Übungen - Bildgenierung

Übung 04.

Jose Jimenez

Angewandte Informatik
Bergische Universität Wuppertal

November 23, 2022

Table of Contents

Aufgabe 10 Perspektivische Projektion

Gefüllte Polygone

Programmieren Sie die perspektivische Projektion, die Überführung in normalisierte Koordinaten sowie die Umwandlung in Gerätekoordinaten. Ergänzen Sie hierzu im Rahmenprogramm proj1.cc im Verzeichnis /home/bildgen/Aufgaben/projektion-1 die entsprechenden Teile der Funktionen ....

Aufgabe 10 Perspektivische Projektion

Rahmen Program

```
Matrix4x4 berechneTransformation(const Vec3D& cop, const Vec3D& vrp,
                                 const Vec3D& vup, int w, int h,
                                 double& un, double& vn)
```

```
void maleLinien(Drawing& pic, const vector<Kante>& kanten,
                const Matrix4x4& t, double un, double vn)
```

Aufgabe 10 Perspektivische Projektion

Rahmen Program

```
Matrix4x4 berechneTransformation(const Vec3D& cop, const Vec3D& vrp,
                                 const Vec3D& vup, int w, int h,
                                 double& un, double& vn)
```

```
void maleLinien(Drawing& pic, const vector<Kante>& kanten,
                const Matrix4x4& t, double un, double vn)
```

Was ist neu für uns?

Aufgabe 10 Perspektivische Projektion

Rahmen Program

```
Matrix4x4 berechneTransformation(const Vec3D& cop, const Vec3D& vrp,
                                 const Vec3D& vup, int w, int h,
                                 double& un, double& vn)
```

```
void maleLinien(Drawing& pic, const vector<Kante>& kanten,
                const Matrix4x4& t, double un, double vn)
```

Was ist neu für uns?

- Matrix $4 \times 4 \rightarrow$ doc
- Vec3D $\rightarrow$ doc
- Kante $\rightarrow$

see doc

Aufgabe 10 Perspektivische Projektion

Rahmen Program

1. Verschiebung von vrp in den Ursprung
Das haben wir beim letzten Mal gelernt. Quasi.

Heute brauchen wir nur eine Verschibung. Was ist der T-Matrix? Wie können wir in C++ schreiben?

Aufgabe 10 Perspektivische Projektion

Rahmen Program

1. Verschiebung von vrp in den Ursprung
Das haben wir beim letzten Mal gelernt. Quasi.

Heute brauchen wir nur eine Verschibung

```
tvrp.el[0][0] = tvrp.el[1][1] = tvrp.el[2][2] = tvrp.el[3][3] = 1;
 tvrp.el[0][3] = -vrp.el[0];
 tvrp.el[1][3] = -vrp.el[1];
 tvrp.el[2][3] = -vrp.el[2];
```

Aufgabe 10 Perspektivische Projektion

Rahmen Program

2. a) Bestimme (u,v,n)-Koordinatensystem (Seiten 4-15 —-4-18)

Aufgabe 10 Perspektivische Projektion

Rahmen Program

2. a) Bestimme (u,v,n)-Koordinatensystem (Seiten 4-15 —-4-18) Wir brauchen n,v und u.

- n ist der Normalenvektor, und wir haben ihn schon. (vpn).

```
n = vpn / norm(vpn);
```

Aufgabe 10 Perspektivische Projektion

Rahmen Program

2. a) Bestimme (u,v,n)-Koordinatensystem (Seiten 4-15 —-4-18) Wir brauchen n,v und u.

- n ist der Normalenvektor, und wir haben ihn schon. (vpn).
- v:Orthogonalprojektion von up auf Projektionsebene.

```
n = vpn / norm(vpn);

v = vup - skalarprod(vup, n) * n;
v = v / norm(v);
```

Aufgabe 10 Perspektivische Projektion

Rahmen Program

2. a) Bestimme (u,v,n)-Koordinatensystem (Seiten 4-15 —-4-18) Wir brauchen n,v und u.

- n ist der Normalenvektor, und wir haben ihn schon. (vpn).
- v:Orthogonalprojektion von up auf Projektionsebene.
- Die u-Achse wird so gewählt, dass (u,v,n) (normierte Vektoren) ein rechtshändiges 3D-Koordinatensystem bilden

```
n = vpn / norm(vpn);

v = vup - skalarprod(vup, n) * n;
v = v / norm(v);

u = kreuzprod(v, n);
```

Aufgabe 10 Perspektivische Projektion

Rahmen Program

2. b) Rotation: (x, y, z) (u, v, n) (Seite 4-18, 2.) Das ist natürlich eine matrix.

Aufgabe 10 Perspektivische Projektion

Rahmen Program

2. b) Rotation: (x, y, z) (u, v, n) (Seite 4-18, 2.)

```
 /*
  / ux uy uz   \
  | vx vy vz   |
  | nx ny nz   |
  \          1 /
*/
for (i = 0; i < 3; ++i)
  {
    rot.el[0][i] = u.el[i];
    rot.el[1][i] = v.el[i];
    rot.el[2][i] = n.el[i];
  }
rot.el[3][3] = 1;
```

Aufgabe 10 Perspektivische Projektion

Rahmen Program

3) Verschieben des transformierten Augenpunktes in den Ursprung (Seite 4-19, 3.).
Wie sieht die Matrix aus? (z_p ist $Z_n in Rahmen Programm$).
$Der Augenpunkt liegt nicht im Ursprung$, $sondern an Position$(0, 0, -z_p) (4-17)
(z_p ist Z_n in Rahmen Programm).

Aufgabe 10 Perspektivische Projektion

Rahmen Program

3) Verschieben des transformierten Augenpunktes in den Ursprung (Seite 4-19, 3.).
Wie sieht die Matrix aus? (z_p ist Z_n*inRahmenProgramm*).
DerAugenpunktliegtnichtimUrsprung, *sondernanPosition*(0, 0, -z_p) (4-17)

```
  /*
  / 1          \
  |   1        |
  |     1 -zn  |
  \       1  /
*/
transcop.el[0][0] = transcop.el[1][1] = transcop.el[2][2] = transcop.el[
  = 1;
transcop.el[2][3] = -znear;
```

Aufgabe 10 Perspektivische Projektion

Rahmen Program

4) Standard perspektivische Projektion (Seite 4-24)

Aufgabe 10 Perspektivische Projektion

Rahmen Program

4) Standard perspektivische Projektion (Seite 4-24)

```
/*
   / -zn           \
   |      -zn      |
   |               |
   \          1    /
*/
proj.el[0][0] = proj.el[1][1] = -znear;
proj.el[2][2] = 0;
proj.el[3][2] = 1;
```

Aufgabe 10 Perspektivische Projektion

Rahmen Program

5. b) Translation des Bereichs [umin; umax] $\times$ [vmin; vmax] so dass (umin, vmin) $\rightarrow$ (0, 0)

Aufgabe 10 Perspektivische Projektion

Rahmen Program

5. b) Translation des Bereichs [umin; umax] $\times$ [vmin; vmax] so dass (umin, vmin) $\rightarrow$ (0, 0)

```
/*
  / 1      -umin \
  |   1    -vmin |
  |     ?        |
  \          1   /
*/
t.el[0][0] = t.el[1][1] = t.el[3][3] = 1;
t.el[2][2] = 0;
t.el[0][3] = -umin;
t.el[1][3] = -vmin;
```

Aufgabe 10 Perspektivische Projektion

Rahmen Program

5. b) Skalierung, so dass das Fenster in das Einheitsquadrat passt.

```
double ft;          // finaler Streckfaktor
```

Aufgabe 10 Perspektivische Projektion

Rahmen Program

5. b) Skalierung, so dass das Fenster in das Einheitsquadrat passt.

```
/*
  / ft         \
  |    ft      |
  |       ?    |
  \         1 /
*/
double fu = 1.0 / (umax - umin); // Faktor für u
double fv = 1.0 / (vmax - vmin); // Faktor für v
double ft = min(fu, fv);         // finaler Streckfaktor
s.el[0][0] = s.el[1][1] = ft;
//s.el[2][2] = 0; // Die z-Koordinate interessiert uns hier nicht
s.el[3][3] = 1;
```

Aufgabe 10 Perspektivische Projektion

Rahmen Program

6. Multiplikation der einzelnen Transformationen.
Einfach: Wir haben alle Transformationsmatrizen...

s: Skalierung.

t: Translation des Bereichs [umin; umax] $\times$ [vmin; vmax].

proj: Standard perspektivische Projektion.

transcop: Verschieben des transformierten Augenpunktes in den Ursprung

rot: Rotation

trvp: Verschiebung von vrp in den Ursprung

Aufgabe 10 Perspektivische Projektion

Rahmen Program

6. Multiplikation der einzelnen Transformationen.
Einfach: Wir haben alle Transformationsmatrizen...

```
double ft;          // finaler Streckfaktor
mzen =  s * t * proj * transcop * rot * tvrp;
```

Aufgabe 10 Perspektivische Projektion

Rahmen Program

Zum Zeichnen des Bildes ist dann noch Folgendes in maleLinien() zu tun:

1. Anwenden der Transformationsmatrix auf Anfangs- und Endpunkt der einzelnen Linien (der entsprechende Code-Abschnitt ist vorgegeben).
2. Umwandlung der homogenen Koordinaten in 2D-Koordinaten.
3. Skalierung auf Fenstergröße (Gerätekoordinaten) unter Verwendung von u_m und v_n

Aufgabe 10 Perspektivische Projektion

Rahmen Program

Zum Zeichnen des Bildes ist dann noch Folgendes in maleLinien() zu tun:

1. Anwenden der Transformationsmatrix auf Anfangs- und Endpunkt der einzelnen Linien (der entsprechende Code-Abschnitt ist vorgegeben).
2. Umwandlung der homogenen Koordinaten in 2D-Koordinaten.
3. Skalierung auf Fenstergröße (Gerätekoordinaten) unter Verwendung von u_m und v_n

Umwandlung der homogenen Koordinaten in 2D-Koordinaten.

```
void maleLinien(Drawing& pic, const vector<Kante>& kanten,
                const Matrix4x4& t, double un, double vn){
    Vec4D nanf, nend;     // Anfangs- und Endpunkt nach Normalisierung
    DPoint2D panf, pend; // Anfangs- und Endpunkt im Bild pic
    ...
    // 2. Perspektivendivision, Umwandlung von homogenen in 2D-Koordinaten
    // *** Hinweise:
    //     nanf -> panf, nend -> pend
    //     Zugriff auf die Elemente von nanf, nend: nanf.el[#], # aus [0,3]
    //     Zugriff auf die Elemente von panf, pend: panf.x, panf.y
    }
```

Aufgabe 10 Perspektivische Projektion

Rahmen Program

Umwandlung der homogenen Koordinaten in 2D-Koordinaten.

```
void maleLinien(Drawing& pic, const vector<Kante>& kanten,
                const Matrix4x4& t, double un, double vn){
    Vec4D nanf, nend;     // Anfangs- und Endpunkt nach Normalisierung
    DPoint2D panf, pend; // Anfangs- und Endpunkt im Bild pic
    ...
      panf.x = nanf.el[0] / nanf.el[3];
      panf.y = nanf.el[1] / nanf.el[3];
      pend.x = nend.el[0] / nend.el[3];
      pend.y = nend.el[1] / nend.el[3];
    }
```

Aufgabe 10 Perspektivische Projektion

Rahmen Program

Skalierung auf Fenstergröße (Gerätekoordinaten) unter Verwendung von u_m und v_n . Die Pixel liegen in beiden Richtungen gleich dicht.
Auch einfach:

```
void maleLinien(Drawing& pic, const vector<Kante>& kanten,
                const Matrix4x4& t, double un, double vn){

// Skalierungsfaktoren:
    double sx = pic.getWidth() / un;  //Breite
    double sy = pic.getHeight() / vn;  //Höhe

    DPoint2D panf, pend; // Anfangs- und Endpunkt im Bild pic
    ...
      panf.x *= sx;
      panf.y *= sy;
      pend.x *= sx;
      pend.y *= sy;
    }
```

Aufgabe 11 Strecken-Clipping nach Cohen und Sutherland

Algorithmus

Ihr habt den Algorithmus schon gelernt, Ihr muss nur es anwenden.

Rechteck $[2;8]x[1;5]$ **Linien**

a) $P_1 = \begin{pmatrix}1\\2\end{pmatrix}, P_2 = \begin{pmatrix}10\\4\end{pmatrix}$

b) $P_1 = \begin{pmatrix}3\\3\end{pmatrix}, P_2 = \begin{pmatrix}6\\0\end{pmatrix}$

c) $P_1 = \begin{pmatrix}7\\0\end{pmatrix}, P_2 = \begin{pmatrix}10\\2\end{pmatrix}$

d) $P_1 = \begin{pmatrix}5\\6\end{pmatrix}, P_2 = \begin{pmatrix}11\\4\end{pmatrix}$

Aufgabe 11 Strecken-Clipping nach Cohen und Sutherland

Algorithmus

Rechteck $[2;8]x[1;5] \rightarrow \overline{u} = 8, \quad \overline{v} = 5, \quad \underline{u} = 2, \quad \underline{v} = 1.$

Linien

a) $P_1 = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, P_2 = \begin{pmatrix} 10 \\ 4 \end{pmatrix}$

b) $P_1 = \begin{pmatrix} 3 \\ 3 \end{pmatrix}, P_2 = \begin{pmatrix} 6 \\ 0 \end{pmatrix}$

c) $P_1 = \begin{pmatrix} 7 \\ 0 \end{pmatrix}, P_2 = \begin{pmatrix} 10 \\ 2 \end{pmatrix}$

d) $P_1 = \begin{pmatrix} 5 \\ 6 \end{pmatrix}, P_2 = \begin{pmatrix} 11 \\ 4 \end{pmatrix}$

$\text{code1} = (u1 > \overline{u}, v1 > \overline{v}, u1 < \underline{u}, v1 < \underline{v})$

$\text{code2} = (u2 > \overline{u}, v2 > \overline{v}, u2 < \underline{u}, v2 < \underline{v})$

Aufgabe 11 Strecken-Clipping nach Cohen und Sutherland

Algorithmus

Rechteck $[2;8]x[1;5] \rightarrow \overline{u} = 8, \quad \overline{v} = 5, \quad \underline{u} = 2, \quad \underline{v} = 1.$

Linien

d) $P_1 = \begin{pmatrix} 5 \\ 6 \end{pmatrix}, P_2 = \begin{pmatrix} 11 \\ 4 \end{pmatrix}$

$\text{code1} = (u1 > \overline{u}, v1 > \overline{v}, u1 < \underline{u}, v1 < \underline{v})$

$\text{code2} = (u2 > \overline{u}, v2 > \overline{v}, u2 < \underline{u}, v2 < \underline{v})$

Aufgabe 11 Strecken-Clipping nach Cohen und Sutherland

Algorithmus

Rechteck $[2;8]x[1;5] \rightarrow \overline{u} = 8, \quad \overline{v} = 5, \quad \underline{u} = 2, \quad \underline{v} = 1.$

Linien

b) $P_1 = \begin{pmatrix} 3 \\ 3 \end{pmatrix}, P_2 = \begin{pmatrix} 6 \\ 0 \end{pmatrix}$

c) $P_1 = \begin{pmatrix} 5 \\ 6 \end{pmatrix}, P_2 = \begin{pmatrix} 11 \\ 4 \end{pmatrix}$

Aufgabe 12 Strecken-Clipping nach Cyrus, Beck, Liang und Barsky

Algorithmus

Ihr habt den Algorithmus schon gelernt, Ihr muss nur es anwenden.

Octave script

Rechteck $[2;8]x[1;5]$ **Linien**

a) $P_1 = \begin{pmatrix}1\\2\end{pmatrix}$, $P_2 = \begin{pmatrix}10\\4\end{pmatrix}$

b) $P_1 = \begin{pmatrix}3\\3\end{pmatrix}$, $P_2 = \begin{pmatrix}6\\0\end{pmatrix}$

c) $P_1 = \begin{pmatrix}7\\0\end{pmatrix}$, $P_2 = \begin{pmatrix}10\\2\end{pmatrix}$

d) $P_1 = \begin{pmatrix}5\\6\end{pmatrix}$, $P_2 = \begin{pmatrix}11\\4\end{pmatrix}$