Портирование Next-gen 3D-приложений на Android: Практическое руководство

Терентьев Илья,
Developer Technologies Engineer,
NVIDIA

Вступление

- High-end игры и приложения на платформе Android
- Целевые версии API и Android
- Этапы портирования приложения
- Совместимость

Целевая платформа Android

- Какие разрешения экрана и устройства вы будете поддерживать?
  - Планшеты? Телефоны? И то, и другое?
  - Альбомная ориентация? Вертикальная? И то, и другое?
- Какие устройства ввода?
  - Тач? Мультитач?
  - Акселерометр/гироскоп/компас?
- На каких версиях Android будет работать ваше приложение?
  - Включать ли поддержку Éclair (2.1) и Froyo (2.2)?
  - Поддерживать только Gingerbread (2.3) и старше?
- Минимальные требования к HW-рендеренгу?

High-end игры на Android

- Источники:
  - Игры, портированные с консолей или PC
  - Игры, портированные с других мобильных платформ
  - Игры, разработанные для нескольких мобильных платформ
- Этапы портирования:
  - “Подъем”
  - “Настройка”
  - “Продуктирование”
- Планируйте портирование заранее…

Android и нативный код

- На верхнем уровне всех Android-приложений находится Java
  - Но это не значит, что вам придется обязательно столкнуться с Java-кодом…
- Используя нативный код можно решить широкий спектр задач; но не все
- Нативный усложняет ваше приложение, но
  - Им можно эффективно управлять
  - Раскрывает максимальную производительность
  - Упрощает портирование существующего кода
- Практически все разработчики, с которыми мы работаем, широко используют нативный код

5

NDK: Что внутри

- Набор кросс-компиляторов
  - Gcc для архитектуры ARM
  - Работает на Windows, Linux или Mac OS
- Набор “стабильных API”
  - Функционал Android, который можно использовать напрямую из нативного кода
  - Безопасные и совместимые “снизу вверх” системные и Android API
- Реализация JNI – Java Native Interface
  - Нативный код на Android всегда вызывается из Java!

NDK: Чего нет

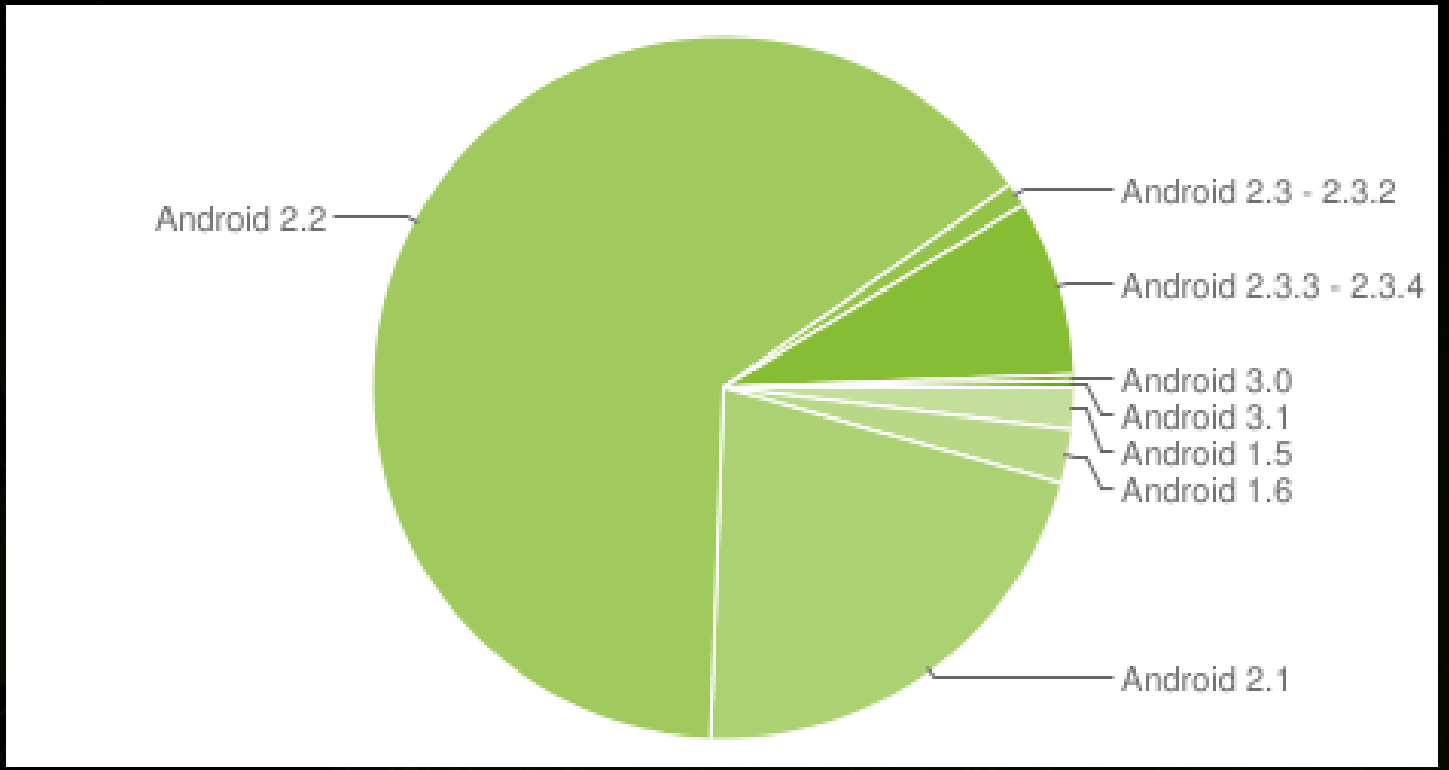
- NDK не позволит вам разрабатывать Linux приложения для Android
  - ОС защищает устройство от различных форм доступа
  - Реализованы не все POSIX API
- NDK не позволяет полностью забыть про Java
  - На старых версиях Android невозможно использовать нативный код без Java
  - Новые версии Android позволяют иногда не писать Java-код
  - Некоторый функционал доступен только из Java даже в последних версиях Android
- Это не среда разработки
  - Используйте другие IDE (например Eclipse)

Уровни Android SDK

- Выбор минимального уровня Android SDK означает:
  - Функционал, который вы можете использовать (чем выше, тем больше)
  - API, которые доступны из C/C++ (чем выше, тем больше)
  - Минимальную версию ОС, которую будет поддерживать ваша игра (чем ниже, тем больше рынок)
- Выбор. Решения.

Рынок Android (на 1 июня 2011)

Версия Android		Доля рынка
Honeycomb	(3.1)	0.3%
Honeycomb	(3.0)	0.3%
Gingerbread	(2.3.x)	9.2%
Froyo	(2.2)	64.6%
Eclair	(2.0/2.1)	21.2%
До Eclair	(1.6,1.5)	4.4%



^ "Platform Versions". *Android Developers*. http://developer.android.com/resources/dashboard/platform-versions.html. "based on the number of Android devices that have accessed Android Market within a 14-day period ending on the data collection date noted below"

Android 2.1/2.2 (Éclair/Froyo)

- Доступно для нативного кода (C/C++):
  - POSIX threads, работа с файлами, math, сокеты
  - OpenGL ES 2.0
  - (2.2) Доступ CPU к экранным плоскостям
- Недоступно:
  - EGL
  - Пользовательский ввод
  - Работа со звуком
  - Проигрывание видео
  - Работа с UI
- (2.1) Приложение должно быть **полностью** установлено во внутреннюю память (максимальный размер ~50MB)
- (2.2) Доступ к внешней пямяти (SD-карта)

SDK API уровень 7-8+

Android 2.3 (Gingerbread)

SDK API уровень 9+

- Доступно для нативного кода (C/C++):
  - Нативные приложения (NativeActivity)
  - Звук через OpenSL ES
  - EGL! (Инициалиация 3D)
  - Работа с окнами
  - Пользовательский ввод (тач, кнопки)
  - Датчики (акселерометр, гироскоп, компас)
  - Загрузка установленных ресурсов
- Теперь практически все игры могут обойтись без Java-кода (кроме воспроизведения видео)
- Но в результате не будут работать на Éclair или Froyo

NativeActivity

- В API 9 (Gingerbread) добавлен класс NativeActivity
- Поддерживается
  - Весь жизненный цикл приложения
  - Пользовательский ввод
- Возможность реализовать приложение на 100% нативном C/C++ коде
- Это ключ к минимизации или отказа от использования Java
- Обратите внимание на native_app_glue!

Интерфейсы Dalvik (Java)

- Доступно только из Java:
  - Воспроизведение видео
  - Работа с камерой
  - Вывод UI
  - API системной интеграции
- Внимательно выбирайте “только нативный код”

Этапы портирования

- “Подъем”
- “Настройка”
- “Продуктирование” или коммерческое внедрение

Этапы портирования: “Подъем”

- Что необходимо сделать:
  - Интеграцию с системой сборки
  - Системные вызовы (сокеты, работа с файлами, ввод, и.т.п.)
  - Рендеринг (OpenGL ES)
  - Ре-экспорт/пересборку ресурсов
  - Начальную отладку
- Даже на этом этапе не упускайте из вида “продуктирование”

Интеграция с системой сборки

- “Сборка” приложения для Android возможна на различных платформах:
  - Windows
  - Linux
  - Mac OS
- Eclipse – самая распространенная IDE
  - Но сборка из командной строки также возможна на всех платформах
- NVIDIA DevTech помогает разработчикам интегрировать сборку Android-приложения в любой мультиплатформенный сборочный процесс

Портирование системных вызовов

- Хорошо отработанный процесс для всех мультиплатформенных “движков”
- Если у вас уже есть POSIX-порт, вы на верном пути!
- Несколько ограничений касающихся Android
  - Ограничения файловой системы, особенно на запись
  - До Android 2.3 отсутствует поддержка `wchar_t`
    - И даже после 2.3 она введена для “migrating existing code”
  - Поддержка C++ STL добавлена недавно и не полностью
  - Жесткое требование к выравниванию полей структур (не только на Android)
    - Иногда приводит к трудностям с сериализацией

Совет: знай уровень доступа!

- У каждого Android-приложения есть манифест в XML-файле
- В манифесте задаются настройки приложения, мэппинги для Java-классов и т.п.
- Также в манифесте запрашиваются уровни доступа
- Если нет какого-либо доступа, может произойти следующее:
  - Нет INTERNET доступа? Вызов сокетов вернется с ошибкой…
  - Нет WRITE_EXTERNAL_STORAGE? Запись на SD-карту будет невозможна…
- Это касается и Java-, И нативного кода
- Узнайте больше

http://developer.android.com/reference/android/Manifest.permission.html

Многопоточность

- Сегодня многоядерные чипы, такие как NVIDIA Tegra, – стандарт
- Многопоточность – это ключ к максимальной производительности
- Распараллельте:
  - Физику
  - Частицы
  - Игровую логику
  - Отрисовку
  - Работу с сетью
- Думайте о будущем
  - Не ограничивайтесь двумя ядрами!

“Подъем” рендерера

- Важнейшая часть любого порта
- OpenGL ES (GLES, ES) – 3D API на Android
  - Игры для мобильных платформ скорее всего сделаны под ES
  - PC и консольные игры – нет
- Вы должны выбрать между OpenGL ES 1.x и 2.0
  - Next-gen игры должны быть под ES 2.0
  - Вскоре все приложения будут под ES 2.0
- OpenGL ES может быть и на PC!
  - Существуют Linux/Windows OpenGL ES эмуляторы
  - Результат может сильно отличаться от конечного устройства

PC профиль OpenGL ES2

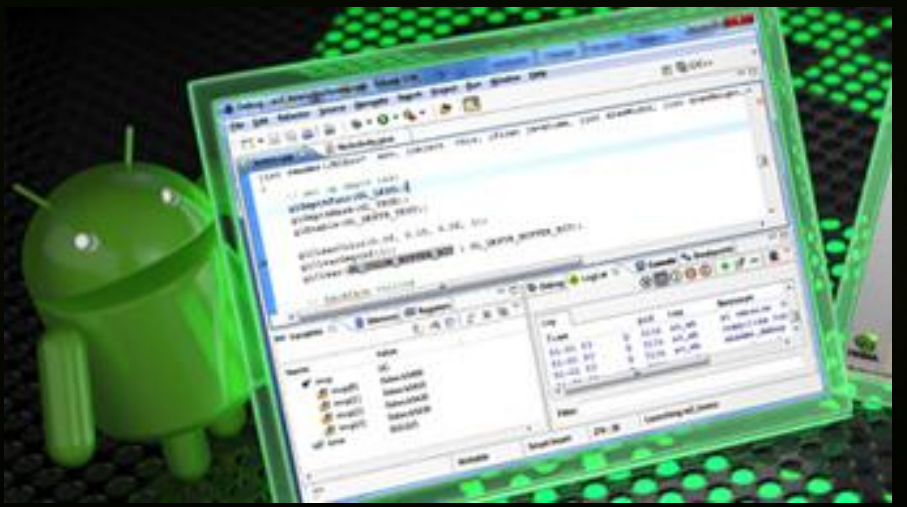
- Эмулятор Android на текущий момент не поддерживает GLES2
  - И даже если будет – он предпологает только частично работающий порт!
- Но многие приложения для Android работают и на Windows/Linux
- Используйте `EXT_create_context_es2_profile`
  - Портируйте рендерер на ES2 на PC
  - Используйте утилиты, которые вы знаете
  - Распараллельте процесс портирования
- Запущенный на Android рендерер может быть вам уже хорошо знаком

Отладка

- Две разные истории:
  - Java: отлично!
  - Нативный код: проблематично...
- Извесные трудности:
  - Настройка удаленной отладки gdb/gdbserver
  - Java/Native отлаживаются разными отладчиками
  - gdbserver сломан в некоторых версиях NDK
  - Невозможно отлаживать многопоточное приложение до версии Android 2.3
- Улучшается день ото дня
  - Но воз и ныне там...

Менеджер отладки NVIDIA

- Плагин для Eclipse, который упрощает отладку нативного C/C++ кода на платформе Tegra
- Сквозная отладка Java и нативного кода
- Регулярно обновляется для поддержки последних версий NDK
- Работает на
  - Windows 7
  - Mac OS X
  - Linux



Этапы портирования: настройка

- Настройка геймплея
  - Настройка чувствительности устройств ввода/датчиков
  - Настройка на размер экрана и форм-фактор (экранный “геймпад”)
- Важно иметь под рукой несколько устройств
- Настройка производительности
  - Выявление основных узких мест (CPUs, GPU, ширина канала памяти)
  - Используйте имеющийся инструментарий

Графическая совместимость

- Качество графики и “фичи” – часто являются основой игр
  - Игры выжимают все возможное из целевой платформы
- Возможности 3D-железа разные на каждом устройстве
- В результате можно получить некорректную отрисовку или сбой приложения
- Пользователь может слишком поздно понять, что ваше приложение не подходит для его устройства…

Особенности OpenGL ES: будьте гибкими

- EGL – API для конфигурирования и создания буферов и контекстов OpenGL ES
- Напишите свой код для выбора конфигурации EGL
- Будьте готовы “откатиться”, если нужная конфигурация недоступна
- Минимизируйте число абсолютных требований
- Различные платформы имеют различную поддержку
  - Сжатия текстур
  - Анти-алиасинга
  - Форматов буферов цвета и глубины

Аппаратное сжатие текстур

- Аппаратное сжатие текстур необходимо!
- Формат ETC1 – универсальный
  - Но не содержит alpha-канал
- Для сжатия RGBA-текстур разработчики должны использовать форматы производителей GPU:
  - DXT3/5 (S3TC)
  - PVRTC
  - ATITC
- Это значит, необходимо подготовить различные ресурсы для загрузки на целевую платформу
  - Здесь могут помочь фильтры Android Market

Фильтры Android Market

- С фильтрами вы можете быть уверены, что ваше приложение будет работать на устройстве пользователя
- Тэг <supports-gl-texture> позволяет фильтровать по форматам сжатия текстур
- Тэг <uses-feature> позволяет фильтровать по:
  - openGlEsVersion: версии OpenGL ES (например, ES 2.0)
  - android.hardware.touchscreen.multitouch.distinct: наличию точной независимой обработки касаний
  - android.hardware.touchscreen.multitouch.jazzhand: наличию точной обработки >=5 касаний

Выводы

- Тщательно планируйте порт
  - Четко обозначте требования вашей игры
  - Изучите необходимый функционал NDK
  - Выберите целевую платформу API/SDK/рынок
- Используйте лучший инструментарий
- Тщательно настраивайте ваш порт
- Протестируйте ваш порт в различных ситуациях
  - Входящий звонок, низкий заряд акуумулятора, и т.п.
  - И на различных устройствах!
- Используйте фильтры для устройств, которые не поддерживает ваша игра

Спасибо за внимание!

iTERENTIEV@NVIDIA.COM