OpenGL ES CSCI 4239/5239 Advanced Computer Graphics Spring 2025





- 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, ...
- Embeded 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

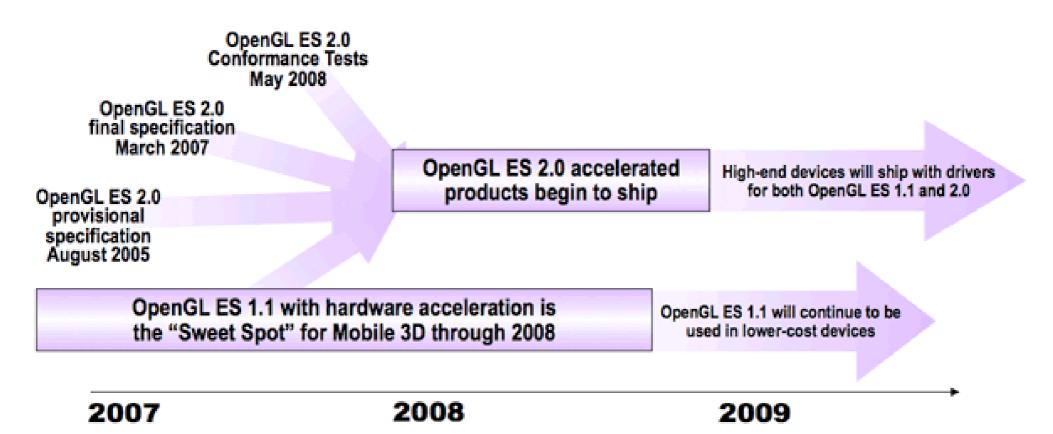
OpenGL ES Evolution

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

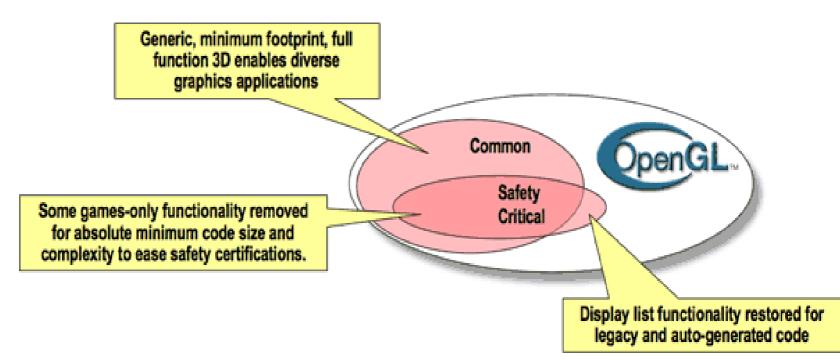
- The determine next-generation requirements



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

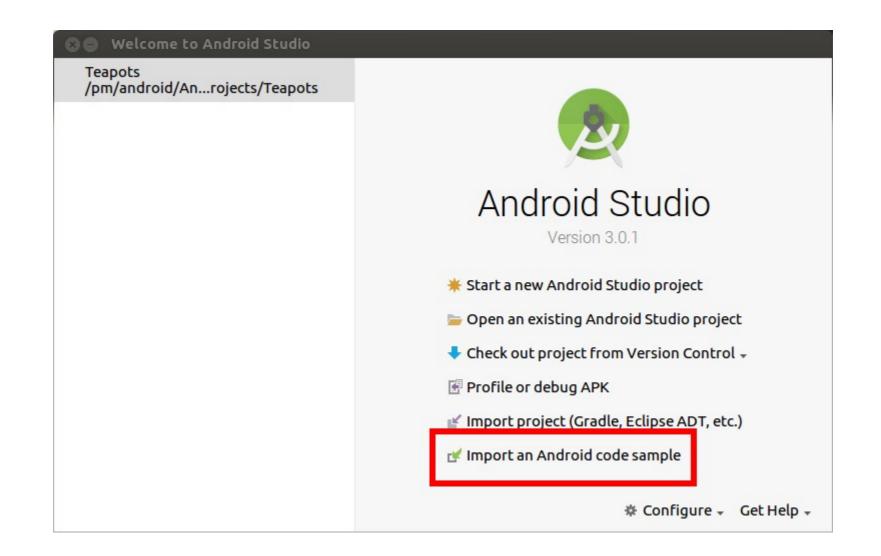
Initial SDK Configuration

Welcome to Android Studio Teapots /pm/android/An...rojects/Teapots Android Studio Version 3.0.1 * Start a new Android Studio project Dpen an existing Android Studio project Check out project from Version Control + Profile or debug APK Import project (Gradle, Eclipse ADT, etc.) Import an Android code sample & Configure -Get Help 🚽

Select and install SDK Tools

	Appearance & Behavior > System Settings > Android SDK		
Appearance & Behavior	Manager for the Android SDK and Tools used by Android Stud	lio	
Appearance	Android SDK Location: /pm/android/SDK	Edi	t
Menus and Toolbars	SDK Platform: SDK Tools DK Update Sites		
System Settings			
Passwords	Below are the available SDK developer tools. Once installed, automatically check for updates. Check "show package deta		
	versions of an SDK Tool.		
HTTP Proxy	Name	Version	Status
Updates	Android SDK Build-Tools		Installed
Usage Statistics	Chake		Not Installed Installed
Android SDK			Not Installed
Notifications	Android Auto API Simulators	1	Not installed
Ouick Lists	Android Auto Desktop Head Unit emulator	1.1	Notinstalled
Path Variables	 Android Emulator Android SDK Platform-Tools 	27.1.7 27.0.1	Installed Installed
	Android SDK Platform-Tools	26.1.1	Installed
Keymap		1	Not installed
Editor	🗌 Google Play APK Expansion library	1	Not installed
Plugins	Google Play Licensing Library	1	Notinstalled
Build, Execution, Deployment	Google Play services	46 2	Not installed Not installed
rools	Instant Apps Development SDK	1.1.0	Not installed
0013	✓ NDK	16.1.4479499	Installed
	Cuppert Repository		
	ConstraintLayout for Android		Update Available: 1
	 Solver for ConstraintLayout Android Support Repository 	47.0.0	Update Available: 1 Installed
	Google Repository	58	Installed
			Show Package Det

Import Code Example



Teapots NDK/OpenGL ES 2.0

 \otimes

Import Sample



Browse Samples Android Studio

Select a sample to import

Q- Tea ▼ Android o preview

Downloadable Fonts (Kotlin)

- Connectivity
 Beam Large Files
- Content

Scoped Directory Access

Getting started

Wear Verify Remote App

Ndk

Teapots

V UI

Downloadable Fonts Downloadable Fonts (Kotlin) Wear Verify Remote App Teapots is an collection of Android C++ samples that uses a Teapot rendering to demostrate Android NDK platform features: - classic-teapot: Rendering classic teapot mesh using GLES 2.0 API and <u>NativeActivity</u>. - more-teapots: Rendering multiple instances of Classic Teapot with GLES 3.0 Instance Rendering - Choreographer-30fps: demonstrates multiple frame rate throttoling techniques based on API level using Chreographer API and EGL Android presentation time extension. This sample uses the new Android Studio CMake plugin with C++ support.

Tags: ndk

Browse source in GitHub

60.10 FPS

Previous Next

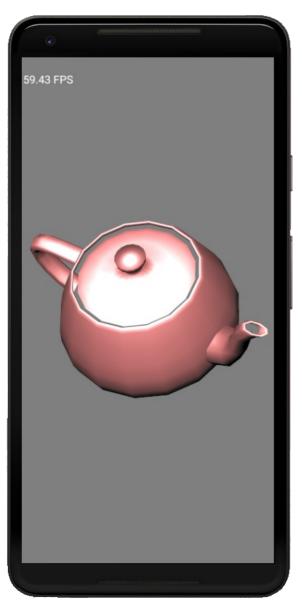
Select Target, Build and Run

🙁 📾 🔳 Teapots - [/pm/android/AndroidStudioProjects/Teapots] - Android Studio 3.0.1																		
<u>F</u> il	e <u>E</u> dit <u>V</u> iew	<u>N</u> avigate	<u>C</u> ode	Analyze	Refacto	r <u>B</u> uild	Run	Tools	VCS	Window	v <u>H</u> el	р						
í.	Teapots						~	🔚 clas	sic-tea	pot 🚽 🕨	4 1	i	01		1 R		Q,	
IJ	🚔 Android		-	⊕ ‡	\$÷ 1∺													٢
1: Project	🕨 📑 choreo	grapher-30	fps															🕑 Gradle
1:1	classic-	-																dle
2	more-te	and the second se			_													1
Ire	Gradle S																	
7: Structure	Stress	l Build Files																
2 2					э	rch Ever	ywhe	re Dout	ole Shi	ft								
to File Ctrl+Shift+N																		
1						e <mark>nt F</mark> iles	Ctrl+	E										
	Gradle Console	2				iastian (killer:	~							*	· 🛓	
Build Variants																		
÷																		Devic
* 2: Favorites																		Device File Explorer
	TODO E	<u>6</u> : Logcat	Term	inal								Q	Event Lo	og (🗉 Gradl	le Cons	ole	
	Gradle sync f	inished in 2	74ms (f	rom cach	ed state)	(a minut	e ago)				(Contex	t: <no< td=""><td>contex</td><td><t></t></td><td>ъ</td><td>#</td></no<>	contex	<t></t>	ъ	#

Create Virtual Device using Wizard

8 Select Deployment Target			
Connected Devices			
📕 Pixel 2 XL API 27 (Android 8.1.0, API 27)			
Create New Virtual Device			
	ОК	Cancel	Help

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 higer 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