



# 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
  - ただし直接 OpenGL のプログラミングは行いません





# 課題ファイルのダウンロード

---

<https://www.wakayama-u.ac.jp/~tokoi/lecture/seminar1/>



コンピューター



Style Builder  
2016



ごみ箱



Epic Games  
Launcher



Mozilla  
Thunderbird



Unity 5.4.2f2  
(64-bit)



Mozilla Firefox



マイドキュメント



Google Chrome



質問



Google Earth



LayOut 2016



SketchUp 2016

Firefox



17:44  
2017/04/27

システム工学部 ポータルサイト  
http://portal.sys.wakayama-u.ac.jp/

https://www.wakayama-u.ac.jp/~tokoi/lecture/seminar1/  
を開く



### A803演習室

現在の授業ページへ

システム情報学センターのユーザ名

ユーザー名とパスワードを入力してください

https://www.wakayama-u.ac.jp がユーザー名とパスワードを要求しています。サイトからのメッセージ: "CIS Account"

ユーザー名:

パスワード:

OK キャンセル

システム情報学センターのパスワード

#### A803演習室予定表

	4/27(Thu)	4/28(Fri)
9:00		
10:00		
11:00		
12:00		
13:00	13:00~16:20 MDセミナー1	13:10~16:20 ウェブデザイン演習
14:00		
15:00		
16:00		
17:00		
18:00		

管理者 2016/11/16  
安江 2016/08/24

【8/29実施】演習室メンテナンスに伴う演習室利用停止について

フラット表示へ 1 2 3 4 5 次 ▶ 5件 ▼

#### 講義時間割一覧 (※当日の利用予定は右の予定表をご覧ください)

2017年 前期 シス工 情通 光メカ 精密 環境 デザイン A301 A601 A803

凡例 ●:1年 ●:2年 ●:3年

	月	火	水	木	金
1					

- システム工学部NEWS
- 4/18 和歌山大学大学院システム工...
  - 4/10 日刊工業新聞において聴覚メ...
  - 4/6 入学式が行われました
  - 3/27 和歌山大学システム工学部と...

- 和歌山大学NEWS
- 4/21 【OTR・観光学部】5/8(月)13時...

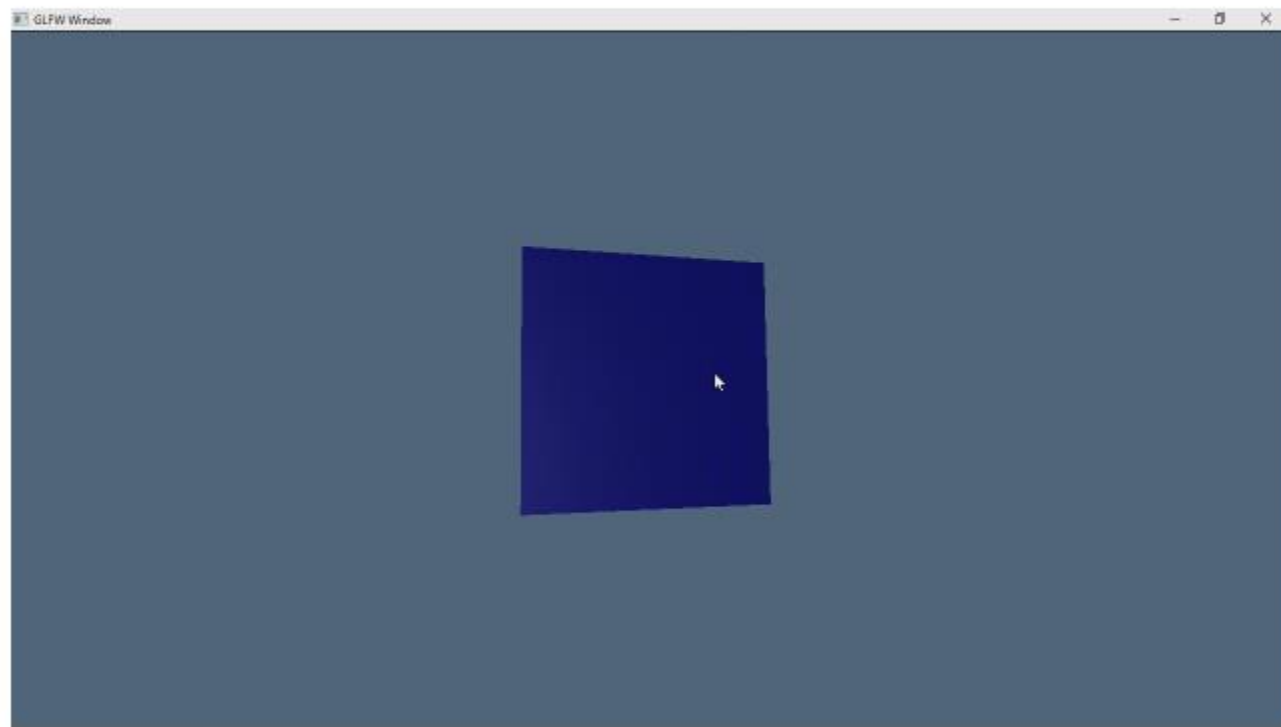
www.wakayama-u.ac.jp の応答を待っています...

# メディアデザイン セミナー1

## 課題の概要

こういうポリゴンを一枚表示するプログラムをテンプレートとして用意しています。

この演習の Web ページ



これをもとに次のようなビデオエフェクトを付けるプログラムを作成してください。



下にスクロール



この資料

## 課題の資料

- [グラフィックスと画像処理 課題資料 \(PDF\)](#)
  - 課題テンプレート - [seminar1.zip](#)
  - 補助資料 - [Visual Studio 2013 の新規プロジェクト作成 \(PDF\)](#)
  - 補助資料 - [GLFW による OpenGL 入門 \(PDF\)](#)
  - 補助資料 - [ゲームグラフィックス特論](#)
  - 参考資料 - [OpenGL - The Industry S](#)
  - 参考資料 - [OpenCV Library \(外部サイト\)](#)
  - 参考資料 - [OpenCV.jp \(外部サイト\)](#)
- 【以前の課題】[ばね質点モデル 課題資料 \(PDF\)](#)
  - 課題テンプレート (糸) - [spring.zip](#)
  - 課題テンプレート (布) - [cloth.zip](#)
- 【以前の課題】[GLUT による「手抜き」OpenGL 入門](#)

seminar1.zip をダウンロード

全画面領域切り取り(S)

## 課題の提出

完成した課題のファイルをアップロードしてください。ファイル名は“数字.拡張子”にしてください。

1. 「参照」(あるいは「選択」)をクリックして、アップロードするファイルを選択してください。(上限 50MB).

ファイル名:  ファイルが選択されていません。

2. ファイルの選択が完了したら、「送信」をクリックしてください。

- ファイルを指定せずに送信すれば、既にアップロードされているものが表示されます。
- アップロードできるファイルの形式(拡張子)は PNG, JPG, GIF, PDF, C, CPP です。
- 以前にアップロードした同じファイル名のファイルは上書きされます。
- 十分にテストしていないので、問題があれば教員に連絡してください。

## Lhaplus を使って開く

## 課題の資料

- [グラフィックスと画像処理 課題資料 \(PDF\)](#)
  - 課題テンプレート - [seminar1.zip](#)
  - 補助資料 - [Visual Studio 2013 の新規プロジェクト](#)
  - 補助資料 - [GLFW による OpenGL 入門 \(PDF\)](#)
  - 補助資料 - [ゲームグラフィックス特論宿題補助](#)
  - 参考資料 - [OpenGL - The Industry Standard fo](#)
  - 参考資料 - [OpenCV Library \(外部サイト\)](#)
  - 参考資料 - [OpenCV.jp \(外部サイト\)](#)
- 【以前の課題】[ばね質点モデル 課題資料 \(PDF\)](#)
  - 課題テンプレート (糸) - [spring.zip](#)
  - 課題テンプレート (布) - [cloth.zip](#)
- 【以前の課題】[GLUT による「手抜き」OpenGL 入門](#)

## 課題の提出

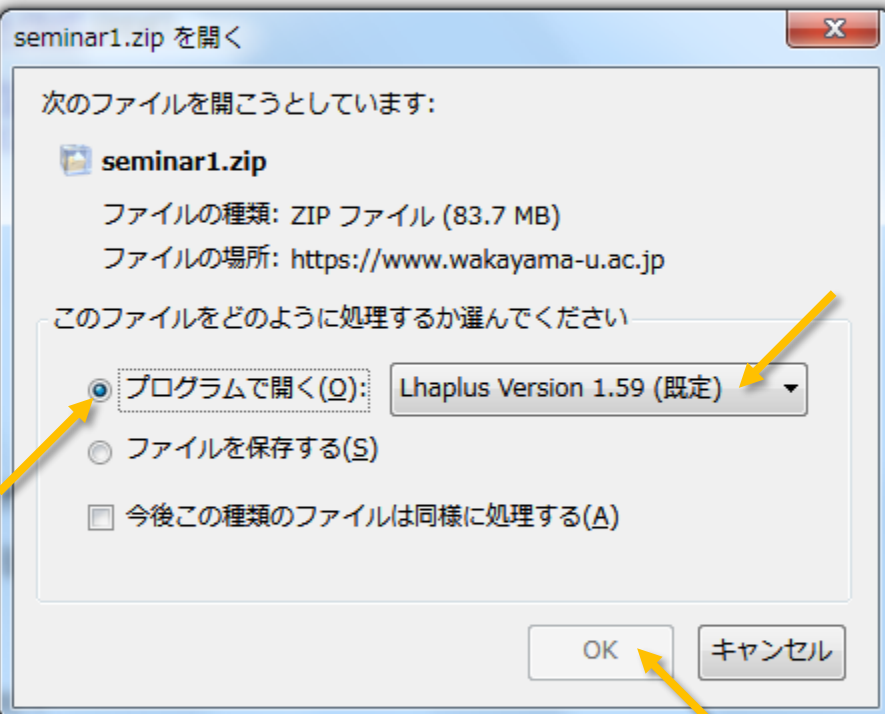
完成した課題のファイルをアップロードしてください。ファイル名

1. 「参照」(あるいは「選択」) をクリックして、アップロードす

ファイル名:  ファイルが選択されていません。

2. ファイルの選択が完了したら、「送信」をクリックしてくだ

- ファイルを指定せずに送信すれば、既にアップロードされているものが表示されます。
- アップロードできるファイルの形式(拡張子)は PNG, JPG, GIF, PDF, C, CPP です。
- 以前にアップロードした同じファイル名のファイルは上書きされます。
- 十分にテストしていないので、問題があれば教員に連絡してください。



## 課題の資料

- [グラフィックスと画像処理 課題資料 \(PDF\)](#)
  - 課題テンプレート - [seminar1.zip](#)
  - 補助資料 - [Visual Studio 2013 の新規プロジェクト作成 \(PDF\)](#)
  - 補助資料 - [GLFW による OpenGL 入門 \(PDF\)](#)
  - 補助資料 - [ゲームグラフィックス特論宿題補助プログラム](#)
  - 参考資料 - [OpenGL - The Industry Standard for High Performance Graphics \(外部サイト\)](#)
  - 参考資料 - [OpenCV Library \(外部サイト\)](#)
  - 参考資料 - [OpenCV.jp \(外部サイト\)](#)
- 【以前の課題】[ばね質点モデル 課題資料 \(PDF\)](#)
  - 課題テンプレート (糸) - [spring.zip](#)
  - 課題テンプレート (布) - [cloth.zip](#)
- 【以前の課題】[GLUT による「手抜き」OpenGL 入門](#)

## 課題の提出

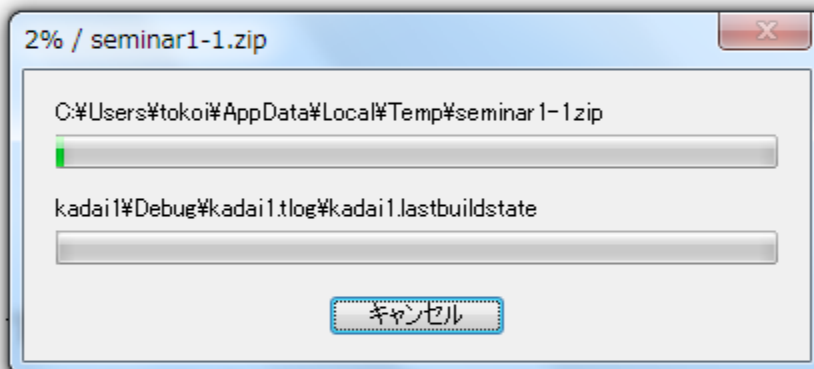
完成した課題のファイルをアップロードしてください。ファイル名は

1. 「参照」(あるいは「選択」) をクリックして、アップロードするファイルを選択してください。(上限 50MB).

ファイル名:  ファイルが選択されていません。

2. ファイルの選択が完了したら、「送信」をクリックしてください。

- ファイルを指定せずに送信すれば、既にアップロードされているものが表示されます。
- アップロードできるファイルの形式(拡張子)は PNG, JPG, GIF, PDF, C, CPP です。
- 以前にアップロードした同じファイル名のファイルは上書きされます。
- 十分にテストしていないので、問題があれば教員に連絡してください。



ダウンロード中

### 課題の資料

- グラフィックスと画像処理 課題
  - 課題テンプレート - sem
  - 補助資料 - Visual Studi
  - 補助資料 - GLFW (こよ
  - 補助資料 - ゲームグラ
  - 参考資料 - OpenGL - T
  - 参考資料 - OpenCV Lib
  - 参考資料 - OpenCV.jp (
- 【以前の課題】ばね質点モデル
  - 課題テンプレート (糸) -
  - 課題テンプレート (布) -
- 【以前の課題】GLUT による「手

### 課題の提出

完成した課題のファイルをアップロー

- 「参照」(あるいは「選択」) をク

ファイル名:  ファイルが

- ファイルの選択が完了したら、

- ファイルを指定せずに送信す
- アップロードできるファイルの形
- 以前にアップロードした同じフ
- 十分にテストしていないので、問題があれば教員に連絡してください。

File Explorer window showing the contents of the 'seminar1' folder on the desktop. The search bar contains 'seminar1の検索'. The file list includes folders for 'glfw', 'kadai1', and 'opencv', and files for '.gitignore' and 'seminar1.sln'. The 'seminar1.sln' file is selected and highlighted.

名前	更新日時	種類	サイズ
glfw	2017/04/27 17:57	ファイル フォル...	
kadai1	2017/04/27 17:57	ファイル フォル...	
opencv	2017/04/27 17:57	ファイル フォル...	
.gitignore	2017/04/27 0:37	テキスト ドキュ...	5 KB
seminar1.sln	2017/04/27 8:29	Microsoft Visual...	2 KB

Bottom status bar for 'seminar1.sln':  
 更新日時: 2017/04/27 8:29    作成日時: 2017/04/27 17:57  
 Microsoft Visual Studio Solution    サイズ: 1.26 KB

プロジェクトのフォルダが開く



コンピューター



Style Builder  
2016



ごみ箱



Unity 5.4.2f2  
(64-bit)



Epic Games  
Launcher



マイドキュメント



Google Earth



質問

プロジェクトのフォルダはデスクトップにある



LayOut 2016



seminar1



Mozilla Firefox



Google Chrome



Mozilla  
Thunderbird



SketchUp 2016





# Visual Studio 2017 の起動

---

ソリューションファイルを開く

### 課題の資料

- [グラフィックスと画像処理 課題](#)
  - 課題テンプレート - [seminar1](#)
  - 補助資料 - [Visual Studio](#)
  - 補助資料 - [GLFW \(による\)](#)
  - 補助資料 - [ゲームグラフィックス](#)
  - 参考資料 - [OpenGL - T](#)
  - 参考資料 - [OpenCV Lib](#)
  - 参考資料 - [OpenCV.jp](#)
- 【以前の課題】[ばね質点モデル](#)
  - 課題テンプレート (糸) -
  - 課題テンプレート (布) -
- 【以前の課題】[GLUT による「手](#)

### 課題の提出

完成した課題のファイルをアップロード

1. 「参照」(あるいは「選択」) をクリック

ファイル名:  ファイルが

2. ファイルの選択が完了したら、

- ファイルを指定せずに送信すると
- アップロードできるファイルの形
- 以前にアップロードした同じファイル
- 十分にテストしていないので、問題があれば教員に連絡してください。

File Explorer window showing the contents of the 'seminar1' folder on the desktop. The search bar contains 'seminar1の検索'. The file list is as follows:

名前	更新日時	種類	サイズ
glfw	2017/04/27 17:57	ファイルフォル...	
kadai1	2017/04/27 17:57	ファイルフォル...	
opencv	2017/04/27 17:57	ファイルフォル...	
.gitignore	2017/04/27 0:37	テキストドキュ...	5 KB
seminar1.sln	2017/04/27 8:29	Microsoft Visual...	2 KB

The 'seminar1.sln' file is selected and highlighted with a yellow arrow pointing to it. A yellow callout box with black text says: **seminar1.sln (ソリューションファイル) を開く**

At the bottom of the window, a summary for 'seminar1.sln' is shown:   
 seminar1.sln      更新日時: 2017/04/27 8:29      作成日時: 2017/04/27 17:57  
 Microsoft Visual Studio Solution      サイズ: 1.26 KB

### 課題の資料

- [グラフィックスと画像処理 課題](#)
  - 課題テンプレート - [semi](#)
  - 補助資料 - [Visual Studio](#)
  - 補助資料 - [GLFW \(による\)](#)
  - 補助資料 - [ゲームグラフィ](#)
  - 参考資料 - [OpenGL - T](#)
  - 参考資料 - [OpenCV Lib](#)
  - 参考資料 - [OpenCV.jp](#)
- 【以前の課題】[ばね質点モデル](#)
  - 課題テンプレート (糸) -
  - 課題テンプレート (布) -
- 【以前の課題】[GLUT による「手](#)

### 課題の提出

完成した課題のファイルをアップロー

1. 「参照」(あるいは「選択」) をク

ファイル名:  ファイルが

2. ファイルの選択が完了したら、

- ファイルを指定せずに送信す
- アップロードできるファイルの形
- 以前にアップロードした同じフ
- 十分にテストしていないので、問題があれば教員に連絡

# Visual Studio

整理 開く

★ お気に入り

- ダウンロード
- デスクトップ
- 最近表示した場所

ライブラリ

- ドキュメント
- ピクチャ
- ビデオ
- ミュージック

コンピューター

- ローカル ディス
- tokoi (¥fs-a) (
- work (¥fs-b) (
- A360 Drive

ネットワーク

- seminar1.slr
- Microsoft Vis

検索

サイズ

フォル...	
フォル...	
フォル...	
ドキュ...	5 KB
Visual...	2 KB

しばらく待つ

### 課題の資料

- グラフィックスと画像処理 課題
  - 課題テンプレート - semi
  - 補助資料 - Visual Studio
  - 補助資料 - GLFW (による)
  - 補助資料 - ゲームグラフィ
  - 参考資料 - OpenGL - T
  - 参考資料 - OpenCV Lib
  - 参考資料 - OpenCV.jp (
- 【以前の課題】ばね質点モデル
  - 課題テンプレート (糸) -
  - 課題テンプレート (布) -
- 【以前の課題】GLUT による「手

### 課題の提出

完成した課題のファイルをアップロー

- 「参照」(あるいは「選択」) をク

ファイル名:  ファイルが

- ファイルの選択が完了したら、

- ファイルを指定せずに送信す
- アップロードできるファイルの形
- 以前にアップロードした同じフ
- 十分にテストしていないので、問題があれば教員に連絡してください。

## Visual Studio

ようこそ。  
すべての開発者サービスをご利用ください。

サインインして、Azure クレジットの使用開始、プライベート Git リポジトリへのコードの発行、設定の同期、IDE のロック解除を行います。

[詳細の表示](#)

アカウントをお持ちでない場合、[サインアップ](#)してください

[後で行う。](#)

マイクロソフトアカウントを  
 ・ 持っていない場合は「後で行う。」  
 ・ 持っていれば「サインイン(I)」

### 課題の資料

- グラフィックスと画像処理 課題
  - 課題テンプレート - semi
  - 補助資料 - Visual Studio
  - 補助資料 - GLFW (による)
  - 補助資料 - ゲームグラフィ
  - 参考資料 - OpenGL - T
  - 参考資料 - OpenCV Lib
  - 参考資料 - OpenCV.jp (
- 【以前の課題】ばね質点モデル
  - 課題テンプレート (糸) -
  - 課題テンプレート (布) -
- 【以前の課題】GLUT による「手

### 課題の提出

完成した課題のファイルをアップロー

- 「参照」(あるいは「選択」) をク  
ファイル名:  ファイルが
- ファイルの選択が完了したら、

- ファイルを指定せずに送信す
- アップロードできるファイルの形
- 以前にアップロードした同じフ
- 十分にテストしていないので、問題があれば教員に連絡してください。

The screenshot shows the Visual Studio installation wizard. The main window title is "Visual Studio" and the subtitle is "慣れた環境で開始します". Under "開発設定(V):", the "全般" (General) option is selected. Below this, "配色テーマの選択" (Choose a color theme) is shown with three options: "淡色" (Light), "濃色" (Dark), and "青" (Blue). The "青" theme is selected. A dropdown menu is open, showing a list of themes: "全般", "JavaScript", "Visual Basic", "Visual C#", "Visual C++", "Web 開発", "Web 開発 (コードのみ)", and "全般". A yellow arrow points from the "全般" dropdown to the "Visual C++" option in the list. At the bottom, there is a button labeled "Visual Studio の開始(S)".

配色はお好みで

Visual C++ を選んだという前提で説明します

### 課題の資料

- [グラフィックスと画像処理 課題](#)
  - 課題テンプレート - [seminar1](#)
  - 補助資料 - [Visual Studio](#)
  - 補助資料 - [GLFW \(による\)](#)
  - 補助資料 - [ゲームグラフィックス](#)
  - 参考資料 - [OpenGL - T](#)
  - 参考資料 - [OpenCV Lib](#)
  - 参考資料 - [OpenCV.jp](#)
- 【以前の課題】[ばね質点モデル](#)
  - 課題テンプレート (糸) -
  - 課題テンプレート (布) -
- 【以前の課題】[GLUT による「手](#)

### 課題の提出

完成した課題のファイルをアップロード

1. 「参照」(あるいは「選択」) をクリック

ファイル名:  ファイルが選択

2. ファイルの選択が完了したら、「送信」

- ファイルを指定せずに送信すると、
- アップロードできるファイルの形式が、
- 以前にアップロードした同じファイルが、
- 十分にテストしていないので、問題があれば教員に連絡してください。

整理
開く

★ お気に入り

- ダウンロード
- デスクトップ
- 最近表示した場所

ライブラリ

- ドキュメント
- ピクチャ
- ビデオ
- ミュージック

コンピューター

- ローカル ディスク
- tokoi (¥¥fs-a) (C:)
- work (¥¥fs-b) (D:)
- A360 Drive

ネットワーク

- seminar1.sl
- Microsoft Vis

## Visual Studio

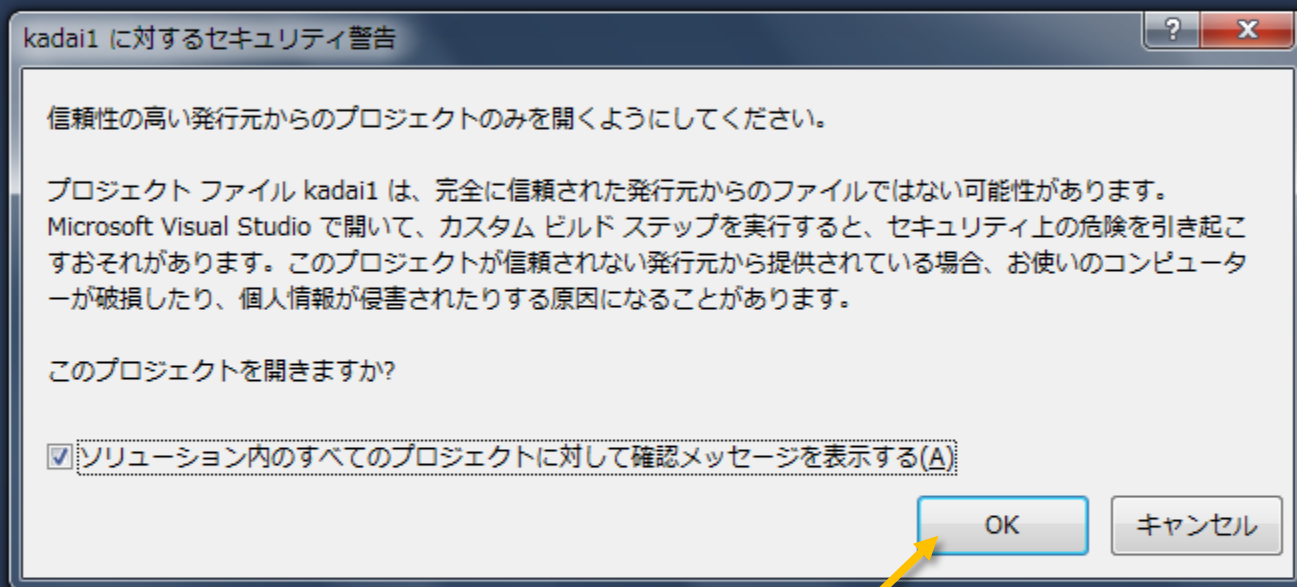
初めて使用するための準備をしています

この処理には数分かかる場合があります。

しばらく待つ

検索

名前	サイズ
フォル...	
フォル...	
フォル...	
ドキュ...	5 KB
Visual...	2 KB



とりあえず信用してください

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

ソリューション... クラス ビュー プロパティ... チーム エク...

# Visual Studio 2017 起動



# 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

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

ソリューション エクスプローラー

ソリューション... クラス ビュー プロパティ... チーム エク...

kadai1 の▶をクリック

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)
1 // 共通の関数
2 #include "common.h"
3
4 // 標準ライブラリ
5 #include <cmath>
6
7 // メッシュの幅と高さ
8 const int width(100), height(100);
9
10 // 点データ
11 float point[height][width][3];
12
13 // 色データ
14 unsigned char color[height][width][3];
15
16 // (一度だけ実行されます)
17
18 //
19 bool setup()
20 {
21     for (int y = 0; y < height; ++y)
22     {
23         for (int x = 0; x < width; ++x)
24         {
25             // x 方向のパラメータ (0 ≤ u ≤ 1)
26             const float u(float(x) / float(width - 1));
27
28             // y 方向のパラメータ (0 ≤ v ≤ 1)
29             const float v(float(y) / float(height - 1));
30
31             // 頂点の位置
32             point[y][x][0] = u * 2.0f - 1.0f;
33             point[y][x][1] = v * 2.0f - 1.0f;
34             point[y][x][2] = 0.0f;
35
36             // 頂点の色
37         }
38     }
39 }
```

このソースファイル kadai1.cpp を修正する

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

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... クラス ビュー プロパティ... チーム エク...

実行 (デバッグ)

```
1 // 共通の関数
2 #include "common.h"
3
4 // 標準ライブラリ
5 #include <cmath>
6
7 // メッシュの幅と高さ
8 const int width(100), height(100);
9
10 // 点データ
11 float point[height][width][3];
12
13 // 色データ
14 unsigned char color[height][width][3];
15
16 //
17 // 設定 (最初に一度だけ実行されます)
18 //
19 bool setup()
20 {
21     for (int y = 0; y < height; ++y)
22     {
23         for (int x = 0; x < width; ++x)
24         {
25             // x 方向のパラメータ (0 ≤ u ≤ 1)
26             const float u(float(x) / float(width - 1));
27
28             // y 方向のパラメータ (0 ≤ v ≤ 1)
29             const float v(float(y) / float(height - 1));
30
31             // 頂点の位置
32             point[y][x][0] = u * 2.0f - 1.0f;
33             point[y][x][1] = v * 2.0f - 1.0f;
34             point[y][x][2] = 0.0f;
35
36             // 頂点の色
37         }
38     }
39 }
```

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

kadai1 (グローバルスコープ)

```
1 // 共通の関数
2 #include <math.h>
3
4 //
5 #include <string.h>
6
7 //
8 const int width = 100;
9
10 //
11 float u = 0.5f;
12
13 //
14 unsigned char *img;
15
16 //
17 int main()
18 {
19     //
20     //
21     //
22     //
23     //
24     //
25     //
26     //
27     //
28     //
29     const float v(float (y) / float (height - 1));
30
31     // 頂点の位置
32     point [y][x][0] = u * 2.0f - 1.0f;
33     point [y][x][1] = v * 2.0f - 1.0f;
34     point [y][x][2] = 0.0f;
35     // 頂点の色
```

Microsoft Visual Studio

このプロジェクトは変更されています(I):

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

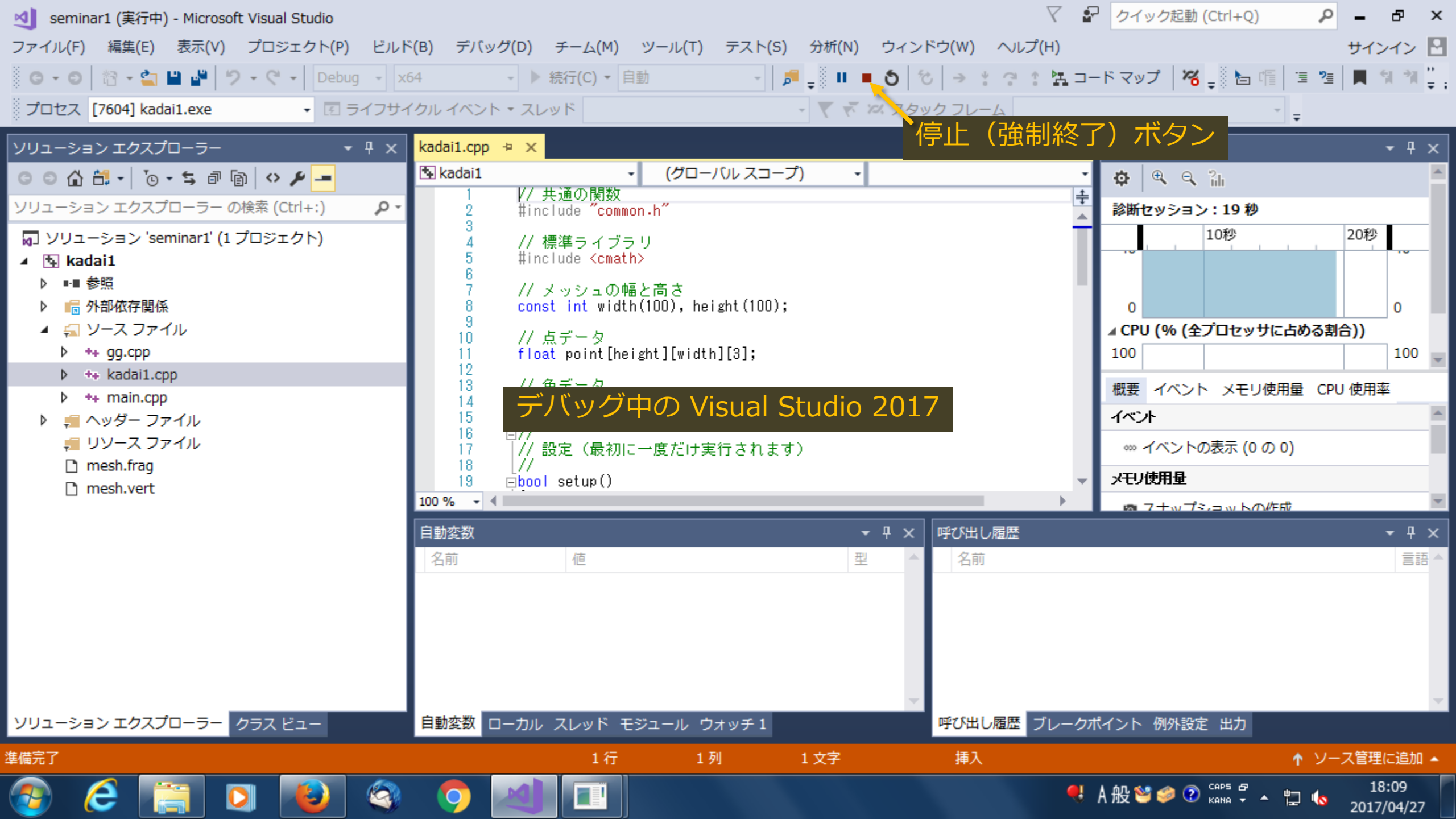
クローズボタンをクリックするか Esc キーをタイプすれば終了する

ビルドに成功すれば実行される

診断ツール  
診断セッション: 41 秒  
CPU (%) (全プロセッサに占める割合)  
100 | 100  
概要 イベント メモリ使用量 CPU 使用率  
イベント  
イベントの表示 (0 の 0)  
メモリ使用量  
スナップショットの作成

呼び出し履歴  
名前 言語

呼び出し履歴 ブレークポイント 例外設定 出力



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

```
1 // 共通の関数
2 #include "common.h"
3
4 // 標準ライブラリ
5 #include <cmath>
6
7 // メッシュの幅と高さ
8 const int width(100), height(100);
9
10 // 点データ
11 float point[height][width][3];
12
13 // 色データ
14
15
16
17 // 設定 (最初に一度だけ実行されます)
18
19 bool setup()
```

停止 (強制終了) ボタン

デバッグ中の Visual Studio 2017

診断セッション: 19 秒

0 10秒 20秒

0 0

▲ CPU (% (全プロセッサに占める割合))

100 100

概要 イベント メモリ使用量 CPU 使用率

イベント

イベントの表示 (0 の 0)

メモリ使用量

マップシートの作成

自動変数

名前	値	型
----	---	---

呼び出し履歴

名前	言語
----	----

ソリューション全体の保存

- 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

```
1 // 共通の関数
2 #include "common.h"
3
4 // 標準ライブラリ
5 #include <cmath>
6
7 // メッシュの幅と高さ
8 const int width(100), height(100);
9
10 // 点データ
11 float point[height][width][3];
12
13 // 色データ
14 unsigned char color[height][width][3];
15
16 //
17 // 設定 (最初に一度だけ実行されます)
18 //
19 bool setup()
20 {
21     for (int y = 0; y < height; ++y)
```

100 %

出力

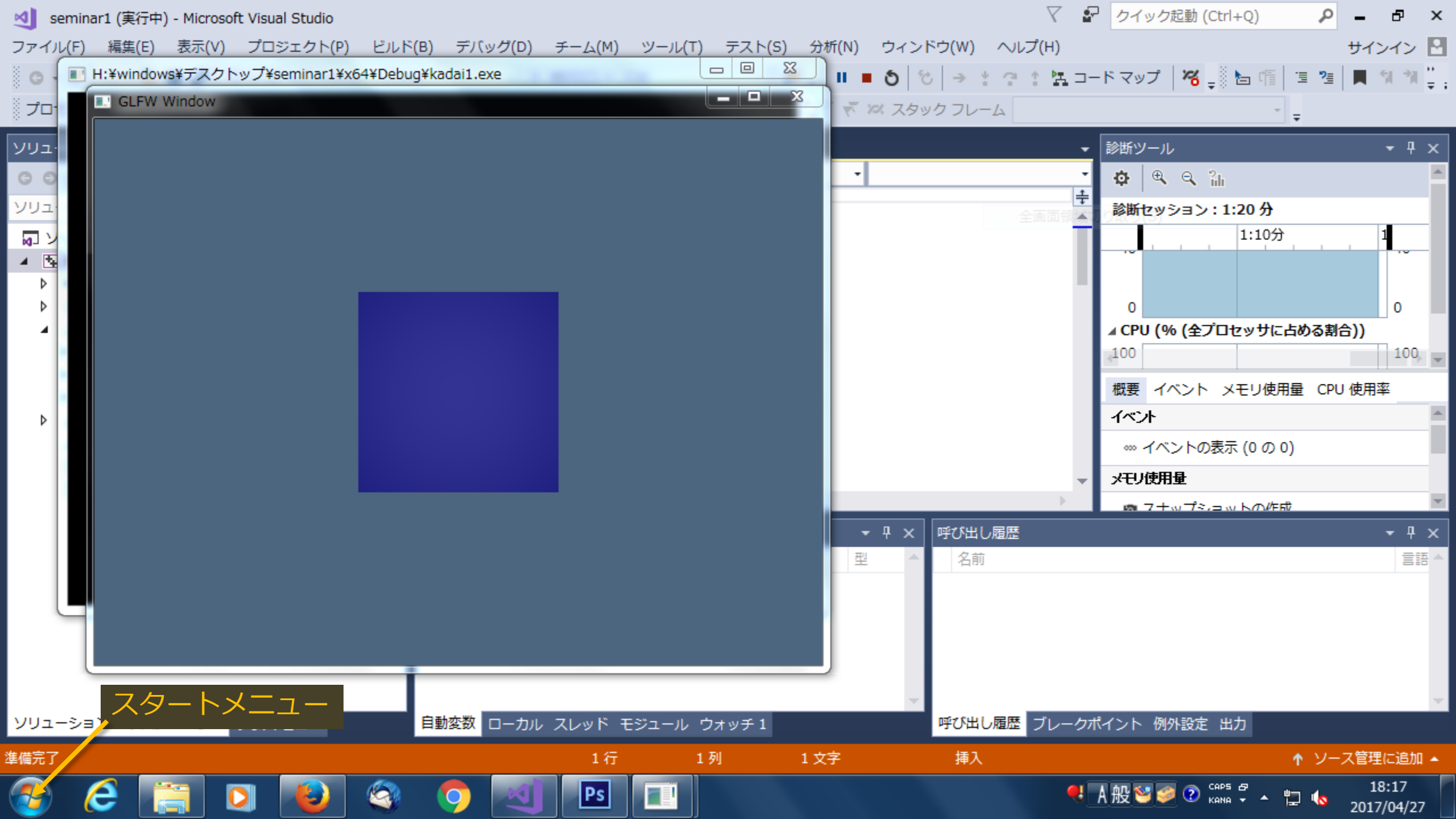
出力元(S): デバッグ

```
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。
```



# スクリーンショットの取得

Snipping Tool を使う



スタートメニュー

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

H:\windows\デスクトップ\seminar1\x64\Debug\kadai1.exe

GLFW Window

コードマップ  
スタックフレーム

診断ツール

診断セッション: 1:20分

1:10分

0 0

▲ CPU (% (全プロセッサに占める割合))

100 100

概要 イベント メモリ使用量 CPU 使用率

イベント

∞ イベントの表示 (0 の 0)

メモリ使用量

スナップショットの作成

呼び出し履歴

型	名前	言語
---	----	----

ソリューション 自動変数 ローカル スレッド モジュール ウォッチ 1 呼び出し履歴 ブレークポイント 例外設定 出力

準備完了 1行 1列 1文字 挿入 ソース管理に追加



- Excel 2016
- Word 2016
- PowerPoint 2016
- Internet Explorer
- Mozilla Firefox
- Google Chrome
- Mozilla Thunderbird

床井 浩平

コンピューター

コントロール パネル

デバイスとプリンター

既定のプログラム

ヘルプとサポート

ファイル名を指定して実行...

すべてのプログラム

プログラムとファイルの検索

シャットダウン

すべてのプログラム



スレッド モジュール ウォッチ 1

1行 1列 1文字 挿入

クイック起動 (Ctrl+Q)

コードマップ

スタックフレーム

診断ツール

診断セッション: 4:24分

4:10分 4:20分

0 0

▲ CPU (% (全プロセッサに占める割合))

100 100

概要 イベント メモリ使用量 CPU 使用率

イベント

∞ イベントの表示 (0 の 0)

メモリ使用量

スナップショットの作成

呼び出し履歴

型	名前	言語

呼び出し履歴 ブレークポイント 例外設定 出力



Windows Start menu search results for 'Snipping Tool':

- TeraPad
- Unity 5.4.2f2 (64-bit)
- Visual Studio 2017
- Windows Kits
- アクセサリ
- Snipping Tool** (highlighted with a yellow arrow)
- エクスプローラー
- コマンド プロンプト
- サウンド レコーダー
- ネットワーク プロジェクターへの接続
- はじめに
- ファイル名を指定して実行
- プロジェクターへの接続
- ペイント
- メモ帳
- リモート デスクトップ接続
- ワードパッド
- 数式入力パネル
- 電卓
- 同期センター
- 付箋
- Tablet PC
- Windows PowerShell
- コンピューターの簡単操作
- システム ツール

検索: プログラムとファイルの検索

Windows Start menu search results for '床井 浩平':

- 床井 浩平
- コンピューター
- コントロール パネル
- デバイスとプリンター
- 既定のプログラム
- ファイル名を指定して実行...

Snipping Tool は「アクセサリ」の中にある

Visual Studio interface showing a code editor window with a dark background and a stack frame window.

スタックフレーム

Visual Studio interface showing the '診断ツール' (Diagnostic Tools) window.

診断セッション: 5:04 分

4:50分 5:00分

0 0

▲ CPU (% (全プロセッサに占める割合))

100 100

概要 イベント メモリ使用量 CPU 使用率

イベント

∞ イベントの表示 (0 の 0)

メモリ使用量

スナップショットの作成

Visual Studio interface showing the '呼び出し履歴' (Call Stack) window.

呼び出し履歴

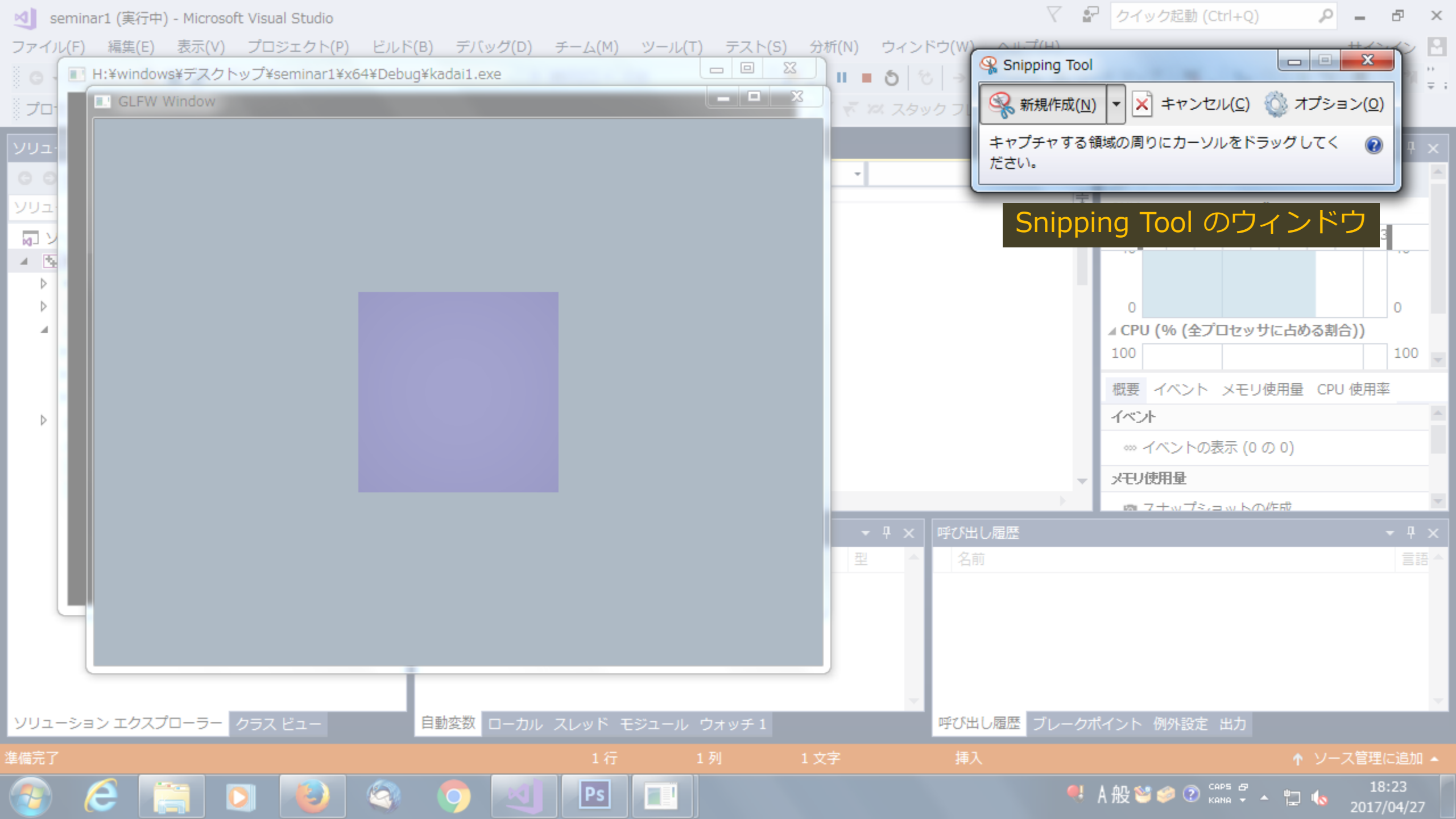
型	名前	言語

呼び出し履歴 ブレークポイント 例外設定 出力

Visual Studio interface showing the 'スレッド モジュール ウォッチ 1' (Thread Module Watch 1) window.

スレッド モジュール ウォッチ 1

1行 1列 1文字 挿入



Snipping Tool

新規作成(N) キャンセル(C) オプション(O)

キャプチャする領域の周りにカーソルをドラッグしてください。

Snipping Tool のウィンドウ

0 0

▲ CPU (% (全プロセッサに占める割合))

100 100

概要 イベント メモリ使用量 CPU 使用率

イベント

∞ イベントの表示 (0 の 0)

メモリ使用量

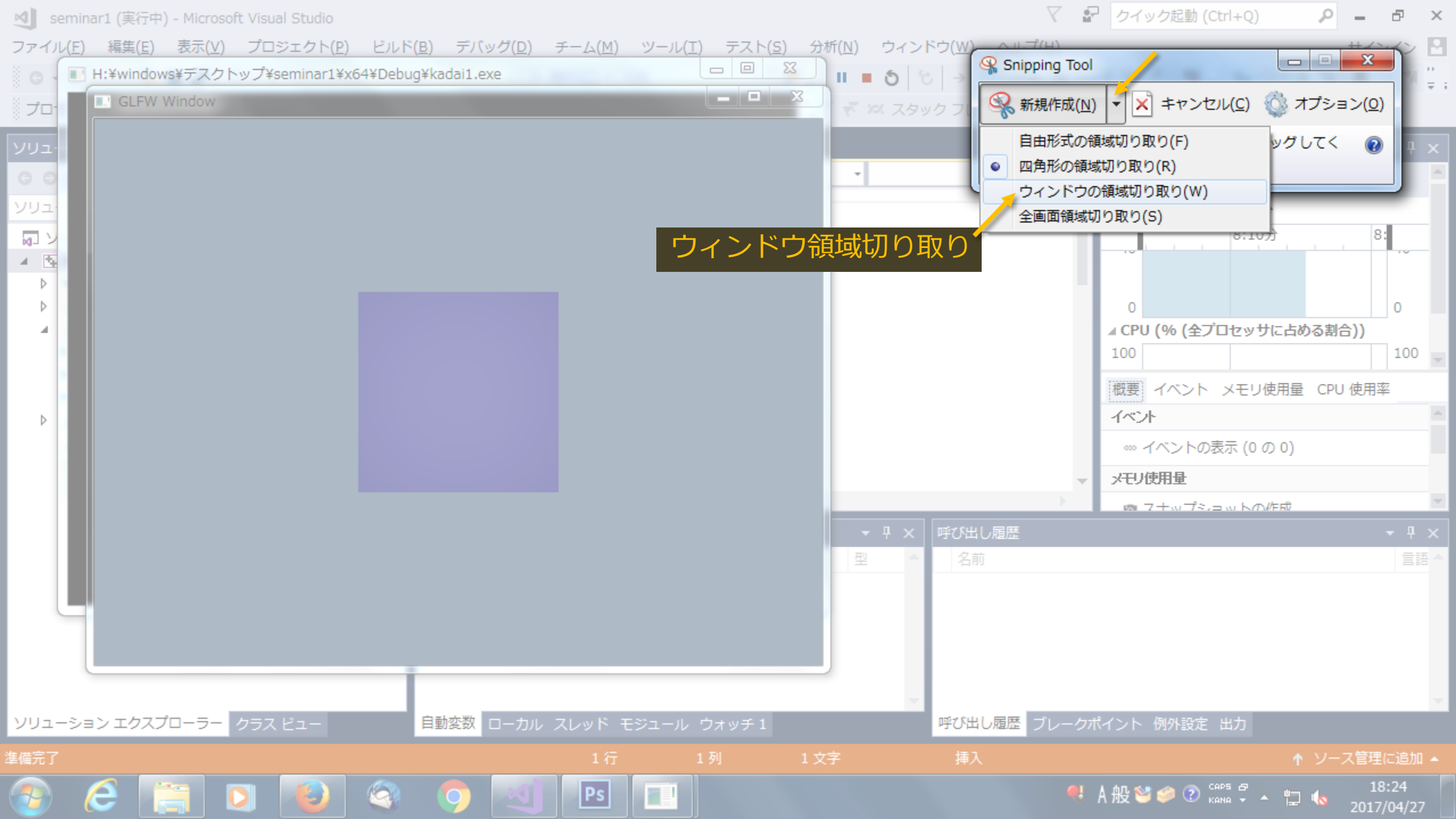
スナップショットの作成

呼び出し履歴

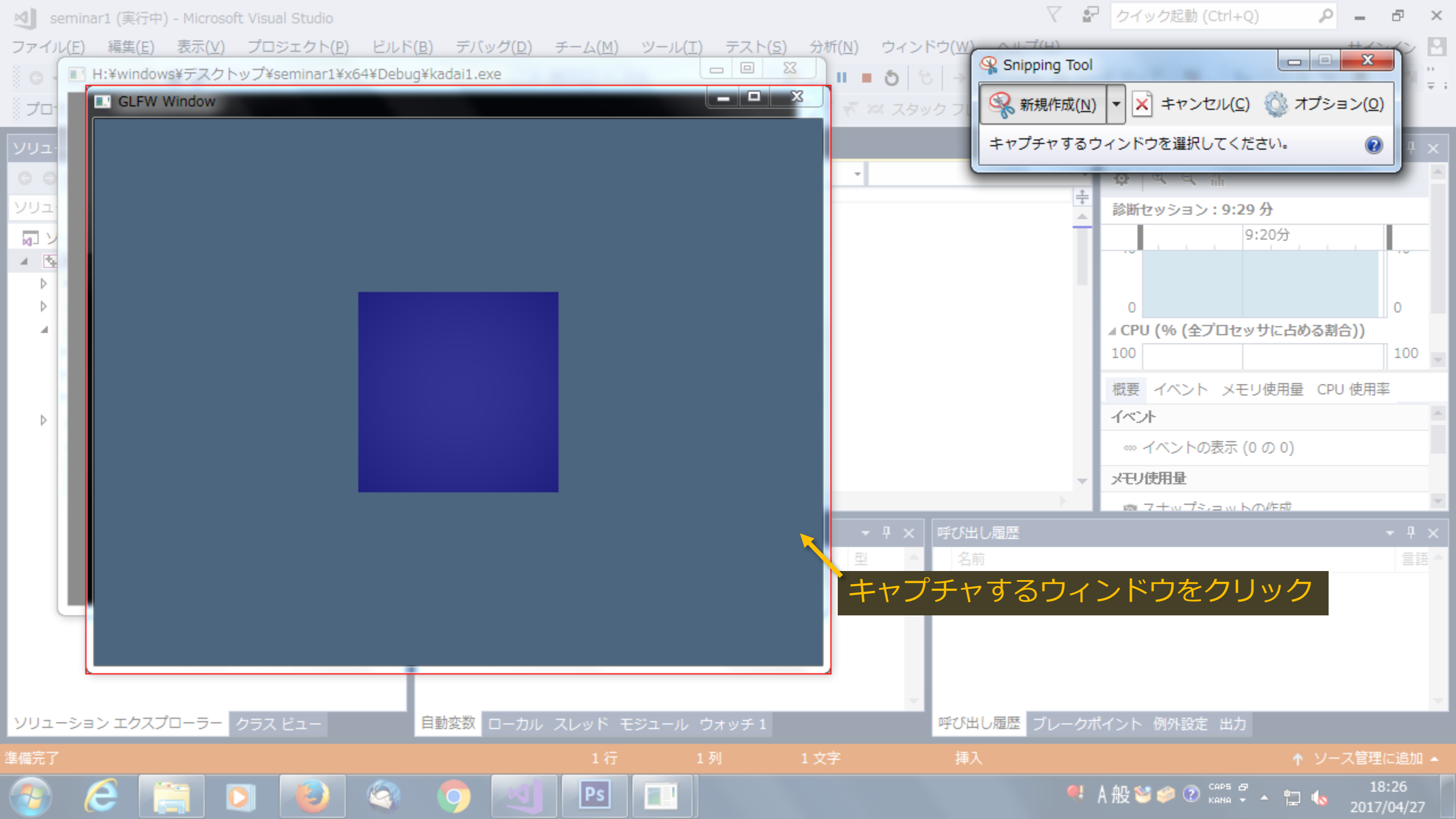
型	名前	言語
---	----	----

ソリューション エクスプローラー クラス ビュー 自動変数 ローカル スレッド モジュール ウォッチ 1 呼び出し履歴 ブレークポイント 例外設定 出力

準備完了 1行 1列 1文字 挿入 ソース管理に追加



ウィンドウ領域切り取り



Snipping Tool

新規作成(N) キャンセル(C) オプション(O)

キャプチャするウィンドウを選択してください。

キャプチャするウィンドウをクリック

診断セッション : 9:29 分

9:20分

0 0

▲ CPU (% (全プロセッサに占める割合))

100 100

概要 イベント メモリ使用量 CPU 使用率

イベント

∞ イベントの表示 (0 の 0)

メモリ使用量

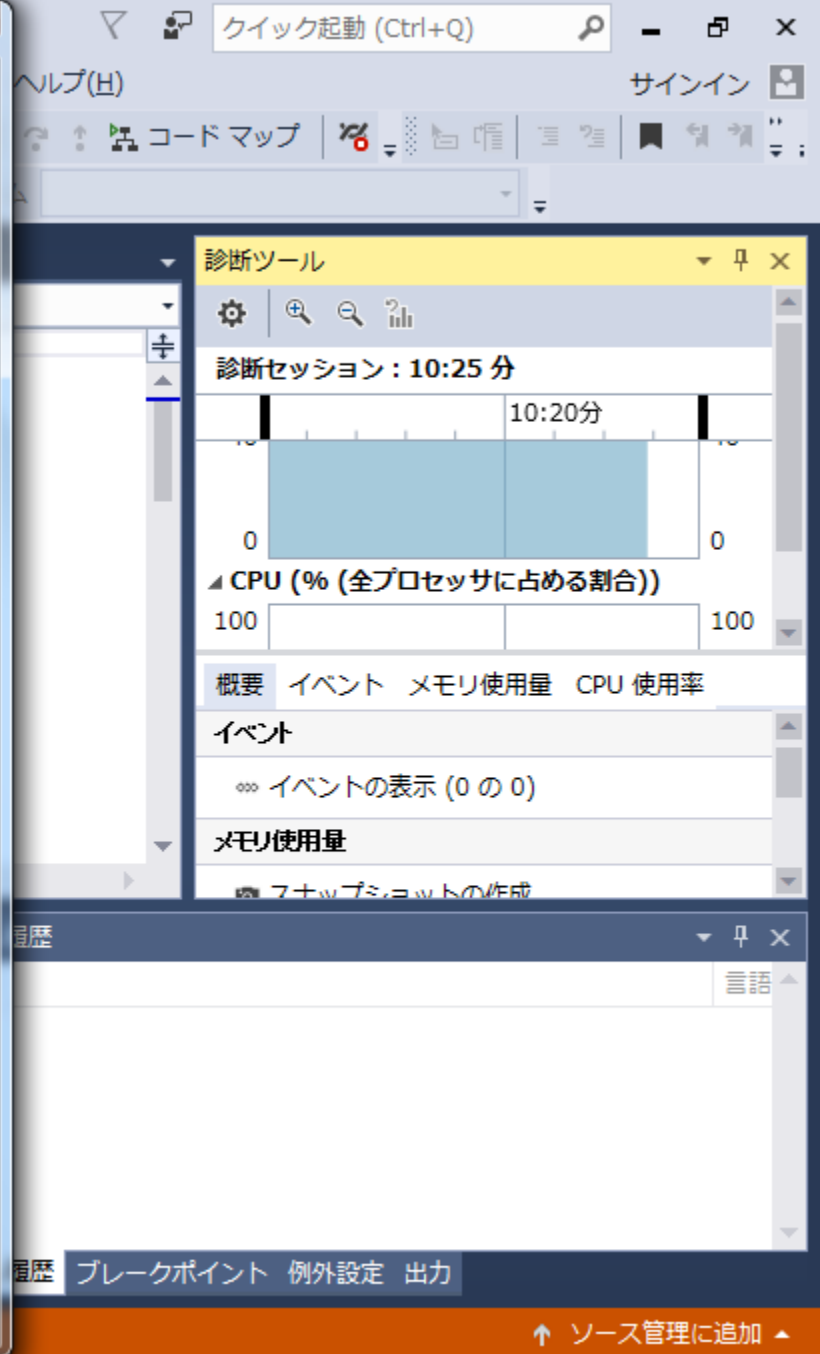
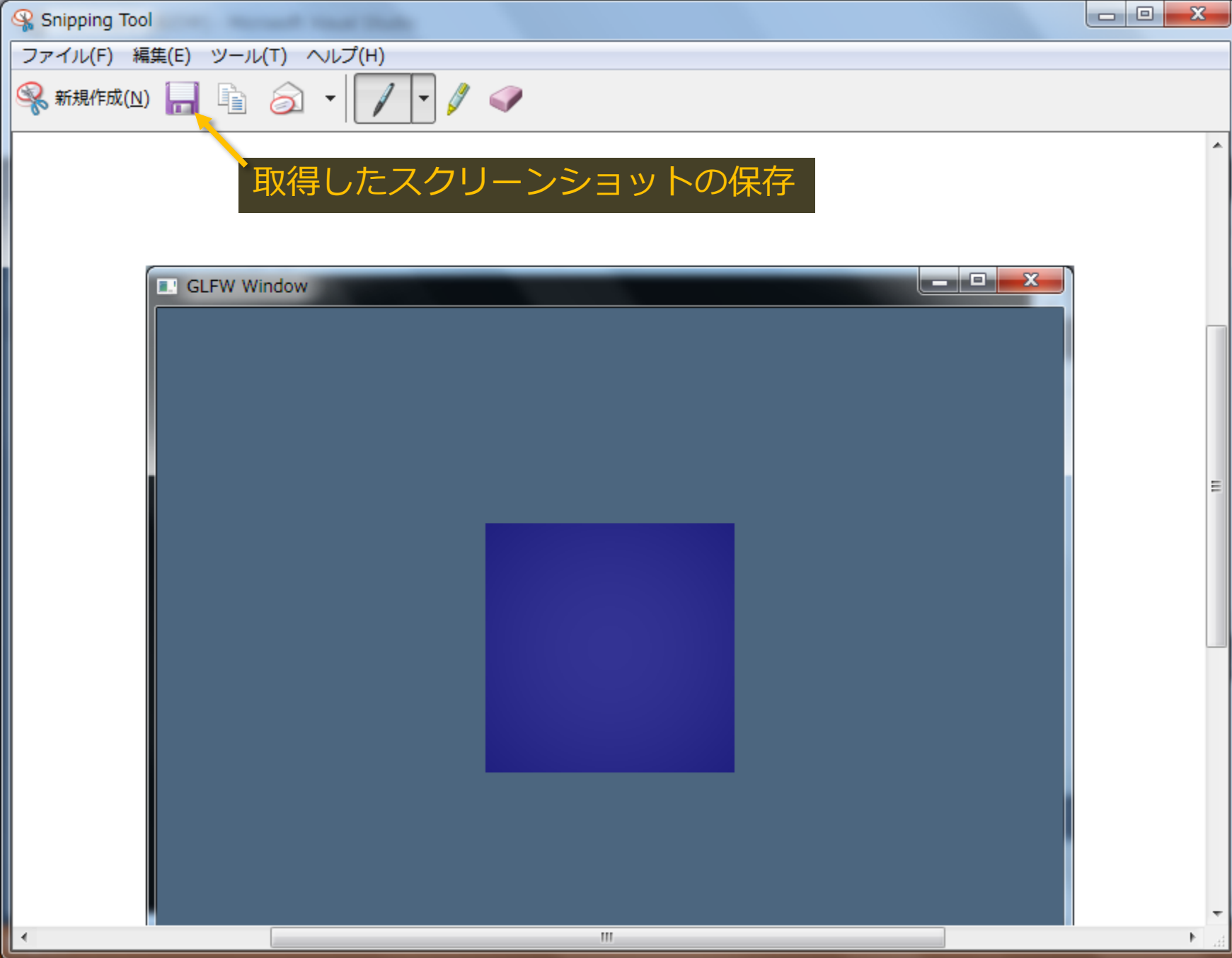
スナップショットの作成

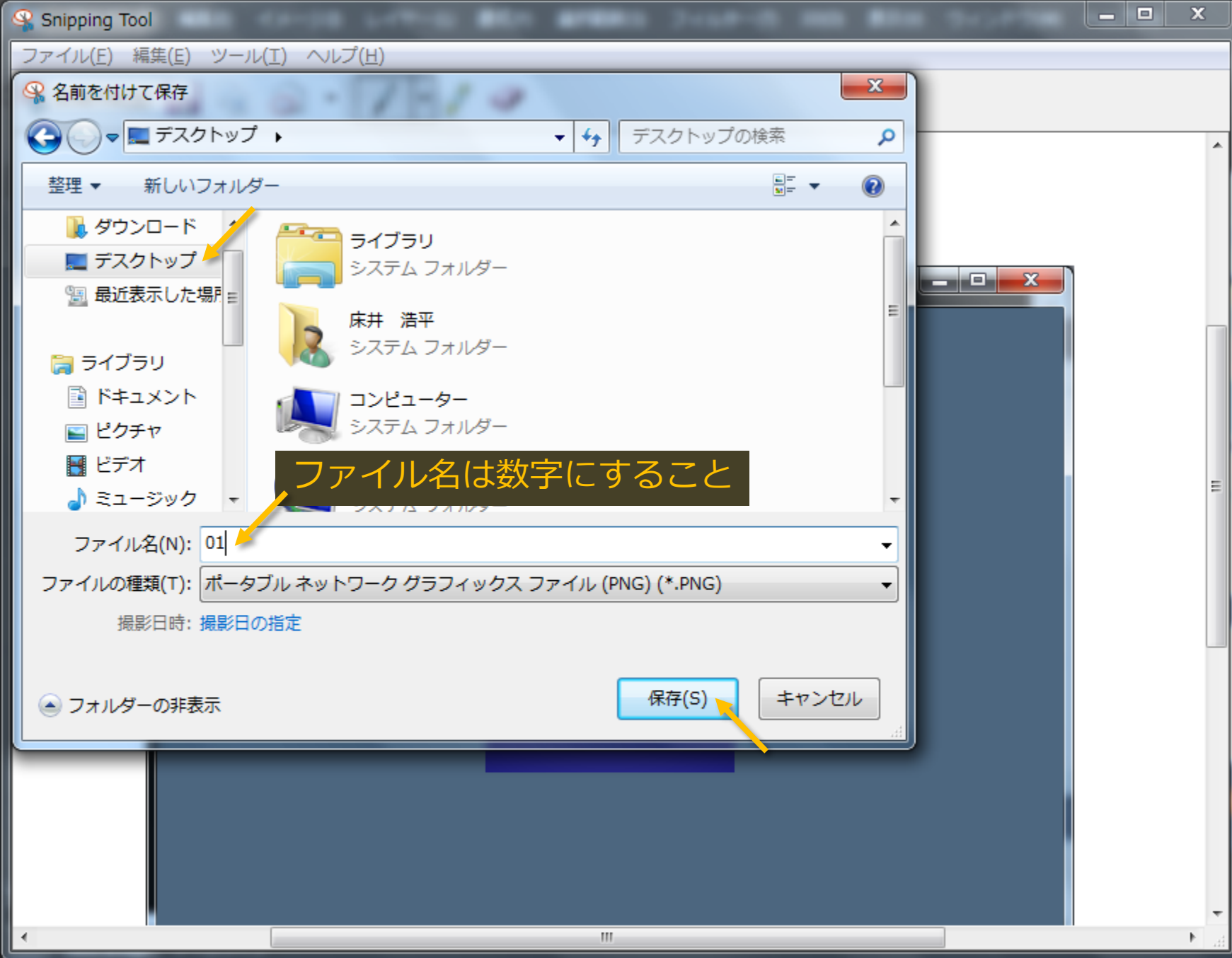
呼び出し履歴

型	名前	言語
---	----	----

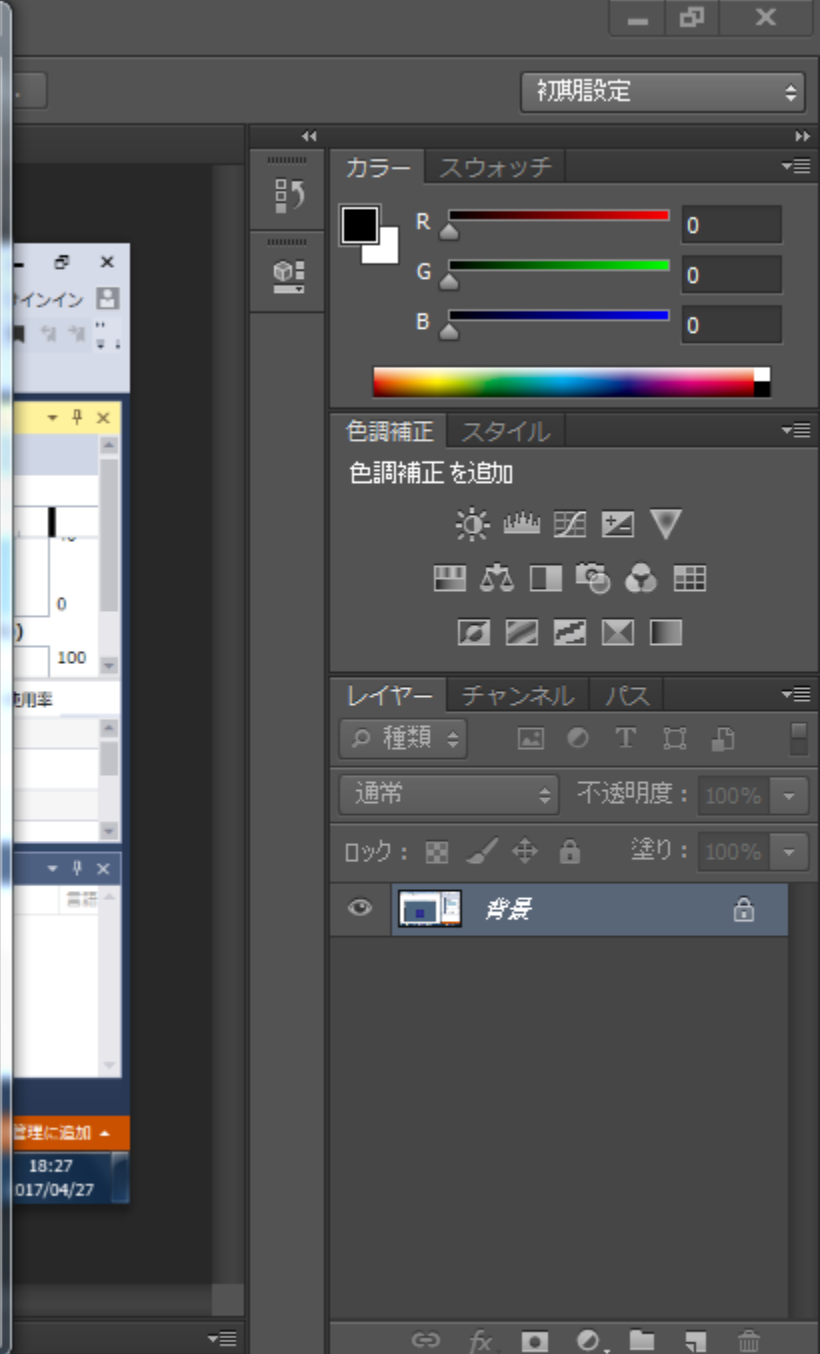
ソリューション エクスプローラー クラス ビュー 自動変数 ローカル スレッド モジュール ウォッチ 1 呼び出し履歴 ブレークポイント 例外設定 出力

準備完了 1行 1列 1文字 挿入 ソース管理に追加



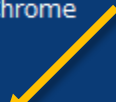


ファイル名は数字にすること



- コンピューター
- Style Builder 2016
- ごみ箱
- Unity 5.4.2f2 (64-bit)
- Epic Games Launcher
- マイドキュメント
- Google Earth
- 質問
- LayOut 2016
- seminar1
- Mozilla Firefox
- Google Chrome
- Mozilla Thunderbird
- 01.PNG
- SketchUp 2016

スクリーンショット





# 課題の提出

スクリーンショットのアップロード

## 課題の資料

この演習の Web ページを開く

- [グラフィックスと画像処理 課題資料 \(PDF\)](#)
  - 課題テンプレート - [seminar1.zip](#)
  - 補助資料 - [Visual Studio 2013 の新規プロジェクト作成 \(PDF\)](#)
  - 補助資料 - [GLFW による OpenGL 入門 \(PDF\)](#)
  - 補助資料 - [ゲームグラフィックス特論宿題補助プログラム](#)
  - 参考資料 - [OpenGL - The Industry Standard for High Performance Graphics \(外部サイト\)](#)
  - 参考資料 - [OpenCV Library \(外部サイト\)](#)
  - 参考資料 - [OpenCV.jp \(外部サイト\)](#)
- 【以前の課題】[ばね質点モデル 課題資料 \(PDF\)](#)
  - 課題テンプレート (糸) - [spring.zip](#)
  - 課題テンプレート (布) - [cloth.zip](#)
- 【以前の課題】[GLUT による「手抜き」OpenGL 入門](#)

## 課題の提出

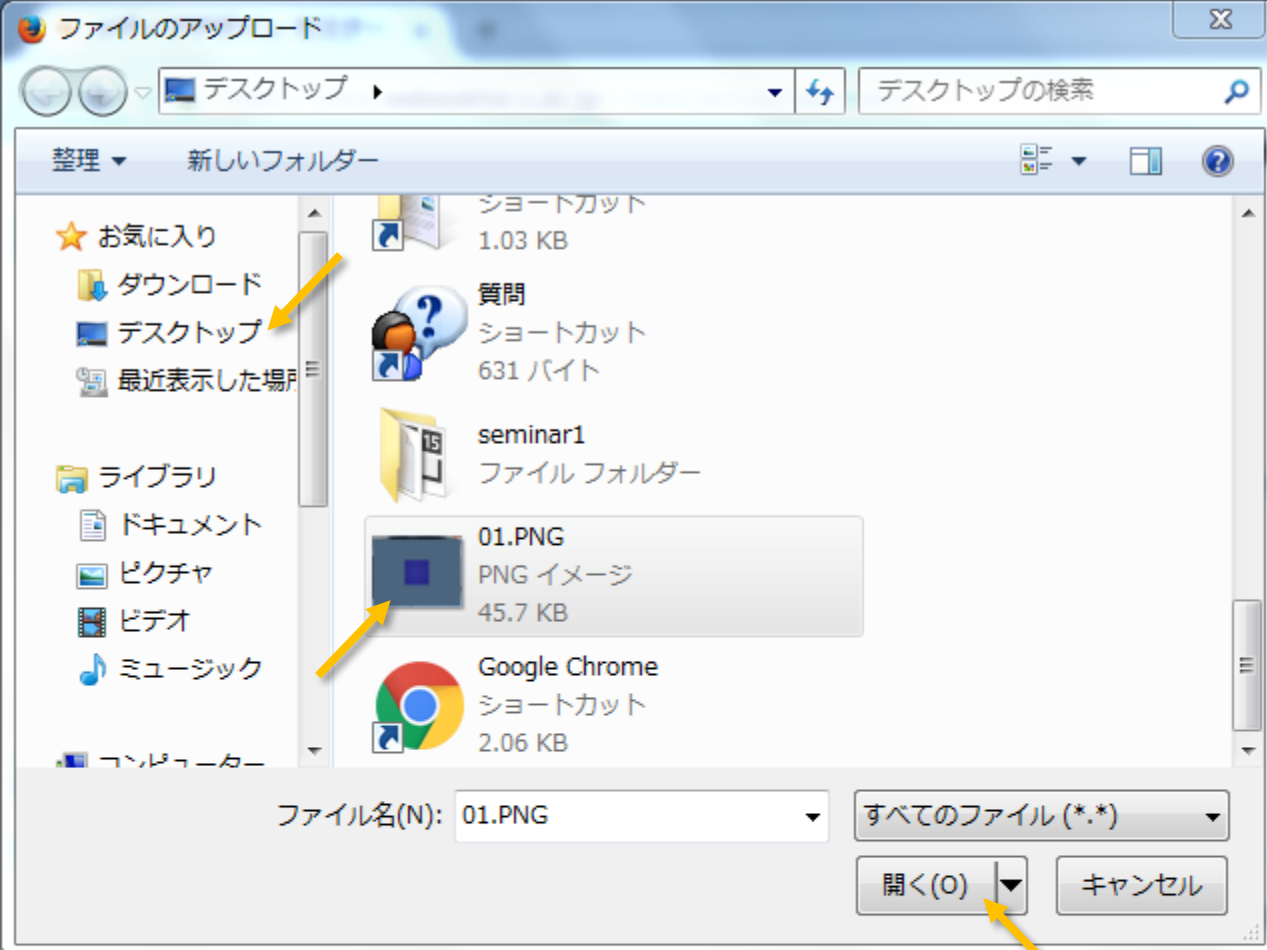
完成した課題のファイルをアップロードする際、「参照」をクリックし、「数字.拡張子」にしてください。

1. 「参照」(あるいは「選択」)をクリックして、アップロードするファイルを選択してください (上限 50MB)。

ファイル名:  ファイルが選択されていません。

2. ファイルの選択が完了したら、「送信」をクリックしてください。

- ファイルを指定せずに送信すれば、既にアップロードされているものが表示されます。
- アップロードできるファイルの形式 (拡張子) は PNG, JPG, GIF, PDF, C, CPP です。
- 以前にアップロードした同じファイル名のファイルは上書きされます。
- 十分にテストしていないので、問題があれば教員に連絡してください。



部サイト)

限 50MB).

ファイル名:  ファイルが選択されていません。

2. ファイルの選択が完了したら、「送信」をクリックしてください。

- ファイルを指定せずに送信すれば、既にアップロードされているものが表示されます。
- アップロードできるファイルの形式(拡張子)は PNG, JPG, GIF, PDF, C, CPP です。
- 以前にアップロードした同じファイル名のファイルは上書きされます。
- 十分にテストしていないので、問題があれば教員に連絡してください。

## 課題の資料

- [グラフィックスと画像処理 課題資料 \(PDF\)](#)
  - 課題テンプレート - [seminar1.zip](#)
  - 補助資料 - [Visual Studio 2013 の新規プロジェクト作成 \(PDF\)](#)
  - 補助資料 - [GLFW による OpenGL 入門 \(PDF\)](#)
  - 補助資料 - [ゲームグラフィックス特論宿題補助プログラム](#)
  - 参考資料 - [OpenGL - The Industry Standard for High Performance Graphics \(外部サイト\)](#)
  - 参考資料 - [OpenCV Library \(外部サイト\)](#)
  - 参考資料 - [OpenCV.jp \(外部サイト\)](#)
- 【以前の課題】[ばね質点モデル 課題資料 \(PDF\)](#)
  - 課題テンプレート (糸) - [spring.zip](#)
  - 課題テンプレート (布) - [cloth.zip](#)
- 【以前の課題】[GLUT による「手抜き」OpenGL 入門](#)

## 課題の提出

完成した課題のファイルをアップロードしてください。ファイル名は“数字+拡張子”にしてください。

1. 「参照」(あるいは「選択」)をクリックしてください。(上限 50MB).

ファイル名:  01.PNG

2. ファイルの選択が完了したら、「送信」をクリックしてください。

- ファイルを指定せずに送信すれば、既にアップロードされているものが表示されます。
- アップロードできるファイルの形式(拡張子)は PNG, JPG, GIF, PDF, C, CPP です。
- 以前にアップロードした同じファイル名のファイルは上書きされます。
- 十分にテストしていないので、問題があれば教員に連絡してください。

ファイル名が設定されている

送信

# アップロード結果

01.PNG のアップロードに成功しました。以前にアップロードしたものは上書きされました。  
下記をクリックして内容を確認し、問題があれば再度アップロードしてください。

- [01.PNG](#)

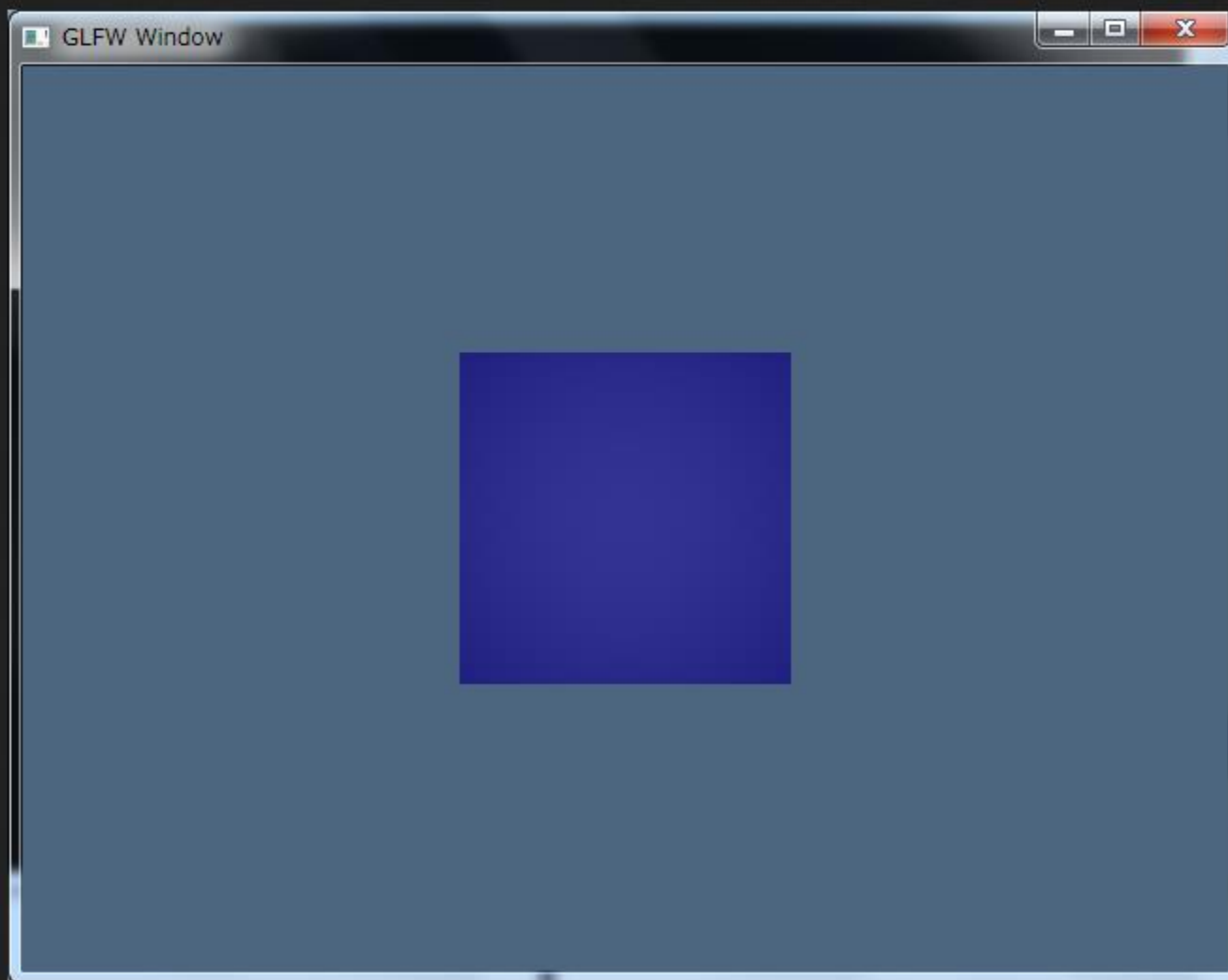
## 提出済の課題

- [01.PNG](#)

連絡先: [tokoi@sys.wakayama-u.ac.jp](mailto:tokoi@sys.wakayama-u.ac.jp)

戻る

アップロード成功



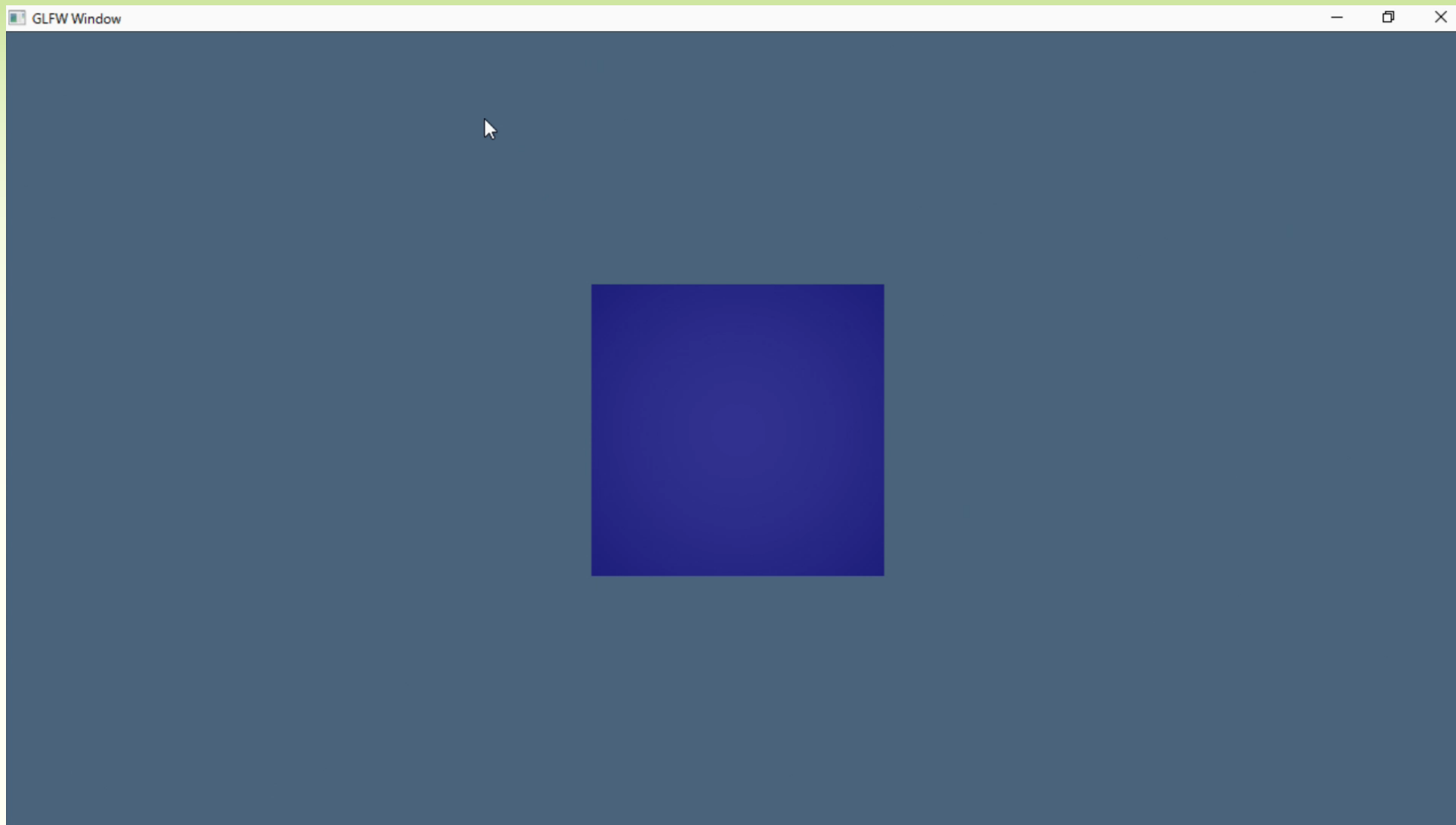
アップロードしたファイルの確認



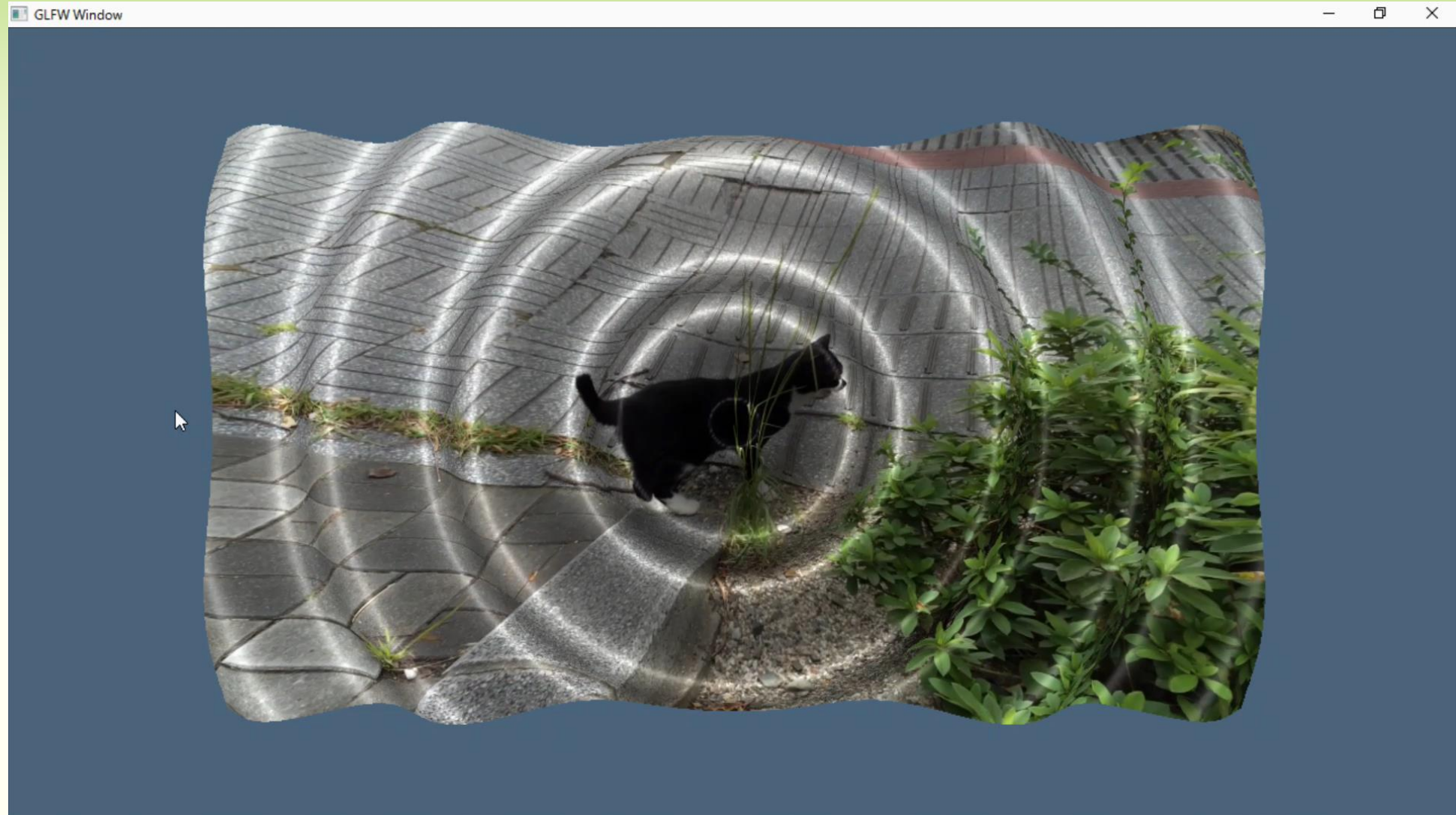
# 課題

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()  
{
```

ここにプログラムを書きます

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

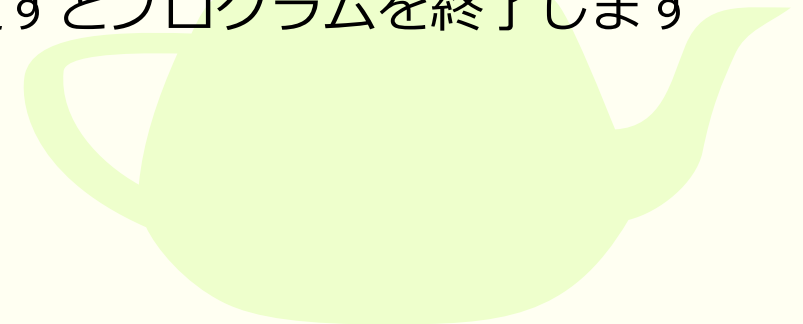
```
//  
// 更新（毎回実行されます）  
//
```

```
bool update()  
{
```

ここにプログラムを書きます

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

- 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];

:
:

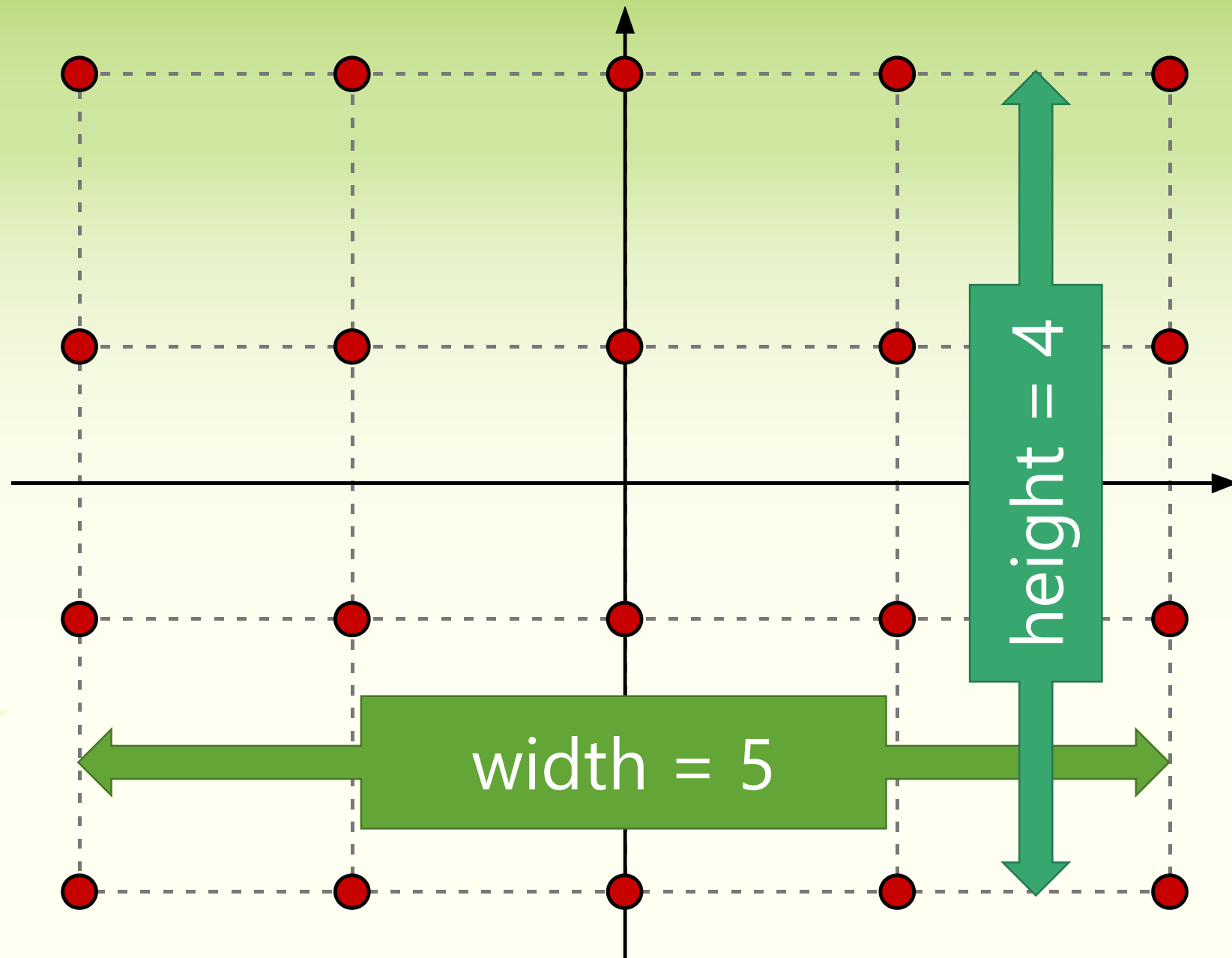
// 点のデータを保持するメモリを確保してそこにデータを転送する
createPoint(width, height, point);
```

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];

:
:

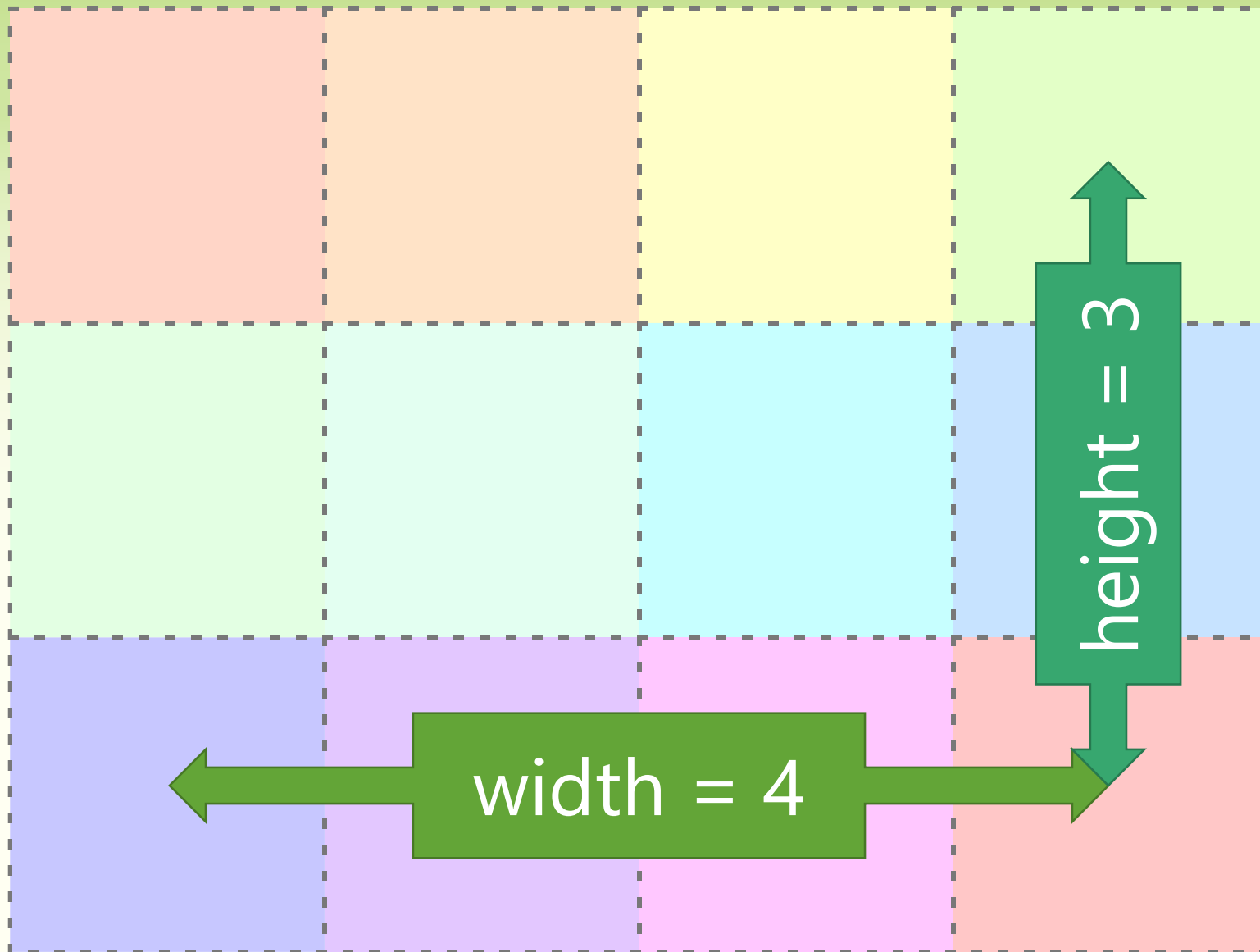
// 色のデータを保持するメモリを確保してそこにデータを転送する
createColor(width, height, color);
```

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

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

$(-1, 1, 0)$     $(-0.5, 1, 0)$     $(0, 1, 0)$     $(0.5, 1, 0)$     $(1, 1, 0)$



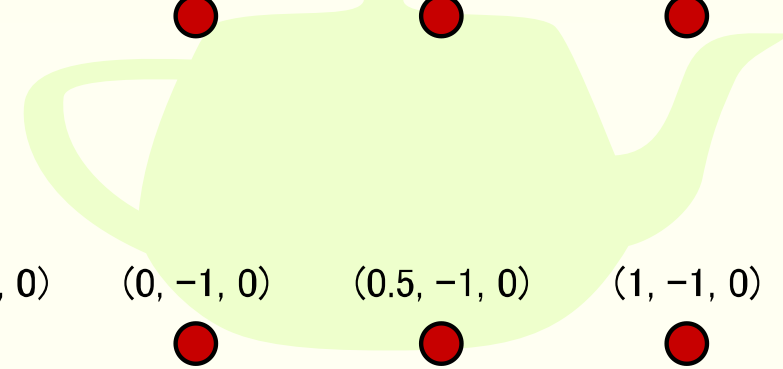
$(-1, 0.33, 0)$     $(-0.5, 0.33, 0)$     $(0, 0.33, 0)$     $(0.5, 0.33, 0)$     $(1, 0.33, 0)$



$(-1, -0.33, 0)$     $(-0.5, -0.33, 0)$     $(0, -0.33, 0)$     $(0.5, -0.33, 0)$     $(1, -0.33, 0)$

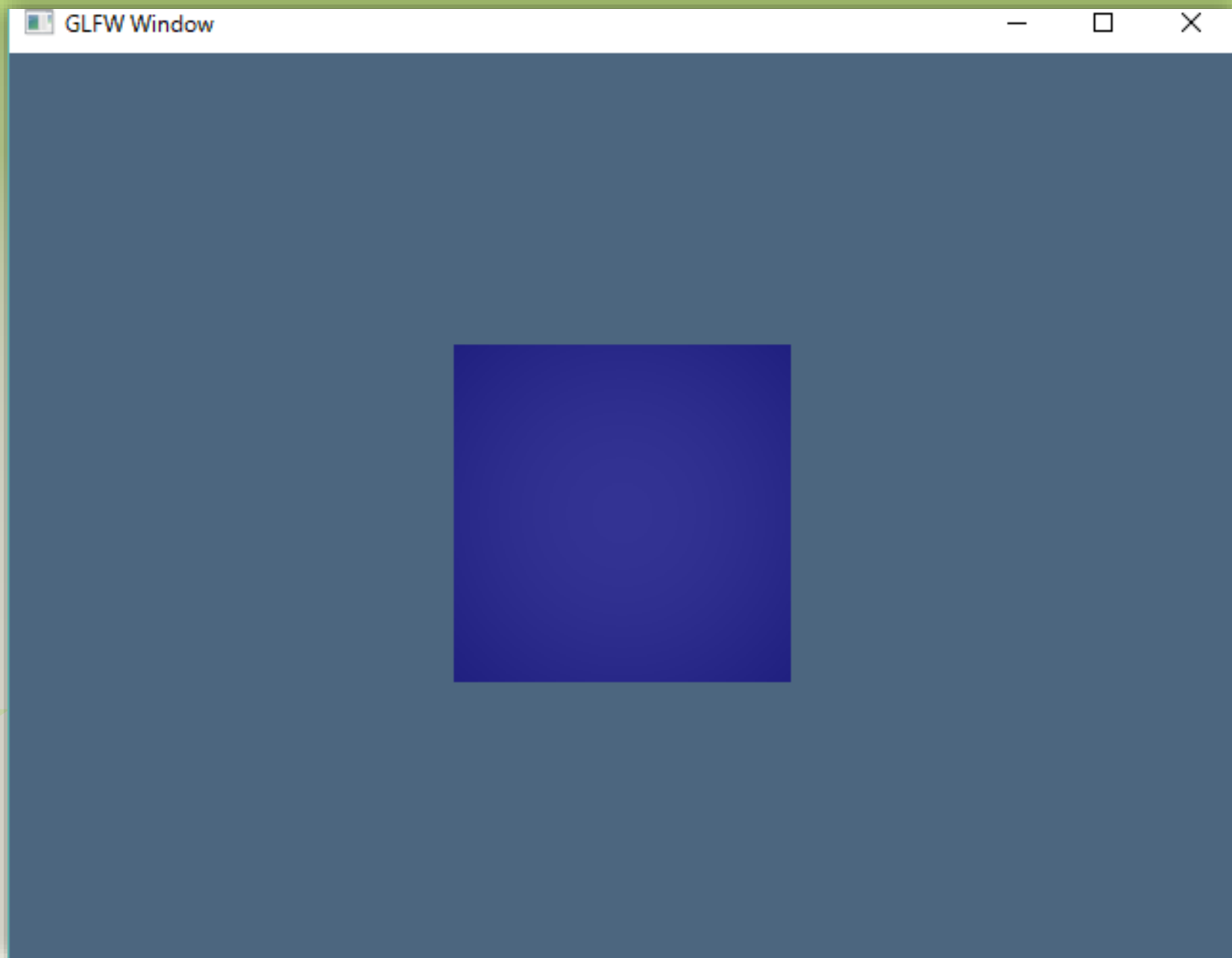


$(-1, -1, 0)$     $(-0.5, -1, 0)$     $(0, -1, 0)$     $(0.5, -1, 0)$     $(1, -1, 0)$



## 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));

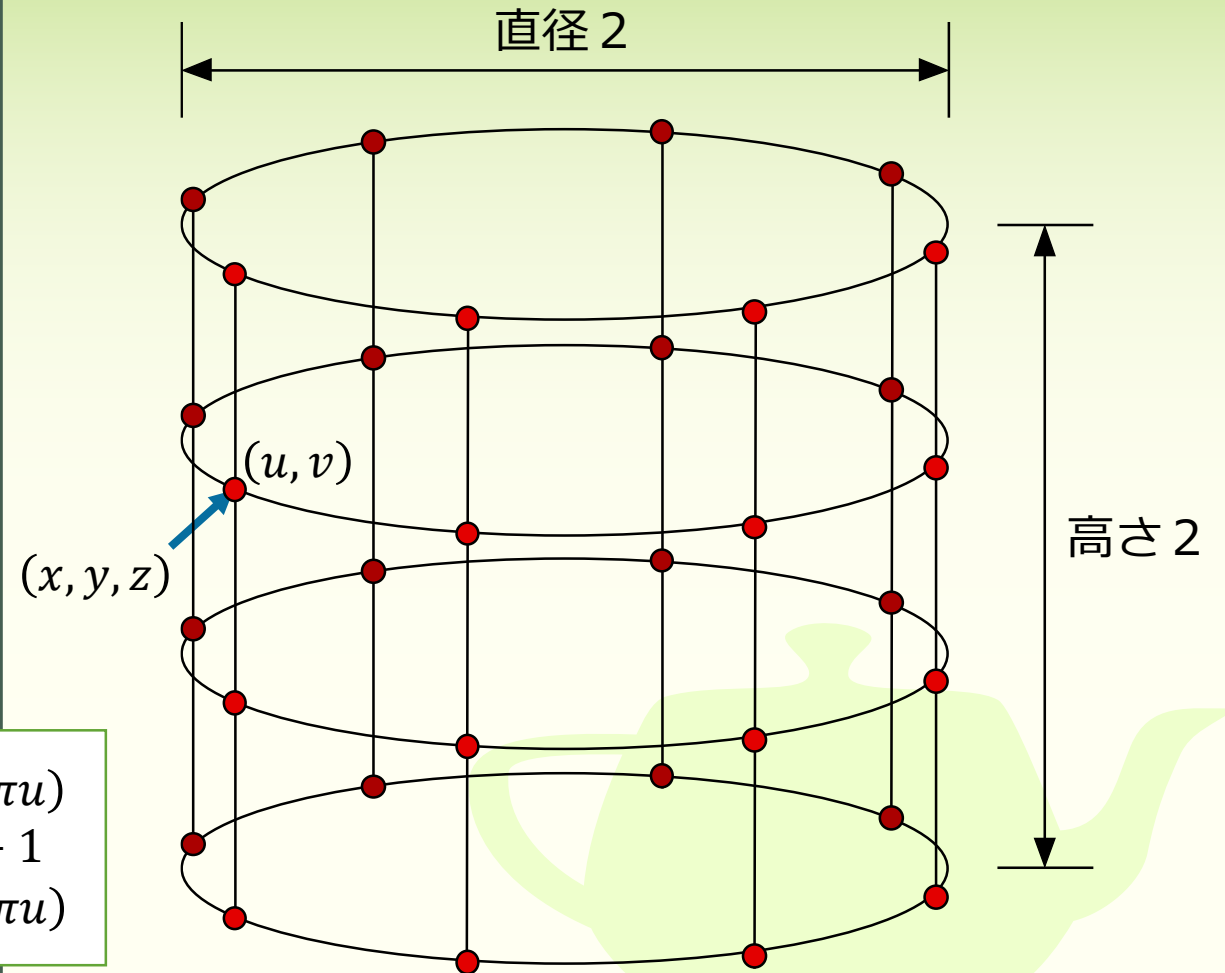
        // 経度
        const float longtude(2.0f * pi * u);

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

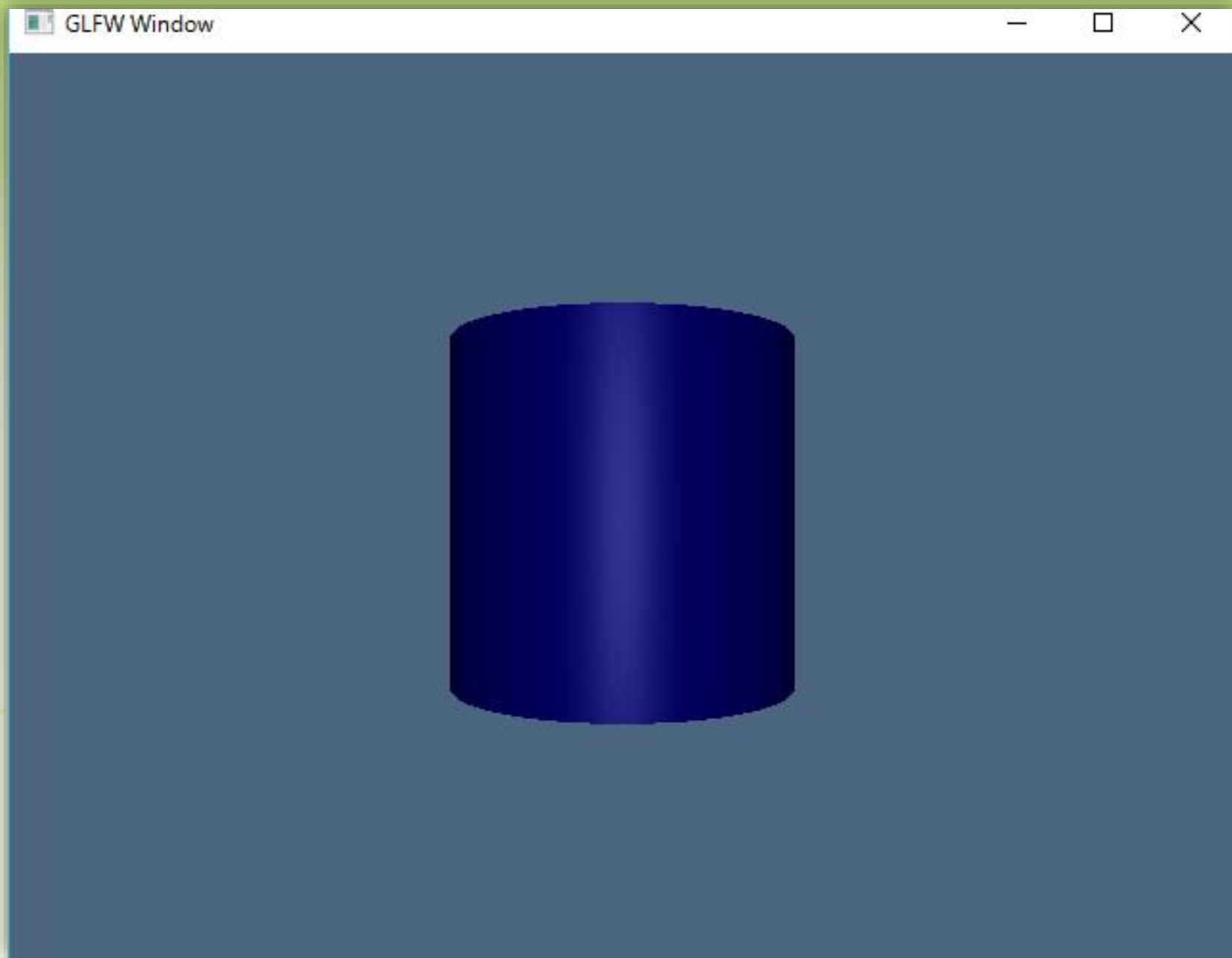
        // 頂点の位置
        point[j][i][0] = sin(longtude);
        point[j][i][1] = v * 2.0f - 1.0f;
        point[j][i][2] = cos(longtude);

        (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 longitude(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)

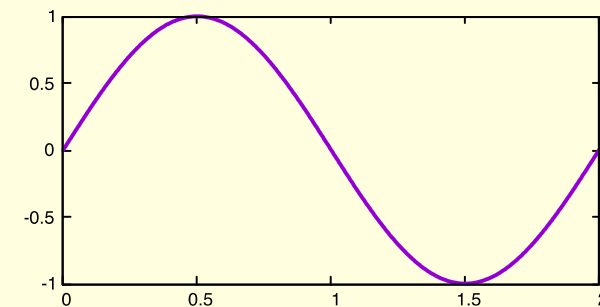
        // 頂点の色
        color[j][i][0] = 100;
        color[j][i][1] = unsigned char(255 * latitude_color);
        color[j][i][2] = 0;
    }
}

```

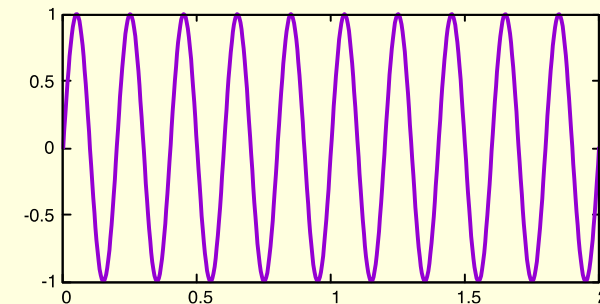
色のデータの最大値

unsigned char 型には 0~255 の整数値を格納できる

$\sin(\pi v)$



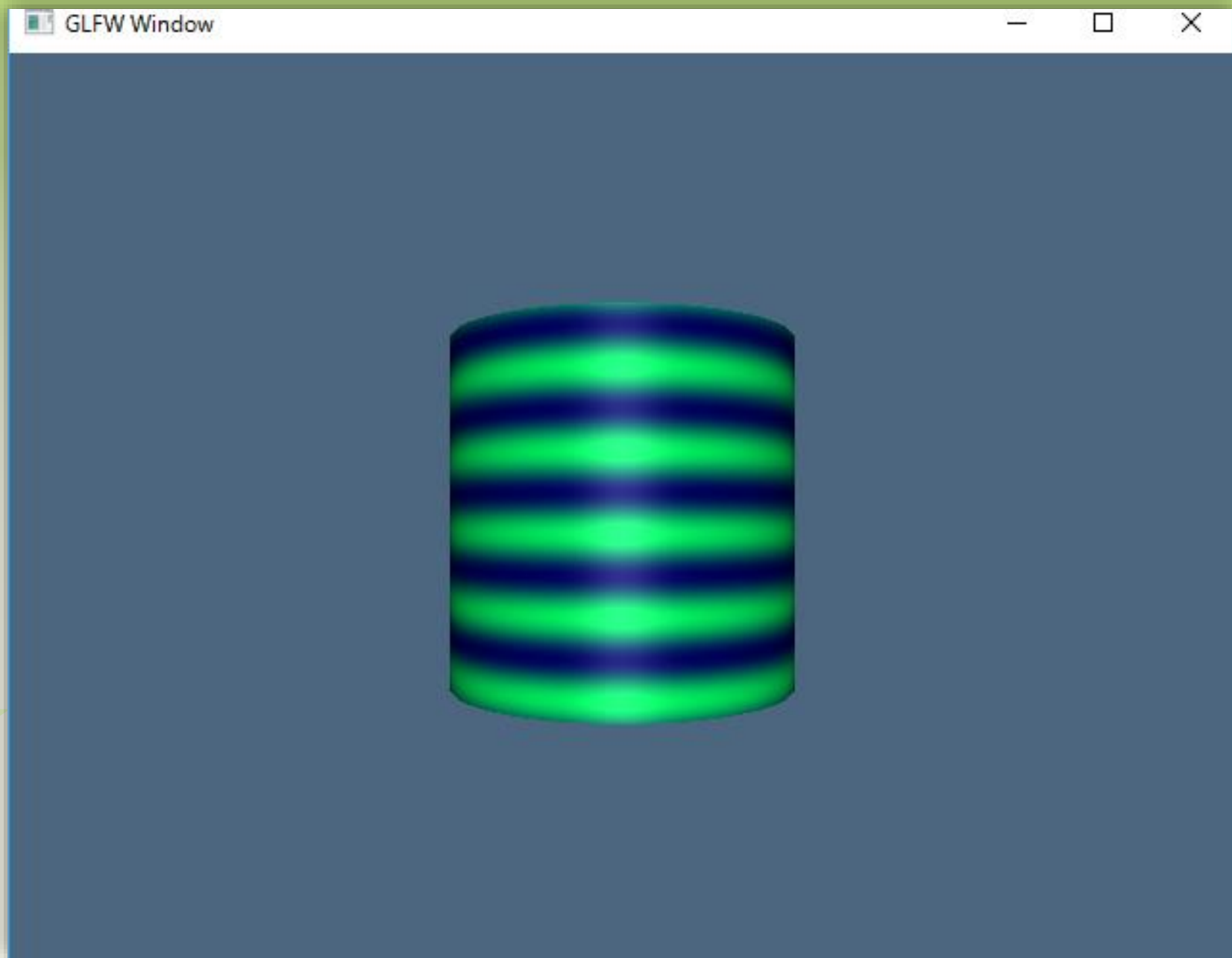
$\sin(10\pi v)$



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

$$0 \leq \text{latitude\_color} \leq 1$$

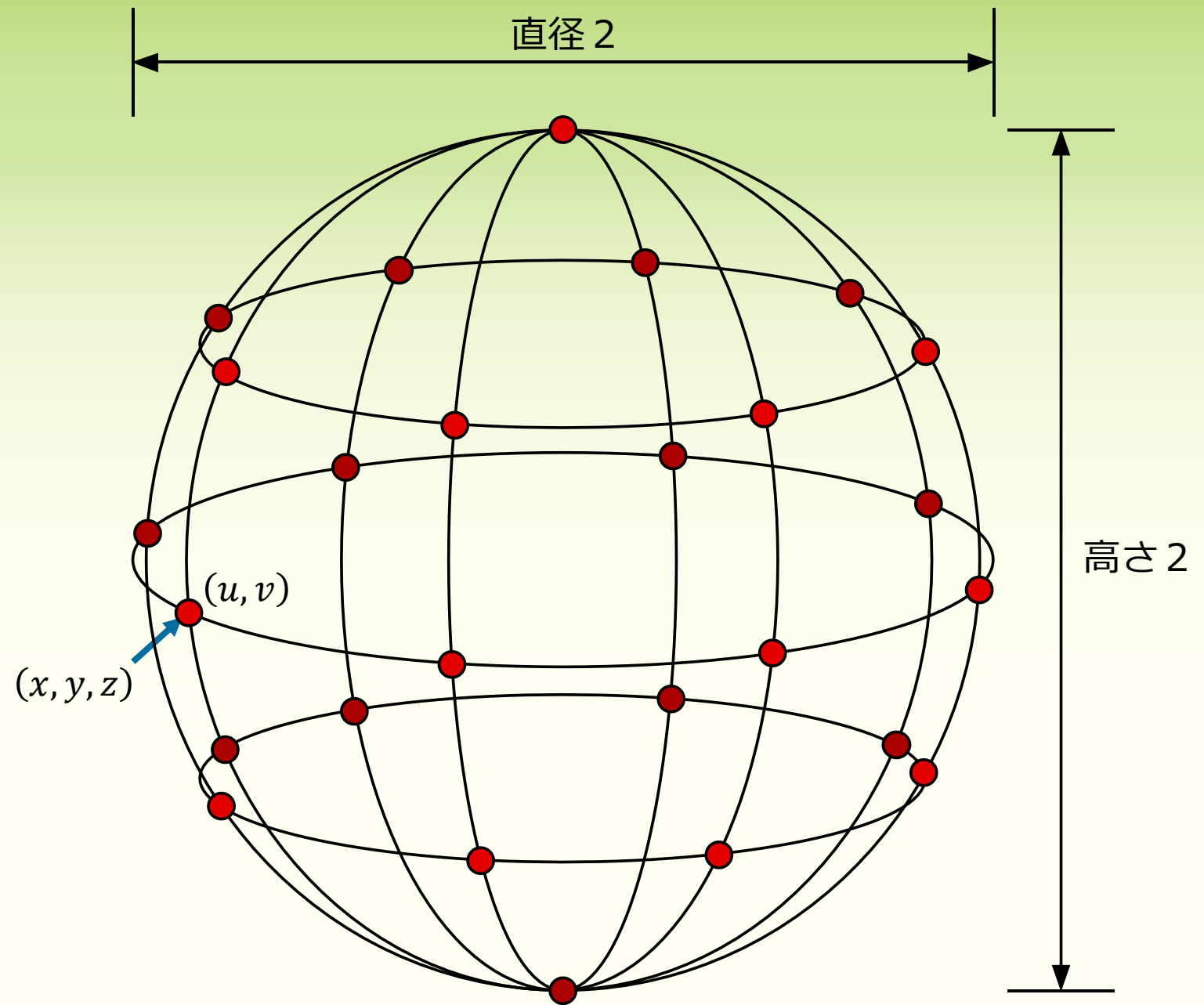
# 実行結果



# 課題1

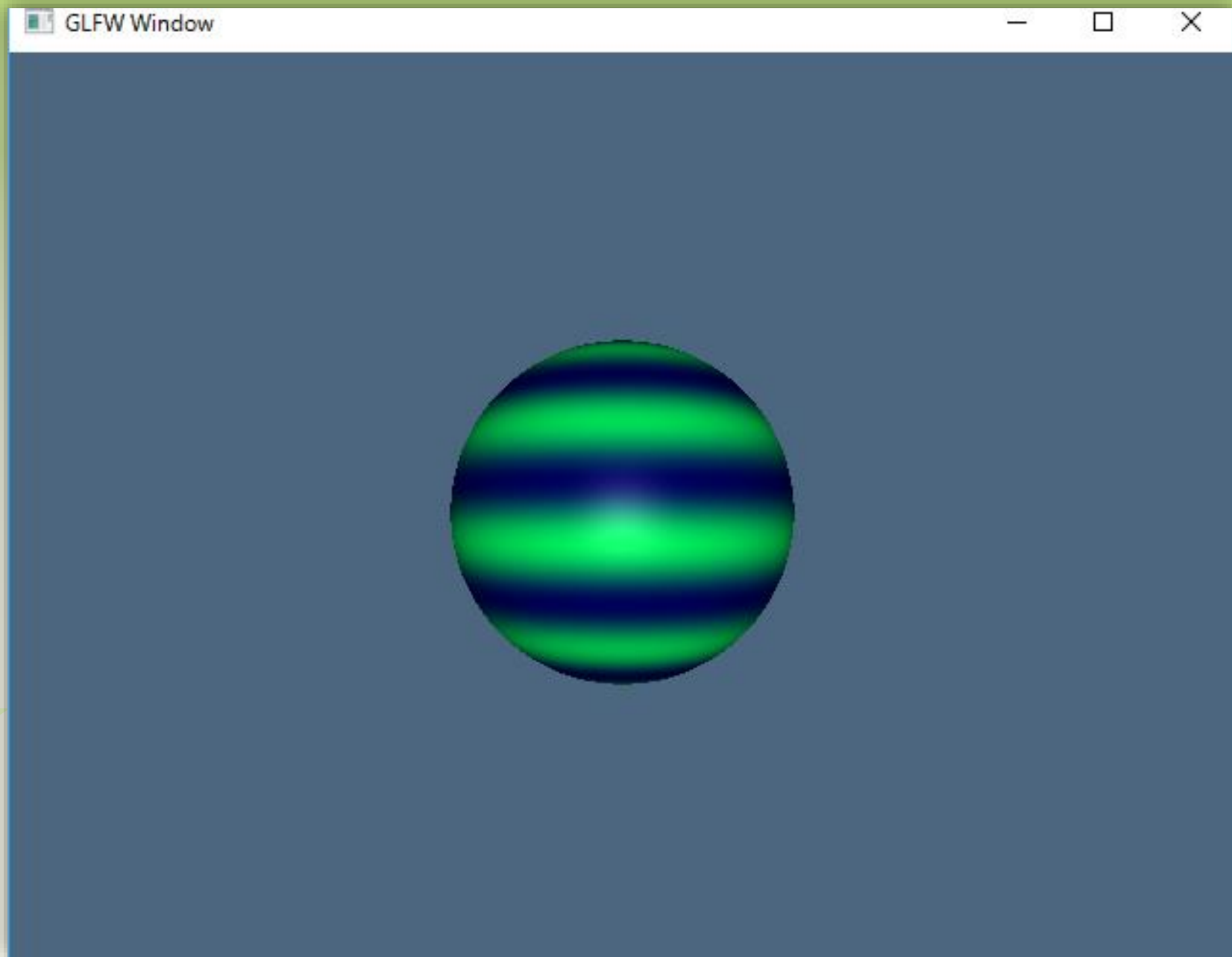
- 球を描いてください
  - 円柱をベースにして結構です
  - その場合北極と南極は複数の点が重なっていても構いません

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



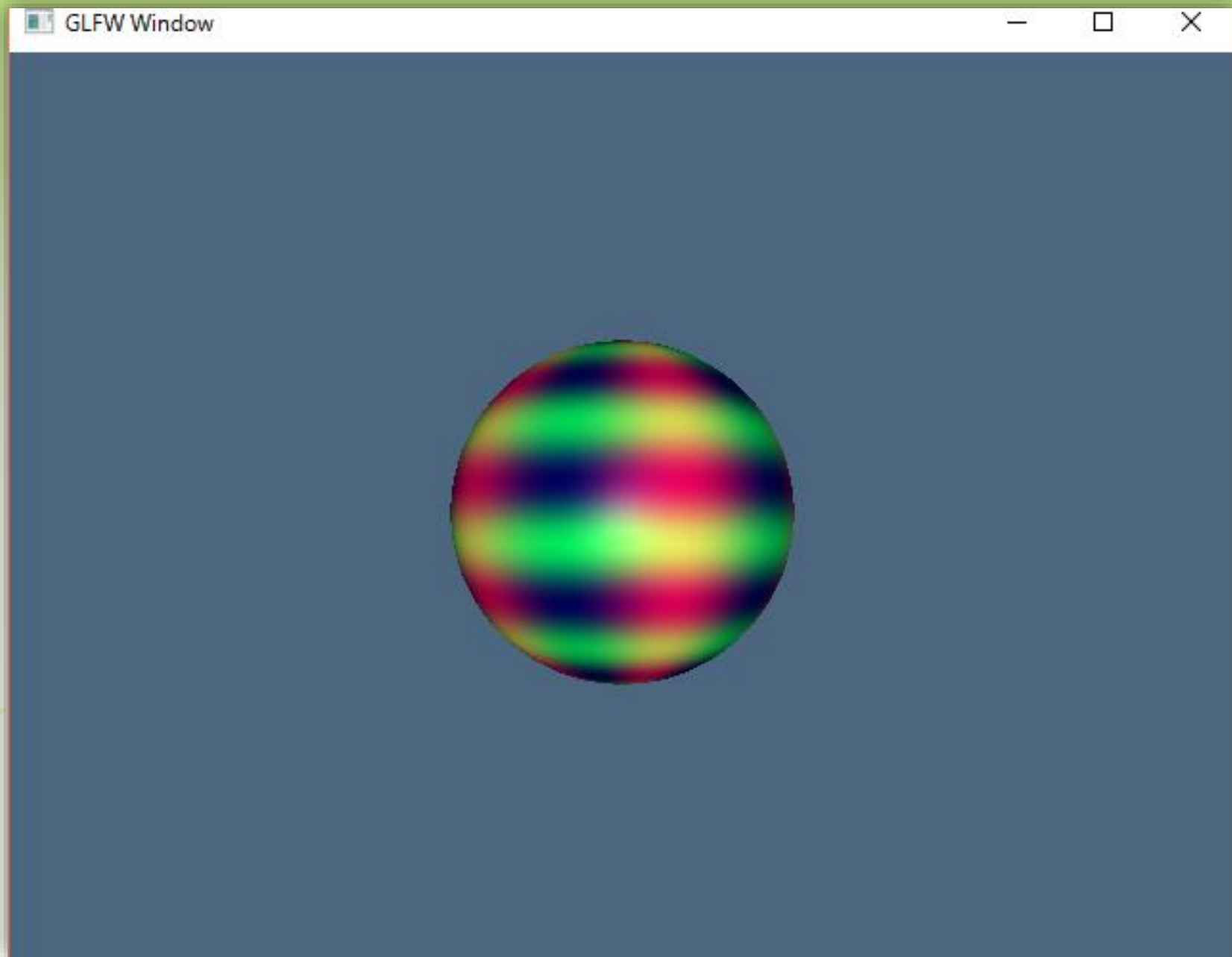
## 課題1 実行結果

- 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乘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);
```

← 追加

```
//
```

```
// 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 ( $0 \leq u \leq 1$ )
```

```
            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);
```


→ 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);
```

 update() にコピー

```
// 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);
```

 update() に移動

```
// 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);
}
}
```

# 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;
}
```

point は  
省略

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

```
    // 時間によって位置を変化させる
    const float offset(sin(2.0f * pi * fmod(getTime(), cycle) / cycle));
```

← 追加

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

← setup() からコピー

# kadai1.cpp の変更点 (4/5)

```
{  
  // x 方向のパラメータ ( $0 \leq u \leq 1$ )  
  const float u(float(i) / float(width - 1));  
  
  // 経度  
  const float longitude(2.0f * pi * u);  
  
  // 経度に対応した色のスケール  
  const float longitude_color(sin(10.0f * pi * u) * 0.5f + 0.5f);  
  
  // y 方向のパラメータ ( $0 \leq v \leq 1$ )  
  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);
```

← setup() からコピー

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

← 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 * ( ... );
```

update() の中で  
point に代入

← setup() から移動して  
各要素を **scale** 倍

```
}
}
```

← setup() からコピー

// 点データを登録する

```
submitPoint(width, height, point);
```

point を転送する

← 追加

// プログラムを終了しない

```
return true;
```

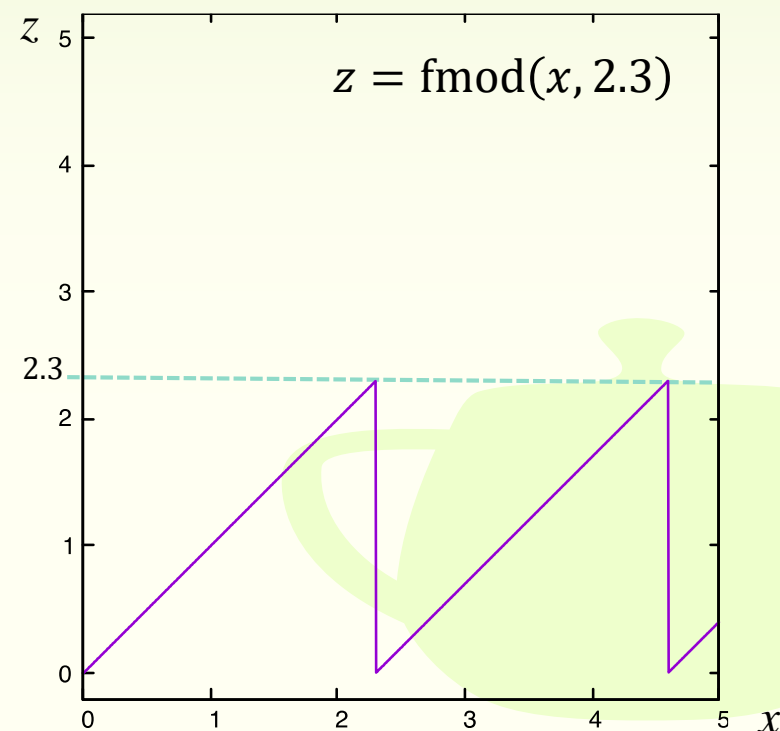
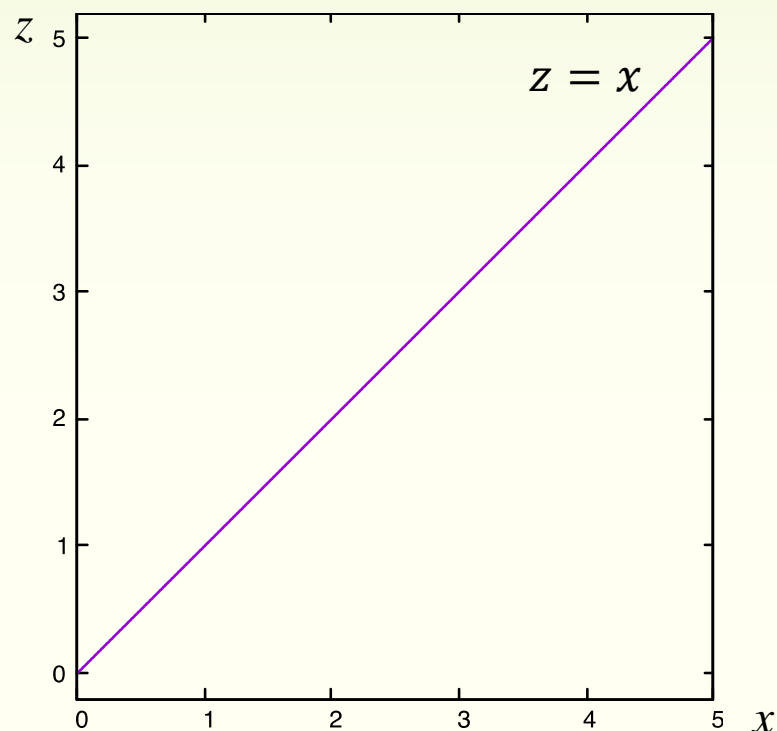
```
}
```

# fmod() 関数

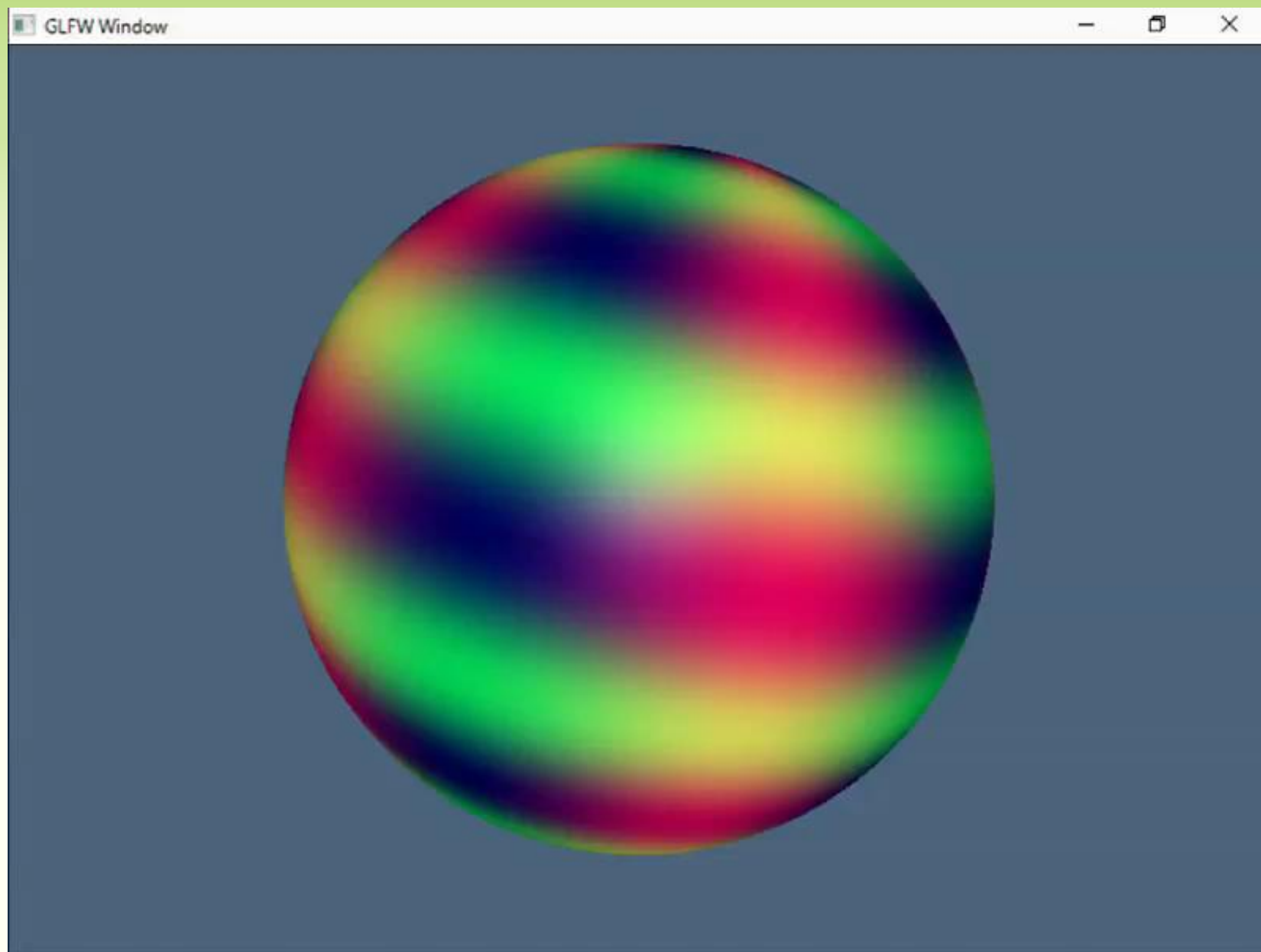
## ■ 実数の剰余を求める

□  $\text{fmod}(x, y) \rightarrow x - [x/y]y$

■  $\text{fmod}(15.6, 2.3) \rightarrow 15.6 - [15.6/2.3] \times 2.3 = 15.6 - 6 \times 2.3 = 1.8$

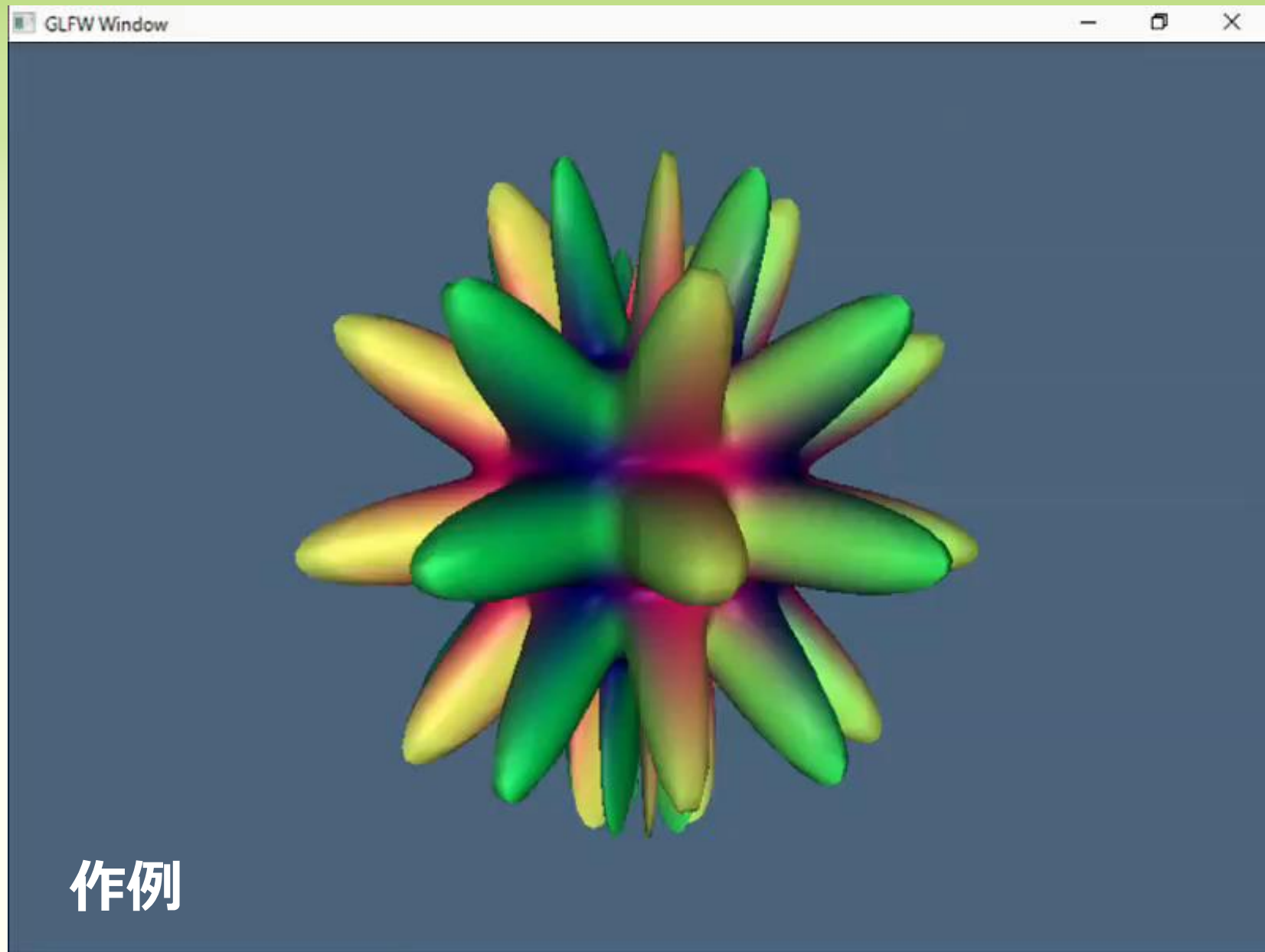


# 実行結果



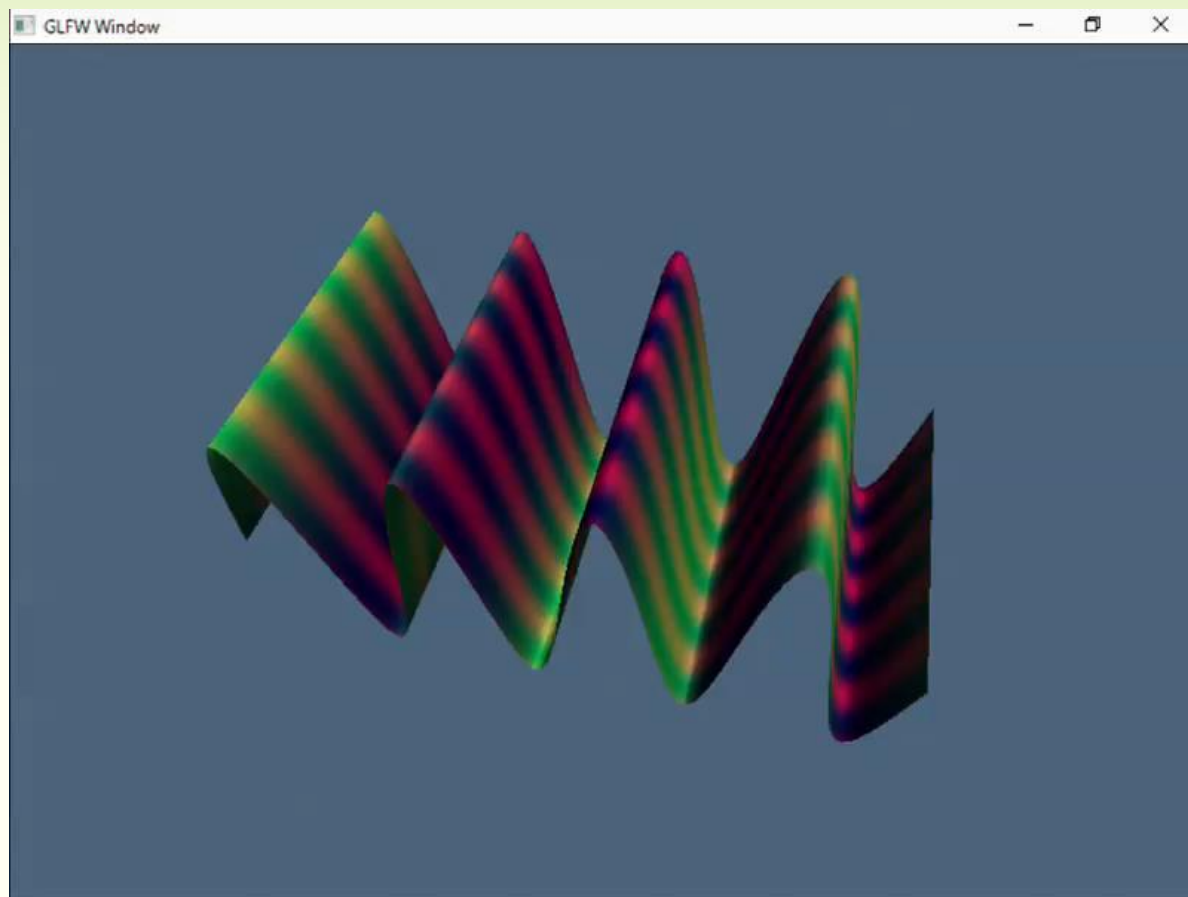
## 課題3

- 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

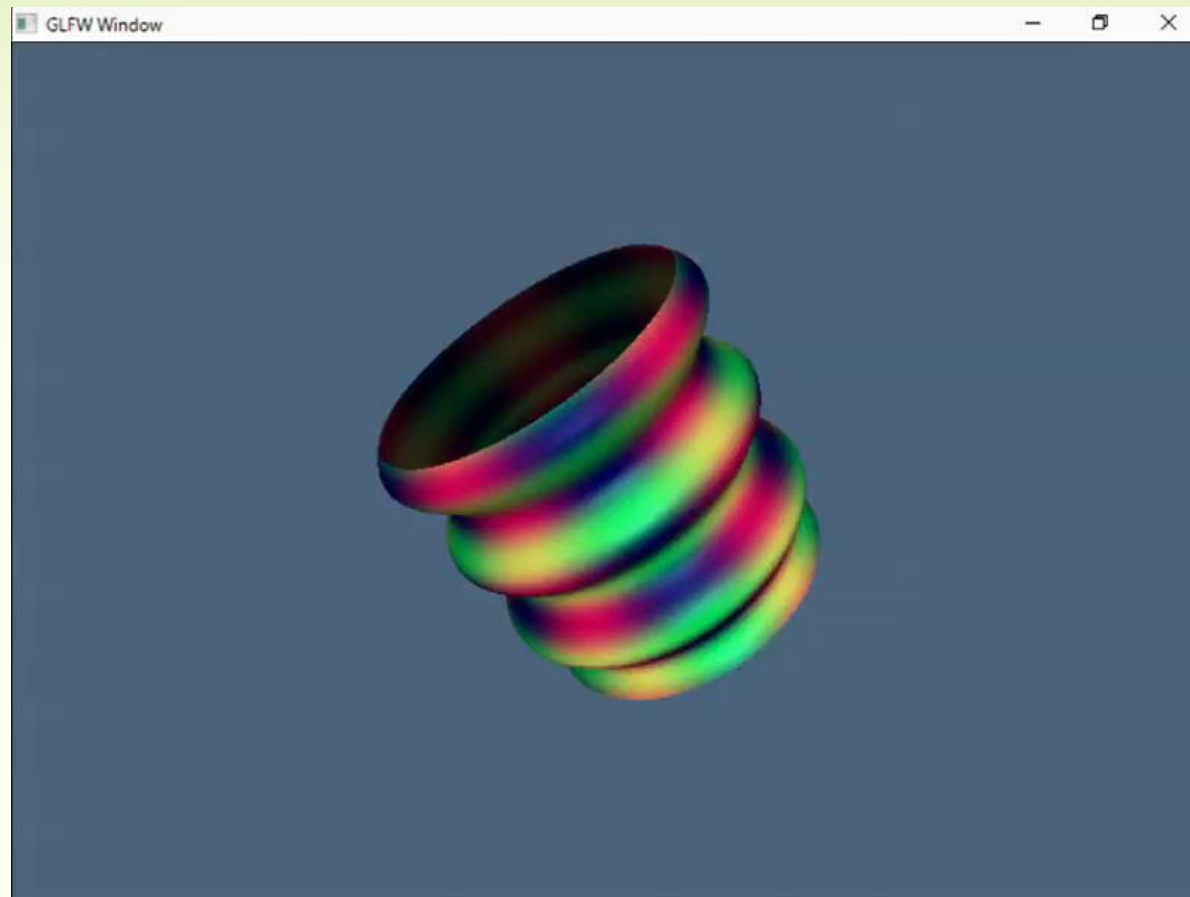


# 課題3のそのほかの作例

## 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

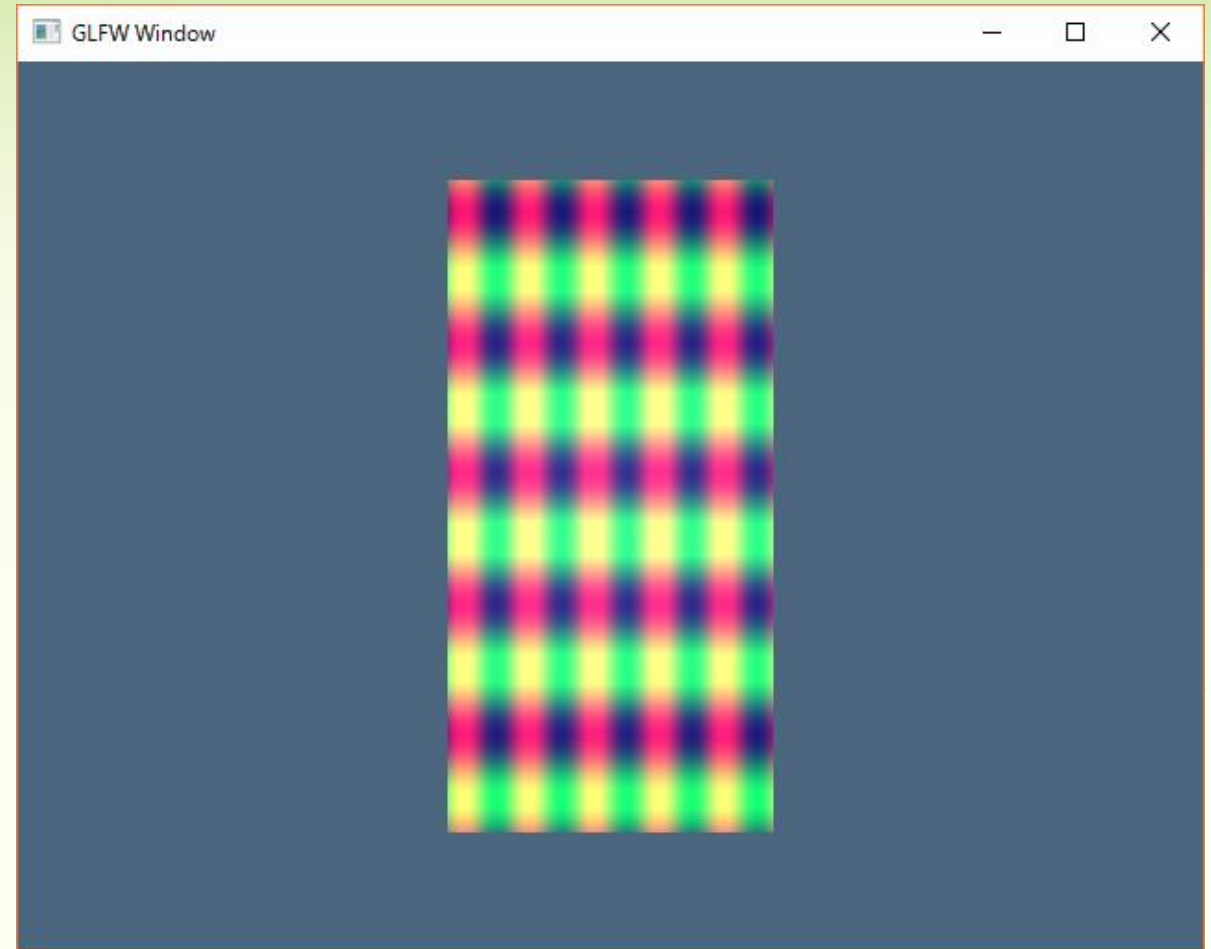
```
//
// 更新 (毎回実行されます)
//
bool update()
{
    for (int j = 0; j < height; ++j)
    {
        for (int i = 0; i < width; ++i)
        {
            // x 方向のパラメータ ( $0 \leq u \leq 1$ )
            const float u(float(i) / float(width - 1));

            // y 方向のパラメータ ( $0 \leq v \leq 1$ )
            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

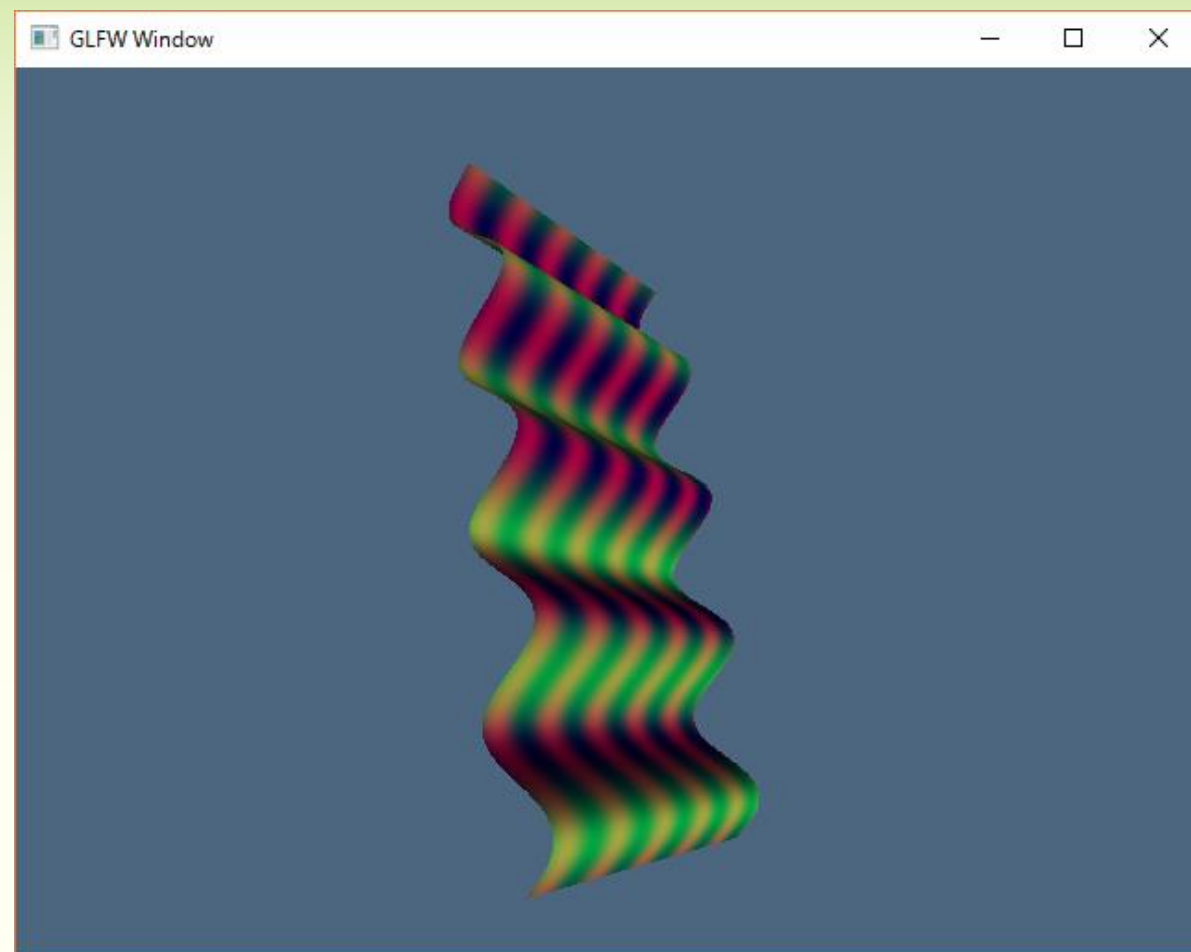
```
//
// 更新 (毎回実行されます)
//
bool update()
{
    for (int j = 0; j < height; ++j)
    {
        for (int i = 0; i < width; ++i)
        {
            // x 方向のパラメータ (0 ≤ u ≤ 1)
            const float u(float(i) / float(width - 1));

            // 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()
{
    // 時間によ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));

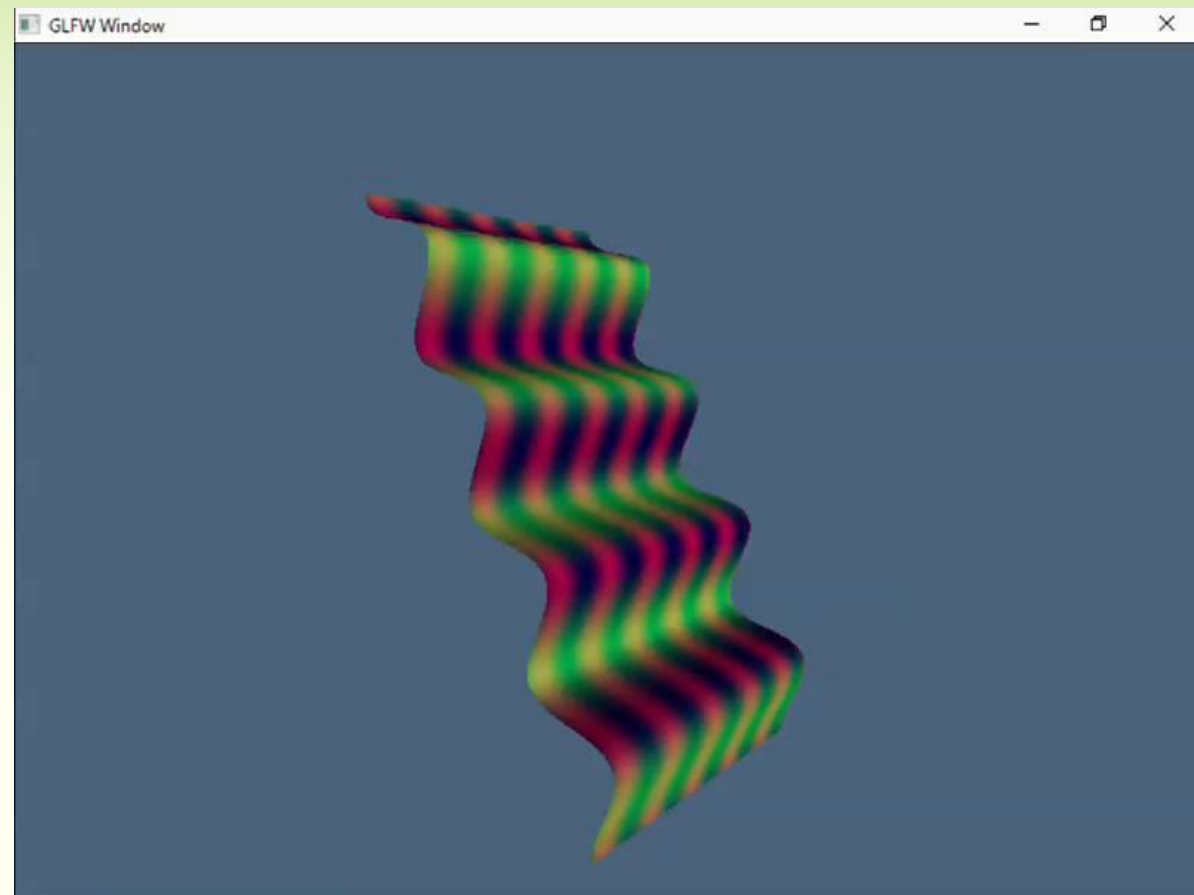
            // y 方向のパラメータ (0 ≤ v ≤ 1)
            const float v(float(j) / float(height - 1));

            // 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);

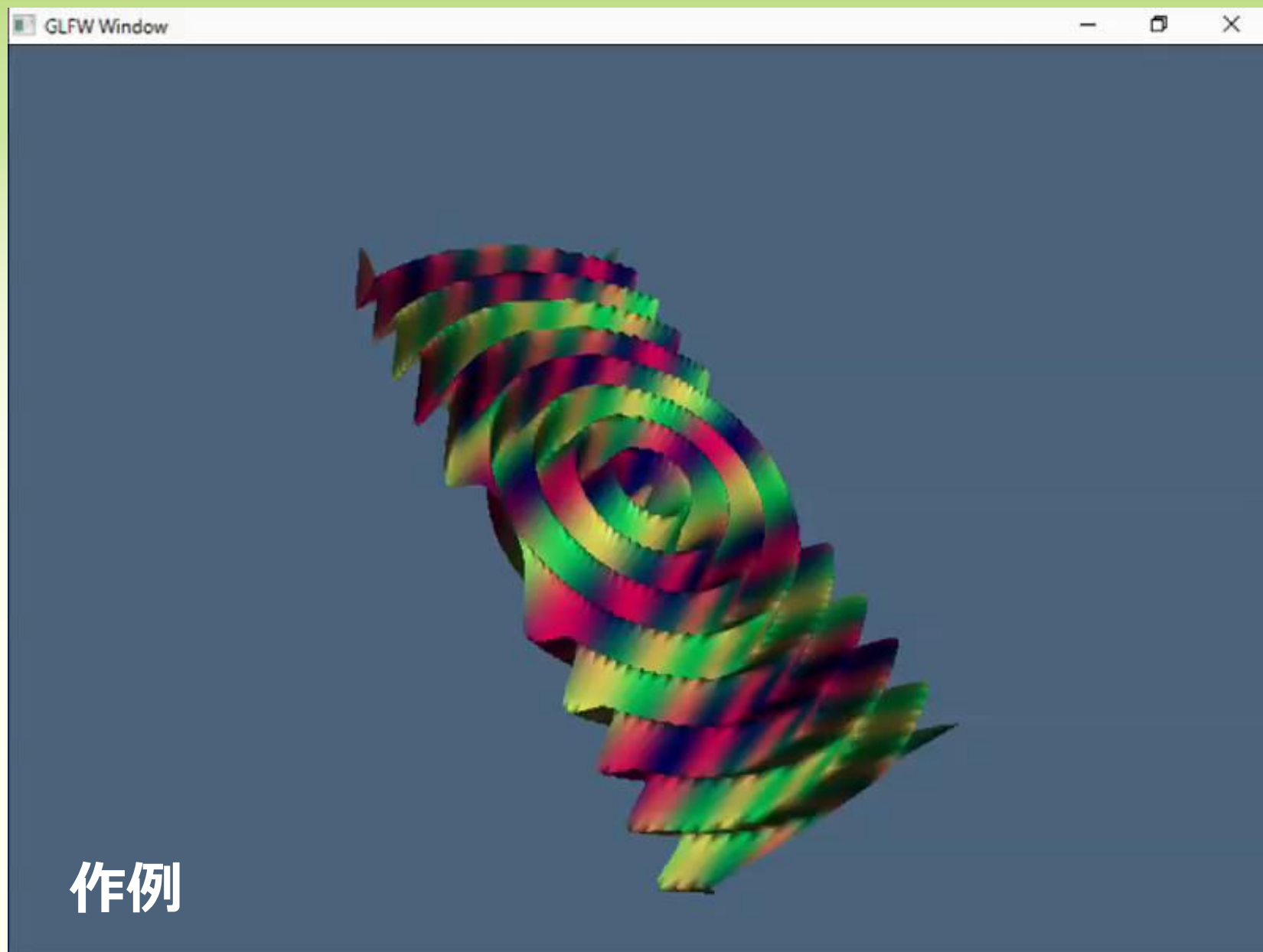
    // プログラムを終了しない
    return true;
}
```

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 (入っていないければ `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

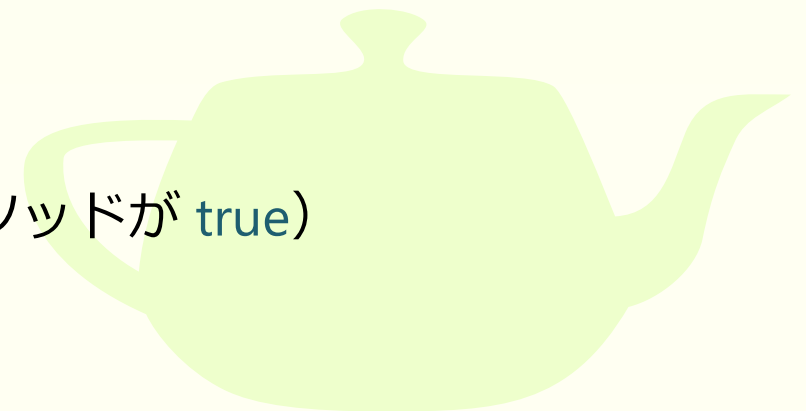
## ■ `cv::imread(const std::string &filename, int flags = 1)`

### □ 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

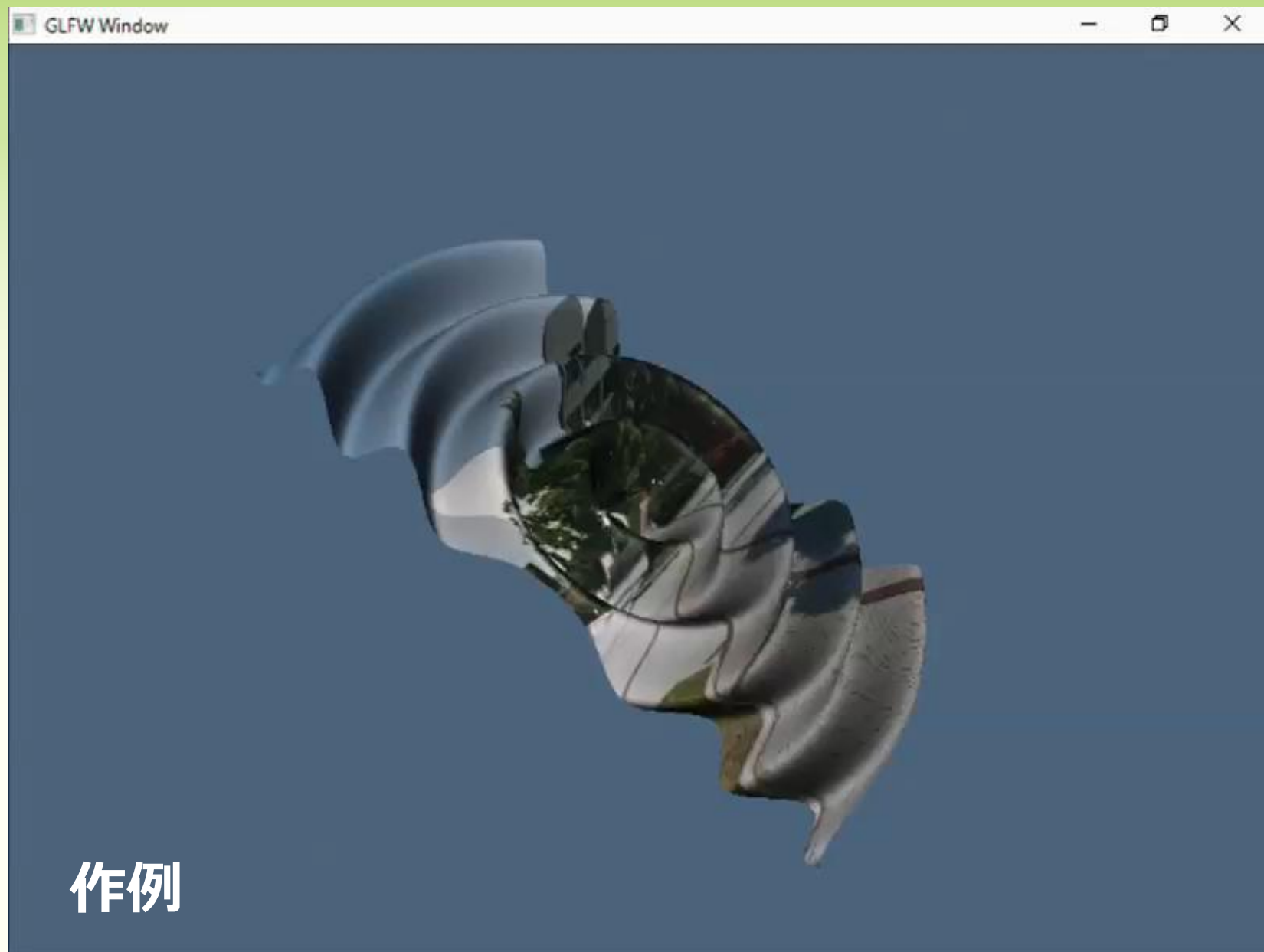
```
//  
// 設定 (最初に一度だけ実行されます)  
//  
bool setup()  
{  
    (省略)  
  
    // 画像の読み込み  
    const cv::Mat image(cv::imread("image.jpg"));  
  
    // 画像の読み込みに失敗した  
    if (image.empty()) return false;  
  
    // 色データを登録する  
    createColor(image.cols, image.rows, image.data);
```

- 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 propld, double value)`: プロパティ `propld` を設定  
する, `get(int propld)`: プロパティ `propld` 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つ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)
        }
    }

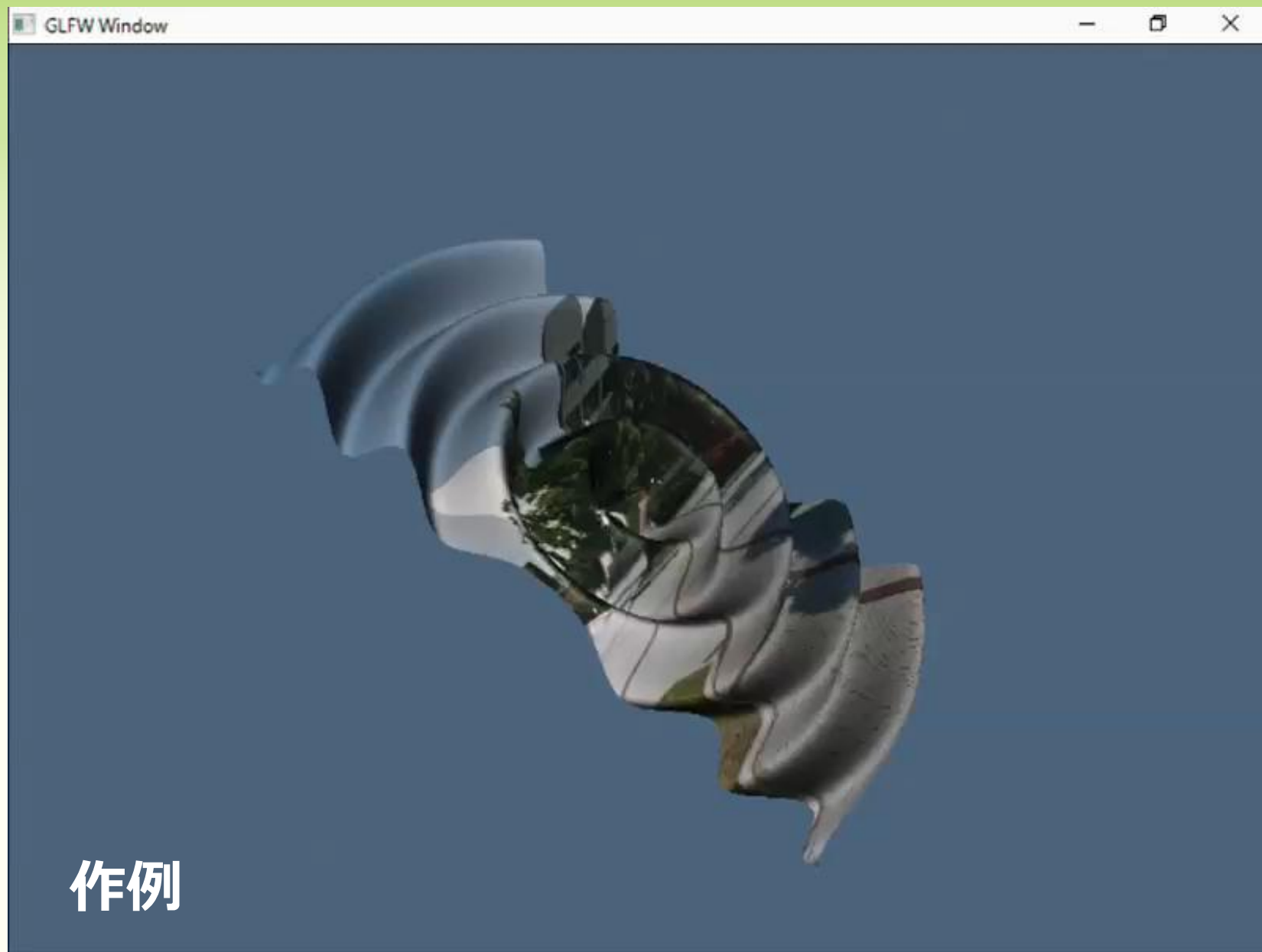
    // プログラムを終了しない
    return true;
}
```

- 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;

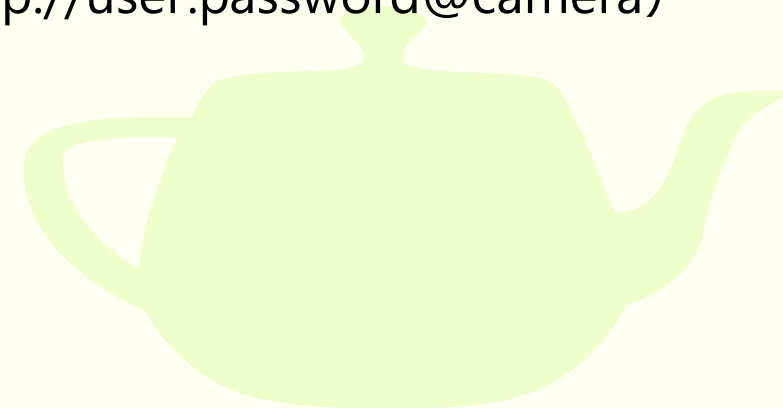
    // 最初のフレームを取得する
    video >> frame;

    // 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 カメラ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);

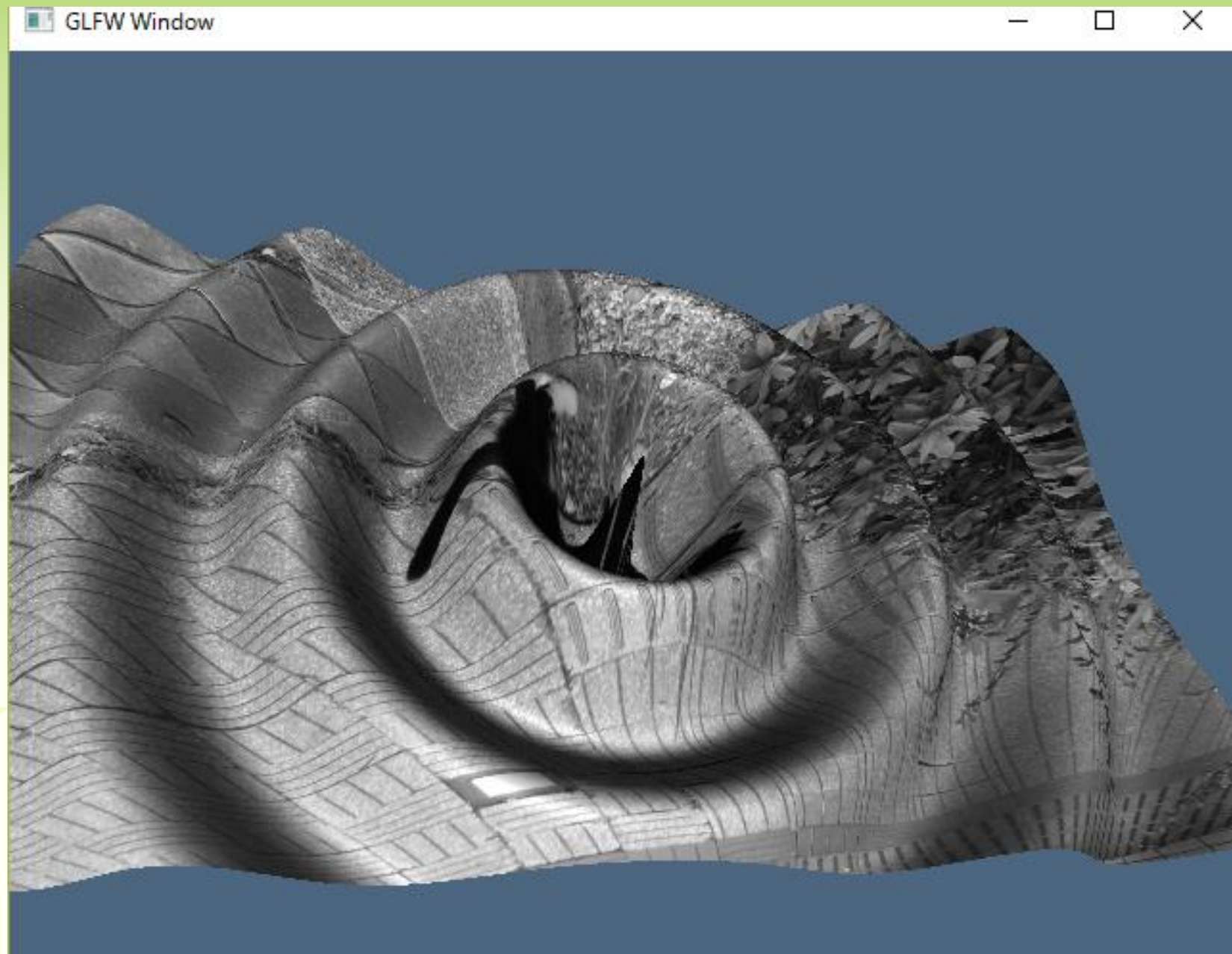
            // 保存したフレームを転送する
            submitColor(frame.cols, frame.rows, frame.data);
        }
    }

    // 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);

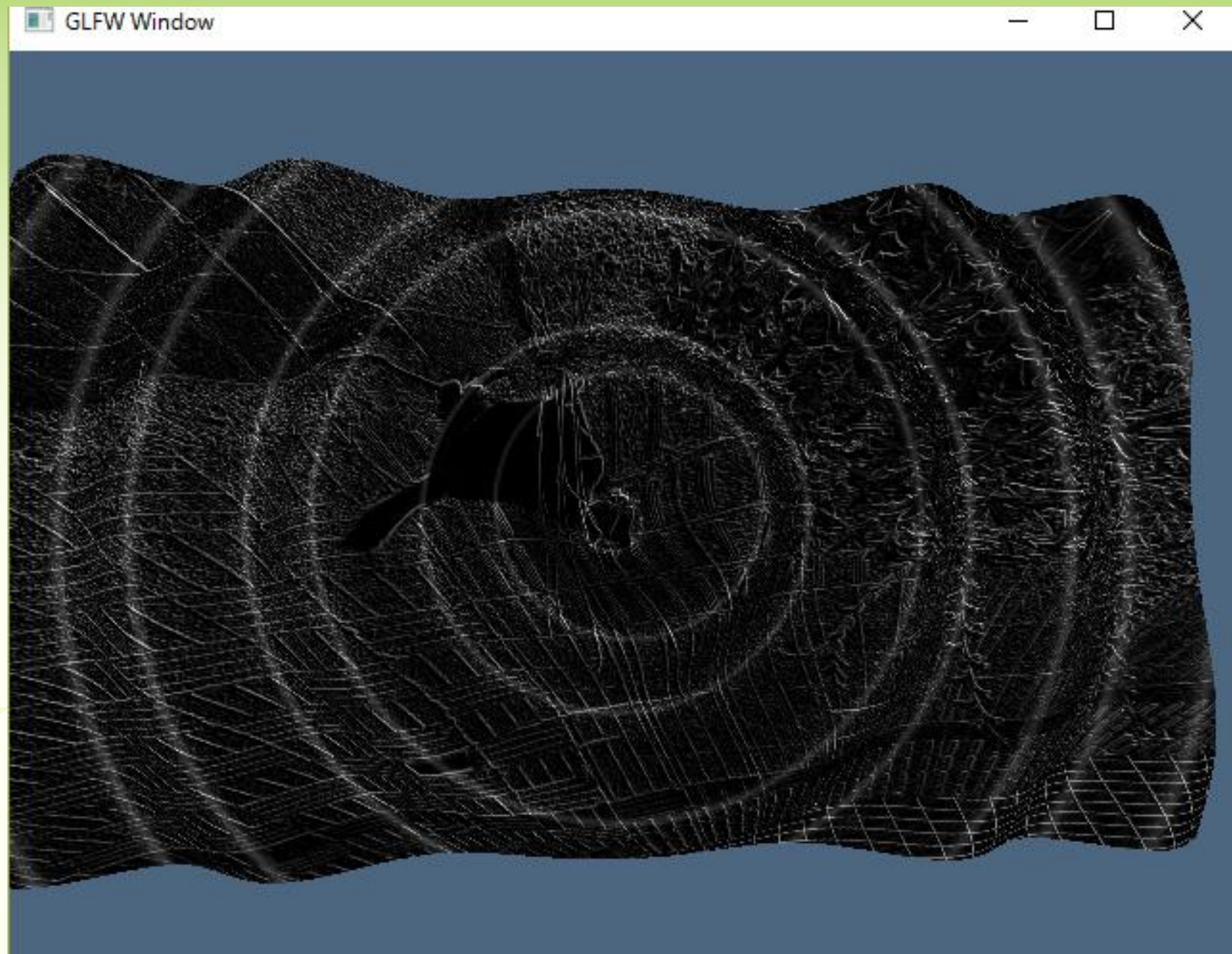
// 保存したフレームを転送する
submitColor(frame.cols, frame.rows, frame.data);
```

(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);

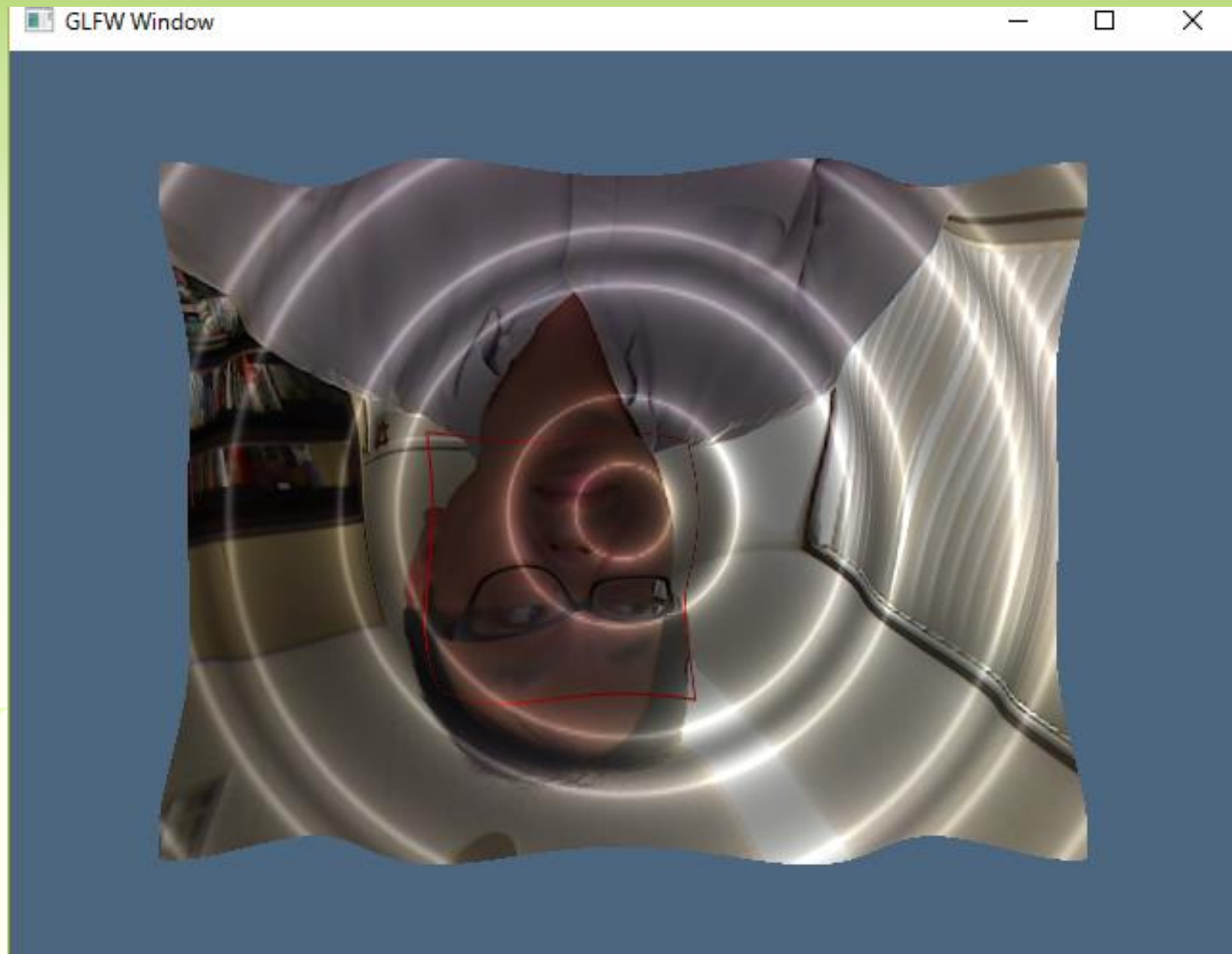
            // グレースケール化
            cv::Mat gray;
            cv::cvtColor(frame, gray, CV_BGR2GRAY);

            // 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);
}

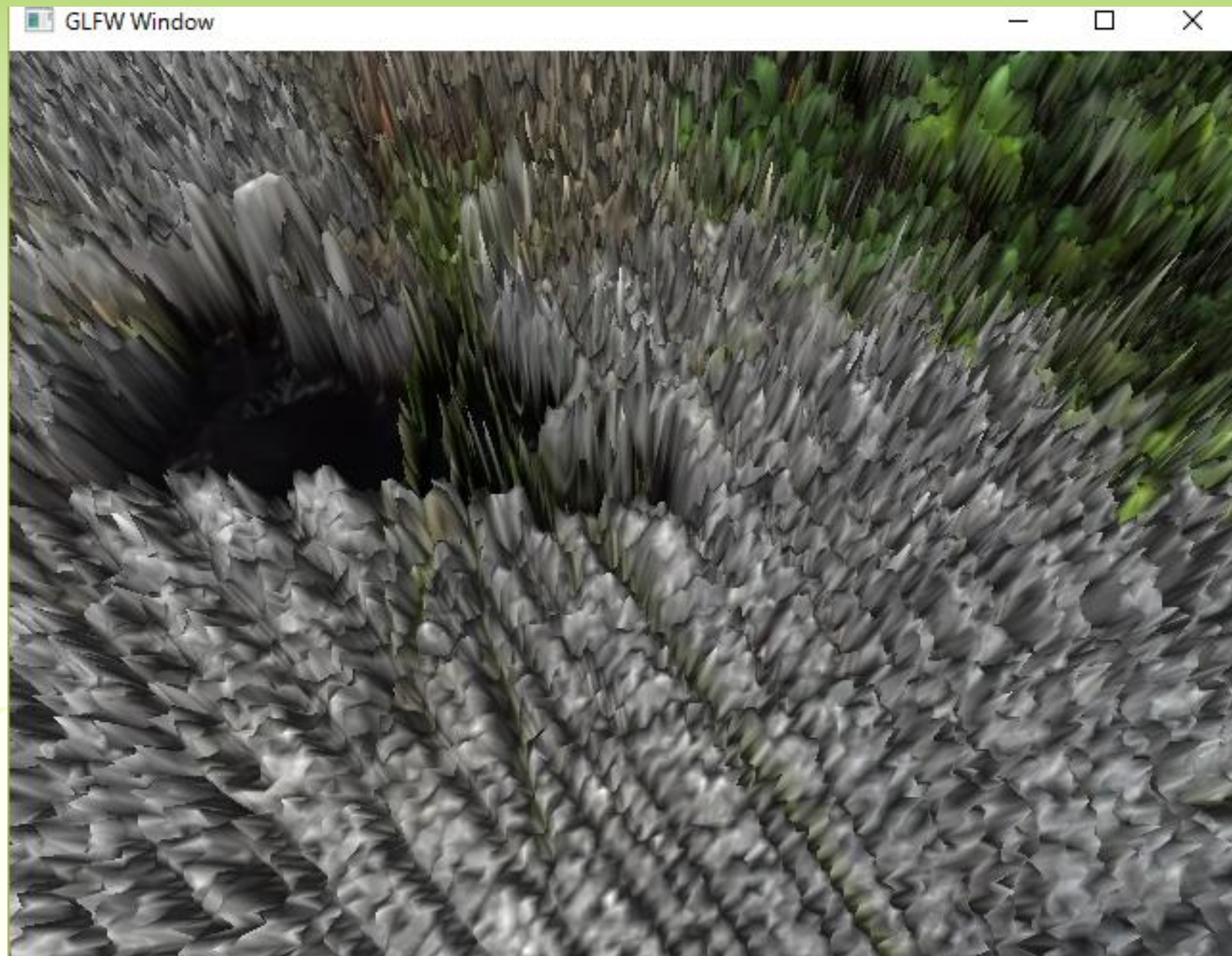
// プログラムを終了しない
return true;
}
```

**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 Effects」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

