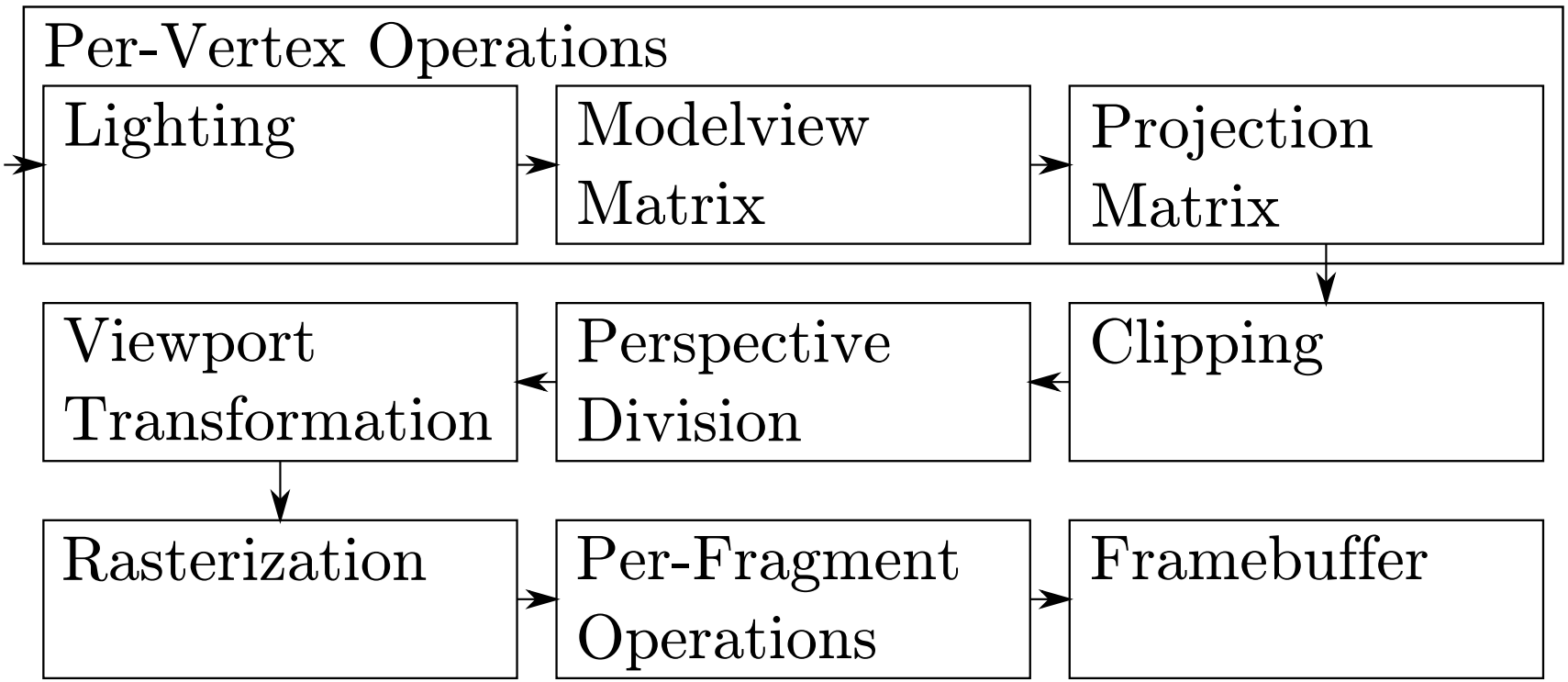
Dreiecke verzerren mit OpenGL ES 1.1

Philipp Klaus Krause

25. Juni 2019

Inhalt

OpenGL ES 1.1 Graphikpipeline



Koordinatentransformation

- Jede Ecke hat 4 Koordinaten (wenn nicht angegeben, ist die 3. 0, die 4. 1), kann also als Vektor mit 4 Einträgen aufgefasst werden (typischerweise in einem dreidimensionalem projektiven Raum).
- Diese werden mit Matrizen multipliziert: Zuerst mit der Model-View Matrix, dann der Projection Matrix.
- Danach erfolgt eine Perspective Division (wir erhalten nun Koordinaten in einem dreidimensionalen Raum).
- Zuletzt gibt es noch eine Viewport Transformation, um Koordinaten im Fenster zu erhalten (links unten liegt $(0, 0)$).

Matrizen

Wähle Matrix aus, ändere diese mit Funktionen:

- glMatrixMode: Auswahl
- glLoadMatrixf: Matrix ersetzen
- glLoadIdentity: Matrix durch Einheitsmatrix ersetzen
- glPushMatrix: Matrix auf Matrixstack sichern
- glPopMatrix: Matrix vom Stack wiederherstellen
- glMultMatrix: Matrix mit anderer Multiplizieren
- glOrthof: Matrix mit Matrix für Orthogonale Projektion multiplizieren
- glRotatef: Matrix mit Rotationsmatrix Multiplizieren
- glTranslatef: Matrix mit Verschiebematrix multiplizieren
- glScalef: Matrix mit Skalierungsmatrix multiplizieren

glMatrixMode

```
void MatrixMode(GLenum mode);
```

- GL_PROJECTION für Projection Matrix, GL_MODELVIEW für Modelview Matrix
- Alle folgenden Funktionen zur Änderung von Matrizen beziehen sich auf die zuletzt so gewählte.

glLoadMatrixf, glLoadIdentity

```
void glMatrixMode(GLenum mode);
void glLoadIdentity(GLenum mode);
```

- Matrizen werden als Felder gespeichert, spaltenweise.
- Alternativ kann man sich vorstellen, dass die Matrizen zeilenweise gespeichert werden, und die Vektoren von links an die Matrix multipliziert werden.

Die folgenden beiden Funktionsaufrufe bewirken das selbe:

```
const GLfloat id[16] = {
  1.0f, 0.0f, 0.0f, 0.0f,
  0.0f, 1.0f, 0.0f, 0.0f,
  0.0f, 0.0f, 1.0f, 0.0f,
  0.0f, 0.0f, 0.0f, 1.0f};
glLoadMatrixf(id);
glLoadIdentity();
```

glPushMatrix, glPopMatrix

- Für jede Matrix gibt es einen Stapel, auf den Matrizen gespeichert werden können
- Nützlich, um ein einzelnes Objekt mit anderen Matrizen (verschoben, skaliert, etc) zu zeichen, und danach wie zuvor weiterzumachen.

Die folgenden beiden Funktionsaufrufe bewirken das selbe:

```
glLoadMatrixf(m1);
...
glPushMatrix()
glLoadMatrixf(m1);
...
glPopMatrix()
// Hier ist m2 wieder die aktuelle Matrix.
```

glOrthof

```
void glOrthof(GLfloat l, GLfloat r, GLfloat b,
  GLfloat t, GLfloat n, GLfloat f);
```

Matrix für orthogonale Projektion (z.B. für 2D- oder isometrische Graphik). l gibt die x-Koordinate des linken Rands an, r die des rechten. b gibt die y-Koordinate des unteren Rands an, t die des oberen. Für ein Fenster mit Seitenverhältnis `ratio` oberen Rand bei 1, unteren bei -1 wählen:

```
glOrthof(-ratio, ratio, -1.0f, 1.0f, 1.0f, -1.0f);
```

glRotatef

```
void glRotatef(GLfloat t, GLfloat x, GLfloat y, GLfloat z);
```

Rotation um t Grad gegen den Uhrzeigersinn um (x, y, z). Drehung in der Ebene um 45°:

```
glRotatef(45.0f, 0.0f, 0.0f, 1.0f);
```

glTranslatef

```
void glTranslatef(GLfloat x, GLfloat y, GLfloat z);
```

Verschiebung um (x, y, z). Verschiebung um 1 in x-Richtung:

```
void glTranslatef(1.0f, 0.0f, 0.0f);
```

glScalef

```
void glScalef(GLfloat x, GLfloat y, GLfloat z);
```

Skalierung um x in x-Richtung, etc. Uniform skalieren mit Faktor 2:

```
void glScalef(2.0f, 2.0f, 2.0f);
```

Inhalt

GLFW

GLFW bietet Plattformunabhängig:

- Erstellen von Fenstern mit OpenGL ES-Kontext
- Tastatureingabe
- Mauseingabe

glfwSetKeyCallback, glfwSetCharCallback

```
GLFWkeyfun glfwSetKeyCallback(GLFWwindow *window, GLFWkeyfun cbfun);
GLFWcharfun glfwSetCharCallback(GLFWwindow *window, GLFWcharfun cbfun);
```

Setzt Callbacks für Tastaturerignisse. `glfwSetKeyCallback` für Tastendrücke (Codes entsprechen US-Tastaturbelegung), `glfwSetCharCallback` für Zeicheneingabe. Beispiel:

```
void key_click(GLFWwindow *window,
  int key, int scancode, int action, int mods)
{
  if (action == GLFW_PRESS && key == GLFW_KEY_MINUS)
    printf("Minustaste gedrückt.\n");
}
void char_click(GLFWwindow *window, unsigned int codepoint)
{
  if (codepoint == L'-')
    printf("Minus eingegeben.\n");
}
…
glfwSetKeyCallback (w, &key_click);
glfwSetCharCallback(w, &char_click);
```

glfwGetCursorPos, glfwGetFramebufferSize

```
void glfwGetCursorPos(GLFWwindow *window,
  double *xpos, double *ypos);
void glfwGetFramebufferSize(GLFWwindow *window,
  int *width, int *height);
```

`glfwGetCursorPos` schreibt die aktuelle Cursorposition nach `*xpos` und `*ypos`, wobei $(0, 0)$ die obere linke Ecke des Fensters ist (also anders als bei OpenGL ES). `glfwGetFramebufferSize` schreibt die Größe des Framebuffers nach `*width` und `*height`.

glfwSetMouseButtonCallback

```
GLFWmousebuttonfun
glfwSetMouseButtonCallback(GLFWwindow *window,
  GLFWmousebuttonfun cbfun);
```

Setzt ein Callback für Ereignisse von Maustasten. Beispiel:

```
void mouse_click(GLFWwindow* window,
  int button, int action, int mods)
{
  if (action == GLFW_PRESS &&
    (button == GLFW_MOUSE_BUTTON_LEFT || GLFW_MOUSE_BUTTON_RIGHT))
  {
    double x, y;
    glfwGetCursorPos(window, &x, &y);
    printf("%s Maustaste bei (%f, %f) gedrückt.\n",
      button == GLFW_MOUSE_BUTTON_LEFT ? "Linke" : "Rechte", x, y);
  }
}
…
glfwSetMouseButtonCallback (w, &mouse_click);
```