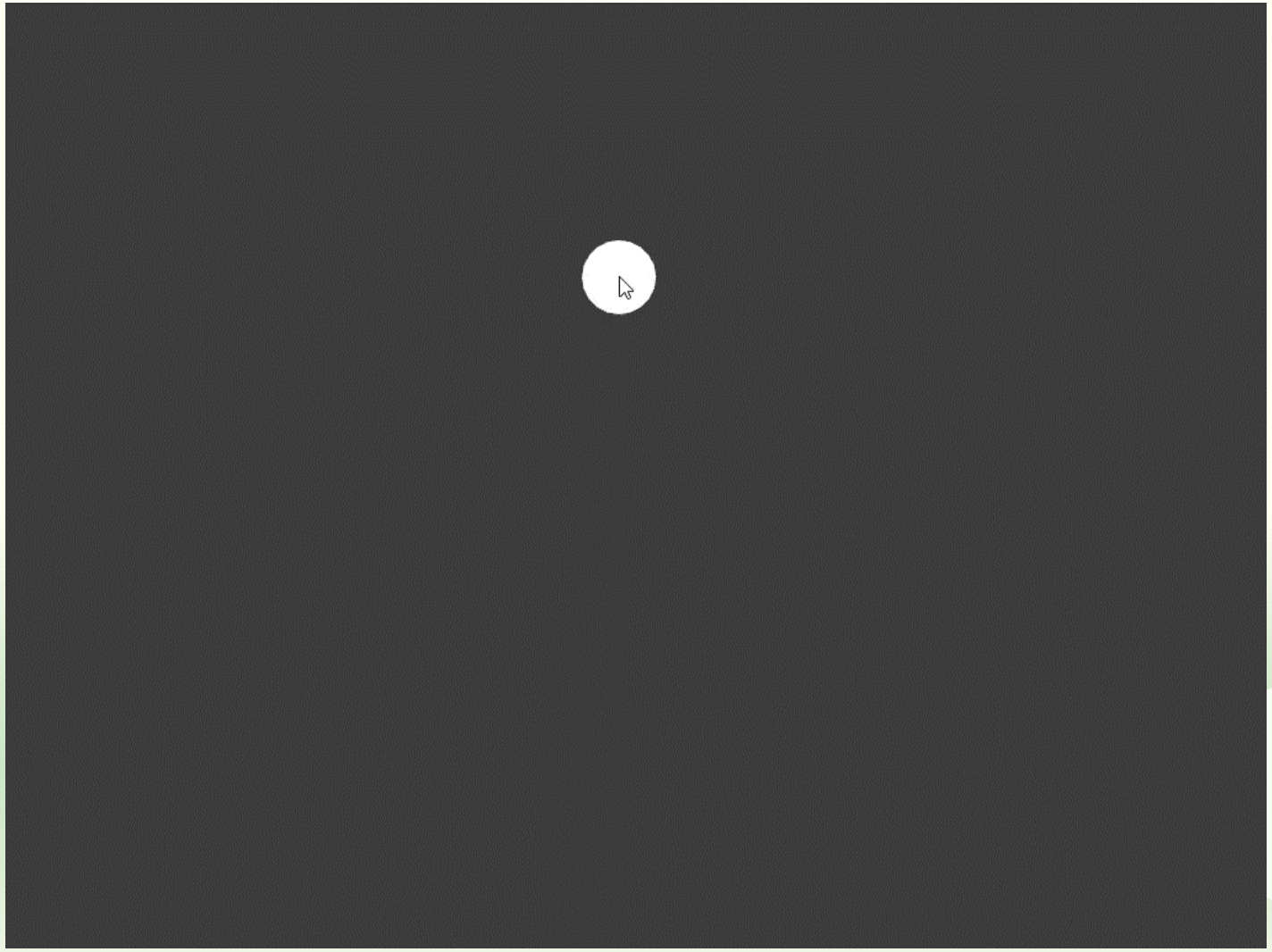
メディアプログラミング演習

第2回

本日は円を放り投げるだけのアプリの作成

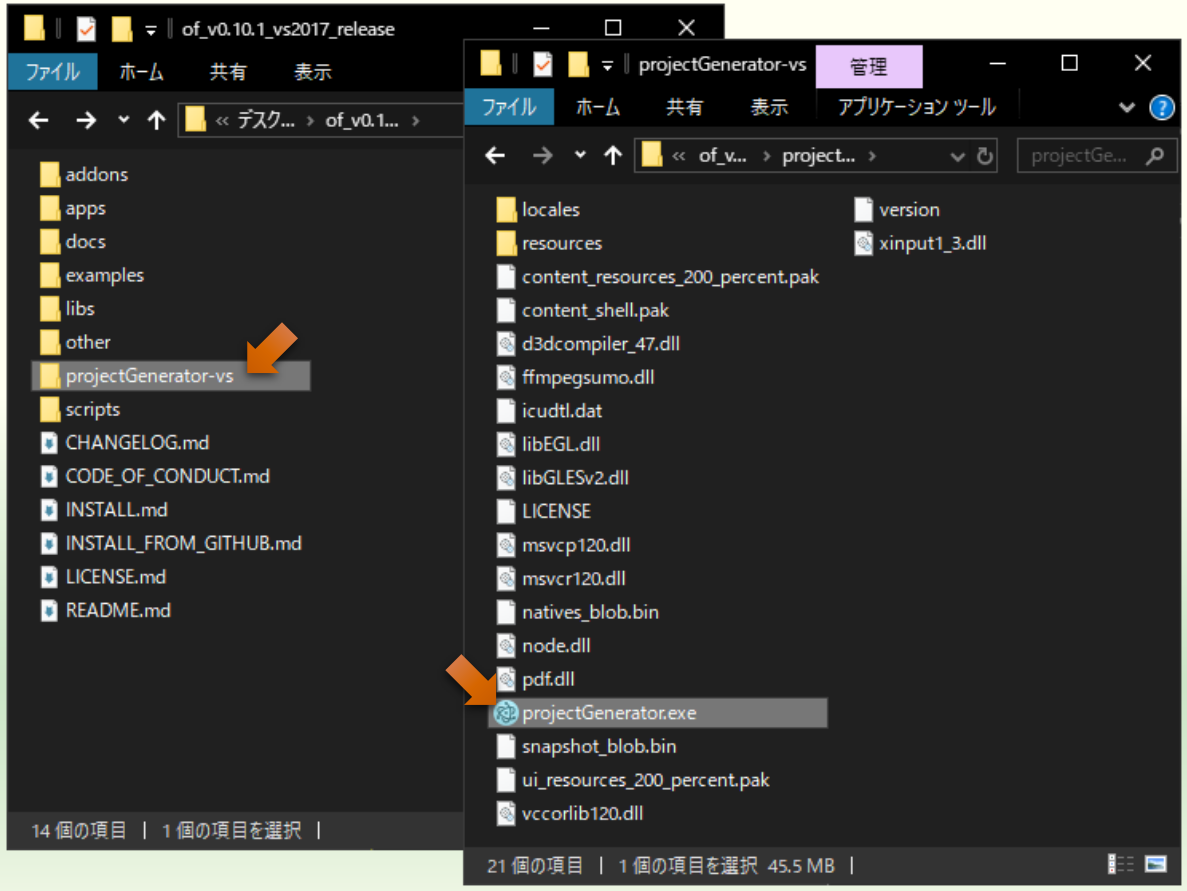


準備

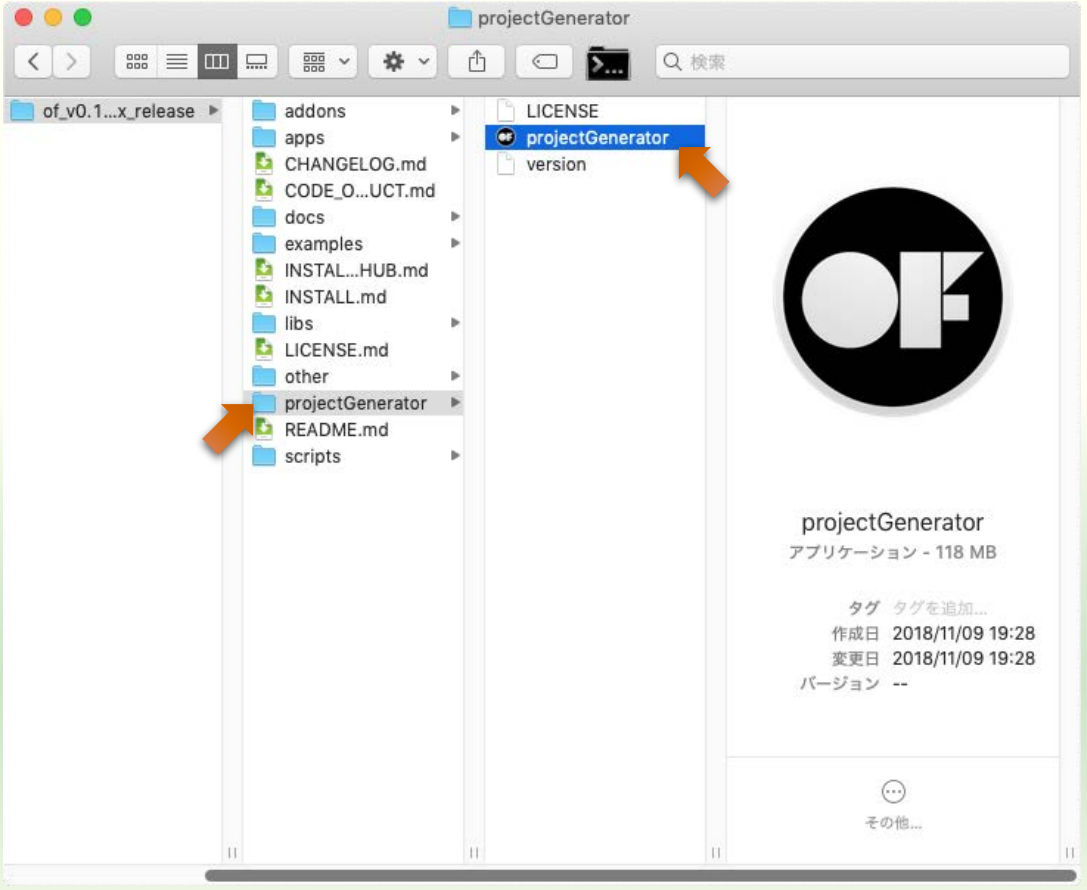
プロジェクトの作成

projectGenerator を起動する

windows 版のパッケージ



macOS 版のパッケージ

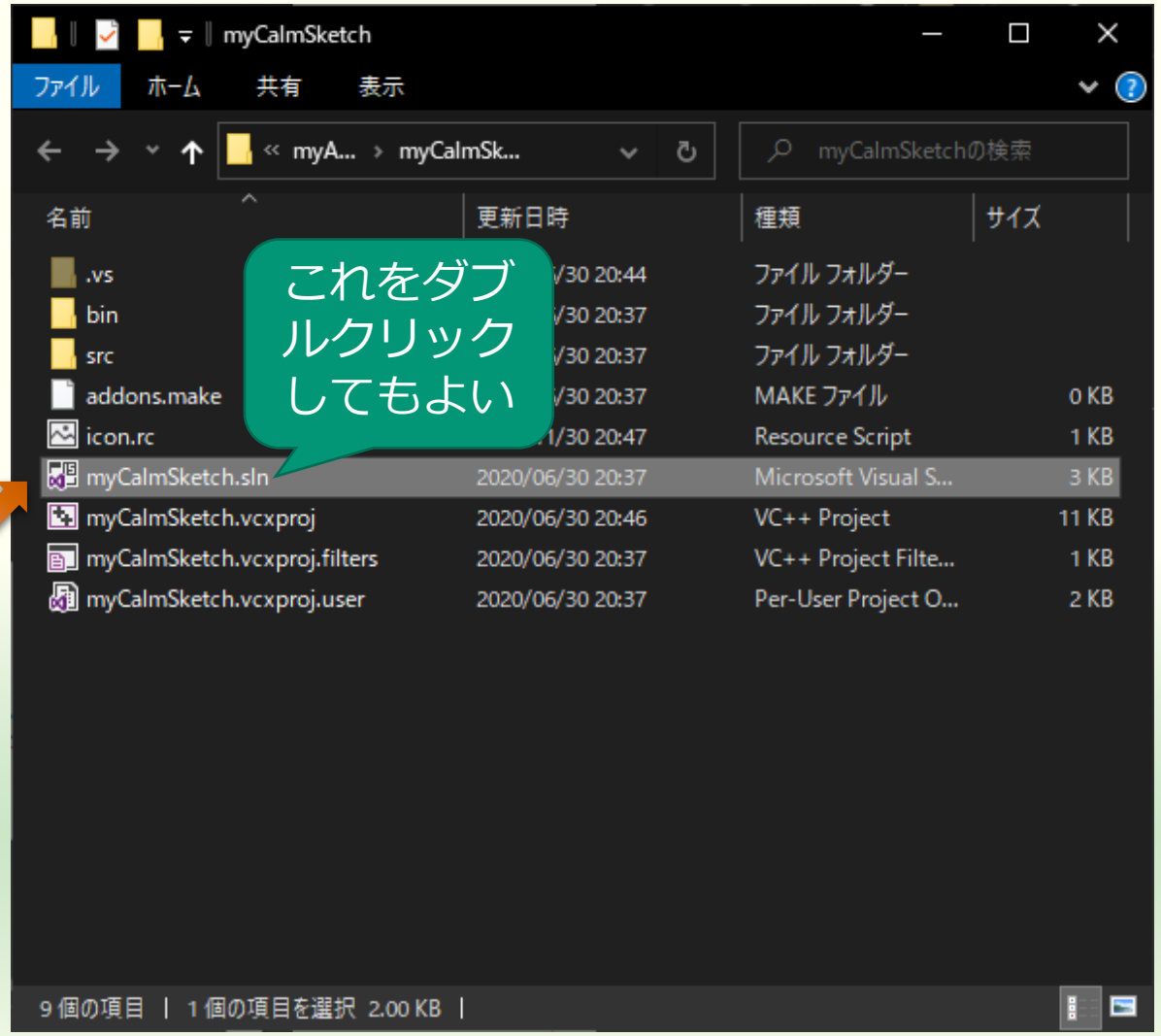


空のプロジェクトの作成

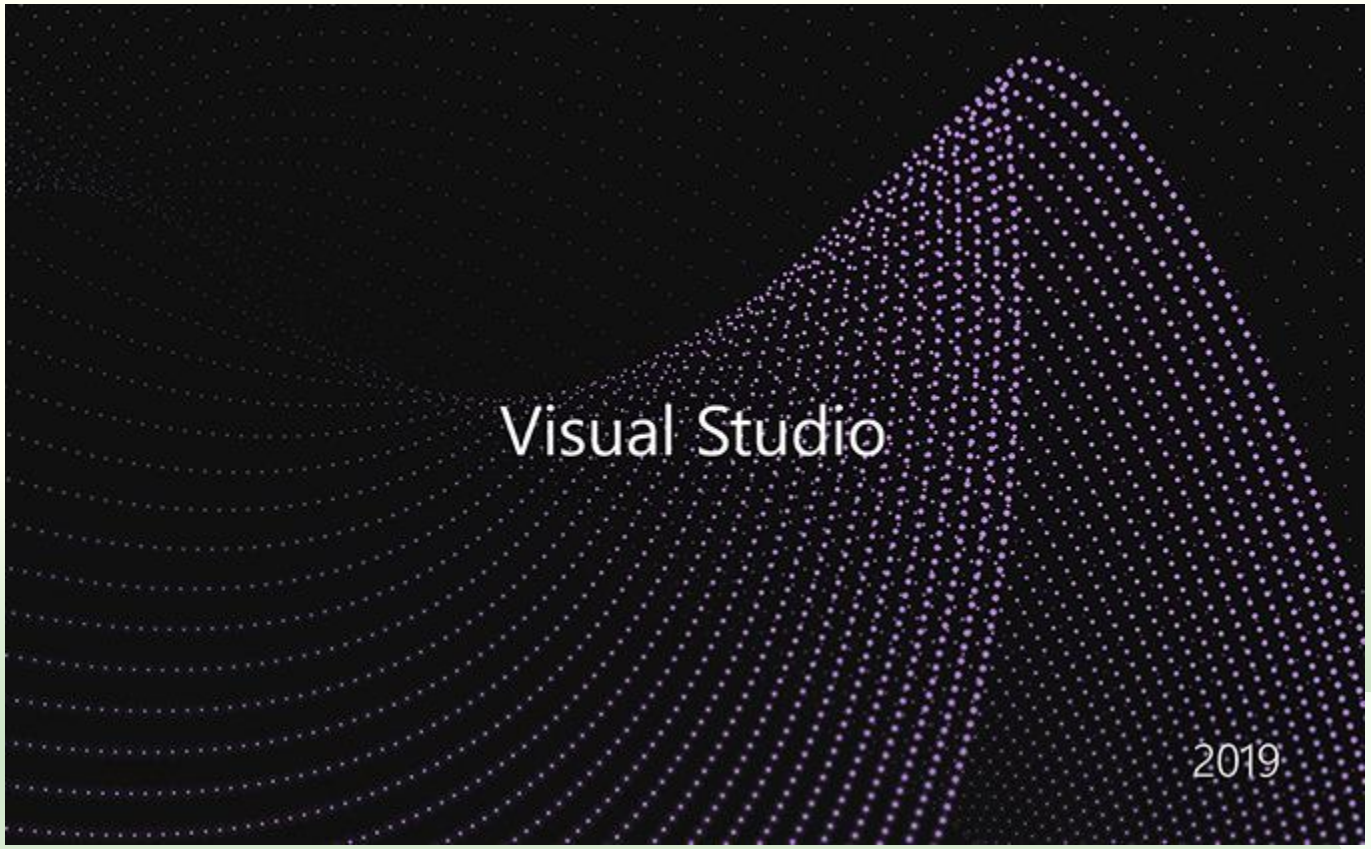
- Project name:
  - 作成するプロジェクト（プログラム）の名前
- Project path:
  - 作成するプロジェクトのファイルを置く場所
  - openFrameworks のパッケージを展開した場所の中の apps¥myApps

プロジェクトの作成成功

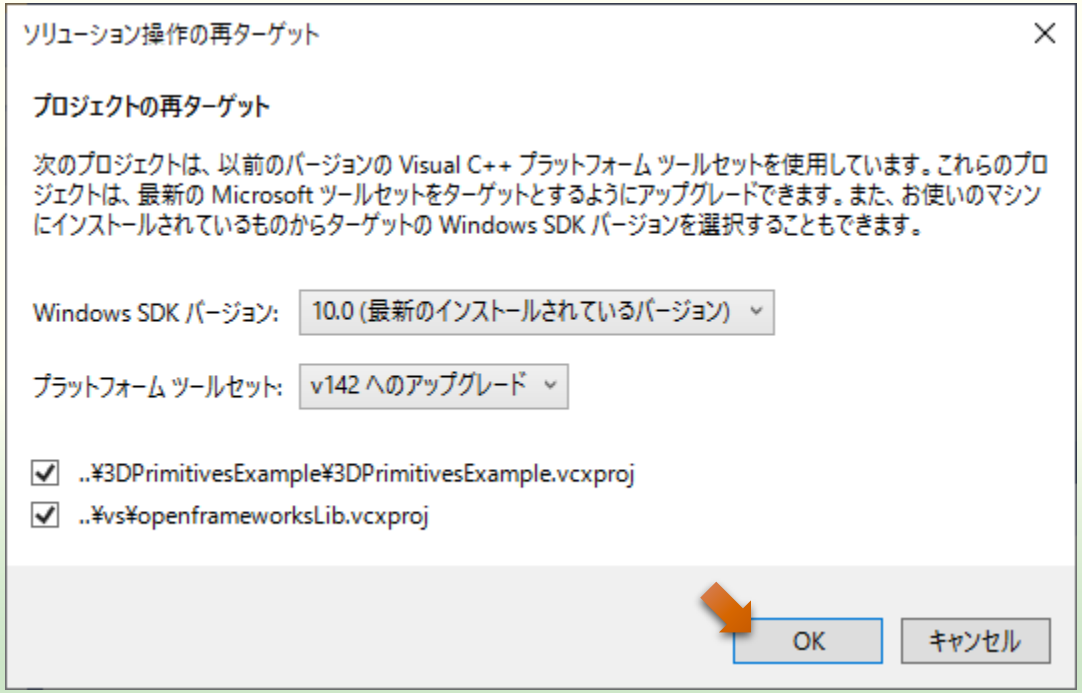
クリックすると開く



Visual Studio 2019 が起動する

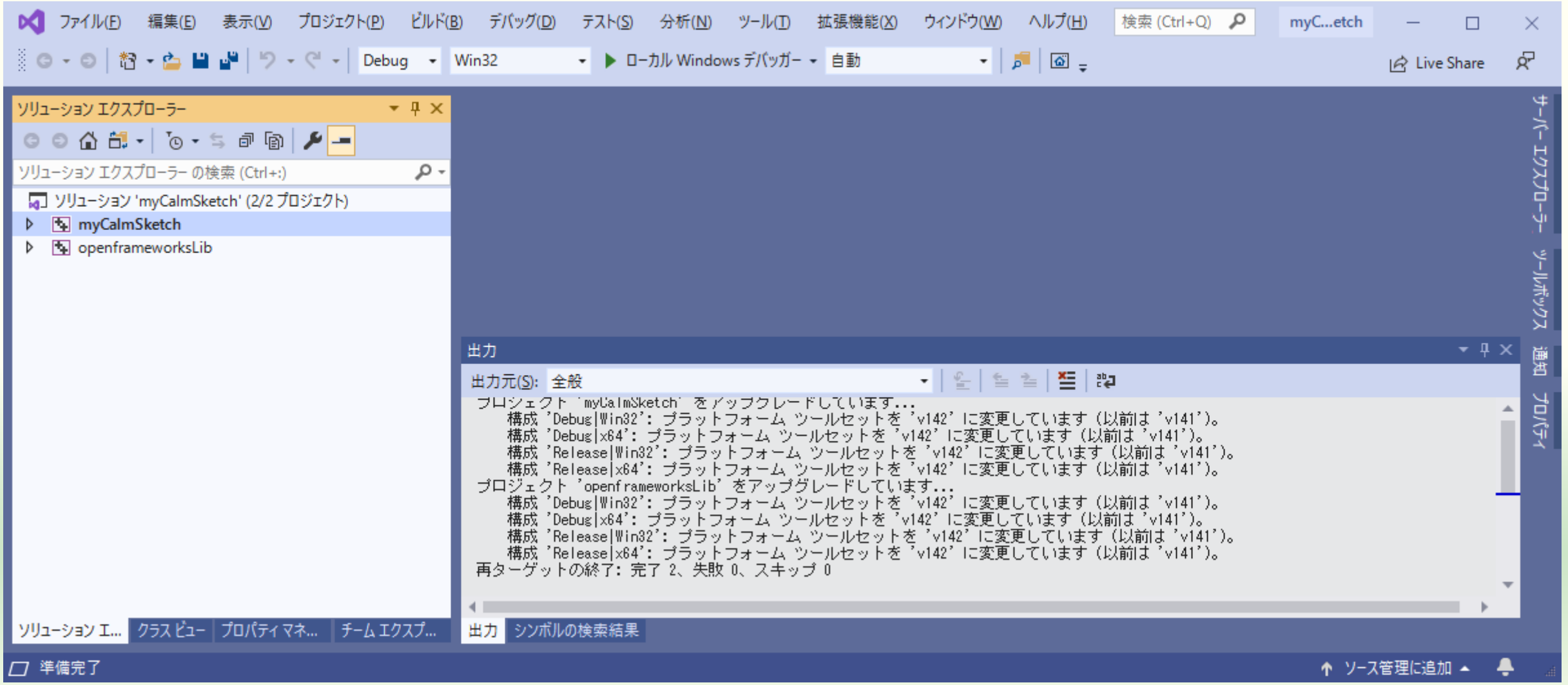


ソリューションの再ターゲット



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

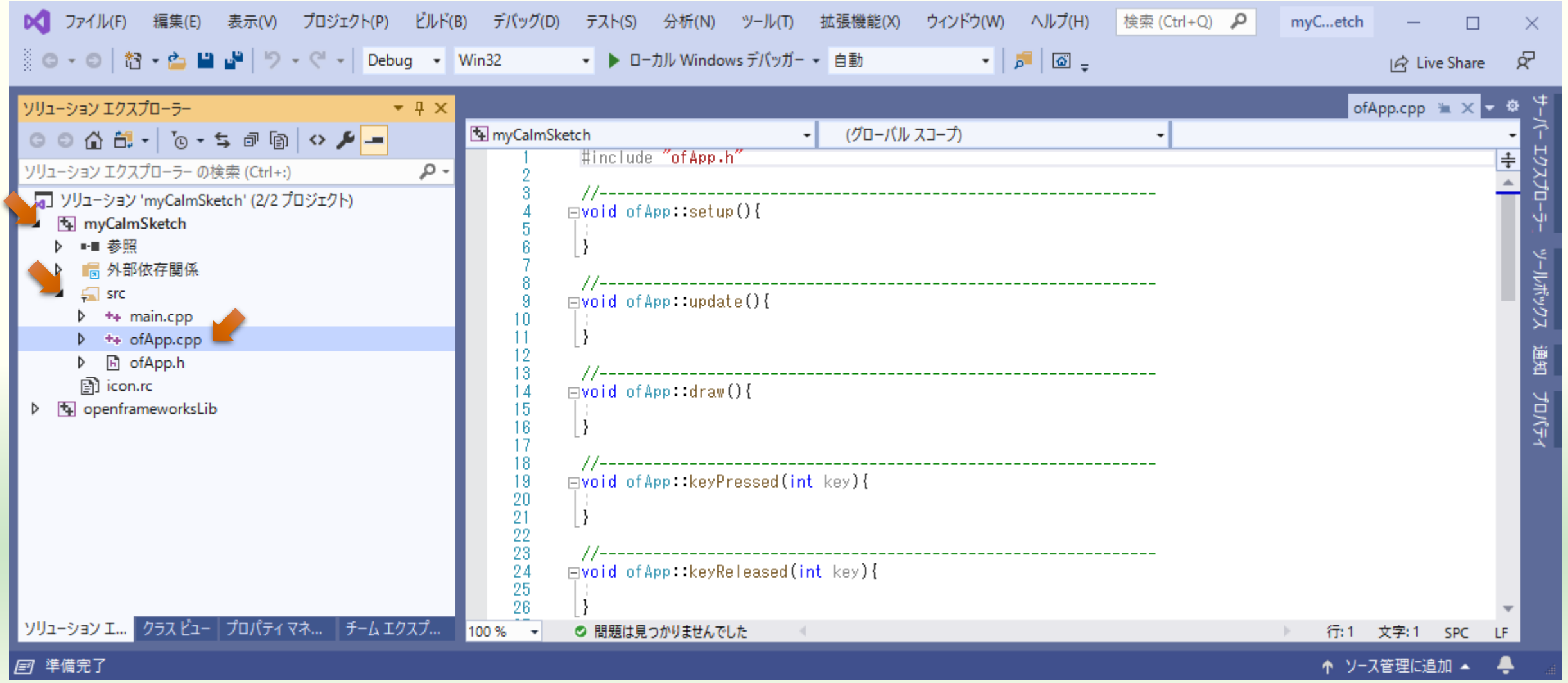
Visual Studio 起動



変数・定数とデータ型

データを記憶する

ofApp.cpp を開く



ofDrawCircle() で円を描く

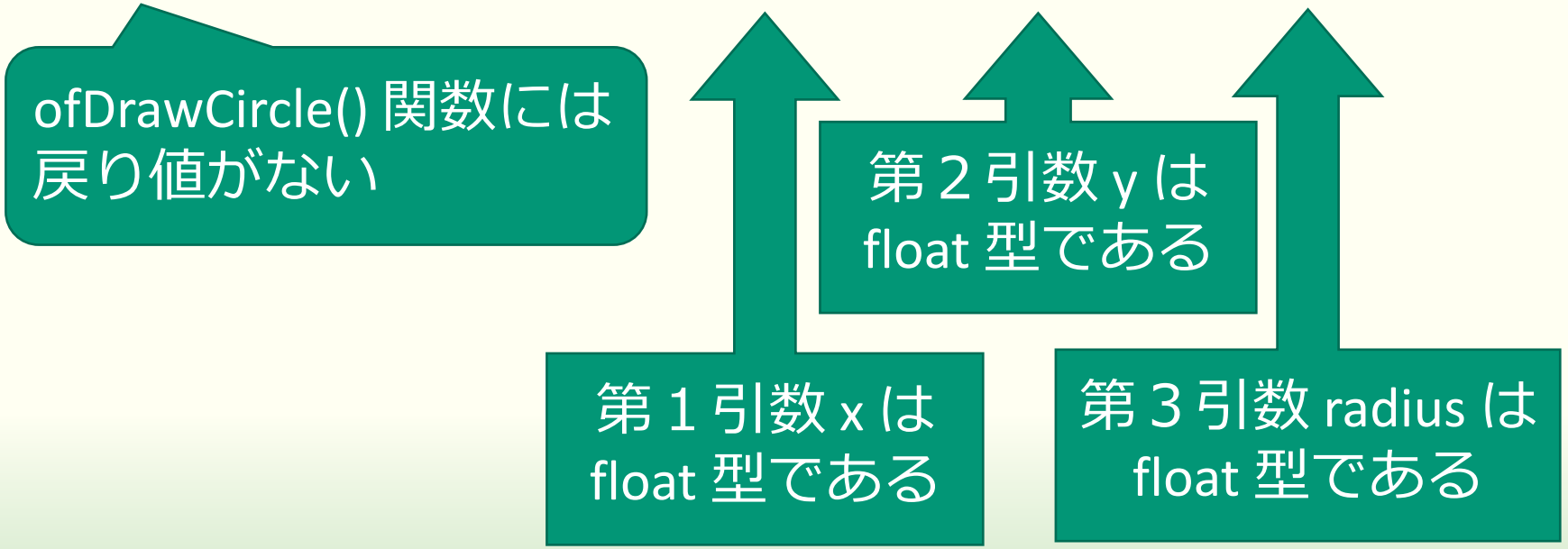
```
//--------------------------------------------
void ofApp::setup(){

}

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

}

//--------------------------------------------
void ofApp::draw(){
  ofDrawCircle(0.0f, 0.0f, 30.0f);
}

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

}
```

中心位置　半径

- (0, 0) を中心に半径 30 の円を ofDrawCircle() 関数で描く

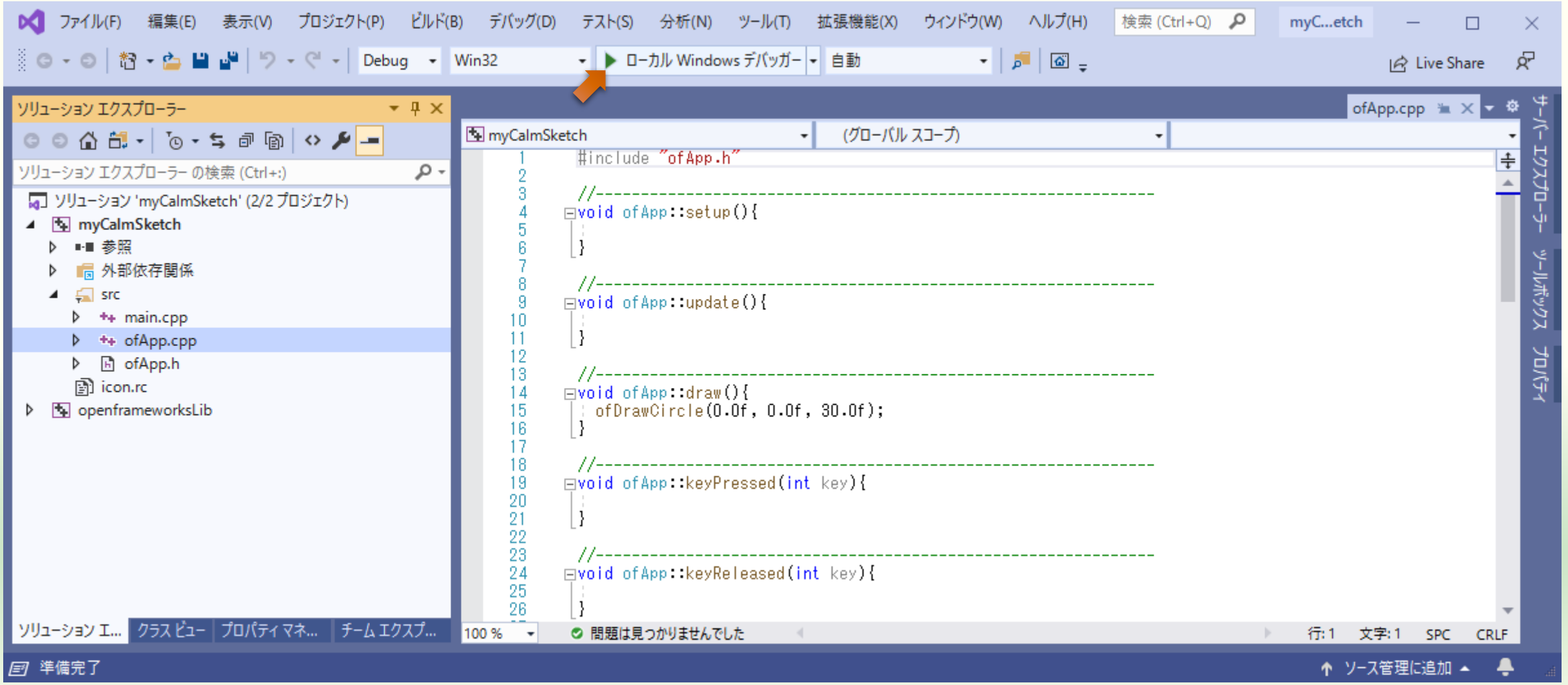
ofDrawCircle() の宣言

- **void ofDrawCircle(float x, float y, float radius)**

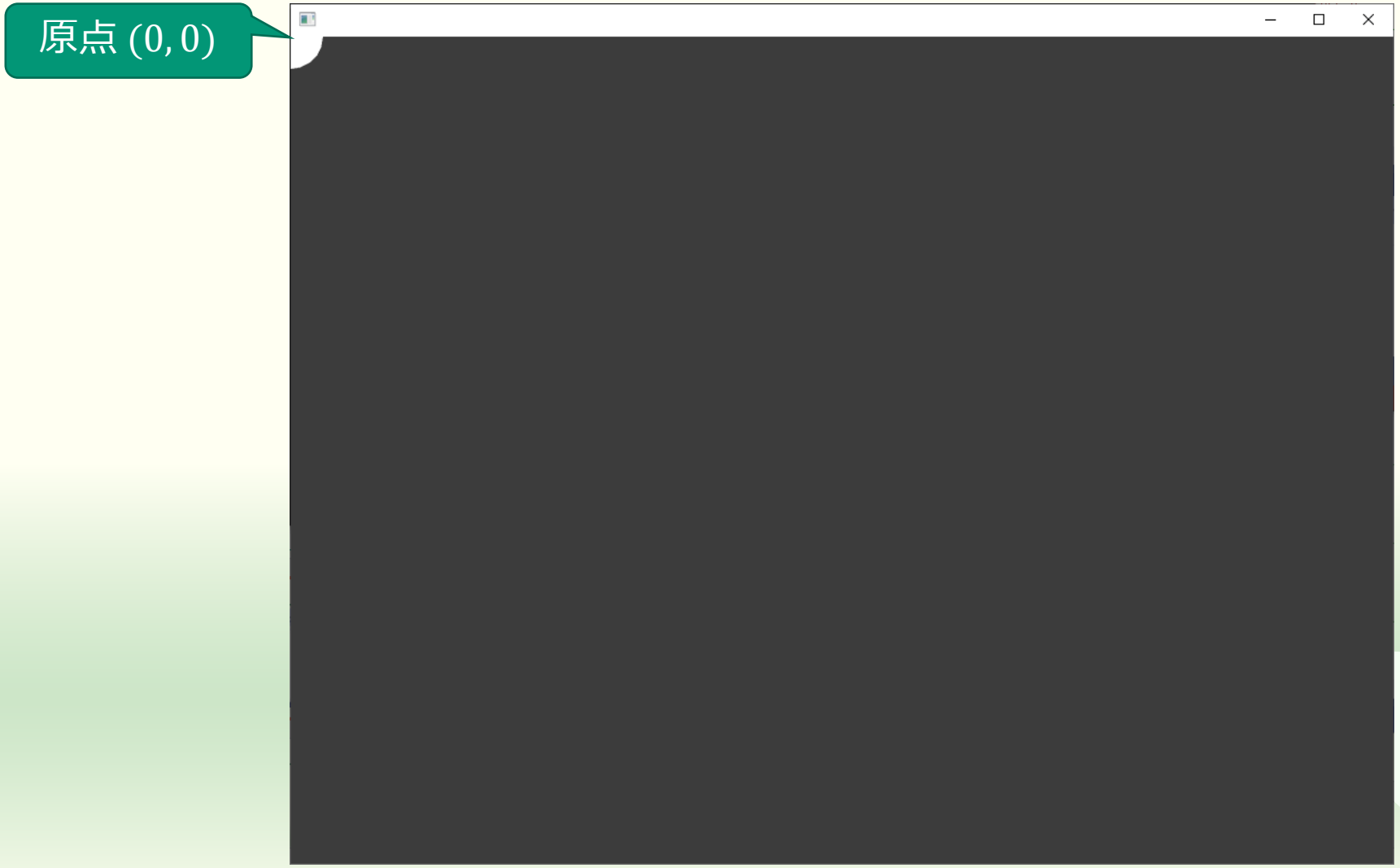


- 仮引数 x, y, radius が何を表すのかは示されていない
  - マニュアルやコメントで説明されている（はず）
  - 宣言では引数名 x, y, radius が省略されている場合がある

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)
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 'myCalmSketch' (2/2 プロジェクト)
myCalmSketch
参照
外部依存関係
src
main.cpp
ofApp.cpp
ofApp.h
icon.rc
openframeworksLib
ofApp.cpp
myCalmSketch
(グローバル スコープ)
#include "ofApp.h"
//--
void ofApp::setup(){
}
//--
void ofApp::update(){
}
//--
void ofApp::draw(){
ofDrawCircle(0.0f, 0.0f, 30.0f);
}
//--
void ofApp::keyPressed(int key){
}
//--
void ofApp::keyReleased(int key){
}
100 %
問題は見つかりませんでした
行: 1
文字: 1
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
準備完了
ソース管理に追加

左上の原点を中心に円が描かれる



定数

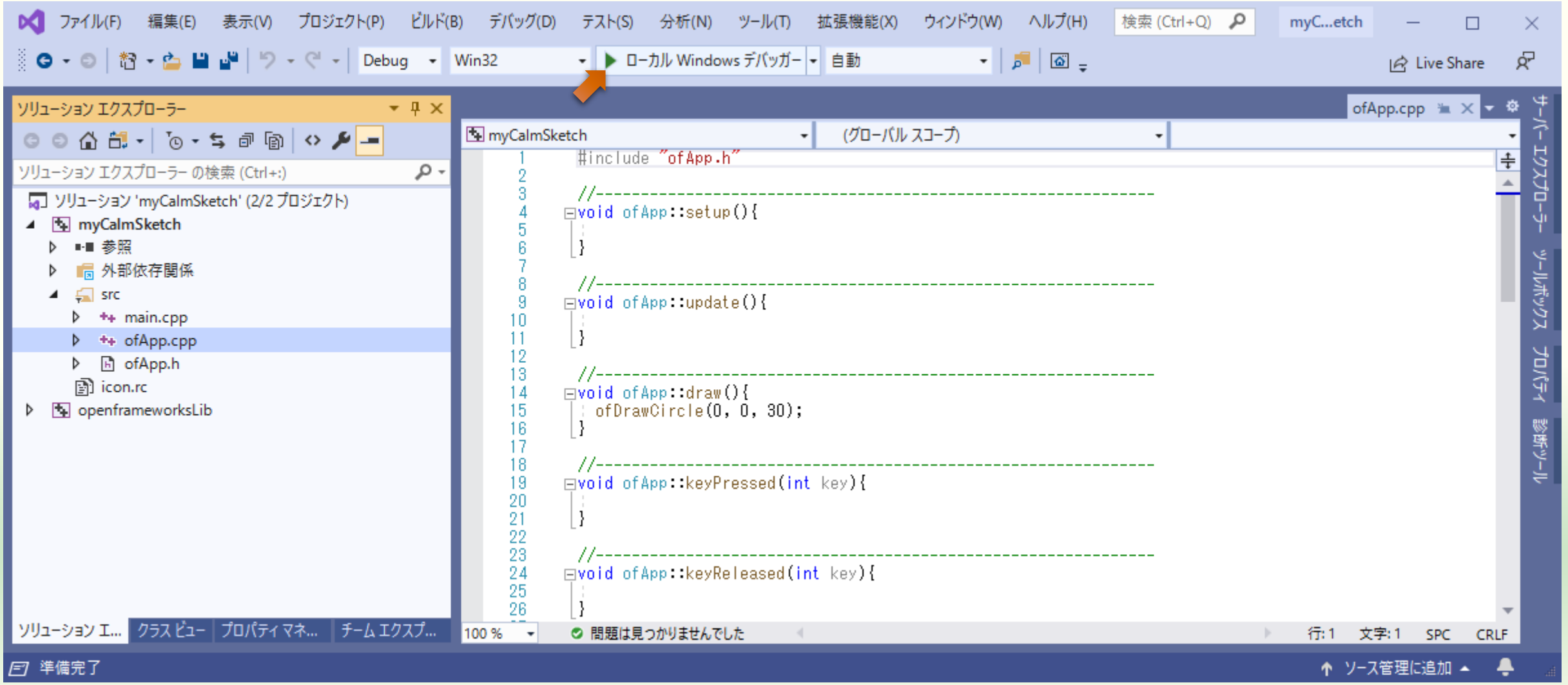
- 100
  - int 型の定数 (10 進数)
- 100.0
  - double 型の定数
- 100.0f
  - float 型の定数
- 1.0e2f
  - float 型の定数
    - $1.0 \times 10^2 = 100$
- 0100
  - int 型の定数 (8 進数)
    - 10 進数の 64
- 0b100
  - int 型の定数 (2進数)
    - 10進数の 4
- 0x100
  - int 型の定数 (16進数)
    - 10進数の 256

引数を整数にしてみる

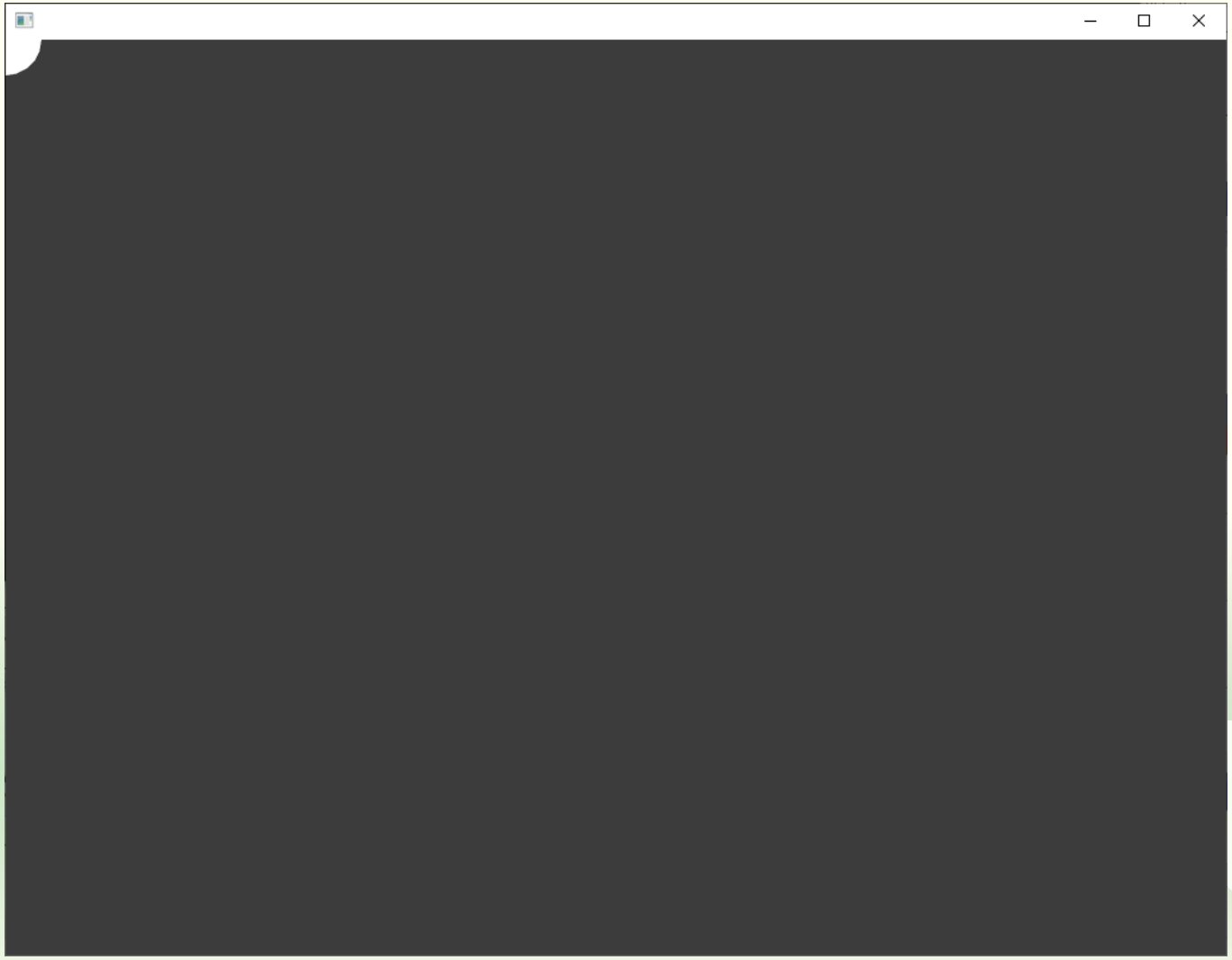
```
//--------------------------------------------
void ofApp::setup(){

}

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

}

//--------------------------------------------
void ofApp::draw(){
  ofDrawCircle(0, 0, 30);
}

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

}
```

- int から float への型変換が自動的に行われる
  - **暗黙の型変換**

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 'myCalmSketch' (2/2 プロジェクト)
myCalmSketch
参照
外部依存関係
src
main.cpp
ofApp.cpp
ofApp.h
icon.rc
openframeworksLib
ofApp.cpp
myCalmSketch
(グローバル スコープ)
#include "ofApp.h"
//--
void ofApp::setup(){
}
//--
void ofApp::update(){
}
//--
void ofApp::draw(){
ofDrawCircle(0, 0, 30);
}
//--
void ofApp::keyPressed(int key){
}
//--
void ofApp::keyReleased(int key){
}
100 %
問題は見つかりませんでした
行: 1 文字: 1 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
ソース管理に追加

特に変わらない



引数を変数で与える

```
//--------------------------------------------
void ofApp::setup(){

}

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

}

//--------------------------------------------
void ofApp::draw(){
  float x, y;
  float radius;

  x = 0.0f;
  y = 0.0f;
  radius = 30.0f;

  ofDrawCircle(x, y, radius);
}
```

変数宣言

代入

- 変数は**宣言**すれば使えるようになる
  - データの記憶場所がメモリ上に確保される
  - コンマ (,) で区切って複数の変数を宣言できる
- **代入**により変数に値を格納する

変数宣言に const を付けると代入できない

```
//--------------------------------------------
void ofApp::setup(){

}

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

}

//--------------------------------------------
void ofApp::draw(){
  float x, y;
  const float radius;

  x = 0.0f;
  y = 0.0f;
  radius = 30.0f;

  ofDrawCircle(x, y, radius);
}
```

エラー

- **const** を付けて宣言した変数は値を**変更できない**
  - 値を変更（代入）しようとするとエラーになる
  - したがって宣言の時にも値を設定（初期化）しないとエラーになる

初期化は変数宣言のときに値を設定する

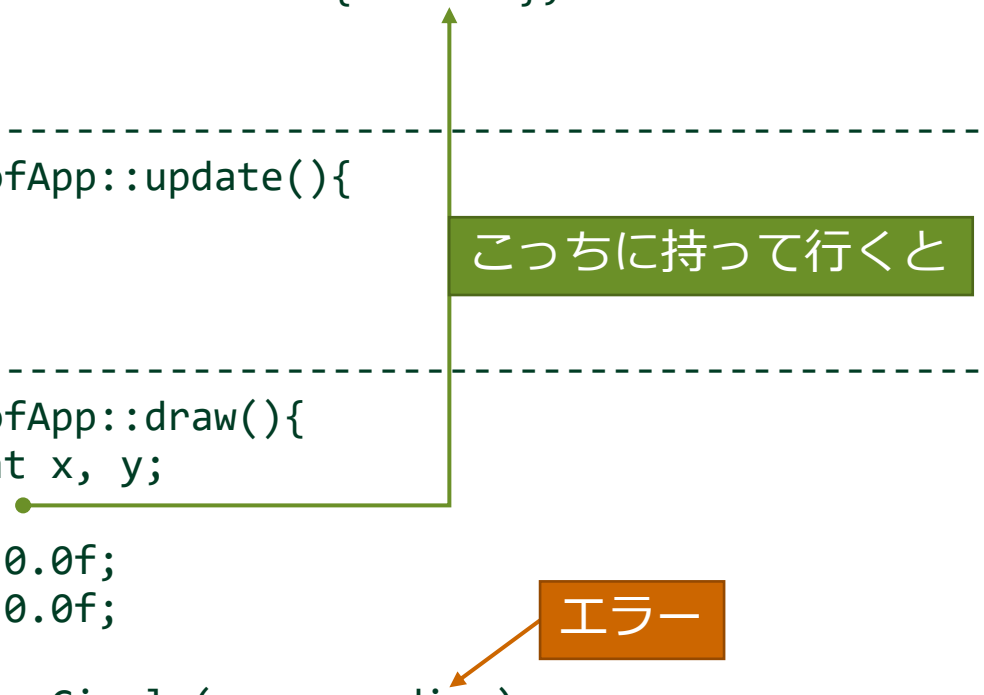
```
//--------------------------------------------
void ofApp::setup(){

}

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

}

//--------------------------------------------
void ofApp::draw(){
  float x, y;
  const float radius{ 30.0f };

  x = 0.0f;
  y = 0.0f;

  ofDrawCircle(x, y, radius);
}
```

初期化

- **初期化**は変数宣言の時に確保されたメモリに値を設定する
  - const float radius{ 30.0f };
    - radius を 30.0f で初期化する
  - const float radius = 30.0f; と書くこともできる
    - これは**代入と紛らわしい**ので勧めない

変数は宣言された { } 内でしか使えない

```
//--------------------------------------------
void ofApp::setup(){
  const float radius{ 30.0f };
}

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

}

//--------------------------------------------
void ofApp::draw(){
  float x, y;

  x = 0.0f;
  y = 0.0f;

  ofDrawCircle(x, y, radius);
}
```



- { } 内で宣言された変数は同じ {} 内でしか使えない
  - **局所変数（ローカル変数）**
- 変数が使える範囲のことを変数の**スコープ**という

変数宣言を関数の外に置くと共有できる

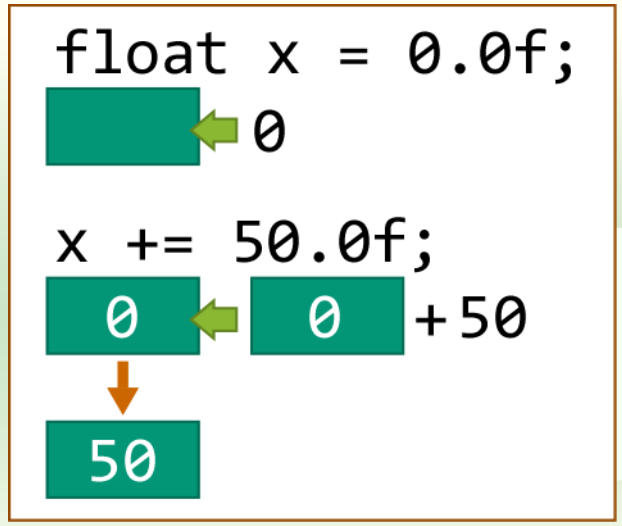
```
const float radius{ 30.0f };

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

}

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

}

//--------------------------------------------
void ofApp::draw(){
  float x, y;

  x = 0.0f;
  y = 0.0f;

  ofDrawCircle(x, y, radius);
}
```

- 変数宣言を関数の外に置くと**後に続く関数**で共通に使用できる
- **const** float radius{ 100.0f };
  - 関数の外で宣言した変数でも const を付けると ofApp.cpp 内以外からは参照できない
    - 大域変数（後述）にならない

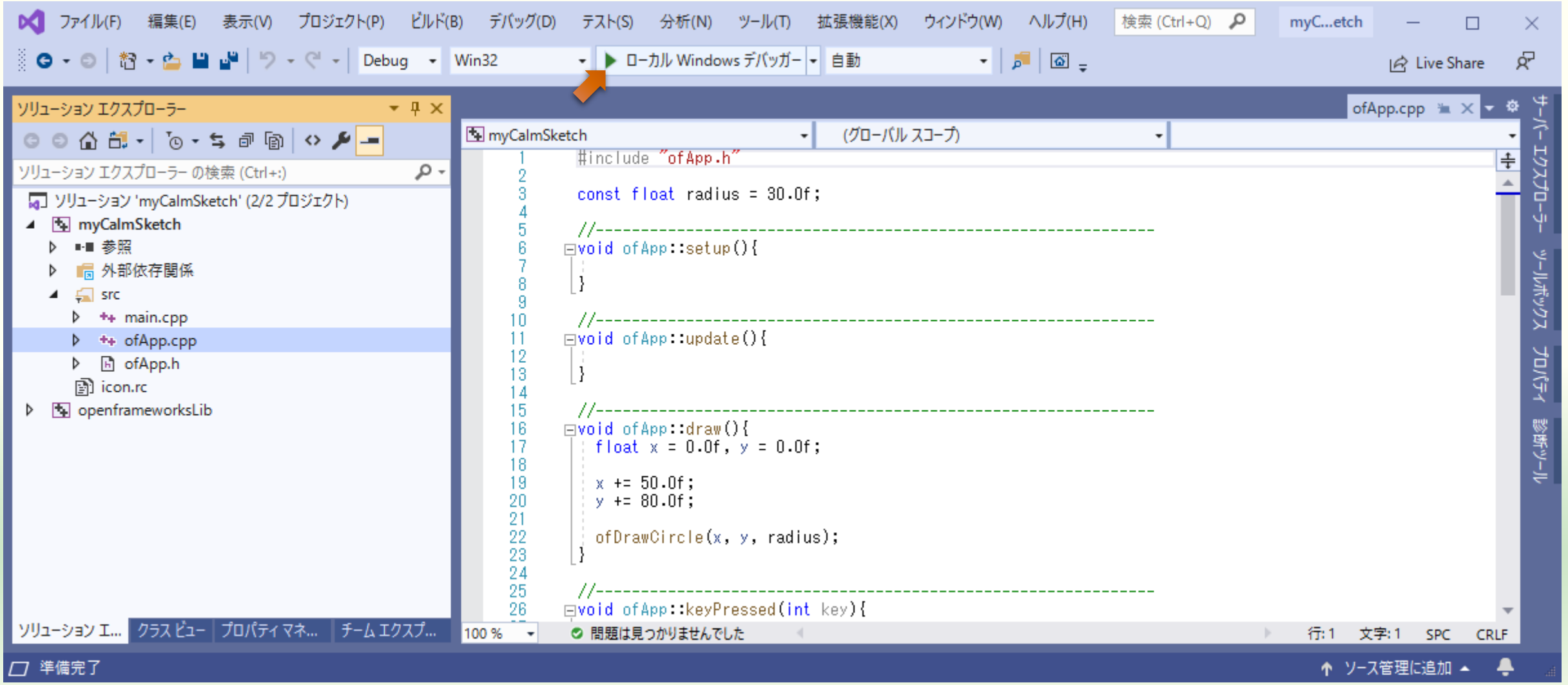
const を付けない変数の初期化は初期値

```
const float radius{ 30.0f };

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

}

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

}

//--------------------------------------------
void ofApp::draw(){
  float x{ 0.0f }, y{ 0.0f };

  x += 50.0f;
  y += 80.0f;

  ofDrawCircle(x, y, radius);
}
```

- x, y の初期値を 0 にする
- **x += 50.0f;**
  - x に 50 を加える
  - x = x + 50.0f; と同じ



ビルドと実行

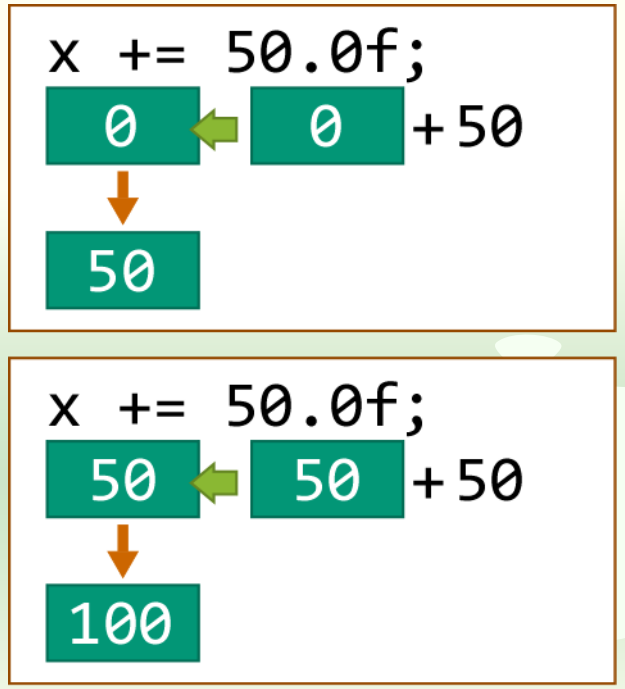

ファイル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(B) デバッグ(D) テスト(S) 分析(N) ツール(T) 拡張機能(X) ウィンドウ(W) ヘルプ(H)
検索 (Ctrl+Q)
Debug
Win32
ローカル Windows デバッガー
自動
Live Share
ソリューション エクスプローラー
ソリューション エクスプローラー の検索 (Ctrl+;)
ソリューション 'myCalmSketch' (2/2 プロジェクト)
myCalmSketch
参照
外部依存関係
src
main.cpp
ofApp.cpp
ofApp.h
icon.rc
openframeworksLib
ofApp.cpp
myCalmSketch
(グローバル スコープ)
#include "ofApp.h"
const float radius = 30.0f;
//--
void ofApp::setup(){
}
//--
void ofApp::update(){
}
//--
void ofApp::draw(){
float x = 0.0f, y = 0.0f;
x += 50.0f;
y += 80.0f;
ofDrawCircle(x, y, radius);
}
//--
void ofApp::keyPressed(int key){
100 %
問題は見つかりませんでした
行: 1 文字: 1 SPC CRLF
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
ソース管理に追加

中心が (50, 80) に移動する

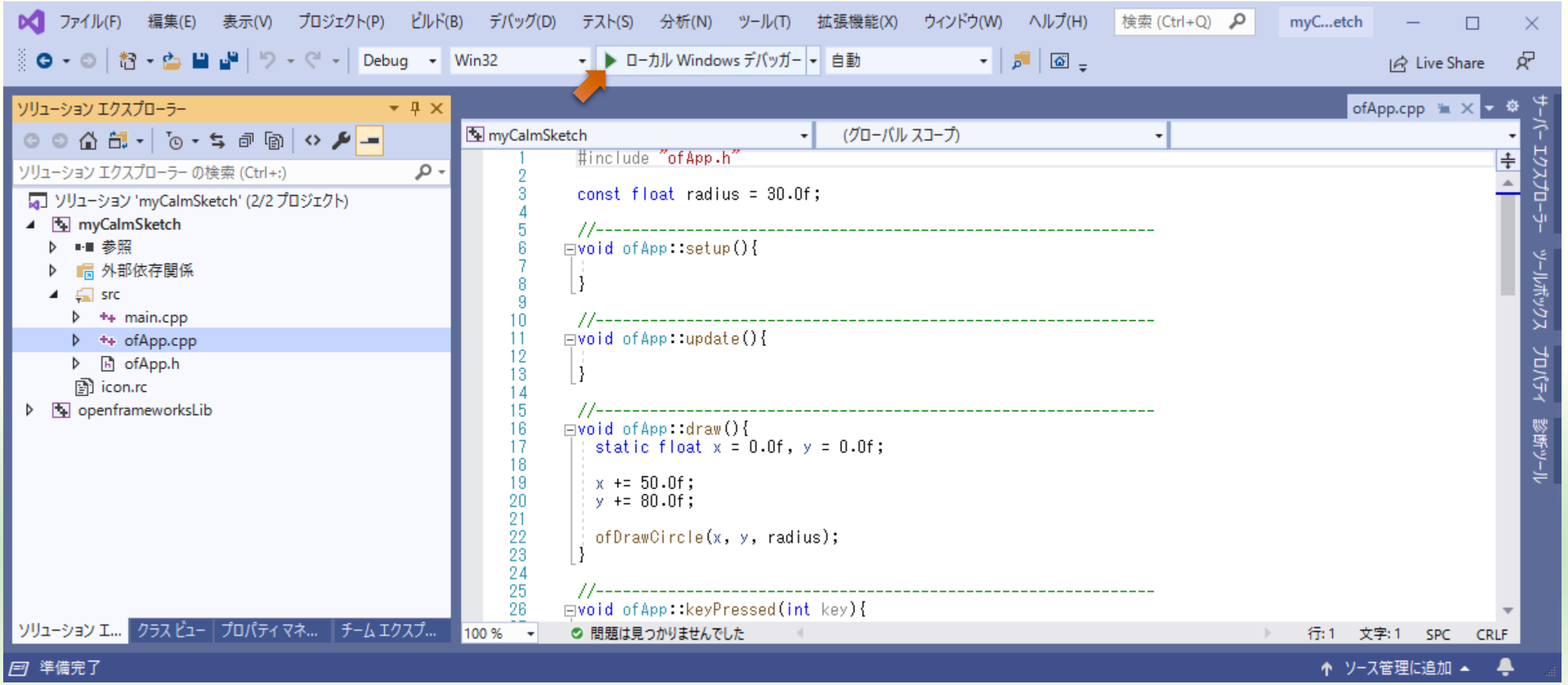
変数宣言に static を付けると値を保持する

```
const float radius{ 30.0f };

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

}

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

}

//--------------------------------------------
void ofApp::draw(){
  static float x{ 0.0f }, y{ 0.0f };

  x += 50.0f;
  y += 80.0f;

  ofDrawCircle(x, y, radius);
}
```

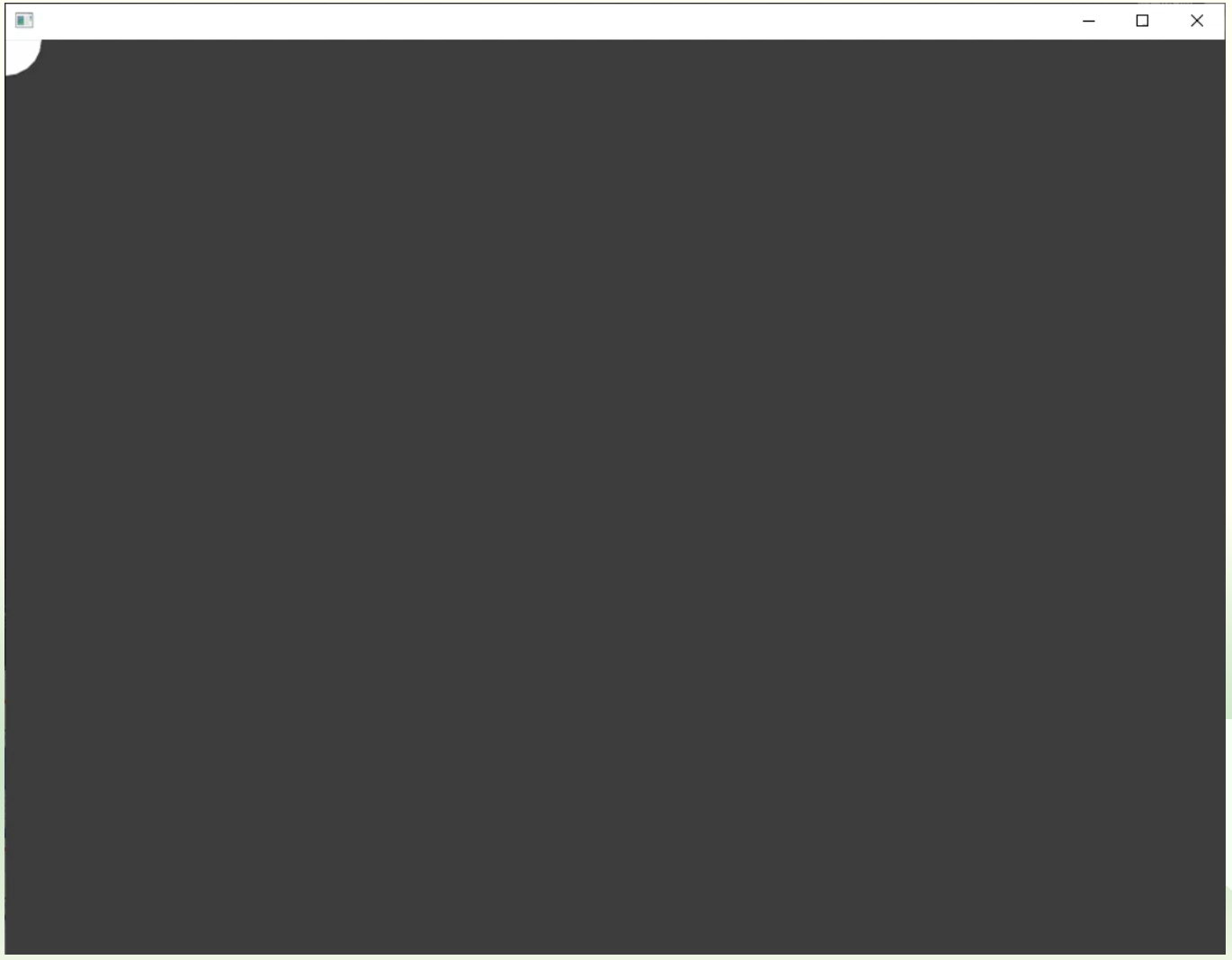
- draw() は繰り返し何度も実行されている
  - x += 50.0f も繰り返す



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 'myCalmSketch' (2/2 プロジェクト)
myCalmSketch
参照
外部依存関係
src
main.cpp
ofApp.cpp
ofApp.h
icon.rc
openframeworksLib
ofApp.cpp
myCalmSketch
(グローバル スコープ)
#include "ofApp.h"
const float radius = 30.0f;
//--
void ofApp::setup(){
}
//--
void ofApp::update(){
}
//--
void ofApp::draw(){
static float x = 0.0f, y = 0.0f;
x += 50.0f;
y += 80.0f;
ofDrawCircle(x, y, radius);
}
//--
void ofApp::keyPressed(int key){
100 %
問題は見つかりませんでした
行: 1 文字: 1 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
ソース管理に追加

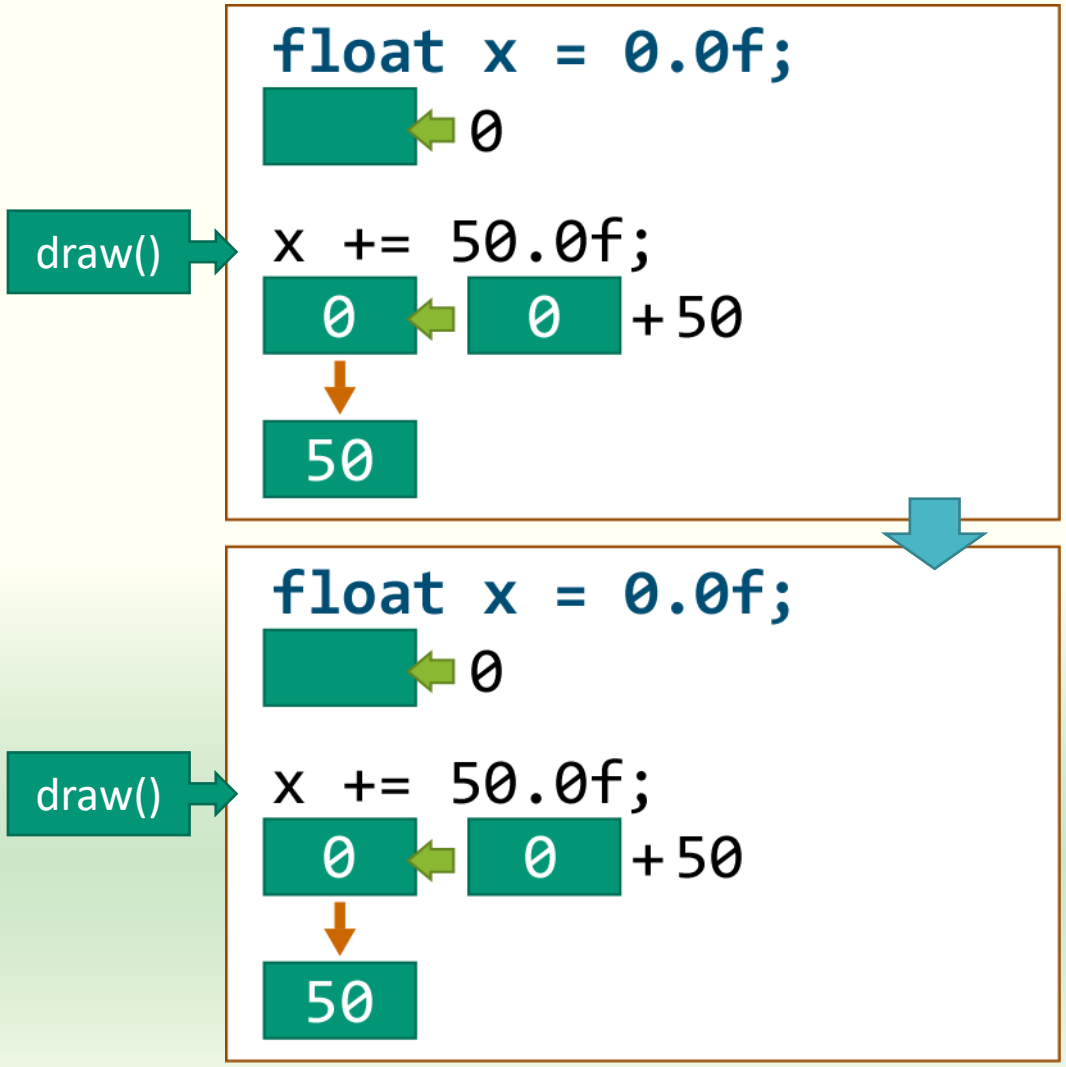
一瞬で消える

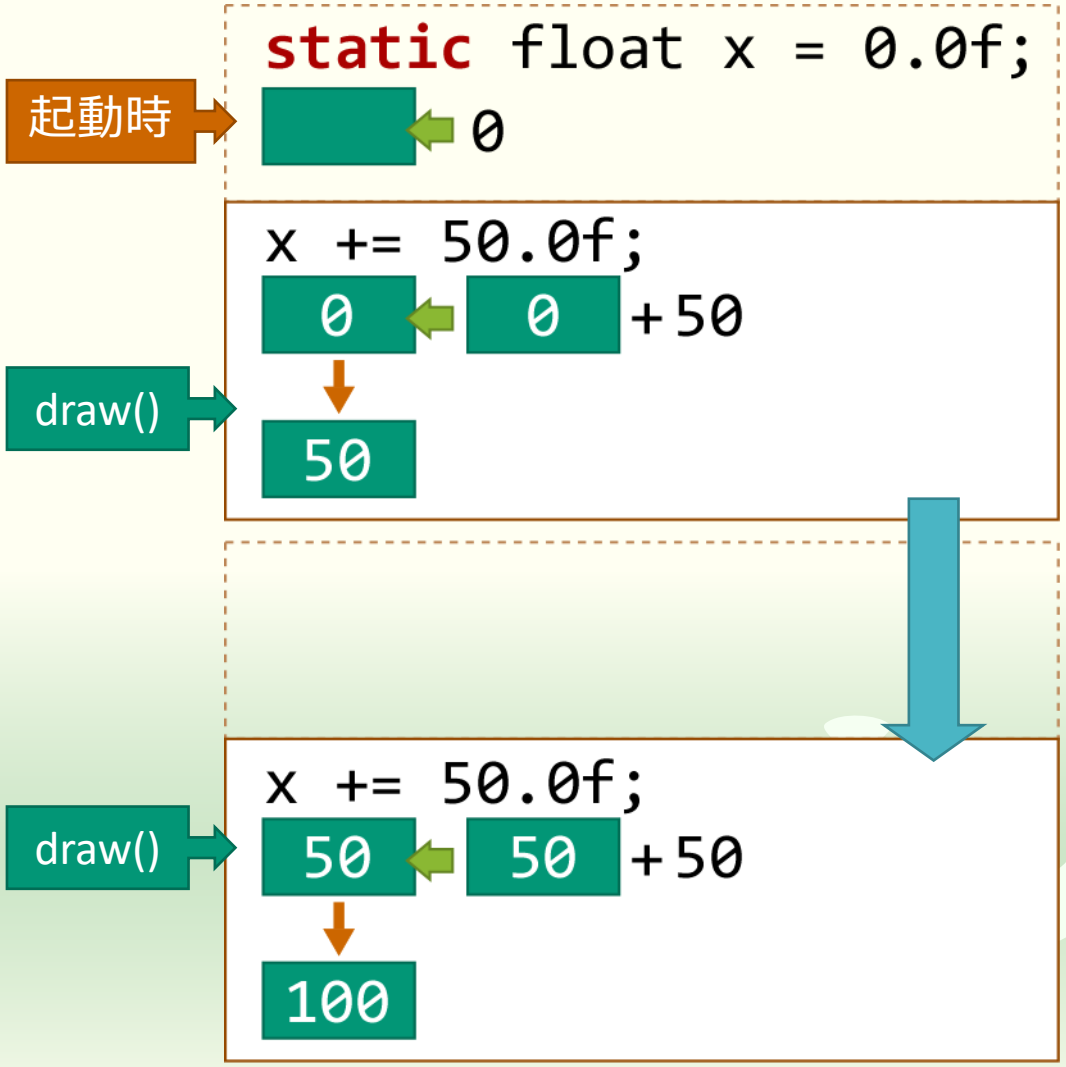


変数のメモリが確保されるタイミング

	static を付けない変数	static を付けた変数
メモリの確保	プログラムの実行が 変数のスコープに**入った**とき	プログラムの**起動**時
メモリの解放	プログラムの実行が 変数のスコープから**出た**とき	プログラムの**終了**時
初期化	変数宣言のところで **毎回**初期化される	起動時に**一度だけ** 初期化される
	自動変数	**静的変数**

静的変数の初期化は起動時に行われる





x, y の値の変更を update() で行う

```
const float radius{ 30.0f };
static float x{ 0.0f }, y{ 0.0f };

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

}

//--------------------------------------------
void ofApp::update(){
  x += 50.0f;
  y += 80.0f;
}

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

  ofDrawCircle(x, y, radius);
}
```

- x, y の変数宣言を関数の外の update() と draw() の両方より前に置く
- **static** float x{ 0.0f }, y{ 0.0f };
  - static を付けて関数の外で宣言した変数は ofApp.cpp 内でのみ使用できる

static を付けずに関数の外で変数宣言する

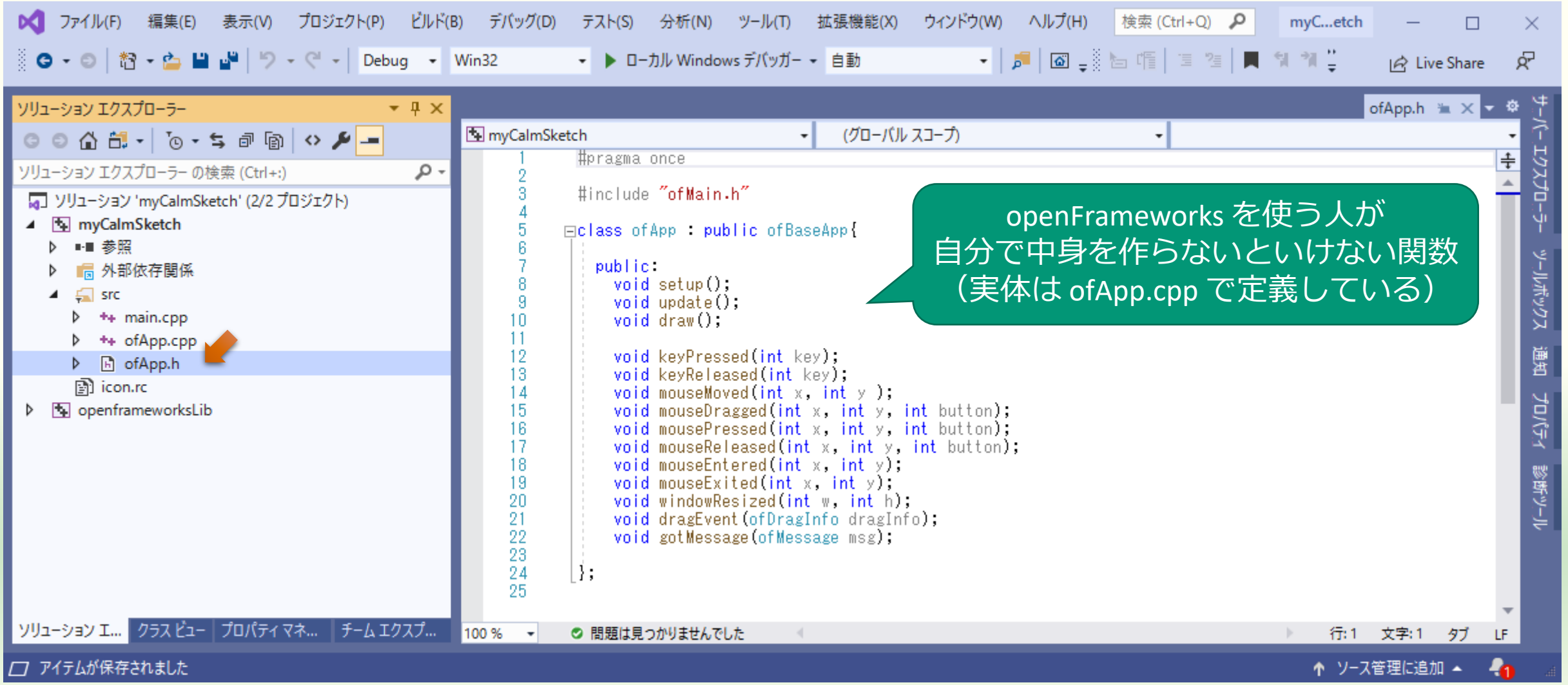
```
const float radius{ 30.0f };
float x{ 0.0f }, y{ 0.0f };

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

}

//--------------------------------------------
void ofApp::update(){
  x += 50.0f;
  y += 80.0f;
}

//--------------------------------------------
void ofApp::draw(){
  ofDrawCircle(x, y, radius);
}
```

- static を付けないで関数の外で宣言した変数は ofApp.cpp 以外にあるどの関数からも使える
  - **大域変数 (グローバル変数)**
  - メモリは静的（プログラム起動時）に確保される

大域変数はできるだけ使用しない

- 大域変数はプログラム全体のどこからも利用できる
  - 意図しないところで変数の内容を変えてしまうことがある
  - 大域変数と同じ名前の変数を使ってしまうことがある
  - 関数同士の依存性が高まってプログラムが複雑になる
    - 異なる関数がどこかにある同じ大域変数を使っていると、一方を変更したときの影響がもう一方に出ることがある
- どうしても大域変数を使わざるを得ない場合はある
  - それ以外は使用を避ける

そこで ofApp.h を開く

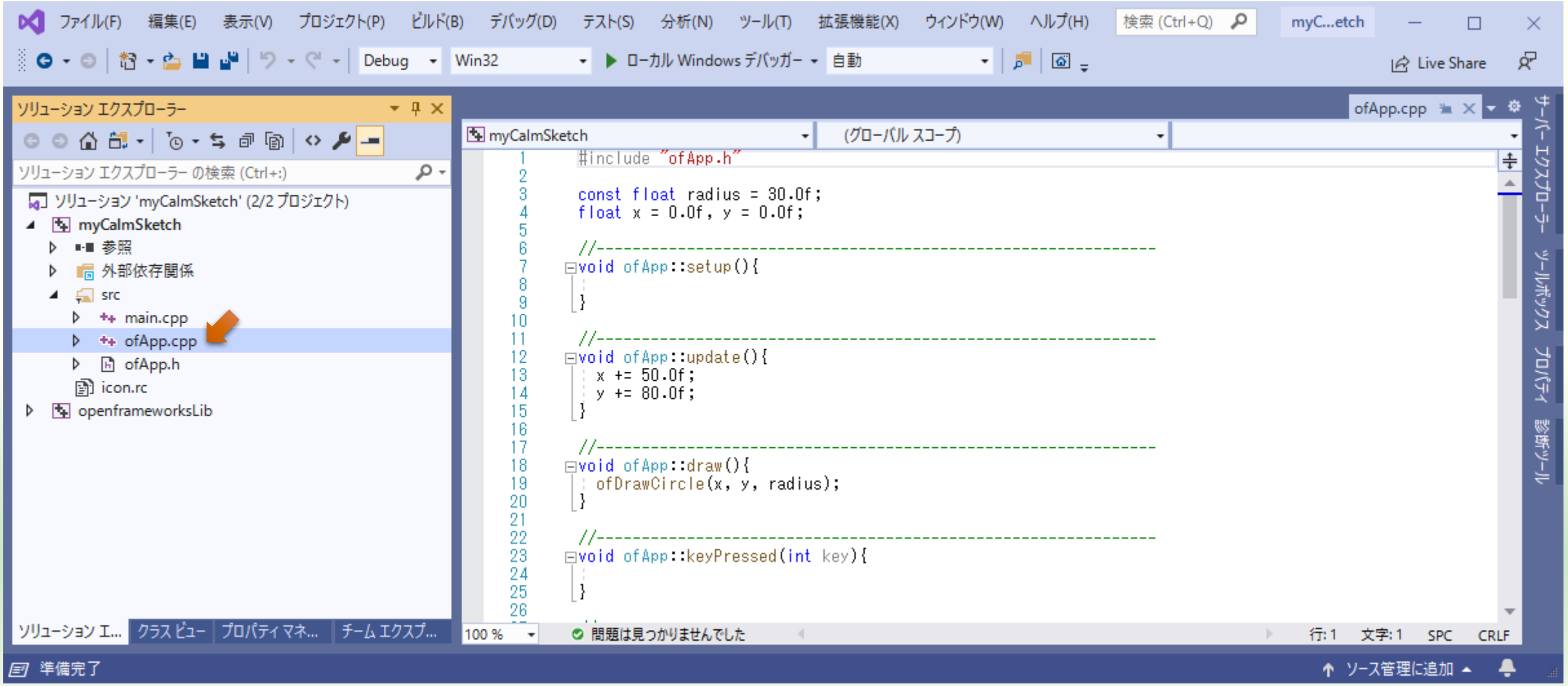


ofApp.h の ofApp クラスで x, y を変数宣言する

```
class ofApp : public ofBaseApp{
  float x, y;

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

  void keyPressed(int key);
  void keyReleased(int key);
  void mouseMoved(int x, int y);
  void mouseDragged(int x, int y, int button);
  void mousePressed(int x, int y, int button);
  void mouseReleased(int x, int y, int button);
  void mouseEntered(int x, int y);
  void mouseExited(int x, int y);
  void windowResized(int w, int h);
  void dragEvent(ofDragInfo dragInfo);
  void gotMessage(ofMessage msg);

};
```

- ofApp.h は **ofApp** という**クラス**を宣言している
  - クラスは自分で定義するデータ型
- x, y を変数宣言する
  - **メンバ変数**
  - 下の setup() 以降の**メンバ関数**で共通に使える
    - これら以外からは使えない

再び ofApp.cpp を開く



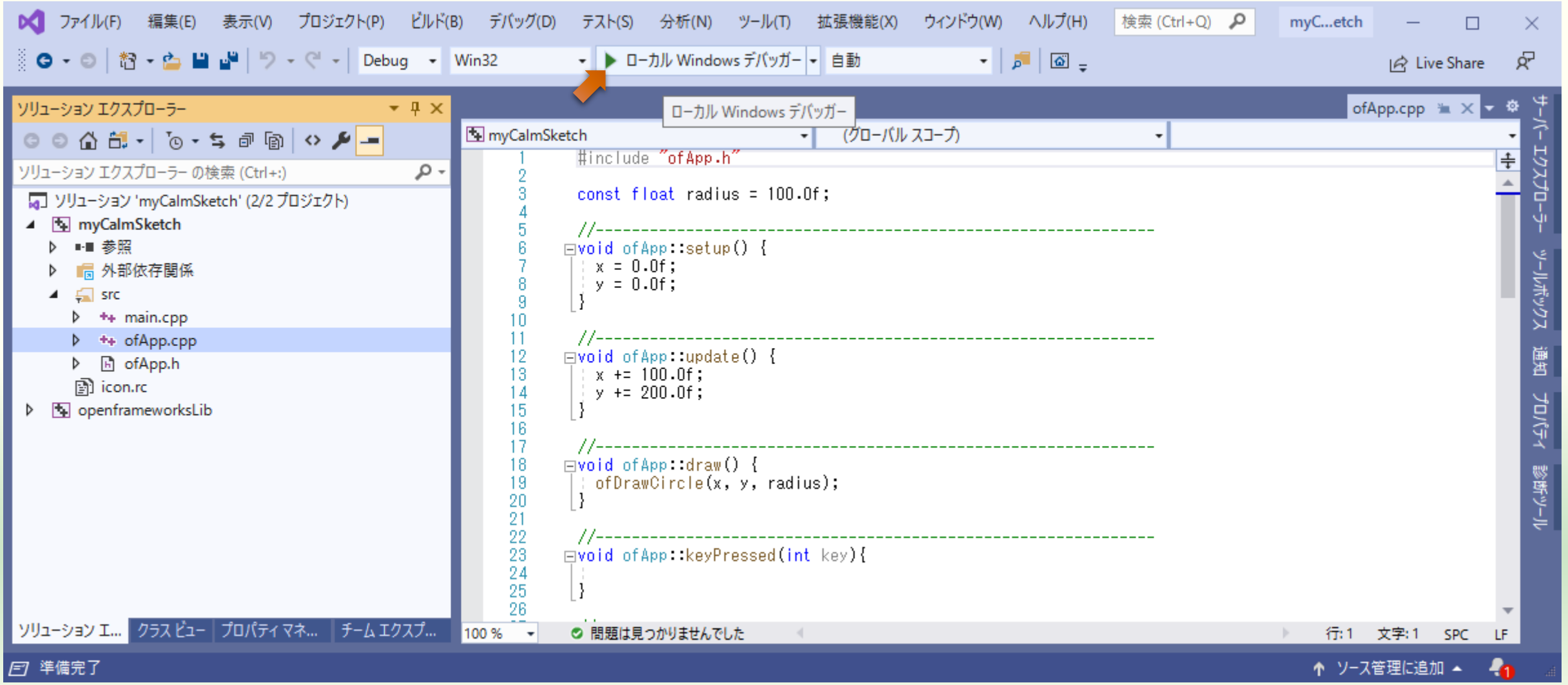
x, y の変数宣言を ofApp.cpp から削除する

```
const float radius{ 30.0f };
float x{ 0.0f }, y{ 0.0f };

//--------------------------------------------
void ofApp::setup(){
  x = 0.0f;
  y = 0.0f;
}

//--------------------------------------------
void ofApp::update(){
  x += 50.0f;
  y += 80.0f;
}

//--------------------------------------------
void ofApp::draw(){
  ofDrawCircle(x, y, radius);
}
```

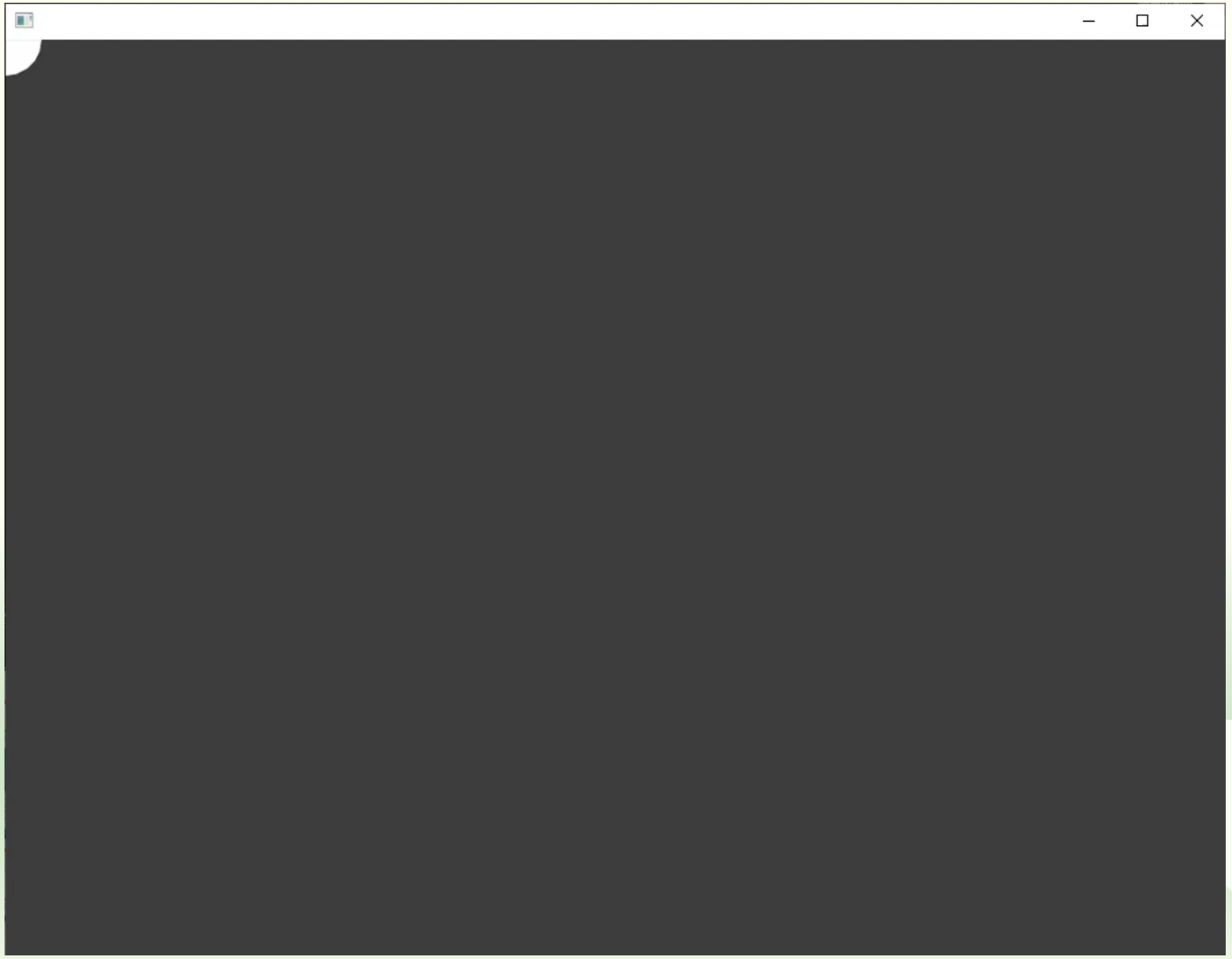
削除

- ofApp.h で初期化は行っていないので setup() で初期値を設定する

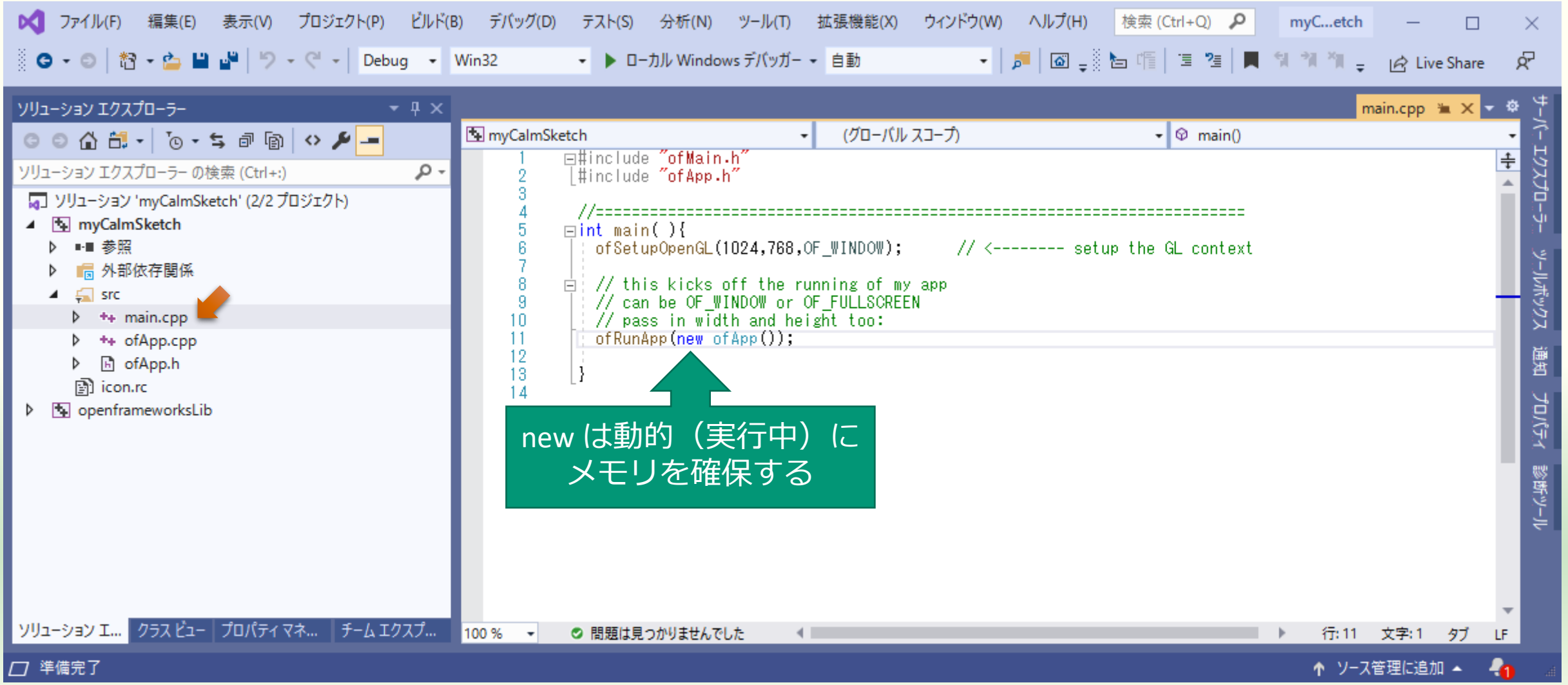
ビルドと実行


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)
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 'myCalmSketch' (2/2 プロジェクト)
myCalmSketch
参照
外部依存関係
src
main.cpp
ofApp.cpp
ofApp.h
icon.rc
openframeworksLib
ofApp.cpp
myCalmSketch
(グローバル スコープ)
#include "ofApp.h"
const float radius = 100.0f;
//--
void ofApp::setup() {
x = 0.0f;
y = 0.0f;
}
//--
void ofApp::update() {
x += 100.0f;
y += 200.0f;
}
//--
void ofApp::draw() {
ofDrawCircle(x, y, radius);
}
//--
void ofApp::keyPressed(int key){
}
100 %
問題は見つかりませんでした
行: 1 文字: 1 SPC LF
準備完了
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 通知 プロパティ 診断ツール

特に変わらない（やっぱり一瞬で消える）



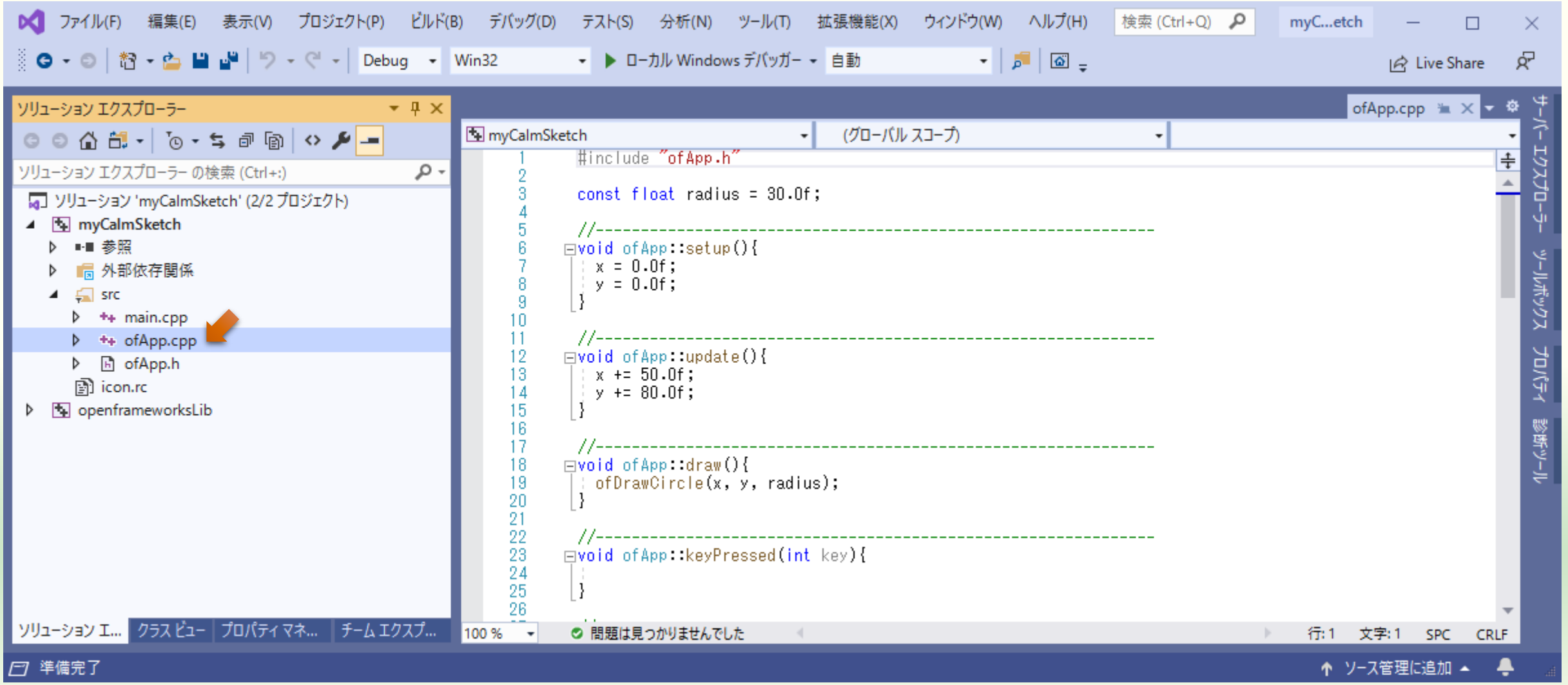
x, y のメモリは main() で確保している



クラスとオブジェクト

- **クラス**
  - ユーザ（プログラムを書く人）が自分で定義するデータ型
  - データを保持する**メンバ変数**と、それを操作する**メンバ関数**（データを操作する**メソッド**）を内包している
- **オブジェクト**
  - クラスによって定義された構造を持つメモリ上のデータ
    - クラスをデータ型とする変数に確保されたメモリ上のデータ
    - クラスをデータ型として動的に確保されたメモリ上のデータ
  - そのクラスの**インスタンス**とも呼ばれる

ofApp.cpp を開く



{ } の内側にある変数宣言が優先される

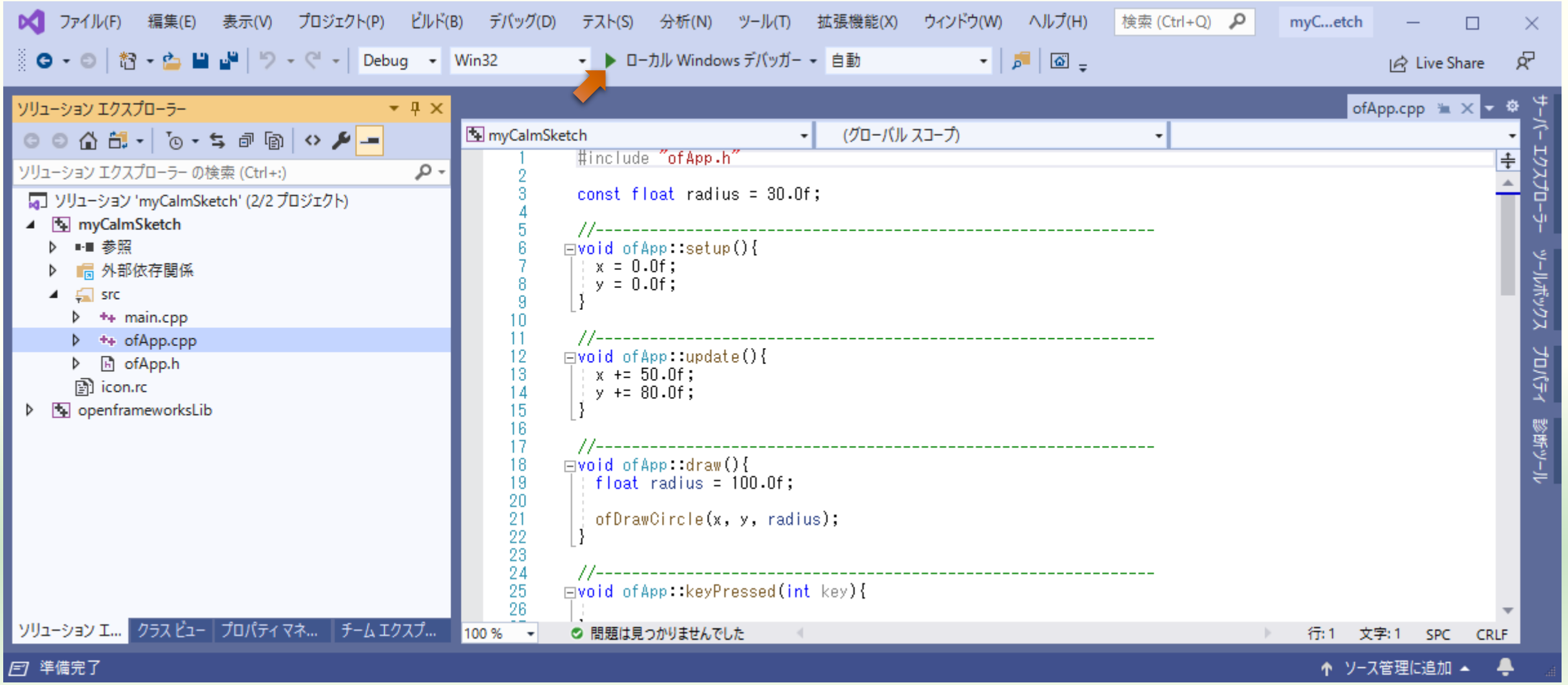
```
const float radius{ 30.0f };

//--------------------------------------------
void ofApp::setup(){
  x = 0.0f;
  y = 0.0f;
}

//--------------------------------------------
void ofApp::update(){
  x += 100.0f;
  y += 200.0f;
}

//--------------------------------------------
void ofApp::draw(){
  float radius{ 300.0f };

  ofDrawCircle(x, y, radius);
}
```

別物

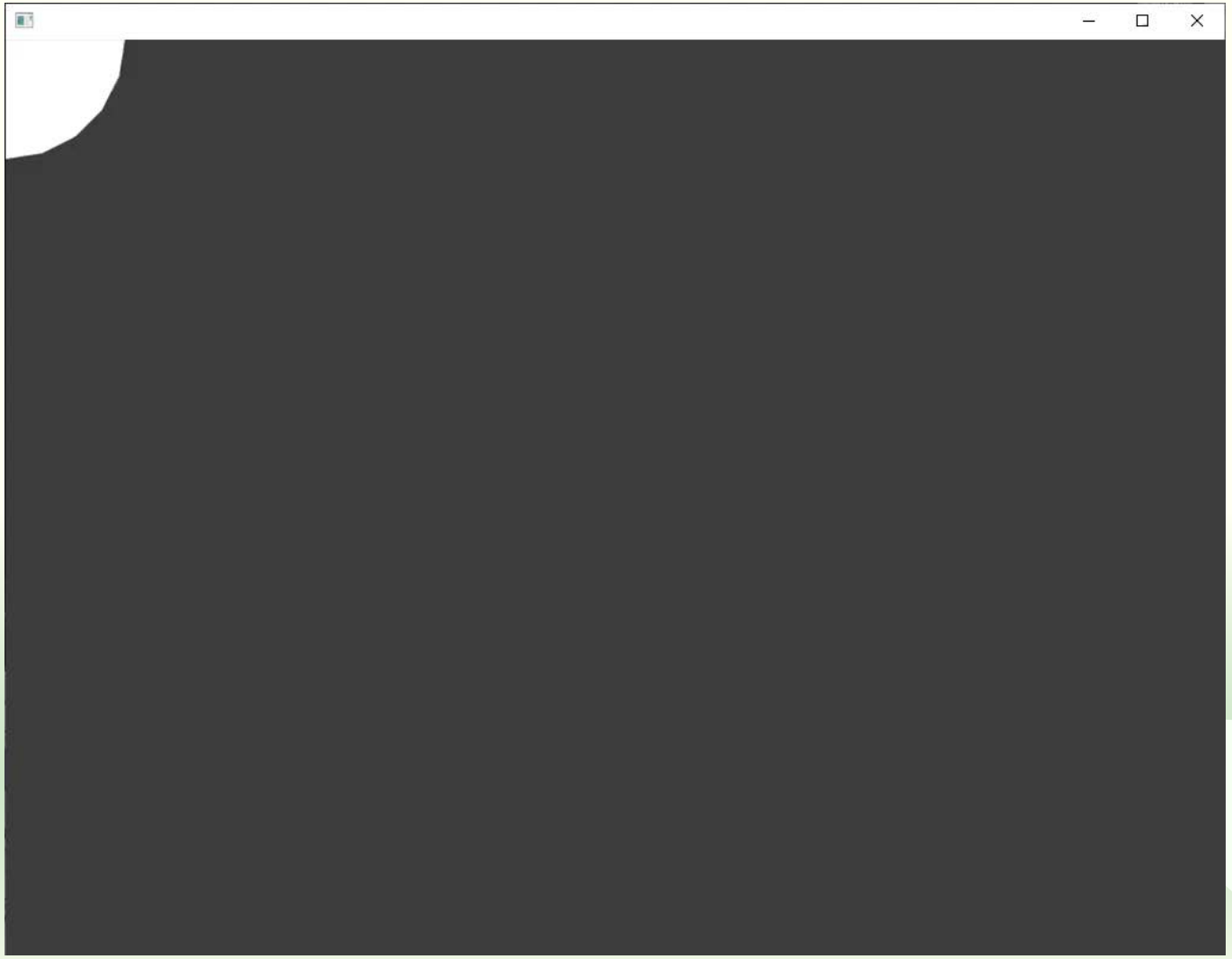
同じ

- { } の外にある変数と同じ変数名の変数を { } 内で変数宣言する
  - 外にある変数が使えなくなる
  - 別の変数になる（データを保存するメモリの場所が異なる）ので互いに影響は与えない

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 'myCalmSketch' (2/2 プロジェクト)
myCalmSketch
参照
外部依存関係
src
main.cpp
ofApp.cpp
ofApp.h
icon.rc
openframeworksLib
ofApp.cpp
myCalmSketch
(グローバル スコープ)
#include "ofApp.h"
const float radius = 30.0f;
//--
void ofApp::setup(){
x = 0.0f;
y = 0.0f;
}
//--
void ofApp::update(){
x += 50.0f;
y += 80.0f;
}
//--
void ofApp::draw(){
float radius = 100.0f;
ofDrawCircle(x, y, radius);
}
//--
void ofApp::keyPressed(int key){
100 %
問題は見つかりませんでした
行: 1 文字: 1 SPC CRLF
準備完了
ソース管理に追加

円が大きくなる（しかし一瞬で消える）



変数に関して覚えておいて欲しいこと

- **データ型**
  - 整数型と実数型がある
  - それぞれ精度の異なるデータ型がある
- **代入**
  - 変数の値を変更する
- **初期化**
  - 変数の初期値を設定する
- **スコープ**
  - 局所（ローカル）変数
  - 大域（グローバル）変数
  - メンバ変数
- **記憶クラス**
  - 自動変数
  - 静的変数
- **動的なメモリ確保**
  - 変数には結びつけられない
  - ポインタ（後述）を保存しておく

速度を考慮する

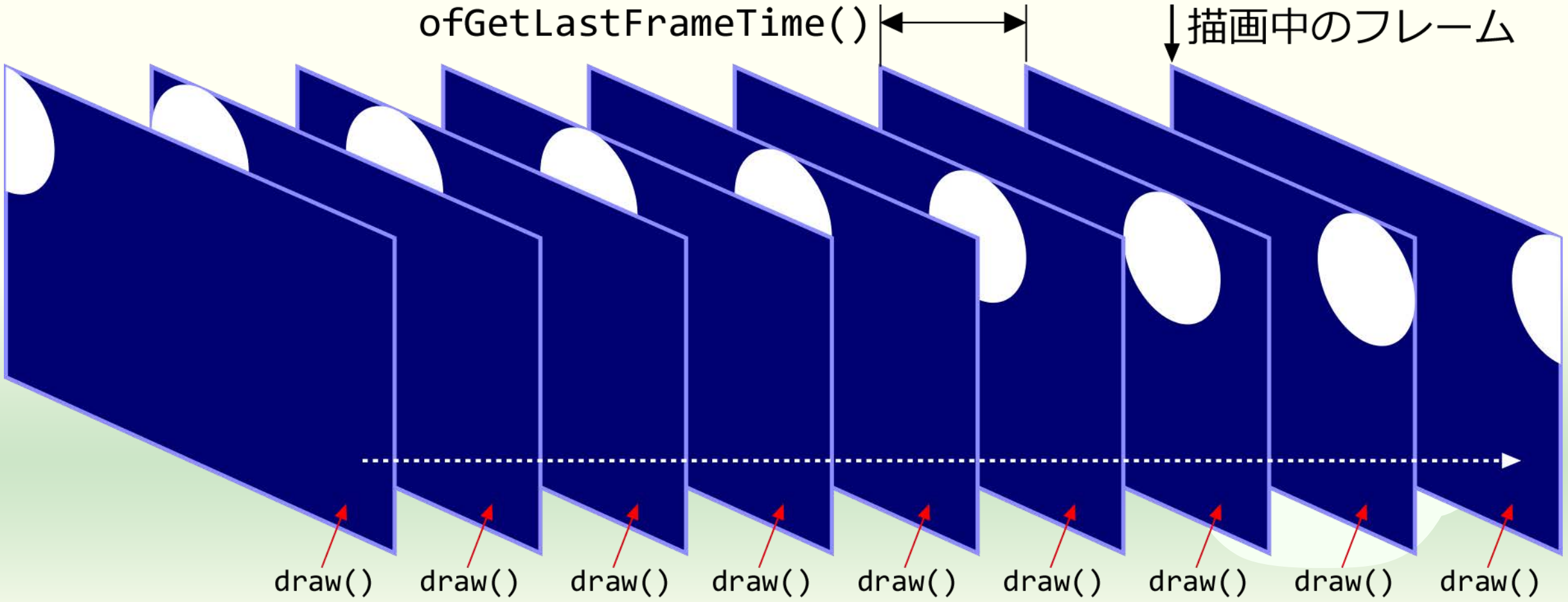
アニメーションの速度を描画時間に依存させない

アニメーション

- パソコンの画面は静止画を**繰り返し表示**している
  - openFrameworks ではそのタイミングで draw() を実行する
- パソコンの１枚の画面表示を**フレーム**という
  - 一般的なディスプレイは１秒間に６０フレーム表示可能
  - １フレームにかかる時間は $1/60 = 0.016666...$ 秒
  - それより速いディスプレイもある（ゲーミングディスプレイ）
- ofGetLastFrameTime()
  - 直前のフレームを表示してから現在のフレームを表示するまでの時間を調べる

double ofGetLastFrameTime()

- 直前のフレームの描画にかかった時間を返す



位置に速度と時間の積を加えて更新する

```
const float radius{ 30.0f };
const float velocity{ 300.0f };

//--------------------------------
void ofApp::setup(){
  x = radius;
  y = radius;
}

//--------------------------------------------
void ofApp::update(){
  x += velocity * ofGetLastFrameTime();
  y += 0.0f;
}

//--------------------------------------------
void ofApp::draw(){
  float radius{ 300.0f };

  ofDrawCircle(x, y, radius);
}
```

1秒間に300画素移動ということ

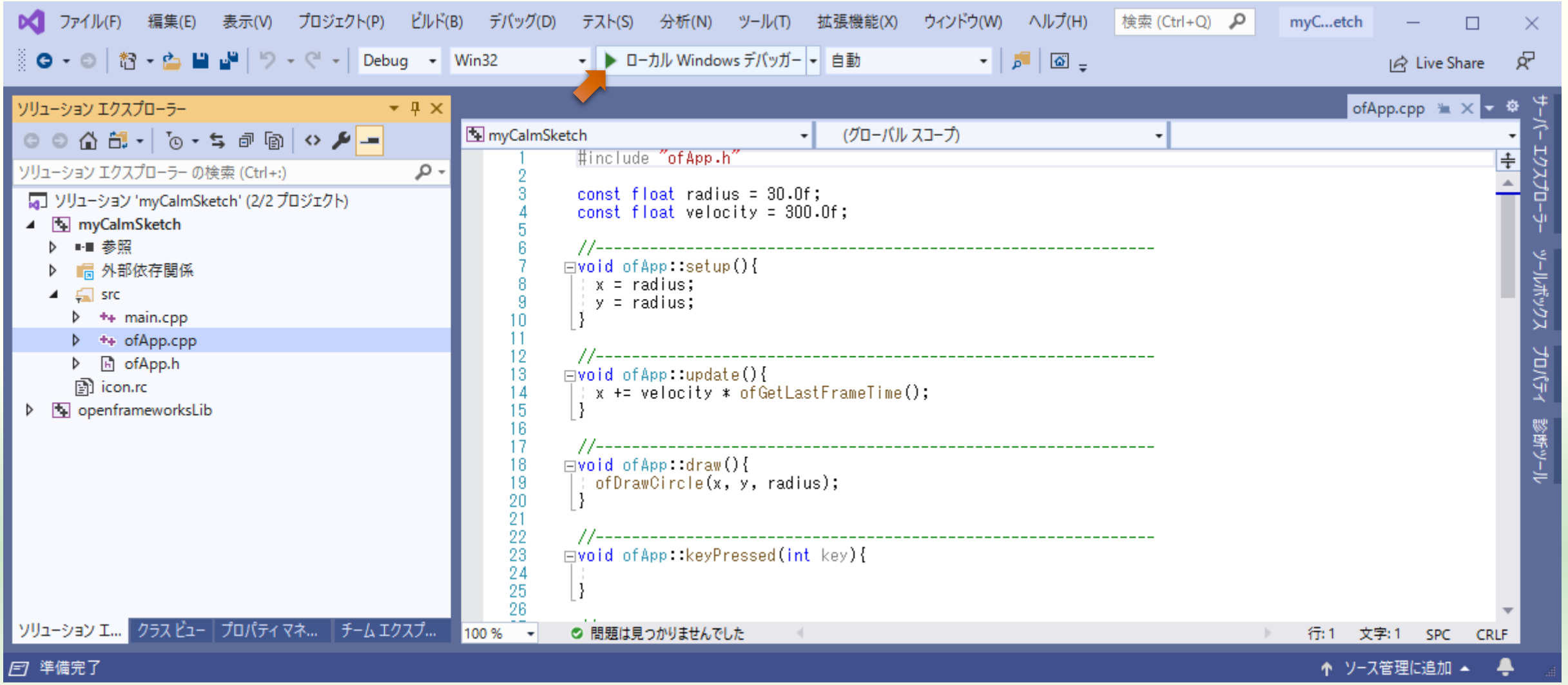
削除

- 現在位置x, 次の位置x', 速度v, 経過時間Δt

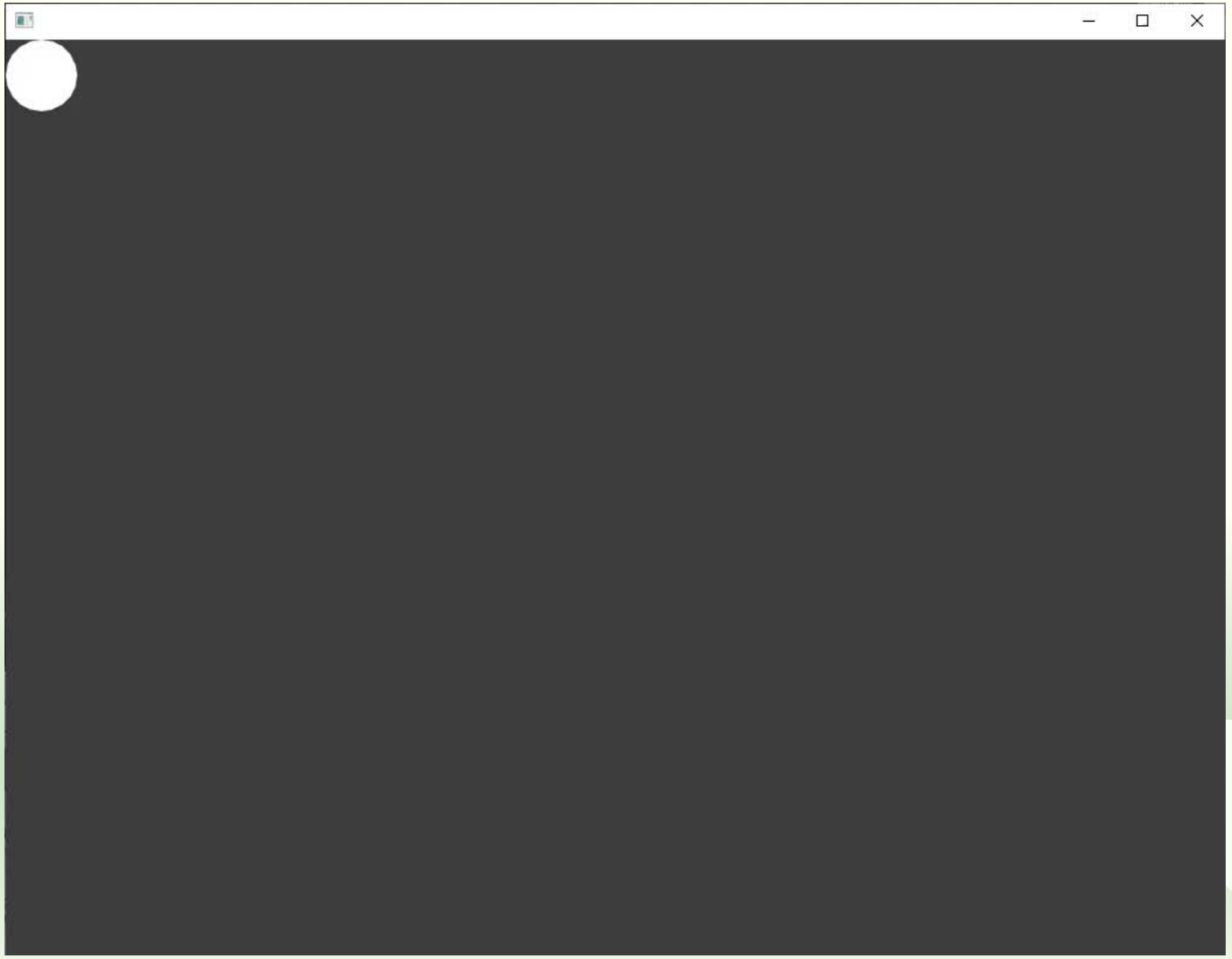
$$x' = x + v\Delta t$$

- Δt: ofGetLastFrameTime()
- y 方向には移動しない
- x, y の初期値を円の半径にしておけば起点でウィンドウからはみ出ない

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 'myCalmSketch' (2/2 プロジェクト)
myCalmSketch
参照
外部依存関係
src
main.cpp
ofApp.cpp
ofApp.h
icon.rc
openframeworksLib
ofApp.cpp
myCalmSketch
(グローバル スコープ)
#include "ofApp.h"
const float radius = 30.0f;
const float velocity = 300.0f;
//--
void ofApp::setup(){
x = radius;
y = radius;
}
//--
void ofApp::update(){
x += velocity * ofGetLastFrameTime();
}
//--
void ofApp::draw(){
ofDrawCircle(x, y, radius);
}
//--
void ofApp::keyPressed(int key){
}
100 %
問題は見つかりませんでした
行: 1 文字: 1 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
ソース管理に追加

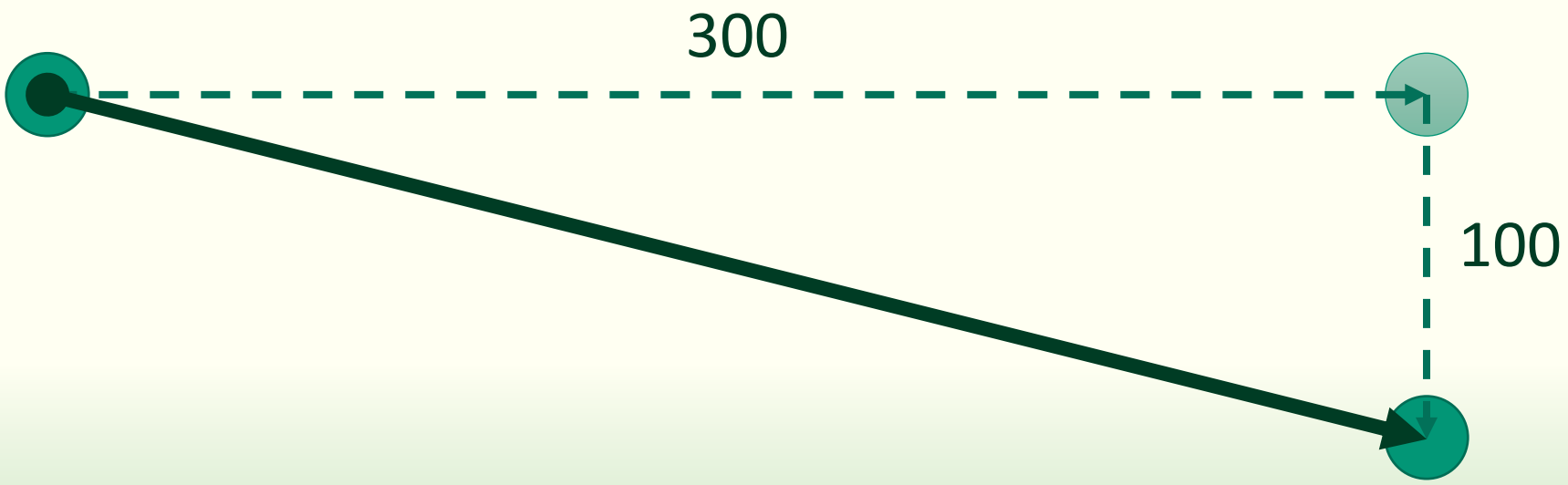
横に動く



課題２－１

斜めに動かしてみる

y 方向の速度を 100画素/秒に設定しなさい



ofApp.cpp を修正して「ビルドと実行」しなさい

ベクトルを使う

線形代数のアレ

ofDrawCircle() のマニュアル

global functions

- ofBackground()
- ofBackgroundGradient()
- ofBackgroundHex()
- ofBeginSaveScreenAsPDF()
- ofBeginSaveScreenAsSVG()
- ofBeginShape()
- ofBezierVertex()
- ofClear()
- ofClearAlpha()
- ofCurveVertex()
- ofCurveVertices()
- ofDisableAlphaBlending()
- ofDisableAntiAliasing()
- ofDisableBlendMode()
- ofDisableDepthTest()
- ofDisablePointSprites()
- ofDisableSmoothing()
- ofDrawBezier()
- ofDrawBitmapString()
- ofDrawBitmapStringHighlight()
- ofDrawCircle()

ofDrawCircle(...)

```
void ofDrawCircle(const glm::vec3 &p, float radius)
```

ofDrawCircle(...)

```
void ofDrawCircle(const glm::vec2 &p, float radius)
```

ofDrawCircle(...)

```
void ofDrawCircle(float x, float y, float radius)
```

Documentation from code comments

Draws a circle, centered at x,y, with a given radius.

```
void ofApp::draw(){
    ofDrawCircle(150,150,100);
}
```

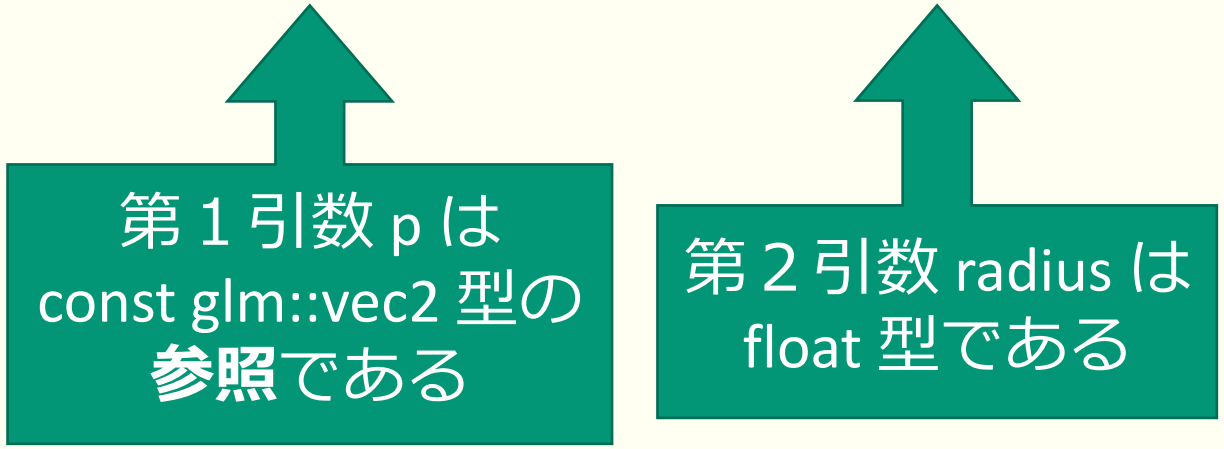
Please keep in mind that drawing circle with different outline color and fill requires calling ofNoFill and ofSetColor for drawing stroke and ofFill and again ofSetColor for filled solid color circle.

ofDrawCircle(...)

```
void ofDrawCircle(float x, float y, float z, float radius)
```

ofDrawCircle() のもう一つの宣言

- **void ofDrawCircle(const glm::vec2 &p, float radius)**



マニュアル

https://openframeworks.cc//documentation/graphics/ofGraphics/#!show_ofDrawCircle

スコープ解決演算子 ::

- プログラムはいろんなライブラリを組み合わせて作る
  - 他の誰かが作ったものを使う
  - 他の誰かが同じ変数名や関数名を使っているかもしれない
  - 変数名や関数名が同じだと使えない（**名前の衝突**）
- **名前空間**
  - 変数や関数のグループの外側に付けた名前
  - 名前空間を変えれば変数名や関数名が同じでも衝突しない
  - std:: や glm:: のようにして名前空間を指定する

glm

- **OpenGL Mathematics (GLM)** の名前空間
  - https://glm.g-truc.net/
  - OpenGL Shading Language (GLSL) というグラフィックス用のプログラミング言語の仕様に倣った数学ライブラリ
  - openFrameworks はグラフィックス表示に OpenGL というライブラリを使っている
- **std** は標準ライブラリの名前空間

glm::vec2

- ２次元のベクトルのクラス

```
glm::vec2 p;                          // vec2 型の変数 p の宣言
p.x = 10.0f;                          // p の x 要素への代入
p.y = 20.0f;                          // p の y 要素への代入
p = vec2{ 10.0f, 20.0f };             // p の x, y 要素への代入
p = vec2{ 0.0f };                     // p の x, y 要素への代入
p += vec2{ 10.0f, 20.0f };            // p の x, y 要素への加算
p += vec2{ 10.0f };                   // p の x, y 要素への加算
p += 10.0f;                           // p の x, y 要素への加算
glm::vec2 q{ 3.0f, 4.0f };            // q の x, y を初期化
glm::vec2 q{ 0.0f };                  // q の x, y を初期化
```

参照演算子 &

引数の値渡し

```
#include <iostream>

int sub(int x)
{
  x = 10;
}

int main()
{
  int x{ 0 };

  std::cout << x;   // 0 が出力される

  sub(x);

  std::cout << x;   // 0 が出力される
}
```

値 (0) がコピーされる
（使われるメモリが異なる）

引数の参照渡し

```
#include <iostream>

int sub(int &x)
{
  x = 10;
}

int main()
{
  int x{ 0 };

  std::cout << x;   // 0 が出力される

  sub(x);

  std::cout << x;   // 10 が出力される
}
```

同じものになる
（使われるメモリが同じ）

参照演算子 & と const

const 変数を参照渡しするとエラー

```
#include <iostream>

int sub(int &x)
{
  x = 10;
}

int main()
{
  const int x{ 0 };

  std::cout << x;   // 0 が出力される

  sub(x);

  (以下略)
}
```

参照渡しの仮引数を変更すると実引数が変更される

エラー

仮引数を const にした参照渡し

```
#include <iostream>

int sub(const int &x)
{
  std::cout << x;
}

int main()
{
  const int x{ 0 };

  std::cout << x;   // 0 が出力される

  sub(x);

  (以下略)
}
```

const を付けた仮引数は変更できない

この関数 sub() は実引数 x を変更しないことを保証している

ofApp クラスのメンバ x, y をベクトルにする

```
#pragma once

#include "ofMain.h"

using namespace glm;

class ofApp : public ofBaseApp{
    vec2 position;

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

   (以下略)
```

- float x, y; を vec2 position; に変更する
- **using namespace glm;** を先に書いておくと glm:: を省略できる

ofApp.cpp を変更する

```
const float radius = 30.0f;
const vec2 velocity{ 300.0f, 200.0f };

//--------------------------------------------
void ofApp::setup(){
  position = vec2{ radius };
}

//--------------------------------------------
void ofApp::update(){
  position += velocity * ofGetLastFrameTime();
}

//--------------------------------------------
void ofApp::draw(){
  ofDrawCircle(position, radius);
}
```

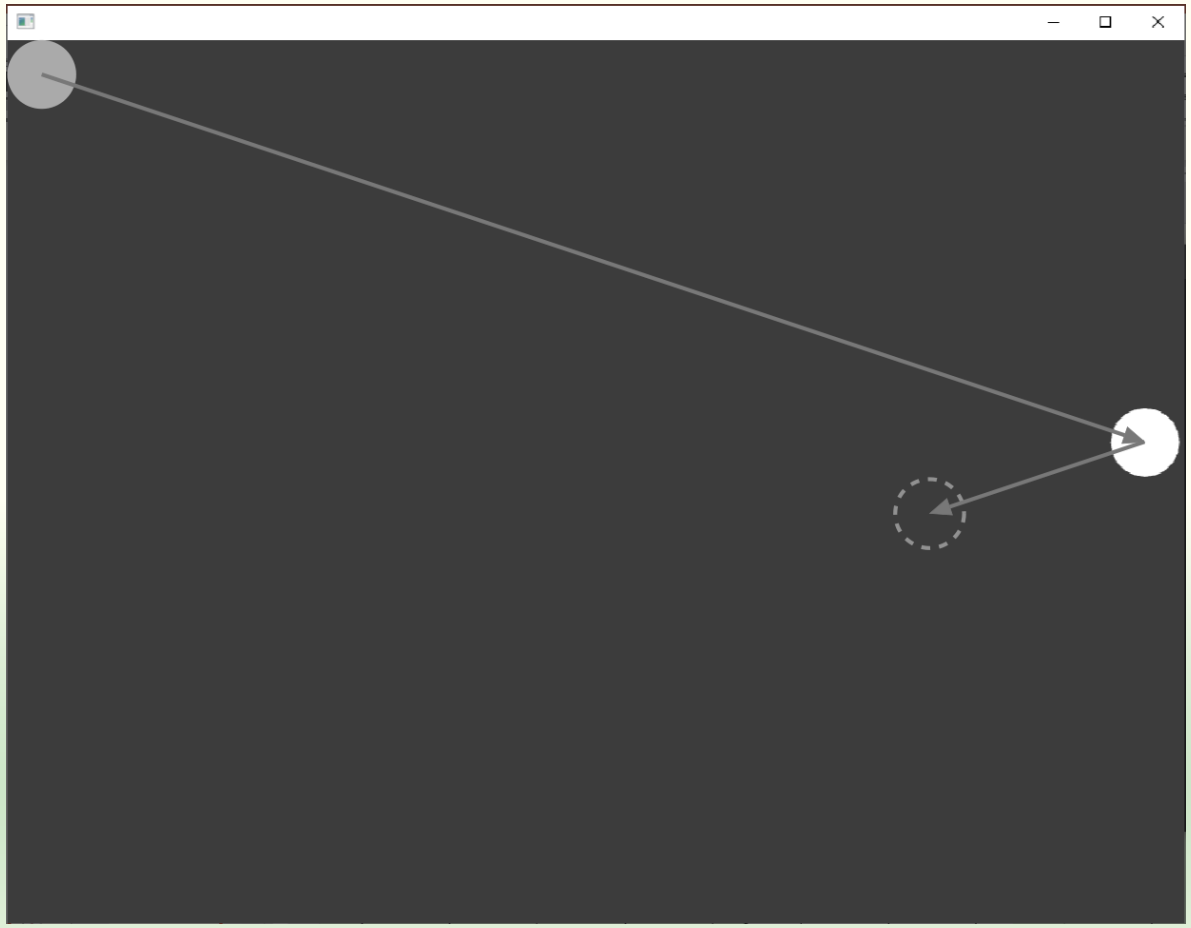
「ビルドと実行」

- velocity に速度を設定する
- position の x, y の両方の要素に radius を初期値として代入する
- position に velocity と経過時間の積を加えて円の位置を更新する
- position の位置に半径 radius の円を描く

課題２－２

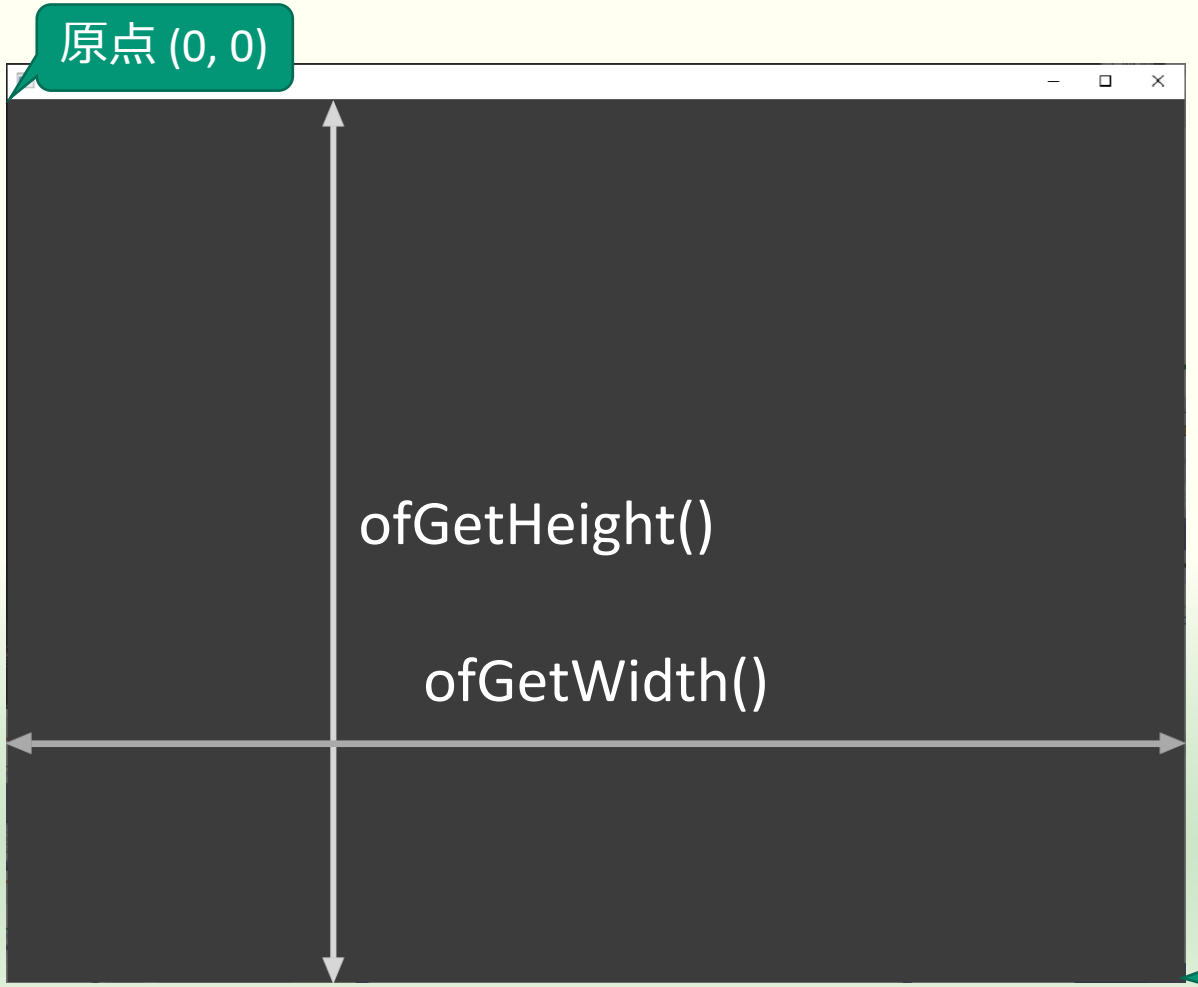
跳ね返らせてみる

円を壁（ウィンドウの端）で跳ね返らせる



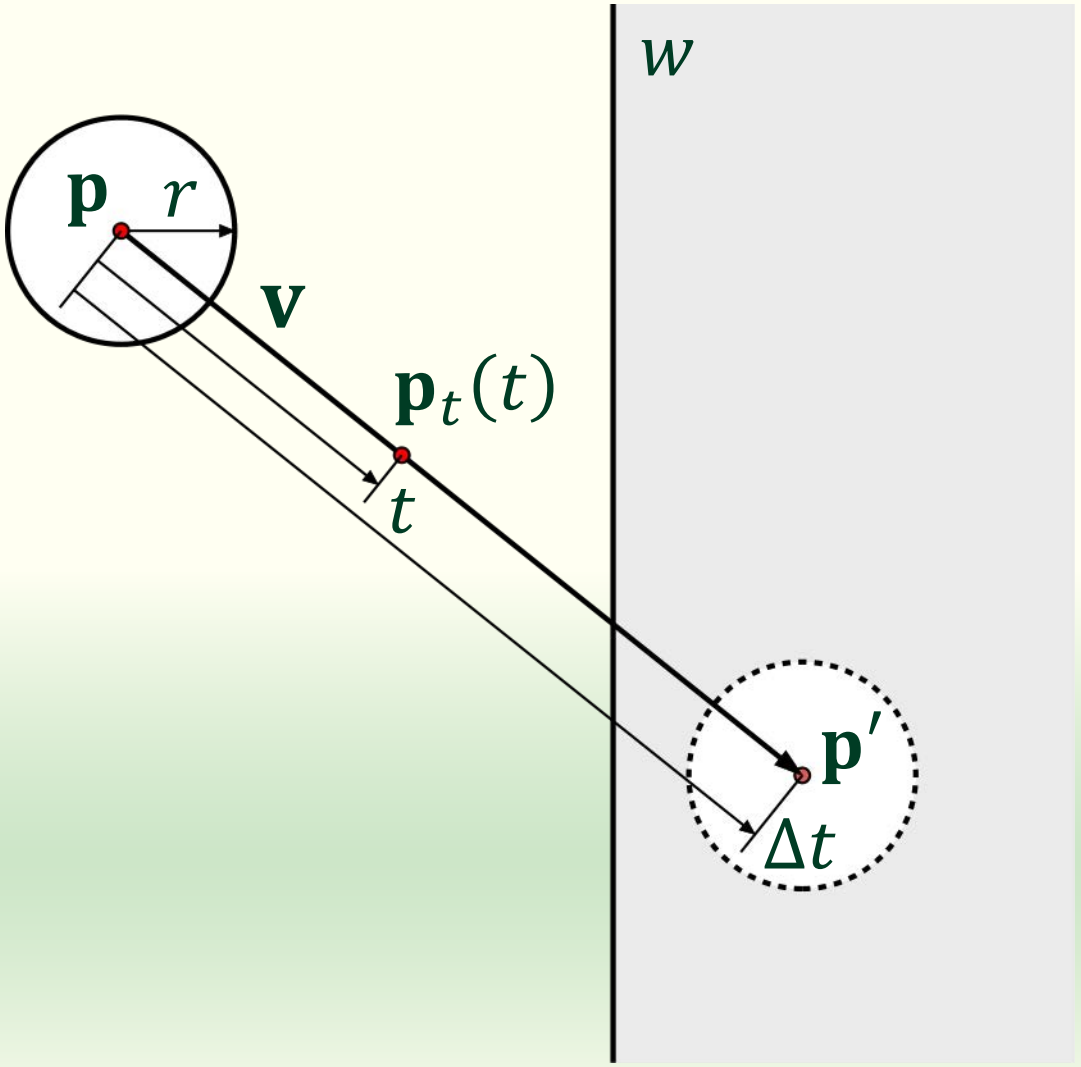
- 円がウィンドウから飛び出して帰ってこない
- 円が壁に当たったら跳ね返るようにする

openFrameworks のウィンドウの座標系



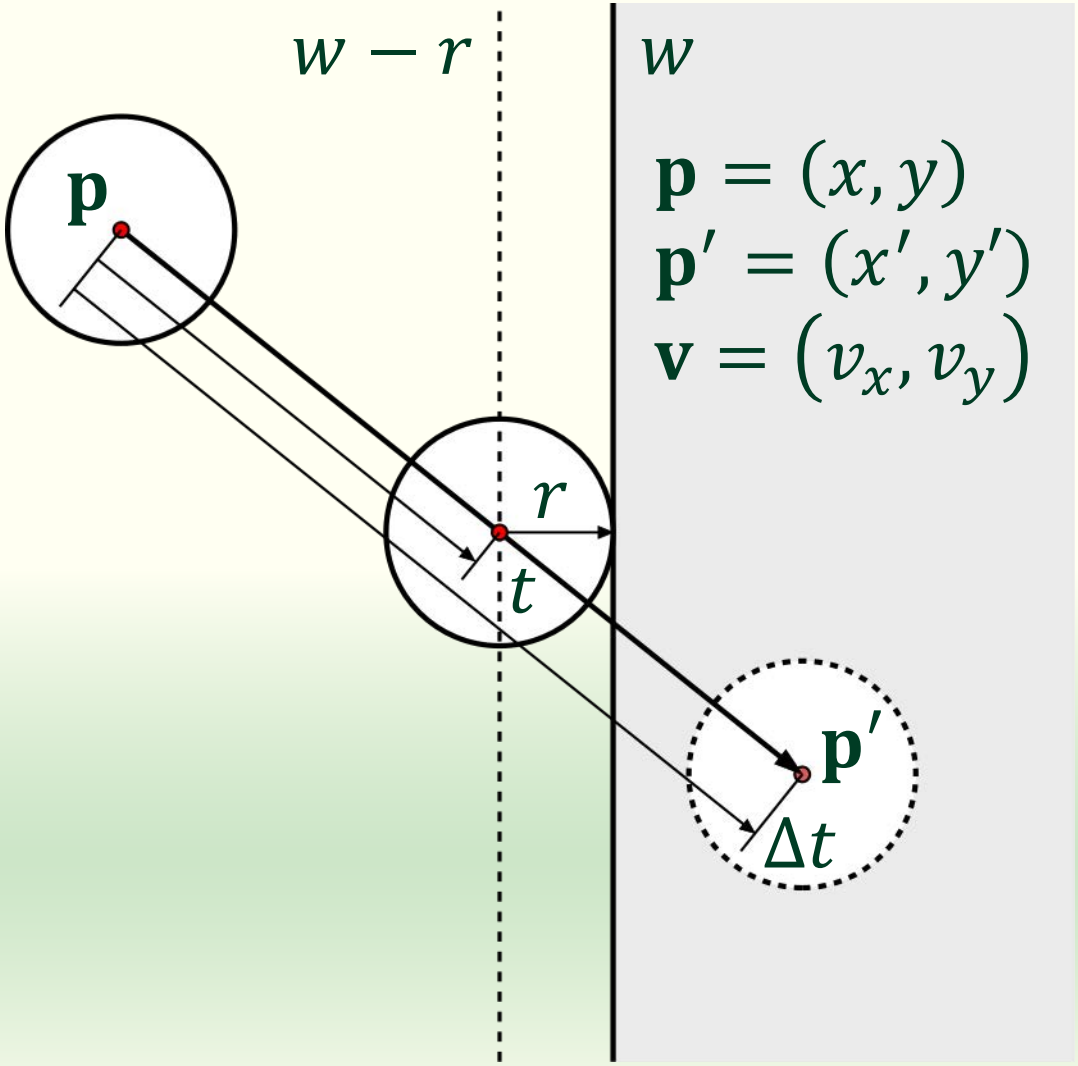
- int ofGetWidth()
  - ofApp() のウィンドウの幅を得る
- int ofGetHeight()
  - ofApp() のウィンドウの高さを得る

円の移動



- 円の移動
  - $\mathbf{p}_t(t) = \mathbf{p} + \mathbf{v}t$
    - $\mathbf{p}_t(\Delta t) = \mathbf{p}'$
- 変数
  - $\mathbf{p}$: position, $\mathbf{v}$: velocity
  - Δt: ofGetLastFrameTime()
  - w: ofGetWidth() - 1
  - r: radius

円が壁に衝突した時刻



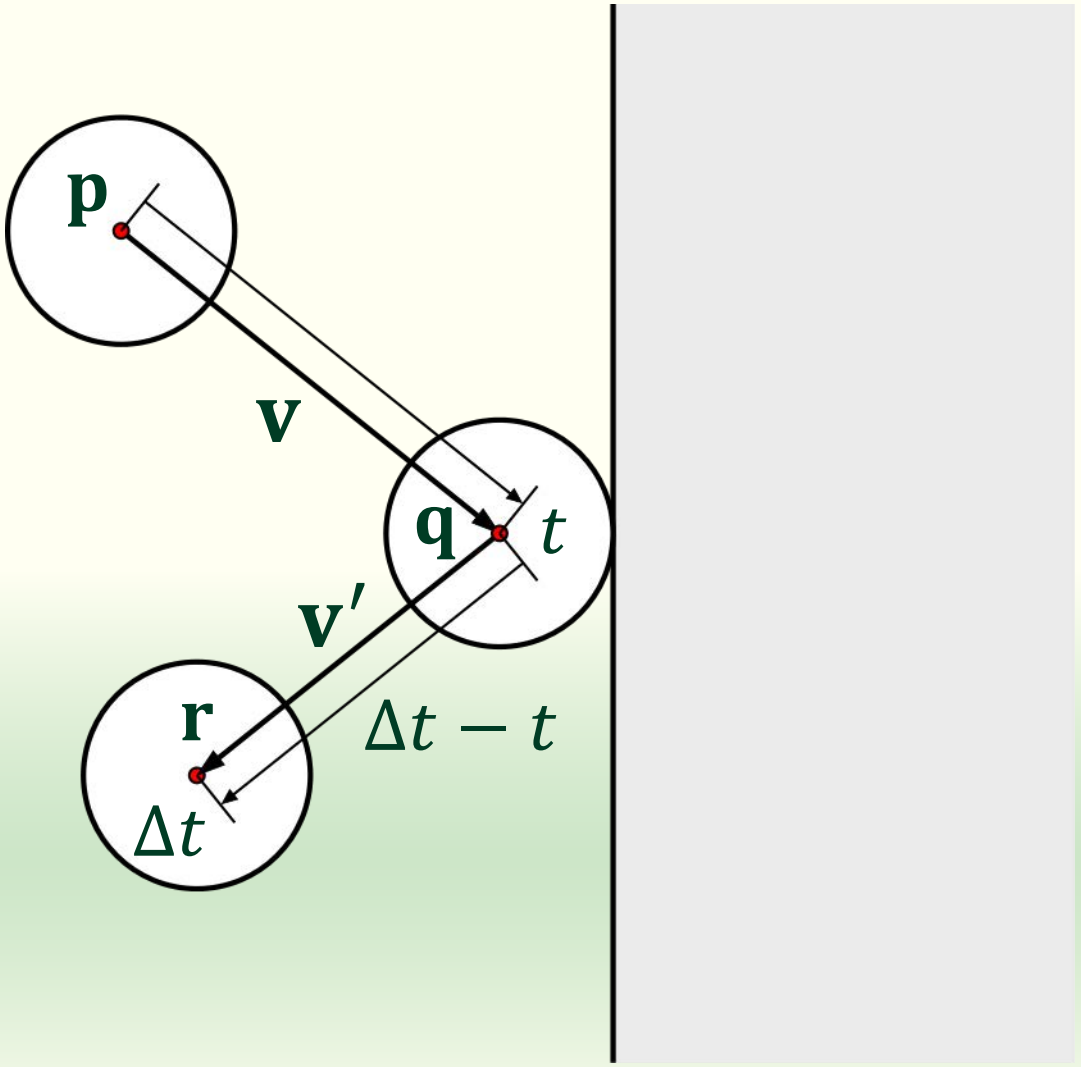
- 現在時刻から Δt 後の円の位置が $x' \geq w - r$ なら壁と衝突している
- そのとき衝突した時刻は

$$x + v_x t = w - r$$

より

$$t = \frac{w - r - x}{v_x}$$

衝突後の円の位置



- 衝突位置

$$\mathbf{q} = (w - r, y + v_y t)$$

- 衝突後の円の速度

$$\mathbf{v}' = (-v_x, v_y)$$

- 円の Δt 後の位置

$$\mathbf{r} = \mathbf{q} + \mathbf{v}'(\Delta t - t)$$

ofApp クラスで velocity を変数宣言する

```
class ofApp : public ofBaseApp{
  vec2 position, velocity;

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

  void keyPressed(int key);
  void keyReleased(int key);
  void mouseMoved(int x, int y );
  void mouseDragged(int x, int y, int button);
  void mousePressed(int x, int y, int button);
  void mouseReleased(int x, int y, int button);
  void mouseEntered(int x, int y);
  void mouseExited(int x, int y);
  void windowResized(int w, int h);
  void dragEvent(ofDragInfo dragInfo);
  void gotMessage(ofMessage msg);

};
```

- velocity を ofApp クラスのメンバ変数にする

setup() の変更

```
const float radius = 30.0f;
const vec2 velocity{ 300.0f, 100.0f };

//--------------------------------------------
void ofApp::setup(){
  position = vec2{ radius };
  velocity = vec2{ 300.0f, 100.0f };
}
```

削除

- velocity はメンバ変数にしたので削除する
- setup() で velocity に初期値を設定する

update() の変更

```
void ofApp::update(){

  // 直前のフレームの時間間隔を保存しておく
  const float dt{ float(ofGetLastFrameTime()) };

  // 現在の円の位置を記録しておく
  const vec2 p{ position };

  // 円の位置を更新する
  position += velocity * dt;
```

変数の値を変更しないなら変数宣言に **const** を付ける

w: ofGetWidth() - 1

```
  // w - r を計算する
  const float wr{ ofGetWidth() - 1.0f - radius };

  // もし円がウィンドウからはみ出たら
  if (position.x >= wr){

    // 壁と衝突した時刻を求める
    const float t{ (wr - p.x) / velocity.x };

    // 衝突した位置を求める
    const vec2 q{ wr, p.y + velocity.y * t };

    // 衝突後の速度を変更する
    velocity = vec2{ -velocity.x, velocity.y };

    // 円の位置を変更する
    position = q + velocity * (dt - t);
  }
}
```

円がウィンドウからはみ出たら速度 velocity と位置 position を変更する

if (position.x >= wr){ … }

```
if (position.x >= wr){

  // この部分は
  // position.x ≧ wr
  // のときのみ実行される

}
```

- もし position.x ≧ wr なら { … } 内の処理を行う
- **条件分岐**
  - if 文

関係演算

x > y	x が y より大きいなら true
x >= y	x が y 以上なら true
x < y	x が y より小さいなら true
x <= y	x が y 以下なら true
x == y	x と y が等しいなら true
x != y	x と y が等しくないなら true

- 数値の比較を行う演算式
- この比較を行う演算子を**関係演算子**という
- 演算の結果は **true** (真) または **false** (偽) の論理値

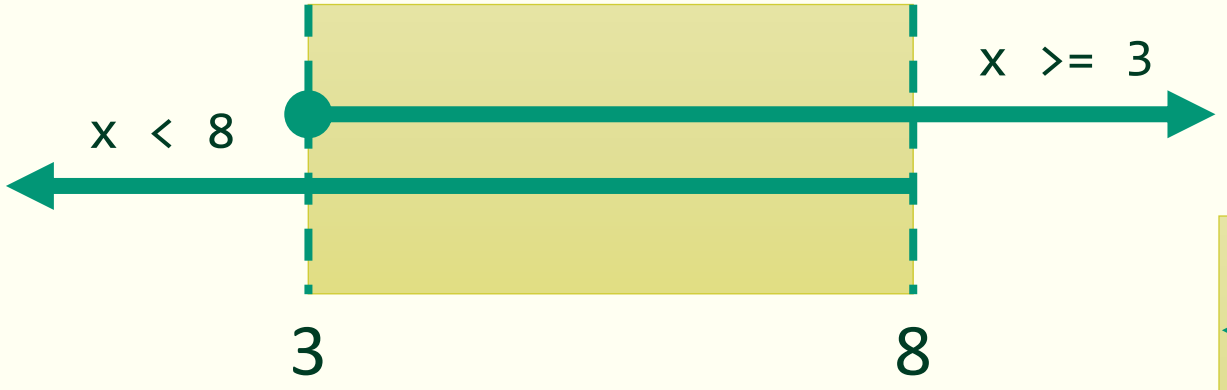
論理演算

x **&&** y	x と y が両方とも true のとき true
x **||** y	x と y のいずれかが true のとき true
!x	x が false なら true、true なら false （論理反転）

- 論理値 **true** と **false** を対象にした演算式
  - x, y が数値なら 0 は false、0 以外は true として扱う
- この演算を行う演算子を論理演算子という
- 演算の結果は true または false

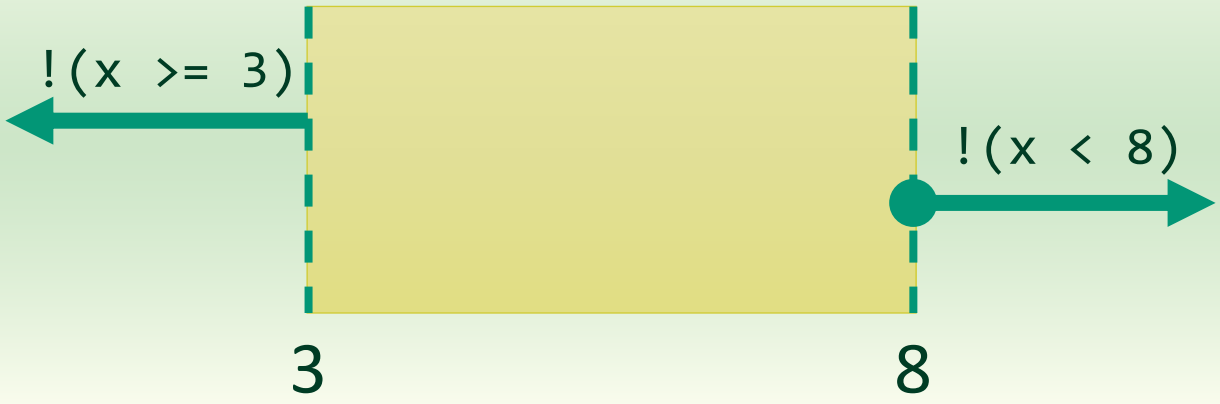
関係演算と論理演算の組み合わせ

```
x >= 3 && x < 8
```




```
!(x >= 3) || !(x < 8)
```


```
!(!(x >= 3) || !(x < 8))
```



条件分岐（if 文）

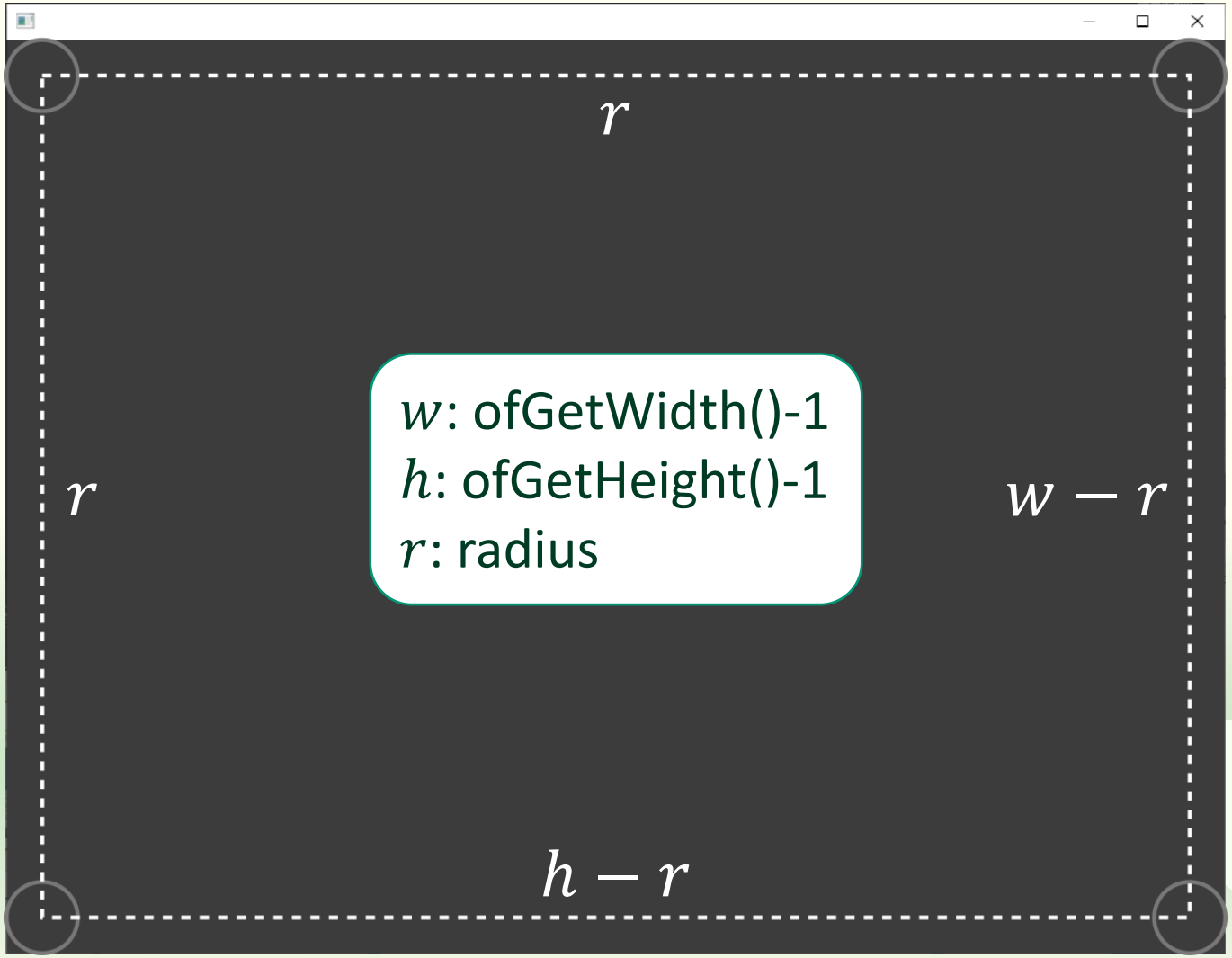
```
if (条件) {
  <条件が true なら実行>
}

if (条件) {
  <条件が true なら実行>
}
else {
  <条件が true でないなら実行>
}

if (条件1) {
  <条件1が true なら実行>
}
else if (条件2) {
  <条件1が true でなく条件2が true なら実行>
}
else {
  <条件1も条件2も true でないなら実行>
}
```

```
if (条件1) {
  <条件1が true なら実行>

  if (条件2) {
    <条件1が true で条件2も true なら実行>
  }
  else {
    <条件1が true で条件2が false なら実行>
  }
}
else {
  if (条件3) {
    <条件1が false で条件3が true なら実行>
  }
  else {
    <条件1が false で条件3が false なら実行>
  }
}
```

ソースプログラムの式と対応する数式

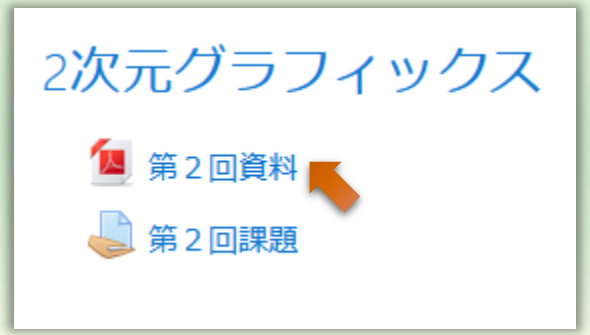
- `const float t{ (wr - p.x) / velocity.x };`
  - $t = \frac{w-r-x}{v_x}$
- `vec2 q{ wr, p.y + velocity.y * t };`
  - $\mathbf{q} = (w - r, y + v_y t)$
- `velocity = vec2{ -velocity.x, velocity.y };`
  - $\mathbf{v}' = (-v_x, v_y)$
- `position = q + velocity * (dt - t);`
  - $\mathbf{r} = \mathbf{q} + \mathbf{v}'(\Delta t - t)$

他の壁に対しても跳ね返るようにしなさい



動画キャプチャのアップロード

- 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、**2-2.mp4** というファイル名で Moodle の第２回課題にアップロードしてください
  - 画面上の動画は「Windows キー」＋「G」でキャプチャできます（macOS では QuickTime Player の「新規画面収録」が使えます）
  - 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 2-2.png というファイル名でアップロードしてください



課題２－３

発射位置を指定する

マウスをクリックした位置から発射する

- void ofApp::mousePressed(int x, int y, int button)
  - マウスのボタンを押したときに呼ばれる
  - x, y にはマウスのボタンを押した位置が入っている
  - button は押されたボタンの番号で 0:左, 1:中央, 2:右
- 速度 velocity の**初期値**は 0 にしておく
- マウスボタンを押したとき
  - マウスボタンを押した位置を円の位置 position に設定する
  - 速度 velocity に 0 でない値を設定する

課題２－４

発射方向を指定する

マウスボタンを離したときに発射する

- void ofApp::mouseReleased(int x, int y, int button)
  - マウスのボタンを離したときに呼ばれる
  - x, y にはマウスのボタンを押した位置が入っている
  - button は押していたボタンの番号で 0:左, 1:中央, 2:右
- マウスボタンを押したとき
  - マウスボタンを押した位置を覚えておく
  - 速度 velocity を **0 に設定**する
- マウスボタンを離したとき
  - マウスボタンを離した位置から押した位置に向かうベクトルを速度に設定する（定数倍してもよい）

マウスのドラッグ中にマウスで円を動かす

- void ofApp::mouseDragged(int x, int y, int button)
  - マウスをドラッグすると呼ばれる
  - x, y はマウスを移動した先の位置が入る
  - button は押されているボタンの番号で 0:左, 1:中央, 2:右
- ドラッグ中のマウスの位置 x, y を position に設定すると円がマウスに追従して動く
  - このときマウスの速度 velocity が 0 になってないとマウスを止めたときに円が勝手にマウスカーソルから離れていく

課題２－５

重力を与えてみる

重力

- ウィンドウの下の方向（y の正の方向）に重力がかかっているとして円の位置を更新しなさい
  - 重力加速度を $\mathbf{g} = (0, 200)$ とする
- 速度は加速度によって変化する
  - Δt 後の速度を $\mathbf{v}' = \mathbf{v} + \mathbf{g}\Delta t$ により求める
  - これはマウスボタンを**押していないときだけ**実行する
- この速度を使って位置を更新する
  - Δt 後の位置を $\mathbf{p}' = \mathbf{p} + \mathbf{v}'\Delta t$ により求める

bool ofGetMousePressed(int button)

- マウスボタンが押されている間 true を返す

```
if (ofGetMousePressed()) {
  // いずれかのボタンを押していればここの処理が実行される
}
```

```
if (!ofGetMousePressed(2)) {
  // 右ボタンを押して**いなければ**ここの処理が実行される
  // 0: 左，1: 中央，2 右
}
```

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、 **2-5.mp4** というファイル名で Moodle の第２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 2-5.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２回課題にアップロードしてください

時間の余った人向け課題

複数の円を扱えるようにしてください

マウスをクリックするたびに円を増やす

発展課題

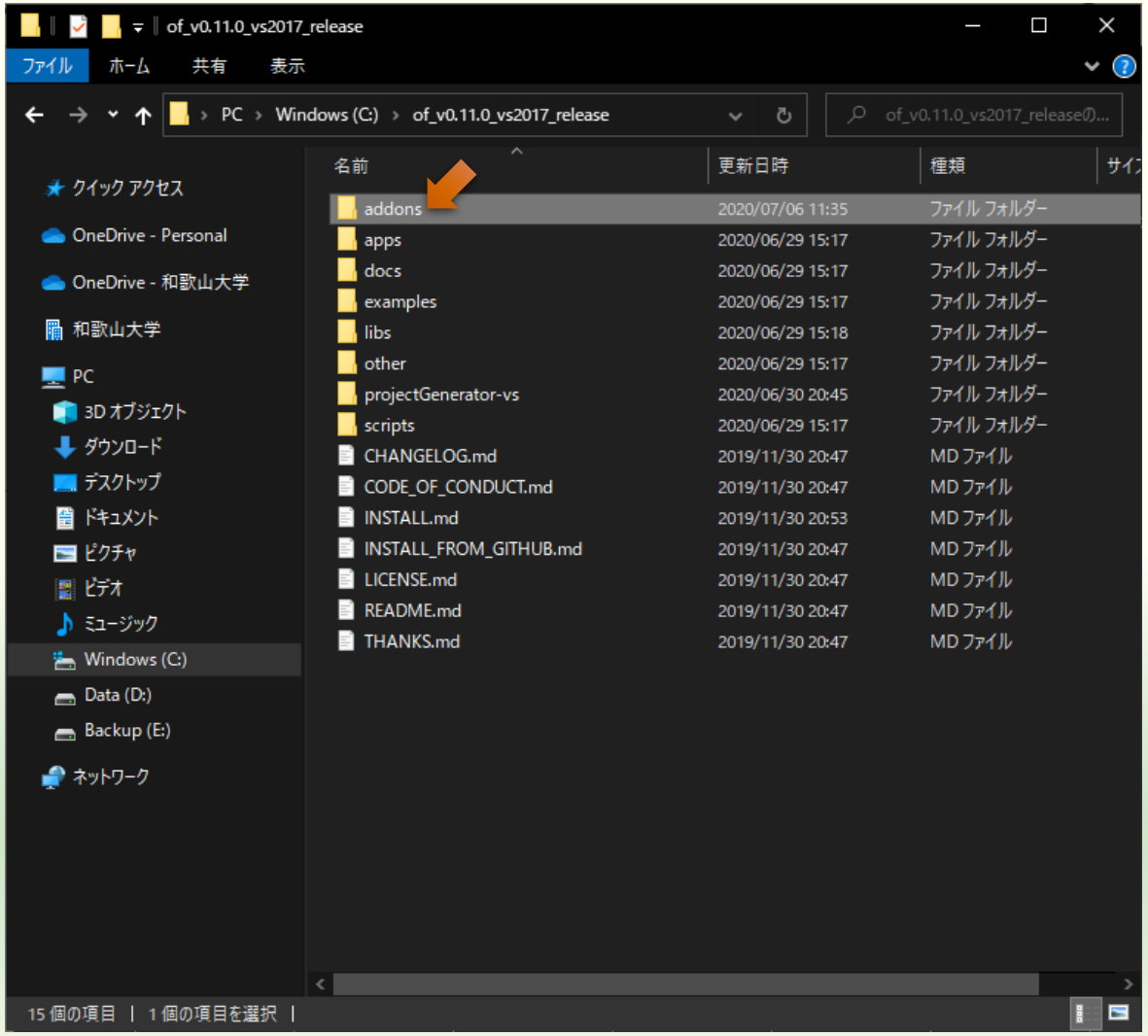
ofxBox2d を使うと楽だけど openFrameworks のバージョン 11 にはまだ対応していないみたいなのでやらなくていいです（対応方法は調査中）

ofxBox2d

- openFrameworks で使える 2 次元の物理エンジン
  - https://github.com/vanderlin/ofxBox2d
  - Box2d (https://box2d.org/) という 2 次元の物理エンジンを openFrameworks で使えるようにしたもの
- これを使うと今日のような課題が楽にできる
  - 多数の物体を扱うことができる
  - 物体同士の衝突も扱うことができる
  - 円以外も扱うことができる
  - 回転も考慮できる

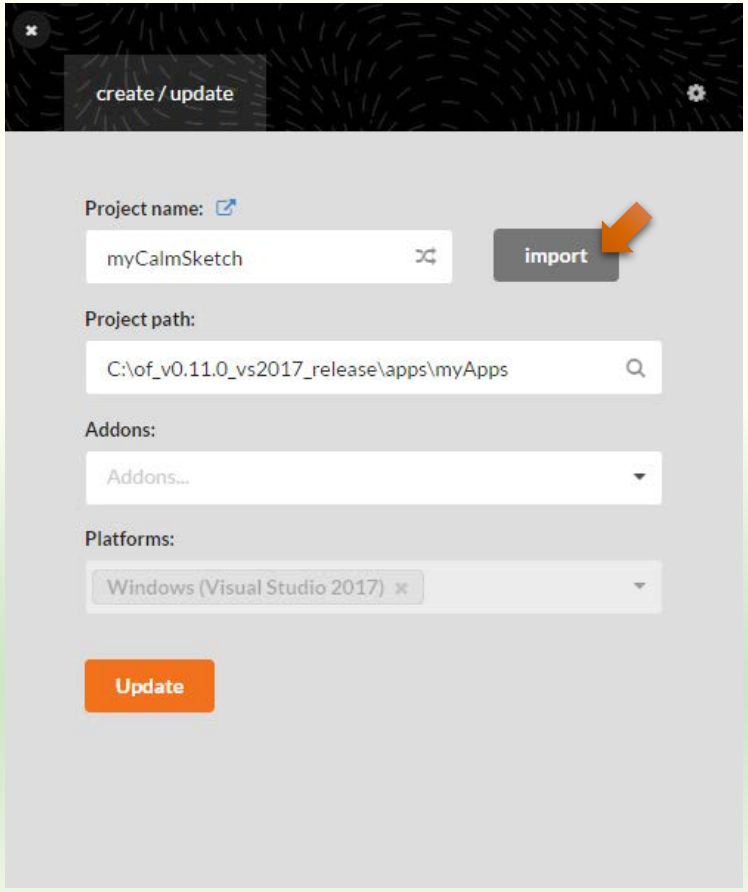
インストール

- https://github.com/vanderlin/ofxBox2d/ をダウンロード
- 展開したフォルダを <openframeworks の展開先>/addons に **ofxBox2d** というフォルダ名で配置

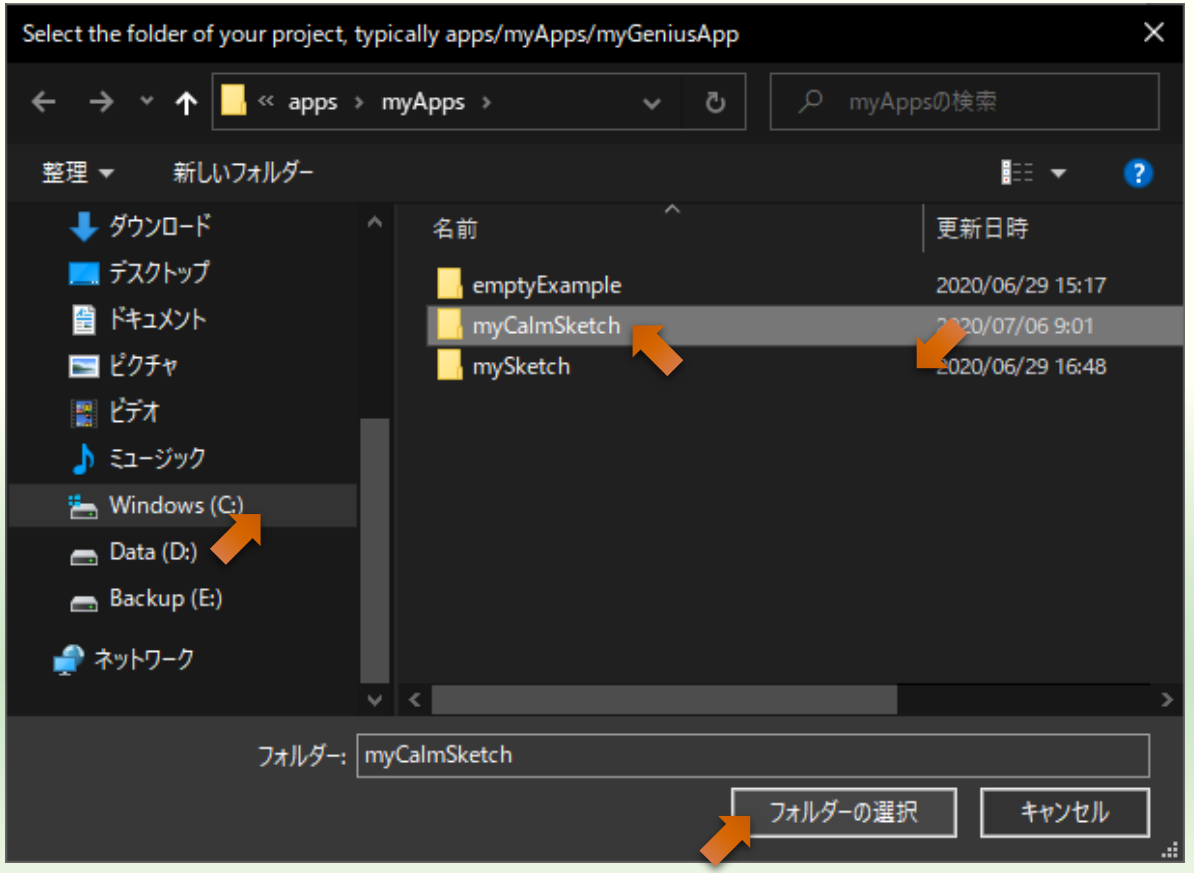


projectBuilder にプロジェクトを読み込む

import をクリック

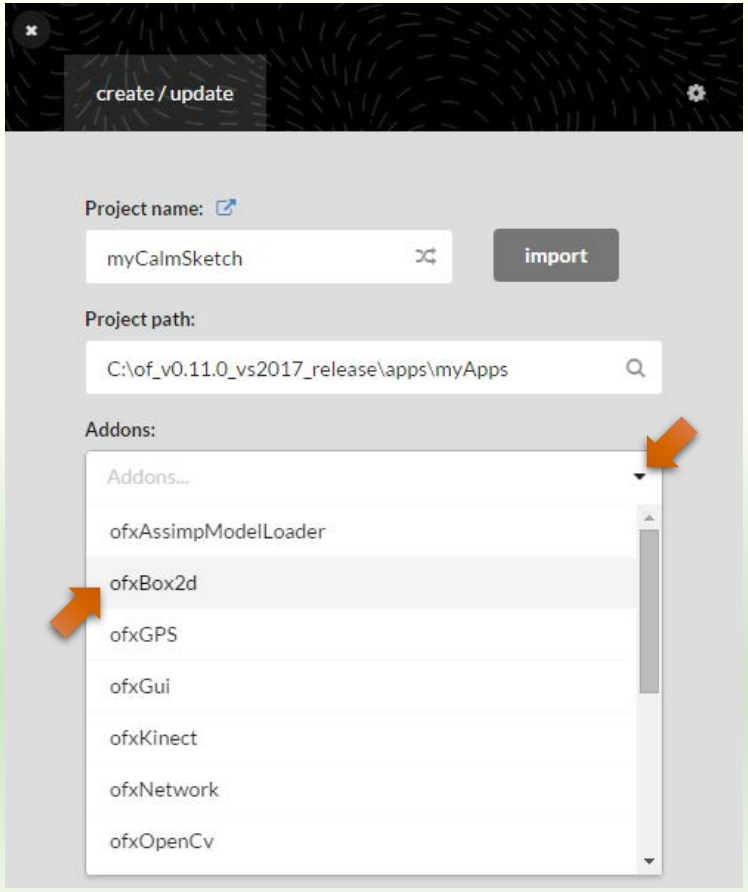


プロジェクトのフォルダを選択

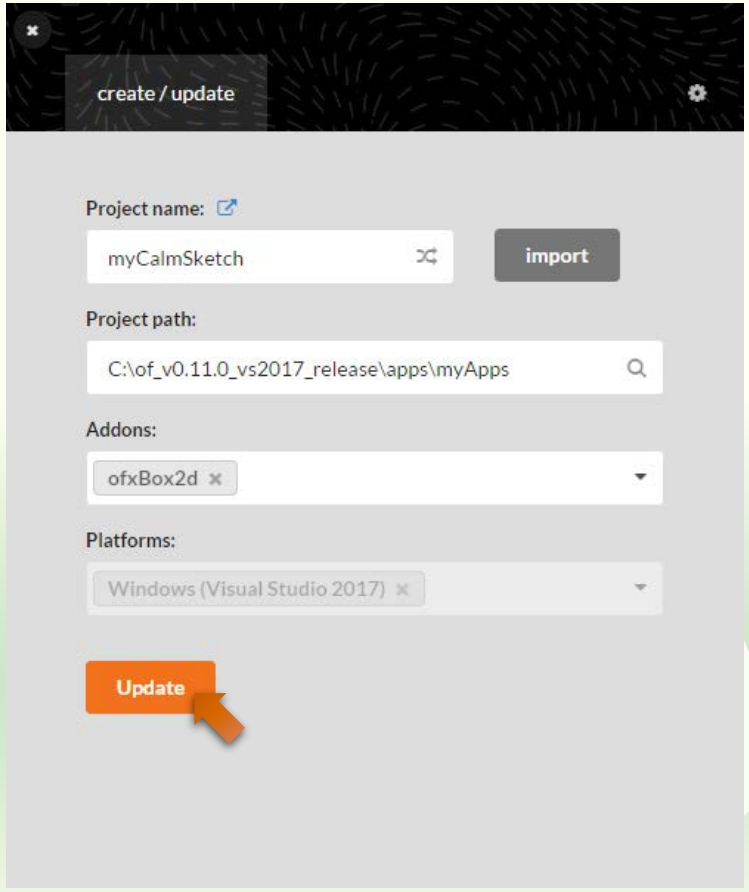


ofxBox2d をプロジェクトに追加する

Addons の中から ofxBox2d を選択



Update してプロジェクトを更新



ofApp.h の変更点

- メンバ変数に以下の内容を追加する

```
#pragma once

#include "ofMain.h"
#include "ofxBox2d.h"

class ofApp : public ofBaseApp{
  ofxBox2d box2d;
  vector<shared_ptr<ofxBox2dCircle>> circles;
```

ofApp.cpp

- setup() で円を生成する

```
// 円を１個生成する
auto circle{ std::make_shared<ofxBox2dCircle>() };

// 生成した円にパラメータを設定して circles に追加する
circle->setPhysics(3.0, 0.53, 0.1);
circle->setup(box2d.getWorld(), 100, 100, 10);
circles.push_back(circle);
```

ofApp.cpp

- draw() で円を描画する

```
    // circles のすべての circle について
    for (auto &circle : circles){

    // 円を 1 個描画する
    circle->draw();
}
```