# OpenGL™

# part 1

Martin Samuelčík
http://www.sccg.sk/~samuelcik
Room I4

# Course syllabus

- Introduction, basic settings
- Drawing, vertex attributes
- Transformations
- Shaders (vertex, fragment)
- Rasterization
- Textures, images
- Buffers, fragment operations & tests
- Extensions, GLEW
- GLU, WGL
- Additional shaders
- WebGL, OpenGL ES

# Course evaluation

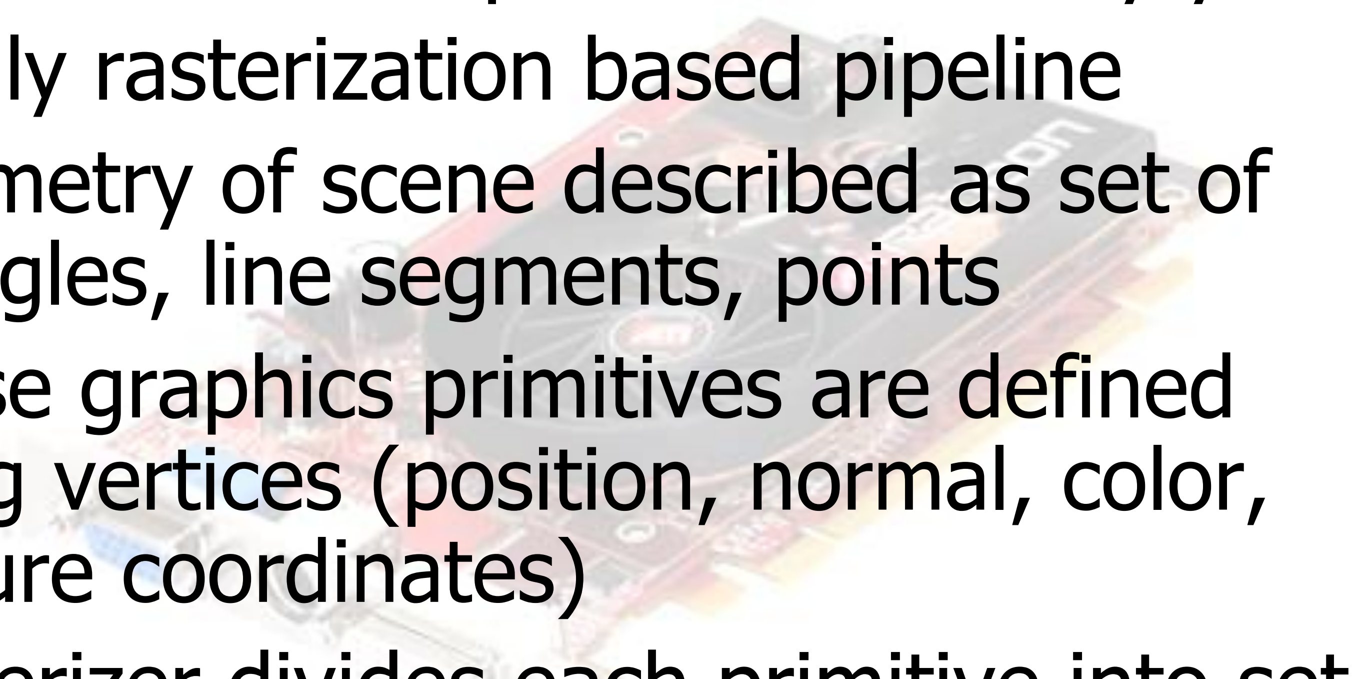
- Evaluation is based on one project that will be personally presented
- Project = computer interactive game
- Possibility to use external libraries
- Basic OpenGL functions and evaluated functionality must be programmed by you
- Arbitrary programming language
- Preferred platform Win32, other platforms possible, but student must provide necessary hardware and software

# Course evaluation

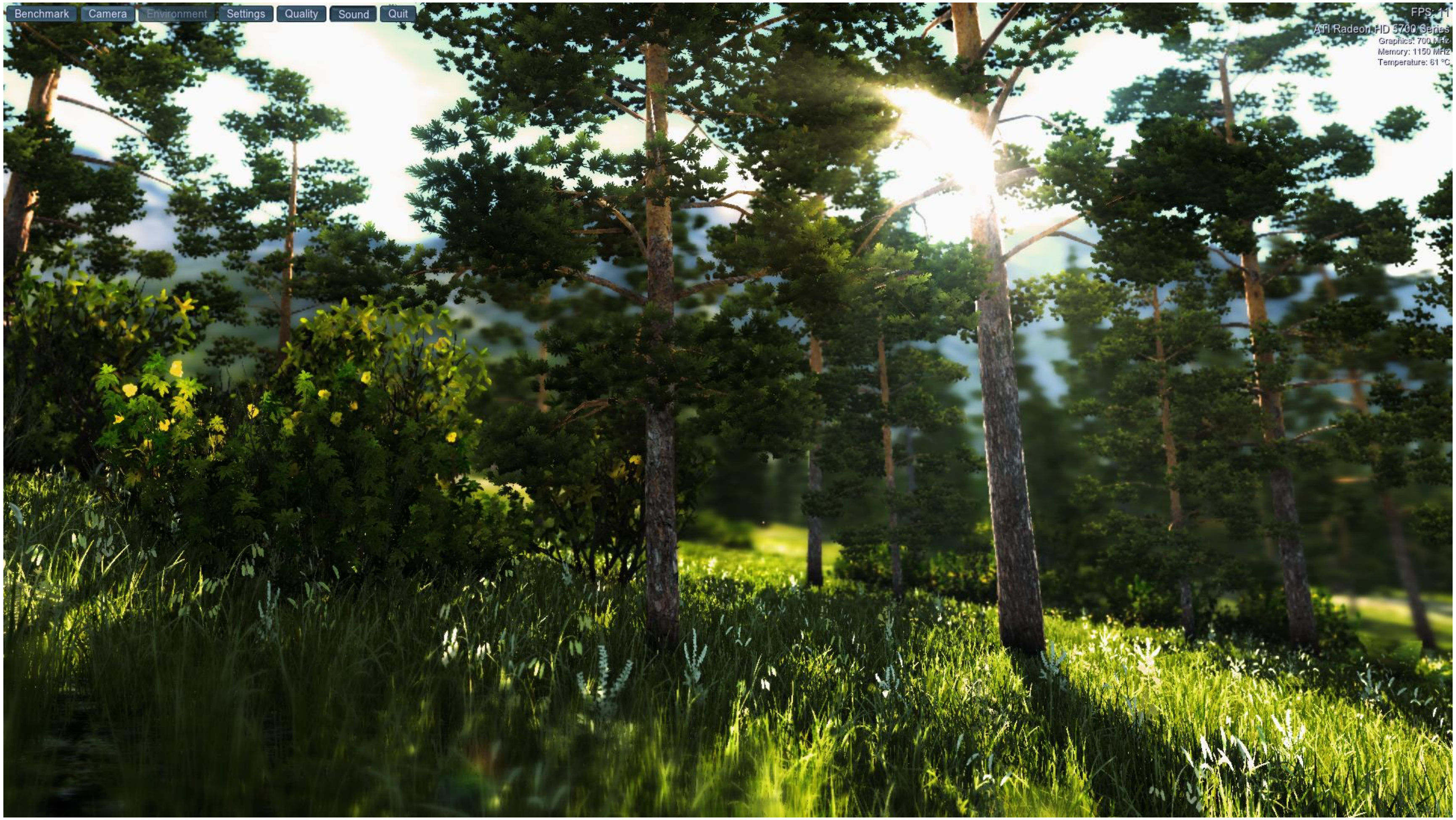
- Conditions for project can be found at http://www.sccg.sk/~samuelcik
- Showcase of projects from previous years https://vimeo.com/album/2436376
- Grades:
  - **A**: 100-90 pts
  - **B**: 89-80 pts
  - **C**: 79-70 pts
  - **D**: 69-60 pts
  - **E**: 59-50 pts
  - **Fx**: 49-00 pts

# Graphics hardware

- Great and fast improvements every year
- Mainly rasterization based pipeline
- Geometry of scene described as set of triangles, line segments, points
- These graphics primitives are defined using vertices (position, normal, color, texture coordinates)
- Rasterizer divides each primitive into set of small fragments (these fragments become pixels on screen at the end)



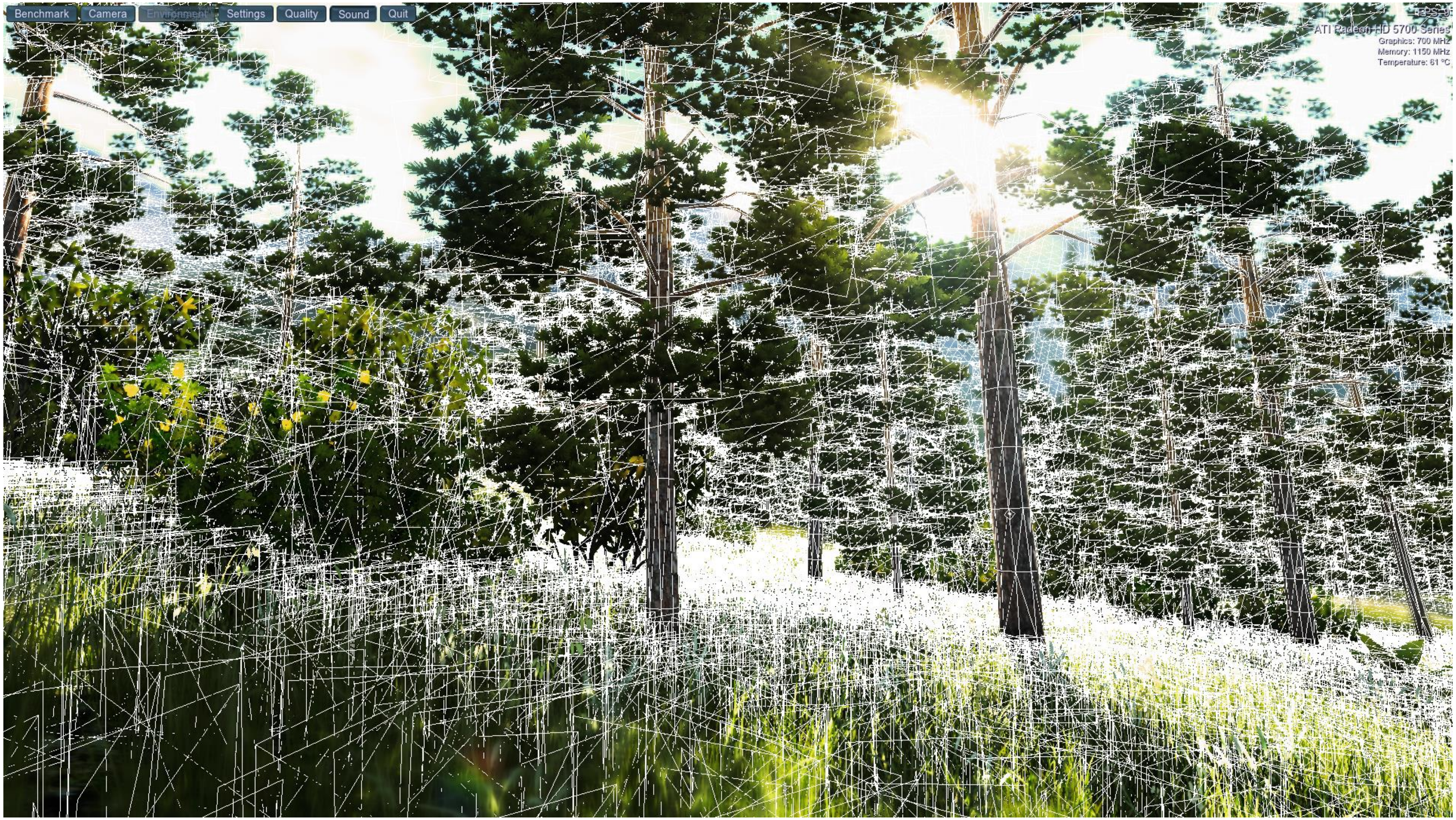
# Graphics hardware



FPS: 11
ATI Radeon HD 5700 Series
Graphics: 700 MHz
Memory: 1150 MHz
Temperature: 61 °C

http://unigine.com/products/valley/

# Graphics hardware



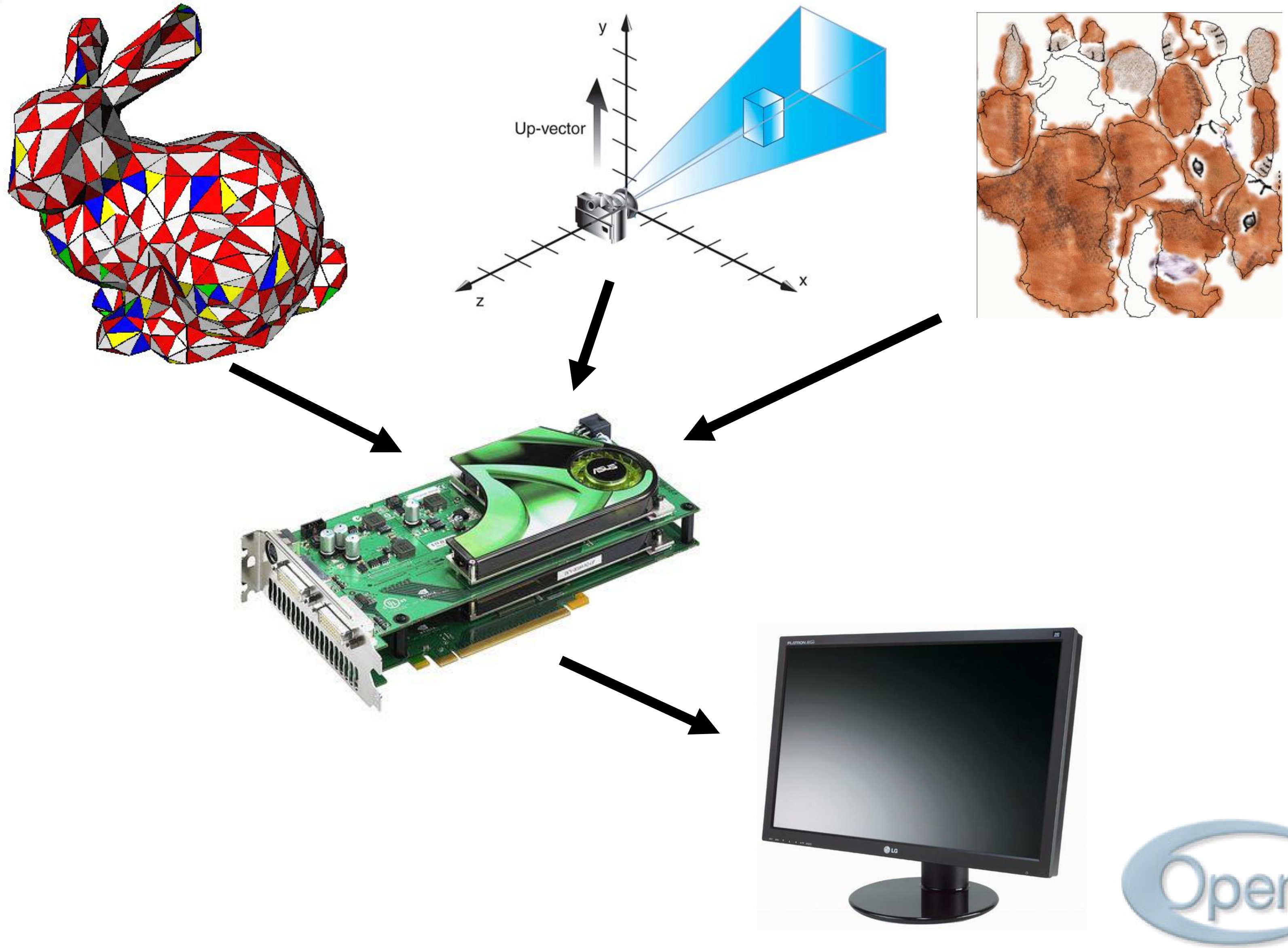
ATI Radeon HD 5700 Series
Graphics: 700 MHz
Memory: 1150 MHz
Temperature: 61 °C

OpenGL®

# What is OpenGL?

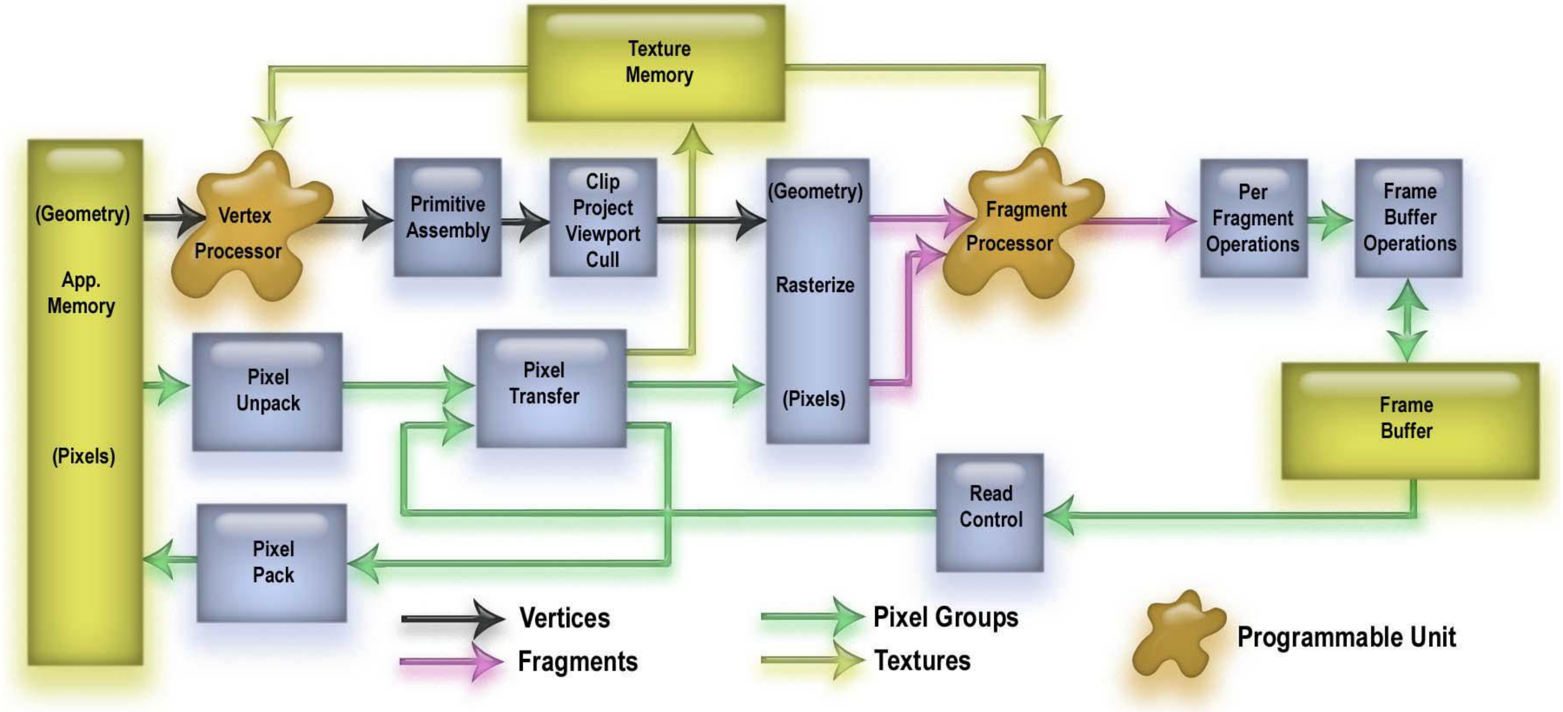
- Bringing 3D virtual world to 2D screen
- API – application programming interface – set of functions for defining virtual world and rendering it
- Support in graphics hardware = rendering is optimized
- Support for many programming languages
- Support for many operating systems
- It is programming!!!!!!!!!!!

# OpenGL model

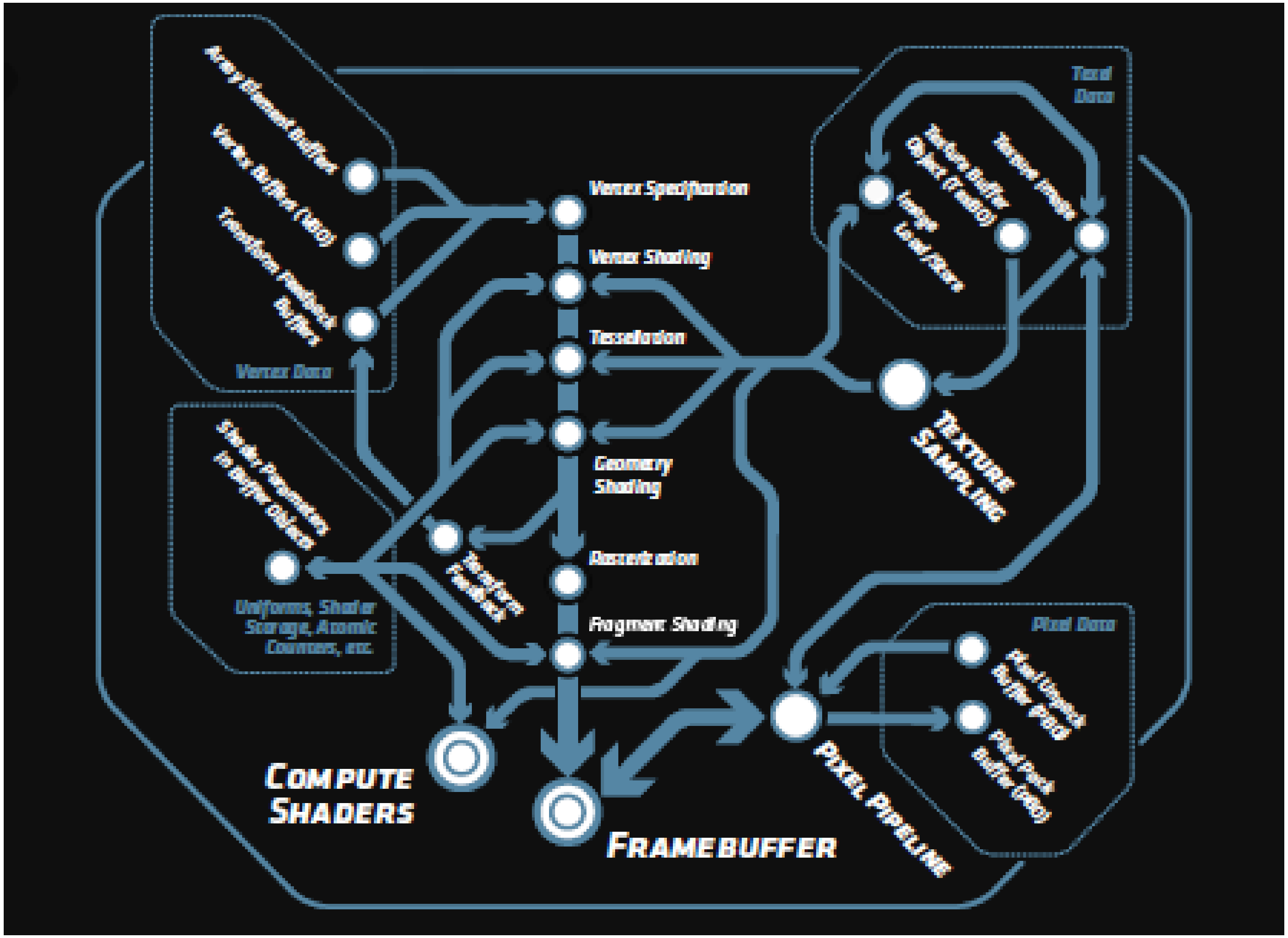


# OpenGL 2.0 pipeline

- Flow of data inside OpenGL
- We will learn about all boxes in diagram



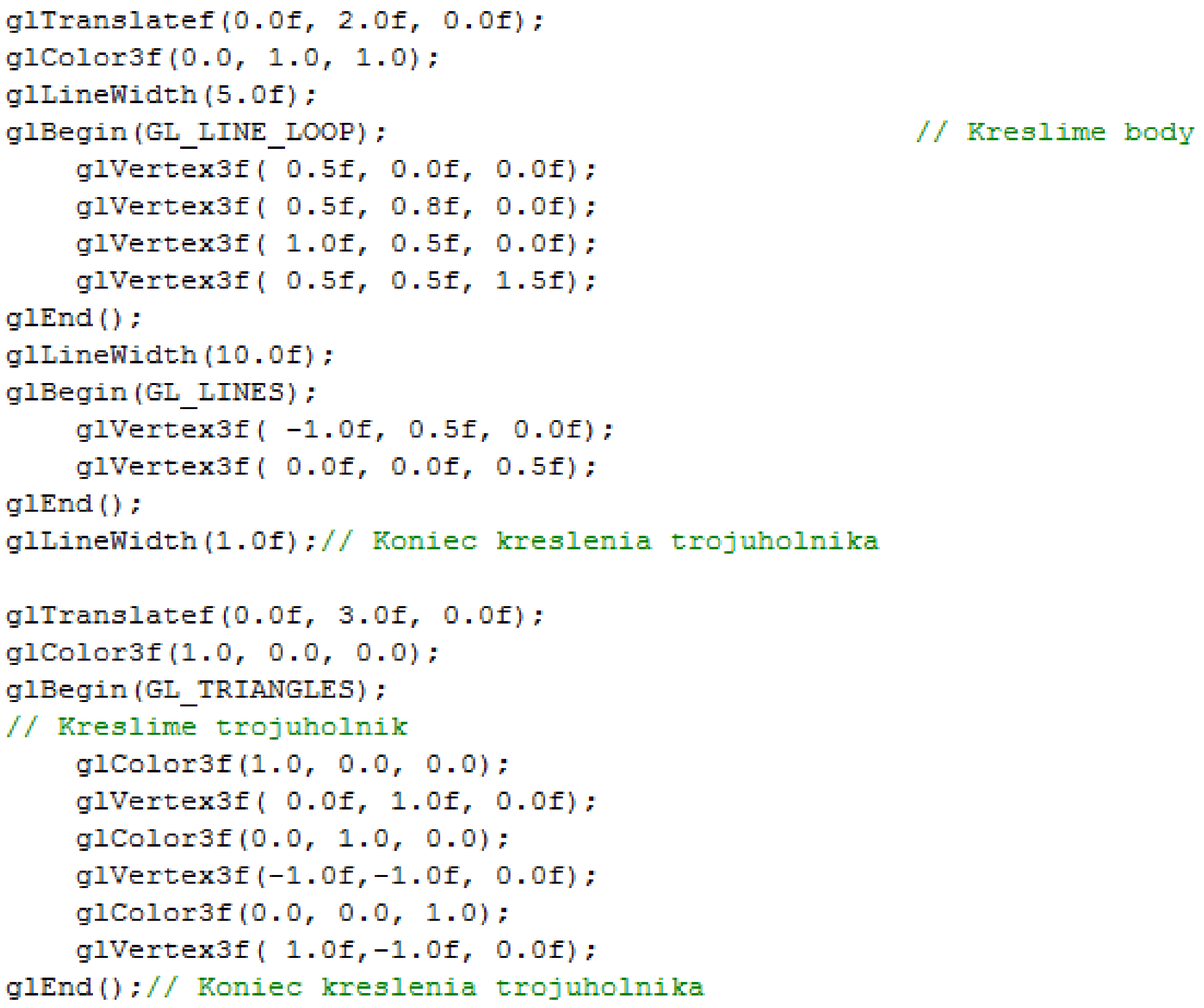
# OpenGL 4.3 pipeline



# Specifications

- Specification – description of whole functionality (functions, parameters, constants, tokens) of library
- Versions 1.0 – 4.5 (change 3.3), since 1992 – we will work with version 2.0
- Currently maintained by Khronos Group
- Low level functions, basic necessary functionality
- Many support libraries – GLU, GLUT, GLEW, ……
- Implementations of specification - Window system creators, Graphics cards vendors, Software implementations (Mesa), …

# Source code example

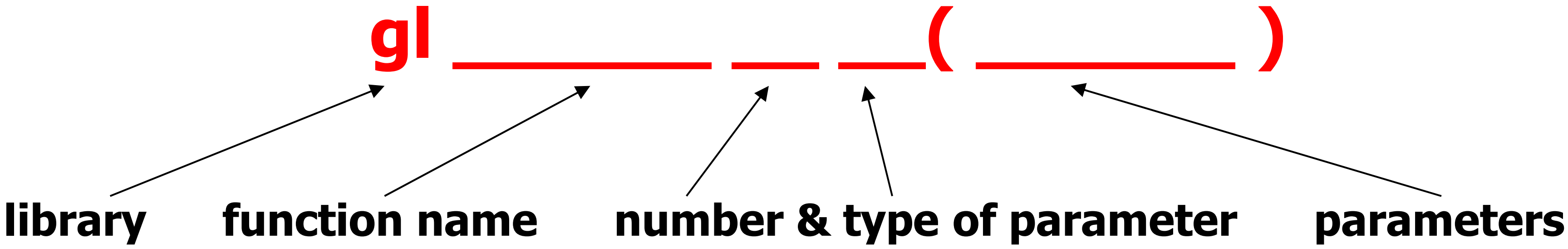


```
glTranslatef(0.0f, 2.0f, 0.0f);
glColor3f(0.0, 1.0, 1.0);
glLineWidth(5.0f);
glBegin(GL_LINE_LOOP);                              // Kreslime body
    glVertex3f( 0.5f, 0.0f, 0.0f);
    glVertex3f( 0.5f, 0.8f, 0.0f);
    glVertex3f( 1.0f, 0.5f, 0.0f);
    glVertex3f( 0.5f, 0.5f, 1.5f);
glEnd();
glLineWidth(10.0f);
glBegin(GL_LINES);
    glVertex3f( -1.0f, 0.5f, 0.0f);
    glVertex3f( 0.0f, 0.0f, 0.5f);
glEnd();
glLineWidth(1.0f);// Koniec kreslenia trojuholnika

glTranslatef(0.0f, 3.0f, 0.0f);
glColor3f(1.0, 0.0, 0.0);
glBegin(GL_TRIANGLES);
// Kreslime trojuholnik
    glColor3f(1.0, 0.0, 0.0);
    glVertex3f( 0.0f, 1.0f, 0.0f);
    glColor3f(0.0, 1.0, 0.0);
    glVertex3f(-1.0f,-1.0f, 0.0f);
    glColor3f(0.0, 0.0, 1.0);
    glVertex3f( 1.0f,-1.0f, 0.0f);
glEnd();// Koniec kreslenia trojuholnika
```

# Writing conventions

- C style
- Constants: starting with GL_
- Defined types: starting with GL
- Functions:



Examples: GL_TRUE, GLfloat, glColor3f(1.0, 1.0, 0.25), gluPerspective(45, 1.25, 0.0, 10.0)

# Parameter types

| Type identifier | Data type | C,C++ data type | OpenGL data type |
|---|---|---|---|
| b | 8-bit integer | signed char | GLbyte |
| s | 16-bit integer | short | GLshort |
| i | 32-bit integer | int, long | GLint, GLsizei |
| f | 32-bit floating point | float | GLfloat, GLclampf |
| d | 64-bit floating point | double | GLdouble, GLclampd |
| ub | 8-bit unsigned number | undigned char | GLubyte, GLboolean |
| us | 16-bit unsigned number | unsigned short | GLushort |
| ui | 32-bit unsigned number | unsigned int or unsigned long | GLuint, GLenum, GLbitfield |

# OpenGL as state machine

- Very important paradigm
- OpenGL - black box accessed by functions (imagine it as class with many public functions and some private functionality)
- State = set of state variables and its current values + other states of system
- OpenGL remains in one state until it is changed with API functions
- Lots of state variables: color, transformation matrix, normal, …

# Preparing OpenGL

- Installing newest graphics card driver
- Choosing programming environment and language (we will use Visual C++)
- Rendering to window – window system dependent feature, not OpenGL feature
- Using auxiliary libraries for system independent development (GLUT, GLEW, …)
- Adding OpenGL (like other)
  - Definition of functions,.. - header files (.h)
  - Implementation – library files (.lib, .dll)
  - Copy .h files, include .lib files

# GLUT

- Fast & easy work with platform dependant features
- Functions for managing OpenGL windows, more windows for OpenGL rendering
- Input events managing, supports more input devices
- Timers and idle programs, pop-up menus
- Generates basic graphics primitives
- http://www.opengl.org/resources/libraries/glut/

# Initialization using GLUT

| Function | Description | Example |
|---|---|---|
| **void glutInit ( int argc, char** argv )** | Glut initialization | glutInit( &argc, argv ) |
| **void glutInitDisplayMode ( int mode )** | Initialization of rendering modes | glutInitDisplayMode(GLUT_RGB \| GLUT_DOUBLE) |
| **void glutInitWindowSize ( int width, int height )** | Setting render window size | glutInitWindowSize(640, 480) |
| **void glutInitWindowPosition ( int x, int y )** | Setting render windows position | glutInitWindowPosition(10, 10) |
| **void glutCreateWindow ( const char *title )** | Creating render window and creating OpenGL state machine | glutCreateWindow(“Render window”) |

# Callbacks initialization

- Callbacks – functions assigned to system events that triggers on given event
- For controlling input and output
- Callbacks for mouse clicks, moves, key strokes
- Callbacks for events when window should be repainted, resized
- Callbacks for timer (called in given time interval) or for idle (called when processor is in idle state)

# GLUT callbacks

**glut ________ Func( __________ )**

| **Part of initialization function** | **Callback function (can be with arbitrary name)** |
|---|---|
| Display | myDisplay( ) |
| Reshape | myReshape( int width,int height ) |
| Mouse | myMouse( int button, int state, int x, int y ) |
| PassiveMotion | myMotion( int x, int y ) |
| Keyboard | myKeyboard( uchar key, int x, int y ) |
| Special | mySpecial( int key, int x, int y ) |
| Timer | myTimer( int id ) |
| Idle | myIdle( ) |

# End of initialization

- **void glutMainLoop(void)**
- GLUT starts infinity loop and is waiting for messages from system
- When message from system arrives, it is transferred to appropriate callback
- Exit from loop: **void exit(int status)**, **void glutLeaveMainLoop(void)**
- Exit from loop must be defined in some callback function, otherwise the loop never ends

# Display callback

- Function that is called every time the window have to be rendered
- Main function for calling OpenGL rendering functions
- The frame rendering is finished by calling **void glutSwapBuffers(void)**
- Usage of double buffering, for flicker-free animation, will be explained later

# GLUT Basic objects

- Solid & wire objects

```
void glutSolidSphere(GLdouble radius, GLint slices, GLint stacks);
void glutWireSphere(GLdouble radius, GLint slices, GLint stacks);
void glutSolidCube(GLdouble size);
void glutWireCube(GLdouble size);
void glutSolidCone(GLdouble base, GLdouble height, GLint slices, GLint stacks);
void glutWireCone(GLdouble base, GLdouble height, GLint slices, GLint stacks);
void glutSolidTorus(GLdouble innerRadius, GLdouble outerRadius, GLint nsides, GLint rings);
void glutWireTorus(GLdouble innerRadius, GLdouble outerRadius, GLint nsides, GLint rings);
void glutSolidDodecahedron(void);
void glutWireDodecahedron(void);
void glutSolidOctahedron(void);
void glutWireOctahedron(void);
void glutSolidTetrahedron(void);
void glutWireTetrahedron(void);
void glutSolidIcosahedron(void);
void glutWireIcosahedron(void);
void glutSolidTeapot(GLdouble size);
void glutWireTeapot(GLdouble size);
```

# The End!

Questions?