



penGL ES

CSCI 4239/5239

**Advanced Computer Graphics
Spring 2021**

OpenGL ES



- OpenGL for Embedded Systems
 - Phones
 - Game consoles
 - Appliances
 - Avionics
 - Subsystems (e.g. browsers)
 - ...
- Cross-platform, open, standard

What is it?

- OpenGL adapted for Embedded Systems
 - Less capable hardware
 - Limited memory
 - Limited processing power
 - Lower clock frequencies
 - Lower power consumption
 - Less heat dissipation
- Same familiar API
 - Subset of full OpenGL API
 - Powerful 3D graphics in your pocket

OpenGL ES Advantages

- Standard and Royalty Free
- Small footprint
- Low power consumption
- Seamless hardware acceleration
- Extensible and evolving
- Easy to use
- Well documented

Current Applications

- Mobile devices
 - iPhone/iPod/iPad
 - Android
- WebGL
 - Chrome, Firefox, Safari, Opera, ...
- Embedded systems
 - 3D displays

OpenGL ES 1.1

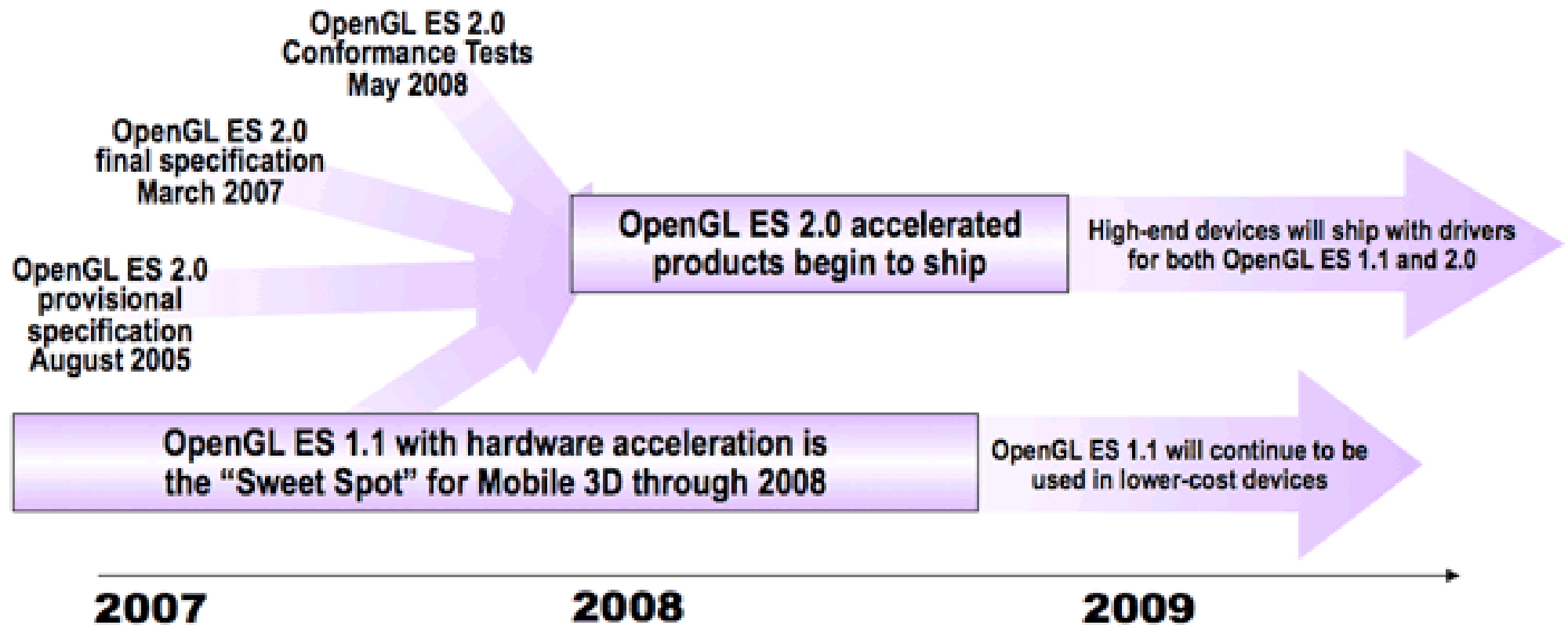
- Feature upgrade from OpenGL ES 1.0
- Defined relative to OpenGL 1.5
- Fixed pipeline (no shaders)
- Removes some functionality
 - No glBegin() ... glEnd()
 - Replaced with glDrawArrays() & glDrawElements()
 - No GL_QUAD or GL_POLYGON
 - No display lists
- Still provides lighting, textures, etc.

OpenGL ES 2.0 and later

- **Not backwards compatible with ES 1.1**
- Defined relative to OpenGL 2.0
- Shaders only (no fixed pipeline)
 - No lighting except in shaders
 - Textures only in shaders
- Removes transformation functions
 - No `glRotate()` `glScale()` `glTranslate()`
- OpenGL ES 3.0 adds feature upgrades

OpenGL ES Evolution

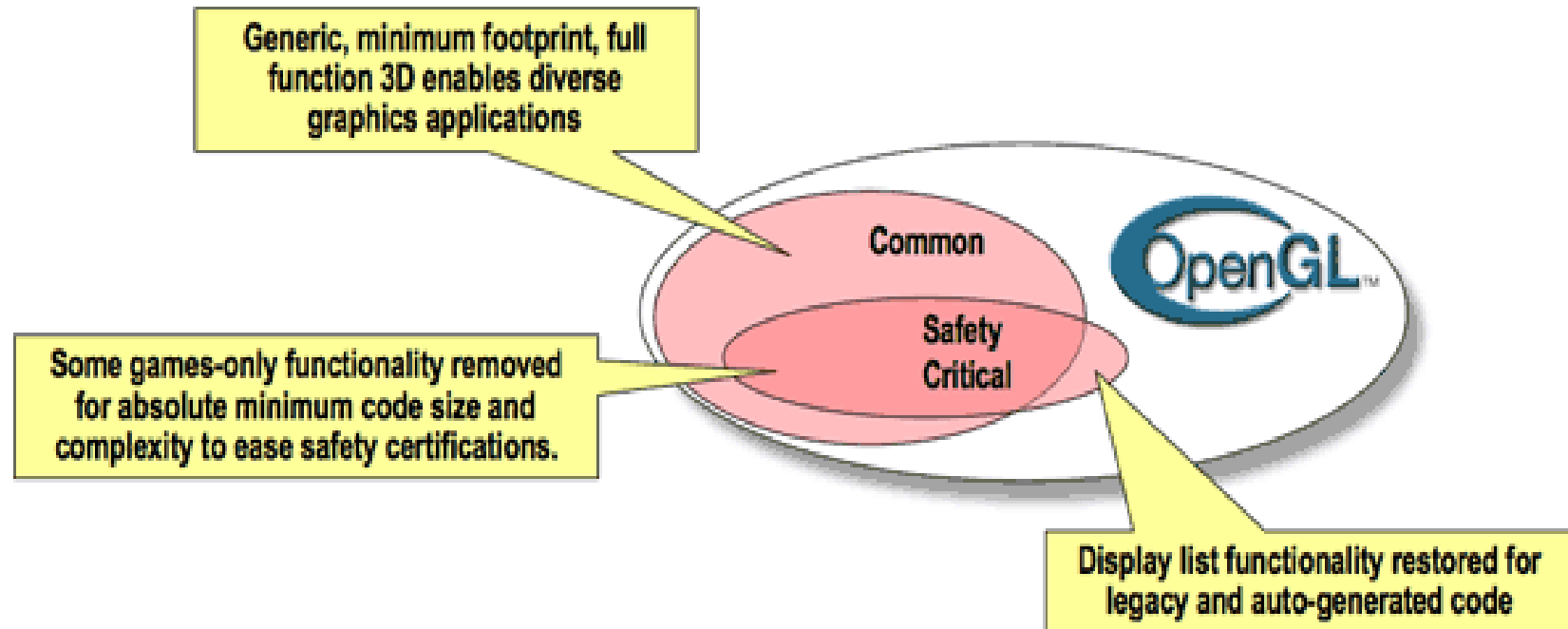
- **OpenGL ES 2.0 silicon implementations now shipping**
 - Shader-based graphics comes to mobile
 - Conformance tests shipping in May 2008
- **Listening carefully to implementation and developer feedback**
 - The determine next-generation requirements



OpenGL SC



- OpenGL for Safety Critical applications
 - Avionics
 - Automotive
 - Industrial
 - Medical
 - Military



OpenGL SC Features

- SC 1.0 starts with OpenGL ES 1.0
 - Adds back some features
 - Begin/End
 - Display Lists
 - Some raster ops
 - Anti-aliasing
 - Removes some features
 - Fog
- SC 2.0 starts with OpenGL ES 2.0

WebGL

- OpenGL ES 2.0 for the web
- Extends Javascript
- Operates on HTML5 canvas element
- Prohibits client side arrays
 - All vertex, normal, color, ... must be stored in Vertex Buffer Object on video card
- Becoming more mainstream
 - Still a work in progress

WebGL Platforms

- Supported by most browsers
 - Chromium
 - Firefox
 - Safari
 - Opera
 - Explorer/Edge/...
- Update to recent version
 - Local file access workarounds

Apple iOS Devices

- Supports OpenGL ES 1.1 and 2.0
- User interface is Objective C
 - Links to C and C++ code
- Develop with Xcode on Mac only
- Emulator for all devices
 - Slower than native devices
 - Almost perfect emulation
- Apple is replacing OpenGL with Metal

Getting iOS Tools

- Download Xcode from Apple
 - many GB download
- Provides compiler, frameworks, etc
- Create project in Xcode
- Select target iPhone/iPad
- Emulator launched on run
- Get command line tools also

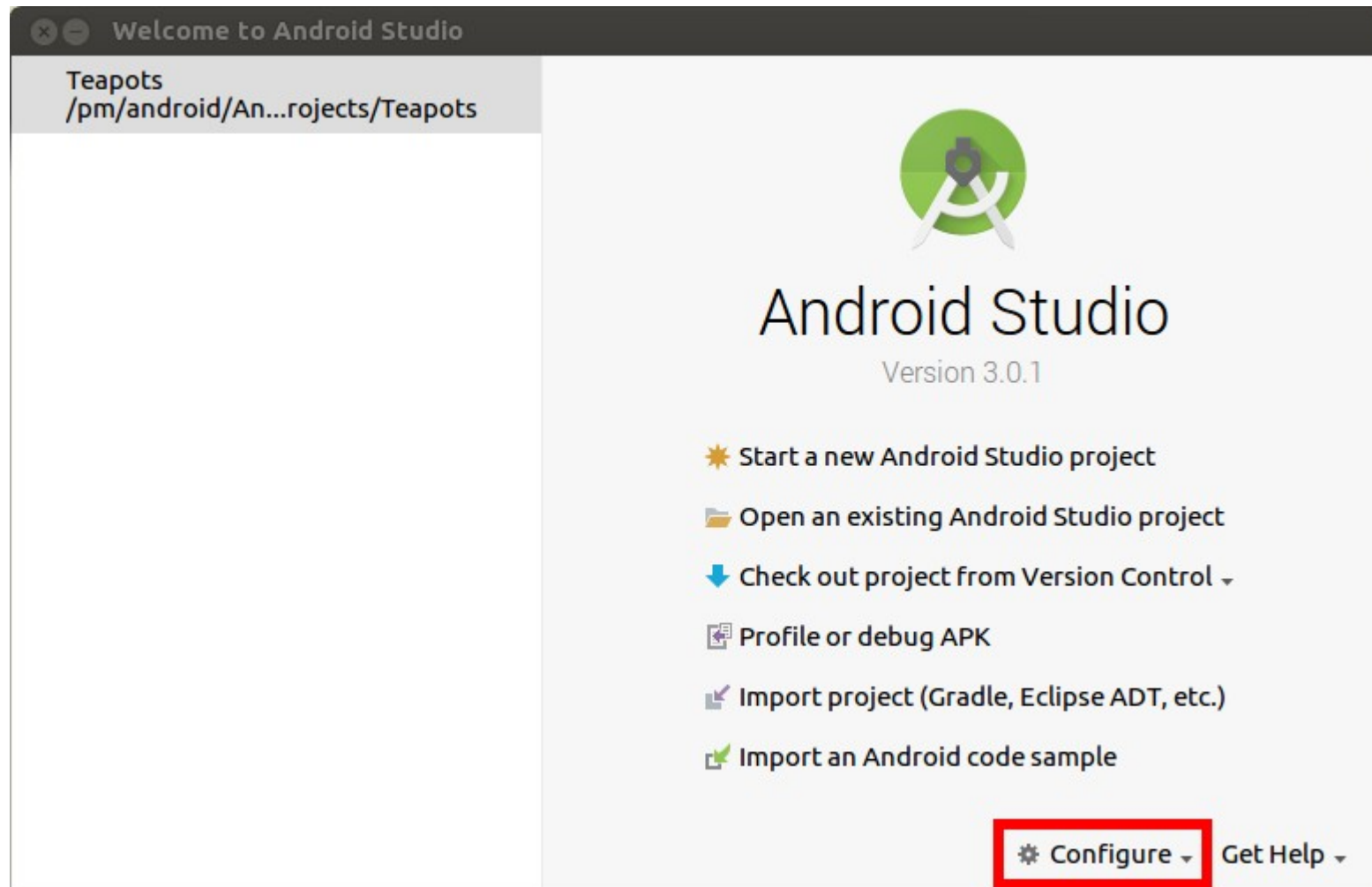
Android Devices

- Supports OpenGL ES 1.1 or 2.0
 - Higher end devices support 1.1 AND 2.0
 - Low end devices may only do 1.1
- User interface is Java
 - Link to C/C++ code with JNI
- Develop with NDK
- Emulator for phones and tablets
 - Slower than native devices
 - Hardware acceleration much improved

Android Tools

- Download Android Studio
 - <https://developer.android.com/studio/>
 - About 750MB ZIP file
- Unzip and find studio.sh or studio.exe
- Use Configure to download SDK, NDK
- Add Android tools to PATH
 -/SDK/tools
 -/SDK/platform-tools
 -/SDK/ndk-bundle
- Create AVDs

Initial SDK Configuration



Select and install SDK Tools

Default Settings

Appearance & Behavior > System Settings > Android SDK

Manager for the Android SDK and Tools used by Android Studio

Android SDK Location: [Edit](#)

SDK Platforms: **SDK Tools** SDK Update Sites

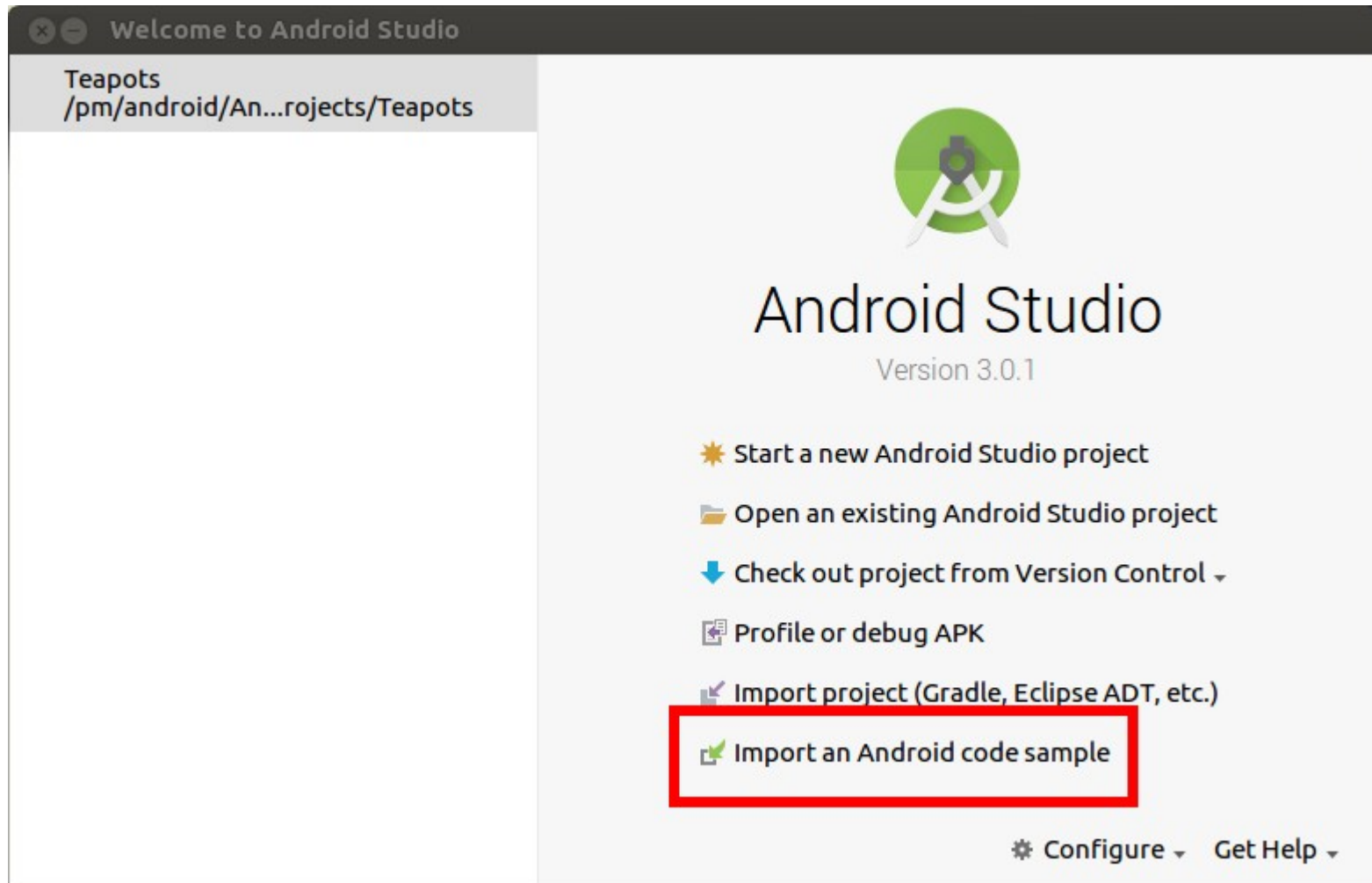
Below are the available SDK developer tools. Once installed, Android Studio will automatically check for updates. Check "show package details" to display available versions of an SDK Tool.

Name	Version	Status
<input checked="" type="checkbox"/> Android SDK Build-Tools		Installed
<input type="checkbox"/> Android Studio Debugging tools		Not Installed
<input checked="" type="checkbox"/> CMake		Installed
<input type="checkbox"/> LLD		Not Installed
<input type="checkbox"/> Android Auto API Simulators	1	Not installed
<input type="checkbox"/> Android Auto Desktop Head Unit emulator	1.1	Not installed
<input checked="" type="checkbox"/> Android Emulator	27.1.7	Installed
<input checked="" type="checkbox"/> Android SDK Platform-Tools	27.0.1	Installed
<input checked="" type="checkbox"/> Android SDK Tools	26.1.1	Installed
<input type="checkbox"/> Documentation for Android SDK	1	Not installed
<input type="checkbox"/> Google Play APK Expansion library	1	Not installed
<input type="checkbox"/> Google Play Licensing Library	1	Not installed
<input type="checkbox"/> Google Play services	46	Not installed
<input type="checkbox"/> Google Web Driver	2	Not installed
<input type="checkbox"/> Instant Apps Development SDK	1.1.0	Not installed
<input checked="" type="checkbox"/> NDK	16.1.4479499	Installed
<input type="checkbox"/> Support Repository		
<input type="checkbox"/> ConstraintLayout for Android		Update Available: 1
<input type="checkbox"/> Solver for ConstraintLayout		Update Available: 1
<input checked="" type="checkbox"/> Android Support Repository	47.0.0	Installed
<input checked="" type="checkbox"/> Google Repository	58	Installed

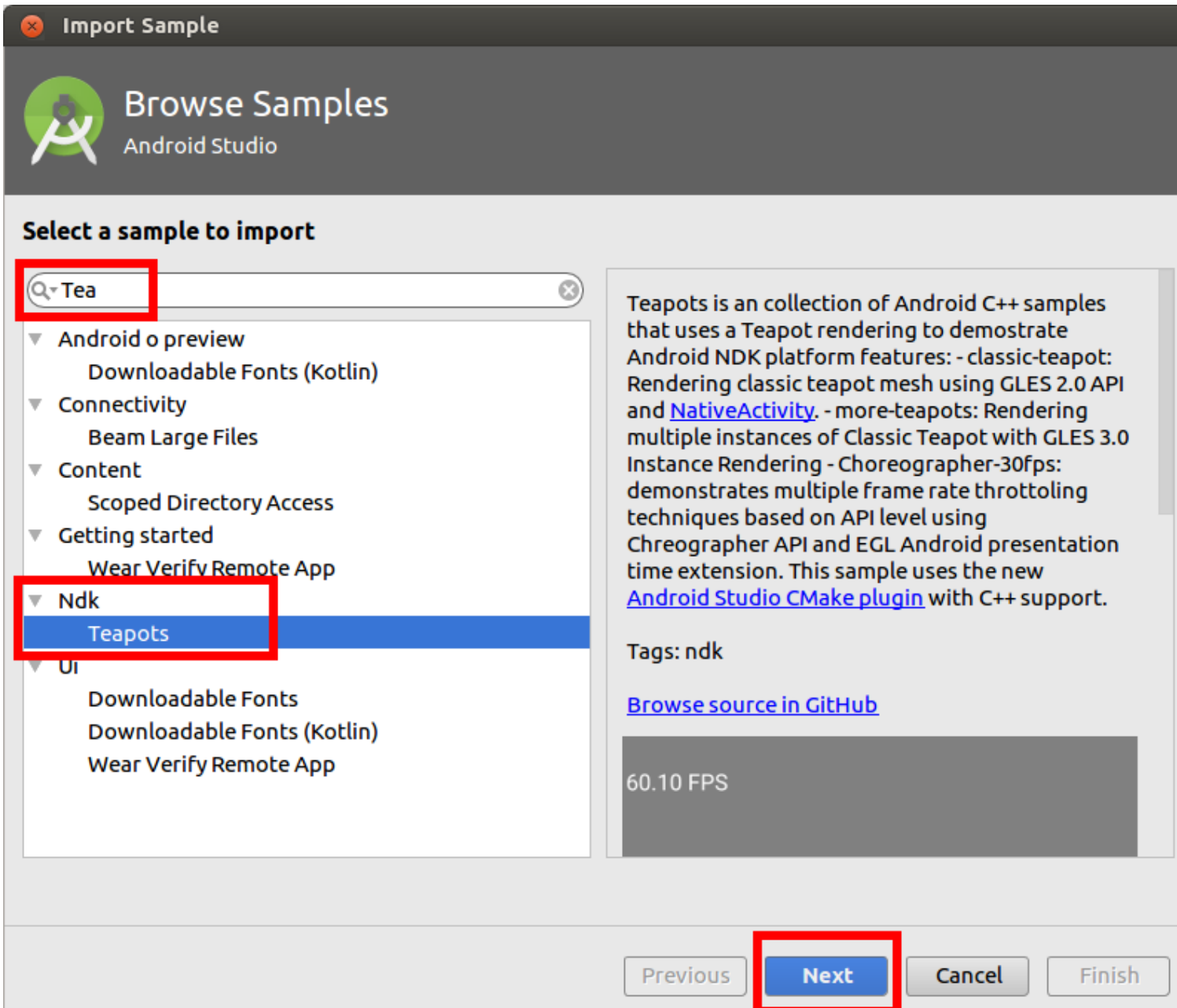
Show Package Details

OK **Cancel** **Apply** **Help**

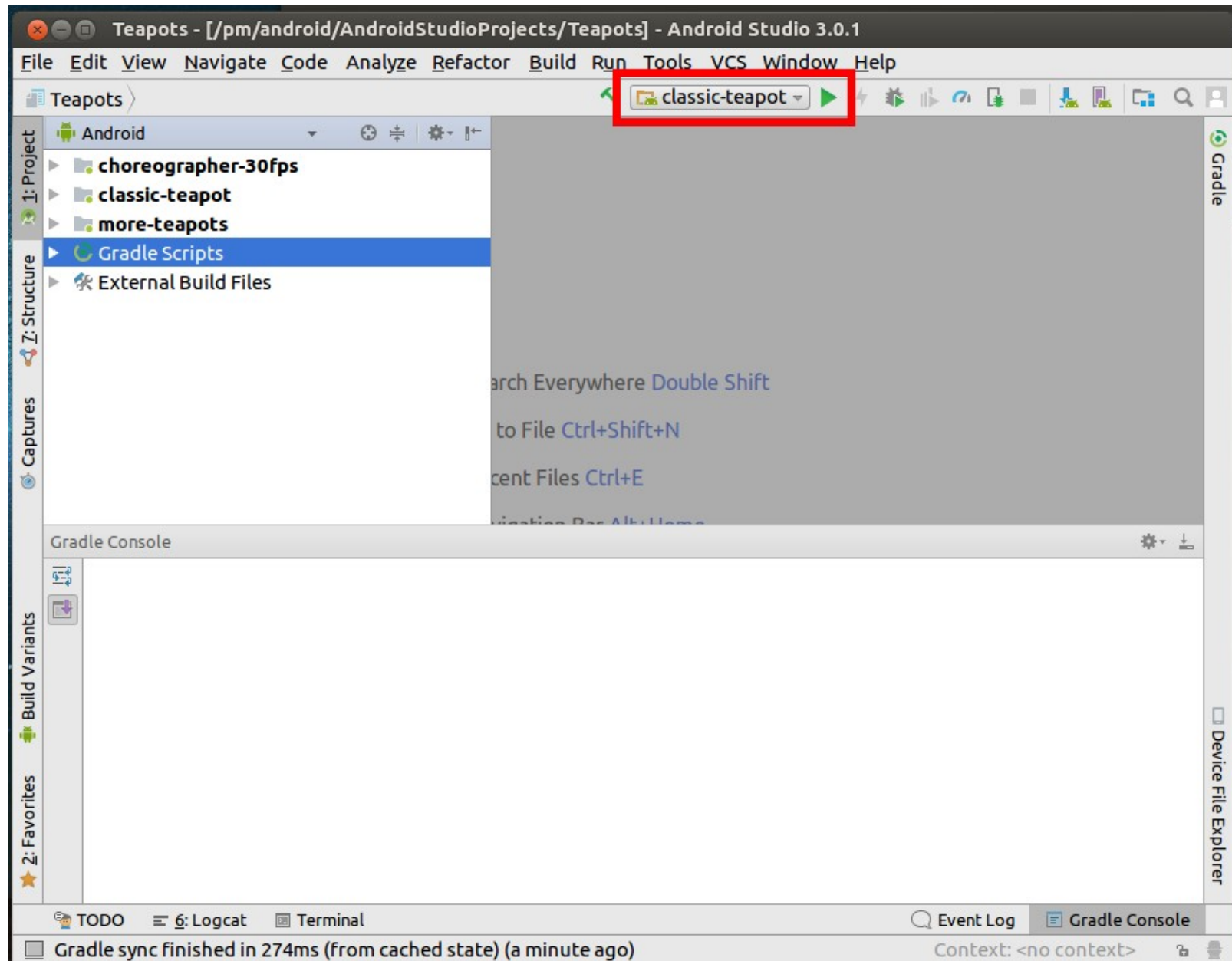
Import Code Example



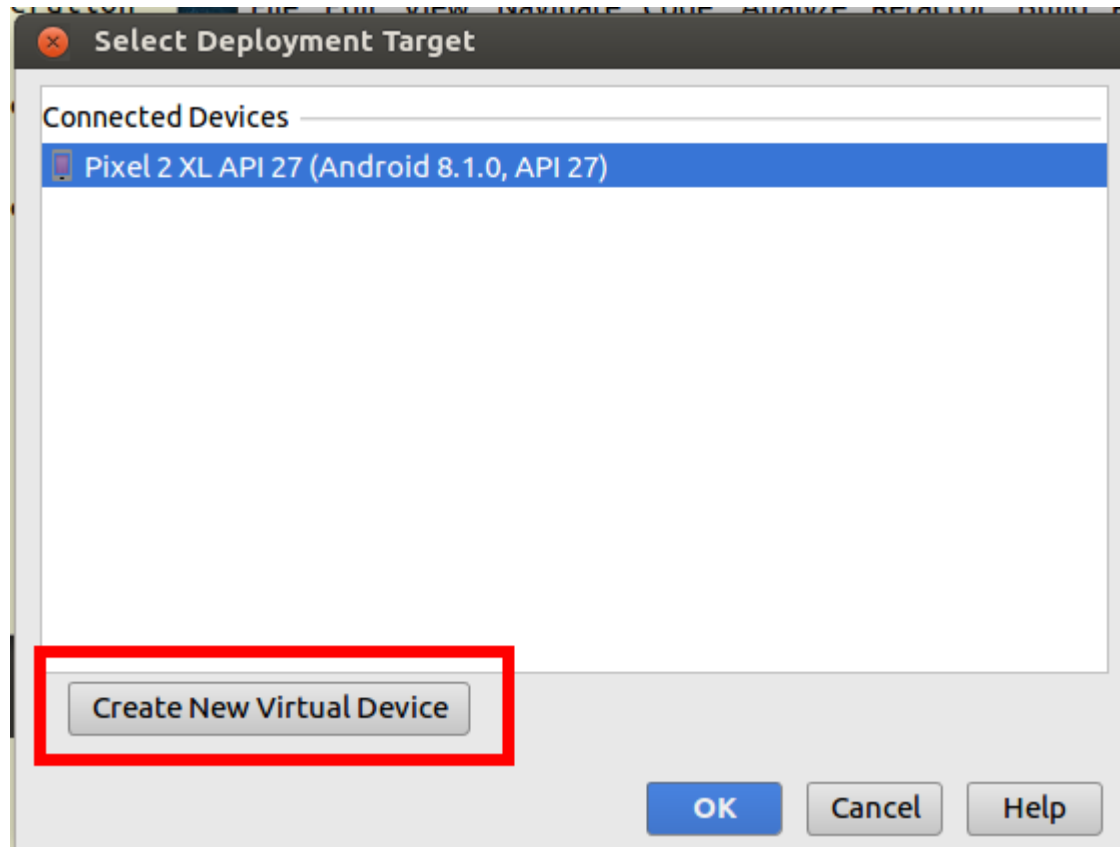
Teapots NDK/OpenGL ES 2.0



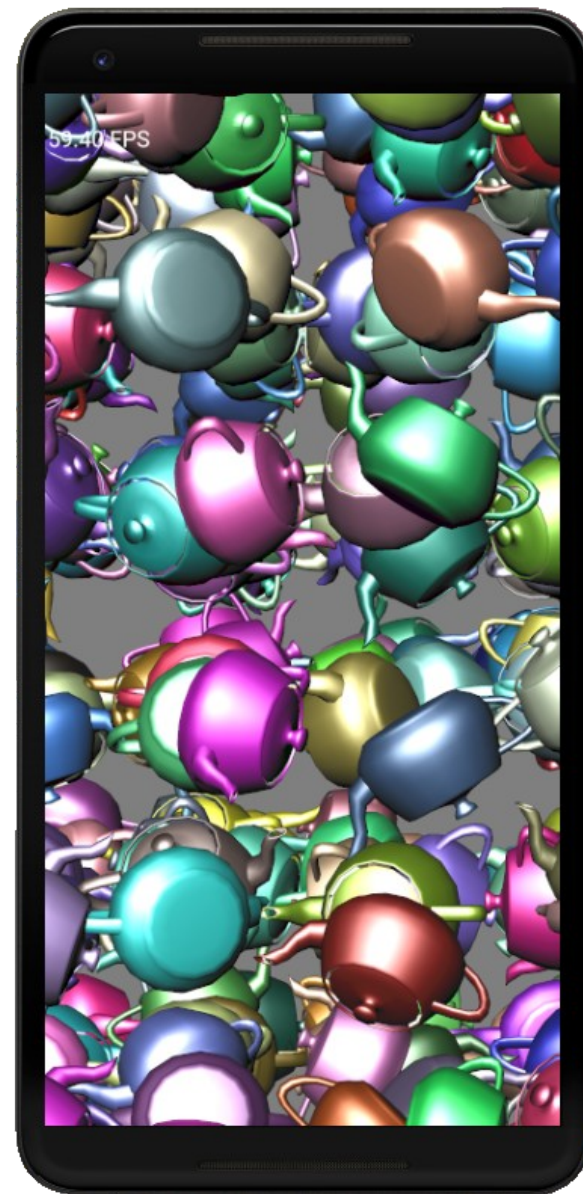
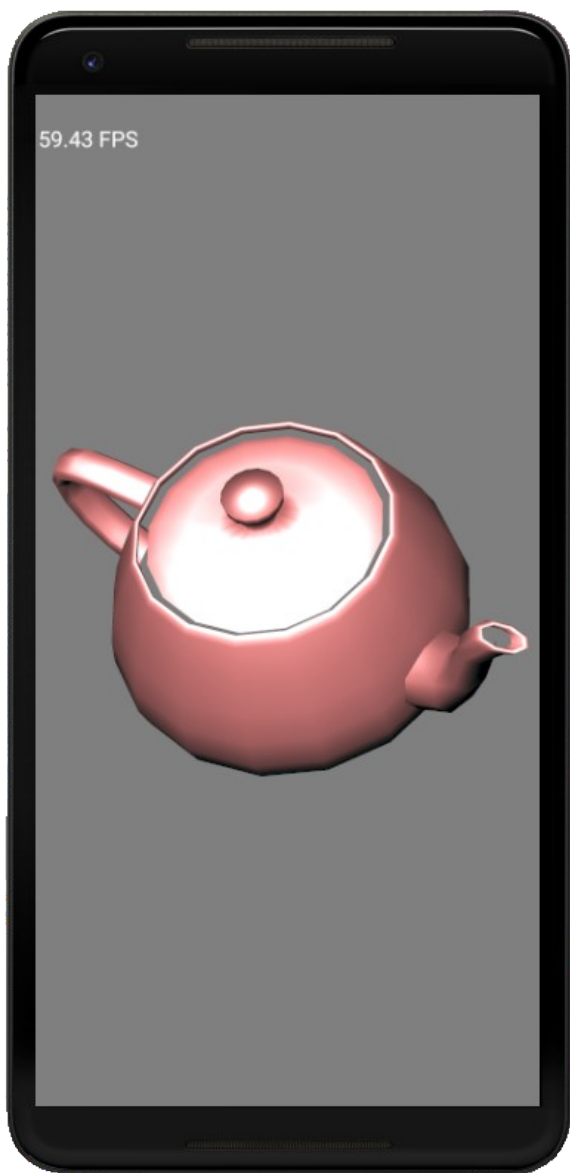
Select Target, Build and Run



Create Virtual Device using Wizard



teapot, textured-teapot and more-teapots



Portable OpenGL ES Code

- Write the bulk of the code in C++
 - OpenGL ES 1.1 will run on all devices
 - OpenGL ES 2.0 will run on higher end devices
- Write minimal code in interface language
 - Objective C – link to C/C++
 - Java – call C/C++ using JNI
- Qt 5 for iOS/Qt 5 for Android
 - Later builds are better

Assignment 5

- Create a scene that can be viewed in 3D using WebGL or IOS or Android
 - Must support lighting and textures
 - Objects must be created in code
 - I want you to get some experience using vertex buffer objects
 - WebGL may use mat4.js or CanvasMatrix library, but NOT Three.js or similar high level libraries
- Explore features like buttons