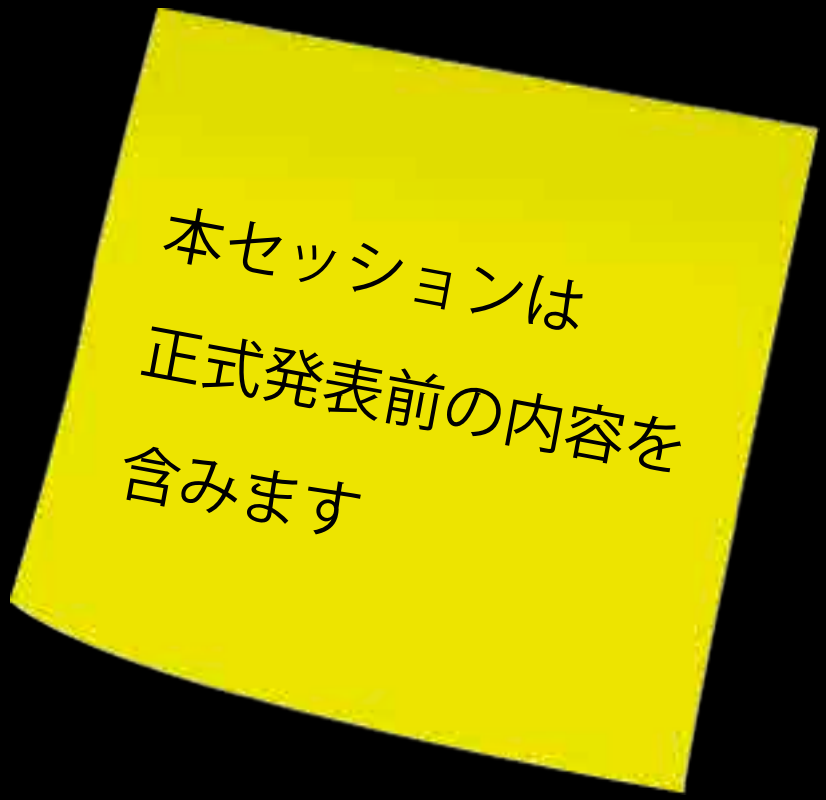
Flash Player 描画機能の最新情報
GPU 編

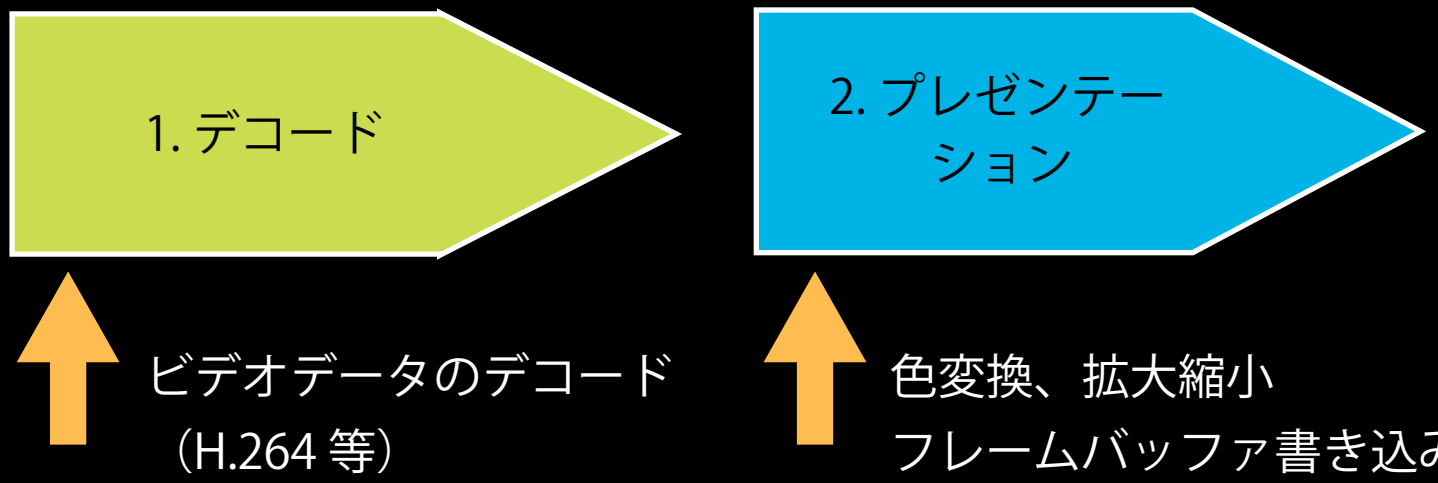
2010 年 11 月 25 日

上条晃宏

はじめに

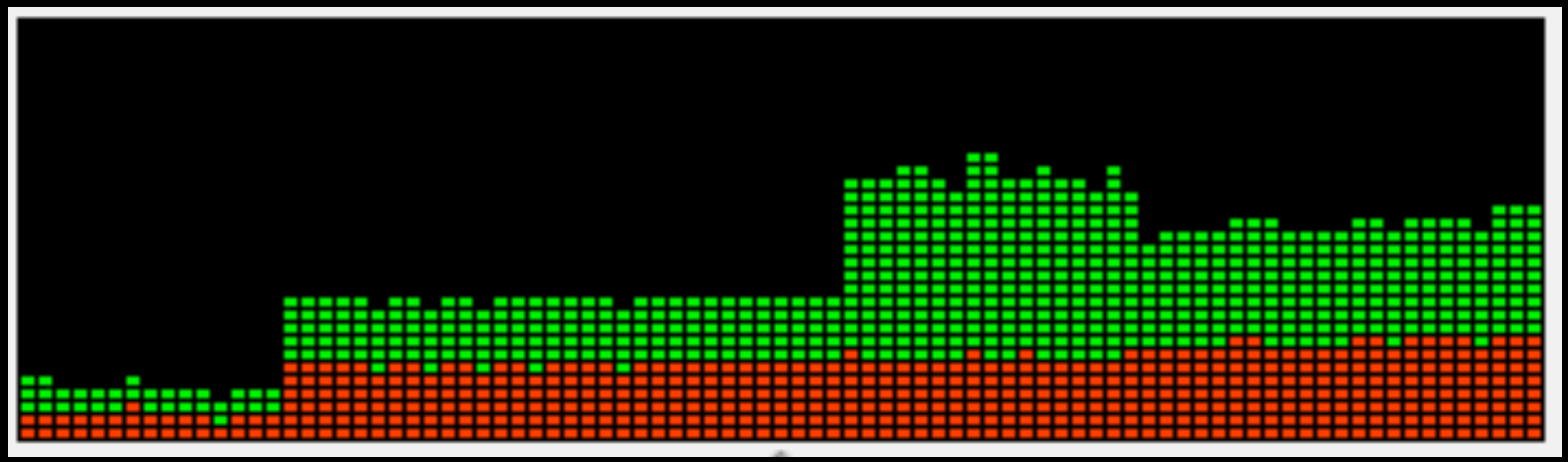


Stage Video



- StageVideo では、両方の処理が H/W で行われる
- 従来、プレゼンテーション処理は常に S/W で行われていた

SatgeVideo と従来の Video の比較 （CPU 使用率）



StageVideo

StageVideo +
オーバーレイ

Video +
オーバーレイ

Video のみ

- Video は wmode = direct で表示した
  - デコードは Stage Video、Video 共に H/W による処理
  - プレゼンテーション処理は Stage Video が H/W、Video が S/W

flash.media.StageVideo

- 基本的な使い方は、従来の Video クラスと同様
- H/W の仕様によって、複数の StageVideo インスタンスを利用できる
  - スクリプトからインスタンスを作成することはできない

```
var stageVideo:StageVideo;
if ( stage.stageVideos.length >= 1 ) {
  stageVideo = stage.stageVideos[0];
}
```

- 表示領域は矩形のみ
  - 大きさや位置の変更は可能

```
stageVideo.viewPort = new Rectangle(10,10,320,240);
```

flash.media.StageVideo つづき

- 状態の通知は StageVideoEvent で

```
stageVideo.addEventListene(StageVideoEvent.RENDER_STATE,
                           onStageVideoEvent);
```

- 表示オブジェクトと StageVideo の合成は H/W が行う
- AS3 からビデオデータを BitmapData に取り込むことは出来ない

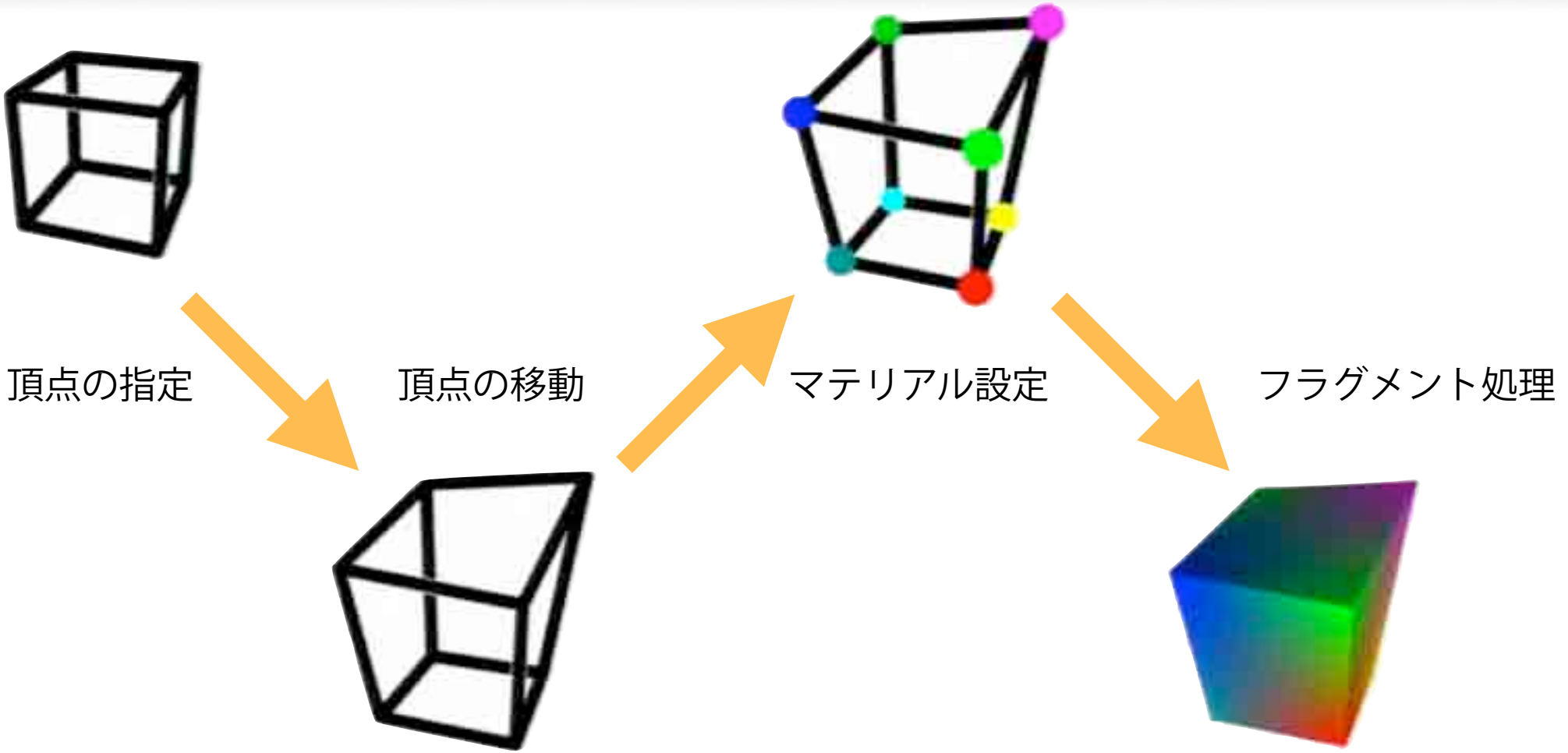
Stage Video 公開次期

- Flash Player 10.2 ベータ版から利用可能
  - http://labs.adobe.com/technologies/flashplayer10/stagevideo.html
  - 公開は来年前半を予定
- AIR for TV 用の ASDoc が公開されている
  - http://help.adobe.com/en_US/FlashPlatform/reference/actionscript/3/flash/media/StageVideo.html

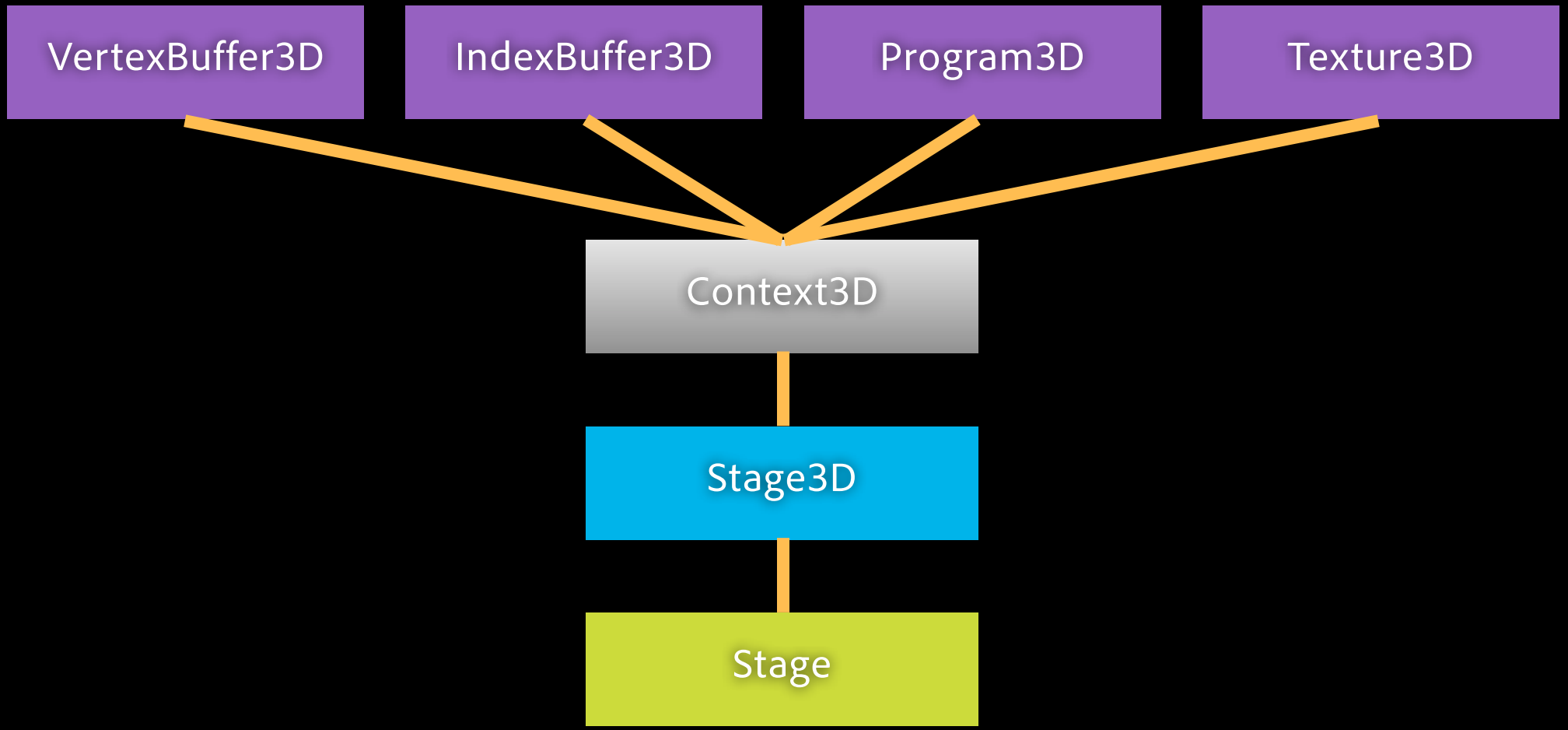
Molehill （Stage 3D）

- Flash Player の 3D API
- 従来の 2.5D とは異なり、完全に GPU により描画される
- DirectX 9, Open GL 1.3, Open GL ES 2.0 が前提
- GPU が利用できない環境では S/W にフォールバック
  - TransGaming SwiftShader を利用

Molehill 3D 処理プロセス



Molehill 主要クラス



＊仕様は Adobe MAX 2010 時点のもの

Mo;ehill コードサンプル

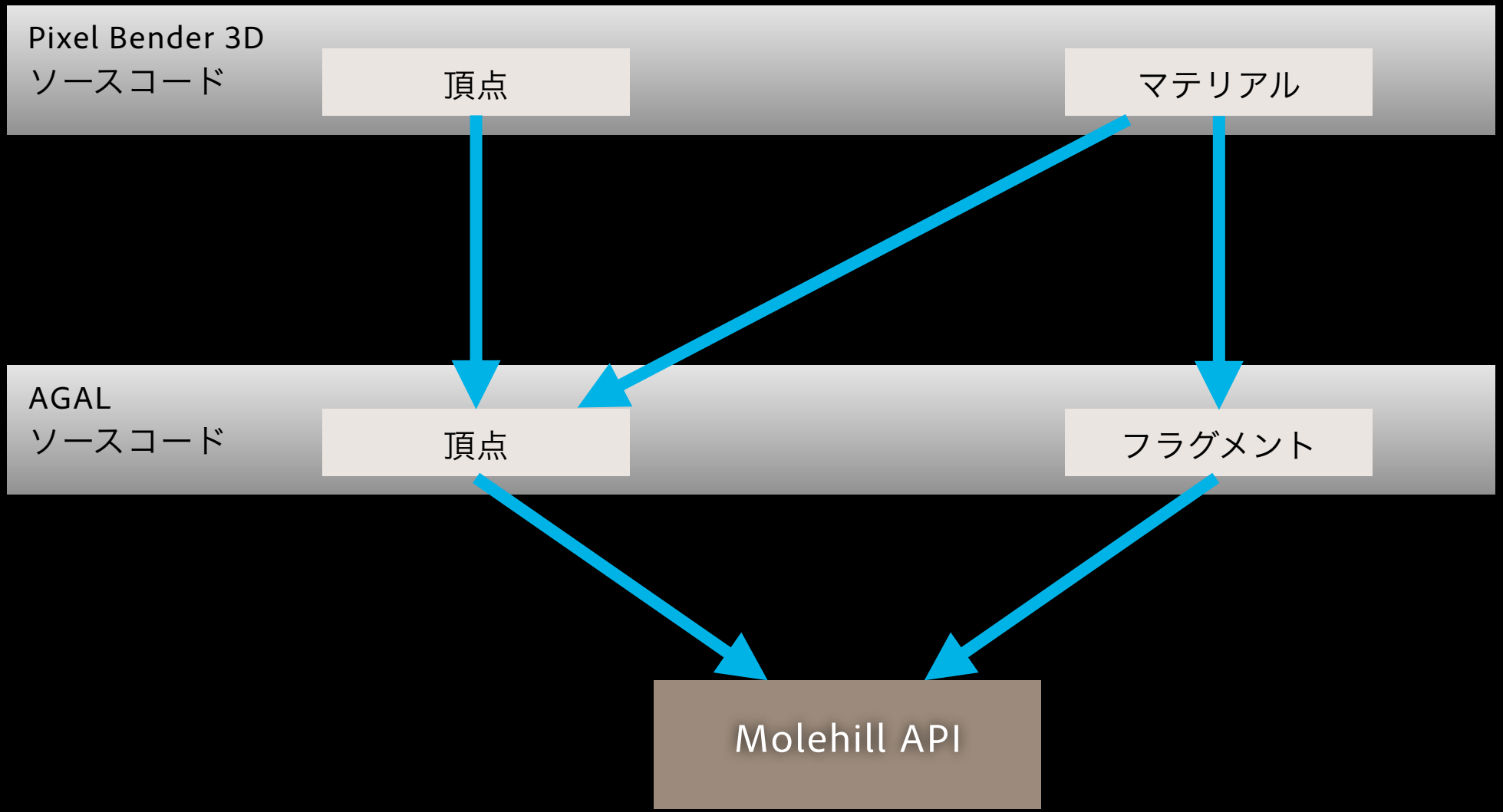
```
// 頂点プログラム
var av : AGALMiniAssembler = new AGALMiniAssembler();
as.assemble( Context3DProgramType.VERTEX,
    "m44 op, va0, vc0\n"
    "mov v0, va1\n"
);
// フラグメントプログラム
var af : AGALMiniAssembler = new AGALMiniAssembler();
af.assemble( Context3DProgramType.FRAGMENT,
    "mov oc, v0"
);
```

- 3D プログラム（赤字の箇所）は AGAL（Adobe Graphics Assembly Language）で記述する（＊仕様は Adobe MAX 2010 時点のもの）

Molehill に対応予定の 3D フレームワーク

- 以下のフレームワークが Molehill 用のコード生成機能を持つ予定
  - Proscenium （Adobe のフレームワーク）
  - Flare 3D
  - Alternativa 3D
  - Sophie 3D
  - Yogurt 3D
  - Away 3D
  - などなど

Pixel Bender 3D



Pixel Bender 3D サンプルコード

```
<languageVersion : 1.0;>
vertex kernel Bulge
<
    namespace : "AIF Test";
    version : 1;
>
{
    parameter float scale;
    parameter float4x4 objectToClipSpaceTransform;
    input vertex float3 vertexPosition
    <
        id : "PB3D_POSITION";
    >;
    output float4 vertexClipPosition;

    void evaluateVertex() {
        float3 t = vertexPosition;
        t += bulgeVector * scale;
        vertexClipPosition = float4( t.x, t.y, t.z, 1.0 )
        * objectToClipSpaceTransform;
    }
}
```

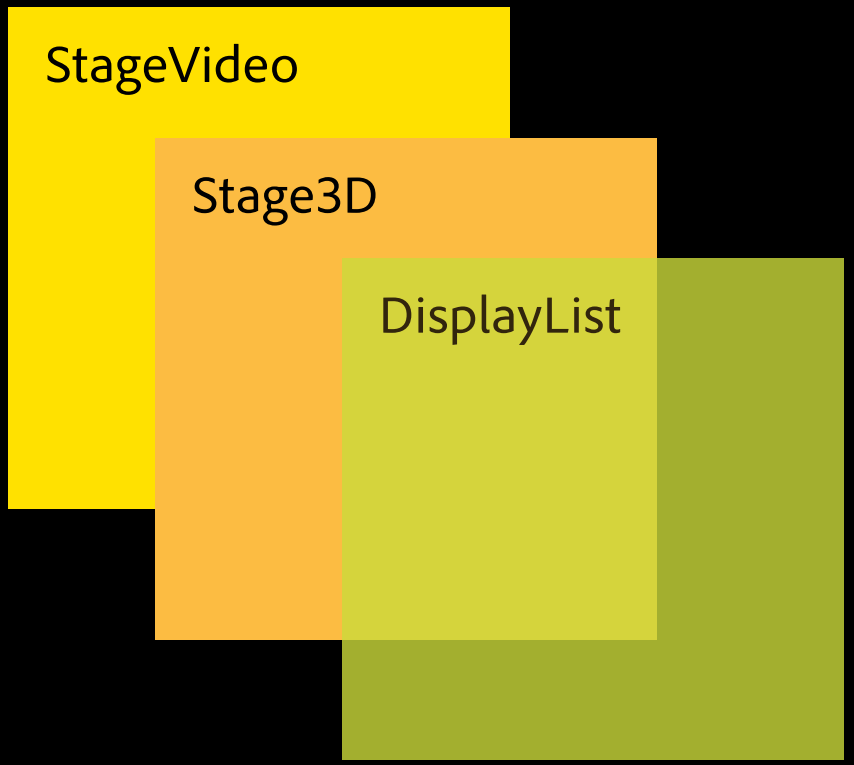
＊仕様は Adobe MAX 2010 時点のもの

Molehill 公開予定

- 2011 年前半にベータ版公開を予定
- Flash Player バージョンアップごと段階的に機能追加を行う予定

３つの描画プレーン

- それぞれH/W の独立した面に描画される
- 重なり順は、
  - 表示リスト → Stage3D → StageVideo
- 他の面上のデータにはアクセスできない
- 描画のタイミングも個別



2 種類の GPU 描画機能 （表示リストの描画）

描画モード	ラスタライズ	合成	Flash Player	AIR
GPU 合成	CPU	GPU	10 以降	2.0 以降
GPU ベクター	GPU	GPU	10.1 以降	2.5 以降

- GPU 合成の前提は OpenGL ES 1.1
- GPU ベクターの前提は OpenGL ES 2.0
- 現時点ではデバイスのみ上記機能を利用可能
  - 明示的に wmode の指定がない場合、Flash Player が GPU の利用可否を判断
  - AIR では GPU の利用を明示的に指定する

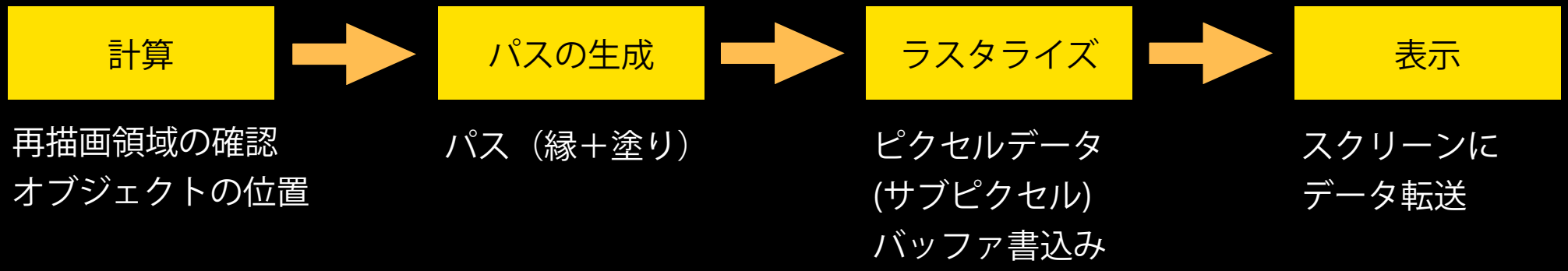
AIR RenderMode（mobileDevice プロファイルのみ）

- アプリケーション記述ファイル内の renderMode タグに設定する

```
<initialWindow>
    <renderMode>gpu</renderMode>
    <content>Foo.swf</content>
    ...
</initialWindow>
```

- Flash Professional CS5 の設定パネルから指定できる

Flash Player 描画プロセス（CPU）



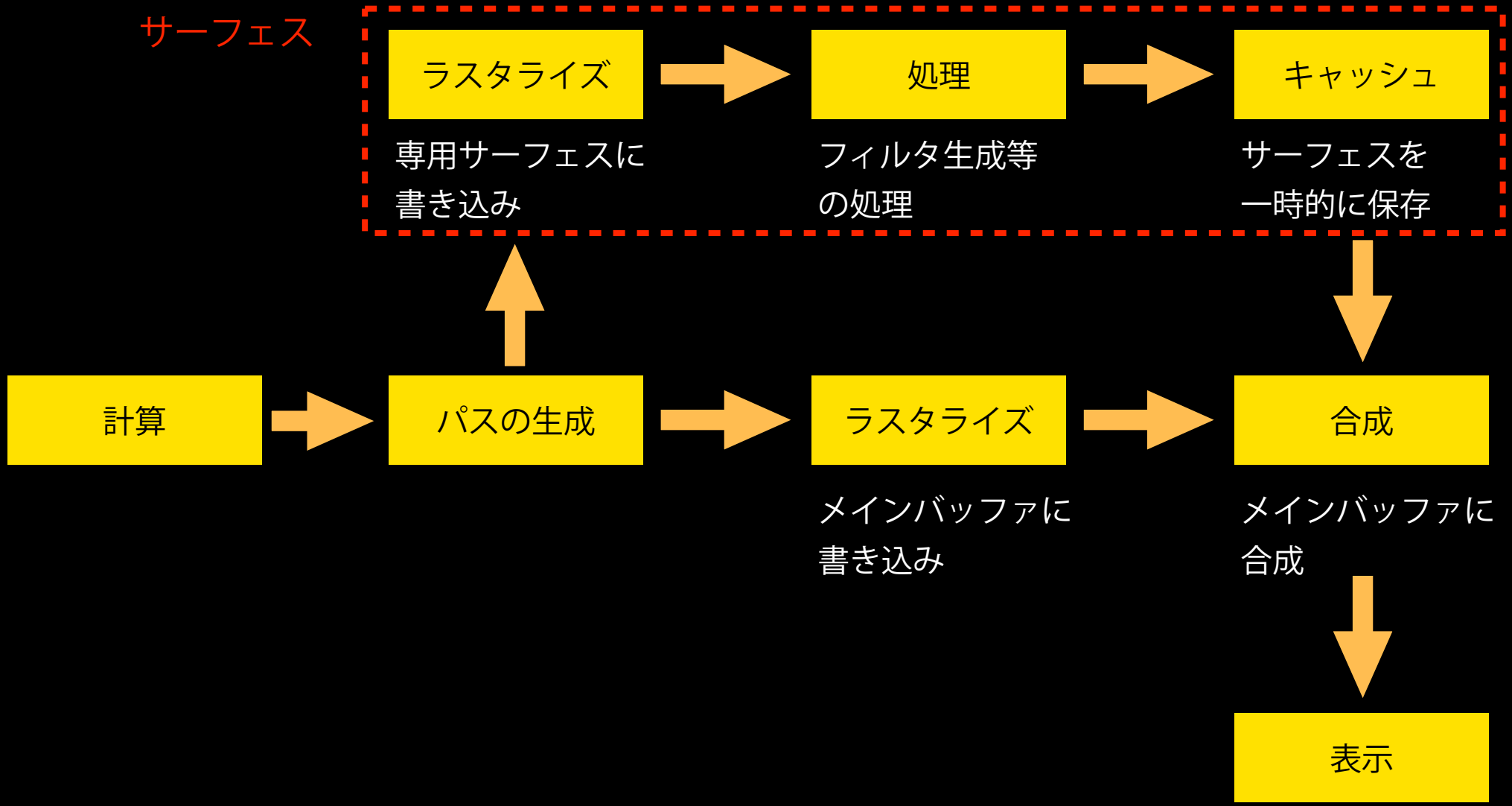
- 処理は１ピクセルあたり１度だけ

注意点：
再描画領域の大きさ
階層の深さ
エッジの数
アルファ
パスの重なり
ピクセルアラインメント

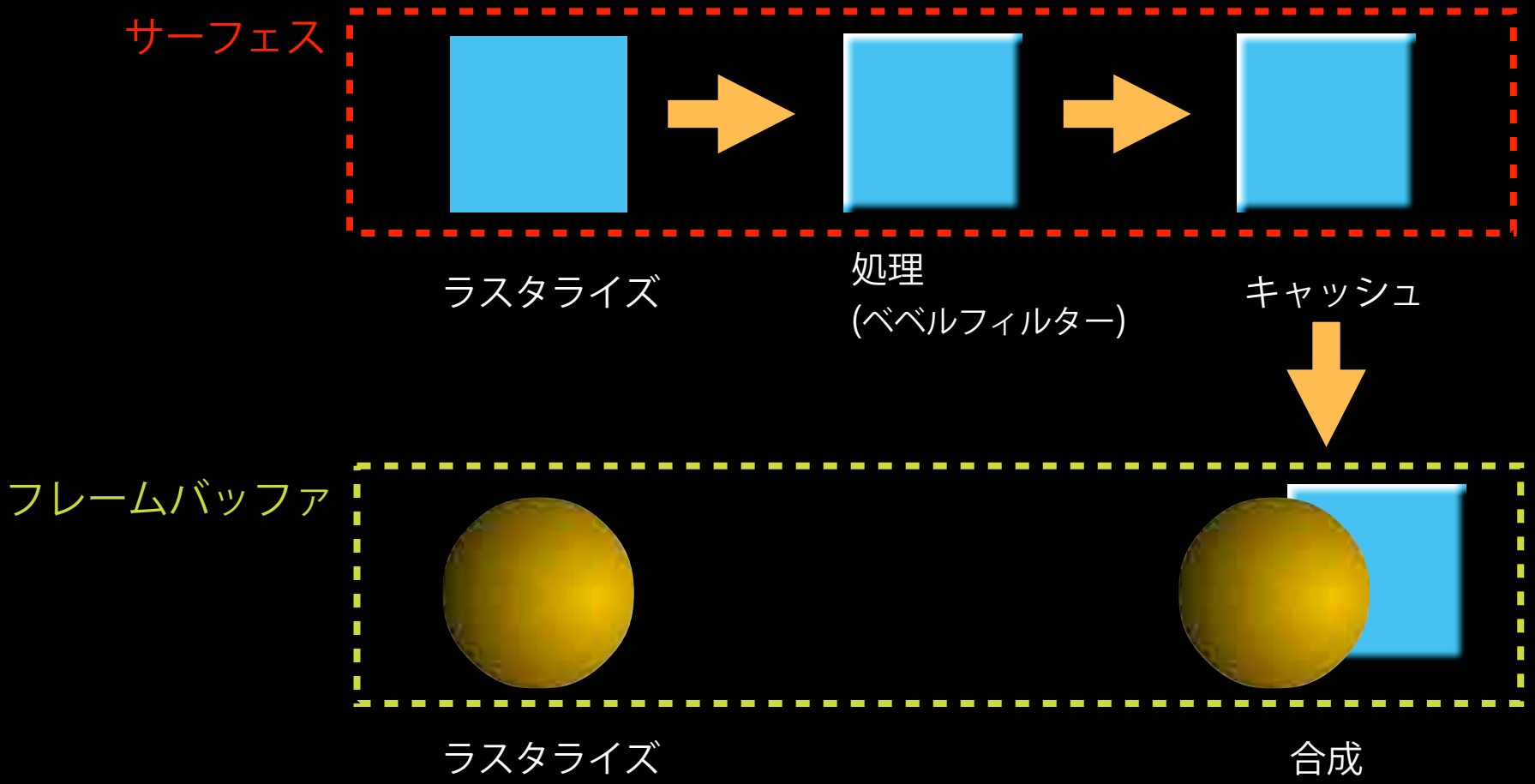
サーフェス描画

- Flash Player 8 から追加された
- 独立したサーフェスに描画し、後でメインのピクセルバッファに合成
- 以下の機能が利用する
  - cacheAsBitmap
  - 2.5D
  - フィルター
  - opaqueBackground 属性設定時の scrollRect
  - （一部の）デバイステキスト

サーフェス描画のフロー

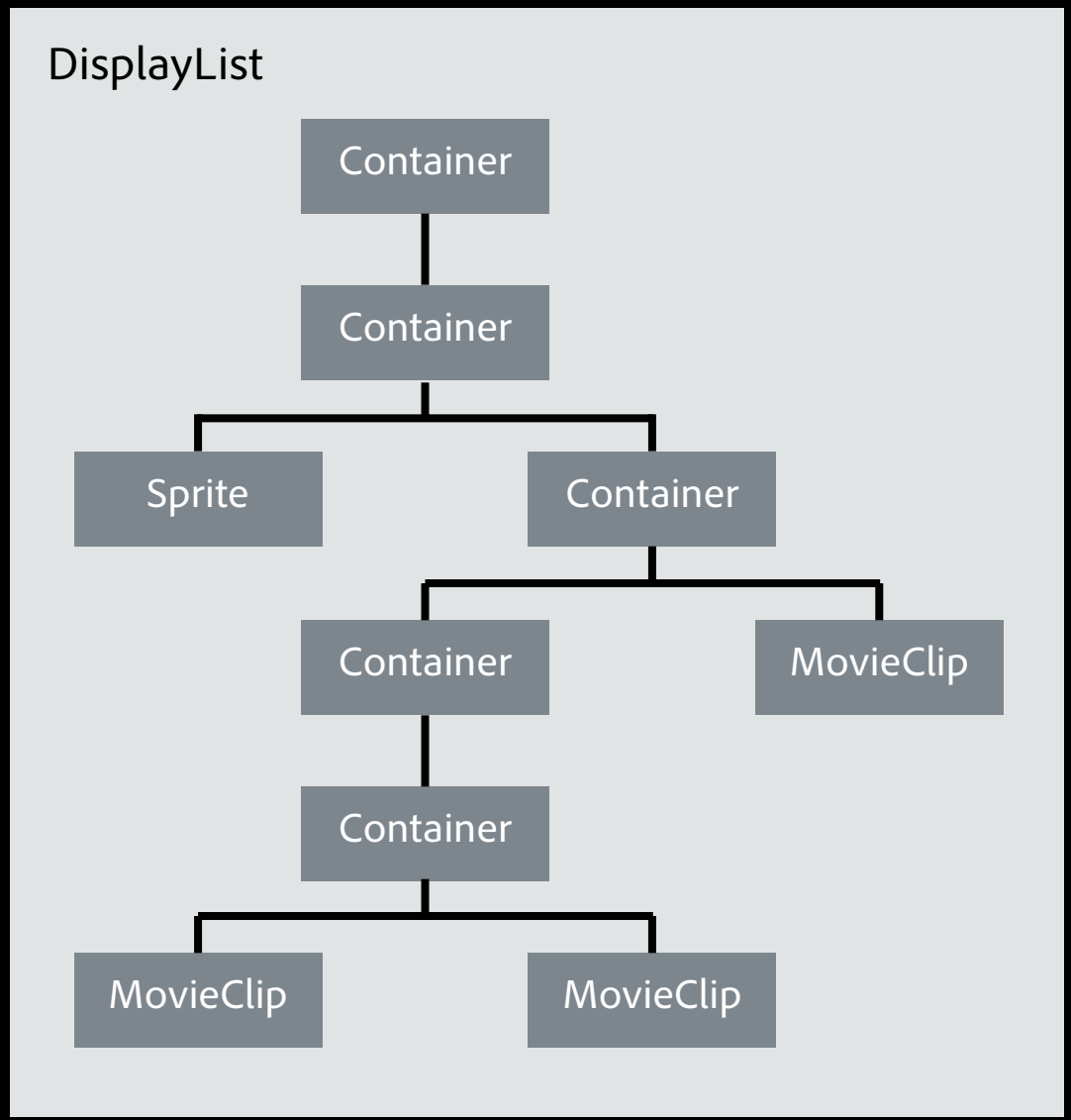


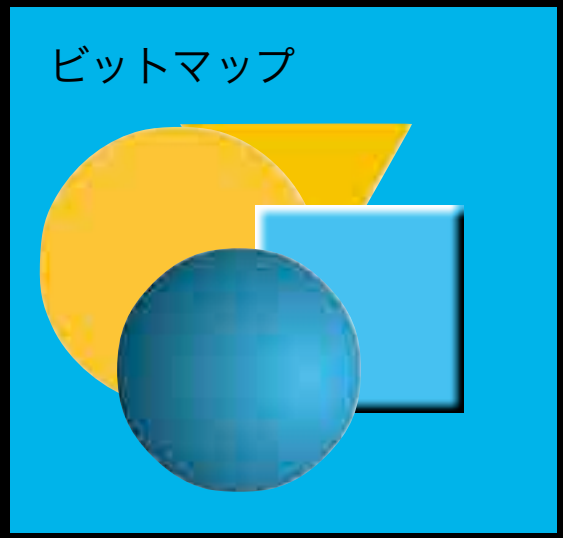
サーフェス描画の例（フィルタの場合）



CacheAsBitmap

- 表示オブジェクトをビットマップ形式に変換して、サーフェスにキャッシュ

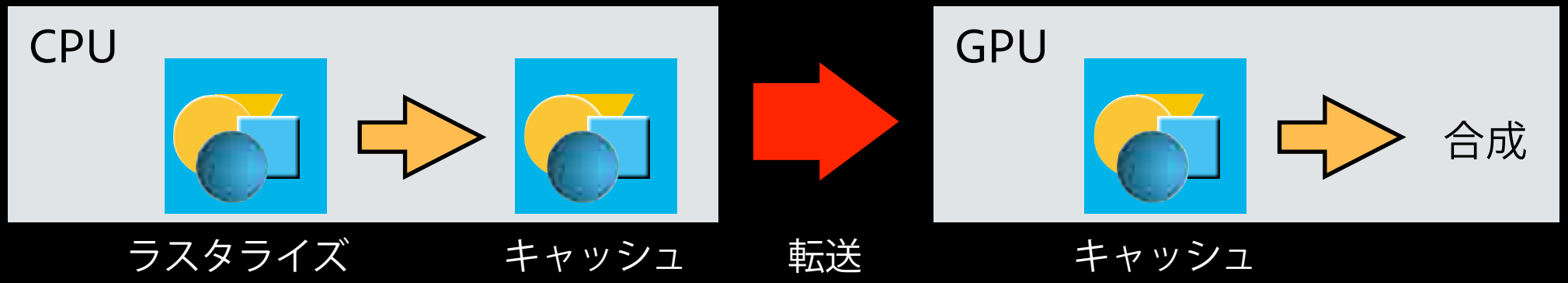




DisplayObject.cacheAsBitmap

- 値が true の場合、表示オブジェクトをビットマップ形式でキャッシュ
- キャッシュされた描画データは複数フレーム間で使用できる
- 表示オブジェクトの位置が変わっても、キャッシュが再利用される
  - フレームごとにラスタライズ処理が発生しない
- 大きさや傾きが変わると、キャッシュは破棄される
- メモリ使用量が増える（縦 × 横 × 4 bytes）
- フィルター使用時は、値が自動的に true になる

CacheAsBitmap と “GPU 合成”



- サーフェスにキャシュされたビットマップを GPU 側に転送しキャッシュ
- 合成は GPU 側で行う
- キャッシュが再利用されるとその描画領域のデータ転送が発生しない
  - GPU へのデータ転送量削減の効果がプラスされる
- キャッシュされたオブジェクトがステージ上動き回るケースに有効
- 頻繁にキャッシュが更新される場合の負荷は高い
- GPU メモリは貴重

CacheAsBitmapMatrix

- cacheAsBitmapMatrix を使うと、拡大や回転時でもサーフェスが再利用される

機能	移動	拡大	回転	visible = false
cacheAsBitmap	○			
cacheAsBitmap + cacheAsBitmapMatrix	○	○	○	○
2.5D	○	○	○	

CacheAsBitmapMatrix つづき

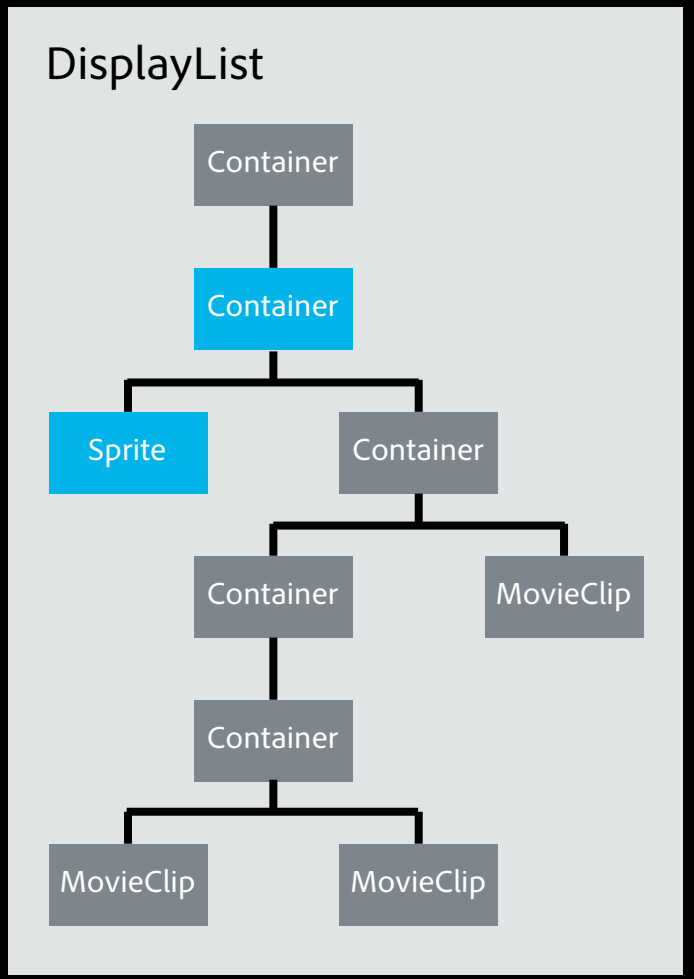
- 表示オブジェクトの 2D 属性を操作
- cacheAsBitmap と組み合わせて使用する

```
matrix:Matrix = new Matrix();
displayObject.cacheAsBitmap = true;
displayObject.cacheAsBitmapMatrix = matrix;
```

- AIR の mobileDevice プロファイルのみで利用可能 （現時点）
- 2.5D 属性を設定すると機能しない
- マトリクスを操作するとサーフェスが再度ラスタライズされる

CacheAsBitmap と表示リスト

- 子の表示オブジェクトが親から見て変化する場合
  - 親：true、子：false　→　×
  - 親：false、子：true　→　○
- cacheAsBitmap = true に設定すると、描画領域に変更があるたびに、キャッシュが再作成される
- コンテナへの cacheAsBitmap の使用は注意
- 階層を深くしない！
  - 例：Sprite.graphics.draw()



チップセット

製造	モデル	クロック	GPU	プロセス	スマートフォン
Qualcomm Snapdragon	QSD8250 QSD8260	1 GHz	Adreno 200	65 nm	HTC Desire, Sharp SH-03C, IS03, Fujitsu Toshiba T-01C, IS04, SIRIUS IS06
Qualcomm Snapdragon	MSM7630 MSM8255	1 GHz	Adreno 205	45 nm	HTC Desire HD, Sharp IS05, 003SH, 005SH
Qualcomm Snapdragon	MSM8260 MSM8660	1.2 GHz Dual	Adreno 220	45 nm	2010 Q3 出荷開始
Samsung Hummingbird	S5PC110 S5PV210	1 GHz	PowerVR SGX 540	45 nm	Galaxy S, (iPhone 4G)

- GPU 性能は機種により数倍の差’（次世代のチップでは更に差が拡大）

解像度とピクセル密度

デバイス	解像度	サイズ	PPI
Nexus One / HTC Desire	800x480	3.7’	254
Galaxy S	800x480	4.0’	232
HTC Desire HD	854x480	4.3’	217
Droid X	854x480	4.3’	240
iPhone 3GS	480x320	3.5’	163
iPhone 4	960x640	3.5’	326
iPad	1024x768	9.7’	132
Galaxy Tab	1024x600	7.0’	170

- CPU 描画の場合、解像度による影響が大きい

ありがとうございました！！

- つづきはブログで： http://cuaoar.jp