Umeå University
Department of Computing Science
Computer Graphics

Stefan Johansson, stefanj@cs.umu.se

Exercise 2

Matrix multiplication routines

Introduction

In the course project, we will need a way to do simple matrix multiplications in our software. The matrices are 4×4 and usually structured which we can use to reduce the number of operations.

Some of the operations we need are shown below. In the lecture about transformations we will cover and deduce these matrices, but you can prepare and start to see how these operations can be done in C++. Below each operation you can see a proposed function header with a given set of parameters. This may of course be altered and complemented.

Translation

$$T \leftarrow \begin{bmatrix} 1 & & & dx \\ & 1 & & dy \\ & & 1 & dz \\ & & & 1 \end{bmatrix} \cdot T$$

```
translate(dx,dy,dz)
```

Scaling

$$T \leftarrow \begin{bmatrix} sx & & & \\ & sy & & \\ & & sz & \\ & & & 1 \end{bmatrix} \cdot T$$

```
scale(sx,sy,sz)
```

Rotation around z-axis

$$T \leftarrow \begin{bmatrix} \cos(\alpha_z) & -\sin(\alpha_z) & & \\ \sin(\alpha_z) & \cos(\alpha_z) & & \\ & & 1 & \\ & & & 1 \end{bmatrix} \cdot T$$

```
rotatez(a)
```

Rotation around y-axis

$$T \leftarrow \begin{bmatrix} \cos(\alpha_y) & & -\sin(\alpha_y) & \\ & 1 & & \\ \sin(\alpha_y) & & \cos(\alpha_y) & \\ & & & 1 \end{bmatrix} \cdot T$$

```
rotatey(a)
```

Rotation around x-axis

$$T \leftarrow \begin{bmatrix} 1 & & & \\ & \cos(\alpha_x) & -\sin(\alpha_x) & \\ & \sin(\alpha_x) & \cos(\alpha_x) & \\ & & & 1 \end{bmatrix} \cdot T$$

```
rotatex(a)
```

A simple search on the web will give several hits for libraries that handle this, both general libraries and specialized for computer graphics. Boost uBlas is a general numeric library and is installed on CS (http://www.boost.org/doc/libs/1_55_0/libs/numeric/ublas/doc/index.htm).

Another library is Generic Graphics Toolkit, GGT (http://ggt.sourceforge.net/).

However, small routines for the multiplications is not difficult to do yourself (but do some basic testing and sanity checks). First thing is to define your data type, or to use those provided by a library. In Workshop 1 and its given code, the Qt classes `QVector2D` and `QMatrix4x4` are used.

Task – Matrix routines

It does not matter if you like to build your own or use existing libraries, but **find a way to do basic matrix operations as described above and find a representation that seems efficient**.

Consider the following:

- It should be fast to copy a matrix and a vector to an array buffer.
- Use float (not double).
- Other routines that may come in handy (but certainly not necessarily) are
  - matrix-vector multiplication,
  - cross-product.
- Matrices are always 4×4.
- Vectors multiplied with a matrix is always 4×1.
- A vertex (a coordinate) is typically 3D or 4D (with the 4^{th} element set to 1) and implemented as a struct (or a class), and it should be possible to copy it directly into an array buffer (for speed).