In this demo, I implemented a software raster renderer by using vs2019 and C++. I used CPU to simulate the process of drawing model data by GPU in 3D games.

#### Main functions:

1. 3D rendering pipeline is implemented in this demo:

model data -> model space -> world space (vertex lighting) -> view space (back face culling) ->clip space (clipping) -> screen space (assembly of pixels, interpolation and rasterization, pixel operation, write the pixel into backbuff-> draw the backbuff on the screen

2. Rendering in Wireframe render mode, texture render mode and vertex color mode

3. Perspective correction interpolation for texture uv coordinates and vertex colors, etc

4. Bilinear filter sampling, point sampling for texture

5. Back Face Culling

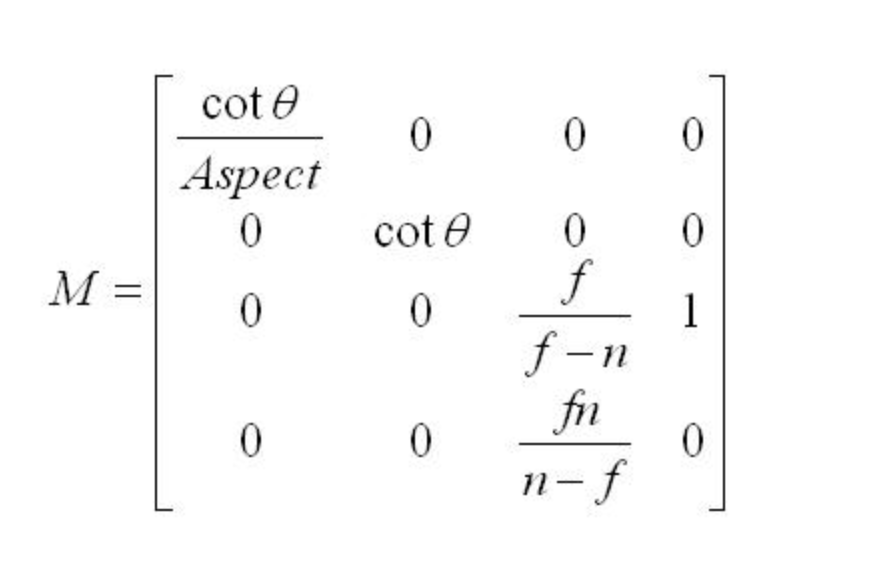
6. Clipping in CVV

7. Vertex lighting

#### Convention:

1. In this program, vertex index is used to organize vertex data, and the normals of triangle composed of vertices organized in counterclockwise are pointed out of the screen.

2. The projection matrix used in this program is Direct3D style as follow.



Parameters：

f far plane

n near plane

Aspect Aspect ratio of the camera

θ Observation Angle

The vertices in the view space multiplied by this matrix are then transformed into a homogeneous clipping space, and the w component of the vector holds z information in view space.

In the homogeneous clipping space we can clip vertices by using range as follow:

-w <= x <= w

-w<= y <= w

0 <= z <= w

In other words, vertices that are not in this range will be clipped and not rendered.

3. uv coordinate system is Direct3D style in the program.

#### Main files and functions of the program:

###### Header files :

**Camera.h** Declares some properties of the scene camera

**Color.h**  Some properties and operations of the color are declared and defined, and the operator is overloaded to facilitate color calculation

**CubeMeshData.h**  Defines the cube data to be used in the demo to render

**Enum.h**  Enums are used in the program

**Light.h**  Properties of the light in the scene declared in the file. These values are used to perform vertex lighting on the object.

**Material.h**  Material of the cube.

**MathTools.h**  Interface declaration for mathematical methods to be used in the rendering process

**Matrix4x4.h**  Declares the interface for matrix operations, and overloaded operators

**Point2D.h**  Declares a point struct in a 2d plane

**Vector3D.h**  Some properties and operations of 3D vector are declared and defined, and the operator is overloaded

**Vertex.h**  Vertex data for the geometric model

###### Source files:

**MathTools.cpp**  Implementation of mathematical function in mathtools.h.

Including functions to get translation matrix, scale matrix, rotation matrix, view matrix, project matrix and linear interpolation

**Matrix4x4.cpp**  Implementation of mathematical function in Matrix4x4.h.

Including functions to figure out determinant of the matrix，adjoint matrix and inverse matrix

**Main.cpp**  The main file of the program, to implement a 3D pipeline, and use windows GDI draw backbuff on the screen.

**Texture.jpg** The texture map used in rendering process

**Ps**: In addition to the README, the source code in the project also has some detailed comments.

#### Instructions for use:

This program was developed using vs2013, and I have submitted the whole project document.You can use the corresponding version of Visual Studio to open SoftWareRenderer/softwarerender.sln and check the entire project.

To see the demo run directly, you can run SoftWareRenderer/Release/ softwarerenderer.exe directly

When running the demo, you can

**Press Q** to switch render modes, including texture mode, vertex color mode and wireframe mode

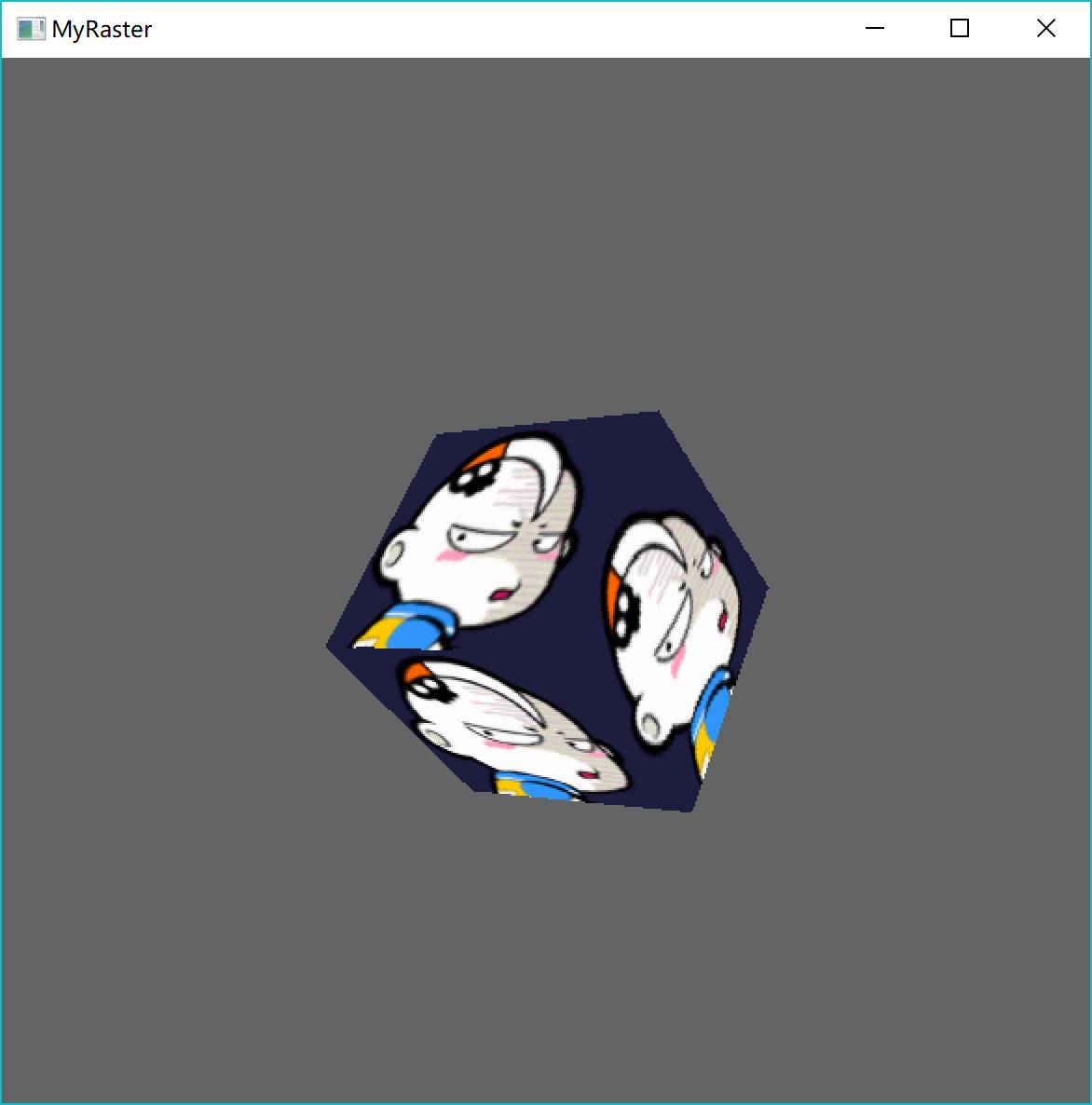
**Press W** to switch the light mode

**Press E** to switch between different texture sampling modes, including point sampling and bilinear sampling

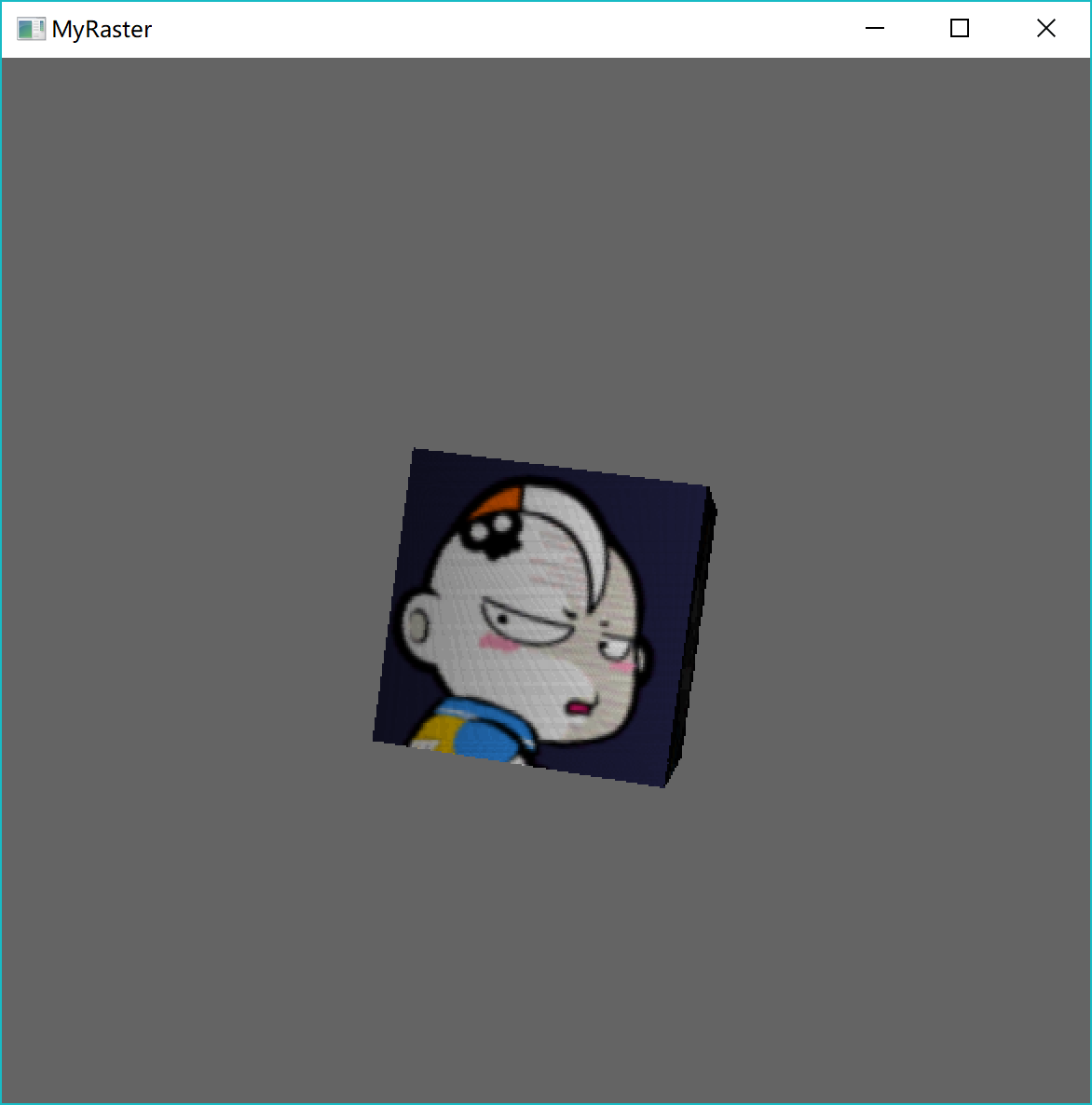
The speed of rotation of the cube can be adjusted by pressing **UP or DOWN**.

#### Effect:

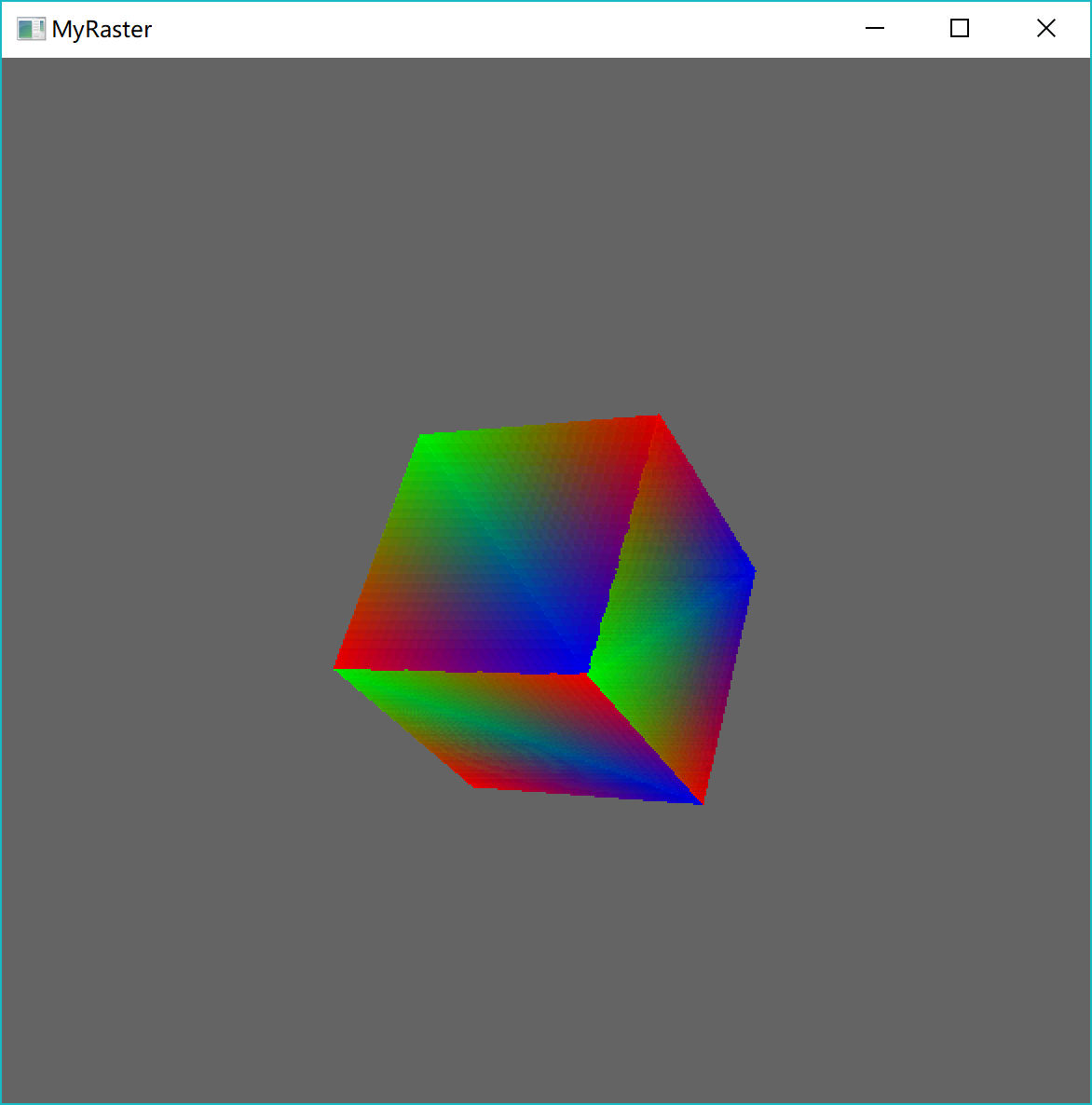
1.running in texture mode and set LightMode::Off



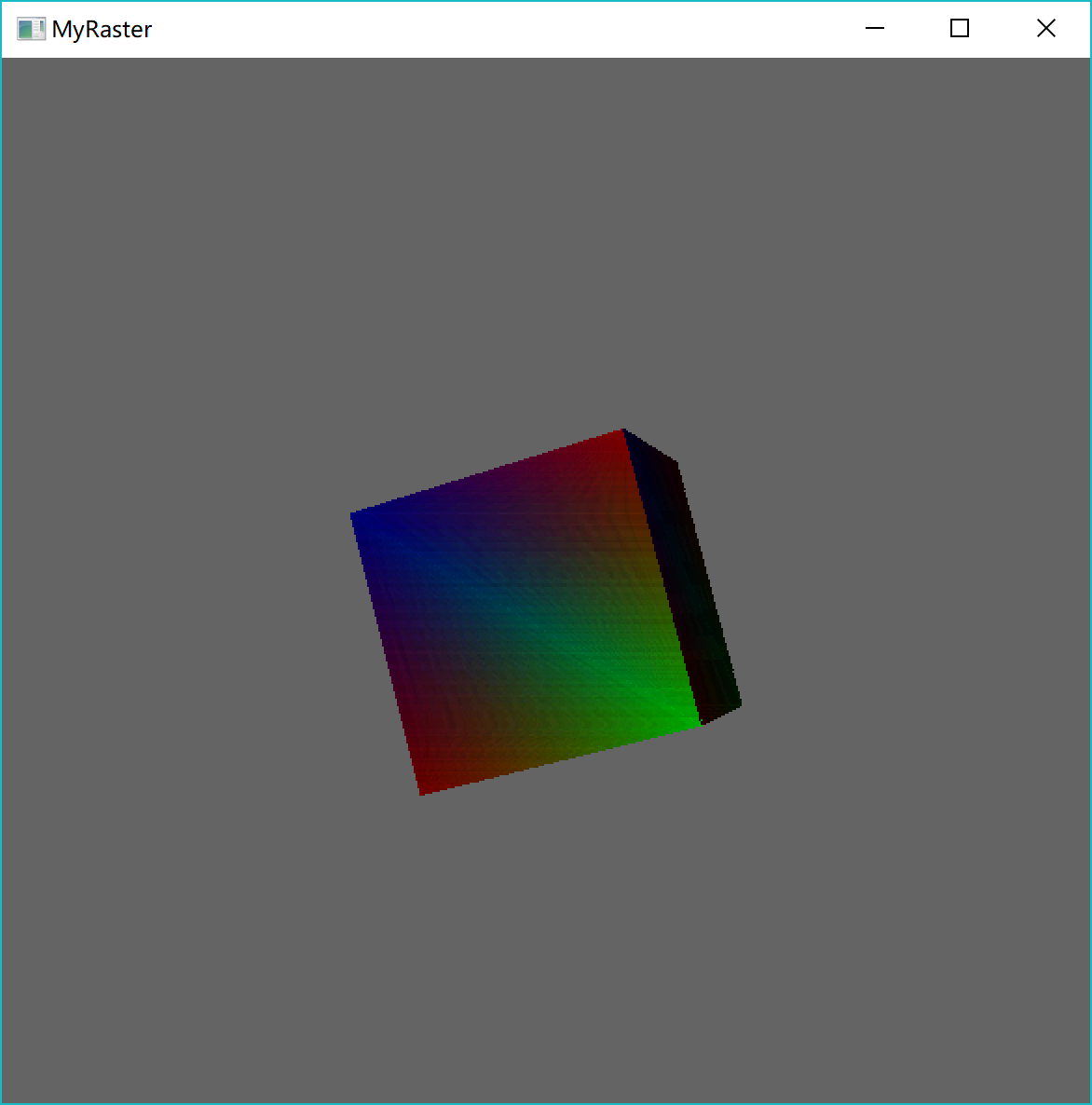
2. running in texture mode and set LightMode::On



3. running in vertex color mode and set LightMode::Off



4 . running in vertex color mode and set LightMode::On



5. running in wireframe mode

