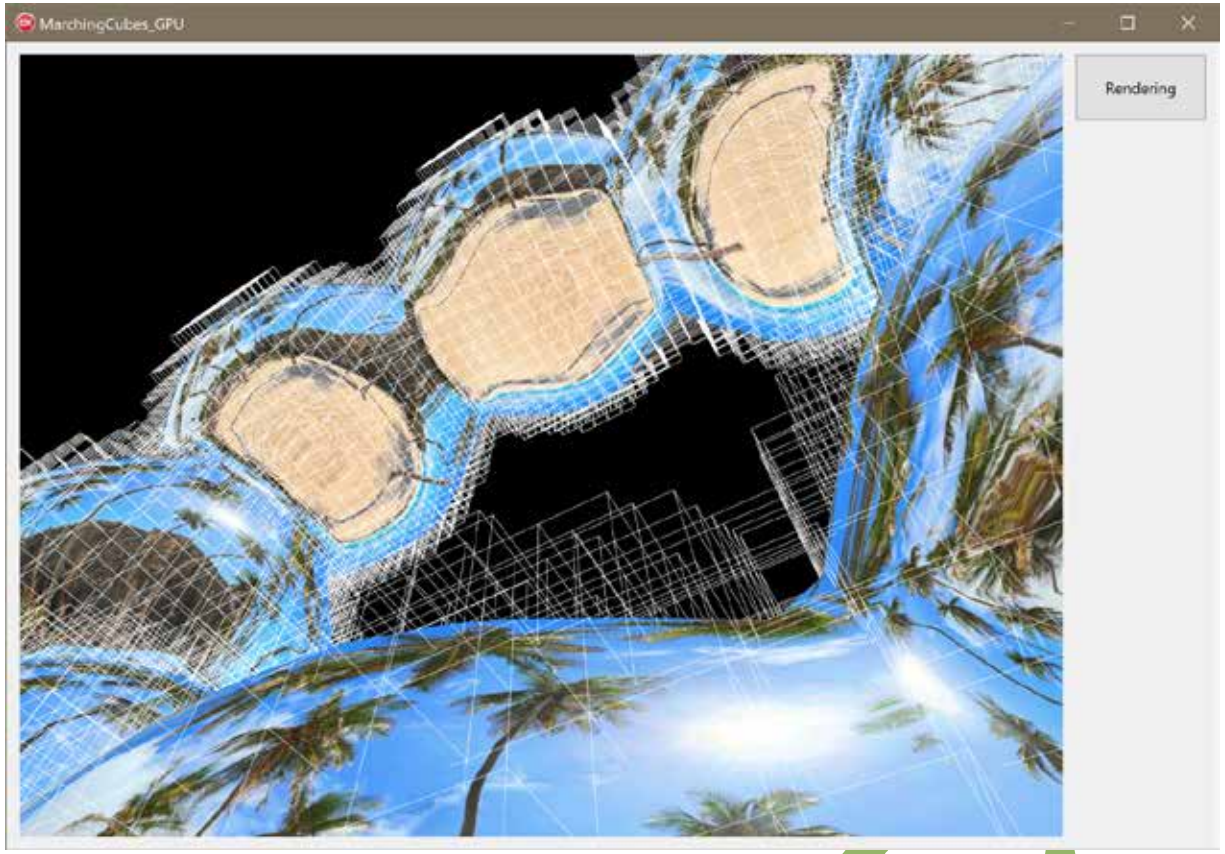
ボクセルで３Ｄモデリング

～ブロックを積むだけの簡単なお仕事～

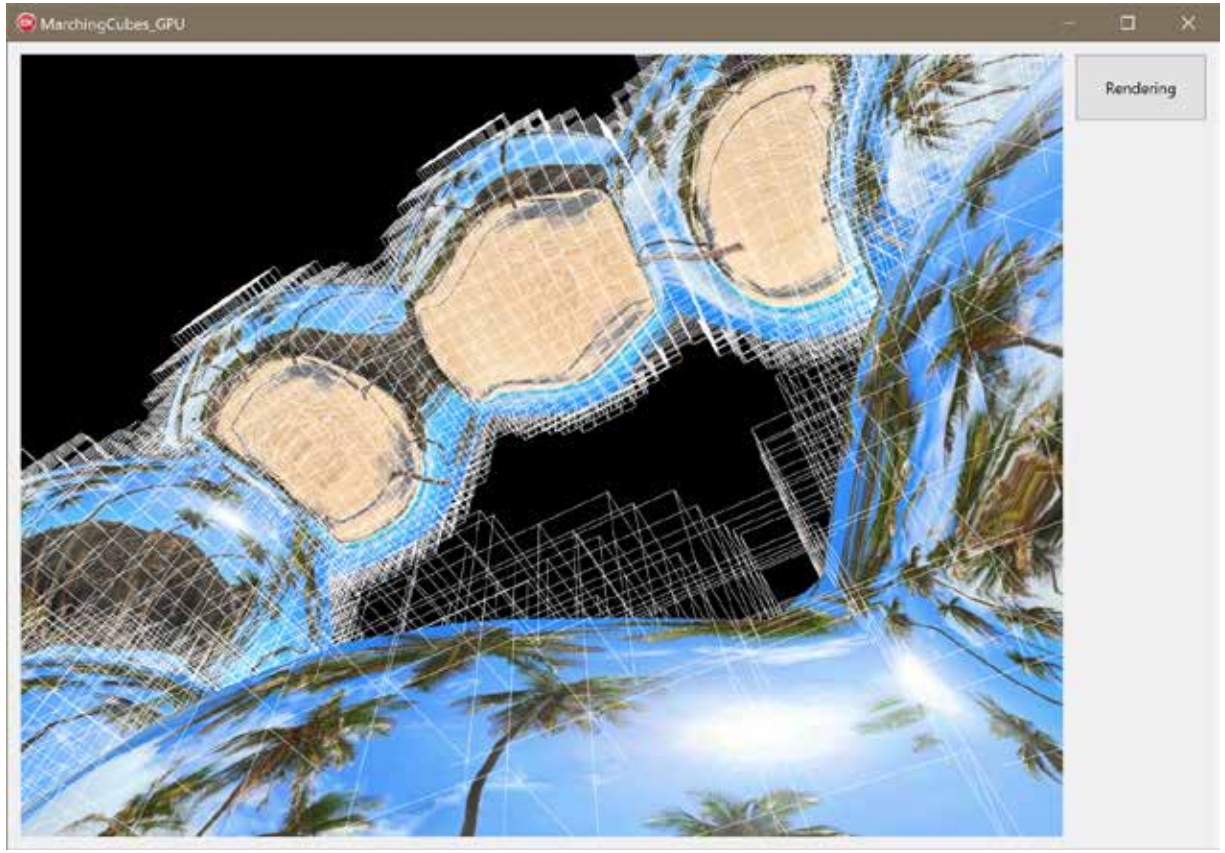
第35回 エンバカデロ・デベロッパーキャンプ

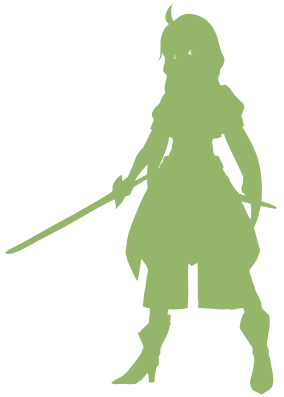


慶應義塾大学 大学院工学研究科 藤代研究室
リサーチフェロー　**中山 雅紀**

embarcadero®
DEVELOPER CAMP

§はじめに

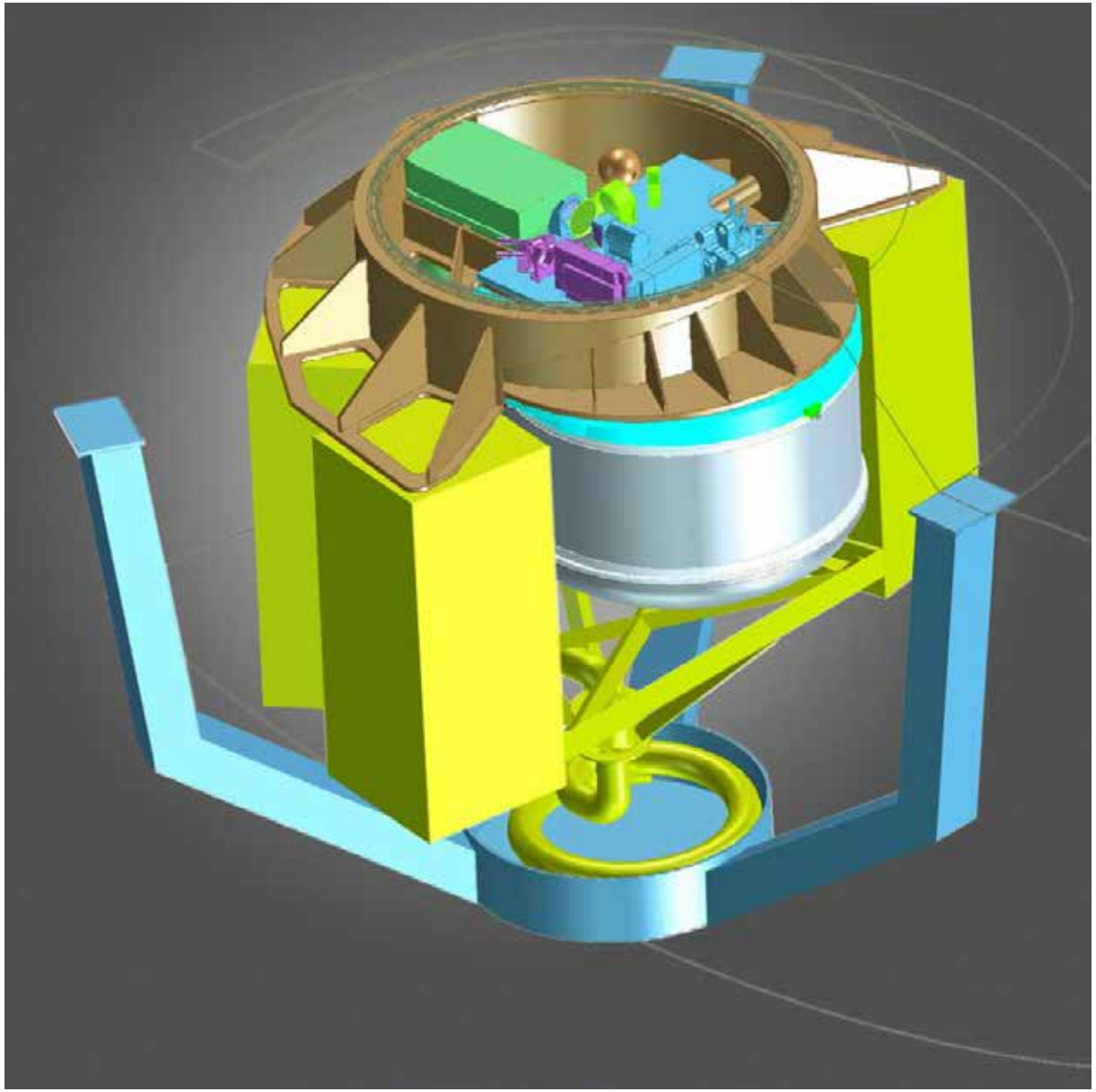




embarcadero
DEVELOPER CAMP

アジェンダ .はじめに

- ３Ｄモデルの作り方
  - FireMonkey の場合
  - OpenGL の場合
- モデリングの方法論
  - 陽関数でモデリング
  - 陰関数でモデリング
  - ブロックでモデリング
  - ボリュームでモデリング
- 断層写真の立体化
- ポリゴンをボクセルへ変換
- テトラヘドライゼーション

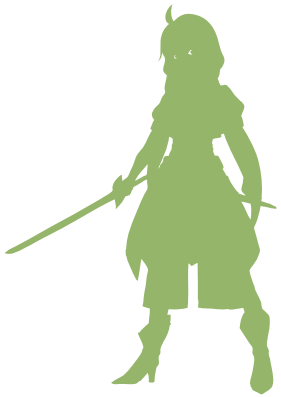


© en.wikipedia.org/wiki/3D_modeling

§自己紹介

- § 怪しい人じゃないよ。

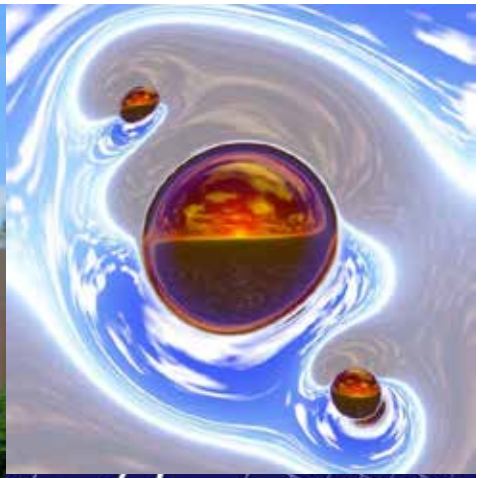
https://deepart.io

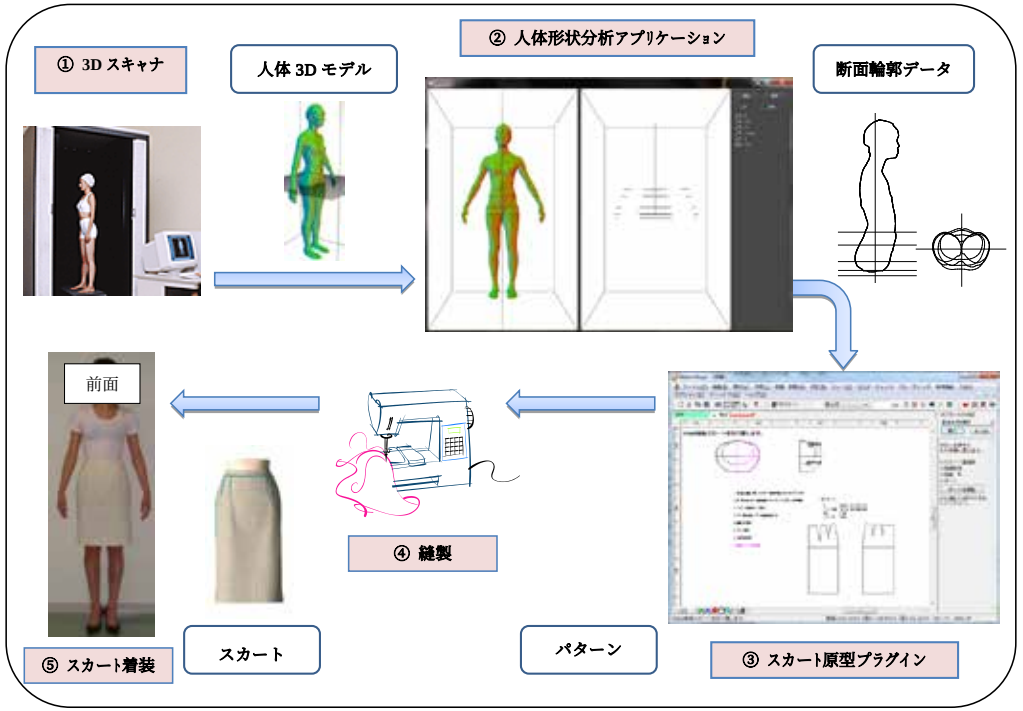


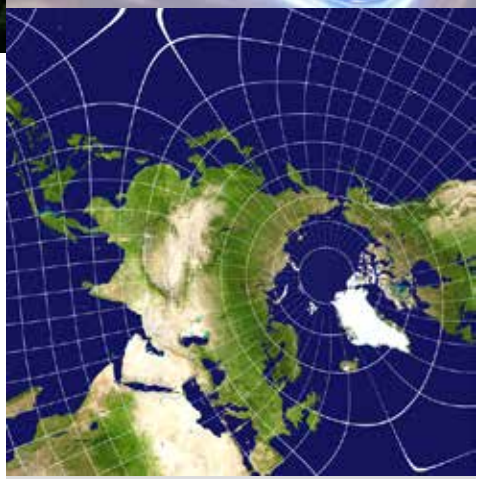
embarcadero DEVELOPER CAMP

専門 .自己紹介

- § 大学
  - 慶應義塾大学 藤代研究室
    - リサーチフェロー
  - 和洋女子大学 山本研究室
    - 共同研究員
- § 研究
  - 球面上の信号処理
  - フォトリアルＣＧ
  - 立体視映像
  - ３Ｄアパレル設計
  - ３Ｄプリンタ
  - ボリュームモデリング

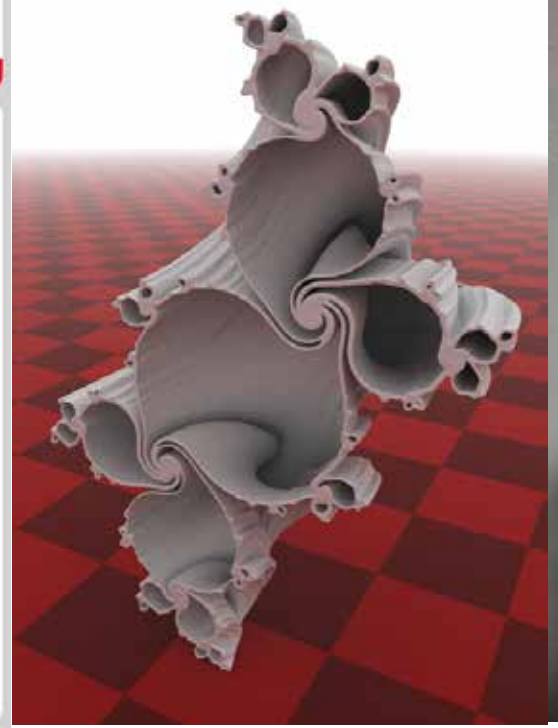






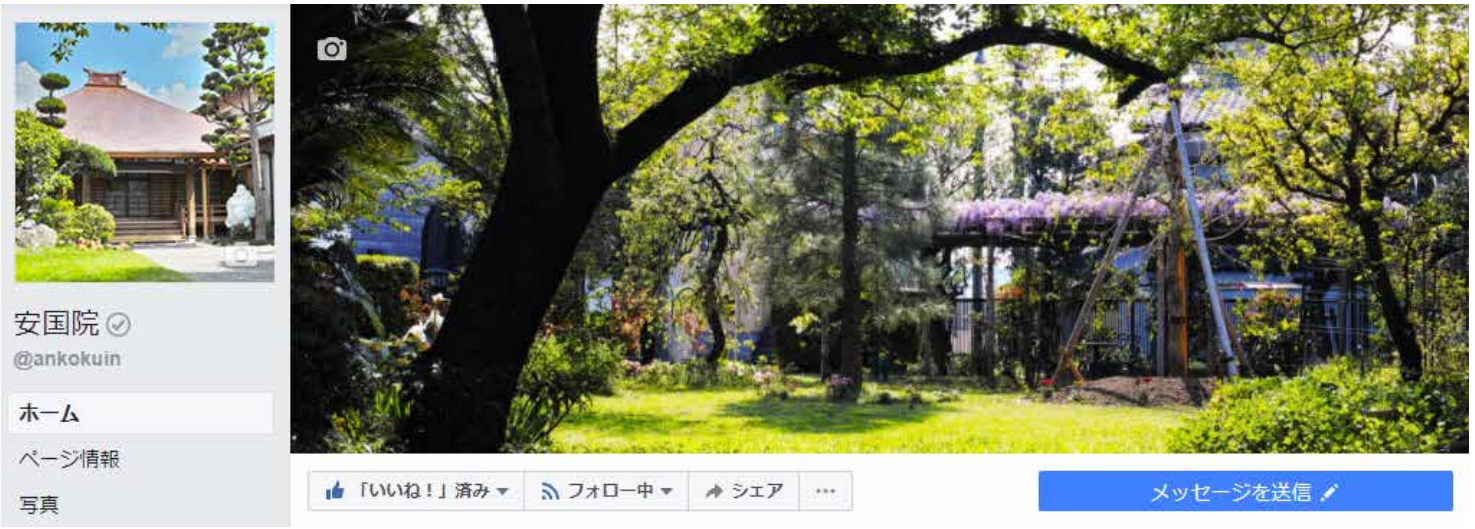
趣味 .自己紹介

- アルゴリズムで生成されるメディアアート制作
  - 中学生時代からフラクタルに心酔
  - 絶滅危惧種のステレオグラム作家



本業 .自己紹介

- 玉泉山 安国院
  - 千葉県市川市
  - 住職



www.facebook.com/ankokuin

連絡先 .自己紹介

§ 是非お友達になって下さい！

facebook.com/luxidea

twitter.com/luxidea

facebook.com/fujishiro.lab

twitter.com/fujishirolab

facebook.com/luxophia

twitter.com/luxophia

コミュニティ .自己紹介

- ＣＧ技術を学問的に探究するコミュニティ（Delphiに限らない）

facebook.com/academia.graphics

twitter.com/AxGRAPH

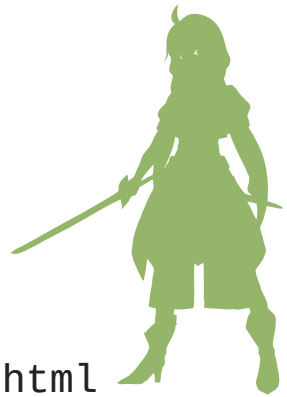
- ＣＧ技術の Delphi による実装を探究するコミュニティ

facebook.com/DelphiCG

twitter.com/delphi_cg

§３Ｄコンポーネントの作り方

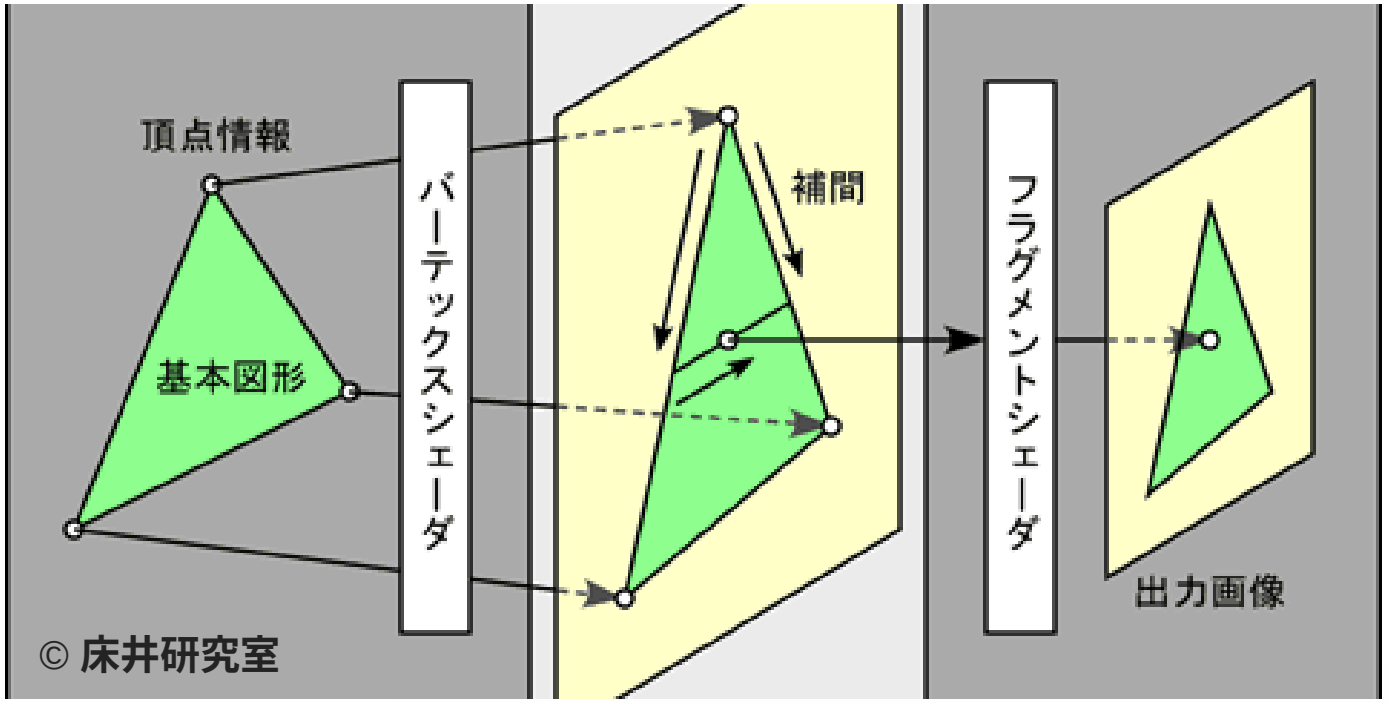
© ht-deko.com/AppMethod/summary/fmx_3d.html

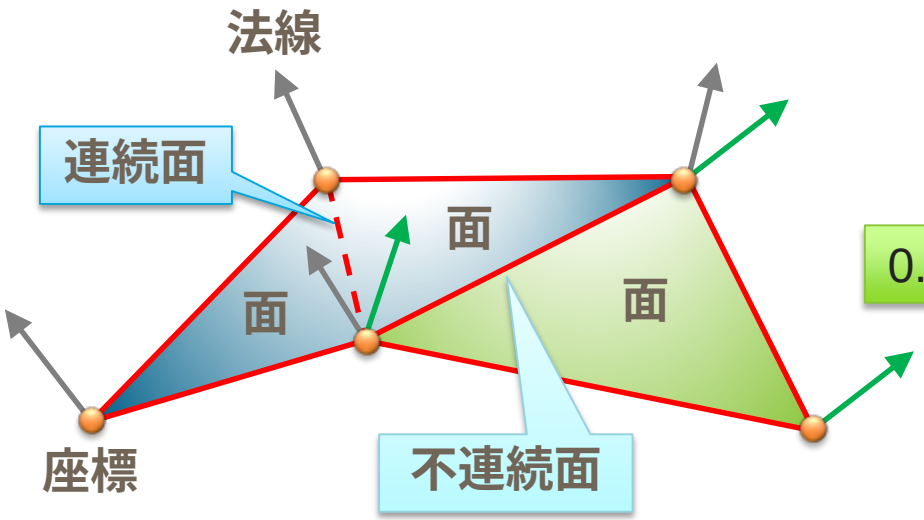


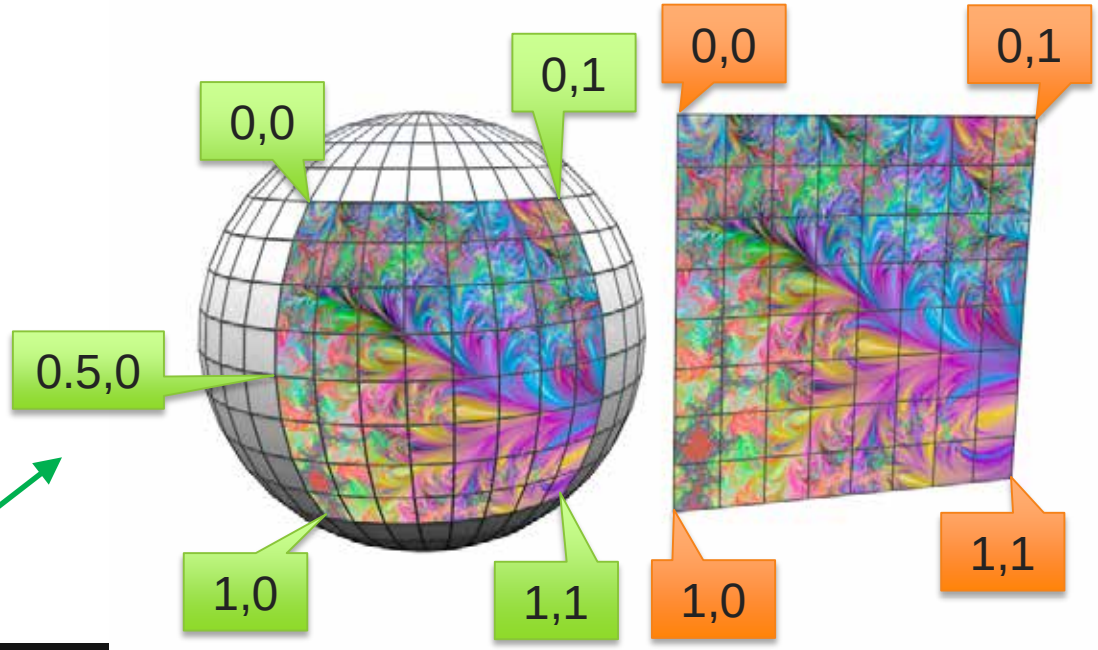
embarcadero DEVELOPER CAMP

ＧＰＵに描かせる方法

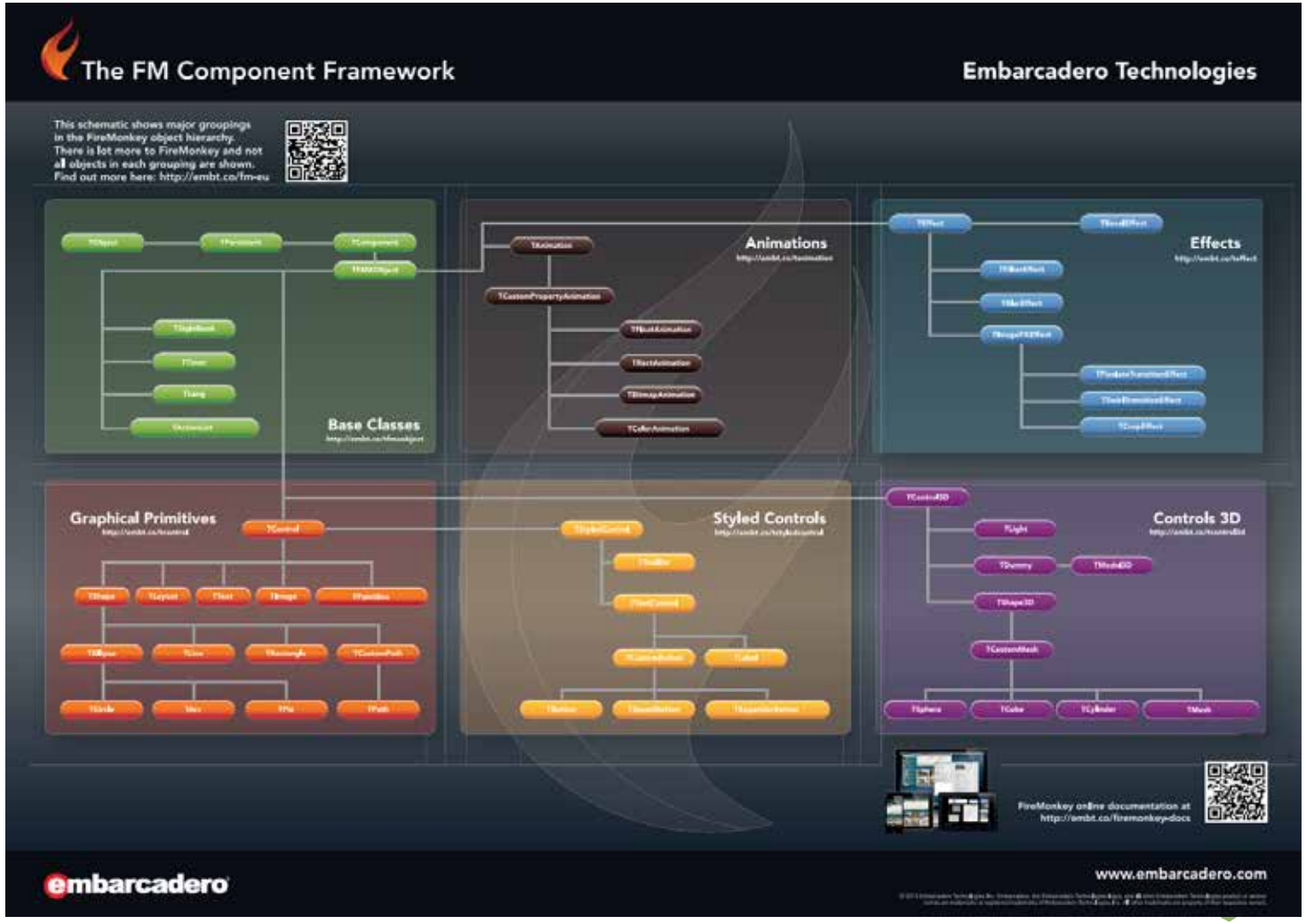
- § 点･線･三角しか描けない
  - 頂点
    - 位置座標（３Ｄ）
    - 法線：面の垂直ベクトル（３Ｄ）
    - テクスチャ座標（２Ｄ）
  - 面
    - ３つの頂点への参照







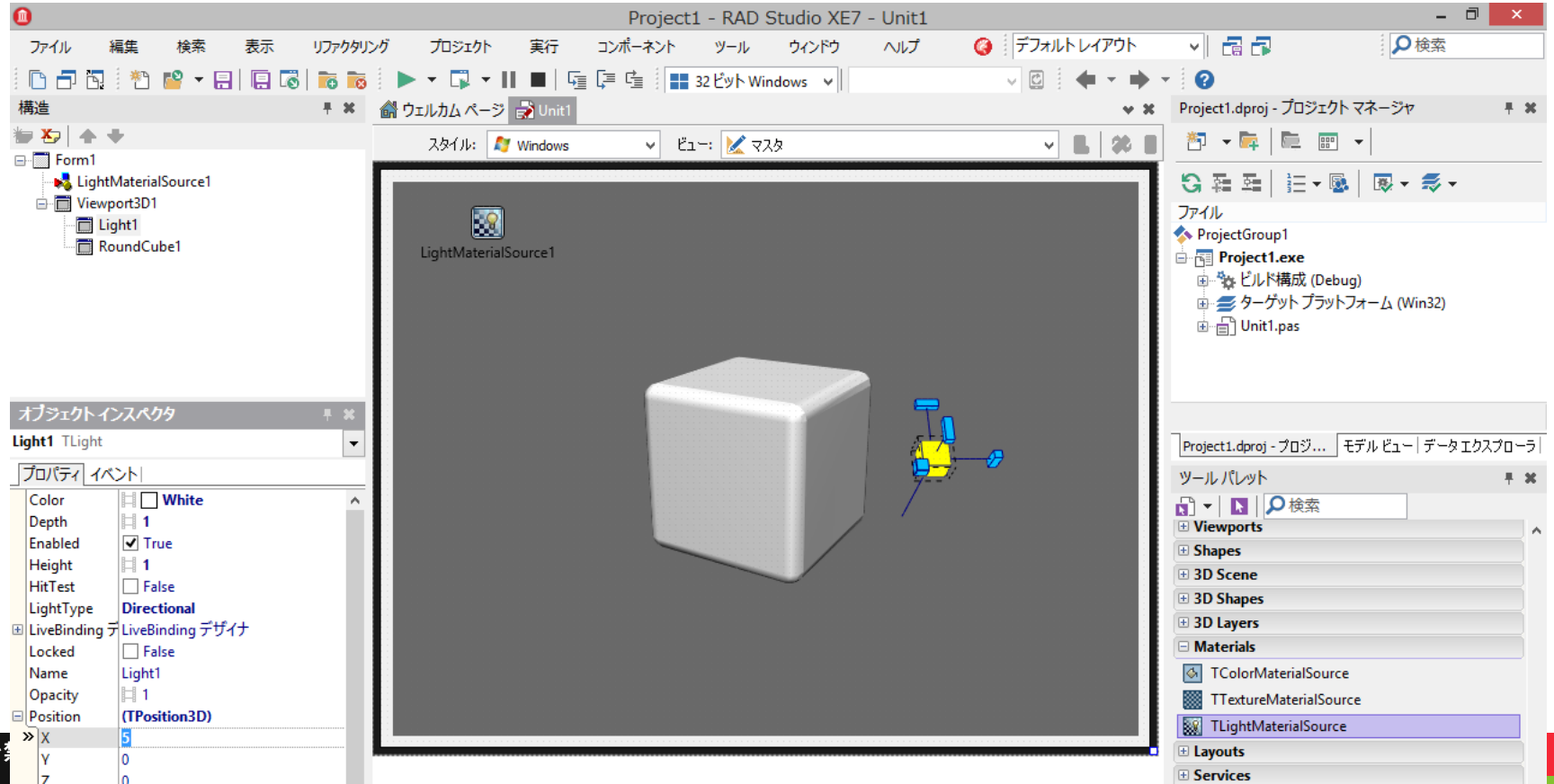
§FireMonkey の場合



embarcadero DEVELOPER CAMP

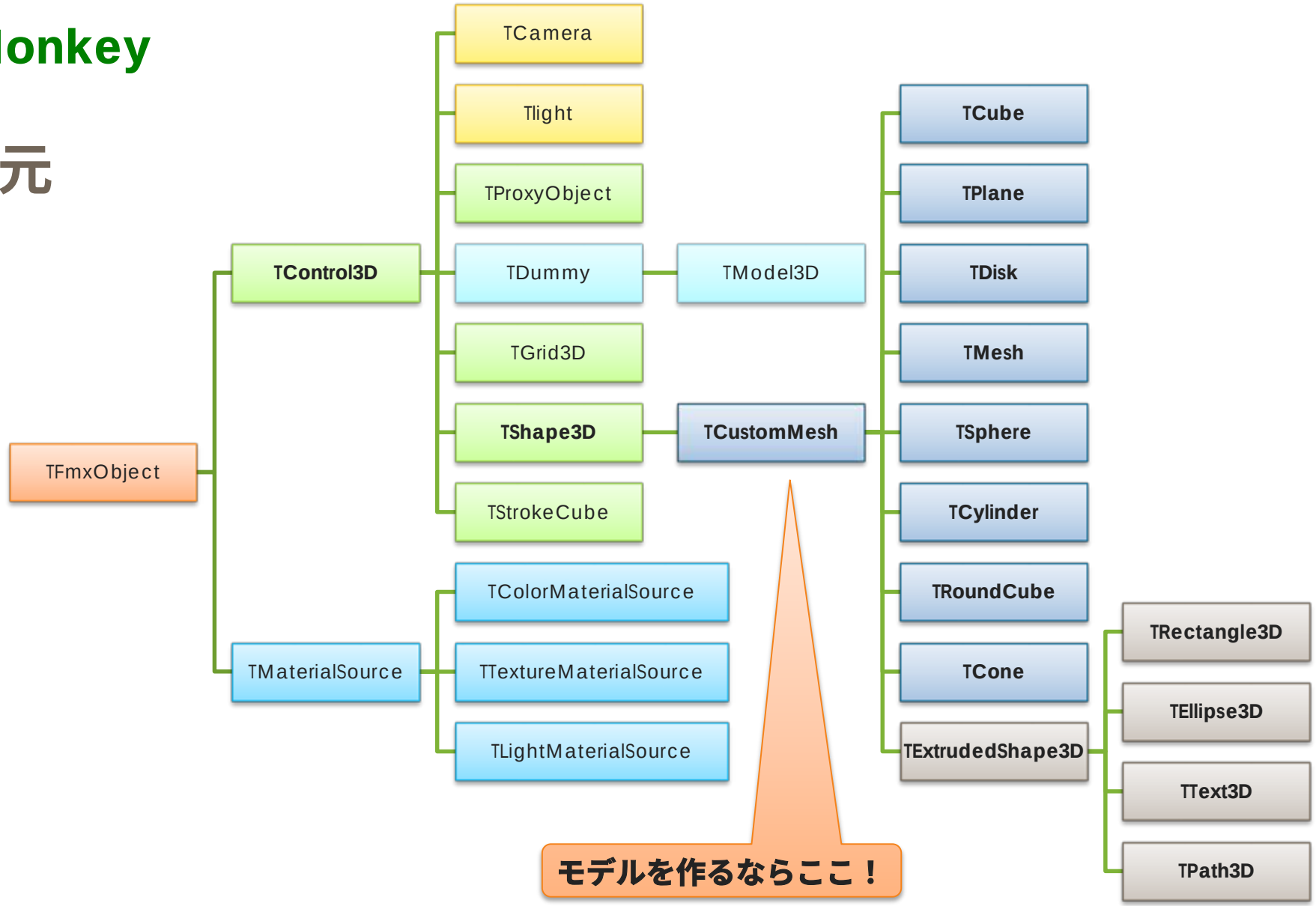
コンポーネント .FireMonkey

- ＧＰＵを活用したリアルタイムな３ＤＵＩを提供可能
  - Windows：Direct3D
  - macOS，iOS, Android：OpenGL



３Ｄクラス .FireMonkey

- TControl3D が大元
  ↓
- TShape3D
  ↓
- TCustomMesh



転送バッファ .FireMonkey

- TShape3D = class(TControl3D)
  - property MaterialSource :TMaterialSource
    - マテリアルコンポーネントへの参照
- TCustomMesh = class(TShape3D)
  - property Data :TMeshData
    - ポリゴンデータを保持する
- TMeshData = class(TPersistent)
  - property VertexBuffer :TVertexBuffer
    - 頂点配列
  - property IndexBuffer :TIndexBuffer
    - 面配列

頂点バッファ .FireMonkey

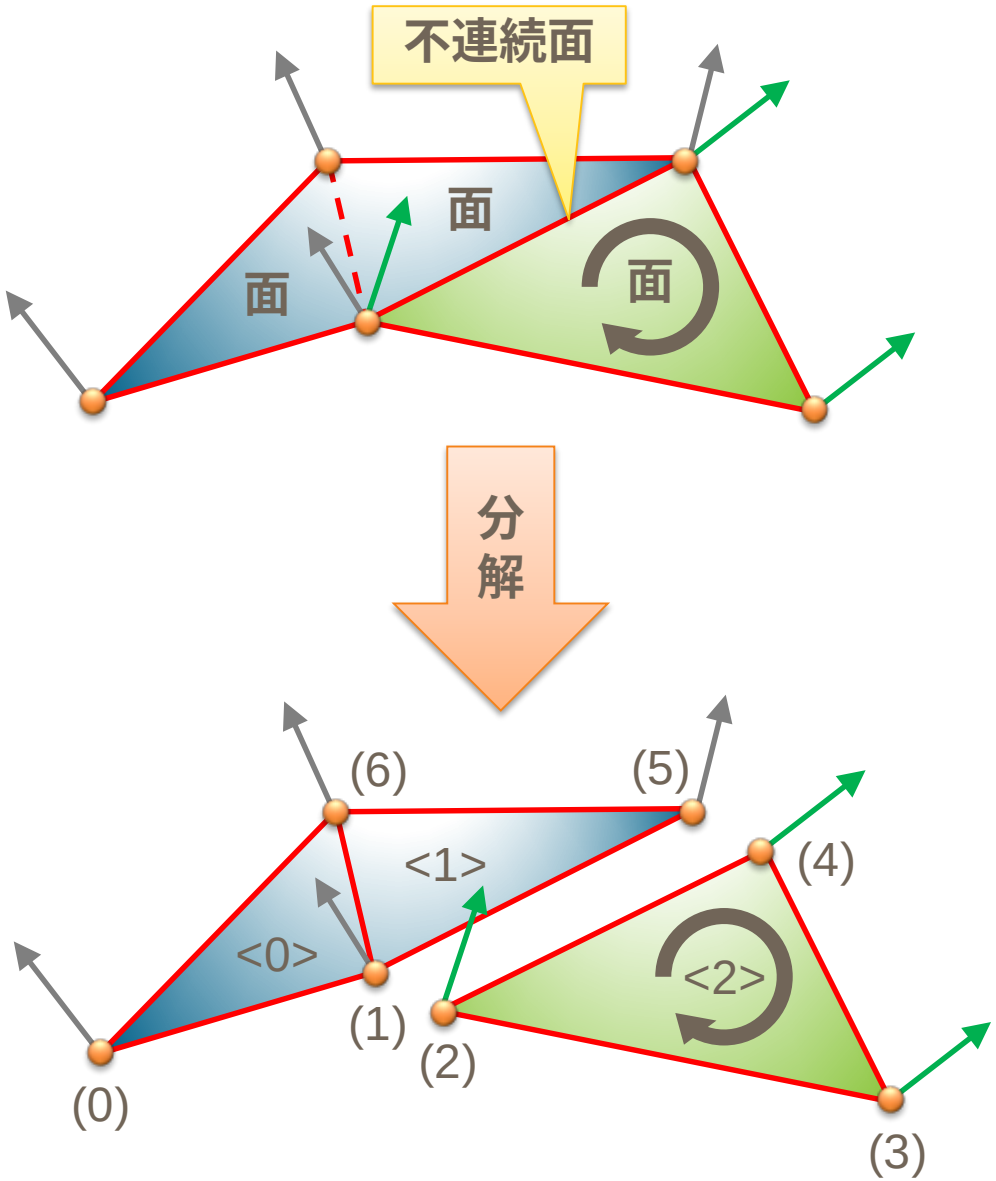
- TVertexBuffer = class(TPersistent)
  - property Length :Integer
    - 頂点数
  - property Vertices[AIndex:Integer] :TPoint3D
    - 頂点の位置座標
  - property Normals[AIndex:Integer] :TPoint3D
    - 頂点の法線ベクトル
  - property TexCoord0[AIndex:Integer] :TPointF
    - 頂点のテクスチャ座標

面バッファ .FireMonkey

- TIndexBuffer = class(TPersistent)
  - property Length ：Integer
    - 面数×３ ：ポリゴン
    - 線数×２ ：ワイヤーフレーム
    - 点数×１ ：ポイントクラウド
  - property Indices[AIndex:Integer] :Integer
    - N番目のポリゴンの属する頂点のインデックス番号
    - Indices[3×N＋0] ＝頂点①
    - Indices[3×N＋1]＝頂点②
    - Indices[3×N＋2]＝頂点③

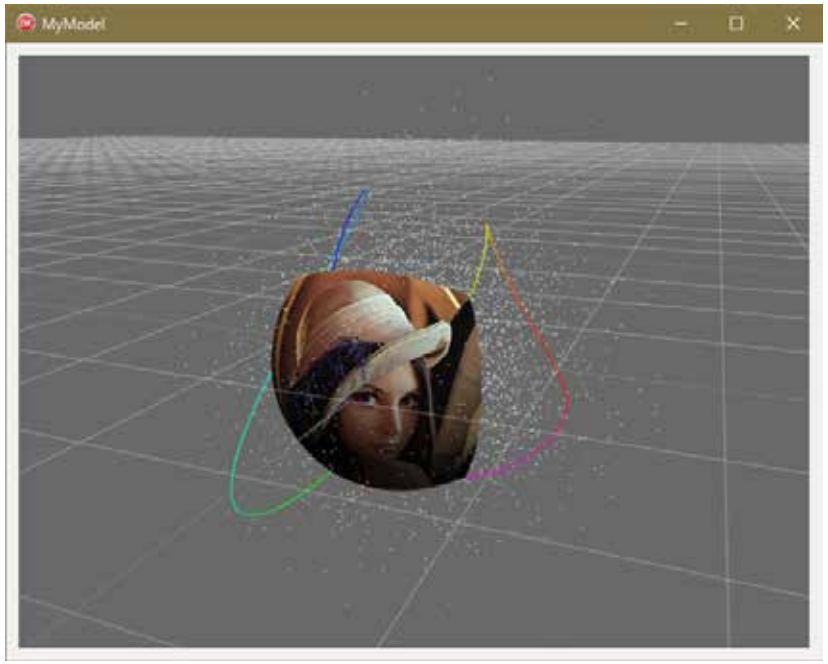
実装 .FireMonkey

```
with Data do
begin
  with VertexBuffer do
  begin
    Length := 7{頂点数};
    Vertices [ 0 ] := TPoint3D.Create( *, *, * ){位置座標};
    Normals  [ 0 ] := TPoint3D.Create( *, *, * ){法線ベクトル};
    TexCoord0[ 0 ] := TPointF .Create( *, *    ){テクスチャ座標};
      ～
    Vertices [ 6 ] := TPoint3D.Create( *, *, * );
    Normals  [ 6 ] := TPoint3D.Create( *, *, * );
    TexCoord0[ 6 ] := TPointF .Create( *, *    );
  end;
  with IndexBuffer do
  begin
    Length := 3{頂点数/ポリゴン} * 3{ポリゴン数};
    Indices[ 3*0+0 ] := 6;
    Indices[ 3*0+1 ] := 1;
    Indices[ 3*0+2 ] := 0;
    Indices[ 3*1+0 ] := 6;
    Indices[ 3*1+1 ] := 5;
    Indices[ 3*1+2 ] := 1;
    Indices[ 3*2+0 ] := 4;
    Indices[ 3*2+1 ] := 3;
    Indices[ 3*2+2 ] := 2;
  end;
end;
```

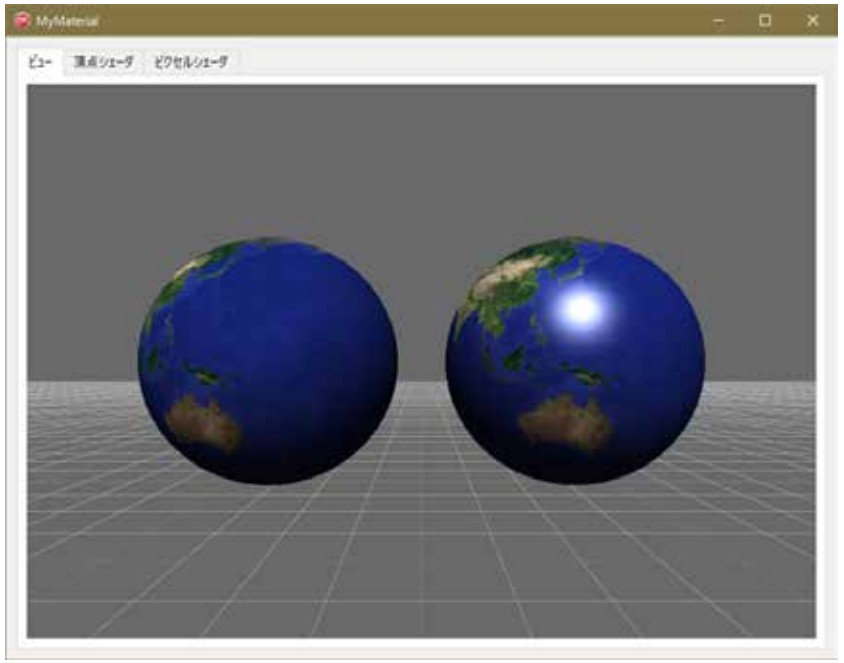


デモ .FireMonkey

§ Direct3D（HLSL）を使用



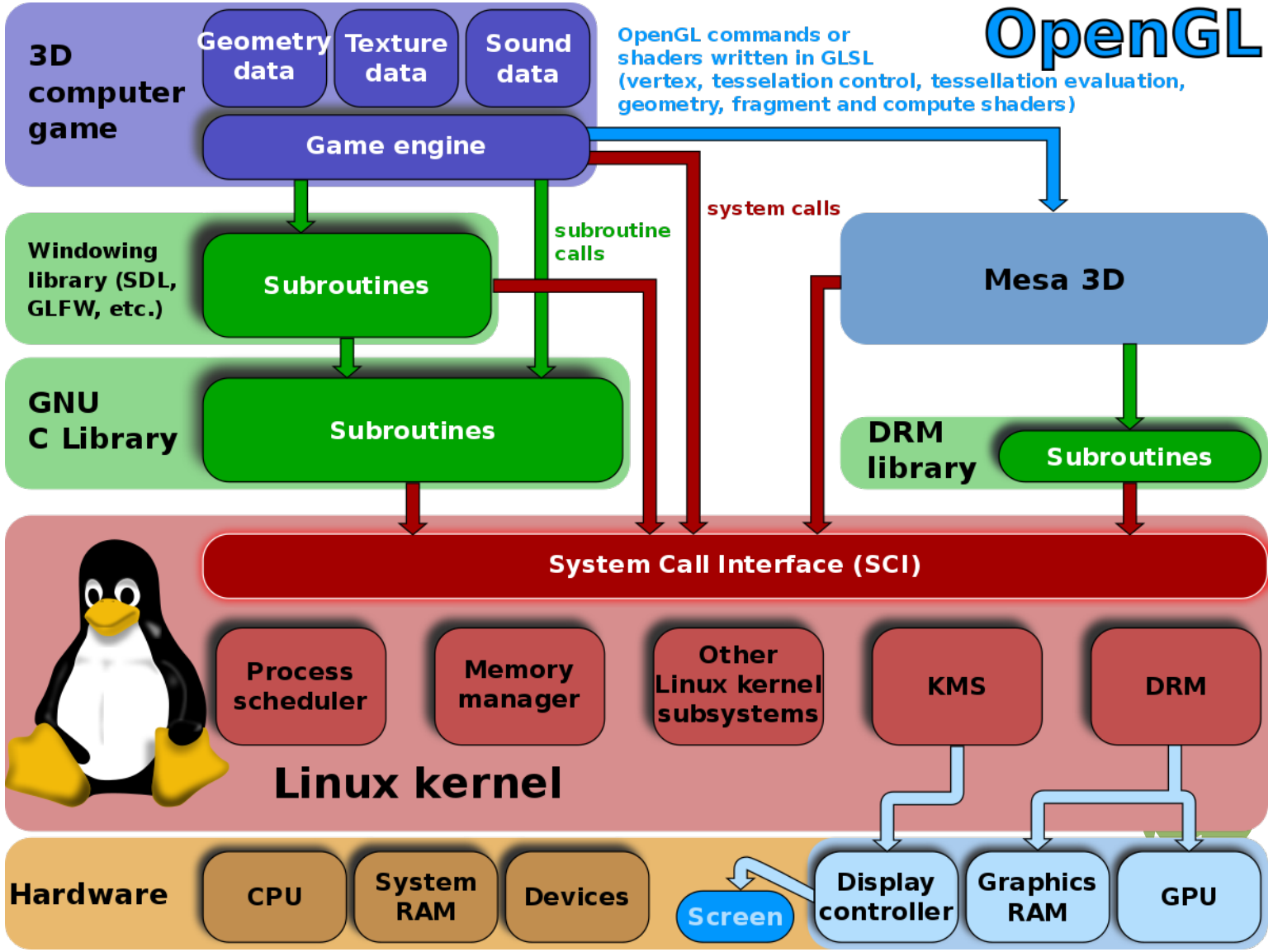
github.com/LUXOPHIA/MyModel



github.com/LUXOPHIA/MyMaterial

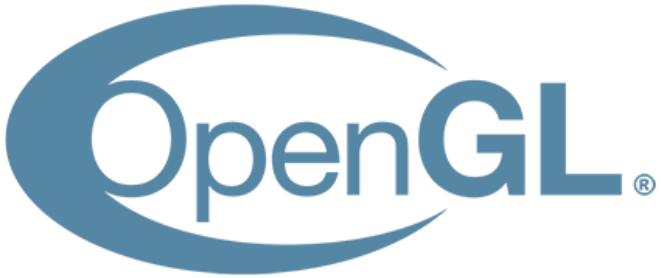
github.com/LUXOPHIA/MyAsset

§OpenGL の場合

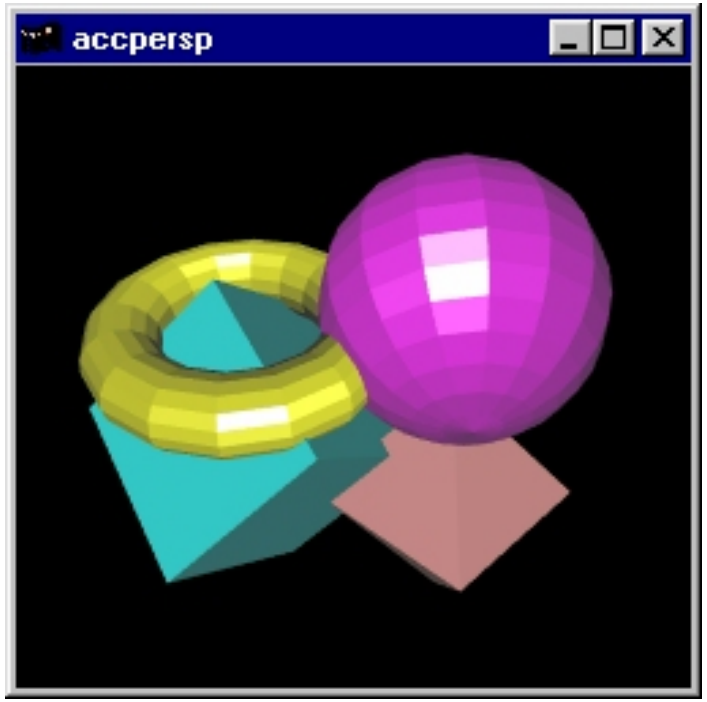


embarcadero DEVELOPER CAMP

OpenGL とは？ .OpenGL



- **リアルタイム２Ｄ/３ＤＣＧ用ＡＰＩ**
  - **GPU：**Graphics Processing Unit **を活用**
    - **コンシューマ向けグラフィックボード**
    - **クリエイター向けグラフィックボード**
- Khronos Group **が策定**
  - www.khronos.org
- **クロスプラットフォーム**
  - **Windows**, macOS, Android, iOS, Linux...
- C-API **ベース**
  - **ラッパー作りが容易**
  - Windows
    - **gl.h → Winapi.OpenGL.pas**
    - **glext.h → Winapi.OpenGLext.pas**

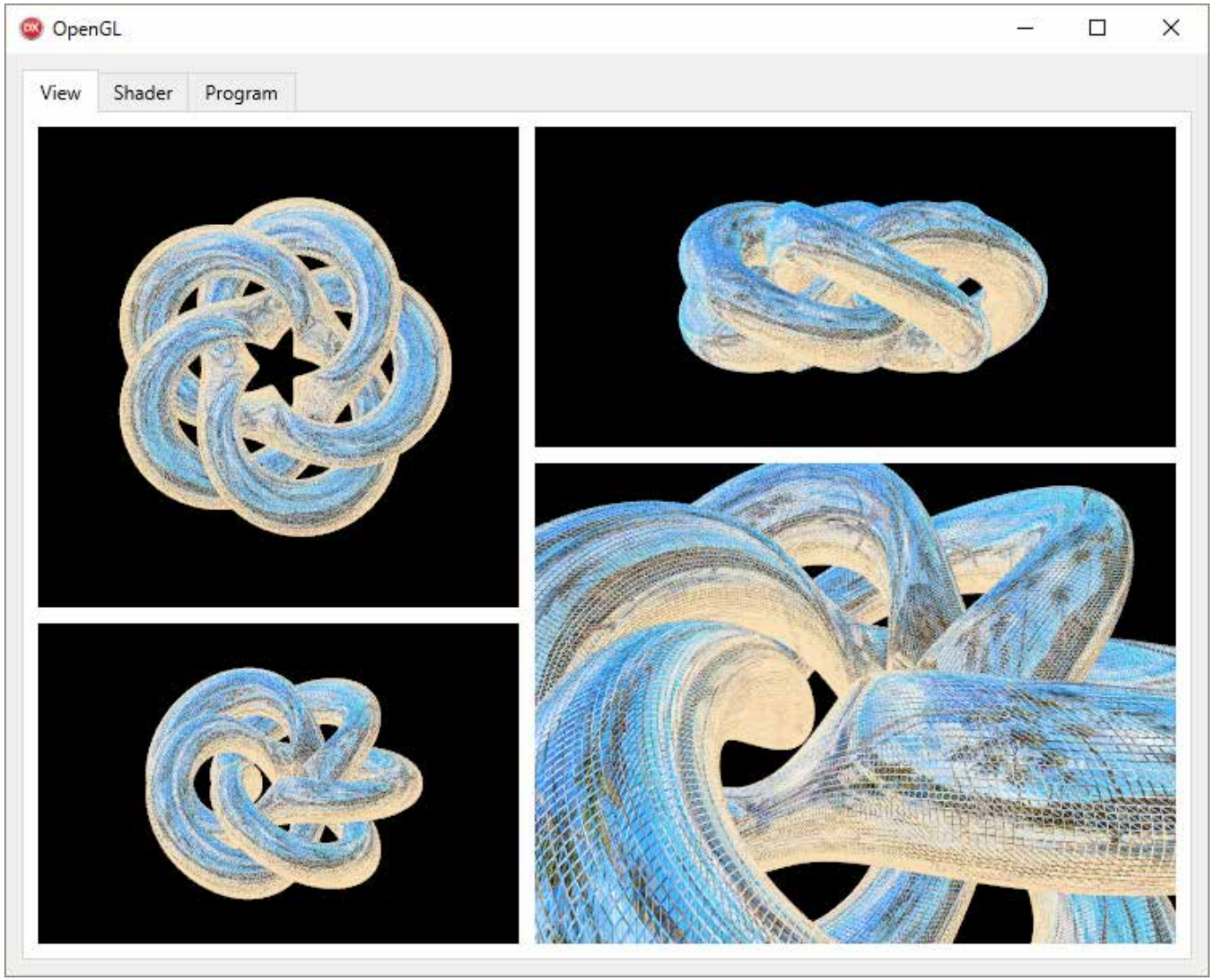


中山ライブラリ .OpenGL

- FMX と VCL の両方に対応
- OpenGL のスペックを
  完全に引き出せる

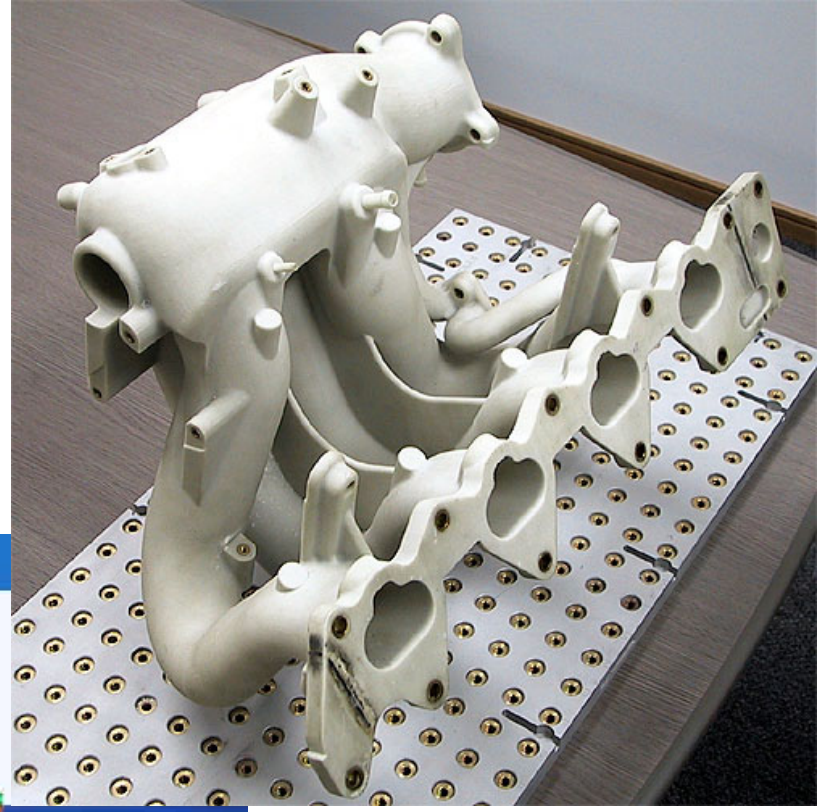
github.com/LUXOPHIA
/OpenGL

github.com/LUXOPHIA
/OpenGL_VCL



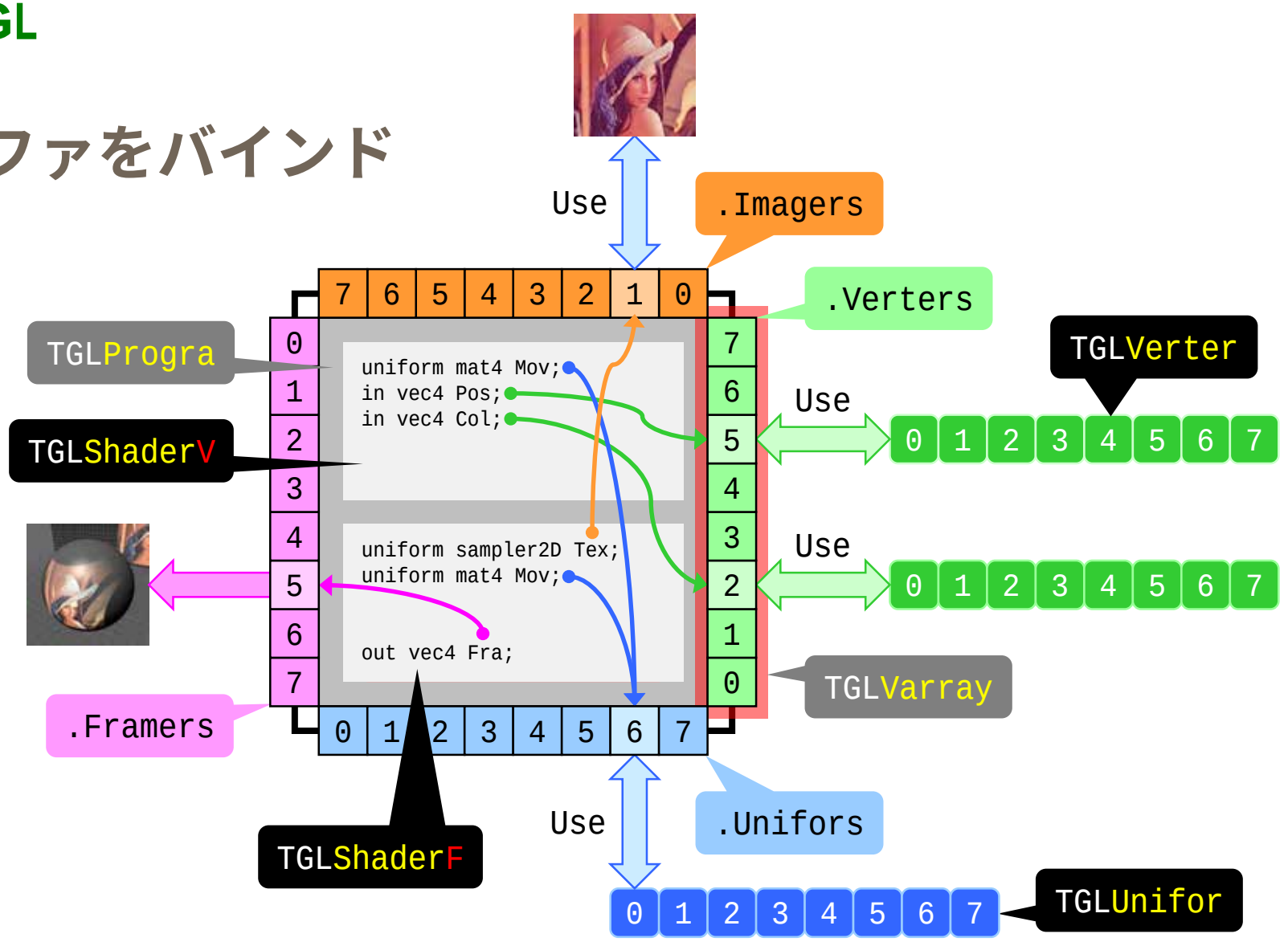
実績 .中山ライブラリ.OpenGL

§ ３Ｄプリンタ制御ソフトの（表示部分の）
開発に携わらせて頂きました！



転送バッファ .OpenGL

§ シェーダへ各種バッファをバインド

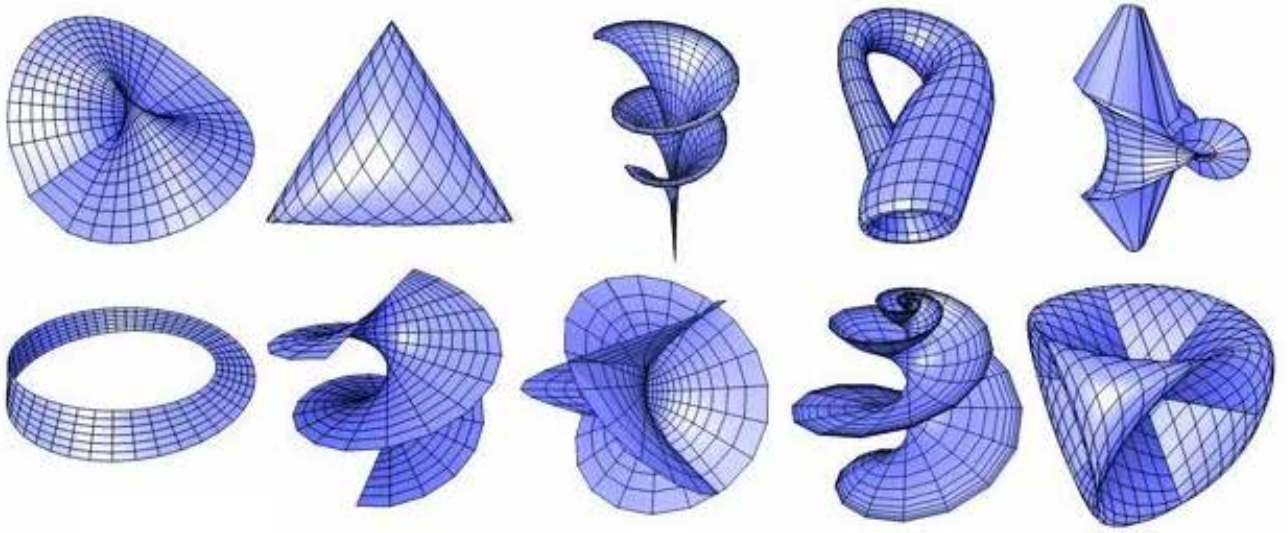


§モデリングの方法論

embarcadero DEVELOPER CAMP

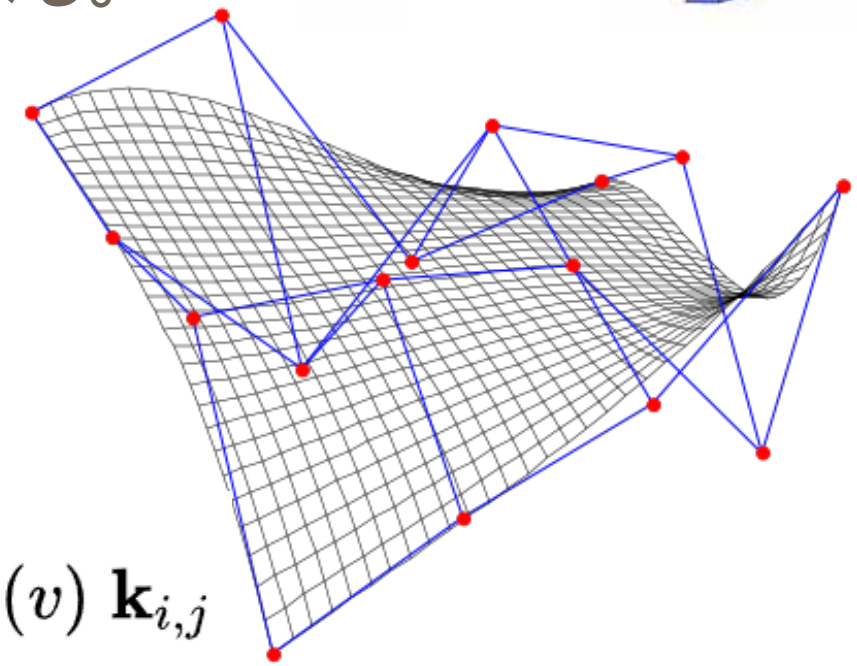
© www.3d-meier.de/tut3/Seite0.html

陽関数 . モデリング



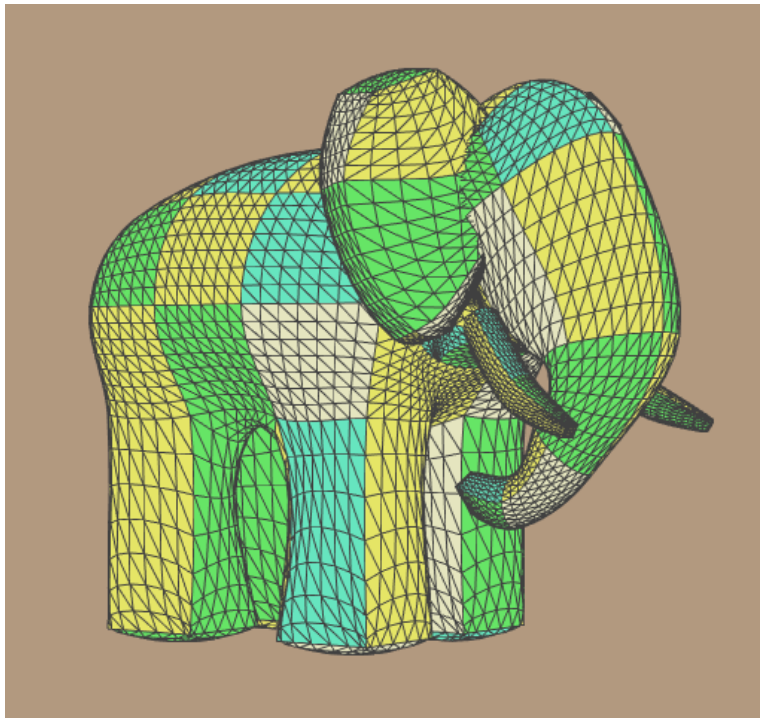
§ 陽関数表現：x,y,z = f(u,v)
- パラメトリック曲面
- パラメータに従って
  表面座標が一意に決まる。
- 計算コスト：低

§ 設計自由度の向上
- 陽関数のツギハギ
  - ベジエ曲面
  - Ｂスプライン曲面
  - ＮＵＲＢＳ曲面



$$\mathbf{p}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} B_i^n(u)\, B_j^m(v)\, \mathbf{k}_{i,j}$$

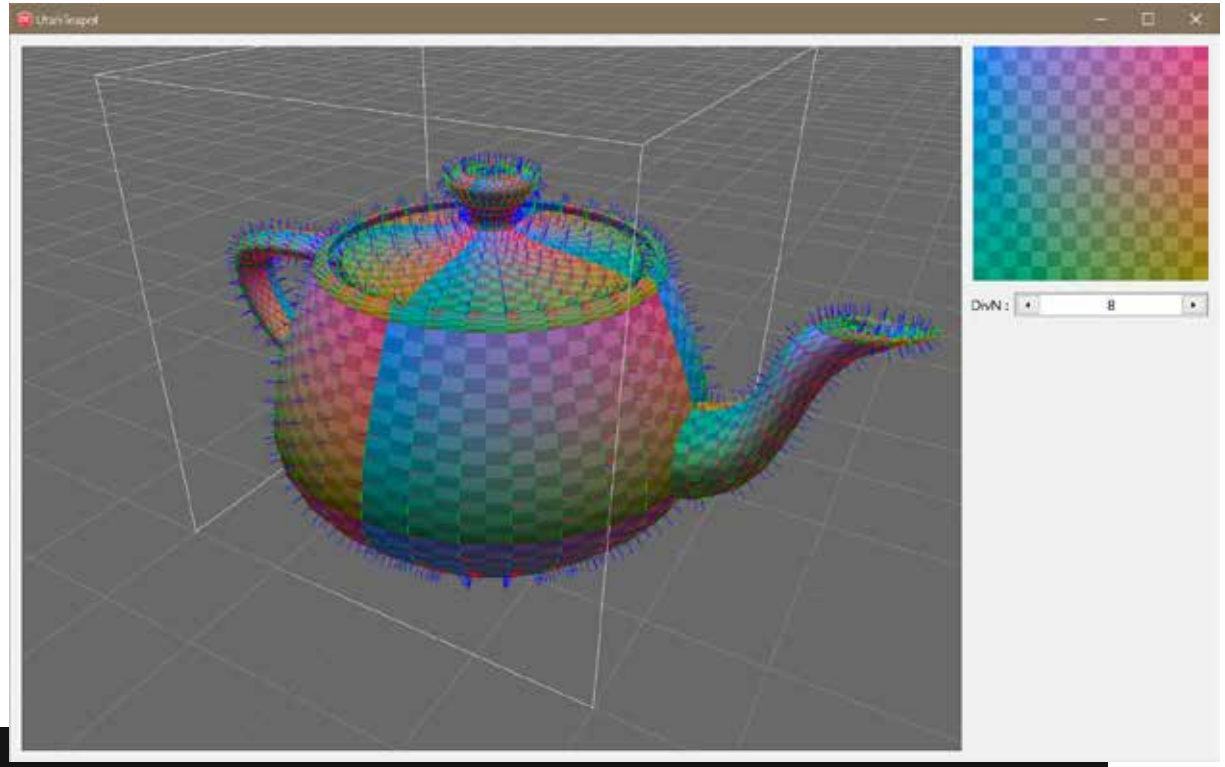
© en.wikipedia.org/wiki/Bézier_surface



UtahTeapot .陽関数.モデリング

§ ベジェ曲面を実装

github.com/LUXOPHIA
/UtahTeapot



電子情報通信学会100年史

5.2 コンピュータグラフィックス

藤 代 一 成（慶大）

コンピュータグラフィックス（CG）技術[1),2)]の源流は，1963 年に米国 MIT のサザランド（Sutherland）によって発表された SketchPad とされている．人間が紙の上にイラストや絵を自由に描くのと同様に，コンピュータにもディスプレイモニタ上に図形を描かせることにより，利用者と[illegible]で知的な挑戦から CG は誕生した．その後の半世紀を超える研究開発の結果，現代の CG は，各種工業製品の形状設計や映像作品/ビデオゲームの制作だけでなく，バーチャルリアリティやデータの可視化等の要素技術としても幅広く利用されている．

本節ではその進展をシステムの観点からまとめた後，モデリング，レンダリング，アニメーションという主要構成技術ごとに概説する．**図 2.88** は，CG 分野でよく知られた手本形状 Utah ティーポットを 4 種類の異なる CG 関連技法によって表現したものである．

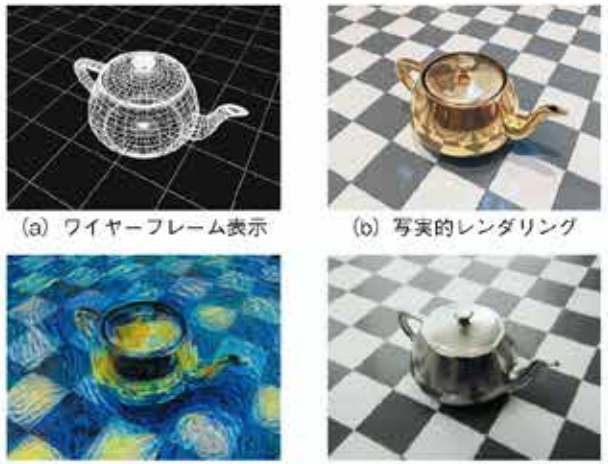
5.2.1 シ ス テ ム

SketchPad 誕生当時の CRT ディスプレイは，ランダムスキャン方式であり，一筆書きの線画しか描けなかった（ベクタグラフィックス，図 2.88(a)）．その後，ラスタスキャン方式が登場して図形内部が塗り潰せるようになり，各種の技法を用いて写実性を増したラスタグラフィックス（図 2.88(b)）に移行し，現在に至る．

ディジタル画像は，フレームバッファに一時的に記憶され，瞬間的にその全内容が走査・表示される．リフレッシュレート（1/30～1/120 秒）以内で後続の画像が準備できれば，仮現運動の原理に従って時間変化するコンテンツは連続して見え，アニメーションが成立する．

表示画像の品質は，解像度（画素密度）と色数（階調数）で決まり，機器や用途により様々な規格が知られている．画素数では 4 K（3,840×2,160 個）やその 4 倍の 8 K といった規格まで登場してきた一方，色数に関しても 24 ビット（約 1,678 万色）のフルカラーを超えて，ダイナミックレンジの広い画像が表示できる HDR [illegible]

両眼視差の原理を利用して立体視を可能にするヘッドマウントディスプレイは，既に 1990 年代に登場していた（アイデア自体は 1960 年代のサザランドの発明に遡る）が，2015 年を境に安価で高性能な市販品が次々と登場し始め，没入的コンテンツ普及への大きな刺激と



(a) ワイヤーフレーム表示　(b) 写実的レンダリング

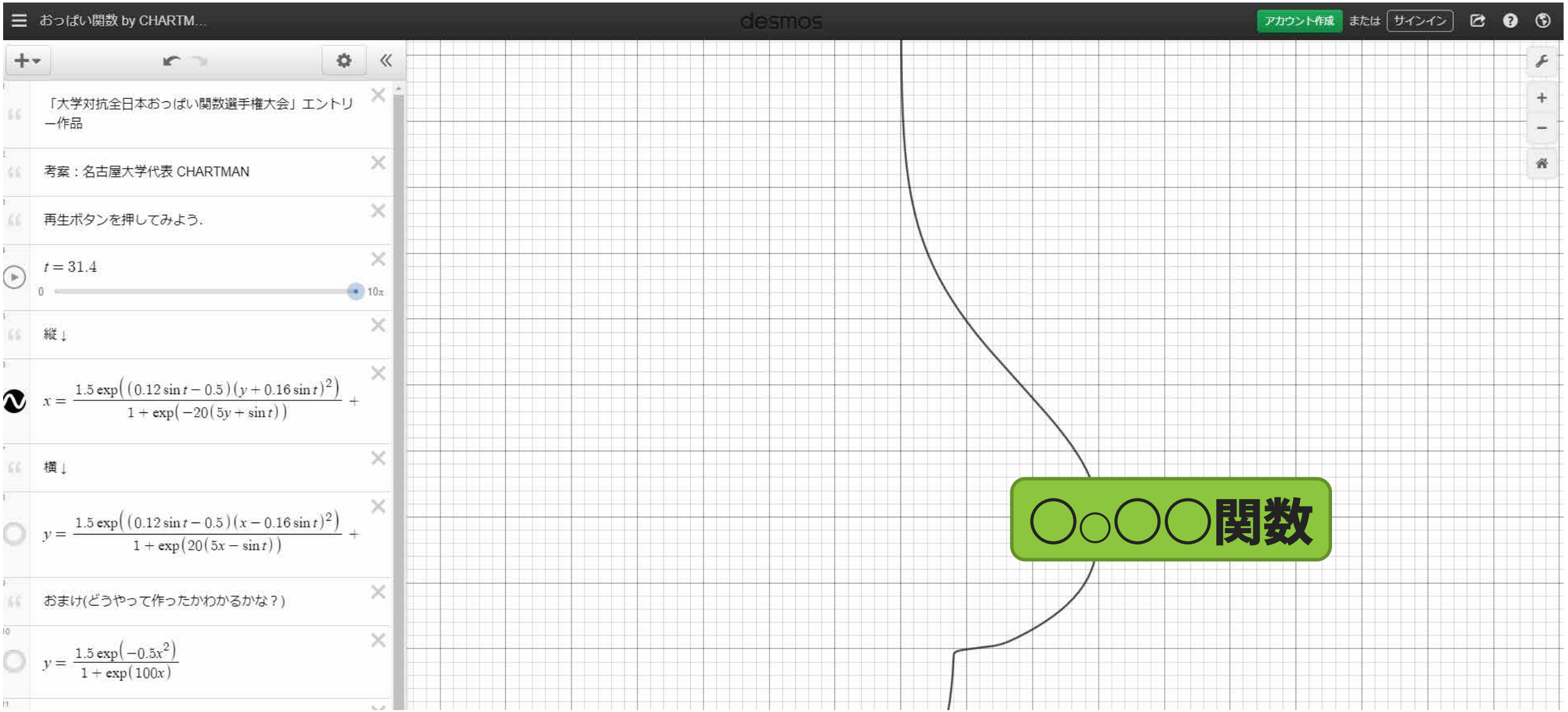
(c) 非写実的レンダリング　(d) 三次元実体化

図 2.88 Utah ティーポット　((a)-(c)：©2016 中山雅紀，慶應義塾大学　(d)：©1991 斎藤隆文，高橋時市郎，NTT Seafood Division)

Utah Teapot @ 電子情報通信学会100年史

youtu.be/CRvwtUCjRto

森羅万象を数式で表そうとする賢者

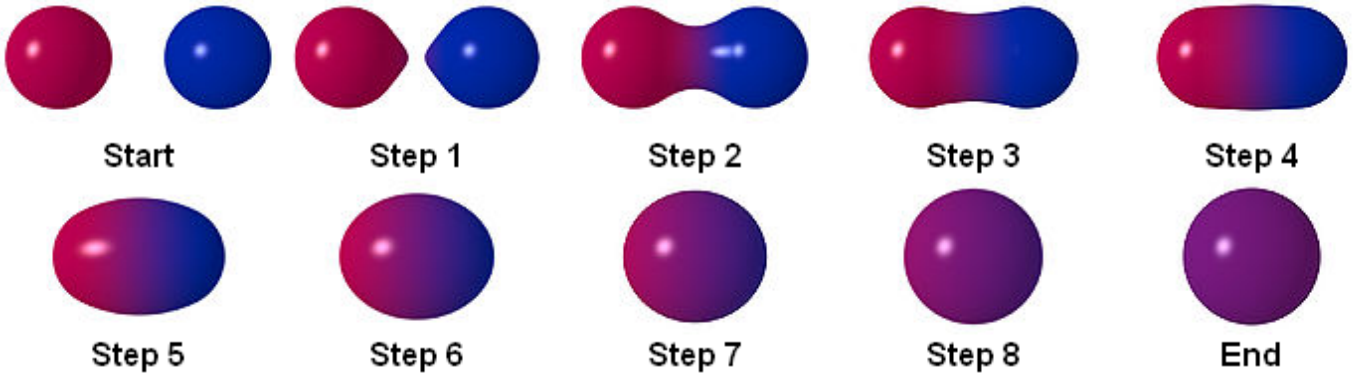


$$\sin x \cos y + \sin y \cos z + \sin z \cos x = 0$$

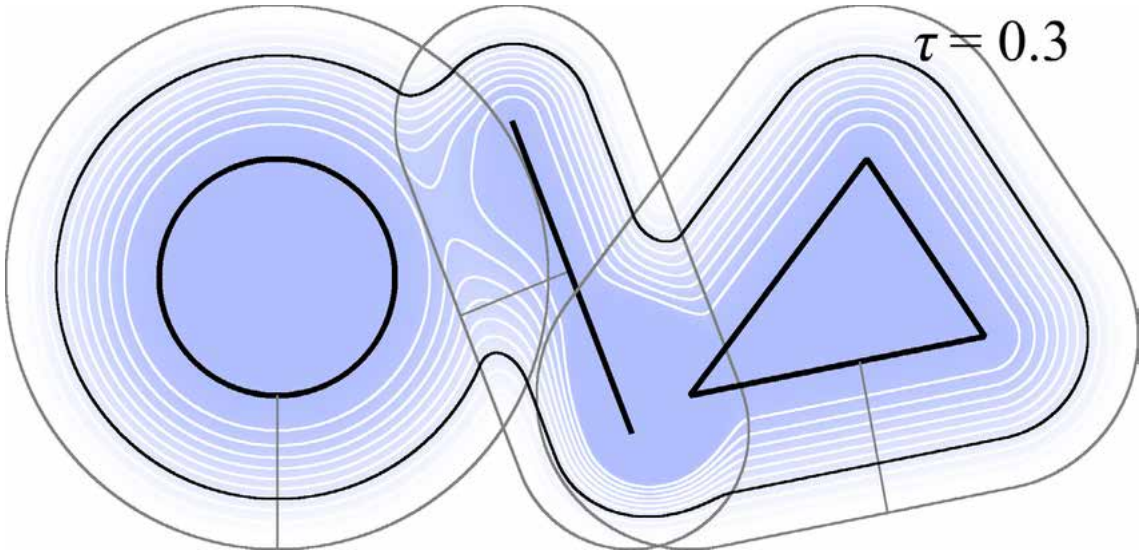
陰関数 . モデリング

- 陰関数表現：f(x, y, z) = 0
  - 等値面（Isosurface）が表面
  - 表面座標は方程式を解くまで分からない
  - 計算コスト：大
- 設計自由度の向上
  - 陰関数をツギハギ
    - メタボール

© www.bathsheba.com

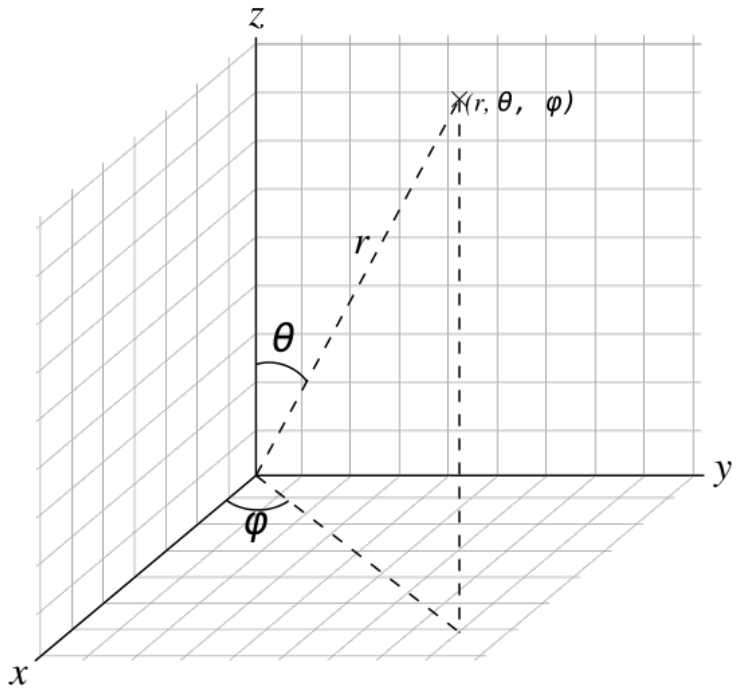


© en.wikipedia.org/wiki/Metaballs

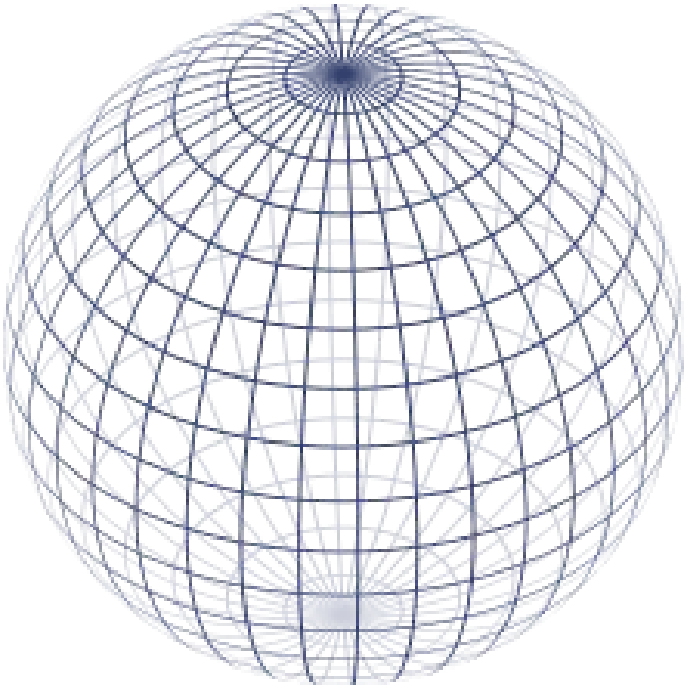


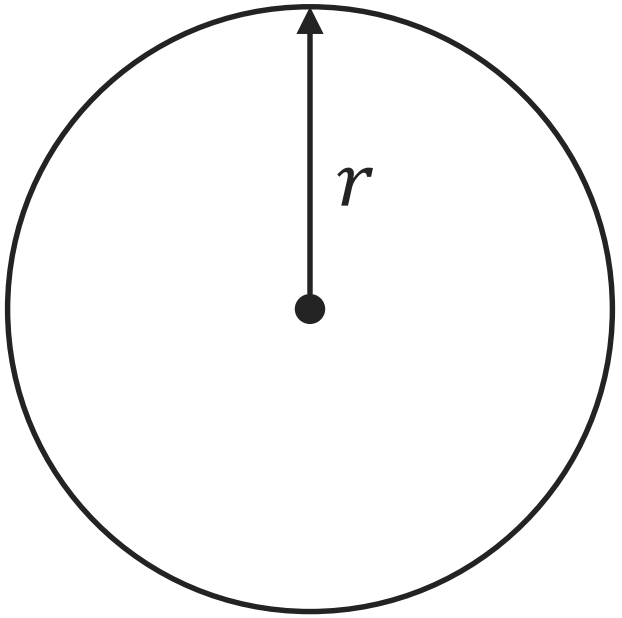
© Level-of-detail visualization of clustered graph layouts

陽関数 vs 陰関数 . モデリング



$$\begin{cases} x = r\sin\theta\ \cos\phi \\ y = r\sin\theta\ \sin\phi \\ z = r\cos\theta \end{cases}$$

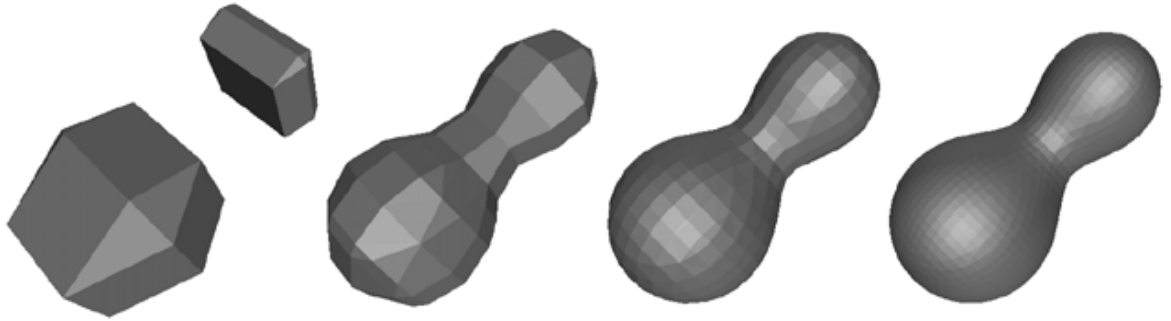




$$x^2 + y^2 + z^2 - r^2 = 0$$

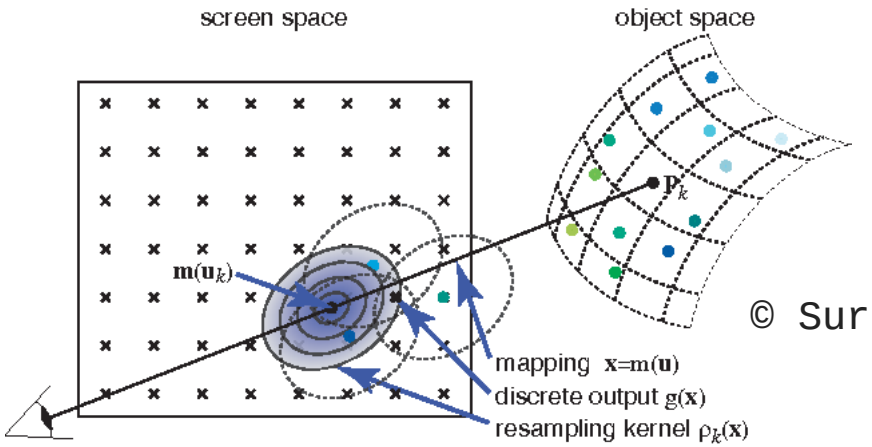
レンダリング .陰関数.モデリング

© paulbourke.net/geometry/polygonise



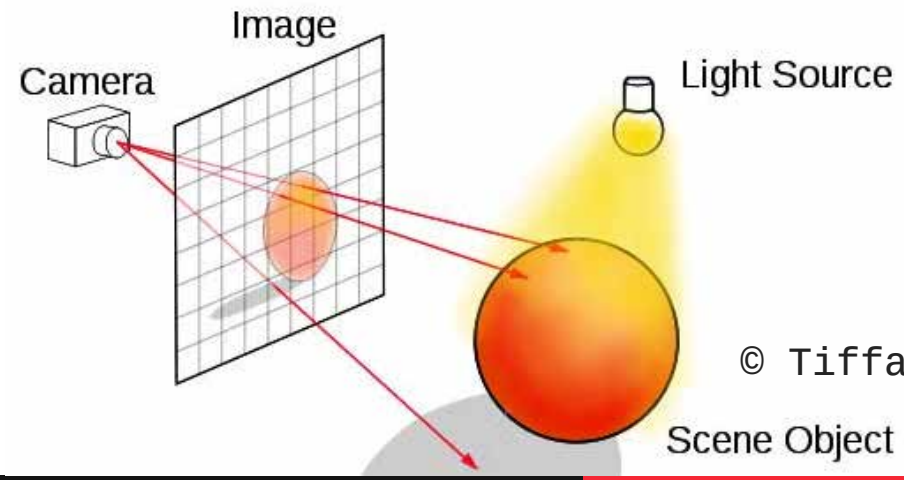
§ ポリゴン化
- 等値面にポリゴンを貼る
- 離散化曲面

§ ポイントクラウド化
- 等値面上に点群を発生させる
  - Splatting 法
- 離散化曲面



© Surface splatting

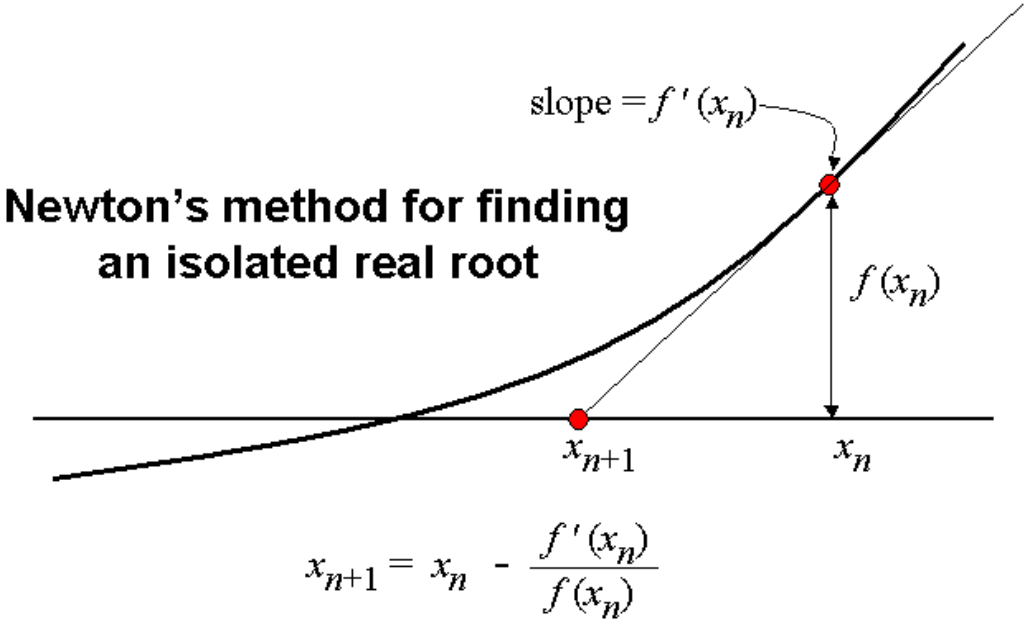
§ レイトレーシング
- 光線と陰関数の交点を求める
- 連続曲面



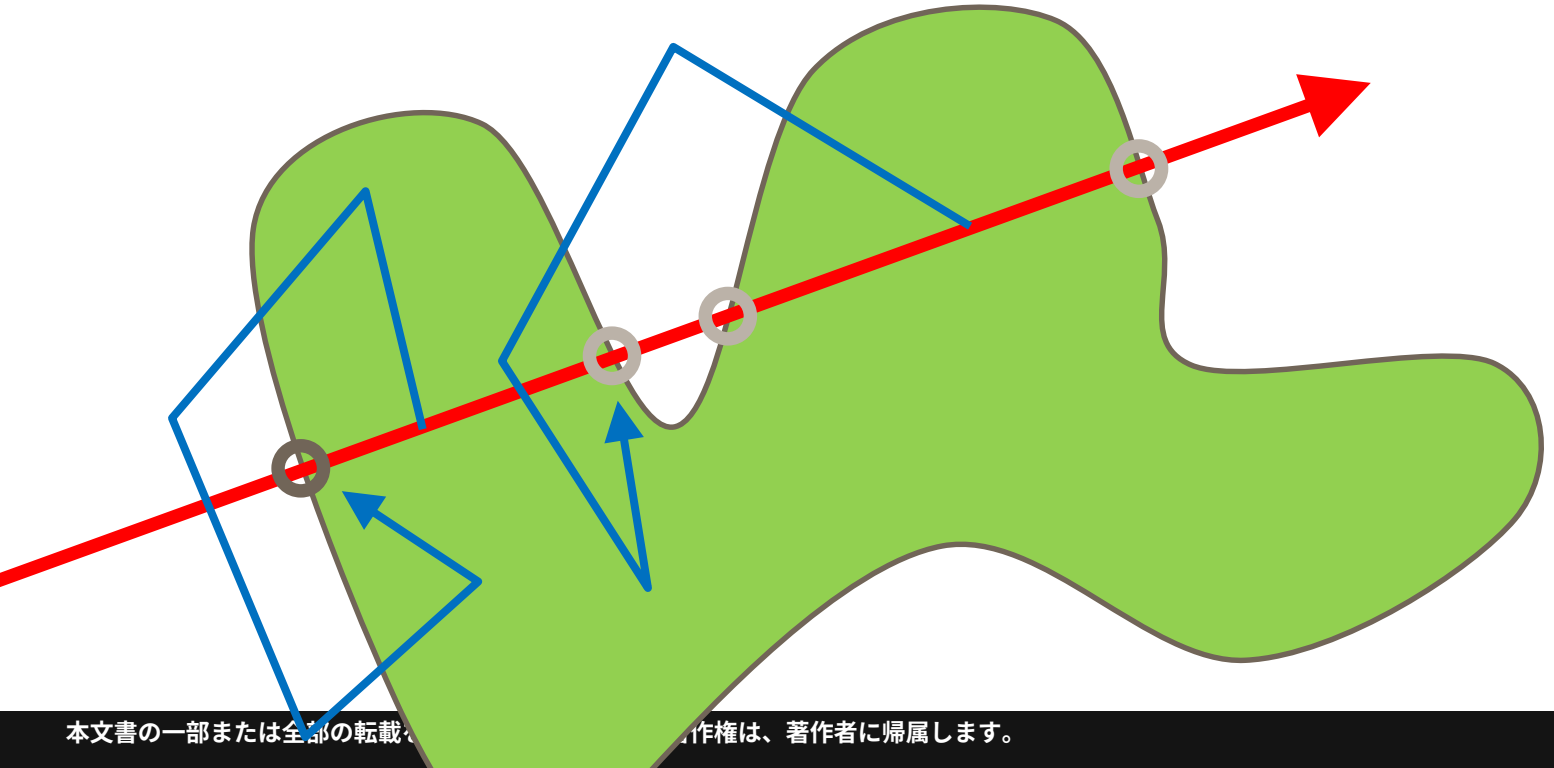
© Tiffany C. Inglis

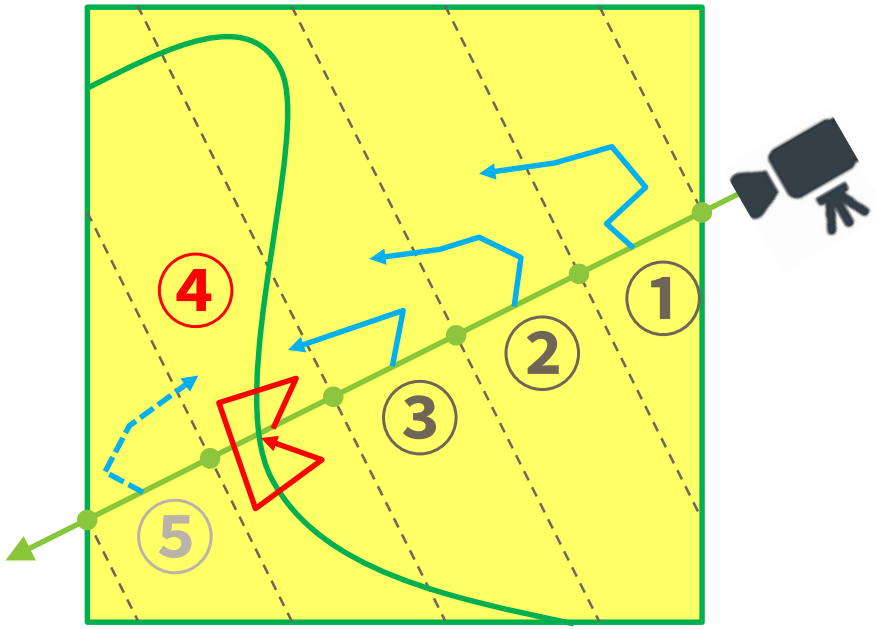
交差判定 .レンダリング.陰関数.モデリング

- 陰関数と視線との交点は複数
  - ニュートン法は収束する解を選べない
  - 始点に最も近い解を探すのが難しい
- 始点から少しずつ伸ばして探索

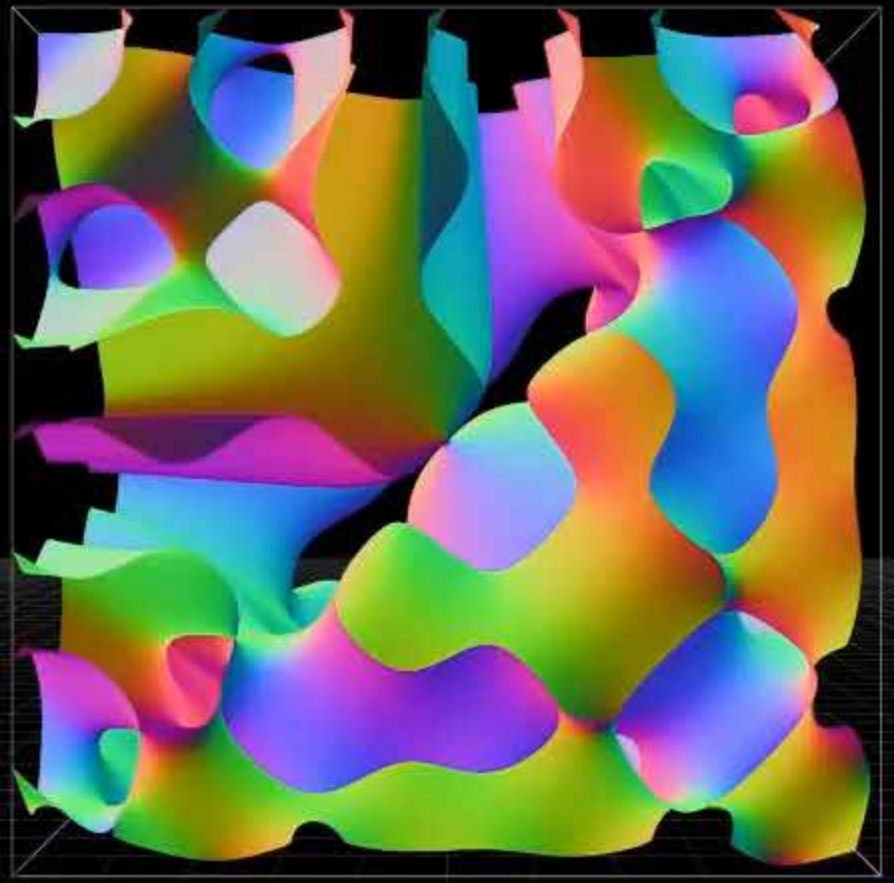


$$x_{n+1} = x_n - \frac{f'(x_n)}{f(x_n)}$$





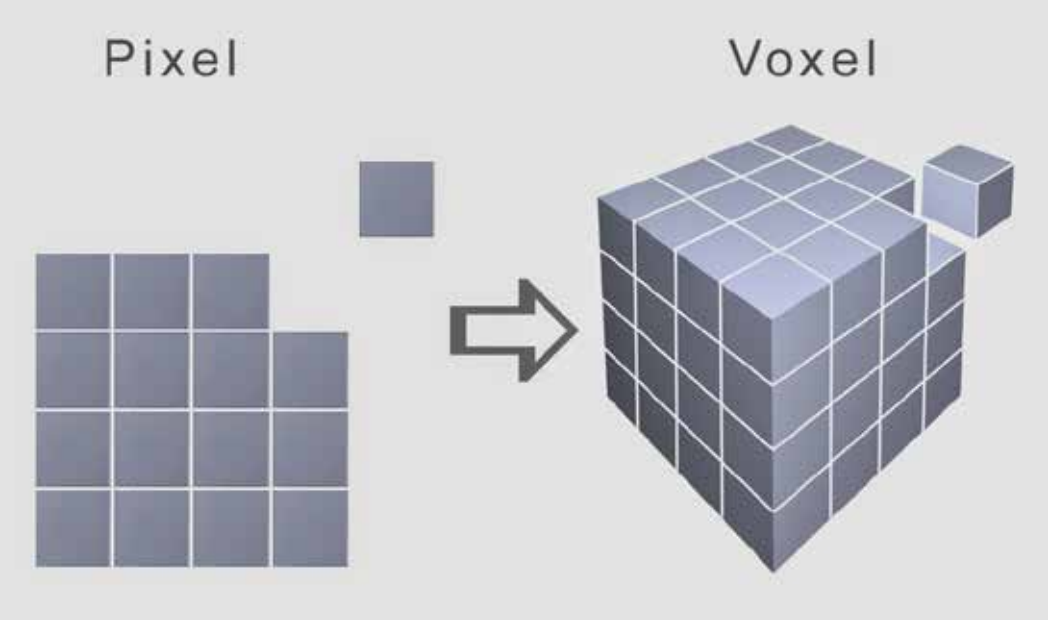
github.com/LUXOPHIA/ImplicitRender



Real-Time Implicit Rendering @ Delphi

ブロック .モデリング

- ３Ｄのドット絵
- ゲーム「Minecraft」で有名
- ボクセルが**二値**を保持
- Undo が容易



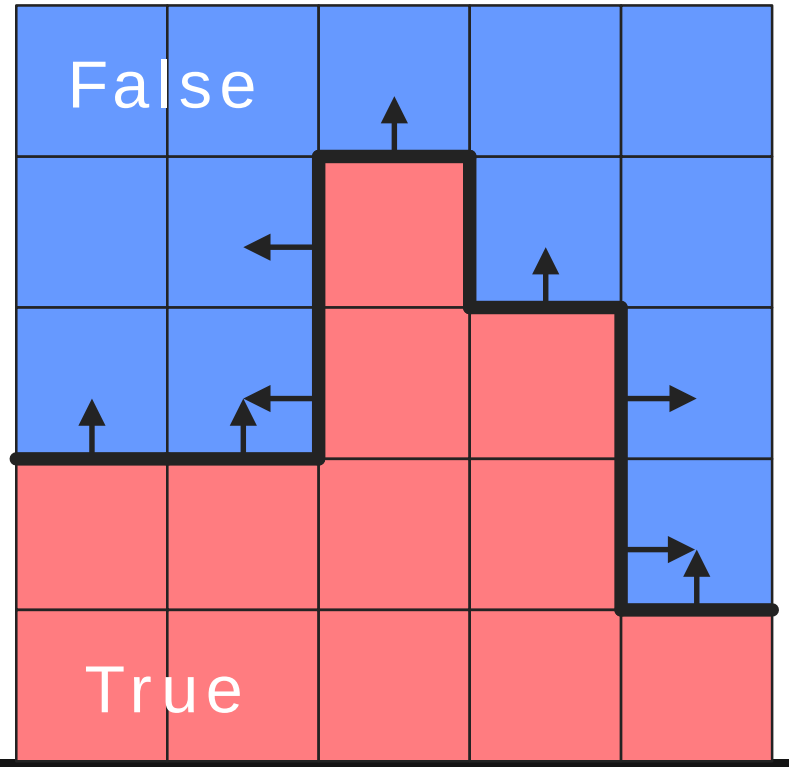
© voxelart.blogspot.jp/2014/04/que-es-voxel-y-el-voxel-art.html

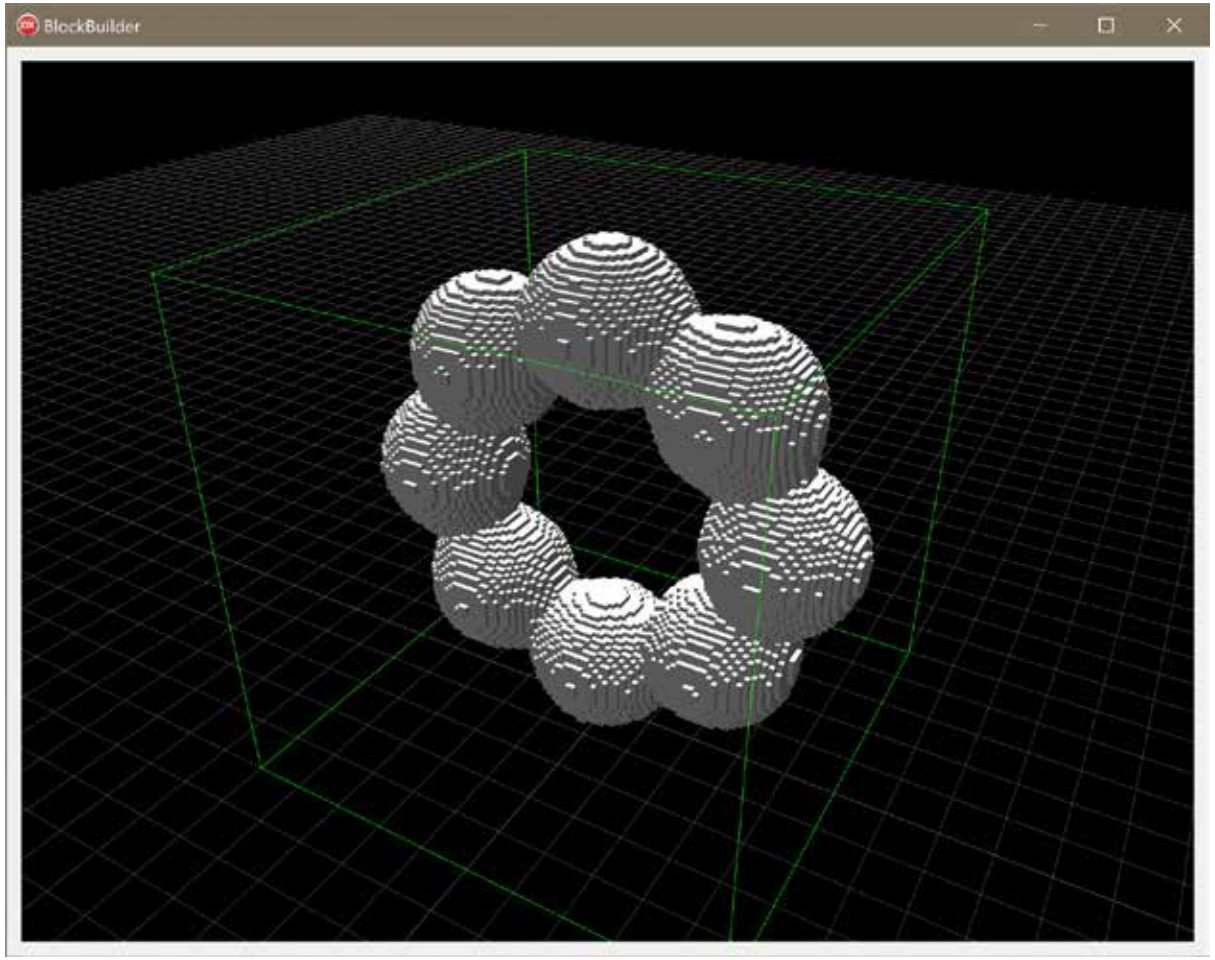
© Super Mario Mash-Up Pack @ Mojang

実装 .ブロック.モデリング

github.com/LUXOPHIA
/BlockBuilder

- ３次元配列内に**存在**フラグを立てて形状を定義
- フラグの境界にポリゴンを貼る
- 法線は False の方を向く





ボクセル配列 .ブロック.モデリング

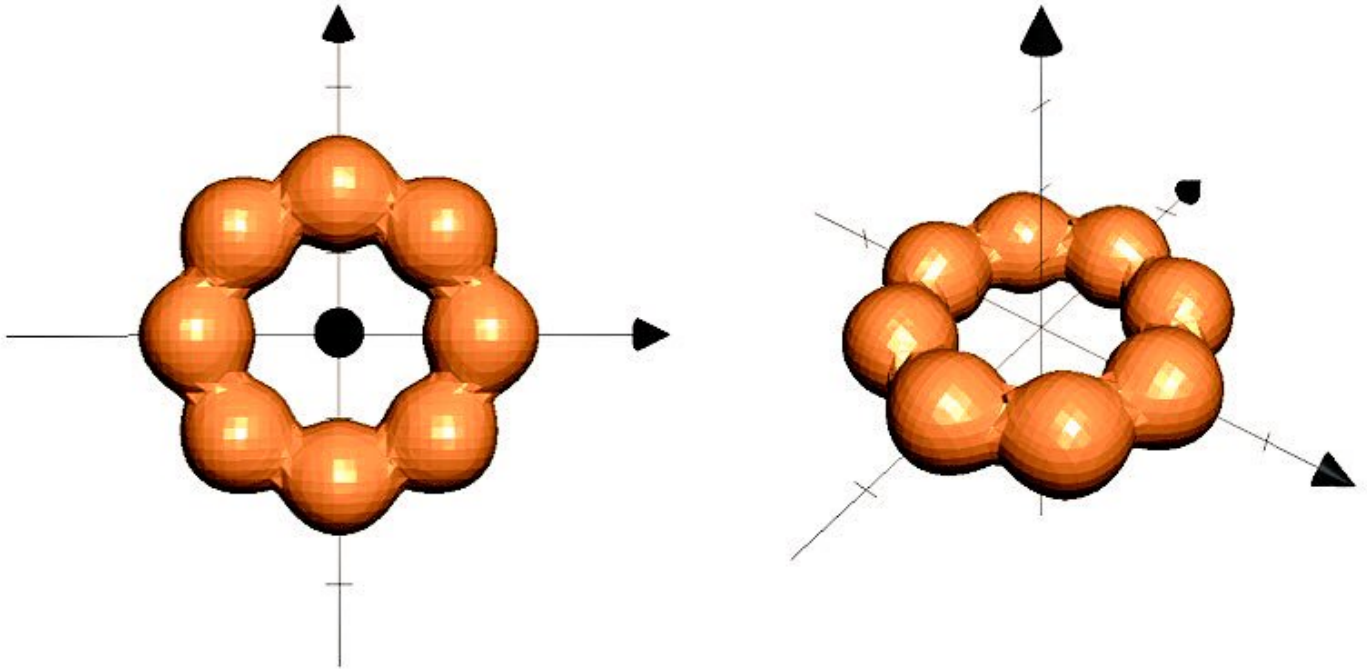
§ ボクセル値は Boolean

```
function Pãodering( const P_:TPoint3D ) :Single;
var
   X2, Y2, Z2, A :Single;
begin
     X2 := Sqr( P_.X );
     Y2 := Sqr( P_.Y );
     Z2 := Sqr( P_.Z );

     A := Abs( Sqr( ( X2 - Y2 ) / ( X2 + Y2 ) ) - 0.5 );

     Result := Sqr( Sqrt( X2 + Y2 ) - 8 - A ) + Z2 - Sqr( 2 + 3 * A );
end;
```

```
procedure TForm1.MakeBlocks;
var
   X, Y, Z :Integer;
   P :TPoint3D;
begin
     with _Blocks do
     begin
          BeginUpdate;

          for Z := 0 to BricsZ-1 do
          begin
               P.Z := 24 * ( ( Z + 0.5 ) / BricsZ - 0.5 );

               for Y := 0 to BricsY-1 do
               begin
                    P.Y := 24 * ( ( Y + 0.5 ) / BricsY - 0.5 );

                    for X := 0 to BricsX-1 do
                    begin
                         P.X := 24 * ( ( X + 0.5 ) / BricsX - 0.5 );

                         Brics[ X, Y, Z ] := ( Pãodering( P ) < 0 );
                    end;
               end;
          end;

          EndUpdate;
     end;
end;
```

ポン・デ・リング関数

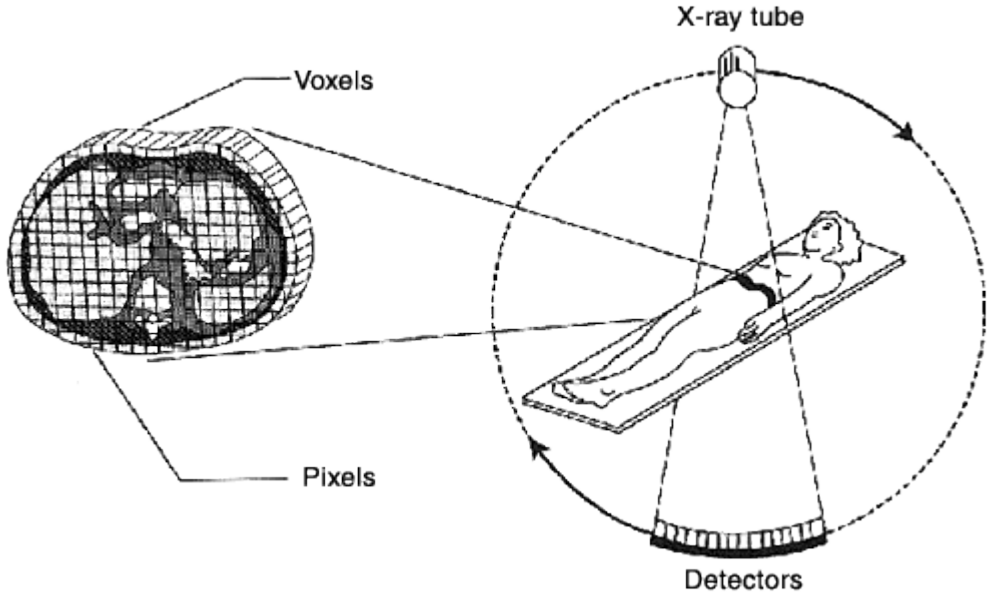
単一の数式によるポン・デ・リング（3D）



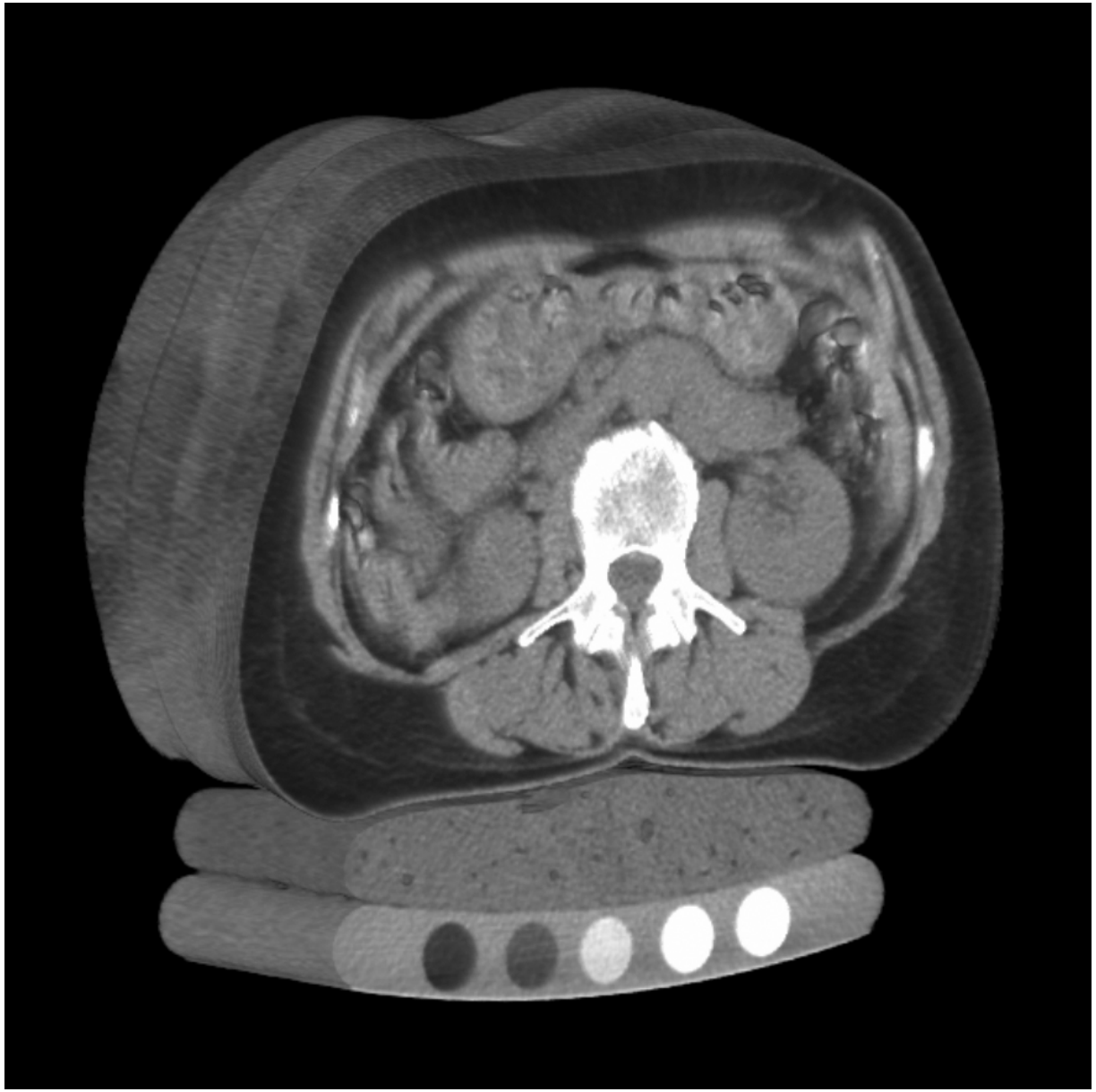
$$\left(\sqrt{x^2+y^2}-8-\left|\left(\frac{x^2-y^2}{x^2+y^2}\right)^2-\frac{1}{2}\right|\right)^2+z^2=\left(2+3\left|\left(\frac{x^2-y^2}{x^2+y^2}\right)^2-\frac{1}{2}\right|\right)^2$$

ボリューム .モデリング

- ３次元画像
- 医療用３ＤＣＴなどで有名
- ボクセルが**実数値**を保持
- 離散化陰関数



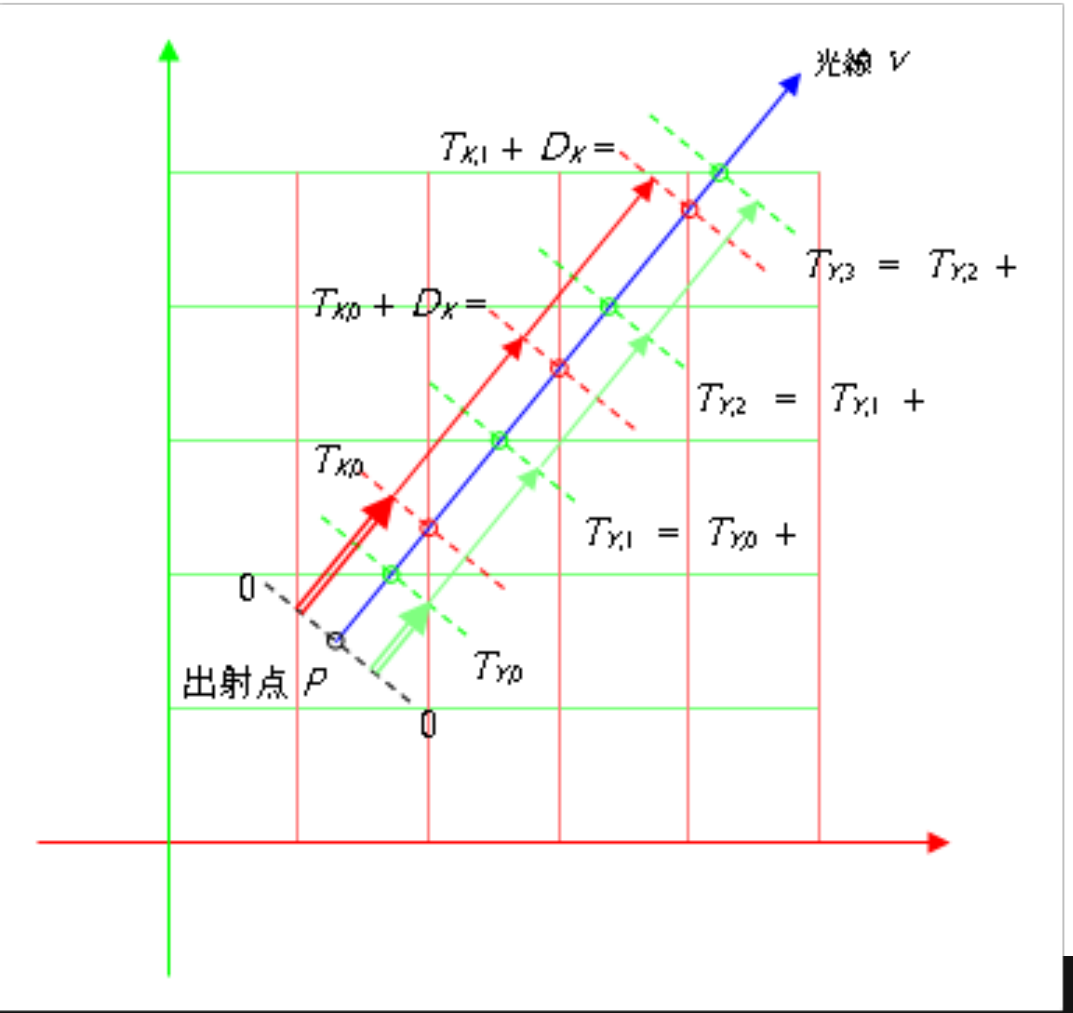
© www.physicscentral.com/explore/action/scans.cfm

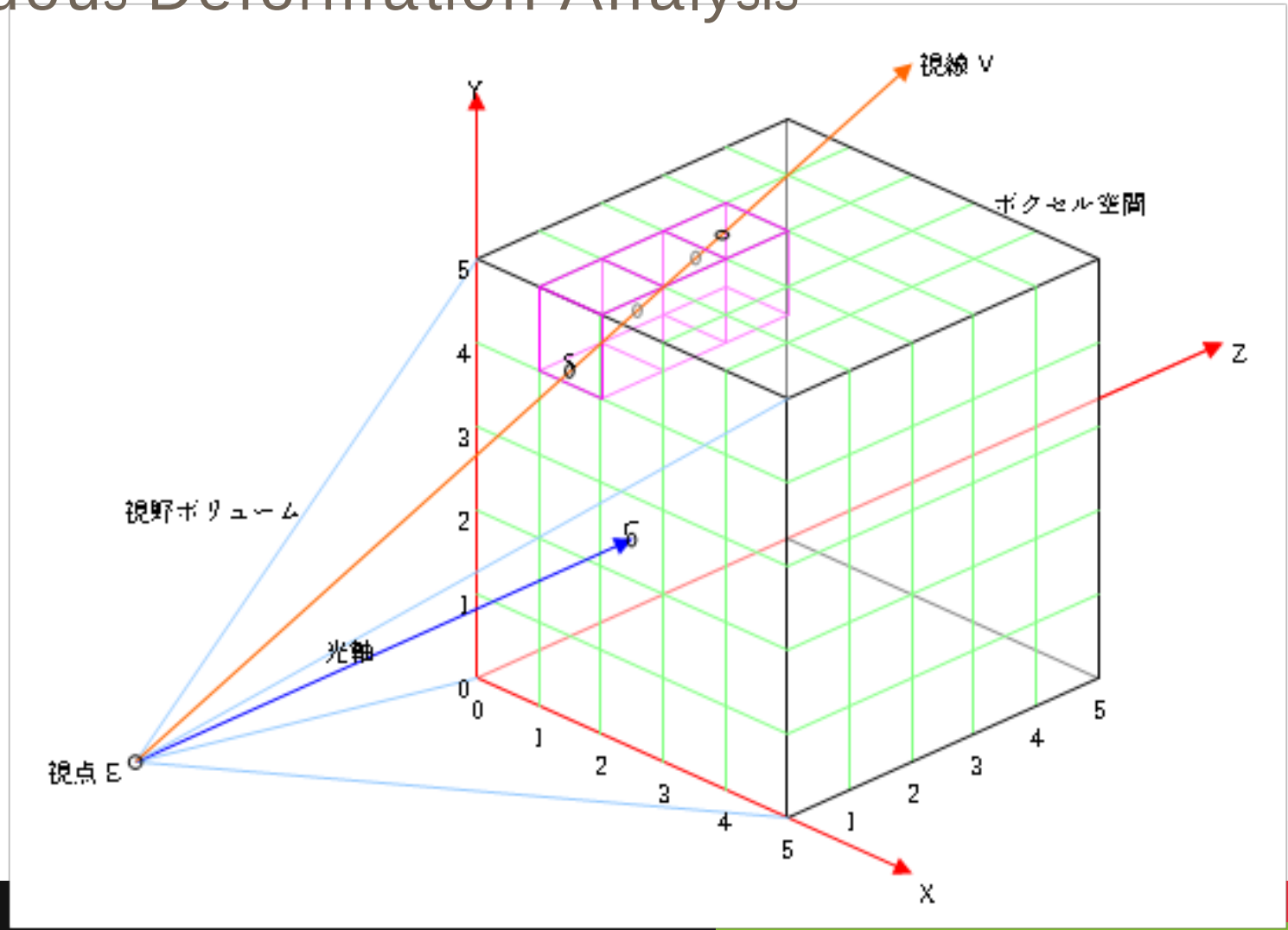


© en.wikipedia.org/wiki/Volume_rendering

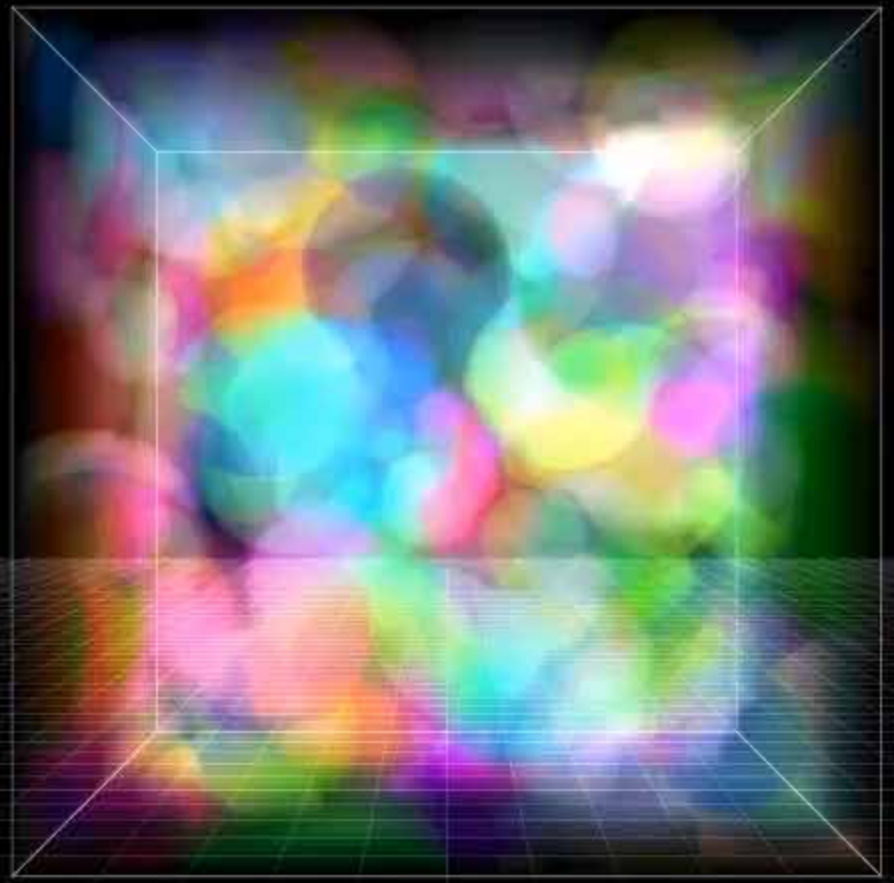
レイトレーシング .ボリューム.モデリング

§ 3D-DDA：3D Discontinuous Deformation Analysis





github.com/LUXOPHIA/VolumeRender

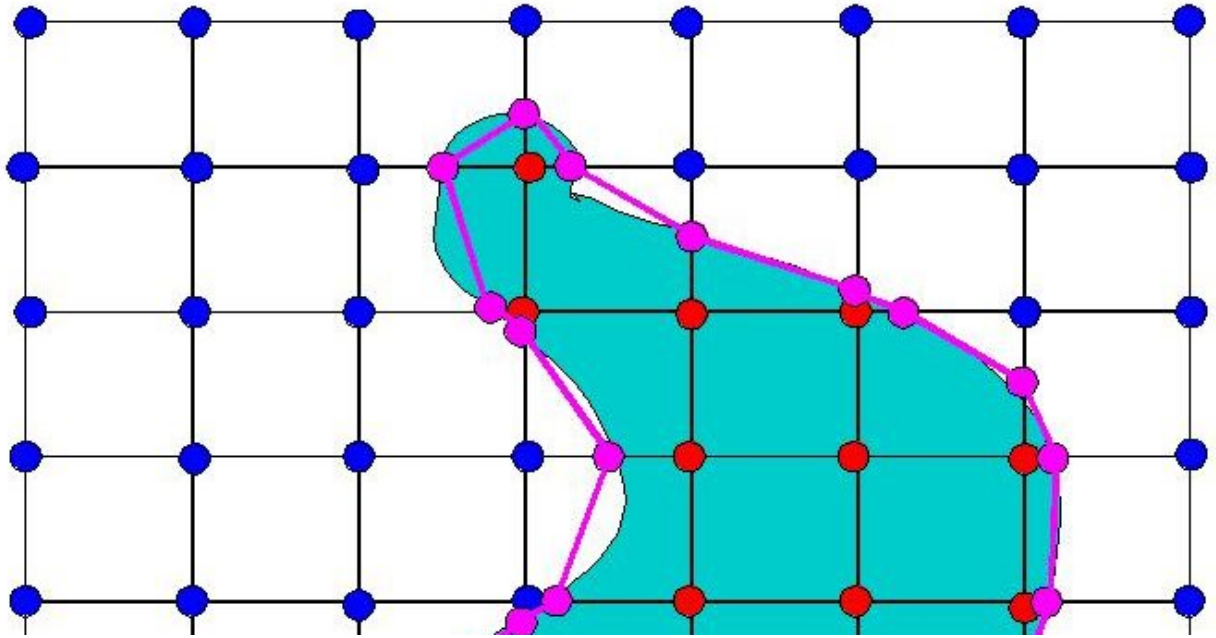


Real-Time Volume Rendering @ Delphi

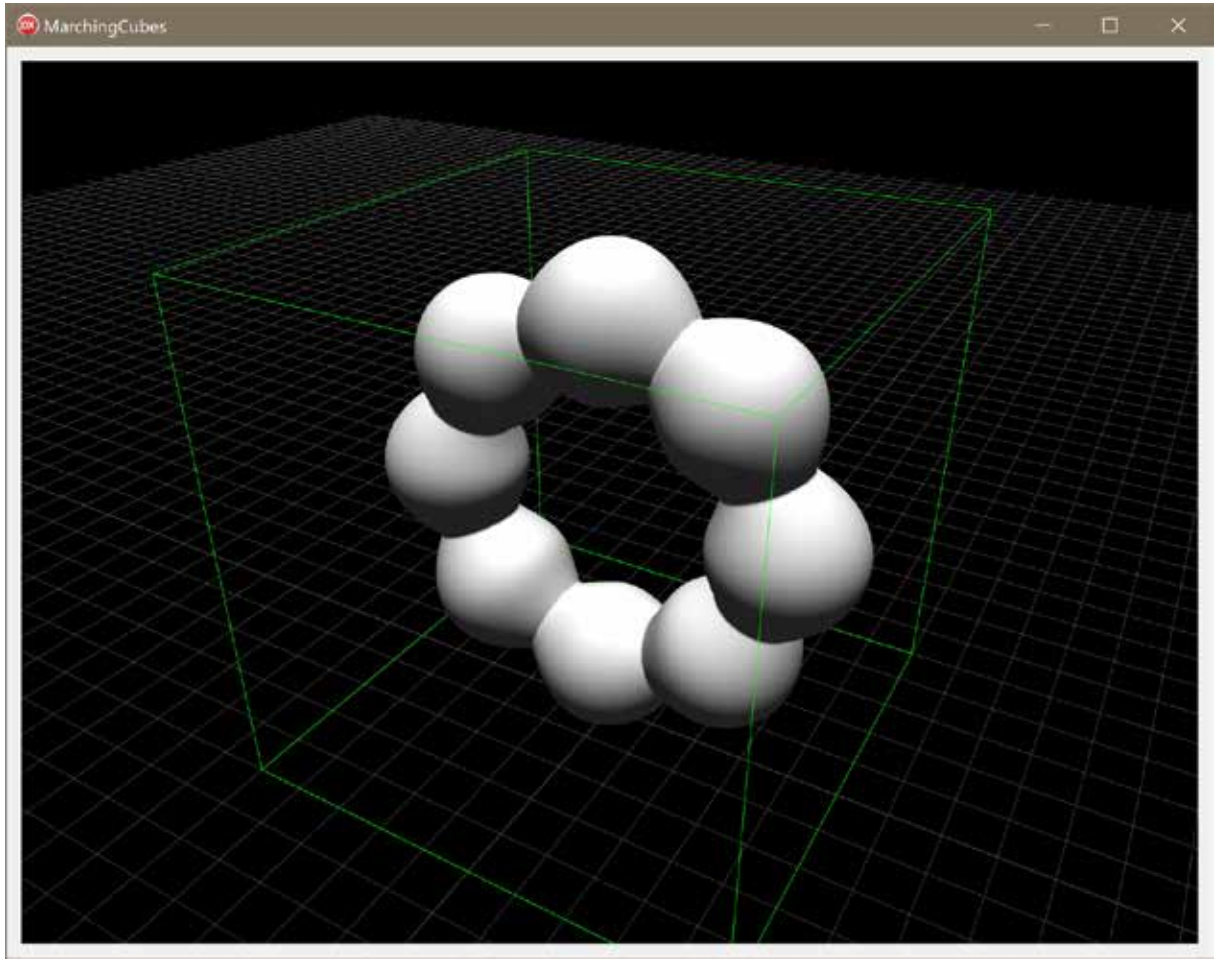
実装 .ボリューム .モデリング

github.com/LUXOPHIA/MarchingCubes

- ３次元配列内に**存在率**を指定して形状を定義
- ボクセルが**実数値**を保持
- 等値面にポリゴンを貼る
  - **マーチングキューブ法**

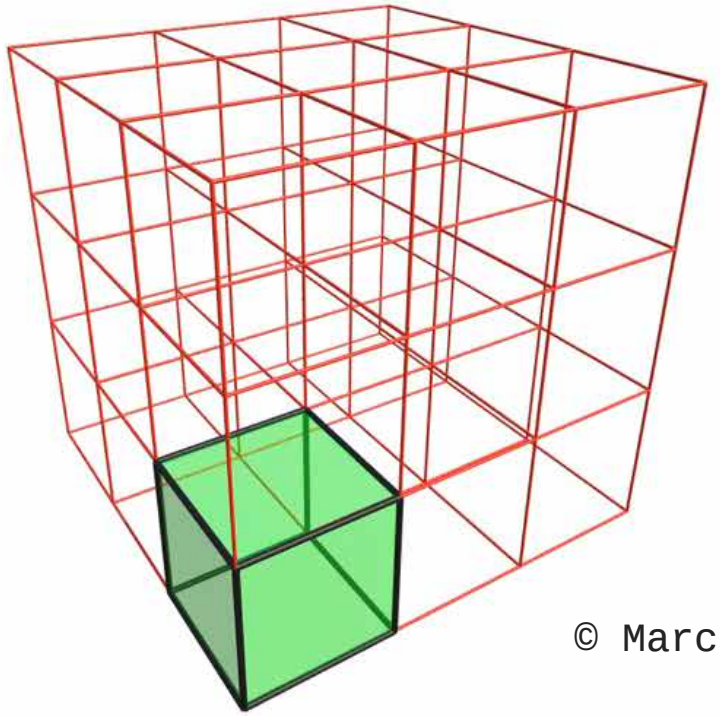


© www.cs.carleton.edu/cs_comps/0405/shape/



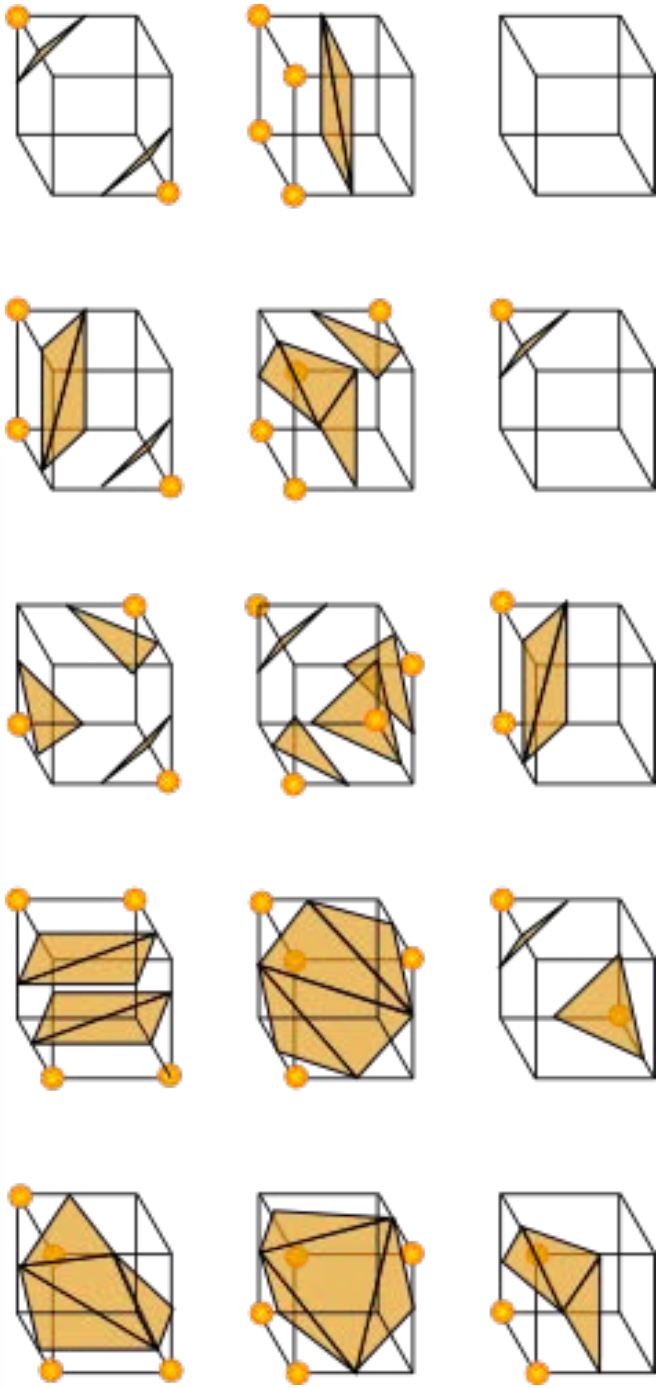
マーチングキューブ法 .ボリューム.モデリング

- ボクセルの角の８頂点は
  等値面の中か外かのどちらかに属する
- ８頂点の内外の組合わせは 256通り ある
- ポリゴン構造としては 15種類 に分けられる



youtu.be/B_xk71YopsA

© Marching Cubes Animation



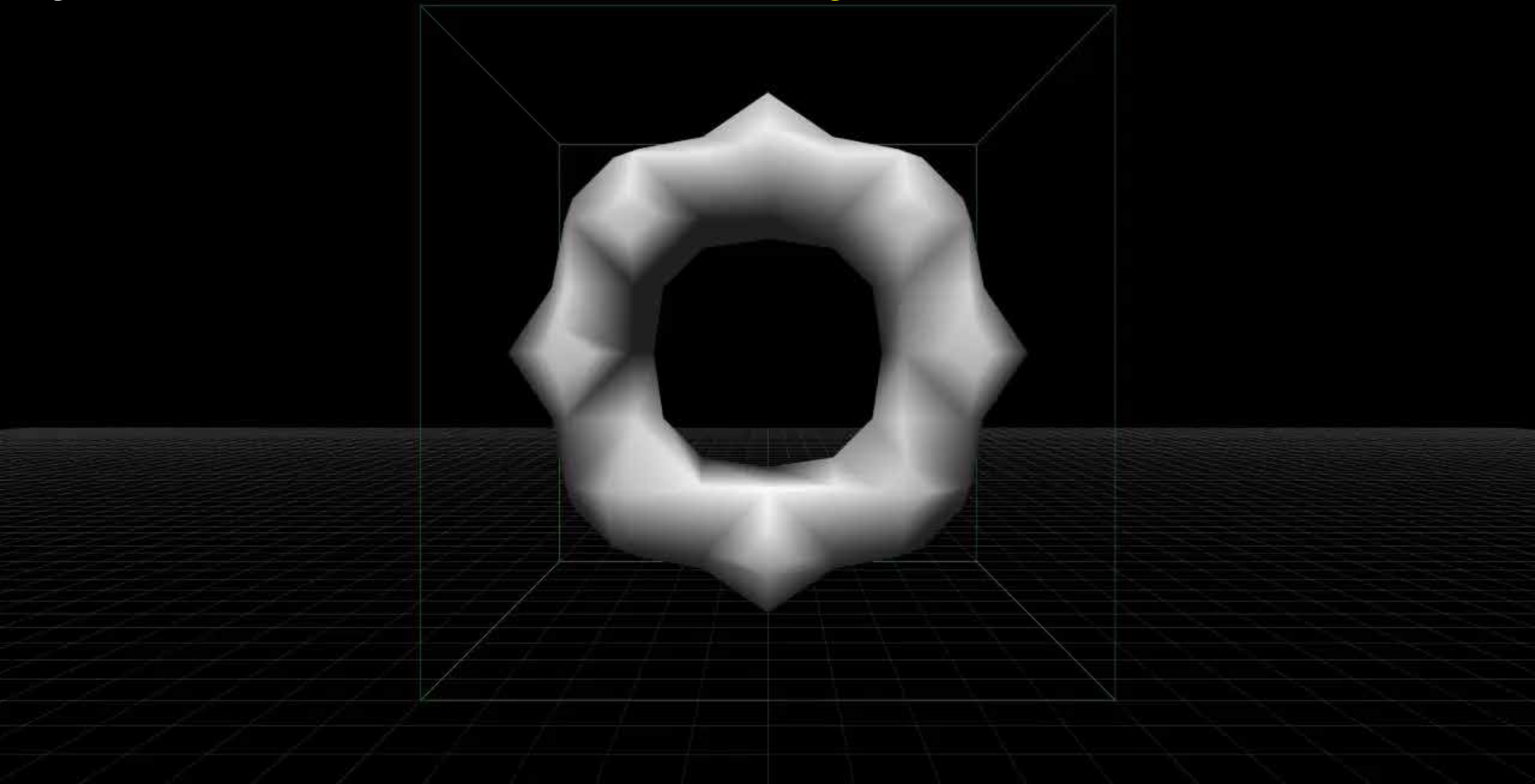
ポリゴンテーブル .マーチングキューブ法.ボリューム.モデリング

§ 最大５枚 / ボクセル

▼LUX.Marcubes.pas

```
initialization //$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ 初期化

     TRIAsTABLE := [
          [                                                             ],
          [ [ 00, 04, 08 ]                                              ],
          [ [ 00, 09, 06 ]                                              ],
          [ [ 09, 04, 08 ], [ 06, 04, 09 ]                              ],
          [ [ 01, 10, 04 ]                                              ],
          [ [ 01, 08, 00 ], [ 10, 08, 01 ]                              ],
          [ [ 00, 09, 06 ], [ 04, 01, 10 ]                              ],
          [ [ 01, 09, 06 ], [ 01, 10, 09 ], [ 10, 08, 09 ]              ],
          [ [ 06, 11, 01 ]                                              ],
          [ [ 06, 11, 01 ], [ 00, 04, 08 ]                              ],
          [ [ 00, 11, 01 ], [ 09, 11, 00 ]                              ],
          [ [ 04, 11, 01 ], [ 04, 08, 11 ], [ 08, 09, 11 ]              ],
          [ [ 06, 10, 04 ], [ 11, 10, 06 ]                              ],
          [ [ 06, 08, 00 ], [ 06, 11, 08 ], [ 11, 10, 08 ]              ],
          [ [ 00, 10, 04 ], [ 00, 09, 10 ], [ 09, 11, 10 ]              ],
          [ [ 09, 11, 08 ], [ 08, 11, 10 ]                              ],
          [ [ 08, 05, 02 ]                                              ],
          [ [ 00, 05, 02 ], [ 04, 05, 00 ]                              ],
          [ [ 09, 06, 00 ], [ 02, 08, 05 ]                              ],
          [ [ 09, 05, 02 ], [ 09, 06, 05 ], [ 06, 04, 05 ]              ],
          [ [ 04, 01, 10 ], [ 08, 05, 02 ]                              ],
          [ [ 05, 01, 10 ], [ 05, 02, 01 ], [ 02, 00, 01 ]              ],
          [ [ 06, 00, 09 ], [ 04, 01, 10 ], [ 02, 08, 05 ]              ],
          [ [ 01, 10, 05 ], [ 06, 01, 05 ], [ 06, 05, 02 ], [ 06, 02, 09 ] ],
```

github.com/LUXOPHIA/MarchingCubes



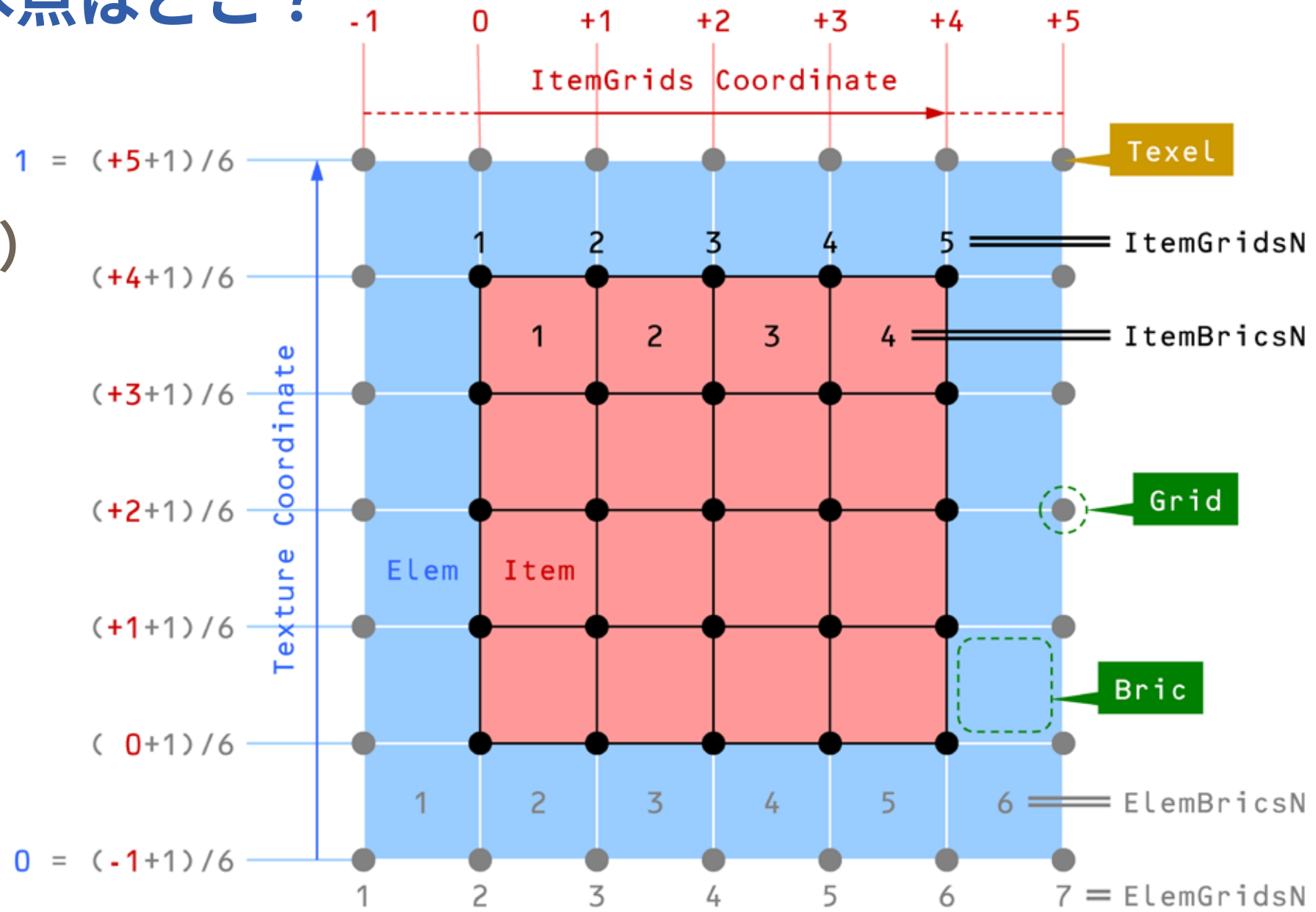
ボクセル配列 .ボリューム .モデリング

§ ボクセル値は Single

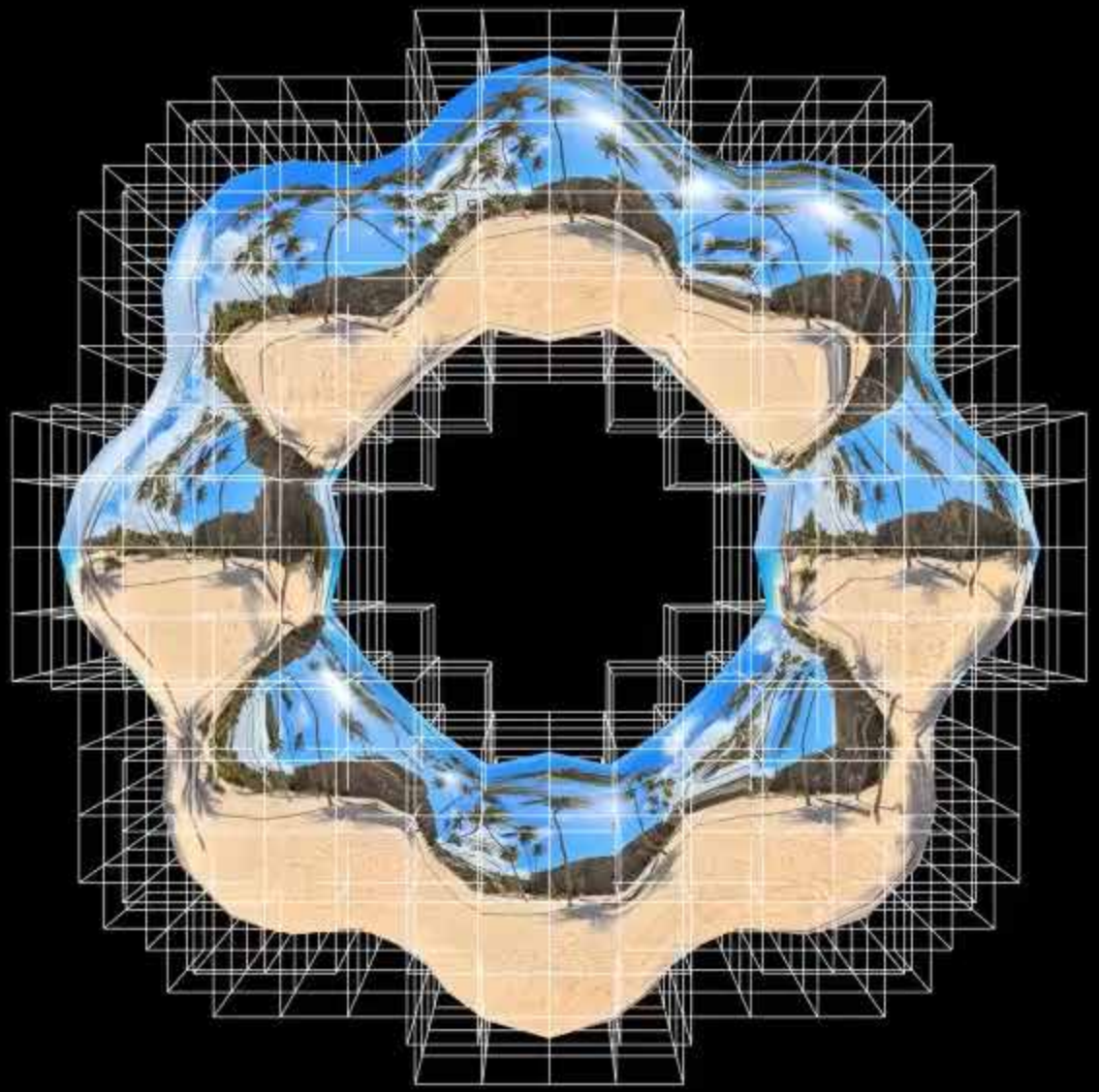
```
procedure TForm1.MakeVoxels;
var
   X, Y, Z :Integer;
   P :TPoint3D;
begin
     with _Marcubes do
     begin
          with Grids do
          begin
               for Z := -1 to GridsZ do
               begin
                    P.Z := 24 * ( Z / BricsZ - 0.5 );

                    for Y := -1 to GridsY do
                    begin
                         P.Y := 24 * ( Y / BricsY - 0.5 );

                         for X := -1 to GridsX do
                         begin
                              P.X := 24 * ( X / BricsX - 0.5 );

                              Grids[ X, Y, Z ] := Pãodering( P );
                         end;
                    end;
               end;
          end;

          MakeModel;  // ポリゴンモデル生成
     end;
end;
```

ボクセルの標本点はどこ？

- § ピクセル
  - 体中心
- § ボクセル（Bric）
  - 体中心
- § 格子点（Grid）
  - ボクセルの角



github.com/LUXOPHIA/MarchingCubes_GPU



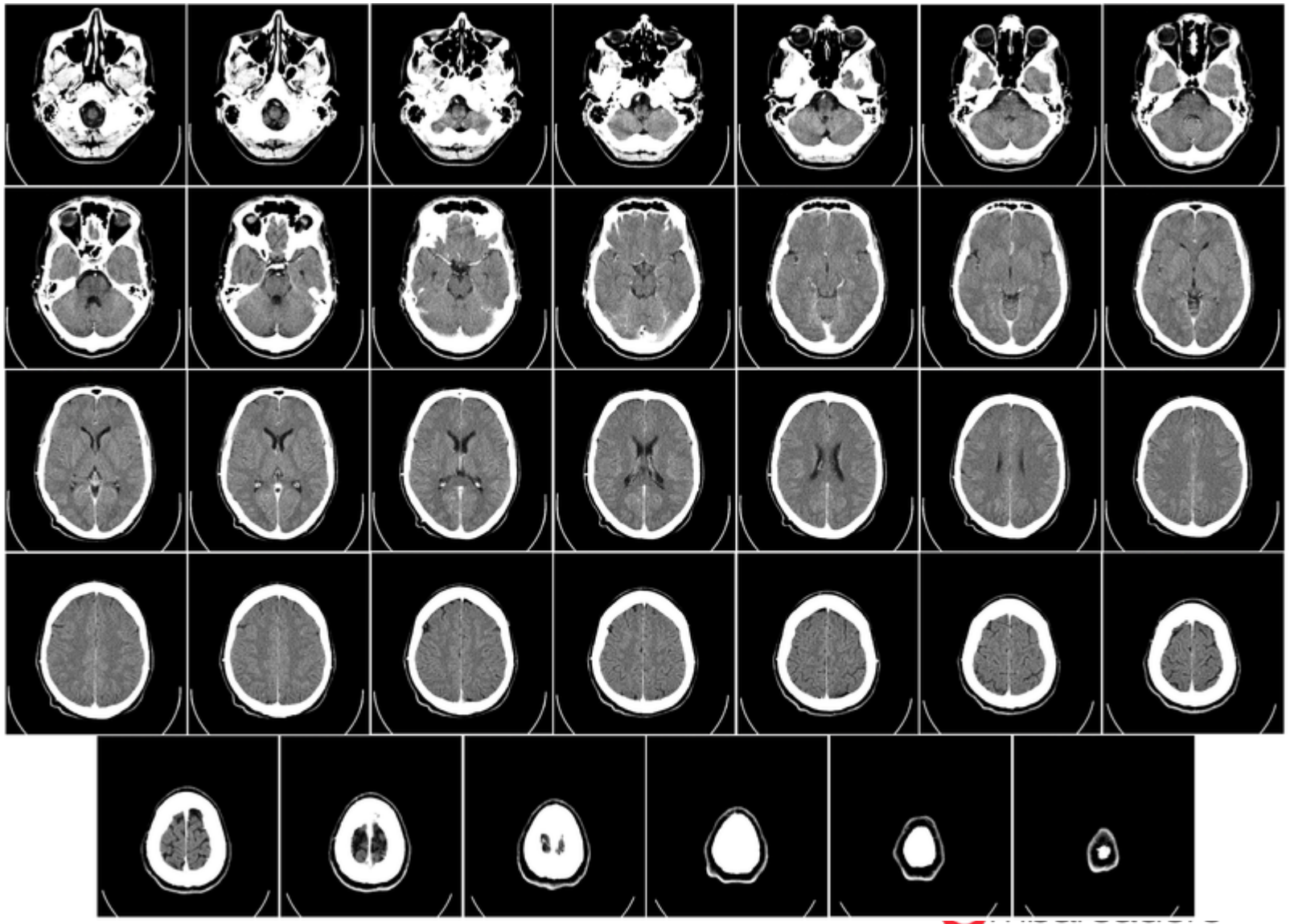
ポリゴンテーブル .マーチングキューブ法.ボリューム.モデリング

§ GLSL 言語は多次元配列や動的配列に未対応

▼LUX.GPU.OpenGL.Shaper.Preset.TMarcubes.Faces_G.glsl

```
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%【定数】

const TTrias TRIASTABLE[ 256 ] = TTrias[ 256 ](
  TTrias( 0, ivec3[ 5 ]( ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ) ) ),
  TTrias( 1, ivec3[ 5 ]( ivec3(  0,  4,  8 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ) ) ),
  TTrias( 1, ivec3[ 5 ]( ivec3(  0,  9,  6 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ) ) ),
  TTrias( 2, ivec3[ 5 ]( ivec3(  9,  4,  8 ), ivec3(  6,  4,  9 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ) ) ),
  TTrias( 1, ivec3[ 5 ]( ivec3(  1, 10,  4 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ) ) ),
  TTrias( 2, ivec3[ 5 ]( ivec3(  1,  8,  0 ), ivec3( 10,  8,  1 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ) ) ),
  TTrias( 2, ivec3[ 5 ]( ivec3(  0,  9,  6 ), ivec3(  4,  1, 10 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ) ) ),
  TTrias( 3, ivec3[ 5 ]( ivec3(  1,  9,  6 ), ivec3(  1, 10,  9 ), ivec3( 10,  8,  9 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ) ) ),
  TTrias( 1, ivec3[ 5 ]( ivec3(  6, 11,  1 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ) ) ),
  TTrias( 2, ivec3[ 5 ]( ivec3(  6, 11,  1 ), ivec3(  0,  4,  8 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ) ) ),
  TTrias( 2, ivec3[ 5 ]( ivec3(  0, 11,  1 ), ivec3(  9, 11,  0 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ) ) ),
  TTrias( 3, ivec3[ 5 ]( ivec3(  4, 11,  1 ), ivec3(  4,  8, 11 ), ivec3(  8,  9, 11 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ) ) ),
  TTrias( 2, ivec3[ 5 ]( ivec3(  6, 10,  4 ), ivec3( 11, 10,  6 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ) ) ),
  TTrias( 3, ivec3[ 5 ]( ivec3(  6,  8,  0 ), ivec3(  6, 11,  8 ), ivec3( 11, 10,  8 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ) ) ),
  TTrias( 3, ivec3[ 5 ]( ivec3(  0, 10,  4 ), ivec3(  0,  9, 10 ), ivec3(  9, 11, 10 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ) ) ),
  TTrias( 2, ivec3[ 5 ]( ivec3(  9, 11,  8 ), ivec3(  8, 11, 10 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ) ) ),
  TTrias( 1, ivec3[ 5 ]( ivec3(  8,  5,  2 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ) ) ),
  TTrias( 2, ivec3[ 5 ]( ivec3(  0,  5,  2 ), ivec3(  4,  5,  0 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ) ) ),
  TTrias( 2, ivec3[ 5 ]( ivec3(  9,  6,  0 ), ivec3(  2,  8,  5 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ) ) ),
  TTrias( 3, ivec3[ 5 ]( ivec3(  9,  5,  2 ), ivec3(  9,  6,  5 ), ivec3(  6,  4,  5 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ) ) ),
  TTrias( 2, ivec3[ 5 ]( ivec3(  4,  1, 10 ), ivec3(  8,  5,  2 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ) ) ),
  TTrias( 3, ivec3[ 5 ]( ivec3(  5,  1, 10 ), ivec3(  5,  2,  1 ), ivec3(  2,  0,  1 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ) ) ),
  TTrias( 3, ivec3[ 5 ]( ivec3(  6,  0,  9 ), ivec3(  4,  1, 10 ), ivec3(  2,  8,  5 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ) ) ),
  TTrias( 4, ivec3[ 5 ]( ivec3(  1, 10,  5 ), ivec3(  6,  1,  5 ), ivec3(  6,  5,  2 ), ivec3(  6,  2,  9 ), ivec3( -1, -1, -1 ) ) ),
  TTrias( 2, ivec3[ 5 ]( ivec3(  6, 11,  1 ), ivec3(  2,  8,  5 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ) ) ),
  TTrias( 3, ivec3[ 5 ]( ivec3(  0,  5,  2 ), ivec3(  0,  4,  5 ), ivec3(  1,  6, 11 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ) ) ),
  TTrias( 3, ivec3[ 5 ]( ivec3(  0, 11,  1 ), ivec3(  0,  9, 11 ), ivec3(  2,  8,  5 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ) ) ),
  TTrias( 4, ivec3[ 5 ]( ivec3(  2,  9, 11 ), ivec3(  2, 11,  4 ), ivec3(  2,  4,  5 ), ivec3(  1,  4, 11 ), ivec3( -1, -1, -1 ) ) ),
  TTrias( 3, ivec3[ 5 ]( ivec3( 10,  6, 11 ), ivec3( 10,  4,  6 ), ivec3(  8,  5,  2 ), ivec3( -1, -1, -1 ), ivec3( -1, -1, -1 ) ) ),
```

§断層写真の立体化



DEVELOPER CAMP

DICOM .断層写真

§ 医用画像ファイル形式

No.	Grup	Elem	OriVR	ExpVR	Size	Desc
1	0002	0000	UL	UL	4	File Meta Information Group Length
2	0002	0001	OB	OB	2	File Meta Information Version
3	0002	0002	UI	UI	26	Media Storage SOP Class UID
4	0002	0003	UI	UI	52	Media Storage SOP Instance UID
5	0002	0010	UI	UI	22	Transfer Syntax UID
6	0002	0012	UI	UI	28	Implementation Class UID
7	0008	0005	CS	CS	16	Specific Character Set
8	0008	0008	CS	CS	16	Image Type
9	0008	0016	UI	UI	26	SOP Class UID
10	0008	0018	UI	UI	52	SOP Instance UID
11	0008	0020	DA	DA	8	Study Date
12	0008	0021	DA	DA	8	Series Date
13	0008	0022	DA	DA	8	Acquisition Date
14	0008	0023	DA	DA	8	Content Date
15	0008	0030	TM	TM	10	Study Time
16	0008	0031	TM	TM	10	Series Time
17	0008	0032	TM	TM	10	Acquisition Time
18	0008	0033	TM	TM	10	Content Time
19	0008	0050	SH	SH	10	Accession Number
20	0008	0060	CS	CS	2	Modality
21	0008	0070	LO	LO	4	Manufacturer
22	0008	0080	LO	LO	14	Institution Name
23	0008	0090	PN	PN	66	Referring Physician's Name
24	0008	1010	SH	SH	6	Station Name
25	0008	1030	LO	LO	0	Study Description

github.com/LUXOPHIA/DICOMLoader

DICOM℠
Digital Imaging and Communications in Medicine

NEMA, Suite 900
1300 North 17th Street
Rosslyn, VA 22209
Ph: (703) 475-9217
http://dicom.nema.org
dicom@medicalimaging.org

DICOM is managed by the Medical Imaging & Technology Alliance – a division of NEMA

The DICOM Standard

6 Registry of DICOM Data Elements

Note

For data elements that were present in ACR-NEMA 1.0 and 2.0 and that have been retired, the specifications of Value Representation and Value Multiplicity provided are recommendations for the purpose of interpreting their values in objects created in accordance with earlier versions of this standard. These recommendations are suggested as most appropriate for a particular data element; however, there is no guarantee that historical objects will not violate some requirements or specified VR and/or VM.

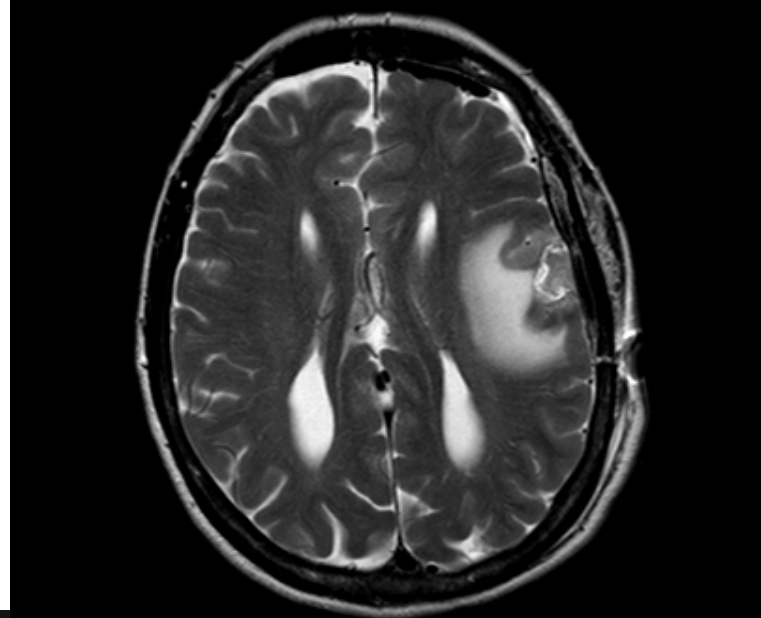
Table 6-1. Registry of DICOM Data Elements

Tag	Name	Keyword	VR	VM	
(0008,0001)	*Length to End*	*LengthToEnd*	*UL*	*1*	*RET*
(0008,0005)	Specific Character Set	SpecificCharacterSet	CS	1-n	
(0008,0006)	Language Code Sequence	LanguageCodeSequence	SQ	1	
(0008,0008)	Image Type	ImageType	CS	2-n	
(0008,0010)	*Recognition Code*	*RecognitionCode*	*SH*	*1*	*RET*
(0008,0012)	Instance Creation Date	InstanceCreationDate	DA	1	
(0008,0013)	Instance Creation Time	InstanceCreationTime	TM	1	
(0008,0014)	Instance Creator UID	InstanceCreatorUID	UI	1	
(0008,0015)	Instance Coercion DateTime	InstanceCoercionDateTime	DT	1	
(0008,0016)	SOP Class UID	SOPClassUID	UI	1	
(0008,0018)	SOP Instance UID	SOPInstanceUID	UI	1	
(0008,001A)	Related General SOP Class UID	RelatedGeneralSOPClassUID	UI	1-n	
(0008,001B)	Original Specialized SOP Class UID	OriginalSpecializedSOPClassUID	UI	1	
(0008,0020)	Study Date	StudyDate	DA	1	
(0008,0021)	Series Date	SeriesDate	DA	1	

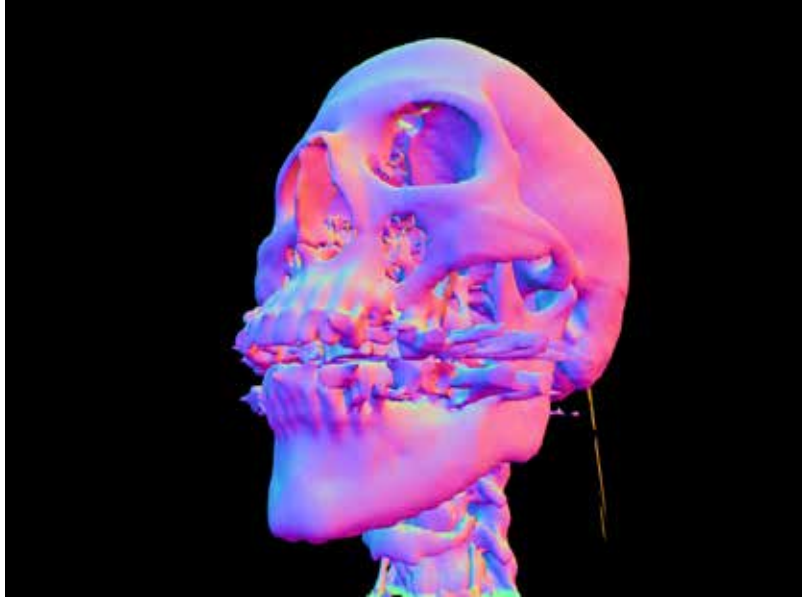
画像 .DICOM.断層写真

§ 画像の読込みにも成功

github.com/LUXOPHIA/DICOMViewer

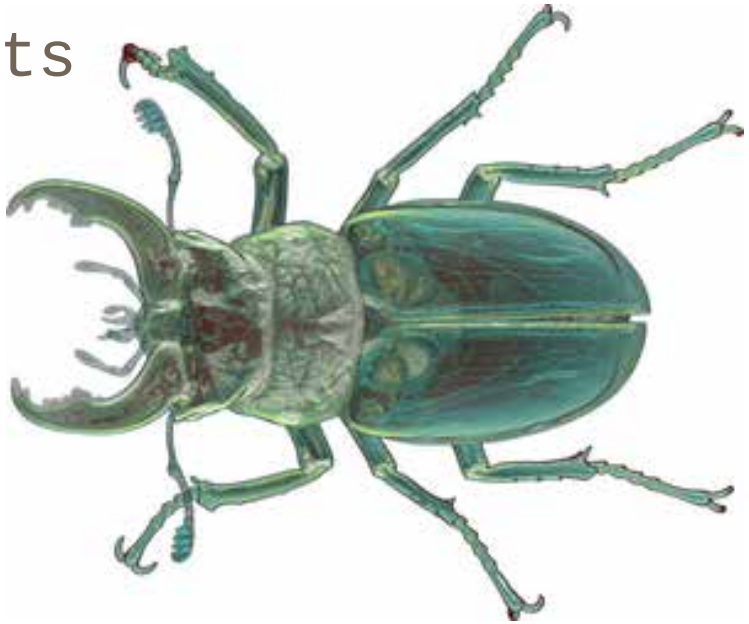


３ＤＣＴサンプル .断層写真



§ The Stanford volume data archive
  • graphics.stanford.edu/data/voldata

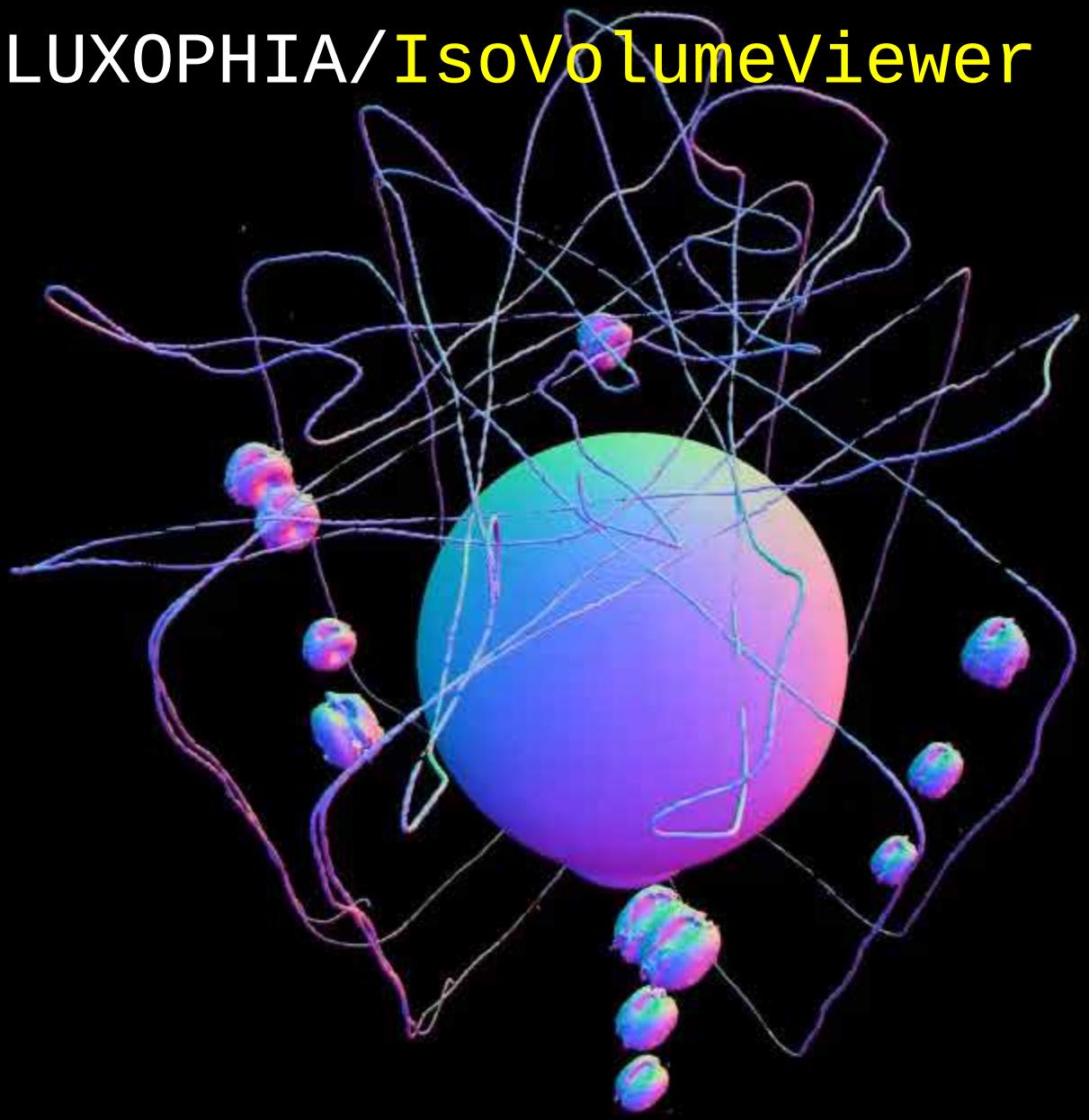
§ Visualization Group Data sets @ TU Wien
  • www.cg.tuwien.ac.at/research/vis/datasets



§ Sample Data Sets @ IDAV
  • graphics.cs.ucdavis.edu/~okreylos
    /PhDStudies/Spring2000/ECS277

github.com/LUXOPHIA/IsoVolumeViewer

github.com/LUXOPHIA/IsoVolumeViewer

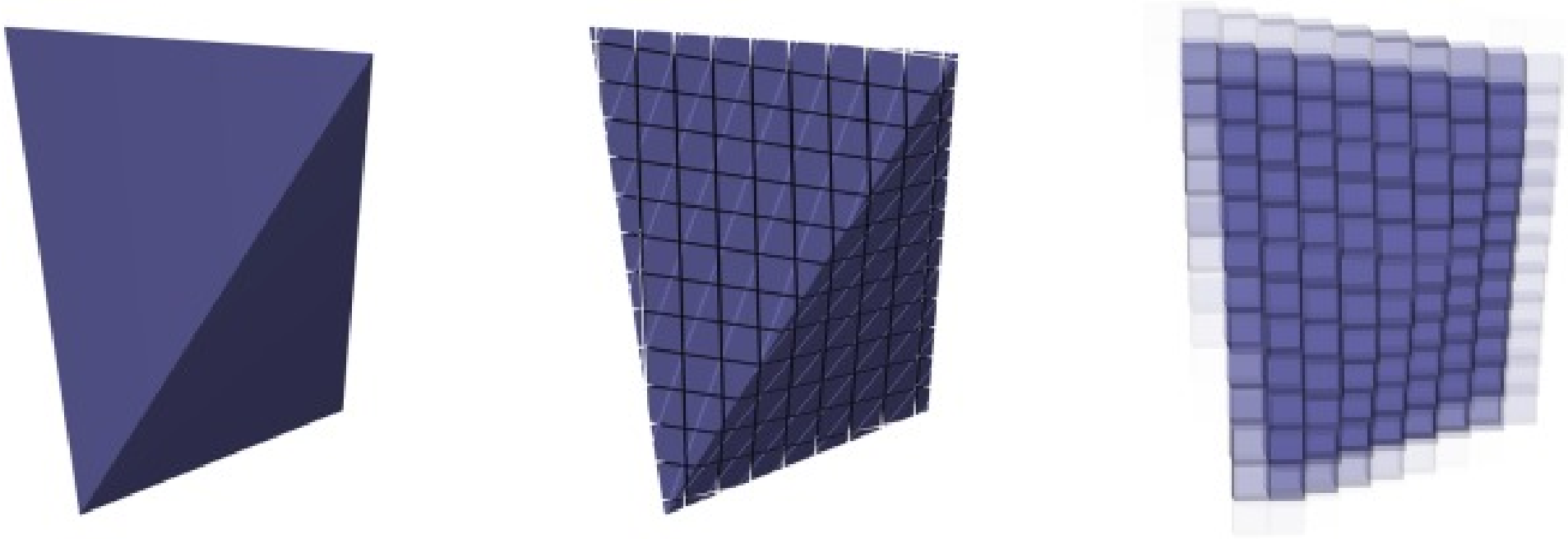


§ポリゴンをボクセルへ変換

© viscomp.alexandra.dk/?p=3836

Voxelization

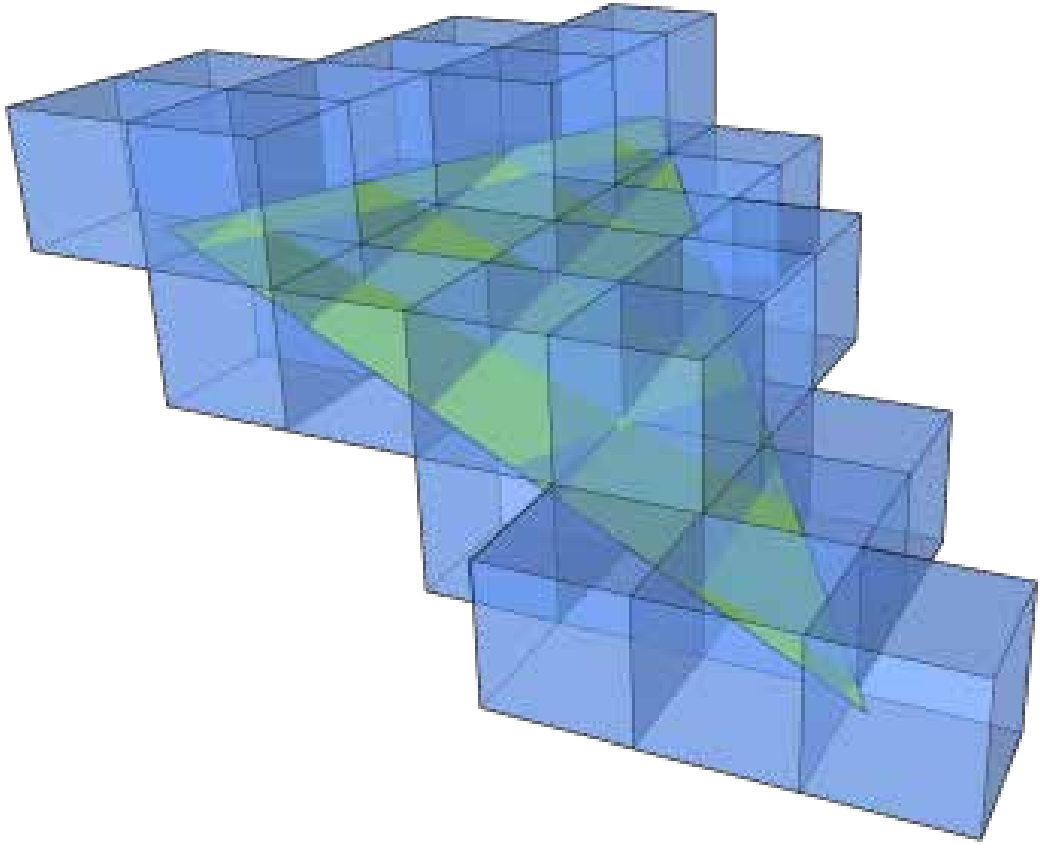
§ ポリゴンモデルをボクセルへ分解する処理



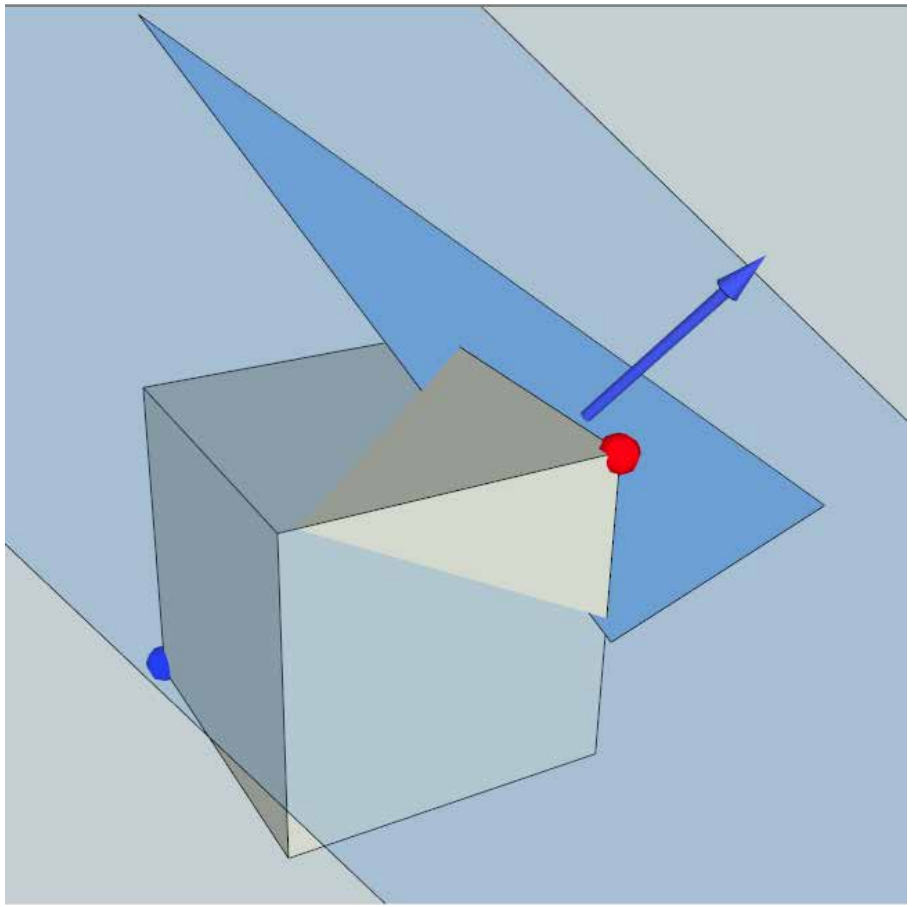
© An exact general remeshing scheme applied to physically conservative voxelization

衝突判定 .Voxelization

§ ポリゴン（三角）とボクセル（直方体）の衝突判定が必要



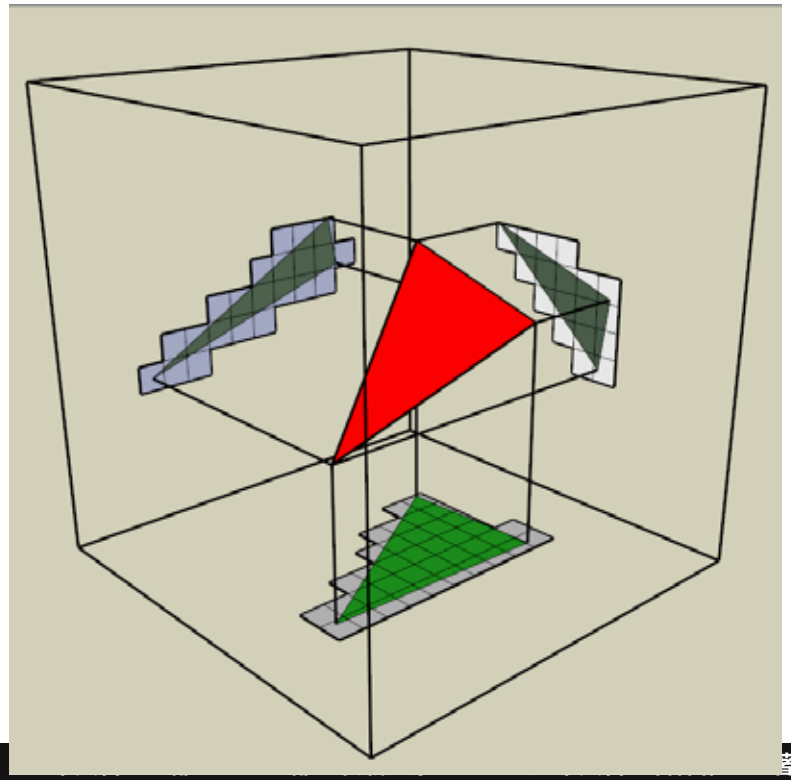
© blogs.nvidia.com/blog/2014/09/19/maxwell-and-dx12-delivered

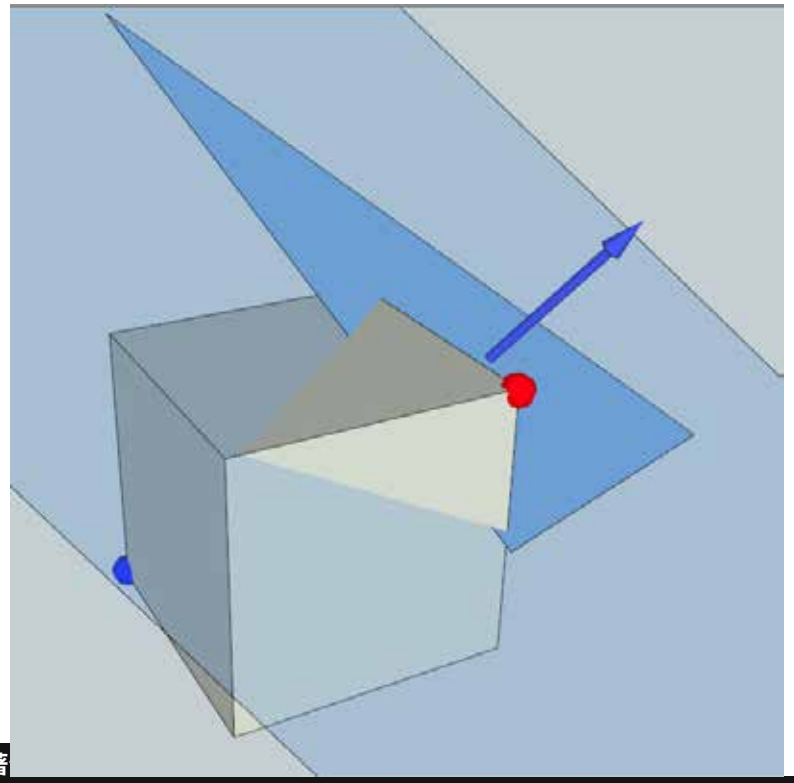


© shikihuiku.wordpress.com/2012/08/02/gpu上でのvoxel構築手法

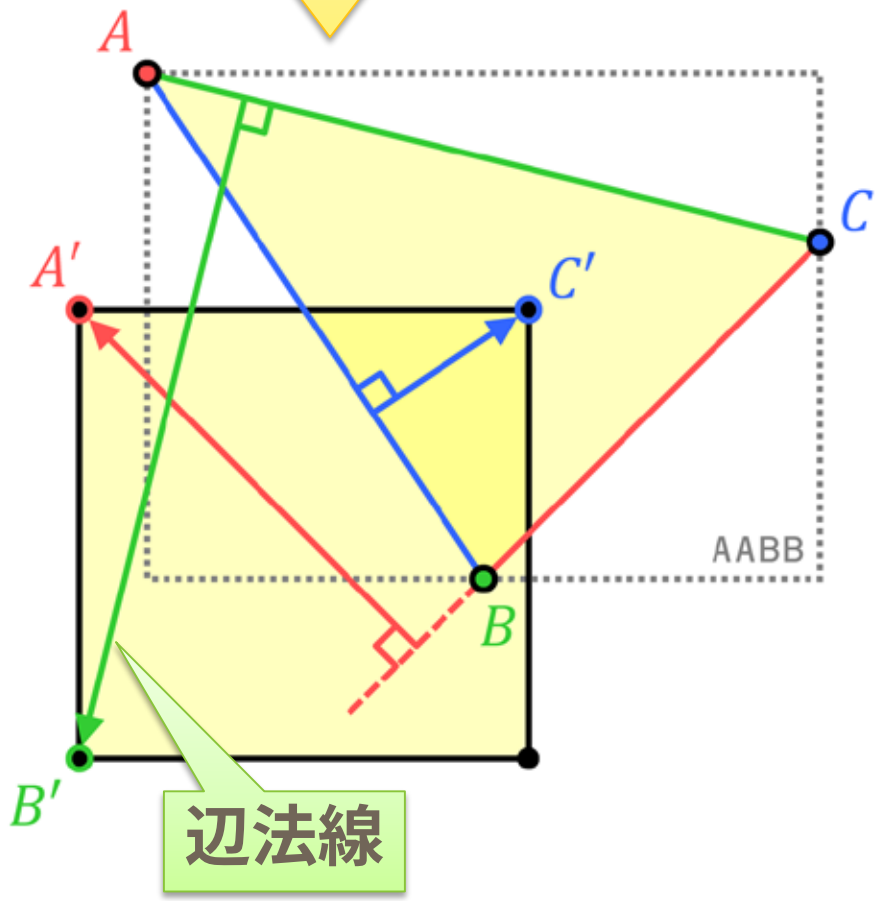
アルゴリズム① .衝突判定.Voxelization

§ **PEF** : Projected Edge Function
- 三軸方向へ射影して２Ｄ処理で判定
- 三角面の法線方向の衝突判定と併用



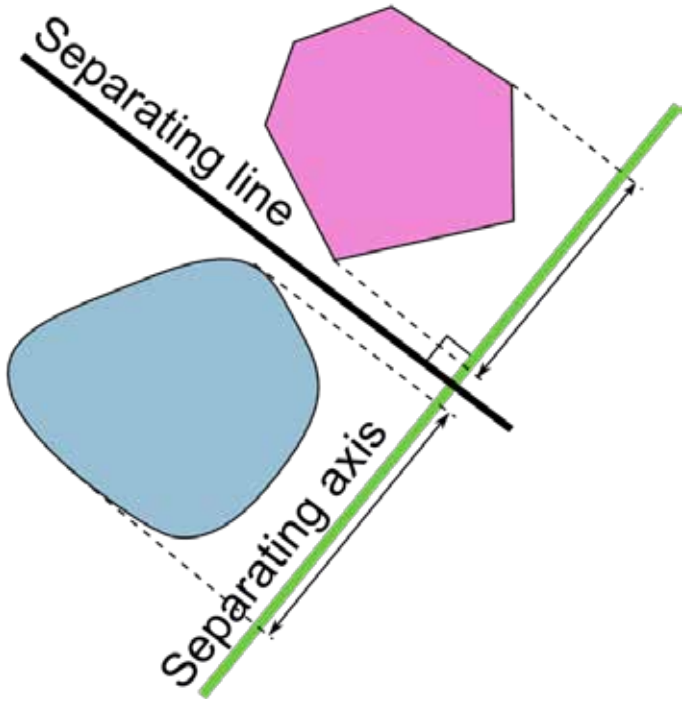


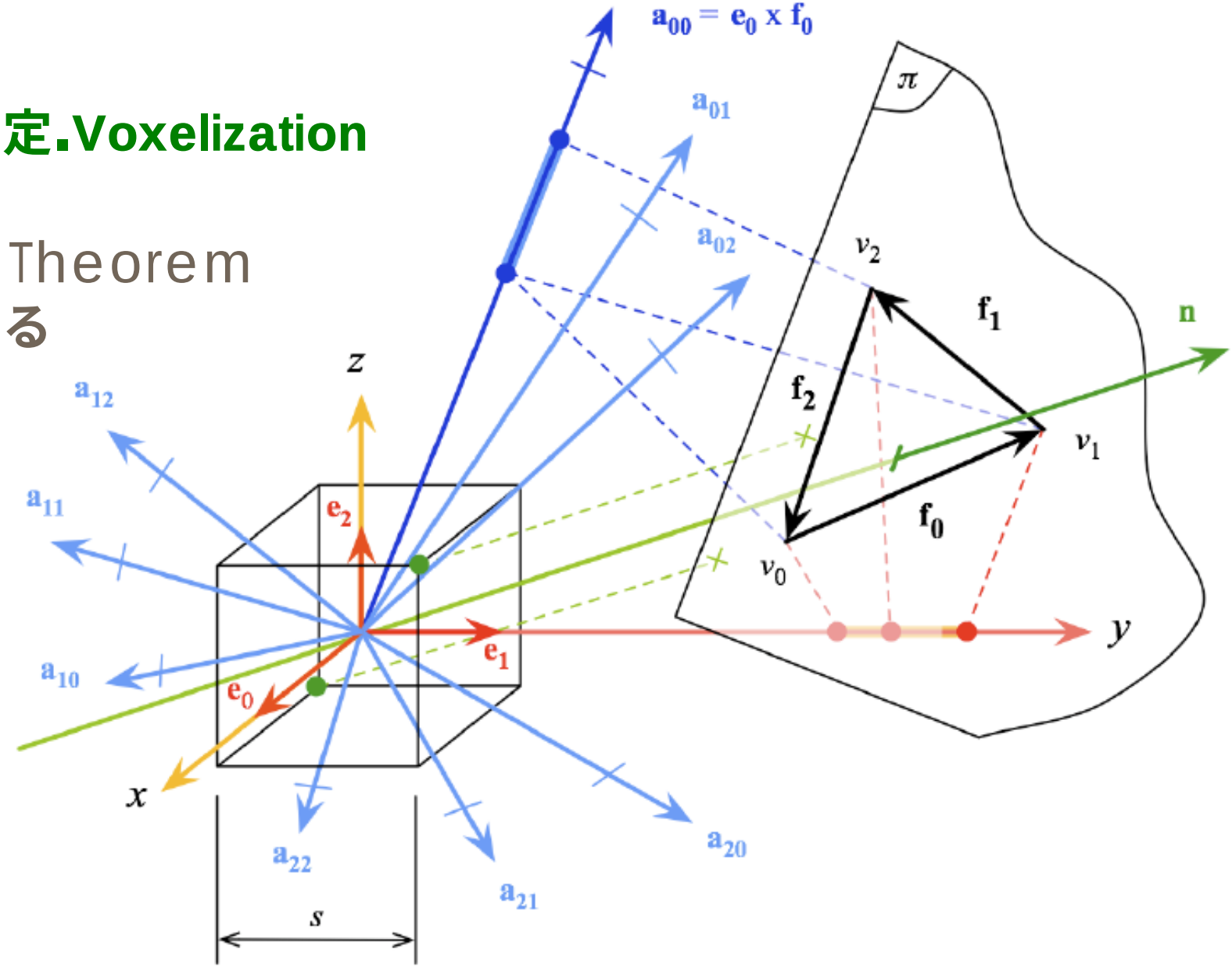
ＡＡＢＢが衝突
＆
辺法線に沿って最も前方の頂点が辺の内側に入る



アルゴリズム② .衝突判定.Voxelization

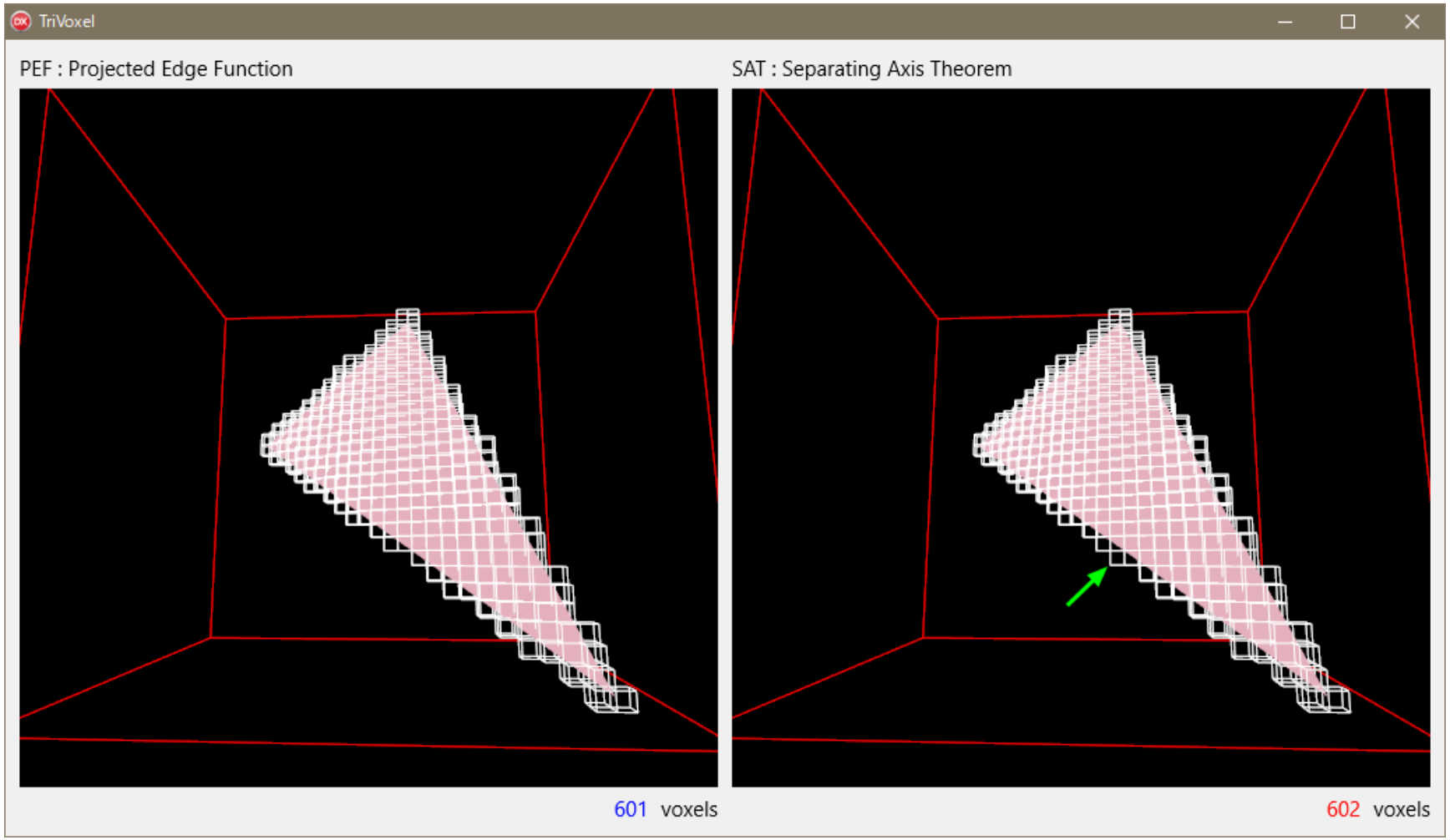
§ SAT：Separating Axis Theorem
- 分離超平面定理を利用する
- 凸体のみ対応
- １３方向を調べればいい





© Improvements of the Voxmap-PointShell Algorithm - Fast Generation of Haptic Data-Structures

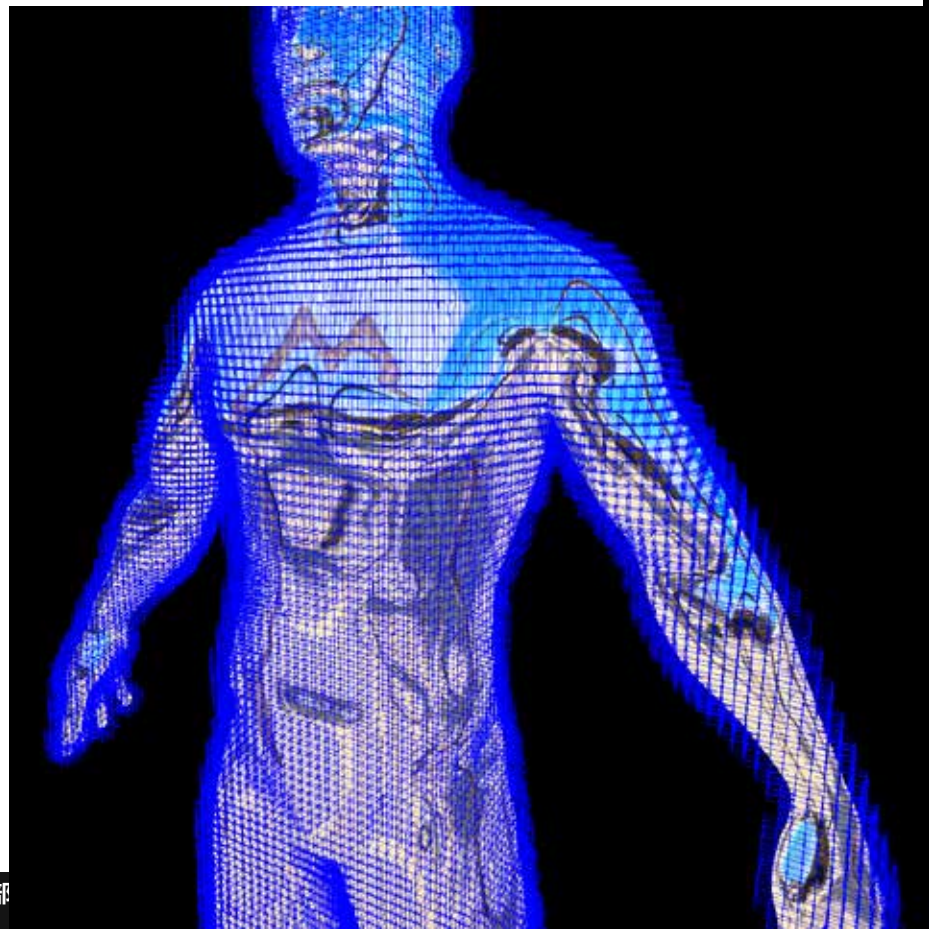
アルゴリズム比較 .衝突判定.Voxelization

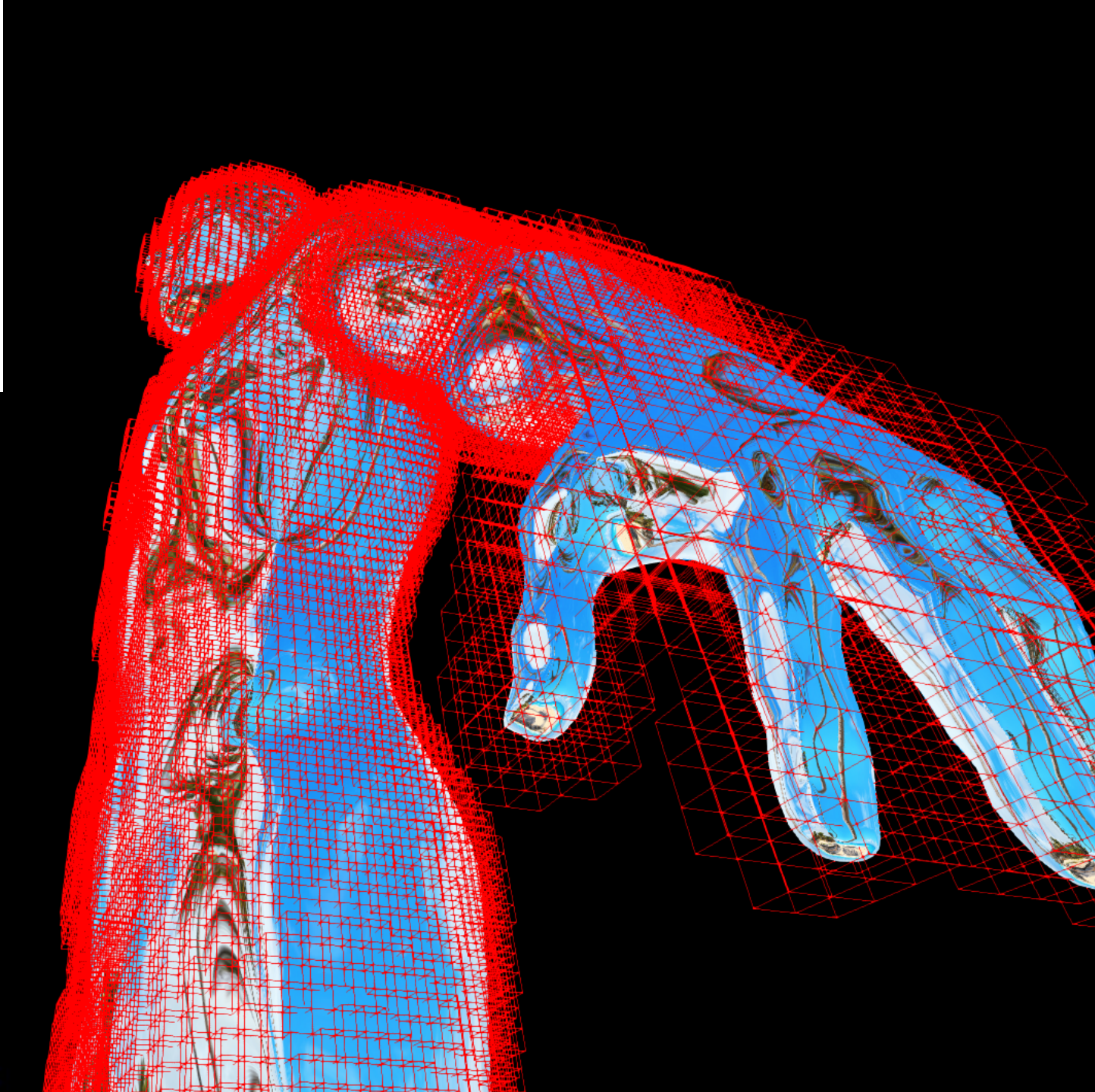


github.com/LUXOPHIA/TriVoxel

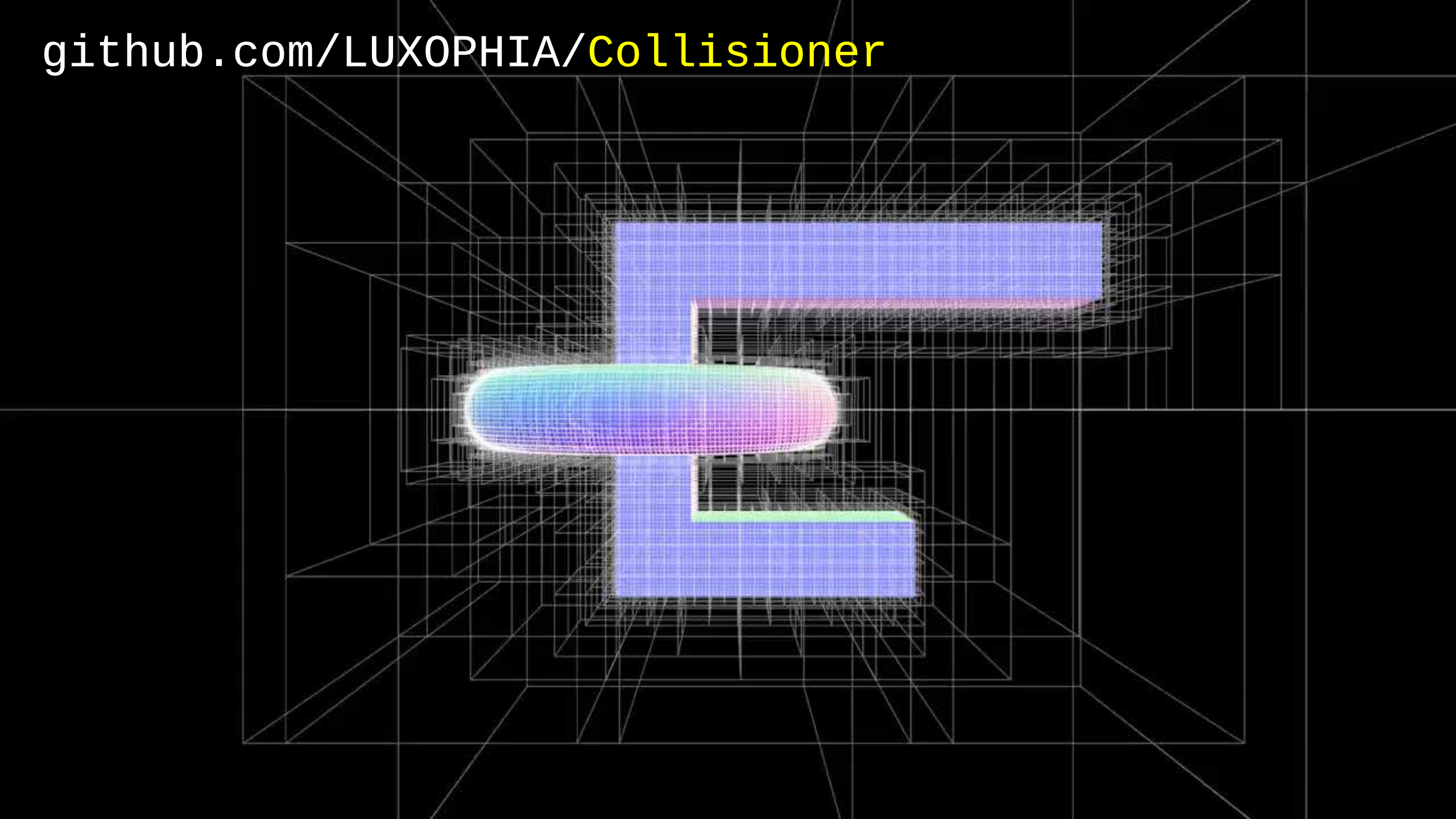
VoxMan .Voxelization

§ 人体のスキャンモデルを
ボリュームデータへ変換





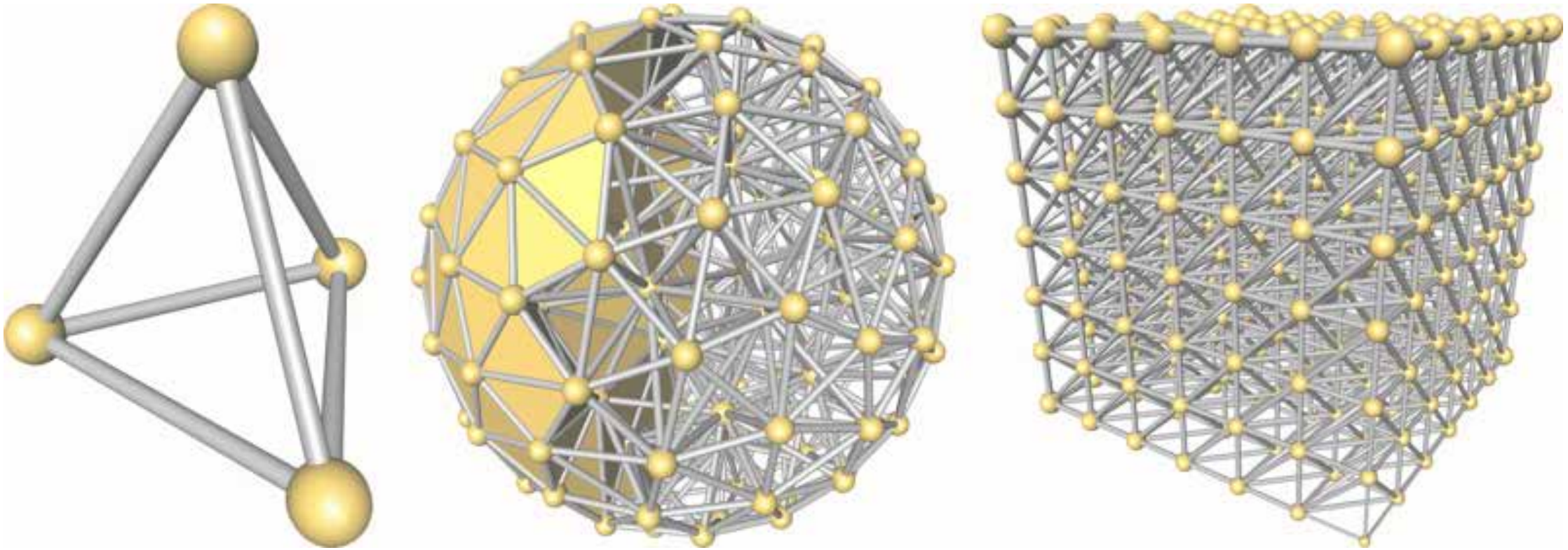
github.com/LUXOPHIA/Collisioner



§テトラヘドライゼーション

Tetrahedralization

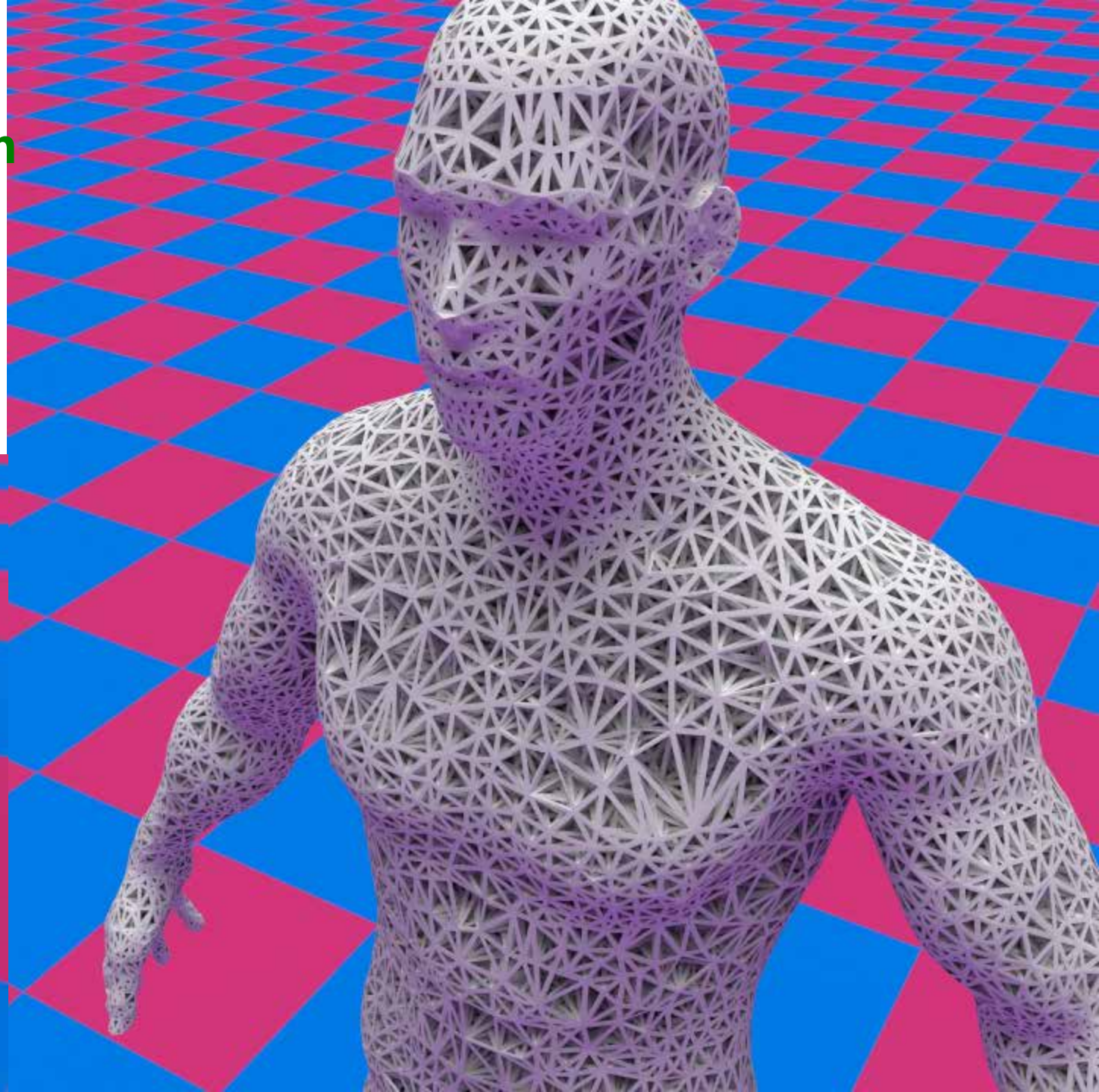
§ 等値面内部を四面体格子に変換する処理



© sites.google.com/site/introduction2cgal/manyuaru/37-3d-triangulations

TetraMan .Tetrahedralization

§ 人体を四面体メッシュ化

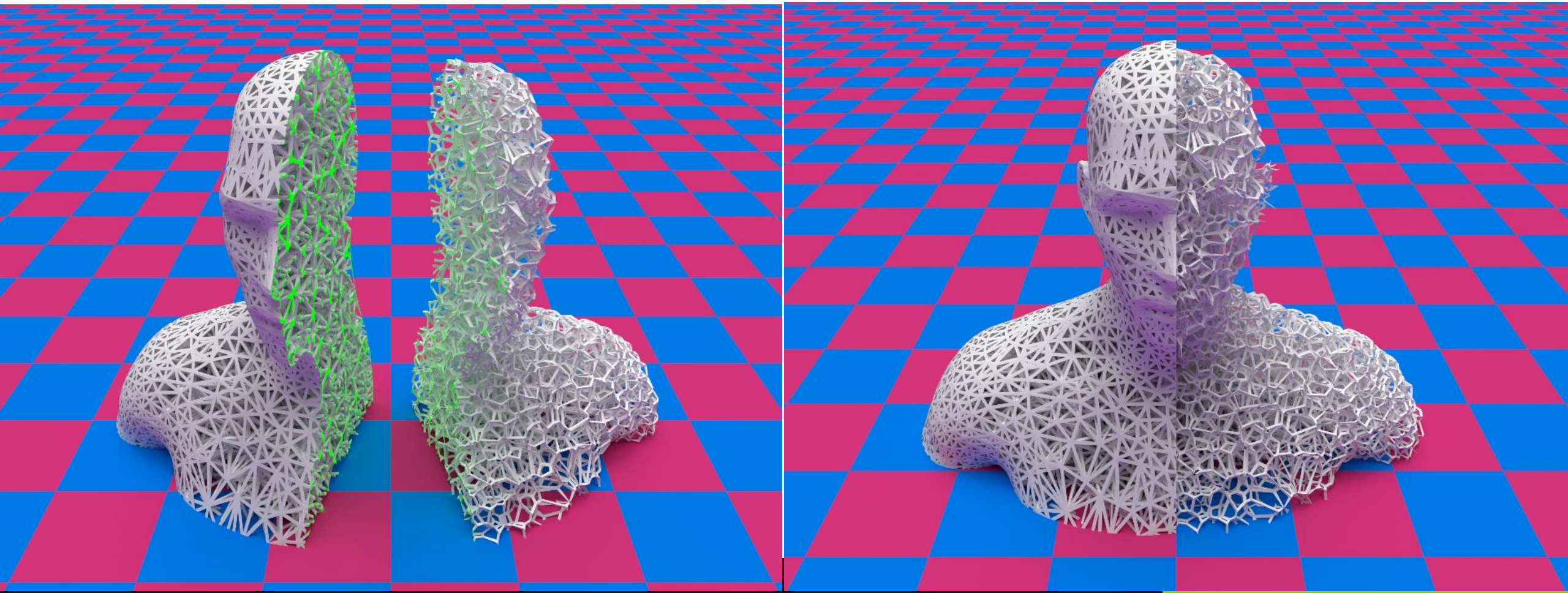


CGI2017 .TetraMan.Tetrahedralization

§ 慶應義塾大学主催の学会を飾る

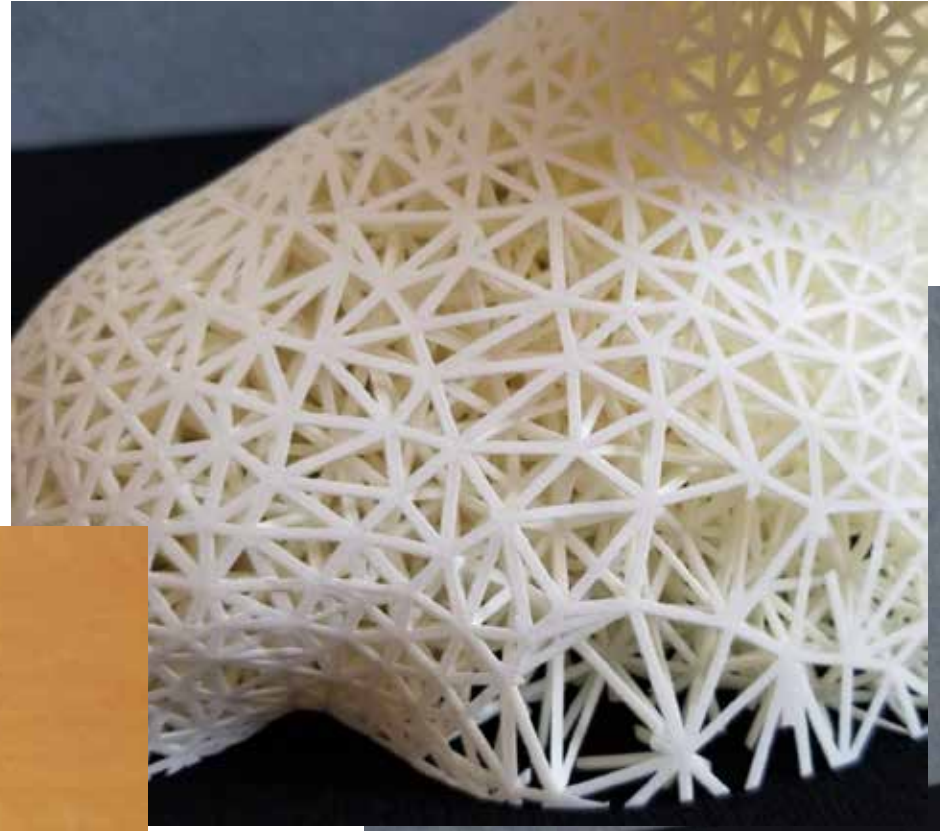
Asian Digital Modeling Contest 2017 .TetraMan.Tetrahedralization

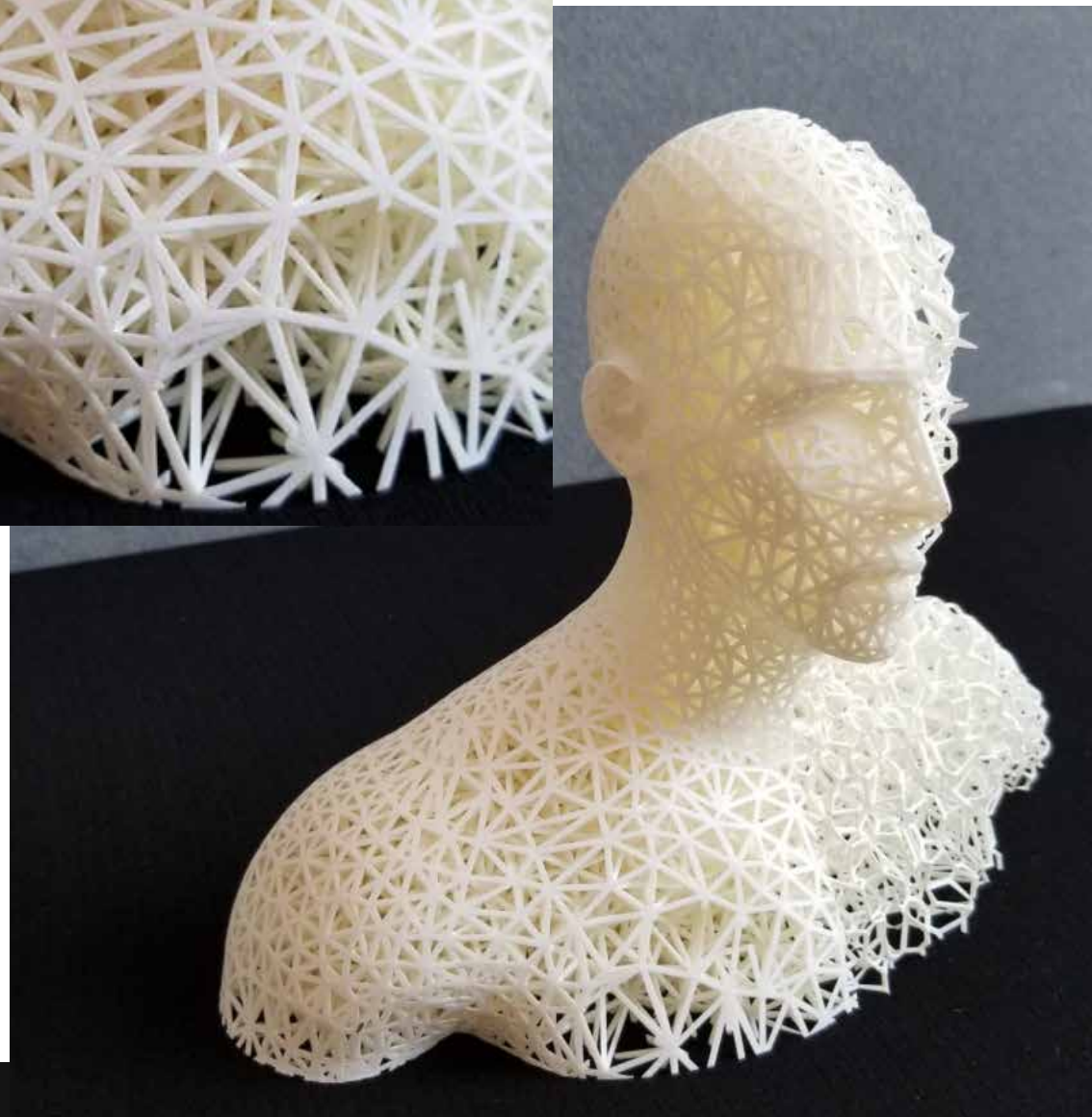
§ 作品「Duality」を出展。



３Ｄプリント .ADMC2017

§ 最優秀賞を頂きました





第35回 エンバカデロ・デベロッパーキャンプ