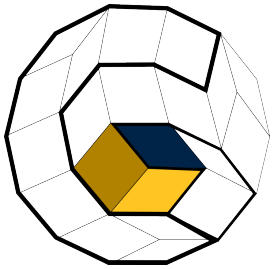
TU BRAUNSCHWEIG
Dipl. Inform. Martin Eisemann
Institut für Computergraphik
Lorenz Rogge (rogge@cg.tu-bs.de)

April 19, 2011



Echtzeit Computergraphik SS 2011
Assignment 2

You have two weeks to complete this assignment. So please present your solution to this exercise on Monday, May 2nd, 2011.

2.1 Extend the OBJ Loader from the last assignment (30 Points)

Implement the now existing OBJLoader class that doesn't just import one 3D mesh, but stores multiple meshes for later use. A `std::map` is used for holding the meshes identified by a string parameter. Later on, the OBJLoader class should return a previously loaded mesh by calling `getMeshObj()`.
To comfortably handle loaded mesh data, the class `MeshObj` has to be completed. It serves as a storage container holding a mesh's vertex information and is able to render itself to screen. Implement all missing methods, especially `setData()` used for copying 3D geometry data and `render()` used for rendering a mesh. In the rendering routine, simply use the `glBegin()` - `glEnd()` style explained in the lecture, instead of trying to implement the use of faster but much more complicated vertex buffer objects.
Using a `MeshObj` it is possible to render the exact same mesh data multiple times without much effort and the OBJLoader provides a convenient storage for many meshes loaded from disk.

2.2 Implement a trackball structure (30 Points)

In this task implement a view controller allowing the user to freely look around and move through the scene. Therefore the three empty GLUT callback functions in `Ex02.cpp` have to be implemented and the class `Trackball` needs to be completed.
To control the viewing direction, capture the mouse position when a mouse button is pressed and compute the difference to the mouse position when moving the mouse with the button pressed.
This 2D difference vector allows to control the rotation in the horizontal plane and an up-down-rotation. See *Figure 1* for a visualization.
The difference in x should control the rotation around the y-axis denoted by the angle θ. Then rotate around the x-axis of the rotated view using the difference in y using the angle ϕ.
Also allow the user to move the camera's origin using the keys W, S, A and D. Moving forward and backward results in a movement corresponding to the viewing direction. Therefore, when looking slightly up, a forward motion results in a change of the y-value of the camera's origin. Moving left or right, however, only results in a motion on the x-z-plane.
To control the viewing of the OpenGL camera, use `gluLookAt(...)`, where the camera's origin are the first three values, the viewing direction from this point as the next three value and finally the up-vector of the camera $(0, 1, 0)$ the last three values. This way, calling the method `rotateView()` before rendering anything, the correct view is established for any later rendering. Combine the GLUT callback functions and the update methods of the trackball to create an interactive rendering application.

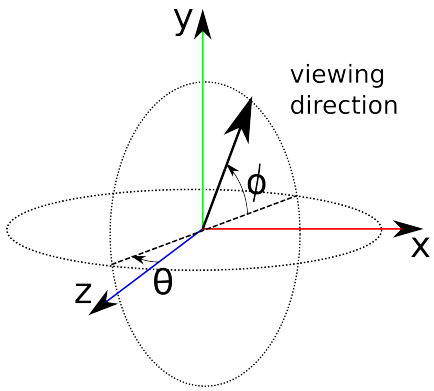


Figure 1: Used angles describing a viewing direction. θ corresponds to a rotation in the x-z-plane, while ϕ describes the tilt around the rotated x-axis.

2.3 Display a test file in 3D (20 Points)

Using your previously implemented OBJLoader it is possible to load and store 3D meshes for later rendering. In the `main()` of `Ex02.cpp` a list of symbols is imported from the directory `./meshes/`. Use these meshes to dynamically display a text in 3D. The *second* parameter passed to `main()` defines a text file to read from. Read in the text file line by line and render the `MeshObj` corresponding to each letter or symbol. Make use of `glTranslatef(x,y,z)` to arrange the letters in lines and every new line below the previous one. The methods `glPushMatrix()` and `glPopMatrix()` allow to control the transformation matrix stack used for model transformations such as translations or rotations. Clever use of these makes letter and line arrangement quite easy. After reading in a given text file, the result should look like this:

Figure 2: A dynamically created text in 3D