Australian National University

COMP4610/8610
Computer Graphics

Computer Lab Homework Assignment #1, S1 2024

Topic: Environment preparation and basic geometry transformations.

Date Issued: see Wattle page
Due Date: see Wattle page

Weighting: 12%

Instruction:

All homework assignments must be completed individually.

We encourage you to discuss the assignments with other students. However, you should not share any of your codes with anyone else. Each student is responsible for implementing the assignment on their own. You may assist other in debugging their codes, but you should not copy and paste. ANU is using Turnitin to detect possible duplications. Consulting with previous year students who enrolled in this course on specific assignment is also not allowed. You may use the internet as a resource for learning the materials, but you should not borrow any existing codes found online.

The homework assignments involve a significant amount of C/C++ programming. However, for most cases, a skeletal code base is provided, and you only need to fill in the missing parts, and/or fix bugs if any.

You will submit a single ZIP file as your submission, which must contain the following files:

(1) All source codes (ending in .h, or .hpp, or .cpp), and CMakeLists.txt. Please include all needed source codes for successful compilation. Please also remove all intermediate files and folders (such as `.vscode/` and `build/`) that are not needed for the compilation – Failing to do so will lead to penalty to the marks.

(2) A written CLab1-Report (minimum 10-point font size, single column A4, in PDF format, with task statement, methods used, any new features that you have implemented, any known bugs in your code, answer any questions that have been asked in the task, instruction for the tutor to use your code, example experiment results.)

Your ZIP file must be named as "COMPX610_2024_HW1_UID.zip". Replace 'X' with 4 or 8. Replace the UID with your Uxxxxxxxx; Please submit your ZIP file to Wattle before the deadline. Late submission will lead to penalty as per the ANU policy. Later-than-one-week

submission will not be accepted, which may result zero mark, unless a pre-approval for special consideration is obtained in written before the submission deadline.

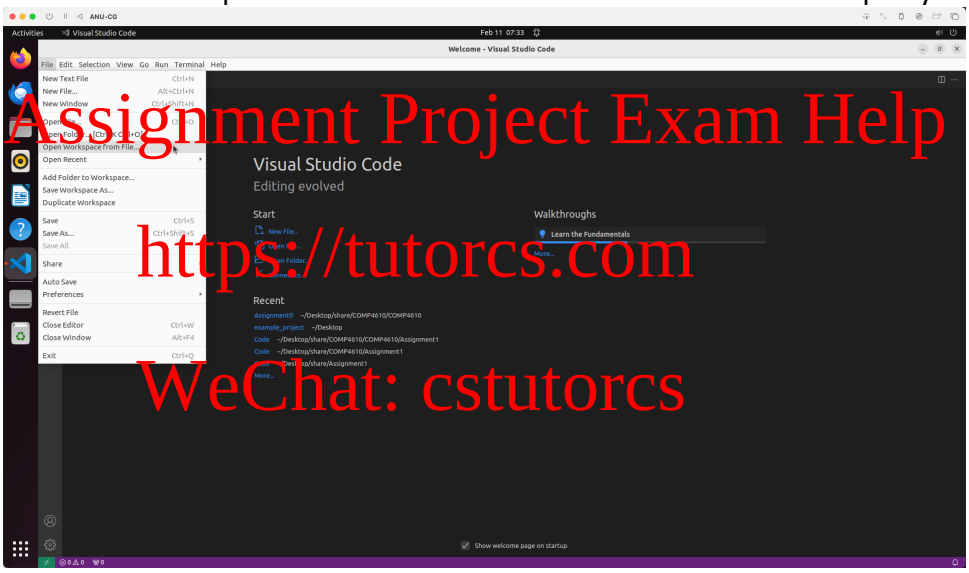
Tasks for HW1:

Task-1: C/C++ Programming Basics

We will introduce basic C/C++ programming knowledge including development environment, fundamental syntax and constructs of C/C++ project that will be used in the homework assignments. If you come across any further problems with C/C++, please refer to online resources C++ documentation — DevDocs, Stack Overflow – Where Developers Learn, Share, & Build Careers or post on Ed Forum.

1.1 Development Environment

We use Visual Studio Code (VSCode) as our IDE (Integrated Development Environment). It's already installed and set up in the virtual machine. You can use VSCode to open your project.



1.2 C++ Programming

1.2.1 Basic programming

First, you need to include essential header files at the front of the code.

```
// Include essential header files that will be used in this file.
#include <Eigen/Core> // Eigen is used for vector calculation
#include <Eigen/Dense>
#include <cmath> // Some useful math functions
#include <iostream>
```

Define a `main` function as the entry of your program.

```cpp
// The "main" function is the entry of a C/C++ program.
int main() {
```

Call some mathematic functions.

```cpp
// Basic Example of cpp
  std::cout << "Example of cpp \n";
  float a = 1.0, b = 2.0;
  std::cout << a << std::endl;
  std::cout << a / b << std::endl;
  std::cout << std::sqrt(b) << std::endl;
  std::cout << std::acos(-1) << std::endl;
  std::cout << std::sin(30.0 / 180.0 * acos(-1)) << std::endl;
```


1.2.2 Vector manipulation with `Eigen` library

`Eigen` is a C++ library for linear algebra, matrix and vector operations. Please refer to `use_eigen.cpp` for some common usage of `Eigen` and more information at Eigen: Quick reference guide.

1.3 Organize your project with CMake

CMake is an open-source, cross-platform family of tools designed to build, test and package software. CMake is used to control the software compilation process using simple platform and compiler independent configuration files and generate native makefiles and workspaces that can be used in the compiler environment of your choice.

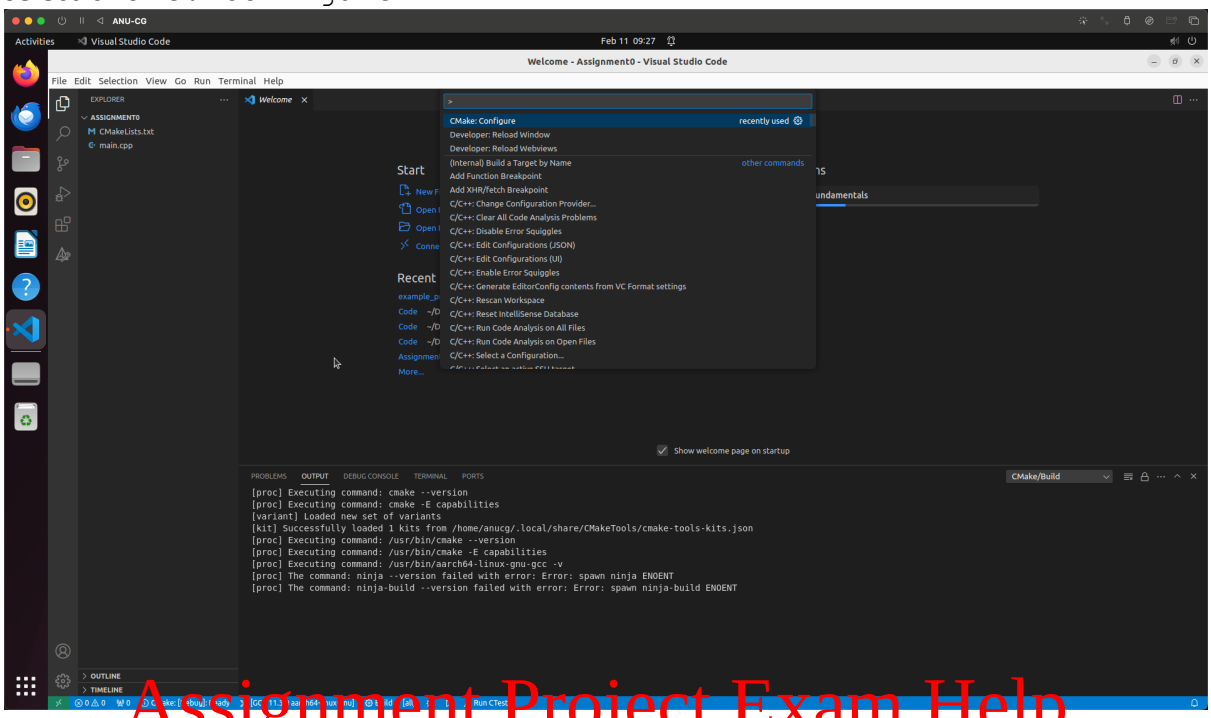
1.3.1 Writing a `CMakeLists.txt`

CMake uses a configuration file called CmakeLists.txt to organize a project. Below is a minimum example of CmakeLists.txt with detailed comments.

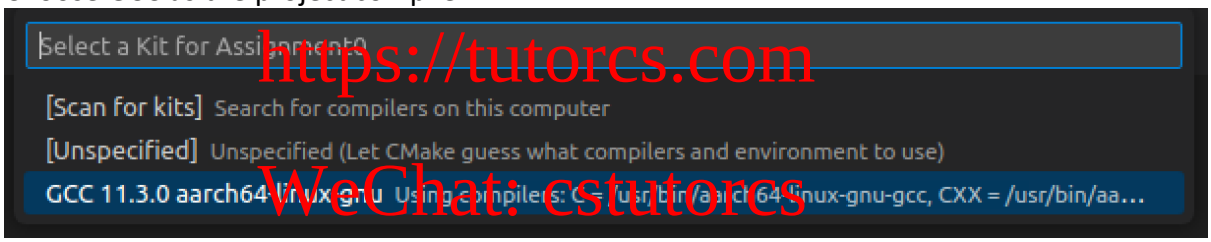
```cmake
# The version of CMake that this project was created with
cmake_minimum_required(VERSION 3.5)

# The name of this project is "Homework1"
project(Homework1)

# Find essential libraries in the system, such as Eigen3 and OpenCV
find_package(Eigen3 REQUIRED)
find_package(OpenCV REQUIRED)

# Include the directories of the libraries so that the compiler can find the header files
include_directories(${EIGEN3_INCLUDE_DIR} ${OpenCV_INCLUDE_DIRS})

# Add the executable file to the project, "hw1" is the name of the executable file, and "main.cpp" is the source file, you can also add more source files by "add_executable (hw1 main.cpp file1.cpp file2.cpp ...)"
add_executable(hw1 main.cpp)

# Link the libraries to the executable file, so that the compiler will link the libraries to the executable file
target_link_libraries(hw1 ${OpenCV_LIBS})
```

1.3.2 Configure with CMake

To use CMake in VSCode, press `CTRL + SHIFT + P` to open the Command Palette and select `CMake: Configure`.



Choose GCC as the project compiler.

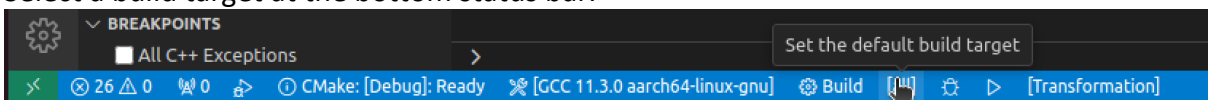


Then, VSCode will invoke CMake to setup the project according to the provided `CmakeLists.txt` and generate a `build` folder under the project root directory.
Build the project in the terminal by `cmake .. && make`:

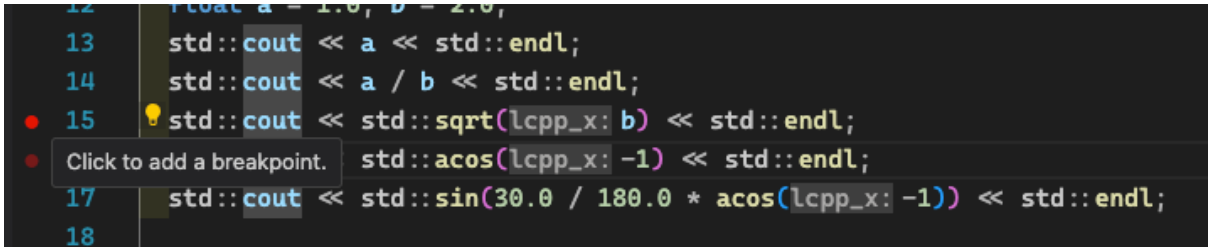
```
anucg@anucg-vm:~/Assignment0$ cd build
anucg@anucg-vm:~/Assignment0/build$ cmake ..
CMake Deprecation Warning at CMakeLists.txt:1 (cmake_minimum_required):
  Compatibility with CMake < 2.8.12 will be removed from a future version of
  CMake.

  Update the VERSION argument <min> value or use a ...<max> suffix to tell
  CMake that the project does not need compatibility with older versions.


-- The C compiler identification is GNU 11.4.0
-- The CXX compiler identification is GNU 11.4.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /usr/bin/cc - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /home/anucg/Assignment0/build
anucg@anucg-vm:~/Assignment0/build$ make
[ 50%] Building CXX object CMakeFiles/Transformation.dir/main.cpp.o
[100%] Linking CXX executable Transformation
[100%] Built target Transformation
```

1.4 Debug with VSCode

Select a build target at the bottom status bar.



Set a breakpoint at the target line by click or press `F9`.



Press `CTRL+F5` to start debugging.

1.5 Check if a point is inside a triangle:

In computer graphics, determining whether a point is inside a triangle on the plane is a very popular algorithm. You need to implement this algorithm according to the lecture in the provided code framework `task1.cpp`.

Task-2: OBJ Mesh file I/O and operations:

Get yourself familiar with Wavefront OBJ file format by reading the Wikipedia page: Wavefront .obj file – Wikipedia.

You need to complete the following two sub-tasks:

(a) Find the provided mesh `Cottage.obj` in the `model` folder and visualize it with `MeshLab`.

(b) Create a simple house mesh by hand (which means you can only create the mesh in a text editor without any 3D software) and save it as `house.obj` in the model folder. Basically, the house mesh should contain a cuboid base, a triangular prism roof and a chimney. Note that you only need to create the vertices and faces of the mesh.

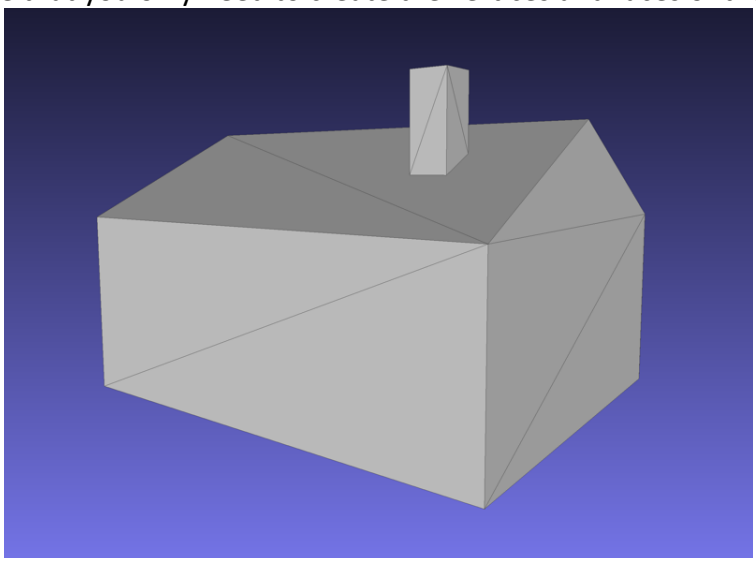


Figure: An example of a simple house mesh.

Task-3: Spatial Transformations for Graphics Rendering

From the first two weeks, you have learned how to use transformation matrices to prepare an object for graphics rendering. Now it is time to put them into practice. In this task, we will create a house mesh and display it in the screen. You need to finish the task by following steps:

(a) Implement the `get_model_matrix` function that calculates a transformation matrix given the rotation angle, translation and scale.

(b) Implement a `draw_circle` function according to the lecture slides.

(c) Currently, the camera is fixed and always looking towards the -z direction. Please update your code to enable camera movement by keyboard and ensure the camera is always looking at the origin. To do this, you may need to implement the `look_at` function and used it in the `get_view_matrix` function.

(d) Run your code to see the result.

Task-4: Spatial transformations

4.1. In order to perform geometric transformations of a rigid body in 3D space, we often use 4x4 homogeneous matrices. Write down any 4x4 matrix which represents a rigid Euclidean transformation and prove that your specific 4x4 matrix is indeed a rigid Euclidean transformation.

4.2. Write the homogeneous transformation matrix that rotates a point around the Y-axis by 90 degrees with the centre at point (2, 3, 4). Please list the steps of your calculation.

4.3. Given the 3D cube mesh `cube.obj` with 12 triangles. Use this cube as the primitive shape, write some C++ codes to model the following cubic dog using transformed primitive cuboids, and render it as wireframe. Report your rendering result in your Lab report.

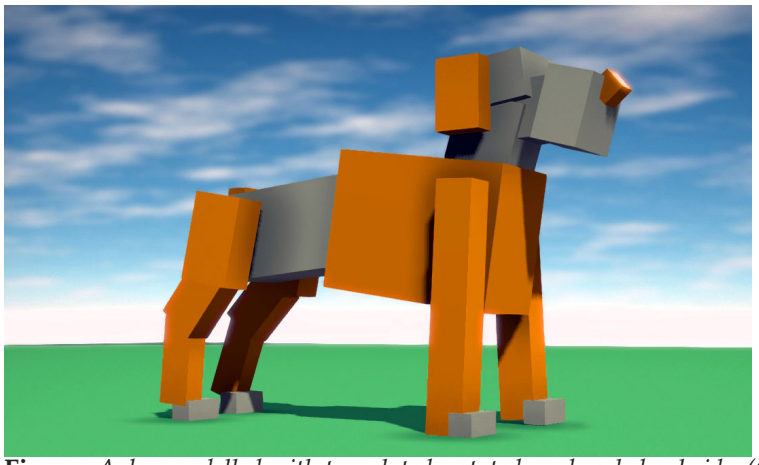


Figure: *A dog modelled with translated, rotated, and scaled cuboids. (Courtesy of* Morgan McGuire).

How to compile and run your codes?

Library dependencies: Eigen, and OpenCV;
Please use the following commands in order, to compile and test your code.

(1) mkdir build
(2) cd build
(3) cmake ..
(4) make -j4

Report Template and Marking Criteria:

Please use "CLab#1 report template" to write your Lab report. Please also note the "HW1 Marking Criteria" posted on Wattle.

Your ZIP file must be named as "COMPX610_2024_HW1_UID.zip".

== END OF CLAB-1 ==