Intel, Delta Course.

Nizhny Novgorod, 2015

Development of C/C++ applications for Android* OS

Alexey Moskalev

About.me/moslex

Moskalev Alexey

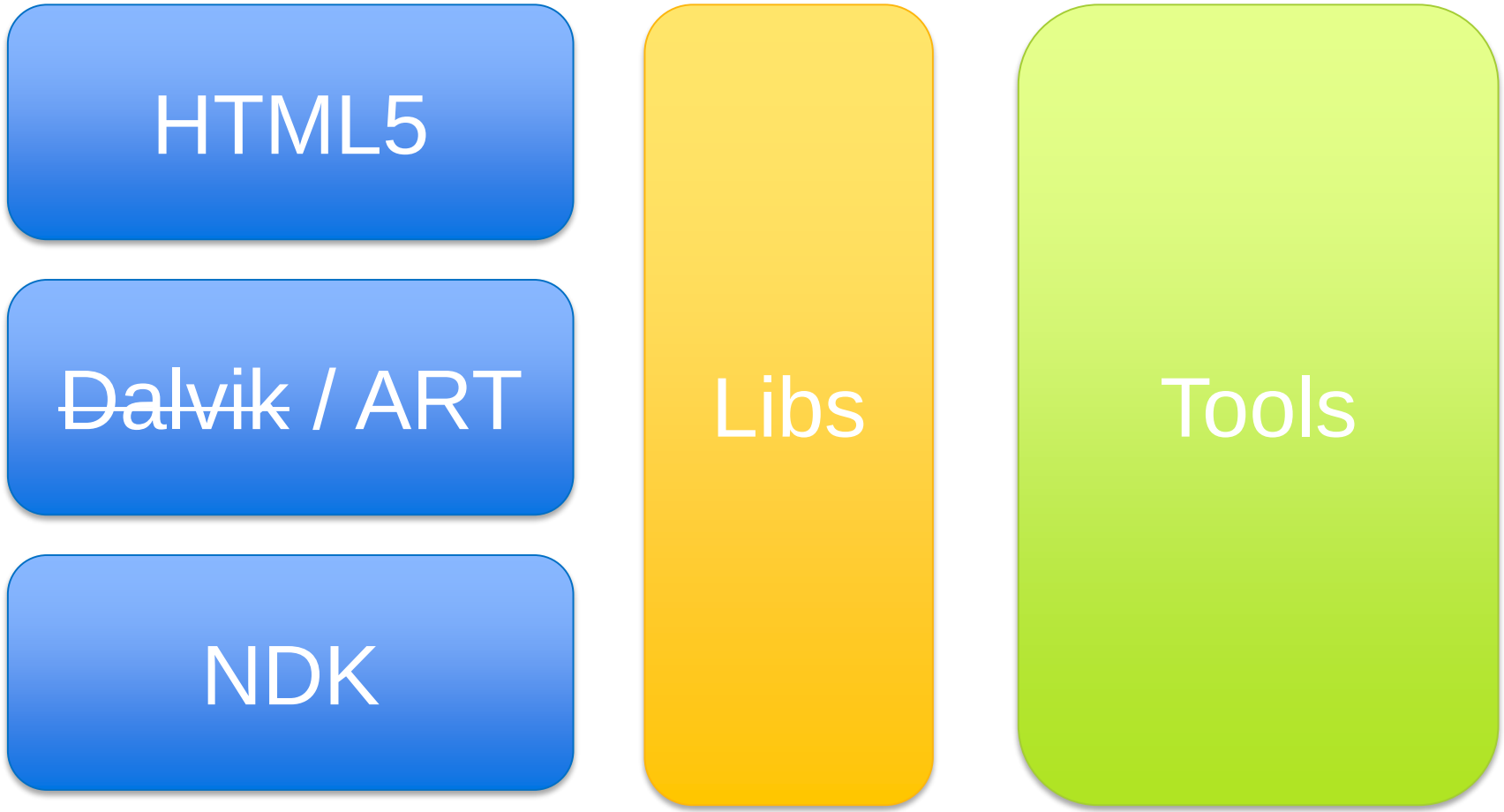
PRM at Intel.
Product: Intel® Threading Building Blocks (Intel® TBB)
Experience: 7 years

Education:
NNSU - CMC (2005-2010)
HSE - MBA (2011-2013)

Summary:

- Android* development ecosystem:
  - Specific of development process for Android* OS
  - Developer's tools:
    - IDE: Eclipse; Android System Studio
    - SDK, NDK
- Developer's tools from Intel:
    - Intel ® C++ Compiler for Android
    - Intel® Cilk™ Plus
    - Intel® Threading Building Blocks (Intel® TBB)
    - Intel® SDK for OpenCL™ Applications
    - Intel® Integrated Native Developer Experience
- Samples & Tips:
  - hello-jni
  - TBB examples:
    - Simple & Deterministic reduce
    - Tachyon

Programming methods for Android ?



http://habrahabr.ru/post/201346/
http://habrahabr.ru/company/intel/blog/263873/

Specific of development process for Android OS

Host

Windows OS
Linux OS (Ubuntu)
OSX

IDE + Android SDK + NDK

Target

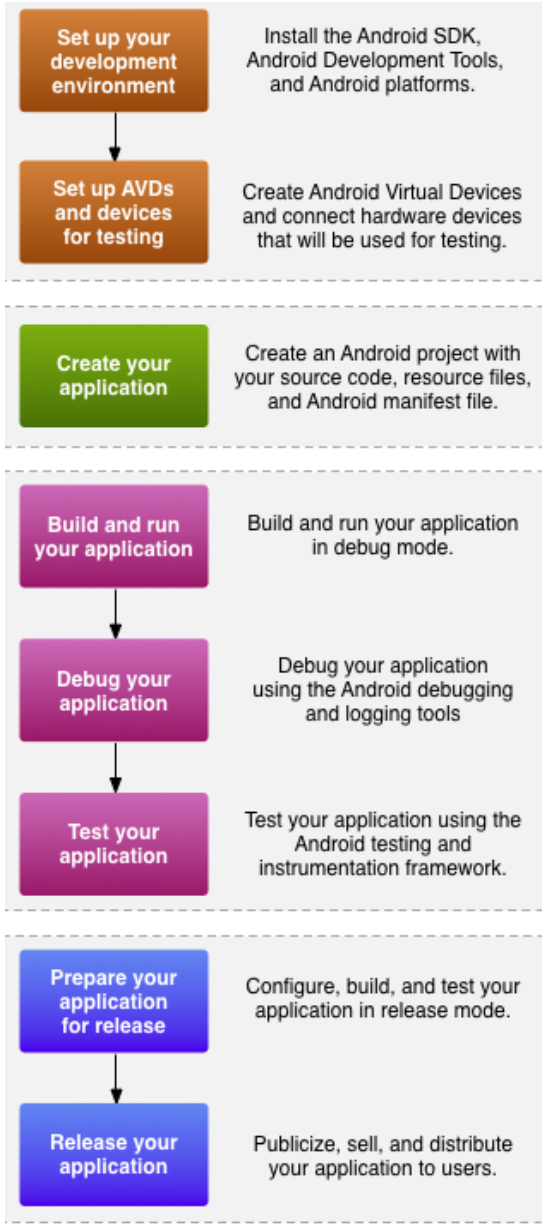
Device

Emulator (+HAXM*)

ADT Bundle

http://habrahabr.ru/company/intel/blog/146114/
http://habrahabr.ru/company/intel/blog/265791/

Specific of development process for Android OS



Setup

Development

Debugging and testing

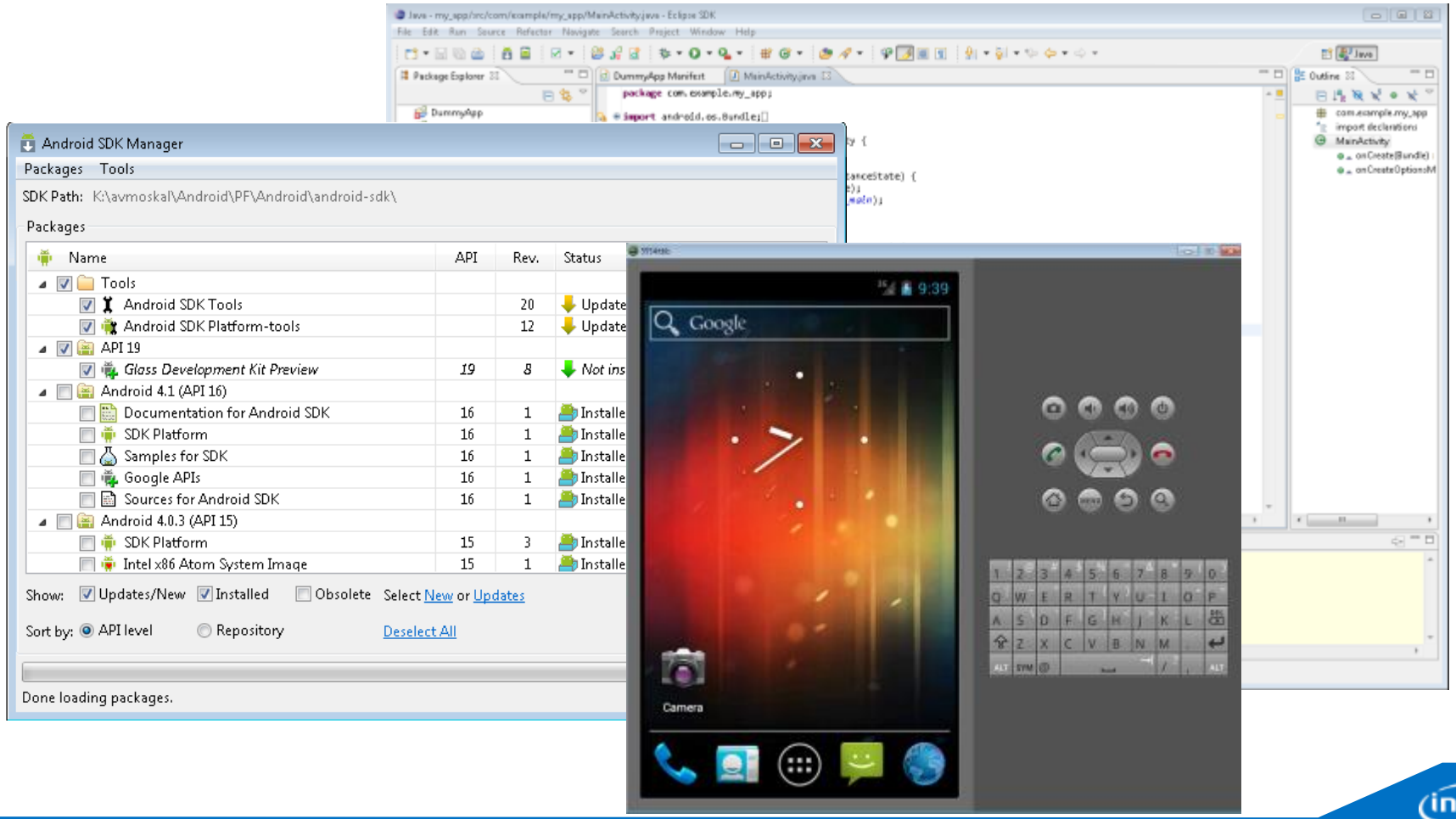
Publishing

https://developer.android.com/tools/workflow/index.html

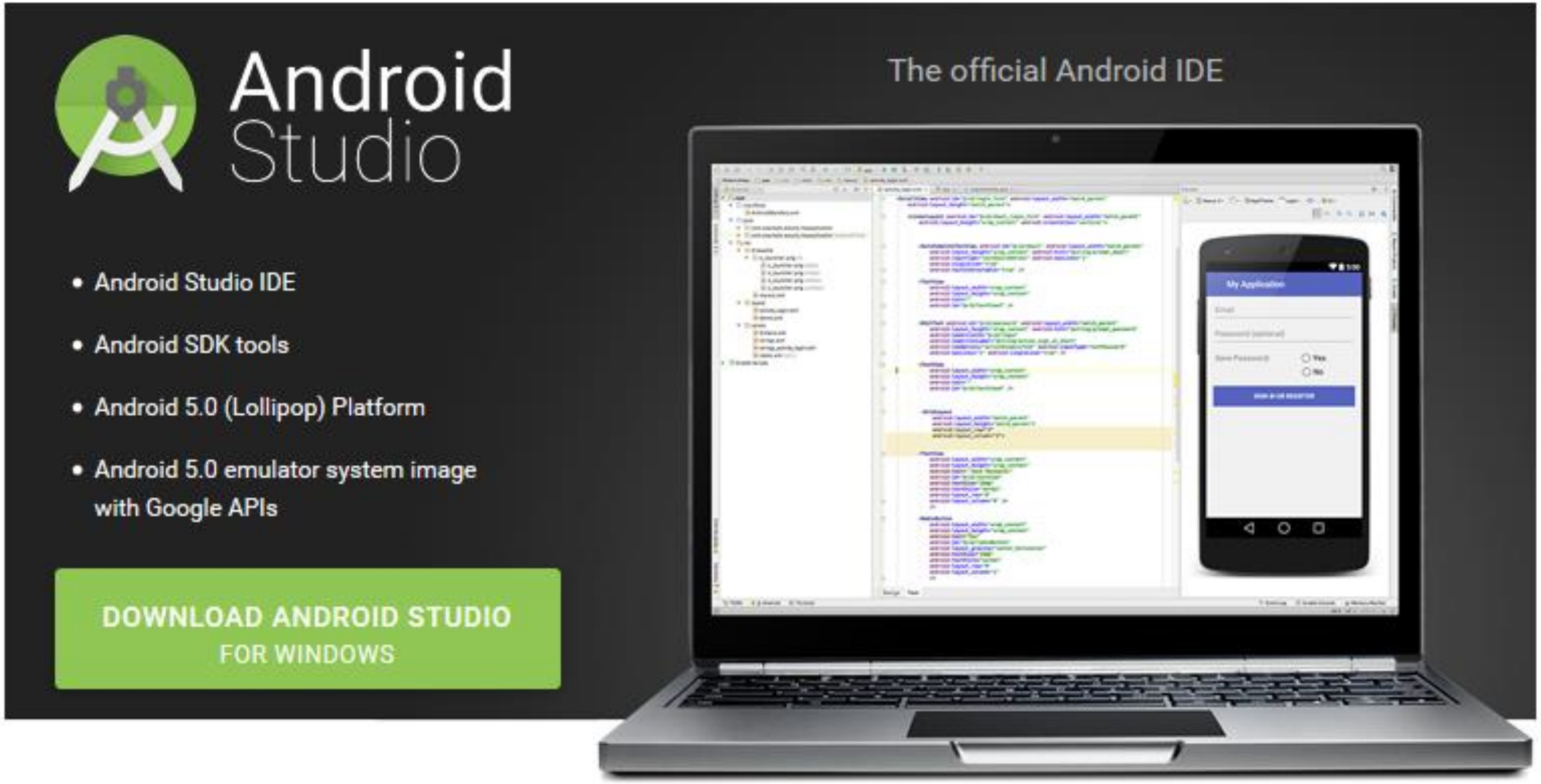
Developer's tools:

IDE: Eclipse ADT bundle with Android SDK

IDE (Integrated Development Environment): http://eclipse.org/mobile/
ADT (Android Developer Tools): https://developer.android.com/sdk/installing/installing-adt.html
SDK (Software Development Kit): https://developer.android.com/sdk/installing/adding-packages.html



Developer's tools: **Android Studio IDE**



http://developer.android.com/sdk/index.html

Developer's tools:

Android NDK (Native Development Kit)

https://developer.android.com/tools/sdk/ndk/index.html

The NDK is a toolset that allows you to implement parts of your app using native-code languages such as C and C++. For certain types of apps, this can be helpful so you can reuse existing code libraries written in these languages, but **most apps do not need the Android NDK.**

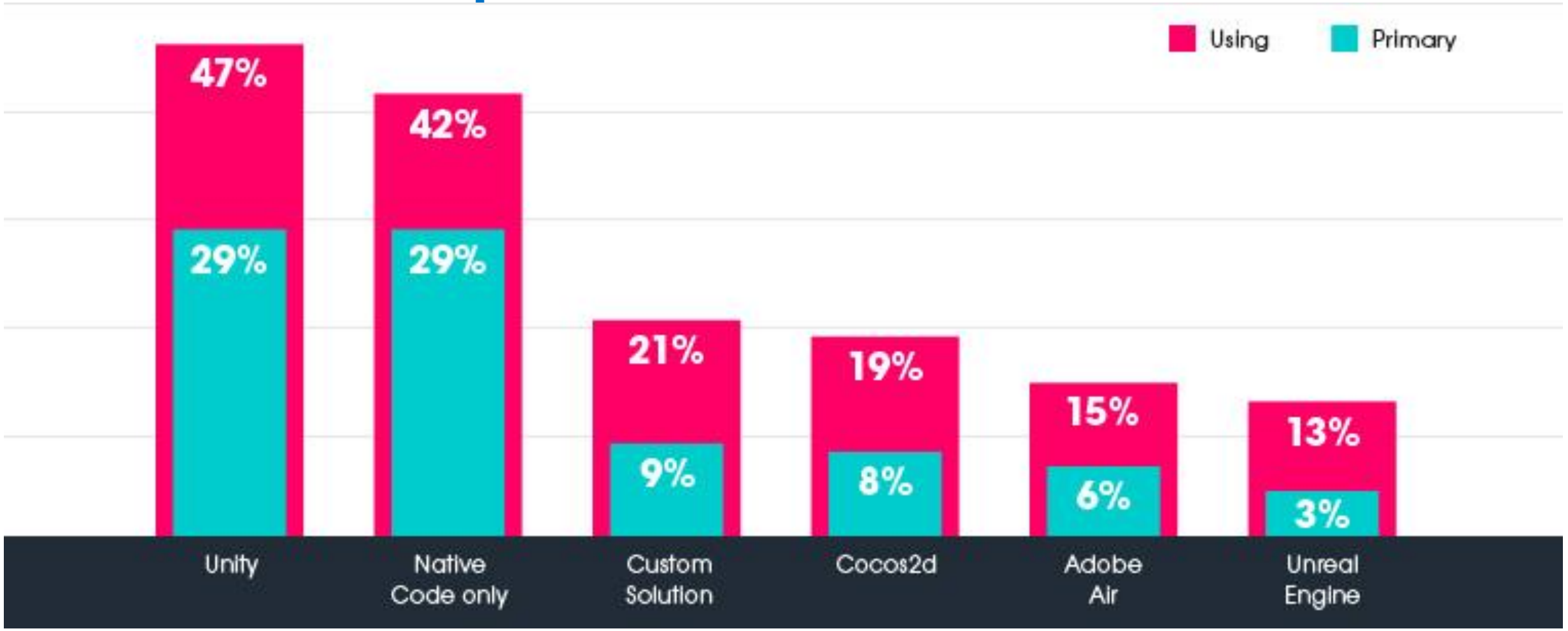
Before downloading the NDK, you should understand that the NDK will not benefit most apps. As a developer, you need to balance its benefits against its drawbacks. Notably, using native code on Android generally does not result in a noticable performance improvement, but it always increases your app complexity. In general, you should only use the NDK if it is essential to your app—never because you simply prefer to program in C/C++.

Typical good candidates for the NDK are **CPU-intensive workloads such as game engines, signal processing, physics simulation**, and so on. When examining whether or not you should develop in native code, think about your requirements and see if the Android framework APIs provide the functionality that you need.

Android platform compatibility

Native Code CPU Architecture Used	Compatible Android Platform(s)
ARM, ARM-NEON	Android 1.5 (API Level 3) and higher
X86	**Android 2.3 (API Level 9) and higher**
MIPS	Android 2.3 (API Level 9) and higher

Games development



unity

NDK (C\C++)

Source: Vision Mobile, Developer Economics, State of the Developer Nation Q3, 2014

Android NDK (Native Development Kit) + Add-on

Compilers:

- GCC
- Clang

- Intel ® C++ Compiler for Android

Native APIs:

- libc (C library) headers
- libm (math library) headers
- JNI interface headers
- libz (Zlib compression) headers
- liblog (Android logging) header
- OpenGL ES 1.1 and OpenGL ES 2.0 (3D graphics libraries) headers
- libjnigraphics (Pixel buffer access) header (for Android 2.2 and above).
- A Minimal set of headers for C++ support
- OpenSL ES native audio libraries
- Android native application APIS

- Intel® Cilk™ Plus
- Intel® Threading Building Blocks (Intel® TBB)
- Intel® SDK for OpenCL™ Applications

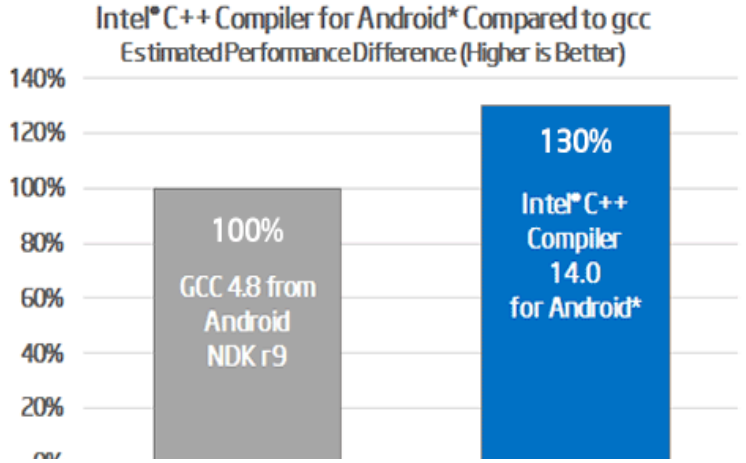
?

Intel ® C++ Compiler for Android

Intel® C++ Compiler 14.0 for Android*

Industry Leading App Performance on Intel Processor-Based Android Devices

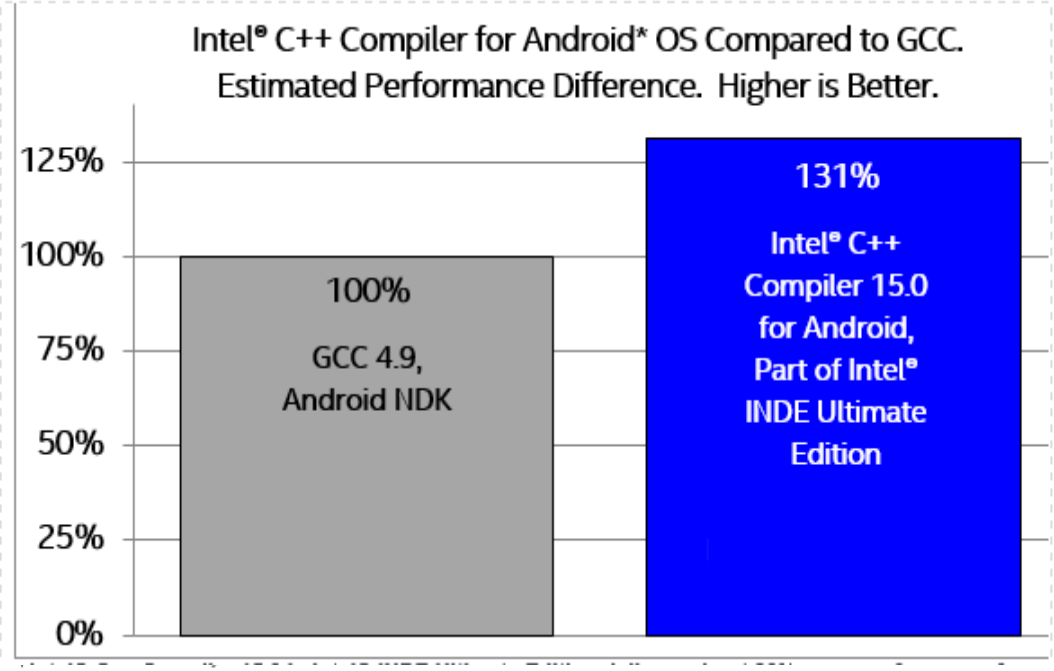
The Intel C++ Compiler brings a heritage of outstanding performance to Android app developers.



OVERVIEW | DOCUMENTATION & SUPPORT | SYSTEM REQUIREMENTS | INTRO VIDEOS

Get Great Android App Performance

- Develop apps for Android mobile devices based on Intel processors
- Outstanding performance in many cases by just recompiling
- Compatible with Android NDK
- Compatible with GNU* C++ in the Android NDK for multi-architecture support
- Develop on Windows*, OS X* or Linux*
- Eclipse* support or command-line
- Android Studio support, a preview feature
- Microsoft Visual Studio support
- Fast and easy download, simplified installation



https://software.intel.com/en-us/c-compilers/inde

When to use Intel® C++ Compiler?

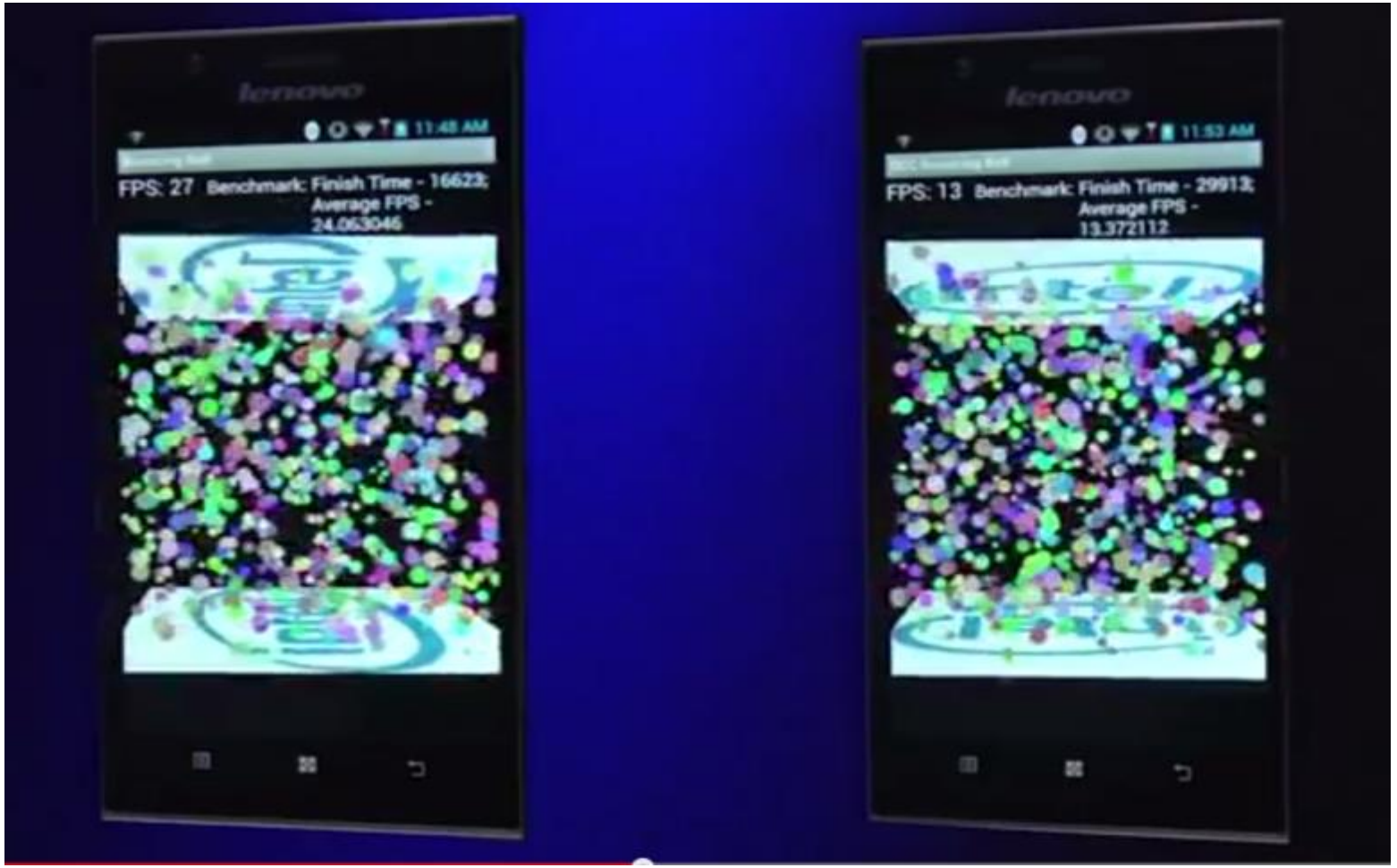
- ICC can only be used for native source code
- You will get better speedup if
  - The app is CPU bound (check with Intel GPA)
  - The hot functions are not written in assembler
  - Code can be vectorized
    - Usually true for multimedia apps & games
  - You want to multithread your application (use Intel® Cilk™ Plus or Intel® TBB)
  - You want to explicitly optimize for the latest CPU generation

General optimization options

- -O1
  - optimize code size, auto vectorization is turned off
- -O2
  - inlining
  - vectorization
- -O3
  - loop optimization
  - data pre-fetching

Example why to use Intel® C++ Compiler

- Collision detection



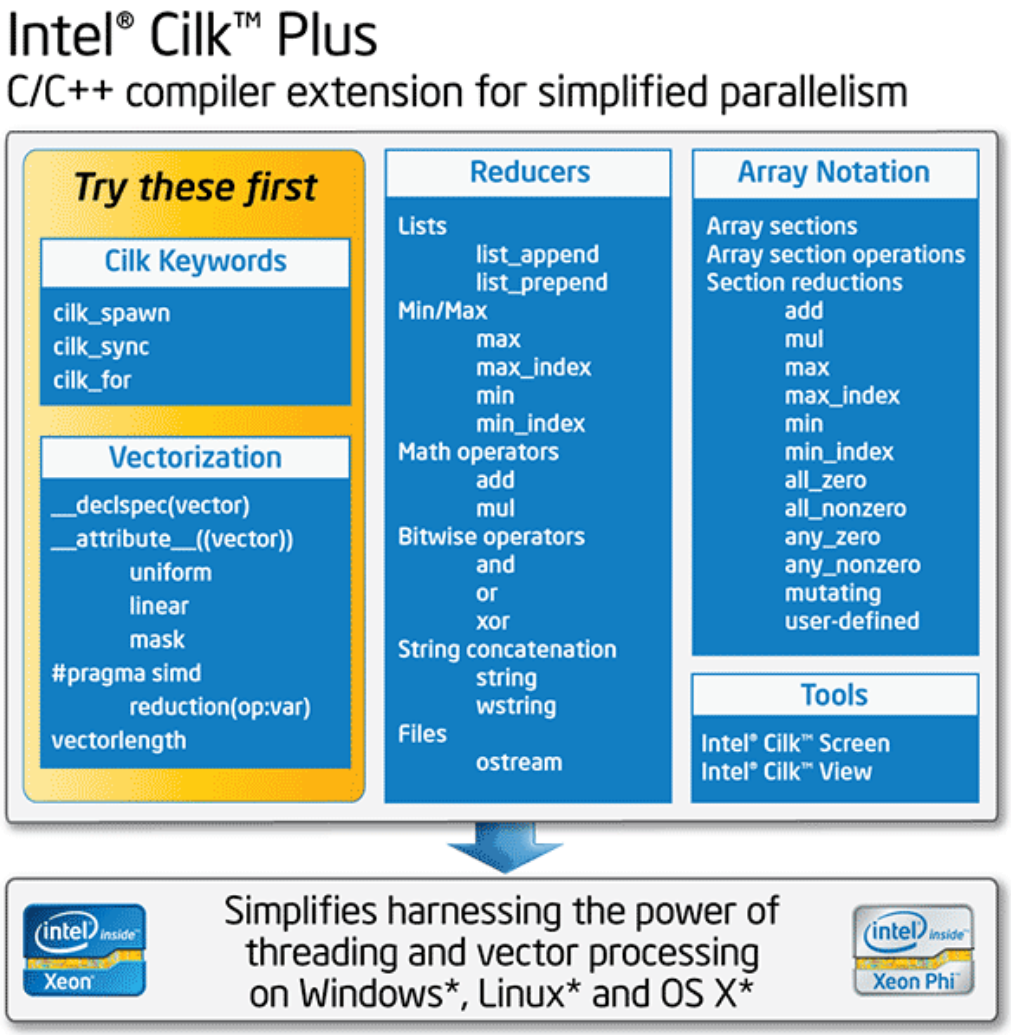
ICC 24 FPS

GCC 4.6 13 FPS

http://www.youtube.com/watch?v=6t7mpyFHuHk

Intel® Cilk™ Plus

Intel® Cilk™ Plus is an extension to C and C++ that offers a quick and easy way to harness the power of both multicore and vector processing.



cilk_spawn and cilk_sync

```
void f()
{
   g();
   work
   work
}

void g()
{
   work
   work
}
```

```
void f()
{
   cilk_spawn  g();
   work
   work
   cilk_sync;
   work
}
```

cilk_for

```
for (int x = 0; x < n; ++x)
{
   cilk_spawn f(x);
}


cilk_for (int x = 0; x < n; ++x)
{
   f(x);
}
```

http://habrahabr.ru/company/intel/blog/204838/

https://software.intel.com/en-us/intel-cilk-plus

Intel® Threading Building Blocks

https://software.intel.com/en-us/intel-tbb

Intel® Threading Building Blocks (Intel® TBB)

Widely used C++ template library for task parallelism

- Rich set of components to efficiently implement higher-level, task-based parallelism
- Future-proof applications to tap multicore and many-core power
- Compatible with multiple compilers and portable to various operating systems

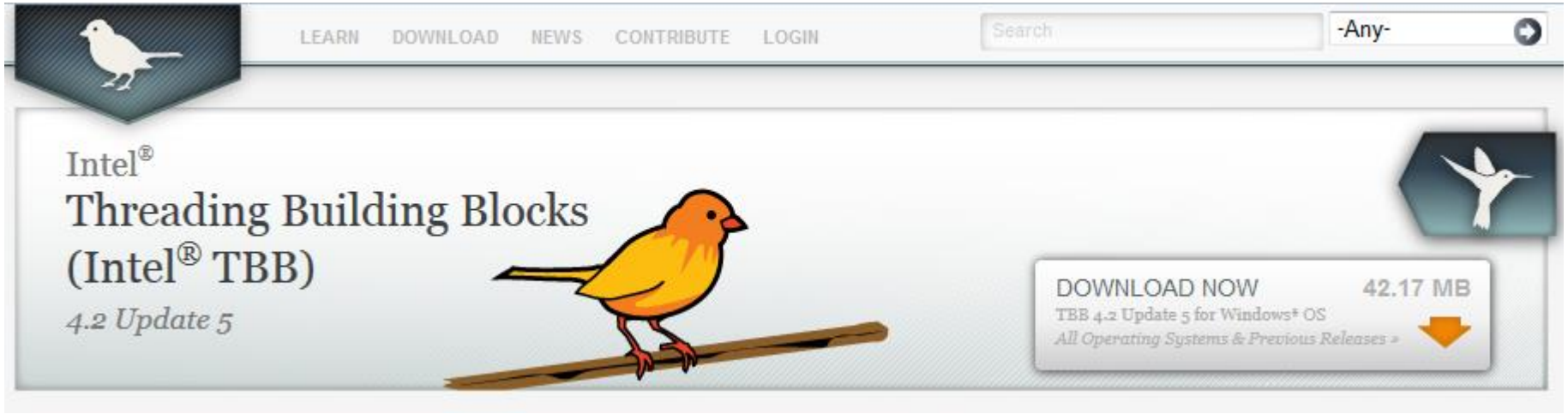
Available in These Suites

Intel® Parallel Studio XE

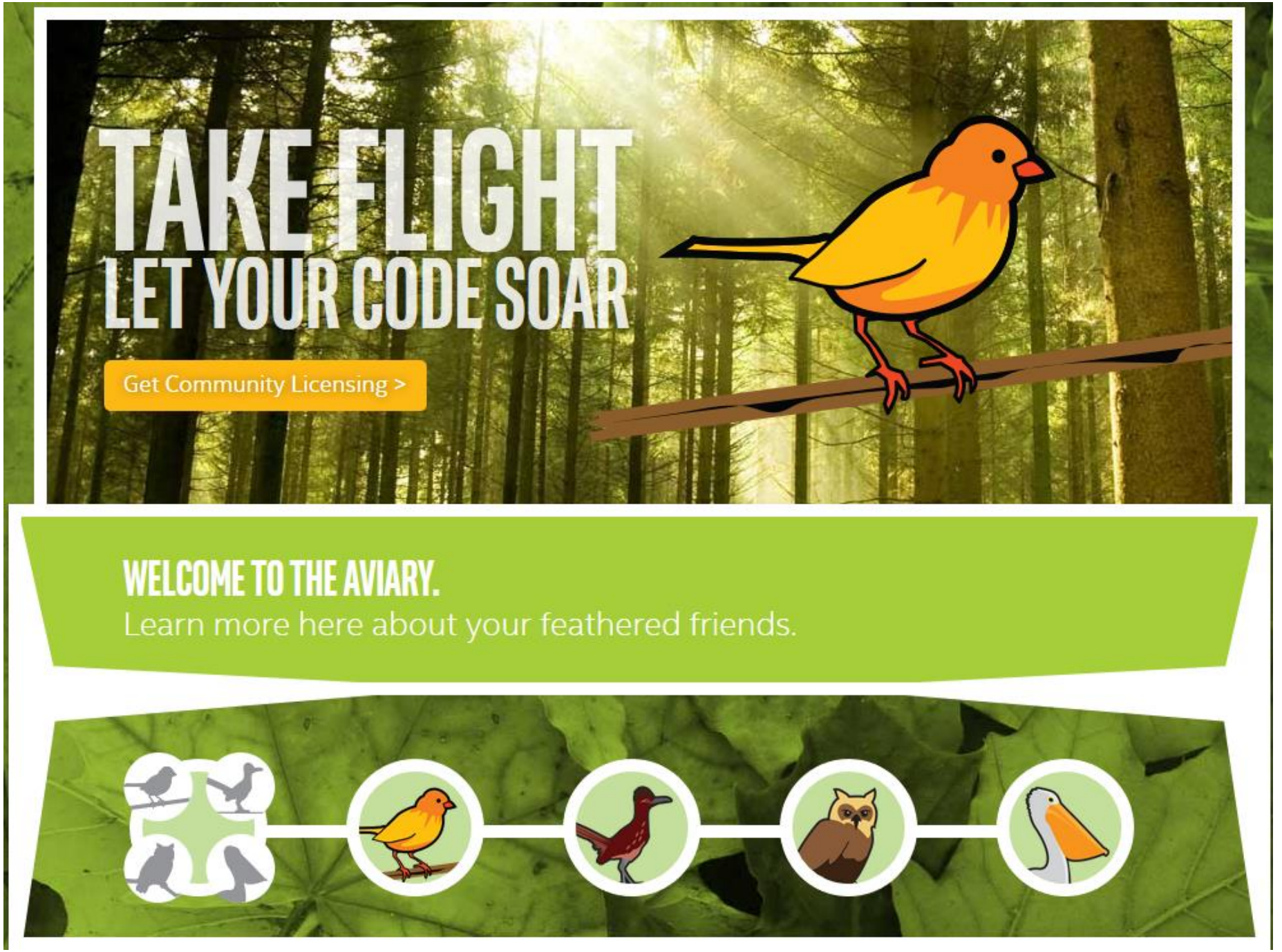
Intel® System Studio

Intel® Integrated Native Developer Experience (Intel® INDE)

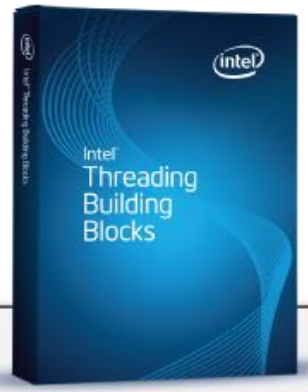
https://www.threadingbuildingblocks.org/



GPLv2



Intel(R) TBB
Intel(R) IPP
Intel(R) MKL
Intel(R) DAAL



Intel® Threading Building Blocks

C and C++ template library for creating high performance, scalable parallel applications

Included in Intel® Parallel Studio XE & Intel® Cluster Studio

Generic Parallel Algorithms	Concurrent Containers	Task Scheduler	Synchronization Primitives	Memory Allocation	Miscellaneous
parallel_for(range)	concurrent_hash_map	task	atomic	tbb_allocator	thread
parallel_reduce	concurrent_queue	task_group	mutex	cache_aligned_allocator	tick_count
parallel_for_each(begin, end)	concurrent_bounded_queue	structured_task_group	recursive_mutex	scalable_allocator	captured_exception
parallel_do	concurrent_vector	task_group_context	spin_mutex	zero_allocator	moveable_exception
parallel_invoke	concurrent_unordered_map	task_scheduler_init	spin_rw_mutex	memory_pool	enumerable_thread_specific
pipeline	concurrent_priority_queue	task_scheduler_observer	queuing_mutex		combinable
parallel_pipeline	concurrent_unordered_set		queuing_rw_mutex		
parallel_sort			reader_writer_lock		
parallel_scan			critical_section		
flow::graph			condition_variable		
parallel_deterministic_reduce			null_mutex		
			null_rw_mutex		

Optimized Threading Functions
running on Windows*, Linux*, OS X* & more

Intel® SDK for OpenCL™ Applications

Intel® SDK for OpenCL™ Applications

Use the Intel® SDK for OpenCL™ Applications to help optimize development time and maximize platform performance with OpenCL™ and Intel® Graphics.

The OpenCL (Open Computing Language) Advantage

- Open, standard, and portable API for heterogeneous computing
- The standard way to program Intel® HD Graphics and Intel® Iris™ Graphics family
- Designed for visual computing applications
- Supported on Intel CPUs, Intel® Xeon Phi™ coprocessors, and Intel® Graphics

The Intel® SDK for OpenCL™ Applications Advantage:

- Free, comprehensive development environment for OpenCL API on Intel® Architecture
- Microsoft Visual Studio* and Eclipse* integration
- Support for Windows* and Linux* operating systems
- Remote development for Android* OS
- Certified OpenCL 1.2 support
- Create, code, compile, advise, and debug with the code builder for OpenCL applications
- Tune OpenCL application with the accompanied Intel® VTune™ Amplifier XE

https://software.intel.com/en-us/vcsource/tools/opencl-sdk

Media Pack for Android*

Media Pack for Android*

Bring professional-quality video & audio to Android*. Part of the Intel® Integrated Native Developer Experience (Intel® INDE).

- Video & audio extensions for Android* enabling camera & screen capture, video editing, video streaming and audio fingerprinting.
- A free download through the Intel® Integrated Native Developer Experience (Intel® INDE).

What is Media Pack for Android*?

Media Pack for Android* is a bundle of cross-platform Java* samples and APIs enabling you to access lower-level audio and video capabilities within the Android software stack with ease, delivering professional-quality usages to your end user. The media pack supports cross-platform development, allowing you to deliver apps that run on Android devices based on ARM* and run best on Intel® Architecture.

Media pack for Android* contains samples, source code, libraries and more enabling:

- Screen sharing
- Screen capturing
- Video streaming – in partnership with Wowza* Media Systems
- Content Recognition – in partnership with Audible Magic*

?

https://software.intel.com/en-us/intel-inde/media-pack

Android NDK (Native Development Kit) + Дополнения

Compilers:

- GCC
- Clang

- Intel ® C++ Compiler for Android

Native APIs:

- libc (C library) headers
- libm (math library) headers
- JNI interface headers
- libz (Zlib compression) headers
- liblog (Android logging) header
- OpenGL ES 1.1 and OpenGL ES 2.0 (3D graphics libraries) headers
- libjnigraphics (Pixel buffer access) header (for Android 2.2 and above).
- A Minimal set of headers for C++ support
- OpenSL ES native audio libraries
- Android native application APIS

- Intel® Cilk™ Plus
- Intel® Threading Building Blocks (Intel® TBB)
- Intel® SDK for OpenCL™ Applications

+

Media Pack for Android*
Intel® Graphics Performance Analyzers

Intel® INDE

https://software.intel.com/en-us/c-compilers/inde/

Intel® Integrated Native Developer Experience

Что такое Intel® INDE?

Intel® INDE – это кроссплатформенный набор инструментов и библиотек, позволяющий создавать приложения для ОС Windows и Android.

Этот продукт будет особенно полезен тем, кто уже разрабатывает или только собирается разрабатывать Android приложения для работы с видео и приложения, использующие нативный код.

Подробнее | Скачать Intel® INDE

В состав продукта входят следующие инструменты Intel:

- Intel® INDE Media Pack
- Intel® C++ Compiler for Android (ICC)
- Intel® Threading Building Blocks (TBB)
- Compute Code Builder beta
- Intel® GPA

*Intel и логотип Intel являются товарными знаками корпорации Intel на территории США и других стран

Создание

- Медиаданные
- Работа с потоками
- Compute Code Builder

Компиляция

- GNU C++Compiler
- Intel® C++Compiler

Отладка и анализ

- Platform Analyzer
- System Analyzer
- Frame Analyzer
- Frame Debugger

Запуск продукта

- Устройства Android 4.3 и выше на базе архитектуры Intel и ARM
- Устройства Microsoft Windows 7—8.1 на базе архитектуры Intel

http://habrahabr.ru/special/intel/inde/about

Intel® Integrated Native Developer Experience

Customize your development toolbox to code native applications, expose underlying architecture, and deliver higher performance and differentiated apps. Your favorite Intel® Integrated Native Developer Experience (Intel® INDE) tools are now available as standalone downloads. Select libraries, SDKs, and tools for code creation, compilation, debugging, and analysis to create high-performance C++/Java applications.

Intel INDE Support

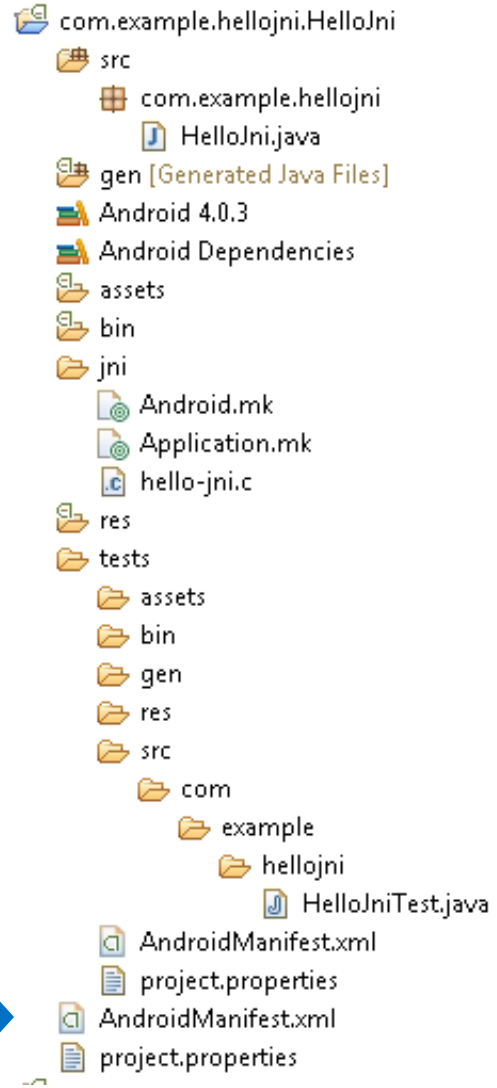
https://software.intel.com/en-us/intel-inde

Talk is cheap. Show me the code.

NDK example: hello-jni

http://developer.android.com/ndk/samples/sample_hellojni.html

This sample guides you through HelloJNI, a minimal application built with the NDK. This sample is in the samples/hello-jni/ directory under the root directory of your NDK installation.

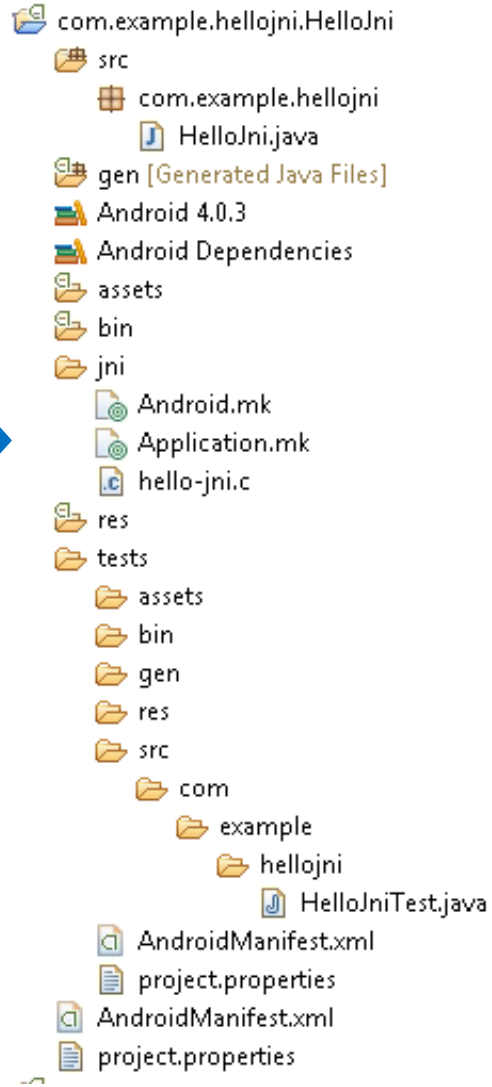


AndroidManifest.xml – application components description.

- Access permissions
- Minimum API Level
- Required HW, SW
- 3-rd party API libs

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.example.hellojni"
      android:versionCode="1"
      android:versionName="1.0">
    <uses-sdk android:minSdkVersion="3" />
    <application android:label="@string/app_name"
                 android:debuggable="true">
        <activity android:name=".HelloJni"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

NDK example: hello-jni



/jni /

Application.mk

```
APP_ABI := all // all architectires
```

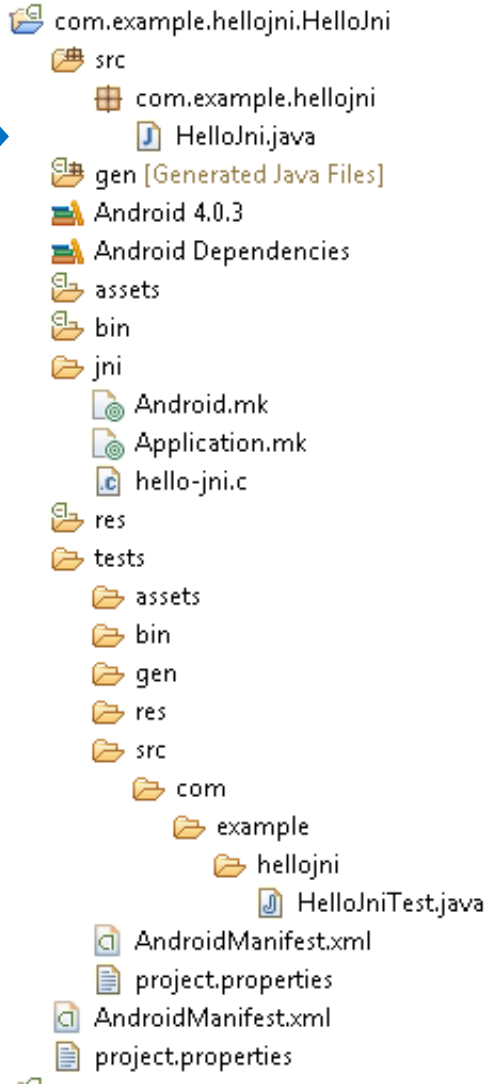
Android.mk

```
LOCAL_PATH := $(call my-dir)
include $(CLEAR_VARS)
LOCAL_MODULE    := hello-jni
LOCAL_SRC_FILES := hello-jni.c
include $(BUILD_SHARED_LIBRARY)
```

hello-jni.c:

```
#include <string.h>
#include <jni.h>
jstring
Java_com_example_hellojni_HelloJni_stringFromJNI( JNIEnv* env, jobject thiz )
{
 return (*env)->NewStringUTF(env, "Hello from JNI !  Compiled with ABI " ABI ".");
 }
```

NDK example: hello-jni



/src /
HelloJni.java

```
public class HelloJni extends Activity
{
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);

        /* Create a TextView and set its content.
        TextView  tv = new TextView(this);
        tv.setText( stringFromJNI() );
        setContentView(tv);
    }

    /* A native method that is implemented by the
     * 'hello-jni' native library, which is packaged
     * with this application.
     */
    public native String  stringFromJNI();

    /* This is another native method declaration that is *not*
    public native String  unimplementedStringFromJNI();

    /* this is used to load the 'hello-jni' library on application
     * startup. The library has already been unpacked into
     * /data/data/com.example.hellojni/lib/libhello-jni.so at
     * installation time by the package manager.
     */
    static {
        System.loadLibrary("hello-jni");
    }
}
```

Android application fundamentals

Classic “C”

```
#include <stdio.h>
int main() {
  printf("Hello World\n");
  return 0;
}
```

Classic “Java”

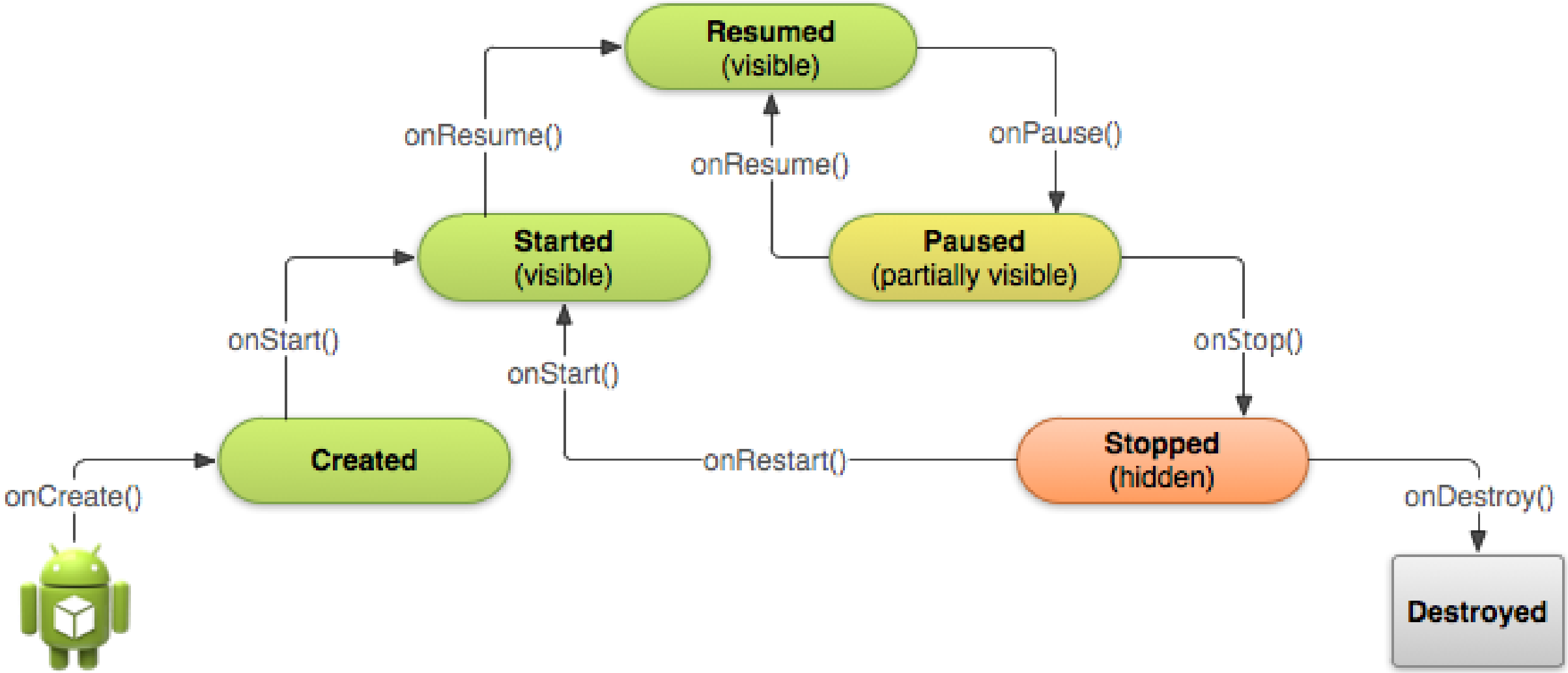
```
class HelloWorld {
    public static void main(String[] args)
{
        System.out.println("Hello
World!");
    }
}
```

Applications for Android:

```
 public class Activity extends ApplicationContext {
    protected void onCreate(Bundle savedInstanceState);

    protected void onStart();

    protected void onRestart();

    protected void onResume();

    protected void onPause();

    protected void onStop();

    protected void onDestroy();
}
```

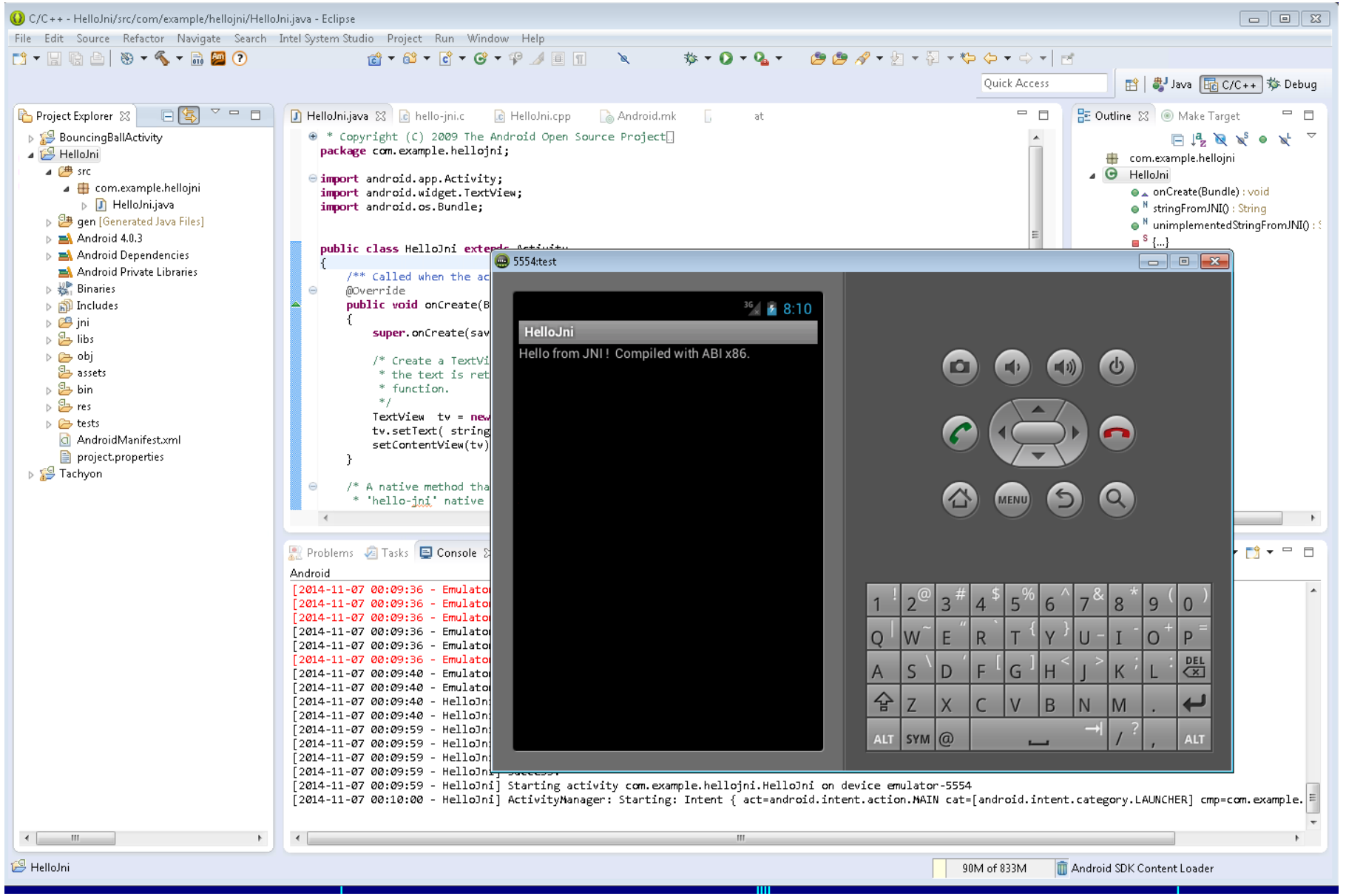
http://developer.android.com/reference/android/app/Activity.htm

Android application fundamentals: Class Activity



http://developer.android.com/reference/android/app/Activity.htm

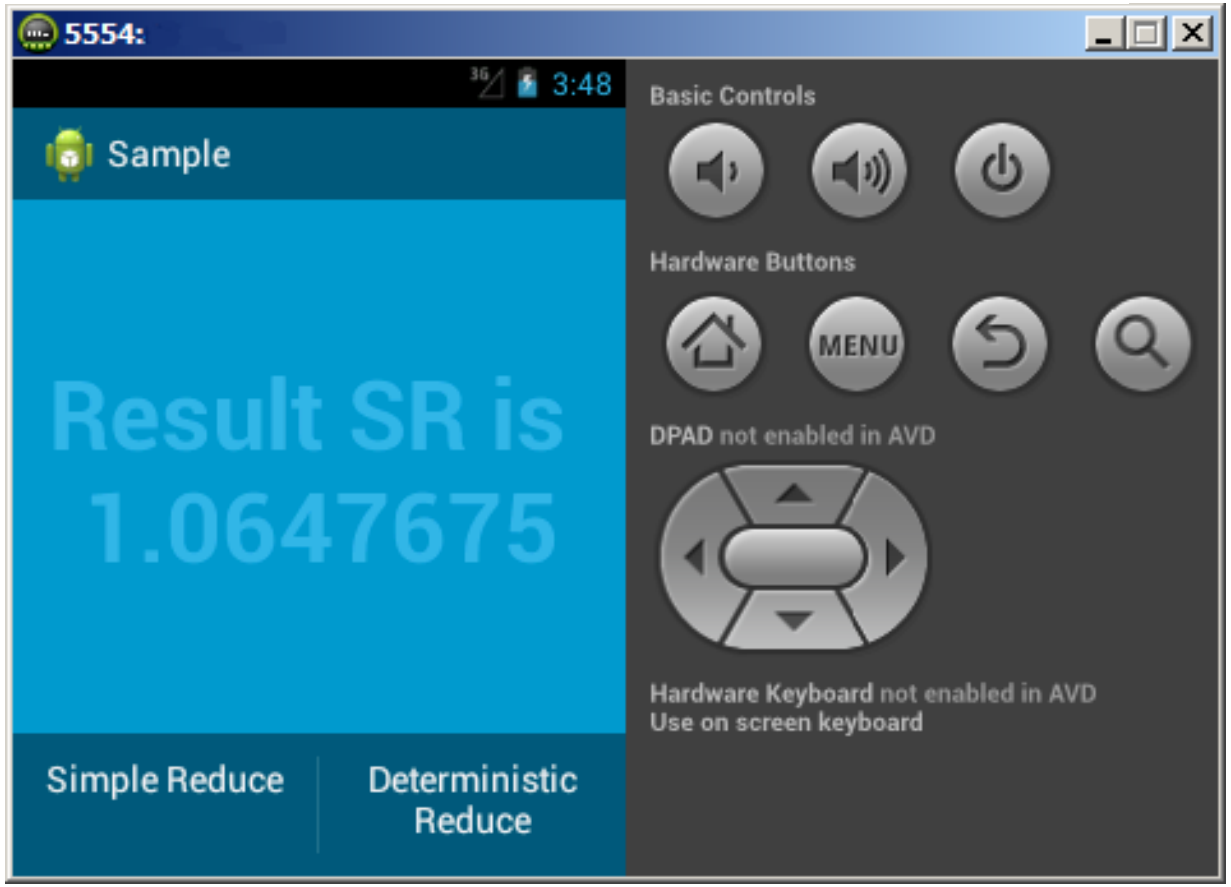
NDK example: hello-jni



TBB examples: Simple & Deterministic reduce

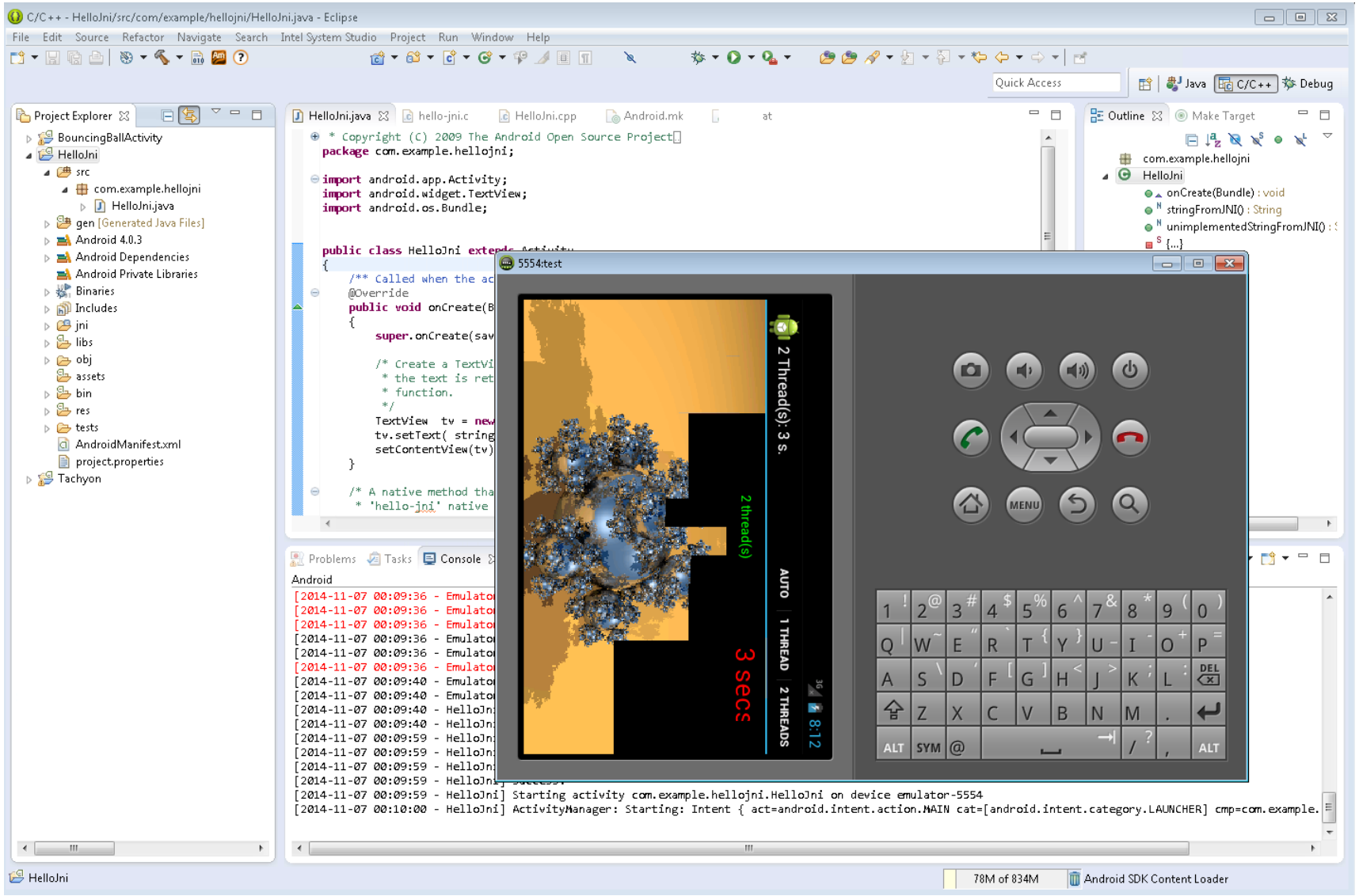
21 декабря 2012 в 16:27

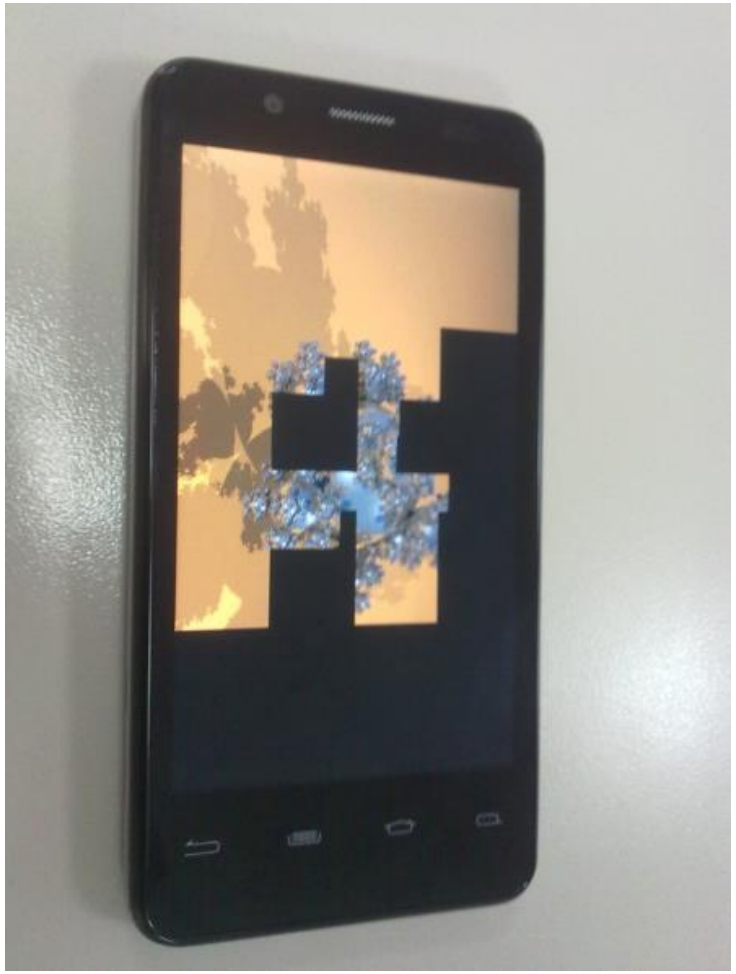
Android: Написание многопоточных приложений с помощью Intel® Threading Building Blocks tutorial



http://habrahabr.ru/company/intel/blog/161527

TBB examples: Tachyon

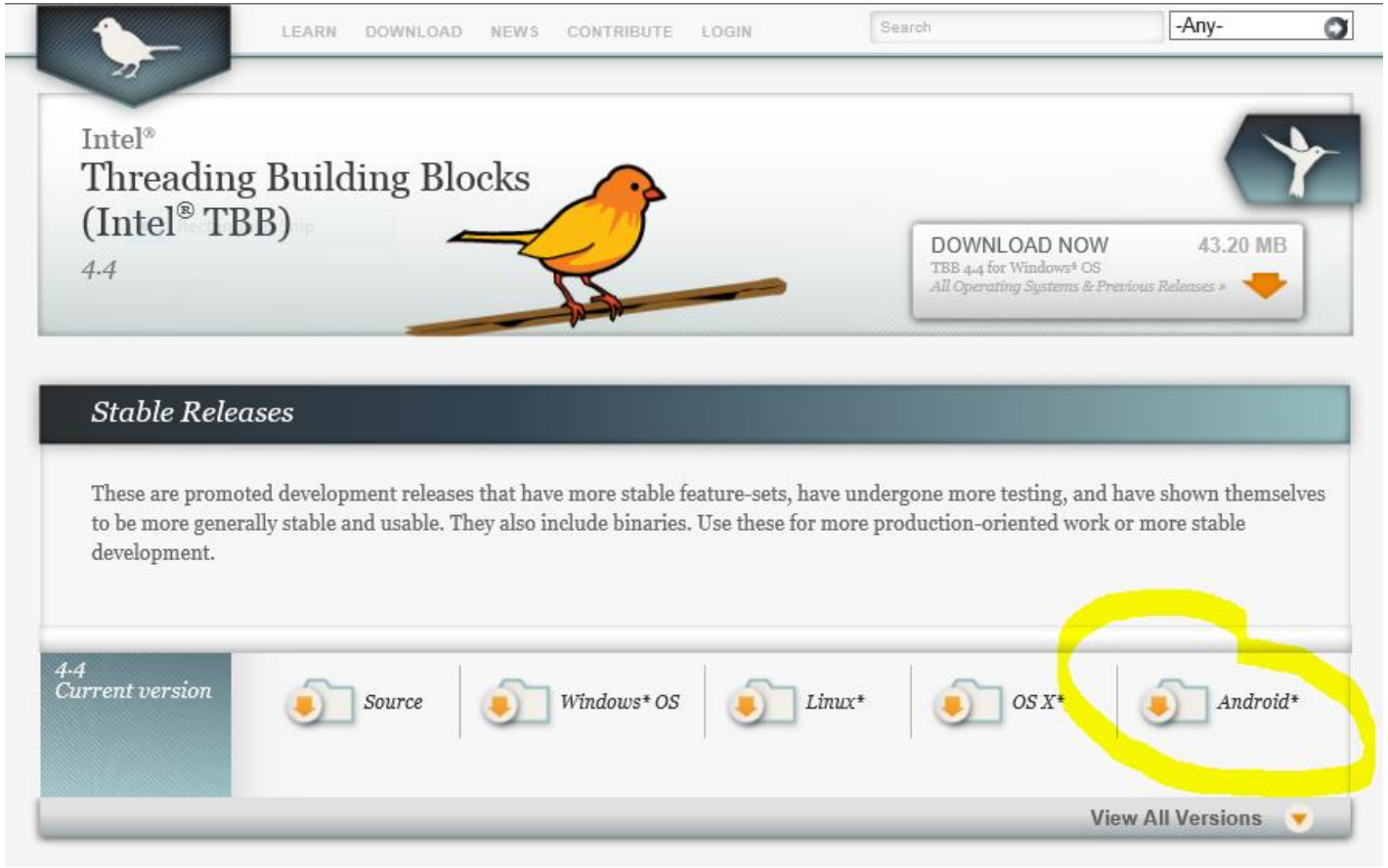




http://threadingbuildingblocks.org/

How to run Tachyon example in Eclipse

1. Where can I get Tachyon example?



Get TBB library package for Android
https://www.threadingbuildingblocks.org/download

Prepare the workspace folder for Eclipse:

- Create a folder
- Put the contents of the downloaded archive there
- Android Tachyon project path is **<workspace_path>\tbb44_20150728oss\examples\parallel_for\tachyon\android**

2. What SW do I need to install to run Tachyon example in Windows?

- Java SDK http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
- Android SDK http://developer.android.com/sdk/index.html

  Installation guide: http://developer.android.com/sdk/installing/index.html
- Android NDK http://developer.android.com/ndk/downloads/index.html

  Installation guide: http://developer.android.com/ndk/guides/setup.html
- Eclipse for Java development + C++ Development Tool and Android Development Tool plugins

  http://www.eclipse.org/downloads/packages/eclipse-ide-java-developers/mars1

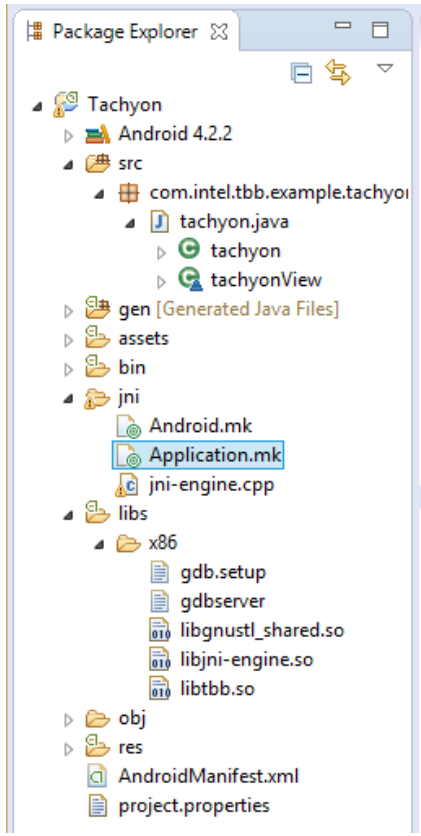
  https://eclipse.org/cdt/

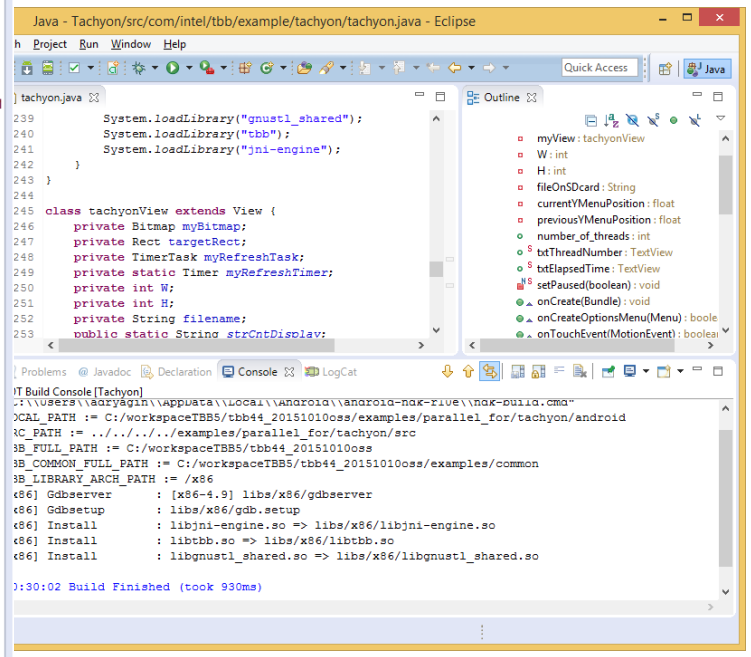
  https://marketplace.eclipse.org/content/android-development-tools-eclipse

  Installation guide: http://developer.android.com/sdk/installing/installing-adt.html
- Set emulators with necessary configurations (e.g. Intel_Nexus_7 with x86 architecture to have the Intel Hardware Accelerated Execution Manager (Intel® *HAXM*) run with this emulator to make it work faster)

3. How to run Tachyon example in Eclipse

- Create a new Android project **from the existing source code**
- Build the project
- After successfully building the project you'll get the structure in the Package Explorer:
- The native code and make files are in the JNI section of the project
- The structure of the Android.mk and Application.mk files:





Android.mk

```
include $(CLEAR_VARS)
LOCAL_MODULE    := jni-engine
LOCAL_SRC_FILES := jni/jni-engine.cpp $(TBB_PATH)/examples/common/gui/convideo.cpp $(SRC_PATH)/trace.tbb
LOCAL_CFLAGS += -std=c++11 -fexceptions -Wdeprecated-declarations  -I$(TBB_FULL_PATH)/include -I$(TBB_CO
LOCAL_LDLIBS := -lm -llog -ljnigraphics -L./ -L$(TBB_LIBRARY_FULL_PATH)
LOCAL_SHARED_LIBRARIES += libtbb
include $(BUILD_SHARED_LIBRARY)

LOCAL_PATH := $(TBB_LIBRARY_FULL_PATH)
include $(CLEAR_VARS)
LOCAL_MODULE    := libtbb
LOCAL_SRC_FILES := libtbb.so
include $(PREBUILT_SHARED_LIBRARY)
```

Application.mk

```
APP_ABI:= x86
APP_STL:=gnustl_shared
APP_GNUSTL_FORCE_CPP_FEATURES := exceptions rtti
APP_PLATFORM:=android-15
NDK_TOOLCHAIN_VERSION:=4.9
```

Development tips: Code optimization for Android 5.0

31 июля в 13:00

Пять способов оптимизации кода для Android 5.0 Lollipop

перевод

Разработка под Android*, Блог компании Intel

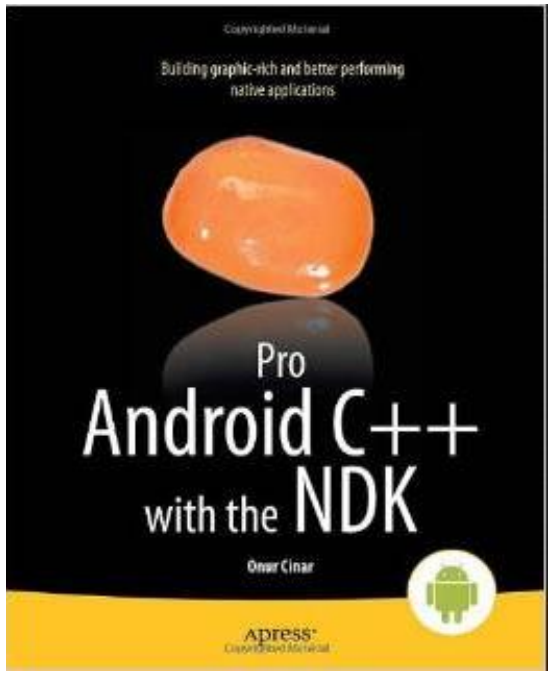
Совет №4. Избегайте вызывать маленькие методы через JNI

```
class A {
  public final int factorial(int x){
    int f = 1;
    for (int i =2; i <= x; i++)
      f *= i;
    return f;
  }
  public int compute (){
    int sum = 0;
    for (int i = 0; i < 1000; i++)
      sum += factorial(i % 5);
//если мы воспользуемся здесь JNI-вариантом функции factorial(),
// приложение будет работать заметно медленнее,
// так как вызов происходил бы в цикле
// а это лишь усиливает нагрузку на систему в ходе JNI-вызовов
    return sum;
  }
}
```

http://habrahabr.ru/company/intel/blog/263873/

Links, books

http://developer.android.com
https://software.intel.com/
http://habrahabr.ru/company/intel/



Cinar O. — Pro Android C++ with the NDK – 2012.

Android on x86
An Introduction to Optimizing for Intel(R) Architecture

«Android NDK. Разработка приложений под Android на C/C++»
Сильвен Ретабоуил

Links and the latest news:

- Building Native Android* Apps Using Intel(R) C++ Compiler in Android Studio* 1.0.1 https://software.intel.com/en-us/articles/building-native-android-apps-using-intelr-c-compiler-in-android-studio-101
- NDK samples https://github.com/googlesamples/android-ndk
- Android Studio + Gradle + NDK: http://habrahabr.ru/company/intel/blog/216353/
- https://www.youtube.com/watch?v=okLKfxfbz40
- Google announced end of development and official support for the Android Developer Tools (ADT) in Eclipse at the end of the year. They plan to focus efforts on developing Android Studio, official Android IDE. See http://android-developers.blogspot.com/2015/06/an-update-on-eclipse-android-developer.html for details.

Questions?


intel