CSC4140 Assignment II

Computer Graphics

February 16, 2022

Transformation

This assignment is 10% of the total mark.

Strict Due Date: 11:59PM, Feb 25^{th}, 2022

Student ID: 118010077

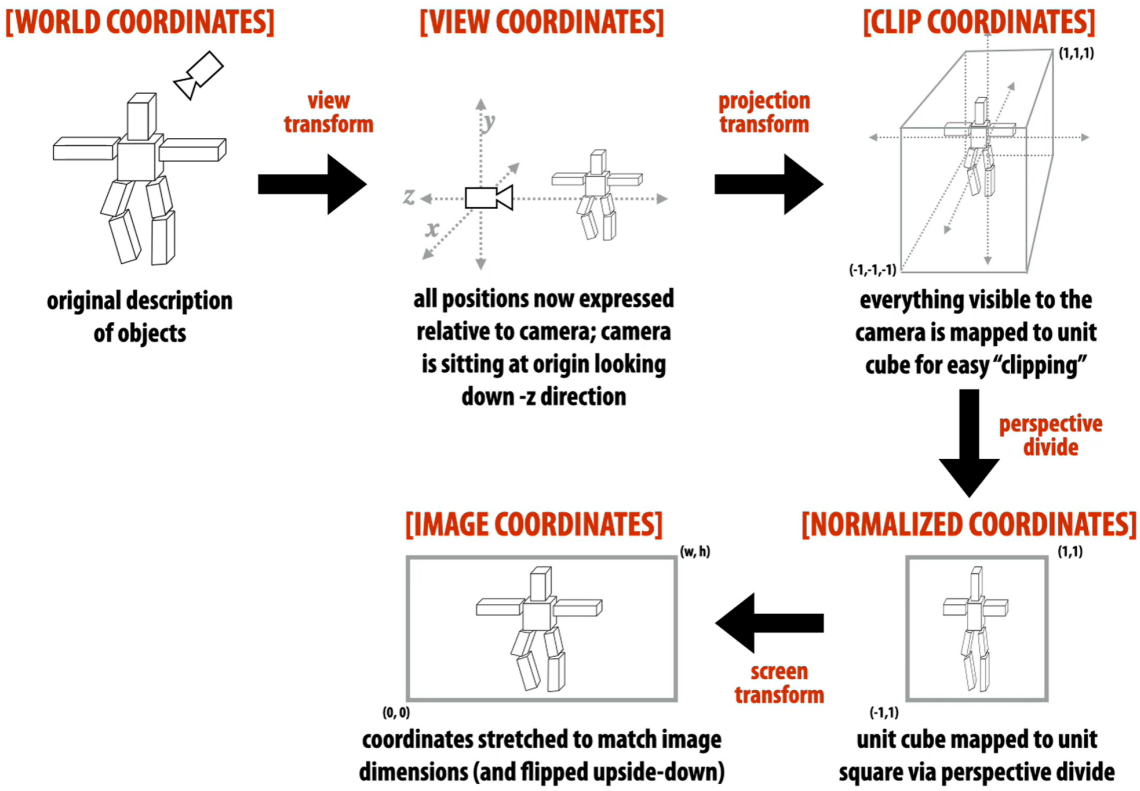
Student Name: 高梓骐

This assignment represents my own work in accordance with University regulations.

Signature: 

1. From Model to Screen

If we want to get a model to screen, we need to the following steps:



We need to conduct all kinds of transformation (translation, scale, rotation, coordinates transformation) to finish those steps.

With the help of the homogeneous coordinate, all of the transformation can be done through matrix multiplication.

1.1 Implement Model Matrix

This method conducts the basic transformation of the model in the world coordinate system (canonical coordinate system).

Here is the introduction to the parameters used in this method:

- rotation_angle: Rotate around an axis determined by other two parameters P0 and P1. **You can change the angle by key "a" mplify and key "d"ecrease**
- T: Translate by T[0], T[1], T[2] along x, y, z direction.
- S: Scale by S[0], S[1], S[2] along x, y, z direction.
- P0, P1: The vector P1 - P0 is the rotation axis.

Note: View Matrix (Eye_Pos)

We need to decide the coordinate of the "eye (camera)" in the world.

It is set to be *(100, 100, 200)* in this project.

And the view matrix is used to translate the coordinate of the model from the world system to the view system. That is, we use the basis of the camera to express the location of the model.

1.2 Implement Perspective Projection Matrix

This projection matrix consists of 2 parts.

In the first stage, we need to conduct the orthographic projection, to get a "cuboid" (view frustum).

In the second stage, we want to turn the cuboid to the canonical view cube called viewport transformation.

Here is the introduction to the parameters used in this method:

- eye_fov, this parameter decide what can be seen through the "eye". And have effect on the view frustum. **This is set to be 60 degrees.**
- aspect_ratio, this parameter also decides the shape of the frustum. **This is set to be 1 because over final image is 1024 * 1024**
- zNear, zFar: Two parameters about the view frustum.

1.3 Implement main() function according to your needs

We just need to set the parameters and conduct the above methods to get the MVP matrix, and we can do the rasterize.

2. Compilation and Demo

After you extract the zip file, you can run the compile_run.sh to compile and run the program.

Then press "a" to amplify the angle, and press "d" to decrease the angle.

Press "ESC" to exit the program.

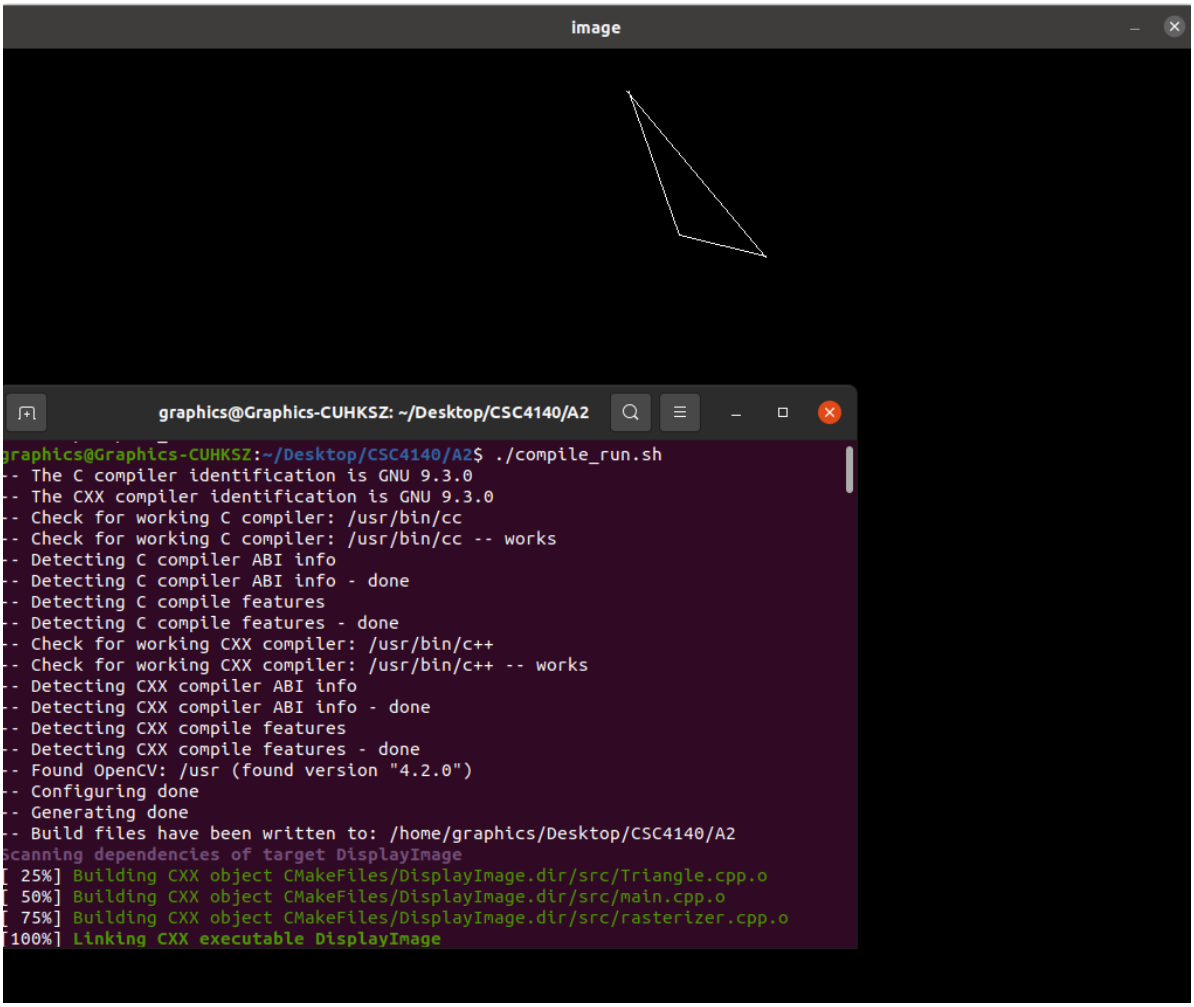


image
graphics@Graphics-CUHKSZ: ~/Desktop/CSC4140/A2
graphics@Graphics-CUHKSZ:~/Desktop/CSC4140/A2$ ./compile_run.sh
-- The C compiler identification is GNU 9.3.0
-- The CXX compiler identification is GNU 9.3.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Found OpenCV: /usr (found version "4.2.0")
-- Configuring done
-- Generating done
-- Build files have been written to: /home/graphics/Desktop/CSC4140/A2
Scanning dependencies of target DisplayImage
[25%] Building CXX object CMakeFiles/DisplayImage.dir/src/Triangle.cpp.o
[50%] Building CXX object CMakeFiles/DisplayImage.dir/src/main.cpp.o
[75%] Building CXX object CMakeFiles/DisplayImage.dir/src/rasterizer.cpp.o
[100%] Linking CXX executable DisplayImage