



Prof. Dr.-Ing. Ralf Steinmetz
Multimedia communications Lab

Dr. Florian Mehm
Dipl. Inf. Robert Konrad



TECHNISCHE
UNIVERSITÄT
DARMSTADT

„Game Technology“ Winter Semester 2016/2017

Exercise 3

For bonus points upload your solutions until **Saturday, November 12, 9:50**

General Information

- The exercises may be solved by teams of up to three people.
- The solutions have to be uploaded to the Git repositories assigned to the individual teams.
- **The submission date (for practical and theoretical tasks) is noted on top of each exercise sheet.**
- If you have questions about the exercises write a mail to game-technology@kom.tu-darmstadt.de or use the forum at <https://www.fachschaft.informatik.tu-darmstadt.de/forum/viewforum.php?f=557>

1. Practical Tasks: Triangle Mesh Rendering (5 Points)

Implement a basic software triangle renderer. Specifically, implement keyboard camera controls by carrying out rotation, translation and projection as described in lecture 3. You may use the `Kore::vec3/4` classes to store data, but not the `Kore` matrix or quaternion classes. As an ungraded bonus, you can try out mouse camera control.

You will find comments in the code that direct you to the correct positions to add your code.

The suggested order is to start with translation first, then the perspective transform, and finally rotations. Note the effect each transformation has on the result. In case the model is not visible anymore after a change you made, check your calculations with some example data, e.g. the point (0, 0, 1), and find out at which position on the screen it ends up in. Keep in mind what your coordinate system is and where your 3D mesh is located (you won't recognize the mesh if your camera is located inside of it). Be especially careful with z values as they can easily mess up your mesh – if that happens find the smallest and largest z value of your mesh and have a close look at what z_{min} / z (for the perspective division) does.

<https://github.com/TUDGameTechnology/Exercise3.git> contains code for triangle rasterizing and mesh loading and a mesh to get you started. You can either copy the code changes manually or just pull them into your own repository using `git pull https://github.com/TUDGameTechnology/Exercise3.git`

Additional Information:

- A video showing the solution for this exercise is available on the website.
- There was a typo in the lecture slides, on slide 25 in line 6, "... + sin(camera.ry) * dz;" should be "... - sin(camera.ry) * dz;" This might have led to some of your rotations behaving unexpectedly.
- The model can end up either very small or very much distorted due to the perspective. In the solution shown in the video, the image is scaled up before `drawTriangle` is called. To achieve this, you can use code like the following:

```
float drawScale = 512.0f;
drawTriangle(
    x1 * drawScale + width / 2, y1 * drawScale + height / 2,
    x2 * drawScale + width / 2, y2 * drawScale + height / 2,
    x3 * drawScale + width / 2, y3 * drawScale + height / 2);
```

Please remember to push into a branch called "exercise3".

2. Theoretical Tasks (5 Points)

2.1 Transformations (2 Points)

a) In the general case, is the order in which only translations are applied to an object relevant? Explain in your own words why or give a counter-example.

b) In the general case, is the order in which any combination of transformations (translation, scale, rotation, shearing) is applied to an object relevant? Explain in your own words why or give a counter-example.

2.2 Angle calculations (2 Point)

You are writing a top-down game in which characters are playing with water guns. You want to write a function to calculate the angle between where a character A is currently facing and where a potential target character B is. The game happens in the X-Z-plane, and Y is up.

A is currently at position $(0, 0, 0)$ and is oriented so that its forward-direction is the normal vector $(1, 0, 0)$. The target is at position $(2, 0, -3)$. Calculate the angle between A's forward direction and the direction towards the target B. Show your calculations as part of the answer.

2.3 Raytracing and Rasterization (1 Point)

Explain in your own words the differences between raytracing and rasterization.