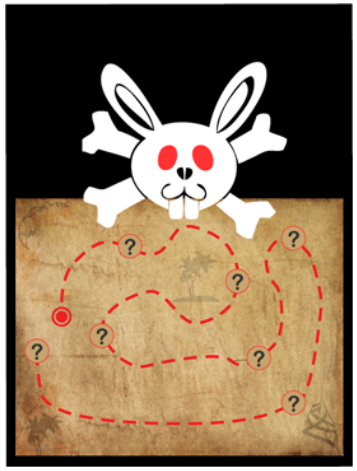
Computer Graphics on Mobile Devices

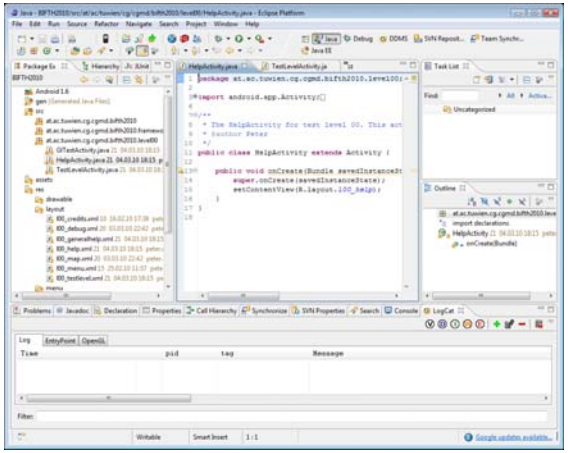
VL SS2010 3.0 ECTS

Peter Rautek

Rückblick

- Motivation
- Vorbesprechung
  - Spiel
  - VL Framework
  - Ablauf
- Android Basics
  - Android Specifics
    - Activity, Layouts, Service, Intent, Permission, etc.
  - Entwicklung mit Eclipse



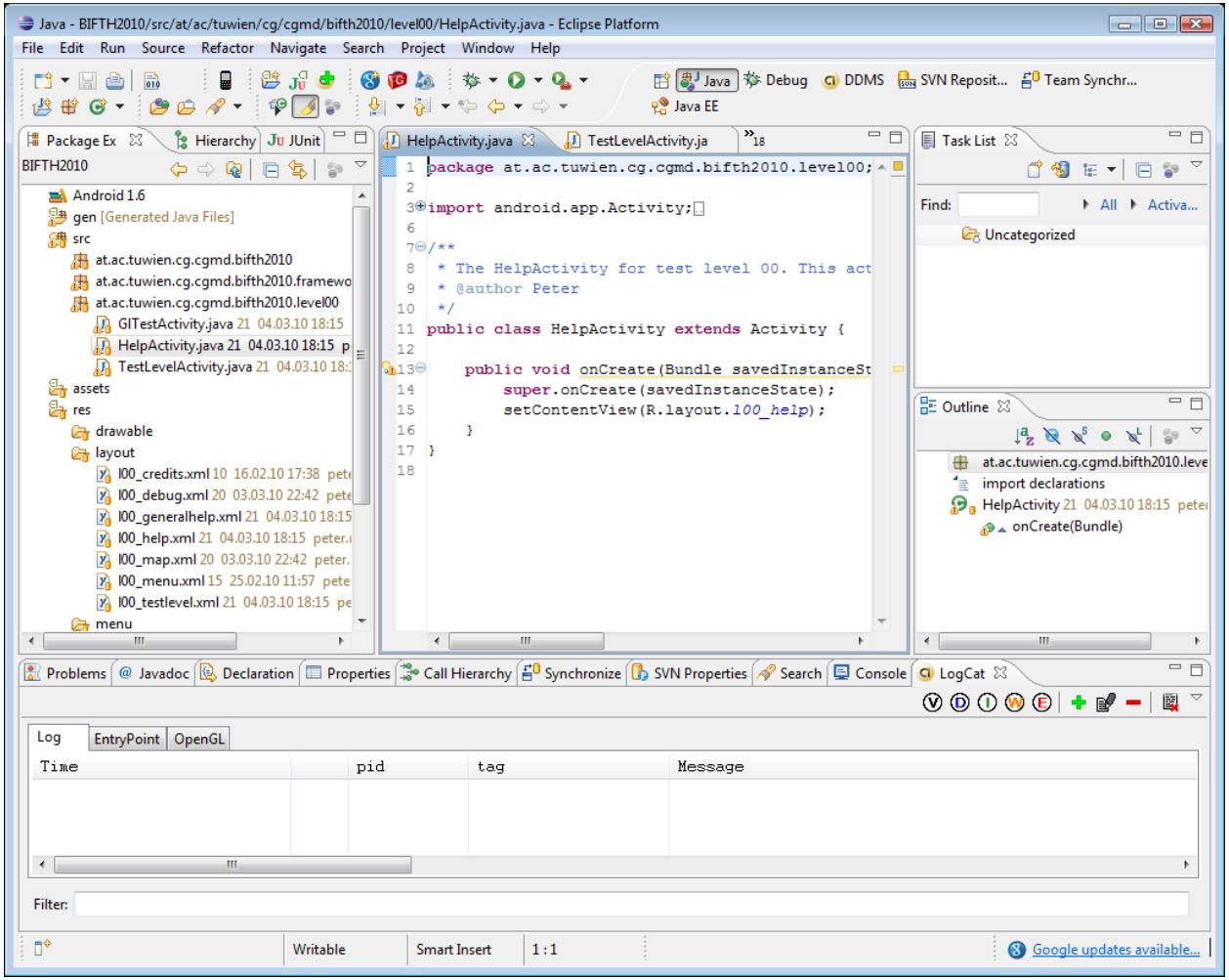


Overview

- Development with Eclipse
  - Overview
  - Demo
- Advanced Android Topics
  - 3D Graphics
    - The Java Way
    - The C/C++ Way
  - Debugging OpenGL
  - Configuration, Resources and Localization
- OpenGL ES
  - History
  - Overview
- Lab Phase II

Development with Eclipse

- Breakpoints
- Variables
- Perspectives
- LogCat
  - Filters
- Callstack
- Subclipse
  - Team item
  - Browser
  - Import



OpenGL „The Java Way“

- Implement the lifecycle methods in Activity
  - onPause
  - onResume
- Extend the GLSurfaceView
  - Call the setRenderer method in GLSurfaceView
- Implement the GLSurfaceView.Renderer
  - onSurfaceCreated
  - onSurfaceChanged
  - onDrawFrame

GLSurfaceView

- Manages memory (surface), composited into the view system
- Manages OpenGL rendering to the surface
- Requires a implementation of the GLSurfaceView.Renderer interface
- Rendering runs in own thread
- On-demand vs. continuous rendering
- OpenGL debugging
- Default: 16-bit R5G6B5, 16-bit depth buffer

Renderer

- Runs in separate thread
- onSurfaceCreated
  - Called when
    - The activity is started
    - The OpenGL context was destroyed and recreated
  - Load Textures
- onSurfaceChanged
  - Called when size/orientation changes
- onDrawFrame

OpenGL „The Java Way“

- Pro
  - Very easy to implement
- Con
  - Lower performance
  - Garbage collection can lead to hickups
  - Threading problem
    - Game engine loop runs in render thread
    - Thread synchronization
- Check out the API demos!

OpenGL “The C/C++ Way”

- Native development kit (NDK)
  - Native implementation
  - Generate make files
  - Build shared library
  - Use SDK tools to build application
    - Load shared library from Java
      - static {
        System.loadLibrary(„mylibrary“);
        }
    - Declare native methods
      - private static native void nativeMyFunction();
    - Call native functions from GLSurfaceView
- Two demo apps in the NDK
  - san-angeles: OpenGL 1.x
  - hello-gl2: OpenGL 2.x
    - Android 2.0 and higher
    - Not running in emulator

OpenGL “The C/C++ Way”

- Pro
  - Performance
- Con
  - Harder to implement
  - Cumbersome development
  - Debugging

- Check out the NDK demos!

Debugging OpenGL

- Context Wrapper (see AboutActivity)

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    mGLSurfaceView.setGLWrapper(new GLWrapper(){
          private Writer logger = new Writer(){
          //implementing the writer interface
          };

          @Override
          public GL wrap(GL gl) {
              return GLDebugHelper.wrap(
                    gl,
                    GLDebugHelper.CONFIG_CHECK_GL_ERROR|GLDebugHelper.CONFIG_LOG_ARGUMENT_NAMES,
                    logger);
          }
    });
}
```


Android Development

- Multithreading
  - User interface (Activity)
  - Rendering (GLSurfaceView.Renderer)
  - Your own threads
- Synchronization
  - Necessary to avoid concurrency problems
  - Handler class provides message queue
  - Example: Display frame rate

Performance

- Always know your frame rate!
- Log (+easy, -floods your Log, -bad visibility)
- Onscreen
  - OpenGL Overlay
    - Write text to bitmap
    - Render as texture
  - GUI Overlay
    - Use FrameLayout
    - Overlay TextView

GUI Overlay

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    // set the layout
    setContentView(R.layout.l00_map);

    // get the frame layout
    FrameLayout mFrameLayout = (FrameLayout) findViewById(R.id.l00_FrameLayout);
    // get the fps text view
    mFpsText = (TextView) findViewById(R.id.l00_TextViewFps);
    // add the surface view
    mMapView = new MapView(this);
    mFrameLayout.addView(mMapView);

    // create a timer
    Timer mFpsUpdateTimer = new Timer();
    mFpsUpdateTimer.schedule(new TimerTask(){

        @Override
        public void run() {

            // get the fps from the surface view
            float fps = mMapView.getFps();
            mFpsString = "fps: " + Float.toString(fps);
            // tell the ui thread to update the fps
            handleUIChanges.sendEmptyMessage(0);
        }
    }, 1000, 1000);
}

// setup a handler in the ui thread
private Handler handleUIChanges = new Handler(){
    @Override
    public void handleMessage(Message msg) {
        super.handleMessage(msg);
        // set the text of the text view
        mFpsText.setText(mFpsString);
    }
};
```

Android Development

- How to add sources
  - No short answer to this
  - <sdk>\platforms\android-1.6\
  - Create sources -> <sdk>\platforms\android-1.6\sources
  - Copy sources to this folder
  - Additional info and sources for other versions: http://code.google.com/p/android/issues/detail?id=979

Resources and Configurations

- Configurations depend on
  - Device
    - Screen size
    - Keyboard
  - User preference
    - Language
  - Situation
    - Orientation
- Solution
  - Multiple resources in one apk

Resources and Configurations

- Change of configuration
  - Destroying activity
  - Restarting activity with new configuration
- Resource folders
  - Loading of resource in appropriate folder
  - Depending on

Example: Different Languages

- Localization
  - res/values/string.xml
    The default must contain all strings
  - res/values-de/string.xml
    The de folder may contain localized strings for German version
  - res/values-fr/string.xml
    The fr folder may contain localized strings for French version

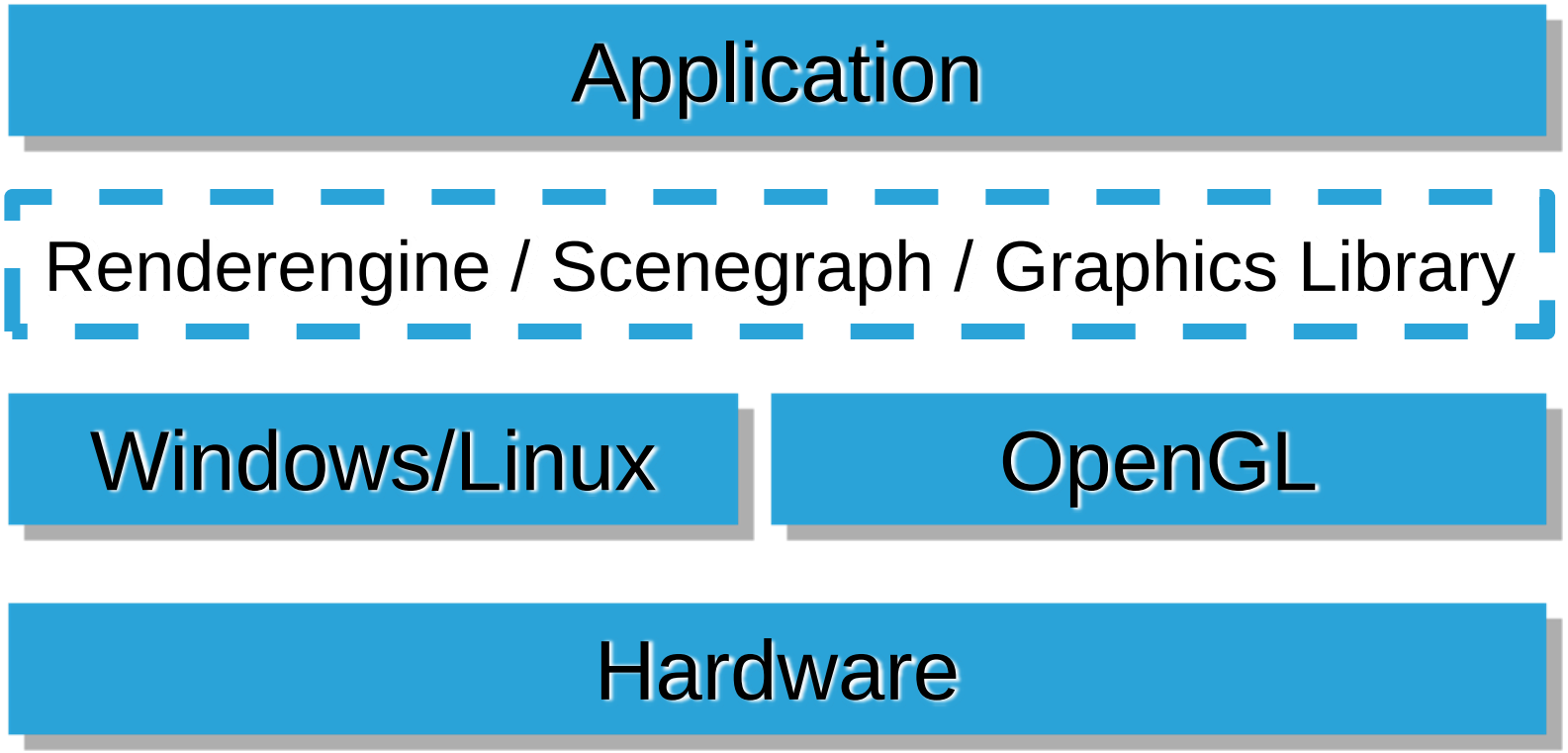
Example: Switching Orientation

- Orientation
  - res/layout-port/main.xml
  - res/layout-land/main.xml
- Handle orientation change
  - Fixed orientation (no change occurs)
    - Specifiy in AndroidManifest file
  - Custom behavior
    - Specify in AndroidManifest file
    - Implement onConfigurationChanged()

Overview

- Development with Eclipse
  - Overview
  - Demo
- Advanced Android Topics
  - 3D Graphics
    - The Java Way
    - The C/C++ Way
  - Debugging OpenGL
  - Configuration, Resources and Localization
- OpenGL ES
  - History
  - Overview
- Lab Phase II

OpenGL – Who, What, and Most Importantly Why?



OpenGL Specification

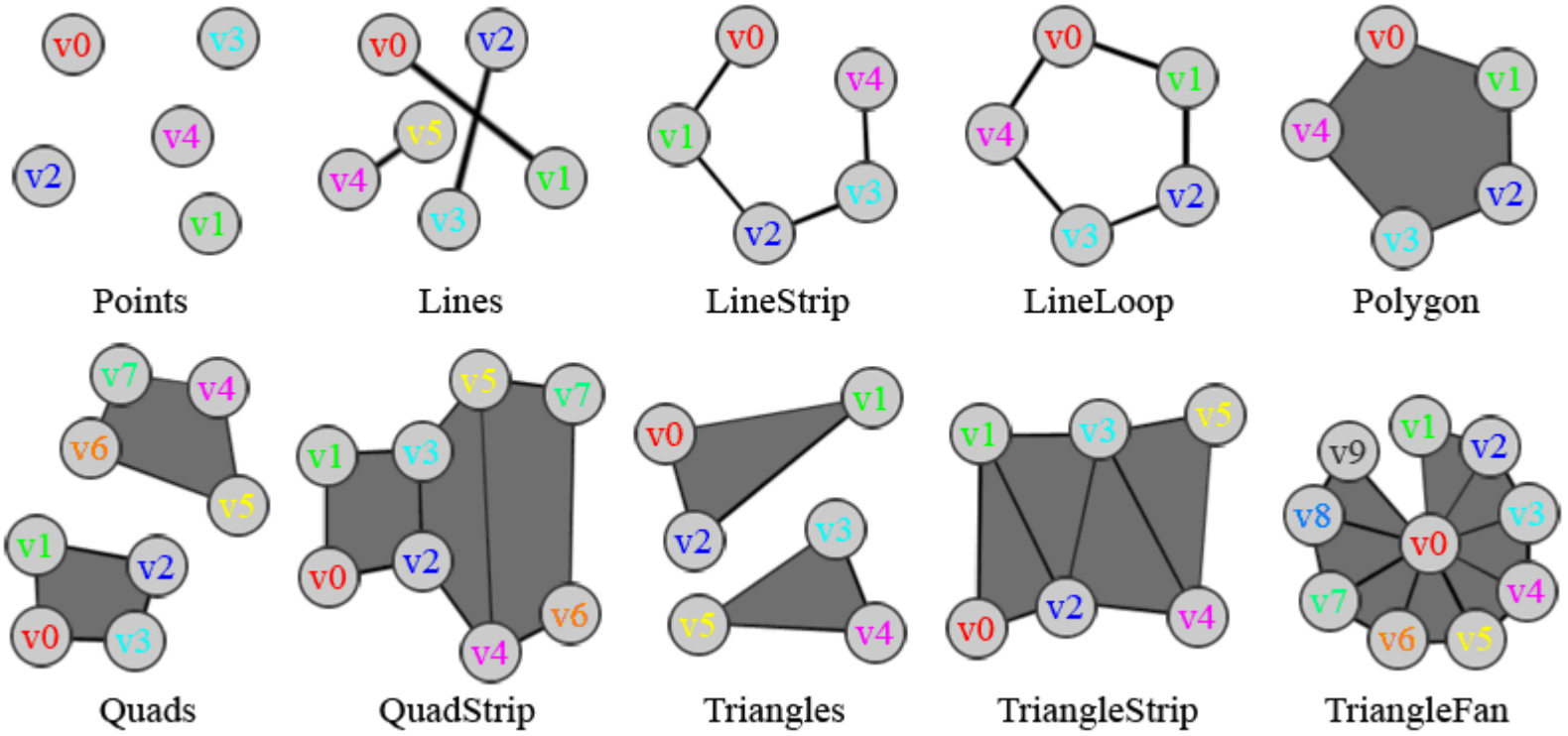
- Since 2006 under control of Khronos Group
  - Non profit consortium
  - Open standards
  - Royalty free
- Working Groups
  - OpenGL, OpenGL ES, OpenCL, COLLADA, OpenVG, OpenSL ES, EGL, WebGL, etc.
- Members
  - AMD, Apple Inc., ARM Holdings, Creative Labs, id Software, Ericsson, Intel Corporation, Motorola, Nokia, Nvidia, Samsung Electronics, Sony Computer Entertainment, Sun Microsystems, Texas Instruments, etc.
- Links: www.khronos.org, www.opengl.org

OpenGL Design Goals

- Platform independent
- Language independent
- Consistency
  - Conformance tests and required verification
  - Not pixel exact, but invariant across passes
- Complete implementations
  - Missing features emulated in software
- Clean interface
  - State machine
  - Most states are orthogonal
- Extensibility
  - Favors innovation

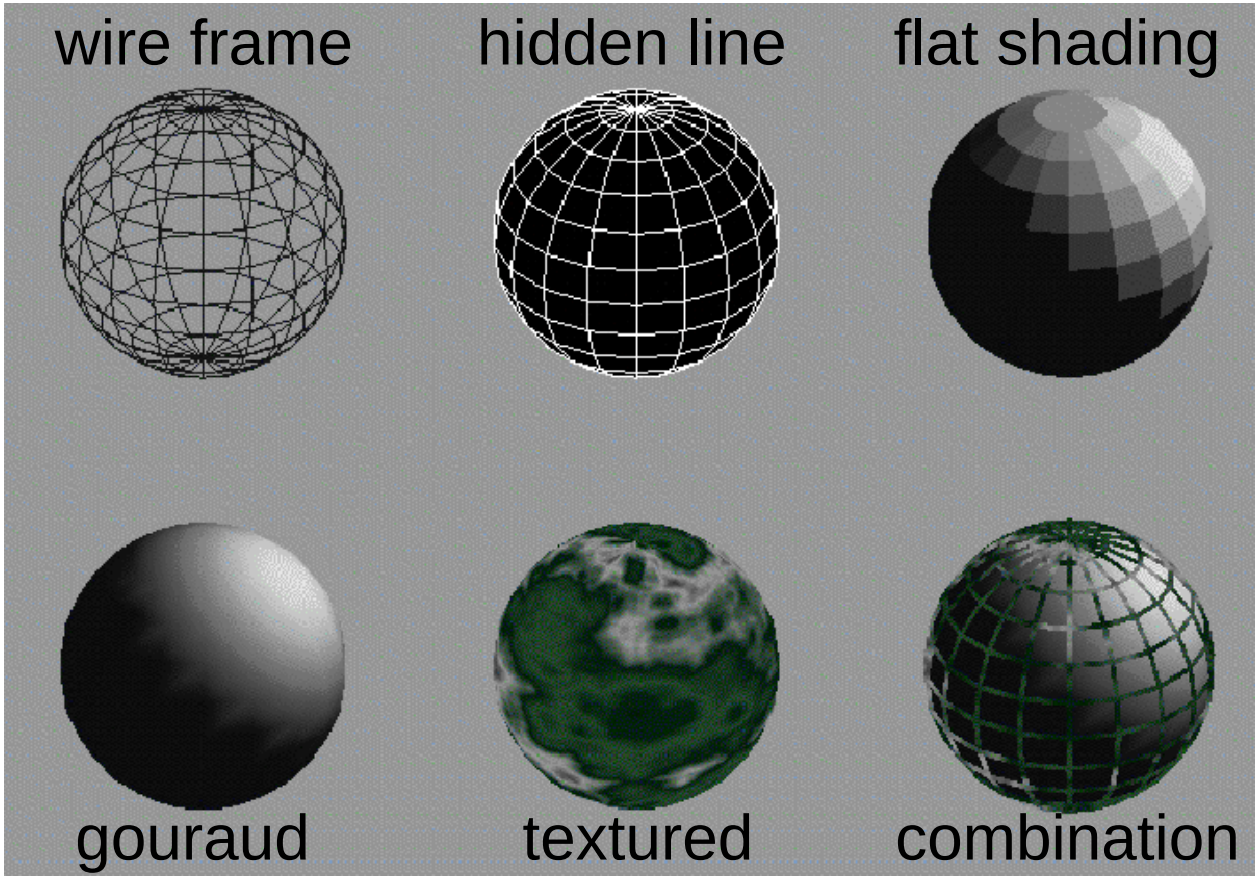
OpenGL in a Nutshell

- Small number of primitives
- Defined by vertices



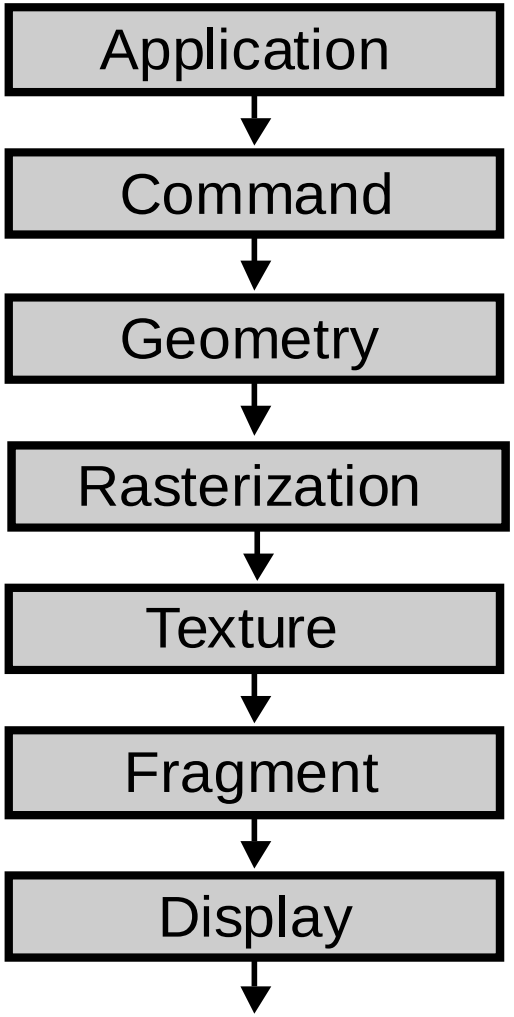
OpenGL in a Nutshell

- Shading



Graphics Pipeline

- Application
  - Scene traversal
  - Object, and camera movement
  - Animation
  - Occlusion/Visibility Culling
  - Level of Detail (LoD) selection
- Geometry
  - Transformation (model, world, view)
  - View Projection
  - Culling (e.g., back-face)
  - Perspective Division
  - Clipping
  - Screen space transformation
- Triangle Setup
- Rasterization
- Texturing
- Fragment
  - Shading
  - Depth Buffering



OpenGL vs. OpenGL ES

- Specification
  - OpenGL ES 1.0 written against OpenGL 1.3
  - OpenGL ES 1.1 written against OpenGL 1.5
  - OpenGL ES 2.0 written against OpenGL 2.0
- Differences
  - Single vs. Double
  - Fixed vs. Floating
  - No glBegin(), glEnd(), glVertex()
  - No display lists
  - etc.

Example

- ✓ Supported
- – Not supported
- † Fixed point
- ◊ Single precision
- ‡ Other restrictions
- ∗ Additional enumerant

The primitives: POINTS, LINES, LINE_STRIP, LINE_LOOP, TRIANGLES, TRIANGLE_STRIP, and TRIANGLE_FAN are supported; the primitives: QUADS, QUAD_STRIP, and POLYGON are not supported.

Color index rendering is not supported. Edge flags are not supported.

OpenGL 1.3	Common	Common-Lite
Begin(enum mode)	–	–
End(void)	–	–
EdgeFlag[v](T flag)	–	–

■ The Begin/End paradigm, while convenient and efficient, leads to a large number of commands that need to be implemented. Correct implementation also involves suppression of commands that are not legal between Begin and End. Tracking this state creates an additional burden on the implementation. Vertex arrays, arguably can be implemented more efficiently since they present all of the primitive data in a single function call. Edge flags are not included, as they are only used when drawing polygons as outlines and support for **PolygonMode** has not been included.

Quads and polygons are eliminated since they can be readily emulated with triangles and it reduces an ambiguity with respect to decomposition of these primitives to triangles, since it is entirely left to the application. Elimination of quads and polygons removes special cases for line mode drawing requiring edge flags (should **PolygonMode** be re-instated). □

2.7 Vertex Specification

The Common profile does not include the concept of Begin and End. Vertices are specified using vertex arrays exclusively. Only float, short, and ubyte coordinate and component types are supported with the exception of ubyte rather than short color components. There is limited support for specifying the current color, normal, and texture coordinate using the fixed-point or floating-point forms of the commands **Color4**, **Normal3**, and **MultiTexCoord4**.

Multitexture texture coordinates are supported, though only a single texture unit needs to be supported.

OpenGL 1.3	Common	Common-Lite
Vertex{234}{**sifd**}[**v**](T coords)	–	–
Normal3f(float coords)	✓	†
Normal3{**bsifd**}[**v**](T coords)	–	–
TexCoord{1234}{**sifd**}[**v**](T coords)	–	–
MultiTexCoord4f(enum texture, float coords)	✓	†
MultiTexCoord123{**sifd**}[**v**](enum texture, T coords)	–	–
Color4f(float components)	✓	†
Color{34}{**bsifd ub us ui**}[**v**](T components)	–	–
Index{**sifd ub**}[**v**](T components)	–	–

■ A handful of *fine grain* commands **Color**, **Normal**, **MultiTexCoord** are included so that per-primitive attributes can be set. For each command, the most general form of the floating-point version of the

Short History of OpenGL (ES)

1991	OpenGL ARB created
1992	OpenGL 1.0 (June 30)
1995	OpenGL 1.1
1996	OpenGL specification made public
1998	OpenGL 1.2
2000	OpenGL goes open source
2001	OpenGL 1.3
2001	OpenGL ES 1.0
2002	OpenGL 1.4
2003	OpenGL 1.5
2003	OpenGL ES 1.1
2004	OpenGL 2.0 (shaders)
2004	OpenGL ES 2.0
2008	OpenGL 3.0
2010	OpenGL 4.0

OpenGL ES 1.0 vs. 2.0

- Main novelty: shading language GLSL
- Vertex and fragment shaders
  - Replace fixed functionality
- Shader: high-level language (C-like)
- OpenGL driver: compiler and linker for shaders
- Vertex-, texture coordinates etc.: abstract input values to shader function
- Arbitrary calculations possible

Overview

- Development with Eclipse
  - Overview
  - Demo
- Advanced Android Topics
  - 3D Graphics
    - The Java Way
    - The C/C++ Way
  - Debugging OpenGL
  - Configuration, Resources and Localization
- OpenGL ES
  - History
  - Overview
- Lab Phase II

Phase II

- Targeting und Localization
  - Language, screen size
- Implement a stub
  - LevelActivity, GLSurfaceView, Renderer
- Framework Integration
  - Icon, return values, documentation
- Implement your level
  - Functions, game play, first hardware tests
- Detailed instructions online

Ankündigung - Nächste VO

- Anton Pirker
- Cross-Platform Development
- Laptop mitnehmen
  - Eclipse
  - Java
  - Android

Where to Start - Links

- Graphics Pipeline Overview (by Dave Salvator)
  - http://www.extremetech.com/article2/0,2845,9722,00.asp
  - Many, many more – google for it!
- Open GL ES Specifications:
  - http://www.khronos.org/opengles/spec/
- Android
  - GLSurfaceView
    http://developer.android.com/reference/android/opengl/GLSurfaceView.html
  - Resources and Internationalization
    http://developer.android.com/guide/topics/resources/resources-i18n.html
  - NDK
    http://developer.android.com/sdk/ndk/index.html
  - API demos
    http://developer.android.com/resources/samples/ApiDemos/src/com/example/android/apis/graphics/index.html

Danke für Ihre Aufmerksamkeit

Fragen?