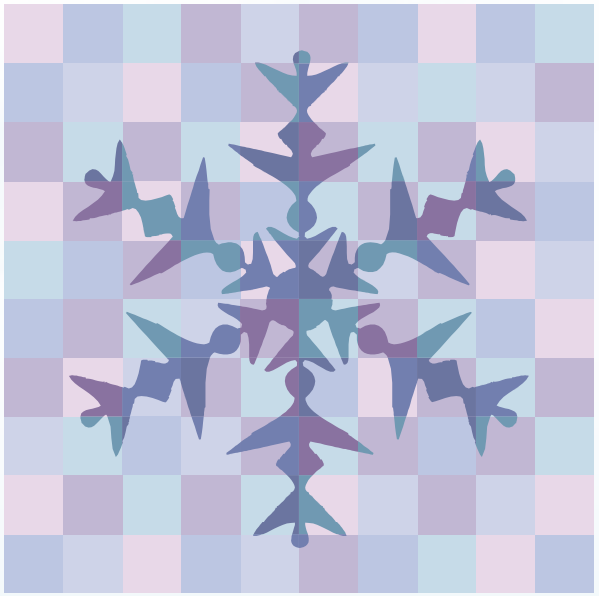
OpenGL & Delphi

Max Kleiner

http://max.kleiner.com/download/openssl_opengl.pdf

OpenGL

http://www.opengl.org



Evolution of Graphics

- Assembler (demo pascalspeed.exe)
- 2D 3D Animation, Simulation (Terrain_delphi.exe)
- Virtual Reality (CAD, Gothic II, trainsimulator)

Computer arbeitet deshalb so schnell, weil er nicht denkt.

Sign or design, that‘s the question

Lang ist der Weg durch Lehren, kurz und wirksam durch Beispiele

OpenGL Overview

- General OpenGL Introduction
- Rendering Primitives (points, lines, polygons and images)
- Rendering Modes
- Lighting
- Texture Mapping
- Imaging

Simple Code Schema

```
drawScene()
  begin
    //set background color
    glClearColor()
    glClear(GL_COLOR_BUFFER_BIT)
    //color of a polygon primitive
    glColor3f()
    //set the polygon
    glBegin(GL_POLYGON)
      glVertex2f()
      glVertex2f()
      .....
    glEnd()
    //draw the GL commands
    glFlush()
  end
```

What's OpenGL?

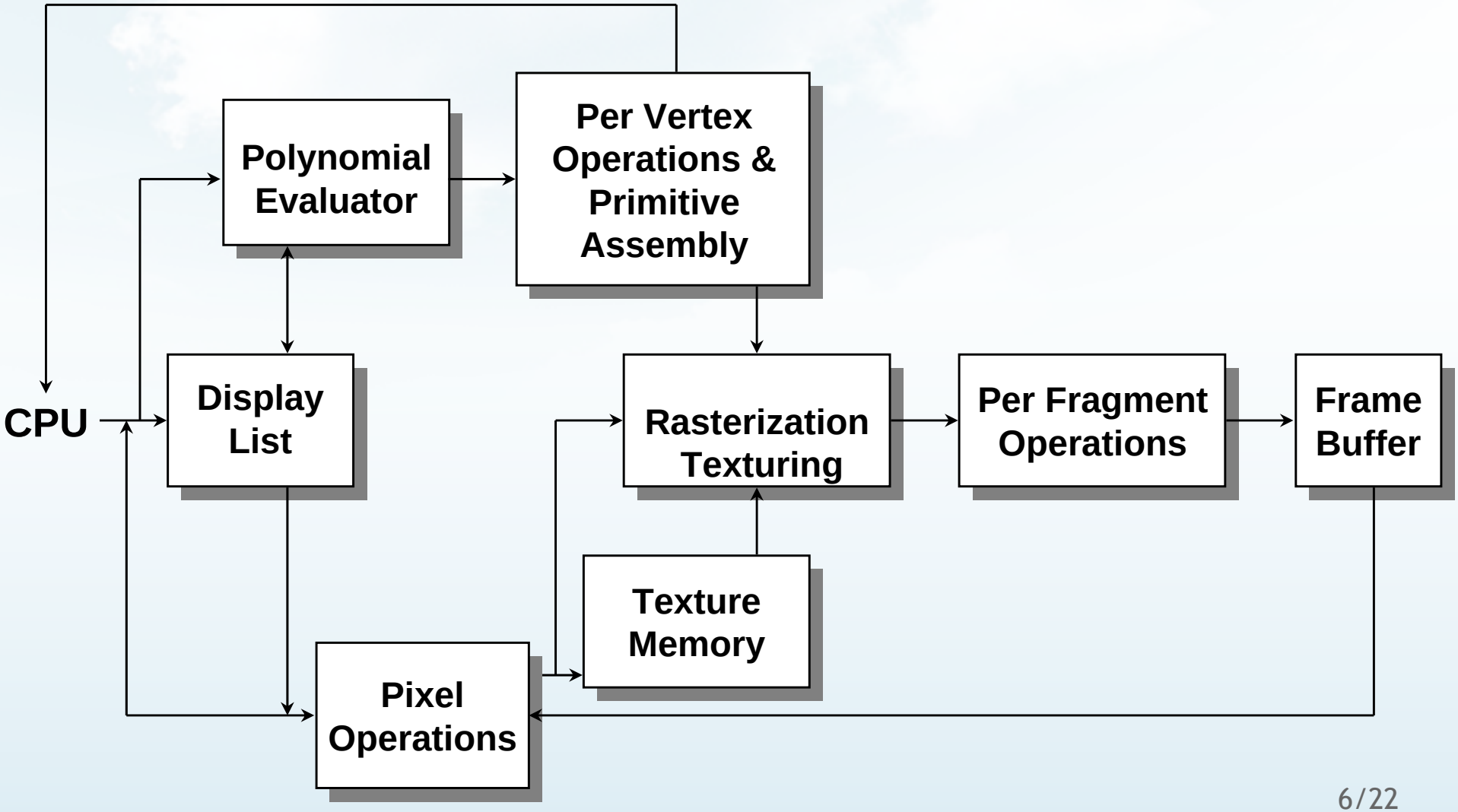
Graphics rendering API

- high-quality color images composed of geometric and image primitives with just 150 commands
- window system independent (context)
- operating system independent
- www.opengl.org /www.sgi.com/OpenGL

www.softwareschule.ch/download/opengldelphidemo.zip

- Open port from freepascal
- http://www.delphi3000.com/articles/article_5166

OpenGL Architecture



OpenGL as a Renderer

- Geometric primitives
  - points, lines and polygons
- Image Primitives
  - images and bitmaps
  - separate pipeline for images and geometry
    - linked through texture mapping
- Rendering depends on state
  - colors, materials, light sources, etc.

Generally, there are two operations that you do with OpenGL:

1. draw something
2. change the state of how OpenGL draws

Related APIs

AGL (mac), GLX, WGL (windows)

- glue between OpenGL and windowing systems

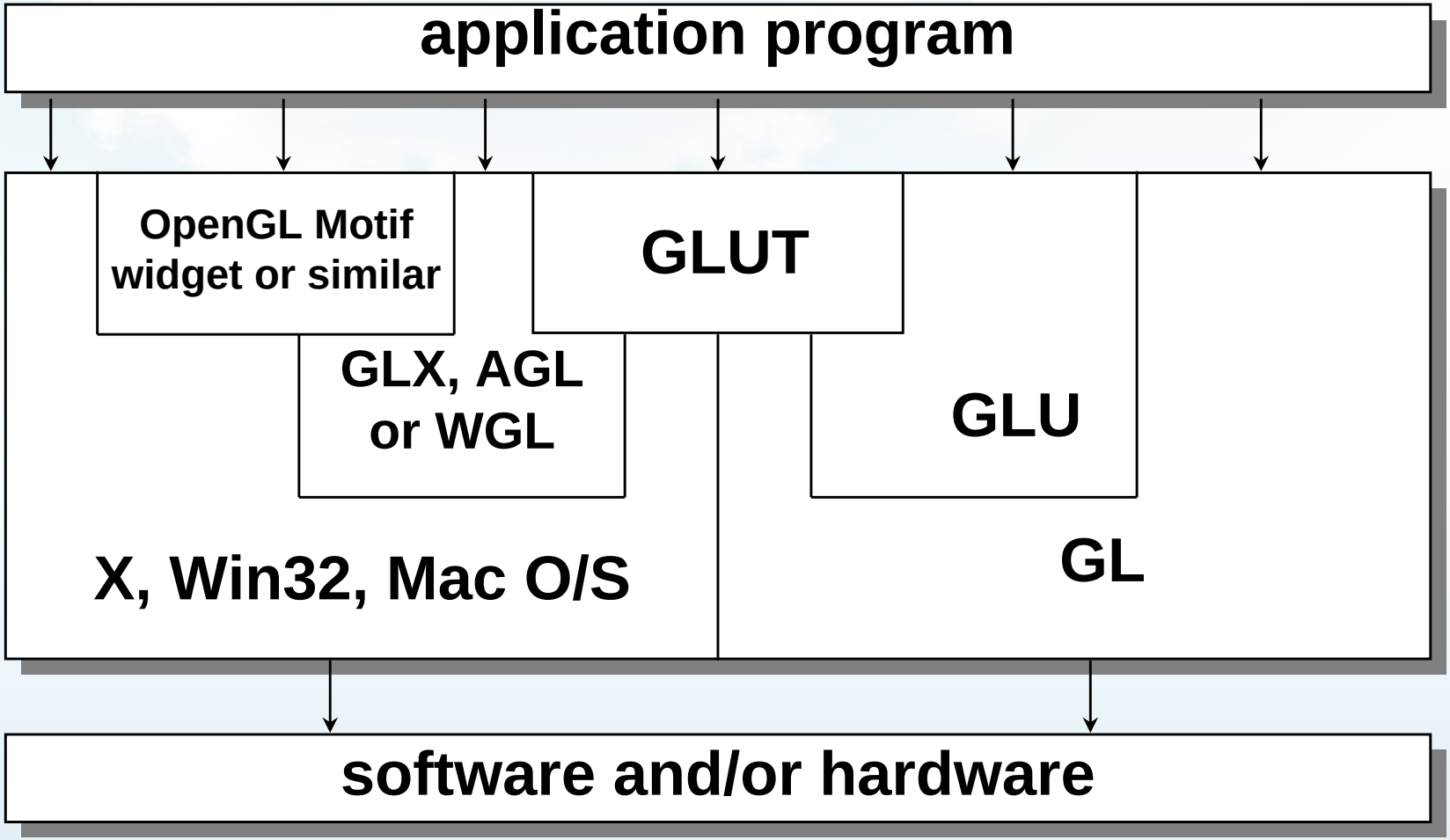
GLU (OpenGL Utility Library)

- part of OpenGL
- NURBS, tessellators, quadric shapes, etc.

GLUT (OpenGL Utility Toolkit)

- portable windowing API
- not officially part of OpenGL
- for making simple OpenGL applications

OpenGL and Related APIs



Preliminaries

Headers Files

uses Buttons, StdCtrls, GL, OpenGLContext_d;
version from Mike Lischke OpenGL12.pas

GL Converted to Delphi by Tom Nuydens :
http://www.delphi3d.net

`glut.h` includes `gl.h` and `glu.h`.

Libraries

opengl32 = 'OpenGL32.dll'; `libGL.so on Linux`
glu32 = 'GLU32.dll';

Enumerated Types

OpenGL defines numerous types for compatibility

GLfloat, GLint, GLenum, etc.

GLUT Basics

Application Structure Process

1. Configure and open window
2. Initialize OpenGL state (background color, light positions and texture maps).
3. Register input callback functions
   1. render, resize
   2. input: keyboard, mouse, etc.
4. Enter event processing loop (e.g. onIdle()

Sa(i)mple Program

```
void main( int argc, char** argv )
{
  int mode = GLUT_RGB|GLUT_DOUBLE;
  glutInitDisplayMode( mode );
  glutCreateWindow( argv[0] );
  init();
  glutDisplayFunc( display );
  glutReshapeFunc( resize );
  glutKeyboardFunc( key );
  glutIdleFunc( idle );
  glutMainLoop();
}
```

OpenGL Initialization

Set up whatever state you're going to use

```
void init( void )
{
  glClearColor( 0.0, 0.0, 0.0, 1.0 );
  glClearDepth( 1.0 );
  glEnable( GL_LIGHT0 );
  glEnable( GL_LIGHTING );
  glEnable( GL_DEPTH_TEST );
}
```

GLUT Callback Functions

Routine to call when something happens

- window resize or redraw
- user input
- Animation (mostly with onIdle)

OnPaint:= OpenGLControl1Paint;

OnResize:= OpenGLControl1Resize;

Idle Callbacks

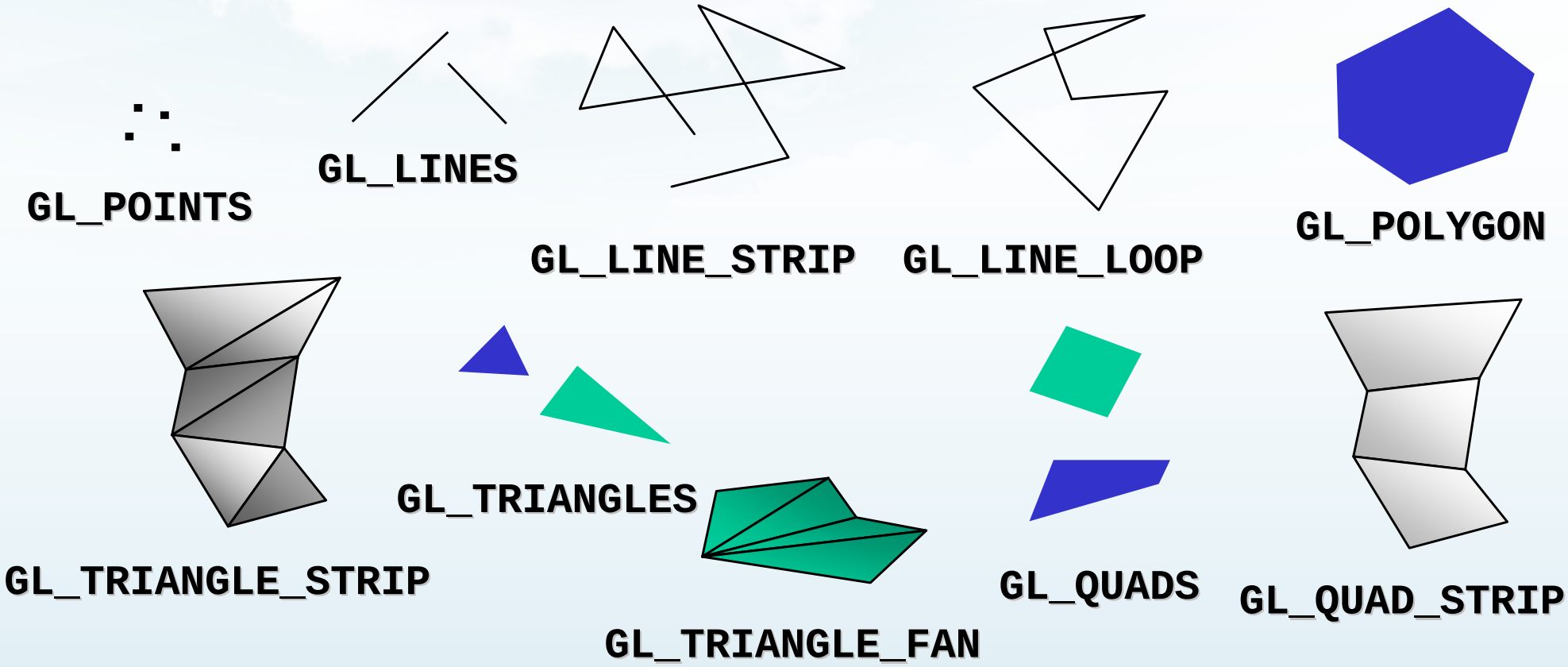
Use for animation and continuous update of motion variables

Application.OnIdle:= IdleFunc; (DEMO)

```
procedure TGLform.IdleFunc(Sender: TObject; var Done:
   Boolean);
begin
  openGLControl1.UpdateFrametimeDiff;
  OpenGLControl1Paint(Self);
  Done:= false;
  if openGLControl1.FrameDiffTimeInMSecs > 2 then
    glform.HintLabel1.Caption:=  'FPS: '
            +  inttoStr((1000 div
                 openGLControl1.FrameDiffTimeInMSecs));
end;
```

OpenGL Geometric Primitives

All geometric primitives are specified by vertices



Simple Example

```
void drawRhombus( GLfloat color[] )
{
    glBegin( GL_QUADS );
    glColor3fv( color );
    glVertex2f( 0.0, 0.0 );
    glVertex2f( 1.0, 0.0 );
    glVertex2f( 1.5, 1.118 );
    glVertex2f( 0.5, 1.118 );
    glEnd();
}
```

OpenGL Command Formats

glVertex3fv(v)

Number of components

```
2 - (x,y)
3 - (x,y,z)
4 - (x,y,z,w)
```

Data Type

```
b  - byte
ub - unsigned byte
s  - short
us - unsigned short
i  - int
ui - unsigned int
f  - float
d  - double
```

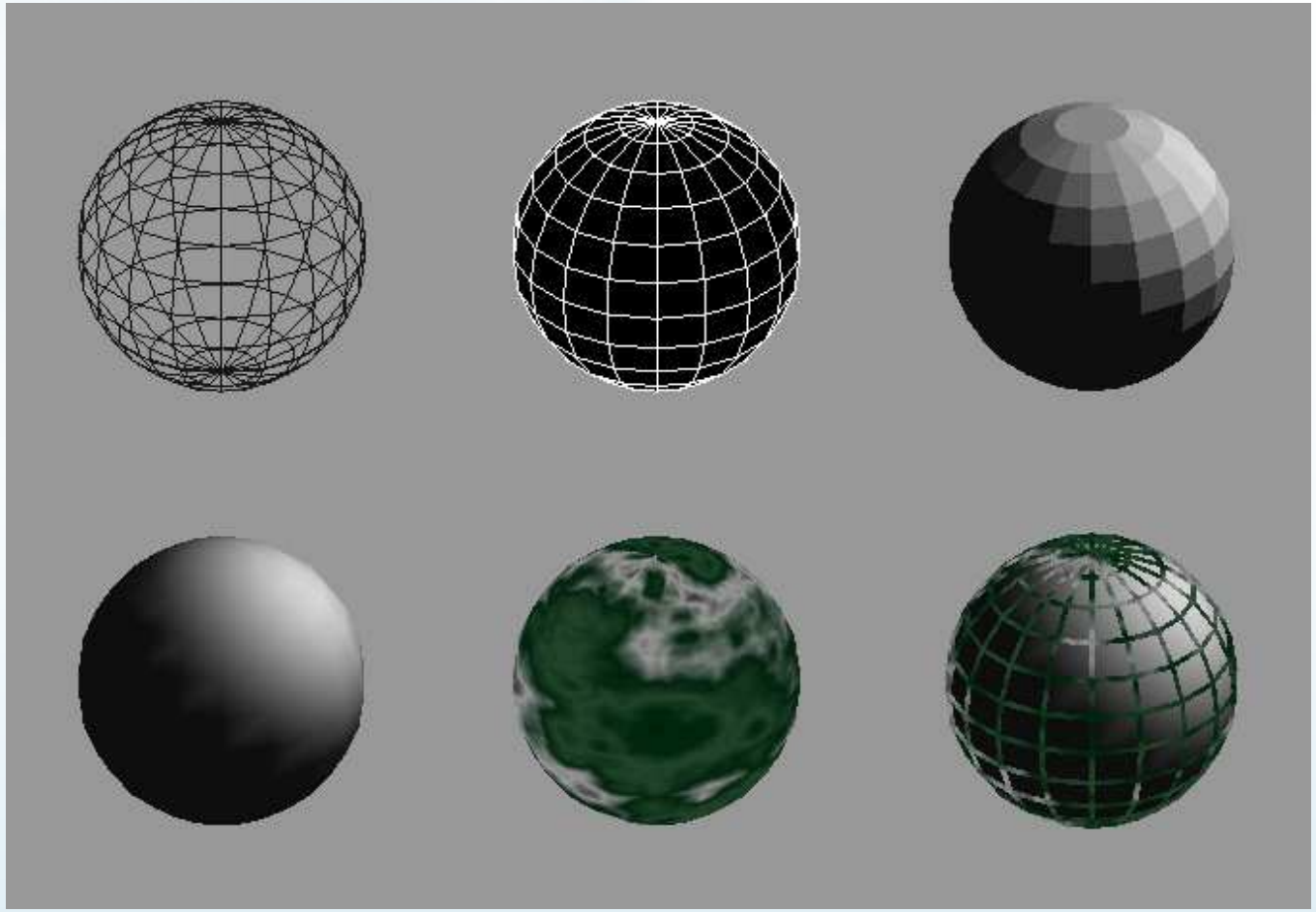
Vector (coords)

omit “v” for scalar form

```
glVertex2f( x, y )
```

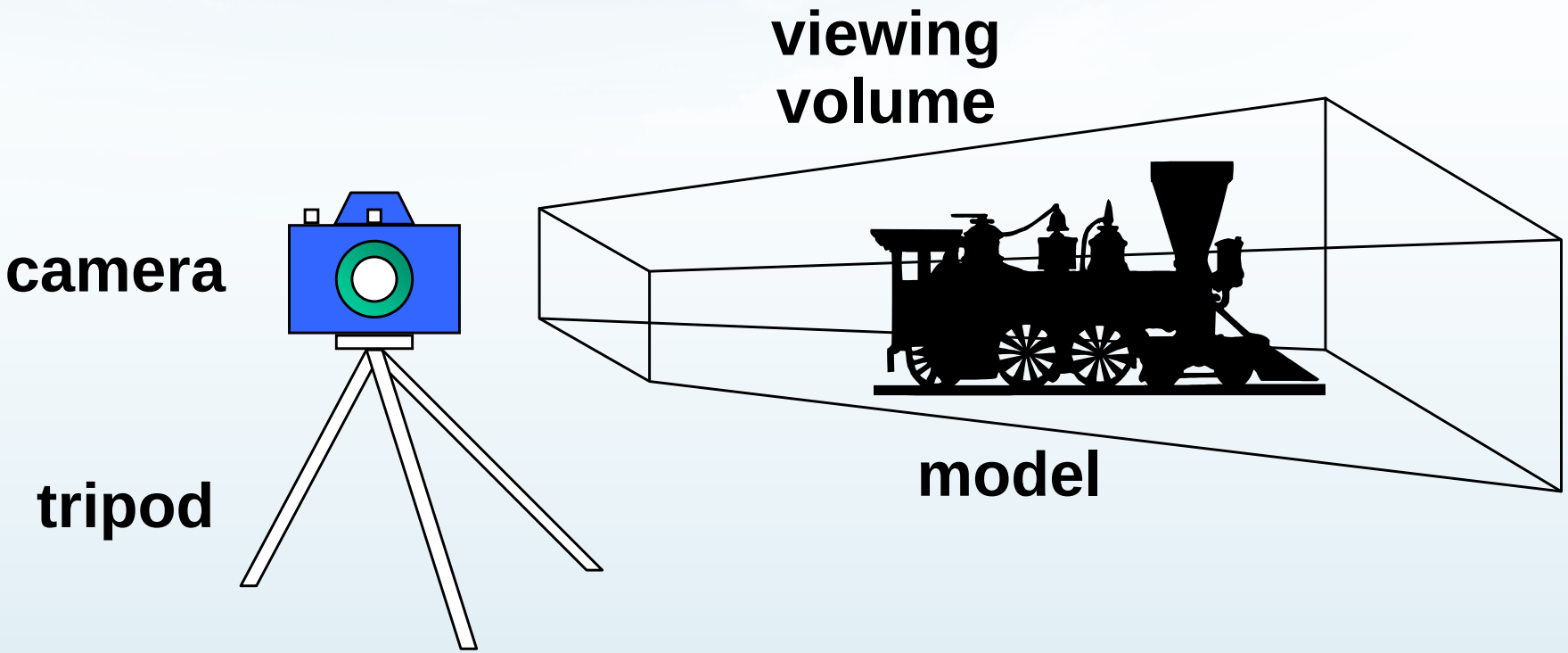
Controlling Rendering Appearance

From Wireframe to Texture Mapped



Transformations in OpenGL Camera Analogy

3D is just like taking a photograph (lots of photographs!)



Camera Analogy and Transformations

- Projection transformations
  - adjust the lens of the camera
- Viewing transformations
  - tripod–define position and orientation of the viewing volume in the world
- Modeling transformations
  - moving the model
- Viewport transformations
  - enlarge or reduce the physical photograph

That’s all Folks ;)

if you can’t see it

you can’t manage IT