

# コンピュータグラフィックス 基礎

## 第1回 イントロダクション

遠藤 結城

endo@cs.tsukuba.ac.jp



# この科目の学習の目標

- コンピュータグラフィックスの基本原理を理解する
- グラフィックスライブラリ OpenGL を用いて、CGを使用したプログラムの開発を行えるようになる

- × 3DCGソフトを使ってカッコいい映像を作る
- × 3DCGゲームを開発する



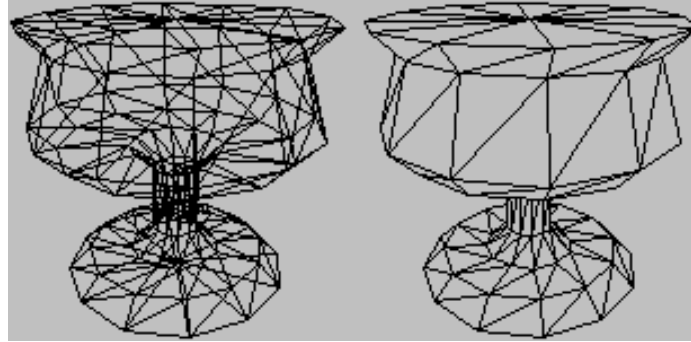
# CGの歩み

実写そっくりの画像（映像）を作る  
幾何(数学)→シミュレーション(物理)

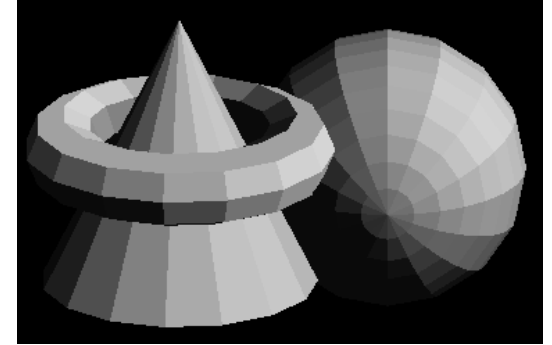


## CGの誕生

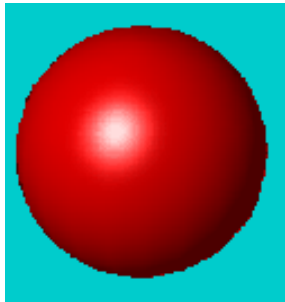
Sutherland, Sketchpad 1959



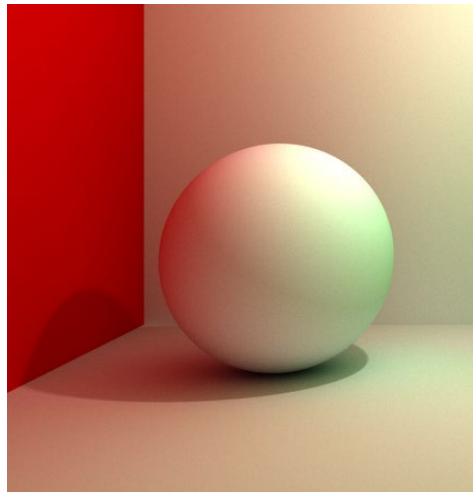
Hidden Surface Removal



Flat Shading



Phong Shading



Global Illumination



Environment Mapping



# エンターテインメント分野におけるCGの例

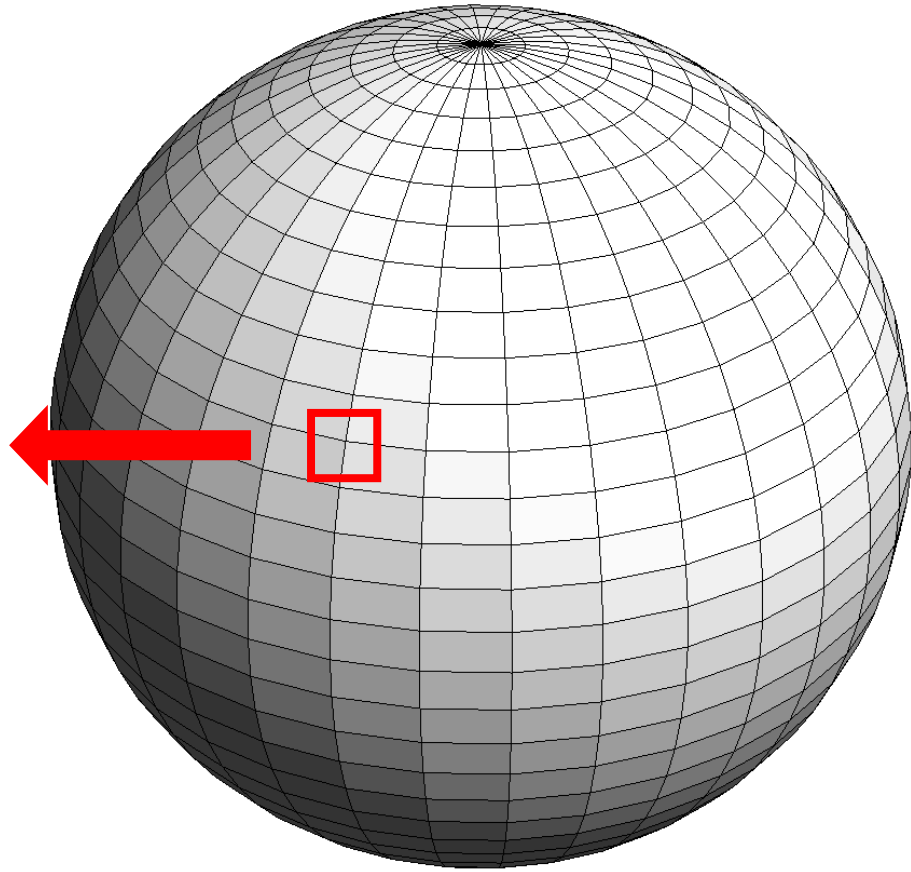
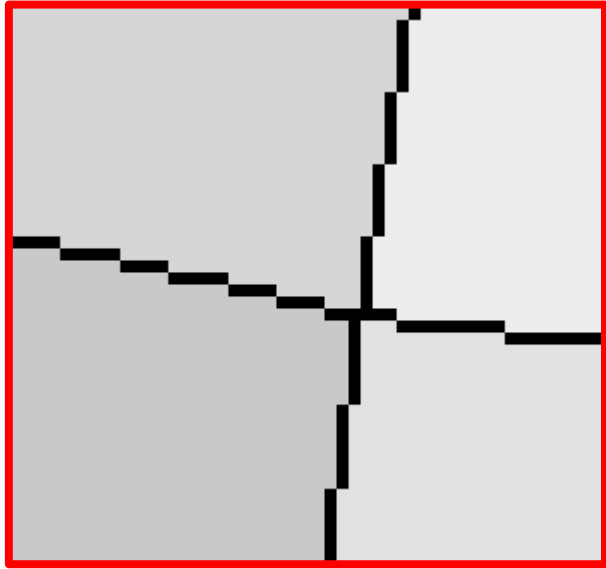
- ファイナルファンタジーVII (PS)
  - <https://www.youtube.com/watch?v=C11CupDUPeo>
- ファイナルファンタジーVII remake (PS4)
  - [https://www.youtube.com/watch?time\\_continue=13&v=FbYxZMGdXUc](https://www.youtube.com/watch?time_continue=13&v=FbYxZMGdXUc)
- 『シン・ゴジラ』 白組によるCGメイキング映像
  - <https://www.youtube.com/watch?v=66SAVZ4JxY8>



# そもそもCGとは？

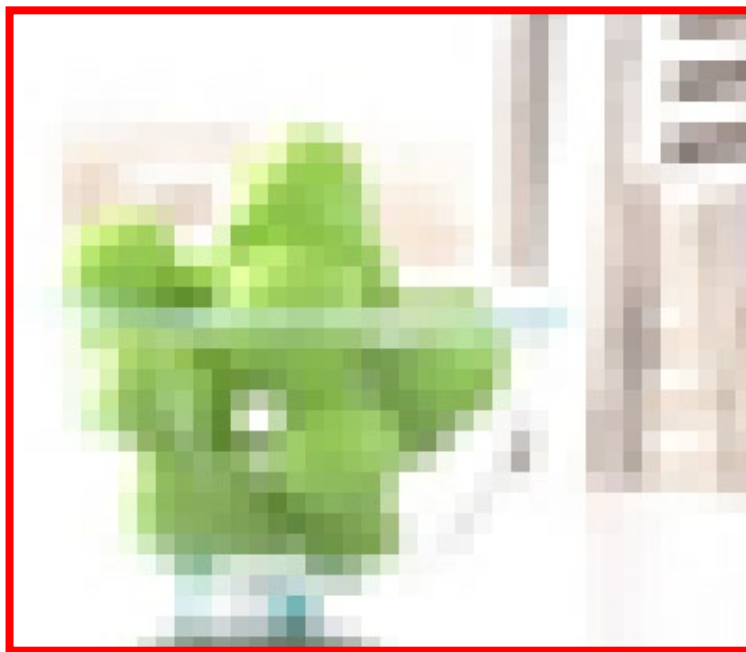


# そもそもCGとは？



# そもそもCGとは？

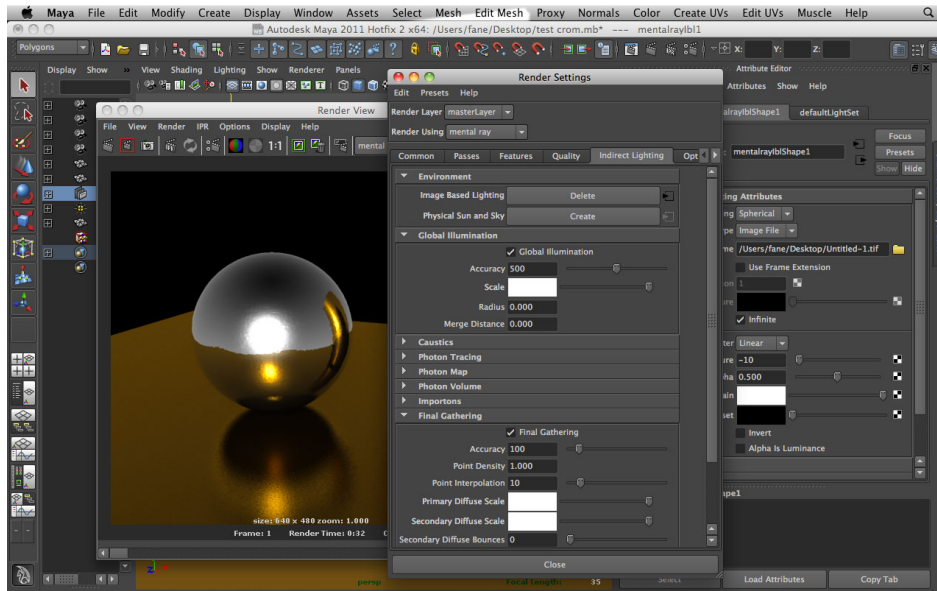
- 画面を構成する「画素」の色を決定すること
- フルHDの場合 1920x1080ピクセル x (R,G,B)
- 色の解像度は一般的に  
 $(R,G,B) = (0 \sim 255, 0 \sim 255, 0 \sim 255)$



# どうやって画素の色を決定するか

- 自分のプログラムで計算する
- ある程度は既存のライブラリを活用する
- × 便利なソフトウェアを利用する

この  
授業で  
扱う





# 大学で学習する一般的なプログラムの例

## C 言語

```
#include <stdio.h>

int main(int argc, char *args[]) {
    printf("Hello, world!¥n");
    return 0;
}
```

## Java 言語

```
public class Example {
    public static void main(String[] args){
        System.out.println("Hello, world!");
    }
}
```



文字列がコンソールに出力される

```
> Hello, world!
```

図形を表示するには？



# OpenGL ライブラリ



- グラフィックライブラリの一つ
  - グラフィックス用の便利な関数が準備されている
  - 関数を実行することで、図形を画面に表示できる
  - どのような関数があるのか、どのように使用するかを覚える（いつでも調べられる）必要がある
- Windows, Linuxなどでソースレベルの互換性、幅広く使用されている（iPhoneアプリの開発でも）
- ウィンドウ、入出力処理をサポートしていない  
（GLUT ライブラリを併用することで対処）



# GLUT ライブラリ

- OpenGL に備わっていない補助機能を備えたライブラリ
  - ウィンドウ表示
  - キーボード、マウスなどの入力処理
- OpenGL を便利に使うための機能も持つ



# 本日の目標

- OpenGL を用いた C 言語のプログラムをコンパイルして実行する
- OpenGLを用いたプログラムの構成を理解する  
(ある程度はそのようなものであると、丸のみせざるを得ない点はある)
- サンプルプログラムを動作させ、パラメータを変更して結果の違いを確認する



# OpenGL + GLUT を用いたプログラムコードの例 ウィンドウの表示

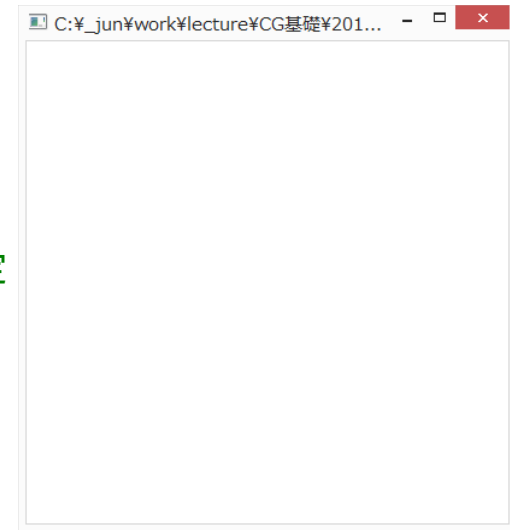
```
#include <GL/glut.h> // ライブラリ用ヘッダファイルの読み込み

// 表示部分をこの関数で記入
void display(void) {
    glClearColor (1.0, 1.0, 1.0, 1.0); // 消去色指定
    glClear (GL_COLOR_BUFFER_BIT );    // 画面消去

    /* ここに描画に関するプログラムコードを入れる */

    glFlush(); // 画面出力
}

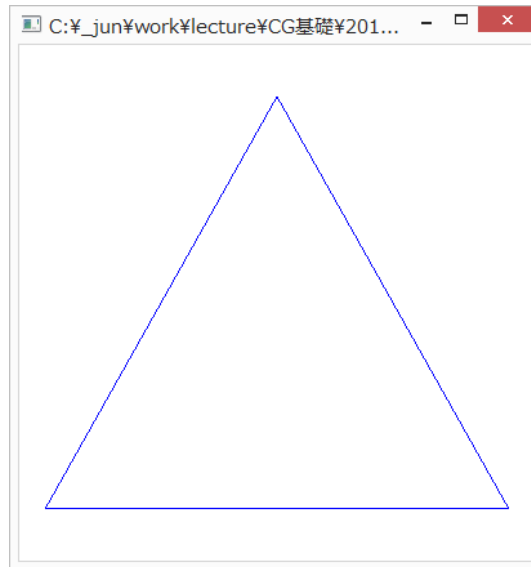
// メインプログラム
int main (int argc, char *argv[]) {
    glutInit(&argc, argv); // ライブラリの初期化
    glutInitWindowSize(400 , 400); // ウィンドウサイズを指定
    glutCreateWindow(argv[0]); // ウィンドウを作成
    glutDisplayFunc(display); // 表示関数を指定
    glutMainLoop(); // イベント待ち
    return 0;
}
```



/\* ここに描画に関するプログラムコードを入れる \*/



```
glColor3d(0.0, 0.0, 1.0); // 色指定(R,G,B)で0~1まで  
glBegin(GL_LINE_LOOP);    // 描画するものを指定  
    glVertex2d(-0.9, -0.8); // 頂点位置の指定(1つめ)  
    glVertex2d( 0.9, -0.8); // 頂点位置の指定(2つめ)  
    glVertex2d( 0.0,  0.8); // 頂点位置の指定(3つめ)  
glEnd();
```



# OpenGLの基本：図形の描画

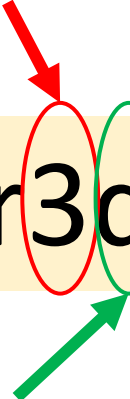
```
glColor3d(■, ■, ■);           // 色を指定する  
glBegin (GL_■■■);              // 何を描画するか指定する  
glVertex2d(x 座標, y 座標);    // 頂点の位置を指定する  
glVertex2d(x 座標, y 座標);    // 頂点の位置を指定する  
// glVertex2d を 必要なだけ繰り返す
```

```
glEnd();
```

GL_POINTS	// 点
GL_LINES	// 線
GL_LINE_STRIP	// 折れ線
GL_LINE_LOOP	// 多角形
GL_TRIANGLES	// 三角形
GL_QUADS	// 四角形



Red, Green, Blueの3つで指定するという意味  
(glColor4d で透明度を含めた指定もできる)



```
glColor3d(1.0, 1.0, 1.0);
```

「double型の値で指定する」  
という意味

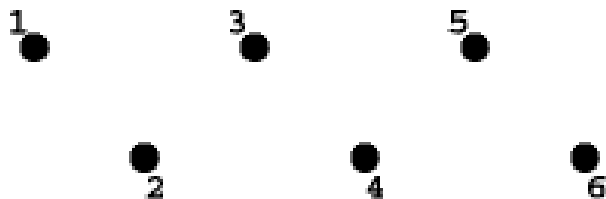
glColor3f なら float型

指定できる値は 0.0 ~ 1.0

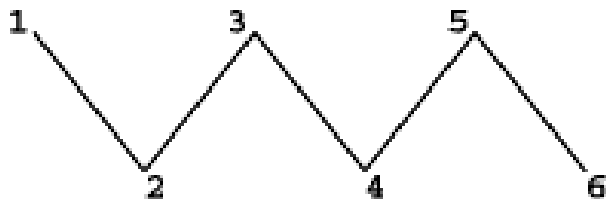




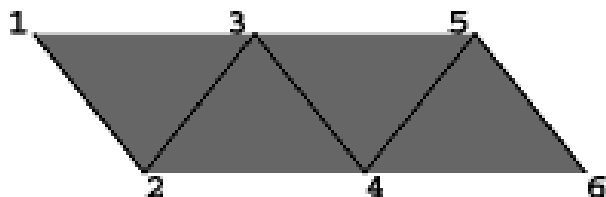
GL\_POINTS



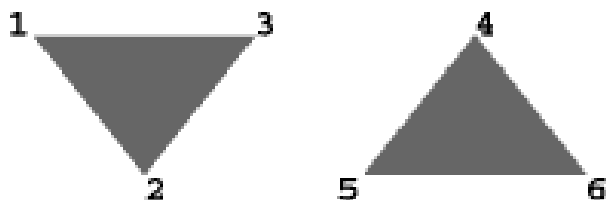
GL\_LINE\_STRIP



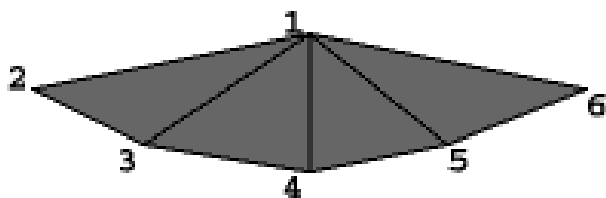
GL\_TRIANGLE\_STRIP



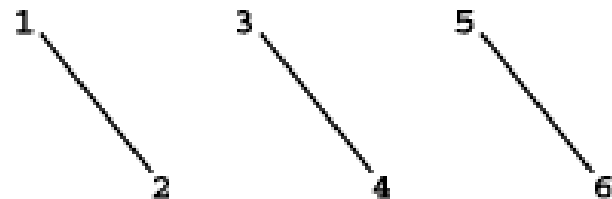
GL\_TRIANGLES



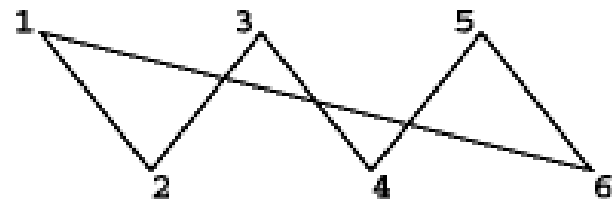
GL\_TRIANGLE\_FAN



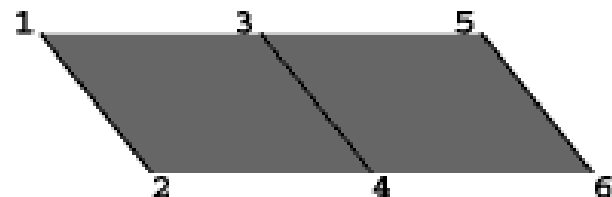
GL\_LINES



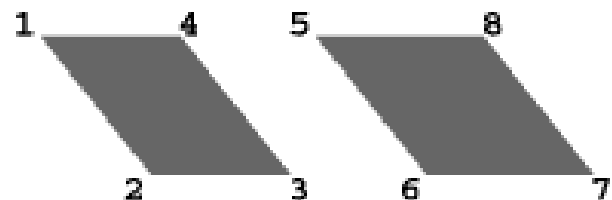
GL\_LINE\_LOOP



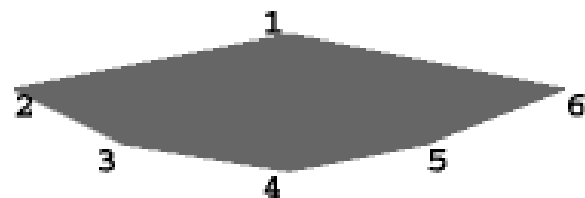
GL\_QUAD\_STRIP



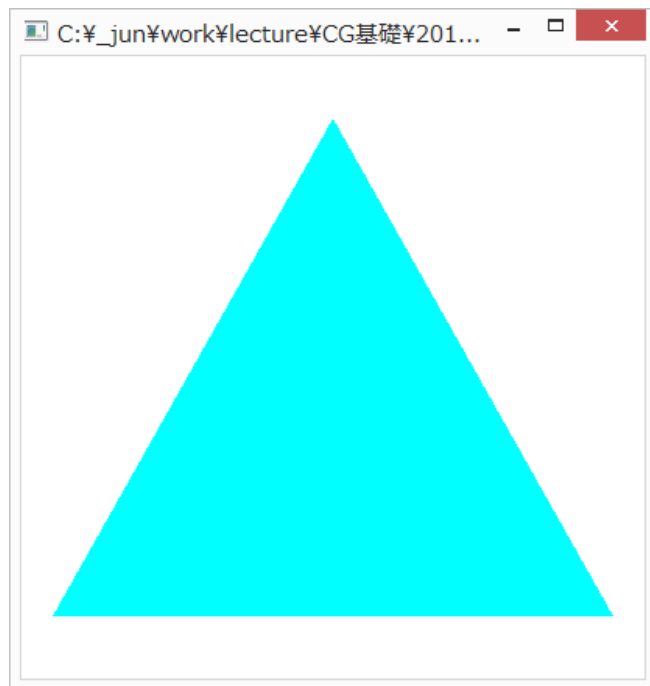
GL\_QUADS



GL\_POLYGON



```
glColor3d(0.0, 1.0, 1.0); // 色指定(R,G,B)で0~1まで  
glBegin(GL_TRIANGLES); // 描画するものを指定  
    glVertex2d(-0.9, -0.8); // 頂点位置の指定(1つめ)  
    glVertex2d( 0.9, -0.8); // 頂点位置の指定(2つめ)  
    glVertex2d( 0.0,  0.8); // 頂点位置の指定(3つめ)  
glEnd();
```



メモ :

円を描くコマンドは存在しない！  
折れ線や多角形で近似する。



「2次元座標」という意味

(glVertex3d で 3次元座標を指定することもできる)

glVertex2d(x 座標, y 座標);

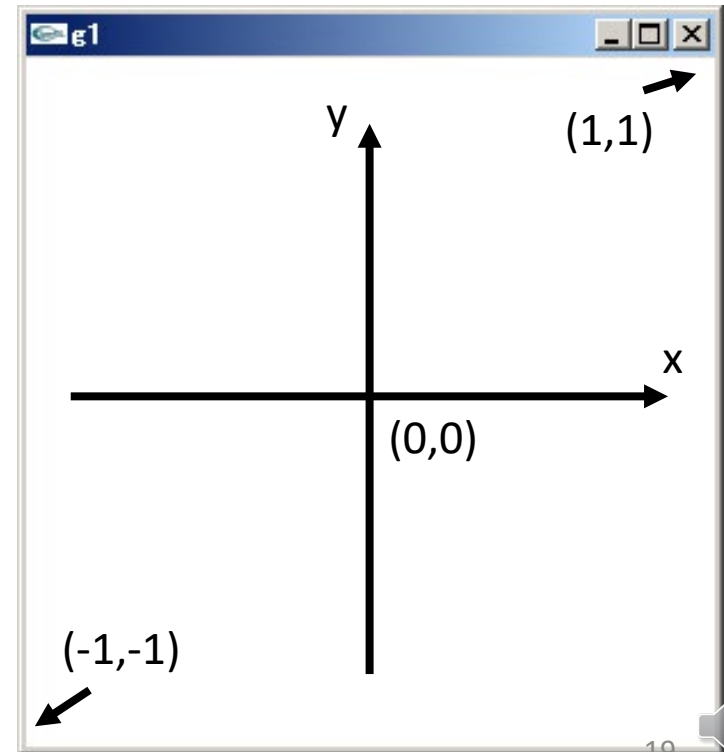
「double型の値で指定する」  
という意味

glVertex2f なら float型

glVertex2i なら int型

glVertex2dv なら double 型の配列

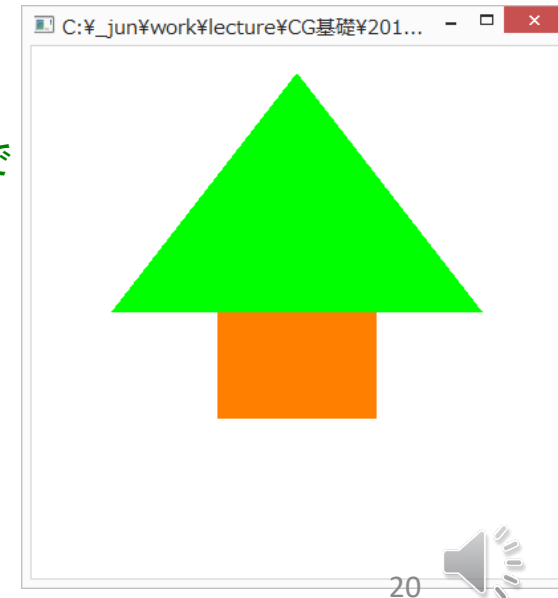
glVertex2fv なら float型の配列



# 複数の図形を描く

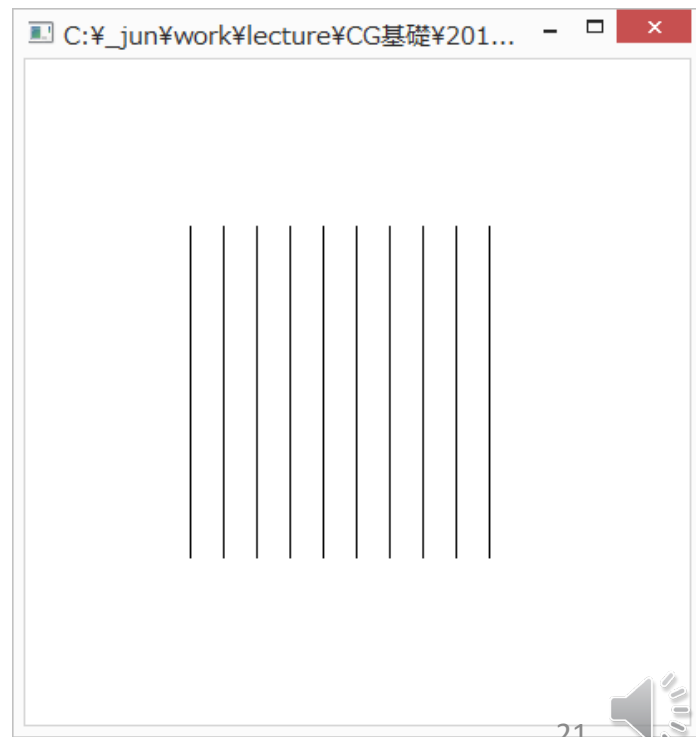
// 表示部分をこの関数で記入

```
void display(void) {  
    glClearColor (1.0, 1.0, 1.0, 1.0); // 消去色指定  
    glClear (GL_COLOR_BUFFER_BIT );     // 画面消去  
  
    // 1つ目の図形  
    glColor3d(1.0, 0.5, 0.0); // 色指定(R,G,B)で0~1まで  
    glBegin(GL_QUADS);        // 描画するものを指定  
        glVertex2d(-0.3, 0.0); // 頂点位置の指定(1つめ)  
        glVertex2d(-0.3, -0.4); // 頂点位置の指定(2つめ)  
        glVertex2d( 0.3, -0.4); // 頂点位置の指定(3つめ)  
        glVertex2d( 0.3, 0.0); // 頂点位置の指定(4つめ)  
    glEnd();  
  
    // 2つ目の図形  
    glColor3d(0.0, 1.0, 0.0); // 色指定(R,G,B)で0~1まで  
    glBegin(GL_TRIANGLES);     // 描画するものを指定  
        glVertex2d( 0.0, 0.9); // 頂点位置の指定(1つめ)  
        glVertex2d(-0.7, 0.0); // 頂点位置の指定(2つめ)  
        glVertex2d( 0.7, 0.0); // 頂点位置の指定(3つめ)  
    glEnd();  
  
    glFlush(); // 画面出力  
}
```



# ループ処理と組み合わせる

```
glColor3d(0.0, 0.0, 0.0);    // 色指定
glBegin(GL_LINES);
for(int i = 0; i < 10; i++) {
    glVertex2d(i*0.1 - 0.5, 0.5);
    glVertex2d(i*0.1 - 0.5, -0.5);
}
glEnd();
```



# ループ処理と組み合わせる

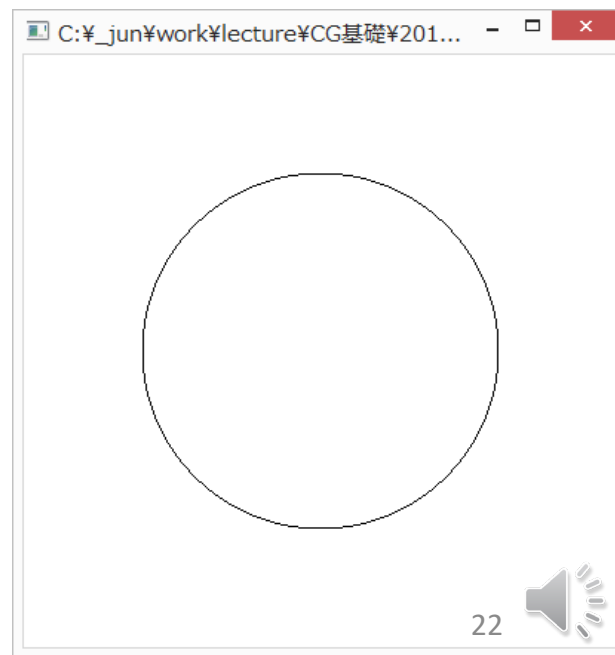
```
#include <math.h>
```

← 三角関数を使うためプログラム冒頭に入れておく

```
glColor3d(0.0, 0.0, 0.0);  
glBegin(GL_LINE_LOOP);  
for(int i = 0; i < 360; i++) {  
    double x = cos(i * 3.14159 / 180.0);  
    double y = sin(i * 3.14159 / 180.0);  
    glVertex2d(x * 0.6, y * 0.6);  
}  
glEnd();
```

- C++ なら `math.h` の代わりに `cmath` をインクルードすることが多い
- 円周率を表す定数マクロ `M_PI` を Visual Studio で使いたければ

```
#define _USE_MATH_DEFINES  
#include <math.h>    // または #include <cmath>
```



2次元



これだけで、自由な図形が描ける！！！！



# 課題

- サンプルコードをコンパイルして実行してみる
- コード中の数字を変えて、結果がどのように変化するか確認する
- 自由にいろいろな図形を表示させてみる ← 提出
- 「コッホ曲線」を描いてみる ← 任意
- 詳しくは授業のページの「第1回 課題」を参照のこと





## コンピュータグラフィックス基礎 第1回 課題

### 課題の目標

- ・ OpenGL を用いたプログラムをコンパイル、実行できるようになる
- ・ 簡単な 2 次元図形を描画するプログラムを作成できる

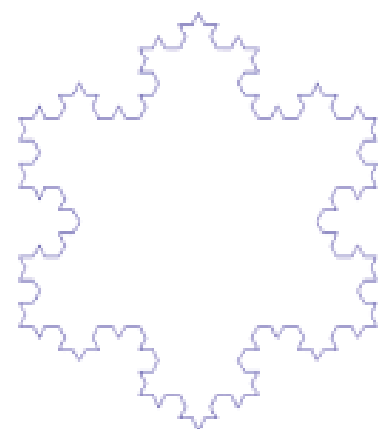
### 課題の内容

1. 授業用 Web ページにある、それぞれのサンプルコードを実行し、プログラムコードと実行結果の様子を観察しなさい。
2. オリジナルの 2 次元図形を画面に表示するプログラムを作成しなさい (できるだけ綺麗な、または楽しい図形を描こう)。プログラムの中では必ず 1 度は `for` ループを用いること。



[発展課題] (必須ではない)

「コッホ曲線」について調べ、右図のような図形を描画するプログラムを作成しなさい (異なる詳細度で描画できればなおよい)

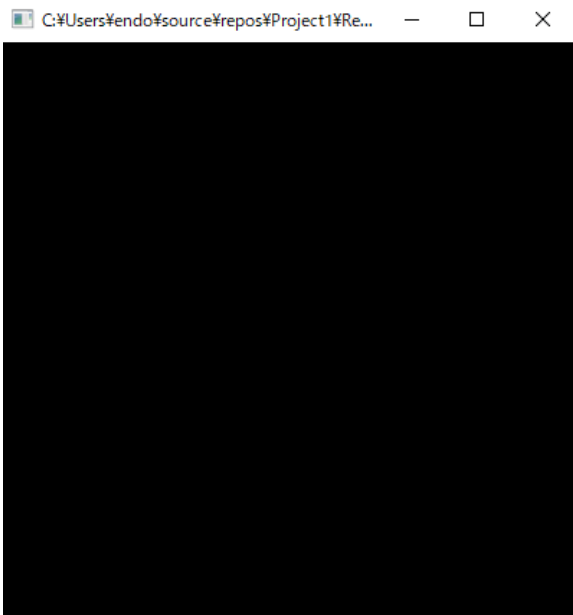


提出するレポートに含めるもの

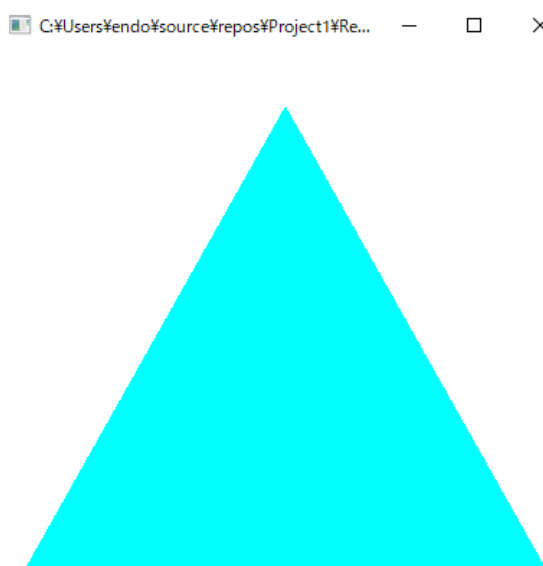
- ・ 上記の課題の 2 に相当するプログラムコードと、実行結果のウィンドウをキャプチャした図 (課題 2 は、最低 1 つ提出すればよいが、異なる複数のプログラムとその結果を含めてよい)。
- ・ 余力があれば、発展課題にも挑戦すること。



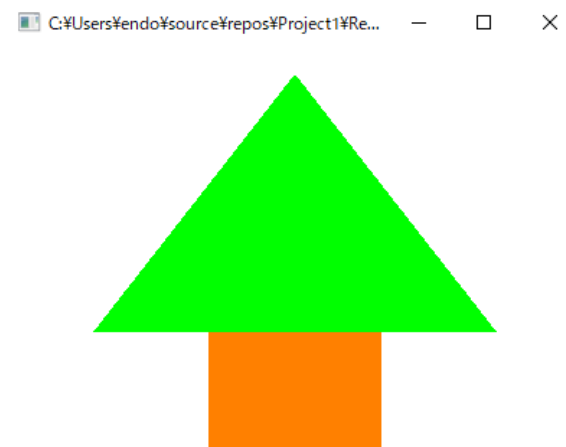
# サンプルコードの実行例



[01 simple window.cpp](#)



[03 draw triangle.cpp](#)



[04 draw figures.cpp](#)



# サンプルコードの注意点

- 拡張子を cpp にしている  
C++ 言語の便利なところを使いたいため
- OpenGLの古典的な使い方をしている  
最新のゲーム開発などで使われている  
OpenGL プログラムコードとは異なる  
(教育的観点から)



# 課題の提出について

- 提出期限: 次の授業の開始まで（期限を過ぎての提出は減点対象です）
- 提出方法：課題は Word ファイルにまとめて manaba で提出
  - プログラムコード、実験結果がわかる画像（スクリーンショットなど）
  - （もしあれば）コメント、感想など
  - プログラムコードはレポートと別々に提出するのではなく、レポートにテキストとしてコピー＆ペーストしてください(プログラムコードのスクリーンショットは不可)
  - 課題サンプルコードはWindows 上 Visual Studio で動作確認している、それ以外の環境で動作確認したものを提出しても構わないがレポートに環境を記載すること

