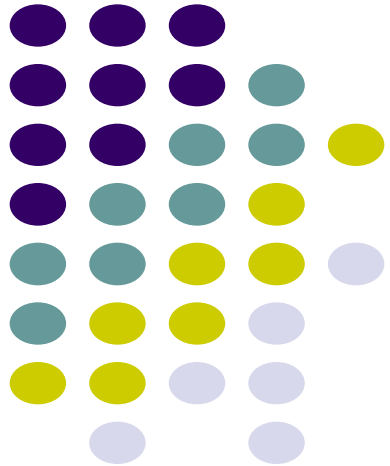
Ubiquitous and Mobile Computing
CS 528: Group 3 Ted Talk

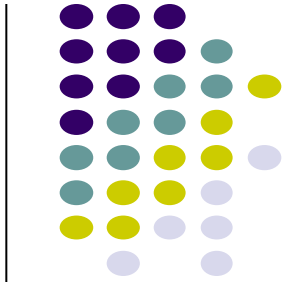


Team 3

Alyssa Herz, Ryan Johnson, Natalia Carvajal Erker, Nicholas Delli Carpini

Computer Science Dept.

Worcester Polytechnic Institute (WPI)

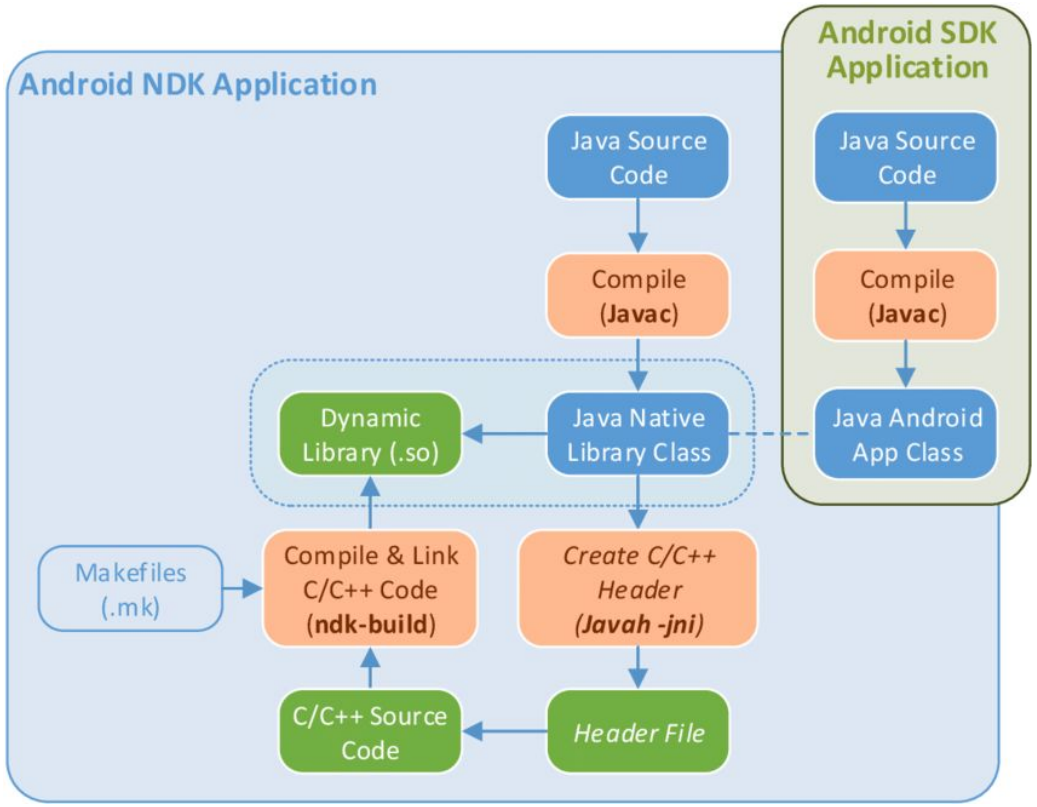


Android NDK

Native Development Kit

What is the NDK?

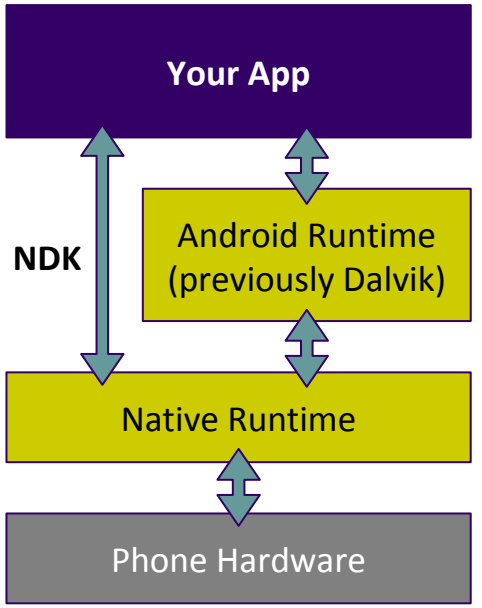
- A way to write native C/C++ code for Android phones, and use it within your Java code!
- Allows access to advanced and performance-critical APIs



Background & Motivation

- Initially Released in 2009 for Android 1.5 (NDK Rev 1, now Rev 21d as of June)
- Developers wanted to be able to run code outside of Dalvik (now Android Runtime)

- Why not just use Java?
  - There is existing code that isn't written for Java
  - Performance in C/C++ can be much higher for computationally-intensive code
  - Deep integration with custom hardware



Problems it Solves

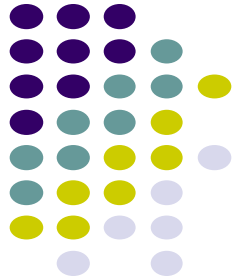
- Creates standard way to add native code in Android apps
  - Toolchain makes portability easier with automatic cross-compiling
- Reuse existing standard/custom libraries, makes porting non-android (and non-java) code to android faster

android studio

CMake
Cross-platform Make

LLVM
COMPILER INFRASTRUCTURE

Problems it Solves



- Performance with computationally-intensive tasks
  - Performance characterization tools (systrace, simpleperf)
  - Lets you gain more performance and/or save battery due to increased efficiency
- Low-level APIs for use with apps with NDK:
  - High Performance Audio
  - OpenGL ES
  - Vulkan
  - Neural Network API
  - Image decoding
  - Camera
  - etc.

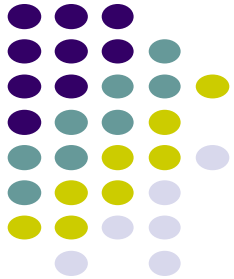
Real World Examples

OpenGL®

OpenMAX

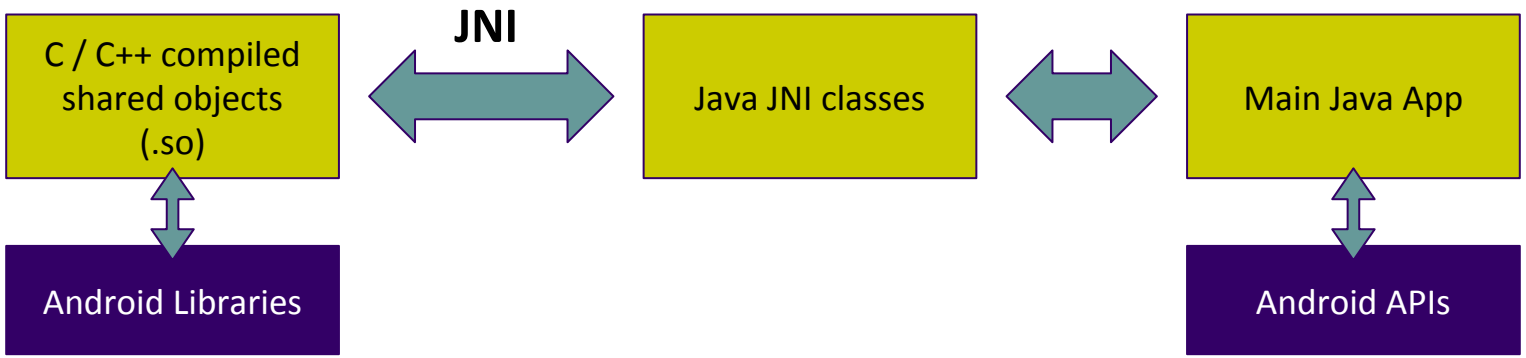
Vulkan™

Overview



Development workflow

1. Write your native C or C++ code, and use Android native APIs (if needed).
2. Add JNI bindings, and make Java JNI class
3. Set up a CMake build system to target a shared object
4. Link build to gradle
5. Run the build, CMake and clang will automatically build native
6. Resulting APK file will hold all native / java code



Code snippet

Java JNI class

```
package com.example.hellojni;


import android.app.Activity;
import android.widget.TextView;
import android.os.Bundle;

public class HelloJni extends Activity
{
   @Override
   public void onCreate(Bundle savedInstanceState)
   {
       super.onCreate(savedInstanceState);
       TextView  tv = new TextView(this);
       tv.setText( stringFromJNI() );
       setContentView(tv);
   }

   public native String  stringFromJNI();
   static {
       System.loadLibrary("hello-jni");
   }
}
```

Load JNI shared object

C native code

Java package path

Class & method name

```
#include <string.h>
#include <jni.h>

jstring
Java_com_example_hellojni_HelloJni_stringFromJNI( JNIEnv* env,
                                                  jobject thiz )
{
   return (*env)->NewStringUTF(env, "Hello from JNI");
}
```

CMake Build Directives

```
cmake_minimum_required(VERSION 3.4.1)

add_library(hello-jni SHARED
            hello-jni.c)

# Include libraries needed for hello-jni lib
target_link_libraries(hello-jni
                      android
                      log)
```