HW3

P76091551 譚至斌

- OS: Ubuntu 20.04
- Environment Setup: `$ sudo apt install build-essential cmake libglfw3-dev libglfw3 libglm-dev #(I just follow the guild slides)`
- Build: `$ mkdir -p build && cd build && cmake .. && make -j`
- Executable file: `$ ./build/bin/Homework01`
- Input format:
  `./Homework01 [model name] [texture name] [vertex shader file name] [fragment shader file name]`
- Run:

```
$ cd build/bin && ./Homework01 resources/model/cube.obj
resources/texture/uv.png Shader/BasicVertexShader.vs.glsl
Shader/BasicFragmentShader.fs.glsl
```

  - or just run ./run.sh

```
$ ./run.sh #(this script will build and run the program automatically)
```

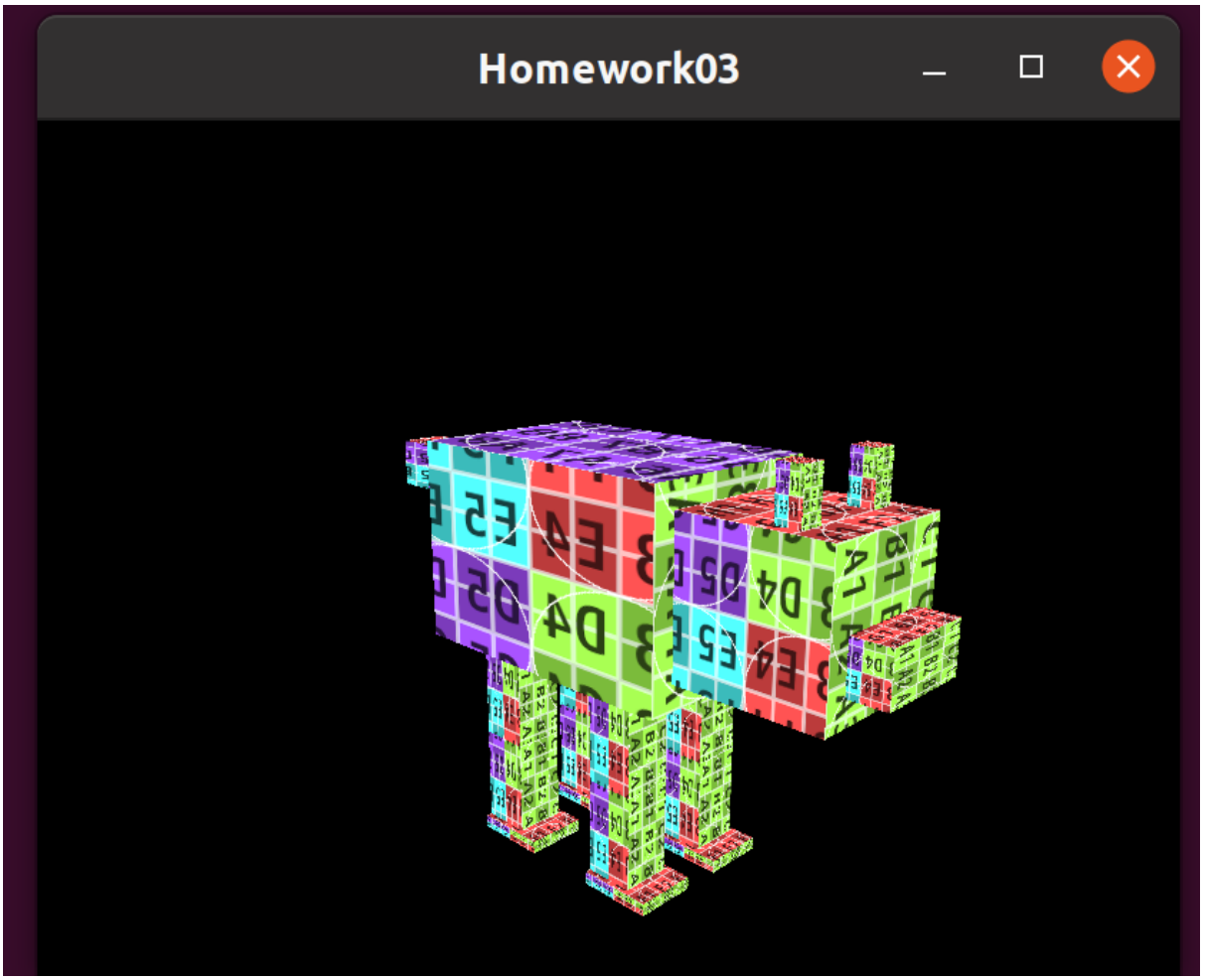
- Program Overview:

```
src/
├── CMakeLists.txt
├── Main.cpp
├── Model
│   ├── Mesh.cpp
│   ├── Mesh.hpp
│   ├── TextureFactory.cpp
│   ├── TextureFactory.hpp
│   └── Vertex.hpp
├── OpenGL
│   ├── Detail
│   │   ├── Set.hpp
│   │   └── Set-inl.hpp
│   ├── OpenGLBufferObject.cpp
│   ├── OpenGLBufferObject.hpp
│   ├── OpenGLException.cpp
│   ├── OpenGLException.hpp
│   ├── OpenGL.hpp
│   ├── OpenGLShader.cpp
```

```
│   ├── OpenGLShader.hpp
│   ├── OpenGLShaderProgram.cpp
│   ├── OpenGLShaderProgram.hpp
│   ├── OpenGLShaderProgram-inl.hpp
│   ├── OpenGLTexture.cpp
│   ├── OpenGLTexture.hpp
│   ├── OpenGLVertexArrayObject.cpp
│   └── OpenGLVertexArrayObject.hpp
├── OpenGLWindow.cpp
├── OpenGLWindow.hpp
├── Scene
│   ├── Component.cpp
│   └── Component.hpp
├── Shader
│   ├── BasicFragmentShader.fs.glsl
│   └── BasicVertexShader.vs.glsl
└── Utils
    ├── Compilers.hpp
    ├── FileIO
    │   ├── Detail
    │   │   ├── Generals.cpp
    │   │   └── Generals.hpp
    │   ├── FileIn.cpp
    │   └── FileIn.hpp
    ├── Global.hpp
    └── StringFormat
    ├── StringFormat.hpp
    └── StringFormat-inl.hpp
```

- src/Scene:
  - The implementation of a basic scene graph.
    - Component represents a node in the graph. It handles all graphic operations, including rotation, translation, and drawing. Every time a component does a graphic operation, it will call a function named ‘propagate’, which is used to propagate the change to its child components recursively.

- Operation Manual



    - Available keys: (all are lowercase on the keyboard)
        - Main
            - Q: quit
            - R: reset
        - Animation
            - D: dance
            - W: walk
        - Control (rotation only, on pitch, roll and yaw)
            - C: camera
            - B: body (torso)
            - H: head
            - T: tail
            - L: leg
            - F: foot
            - 1: left front
            - 2: right front
            - 3: left back
            - 4: right back
            - X: ref_0
            - Y: ref_1
            - Z: ref_2

- Examples:
  - Reset the model to the initial position:
    - press R
  - Let model start to dance:
    - press D
  - Stop dancing:
    - press R
  - Rotate torso on roll:
    - press B, press X, and drag the mouse left and right
  - Rotate torso on yaw:
    - press B, press Y, and drag the mouse left and right
  - Rotate torso on pitch:
    - press B, press Z, and drag the mouse left and right
  - Rotate the left front leg on roll:
    - press L, press 1, press X, and drag the mouse left and right
  - Rotate the right back foot on pitch:
    - press F, press 4, press Z, and drag the mouse left and right
  - Control camera:
    - press C. Drag mouse left and right to rotate camera horizontally, and drag mouse up and down to rotate camera vertically. (Note: the camera always looks at to the original point)