Übungen - Bildgenierung

Übung 04.

Jose Jimenez

Angewandte Informatik
Bergische Universität Wuppertal

November 27, 2024

Table of Contents

Aufgabe 13: 3D-Clipping für perspektivische Projektion

Also, jetzt finden wir die **Werte von** t, bei denen wir einen **Schnittpunkt mit den Clipping-Ebenen** in normalisierten Sichtkoordinaten haben.

Aufgabe 13: 3D-Clipping für perspektivische Projektion

Also, jetzt finden wir die **Werte von** t, bei denen wir einen **Schnittpunkt mit den Clipping-Ebenen** in normalisierten Sichtkoordinaten haben.

Was ist ΔP?
ΔP ist der Vektor, der die Richtung und Länge der Linie beschreibt. Wenn die Linie zwei Punkte $P_0 = (x_0, y_0, z_0)$ und $P_1 = (x_1, y_1, z_1)$ hat, dann ist:

$$\Delta P = P_1 - P_0 = (x_1 - x_0, y_1 - y_0, z_1 - z_0).$$

Das bedeutet, ΔP zeigt, wie weit und in welche Richtung wir uns von P_0 nach P_1 bewegen.

Aufgabe 13: 3D-Clipping für perspektivische Projektion

Wir implementiereng 2 Hilfsfunktionen für **a** und **b**.

- Rahmen Program
- true =: full oder teilweise sichtbar
- false =: nicht sichtbar

Aufgabe 13: 3D-Clipping für perspektivische Projektion

Clipping für Fall a.

```
bool clip3D1(Vec3D& anf, Vec3D& end, double t1, double t2){
  // Clipping-Hilfsfunktion
  Vec3D delta;
  if (t1 > t2) //Annahme
    swap(t1, t2);
  if (t1 > 1 || t2 < 0)   //ii) β γ, zurückweisen
    return false;

  // i) iii) iv)
  delta = end - anf;  //P1- P0
  if (t2 < 1)
    end = anf + t2 * delta; //P1 = P0 +t2ΔP
  if (t1 > 0)
    anf += t1 * delta;   //P0 = P0 +t1ΔP
  return true;
}
```

Aufgabe 13: 3D-Clipping für perspektivische Projektion

Clipping für Fall b.

```
bool clip3D2(Vec3D& anf, Vec3D& end, double t1, double t2){
  // Clipping-Hilfsfunktion
  Vec3D delta;

  delta = end - anf;
  double z1 = anf.el[2] + t1 * delta.el[2];
  double z2 = anf.el[2] + t2 * delta.el[2];
  if (z1 > 0 && z2 > 0)          //i)
    return false;
  if (z1 <= 0 && z2 <= 0)        //ii)
    return clip3D1(anf, end, t1, t2);
  double tu = min(t1, t2);       // unteres t
  if (tu < 0)                    //iii) alpha)
    return false;
  if (tu < 1)                     //iii) gamma)
    end = anf + tu * delta;
  return true;
}
```

Aufgabe 13: 3D-Clipping für perspektivische Projektion

clip3D implementierung

Wir sollen nur

```
bool clip3D(Vec3D& anf, Vec3D& end, double zmin)
```

implementieren.
Das heißt, wir übersetzen, was an der Tafel steht, in C.

Aufgabe 14: Modellierung mit OpenGL

OpenGL

Idee...

Wir zeichnen einen Würfel, dann Teile, dann einen Schneemann und dann Schneemänner...

```
void drawCube(const DrawColour& col)
void drawSnowmanParts()
void drawSnowman()
void drawSnowmen()
```

Aufgabe 14: Modellierung mit OpenGL

OpenGL

Idee...

Wir zeichnen einen Würfel, dann Teile, dann einen Schneemann und dann Schneemänner...

```
void drawCube(const DrawColour& col) //was macht die Funktion?
```

Push und Pop

Es gibt einen Stapel von Transformationsmatrizen... Wir werden nur ein Element in unserem Stapel behalten.

```
void drawSnowmanParts(){

    glPushMatrix();
    /*.
    . Verschibung oder Skalierung   (auch Drehung)
    .*/
     glPopMatrix();
}
```

unterer Würfel... Aber zuerst Farben!

```
void drawSnowmanParts(){
  DrawColour white(255, 255, 255);
  DrawColour black(0, 0, 0);
  DrawColour orange(255, 45, 0);
  }
```

Aufgabe 14: Modellierung mit OpenGL

OpenGL

unterer Würfel

```
void drawSnowmanParts(){
  glPushMatrix();
  glTranslatef(0.0f, 1.0f, 0.0f);
  drawCube(white);
  glPopMatrix();
}
```

Mitlere Würfel und Obere Würfel. Hinweis: glScalef(x, y, z);
Daran denken. 5 min.

Aufgabe 14: Modellierung mit OpenGL

OpenGL

unterer Würfel

```
void drawSnowmanParts(){
  glPushMatrix();
  glTranslatef(0.0f, 1.0f, 0.0f);
  drawCube(white);
  glPopMatrix();
}
```

Mitlere Würfel und Obere Würfel. Hinweis: glScalef(x, y, z);
Daran denken. 5 min. PDF

unterer Würfel

```
void drawSnowmanParts(){
  glPushMatrix();
  glTranslatef(0.0f, 1.0f, 0.0f);
  drawCube(white);
  glPopMatrix();
    // mittlerer Würfel
  glPushMatrix();
  glTranslatef(0.0f, 2.8f, 0.0f);
  glScalef(0.8f, 0.8f, 0.8f);
  drawCube(white);
  glPopMatrix();

}
```

Hut

```
void drawSnowmanParts(){
  // Hut
  glPushMatrix();
  glTranslatef(0.0f, 4.9f, 0.0f);
  glScalef(0.8f, 0.2f, 0.8f);
  drawCube(black);
  glPopMatrix();
  glPushMatrix();
  glTranslatef(0.0f, 5.3f, 0.0f);
  glScalef(0.6f, 0.6f, 0.6f);
  drawCube(black);
  glPopMatrix();
}
```

Snow-men

```
void drawSnowmen()
{
  glPushMatrix();
  glScalef(1.5f, 1.5f, 1.5f);
  for (int i = 0; i < 9; ++i)
    {
      glPushMatrix();

      glTranslatef(i % 3 - 1, 0.0f, i / 3 - 1);
      glScalef(0.2f, 0.2f, 0.2f);
      drawSnowman();

      glPopMatrix();
    }

  glPopMatrix();
}
```