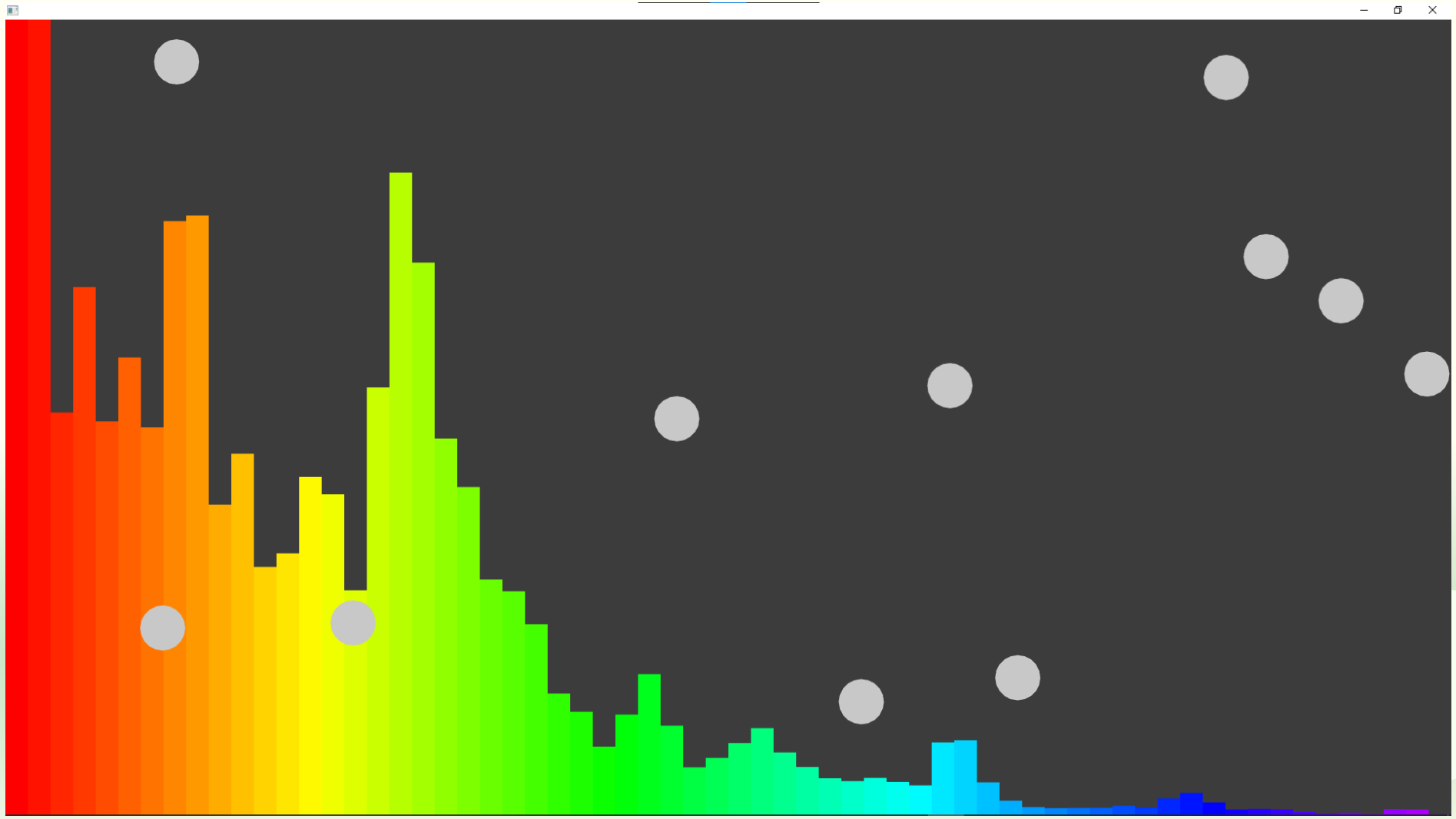
メディアプログラミング演習

第6回

本日はサウンドを使ったアプリの作成

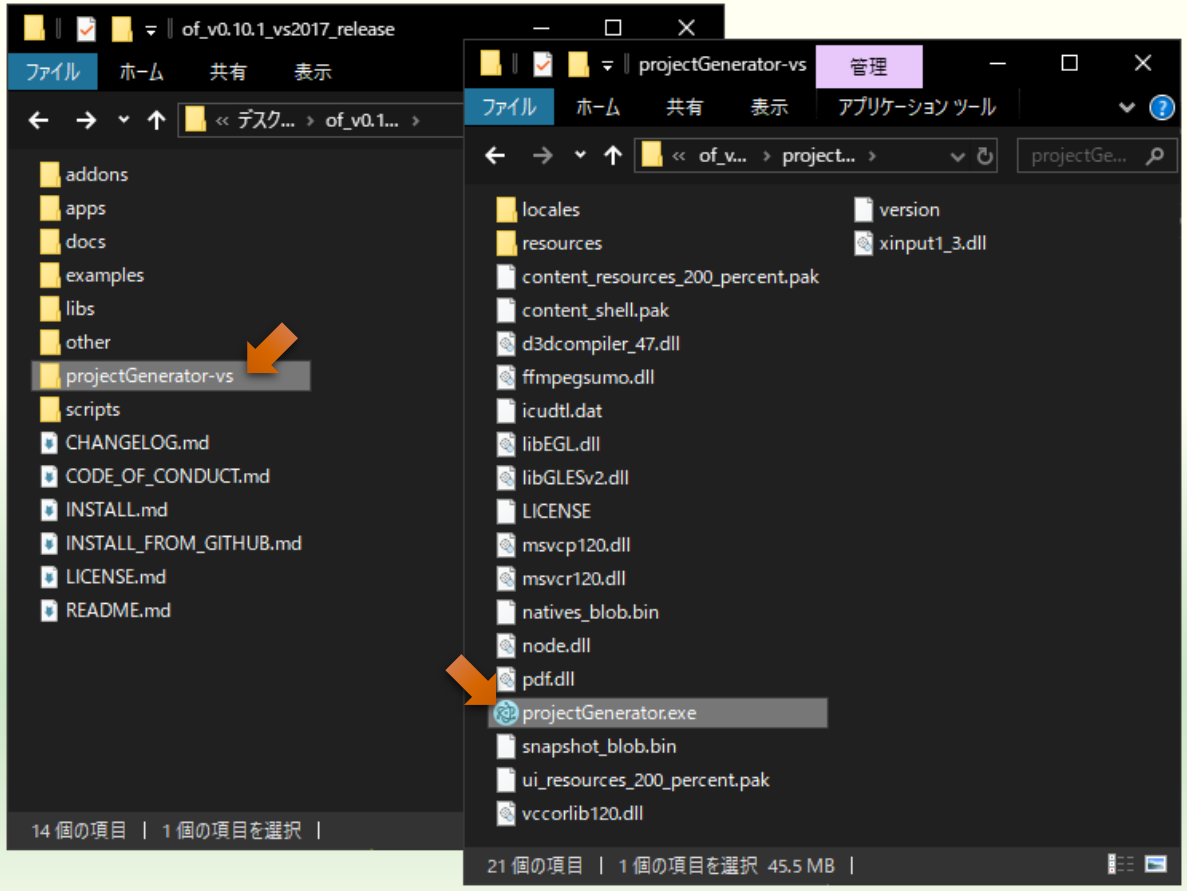


準備

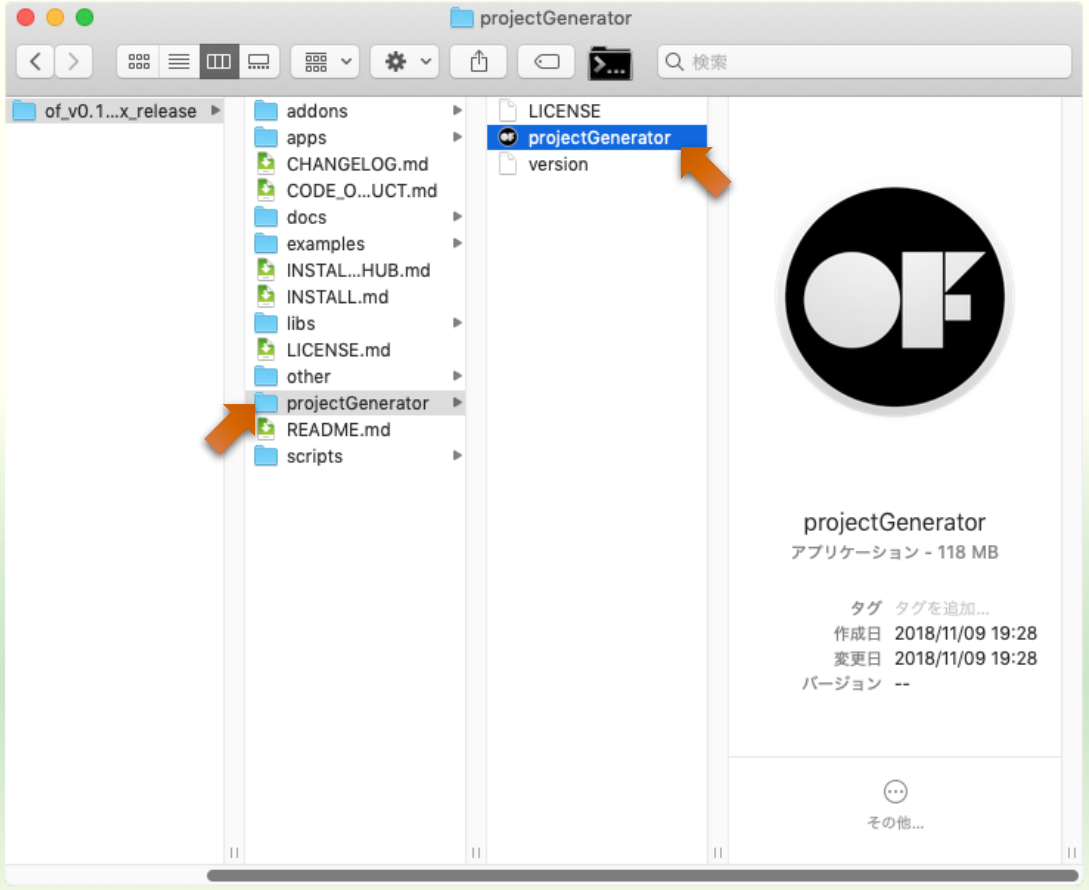
プロジェクトの作成

projectGenerator を起動する

windows 版のパッケージ



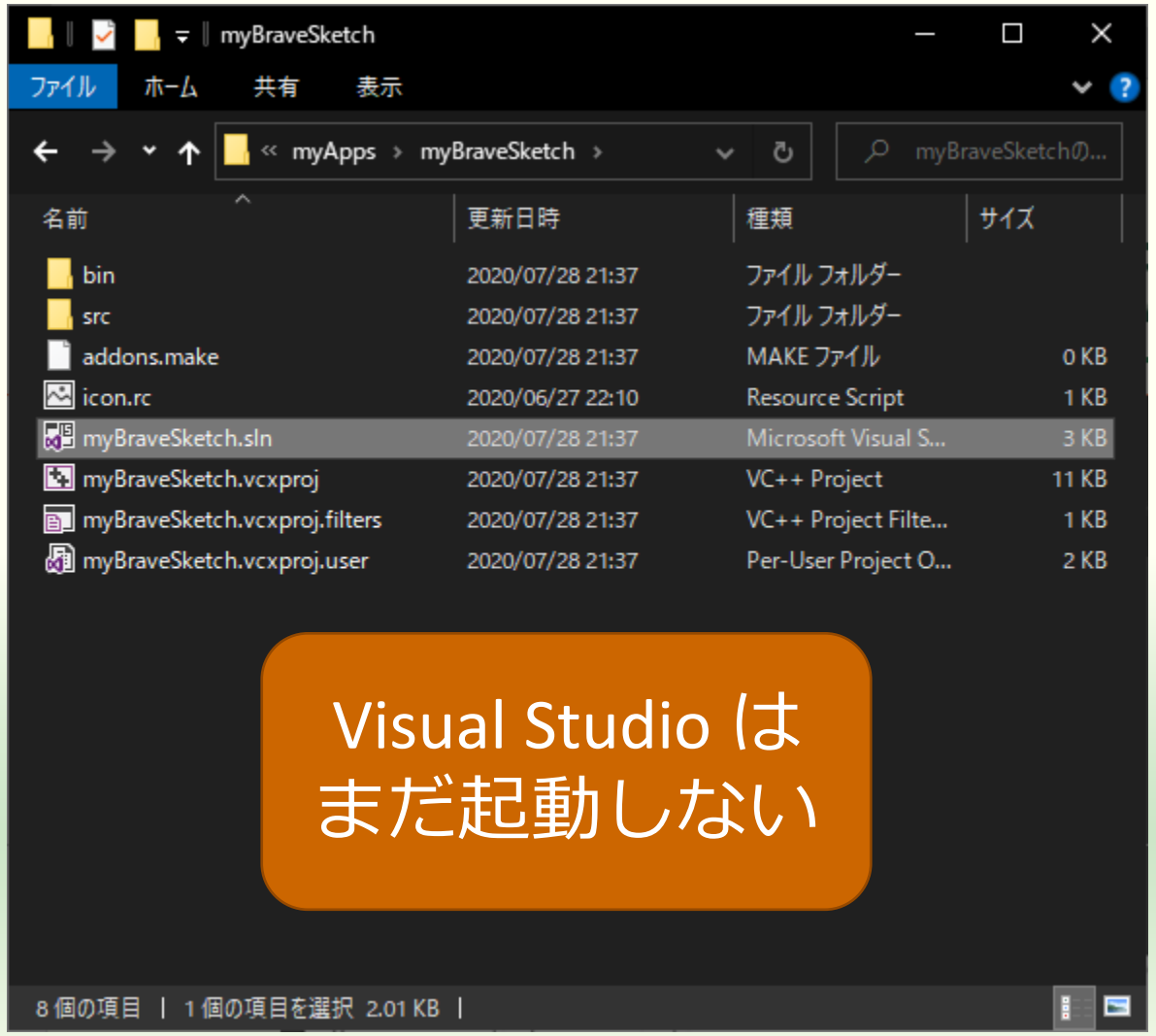
macOS 版のパッケージ



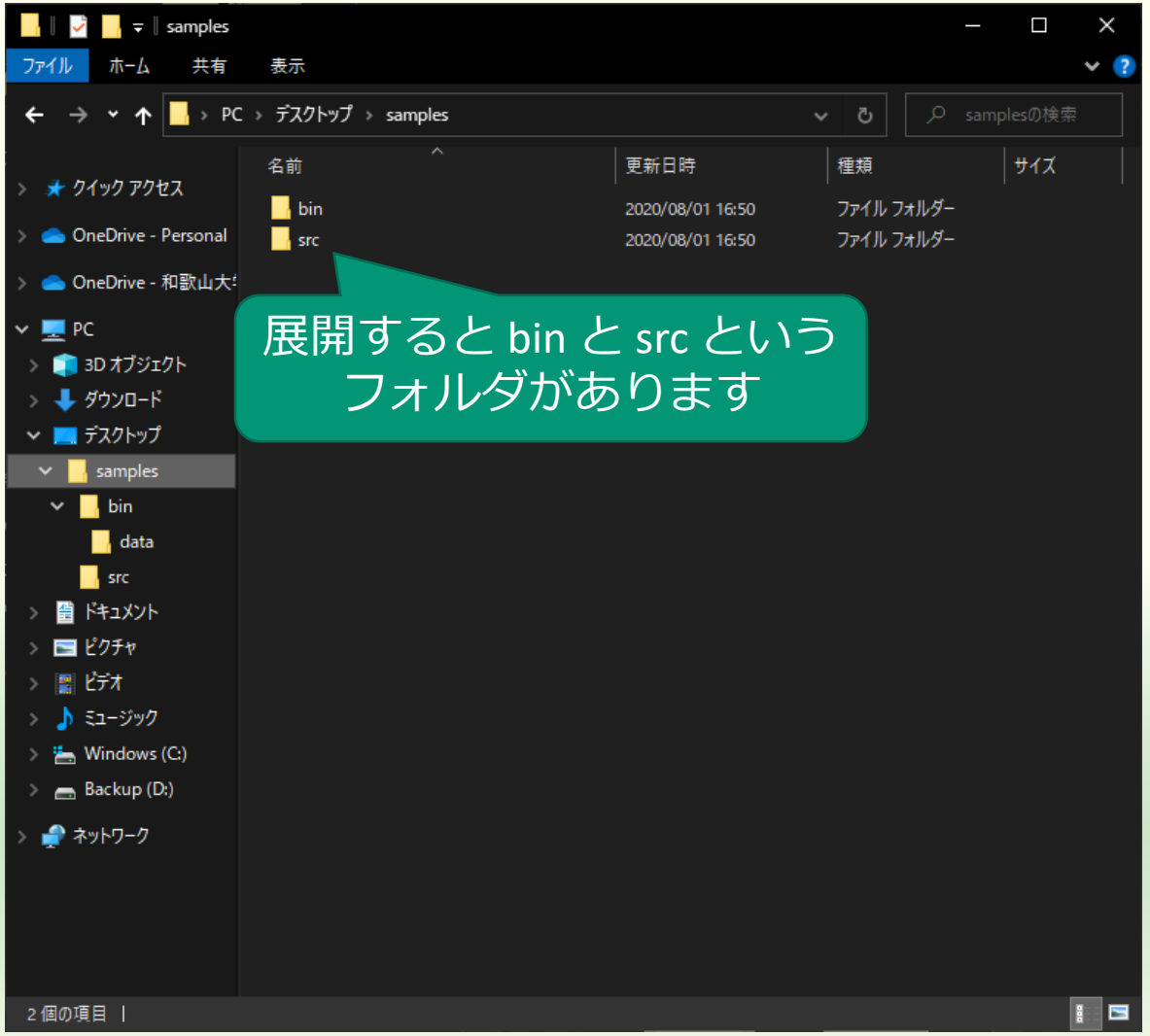
空のプロジェクトの作成

- Project name:
  - 作成するプロジェクト（プログラム）の名前
- Project path:
  - 作成するプロジェクトのファイルを置く場所
  - openFrameworks のパッケージを展開した場所の中の apps¥myApps

プロジェクトの作成成功

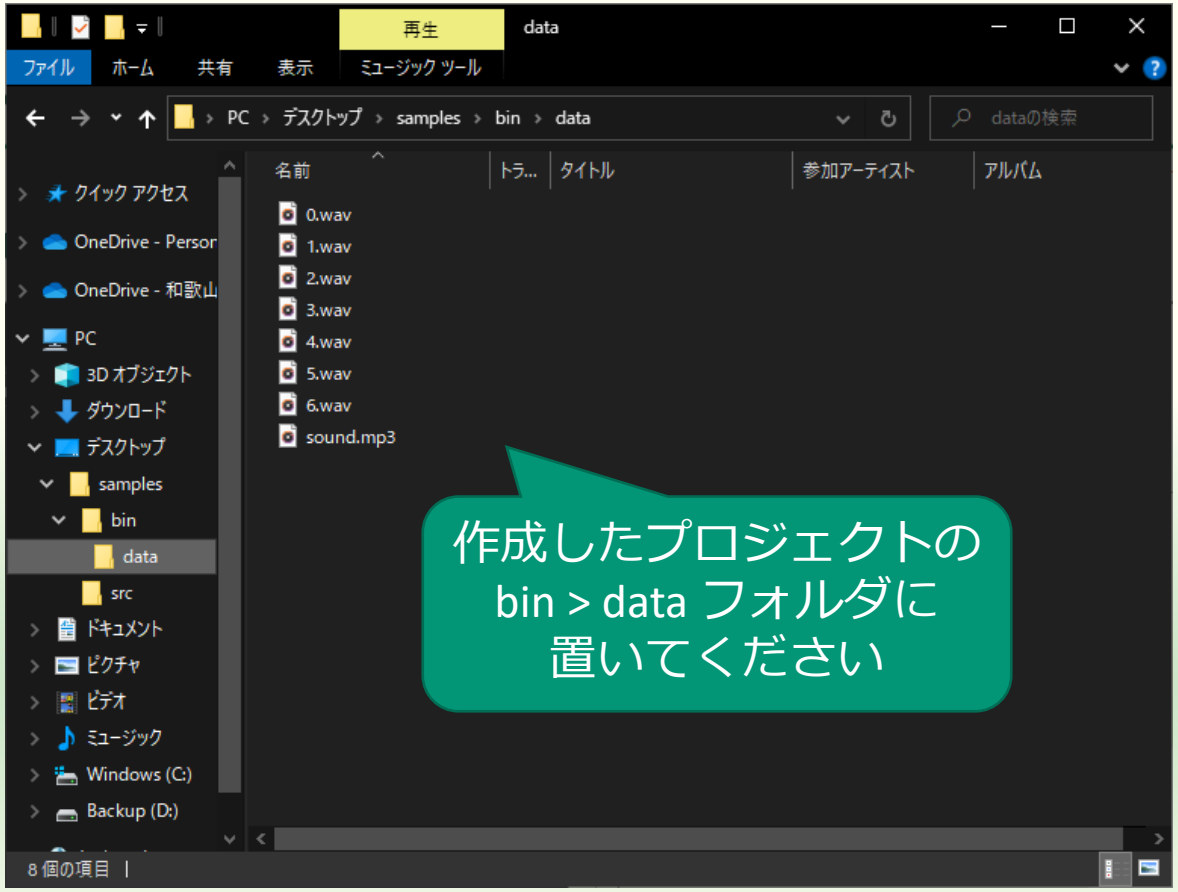


samples.zip のダウンロードと展開

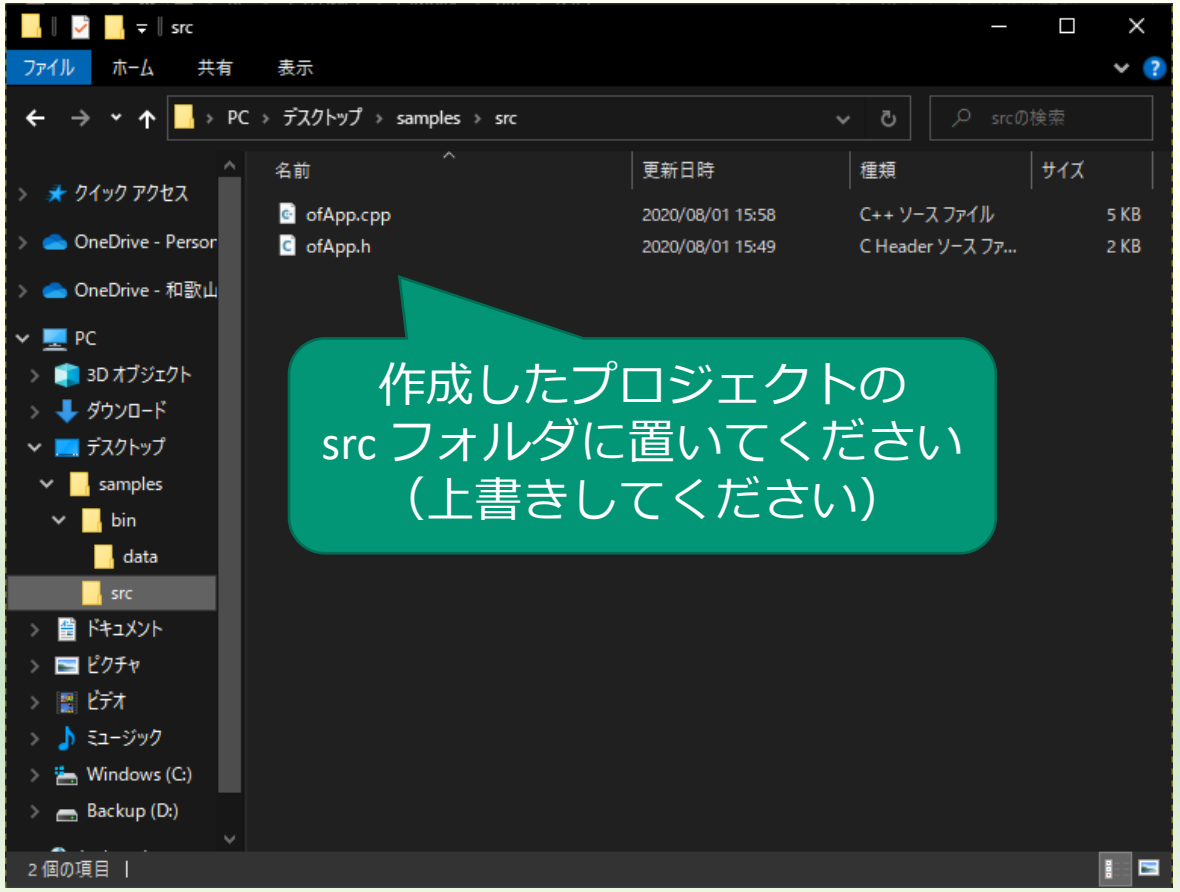


ファイルの配置

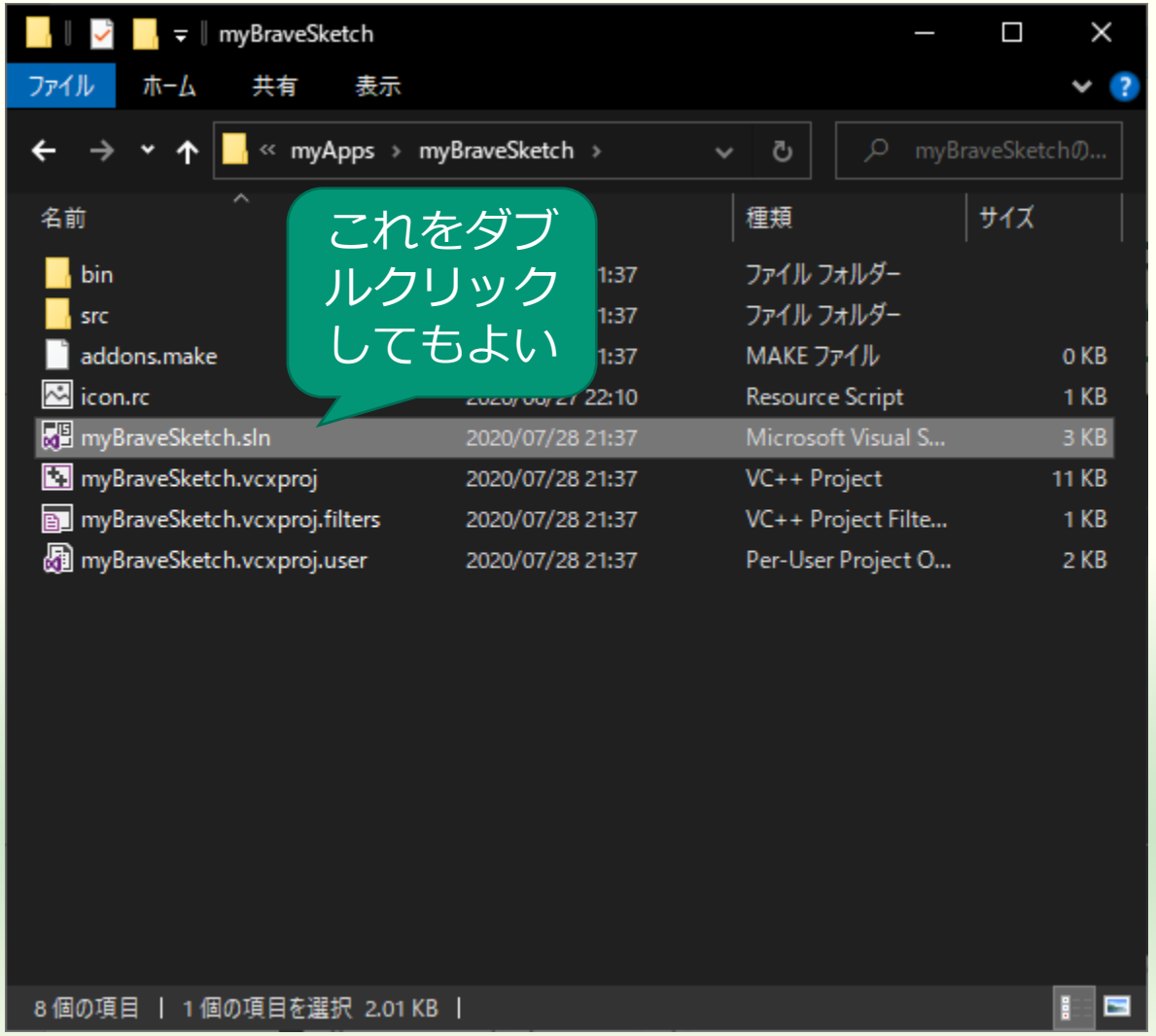
bin > data の内容



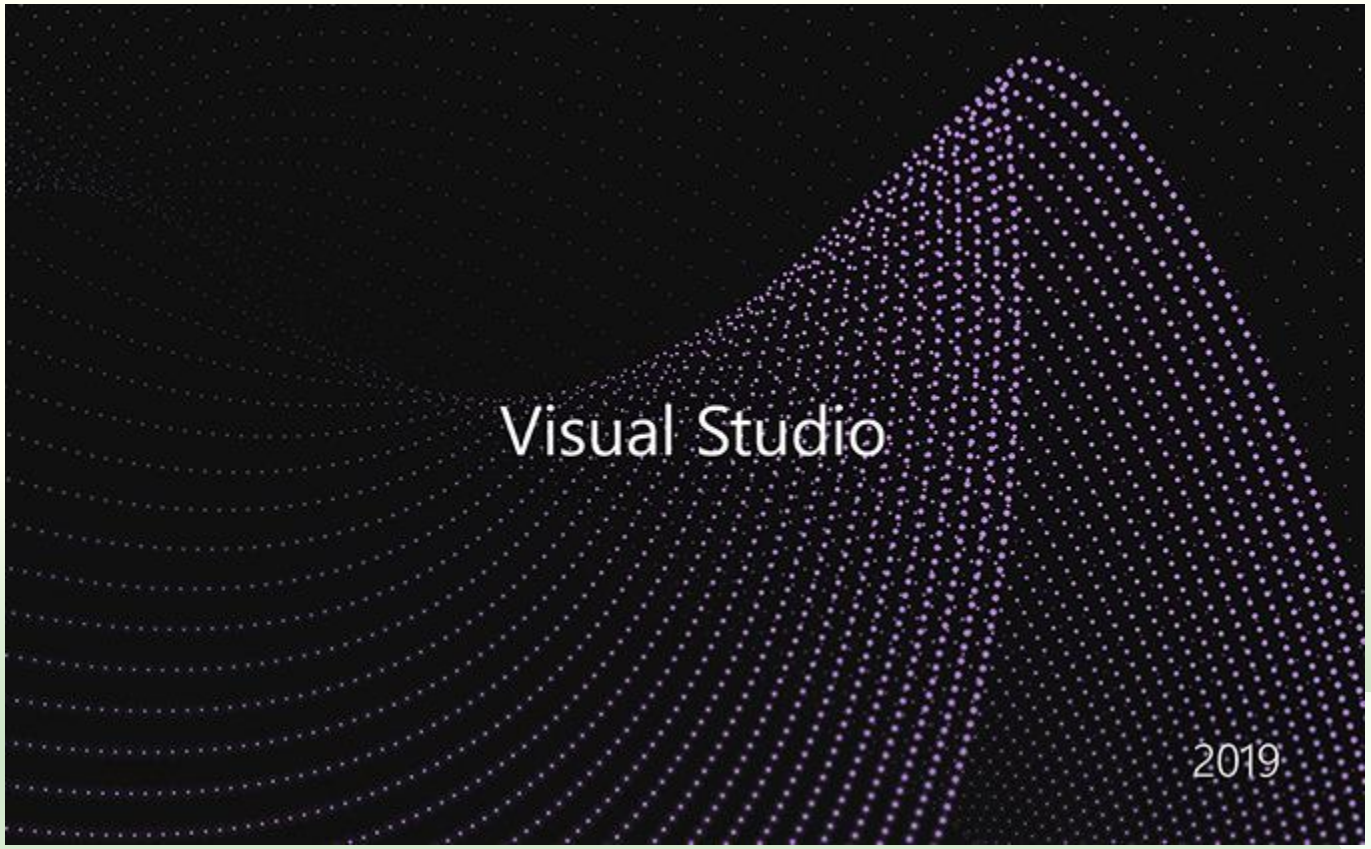
src の内容



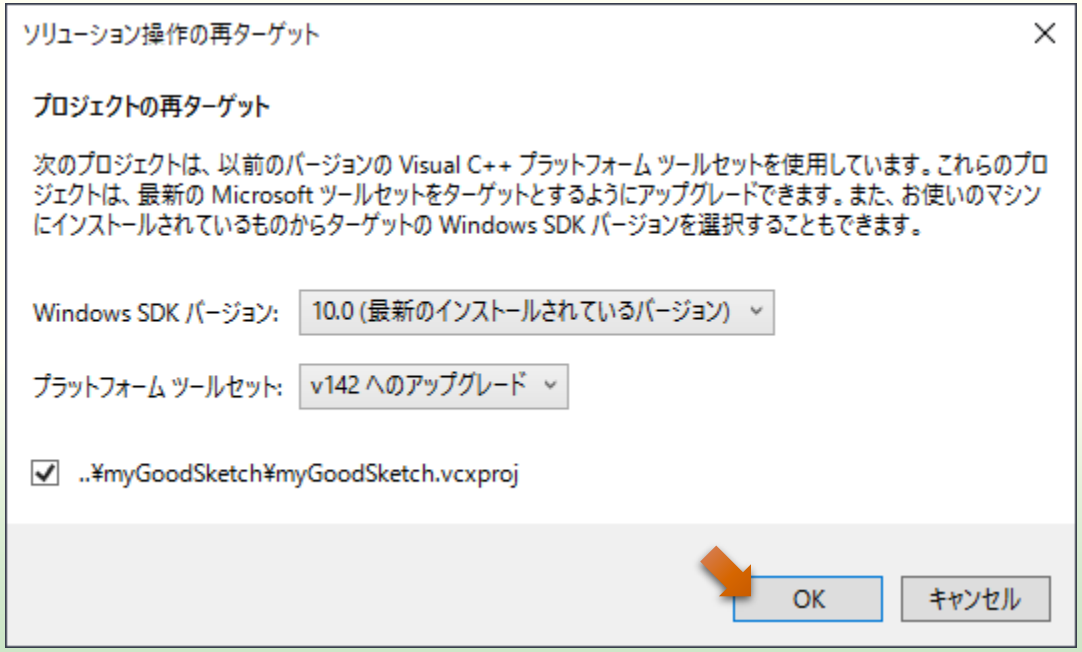
ソリューションファイルを開く



Visual Studio が起動する

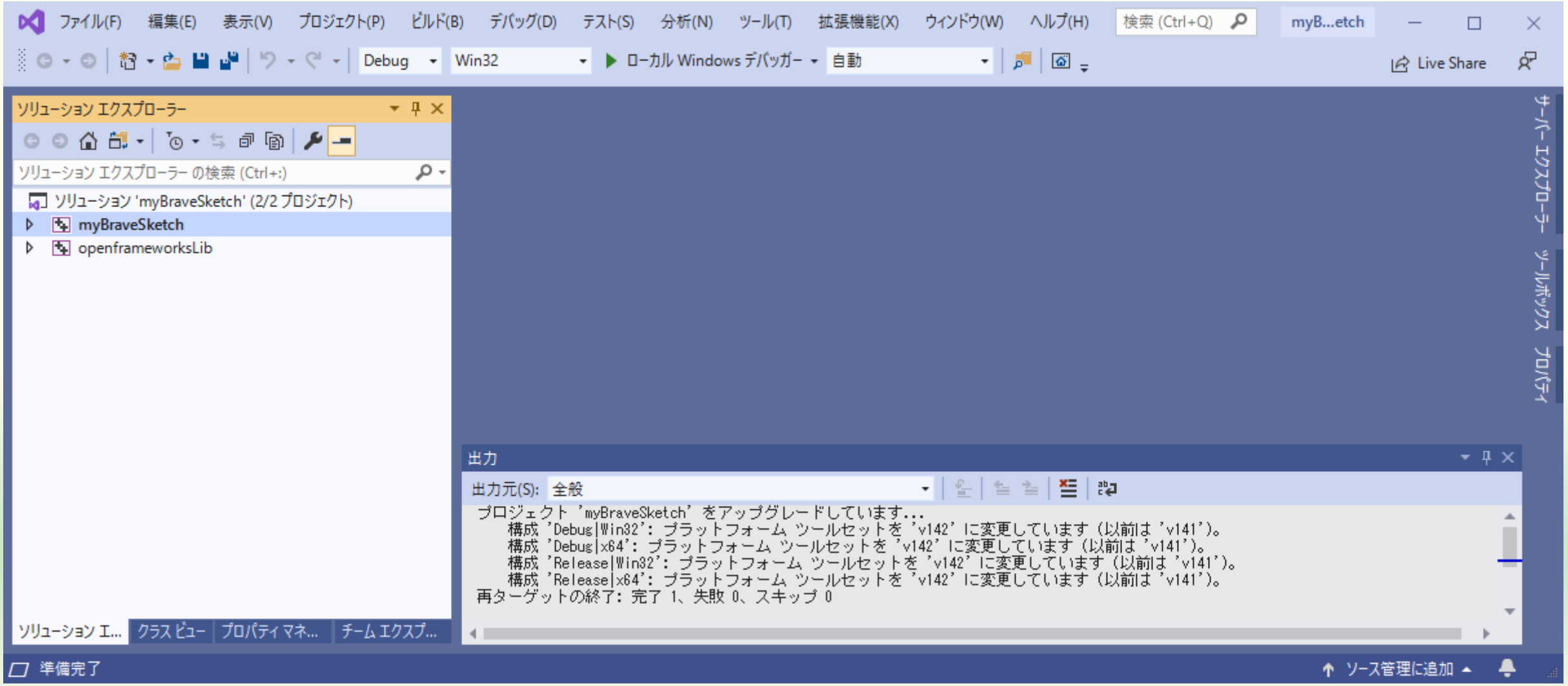


ソリューションの再ターゲット

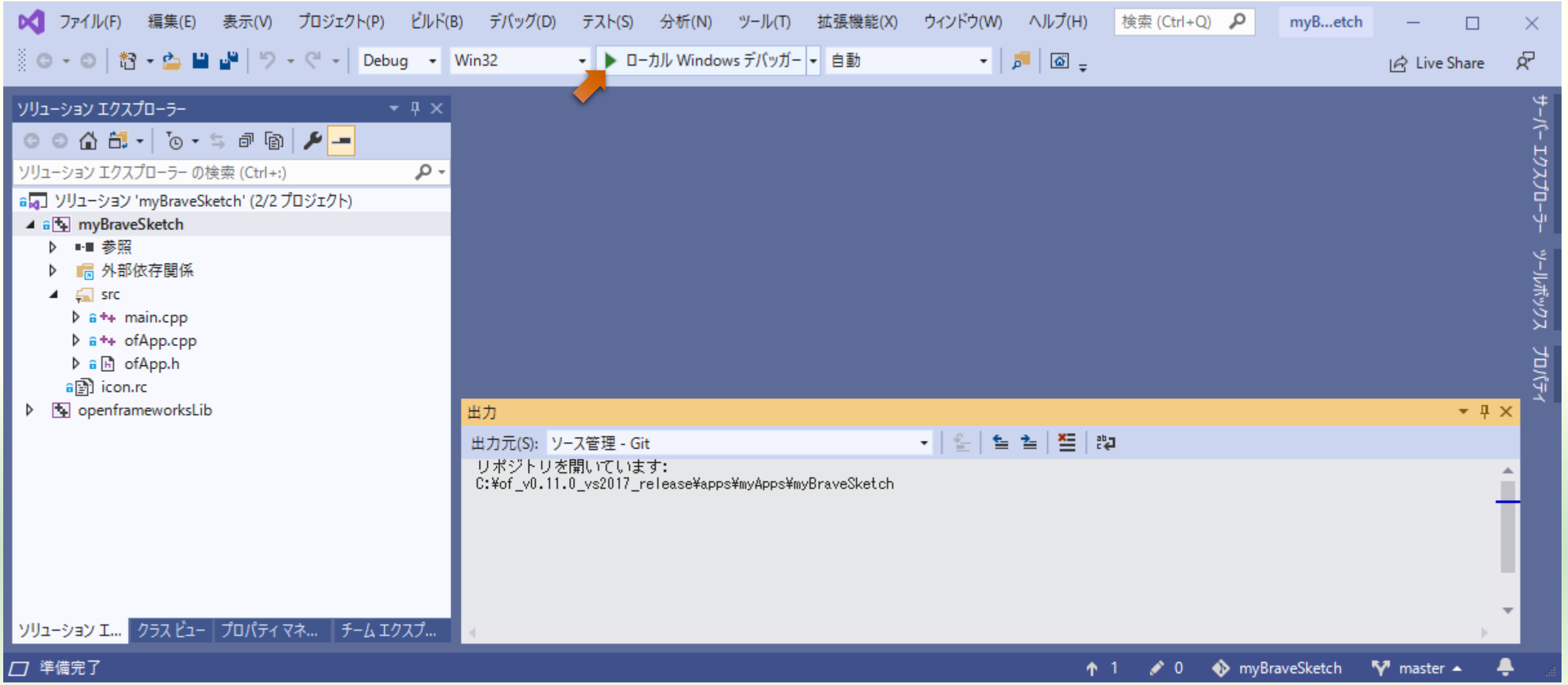


Visual Studio は頻繁に更新しているので皆さんがお使いの Visual Studio SDK のバージョンと合わない場合がある

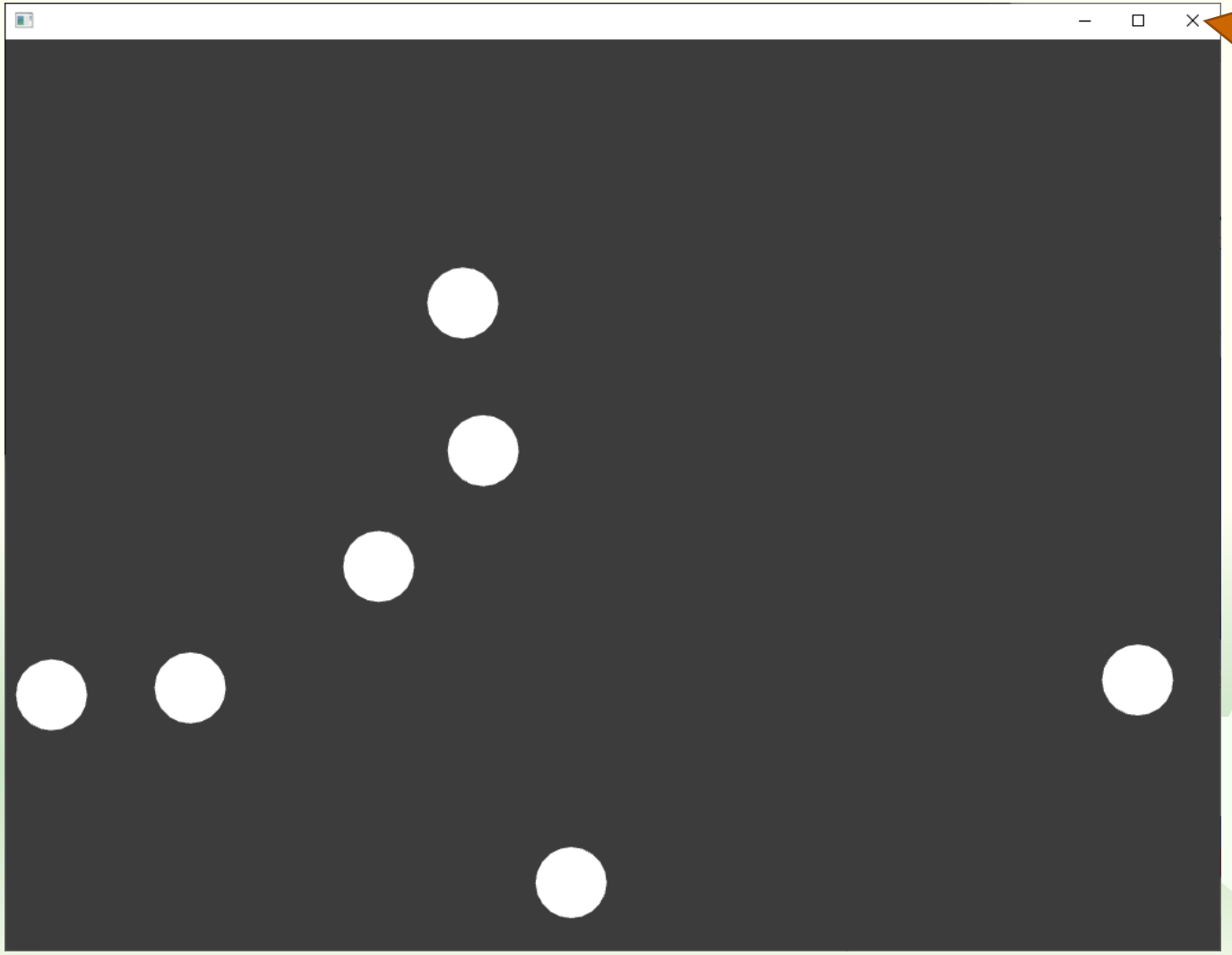
Visual Studio 起動



ビルドと実行


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)
テスト(S)
分析(N)
ツール(T)
拡張機能(X)
ウィンドウ(W)
ヘルプ(H)
検索 (Ctrl+Q)
Debug
Win32
ローカル Windows デバッガー
自動
Live Share
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 'myBraveSketch' (2/2 プロジェクト)
myBraveSketch
参照
外部依存関係
src
main.cpp
ofApp.cpp
ofApp.h
icon.rc
openframeworksLib
出力
出力元(S): ソース管理 - Git
リポジトリを開いています:
C:¥of_v0.11.0_vs2017_release¥apps¥myApps¥myBraveSketch
サーバー エクスプローラー
ツールボックス
プロパティ
ソリューション エ...
クラス ビュー
プロパティ マネ...
チーム エクスプ...
準備完了
1
0
myBraveSketch
master

課題２－５を複数の円に対応したもの



動作を確認したら止める

ofApp.h に Circle クラスの定義を追加している

```
#pragma once

#include "ofMain.h"

using namespace glm;

class Circle{

public:
  vec2 position;
  vec2 velocity;
  float radius;
  ofColor color;
};

class ofApp : public ofBaseApp{
  vector<Circle> circles;
  vec2 startPosition;
  float startTime;

   (以下略)
```

ひとまとまりのデータとして扱う

position など円の４つのデータを１つの vector に保持できる

- class Circle { … };
  - Circle というクラスの定義
    - メンバ変数: position（位置）velocity（速度）radius（半径）color（色）
    - public: 以降にあるメンバ変数・メンバ関数はメンバ関数以外から参照できる
- vector<Circle> circles;
  - Circle クラスの vector として circles を宣言
    - 個々の要素は position, velocity, radius, color のメンバを持つ

マウスボタンが押されたときに円を生成する

```
//--------------------------------------------------------------
void ofApp::mousePressed(int x, int y, int button){
  if (button == 0){                                  // もし押されたのが左ボタンなら
    startTime = ofGetElapsedTimef();                 // 押された時刻（経過時間）を startTime に保存する
    startPosition = vec2{ x, y };                    // 押された位置を startPosition に保存する
    Circle circle;                                   // Circle クラスのオブジェクト circle を一つ生成する
    circle.position = startPosition;                 // circle の位置 position をボタンを押した位置を設定する
    circle.velocity = vec2{ 0.0f, 0.0f };            // circle の速度 velocity を 0 にする
    circle.radius = 30.0f;                           // circle の半径を 30 にする
    circle.color = ofColor{ 200, 200, 200 };         // circle の色を R:200, G:200, B:200 にする
    circles.push_back(circle);                       // vector の circles に circle を追加する
  }
}
```

ofApp::mousePressed() は Circle クラスのメンバではないが、position、velocity、radius、color は public: メンバなので、circle.position のように “.”（ドット演算子）を使ってメンバにアクセスできる

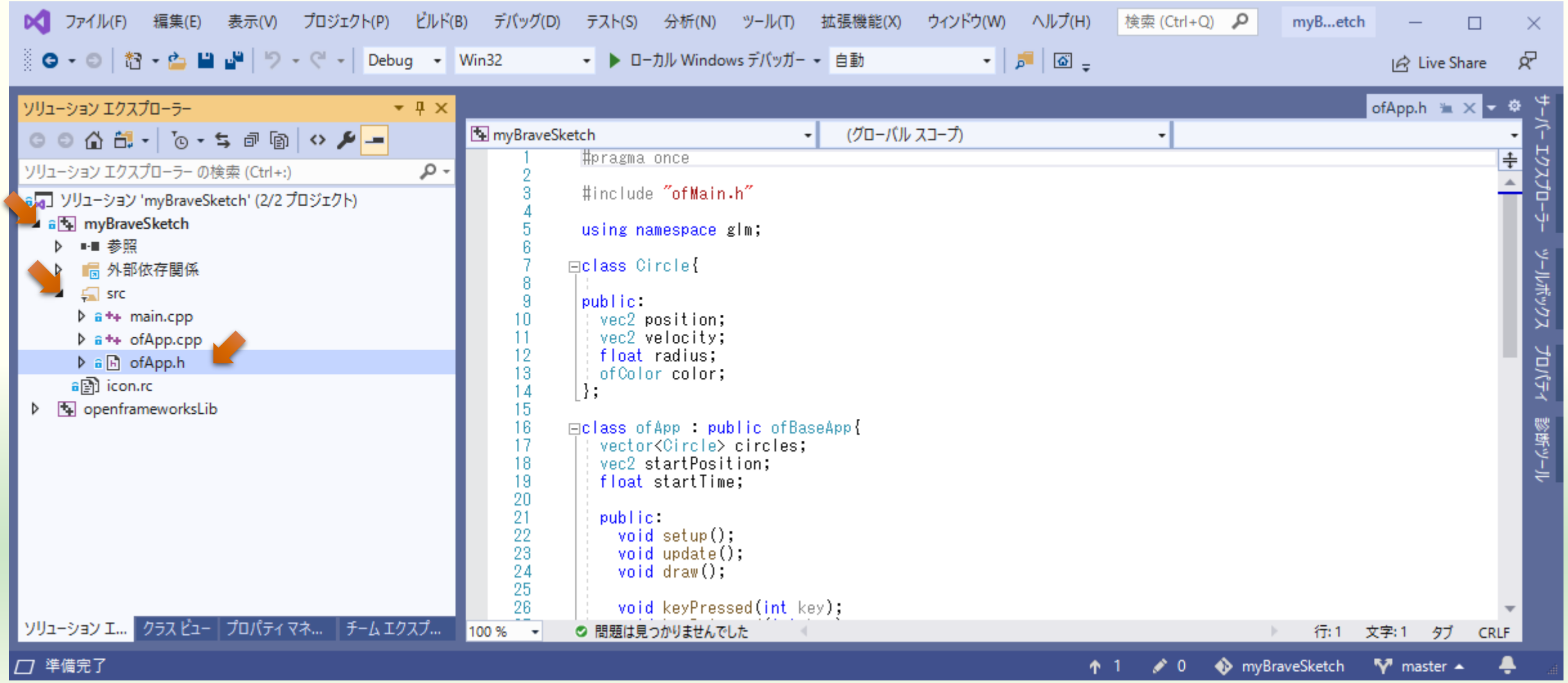
この場合は初期化を用いて以下のように書くこともできる

```
Circle circle{ startPosition, vec2{ 0.0f, 0.0f }, 30.0f, ofColor{ 200, 200, 200 } };
```

音声の再生

音声ファイルの読み込み

ofApp.h を開く

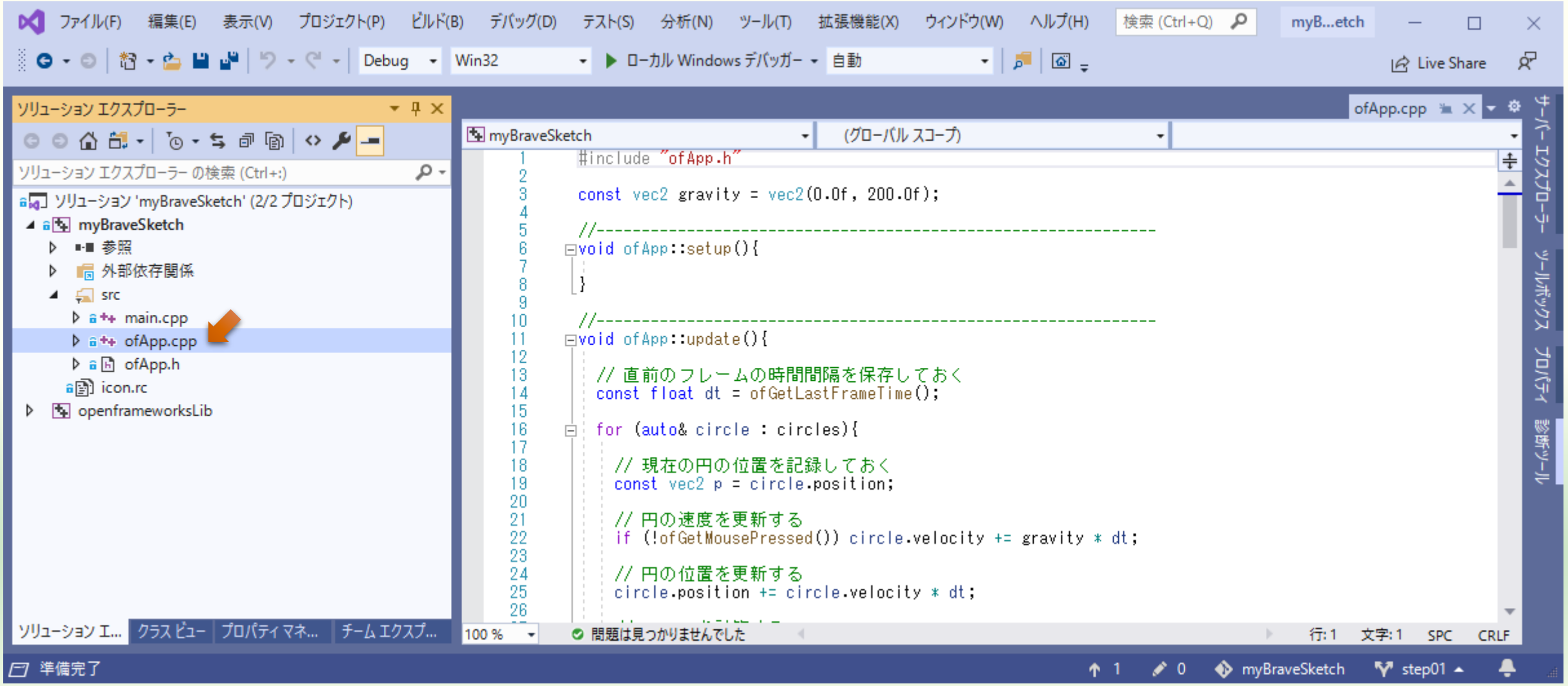

ファイル(F)
編集(E)
表示(V)
プロジェクト(P)
ビルド(B)
デバッグ(D)
テスト(S)
分析(N)
ツール(T)
拡張機能(X)
ウィンドウ(W)
ヘルプ(H)
検索 (Ctrl+Q)
Debug
Win32
ローカル Windows デバッガー
自動
Live Share
ソリューション エクスプローラー
ソリューション エクスプローラー の検索 (Ctrl+;)
ソリューション 'myBraveSketch' (2/2 プロジェクト)
myBraveSketch
参照
外部依存関係
src
main.cpp
ofApp.cpp
ofApp.h
icon.rc
openframeworksLib
ofApp.h
myBraveSketch
(グローバル スコープ)
#pragma once
#include "ofMain.h"
using namespace glm;
class Circle{
public:
vec2 position;
vec2 velocity;
float radius;
ofColor color;
};
class ofApp : public ofBaseApp{
vector<Circle> circles;
vec2 startPosition;
float startTime;
public:
void setup();
void update();
void draw();
void keyPressed(int key);
100 %
問題は見つかりませんでした
行: 1
文字: 1
タブ
CRLF
準備完了
1
0
myBraveSketch
master

ofAppクラスに音声再生のメンバ変数を追加する

```
（以上略）

class ofApp : public ofBaseApp{
  vector<Circle> circles;
  vec2 startPosition;
  float startTime;
  ofSoundPlayer sound;

public:
  void setup();
  void update();
  void draw();

（以下略）
```

- ofSoundPlayer
  - サウンドファイルの読み込みと再生を行うクラス
    - ボリューム、パン、スピード、シーク、マルチプレイのコントロールが可能
    - プラットフォームごとに異なるサウンド再生機能に対して統一したインターフェイスを与えたもの

ofApp.cpp を開く

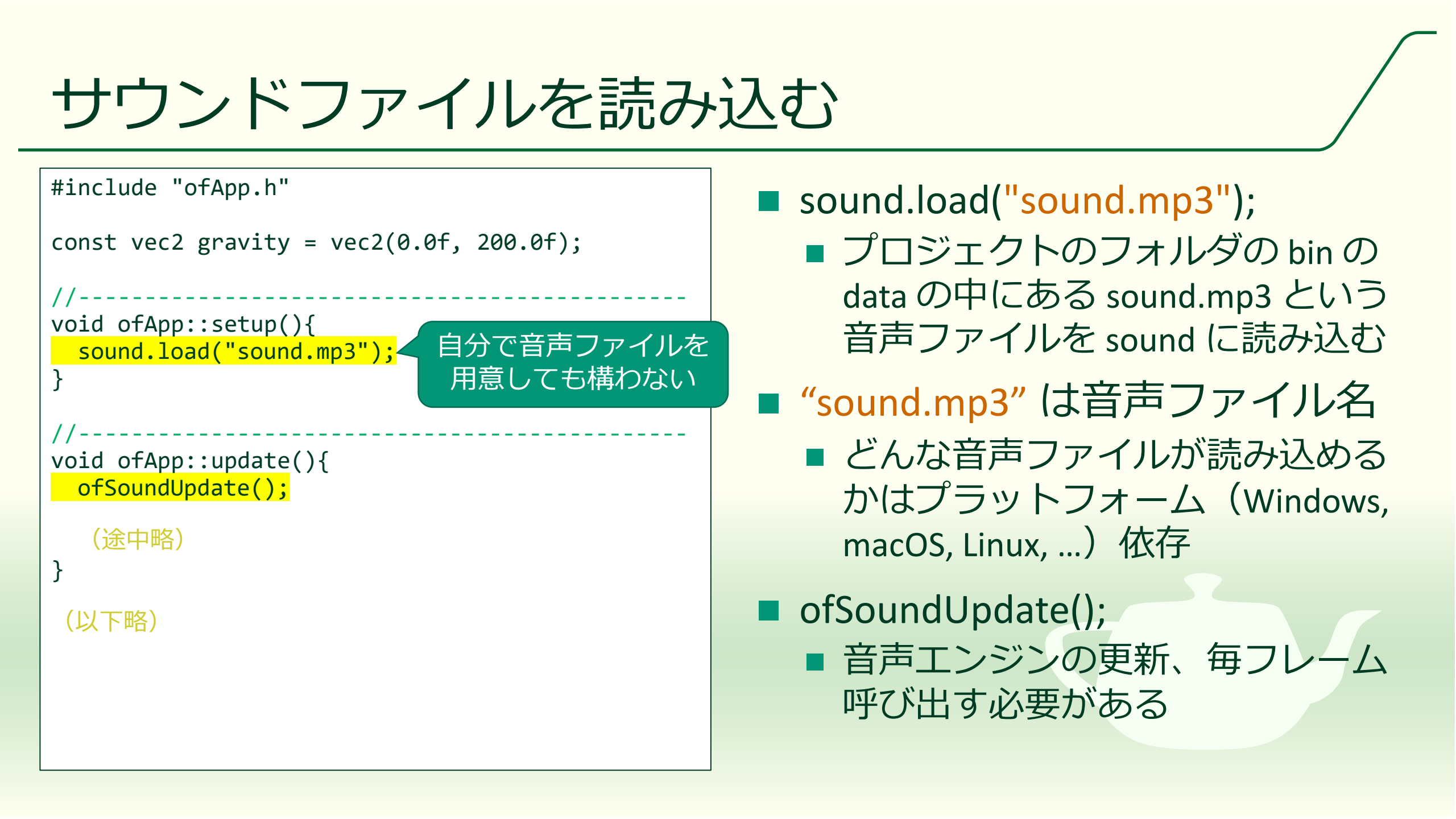


サウンドファイルを読み込む

```
#include "ofApp.h"

const vec2 gravity = vec2(0.0f, 200.0f);

//--------------------------------------------
void ofApp::setup(){
  sound.load("sound.mp3");
}

//--------------------------------------------
void ofApp::update(){
  ofSoundUpdate();

   (途中略)
}

(以下略)
```

自分で音声ファイルを用意しても構わない

- sound.load("sound.mp3");
  - プロジェクトのフォルダの bin の data の中にある sound.mp3 という音声ファイルを sound に読み込む
- “sound.mp3” は音声ファイル名
  - どんな音声ファイルが読み込めるかはプラットフォーム（Windows, macOS, Linux, …）依存
- ofSoundUpdate();
  - 音声エンジンの更新、毎フレーム呼び出す必要がある

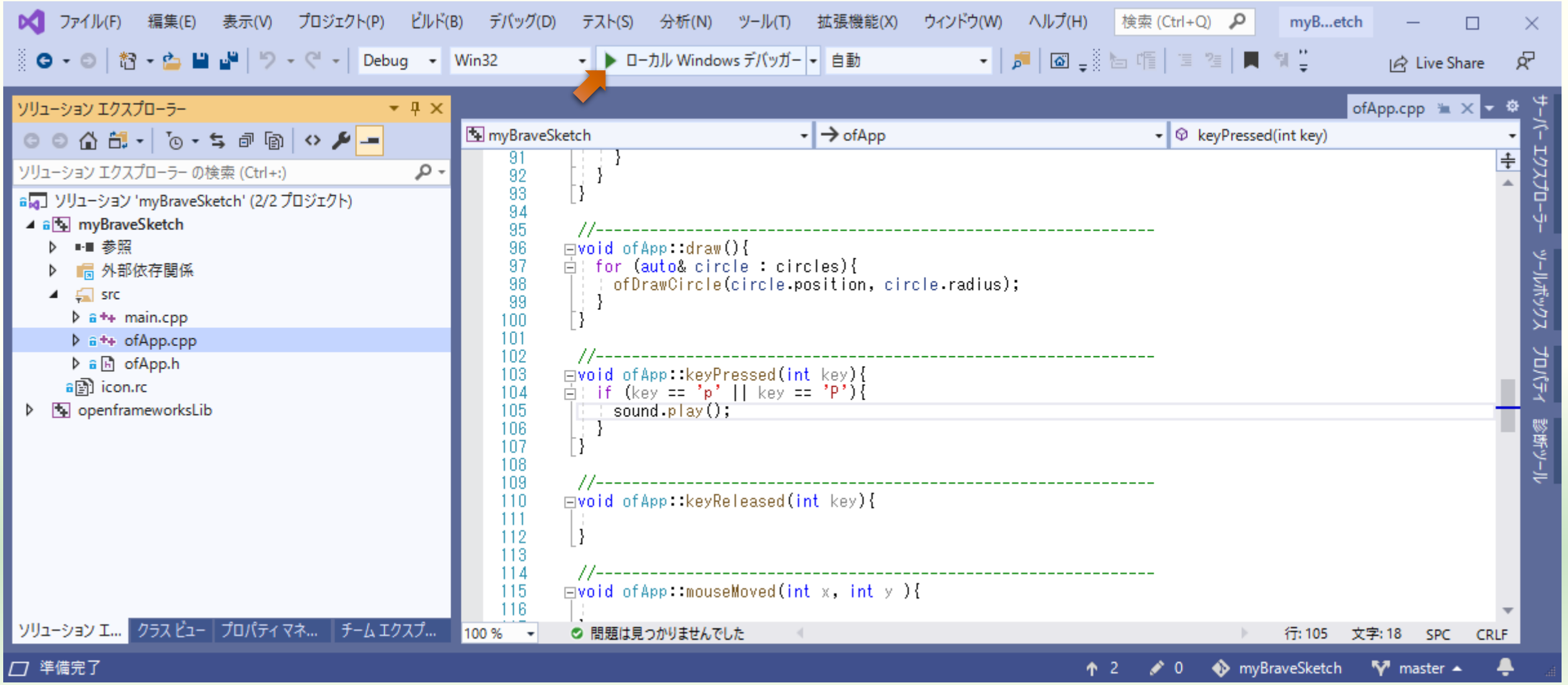


‘p’ または ‘P’ キーのタイプでサウンドファイルを再生する

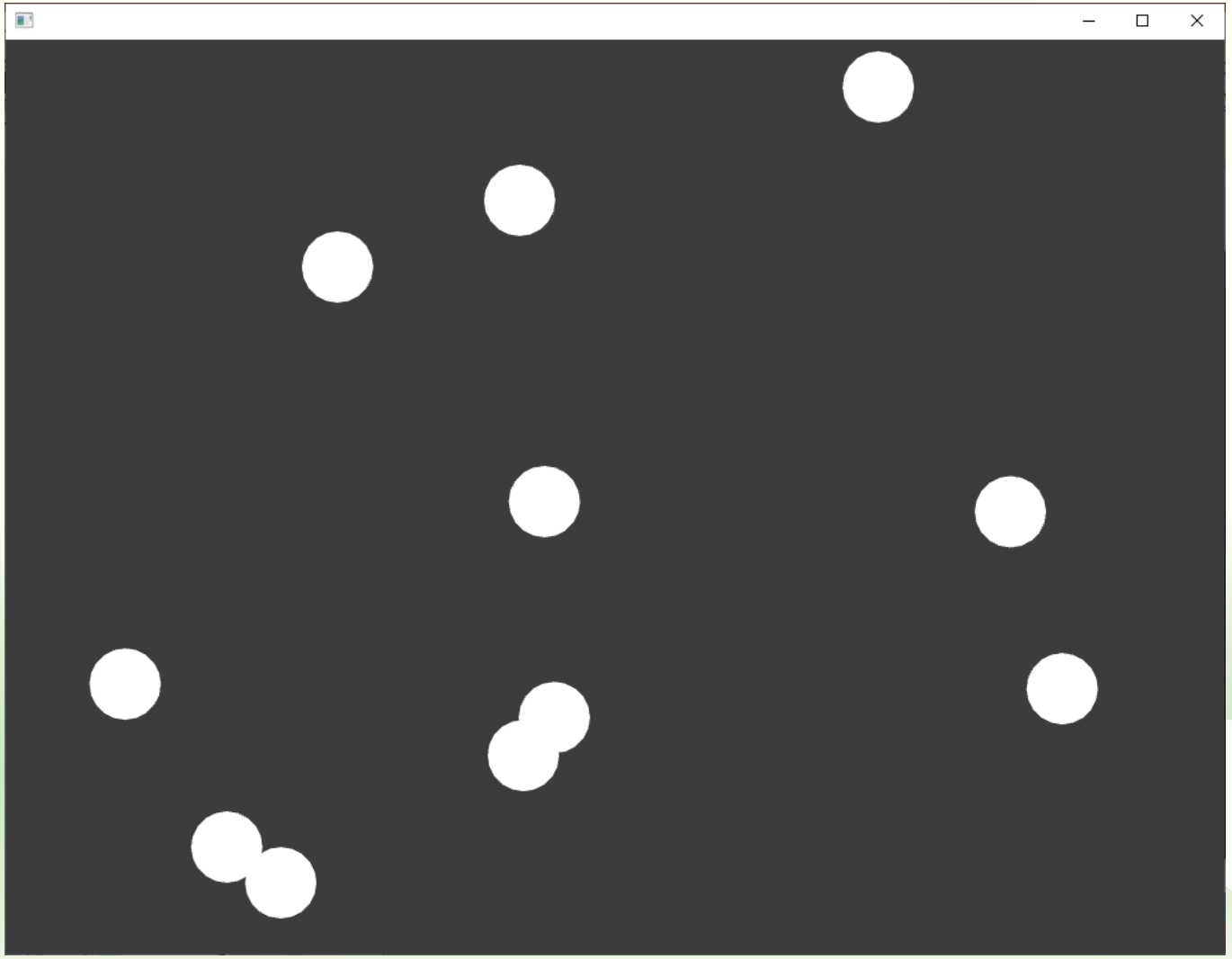
```
（以上略）

//--------------------------------------------
void ofApp::draw(){
   （途中略）
}

//--------------------------------------------
void ofApp::keyPressed(int key){
  if (key == 'p' || key == 'P'){
    sound.play();
  }
}

//--------------------------------------------
void ofApp::keyReleased(int key){

}

（以下略）
```

- sound.play();
  - sound に読み込んだ（load した）音声ファイルを再生する
  - これを setup() で実行するとパソコンによってはうまく再生されないことがある

ビルドと実行


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) テスト(S) 分析(N) ツール(T) 拡張機能(X) ウィンドウ(W) ヘルプ(H)
Debug
Win32
ローカル Windows デバッガー
自動
Live Share
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 'myBraveSketch' (2/2 プロジェクト)
myBraveSketch
参照
外部依存関係
src
main.cpp
ofApp.cpp
ofApp.h
icon.rc
openframeworksLib
ofApp.cpp
myBraveSketch
ofApp
keyPressed(int key)
91 }
92 }
93 }
94
95 //--
96 void ofApp::draw(){
97 for (auto& circle : circles){
98 ofDrawCircle(circle.position, circle.radius);
99 }
100 }
101
102 //--
103 void ofApp::keyPressed(int key){
104 if (key == 'p' || key == 'P'){
105 sound.play();
106 }
107 }
108
109 //--
110 void ofApp::keyReleased(int key){
111
112 }
113
114 //--
115 void ofApp::mouseMoved(int x, int y){
116
100 %
問題は見つかりませんでした
行: 105 文字: 18 SPC CRLF
ソリューション エ...
クラス ビュー
プロパティ マネ...
チーム エクスプ...
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
2
0
myBraveSketch
master

‘p’ か ‘P’ をタイプしてサウンドを再生してみる



課題６－１

跳ね返るときに音を出す

円が壁で跳ね返るときに効果音を再生しなさい

- bin > data に置いた 0.wav ～ 6.wav は 1 秒未満の短い音声ファイルである
- 円が壁で跳ね返るときにこれらを再生するようにしなさい
- 音声ファイルは自分で用意しても構わない
  - Windows の「ボイスレコーダー」アプリや macOS / iOS / iPadOS の「ボイスメモ」アプリで作成した AAC ファイル（拡張子 .m4a）は Windows の ofSoundPlayer クラスでは多分再生できない
  - mp3 か wav に変換する
- 同じ音を同時に鳴らすには load() した後に setMultiPlay(true)

サウンドファイルをループ再生する

```
（以上略）

//--------------------------------------------
void ofApp::setup(){
  sound.load("sound.mp3");
  sound.setLoop(true);
}

（途中略）

//--------------------------------------------
void ofApp::keyPressed(int key){
  if (key == 'p' || key == 'P'){
    sound.play();
  }
  else if (key == 's' || key == 'S'){
    sound.stop();
  }
}

（以下略）
```

- sound.setLoop(true);
  - 音声ファイルをループ再生（エンドレス再生）するようにする
  - 引数の true が false の場合はループ再生しない
- sound.stop();
  - 音声ファイルの再生を停止する

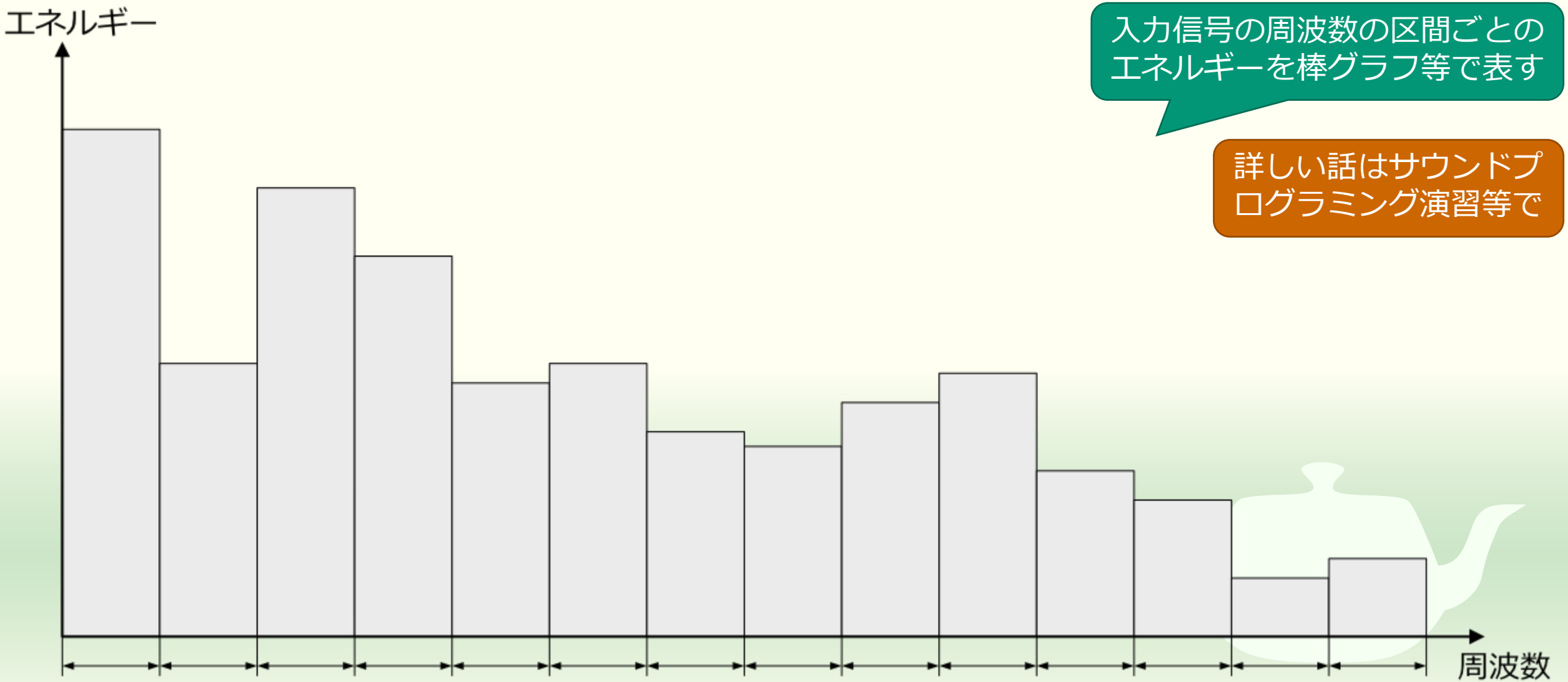
課題のアップロード

- 作成したプログラムの実行中のウィンドウを**５秒以内**で動画キャプチャして、**6-1.mp4** というファイル名で Moodle の第６回課題にアップロードしてください
  - 動画のキャプチャができないときはスクリーンショットを撮って 6-1.png というファイル名でアップロードしてください

スペクトル表示

音の周波数分布

スペクトル分布



入力信号の周波数の区間ごとのエネルギーを棒グラフ等で表す

詳しい話はサウンドプログラミング演習等で

スペクトル分布を格納するメンバ変数を追加

```
（以上略）

class ofApp : public ofBaseApp{
  vector<Circle> circles;
  vec2 startPosition;
  float startTime;
  ofSoundPlayer sound;
  array<float, 64> spectrum{};

public:
  void setup();
  void update();
  void draw();

（以下略）
```

- array<float, 64> spectrum{};
  - array は**固定長**配列
    - サイズ（要素の数）を指定して宣言することによりメモリを確保する
    - vector のように後からデータを追加したり削除したりすることはできない
    - この場合のサイズは 64、spectrum[0] ～spectrum[63] の要素を持つ
- array の初期化
  - 例）array<int, **5**> x{ 3, 1, 2 };
    - x[0] は 3、x[1] は 1、x[2] は 2 に初期化
    - 初期値が指定されていない x[3], x[4] は 0 で初期化される
    - {} だと全部 0 で初期化される

スペクトラム分布の抽出

```
（以上略）

//--------------------------------------------
void ofApp::update(){

  ofSoundUpdate();

  const size_t nBands{ spectrum.size() };
  const float *val{ ofSoundGetSpectrum(nBands) };

   （次ページに続く）
```

- spectrum.size()
  - spectrum の要素数を返す
  - これを周波数の区間 nBands に使う
- float *ofSoundGetSpectrum(int nBands)
  - 再生中の音声から高速フーリエ変換 (Fast Fourier Transform, FFT) を用いてスペクトル分布を求める
  - 結果が格納されたメモリへのポインタを返す
  - nBands は周波数の区間の数

スペクトラム分布のグラフデータの作成

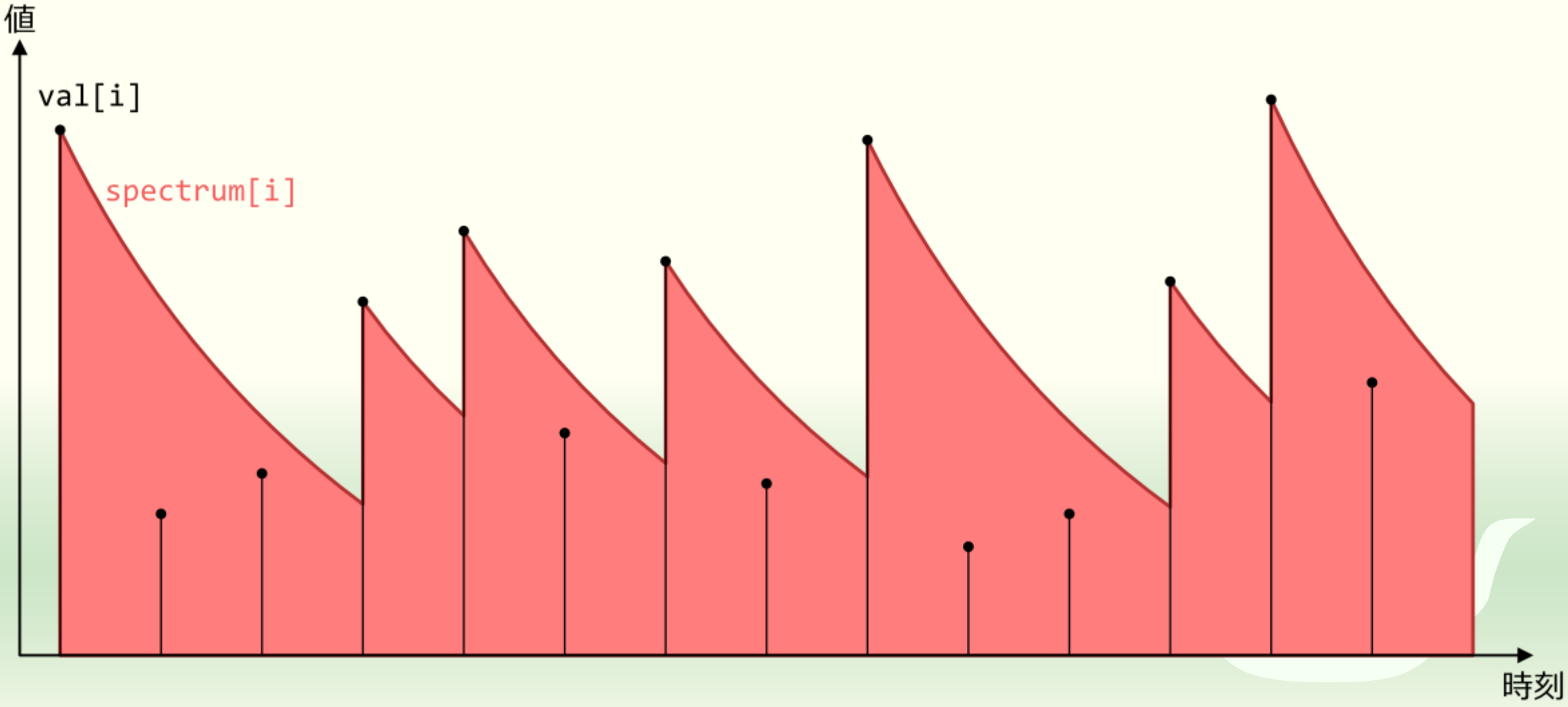
（全ページからの続き）

```
  for (size_t i = 0; i < nBands; ++i){
    spectrum[i] *= 0.96f;
    if (spectrum[i] < val[i]){
      spectrum[i] = val[i];
    }
  }
```

（以下略）

- for (size_t i = 0; i < nBands; ++i) {
  - i を 0 から nBands - 1 まで変化させながら {} 内を繰り返す
  - spectrum[i] *= 0.96f;
    - 以前の値を 0.96 倍することで時間の経過に伴い値が指数関数的に減少する
  - if (spectrum[i] < val[i]){
  - もし入力信号の値 val[i] が現在の値 spectrum[i] を超えていたら
    - spectrum[i] = val[i];
    - spectrum[i] を入力信号の値 val[i] に更新する

スペクトル分布のグラフの変化



課題６－２

スペクトル分布のグラフを描く

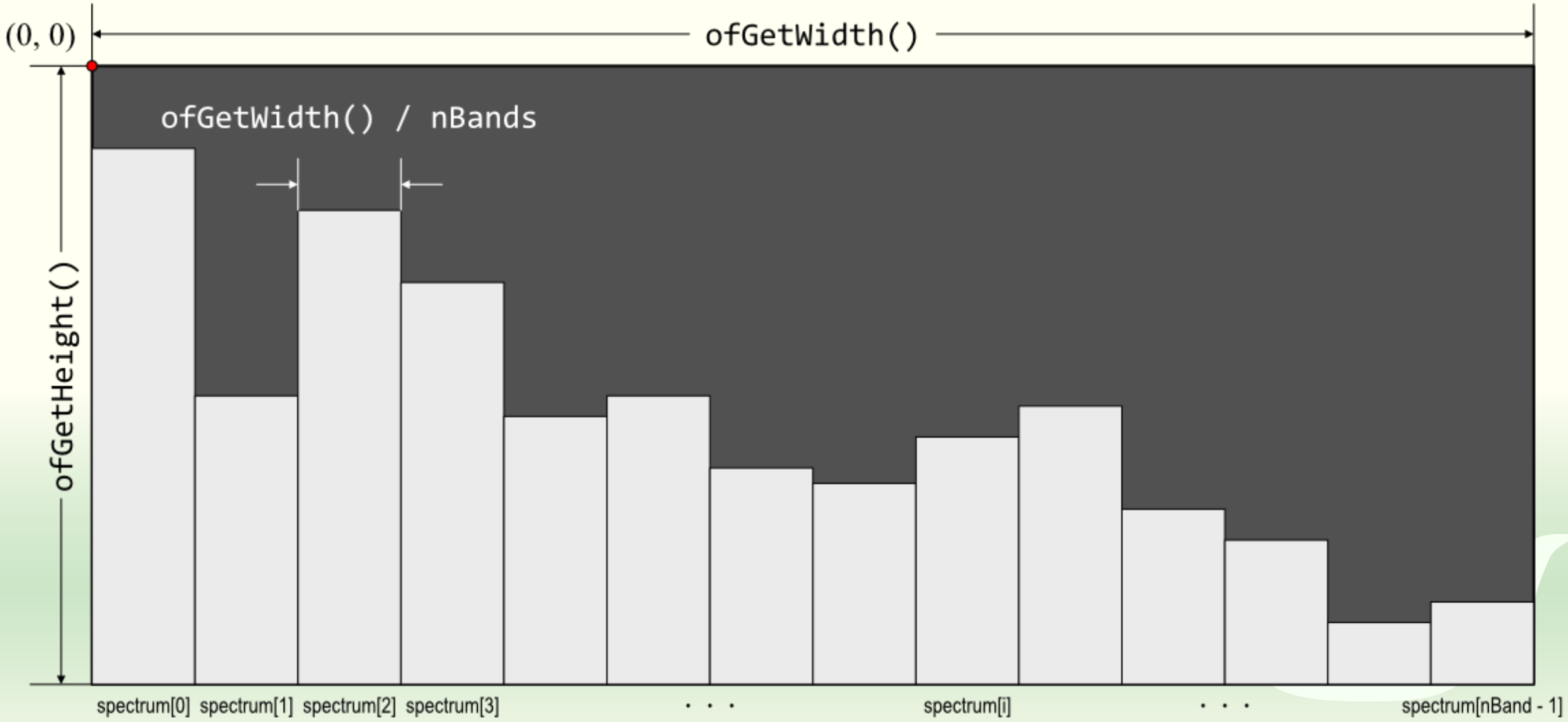
スペクトル分布の棒グラフを描きなさい

- sectrum の値を使ってウィンドウ上にスペクトル分布の棒グラフを円の下に描きなさい
- spectrum[i] には 0～1 の値が入っている
  - したがって棒グラフの高さを spectrum[i] * ofGetHeight() にすれば最大値がウィンドウの高さのグラフになる
  - ただし原点がウィンドウの上端にあるのでそのままではグラフの上下が反転してしまう
- 棒グラフの棒の数は nBands である
  - したがって 1 本の棒グラフの幅は ofGetWidth() / nBands になる

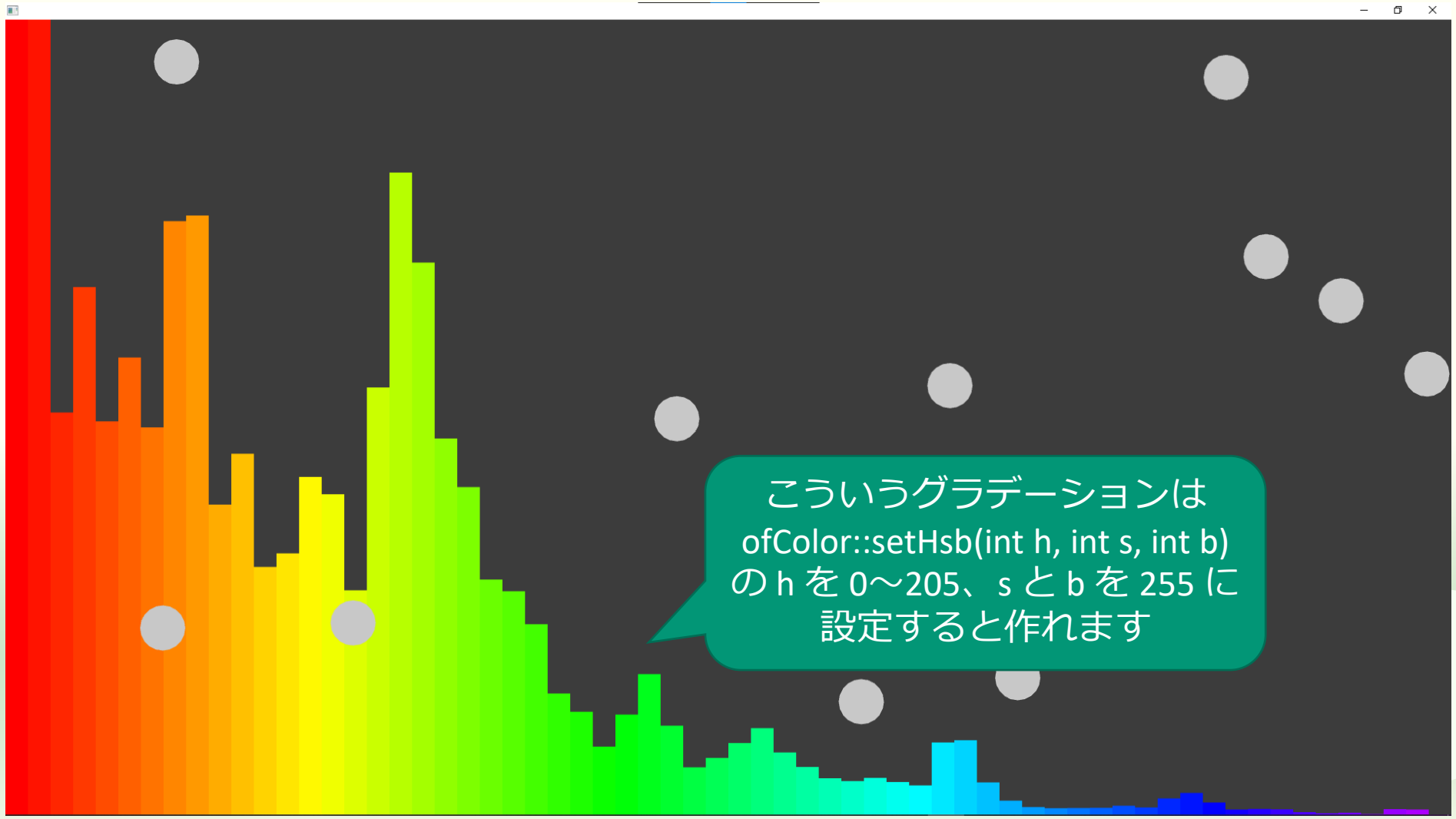
矩形の描画

- void ofDrawRectangle(const glm::vec2 &p, float w, float h)
  - p を左上に幅 w 高さ h の矩形を描く
- void ofDrawRectangle(float x1, float y1, float w, float h)
  - (x1, y1) を左上に幅 w 高さ h の矩形を描く

スペクトル分布の棒グラフのレイアウト



結果の例


こういうグラデーションは
ofColor::setHsb(int h, int s, int b)
の h を 0～205、s と b を 255 に
設定すると作れます

課題のアップロード

- 作成したプログラムの実行中のウィンドウを**５秒以内**で動画キャプチャして、**6-2.mp4** というファイル名で Moodle の第６回課題にアップロードしてください
  - 動画のキャプチャができないときはスクリーンショットを撮って 6-2.png というファイル名でアップロードしてください

音声の入力

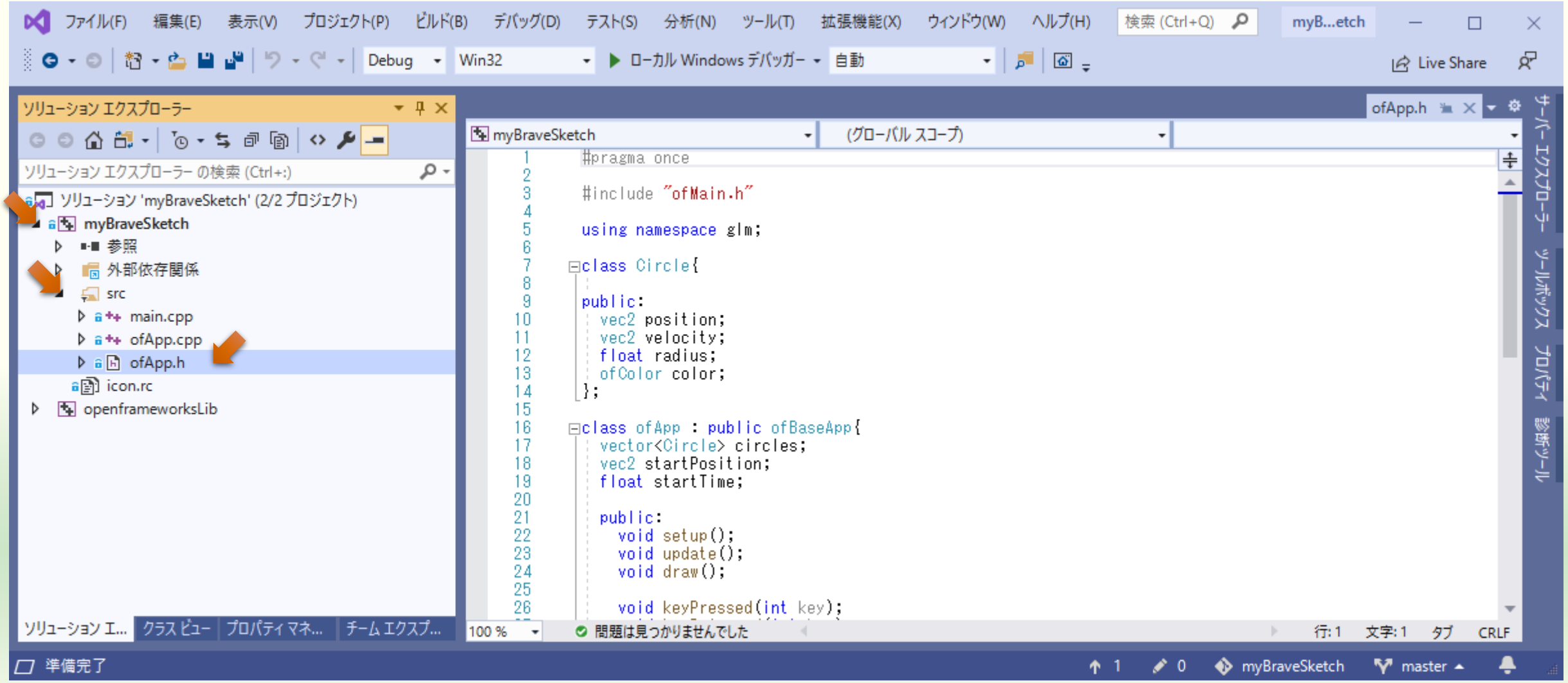
サウンドデータの取り扱い

「設定」でマイクの設定を確認する

「システム」→「サウンド」

「プライバシー」→「マイク」

ofApp.h を開く


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)
テスト(S)
分析(N)
ツール(T)
拡張機能(X)
ウィンドウ(W)
ヘルプ(H)
検索 (Ctrl+Q)
Debug
Win32
ローカル Windows デバッガー
自動
Live Share
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 'myBraveSketch' (2/2 プロジェクト)
myBraveSketch
参照
外部依存関係
src
main.cpp
ofApp.cpp
ofApp.h
icon.rc
openframeworksLib
ofApp.h
myBraveSketch
(グローバル スコープ)
#pragma once
#include "ofMain.h"
using namespace glm;
class Circle{
public:
vec2 position;
vec2 velocity;
float radius;
ofColor color;
};
class ofApp : public ofBaseApp{
vector<Circle> circles;
vec2 startPosition;
float startTime;
public:
void setup();
void update();
void draw();
void keyPressed(int key);
100 %
問題は見つかりませんでした
行: 1
文字: 1
タブ
CRLF
準備完了
1
0
myBraveSketch
master

ofApp クラスに音声入力のメンバ変数を追加する

```
（以上略）

class ofApp : public ofBaseApp{
  vector<Circle> circles;
  vec2 startPosition;
  float startTime;
  ofSoundPlayer sound;
  vector<ofSoundPlayer> effect;
  array<float, 64> spectrum{};
  ofSoundStream soundStream;
  array<float, 256> buffer{};
  float volume;

public:
  void setup();
  void update();
  void draw();
  void audioIn(ofSoundBuffer &input);

（以下略）
```

- ofSoundStream
  - リアルタイムに音声の入出力を行うためのクラス
- void audioIn(ofSoundBuffer &input);
  - input に音声データが、CD 品質なら 44,100Hz で取得される
  - update() や draw() は画面表示のタイミングで実行される（60Hz のディスプレイなら 60秒間に1回）
  - タイミングが合わないので音声は画面表示と並行して処理する

サウンド入力の設定

```
#include "ofApp.h"

const vec2 gravity = vec2(0.0f, 200.0f);
const int channels = 2;

//--------------------------------------------------------------
void ofApp::setup(){
  sound.load("sound.mp3");
  sound.setLoop(true);
   （途中略）

  ofSoundStreamSettings settings;
  settings.setInListener(this);
  settings.sampleRate = 44100;
  settings.numOutputChannels = 0;
  settings.numInputChannels = channels;
  settings.bufferSize = buffer.size() * channels;
  soundStream.setup(settings);
}

 （以下略）
```

2 で音声が入らないときは
1 で試してください

- settings.setInListener(this);
  - このオブジェクト (ofApp) の audioIn() を使って音声データを受け取る
- settings.sampleRate = 44100;
  - サンプリングレートを 44,100Hz（CD 品質）に設定する
- settings.numOutputChannels = 0;
  - このプログラムでは出力しないので出力チャネル数は 0 にする
- settings.numInputChannels = channels;
  - 入力チャネル数は 2（ステレオ）にする
- settings.bufferSize = buffer.size() * channels;
  - 取り出し用のメモリ (buffer) のチャネル数のバッファ（一時メモリ）を確保する

ofApp.cpp に追加するサウンド入力関数

```
//--------------------------------------------------------------
void ofApp::audioIn(ofSoundBuffer &input){

  // 二乗和
  float square = 0.0f;

  for (size_t i = 0; i < input.getNumFrames(); i += channels){

    // 左チャネルだけを保存する
    buffer[i / channels] = input[i];

    // 二乗和を求める
    square += input[i] * input[i];
  }

  // RMS (root mean square, 二乗平均平方根)
  volume = sqrt(square * channels / input.getNumFrames());
}
```

- void ofApp::audioIn(ofSoundBuffer &input){
  - バッファに入力音声データが満たされたら実行される
  - 入力音声データは input に格納されている
- input.getNumFrames()
  - input に格納されている入力音声データの数
    - 音声データはチャネルごとに順番に入っている
    - チャネル数が 2 なら、input[0] は左、input[1] は右、 input[2] は左、input[3] は右、…
  - buffer[i / channels] = input[i];
    - 左チャネルだけ使うので偶数番号のデータだけを buffer にコピーする
  - square += input[i] * input[i];
    - 入力データの二乗和を求めておく
- volume = sqrt(square * channels / <省略>);
  - volume にはバッファの中に格納されている音声データの**音量** (0～1) が入る

音量で円の大きさを制御する

```
#include "ofApp.h”

 （途中略）

//--------------------------------------------
void ofApp::draw(){
  const float cx{ ofGetWidth() * 0.5f };
  const float cy{ ofGetHeight() * 0.5f };
  const float cr{ (cx < cy ? cx : cy) * volume };
  ofSetColor(150, 150, 150);
  ofDrawCircle(cx, cy, cr);

   （途中略）
}

 （以下略）
```

- const float cx = ofGetWidth() * 0.5f;
  - cx はウィンドウの横方向の中心
- const float cy = ofGetHeight() * 0.5f;
  - cy はウィンドウの縦方向の中心
- cx < cy ? cx : cy
  - cx < cy なら cx、でなければ cy
    - cx と cy の小さい方に volume を掛ける
- **３項演算子**
  - 条件 ? 式１ : 式２
    - 条件が true なら式１の値を求め、そうでなければ式２の値を求める

課題６－３

声で円を追加する

一定以上の音量で円を追加するようにしなさい

- マイクに向かって一定以上の声で叫ぶと円が追加されるようにしてください
- ヒント：update() で volume と適当な閾値を比較して volume が閾値を超えたら円を生成します
  - 初期位置はウィンドウの中心にするといいでしょう
  - 初速度の大きさを声の大きさで決定するのも面白いと思います
  - 初速度の方向を疑似乱数で決定するといろんな方向に移動します
    - float ofRandom(float max), float ofRandom(float v0, float v1)
      - それぞれ 0～max、v0～v1 の疑似乱数を返す

課題のアップロード

- 作成したプログラムの実行中のウィンドウを**５秒以内**で動画キャプチャして、**6-3.mp4** というファイル名で Moodle の第６回課題にアップロードしてください
  - 動画のキャプチャができないときはスクリーンショットを撮って 6-3.png というファイル名でアップロードしてください
- ソースプログラム **ofApp.h** と **ofApp.cpp** を Moodle の第６回課題にアップロードしてください