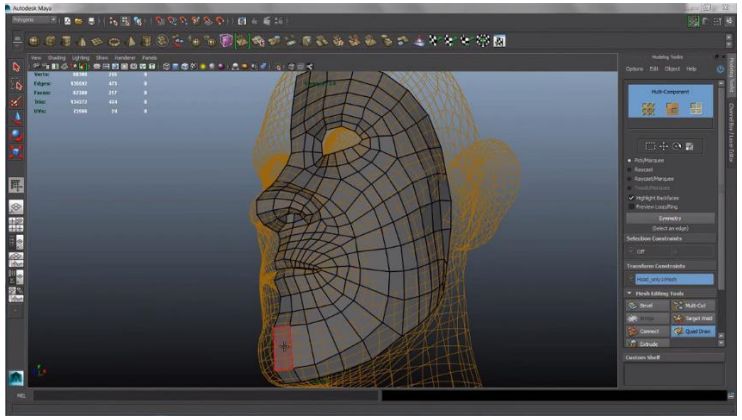
コンピュータグラフィックス

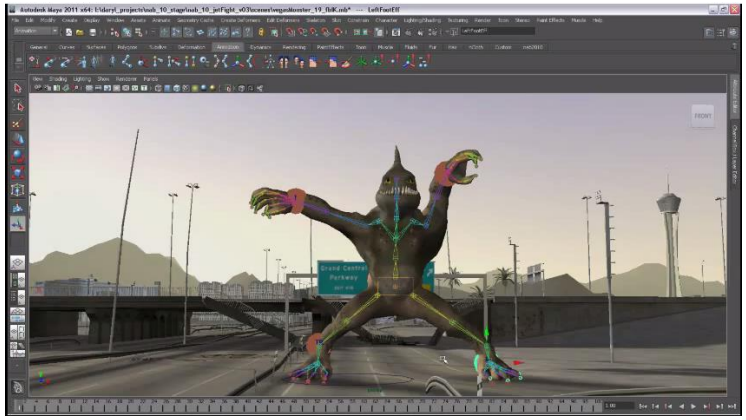
第13回
リアルタイムCG

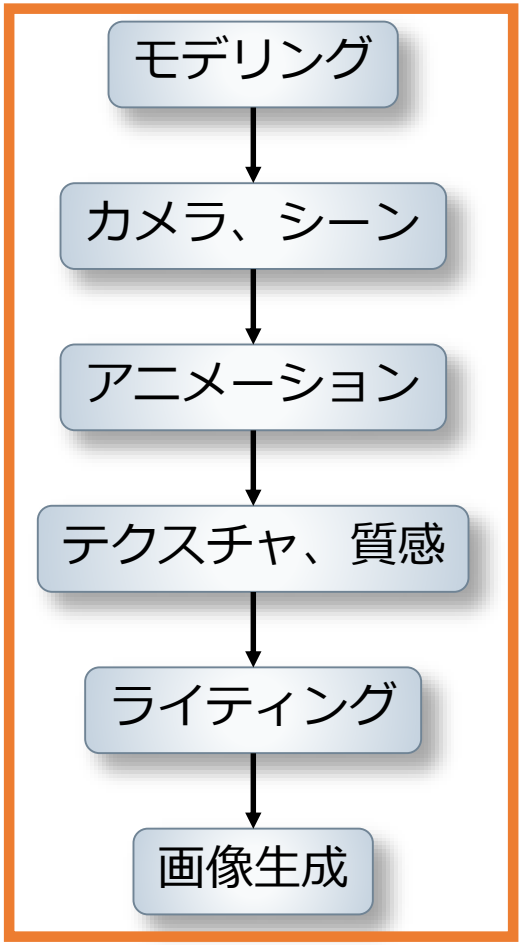
理工学部　兼任講師
藤堂 英樹

CG制作の主なワークフロー

- 3DCGソフトウェアの場合







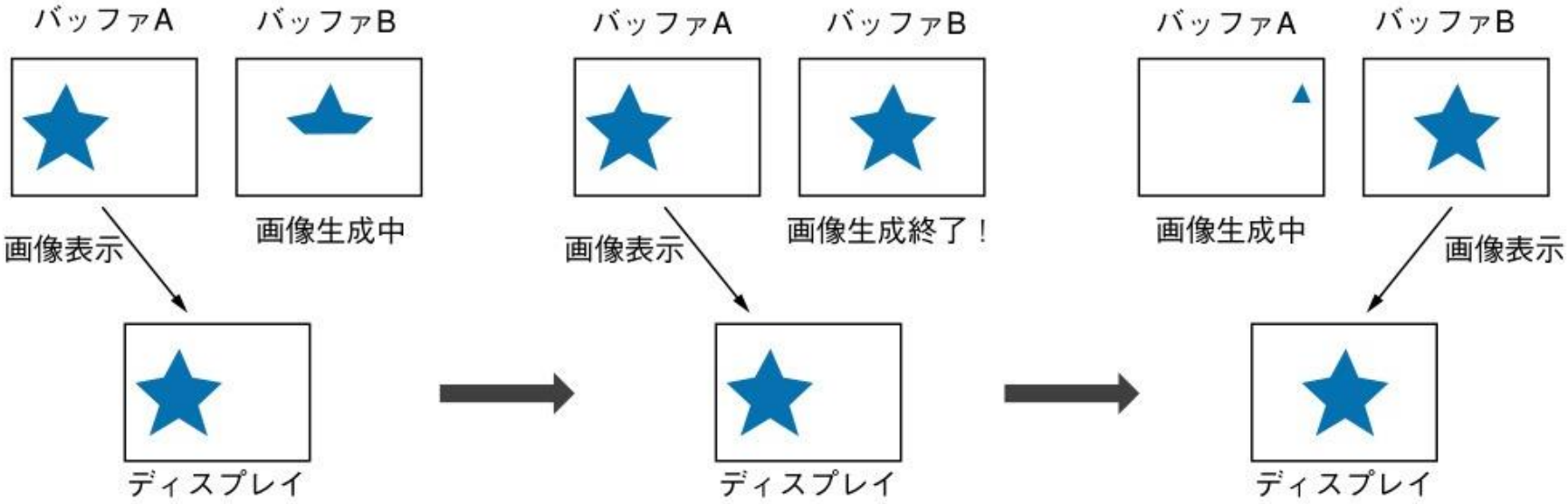
リアルタイムCG

- 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 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

- 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

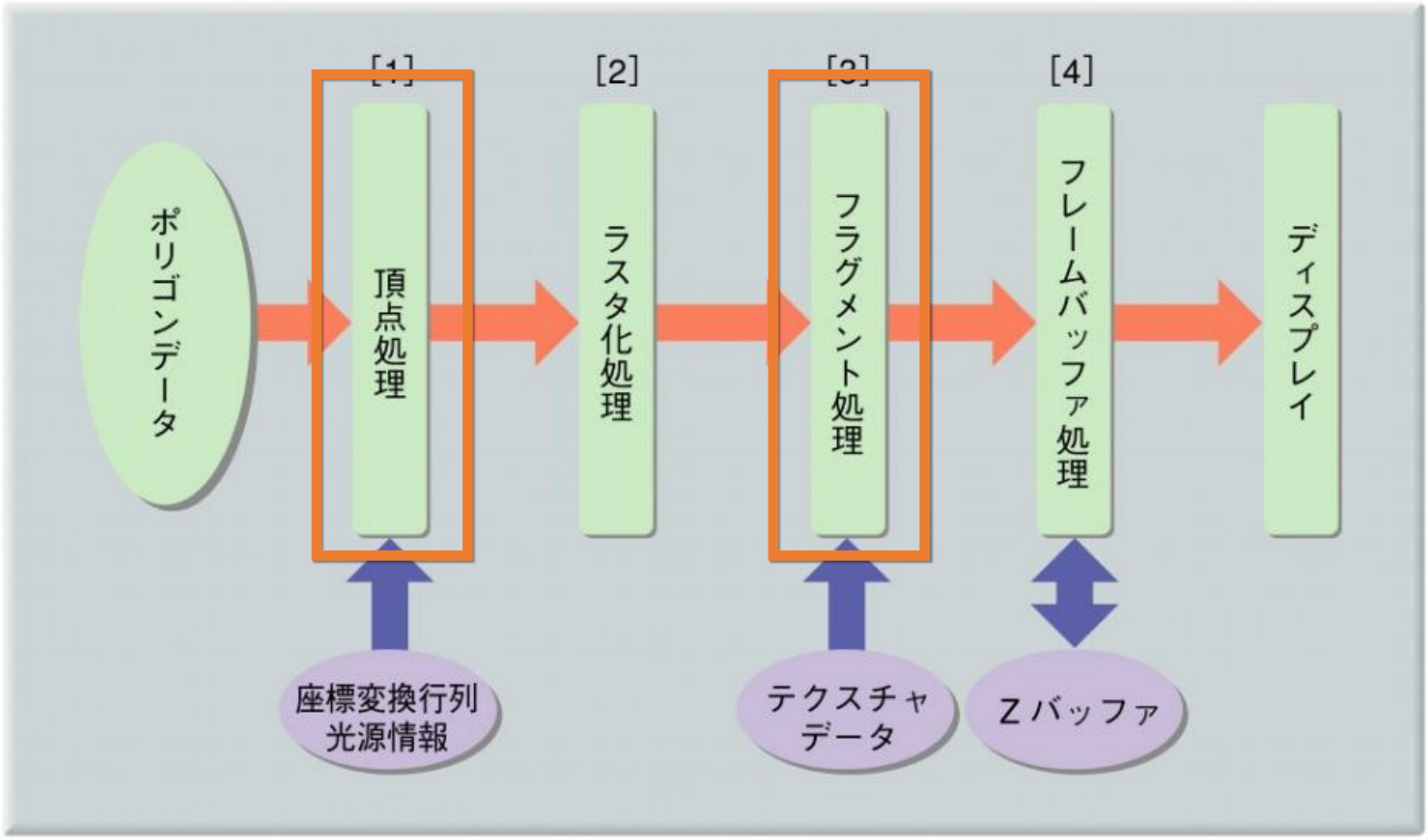
Unity上でのシェーダー開発

- 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

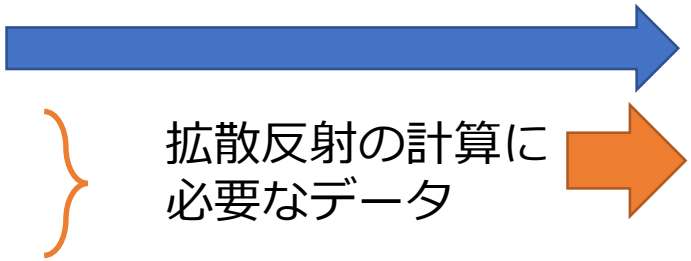
一番シンプルな例

- Unityの公式マニュアル
  - Vertex and Fragment Shader Examples

```
Shader "Custom/SolidColor" {
    SubShader {
        Pass {
            CGPROGRAM

            #pragma vertex vert
            #pragma fragment frag

            float4 vert(float4 v:POSITION) : SV_POSITION {
                return mul (UNITY_MATRIX_MVP, v);
            }

            fixed4 frag() : COLOR {
                return fixed4(1.0,0.0,0.0,1.0);
            }

            ENDCG
        }
    }
}
```

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

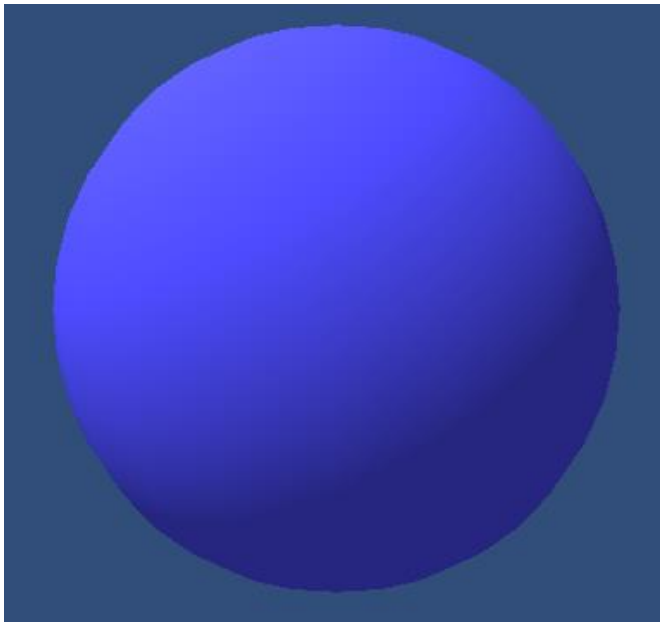



```
43          // 頂点毎の処理
44          // pos, L, N, RVのデータを計算する．
45          vertexOutput vert(appdata_base v) : POSITION
46          {
47              vertexOutput output;
48              // 座標変換後の位置
49              output.pos = mul (UNITY_MATRIX_MVP, v.vertex);
50
51              // 法線ベクトル
52              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

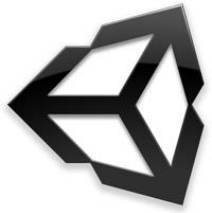
- GPU Gems ©Randima Fernando
  - 様々なリアルタイムCG手法を紹介
  - デモ解説ページ

アニメーションの
GPUによる高速化

サブサーフェス
スキャッタリング

リアルタイムCGシステム

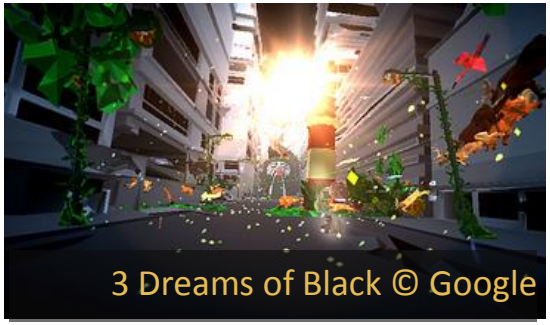
- ゲームエンジン
  - ゲーム作成に有用なデータ
  - インタラクティブなCG処理



Unity

Unreal Engine

- 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

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

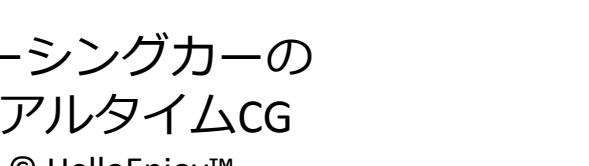
- Unity Web Player
  - 現在は非推奨に⇒**WebGL**のサポート

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

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

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

山岳地形のリアルタイムCG
© mathajie

ストームの可視化
© Callum Prentice

WebGL

水のリアルタイム
シミュレーション
© Evan Wallace

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

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

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

Live2D
2Dを活かした立体表現

Live2DシステムとUnityの連携
© Live2D

SIGGRAPHとは？

- **世界最大**のCGの祭典
  - 8/10～14 Vancouver convention center
  - 14,045人の参加者(75ケ国)
  - 127 論文 / 550 論文投稿
  - 35 セッション

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

- 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