Hochschule Darmstadt
B. Frömmer, B. Meyer

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

15. Oktober 2023

Visual Computing WS 2023
Aufgabe 1

In diesem Aufgabenblatt beschäftigen Sie sich mit dem Zeichnen der Geometrie von Objekten. Konkret sind hier die Kenntnisse zu VertexArrayObjects, VertexBufferObjects und IndexBufferObjects gefragt. Mit der Aufgabenstellung haben Sie ein voll funktionsfähiges Framework bekommen, auf dem die ersten beiden Praktika aufbauen werden. Verwenden Sie also ab diesem Praktikum das vorliegende Projekt.

Hinweis: Das Basis-Projekt wurde mit CLion erstellt und kann mit in dieser IDE einfach geöffnet werden. Alternativ können Sie auch versuchen, das Projekt in einer anderen IDE zum Laufen zu bekommen. Hierfür kann es notwendig sein, die benötigten externen Bibliotheken neu integrieren, evtl. für Ihr Betriebssystem sogar neu zu compilieren. Sie benötigen für dieses Praktikum die Bibliotheken GLEW, GLFW und GLM.

Öffnen Sie das Framework (vorzugsweise in CLion, Rechtsklick "Öffnen als CLion-Projekt"), compilieren und starten Sie es. Hat alles korrekt funktioniert, sollte sich ein schwarzes Fenster öffnen und im Ausgabefenster der IDE die aktuelle Framerate angegeben werden.

1.1 Szenen-Initialisierung

Das gegebene Framework übernimmt das Erstellen und regelmäßige Aktualisieren eines einfachen Fensters. Uns soll in diesem Praktikum allerdings nur die Klasse `Scene` interessieren. Hier werden alle Objekte der anzuzeigenden Geometrie in der Methode `init()` angelegt. Das float-Array `vertices` beinhaltet bereits 2D-Positionsdaten sowie RGB-Farbinformationen, um ein einfaches Haus darzustellen. Die Indizes (also die Reihenfolge, in der die Vertices zu Dreiecken verknüpft werden sollen) sind auch bereits definiert.

Entsprechend der Vorlesung sollen diese Daten jetzt an die GPU gesendet werden. Gehen Sie hierfür wie folgt vor:

a) Erzeugen Sie ein VBO, binden es (setzen es aktiv) und füllen es mit den Vertex-Daten.

b) Erzeugen Sie ein VAO und binden es.

c) Definieren und aktivieren Sie die jeweiligen Vertex-Attribute. Jeder definierte Attrib-Pointer bezieht sich auf das aktuell aktive VBO sowie VAO.

d) Erstellen und Binden eines Indexbuffers. Auch das IBO bezieht sich nun auf das aktive VAO. Ein Ändern der Vorgehensreihenfolge ist daher oft problematisch.

e) Optional kann alles gelöst (engl. "unbind", `bind(0)`) werden, um versehentliche Änderungen an den Buffern und VAO zu vermeiden. Denken Sie daran, das VAO zuerst zu lösen.

1.2 Scene-`render()`

In jedem Frame wird von dem Framework die Methode `render()` der `Scene`-Klasse aufgerufen. Alle Objekte, die dargestellt werden sollen, müssen hier in die Rendering-Pipeline integriert werden. Außerdem muss jeweils zuvor definiert werden, welcher Shader das Rendern übernehmen soll. Um die Verantwortung an den jeweiligen Shader zu übergeben, nutzen Sie bitte die `use()`-Methode des `ShaderPrograms` (steht schon da). Anschließend gehen Sie wie folgt vor:

a) Binden Sie das VAO.

b) Zeichnen Sie die Elemente.

c) Optionales Lösen der Bindung, um versehentliche Änderungen am VAO zu vermeiden.

Haben Sie alles richtig gemacht, sollte das Bild aus Abbildung 1 erscheinen:

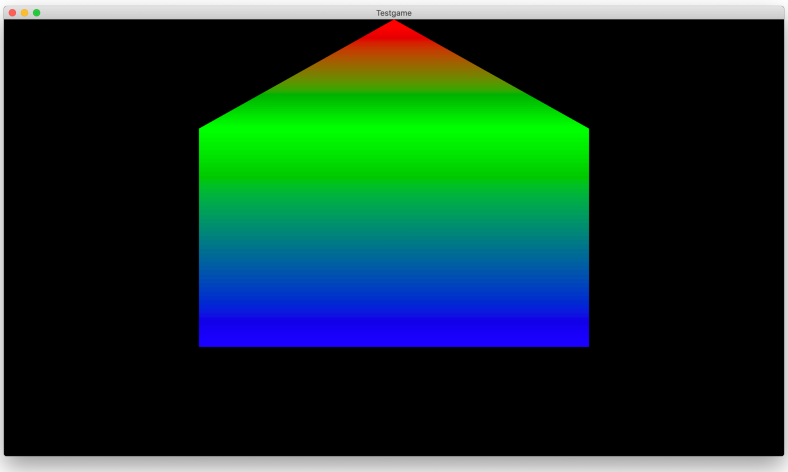
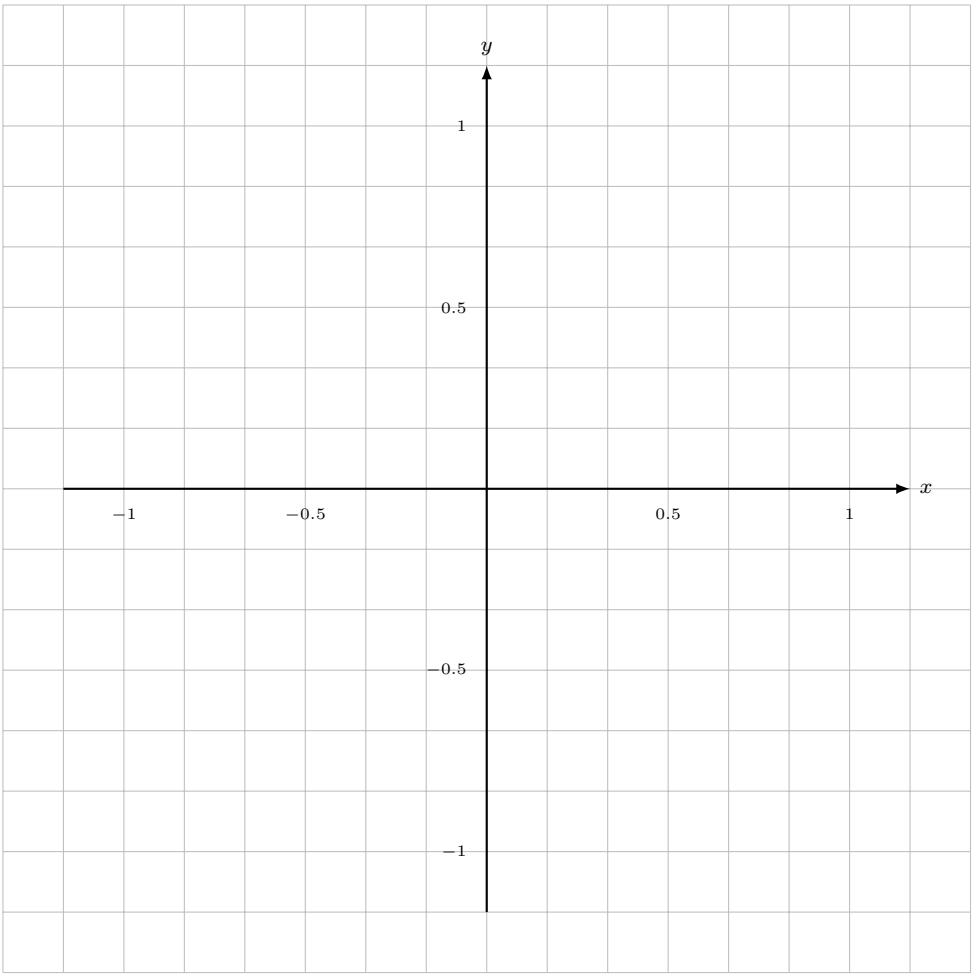


Abbildung 1: Gerendertes Haus

1.3 Initialen

Verändern Sie die Vertex- und Indexdaten, sodass die Initialen Ihres Namens dargestellt werden. Das nachfolgende Koordinatensystem soll Ihnen hierbei behilflich sein, um die entsprechenden Koordinaten zu ermitteln. **Halten Sie hierbei die Schrift einfach. Ein O muss nicht rund sein.**



1.4 Backface culling

Fügen Sie der `Scene.init()`-Methode am Ende folgende Codezeilen hinzu:

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

Jetzt werden vermutlich nicht mehr alle Ihrer definierten Flächen gerendert. Haben Sie eine spontane Idee, woran das liegen kann? Korrigieren Sie gegebenenfalls Ihre Lösung.