メディアデザインセミナー１

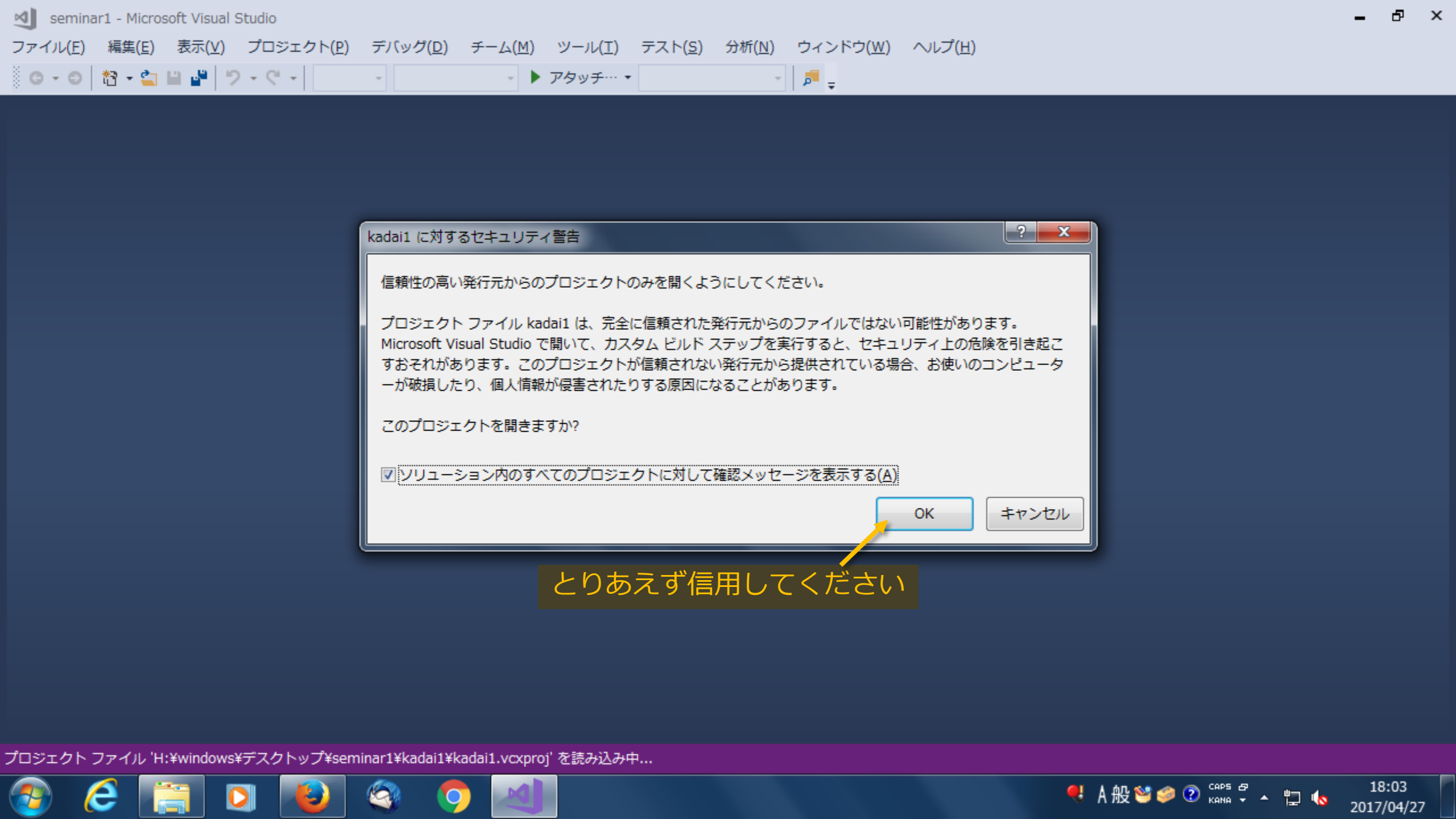
視覚メディア研究室

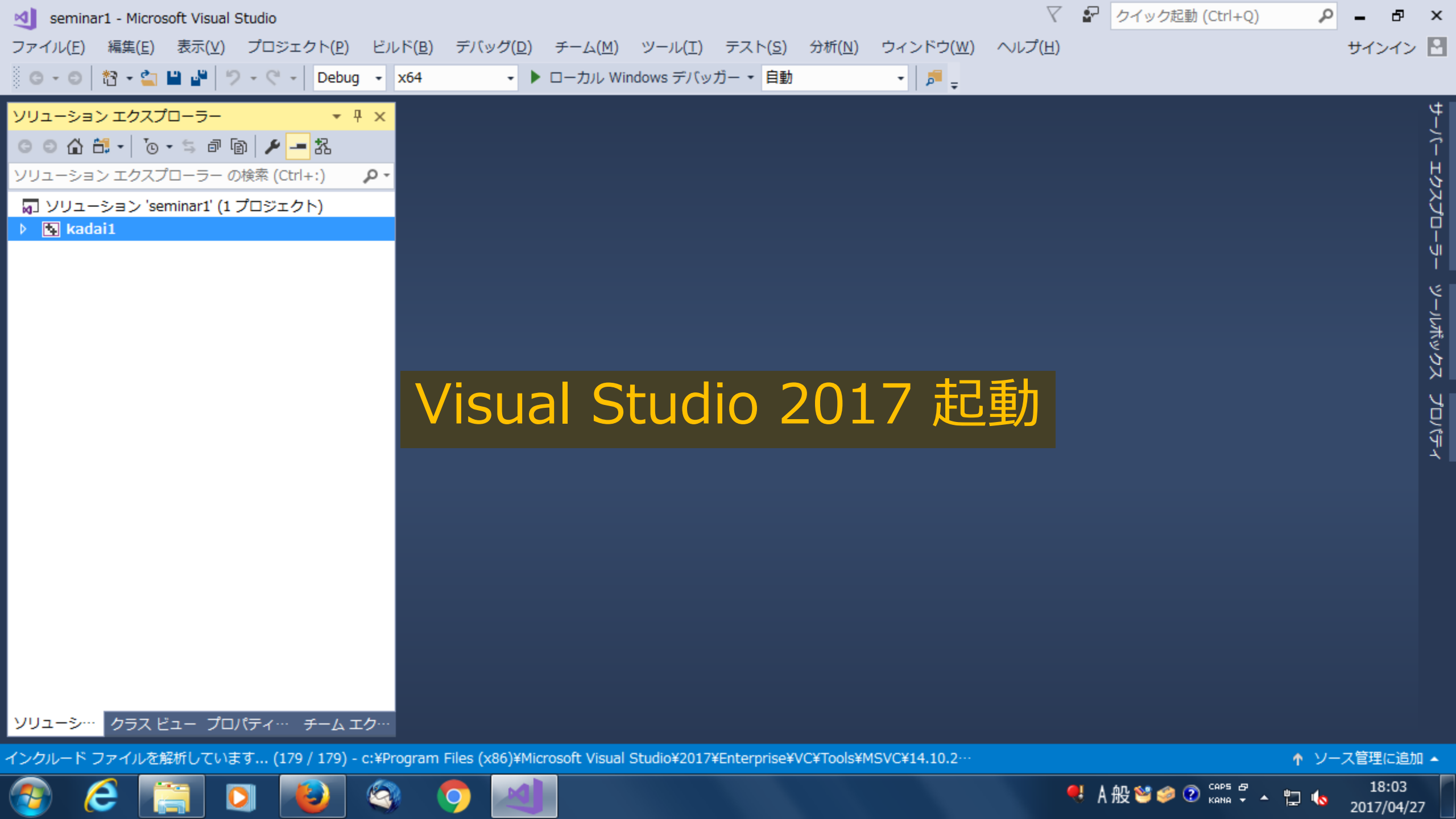
内容

- Visual Studio を用いた C++ プログラミング
- OpenCV によるビデオ入力
- OpenGL による図形表示
  - だだし直接 OpenGL のプログラミングは行いません

Visual Studio 2017 の起動

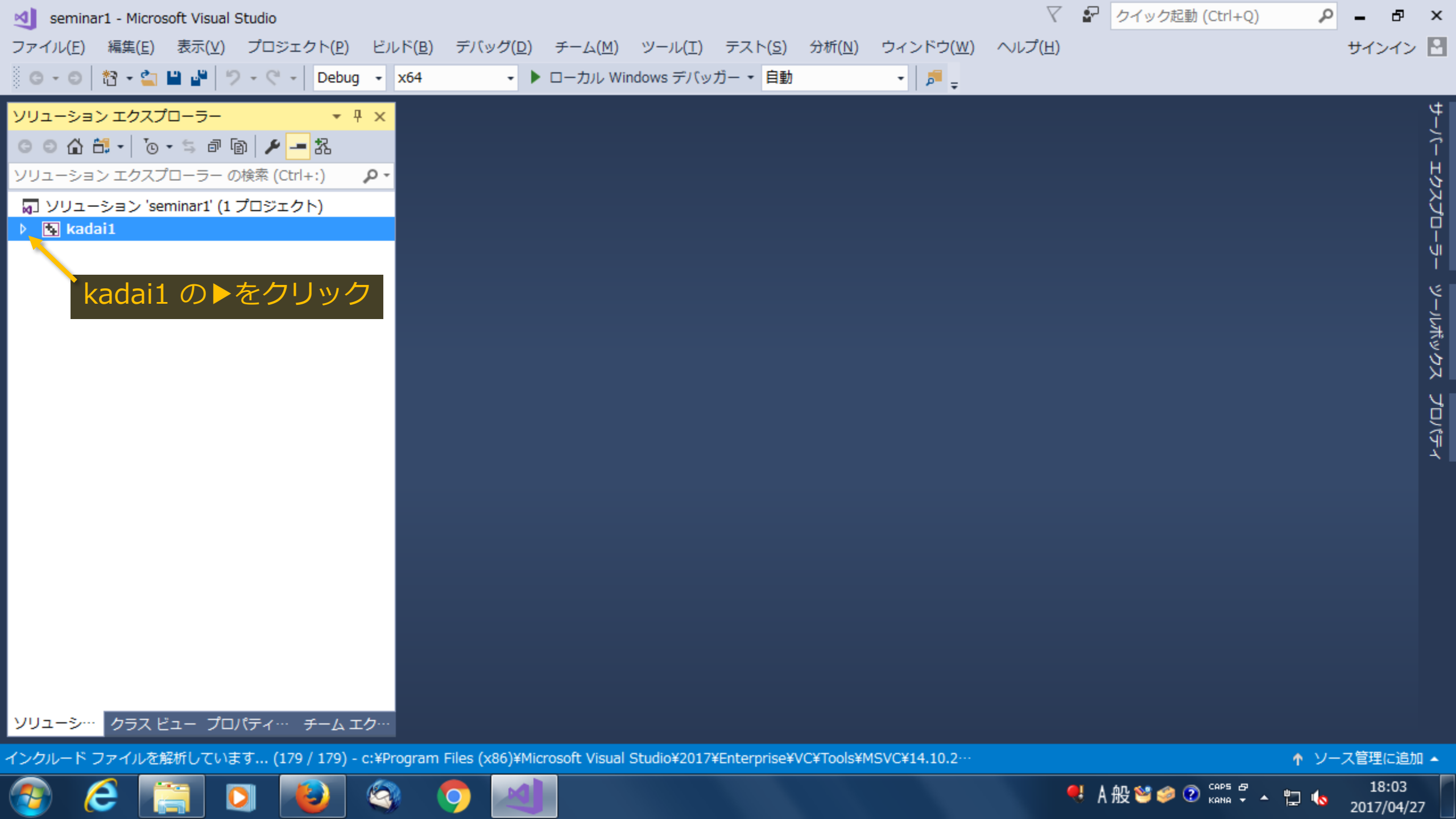
ソリューションファイルを開く

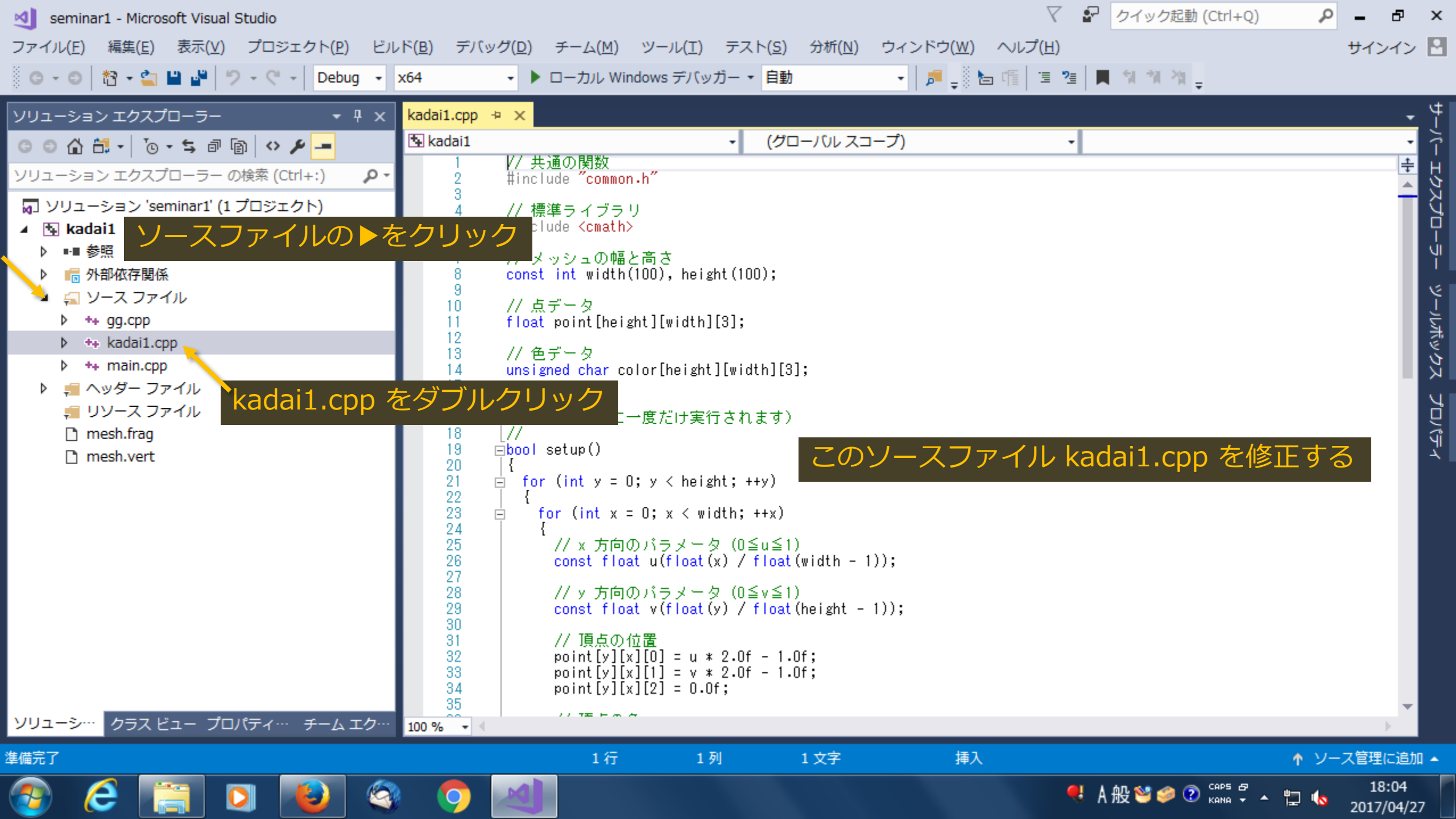

seminar1 - Microsoft Visual Studio
ファイル(F) 編集(E) 表示(V) プロジェクト(P) デバッグ(D) チーム(M) ツール(T) テスト(S) 分析(N) ウィンドウ(W) ヘルプ(H)
アタッチ...
kadai1 に対するセキュリティ警告
信頼性の高い発行元からのプロジェクトのみを開くようにしてください。
プロジェクト ファイル kadai1 は、完全に信頼された発行元からのファイルではない可能性があります。Microsoft Visual Studio で開いて、カスタム ビルド ステップを実行すると、セキュリティ上の危険を引き起こすおそれがあります。このプロジェクトが信頼されない発行元から提供されている場合、お使いのコンピューターが破損したり、個人情報が侵害されたりする原因になることがあります。
このプロジェクトを開きますか?
ソリューション内のすべてのプロジェクトに対して確認メッセージを表示する(A)
OK
キャンセル
とりあえず信用してください
プロジェクト ファイル 'H:¥windows¥デスクトップ¥seminar1¥kadai1¥kadai1.vcxproj' を読み込み中...
18:03
2017/04/27

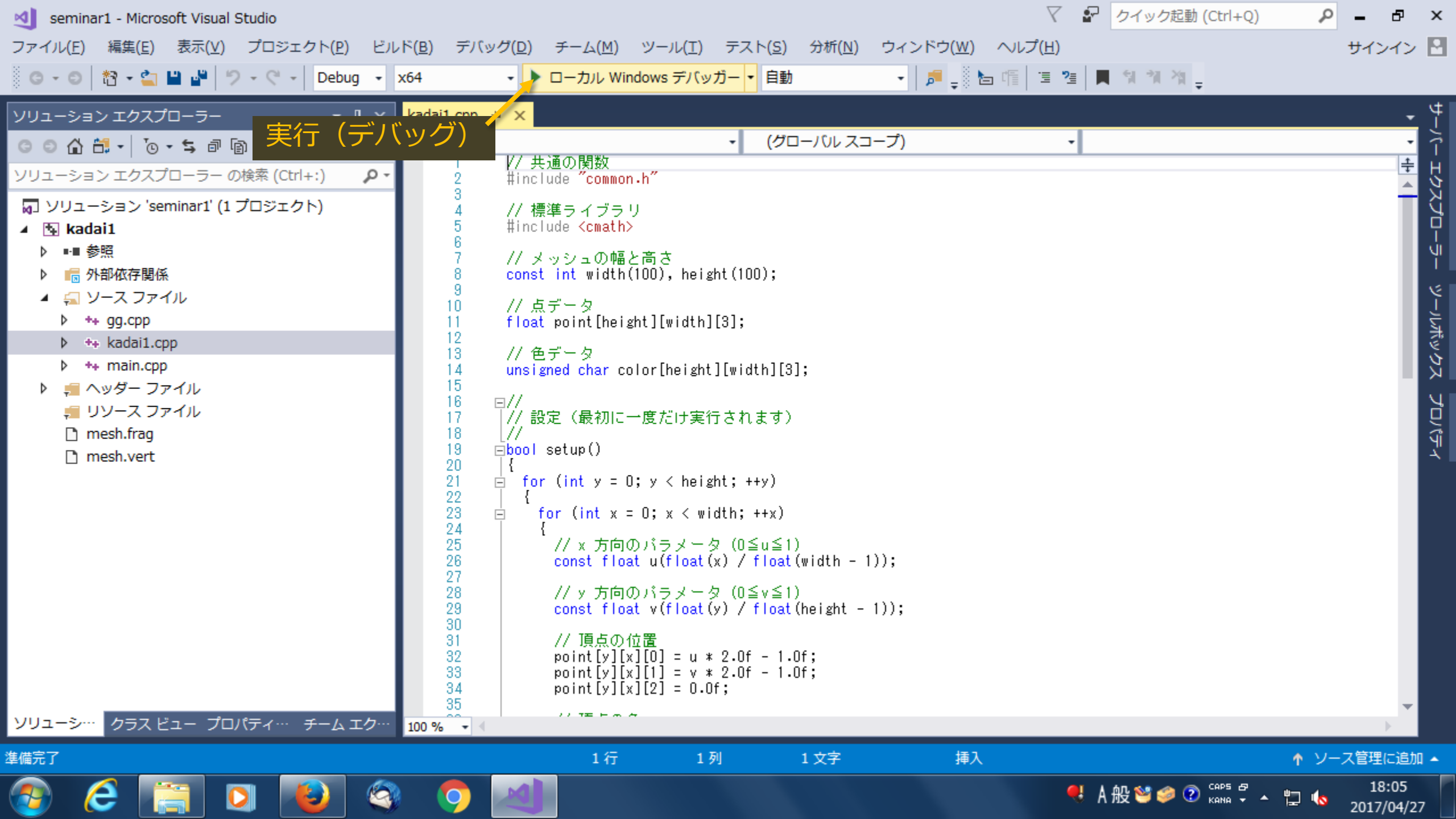

seminar1 - Microsoft Visual Studio
クイック起動 (Ctrl+Q)
ファイル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(B) デバッグ(D) チーム(M) ツール(T) テスト(S) 分析(N) ウィンドウ(W) ヘルプ(H)
サインイン
Debug
x64
ローカル Windows デバッガー
自動
ソリューション エクスプローラー
ソリューション エクスプローラー の検索 (Ctrl+:)
ソリューション 'seminar1' (1 プロジェクト)
kadai1
ソリューシ…
クラス ビュー
プロパティ…
チーム エク…
サーバー エクスプローラー
ツールボックス
プロパティ
Visual Studio 2017 起動
インクルード ファイルを解析しています... (179 / 179) - c:¥Program Files (x86)¥Microsoft Visual Studio¥2017¥Enterprise¥VC¥Tools¥MSVC¥14.10.2…
ソース管理に追加
A 般
CAPS
KANA
18:03
2017/04/27

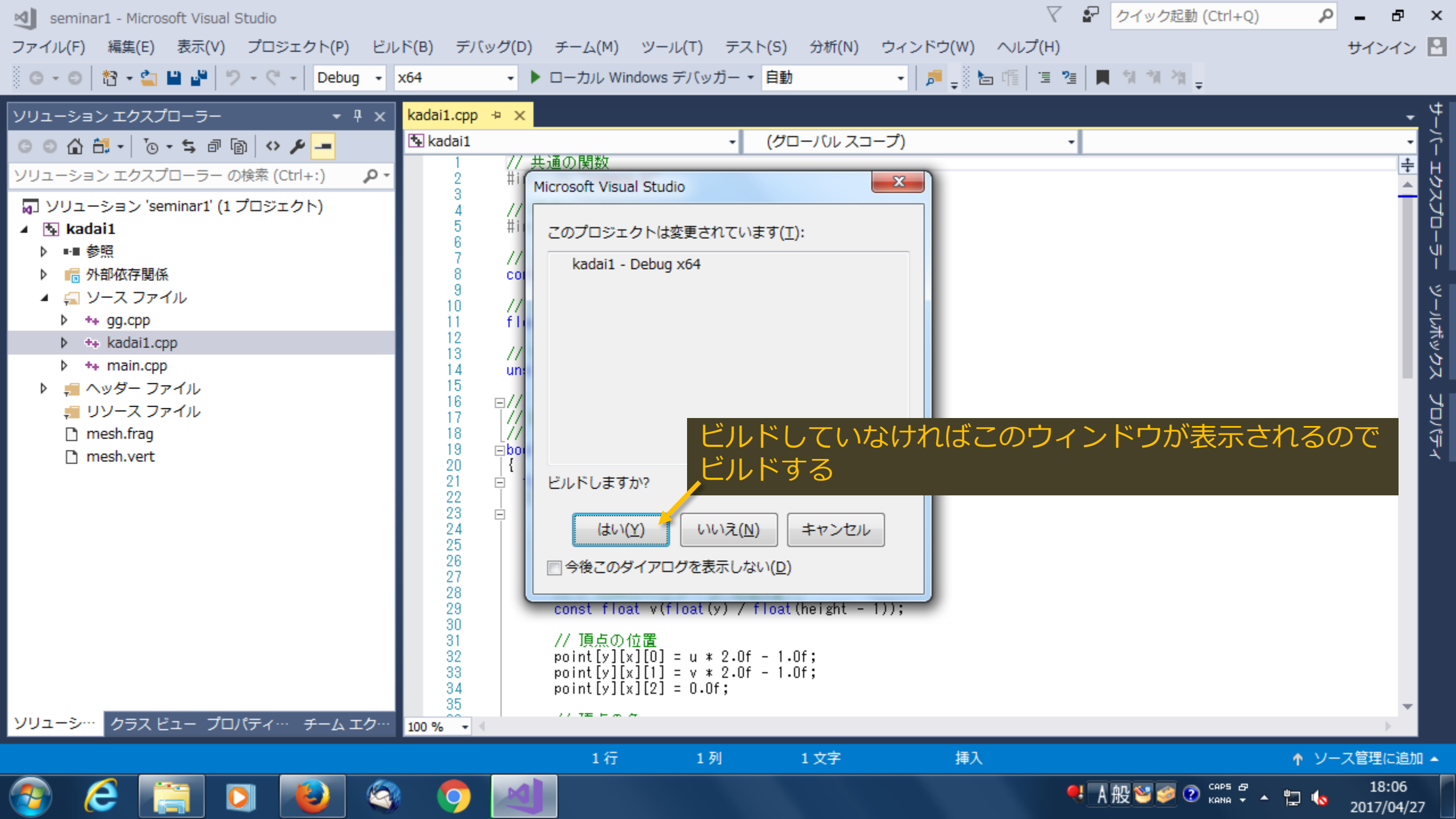
プロジェクトのビルドと実行

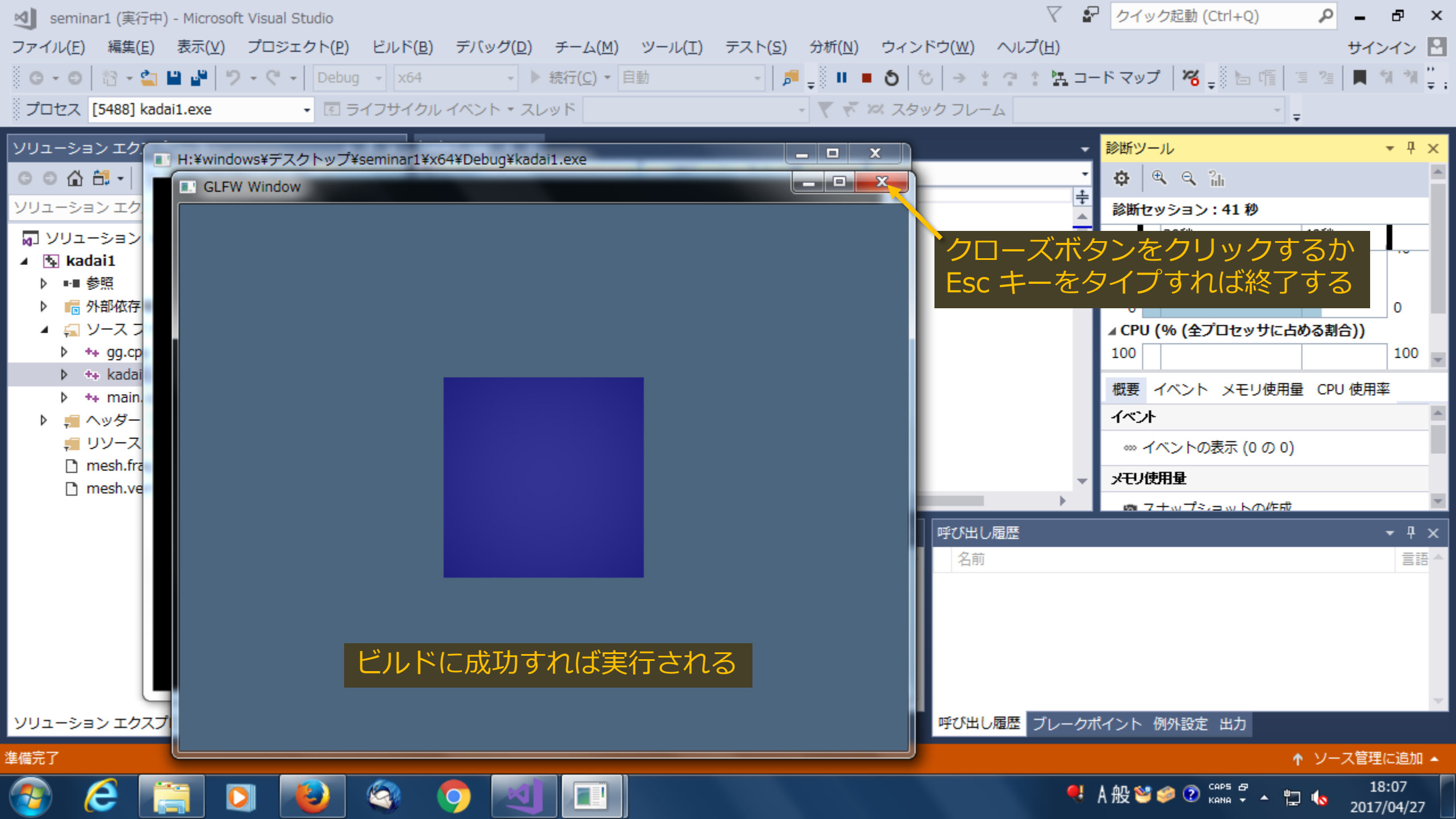
作成したプログラムを動かす

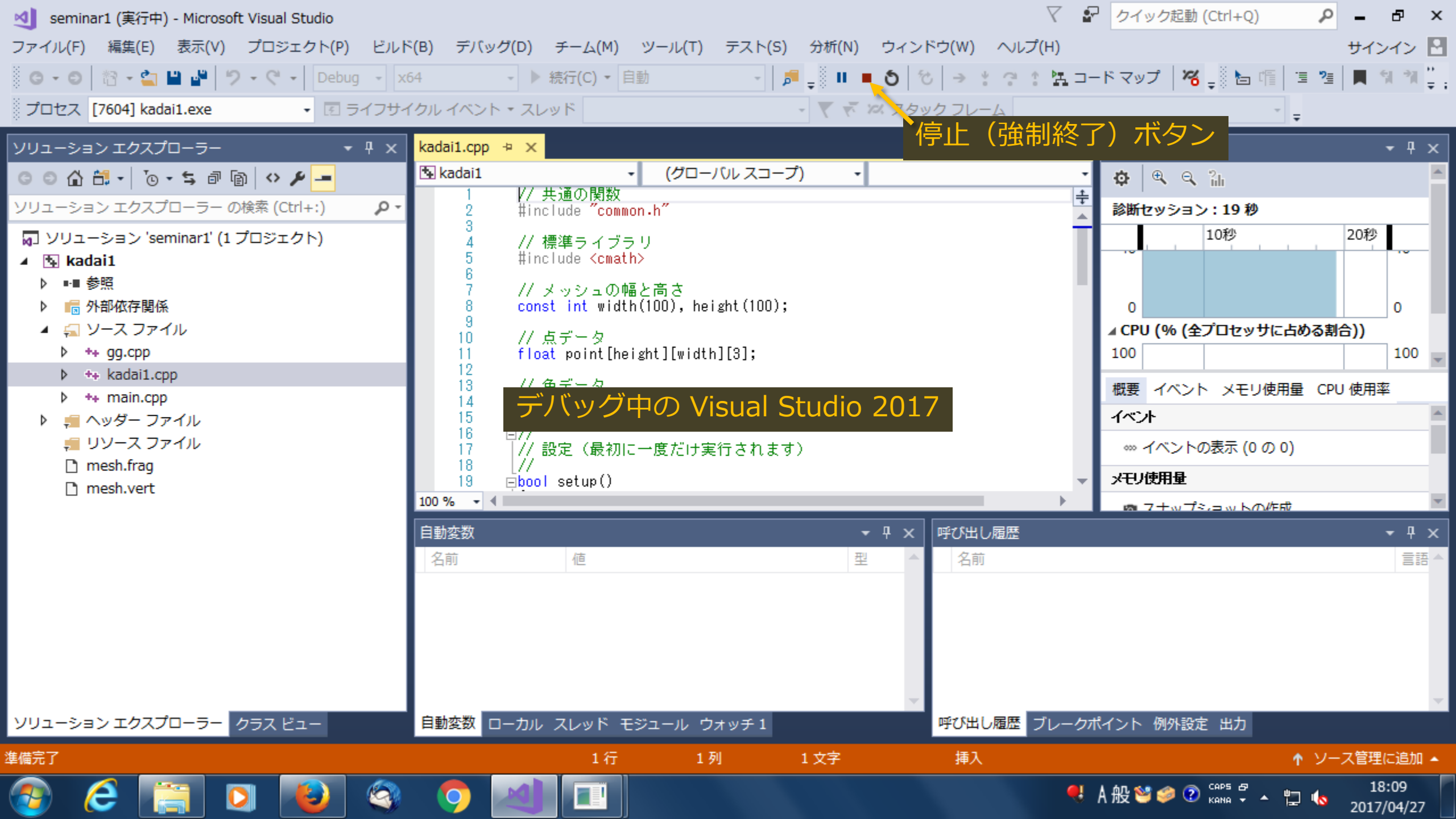

seminar1 - Microsoft Visual Studio
クイック起動 (Ctrl+Q)
ファイル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(B) デバッグ(D) チーム(M) ツール(T) テスト(S) 分析(N) ウィンドウ(W) ヘルプ(H)
サインイン
Debug
x64
ローカル Windows デバッガー
自動
ソリューション エクスプローラー
ソリューション エクスプローラー の検索 (Ctrl+:)
ソリューション 'seminar1' (1 プロジェクト)
kadai1
kadai1 の▶をクリック
ソリューシ…
クラス ビュー
プロパティ…
チーム エク…
サーバー エクスプローラー
ツールボックス
プロパティ
インクルード ファイルを解析しています... (179 / 179) - c:¥Program Files (x86)¥Microsoft Visual Studio¥2017¥Enterprise¥VC¥Tools¥MSVC¥14.10.2…
ソース管理に追加
A 般
CAPS
KANA
18:03
2017/04/27

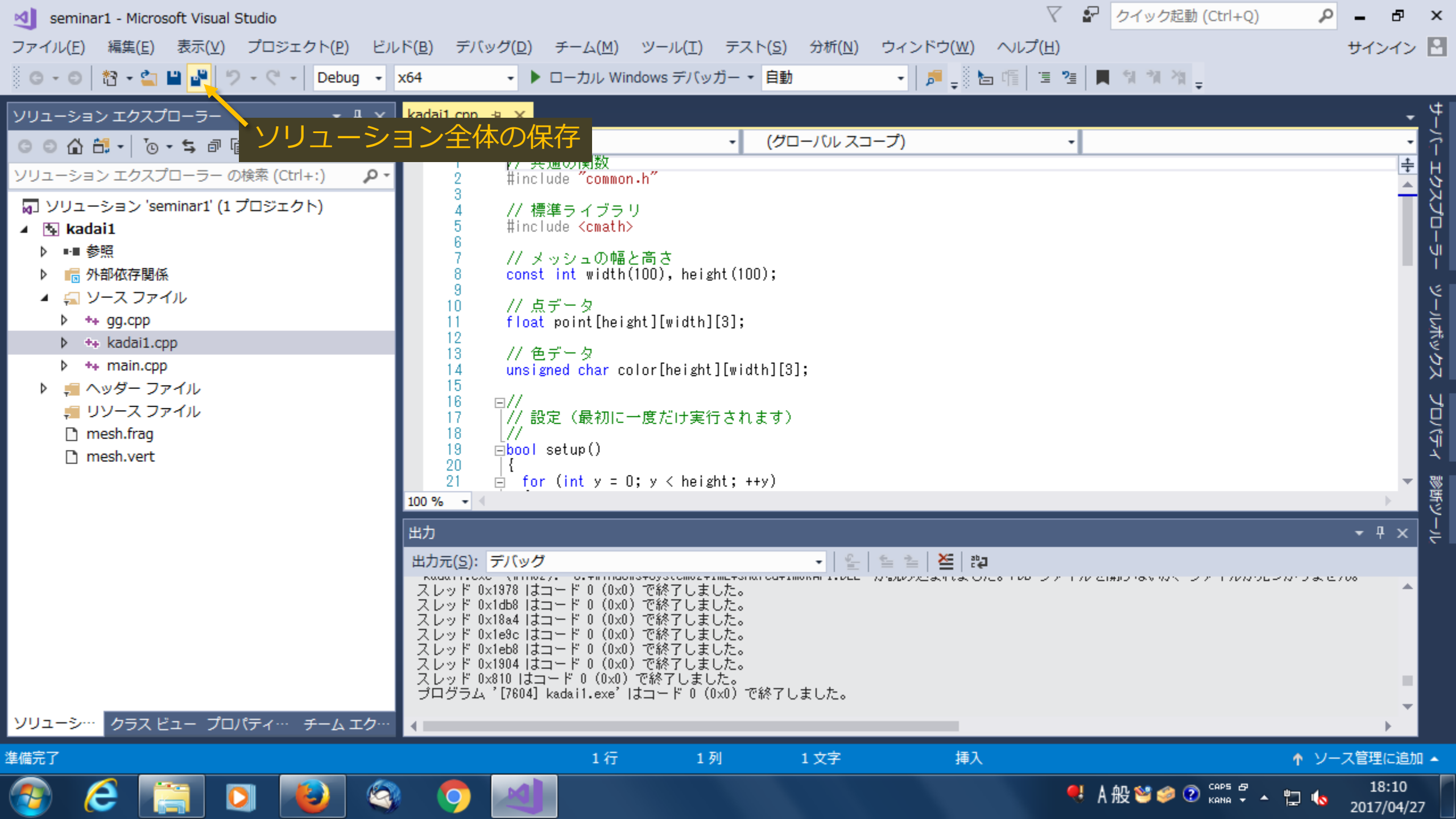

seminar1 - Microsoft Visual Studio
ファイル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(B) デバッグ(D) チーム(M) ツール(T) テスト(S) 分析(N) ウィンドウ(W) ヘルプ(H)
Debug
x64
ローカル Windows デバッガー
自動
ソリューション エクスプローラー
ソリューション エクスプローラー の検索 (Ctrl+:)
ソリューション 'seminar1' (1 プロジェクト)
kadai1
参照
外部依存関係
ソース ファイル
gg.cpp
kadai1.cpp
main.cpp
ヘッダー ファイル
リソース ファイル
mesh.frag
mesh.vert
ソースファイルの▶をクリック
kadai1.cpp をダブルクリック
kadai1.cpp
kadai1
(グローバル スコープ)
// 共通の関数
#include "common.h"
// 標準ライブラリ
clude <cmath>
// メッシュの幅と高さ
const int width(100), height(100);
// 点データ
float point[height][width][3];
// 色データ
unsigned char color[height][width][3];
一度だけ実行されます)
//
bool setup()
{
for (int y = 0; y < height; ++y)
{
for (int x = 0; x < width; ++x)
{
// x 方向のパラメータ (0≦u≦1)
const float u(float(x) / float(width - 1));
// y 方向のパラメータ (0≦v≦1)
const float v(float(y) / float(height - 1));
// 頂点の位置
point[y][x][0] = u * 2.0f - 1.0f;
point[y][x][1] = v * 2.0f - 1.0f;
point[y][x][2] = 0.0f;
このソースファイル kadai1.cpp を修正する
100 %
準備完了
1 行
1 列
1 文字
挿入
ソース管理に追加
サーバー エクスプローラー ツールボックス プロパティ
18:04
2017/04/27


seminar1 - Microsoft Visual Studio
クイック起動 (Ctrl+Q)
ファイル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(B) デバッグ(D) チーム(M) ツール(T) テスト(S) 分析(N) ウィンドウ(W) ヘルプ(H)
サインイン
Debug
x64
ローカル Windows デバッガー
自動
実行（デバッグ）
ソリューション エクスプローラー
ソリューション エクスプローラー の検索 (Ctrl+:)
ソリューション 'seminar1' (1 プロジェクト)
kadai1
参照
外部依存関係
ソース ファイル
gg.cpp
kadai1.cpp
main.cpp
ヘッダー ファイル
リソース ファイル
mesh.frag
mesh.vert
kadai1.cpp
(グローバル スコープ)
// 共通の関数
#include "common.h"
// 標準ライブラリ
#include <cmath>
// メッシュの幅と高さ
const int width(100), height(100);
// 点データ
float point[height][width][3];
// 色データ
unsigned char color[height][width][3];
//
// 設定（最初に一度だけ実行されます）
//
bool setup()
{
for (int y = 0; y < height; ++y)
{
for (int x = 0; x < width; ++x)
{
// x 方向のパラメータ (0≦u≦1)
const float u(float(x) / float(width - 1));
// y 方向のパラメータ (0≦v≦1)
const float v(float(y) / float(height - 1));
// 頂点の位置
point[y][x][0] = u * 2.0f - 1.0f;
point[y][x][1] = v * 2.0f - 1.0f;
point[y][x][2] = 0.0f;
100 %
ソリューシ… クラス ビュー プロパティ… チーム エク…
サーバー エクスプローラー ツールボックス プロパティ
準備完了
1 行
1 列
1 文字
挿入
ソース管理に追加
18:05
2017/04/27


seminar1 - Microsoft Visual Studio
ファイル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(B) デバッグ(D) チーム(M) ツール(T) テスト(S) 分析(N) ウィンドウ(W) ヘルプ(H)
Debug
x64
ローカル Windows デバッガー
自動
ソリューション エクスプローラー
ソリューション 'seminar1' (1 プロジェクト)
kadai1
参照
外部依存関係
ソース ファイル
gg.cpp
kadai1.cpp
main.cpp
ヘッダー ファイル
リソース ファイル
mesh.frag
mesh.vert
kadai1.cpp
kadai1
(グローバル スコープ)
// 共通の関数
Microsoft Visual Studio
このプロジェクトは変更されています(T):
kadai1 - Debug x64
ビルドしますか?
はい(Y)
いいえ(N)
キャンセル
今後このダイアログを表示しない(D)
ビルドしていなければこのウィンドウが表示されるのでビルドする
const float v(float(y) / float(height - 1));
// 頂点の位置
point[y][x][0] = u * 2.0f - 1.0f;
point[y][x][1] = v * 2.0f - 1.0f;
point[y][x][2] = 0.0f;
100 %
1 行
1 列
1 文字
挿入
ソース管理に追加
18:06
2017/04/27


seminar1 (実行中) - Microsoft Visual Studio
H:¥windows¥デスクトップ¥seminar1¥x64¥Debug¥kadai1.exe
GLFW Window
クローズボタンをクリックするか
Esc キーをタイプすれば終了する
ビルドに成功すれば実行される

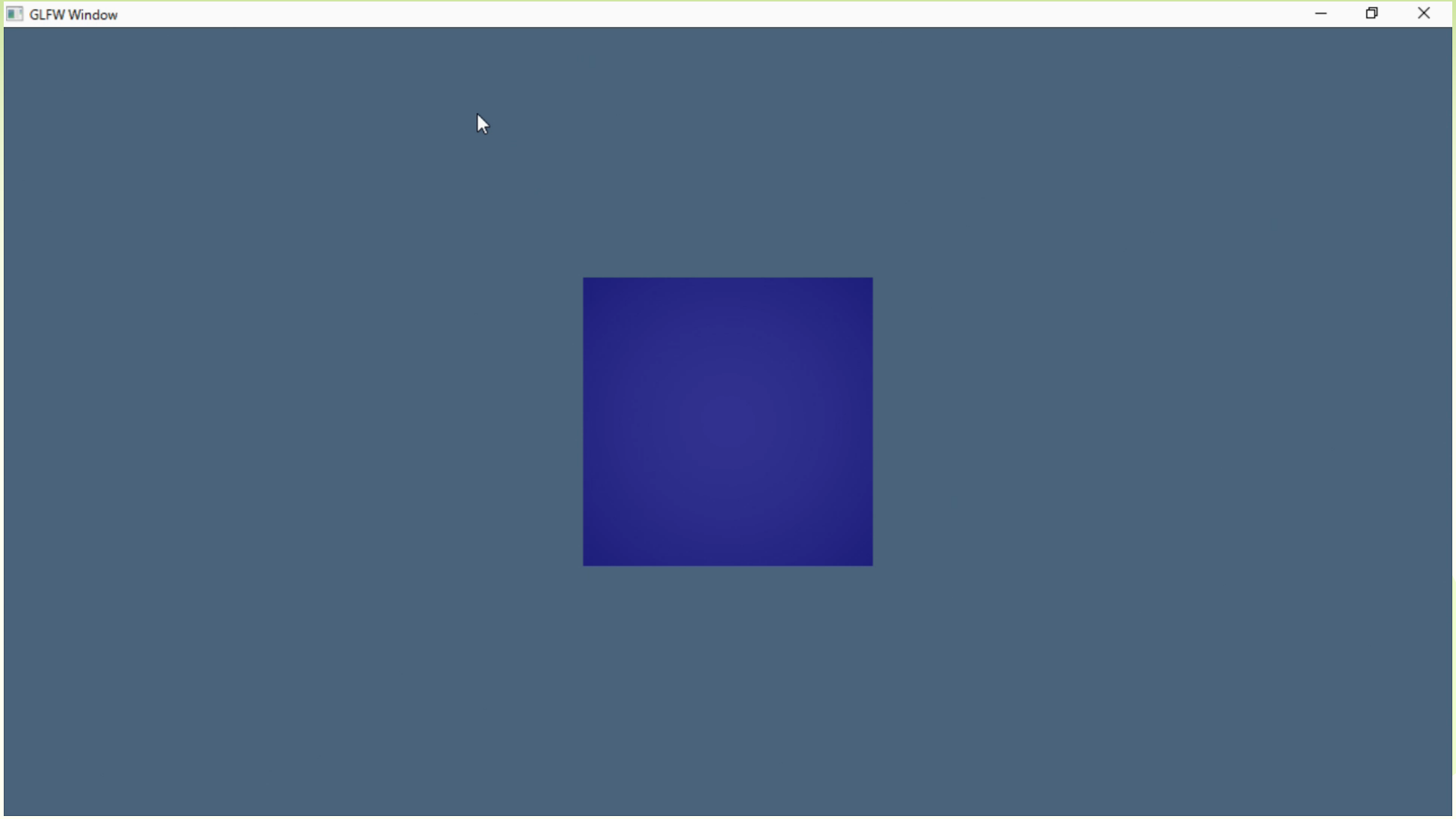

seminar1 (実行中) - Microsoft Visual Studio
停止（強制終了）ボタン
デバッグ中の Visual Studio 2017
kadai1.cpp
// 共通の関数
#include "common.h"
// 標準ライブラリ
#include <cmath>
// メッシュの幅と高さ
const int width(100), height(100);
// 点データ
float point[height][width][3];
// 色データ
//
// 設定（最初に一度だけ実行されます）
//
bool setup()
ソリューション エクスプローラー
ソリューション 'seminar1' (1 プロジェクト)
kadai1
参照
外部依存関係
ソース ファイル
gg.cpp
kadai1.cpp
main.cpp
ヘッダー ファイル
リソース ファイル
mesh.frag
mesh.vert
診断セッション：19 秒
CPU (% (全プロセッサに占める割合))
自動変数
呼び出し履歴
準備完了


seminar1 - Microsoft Visual Studio
クイック起動 (Ctrl+Q)
ファイル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(B) デバッグ(D) チーム(M) ツール(T) テスト(S) 分析(N) ウィンドウ(W) ヘルプ(H)
サインイン
Debug
x64
ローカル Windows デバッガー
自動
ソリューション全体の保存
ソリューション エクスプローラー
ソリューション エクスプローラー の検索 (Ctrl+:)
ソリューション 'seminar1' (1 プロジェクト)
kadai1
参照
外部依存関係
ソース ファイル
gg.cpp
kadai1.cpp
main.cpp
ヘッダー ファイル
リソース ファイル
mesh.frag
mesh.vert
kadai1.cpp
(グローバル スコープ)
// 共通の関数
#include "common.h"
// 標準ライブラリ
#include <cmath>
// メッシュの幅と高さ
const int width(100), height(100);
// 点データ
float point[height][width][3];
// 色データ
unsigned char color[height][width][3];
//
// 設定（最初に一度だけ実行されます）
//
bool setup()
{
for (int y = 0; y < height; ++y)
100 %
出力
出力元(S): デバッグ
スレッド 0x1978 はコード 0 (0x0) で終了しました。
スレッド 0x1db8 はコード 0 (0x0) で終了しました。
スレッド 0x18a4 はコード 0 (0x0) で終了しました。
スレッド 0x1e9c はコード 0 (0x0) で終了しました。
スレッド 0x1eb8 はコード 0 (0x0) で終了しました。
スレッド 0x1904 はコード 0 (0x0) で終了しました。
スレッド 0x810 はコード 0 (0x0) で終了しました。
プログラム '[7604] kadai1.exe' はコード 0 (0x0) で終了しました。
ソリューシ... クラス ビュー プロパティ... チーム エク...
サーバー エクスプローラー ツールボックス プロパティ 診断ツール
準備完了
1 行
1 列
1 文字
挿入
ソース管理に追加
18:10
2017/04/27

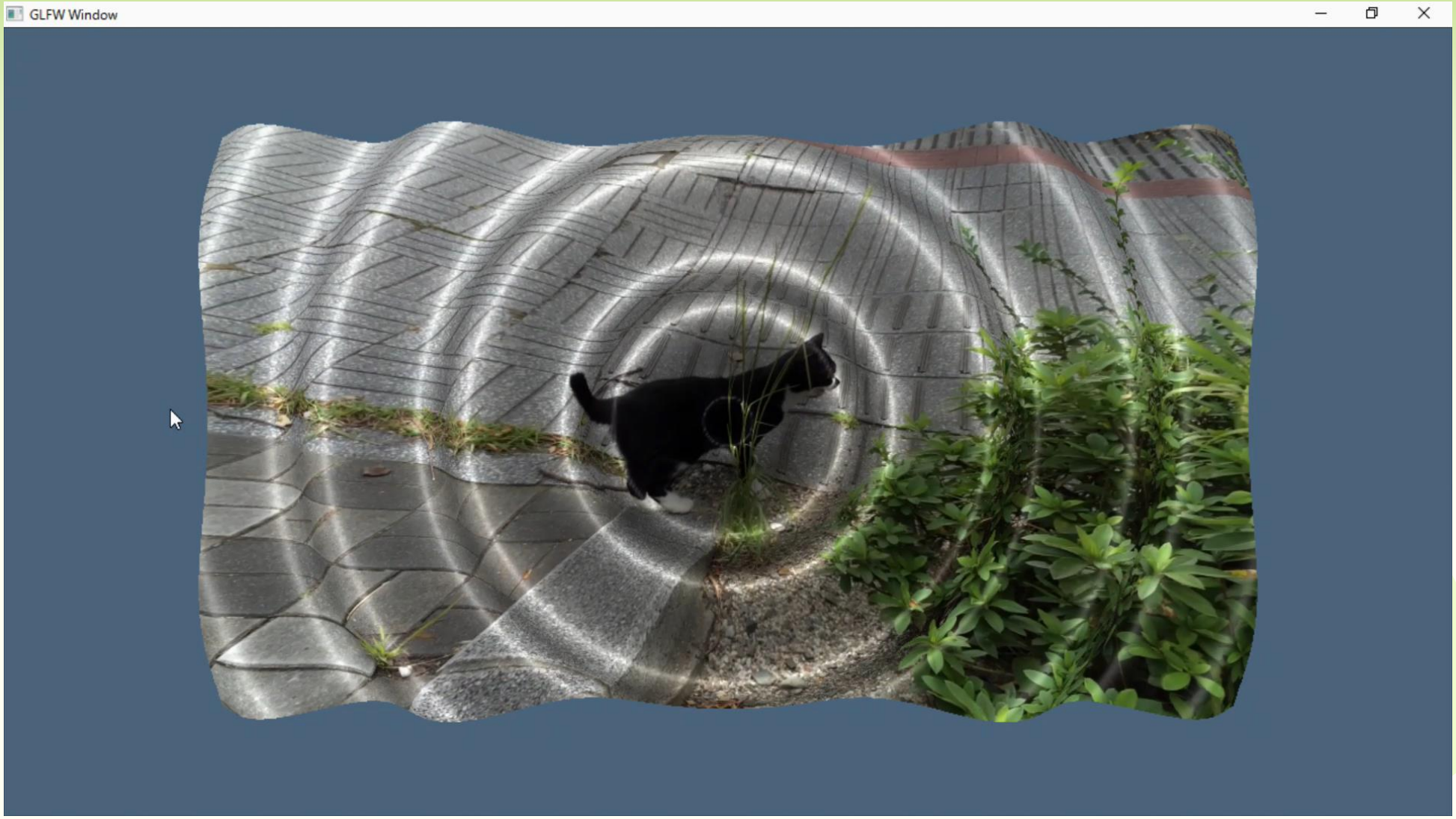
課題

ビデオにエフェクトをかけるプログラムを作る

課題テンプレートの実行結果



目標の実行結果（ビデオエフェクト）



1日目

kadai1.cpp の内容

```
// 共通の関数
#include "common.h"

//
// 設定（最初に一度だけ実行されます）
//
bool setup()
{

  // セットアップに成功した
  return true;
}

//
// 更新（毎回実行されます）
//
bool update()
{

  // プログラムを終了しない
  return true;
}
```

ここにプログラムを書きます

ここにプログラムを書きます

- kadai1.cpp を修正してください
  - setup() 関数と update() 関数の中身を実装します
- setup() 関数
  - プログラムの実行開始時に一度だけ実行されます
  - false を返すとメッセージを表示してプログラムを停止します
- update() 関数
  - 画面表示を行うたびに毎回実行されます
  - false を返すとプログラムを終了します

common.h の内容

```
// 点データの作成
extern void createPoint(int width, int height,
const void *point = nullptr);

// 点データの転送
extern void submitPoint(int width, int height,
const void *point);

// 色データの作成
extern void createColor(int width, int height,
const void *color = nullptr);

// 色データの転送
extern void submitColor(int width, int height,
const void *color);

// 現在時刻の取得
extern float getTime();
```

- common.h にはプログラムの作成に必要な関数を宣言しています
  - もちろん他に C / C++ の関数も使えます
- createPoint()
  - 点のデータを保持するメモリを確保してそこにデータを転送します
- submitPoint()
  - 点のデータを転送します
- createColor()
  - 色のデータを保持するメモリを確保してそこにデータを転送します
- submitColor()
  - 色のデータを転送します
- getTime()
  - 現在時刻を秒で取得します

createPoint(int width, int height, const void *point)

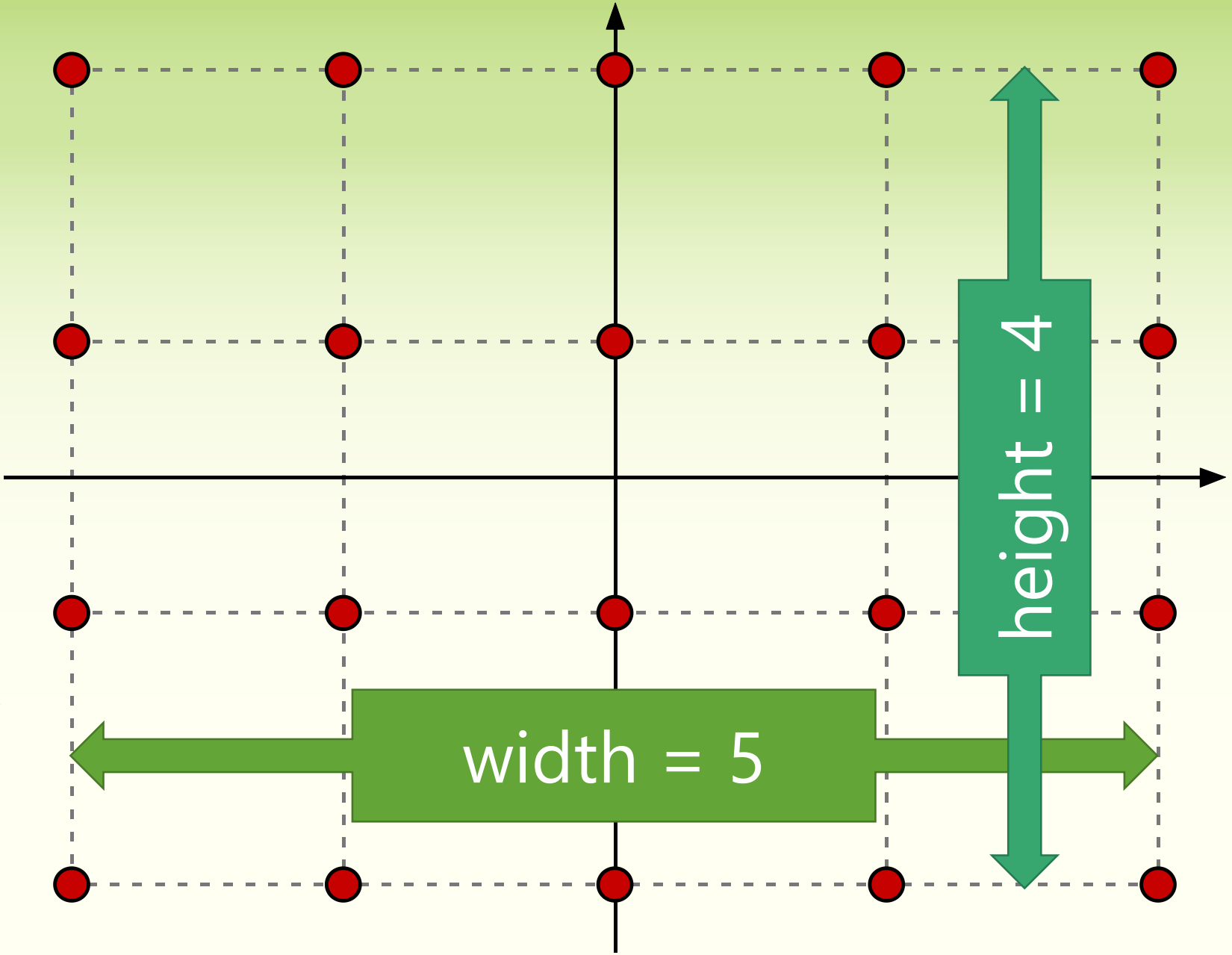
```
// 点群の幅
const int width(160);

// 点群の高さ
const int height(120);

// 点のデータ
float point[height][width][3];

    :
    :

// 点のデータを保持するメモリを確保してそこにデータを転送する
createPoint(width, height, point);
```

これに値を設定します

- 点のデータを保持するメモリを確保してそこにデータを転送します
  - width に点群の幅を指定します
  - height に点群の高さを指定します
  - point は点のデータを格納した配列です
- point
  - 一つの点は x, y, z の三つの値 (座標値) を持ちます
  - これが幅 width 個，高さ height 個並んでいます
  - 値のデータ型は float です
  - point の指定を省略 (あるいは nullptr を指定) した場合はデータを転送しません
    - メモリの確保だけを行います
- setup() で実行する必要があります

点のデータ

点のデータは座標値



submitPoint(int width, int height, const void *point)

```
// 点群の幅
const int width(160);

// 点群の高さ
const int height(120);

// 点のデータ
float point[height][width][3];

    :
    :

// 点のデータを転送する
submitPoint(width, height, point);
```

これに値を設定します

- 点のデータの転送のみを行います
  - width に点群の幅を指定します
  - height に点群の高さを指定します
  - point は点のデータを格納した配列です
- point
  - 一つの点は x, y, z の三つの値 (座標値) を持ちます
  - これが幅 width 個，高さ height 個並んでいます
  - 値のデータ型は float です
  - point の指定は省略できません
- update() で実行する必要があります

createColor(int width, int height, const void *point)

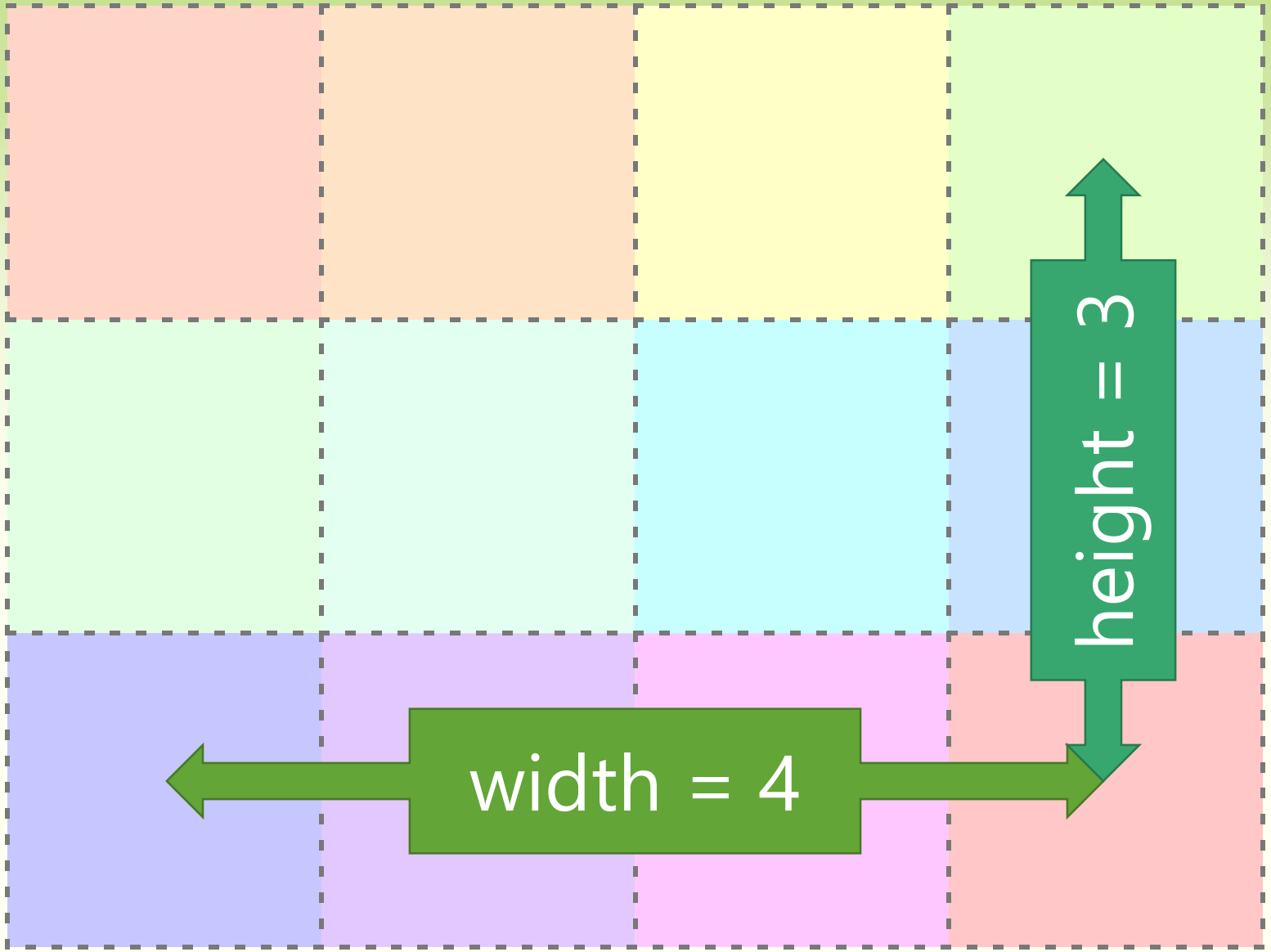
```
// 画像の幅
const int width(640);

// 画像の高さ
const int height(480);

// 色のデータ
unsigned char color[height][width][3];

    :
    :

// 色のデータを保持するメモリを確保してそこにデータを転送する
createColor(width, height, color);
```

これに値を設定します

- 色のデータを保持するメモリを確保してそこにデータを転送します
  - width に画像の幅を指定します
  - height に画像の高さを指定します
  - color は色のデータを格納した配列です
- color
  - 一つの色は r, g, b の三つの値 (三原色) を持ちます
  - これが幅 width 個，高さ height 個並んでいます
  - 値のデータ型は unsigned char です
  - color の指定を省略 (あるいは nullptr を指定) した場合はデータを転送しません
    - メモリの確保だけを行います
- setup() で実行する必要があります

色のデータ

色のデータは三原色



submitColor(int width, int height, const void *point)

```
// 画像の幅
const int width(640);

// 画像の高さ
const int height(480);

// 色のデータ
unsigned char color[height][width][3];

    :
    :

// 色のデータを転送する
submitColor(width, height, color);
```

これに値を設定します

- 色のデータの転送のみを行います
  - width に画像の幅を指定します
  - height に画像の高さを指定します
  - color は色のデータを格納した配列です
- color
  - 一つの色は r, g, b の三つの値 (三原色) を持ちます
  - これが幅 width 個，高さ height 個並んでいます
  - 値のデータ型は unsigned char です
  - color の指定は省略できません
- update() で実行する必要があります

width, height

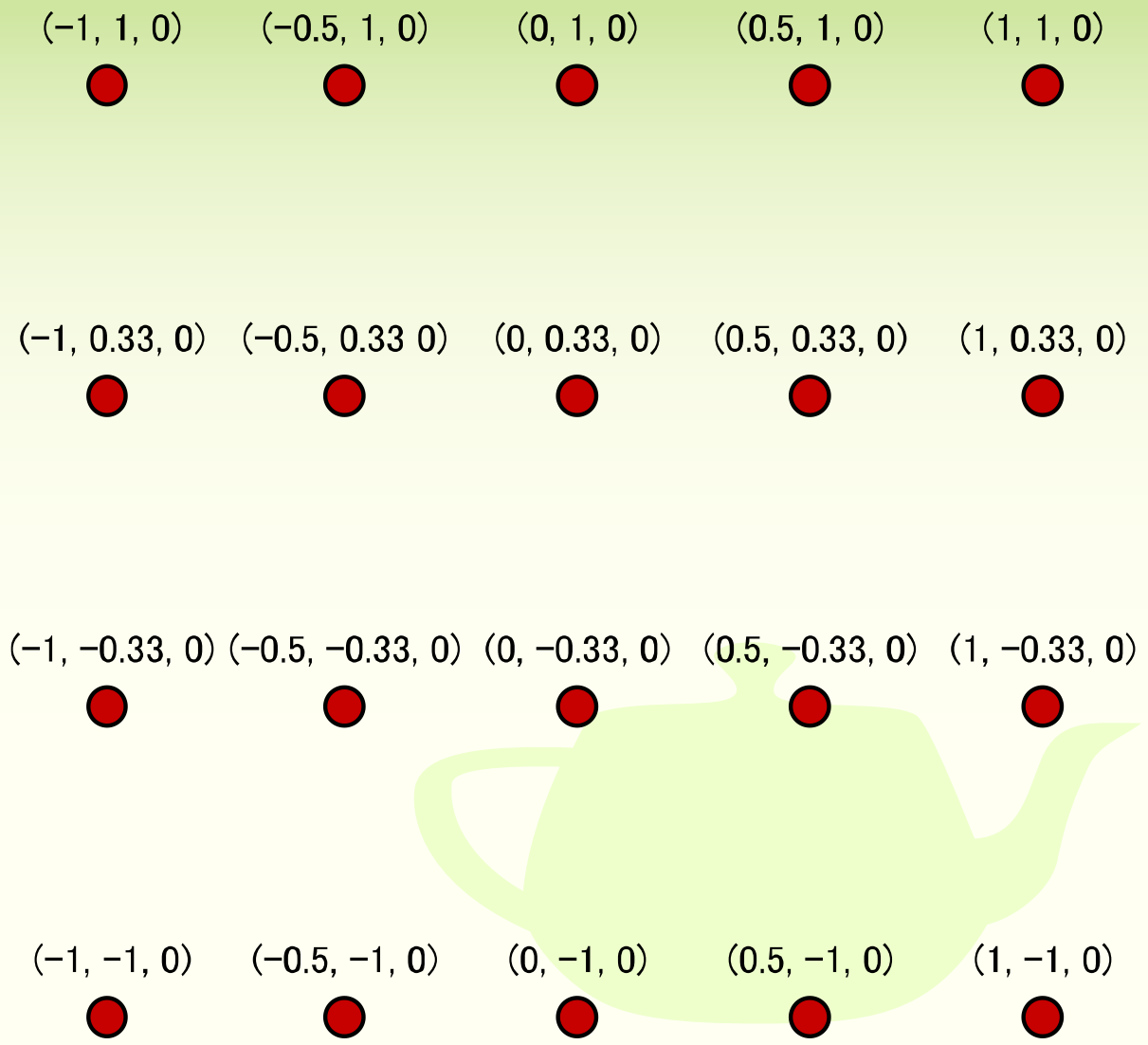
- **点群**の幅 (width) と高さ (height) と**画像**の幅 (width) と高さ (height) は一致している必要はありません
- **submitPoint**(width, height, point) で指定する width, height には **createPoint**(width, height, point) で指定した width, height より大きな値を指定することはできません
- **submitColor**(width, height, color) で指定する width, height には **createColor**(width, height, color) で指定した width, height より大きな値を指定することはできません

点を格子状に並べる

```
// メッシュの幅と高さ
const int width(5), height(4);

// 点データ
float point[height][width][3];

==========

  for (int j = 0; j < height; ++j)
  {
    for (int i = 0; i < width; ++i)
    {
      // x 方向のパラメータ（0≦u≦1）
      const float u(float(i) / float(width - 1));

      // y 方向のパラメータ（0≦v≦1）
      const float v(float(j) / float(height - 1));

      // 頂点の位置
      point[j][i][0] = u * 2.0f - 1.0f;
      point[j][i][1] = v * 2.0f - 1.0f;
      point[j][i][2] = 0.0f;

      （省略）
    }
  }
```

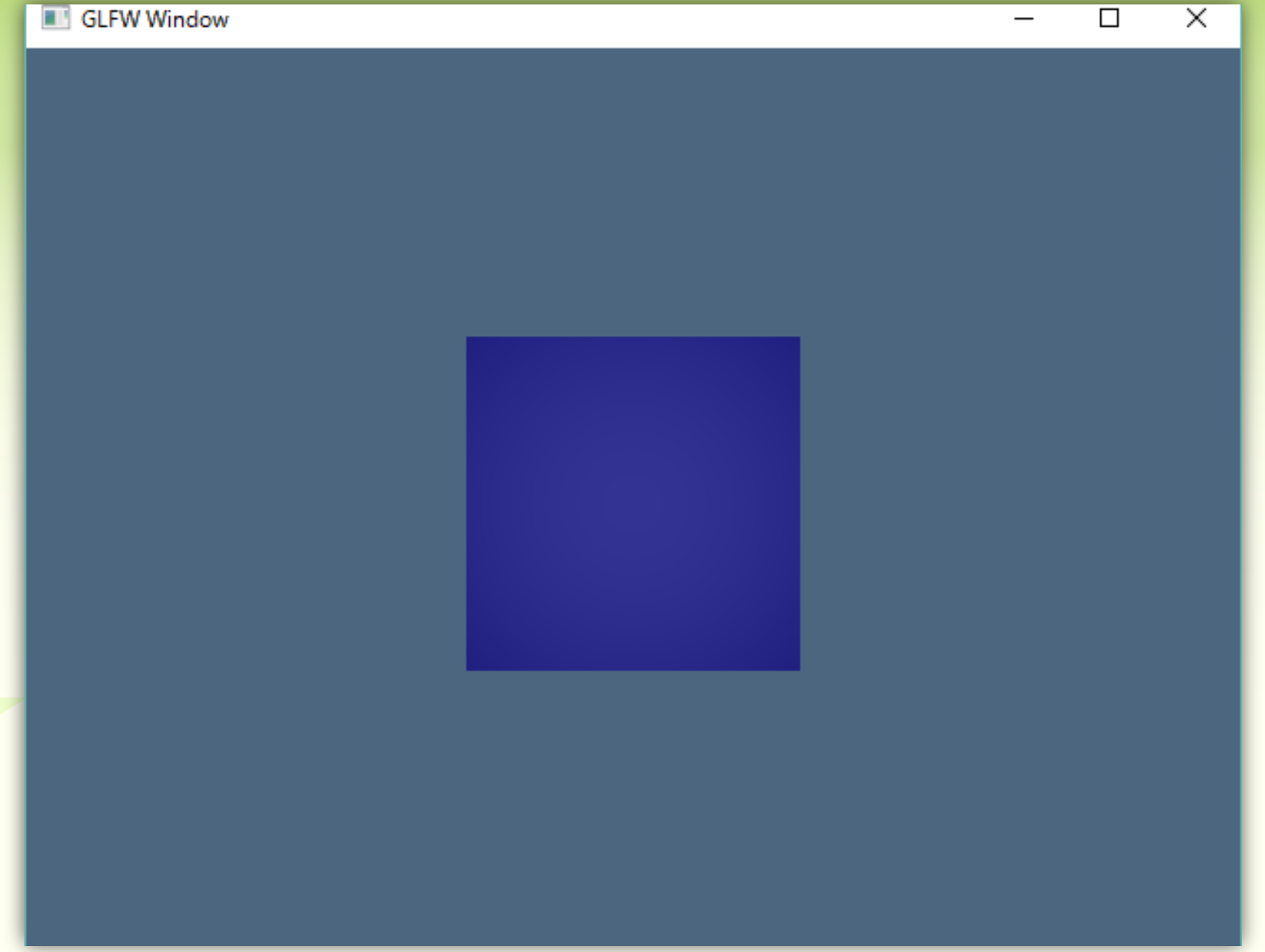
パラメータ
(u, v)
↓
(x, y, z)
点の座標値

$$\begin{cases} x = 2u - 1 \\ y = 2v - 1 \\ z = 0 \end{cases}$$



実行結果

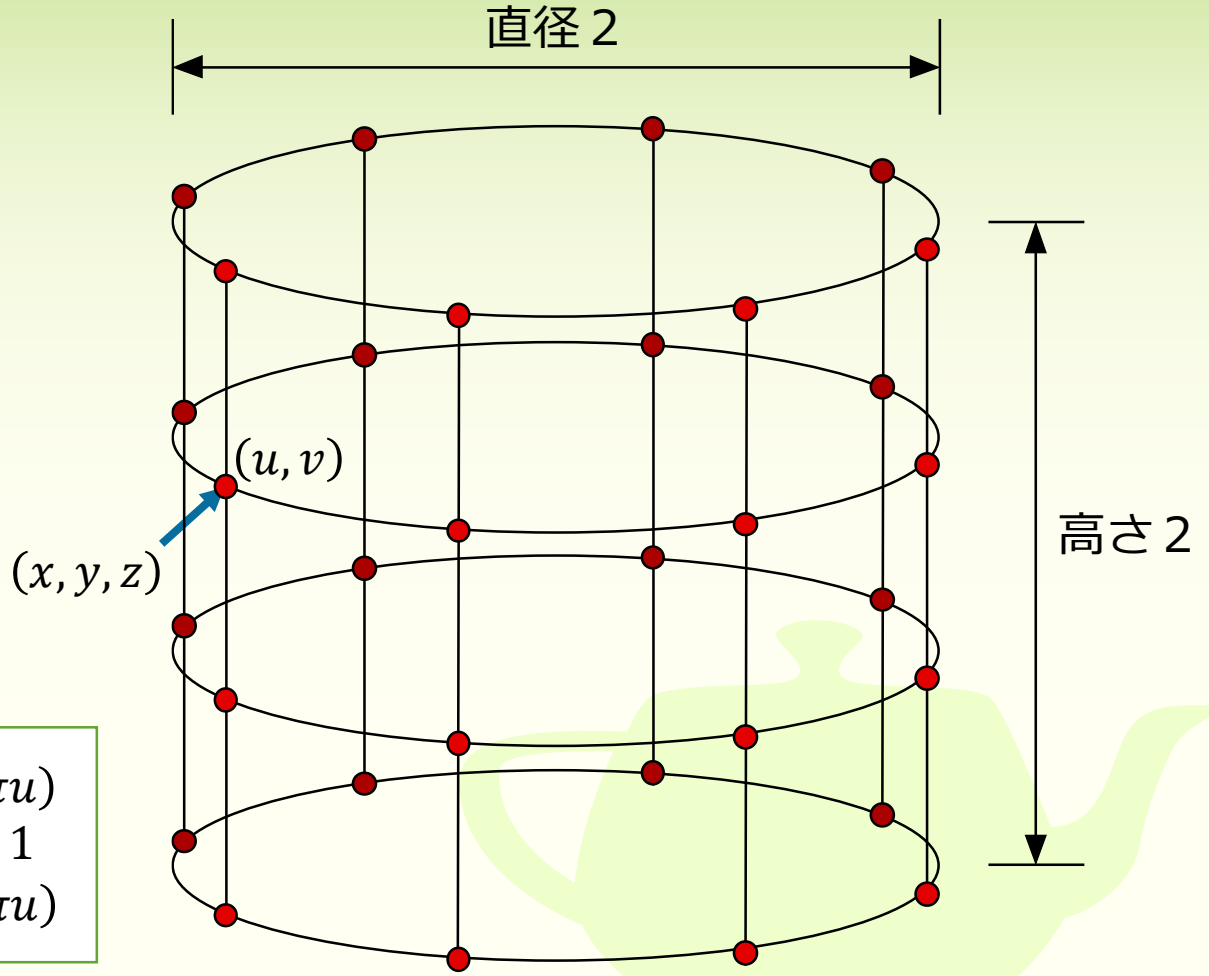
- kadai1.cpp の設定
  - width = 100
  - height = 100



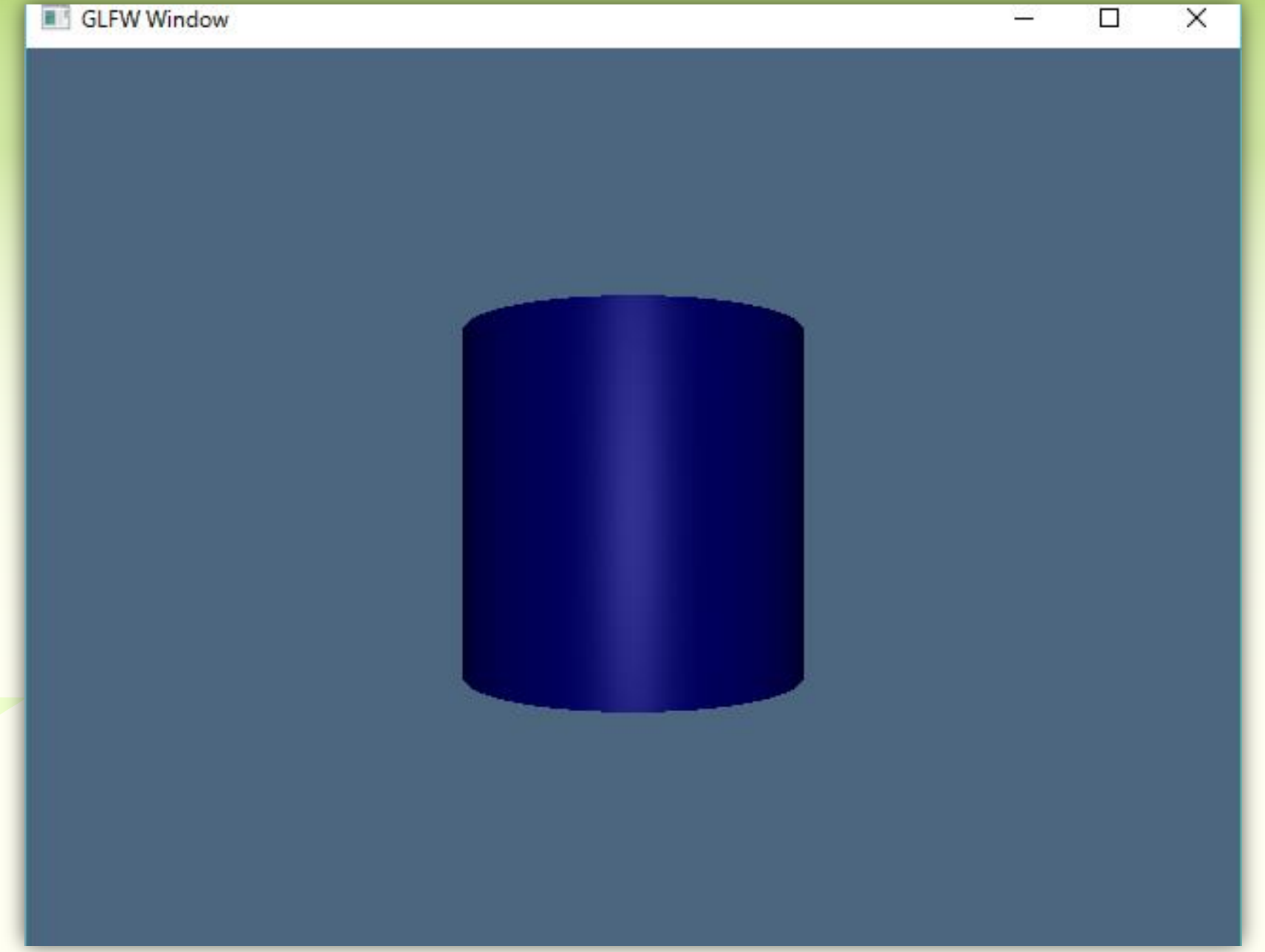
点を円柱状に並べる

```
// 円周率
const float pi(3.14159265f);

=========================

for (int j = 0; j < height; ++j)
 {
   for (int i = 0; i < width; ++i)
   {
     // x 方向のパラメータ (0≦u≦1)
     const float u(float(i) / float(width - 1));

     // 経度
     const float longtude(2.0f * pi * u);

     // y 方向のパラメータ (0≦v≦1)
     const float v(float(j) / float(height - 1));

     // 頂点の位置
     point[j][i][0] = sin(longtude);
     point[j][i][1] = v * 2.0f - 1.0f;
     point[j][i][2] = cos(longtude);

     (省略)
   }
 }
```

$$\begin{cases} x = \sin(2\pi u) \\ y = 2v - 1 \\ z = \cos(2\pi u) \end{cases}$$



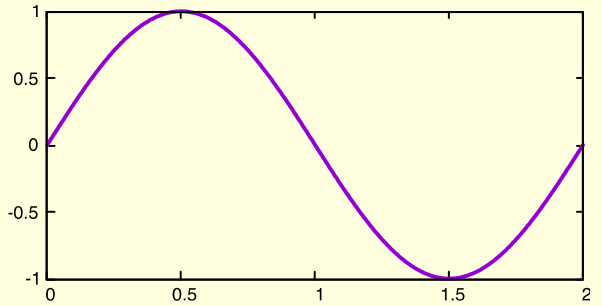
実行結果



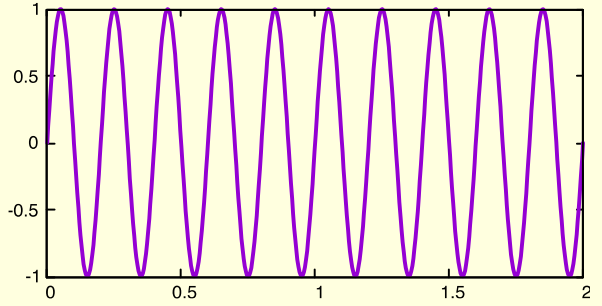
場所によって色を変える

```
for (int j = 0; j < height; ++j)
 {
   for (int i = 0; i < width; ++i)
   {
     // x 方向のパラメータ（0≦u≦1）
     const float u(float(i) / float(width - 1));

     // 経度
     const float longtude(2.0f * pi * u);

     // y 方向のパラメータ（0≦v≦1）
     const float v(float(j) / float(height - 1));

     // 緯度
     const float latitude(pi * v);

     // 緯度に対応した色のスケール
     const float latitude_color(sin(10.0f * pi * v) * 0.5f + 0.5f);

     （省略）

     // 頂点の色
     color[j][i][0] = 100;
     color[j][i][1] = unsigned char(255 * latitude_color);
     color[j][i][2] = 0;
   }
 }
```

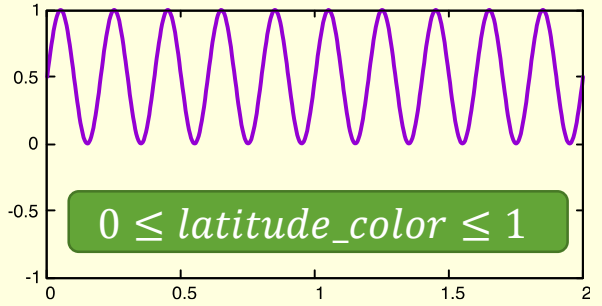
$\sin(\pi v)$



$\sin(10\pi v)$



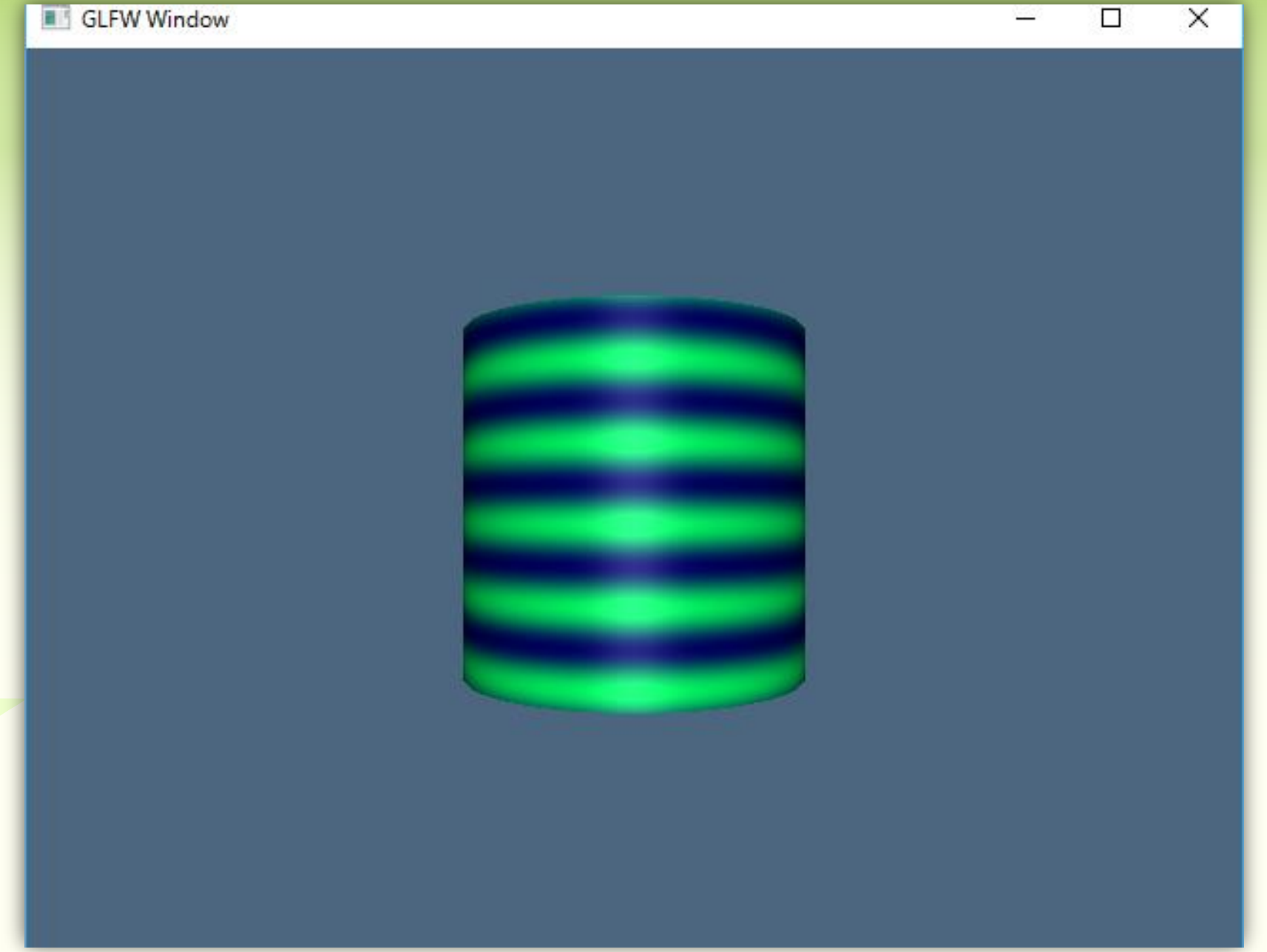
$\frac{1}{2}\sin(10\pi v) + \frac{1}{2}$



色のデータの最大値

unsigned char 型には 0～255 の整数値を格納できる

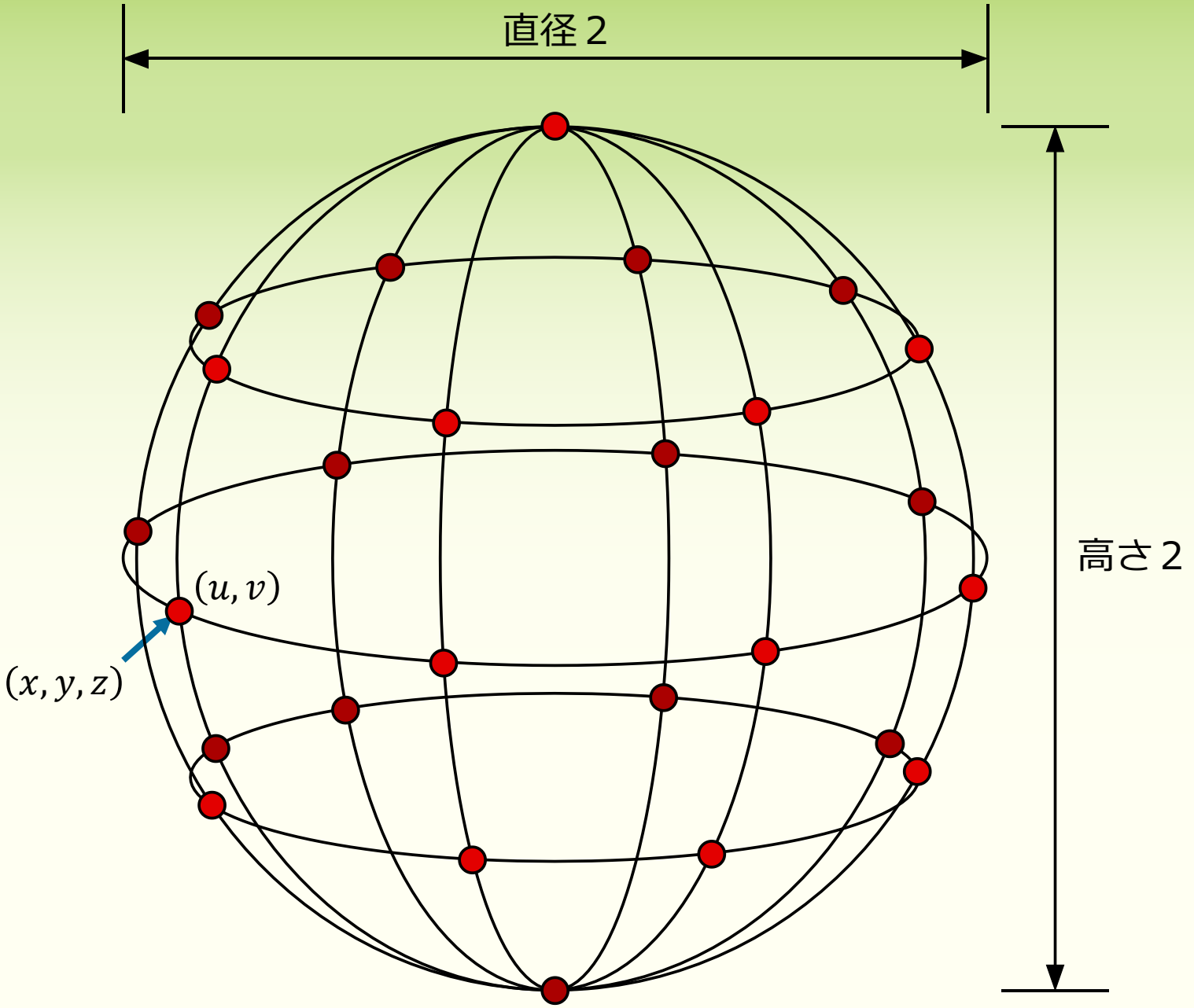
実行結果



課題1

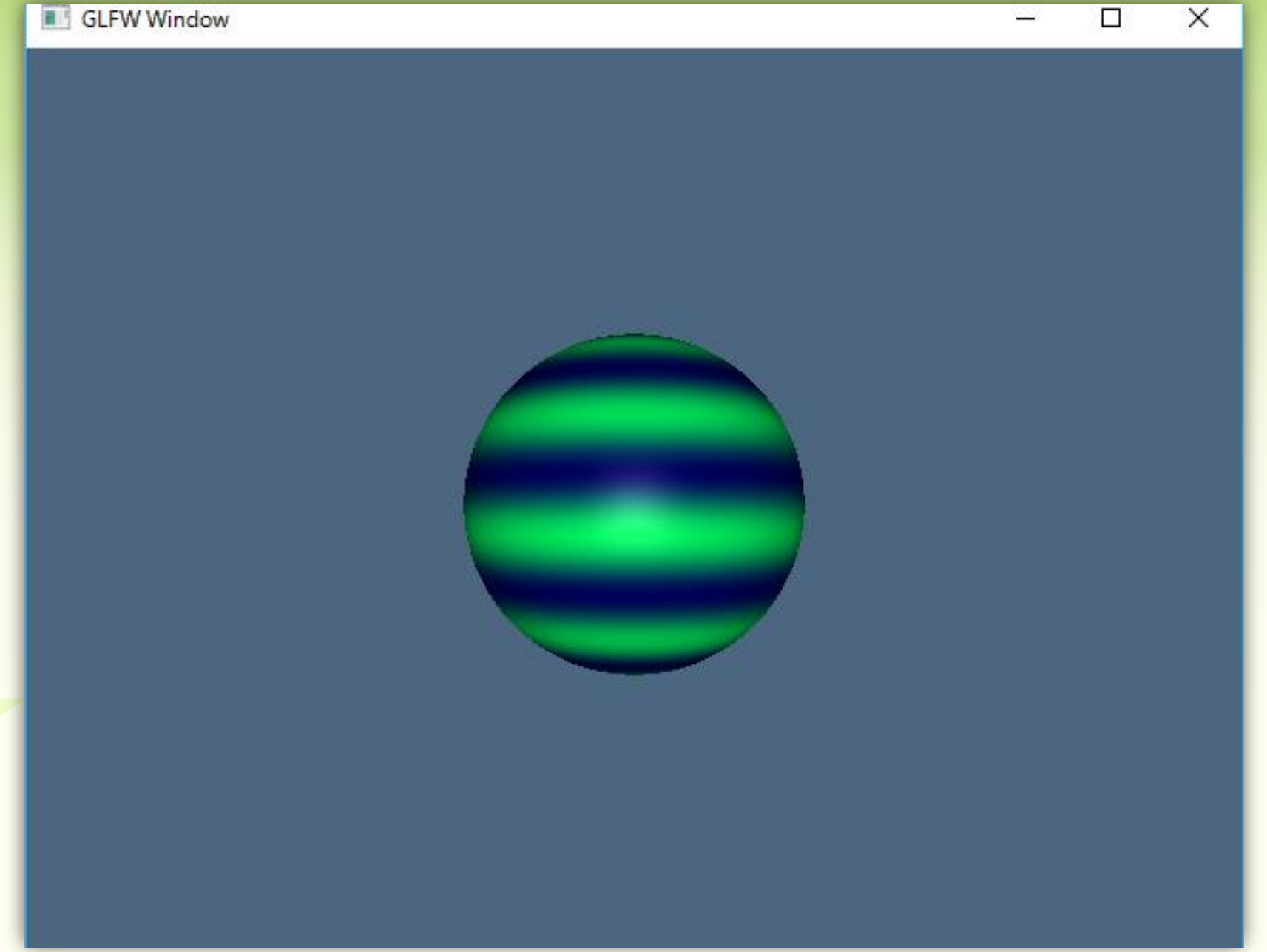
- 球を描いてください
  - 円柱をベースにして結構です
  - その場合北極と南極は複数の点が重なっていても構いません

$$\begin{cases} x = \sin(\pi v)\sin(2\pi u) \\ y = -\cos(\pi v) \\ z = \sin(\pi v)\cos(2\pi u) \end{cases}$$



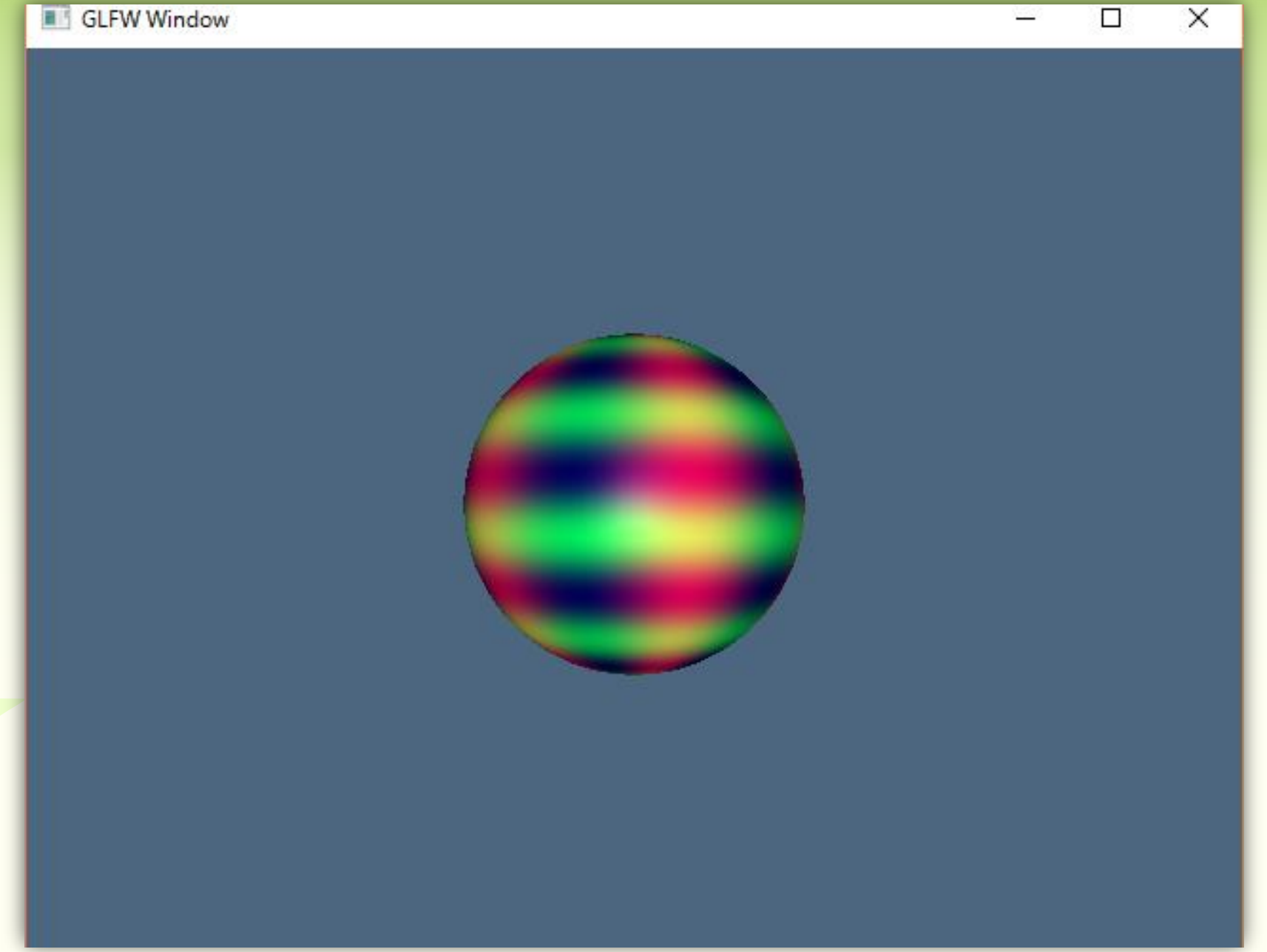
課題1 実行結果

- スクリーンショットを 01.png というファイル名でアップロードしてください
- すでに 01.png がある場合は上書きしてください



課題2

- 縦縞をつけてください
  - あるいは好きな模様をつけてください
- スクリーンショットを 02.png というファイル名でアップロードしてください



時間に従って変形する

- update() で point に値を設定する
  - update() は繰り返し実行される
- point に設定する値に時間の関数を用いる
  - 時間は getTime() 関数で得られる
    - たとえば float scale = sin(2.0f * pi * getTime()) * 0.2f + 1.0f; とすれば scale は 1 秒ごとに 0.8 ～ 1.2 の間を変化する値をとる
    - この scale を point に代入する座標値に乗じれば図形は拡大縮小を繰り返す
- 求めた point の値は submitPoint() を使って転送する
  - submitPoint(width, height, point);

kadai1.cpp の変更点 (1/5)

```
// 周期
const float cycle(1.0f);

//
// 設定（最初に一度だけ実行されます）
//
bool setup()
{
  for (int j = 0; j < height; ++j)
  {
    for (int i = 0; i < width; ++i)
    {
      // x 方向のパラメータ（0≦u≦1）
      const float u(float(i) / float(width - 1));

      // 経度
      const float longtude(2.0f * pi * u);

      // 経度に対応した色のスケール
      const float longtude_color(sin(10.0f * pi * u) * 0.5f + 0.5f);
```

追加

update() にコピー

kadai1.cpp の変更点 (2/5)

```
      // y 方向のパラメータ (0≦v≦1)
      const float v(float(j) / float(height - 1));

      // 緯度
      const float latitude(pi * v);

      // 緯度に対応した色のスケール
      const float latitude_color(sin(10.0f * pi * v) * 0.5f + 0.5f);

      // 頂点の位置
      point[j][i][0] = sin(latitude) * sin(longtude);
      point[j][i][1] = -cos(latitude);
      point[j][i][2] = sin(latitude) * cos(longtude);

      // 頂点の色
      color[j][i][0] = 100;
      color[j][i][1] = unsigned char(255 * latitude_color);
      color[j][i][2] = unsigned char(255 * longtude_color);
    }
  }
```

update() にコピー

update() に移動

kadai1.cpp の変更点 (3/5)

```
  // 点データを登録する
  createPoint(width, height);

  // 色データを登録する
  createColor(width, height, color);

  // セットアップに成功した
  return true;
}

//
// 更新（毎回実行されます）
//
bool update()
{
  // 時間に従って位置を変化させる
  const float offset(sin(2.0f * pi * fmod(getTime(), cycle) / cycle));

  for (int j = 0; j < height; ++j)
  {
    for (int i = 0; i < width; ++i)
```

point は省略

追加

setup() からコピー

kadai1.cpp の変更点 (4/5)

```
    {
      // x 方向のパラメータ (0≦u≦1)
      const float u(float(i) / float(width - 1));

      // 経度
      const float longtude(2.0f * pi * u);

      // 経度に対応した色のスケール
      const float longtude_color(sin(10.0f * pi * u) * 0.5f + 0.5f);

      // y 方向のパラメータ (0≦v≦1)
      const float v(float(j) / float(height - 1));

      // 緯度
      const float latitude(pi * v);

      // 緯度に対応した色のスケール
      const float latitude_color(sin(10.0f * pi * v) * 0.5f + 0.5f);

      // 形状のスケール
      const float scale(offset + 1.0f);
```

← setup() からコピー

← 追加

kadai1.cpp の変更点 (5/5)

```
        // 頂点の位置
        point[j][i][0] = scale * ( ... );
        point[j][i][1] = scale * ( ... );
        point[j][i][2] = scale * ( ... );
      }
    }

    // 点データを登録する
    submitPoint(width, height, point);

    // プログラムを終了しない
    return true;
}
```

update() の中で point に代入

point を転送する

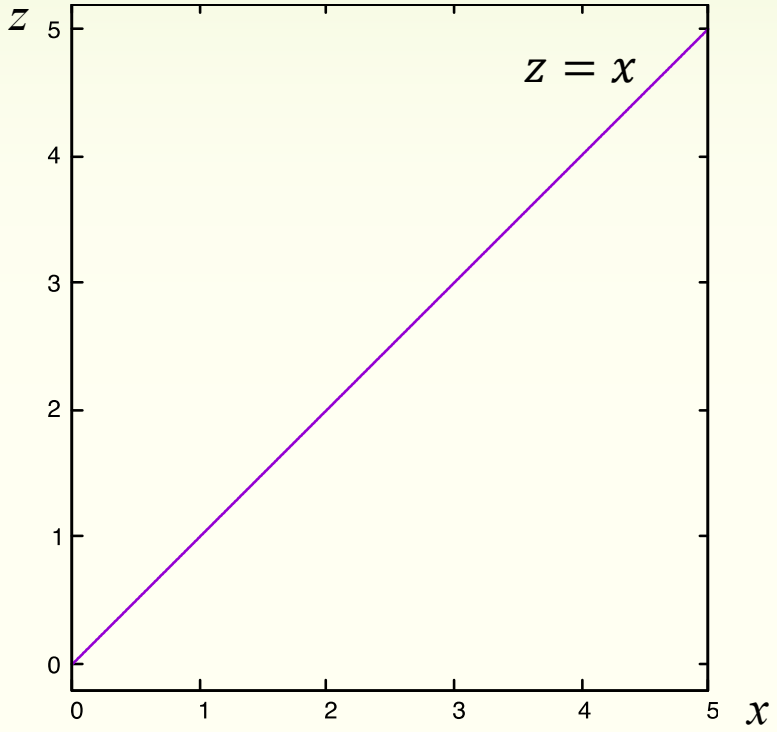
← setup() から移動して各要素を scale 倍

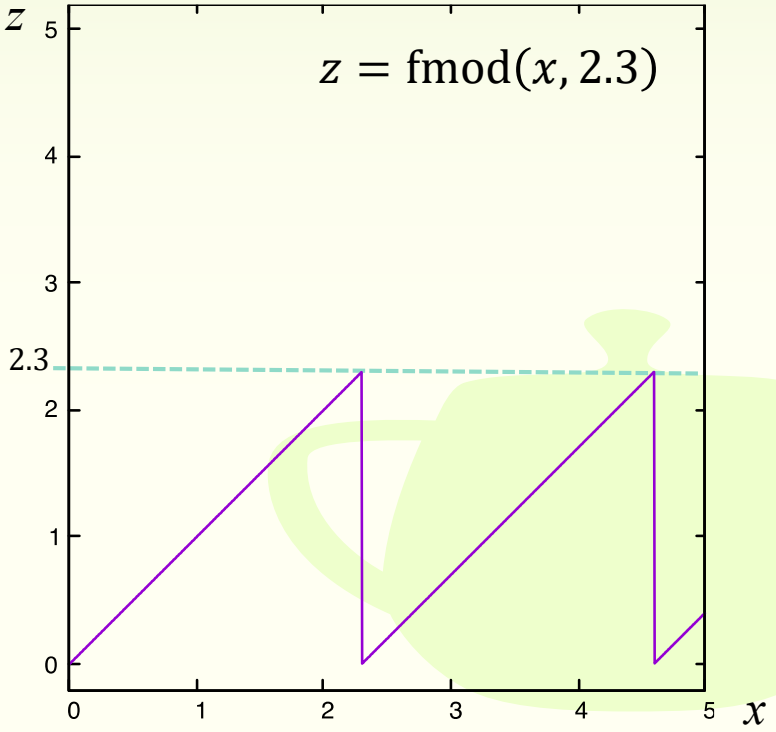
← setup() からコピー

← 追加

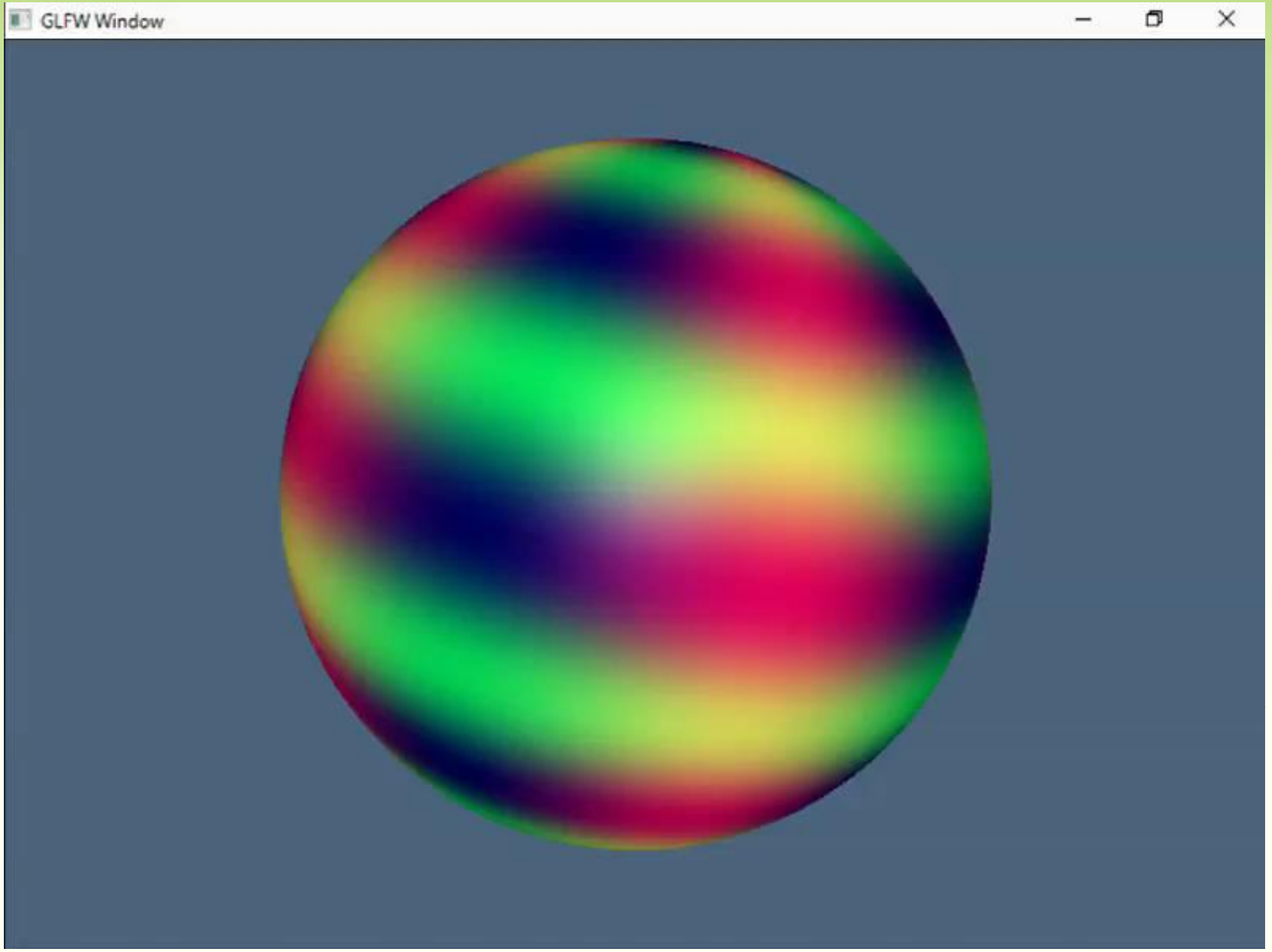
fmod() 関数

- 実数の剰余を求める
  - fmod(x, y) → $x - \lfloor x/y \rfloor y$
    - fmod(15.6, 2.3) → $15.6 - \lfloor 15.6/2.3 \rfloor \times 2.3 = 15.6 - 6 \times 2.3 = 1.8$



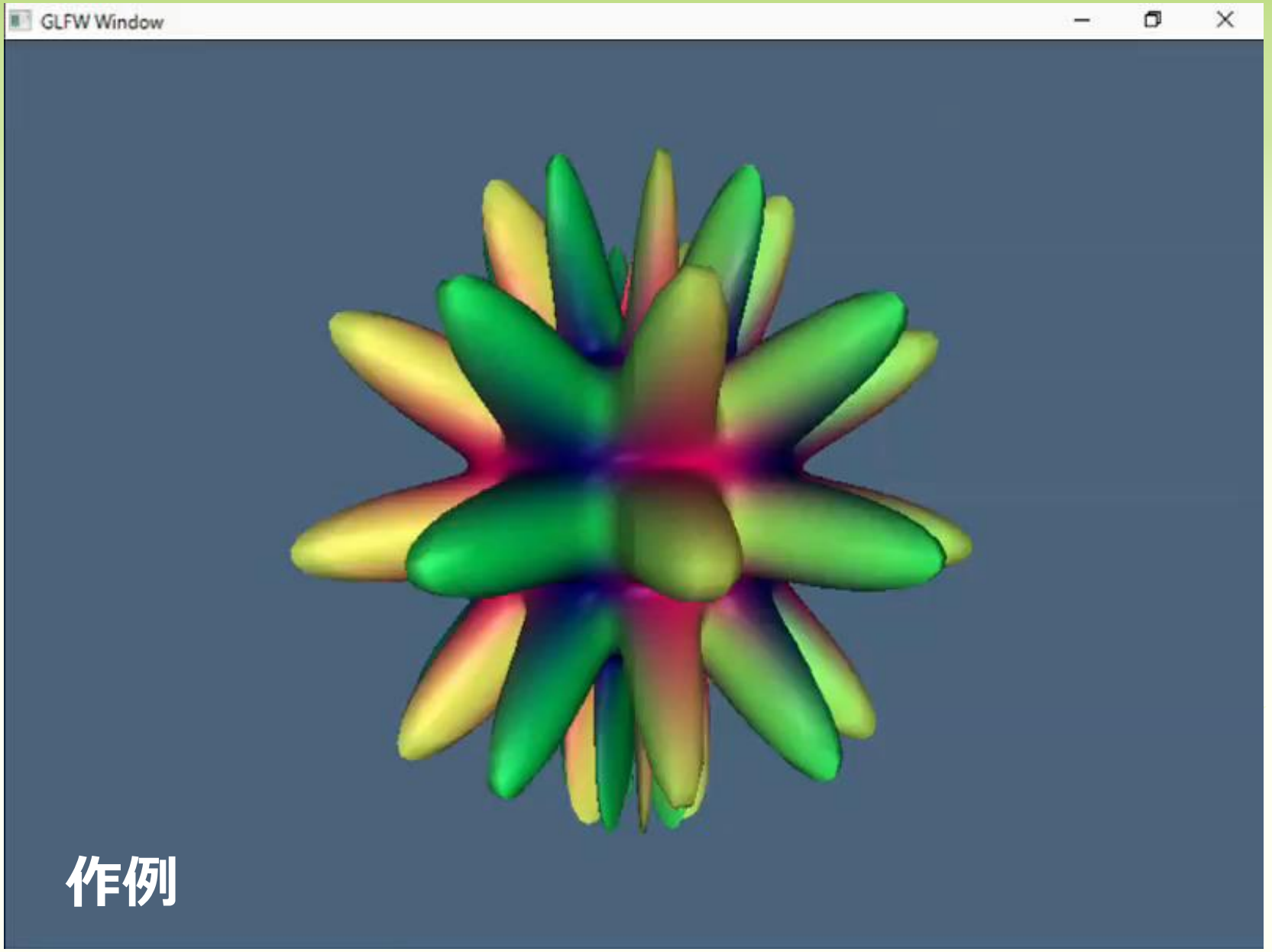


実行結果



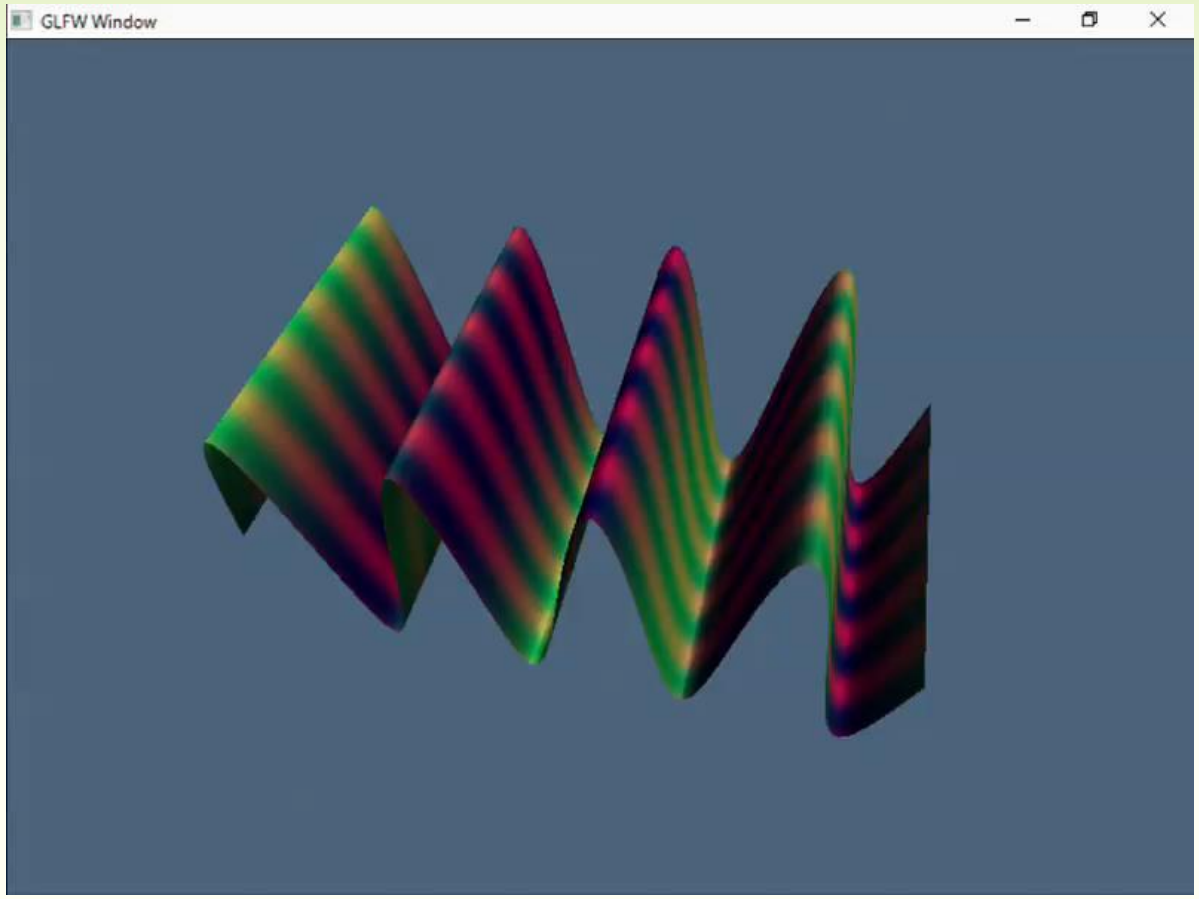
課題3

- 形の変化の仕方が場所によって異なるようにしてください
- スクリーンショットを 03.png というファイル名でアップロードしてください

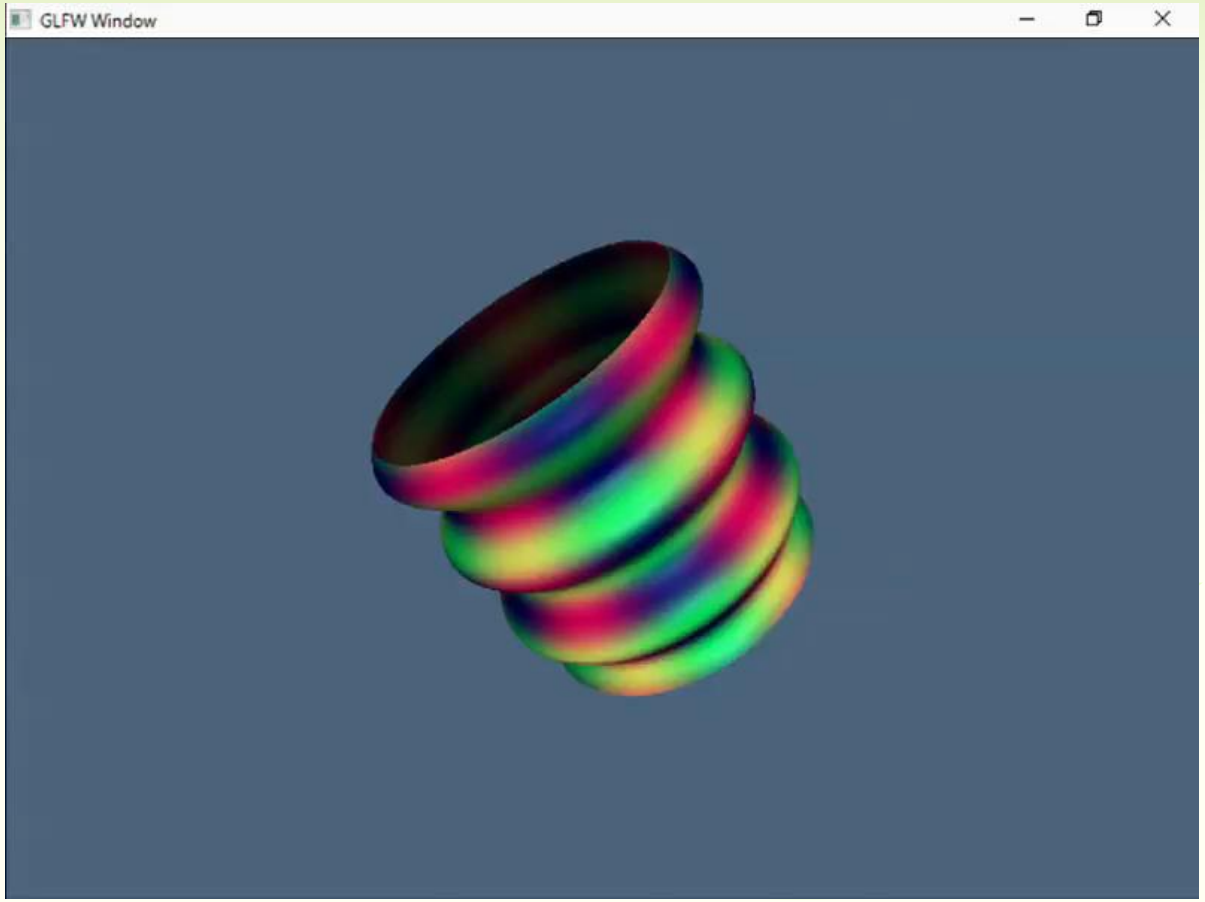


課題3のそのほかの作例

一枚板を変形した場合



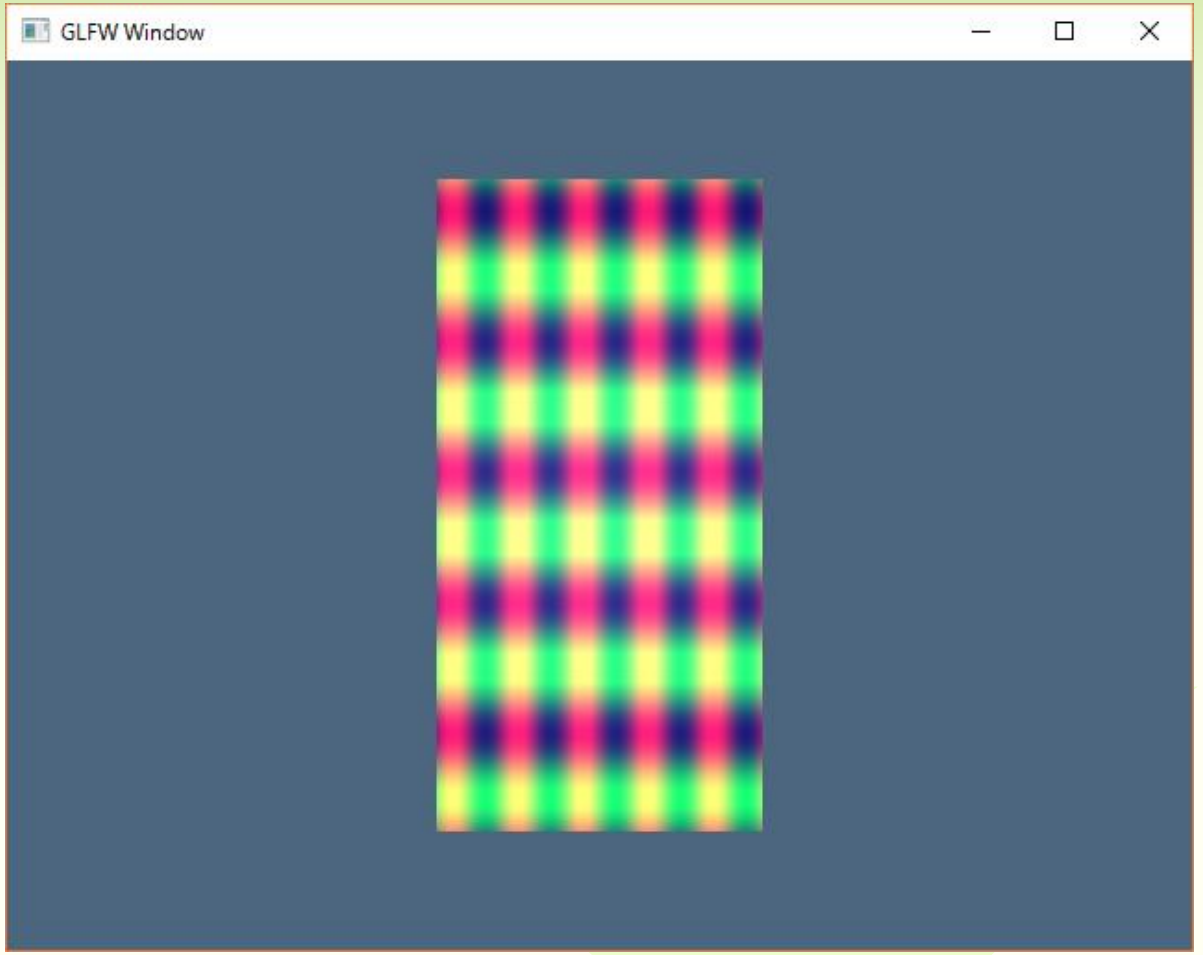
円柱を変形した場合



2 日目

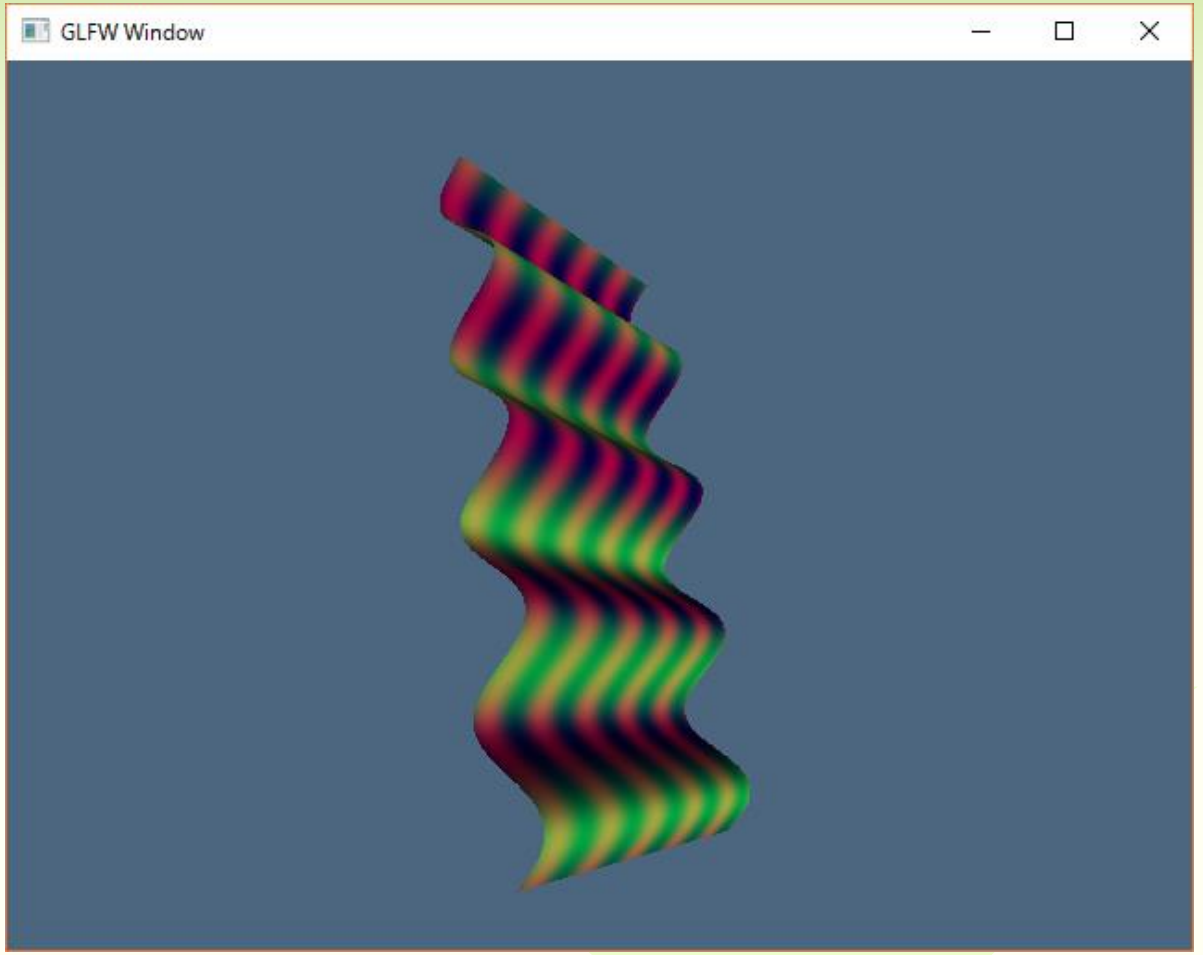
長方形を描く

```
//
// 更新（毎回実行されます）
//
bool update()
{
  for (int j = 0; j < height; ++j)
  {
    for (int i = 0; i < width; ++i)
    {
      // x 方向のパラメータ (0≦u≦1)
      const float u(float(i) / float(width - 1));

      // y 方向のパラメータ (0≦v≦1)
      const float v(float(j) / float(height - 1));

      // 頂点の位置
      point[j][i][0] = 2.0f * u - 1.0f;
      point[j][i][1] = 4.0f * v - 2.0f;
      point[j][i][2] = 0.0f;
    }
  }

  // 点データを登録する
  submitPoint(width, height, point);

  // プログラムを終了しない
  return true;
}
```



波型にする

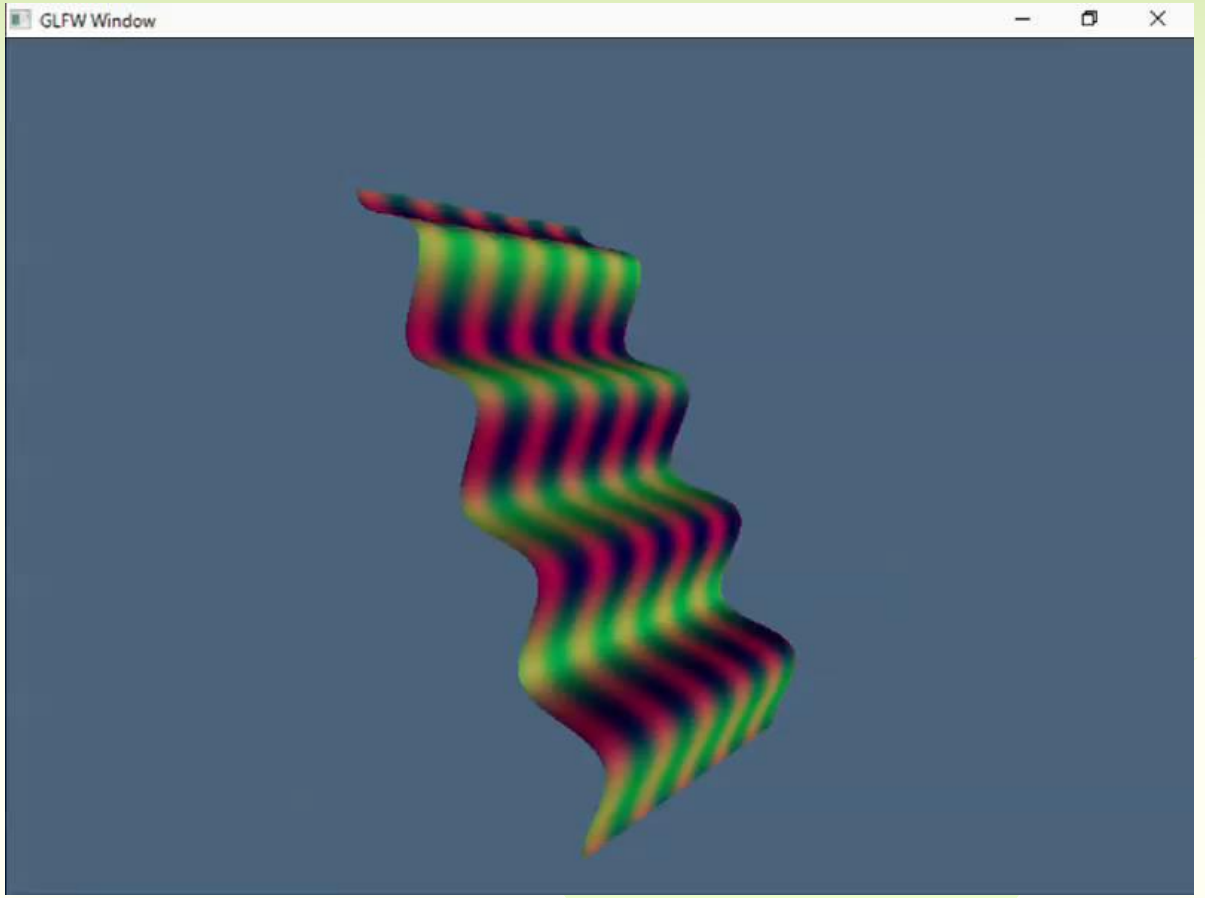
```
//
// 更新（毎回実行されます）
//
bool update()
{
  for (int j = 0; j < height; ++j)
  {
    for (int i = 0; i < width; ++i)
    {
      // x 方向のパラメータ (0≦u≦1)
      const float u(float(i) / float(width - 1));

      // y 方向のパラメータ (0≦v≦1)
      const float v(float(j) / float(height - 1));

      // 頂点の位置
      point[j][i][0] = 2.0f * u - 1.0f;
      point[j][i][1] = 4.0f * v - 2.0f;
      point[j][i][2] = sin(8.0f * pi * v) * 0.2f;
    }
  }

  // 点データを登録する
  submitPoint(width, height, point);

  // プログラムを終了しない
  return true;
}
```



波を動かす

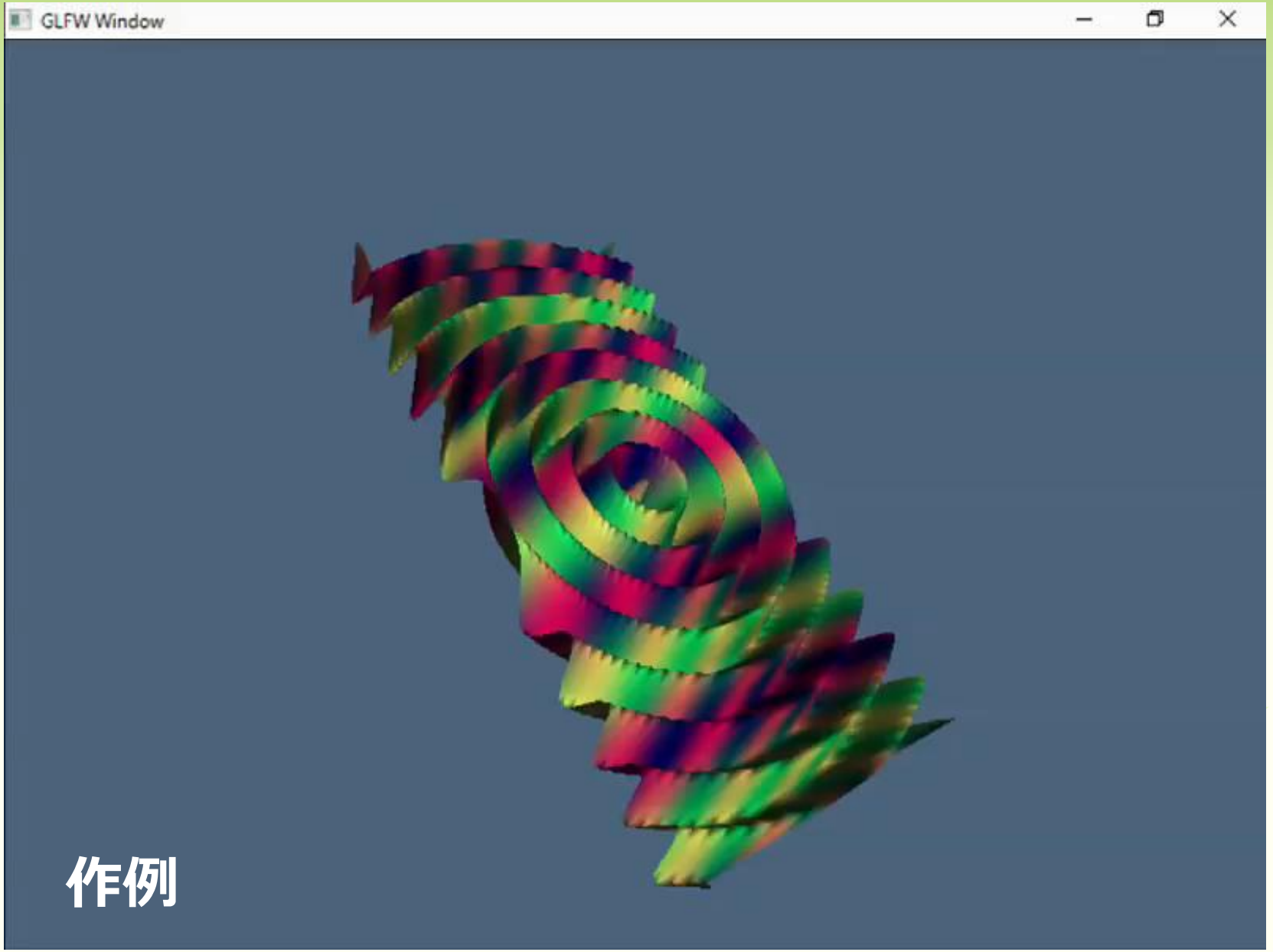
```
//
// 更新（毎回実行されます）
//
bool update()
{
  // 時間に従って位置を変化させる
  const float offset(2.0f * pi * fmod(getTime(), cycle) / cycle);

  for (int j = 0; j < height; ++j)
  {
    for (int i = 0; i < width; ++i)
    {
      // x 方向のパラメータ (0≦u≦1)
      const float u(float(i) / float(width - 1));

      // y 方向のパラメータ (0≦v≦1)
      const float v(float(j) / float(height - 1));

      // 頂点の位置
      point[j][i][0] = 2.0f * u - 1.0f;
      point[j][i][1] = 4.0f * v - 2.0f;
      point[j][i][2] = sin(8.0f * pi * v + offset) * 0.2f;
    }
  }

  // 点データを登録する
  submitPoint(width, height, point);

  // プログラムを終了しない
  return true;
}
```

位相を変化させる



課題4

- 波が板の中心から発生するようにしてください
  - もしギザギザが目立つと感じるなら width と height を**少し**大きくしてみてください
- スクリーンショットを 04.png というファイル名でアップロードしてください

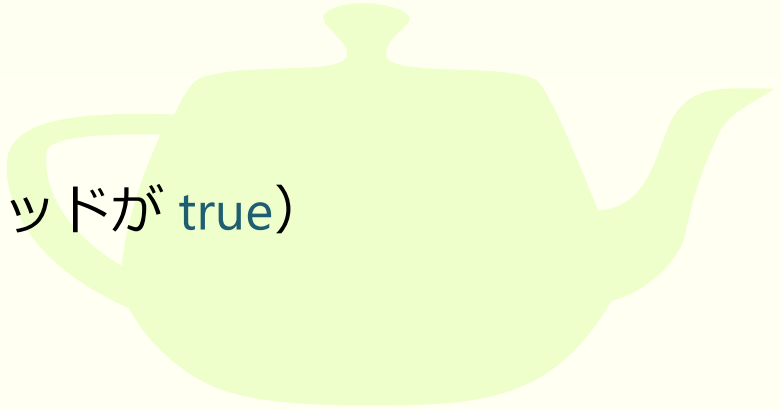


OpenCV の画像データ

- cv::Mat クラス（行列クラス）
  - コンストラクタ
    - cv::Mat(), cv::Mat(int _rows, int _cols, int _type), ほか
      - _rows: 画像の高さ，_cols: 画像の幅
      - _type はデータが符号なし (unsigned) 8 ビット 3 チャンネル (RGB) 画像の時 CV_8UC3
  - メンバ
    - rows: 高さ，cols: 幅，data: データ本体（入っていなければ nullptr），ほか
  - メソッド
    - type(): _type を返す, depth(): ビット深度 (CV_8U とか) を返す，channels(): チャンネル数を返す
    - isContinuous(): これが false だと data にデータが連続して入っていない
      - その場合 data メンバをそのまま使えない

画像の読み込み

- cv::imread(const std::string &filename, int flags = 1)
  - 引数
    - filename: ファイル名の文字列（std::string クラス）
    - flags : （省略した時は 1）
      - flags > 0 ⇒ 強制的に 3 チャンネルカラー画像として読み込む
      - flags = 0 ⇒ 強制的に 1 チャンネルグレースケール画像として読み込む
      - flags < 0 ⇒そのままの画像として読み込まれる
  - 戻り値
    - cv::Mat 型の画像データ
      - 読み込めなかったときは data が nullptr（empty() メソッドが true）



画像を読み込む

```
// 共通の関数
#include "common.h"

// OpenCV
#include <opencv2/opencv.hpp>

// 標準ライブラリ
#include <cmath>

（省略）

//
// 設定（最初に一度だけ実行されます）
//
bool setup()
{
   （省略）

   // 画像の読み込み
   const cv::Mat image(cv::imread("image.jpg"));

   // 色データを登録する
   createColor(image.cols, image.rows, image.data);
```

- プログラムの冒頭で OpenCV のヘッダファイル opencv2/opencv.hpp を #include します
  - cmath より先に #include しないとエラーになるかもしれません
- imread() で image.jpg ファイルを読み込んで image に格納します
  - image.jpg は kadai1 フォルダに入っています
- これは手を抜いています
  - image.data が nullptr でないことを確かめるべきです
  - image.isContinuous() が false でないことを確かめるべきです

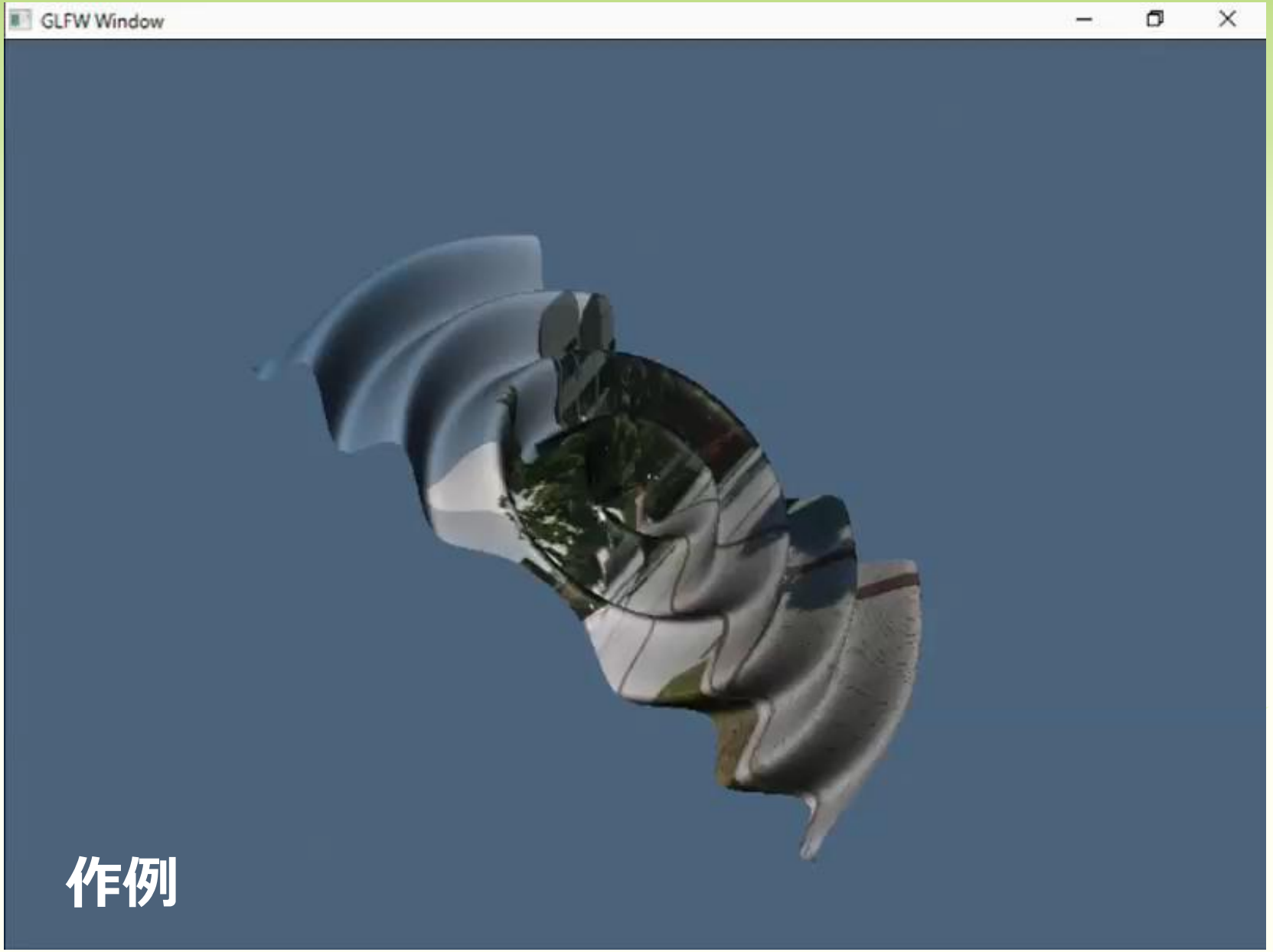
画像の読み込みに失敗した時の処理の追加

```
//
// 設定（最初に一度だけ実行されます）
//
bool setup()
{
   （省略）

  // 画像の読み込み
  const cv::Mat image(cv::imread("image.jpg"));

  // 画像の読み込みに失敗した
  if (image.empty()) return false;

  // 色データを登録する
  createColor(image.cols, image.rows, image.data);
```

- 画像が読み込めなかったときは data メンバが nullptr になっています
- そのとき empty() メソッドは true を返します
  - setup() が false で return すると課題プログラムは**メッセージを出して**終了するように作ってあります
  - ここに自分でエラー処理を追加できます

課題5

- 模様を付ける代わりに画像を貼り付けてください
  - 画像 (image.jpg) は自分で撮影したものなど他のものに入れ替えてください
  - 大学のコンピュータへの保存方法は教員に相談してください
- スクリーンショットを 05.png というファイル名でアップロードしてください



動画の読み込み

- cv::VideoCapture クラス
  - コンストラクタ
    - cv::VideoCapture(), cv::VideoCapture(const std::string &filename), cv::VideoCapture(int device)
      - filename: ムービーファイル名， device: キャプチャデバイス (Web カメラ等) の番号
  - メソッド
    - open(const std::string &filename): ムービーファイルを開く, open(int device): キャプチャデバイスを開く, grab(): フレームを取得する（取得できなければ false を返す）, retrieve(cv::Mat &image, int channel = 0): 取得したフレームを image に格納する, set(int propId, double value): プロパティ propId を設定する, get(int propId): プロパティ propId を取得する
  - 演算子
    - >>: フレームを取得 (grab()) して格納 (retrieve()) する

動画を開く

```
// ビデオ入力
cv::VideoCapture video;

//
// 設定（最初に一度だけ実行されます）
//
bool setup()
{
   （省略）

   // ビデオの取得開始
   video.open("video.mp4");

   // ムービーファイルの1フレームの一時保存先
   cv::Mat frame;

   // 最初のフレームを取得する
   video >> frame;

   // 色データを登録する
   createColor(frame.cols, frame.rows, frame.data);

   // セットアップに成功した
   return true;
}
```

- 動画の場合はフレーム（1枚の画像）を繰り返し読むことになります
- cv::VideoCapture クラスのオブジェクト video を外部変数として宣言して setup() と update() で共有します
- setup() で open() メソッドを使ってムービーファイル video.mp4 を開きます
  - video.mp4 は kadai1 フォルダに入っています
- ムービーファイルのフレームを保存する cv::Mat 型の変数 frame を用意して最初のフレームを取得します
- 取得したフレームを色データとして転送します

フレームを読み込む

```
//
// 更新（毎回実行されます）
//
bool update()
{
   （省略）

   // ムービーファイルの1フレームの一時保存先
   cv::Mat frame;

   // 1フレーム取得する
   video >> frame;

   // 取得したフレームを転送する
   submitColor(frame.cols, frame.rows, frame.data);

   // プログラムを終了しない
   return true;
}
```

- ムービーファイルの１フレームを保存する cv::Mat 型の変数 frame を用意します
- video から１フレーム取得して frame に保存します
- 保存したフレームを転送します
- これは手を抜いています
  - setup() でムービーファイルを正しく開くことができたか確かめるべきです
  - update() でムービーファイルの最後に達した時の処理を追加すべきです

動画が開けなかったときの処理の追加

```
//
// 設定（最初に一度だけ実行されます）
//
bool setup()
{
   （省略）

  // ビデオの取得開始
  if (!video.open("video.mp4")) return false;

  // ムービーファイルの1フレームの一時保存先
  cv::Mat frame;

  // 最初のフレームを取得する
  if (!video.grab()) return false;
  video.retrieve(frame);

  // 色データを登録する
  createColor(frame.cols, frame.rows, frame.data);

  // セットアップに成功した
  return true;
}
```

- ムービーファイルやデバイスが開けないときは open() メソッドの戻り値が false になります
  - ムービーファイルやデバイスが開けたかどうかは isOpened() メソッドでも調べることができます
- >> 演算子の代わりに grab() メソッドと retrieve() メソッドを使ってフレームを取得します
  - フレームが取得できなければ grab() メソッドの戻り値が false になります
  - >> 演算子は grab() メソッドと retrieve() メソッドを組み合わせた処理をします

フレームが取得できなかったときの処理の追加

```
//
// 更新（毎回実行されます）
//
bool update()
{
   （省略）

  // ムービーファイルの1フレームの一時保存先
  cv::Mat frame;

  // 1フレーム取得する
  if (!video.grab()) return false;
  video.retrieve(frame);

  // 取得したフレームを転送する
  submitColor(frame.cols, frame.rows, frame.data);

  // プログラムを終了しない
  return true;
}
```

- grab() メソッドを使って video から１フレーム取り出します
- grab() メソッドが false を返した場合は終了します
  - update() が false で return すると課題プログラムは**何も言わずに**終了するように作ってあります
- 取り出したフレームは retrieve() メソッドを使って保存します

繰り返し再生する

```
//
// 更新（毎回実行されます）
//
bool update()
{
   （省略）

  // 1フレーム取得する
  if (!video.grab())
  {
    // フレームが取得できなかったらムービーファイルを巻き戻す
    video.set(CV_CAP_PROP_POS_MSEC, 0.0);
  }
  else
  {
    // ムービーファイルの1フレームの一時保存先
    cv::Mat frame;

    // 取得したフレームを保存する
    video.retrieve(frame);

    // 取得したフレームを転送する
    submitColor(frame.cols, frame.rows, frame.data);
  }
```

video.grab()の後に移動

- これまでのプログラムはムービーファイルの最後に達すると**何も言わずに**終了します
- そこでムービーファイルの最後に達した時は巻き戻すようにします
  - ムービーファイルの再生位置は CV_CAP_PROP_POS_MSEC プロパティ，CV_CAP_PROP_POS_FRAMES プロパティ，あるいは CV_CAP_PROP_POS_AVI_RATIO プロパティに設定します
  - CV_CAP_PROP_POS_MSEC
    - ミリ秒単位
  - CV_CAP_PROP_POS_FRAMES
    - フレーム単位
  - CV_CAP_PROP_POS_AVI_RATIO
    - 最初を 0，最後を 1 とした 0～1 の割合

最初のフレームの取得時刻を記録する

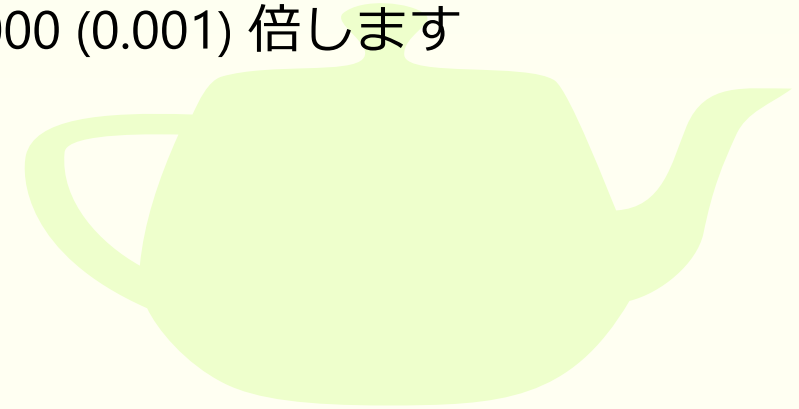
```
// ビデオ入力
cv::VideoCapture video;

// 最初のフレームを取得した時刻
double start;

//
// 設定（最初に一度だけ実行されます）
//
bool setup()
{
   （省略）

  // ムービーファイルのフレームの一時保存先
  cv::Mat frame;

  // 最初のフレームを取得する
  if (!video.grab()) return false;

  // 最初のフレームを取得した時刻を記録しておく
  start = getTime();

  // 取得したフレームを保存する
  video.retrieve(frame);
```

- 演習室の PC の画面表示は通常 60Hz で行われているのに対してムービーファイルは 30fps になっています
- そのため現状では倍速で表示されてしまいます
- そこで画面表示の時刻がムービーファイルの次のフレームの時刻に達していなかったら画像を更新しないようにします
  - 画面表示の時刻は update() が呼び出された時刻です
- まず setup() で最初のフレームを取得した時刻を記録しておきます

ムービーを画面表示のタイミングに合わせる

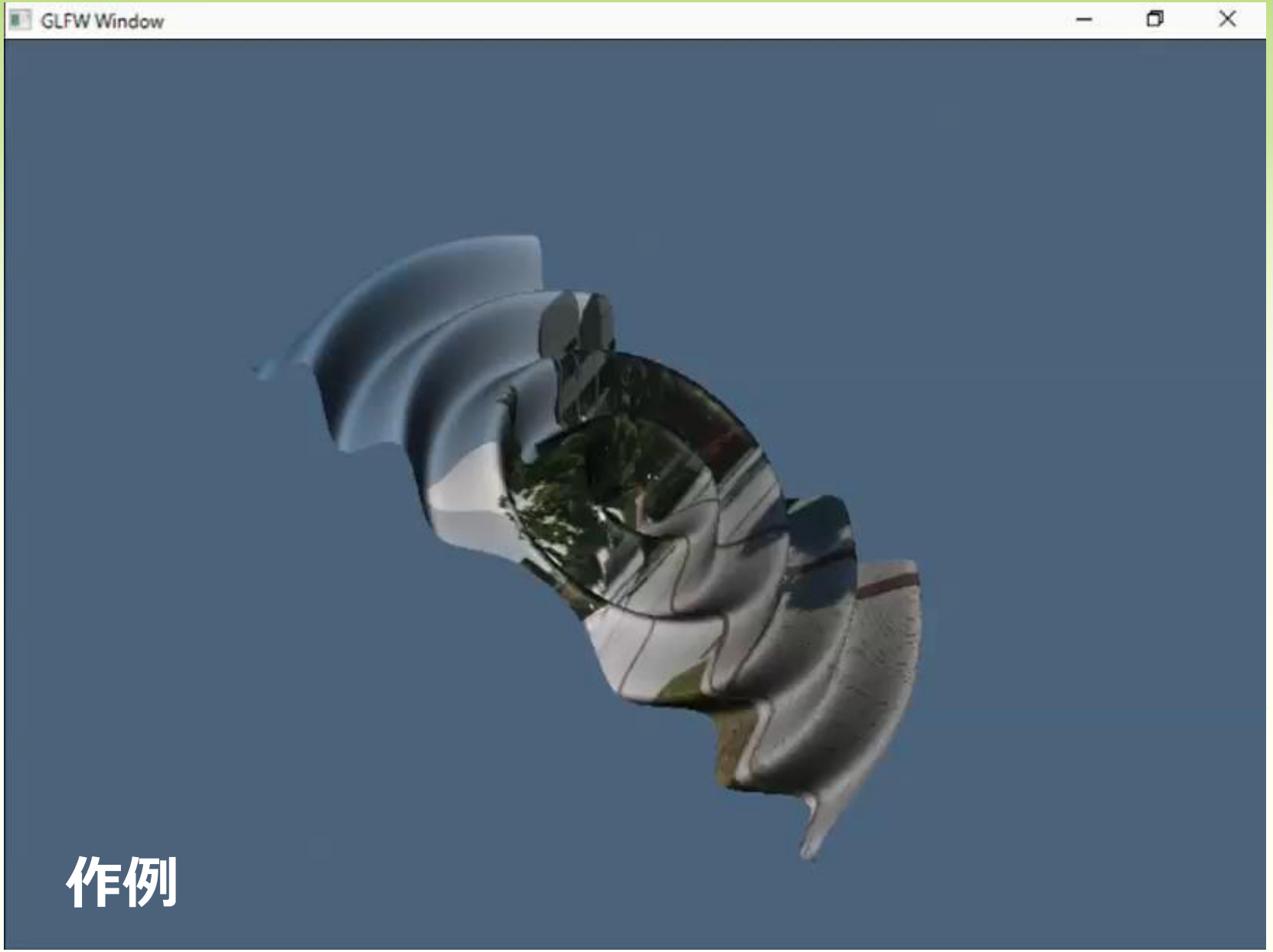
```
//
// 更新（毎回実行されます）
//
bool update()
{
  // 経過時間
  const double elaps(getTime() - start);

  // 経過時間が現在のフレームの時刻に達していたら
  if (elaps >= video.get(CV_CAP_PROP_POS_MSEC) * 0.001)
  {
    // 1フレーム取得する
    if (!video.grab())
    {
      // フレームが取得できなかったらムービーファイルを巻き戻す
      video.set(CV_CAP_PROP_POS_FRAMES, 0.0);

      // 経過時間をリセットする
      start = getTime();
    }
    else
    {
      （省略）
    }
  }

  // プログラムを終了しない
  return true;
}
```

- 現在の時刻から最初のフレームの時刻を引いて経過時間 elaps を求めます
- その経過時間 elaps がこれから取得しようとするフレームの時刻に達していたらフレームを取得します
  - フレームの時刻は CV_CAP_PROP_POS_MSEC プロパティで取得できます
  - ただしミリ秒単位なので秒単位に直すために 1/1000 (0.001) 倍します



課題6

- 画像を貼り付けるかわりにムービーファイルを貼り付けてください
  - ムービーファイル (video.mp4) はk可能なら自分で撮影したものなど他のものに入れ替えてください
  - 大学のコンピュータへの保存方法は教員に相談してください
- スクリーンショットを 06.png というファイル名でアップロードしてください



Web カメラから入力する

```
// ビデオ入力
cv::VideoCapture video;

//
// 設定（最初に一度だけ実行されます）
//
bool setup()
{
   （省略）

  // ビデオの取得開始
  video.open(0);

  // ムービーファイルの1フレームの一時保存先
  cv::Mat frame;

  // 最初のフレームを取得する
  video >> frame;

  // 色データを登録する
  createColor(frame.cols, frame.rows, frame.data);

  // セットアップに成功した
  return true;
}
```

デバイス番号

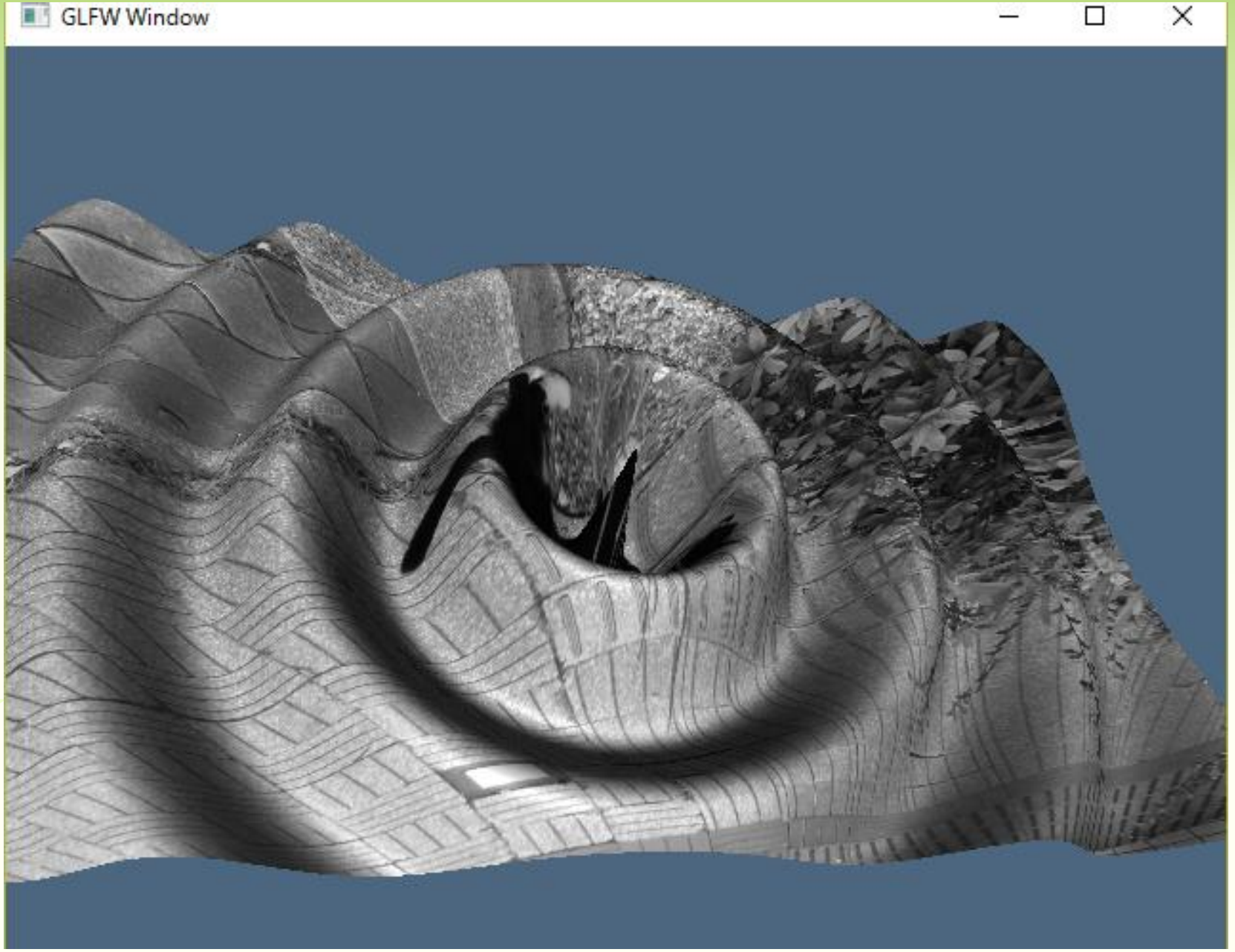
- ムービーファイルの代わりに Web カメラから入力することもできます
- open() メソッドの引数にデバイス番号（PC に Web かめらが一つしかなければ多分 0）を指定します
- cv::VideoCapture クラスはネットワークカメラにも対応しています
  - open() メソッドの引数に URL を指定できます（http://user:password@camera）

グレースケール化

```
//
// 更新（毎回実行されます）
//
bool update()
{
   （省略）

  // 経過時間がフレームの時刻に達していたら
  if (elaps >= video.get(CV_CAP_PROP_POS_MSEC) * 0.001)
  {
    // 1フレーム取得する
    if (!video.grab())
    {
      （省略）
    }
    else
    {
      // ムービーファイルのフレームの一時保存先
      cv::Mat frame;

      // 取得したフレームを保存する
      video.retrieve(frame);

      // グレースケール化
      cv::Mat gray;
      cv::cvtColor(frame, gray, CV_BGR2GRAY);
      cv::cvtColor(gray, frame, CV_GRAY2BGR);

      // 保存したフレームを転送する
      submitColor(frame.cols, frame.rows, frame.data);
    }
  }

  // プログラムを終了しない
  return true;
}
```

- cv::cvtColor() を使えば画像の色空間を変換できます
  - CV_BGR2GRAY を指定すればカラー画像をグレースケール画像（モノクロ画像）に変換します
  - ただしグレースケール画像は1チャンネルしかないので submitColor() の色データには使えません
  - そこでもう一度 cv::cvtColor() を使ってグレースケール画像をカラー画像に変換します (CV_GRAY2BGR)

実行結果

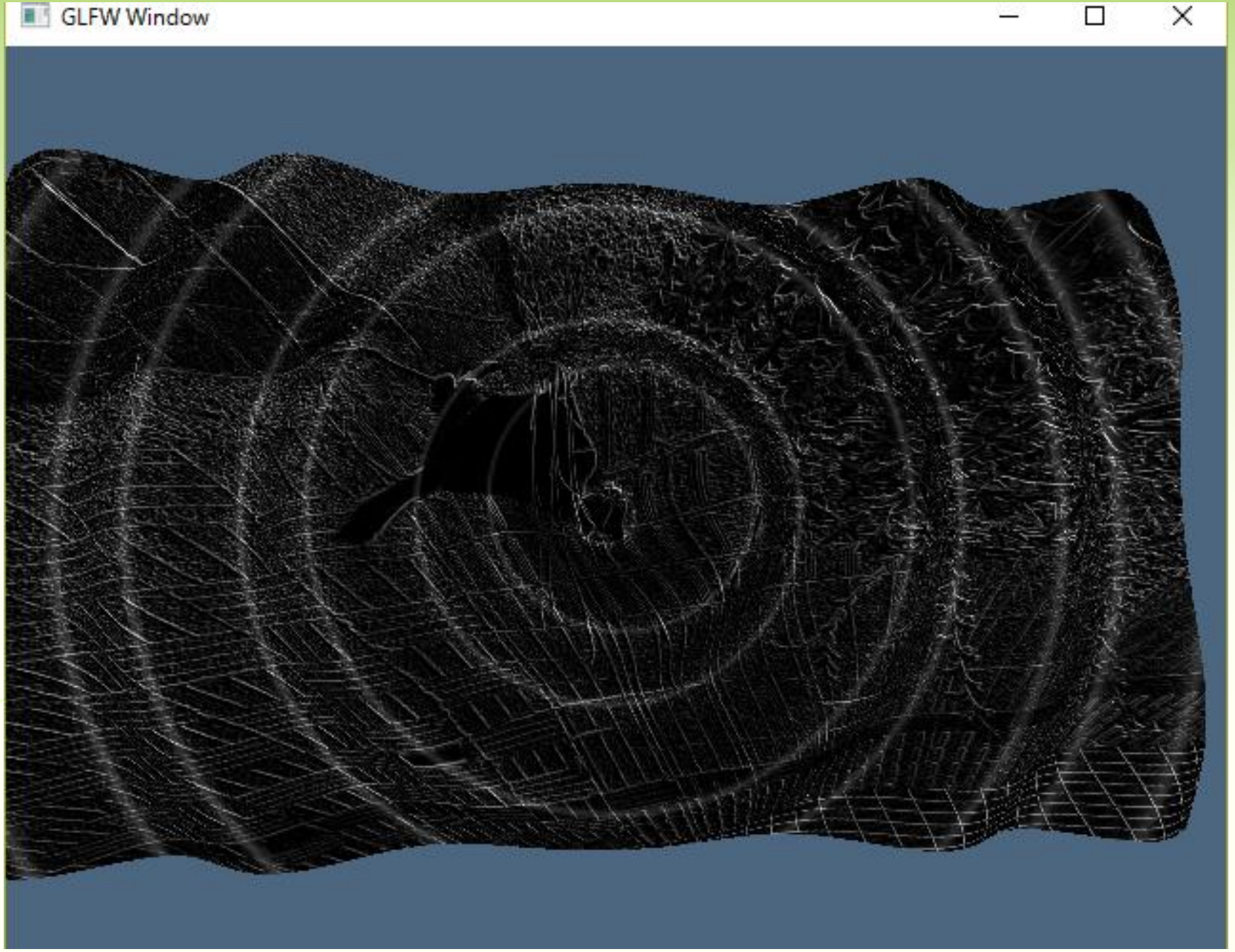


エッジ検出

```
（省略）

// ムービーファイルのフレームの一時保存先
cv::Mat frame;

// 取得したフレームを保存する
video.retrieve(frame);

// グレースケール化
cv::Mat gray;
cv::cvtColor(frame, gray, CV_BGR2GRAY);

// エッジ検出
cv::Mat sobelX, sobelY;
cv::Sobel(gray, sobelX, CV_8U, 1, 0);
cv::Sobel(gray, sobelY, CV_8U, 0, 1);
gray = (sobelX + sobelY) * 0.5;
cv::cvtColor(gray, frame, CV_GRAY2BGR);

// 保存したフレームを転送する
submitColor(frame.cols, frame.rows, frame.data);

（省略）
```

- エッジ検出にはいくつもの手法がありますが，ここでは Sobel オペレータを使ってみます
- Sobel オペレータは画像上のある方向に対する一次微分で，ここでは X 方向と Y 方向について微分した結果の平均を求めています

実行結果



顔検出

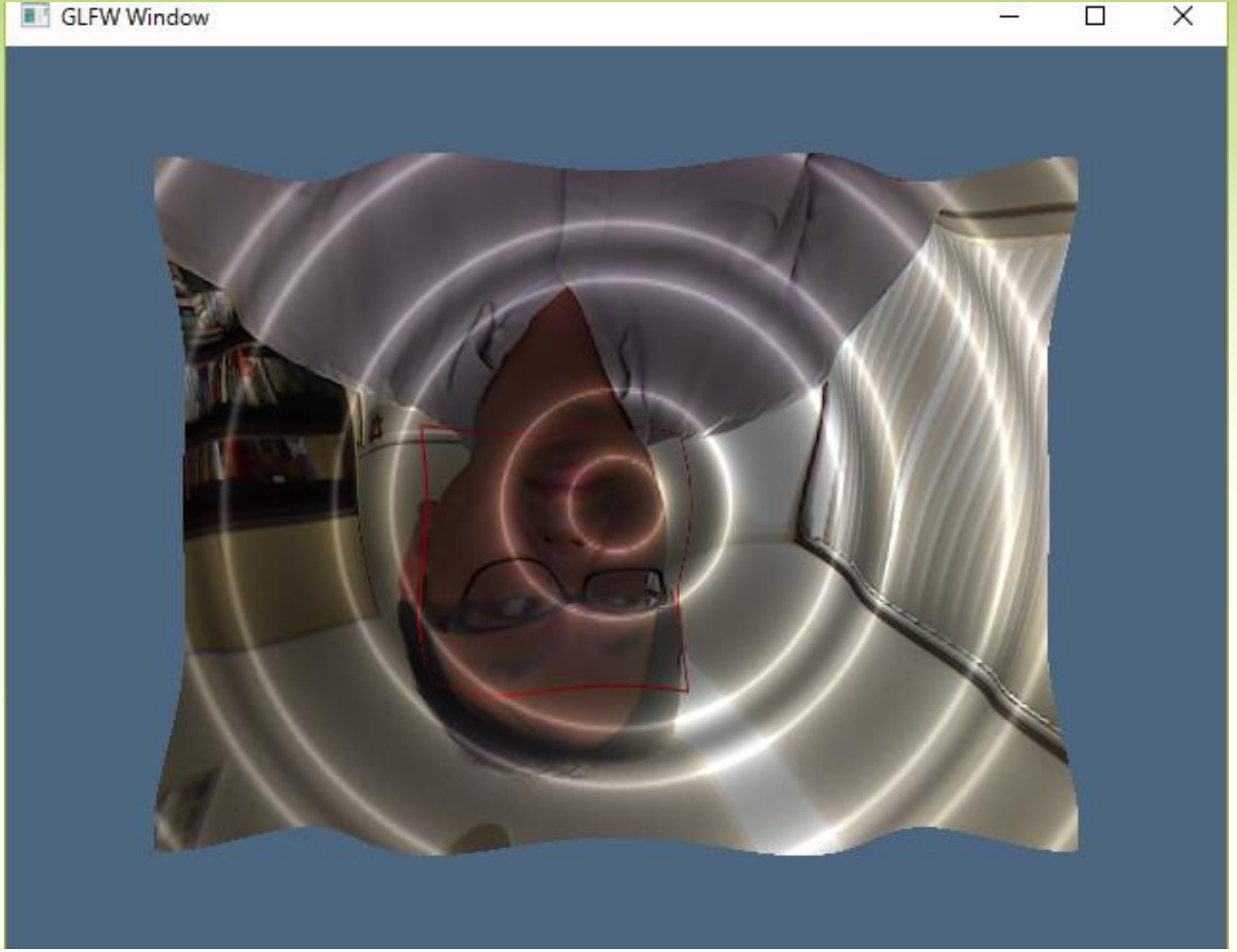
```
（省略）

// カスケード分類器
cv::CascadeClassifier ccls;

//
// 設定（最初に一度だけ実行されます）
//
bool setup()
{
（省略）

  // 色データを登録する
  createColor(frame.cols, frame.rows, frame.data);

  // 顔の分類器を読み込む
  ccls.load("../opencv/etc/haarcascades/haarcascade_frontalface_default.xml");

  // セットアップに成功した
  return true;
}
```

このかわりに "../opencv/etc/haarcascades/haarcascade_eye.xml" を使えば「目」の検出ができる

```
//
// 更新（毎回実行されます）
//
bool update()
{
   （省略）

  // 経過時間がフレームの時刻に達していたら
  if (elaps >= video.get(CV_CAP_PROP_POS_MSEC) * 0.001)
  {
    // 1フレーム取得する
    if (!video.grab())
    {
      （省略）
    }
    else
    {
      // ムービーファイルのフレームの一時保存先
      cv::Mat frame;

      // 取得したフレームを保存する
      video.retrieve(frame);

      // グレースケール化
      cv::Mat gray;
      cv::cvtColor(frame, gray, CV_BGR2GRAY);

      // 物体検出
      std::vector<cv::Rect> objects;
      ccls.detectMultiScale(gray, objects);
      for (auto &o : objects) cv::rectangle(frame, o, cv::Scalar(0, 0, 255));

      // 保存したフレームを転送する
      submitColor(frame.cols, frame.rows, frame.data);
    }
  }
```

実行結果

- 検出した顔に枠を付けます
  - Web カメラを使わないと面白くないです
- 図形の原点が左下（上が y の正方向）なのに対して画像の原点は左上（下が y の正方向）なので上下が反転しています
  - そこは自分で直してください



明るさを高さで表す

```
//
// 更新（毎回実行されます）
//
bool update()
{
   （削除）

  // 経過時間
  const double elaps(getTime() - start);

  // 経過時間がフレームの時刻に達していたら
  if (elaps >= video.get(CV_CAP_PROP_POS_MSEC) * 0.001)
  {
    // 1フレーム取得する
    if (!video.grab())
    {
      （省略）
    }
    else
    {
      // ムービーファイルのフレームの一時保存先
      cv::Mat frame;

      // 取得したフレームを保存する
      video.retrieve(frame);

      // グレースケール化
      cv::Mat gray;
      cv::cvtColor(frame, gray, CV_BGR2GRAY);

      （右に続く）
```

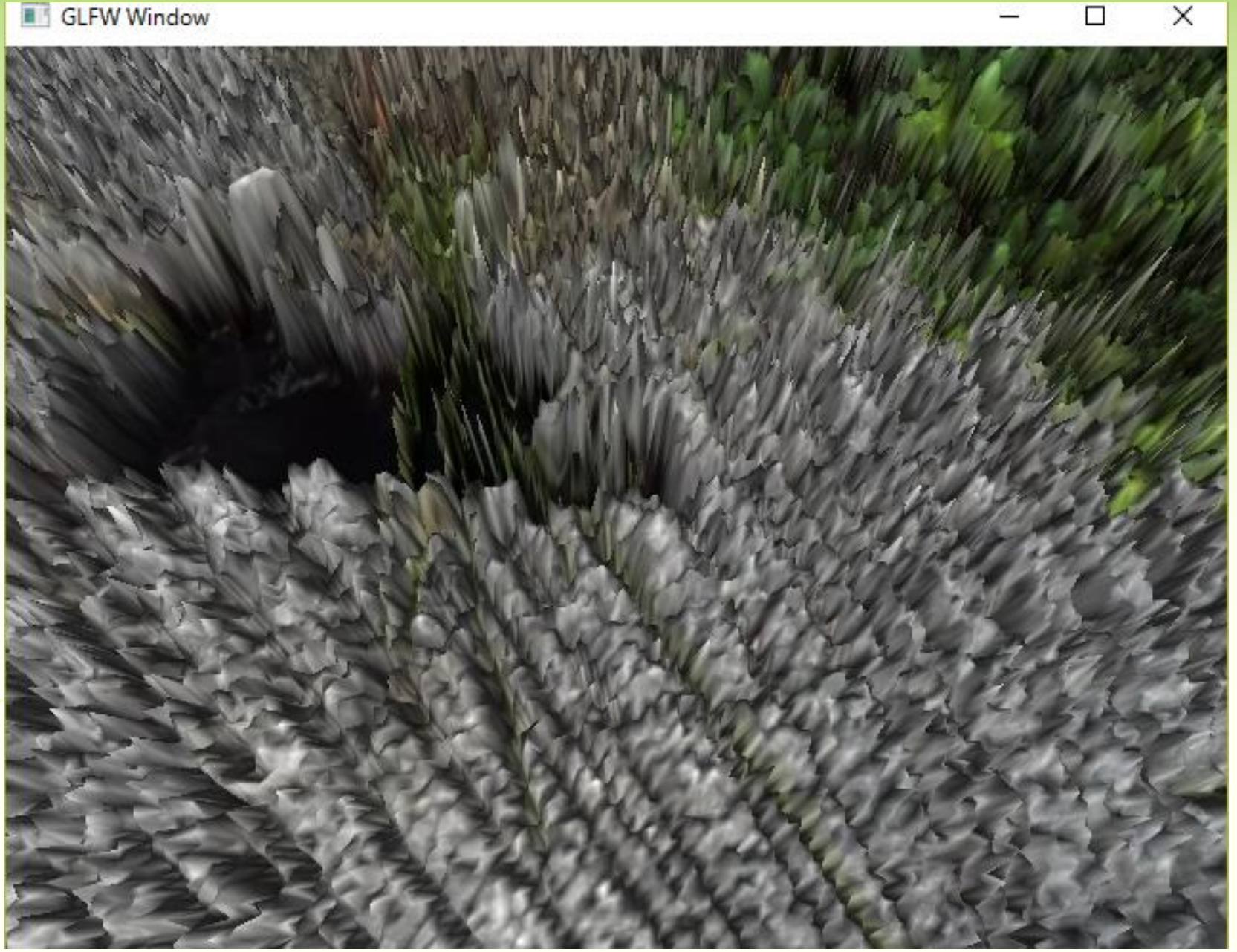
```
      （左から続く）

      // 点データの大きさに拡大縮小
      cv::resize(gray, gray, cv::Size(width, height));

      // グレースケールを点の座標値の z 値に使う
      for (int j = 0; j < height; ++j)
      {
        for (int i = 0; i < width; ++i)
        {
          // 頂点の位置
          point[j][i][0] = float(i * 2 - width + 1)
                         / float(height - 1);
          point[j][i][1] = float(j * 2 - height + 1)
                         / float(height - 1);
          point[j][i][2] = gray.at<unsigned char>(j, i) * 0.001f;
        }
      }

      // 位置データを転送する
      submitPoint(width, height, point);

      // 保存したフレームを転送する
      submitColor(frame.cols, frame.rows, frame.data);
    }
  }

  // プログラムを終了しない
  return true;
}
```

at<型>() メソッドは cv::Mat の要素にアクセスする

- image がグレースケールの場合:
  image.**at**<unsigned char>(j, i)
- image がカラー (BGR) の場合:
  image.**at**<cv::Vec3b>(j, i)[0] ・・・ 青
  image.**at**<cv::Vec3b>(j, i)[1] ・・・ 緑
  image.**at**<cv::Vec3b>(j, i)[2] ・・・ 赤

実行結果

- 例示したプログラムの一部を拡大しています



課題7

- オリジナルなビデオエフェクトを考えて実装してください
- 技術的なことでわからないことがあれば教員に相談してください
- スクリーンショットを 07.png というファイル名でアップロードしてください
- kadai1.cpp をアップロードしてください

- 「SNOW」みたいなもの
- 「スキャニメイト」みたいなもの
- 「Premier」や「After Effetcts」にあるようなもの
- その他オリジナルなビデオエフェクト