HW01 Report

tags: `CV_class`

P76111131 唐飴苹

Environment

- OS : Linux
- C++ & CMake
- OpenGL (4.6.0)
- Libraries
    - glfw (3.3.8)
    - glad
    - glm (0.9.9.8)
    - stb_image (v2.25)
    - tinyobjloader (version 0.9.20)
    - dear imgui (v1.77)

Method Description

OpenGLBufferObject.cpp

1. allocateBufferData

```
/* Allocate a new size bytes data storage for OpenGLBufferObject.
    Initialize with data. */
void OpenGLBufferObject::allocateBufferData(const void *data,
                                            GLsizeiptr size) noexcept
{
    PROGRAM_ASSERT(Detail::isCreated(id_));
    // Fill in the Blank
    glBufferData(type_, size, data, usagePattern_);
}
```

   - ***void glBufferData (GLenum target, GLsizeiptr size, const void * data, GLenum usage)***
     creates and initializes a buffer object's data storage
       - *target*
         Specifies the target to which the buffer object is bound for glBufferData,
         ex.`GL_ARRAY_BUFFER`, `GL_TEXTURE_BUFFER`
       - *usage*
         Specifies the expected usage pattern of the data store.
         ex. `GL_STREAM_DRAW`, `GL_STATIC_READ`

2. bind

```
/* Bind the OpenGLBufferObject to the current OpenGL content. */
void OpenGLBufferObject::bind() noexcept
{
    PROGRAM_ASSERT(Detail::isCreated(id_));
    // Fill in the Blank
    glBindBuffer(type_, id_);
}
```

- ***void glBindBuffer(GLenum target, GLuint buffer)***
  binds the target to a named buffer object, and the object remains active
  - *target*
    Specifies the target to which the buffer object is bound,
    ex.`GL_ARRAY_BUFFER`, `GL_TEXTURE_BUFFER`

3. create

```
/* Create the buffer. */
void OpenGLBufferObject::create()
{
    PROGRAM_ASSERT(!Detail::isCreated(id_));
    // Fill in the Blank
    glGenBuffers(1, &id_);
    if (!Detail::isCreated(id_))
    {
        throw OpenGLException("OpenGLBufferObject failed to instantiate.");
    }
}
```

- ***void glGenBuffers(GLsizei n, GLuint buffers);***
  generate the name of buffer object, return *n* buffer object names in *buffers*

4. release, tidy

```
/* Release the OpenGLBufferObject from the current OpenGL content. */
void OpenGLBufferObject::release() noexcept
{
    PROGRAM_ASSERT(Detail::isCreated(id_));
    // Fill in the Blank
    glDeleteBuffers(1, &id_);
}

/* Clean up and delete the buffer. */
void OpenGLBufferObject::tidy() noexcept
{
    PROGRAM_ASSERT(Detail::isCreated(id_));
    // Fill in the Blank
    glDeleteBuffers(1, &id_);

    id_ = Detail::noId;
}
```

- ***void glDeleteBuffers(GLsizei n, const GLuint buffers)***
  delete *n* named buffer objects

OpenGLShader.cpp

1. compileStatus

```
/* Get the compile status. */
inline bool compileStatus(GLuint id) noexcept
{
    GLint status;
    // Fill in the Blank
    glGetShaderiv(id, GL_COMPILE_STATUS, &status);

    return (status == GL_TRUE);
}
```

- ***void glGetShaderiv(GLuint shader, GLenum pname, GLint params);***
  return a parameter in *params* from shader object *shader*
  - *pname*
    Specifies the object parameter.
    ex. GL_SHADER_TYPE, GL_DELETE_STATUS

2. compileFromSource

```
/* Compile the \a source content into OpenGLShader. */
bool OpenGLShader::compileFromSource(const char *source) noexcept
{
    PROGRAM_ASSERT(Detail::isCreated(id_));
    // Fill in the Blank
    glShaderSource(id_, 1, &source, NULL);
    // Fill in the Blank
    glCompileShader(id_);

    return Detail::compileStatus(id_);
}
```

- ***void glShaderSource(GLuint shader, GLsizei count, const GLchar string, const GLint length)***
  Replaces the source code in shader object *shader*

- ***void glCompileShader(GLuint shader)***
  compiles the source code strings that have been stored in the shader object specified by *shader*

3. create

```
/* Create the shader. */
void OpenGLShader::create()
{
    PROGRAM_ASSERT(!Detail::isCreated(id_));
    // Fill in the Blank
    id_ = glCreateShader(type_);

    if (!Detail::isCreated(id_))
    {
        throw OpenGLException("OpenGLShader failed to instantiate.");
    }
}
```

- ***GLuint glCreateShader(GLenum shaderType)***
  Creates a shader object with specified shader type

        - *shaderType*
          Specifies the type of shader to be created.
          ex. `GL_COMPUTE_SHADER`, `GL_VERTEX_SHADER`

4. tidy

```
/* Clean up and delete the shader. */
void OpenGLShader::tidy() noexcept
{
    PROGRAM_ASSERT(Detail::isCreated(id_));
    // Fill in the Blank
    glDeleteShader(id_);

    id_ = Detail::noId;
}
```

    - ***void glDeleteShader(GLuint shader)***
      Deletes a shader object *shader*

OpenGLShaderProgram.cpp

1. attachShader

```
/* Attach the shader to the OpenGLShaderProgram */
void OpenGLShaderProgram::attachShader(
    std::unique_ptr<OpenGLShader> &&shader) noexcept
{
    PROGRAM_ASSERT(Detail::isCreated(id_));
    // Fill in the Blank
    glAttachShader(id_, shader->id());

    shaders_.push_back(std::move(shader));
}
```

    - ***void glAttachShader(GLuint program, GLuint shader)***
      Attaches shader object *shader* to a program object *program*

2. create

```
/* Create the buffer. */
void OpenGLShaderProgram::create()
{
    PROGRAM_ASSERT(!Detail::isCreated(id_));
    // Fill in the Blank
    id_ = glCreateProgram();

    if (!Detail::isCreated(id_))
    {
        throw OpenGLException("OpenGLShaderProgram failed to instantiate.");
    }
}
```

    - ***GLuint glCreateProgram(void)***
      Creates a program object and returns a non-zero value which could be referenced

3. destroyProgram

```
/* Destroy the shader program. */
void OpenGLShaderProgram::destroyProgram() noexcept
{
    PROGRAM_ASSERT(Detail::isCreated(id_));
    // Fill in the Blank
    glDeleteProgram(id_);

    id_ = Detail::noId;
}
```

- ***void glDeleteProgram(GLuint program)***
  Deletes a program object *program*

4. destroyShaders

```
/*  Destroy the shader which attached to the program. */
void OpenGLShaderProgram::destroyShaders() noexcept
{
    PROGRAM_ASSERT(Detail::isCreated(id_));
    for (auto &shader : shaders_)
    {
        // Fill in the Blank
        glDeleteShader(shader->id());

        shader.reset(nullptr);
    }
    shaders_.clear();
}
```

- ***void glDeleteShader(GLuint shader)***
  Deletes a shader object *shader*

5. disableAttributeArray

```
/* Disable the vertex attribute at index in the OpenGLShaderProgram. */
void OpenGLShaderProgram::disableAttributeArray(GLuint index) noexcept
{
    // Fill in the Blank
    glDisableVertexAttribArray(index);
}
```

- ***void glDisableVertexAttribArray(GLuint index)***
  Disable a generic vertex attribute array specified by *index*

6. enableAttributeArray

```
/*  Enable the vertex attribute at index in the OpenGLShaderProgram. */
void OpenGLShaderProgram::enableAttributeArray(GLuint index) noexcept
{
    // Fill in the Blank
    glEnableVertexAttribArray(index);
}
```

- ***void glEnableVertexAttribArray(GLuint index)***
  Enable a generic vertex attribute array specified by *index*

7. link

```
/* Link the shaders in the OpenGLShaderProgram together. */
void OpenGLShaderProgram::link() noexcept
{
    // Fill in the Blank
    glLinkProgram(id_);
}
```

- ***void glLinkProgram(GLuint program)***
  links the program object specified by *program*.

8. linkStatus

```
/* Gets the link status of the OpenGLShader */
bool OpenGLShaderProgram::linkStatus() const noexcept
{
    GLint status;
    // Fill in the Blank
    glGetProgramiv(id_, GL_LINK_STATUS, &status);

    return (status == GL_TRUE);
}
```

- ***void glGetProgramiv(GLuint program, GLenum pname, GLint params)***
  return a parameter in *params* from a program object *program*

  - *pname*
    Specifies the object parameter.
    ex. `GL_DELETE_STATUS`, `GL_LINK_STATUS`

9. mapAttributePointer

```
/* Set the location and format of the array of attributes at index
    in OpenGLShaderProgram. */
void OpenGLShaderProgram::mapAttributePointer(GLuint index, GLint size,
                                             GLenum type, GLboolean normalized,
                                             GLsizei stride,
                                             int offset) noexcept
{
    // Fill in the Blank
    glVertexAttribPointer(index, size, type, normalized, stride, (void *)offset);
}
```

- ***void glVertexAttribPointer(GLuint index, GLint size, GLenum type, GLboolean normalized, GLsizei stride, const void pointer)***
  define an array of generic vertex attribute data

  - *type*
    Specifies the data type of each component in the array.
    ex. `GL_BYTE`, `GL_UNSIGNED_BYTE`

10. use

```
/* Use the OpenGLShaderProgram to the current rendering state. */
void OpenGLShaderProgram::use() noexcept
{
    // Fill in the Blank
    glUseProgram(id_);
}
```

- ***void glUseProgram(GLuint program)***
  Installs the program object specified by *program* as part of current rendering state.

OpenGLTexture.cpp

1. bind

```
void OpenGLTexture::bind()
{
    PROGRAM_ASSERT(Detail::isCreated(id_));

    // Fill in the Blank
    glBindTexture(GL_TEXTURE_2D, id_);
}
```

- ***void glBindTexture(GLenum target, GLuint texture)***
  bind a named texture *texture* to a texturing target *target*

2. create

```
void OpenGLTexture::create()
{
    PROGRAM_ASSERT(!Detail::isCreated(id_));
    // Fill in the Blank
    glGenTextures(1, &id_);

    if (!Detail::isCreated(id_))
    {
        throw OpenGLException(
            "OpenGLTexture instantiate failed at 'glGenTextures'.");
    }
}
```

- ***void glGenTextures(GLsizei n, GLuint textures)***
  returns *n* texture names in *textures*

3. release, tidy

```
void OpenGLTexture::release()
{
    PROGRAM_ASSERT(Detail::isCreated(id_));

    // Fill in the Blank
    glDeleteTextures(1, &id_);
}

void OpenGLTexture::tidy()
{
    PROGRAM_ASSERT(Detail::isCreated(id_));

    // Fill in the Blank
    glDeleteTextures(1, &id_);

    id_ = 0;
}
```

- ***void glDeleteTextures(GLsizei n, const GLuint textures)***
  deletes *n* textures named by the elements of the array *textures*

4. setMagnificationFilter, setMinificationFilter, setWrapOption

```
void OpenGLTexture::setMagnificationFilter(Filter filter)
{
    PROGRAM_ASSERT(Detail::isCreated(id_));
    magnificationFilter_ = filter;

    bind();
    // Fill in the Blank
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, magnificationFilter_);

    release();
}

void OpenGLTexture::setMinificationFilter(Filter filter)
{
    PROGRAM_ASSERT(Detail::isCreated(id_));
    minificationFilter_ = filter;

    bind();
    // Fill in the Blank
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, minificationFilter_);

    release();
}

void OpenGLTexture::setWrapOption(WrapOption option)
{
    PROGRAM_ASSERT(Detail::isCreated(id_));
    wrapOption_ = option;

    bind();
    // Fill in the Blank
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, wrapOption_);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, wrapOption_);

    release();
}
```

- ***void glTexParameteriv(GLenum target, GLenum pname, const GLint * params)***
  set texture parameters and stored pointer to the value in *params*
  - *target*
    Specifies the target to which the texture is bound for glTexParameter functions.
    ex. `GL_TEXTURE_1D_ARRAY`, `GL_TEXTURE_2D`
  - *pname*
    Specifies the symbolic name of a single-valued texture parameter.
    ex. `GL_TEXTURE_MIN_FILTER`, `GL_TEXTURE_MAG_FILTER`

5. bindbuffer

```
void OpenGLTexture::bindBuffer(const std::vector<unsigned char> &buffer) const
{
    // Fill in the Blank
    // (bind)
    glBindTexture(GL_TEXTURE_2D, id_);

    // (parameter setup: filter and warpping method)
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, wrapOption_);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, wrapOption_);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, minificationFilter_);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, magnificationFilter_);

    // (data specify)
    glTexImage2D(GL_TEXTURE_2D, 0, format_, width_,
        height_, 0, format_, GL_UNSIGNED_BYTE, buffer.data());

    // (generate mipmap)
    glGenerateMipmap(GL_TEXTURE_2D);

}
```

- ***void glTexImage2D(GLenum target, GLint level, GLint internalFormat, GLsizei width, GLsizei height, GLint border, GLenum format, GLenum type, const GLvoid * data)***
  Specify a two-dimensional texture image, the arguments describe the parameters of the texture image (*height*, *width*, *border*) and how the image is represented in memory (*format*, *type*, *data*)
  - *target*
    Specifies the target texture.
    ex. `GL_TEXTURE_2D`, `GL_PROXY_TEXTURE_2D`
  - *format*
    Specifies the format of the pixel data.
    ex. `GL_RED`, `GL_RGB_INTEGER`,
  - *type*
    Specifies the data type of the pixel data.
    ex. `GL_UNSIGNED_BYTE`, `GL_BYTE`

OpenGLVertexArrayObject.cpp

1. bind, release

```
/* Bind the OpenGLVertexArrayObject to the current OpenGL content. */
void OpenGLVertexArrayObject::bind() noexcept
{
    PROGRAM_ASSERT(Detail::isCreated(id_));

    // Fill in the Blank
    glBindVertexArray(id_);
}

/* Release the OpenGLVertexArrayObject from the current OpenGL content. */
void OpenGLVertexArrayObject::release() noexcept
{
    PROGRAM_ASSERT(Detail::isCreated(id_));

    // Fill in the Blank
    glBindVertexArray(0);
}
```

   - ***void glBindVertexArray(GLuint array)***
     binds a vertex array object with name *array*
      - *glBindVertexArray(0)*
        break the existing vertex array object binding.

2. create

```
/* Create the buffer. */
void OpenGLVertexArrayObject::create()
{
    PROGRAM_ASSERT(!Detail::isCreated(id_));
    // Fill in the Blank
    glGenVertexArrays(1, &id_);

    if (!Detail::isCreated(id_))
    {
        throw OpenGLException("OpenGLVertexArrayObject failed to instantiate.");
    }
}
```

   - ***void glGenVertexArrays(GLsizei n, GLuint arrays)***
     returns *n* vertex array object names in *arrays*.

3. tidy

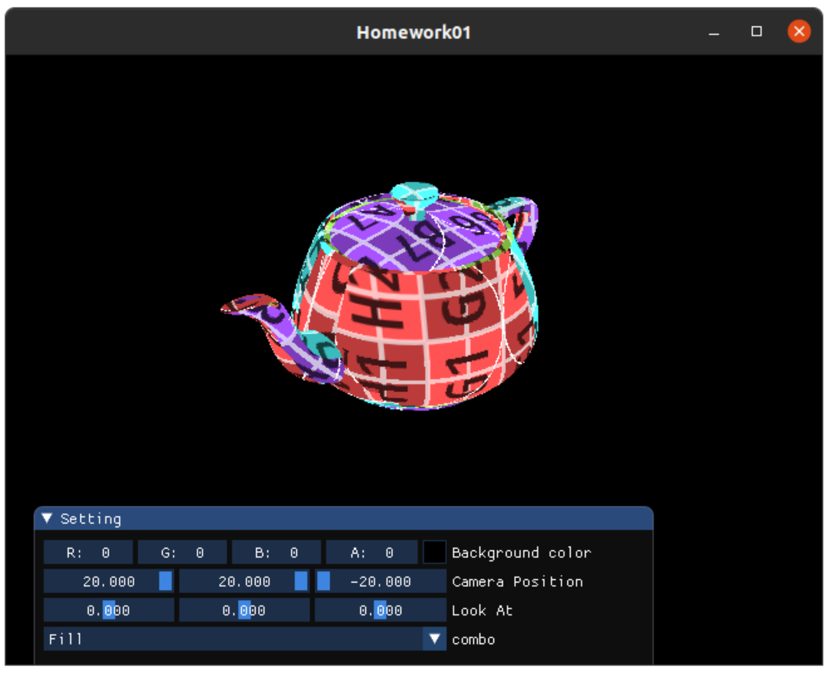
```
/* Clean up and delete the buffer. */
void OpenGLVertexArrayObject::tidy() noexcept
{
    PROGRAM_ASSERT(Detail::isCreated(id_));

    // Fill in the Blank
    glDeleteVertexArrays(1, &id_);

    id_ = Detail::noId;
}
```

   - ***void glDeleteVertexArrays(GLsizei n, const GLuint arrays)***
     delete *n* vertex array objects whose names are stored in the array addressed by *arrays*

How to run the program

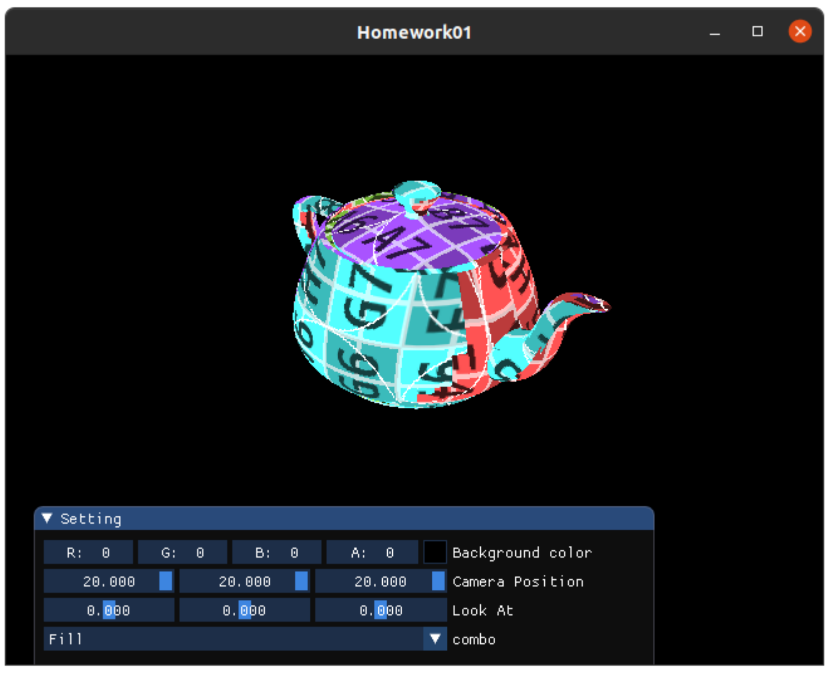
```
$ cd sample_code/build/bin
$ ./Homework01 "resources/model/Utah_teapot_(solid)_texture.obj"
"resources/texture/uv.png" "Shader/BasicVertexShader.vs.glsl"
"Shader/BasicFragmentShader.fs.glsl"
```

Results


Homework01

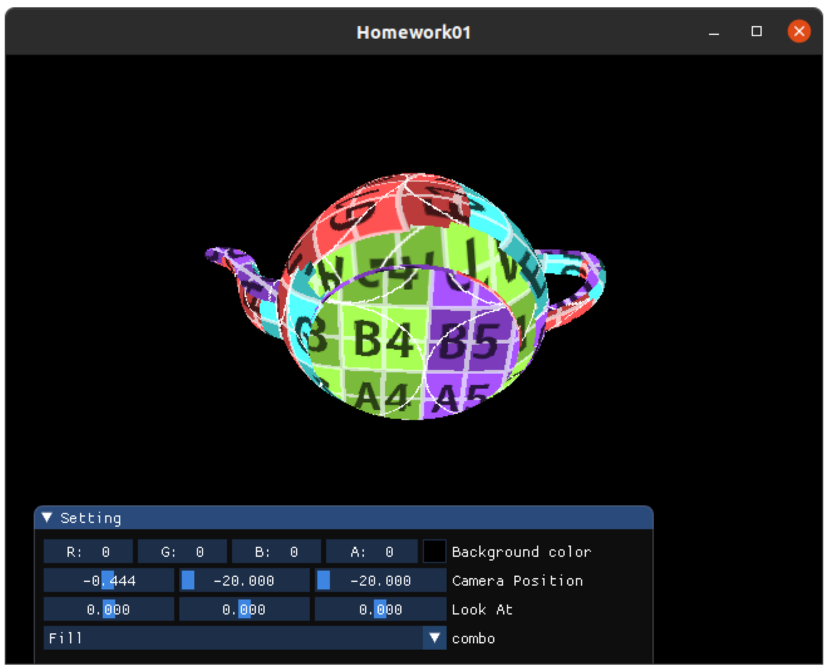
▼ Setting

```
R: 0   G: 0   B: 0   A: 0   Background color
20.000   20.000   -20.000   Camera Position
0.000   0.000   0.000   Look At
Fill   combo
```


Homework01

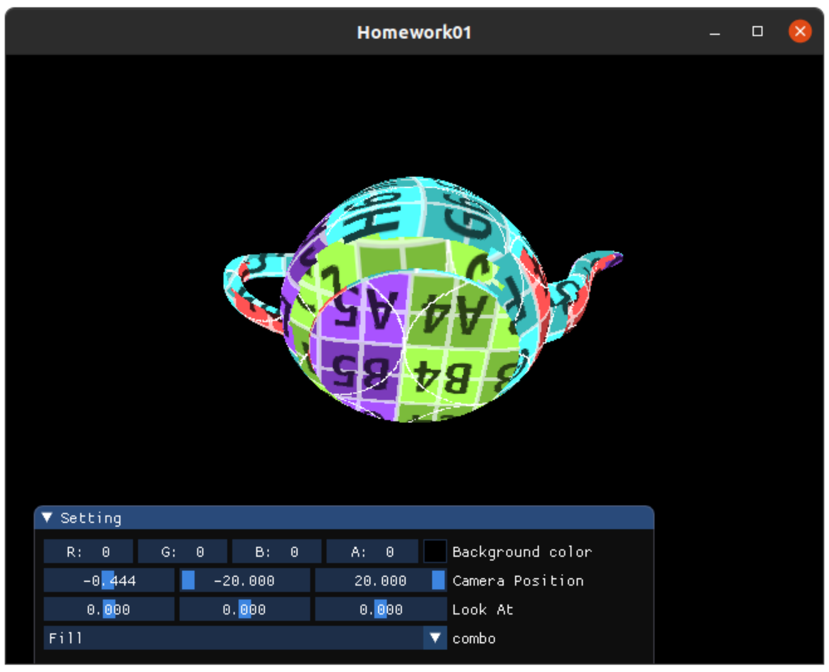
▼ Setting

```
R: 0   G: 0   B: 0   A: 0   Background color
20.000   20.000   20.000   Camera Position
0.000   0.000   0.000   Look At
Fill   combo
```


Homework01

▼ Setting

```
R: 0   G: 0   B: 0   A: 0   Background color
-0.444   -20.000   -20.000   Camera Position
0.000   0.000   0.000   Look At
Fill   combo
```


Homework01

▼ Setting

```
R: 0   G: 0   B: 0   A: 0   Background color
-0.444   -20.000   20.000   Camera Position
0.000   0.000   0.000   Look At
Fill   combo
```