UNIVERSIDAD DE DARMSTADT
B. FRÖMMER, E. HERGENRÖTHER, B. MEYER

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

11 DE ABRIL DE 2023

VISUAL INFORMÁTIC SOSE 2023
A
TAREA 1

En esta hoja de ruta se trata de dibujar la geometría de los objetos. Específicamente, aquí se requiere el conocimiento de VertexArrayObjects, VertexBufferObjects e IndexBufferObjects. Con la tarea, se le ha dado un marco totalmente funcional sobre el que se construirán las dos primeras prácticas. Así que a partir de esta práctica, utiliza este proyecto.

Nota: El proyecto básico fue creado con CLion y puede abrirse fácilmente con este IDE. Alternativamente, también puede intentar que el proyecto se ejecute en otro IDE. Para ello, puede ser necesario reintegrar las librerías externas requeridas, posiblemente incluso recompilarlas para su sistema operativo. Necesitará las bibliotecas GLEW, GLFW y GLM para este curso práctico.

Abre el framework (preferiblemente en CLion, haz clic con el botón derecho del ratón en "Abrir como proyecto CLion"), compílalo e inícialo. Si todo ha funcionado correctamente, debería abrirse una ventana negra e indicarse la velocidad de fotogramas actual en la ventana de salida del IDE.

1.1 Inicialización de la escena

El marco dado se encarga de la creación y actualización periódica de una ventana simple. En este curso práctico, sin embargo, sólo estamos interesados en la clase `Scene`. Aquí, todos los objetos de la geometría a mostrar se crean en el método `init()`. El array float `vertices` ya contiene los datos de posición 2D y la información de color RGB para mostrar una simple casa. Los índices (es decir, el orden en el que los vértices deben enlazarse para formar triángulos) también están ya definidos.

Según la conferencia, ahora hay que enviar estos datos a la GPU. Para ello, proceda de la siguiente manera:

a) Crear un VBO, enlazarlo (activarlo) y llenarlo con los datos de los vértices.

b) Crear un VAO y enlazarlo.

c) Define y activa los respectivos atributos de vértice. Cada puntero de atributo definido hace referencia al VBO actualmente activo, así como al VAO.

d) Crear y vincular un búfer de índice. El IBO ahora también hace referencia al VAO activo. Por lo tanto, cambiar la secuencia de procedimientos suele ser problemático.

e) Opcionalmente, todo puede ser desvinculado (`bind(0)`) para evitar cambios accidentales en los buffers y VAO. Recuerda desvincular primero el VAO.

1.2 Renderizar escena()

En cada fotograma, el framework llama al método `render()` de la `clase Scene`. Todos los objetos que van a ser renderizados deben integrarse aquí en el pipeline de renderizado. Además, debe definirse de antemano qué shader debe hacerse cargo del renderizado. Para pasar la responsabilidad al respectivo shader, por favor utilice el método `use()` del `ShaderProgram` (que ya existe). Luego proceda de la siguiente manera:

a) Vincula el VAO.

b) Dibuja los elementos.

c) Desatado opcional para evitar cambios accidentales en el VAO. Si has hecho todo correctamente, debería aparecer la imagen de la figura 1:

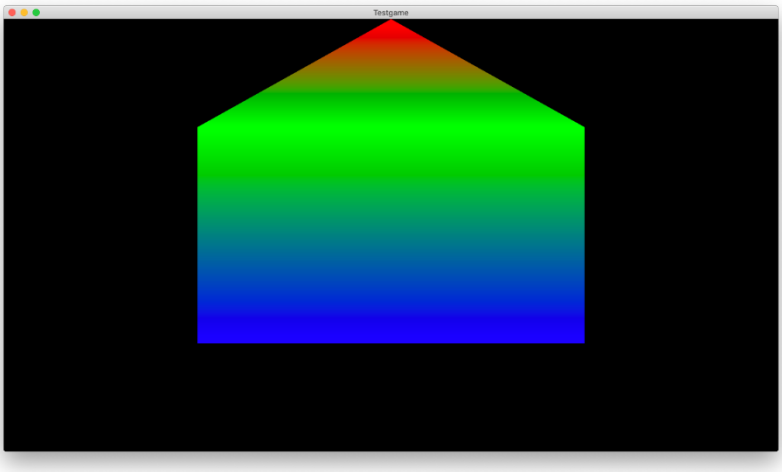
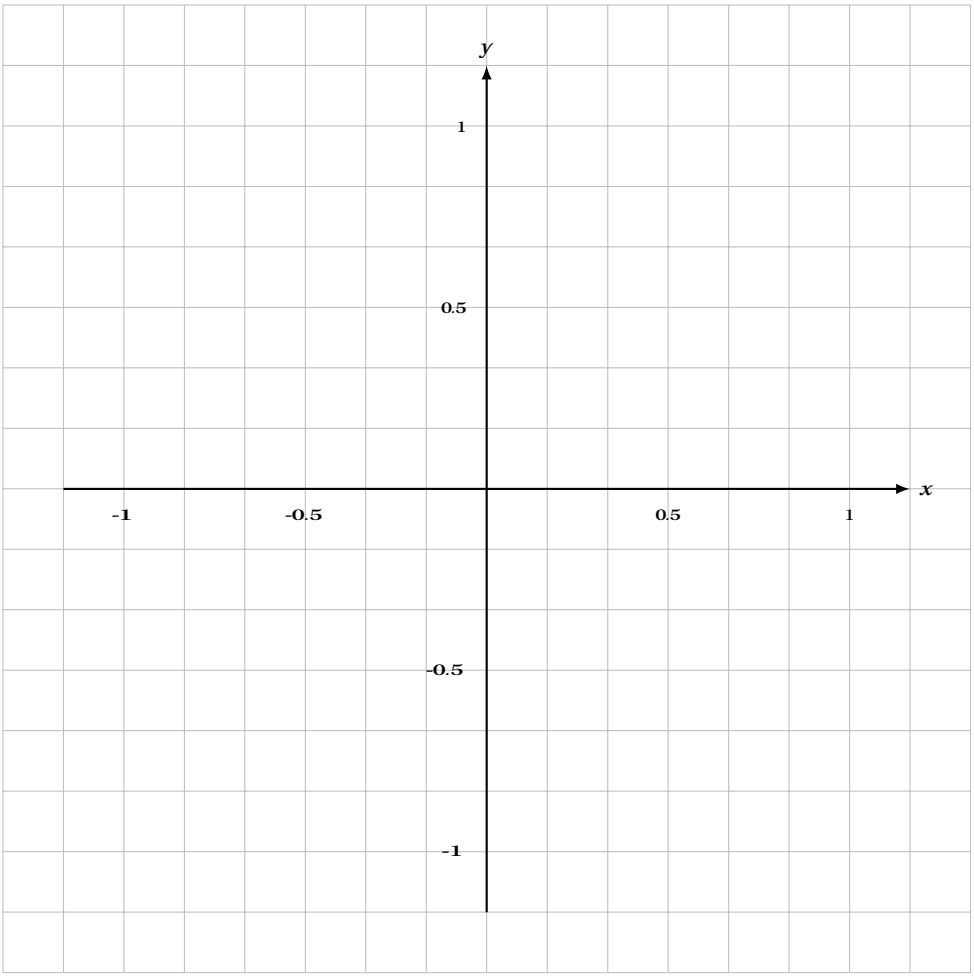


Figura 1: Casa renderizada

1.3 Iniciales

Cambie los datos de vértice e índice para que aparezcan las iniciales de su nombre. El siguiente sistema de coordenadas debería ayudarte a determinar las coordenadas correspondientes. **Mantenga el tipo de letra simple. Una O no tiene por qué ser redonda.**



1.4 Eliminación de rostros

Añade las siguientes líneas de código al final del método `Scene.init()`:

```
bool Escena::init()
...
        glEnable(GL_CULL_FACE);
        glFrontFace(GL_CCW);
        glCullFace(GL_BACK);
...
```

Ahora probablemente no todas sus superficies definidas son renderizadas. ¿Tienes una idea espontánea de por qué puede ser? Corrija su solución si es necesario.