University of Tübingen
Prof. Dr.-Ing. Hendrik P.A. Lensch
Chair of Computer Graphics
Faezeh Zakeri (faezeh-sadat.zakeri@uni-tuebingen.de)
Lukas Ruppert (lukas.ruppert@uni-tuebingen.de)
Raphael Braun (raphael.braun@uni-tuebingen.de)
Andreas Engelhardt (andreas.engelhardt@uni-tuebingen.de)
Philipp Langsteiner (philipp.langsteiner@student.uni-tuebingen.de)

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

25. October 2022

Computer Graphics
Assignment 1

Submission deadline for the exercises: 01. November 2022, 0:00

Source Code Solutions

- Upload **only** the src folder and all the files it contains, plus the CMakeLists.txt and files that are already exist in the exercise.zip. Do NOT include your build folder, results and exercise sheet in the zip folder you upload.
- Zip them first and upload **.zip** folder on Ilias.

Written Solutions

Written solutions have to be submitted digitally as one PDF file via Ilias.

Motivation

In the first exercise sheet we will introduce you to matrix and vector multiplication, transformations and rotation matrices. The presented methods in this exercise are essentials for computer graphics applications, as they can, for example, be used to manipulate perspectives and create movement in scenes.

1.1 Matrix Multiplication (5 + 10 = 15 Points)

Given is the matrix $A = \begin{pmatrix} 2 & 0 & 0 \\ 0 & \sqrt{2} & -\sqrt{2} \\ 0 & \sqrt{2} & \sqrt{2} \end{pmatrix} \in \mathbb{R}^{3\times 3}$.

a) Describe in your own words what effect matrix A has on an arbitrary vector $\vec{a} \in \mathbb{R}^3$, when it is multiplied by the matrix from the left: $A \cdot \vec{a}$

b) Lets assume that $\vec{v} = \begin{pmatrix} 2 \\ \sqrt{2} \\ \sqrt{2} \end{pmatrix} \in \mathbb{R}^3$ and $\vec{w} \in \mathbb{R}^3$, with the elements of $\vec{w}$ being arbitrary numbers in $\mathbb{R}$.
Given is the following relation: $A \cdot \vec{w} = \vec{v}$. Calculate $\vec{w}$ and document all intermediate results.

1.2 Matrices (5 + 10 = 15 Points)

Given are the two matrices $R_\theta = \begin{pmatrix} C_\theta & -S_\theta & 0 \\ S_\theta & C_\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \in \mathbb{R}^{3\times3}$ and $R_\phi = \begin{pmatrix} C_\phi & 0 & S_\phi \\ 0 & 1 & 0 \\ -S_\phi & 0 & C_\phi \end{pmatrix} \in \mathbb{R}^{3\times3}$.

a) Explain how an arbitrary vector would be manipulated by these matrices if we were to multiply them from left onto it.

b) Given are the following relations: $R^{'} = R_\theta \cdot R_\phi$ and $R^{''} = R_\phi \cdot R_\theta$.
Show that, in general, the matrices $R^{'}$ and $R^{''}$ have different effect on an arbitrary vector $\vec{v} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \in \mathbb{R}^3$.

1.3 Rotation Matrix (10 Points)

Let the matrix $R \in \mathbb{R}^{2\times2}$ be a two-dimensional rotations matrix. Show that $R^{-1} = R^T$. In other words, show that a transposed rotation matrix is equal to its inverse.

1.4 Transformations (20 Points)

In the image below three polygons are displayed. Create a transformation matrix step by step in order of the instructions. Give the complete matrix and draw the final result.
Instruction:

- scale by 2 along the y-axis
- shear by 1 along the x-axis
- translate by -6 along the x-axis and -5 along the y-axis
- rotate by 180 degrees around the origin

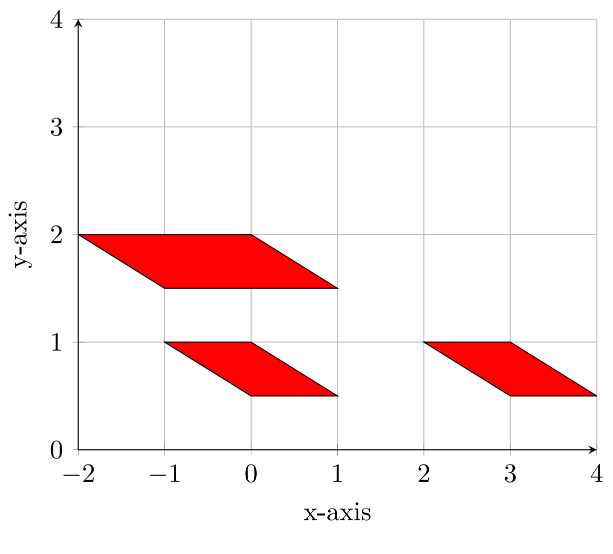


Figure 1: The three polygons to be transformed.

1.5 Coding Exercise (40 Points)

In this exercise, we will transform a given 3D mesh by simply transforming every single one of its vertices. Implement the following basic operations in the file `src/exercise01.cpp`:

- scaling

```
static Vertex scale(Vertex v, float sx, float sy, float sz);
```

- translation

```
static Vertex translate(Vertex v, float tx, float ty, float tz);
```

- rotation around $x = (1, 0, 0)^T$, $y = (0, 1, 0)^T$, or $z = (0, 0, 1)^T$

```
static Vertex rotateX(Vertex v, float angle);
static Vertex rotateY(Vertex v, float angle);
static Vertex rotateZ(Vertex v, float angle);
```

All operations should be applied to the given 3D point, where the reference point for the operation is always the origin $(0, 0, 0)^T$. Note that the `Vertex` class is just an alias of the `Point3D` class implemented in `include/point3d.h`.

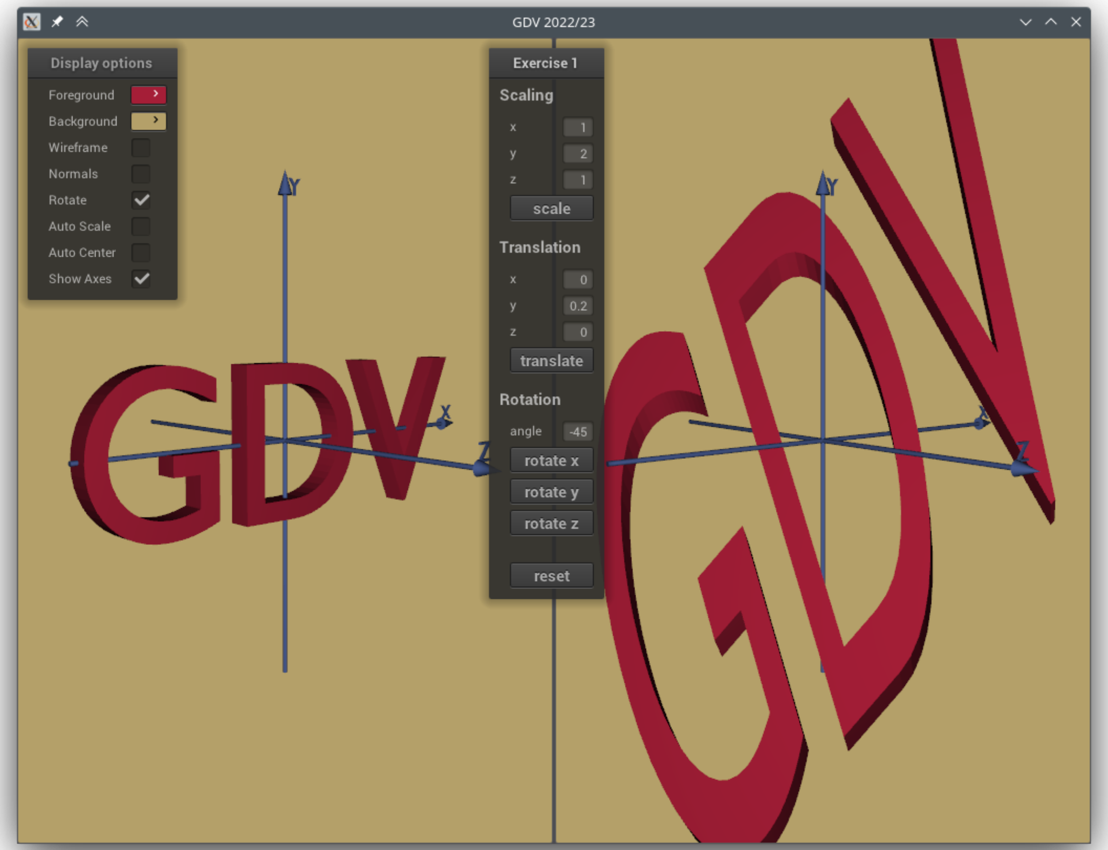
In the given implementation, all operations simply return the input vertex. You can test your methods by transforming the mesh shown on the right half of the screen. The left half of the screen always shows the input mesh. You can reset to the initial state at any time using the "reset" button.

You can see an example of the application in Figure 2.
To compile and run the application, the following commands are necessary (or just use an IDE):

```
mkdir build
cd build
# add -GNinja to use ninja instead of make
cmake .. -DCMAKE_BUILD_TYPE=RelWithDebInfo
# or ninja
make -j8
# make sure to run the program in the build folder, or it will not find its meshes
./exercise01
```

@Windows users: you have to add `"currentDir": "${projectDir}\\out"` to your `launch.vs.json` configuration to run the program in the correct folder.

For your submission, simply upload the `exercise01.cpp` file. You don't have to (and should not) touch any other file.


GDV 2022/23

Display options

Foreground
Background
Wireframe
Normals
Rotate ✓
Auto Scale
Auto Center
Show Axes ✓

Exercise 1

Scaling
x 1
y 2
z 1

Translation
x 0
y 0.2
z 0

Rotation
angle -45

Figure 2: The GUI of the demo application with the input mesh on the left and the transformed mesh on the right. In the display options menu, you can toggle various options to help with debugging.