Übungen - Bildgenierung

Übung 04.

Jose Jimenez

Angewandte Informatik
Bergische Universität Wuppertal

November 30, 2022

Table of Contents

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(bool outline)
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?
```

Aufgabe 14: Modellierung mit OpenGL

OpenGL

Push und Pop

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

```
void drawSnowmanParts(bool outline){

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

unterer Würfel... Aber zuerst Farben!

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

Mitlere Würfel und Obere Würfel. Hinweis: glScalef(x, y, z);

Hut

```
void drawSnowmanParts(bool outline){
  // 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();
}
```

outline

```
void drawSnowmanParts(bool outline){
  // Outline?
}
```

outline

```
void drawSnowmanParts(bool outline){
  /* Outline?
  Wenn x wahr ist, dann zeichnen wir nur die Linien der Polygone.
Andernfalls zeichnen wir farbige Polygone.

  */
}
```

outline

```
//Am amfang:
void drawSnowmanParts(bool outline){
/*  Wenn x wahr ist, dann zeichnen wir nur die Linien der Polygone.
Andernfalls zeichnen wir farbige Polygone.*/
if (outline)
    {
      glPolygonMode(GL_FRONT,GL_LINE);
      glPolygonMode(GL_BACK,GL_LINE);
      glPolygonOffset(2, 2);
      glLineWidth(1.5);
      white = DrawColour(0, 0, 0);
      black = DrawColour(255, 255, 255);
      orange = DrawColour(0, 0, 0);
    }
    //glPolygonOffset: nützlich für das Rendern von Solids mit hervorgehob
}
```

outline

```
void drawSnowmanParts(bool outline){
 if (outline)
  {
    glPolygonMode(GL_FRONT,GL_FILL);
    glPolygonMode(GL_BACK,GL_FILL);
  }
```

Ok, Nice! Jetz Implementieren!

Clipping für Fall 1.

Aufgabe 13: 3D-Clipping für perspektivische Projektion

Clipping für Fall 1.

```
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)   //nichts zu tun
    return false;

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

Clipping für Fall 2.

Aufgabe 13: 3D-Clipping für perspektivische Projektion

Clipping für Fall 2.

```
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

Parameterdarstellung.

ParameterDarstellung.

Wir brauchen die Schnittpositionen t_1 und t_2