工程の手戻りを最小限に
2Dエンジン活用における傾向と対策

小高 輝真(株式会社ウェブテクノロジ)
東田 弘樹(フリーランス・プログラマ)

このスライド(暫定公開版)は、CEDEC 2014 の講演内容です。
http://cedec.cesa.or.jp/2014/session/ENG/6589.html
後日、口頭で説明した内容を補足したバージョンを
http://www.webtech.co.jp/blog/ で公開予定です。

Ver. 2014/9/8

OPTPiX

WebTechnology®

工程の手戻りを最小限に

2Dエンジン活用における傾向と対策

セッションの内容

ゲームエンジンを活用して2Dゲームアプリを開発する際に、事前の調査をしっかり行うことで、手戻りを起こさないための情報を提供します。
UnityとCocos2d-xで動作する同じ仕様のゲーム風アプリを使い、各種端末で性能比較を行います。
Unity、Cocos2d-x上で2Dゲームを開発する際のTipsをご紹介します。

受講スキル

- 小規模チームで、スマホ向け2Dゲーム開発に関わるプログラマ
- 幅広いユーザーに向けたスマホ向け2Dゲームの開発者

受講者が得られる知見

- 2Dゲームエンジンで、性能見積をするための実際的な情報(テストアプリのソース付)
- 主に画像の取扱に関するTips

対象プラットフォーム **Android, iOS**

WebTechnology®

アジェンダ

第１部　2Dゲームエンジンとは

- 主なゲームエンジン、特にUnityとCocos2d-xについてのご紹介

第２部　スプライト描画負荷のベンチマーク

- Cocos2d-x v3とUnityの描画性能差
- Cocos2d-x v2とv3の描画性能差
- 機種による描画性能差
- バッチについて

第３部

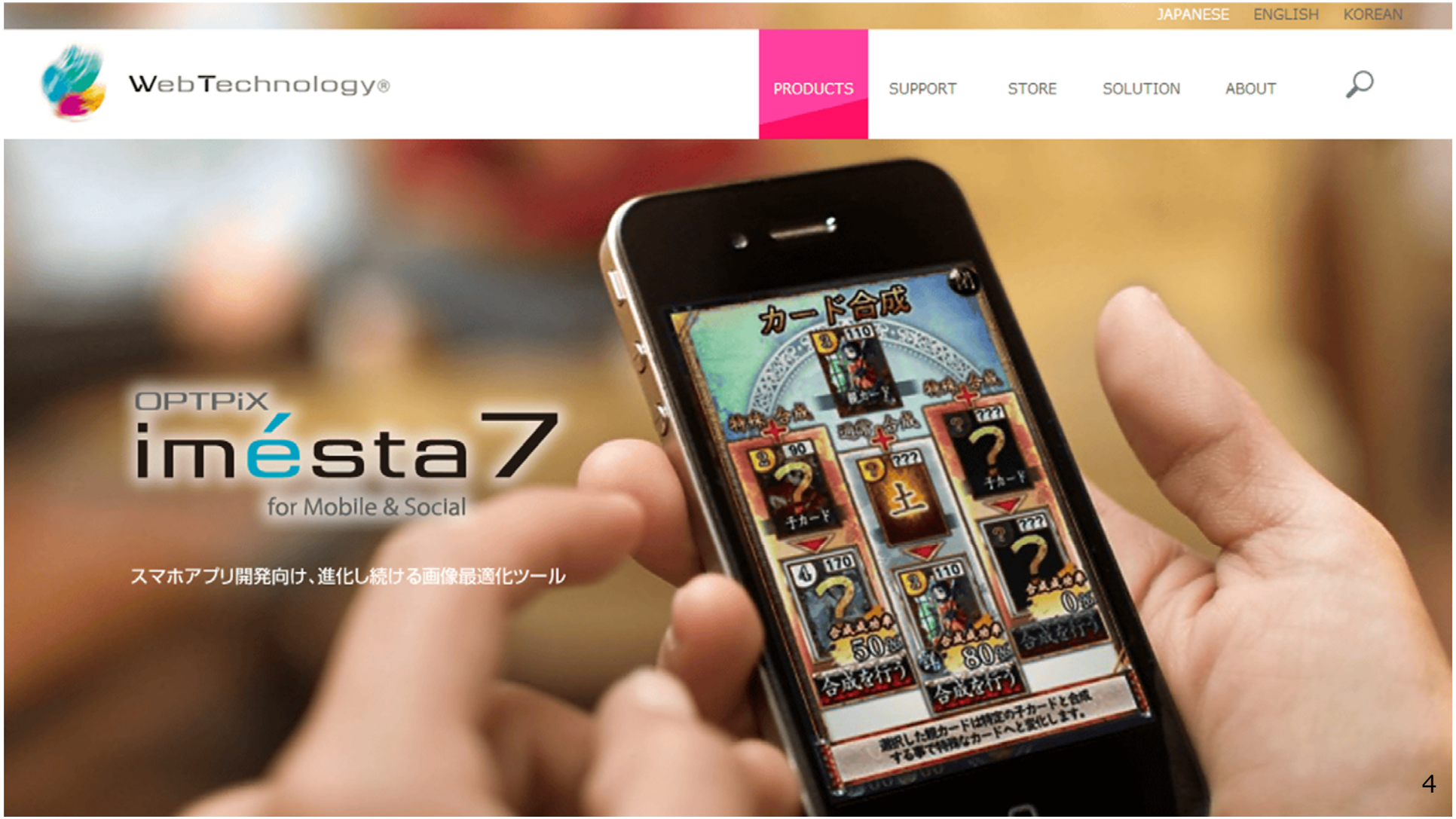
- 8つのTips

講演者紹介

小高　輝真

株式会社ウェブテクノロジ　代表取締役
元プログラマ
元テクニカルライタ
CEDEC運営委員
スポンサープログラムWGリーダー

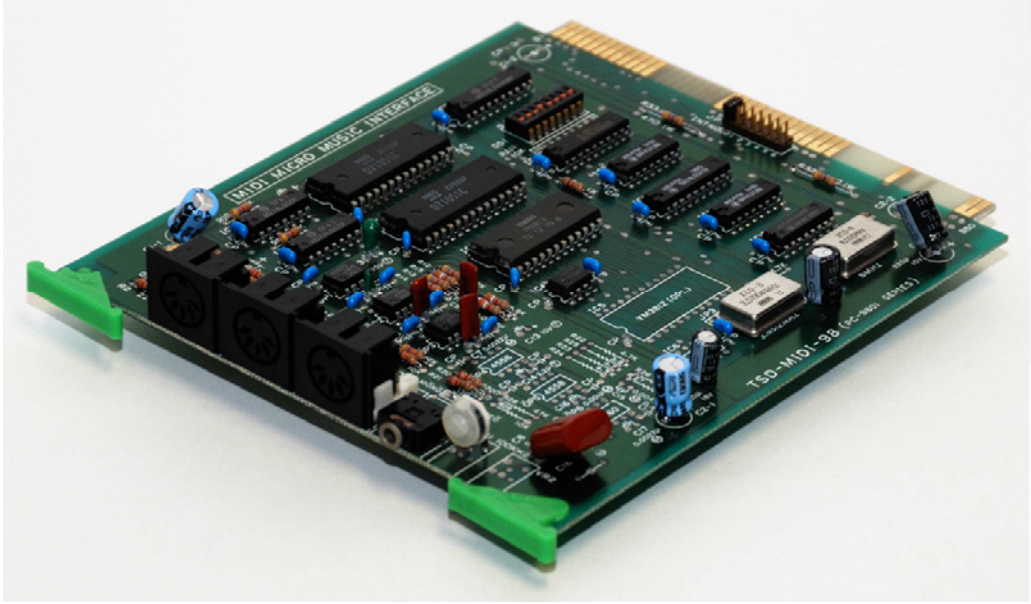
写真は、GameBusiness.jp
「OPTPiXはこうして生まれた！
ウェブテクノロジ設立物語（前編）」より
http://www.gamebusiness.jp/article.php?id=9350

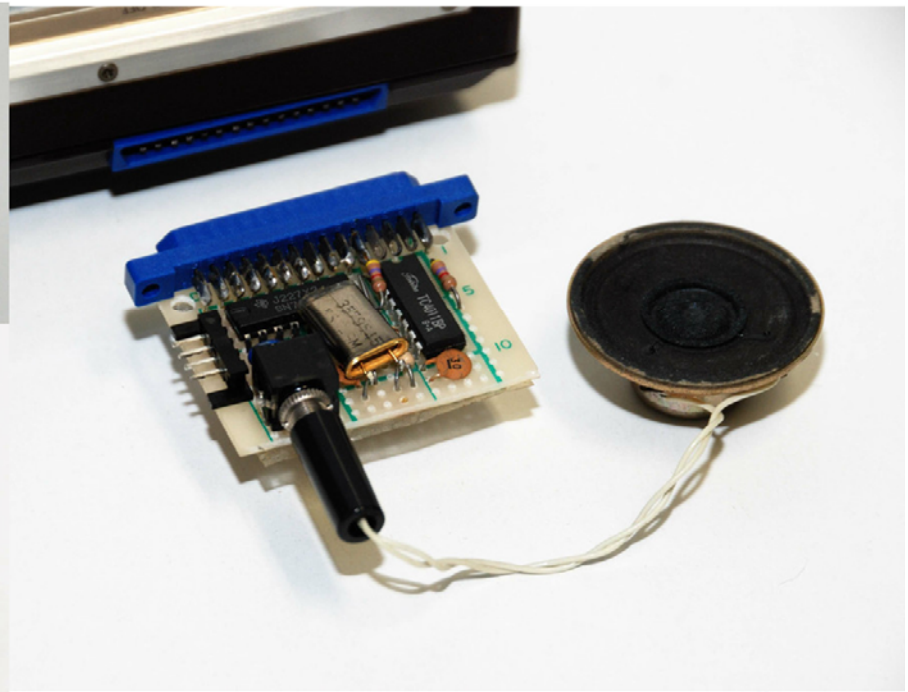

JAPANESE ENGLISH KOREAN
WebTechnology®
PRODUCTS
SUPPORT
STORE
SOLUTION
ABOUT
OPTPiX
imésta 7
for Mobile & Social
スマホアプリ開発向け、進化し続ける画像最適化ツール
カード合成


JAPANESE ENGLISH KOREAN
WebTechnology®
PRODUCTS
SUPPORT
STORE
SOLUTION
ABOUT
OPTPiX
SpriteStudio®
超汎用2Dスプライトアニメーション制作
OPTPiX Help Center
OPTPiX SpriteStudio for Indie

3Dキャラで組み立てる、マンガ作成ツール

講演者紹介　小高　輝真





WebTechnology®

講演者紹介　小高　輝真


PC-9801
スーパーテクニック
小高輝真・清水和文・遠水祐 共著
アスキー出版局


アンドキュメンテイド
UNDOCUMENTED
9801/9821
Vol.
1
インプレス

講演者紹介

東田 弘樹

フリーランス・プログラマ

ゲームボーイの時代頃よりゲームプログラマとして従事。
フリーランスになってからゲーム以外のプログラム開発案件にも関わりつつ
最近はゲーム開発を支援する裏方として、ツール開発などに従事しています。

第１部

2Dゲームエンジンとは

2Dゲームエンジンとは

このセッションでは、2Dゲームエンジンについて

「2Dゲームを作成するために使いやすいゲームエンジン」

と定義

3Dエンジンは除くが、Unityは現在2Dゲーム制作にも広く使われており、
Unityには2D機能も提供されているため、このセッションの対象とする

2Dゲームエンジンのメリット

- マルチプラットフォーム対応
  ゲームエンジンがプラットフォームごとの差異を吸収してくれるので、
  １ソースでマルチプラットフォームの開発が行える

- 高速化
  ハードウェアの性能を出すにはOpenGLの使用が必須
  しかし、OpenGLで2Dの表現をするのは案外面倒
  ゲームエンジンが肩代わりしてくれる

- 時間節約
  2Dであれば、比較的定型の処理が多い
  ゲームエンジンが必要な機能を備えている

代表的な2Dゲームエンジン

Unity

Cocos2d-x

Starling

Sprite Kit

Corona SDK

Marmalade SDK

…

Unity

主な特徴

- プロトタイプの開発が早い。独自エディタを使っての開発が主
- 2Dに3Dの表現を組み合わせることができる
- 自分が作りたいゲームにあったAsset(サードパーティ製のプラグインのようなもの)が見つかれば、非常に生産性が高くなる
- 対応プラットフォーム: iOS, Android, Windows, Mac, Linux, Webブラウザ, PlayStation® 4/3/Vita, Xbox ONE/360, Wii U™
- ライセンス等: Unity Free 無料(制限あり)、 Unity Pro 162,000円
- 開発言語: C#, JavaScript, Boo
- IDE: Unity エディタ, Mono Develop 4（替わりにVisualStudioも使用可 ※UnityVS）
- 物理エンジン: あり
- パーティクル: あり
- http://japan.unity3d.com/

Cocos2d-x

主な特徴

- 2Dゲームエンジン
  C++での開発に馴れたコンシューマ・ゲーム開発経験者には使いやすい
- 対応プラットフォーム: iOS 5.0以降, Android 2.3以降, Windows 7以降, OS X 10.6以降, Windows Phone 8
- ライセンス等: 無償, オープンソース(MITライセンス)
- 開発言語: C++(C++11), Lua/JavaScriptを使っての開発も可
- 使用可能IDE: Visual Studio 2012以降, Xcode 4.6以降
- 物理エンジン: あり(Chipmunk2D, Box2D)
- パーティクル: あり
- http://www.cocos2d-x.org/

上記はCocos2d-x v3の情報です

Starling

主な特徴

- AIR/Flash上で動く
- 対応プラットフォーム: AIR/Flashが動くもの(iOS, Android, Windows, Mac, Linux, Webブラウザ)
- ライセンス等: 無償利用可, オープンソース
- 開発言語: ActionScript 3
- 使用可能IDE :Flash Builder 4.7, FlashDevelop
- 日本語情報: かなり少ない
- 物理エンジン: 標準ではなし
- パーティクル: あり
- http://gamua.com/starling/
- GPUで描画（Stage3Dを使用）
- AIRというと別途ランタイムのインストールが必要そうだが、iOS, Androidはアプリの中にランタイムが組み込まれる
- 気になること…Adobe がAIR/Flashをいつまでサポートするのか

Sprite Kit

主な特徴

- iOS, Mac OS X標準の2Dグラフィック・フレームワーク
- 対応プラットフォーム: iOS 7.0以降, OS X 10.9以降
- ライセンス等: 無償利用可, ソース非公開
- 開発言語: Objective-C(C++, C), Swift
- 使用可能IDE：Xcode
- 日本語情報: 若干。日本語書籍は1冊(2014/8現在)
- 物理エンジン: あり
- パーティクル: あり
- https://developer.apple.com/library/ios/documentation/GraphicsAnimation/Conceptual/SpriteKit_PG/Introduction/Introduction.html
- 気になること… Androidとのマルチプラットフォーム開発ができない

Corona SDK

主な特徴

- クラウドコンパイルのため環境構築がラク
- ライトゲームの作成に向いている
- 対応プラットフォーム: iOS 5.1以降, Android 2.2以降, Windows Phone 8(予定)
- ライセンス等: US$16.00/月～
- 開発言語: Lua ※ライセンス次第でネイティブ言語も合わせて使用可
- 使用可能IDE：Corona Editor（Sublime Text用プラグイン）
- 日本語情報: かなり少ない
- 物理エンジン: あり(Box2D)
- パーティクル: あり
- http://coronalabs.com/
- 気になること
  - クラウドコンパイルのため、サービス継続性のリスク
  - ソースをクラウドにアップロードする必要があるため、それが許容できるかどうか
  - Enterpriseライセンスではオフラインビルドも可能

Marmalade SDK

主な特徴

- 2D/3Dの開発が可能
- 対応プラットフォーム: iOS 5.0以降, Android 2.2以降, Windows Phone 8, BlackBerry, TIZEN, Windows, Mac（Win/MacはUS$499/月ライセンスから可）
- ライセンス等: フリー版あり（サポート限定）, US$15/月～
- 開発言語: C++
- 使用可能IDE: Visual Studio, Xcode
- 日本語情報: かなり少ない
- https://www.madewithmarmalade.com/
- ミドルウェアとの組み合わせで、LuaもしくはObjective-Cを使った開発も可
- Objective-Cを使った組み合わせでは、iOS向けに作ったプロジェクトをAndroidに移植するような用途にも使える

代表的な2Dゲームエンジン

Unity

Cocos2d-x

Starling

Sprite Kit

Corona SDK

Marmalade SDK

…

OPTPiX®

Unity

- 最新版はv4.5.3、v4.6がオープンβ中。v5はまもなく登場！
- 情報量とサポート
  - Unityテクノロジーズジャパン合同会社のサポートが受けられる
  - 日本語書籍は20冊以上（ただ3D機能についてが中心）
  - ネット上にも日本語情報が溢れている

Unity

- v4.6で、UIエディタが搭載
- Unity Cloud Build（β）
  - Git/svnのリポジトリを登録しておくと、自動的にビルド、配信(テスト向け)まで行ってくれる
  - Unity Pro限定

Cocos2d-x

- 最新版はv3.2、v2はv2.2.5
- 情報量
  - 日本語書籍は6冊程度。ただし、v2のものがほとんど
  - ネット上の日本語情報はかなり増えてきている

Cocos2d-x

v3になって何が変わったか？

- Objective-C的パターンからC++11へ
- 新しいレンダラーの導入
  - 自動バッチング、自動カリング、グローバルZオーダー導入

- ラベルの改善
- イベント処理の改善
  - タッチ処理がより扱いやすく
- 3Dモデルが表示可能に（v3.1から）

https://github.com/cocos2d/cocos2d-x/blob/cocos2d-x-3.0/docs/RELEASE_NOTES.md#user-content-highlights-of-v30

Cocos2d-x

気になるところ…

サウンドの機能が弱い、v3でも変わらず

- 最低限のAPIしか用意されていない
- 音ゲーのようなもをお考えの方は事前確認を
- ミドルウェアに頼る手も　CRI ADX2 for Smartphone

第2部

スプライト描画負荷のベンチマーク

基本的なスプライト描画負荷テストアプリ

- 画面上にスプライトを増やしていき、その描画負荷を計測
- スプライトはアニメーションなし
- 画面タッチで描画スプライト数が増加
- 使用テクスチャの枚数により２パターンを検証
  - o 1枚のテクスチャのみで描画
  - o 16枚のテクスチャを順番に切り替えて描画
- テクスチャは、96x96ピクセル、RGBA4444(.pvrに格納)
- アプリ名: 1TexSpr, 16TexSpr
- Unity v4.5.2, Cocos2d-x v3.2, v2.2.5(Batchあり・なし)で検証

OPTPiX

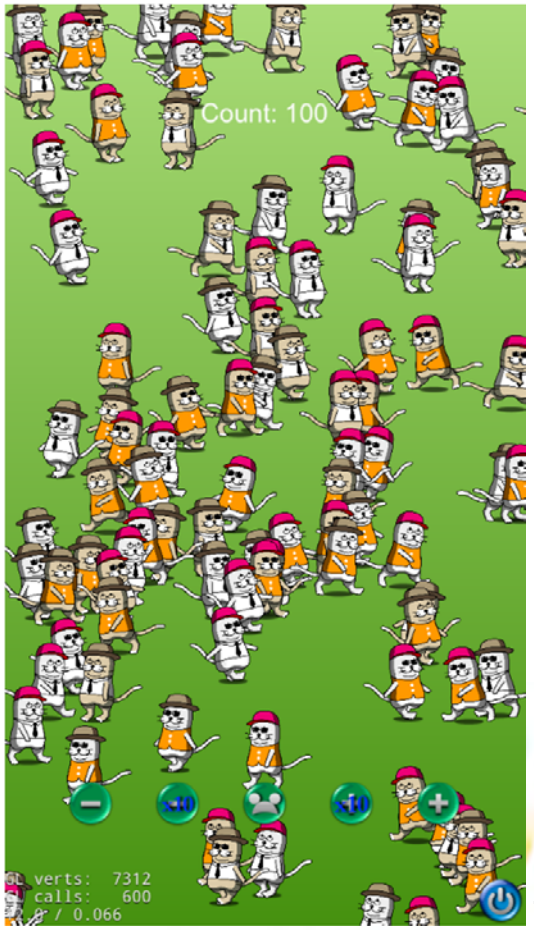
基本的なスプライト描画負荷テストアプリ

96 x 96
(RGBA4444) x1枚

96 x 96
(RGBA4444) x16枚

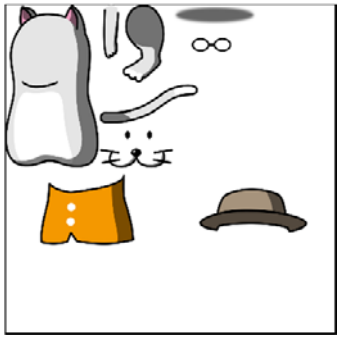
ゲーム風アプリ

- 画面上に多関節アニメするキャラクターを増やしていき、
  その描画負荷を計測
- キャラクターは11パーツ（スプライト）で構成
- ボタンでキャラクターを増減可能
- サウンドあり(BGMをMP3で再生)
- 使用テクスチャの構成により３パターンで検証
  - １枚のみ
  - 白ネコ、黒ネコのテクスチャ２枚
  - 白ネコ、黒ネコ、洋服、帽子、めがね、
    などテクスチャ８枚
- テクスチャフォーマットは、RGBA4444(.pvrに格納)
- アプリ名: Multi
- Unity v4.5.2, Cocos2d-x v3.2, v2.2.5(Batchあり・なし)で検証
  ※テクスチャ８枚構成のv2 Batchありは未計測

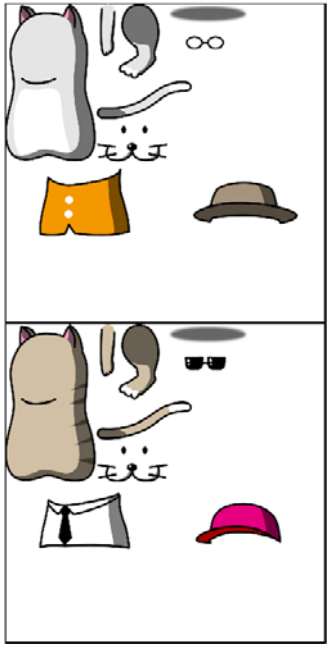


OPTPiX

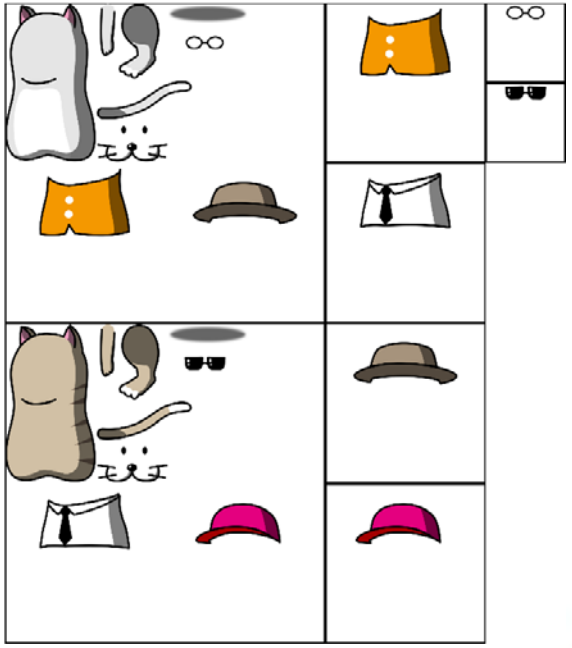
ゲーム風アプリ



1024 x 1024
(RGBA4444) x1枚



1024 x 1024
(RGBA4444) x2枚



8枚　※装備品は単体画像側を使用
1024 x2枚, 512 x4枚, 256 x2枚

スプライト描画性能比較　検証端末

	端末名	発売日	CPU	画面解像度	備考
1	Xperia arc (docomo SO-01C)	2011/03	Qualcomm Snapdragon MSM8255 1GHz Qualcomm Adreno 205 GPU	480×854	Android 2.3.4
2	Nexus 5 (LG-D821)	2013/11	Qualcomm Snapdragon MSM8974 2.26GHz(クアッドコア) Qualcomm Adreno 330 GPU	1920×1080	Android 4.4.0
3	iPhone 4	2010/6	Cortex-A8(Single) 800MHz PowerVR SGX535	960x640	
4	iPhone 4S	2011/10	Coretex-A9(Dual) 800MHz PowerVR SGX543MP2(Dual)	960x640	
5	iPhone 5	2012/9	ARMv7s(Dual) 1.3GHz PowerVR SGX543MP3(Triple)	1136x640	

スプライト描画性能比較

比較内容

1．Cocos2d-x v3とUnityの描画性能差(2パターン)

2．Cocos2d-x v2とv3の描画性能差(3パターン)

3．機種による描画性能差(4パターン)

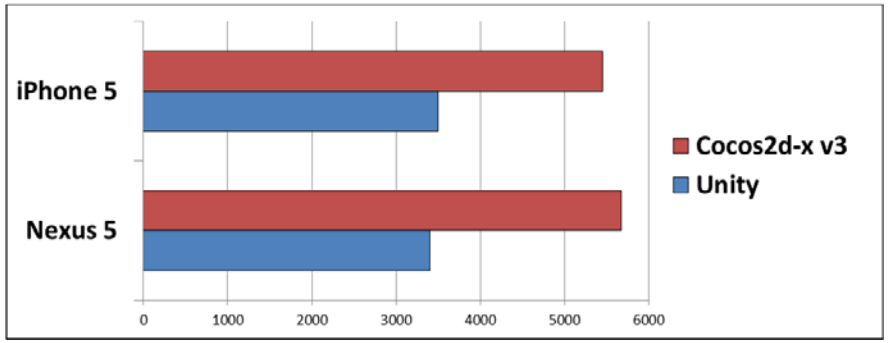
2D描画性能を比較するためにいくつか要素の異なるスプライトを１フレームにいくつ描画できるかを計測
計測方法は、描画するスプライトを徐々に増やしていき、30FPS を維持できる最大数を求めた

スプライト描画性能比較(Part 1)

Cocos2d-x v3とUnityの描画性能差(1)

- 使用アプリ: 1TexSpr (基本的なスプライト描画負荷テストアプリ)
- テストの内容：
  - 1枚のテクスチャ(96x96、RGBA4444)から、スプライトを多数描画するのみ
  - 非常にプリミティブな機能の計測
- テスト結果：

	Cocos2d-x v3	Unity
iPhone 5	5450個	3500個
Nexus 5	5670個	3400個



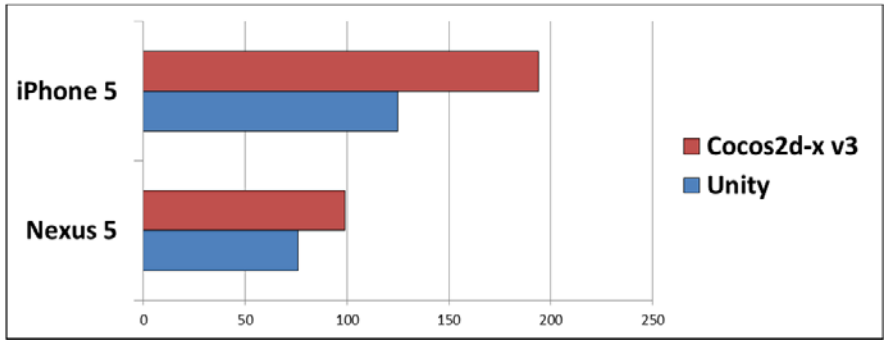
- 考察：
  - **iPhone 5では、Cocos2d-xがUnityの約1.6倍、Nexus 5では約1.7倍のパフォーマンス**
  - iPhone 5とNexus 5はほぼ互角(UnityはiPhone 5が3%優位、Cocos2d-x v3はNexus 5が4%優位)

スプライト描画性能比較(Part 1)

Cocos2d-x v3とUnityの描画性能差(2)

- 使用アプリ: Multi (ゲーム風アプリ)
- テストの内容：
  - 2枚のテクスチャ(1024x1024、RGBA4444)から、複数スプライトで構成されたキャラクタを多数描画
  - UIボタン、サウンドあり
  - 実際のゲームアプリに近い構成
- テスト結果：

	Cocos2d-x v3	Unity
iPhone 5	194個	125個
Nexus 5	99個	76個

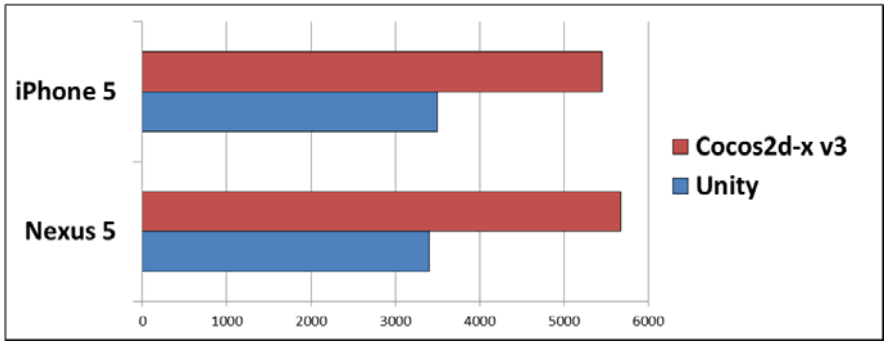


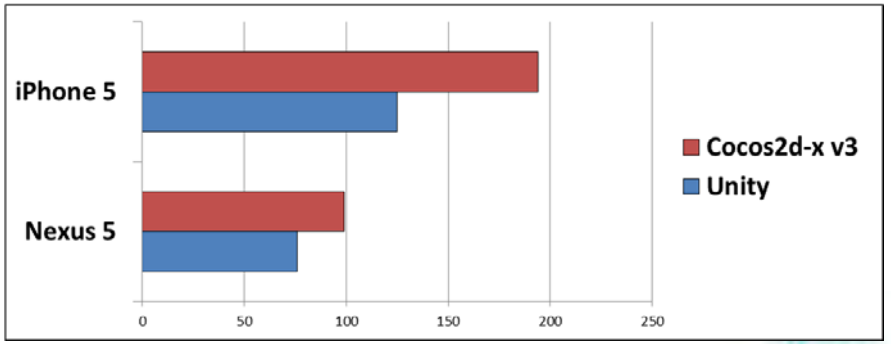
- 考察：
  - **iPhone 5では、Cocos2d-xが Unityの約1.6倍、Nexus 5では約1.3倍のパフォーマンス**
  - この条件ではiPhone 5がかなり優位(Unityで約1.6倍、Cocos2d-x v3で約2倍のパフォーマンス)

スプライト描画性能比較(Part 1)

Cocos2d-x v3とUnityの描画性能差

- 特定のプラットフォーム、特定の機種だけでパフォーマンスチェックしていると、手戻り発生の可能性あり



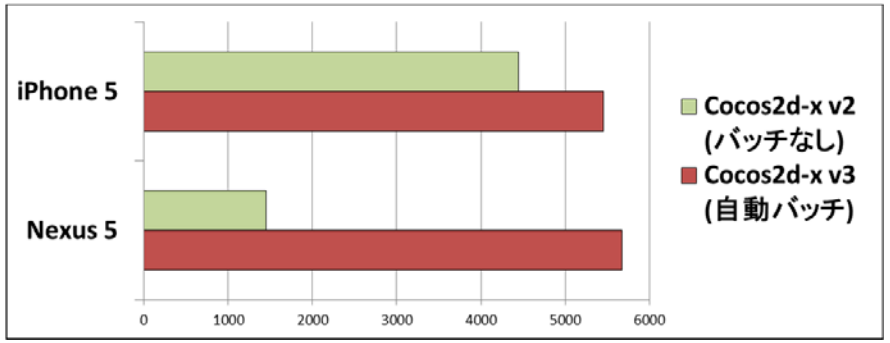


スプライト描画性能比較(Part 2)

Cocos2d-x v2とv3の描画性能差(1)

- 使用アプリ: 1TexSpr (基本的なスプライト描画負荷テストアプリ)
- テストの内容：
  - 1枚のテクスチャ(96x96、RGBA4444)から、スプライトを多数描画するのみ
  - 非常にプリミティブな機能の計測
- テスト結果：

	Cocos2d-x v2 (バッチなし)	Cocos2d-x v3 (自動バッチ)
iPhone 5	4444個	5450個
Nexus 5	1450個	5670個



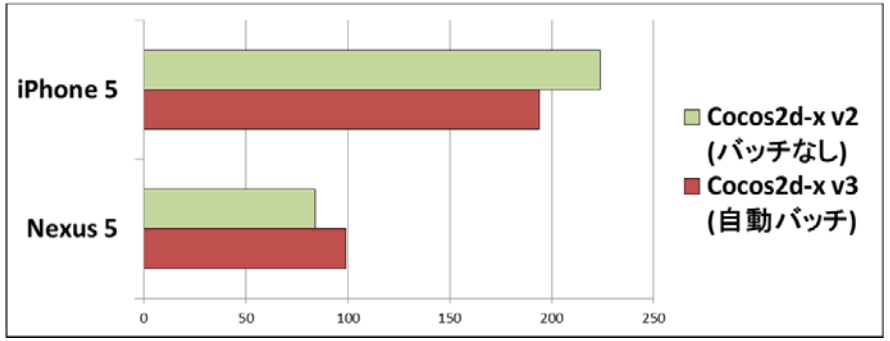
- 考察：
  - **Nexus 5では、v3がv2(バッチなし)の約3.9倍のパフォーマンス。バッチの効果は絶大**
  - **iPhone 5では、v3がv2(バッチなし)の約1.2倍でしかない。というより、バッチなしでも高速**

スプライト描画性能比較(Part 2)

Cocos2d-x v2とv3の描画性能差(2)

- 使用アプリ: Multi (ゲーム風アプリ)
- テストの内容：
  - 2枚のテクスチャ(1024x1024、RGBA4444)から、複数スプライトで構成されたキャラクタを多数描画
  - UIボタン、サウンドあり
  - 実際のゲームアプリに近い構成
- テスト結果：

	Cocos2d-x v2 (バッチなし)	Cocos2d-x v3 (自動バッチ)
iPhone 5	224個	194個
Nexus 5	84個	99個



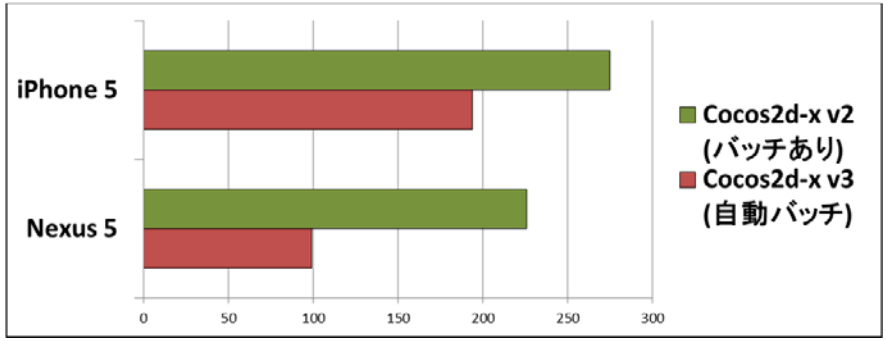
- 考察：
  - **iPhone では、v2(バッチなし)がv3の+15%、Nexus 5 では逆にv3がv2(バッチなし)の+18%のパフォーマンス**
  - **実際のゲームアプリに近い構成では、v3の優位が揺らいでいる**

スプライト描画性能比較(Part 2)

Cocos2d-x v2とv3の描画性能差(3)

- 使用アプリ: Multi (ゲーム風アプリ)
- テストの内容：
  - 2枚のテクスチャ(1024x1024、RGBA4444)から、複数スプライトで構成されたキャラクタを多数描画
  - UIボタン、サウンドあり
  - 実際のゲームアプリに近い構成
- テスト結果：

	Cocos2d-x v2 (バッチあり)	Cocos2d-x v3 (自動バッチ)
iPhone 5	275個	194個
Nexus 5	226個	99個

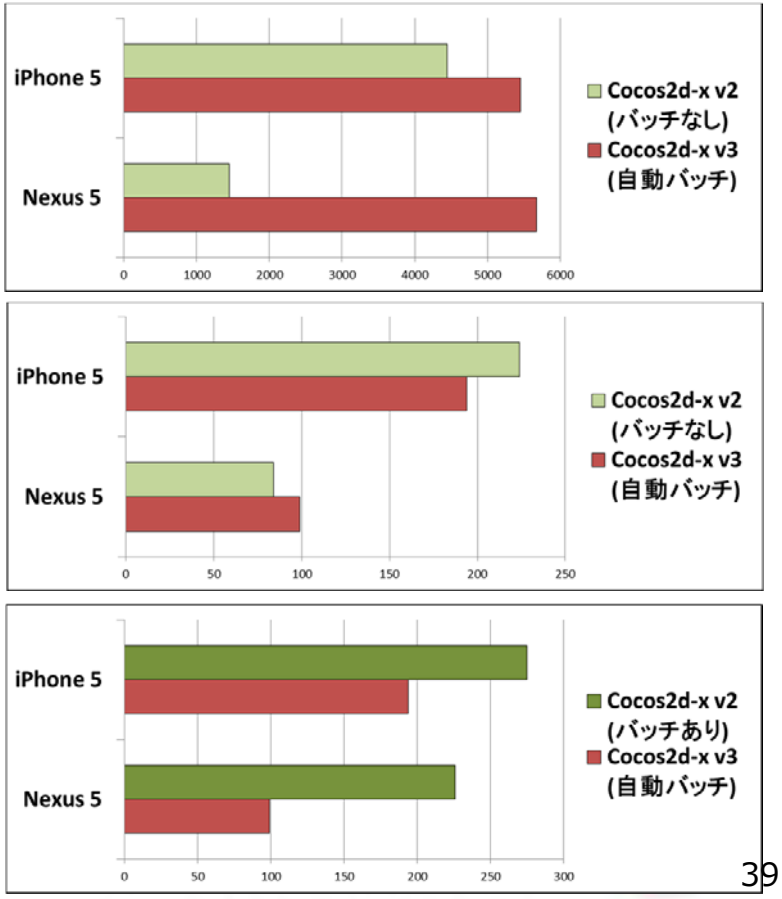


- 考察：
  - **iPhone では、v2(バッチあり)がv3の約1.4倍、Nexus 5 では約2.3倍のパフォーマンス**
  - **Cocs2d-x v2で、手動で理想的なバッチを組めれば、v3よりも高パフォーマンスを出せる**
  - **ただし、このアプリのv2 バッチありでは必ずバッチが効くという理想状態を計測しているので、注意**

スプライト描画性能比較(Part 2)

Cocos2d-x v2とv3の描画性能差

- Cocos2d-x v3で導入された自動バッチは、複雑な描画の場合は、必ずしもバッチなしv2より高性能になるとは限らない



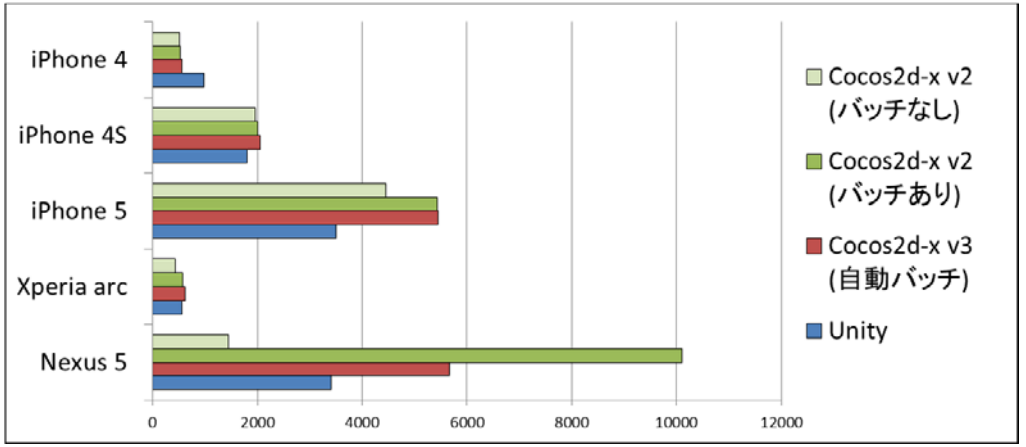
OPTPiX

スプライト描画性能比較(Part 3)

機種による描画性能差

- 使用アプリ: 1TexSpr (基本的なスプライト描画負荷テストアプリ; テクスチャ1枚)
- テスト結果：

	Cocos2d-x v2 (バッチなし)	Cocos2d-x v2 (バッチあり)	Cocos2d-x v3 (自動バッチ)	Unity
iPhone 4	510個	520個	560個	980個
iPhone 4S	1960個	2000個	2050個	1800個
iPhone 5	4444個	5430個	5450個	3500個
Xperia arc	440個	580個	620個	560個
Nexus 5	1450個	10100個	5670個	3400個

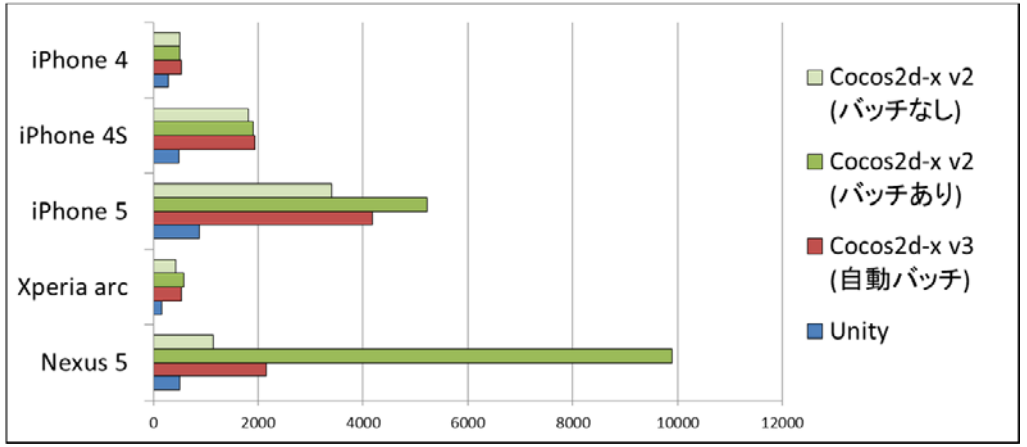


スプライト描画性能比較(Part 3)

機種による描画性能差

- 使用アプリ: 16TexSpr (基本的なスプライト描画負荷テストアプリ; テクスチャ16枚)
- テスト結果：

	Cocos2d-x v2 (バッチなし)	Cocos2d-x v2 (バッチあり)	Cocos2d-x v3 (自動バッチ)	Unity
iPhone 4	510個	505個	530個	280個
iPhone 4S	1820個	1900個	1930個	490個
iPhone 5	3400個	5220個	4180個	880個
Xperia arc	425個	580個	540個	165個
Nexus 5	1150個	9890個	2150個	500個

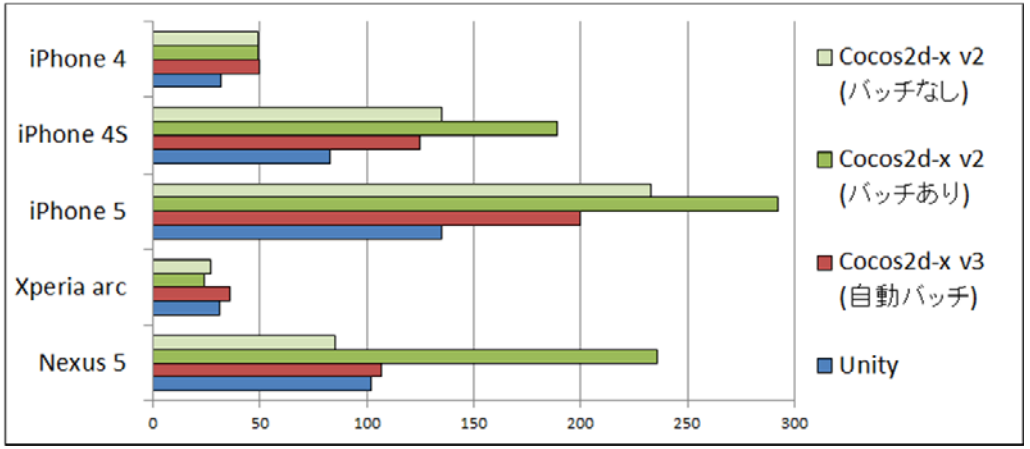


スプライト描画性能比較(Part 3)

機種による描画性能差

- 使用アプリ: Multi (ゲーム風アプリ; **テクスチャ1枚**)
- テスト結果：

	Cocos2d-x v2 (バッチなし)	Cocos2d-x v2 (バッチあり)	Cocos2d-x v3 (自動バッチ)	Unity
iPhone 4	49個	49個	50個	32個
iPhone 4S	135個	189個	125個	83個
iPhone 5	233個	292個	200個	135個
Xperia arc	27個	24個	36個	31個
Nexus 5	85個	236個	107個	102個

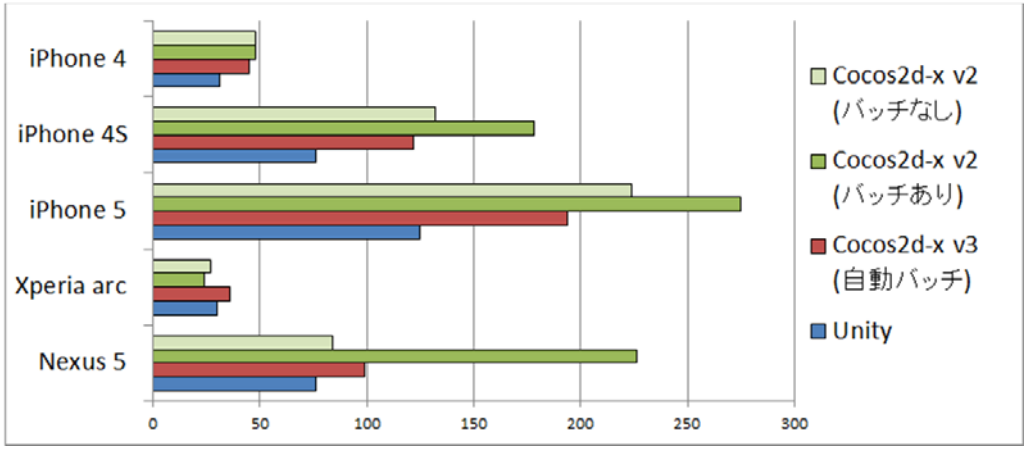


スプライト描画性能比較(Part 3)

機種による描画性能差

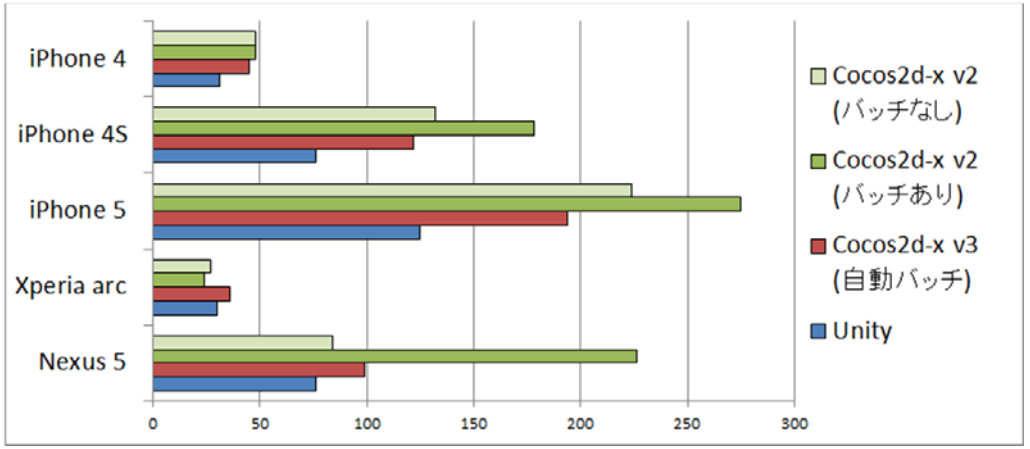
- 使用アプリ: Multi (ゲーム風アプリ; **テクスチャ2枚**)
- テスト結果：

	Cocos2d-x v2 (バッチなし)	Cocos2d-x v2 (バッチあり)	Cocos2d-x v3 (自動バッチ)	Unity
iPhone 4	48個	48個	45個	31個
iPhone 4S	132個	178個	122個	76個
iPhone 5	224個	275個	194個	125個
Xperia arc	27個	24個	36個	30個
Nexus 5	84個	226個	99個	76個



スプライト描画性能比較(Part 3)

- スマホ世代間の性能差は結構大きい
- Cocos2d-xに比べると、Unityはテクスチャの枚数がパフォーマンスにより影響を与える



スプライト描画性能比較

手戻りしないために

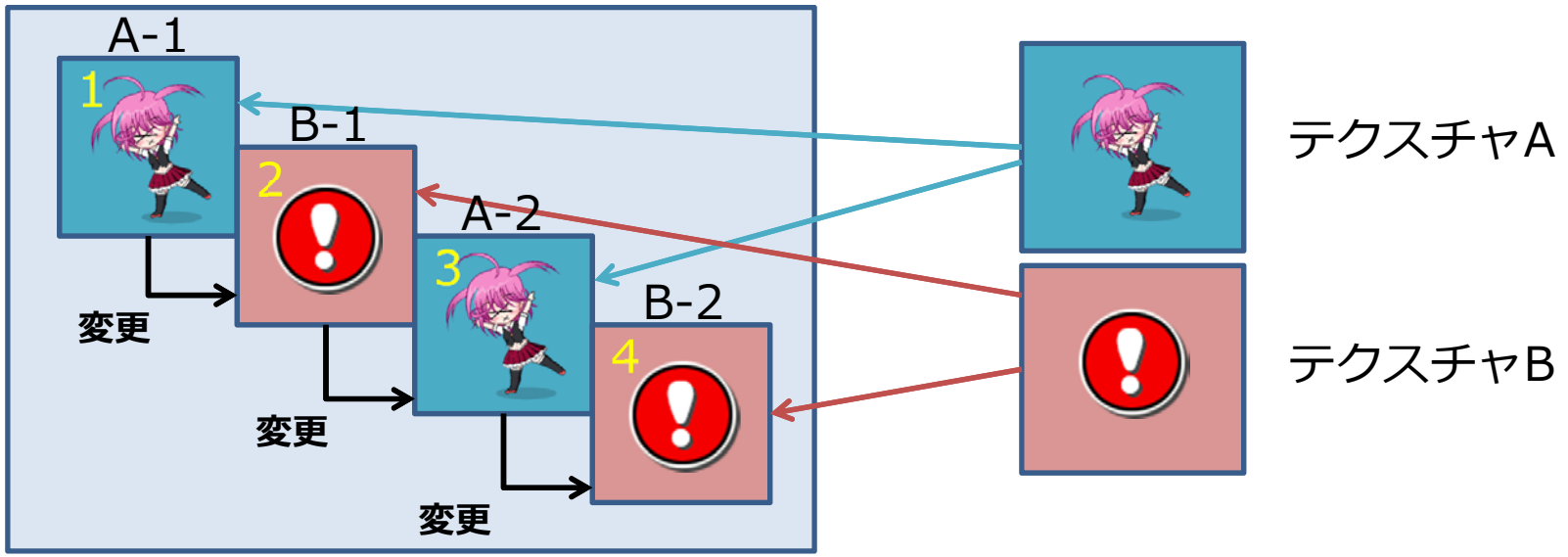
- スマホの世代による性能差はわりと大きい
- スマホの性能は、今後もしばらく向上が予想されるので、リリース時点の状況を予測して、どこまで古い機種をサポートするのかを良く検討した方が良い
- Cocos2d-xのほうがパフォーマンスが安定している。Unityは扱うテクスチャの枚数が増えるとパフォーマンスがより落ちる傾向にある、ただ本来3Dエンジンであることを加味すれば十分健闘している
- パフォーマンスが決定的に重要な場合はCocos2d-xを、そうでなければ開発のしやすさ優先でエンジンを選ぶのが良さそう
- iPhoneで先行して開発していると、Androidで性能が出ない場合がある。
  定期的に、ターゲット範囲の底辺の機種でパフォーマンスを確認しておく

ドローコールとは？

- GPU に対して描画開始を命令すること
- この呼び出し回数が多ければ、多いほど描画負荷が上がる
- バッチングを活用することで、ドローコールを減らすことができる

ドローコールが増える要因

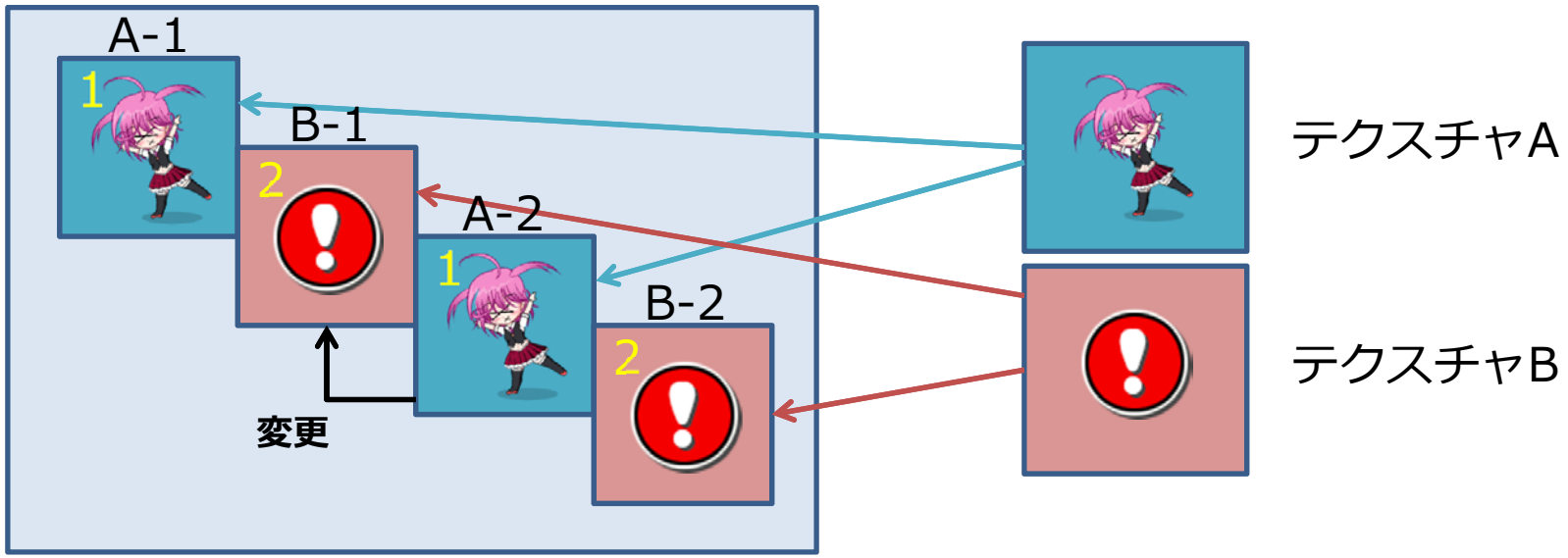
GPUステートの変更(テクスチャやシェーダの変更)によって増える



ドローコール４

バッチングとは？

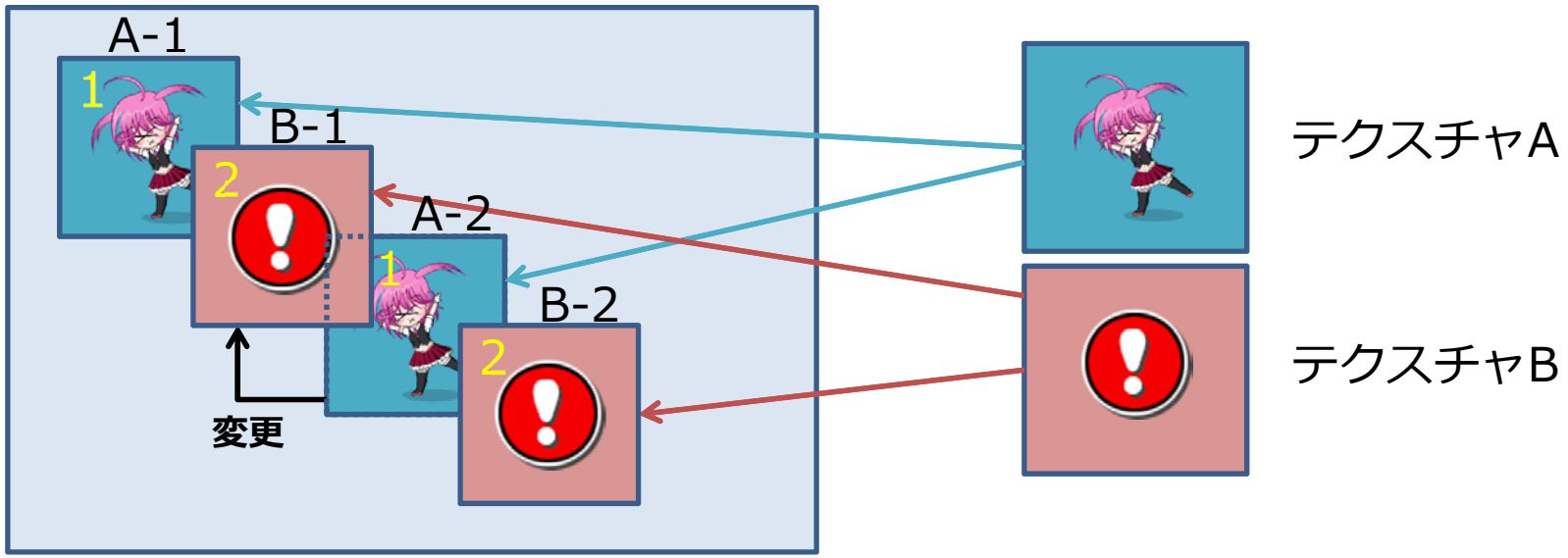
同じテクスチャを使ったものを一度にまとめて描画する



ドローコール2

バッチングと描画順序の関係

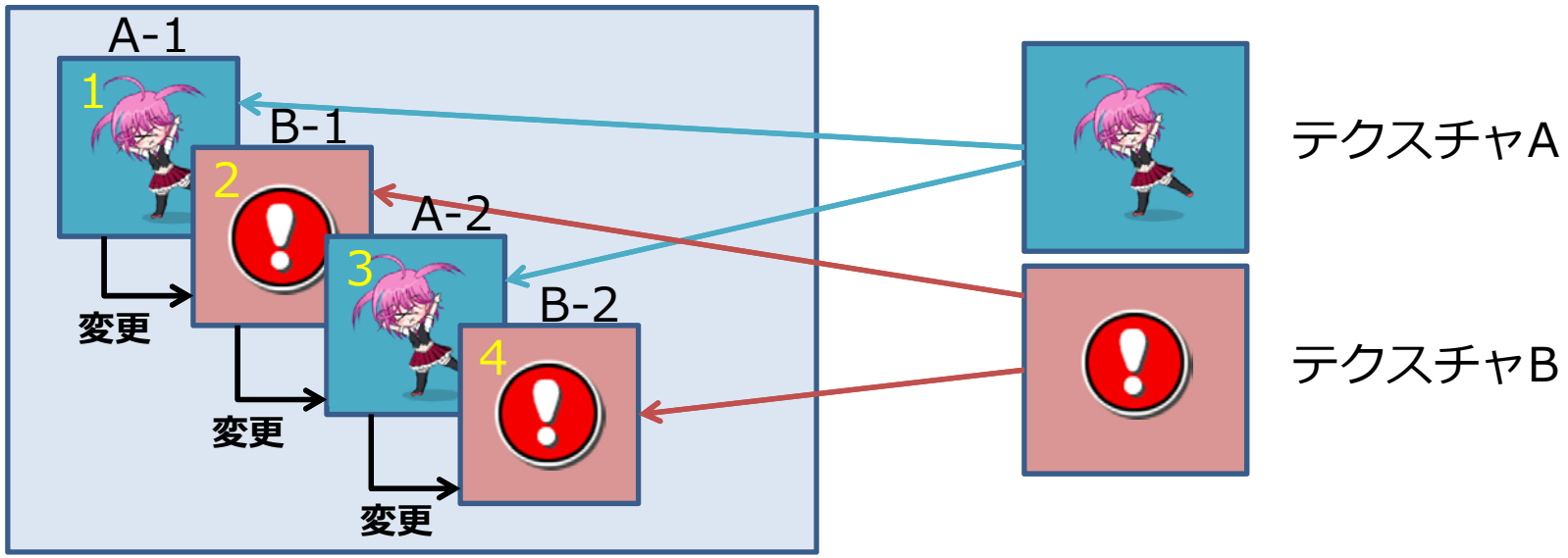
重なっている状態でバッチングすると描画順が狂う



ドローコール2

バッチングと描画順序の関係

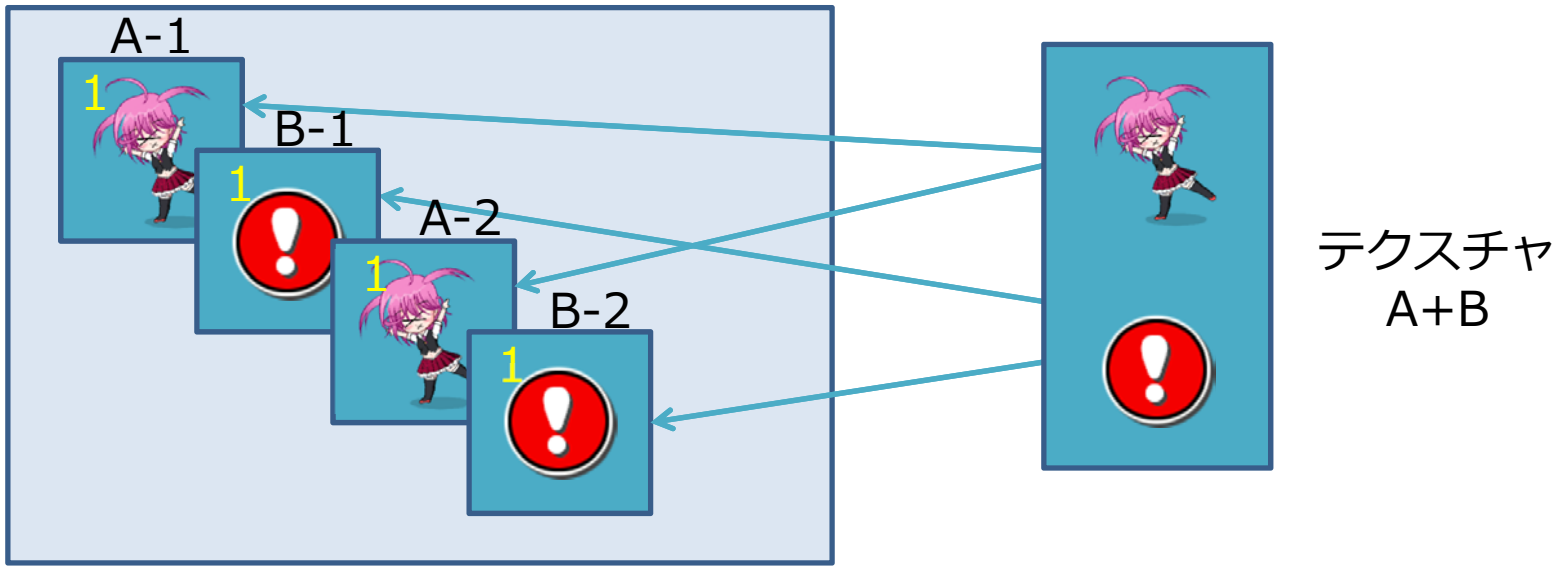
描画順を守るためバッチングできない



ドローコール４

テクスチャのアトラス化

バッチングできるようにテクスチャを１枚に合体する



ドローコール1

※ただし、現実的には必ずしも一枚のテクスチャに全ての画像を収められるとは限らない

バッチングのまとめ

- Unity, Cocos2d-x Ver.3系では自動バッチングに対応している
- Cocos2d-x Ver.2系では手動でバッチングする必要がある
- 自動、手動問わずバッチングを有効にする工夫が重要

- 参考情報
  http://japan.unity3d.com/unite/unite2014/files/DAY1-1700-room3-WebTechnologyAndKLab.pdf