RenderScript

androidzaurus

yapf #16 12/18/2011

はじめに

はじめに

- 4/7/2011にLinux FoundationのCollaboration Summitで

はじめに

- 4/7/2011にLinux FoundationのCollaboration Summitで
- GoogleのShih-wei Liaoさんが

はじめに

- 4/7/2011にLinux FoundationのCollaboration Summitで
- GoogleのShih-wei Liaoさんが
- Android RenderScript on LLVMというタイトルで発表した

はじめに

- 4/7/2011にLinux FoundationのCollaboration Summitで
- GoogleのShih-wei Liaoさんが
- Android RenderScript on LLVMというタイトルで発表した
- スライドのコピペです。

はじめに

- 4/7/2011にLinux FoundationのCollaboration Summitで
- GoogleのShih-wei Liaoさんが
- Android RenderScript on LLVMというタイトルで発表した
- スライドのコピペです。

がいよう

がいよう

- RenderScriptの概要

がいよう

- RenderScriptの概要
- 部品
  - オフライン・コンパイラ
  - オンライン・コンパイラ
  - RenderScriptランタイム

がいよう

- RenderScriptの概要
- 部品
  - オフライン・コンパイラ
  - オンライン・コンパイラ
  - RenderScriptランタイム
- 合体
  - apkファイル生成: HelloComputeの例
  - apk実行: 初回実行時と端末でのリンク

がいよう

- RenderScriptの概要
- 部品
  - オフライン・コンパイラ
  - オンライン・コンパイラ
  - RenderScriptランタイム
- 合体
  - apkファイル生成: HelloComputeの例
  - apk実行: 初回実行時と端末でのリンク
- まとめ

RenderScriptとは

RenderScriptとは

- Androidの3Dレンダリングと数値計算
  - 可搬性（ARM/x86）
  - パフォーマンス
  - 使いやすさ

RenderScriptとは

- Androidの3Dレンダリングと数値計算
  - 可搬性（ARM/x86）
  - パフォーマンス
  - 使いやすさ
- C99にいくつかの拡張
  - ベクタ操作
  - 関数オーバーローディング
  - rsForEach

RenderScriptとは

- Androidの3Dレンダリングと数値計算
  - 可搬性（ARM/x86）
  - パフォーマンス
  - 使いやすさ
- C99にいくつかの拡張
  - ベクタ操作
  - 関数オーバーローディング
  - rsForEach
- 利用例
  - ライブ壁紙と3Dランチャ
  - YouTubeアプリ

RenderScriptとは

- Androidの3Dレンダリングと数値計算
  - 可搬性（ARM/x86）
  - パフォーマンス
  - 使いやすさ
- C99にいくつかの拡張
  - ベクタ操作
  - 関数オーバーローディング
  - rsForEach
- 利用例
  - ライブ壁紙と3Dランチャ
  - YouTubeアプリ
- コンパイラとランタイムの難しさ

RenderScriptの部品

RenderScriptの部品

- オフライン・コンパイラ（llvm-rs-cc）
  - スクリプトファイルをbitcodeとリフレクションされたJavaレイヤに変換

RenderScriptの部品

- オフライン・コンパイラ（llvm-rs-cc）
  - スクリプトファイルをbitcodeとリフレクションされたJavaレイヤに変換
- オンラインJITコンパイラ（libbcc）
  - 可搬型bitcodeを適切な機械語に変換（CPU/GPU/DSPなど）

RenderScriptの部品

- オフライン・コンパイラ（llvm-rs-cc）
  - スクリプトファイルをbitcodeとリフレクションされたJavaレイヤに変換
- オンラインJITコンパイラ（libbcc）
  - 可搬型bitcodeを適切な機械語に変換（CPU/GPU/DSPなど）
- ランタイム・ライブラリ・サポート（libRS）
  - Dalvikレイヤからスクリプトを管理
  - 基本的なサポートライブラリを提供（数値演算など）

オフライン・コンパイラ llvm-rs-cc

オフライン・コンパイラ llvm-rs-cc

- Clang/LLVMから派生
  - 人気のあるオープンソース・コンパイラ・プロジェクト
  - Clang - C/C++/ObjCのコンパイラのフロントエンド
  - LLVM - 最適化コンパイラのバックエンド

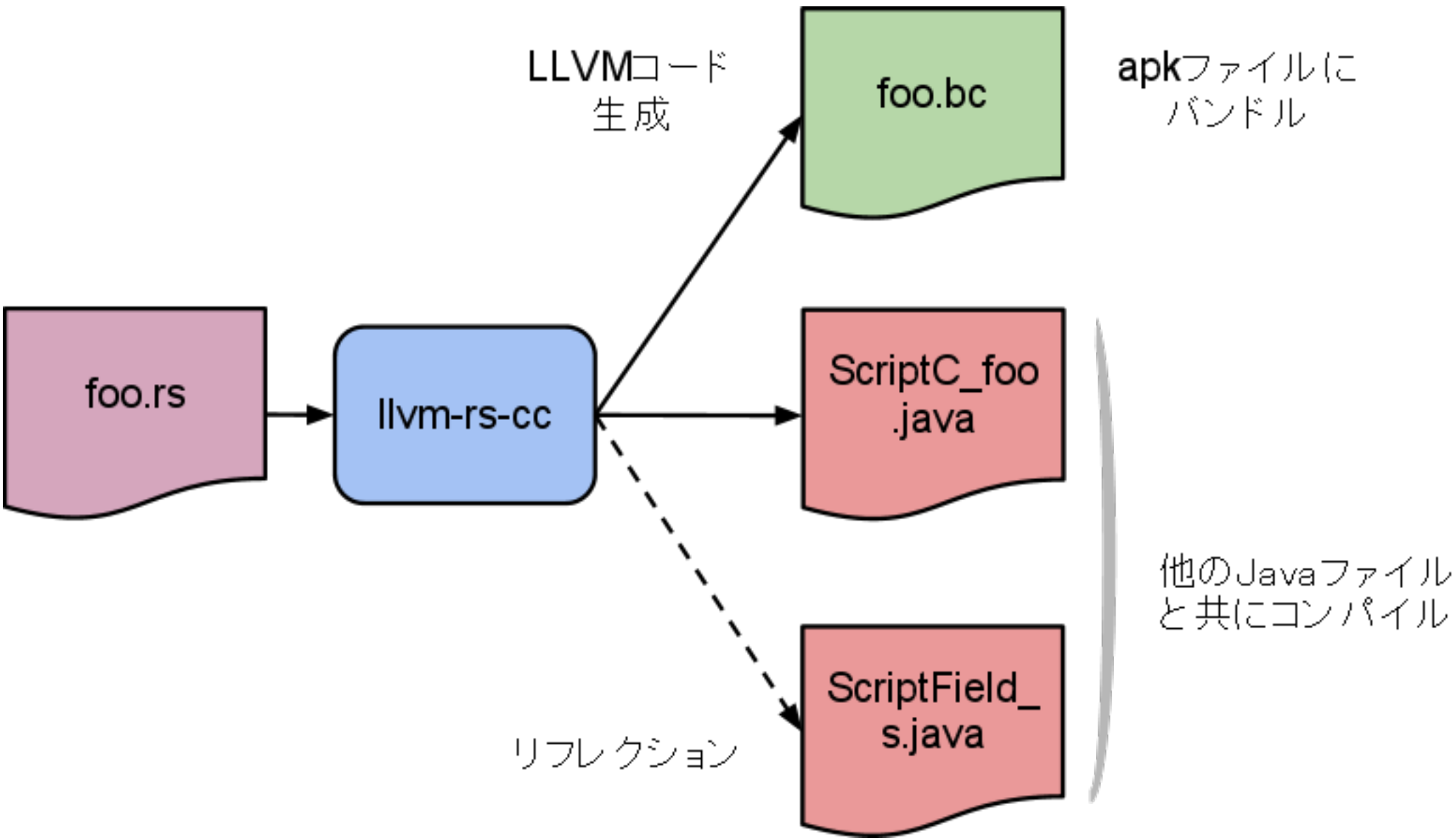
オフライン・コンパイラ llvm-rs-cc

- Clang/LLVMから派生
  - 人気のあるオープンソース・コンパイラ・プロジェクト
  - Clang - C/C++/ObjCのコンパイラのフロントエンド
  - LLVM - 最適化コンパイラのバックエンド
- 主な機能
  - Clangのabstract syntax tree(AST)をうまく利用して、Javaレイヤに情報をリフレクションで戻す
  - bitcodeにメタデータを埋め込む（型など）

オフライン・コンパイラ llvm-rs-cc

- Clang/LLVMから派生
  - 人気のあるオープンソース・コンパイラ・プロジェクト
  - Clang - C/C++/ObjCのコンパイラのフロントエンド
  - LLVM - 最適化コンパイラのバックエンド
- 主な機能
  - Clangのabstract syntax tree(AST)をうまく利用して、Javaレイヤに情報をリフレクションで戻す
  - bitcodeにメタデータを埋め込む（型など）
- すべてのbitcodeはapkにリソースとして提供される

オフライン・コンパイラのフロー



オンラインJITコンパイラ: libbcc

オンラインJITコンパイラ: libbcc

- LLVMベース: Android向けに軽量化
  - 現在ARM/x86をサポート
  - 将来GPU/DSPをターゲットに

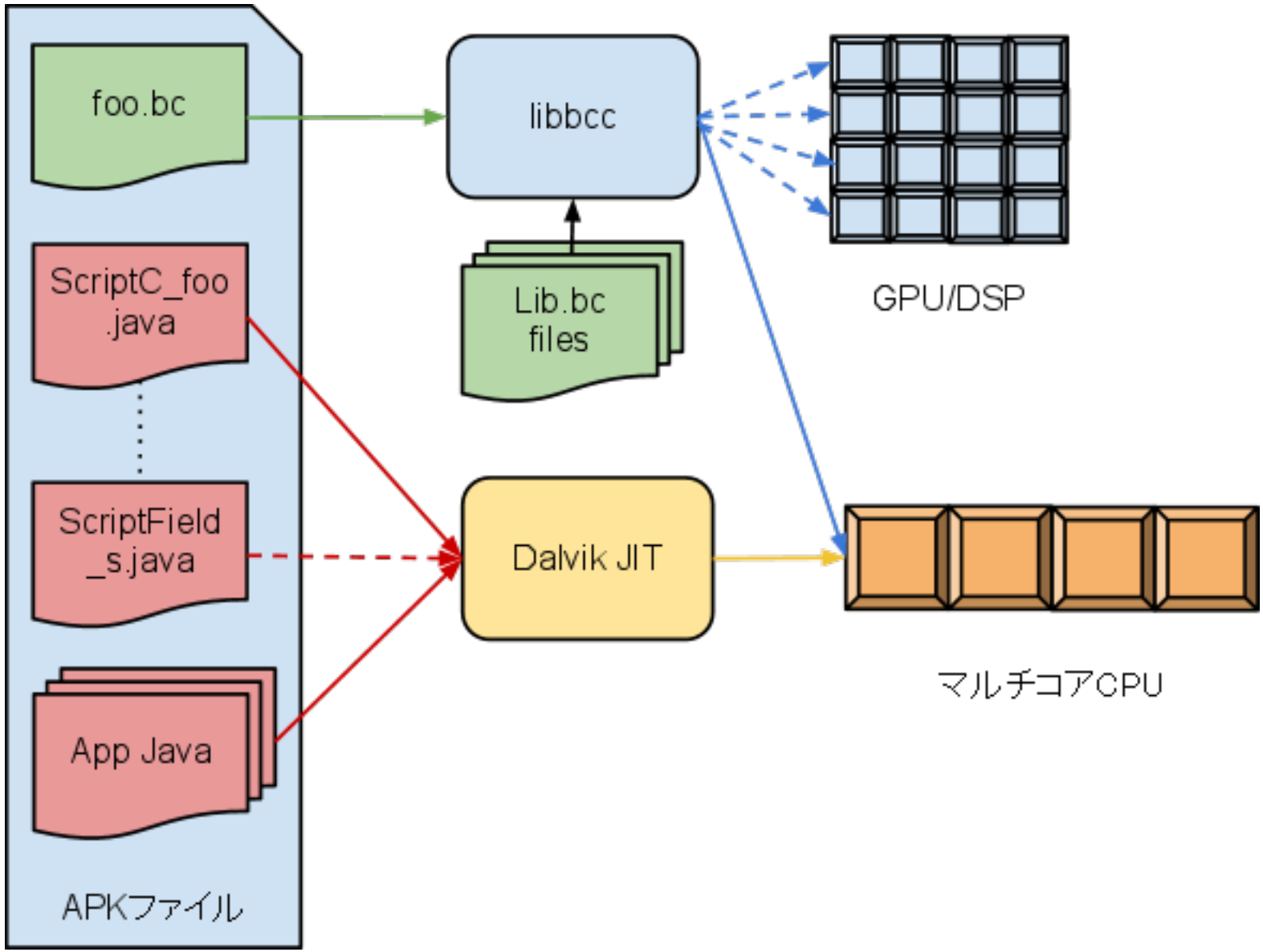
オンラインJITコンパイラ: libbcc

- LLVMベース: Android向けに軽量化
  - 現在ARM/x86をサポート
  - 将来GPU/DSPをターゲットに
- ターゲット向け最適化とコード生成

オンラインJITコンパイラ: libbcc

- LLVMベース: Android向けに軽量化
  - 現在ARM/x86をサポート
  - 将来GPU/DSPをターゲットに
- ターゲット向け最適化とコード生成
- 主要な機能
  - libRSのためのリフレクション: 埋め込まれたメタデータへアクセスするフックを提供
  - JIT済みスクリプトのキャッシュで起動時間を短縮
  - 端末上でのリンク: 数値演算ライブラリなどと動的リンク

オンライン・コンパイラのフロー



RenderScriptランタイム: libRS

RenderScriptランタイム: libRS

- スクリプトとRenderScriptオブジェクトの管理
  - 色々な配列やプリミティブタイプ

RenderScriptランタイム: libRS

- スクリプトとRenderScriptオブジェクトの管理
  - 色々な配列やプリミティブタイプ
  - Script, Type, Element, Allocation, Mesh, ProgramFragment, ProgramRaster, ProgramStore ProgramVertex, Sampler

RenderScriptランタイム: libRS

- スクリプトとRenderScriptオブジェクトの管理
  - 色々な配列やプリミティブタイプ
  - Script, Type, Element, Allocation, Mesh, ProgramFragment, ProgramRaster, ProgramStore ProgramVertex, Sampler
- ランタイムライブラリの提供（数値演算、時間、描画、参照カウンタ）

RenderScriptランタイム: libRS

- スクリプトとRenderScriptオブジェクトの管理
  - 色々な配列やプリミティブタイプ
  - Script, Type, Element, Allocation, Mesh, ProgramFragment, ProgramRaster, ProgramStore ProgramVertex, Sampler
- ランタイムライブラリの提供（数値演算、時間、描画、参照カウンタ）
- アロケーションとスクリプト変数オブジェクトとのバインディング

RenderScriptランタイム: libRS

- スクリプトとRenderScriptオブジェクトの管理
  - 色々な配列やプリミティブタイプ
  - Script, Type, Element, Allocation, Mesh, ProgramFragment, ProgramRaster, ProgramStore ProgramVertex, Sampler
- ランタイムライブラリの提供（数値演算、時間、描画、参照カウンタ）
- アロケーションとスクリプト変数オブジェクトとのバインディング
- 負荷分散（マルチスレッド）

RenderScriptランタイム: libRS

- スクリプトとRenderScriptオブジェクトの管理
  - 色々な配列やプリミティブタイプ
  - Script, Type, Element, Allocation, Mesh, ProgramFragment, ProgramRaster, ProgramStore ProgramVertex, Sampler
- ランタイムライブラリの提供（数値演算、時間、描画、参照カウンタ）
- アロケーションとスクリプト変数オブジェクトとのバインディング
- 負荷分散（マルチスレッド）
- Dalvikとスクリプト間のメッセージ伝達

HelloComputeサンプル

HelloComputeサンプル

- Honeycomb SDKサンプル

HelloComputeサンプル

- Honeycomb SDKサンプル
- ビットマップをグレースケールに変換

HelloComputeサンプル

- Honeycomb SDKサンプル
- ビットマップをグレースケールに変換
- rsForEachを使い、ピクセルごとに並列化
  - RGB値の単純な内積

HelloComputeサンプル

- Honeycomb SDKサンプル
- ビットマップをグレースケールに変換
- rsForEachを使い、ピクセルごとに並列化
  - RGB値の単純な内積
- mono.rs → mono.bc + ScriptC_mono.javaのリフレクション

サンプルスクリプト（mono.rs）

```
#pragma version(1)
#pragma rs java_package_name(com.example.android.rs.
hellocompute)

const static float3 gMonoMult = {0.299f, 0.587f, 0.114f};

void root(const uchar4 *v_in, uchar4 *v_out) {
    float4 f4 = rsUnpackColor8888(*v_in);

    float3 mono = dot(f4.rgb, gMonoMult);
    *v_out = rsPackColorTo8888(mono);
}
```

生成されるScriptC_mono.java 1/2

```
package com.example.android.rs.hellocompute;
import android.renderscript.*;
import android.content.res.Resources;

public class ScriptC_mono extends ScriptC {
    // Constructor
    public  ScriptC_mono(RenderScript rs, Resources resources,
    int id) {
        super(rs, resources, id);
        __U8_4 = Element.U8_4(rs);
    }
    private Element __U8_4;
    private final static int mExportForEachIdx_root = 0;
```

生成されるScriptC_mono.java 2/2 抜粋

```
    public void forEach_root(Allocation ain, Allocation aout) {
        // check ain
        if (!ain.getType().getElement().isCompatible(__U8_4)) {
            throw new RSRuntimeException(
                "Type mismatch with U8_4!");
        }
        // check aout
        if (!aout.getType().getElement().isCompatible(__U8_4)) {
            throw new RSRuntimeException(
                "Type mismatch with U8_4!");
        }
        forEach(mExportForEachIdx_root, ain, aout, null);
    }
}
```

HelloCompute.java 抜粋

```
private void createScript() {
    mRS = RenderScript.create(this);
    mInAllocation = Allocation.createFromBitmap(
                                        mRS, mBitmapIn,
                                        Allocation.MipmapControl.MIP
                                        MAP_NONE,
                                        Allocation.USAGE_SCRIPT);
    mOutAllocation = Allocation.createTyped(
                                            mRS, mInAllocation.getType());
    mScript = new ScriptC_mono(mRS, getResources(), R.raw.mono);
    mScript.forEach_root(mInAllocation, mOutAllocation);
    mOutAllocation.copyTo(mBitmapOut);
}
```

端末でのリンク

端末でのリンク

設計

端末でのリンク

設計

- ランタイムライブラリ（lib*.bc）を提供
  - CPU/GPUベンダが独自のlib*.bcを提供可能に
  - 関数実行の遅延を減らす

端末でのリンク

設計

- ランタイムライブラリ（lib*.bc）を提供
  - CPU/GPUベンダが独自のlib*.bcを提供可能に
  - 関数実行の遅延を減らす
- RenderScriptランタイム（libclcore.bc）での例
  - ベクトル演算
  - 機種により異なる
    - XoomではVFP3-16
    - Nexus SではNEON

端末でのリンク

設計

- ランタイムライブラリ（lib*.bc）を提供
  - CPU/GPUベンダが独自のlib*.bcを提供可能に
  - 関数実行の遅延を減らす
- RenderScriptランタイム（libclcore.bc）での例
  - ベクトル演算
  - 機種により異なる
    - XoomではVFP3-16
    - Nexus SではNEON

実装

端末でのリンク

設計

- ランタイムライブラリ（lib*.bc）を提供
  - CPU/GPUベンダが独自のlib*.bcを提供可能に
  - 関数実行の遅延を減らす
- RenderScriptランタイム（libclcore.bc）での例
  - ベクトル演算
  - 機種により異なる
    - XoomではVFP3-16
    - Nexus SではNEON

実装

- LLVM bitcodeレベルでのリンク

端末でのリンク

設計

- ランタイムライブラリ（lib*.bc）を提供
  - CPU/GPUベンダが独自のlib*.bcを提供可能に
  - 関数実行の遅延を減らす
- RenderScriptランタイム（libclcore.bc）での例
  - ベクトル演算
  - 機種により異なる
    - XoomではVFP3-16
    - Nexus SではNEON

実装

- LLVM bitcodeレベルでのリンク
- リンク時の最適化: インライン化、固定値のプロパゲーション

まとめ

まとめ

- RenderScript
  - 移植性、パフォーマンスにすぐれ、開発しやすい
  - 3Dと数値演算のアクセラレーション

まとめ

- RenderScript
  - 移植性、パフォーマンスにすぐれ、開発しやすい
  - 3Dと数値演算のアクセラレーション
- コンパイラとランタイムで複雑さの隠蔽
  - C99準拠 + rsForEach
  - リフレクション多用
  - ライブラリ
  - 隠蔽状態で管理されている型と参照カウンタ

まとめ

- RenderScript
  - 移植性、パフォーマンスにすぐれ、開発しやすい
  - 3Dと数値演算のアクセラレーション
- コンパイラとランタイムで複雑さの隠蔽
  - C99準拠 + rsForEach
  - リフレクション多用
  - ライブラリ
  - 隠蔽状態で管理されている型と参照カウンタ
- 軽量JIT
  - 起動時間の短縮
  - 端末でのリンク

まとめ

- RenderScript
  - 移植性、パフォーマンスにすぐれ、開発しやすい
  - 3Dと数値演算のアクセラレーション
- コンパイラとランタイムで複雑さの隠蔽
  - C99準拠 + rsForEach
  - リフレクション多用
  - ライブラリ
  - 隠蔽状態で管理されている型と参照カウンタ
- 軽量JIT
  - 起動時間の短縮
  - 端末でのリンク
- 最新情報は http://developer.android.com

おまけ

```
$ cd platform-tools/renderscript/include/
$ ls
rs_allocation.rsh  rs_core.rsh      rs_math.rsh    rs_quaternion.rsh
rs_atomic.rsh      rs_debug.rsh     rs_matrix.rsh  rs_time.rsh
rs_cl.rsh          rs_graphics.rsh  rs_object.rsh  rs_types.rsh
```

おまけ2

- rs_matrix.rsh
  - rsMatrixLoad(rs_matrix4x4 *m, const float *v);
  - rsMatrixMultiply(rs_matrix4x4 *m, rs_matrix4x4 *rhs);
- rs_cl.rsh
  - 三角関数（float, float2, ..., float4）
  - muladd（float, float2, ..., float4）
  - 内積
- rs_math.rsh
  - rsRand
  - rsExtractFrustumPlanes
- rs_debug.rsh
  - rsDebug

ありがとうございました