IOS

Fakultät für Informatik
Labor für Computergrafik
Prof. Dr. G. Umlauf

HTWG
Hochschule Konstanz
University of Applied Sciences

Konstanz, 27.09.2024

Assignment 3

Computer graphics

Deadline 18.12.2024

Preliminary remark: Do not use for this assignment **`OpenGL`**, **`GLUT`**, **`GLAUX`**, or other library-functions for the projection or rotations! You can use the provided vector and matrix classes.

Exercise 5 (Central projection) — 2+1+1+2 points

Implement an application that computes the central projection along the z_v-axis of a simple 3d-scene containing several cuboids:

a. Implement the function

```
CVec4f projectZ(float fFocus, CVec4f pView)
```

for the central projection of an arbitrary 3d-point **`pView`** in homogenous view-coordinates onto the projection plane. Use the setting shown in Figures 1 and 2 where

- the view-origin **`ViewOrigin`** is the origin of the view-coordinate system,
- the view-direction **`ViewDir`** is anti-parallel to the die z_v-axis (and initially also the z_w-axis),
- the view-up-vector **`ViewUp`** (y_v-axis of the image plane) is initially parallel to the y_w-axis,
- the eye-point **`EyePoint`** is on the positive z_v-axis, i.e. (**`0,0,fFocus`**) in view-coordinates,
- the image-plane is the $x_v y_v$-plane,
- the focal-distance **`fFocus`** is the distance of the eye-point to the view-origin, and
- the focal-distance **`fFocus`** is a variable parameter of the function **`projectZ`**.

Initially the world-coordinate system $((0,0,0); x_w, y_w, z_w)$ and the view-coordinate system $(O_v; x_v, y_v, z_v) =$ (**`ViewOrigin`**; **`ViewLeft`**, **`ViewUp`**, **`-ViewDir`**) (aka camera-coordinate system) should have the same coordinates. The view-coordinate system is represented in homogenous world-coordinates.

b. Implement the function

```
void drawProjektedZ (CVec3f Points[8]),
```

that takes eight 2d-points and draws the wireframe of a projected 3d-cuboid. To this end, connect corresponding projected points with lines using the Bresenham algorithm.

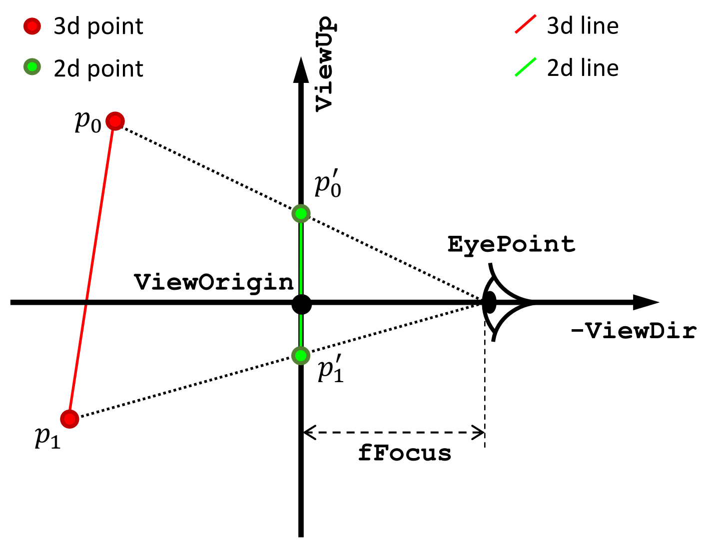

Figure 1 Central projection along the z_v-axis.

c. Define 3d-points for at least three 3d-cuboid. Per cuboid only eight 3d-points need to be stored. Hence, use an array, a structure, or a simple class for the representation of cuboids.

d. Implement the function

```
void drawQuader(CVec3f Cuboid[8],float fFocus, Color c),
```

that
- takes as parameter a cuboid,
- projects the 3d-points using `projectZ(...)` onto the projection plane, and
- draws the respective lines.

Implement these functions in the `display`-function, to display your scene of cuboids.

Exercise 6 (General view) — 7+1 points

So far, the scene is rendered from one perspective only. Here we implement a general view transformation. Thus, extend Exercise 5 to use the following parameters:

- a general position of the eye-point `EyePoint` (in homogenous world-coordinates),
- a general view-direction `ViewDir` (in homogenous world-coordinates), and

- a general view-up-vector **`ViewUp`** (in homogenous world- coordinates).

The eye-point and the two vectors **`ViewDir`** $(= -z_v)$ and **`ViewUp`** $(= y_v)$ define a complete 3d-coordinate system (view-system). The missing x_v-axis (aka **`ViewLeft`** or **`ViewHorizon`**) is computed via $y_v \times z_v$.

a. Implement the function

```
CMat4f getTransform(CVec4f ViewOrigin, CVec4f ViewDir,CVec4f ViewUp),
```

that computes the 4×4- transformation-matrix to converts view-coordinates to world-coordinates, see Figure 2. The inverse of this matrix transforms world-coordinates to view-coordinates.

b. Implement the function

```
CVec4f projectZallg(CMat4f matTransf, float fFocus, CVec4f pWorld),
```

that transforms the point **`pWorld`** in world coordinates via **`matTransf`** to view-coordinates and projects it onto the image plane using **`projectZ`**.

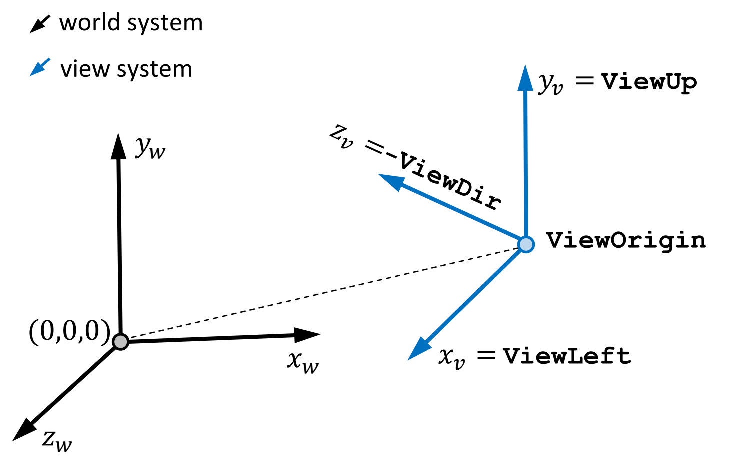

Figure 2 General view transformation.

Exercise 7 (Combination of 5&6) 1+3+3+2+1 points

Combine the functions from Exercises 5 and 6 into one application. Using the functions from Exercise 6 the scene from Exercise 5 is rendered from an arbitrary perspective given the view-coordinate system. Missing is a method to manipulate the view-coordinate system. Use here a simple keyboard-interaction realizing the following key assignments:

a. **`F`** increases the focal-distance and **`f`** decreases the focal-distance.

b. **X**, **Y**, and **Z** rotate the view-coordinate system in positive direction around x_w-, y_w- and z_w-axes of the world-coordinate system and **x**, **y** and **z** rotate in negative direction around the respective axes.
c. **A**, **B**, and **C** respectively **a**, **b**, and **c** rotate the view-coordinate system in respective direction around the respective axes of the view-coordinate system (**A**, **a**: view direction, **B**, **b**: view-up-vector, . . .).
d. **U**, **V**, **W**, **u**, **v**, and **w** translate the view-coordinate system along the ayes of the world-coordinate system in respective directions (**U**, **u**: x_w-axis, **V**, **v**: y_w-axis, **W**, **w**: z_w-axis).
e. **R** resets the view-coordinate system to its initial position (congruent to the world-coordinate system).