コンピュータ実習
第6回 2026年11月12日
グラフィック表示

担当:狩川 大輔, 宍戸 博紀

コンピュータ実習（第6回）

Web教材の第6回のページ「グラフィック表示」
を開いてください.

コンピュータ実習（第６回）

グラフィック表示

OpenGLを用いたグラフィック表示

- OpenGLとは次の特徴を有するソフトウェアインターフェース（API：Application Program Interface）である.

1. 高品質カラーイメージ
2. リアルタイム処理
3. OSに依存しない
4. グラフィックカード対応

コンピュータ実習（第6回）

グラフィック表示

OpenGLで絵を描く

- 普通のグラフィック用関数は，実行すれば線を引くだとか，円を描く，四角を描くなどの機能を持った関数です．

- 一方，OpenGLの場合は，頂点位置の定義をおこない，その頂点を使って，ポリライン（頂点が線でつながれているが，閉じていない，言い換えると線分），ポリゴン（頂点が線でつながれ，さらに閉じている，言い換えれば多角形，面），ポリラインやポリゴンを組み合わせたオブジェクトという考え方をします．
  頂点位置は座標で与えられ，この座標は3次元でも2次元でも扱うことができます．すなわち，3次元であれば立体，2次元であれば平面の絵を描くことが比較的簡単にできます．

- さらに，絵が描かれている空間を様々な方向から見た状態を作り出せます．空間に近づくと拡大，離れると縮小，視点の移動などを簡単に制御することができます．

コンピュータ実習（第6回）

グラフィック表示

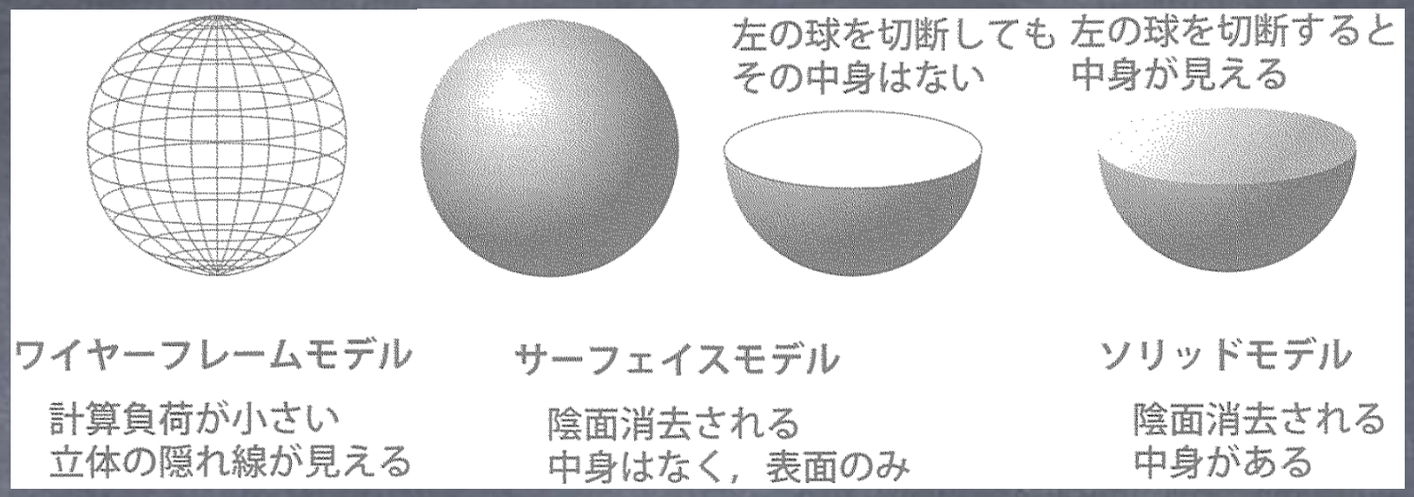
CGの用語

- モデリング（modeling）
  3D物体の形状を定めることをいい，モデリングにより作成されたものをオブジェクトと称する.
- アニメーション（animation）
  オブジェクトの位置や姿勢の動き方を定めること．動きの初めと終りを決めておいてから，これを補間するように中間の動きを逐次的に定めるキーフレーム法などがある.
- レンダリング（rendering）
  3Dオブジェクトとカメラの位置や姿勢を決定して，光を与えて（ライティング　lighting），あたかも写真と同じような画像を作り出す作業をいう.

コンピュータ実習（第6回）

グラフィック表示

CGの用語



- モデル
  - ワイヤーフレームモデル（wire frame model）

    物体を線で構成するもので，通常，物体の奥の隠れ線や面は表示される．このモデルでは，3D物体を辺（edge）と点（vertex）で構成し，辺と点で囲まれた部分を面（face）としている．
  - サーフェイスモデル（surface model）

    ワイヤーフレームモデルの面を色づけしたり，照明処理して目に見えるようにしたもので，曲面は三角形または四角形の集まりで構成される．
  - ソリッドモデル（solid model）

    サーフェイスモデルで中身が詰まったものをいう．したがって，このモデルを断ち切るとその断面が見える．

コンピュータ実習（第６回）

グラフィック表示

CGの用語

- **オブジェクト（object）**
  動きのあるものだけでなく，静的なものも含めて描画される物体をいう．オブジェクトはさらに小さなオブジェクトから成る．最小のオブジェクトのことをプリミティブ（primitive）という．OpenGLではプリミティブとして，点，線，ポリゴンの三つを指す．

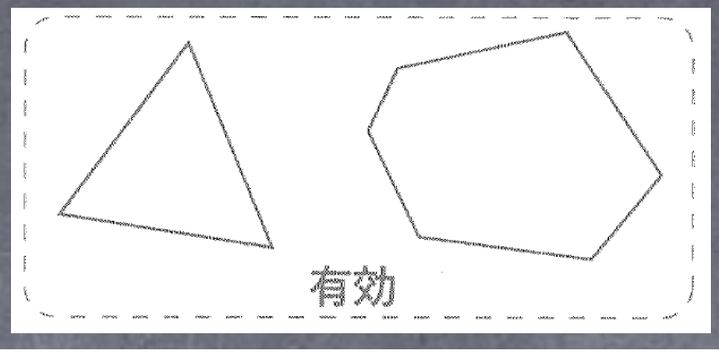
- **ポリゴン（polygon）**
  一般に，ポリゴンは多角形の意味であるが，CGにおけるポリゴンは三つ以上の頂点（vertex）とそれらを結ぶ辺（edge）から成り立つ．最小要素は三角形で，四角形以上のポリゴンは三角形ポリゴンを組み合わせて表示される．

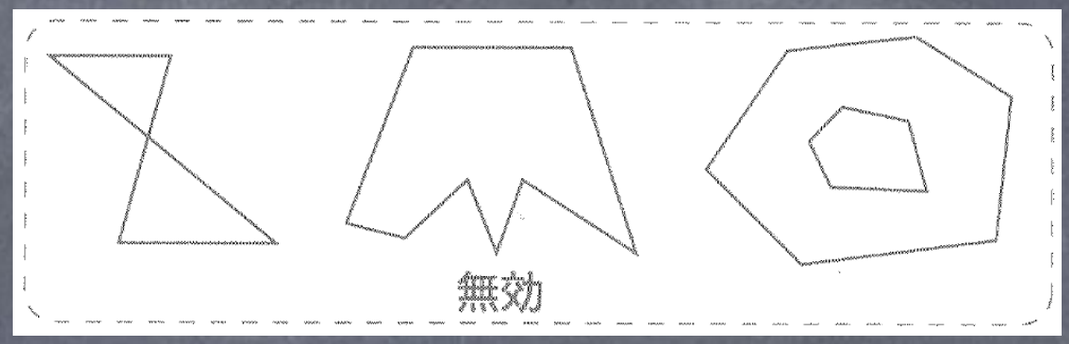
コンピュータ実習（第6回）

グラフィック表示

CGの用語

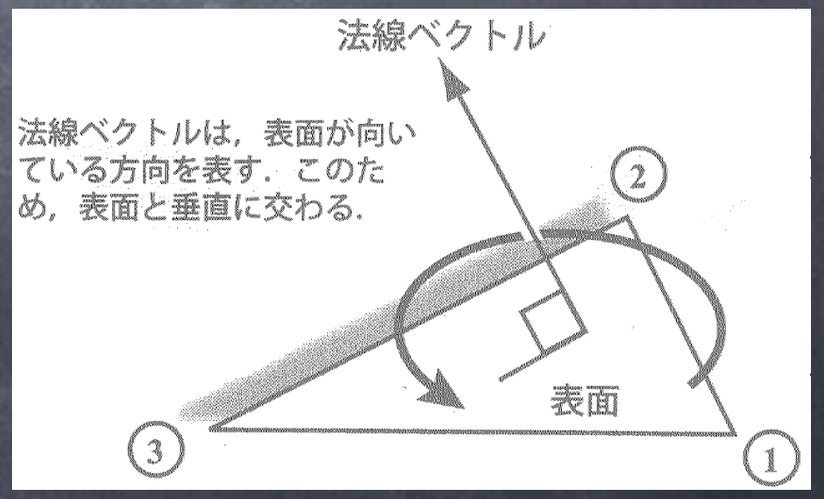
- ポリゴンの制限
  - ポリゴンの辺は交差できない
  - 凸型（convex）ポリゴンである
  - 穴の開いたポリゴンは記述できない





- ポリゴンの表と裏

  視点からポリゴンを見て，頂点が反時計回り（左回り）に配置されていると，目に見えている面が表面，その逆が裏面となる．表面の垂直方向に向いているベクトルを法線ベクトル（normal vector）といい，表面の向く方向を表す．



コンピュータ実習（第6回）

グラフィック表示

CGの用語

- イベント（event）
  - コンピュータの世界では，マウスを動かす，ボタンをクリックする，キーを押す，ユーザがウィンドウの大きさを変えたり移動したりする，などを指す．
  - これらのイベントをどの順番で処理するかの待ち行列が作成される．これをイベント待ち行列（event queue）という．

- コールバック関数（callback function）

  イベントが生じた時にそれに対応して処理する関数．イベントが起こるたびに呼び戻されるためコールバック関数と呼ばれる．表示コールバック関数やマウスコールバック関数等がある．

サンプルプログラムtest4.c
を動かしてみよう！

コンピュータ実習（第６回）

グラフィック表示

エイリアス

- test4.cをコンパイルするにはたくさんのオプションが必要
  cc test4.c –g –O2 –Wall –lglfw –lGL –lGLU –lX11 –lXrandr –o test4

- これを毎回打つのは面倒なので，
  「cc test4.c –g –O2 –Wall –lglfw –lGL –lGLU –lX11 –lXrandr」を別の名前（ここではxcc）で使えるようにする

- つまり，
  cc test4.c –g –O2 –Wall –lglfw –lGL –lGLU –lX11 –lXrandr –o test4
  ＝ xcc test4

コンピュータ実習（第６回）

グラフィック表示

エイリアスの作り方

- ホームディレクトリに，xcc.shというファイルを生成
  中に以下の文章を記述➡保存

```
#! /bin/bash
gcc $1.c -g -O2 -Wall -lglfw -lGLU -lGL -lX11 -lXrandr -lm -o $1
```

- xcc.shに操作権限を与えるために以下のコマンドを入力

```
chmod +x xcc.sh
```

- ホームディレクトリにある.bashrcに以下の一文を追加

```
alias xcc='/home/gakusei2017/xcc.sh'
```

- 上記の変更を反映させるために以下のコマンドを入力

```
source ~/.bashrc
```

コンピュータ実習（第６回）

グラフィック表示

基本物体（プリミティブ）の定義

- 図形を描くには，「描き方」と「頂点」を指定する．そのためglBegin()とglEnd()および，glVertex*()を用いる．
- modeは描く図形の種類を指定し，p0，p1，・・・，pnは座標位置（書式は複数ある）を意味する．

```
glBegin(mode);                  // 描き方（タイプ）を指定
        glVertex* (p0);// 頂点１
        glVertex* (p1);// 頂点２
        ・・・・・
        glVertex* (pn); // 頂点n
glEnd();
```

コンピュータ実習（第６回）

グラフィック表示

modeの種類

- 点の描画
  - GL_POINTS
    各頂点を単独の点として扱う
- 線の描画
  - GL_LINES
    二つの頂点を結んだ直線を生成する
  - GL_LINE_STRIP
    最初の頂点から最後の頂点まで線分を連結して描画する
  - GL_LINE_LOOP
    最初の頂点から最後の頂点まで線分を連結し，さらに最後の点から最初の点も連結して描画する

コンピュータ実習（第6回）

グラフィック表示

modeの種類

- ポリゴンの描画（頂点座標の配置の仕方に注意）
  - GL_TRIANGLES
    三つ一組の頂点を結んだ直線を生成する
  - GL_TRIANGLES_STRIP
    連結した三角形のグループを描画する
  - GL_TRIANGLES_FAN
    連結した三角形のグループを扇形に連結する
  - GL_QUADS
    四つ一組の頂点をそれぞれ独立した四角形として描く
  - GL_QUAD_STRIP
    連結した四角形のグループを描画する
  - GL_POLYGON
    単独のポリゴンを描画する

コンピュータ実習（第6回）

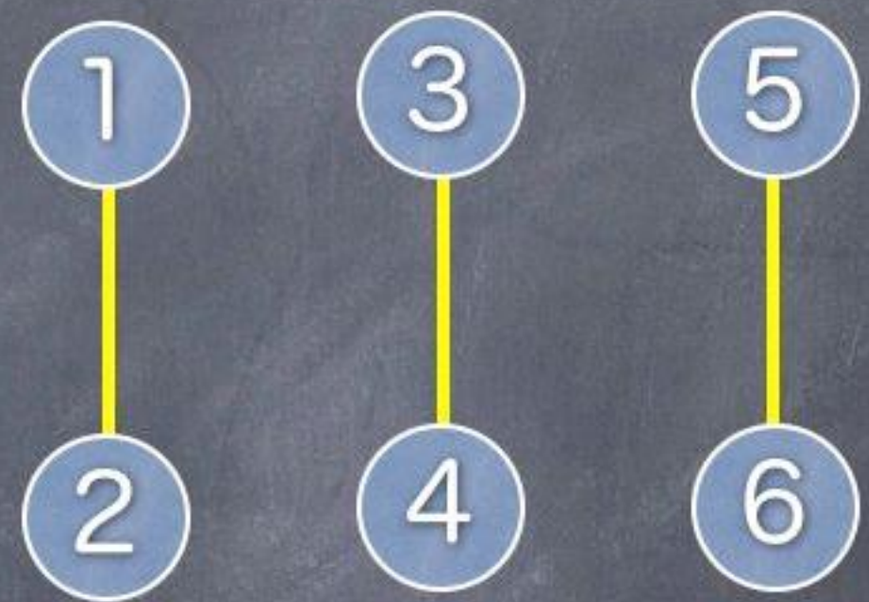
グラフィック表示

glBeginの引数（1）



GL_POINTS

頂点（点）を指定された座標にぽつんと描きます



GL_LINES

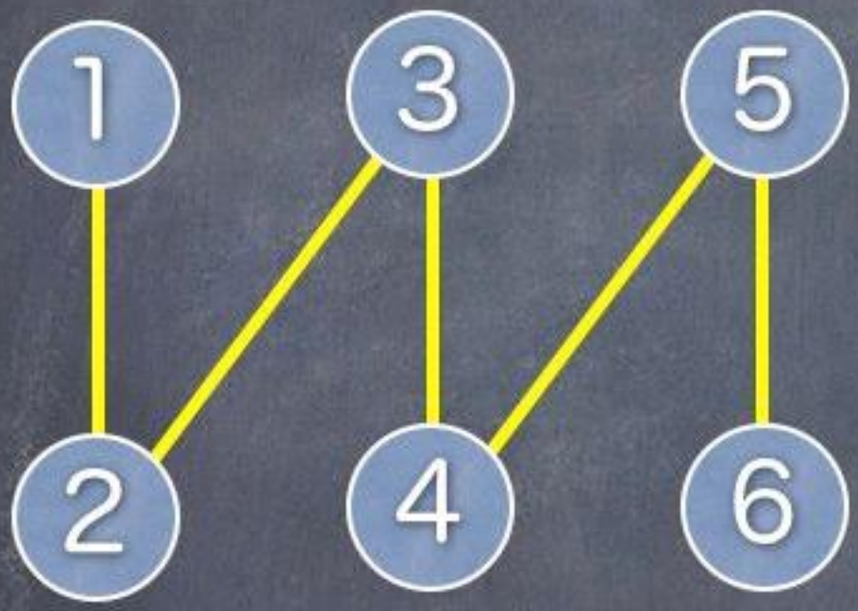
二つの頂点を結ぶ独立した線分を描きます

頂点数が6だとすると，1-2，3-4，5-6という3つの線分です．

コンピュータ実習（第6回）

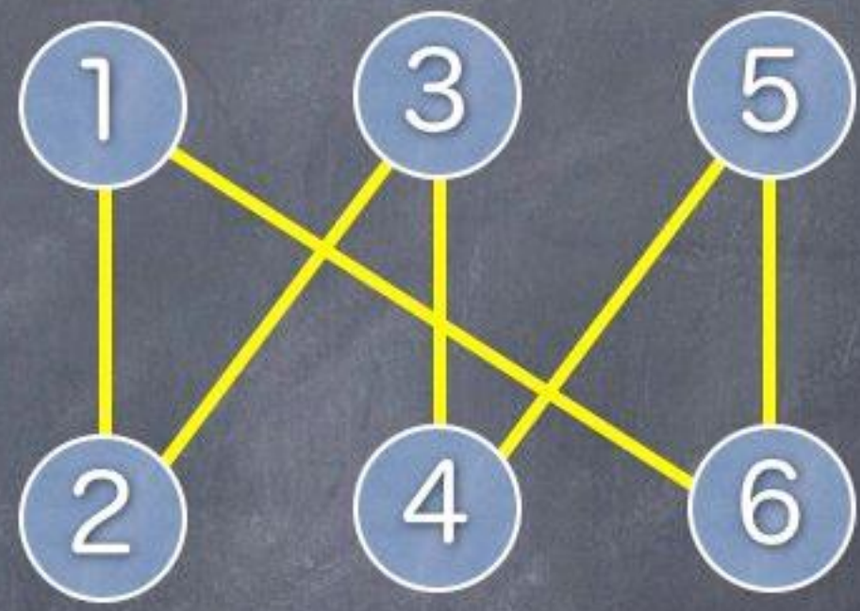
グラフィック表示

glBeginの引数（2）



GL_LINE_STRIPS

頂点を順次結んでいく直線を描きます
頂点数が6だとすると，
1-2-3-4-5-6という折れ線になります．



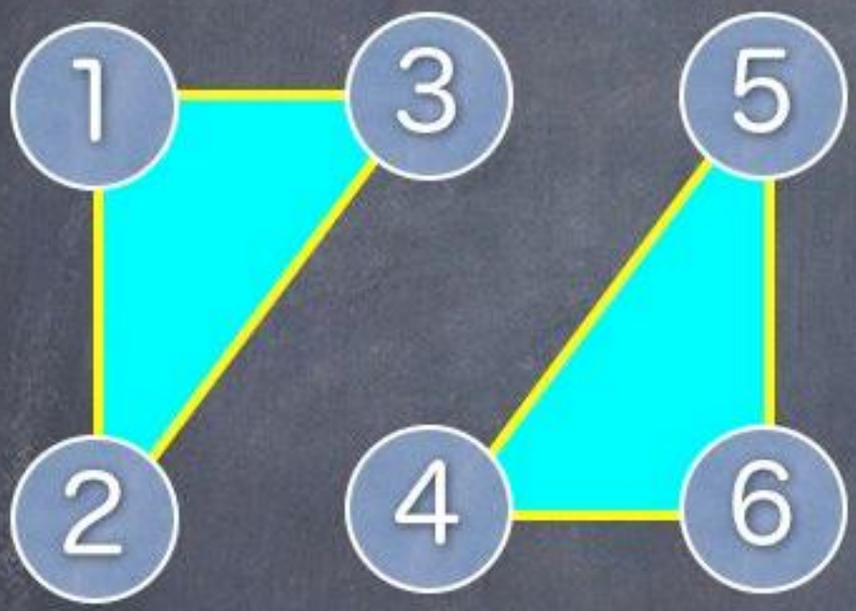
GL_LINE_LOOP

頂点を順次結んでいく直線を描くのは
GL_LINE_STRIPと同じ
しかし，最初の頂点と最後の頂点も結び，
多角形になります頂点数が6だとすると，
1-2-3-4-5-6-1という多角形です．

コンピュータ実習（第6回）

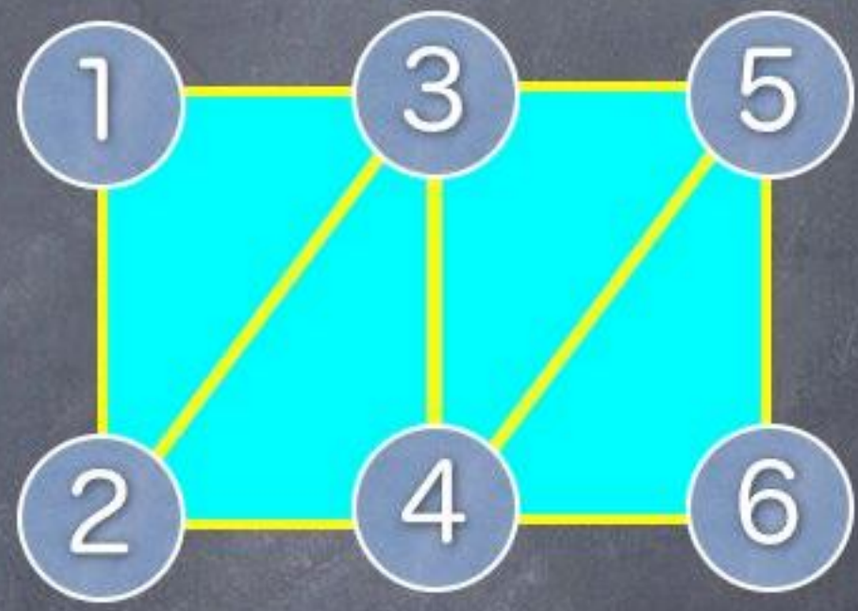
グラフィック表示

glBeginの引数（3）



GL_TRIANGLES

独立した三角形を描き，内部を塗りつぶします
頂点数が6だとすると，
1-2-3，4-5-6という独立した2つの三角形です．



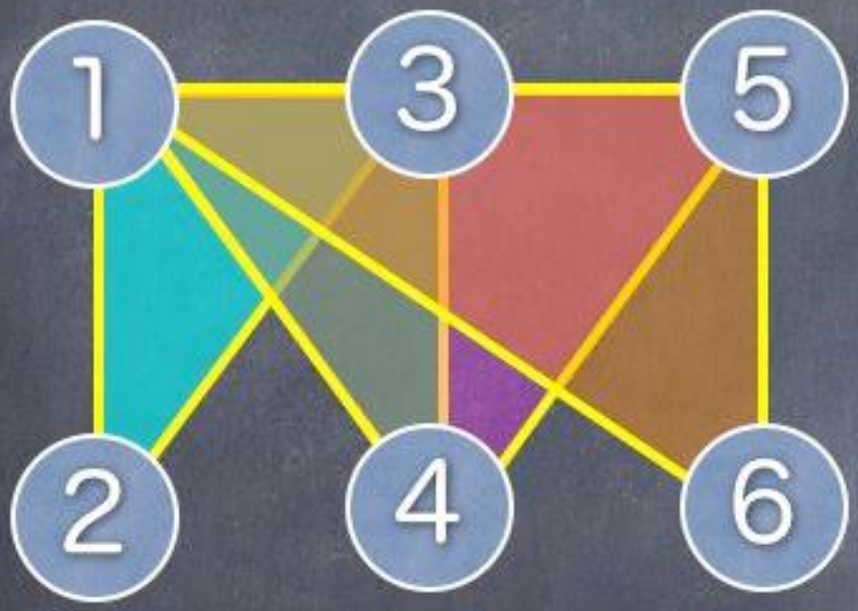
GL_TRIANGLE_STRIP

連続した三角形を描き，内部を塗りつぶします．
頂点数が6だとすると，
1-2-3，2-3-4，3-4-5，4-5-6という3つの三角形です．

コンピュータ実習（第6回）

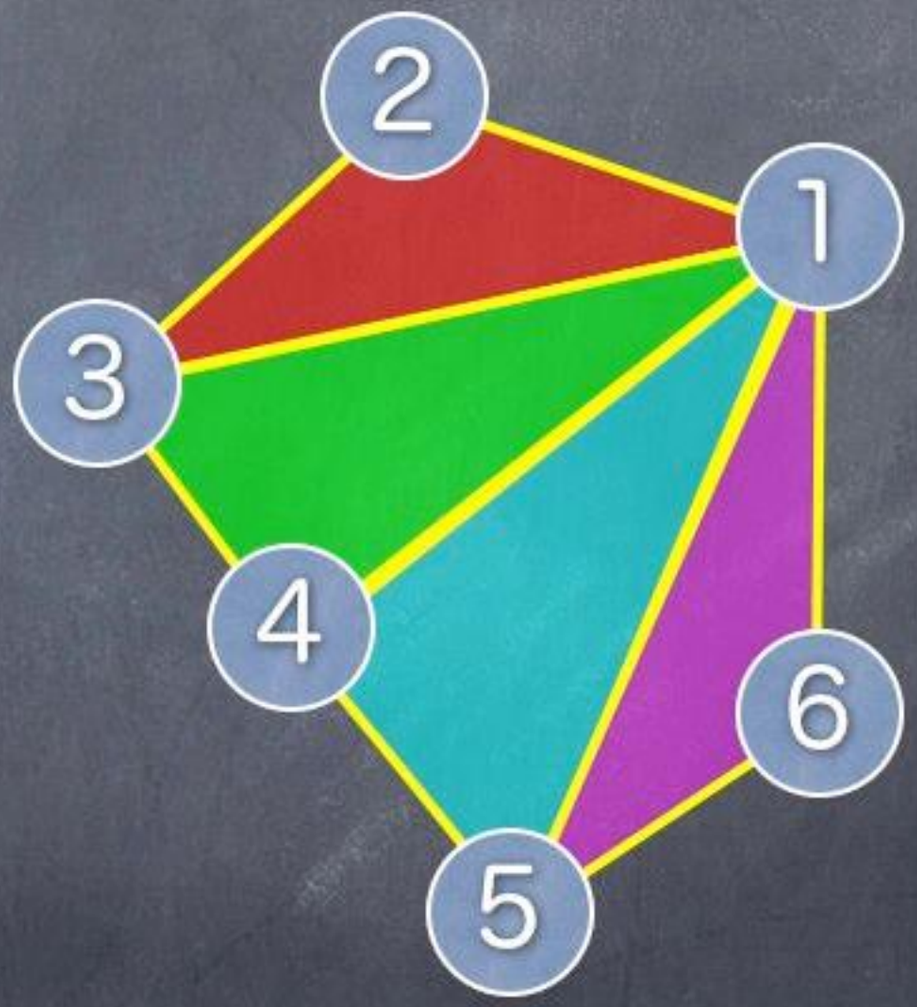
グラフィック表示

glBeginの引数（4）



GL_TRIANGLES_FAN

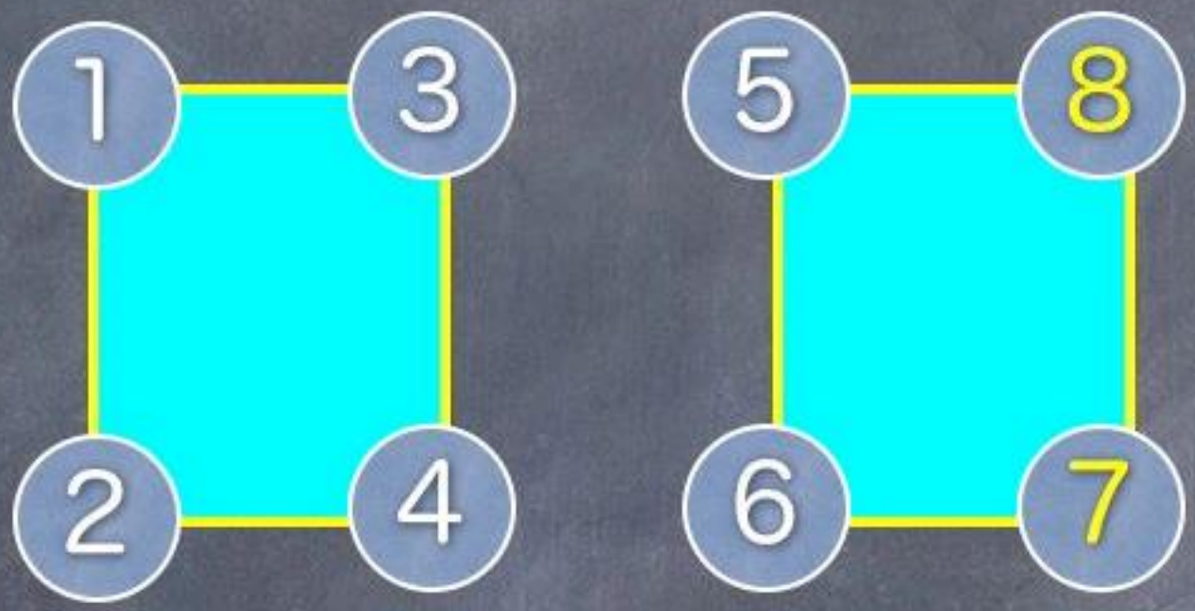
三角形で扇形を描き，内部を塗りつぶします．
頂点数が6だとすると，1-2-3，1-3-4，1-4-5，1-5-6
という頂点1を中心とした
複数の三角形で扇形を描こうとします．



コンピュータ実習（第6回）

グラフィック表示

glBeginの引数（5）



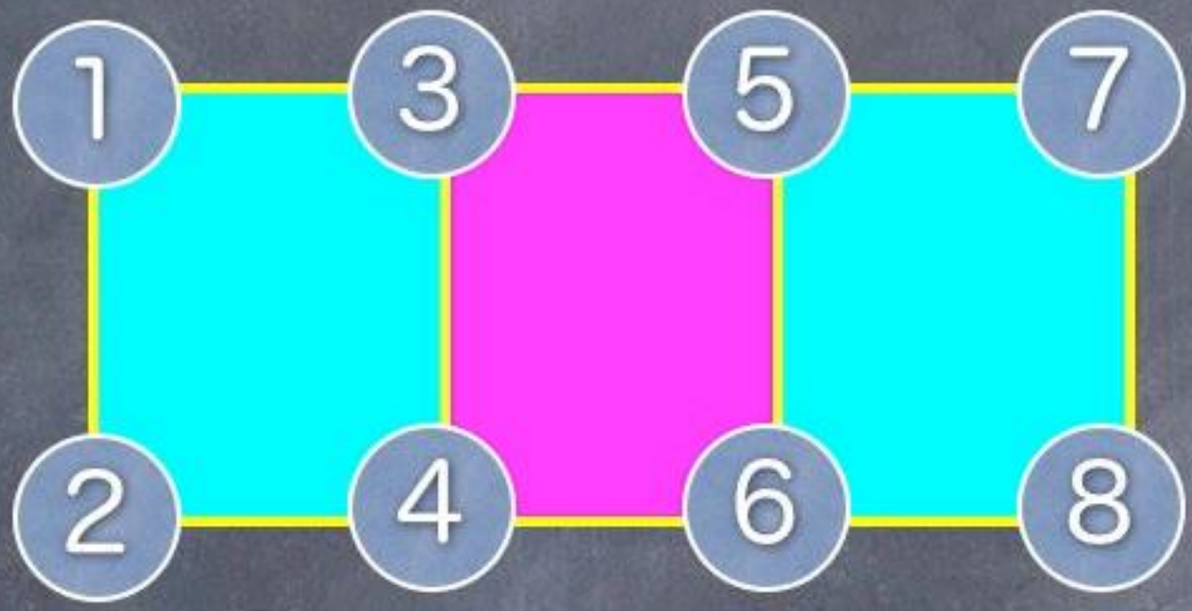
GL_QUADS

独立した四角形を描き，内部を塗りつぶします
頂点数が8だとすると1-2-3-4，5-6-7-8の四角形を描きます．
ただし，図形の一部が凹んだような図形は描けません．

コンピュータ実習（第6回）

グラフィック表示

glBeginの引数（6）



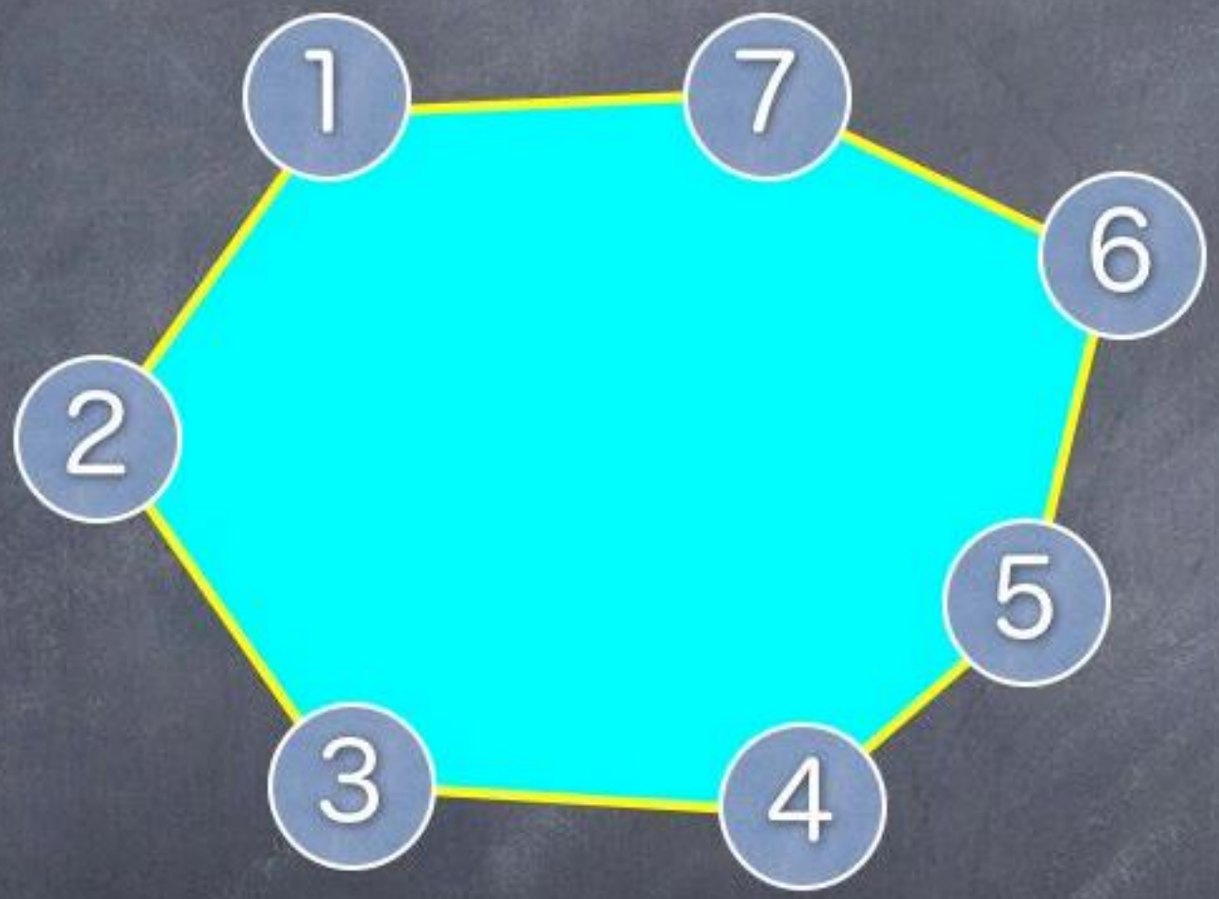
GL_QUADS_STRIP

連続した四角形を描きます

コンピュータ実習（第6回）

グラフィック表示

glBeginの引数（7）



GL_POLYGON

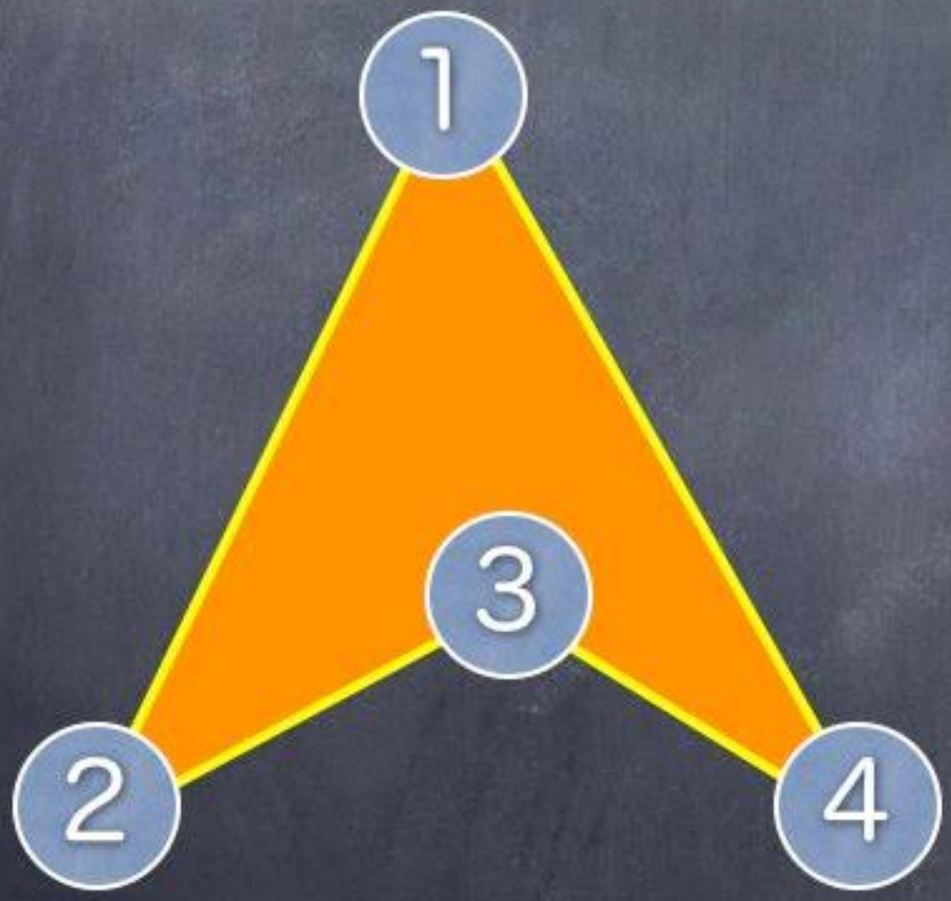
任意の数の頂点を持った多角形を描き，内部を塗りつぶします．
ただし，図形の一部が凹んだような図形は描けません．

コンピュータ実習（第6回）

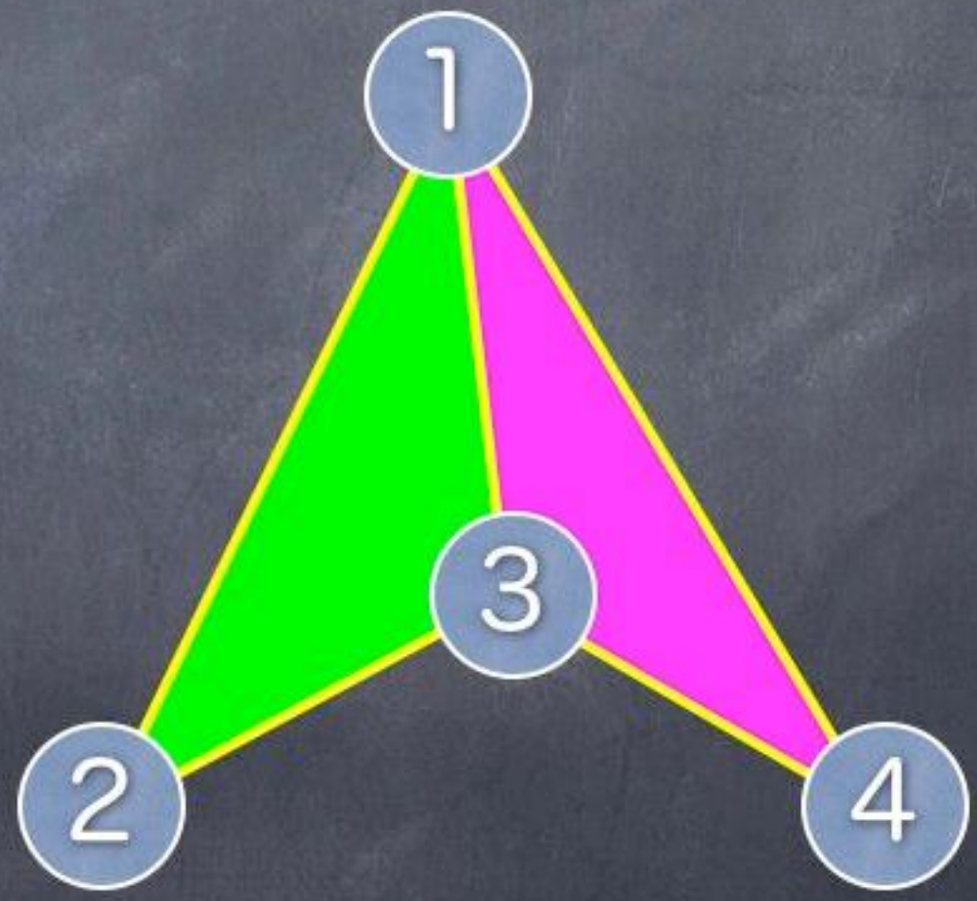
グラフィック表示

glBeginの引数（8）

GL_QUADSやGL_POLYGONでは図形の一部が凹んだような図形は描けません.



これは無理



これならOK

コンピュータ実習（第6回）

グラフィック表示

まとめ：ポリゴン描写の注意点

- 頂点座標を反時計回りに配置する.
  - ポリゴンの表面と裏面（視点から見えない面）を区別するため.
  - これにより，面の裏側を描画することを省くことができ，処理負荷の低減につながる.

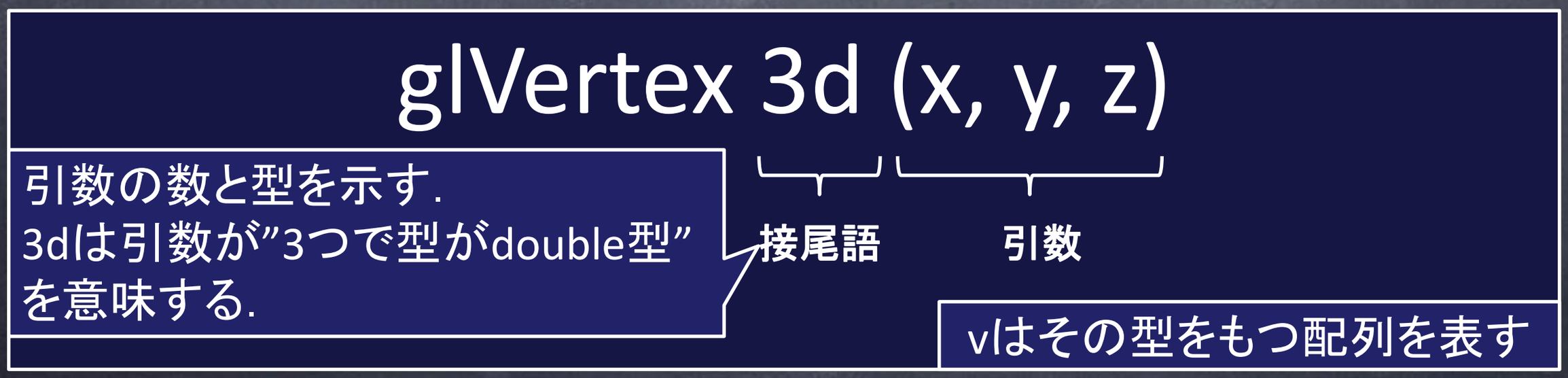
- OpenGLが扱うポリゴンは凸型に限定されている.
  これは，計算幾何学における計算可能性および計算負荷の観点からである．また，辺が交差するポリゴンも扱わないことにする.

コンピュータ実習（第６回）

グラフィック表示

glVertexの意味

- 頂点を定義する関数：glVertexには様々な種類があります．
  - Ex.) glVertex2d, glVertex2f, glVertex2i, glVertex2s, glVertex3d, glVertex3f, glVertex3i, glVertex3s, glVertex2dv, glVertex2fv, glVertex2iv, glVertex2sv, …



コンピュータ実習（第６回）

グラフィック表示

ライブラリの初期化

- glfwInit()
  - GLFWライブラリを初期化する.
  - この1行は必ず書く.

コンピュータ実習（第６回）

グラフィック表示

ウィンドウ設定

- 三次元空間や二次元平面上に描画したオブジェクトを画面上に表示するためには，glfwOpenWindow関数によりウィンドウ環境を設定する必要があります．

glfwOpenWindow (640, 400, 0, 0, 0, 0, 0, 0, GLFW_WINDOW)

- 第1・2引数：ウィンドウの幅（ヨコ）と高さ（タテ）を設定（ドット数）．
- 第3・4・5引数：背景色を原色強度で定義（RGB）．0,0,0で黒，1,1,1で白
- 第6引数：αバッファのビット数（ここでは扱いません）
- 第7引数：depthバッファのビット数（ここでは扱いません）
- 第8引数：stensilバッファのビット数（ここでは扱いません）
- 第9引数：ウィンドウモード選択．GLFW_WINDOWは「ウィンドウをデスクトップウィンドウをして開く」

コンピュータ実習（第6回）

グラフィック表示

モデルビュー変換

- OpenGLではある座標空間内にある図形を，どういう位置からどういう方法で眺めるのかを自由に設定できます．

- まず，「図形をどういう位置からカメラで見ている」というのを定義する必要があります．例えば，カメラが動いてオブジェクトに近づいても，オブジェクトが動いてカメラに近づいても，結果的には同じ絵になります（背景を除く）．言い換えると，カメラが動いてもオブジェクトが動いても，絵は変化します．

- このカメラと図形の相対位置をまとめて変化させるのがモデルビュー変換になります．

コンピュータ実習（第6回）

グラフィック表示

モデルビュー変換

```
glMatrixMode(GL_MODELVIEW)
glLoadIdentity();
```

- カメラとモデルの位置関係を示すモデルビュー変換行列スタックを選定する．以下の変換（平行移動，回転，スケーリング）を指定しないと特に特別な変換はしない．
- 直前に選択されたスタックの最上部に単位行列を格納する
  - スタックとは一時的な記憶用にとられた記憶場所のことであり，スタックに最後に入れた情報は最初に取り出される．

コンピュータ実習（第6回）

グラフィック表示

モデルビュー変換

- OpenGLは，オブジェクトの平行移動（translation），回転（rotation），スケーリング（scaling）の三つのモデリング変換を用意している．
- glScaled(ax, ay, az);
  原点とオブジェクト上の頂点座標の距離にスケーリング係数（ax, ay, az）を乗じて，オブジェクトの拡大縮小を行う．
- glTranslated(dx, dy, dz);
  dx, dy, dzの値だけオブジェクトを平行移動させる．
- glRotated(angle, dx, dy, dz);
  原点と（dx, dy, dz）を結ぶ軸を中心として，angle[deg]だけオブジェクトを左回りに回転させる．

コンピュータ実習（第6回）

グラフィック表示

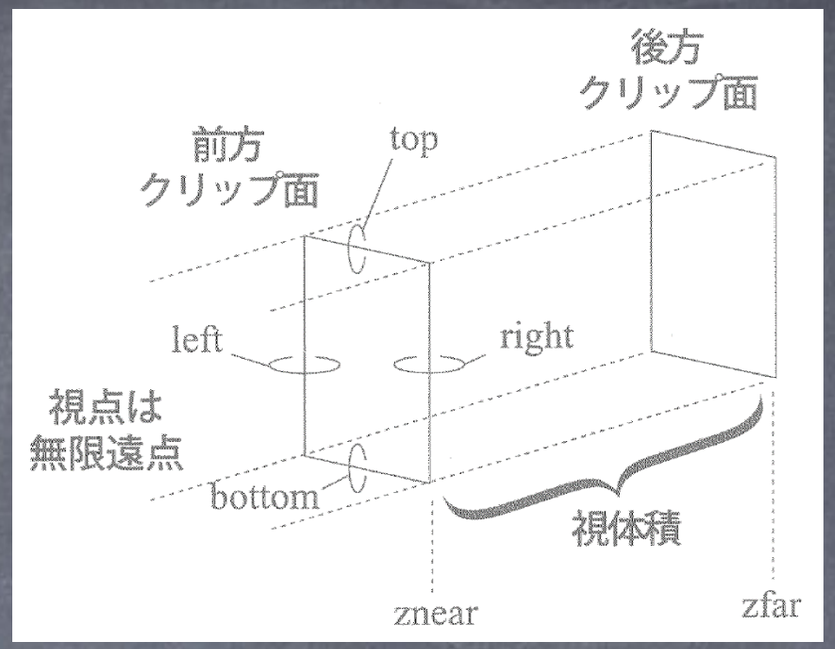
モデルビュー変換

- コンピュータ内で三次元的に図形を配置していても，PCの画面は二次元ですから，当然，カメラで撮影した写真のように二次元平面上に投影した形でしか見ることができません．これを射影と呼びます．
- 自然の摂理では，遠くのものは小さく，近くの物は大きく見えます．これを遠近法だとか，透視射影と呼びます．
- もうひとつは正射影とよばれる遠近をまったく考慮しない投影法です．
- これら平面への写り方を定義するのが投影変換行列です．

コンピュータ実習（第6回）

グラフィック表示

投影変換



- glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  - 投影法の変換行列を投影変換行列スタックを選定する
  - 直前に選択されたスタックの最上部に単位行列を格納する
- glOrtho(left, right, bottom, top, znear, zfar)
  - 直交投影を定義する
  - left, right：左右の垂直のクリップ面の座標を指定する.
  - bottom, top：上下の水平のクリップ面の座標を指定する.
  - znear, zfar：遠近方向のクリップ面までの距離を指定する.

コンピュータ実習（第6回）

グラフィック表示

ビューポート変換

- モデルビュー変換では図形とカメラの設定を行いました．これらで決定された画像をどのように表示するかを設定するのがビューポート変換です．
  ➡ 実際に画面 に切り取る部分や表示する位置を設定するもの．例えば，開いているウィンドウの一部分だけ，あるいはウィンドウの大きさを超えてビューポートを開くこともできます．

- glViewport(x, y , width , height)
  - x, y：ビューポート方形の左下隅の座標を指定する
  - width, height：ビューポート方形の幅と高さを指定する

コンピュータ実習（第6回）

グラフィック表示

ダブルバッファ

- OpenGLの図形描画では，描画処理のちらつきを防止するため，図形が描画されるバッファとよばれる領域が2つ用意されています．このうちどちらか一方はPC上に表示されており，もう一方は表示されていません．表示されている方をフロントバッファ，表示されていない方をバックバッファと呼びます．

- さらに，このふたつのバッファは瞬時に入れ替える事ができます．すなわち今までバックバッファだったものをフロントバッファに，フロントバッファだったものをバックバッファにできます．この仕組みを利用してアニメーションができます．

コンピュータ実習（第６回）

グラフィック表示

ダブルバッファ

- glClearColor(r, g, b, a);
  glClear(GL_COLOR_BUFFER_BIT);
  バッファの初期化設定をします．r,g,bの値をそれぞれ0-1の間で定義して色をつくります．
  aはα値ですが，これは色の混合を行うときに用いますが，今回は使用しませんので0を指定します．
- その後，glClearで実際にバッファを初期化しています．
- この後は先に述べたような方法で図形を描いていくわけですが，これはすべてバックバッファに描かれます．すなわち描き終わってもそのままでは表示されません．これを表示させる，すなわちフロントバッファとバックバッファを入れ替えるために，以下の関数を実行します．
- glfwSwapBuffers();
- これで，図形が表示されました．あとはこの動作の繰り返しです．

コンピュータ実習（第６回）

グラフィック表示

その他の描画関数

- glColor3d(r, g, b)
  描画色の決定．定義色はRGB.

- glLineStripple(factor, pattern)
  - 破線の定義ですが，まずは patternから説明します．
    ここには、破線を描画するドットの並びを2進数で表現し，それを16ビットの数値にしたもので指定します．2進数の 1 は描画するドット，0 は描画しないドットを表します．
    - たとえば，- - - - - - - - - という形の破線を2進数でドットの並びにすると **1010101010101010**となり，これは16進数表記で**0xAAAA**です．
  - さらに，pattern に factor を乗算した値が実際の破線となります（上の例は factor=1)
    - 先ほどの 0xAAAA という値に対して factor を 2 に指定した場合，**1100110011001100110011001100** となります．これは、単純に先ほどの破線を2倍に伸ばした形です．
  - ただし，このgl_line_strippleはそのままでは使用できません．
    glEnable(GL_LINE_STIPPLE)　glDisable(GL_LINE_STIPPLE)
    この関数で囲まれる間だけ使用できるようになります．

コンピュータ実習（第６回）

グラフィック表示

その他の描画関数

- glLineWidth(x)
  描かれる線の幅をドット単位で定義します

- glPointSize(x)
  描かれる点の直径をドット単位で定義します

コンピュータ実習（第6回）

グラフィック表示

その他の機能

- OpenGLは図形を描画するためだけの関数群です．しかし，実際には描画先がウインドウシステムであり，同時にキーボードやマウスも使っています．
- そのため，ウインドウ，キーボード，マウスの操作を扱う関数群も同時に提供されています．

- glfwGetKey(GLFW_KEY_ESC)
  glfwGetWindowParam(GLFW_OPENED)
  glfwGetWindowSize(&width, &height)
  glfwTerminate()
  glfwWaitEvents()　　など

コンピュータ実習（第６回）

グラフィック表示

ここまで理解できたら，

- もう一度，サンプルプログラムtest4.cでどのようにプログラムが書かれているかを確認してみよう.

- 中身が理解できたら，
- 図形の形・大きさを変えてみよう
- モデルビューを変更して，見え方を変えてみよう

コンピュータ実習（第6回）

グラフィック表示

今日の課題

- ex5.cをコンパイルして実行してみる.
- プログラムを改造する.
  1. 重力による放物線の動作の実現
  2. 配列を使って5つ同時に跳ねさせる
  3. 万有引力＆ケプラーの法則の検証シミュレーションを行い，さらに惑星を増やしてみる.
- さらに，はねるボールの形を変えてみる，マウスで指定した位置からボールの動きをスタートさせる，何かキーやボタンを押すとボールが増える，減る，動きが変わるなど，いろいろ改造してみてください.