コンピュータアニメーション特論
Visual Studio による C/C++ 開発環境の設定方法

九州工業大学 情報工学研究院　尾下真樹

本資料について

本資料では、本科目のプログラミング演習を行うために必要な、Windows 10/11 での、Visual Studio 2022 を使った C/C++ 開発環境の設定方法を説明する。

1　Visual Studio のインストールと設定

1.1　インストール

Windows 10 /11 環境に、Visual Studio 2022 をインストールする。Visual Studio には複数のエディションがあるが、Community / Professional / Enterprise のどのエディションをインストールしても構わない。ただし、Visual Studio Code は、C/C++アプリケーションの開発には機能が不足するため、推奨しない。

Visual Studio Community は、マイクロソフトのウェブサイト（https://visualstudio.microsoft.com/ja/）から、無料でダウンロードして利用できる。Visual Studio Professional / Enterprise は、大学がマイクロソフトと契約しているライセンスを使って、インストールできる。基本的な勉強・研究のためのプログラミングを行う場合は、いずれのエディションでも違いはないので、無料の Visual Studio Community を使用して構わない。

インストールを行う際には、C/C++以外の言語・環境を使用しない場合は、「C++によるデスクトップ開発」の基本機能のみを選択してインストールすれば良い（図 1）。他の機能が必要になったときには、後から追加でインストールできる。

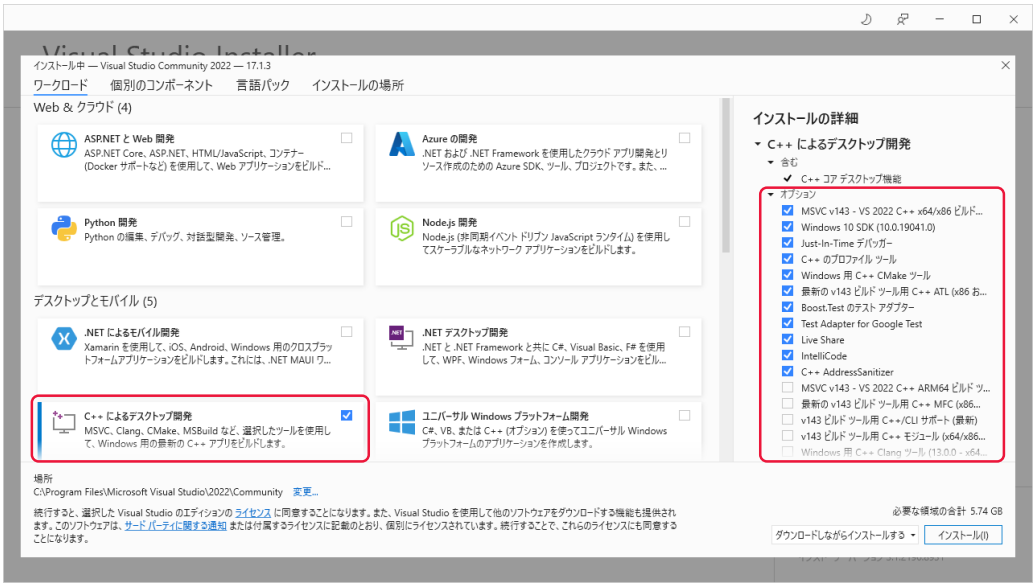

図 1: Visual Studio のインストール画面

1.2　自動書式変更の無効

Visual Studio では、初期設定状態では、設定された書式に従って自動的にコードの書式（インデント・改行・スペースの有無など）が整形されるようになっている。この機能は邪魔になることが多いため、Visual Studio の設定を変更して、この機能を無効にすることを推奨する。

1. メニューの ツール → オプション を実行して、設定画面を表示する。
2. テキストエディター → C/C++ → コードスタイル → 書式設定 にある、オートフォーマット関連の項目を全て無効にする（図 2）。

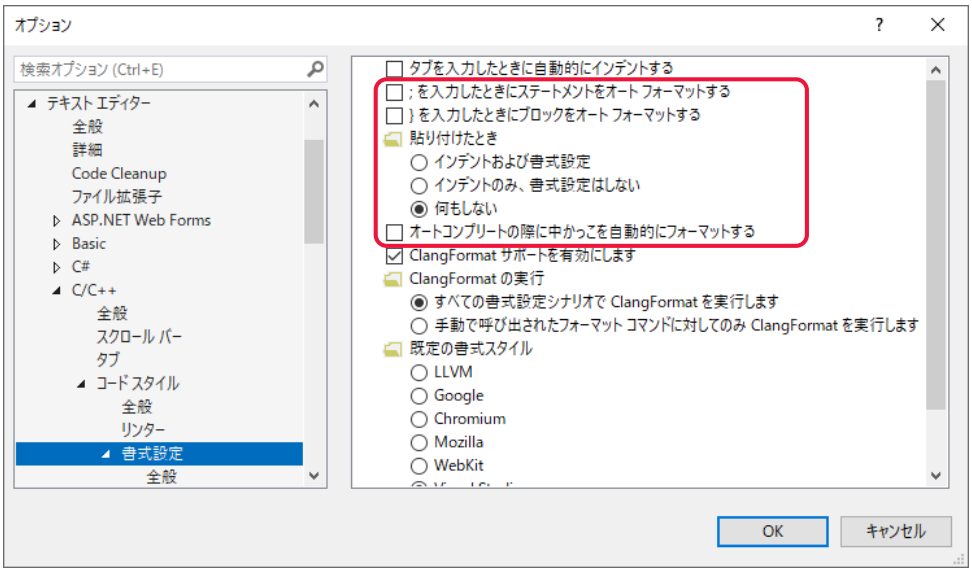

図 2: Visual Studio のオートフォーマットの設定

1.3　インクルード・ライブラリ ディレクトリの設定（古い情報）

本節で説明する機能は、最新版（2023 年 4 月時点）の Visual Studio 2022 では利用できなくなっているため、古い情報となる。最新の Visual Studio では、3.4 節の説明にもとづいて、プロジェクトごとにインクルード・ライブラリディレクトリの設定を行うことが基本となる。代わりになる機能として、複数のプロジェクトで設定を共有するためのローカルプロパティシートの機能が利用できるので、この機能を利用しても良い（https://learn.microsoft.com/ja-jp/cpp/build/create-reusable-property-configurations）。また、古いバージョンの Visual Studio を使用する場合は（基本的には最新版の Visual Studio を使用することを推奨する）、本節の説明にもとづいて設定を行っても良い。

何らかの外部ライブラリを使用する場合、ライブラリのヘッダファイルやライブラリファイルを読み込むディレクトリを設定する必要がある。それぞれのプロジェクトに個別に設定することもできるが、Visual Studio の設定に追加しておくことで、プロジェクト毎に設定しなくとも良くなる。

インクルード・ライブラリ ディレクトリの設定は、以下の手順で行える。

1. 任意のプロジェクトを開いて、プロパティマネージャから、「Microsoft.Cpp.Win32.user」を開く（図 3）。
2. 共通プロパティの VC++ ディレクトリの設定を編集して、「インクルード ディレクトリ」や「ライブラリ ディレクトリ」にパスを追加する（図 3）。

3. 変更したら、「Microsoft.Cpp.Win32.user」を保存して、変更が反映されるようにする。
   使用する外部ライブラリのヘッダファイルが置かれているディレクトリを「インクルード ディレクトリ」に追加し、ライブラリファイルが置かれているディレクトリを「ライブラリ ディレクトリ」に追加する。

4. 上記の 1〜3 は、32 ビット（x86）環境の設定方法である。
   64 ビット（x64）環境でも開発を行う場合は、同様に、「Microsoft.Cpp.Win64.user」も設定する。
   通常のライブラリは、32 ビット（x86）環境用と 64 ビット（x64）環境用に、別のライブラリファイルが提供されているため、環境に応じたライブラリディレクトリを設定する。

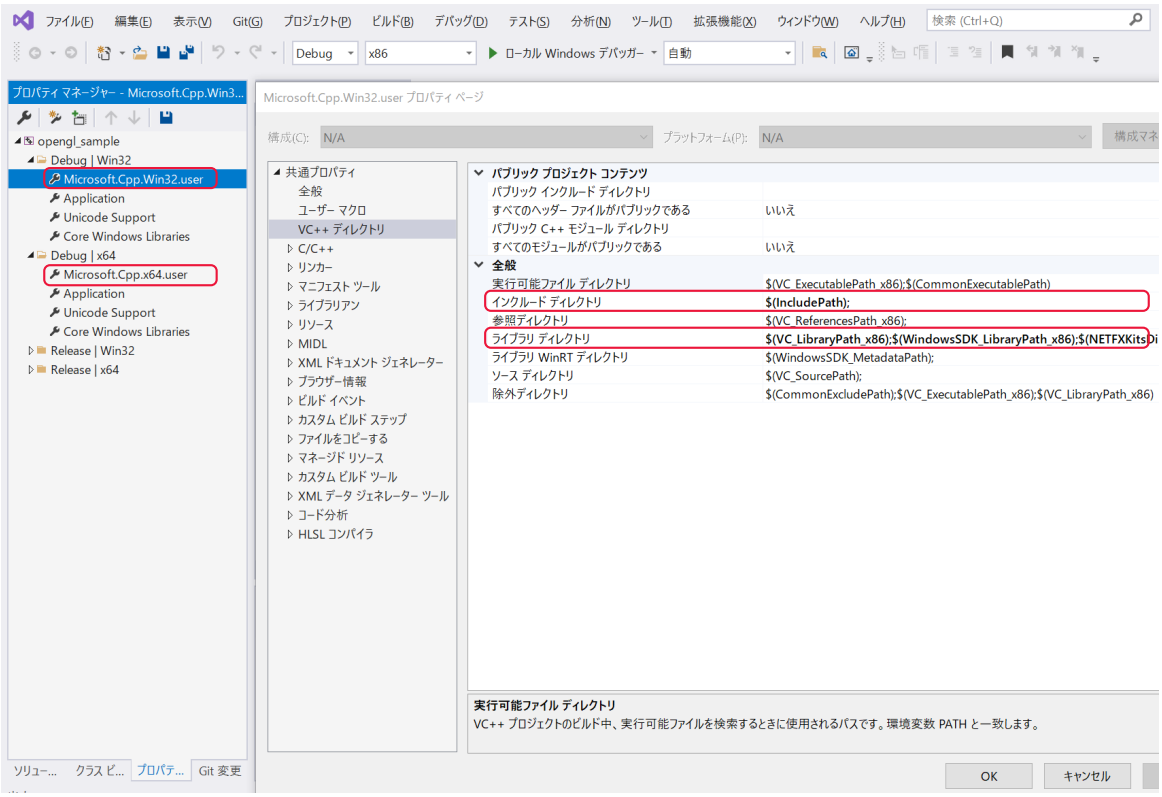

図 3: Visual Studio のディレクトリの設定

2　ライブラリのインストール

本科目のプログラミング演習を行うために必要な外部ライブラリ（fgeeglut、vecmath）の設定方法を説明する。

2.1　freeglut のインストール

freeglut は、GLUT の互換・拡張版ライブラリである。OpenGL を使ったアプリケーションを開発するとき、ウィンドウや入出力の処理に GULT を用いることができるが、GLUT は開発が終了しているため、freeglut を利用する。

freeglut は、静的リンク版と動的リンク版の両方のライブラリが利用できるが、静的リンク版を利用することを推奨する。（動的リンクを使用する場合、コンパイル時に使用したライブラリの DLL ファイルがなければ、プログラムを実行できないため。）

freeglut のインストール方法として、本ページで公開しているコンパイル済みライブラリを使う方法と、自分でライブラリのコンパイルを行う方法の、２つの方法を説明する。特に問題がなければ、前者の方法を使う方が簡単なので、前者の方法を推奨する。

2.1.1 方法 1：コンパイル済みのライブラリをコピーする方法

本ページで公開しているコンパイル済みの freeglut のアーカイブ（freeglut-3.2.1-bin.zip）を展開して、適当なディレクトリにコピーし、1.3 節の説明に従って、インクルード・ライブラリ ディレクトリを設定する。

例えば、コンパイル済みの freeglut を C:/library/freeglut に展開したとすると、以下のようにインクルード・ライブラリ ディレクトリを設定する。

- インクルード ディレクトリ（32 ビット環境・64 ビット環境共通）：C:/library/freeglut/include
- ライブラリ ディレクトリ（32 ビット環境）：C:/library/freeglut/lib/x86
- ライブラリ ディレクトリ（64 ビット環境）：C:/library/freeglut/lib/x64

2.1.2 方法２：自分でライブラリをコンパイルする方法

freeglut のウェブページ（http://freeglut.sourceforge.net/）から、ソースファイルをダウンロードして、自分でコンパイルすることもできる。コンパイルの方法は、和歌山大学 床井先生の OpenGL 入門のページのページ（https://tokoik.github.io/opengl/libglut.html#2.3） の説明が参考になる。

ビルドしたライブラリは、上記のページの説明の通り、Visual Studio のディレクトリにコピーしても良いし（ただし、上記のページは古いバージョンの Visual Studio に合わせたものなので、最新版に合わせてディレクトリを読み替える必要がある）、別のディレクトリにコピーして、インクルード・ライブラリ ディレクトリを設定しても構わない。

後者の方法を用いる場合、例えば、freeglut を C:/library/freeglut に置くとすると、以下の通りにファイルをコピーした上で、2.1.1 節の説明に従って、インクルード・ライブラリ ディレクトリを設定する。

- freeglut ディレクトの /include の下のファイルを全て、C:/library/freeglut/include にコピーする。
- freeglut ディレクトの /build32/Debug と /build32/Release の下のファイルを全て、C:/library/freeglut/lib/x86 にコピーする。
- freeglut ディレクトの /build/Debug と /build/Release の下のファイルを全て、C:/library/freeglut/lib/x64 にコピーする。

また、常に静的リンク版のライブラリを使用するように、glut.h を変更して、

```
#include "freeglut_std.h"
```

の呼び出しの前に、

```
#define FREEGLUT_STATIC
```

の定義を追加する。

2.2 vecmath のインストール

vecmath は、3 次元グラフィックスを扱うために必要となるベクトル・行列・四元数などのデータ構造や演算を扱うライブラリである。元来は Java のライブラリであるが、C++版の非公式ライブラリを利用する。

vecmath C++ は、vecmath C++ のウェブページ（http://objectclub.jp/download/vecmath1）からダウンロードできる。vecmath C++ は、テンプレートライブラリであるため、ライブラリをリンクする必要はなく、ヘッダファイルをインクルードするだけで利用できる。

例えば、vecmath を C:/library/vecmath に展開したとすると、以下のようにインクルード・ライブラリ ディレクトリを設定する。

- インクルード ディレクトリ（Win32・Win64 共通）：C:/library/vecmath

なお、vecmath C++ は、一部に古い仕様に準拠したコードを含むため、Visual Studio の標準のプロジェクト設定では、コンパイルすることができない。プロジェクトの設定から、C/C++ → 言語 と選択して、準拠モードを「いいえ」に変更することで、コンパイルできるようになる。

3　プロジェクトの作成と設定

Visual Studio で、プロジェクトを作成する方法を説明する。Visual Studio でソースファイルをコンパイルするためには、ソリューションやプロジェクトを作成する必要がある。プロジェクトとは、一つまたは複数のソースファイルをコンパイル・リンクして、実行ファイル（またはライブラリ等のファイル）をビルドするための設定をまとめたものである。また、ソリューションは、一つまたは複数のプロジェクトをまとめたものである。ソリューションは拡張子が sln のファイルとして作成され、プロジェクトは拡張子が vcxproj の複数のファイルとして作成される。一つのソースファイルをコンパイルする場合でも、少なくとも一組のソリューションとプロジェクトを作成する必要がある。

3.1　プロジェクトの作成

以下の手順で、プロジェクトとソリューションを作成する。

1. Visual Studio の起動時に表示されるメニュー画面から、「新しいプロジェクトの作成」を選択して、プロジェクトの作成を開始する。もしくは、Visual Studio の起動後に、メニューから、ファイル → 新規作成 → プロジェクト と選択しても良い。

2. OpenGL ＋ GLUT を使ったプログラムを作成するときには、「コンソールアプリ」を選択して、新しいプロジェクトを作成する（図 4）。

3. プロジェクト名には適当な名前、場所にはプロジェクトを作成するディレクトリを指定する（図 5）。
   単一のプロジェクトを作成する場合は、「ソリューションとプロジェクトを同じディレクトリに配置する」のオプションを付けておいた方が、ディレクトリ階層が複雑にならずに良い。

4. プロジェクトが作成されたら、プロジェクトにソースファイルを追加し、必要に応じて設定を変更する。

図 4: Visual Studio のプロジェクトの作成

図 5: Visual Studio のプロジェクトの構成

3.2 既存のプロジェクトのリターゲット

既存のプロジェクトが存在する場合は、プロジェクトを開いて利用できるが、プロジェクトが作成された環境と自分の環境が異なる場合は、プロジェクトのリターゲットが必要になる。

そのようなプロジェクトを Visual Studio で開くと、自動的にリターゲットのダイアログが表示されるため、自分の環境を選択して進めると、リターゲットが行われる（図 6）。Windows SDK バージョンは、Visual Studio で使用するライブラリのバージョンを表す。プラットフォームツールセットは、Visual Studio 自体のバージョンを表す。

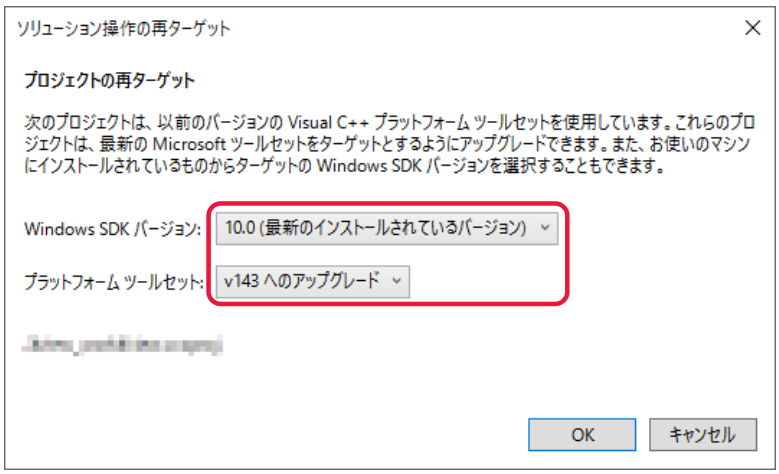

図 6: Visual Studio のプロジェクトの構成・プラットフォーム

プロジェクトを開いた後にリターゲットを行うこともできる。Windows SDK バージョンを変更する場合は、ソリューションエクスプローラーでプロジェクトを右クリックして表示されるメニューから、プロジェクトの再ター

ゲットを実行する。プラットフォームツールセットを変更する場合は、プロジェクトの設定を開いて、構成プロパティ → 全般 → プラットフォームツールセットの設定を変更する。

3.3 プロジェクトのプラットフォーム・構成

Visual Studio を使って C/C++ のプログラムを開発する際に注意が必要になる点として、作成されたプロジェクトは、複数のプラットフォーム・構成を含むため、これらの使い分けや、各プラットフォーム・構成に合わせた設定が必要になる。

通常、プラットフォームとして、32 ビット環境の「Win32 (x86)」と 64 ビット環境の「x64」 の 2 つが自動的に生成される。また、各プラットフォームに対して、デバッグ用の「Debug」と、実行時用・公開用の「Release」の 2 つの構成が自動的に生成される。

プラットフォームは、64 ビット環境のみで動作するプログラムを開発するか、古い 32 ビット環境でも動作するプログラムを開発するかによって、両者を使い分ける。外部ライブラリも全て、使用する環境に合わせる必要がある。実行環境や外部ライブラリに特に制約がなければ、64 ビット環境を利用することを推奨する。

構成は、デバッグ時と実行時用・公開用で、両者を使い分ける。「Debug」構成では、コンパイル時にデバッグに必要なコードを含むため、プログラムのサイズが大きくなり、実行速度も遅くなるが、デバッグモードでの実行が可能となる（ステップ実行や実行中の変数の値の確認が行える）。一方、「Release」構成では、デバッグの機能は使えない代わりに、高速にプログラムを実行できる。

コンパイルに使用する構成・プラットフォームは、Visual Studio のメニューバーから切り替えることができる（図 7）。

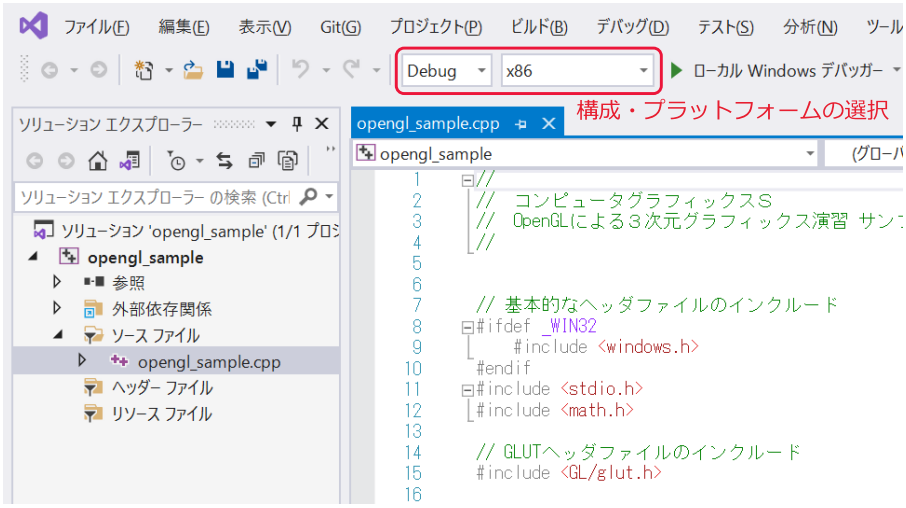

図 7: Visual Studio のプロジェクトの構成・プラットフォーム

3.4 プロジェクトの設定

必要に応じて、プロジェクトの各プラットフォーム・構成の設定を変更する。以下に、変更が必要になる可能性がある項目をいくつか説明する。

- インクルード・ライブラリ ディレクトリの設定
  freeglut や vecmath などの外部ライブラリを用いる場合は、これらの外部ライブラリのファイルが置かれているディレクトリを、インクルード・ライブラリ ディレクトリとして設定する。
  プロジェクトの設定の 「VC++ ディレクトリ」から、インクルードディレクトリ、ライブラリディレクトリを設定できる。1.3 節の説明に従って、別の方法でインクルード・ライブラリ ディレクトリの設定を行っている場合は、この設定は省略できる。

- 作業ディレクトリや出力ディレクトリの設定
  何らかの外部ファイルを読み込むプログラムを開発する場合は、プロジェクトの作業ディレクトリや出力ディレクトリを、読み込むファイルが置かれているディレクトリに設定する。
  プロジェクトの設定の 「全般」 で出力ディレクトリ、「デバッグ」 で作業ディレクトリを設定できる。

- コンソールを表示しないようにする設定
  3.2 節の説明の通りに「コンソールアプリ」を選択してプロジェクトを作成すると、プログラムの実行時に GLUT のウィンドウに加えて、コンソール画面が表示される。コンソール画面は、デバッグ用の情報を表示したりするために使うことができるが、そういった情報の表示が必要ない場合は、邪魔になる。以下のように設定すると、コンソールを表示しないように設定することができる。

  – リンカ設定 → システム のサブシステムを、Window に変更する。
  – リンカ設定 → 詳細設定 のエントリポイントを、mainCRTStartup に変更する。

  例えば、Release 構成のみ上記のように設定をしておけば、Debug 構成を使用するときのみコンソールを表示して、Release 構成を使用するときにはコンソールを表示しないようにできる。

4 開発環境の確認

正しく開発環境を設定し、プロジェクトの作成・設定方法を理解したことを確認するために、以下の確認を行う。

1. OpenGL 基礎プログラミングのサンプルプログラム（opengl_sample.cpp）のコンパイル
   本科目のウェブページで公開されている上記のファイルをダウンロードし、プロジェクトの作成・設定を行って、コンパイル・実行ができることを確認する。

2. キャラクタ・アニメーションのサンプルプログラム（human_sample.zip）のコンパイル
   本科目のウェブページで公開されている上記のファイルをダウンロードし、あらかじめ用意されているプロジェクトを使って、コンパイル・実行ができることを確認する。