

コンピュータグラフィックス

第12回 リアルタイムCG

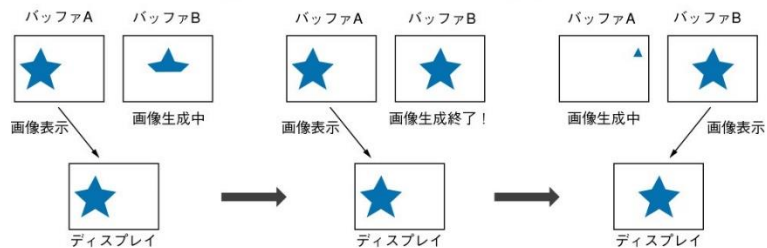
理工学部 兼任講師
藤堂 英樹

本日の講義内容

■リアルタイムCG

- リアルタイムアニメーションの手法
- リアルタイムのCGシステム

■図5.41——ダブルバッファ方式



「コンピュータグラフィックス」2004年 / 財団法人画像情報教育振興協会 (CG-ARTS協会)

■図5.44——シェーダを用いたリアルタイムシェーディングの例 (NVIDIA社のデモコンテンツ)

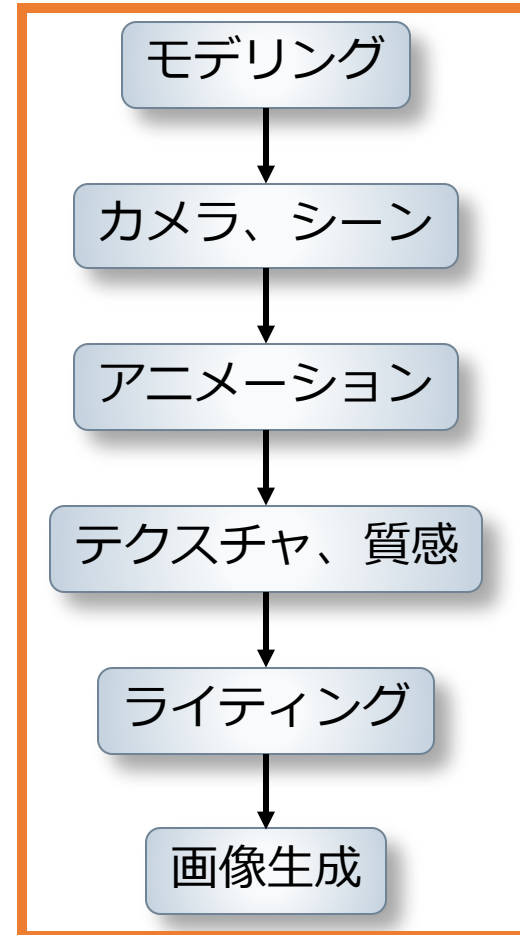


(©NVIDIA Corporation)

「コンピュータグラフィックス」2004年 / 財団法人画像情報教育振興協会 (CG-ARTS協会)

CG制作の主なワークフロー

■3DCGソフトウェアの場合



リアルタイムCG

■CGをリアルタイムにする必要性

- インタラクティブなユーザーとのやり取り
- 映像制作
 - モデリング,...,ライティングの編集集中の表示
- ゲーム
 - ユーザーがキャラクターを操作



■なるべくクオリティが高い物を高速に表示したい

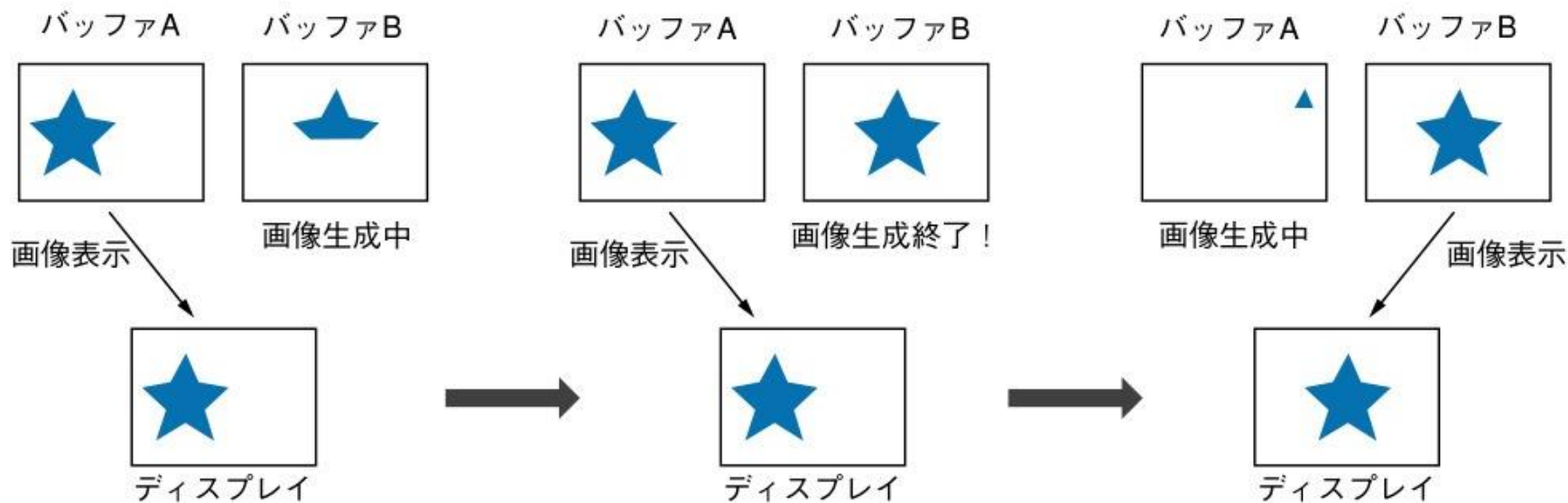
ダブルバッファ方式

■画像の生成には時間がかかる

⇒ 1枚の画面をクリアして描画すると**ちらつく**

■ダブルバッファ方式

- **2枚のバッファ**を切り替えてちらつきを回避



「コンピュータグラフィックス」2004年 / 財団法人画像情報教育振興協会 (CG-ARTS協会)

ダブルバッファの例 © JavaDrive

<http://www.javadrive.jp/applet/thread/ThreadTest6.html>

1枚の画面だけ
(ちらつく)

<http://www.javadrive.jp/applet/thread/ThreadTest7.html>

直線がたくさん描かれた
正方形を右に動かす

ダブルバッファ
(ちらつかない)

ダブルバッファはリアルタイムCGの基本

■OSのUI

- 一番身近なインタラクティブCG
- ダブルバッファ機能は**標準で搭載**されている
- 最近のOSのUIは**3D API**を使用している

■3D API

- OpenGL, DirectXにはダブルバッファ機能が**標準搭載**

■3DCGソフトウェアにおいても**標準搭載**

リアルタイムシェーダー

■ GPUを利用した高速な描画処理

- 3次元CG用に特化された演算装置
- 専用のビデオメモリ

■ NVIDIA Shader Library

- NVIDIA社が開発したGPUで動作する
デモコンテンツ

nVIDIA QUADRO GPU
© NVIDIA Corporation

http://developer.download.nvidia.com/shaderlibrary/webpages/shader_library.html

NVIDIA Shader Library

■ デモコンテンツの概要

- タイトル
- スナップショット
- 動作するGPU
- 3DCGのAPI

■ 各種ダウンロード

- シェーダーのサンプルコード
- 技術内容の簡単な説明
- デモビデオ

GPU処理を行うプログラム言語

■ HLSL:

- **Microsoft**社の**DirectX**上で動作するGPU言語
- **ゲーム**で使われることが多い

■ Cg

- **OpenGL**と**DirectX**の両方に対応したGPU言語
- **Autodesk Maya**や**Unity**でも利用されている

■ GLSL

- **OpenGL専用**のGPU言語
- 研究の現場で用いられることが多い

リアルタイムシェーダー開発ツール

■ FX Composer © NVIDIA

- HLSL, Cgでの開発が可能
- **UIの自動生成**
- シーン, テクスチャの表示
- NVIDIA Shader Libraryとの連携

<https://www.youtube.com/watch?v=OnRUpvZuzx0>

■ RenderMonkey © AMD

- HLSL, GLSLでの開発が可能
- **UIの自動生成**
- シーン, テクスチャの表示

<http://developer.amd.com/tools-and-sdks/archive/legacy-cpu-gpu-tools/rendermonkey-toolsuite/>

Unity上でのシェーダー開発

■ 開発言語

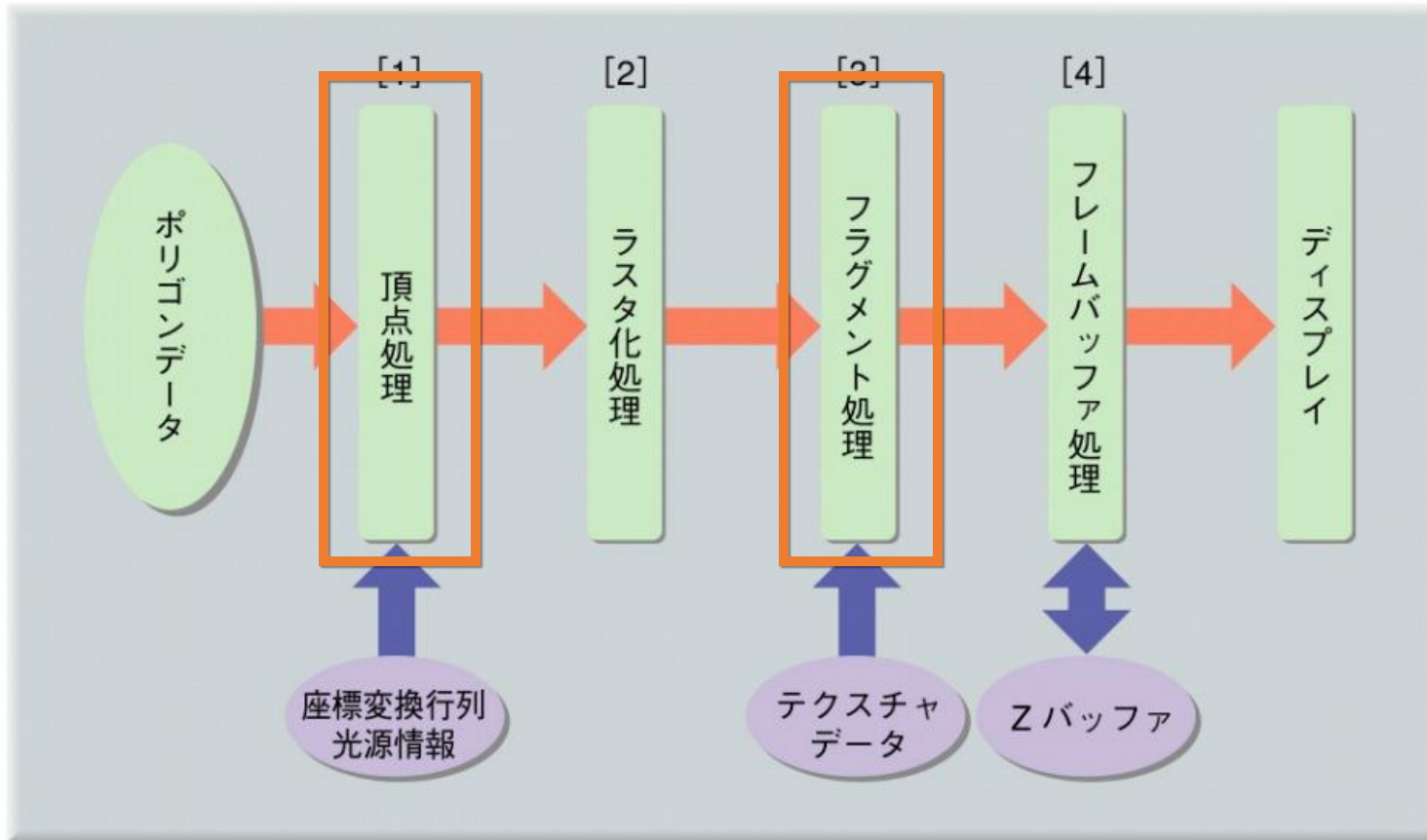
- ShaderLab: Unity独自の開発言語
- HLSL / Cg: GPU処理部分

■ シェーダーの種類

- 固定機能シェーダー
 - ShaderLabで記述
- サーフェスシェーダー
 - ShaderLabで大枠を記述し, HLSL / Cgを補助的に使用
- 頂点シェーダー, ピクセルシェーダー
 - 通常のHLSL / Cgの使い方にかかなり近い
 - ShaderLabをUnityとのやり取りに利用

3次元ハードウェア上での処理

■図8.10——3次元CGハードウェアによるCG描画処理の流れ



「コンピュータグラフィックス」2004年 / 財団法人画像情報教育振興協会 (CG-ARTS協会)

頂点・ピクセルシェーダー

■頂点シェーダー

- **頂点毎の処理**を記述
- 主な処理: **頂点の座標変換, 各種頂点データの計算**
- 計算した**頂点データ**はピクセルに補間され,
ピクセルシェーダーに**転送**される

■ピクセルシェーダー

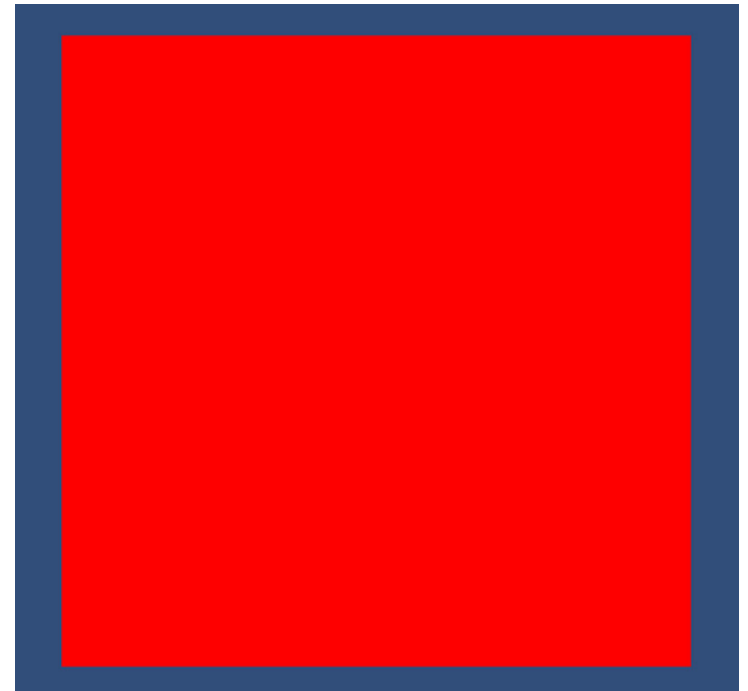
- **画素毎の処理**を記述
- 主な処理: **ライティング, テクスチャマッピング**
- 頂点から送られてきたデータの利用し,
最終的な画素の色を計算する

一番シンプルな例

■Unityの公式マニュアル

- [Vertex and Fragment Shader Examples](#)

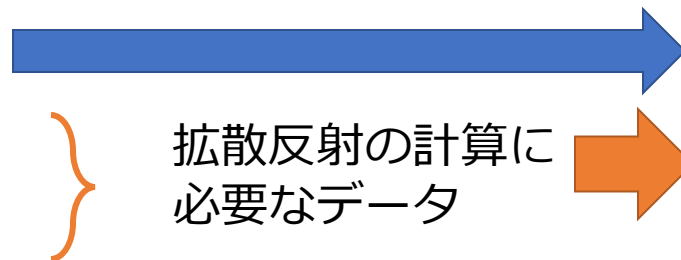
```
1 Shader "Custom/SolidColor" {
2     SubShader {
3         Pass {
4             CGPROGRAM
5
6             #pragma vertex vert
7             #pragma fragment frag
8
9             float4 vert(float4 v:POSITION) : SV_POSITION {
10                return mul (UNITY_MATRIX_MVP, v);
11            }
12
13            fixed4 frag() : COLOR {
14                return fixed4(1.0,0.0,0.0,1.0);
15            }
16
17            ENDCG
18        }
19    }
20 }
```



拡散反射のシェーディング

■ 頂点シェーダー

- 色を塗る位置
- 法線ベクトル
- 光源方向



ピクセルシェーダー
に転送

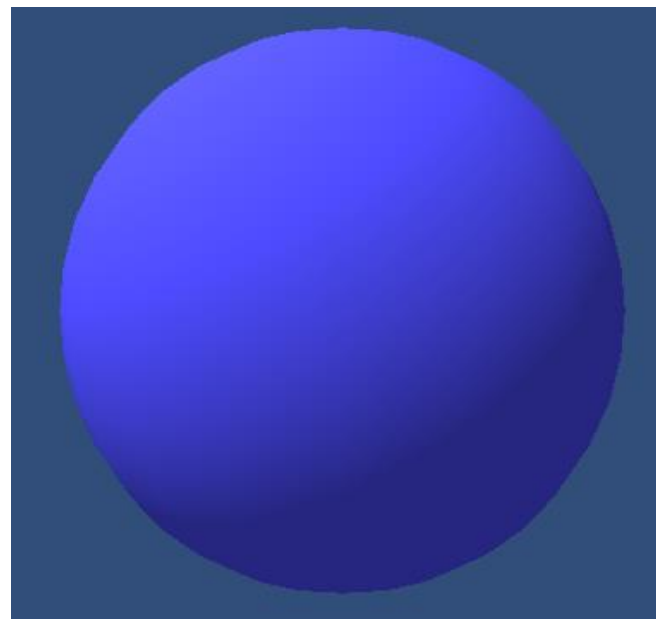
```
43 // 頂点毎の処理
44 // pos, L, N, RVのデータを計算する。
45 vertexOutput vert(appdata_base v) : POSITION
46 {
47     vertexOutput output;
48     // 座標変換後の位置
49     output.pos = mul (UNITY_MATRIX_MVP, v.vertex);
50
51     // 法線ベクトル
52     float3 N = v.normal;
53     output.N = N;
54
55     // ライトベクトル
56     output.L = ObjSpaceLightDir (v.vertex);
57
58     return output;
59 }
```


拡散反射のシェーディング

■ピクセルシェーダー

- 法線ベクトル・光源方向で陰影計算を行う

```
61 // ピクセル毎の処理
62 // LambertShadingのライティング計算を行う
63 float4 frag(vertexOutput input) : COLOR
64 {
65     // 頂点処理で計算したベクトルデータを正規化して取り出す
66     float3 L = normalize( input.L );
67     float3 N = normalize( input.N );
68
69     // 環境光成分I_aを計算
70     float4 I_a = _Ka * _Ambient;
71
72     // 拡散反射成分I_dを計算
73     float LdN = clamp( dot(L, N), 0, 1 );
74     float4 I_d = _Kd * LdN * _Diffuse;
75
76     // 足し合わせて最終的な色を計算する
77     float4 I = I_a + I_d;
78
79     return I;
80 }
```



より複雑なGPU処理

■GPU Gems ©Randima Fernando

- 様々なリアルタイムCG手法を紹介
- [デモ解説ページ](#)

https://developer.nvidia.com/gpugems/GPUGems/gpugems_ch04.html

アニメーションの
GPUによる高速化

https://developer.nvidia.com/gpugems/GPUGems/gpugems_ch16.html

サブサーフェス
スキャタリング

リアルタイムCGシステム

■ゲームエンジン

- ゲーム作成に有用なデータ
- インタラクティブなCG処理

■WebGL

- Web上で動作するCGシステム

■MikuMikuDance

- 初音ミクのダンスCGに特化

レポート課題3

- 2. UnityやWebGL等で作成されたリアルタイムに動作するCGコンテンツを1つ取り上げ、解説・感想を述べなさい
 - 選択するCGコンテンツ：自由に選んでよい
 - 解説・感想：500字以上、図を1つ以上使って説明すること

UnityのリアルタイムCG

■インタラクティブなCG処理

- デザイン時にユーザーにシーン情報を提示
- ゲームプレイ時にダイナミックなシーンを演出
- **Web Palyer**で体感が可能

■Unity公式のデモページ

- <http://japan.unity3d.com/gallery/demos/>

Web上のデモで利用されるUnity

<http://www.esimple.com.tr/en/item/leonardo-da-vinci-museum>

Leonardo da Vinciの
バーチャルミュージアム
© Esimple

<http://www.ddd.co.jp/downloads/unity/>

車のリアルタイムCG
© 株式会社スリーディー

WebGL

■ Web上で3DCGを表示

- OpenGL, **GLSL**を利用した描画処理
- **主要なブラウザ**はほぼ対応

<https://developer.mozilla.org/ja/demos/detail/webglmotion-trackingpure-data/launch>

<http://helloracer.com/webgl/>

カメラでリアルタイムに
モーションをトラッキング

© mathajie

レーシングカーの
リアルタイムCG

© HelloEnjoy™

WebGL

http://alteredqualia.com/three/examples/webgl_terrain_dynamic.html

山岳地形のリアルタイムCG

© mathajie

http://callum.com/apps/storm_tracks_webgl/

山岳地形のリアルタイムCG

© Callum Prentice

WebGL

<http://madebyevan.com/webgl-water/>

水のリアルタイム
シミュレーション

© Evan Wallace

http://alteredqualia.com/three/examples/webgl_materials_skin.html

リアルな皮膚の質感の
レンダリング

© AlteredQualia

Miku Miku Dance (MMD) ©樋口優

- 初音ミクのダンスCGに特化したシステム
 - モーションデザイン・再生
 - 音声・動画ファイルとの連携
 - シンプルな描画処理でインタラクティブにデザイン・再生が可能

<http://www.nicovideo.jp/watch/sm2420025>

<https://www.youtube.com/watch?v=zUv4iVFfiBc>

ニコニコ動画の説明動画

© LaRenuille

MMDを使用して作られたPV

© LaRenuille

MMDのリアルタイムCG

<http://blog.misanyan.com/mmd.js-master/index.html>

<http://3d.nicovideo.jp/works/td16409>

WebGLに移植された
MMDビューアー
© edvakf

ニコニ立体
MMD+Unityによるモデル共有システム
時雨© ぼんぷ長

Live2DシステムとUnityの連携

- 前回の授業で紹介した**Live2DのUnity**連携
 - Live2Dの立体表現で**インタラクティブなCG処理**
 - ゲーム作成への応用が可能

<https://www.youtube.com/watch?v=RqtqFEslyW0>

Live2D
2Dを活かした立体表現

Live2DシステムとUnityの連携
© Live2D

レポート課題3

- 3. SIGGRAPH 2014の技術論文から 1 つ論文を選び、考察・感想を述べなさい。
 - 選択する論文：自由に選んでよい
 - 考察・感想：500字以上，図を1つ以上使って説明すること

■ 課題解決のヒント

- Ke-Sen Huangによる論文リスト
- デモ動画：手法の概要，新規性部分
- Abstract(概要)：新規性部分，アイデア，実験結果

SIGGRAPHとは？

■ **世界最大**のCGの祭典

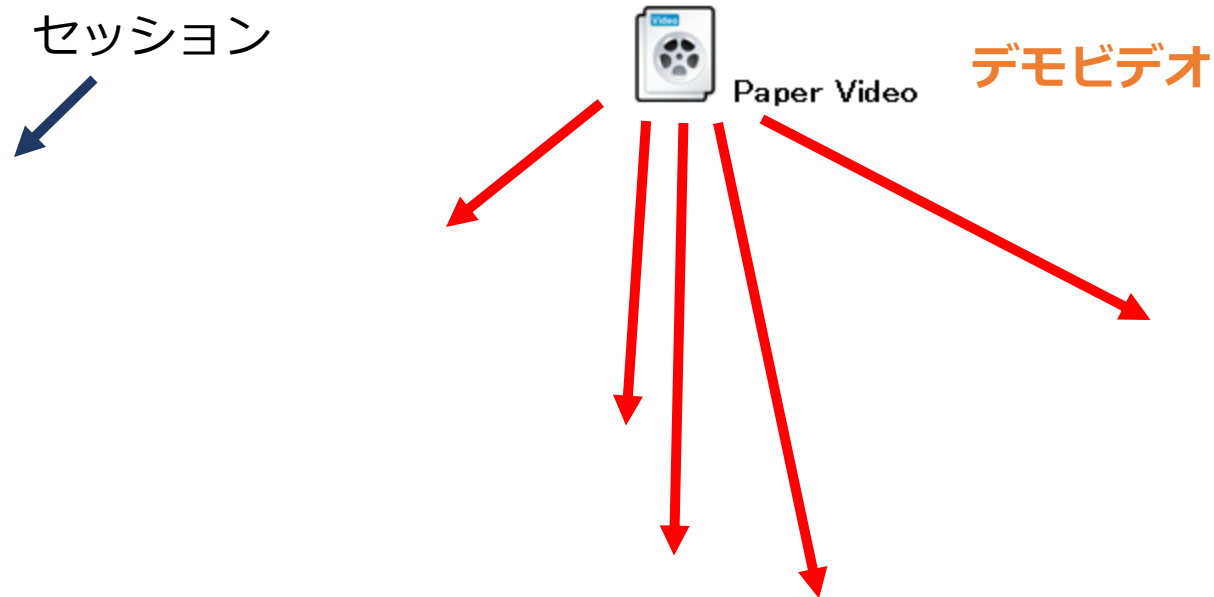
<http://s2014.siggraph.org/>

- 8/10～14 Vancouver convention center
- 14,045人の参加者(75ヶ国)
- 127 論文 / 550 論文投稿
- 35 セッション

https://www.youtube.com/watch?feature=player_embedded&v=u3Z1hDwGEmM

Ke-Sen Huangによる論文リスト

- 各論文がセッション毎にまとめられたサイト
 - <http://kesen.realtimerendering.com/sig2014.html>



論文のWebページ

著者情報



論文タイトル			
著者 1	著者 2	著者 3	...
所属 1	所属 2	著者 3	...

デモビデオ



論文のWebページ

■ Abstract (概要)

- 技術の**新規性部分**
- キーとなる**アイデア**
- **実験結果**から得られる効果

■ 論文のPDF

- **技術の詳細**が記述されている
- **画像**だけからでもある程度内容がわかる

次回

■研究最前線紹介

- レポート
- 国内海外のCG技術動向
- 全体のまとめ