

Prof. Andreas Butz | Dipl.Inf. Otmar Hilliges

# Programmierpraktikum 3D Computer Grafik

Einführung die Computergrafik:  
GLUT und OpenGL.





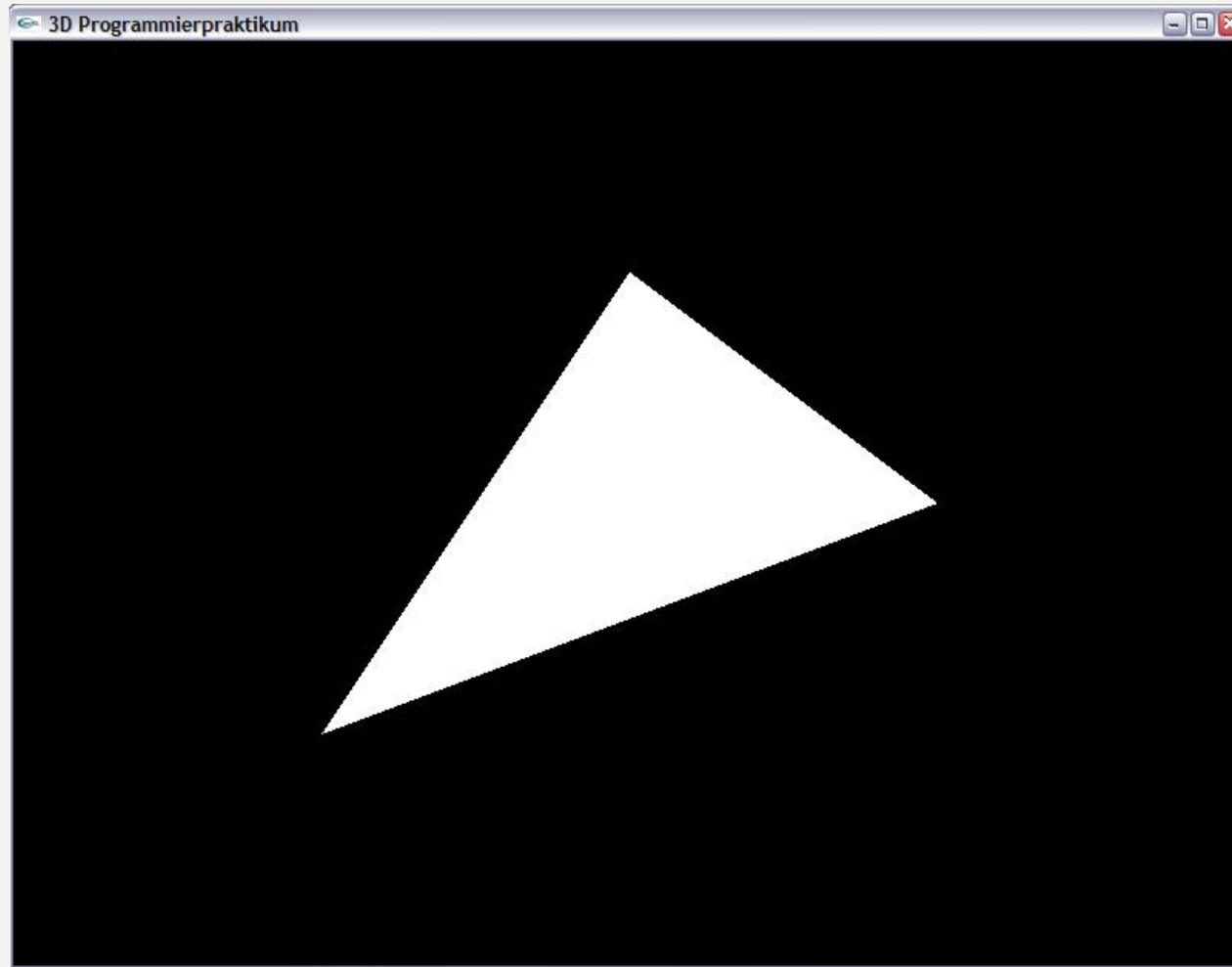
Belzner	Lenz
Bertram	Daniel
Filonik	Daniel
Frankenberger	Florian
Grupp	Nico
Hammer	Felix
Lang	Alexander
Lindemann	Florian
Meindl	Stefan
Plötner	Mark
Schmidt	Fabian
Schulz	Florian
Sharanova	Darina
Wenz	Stefan

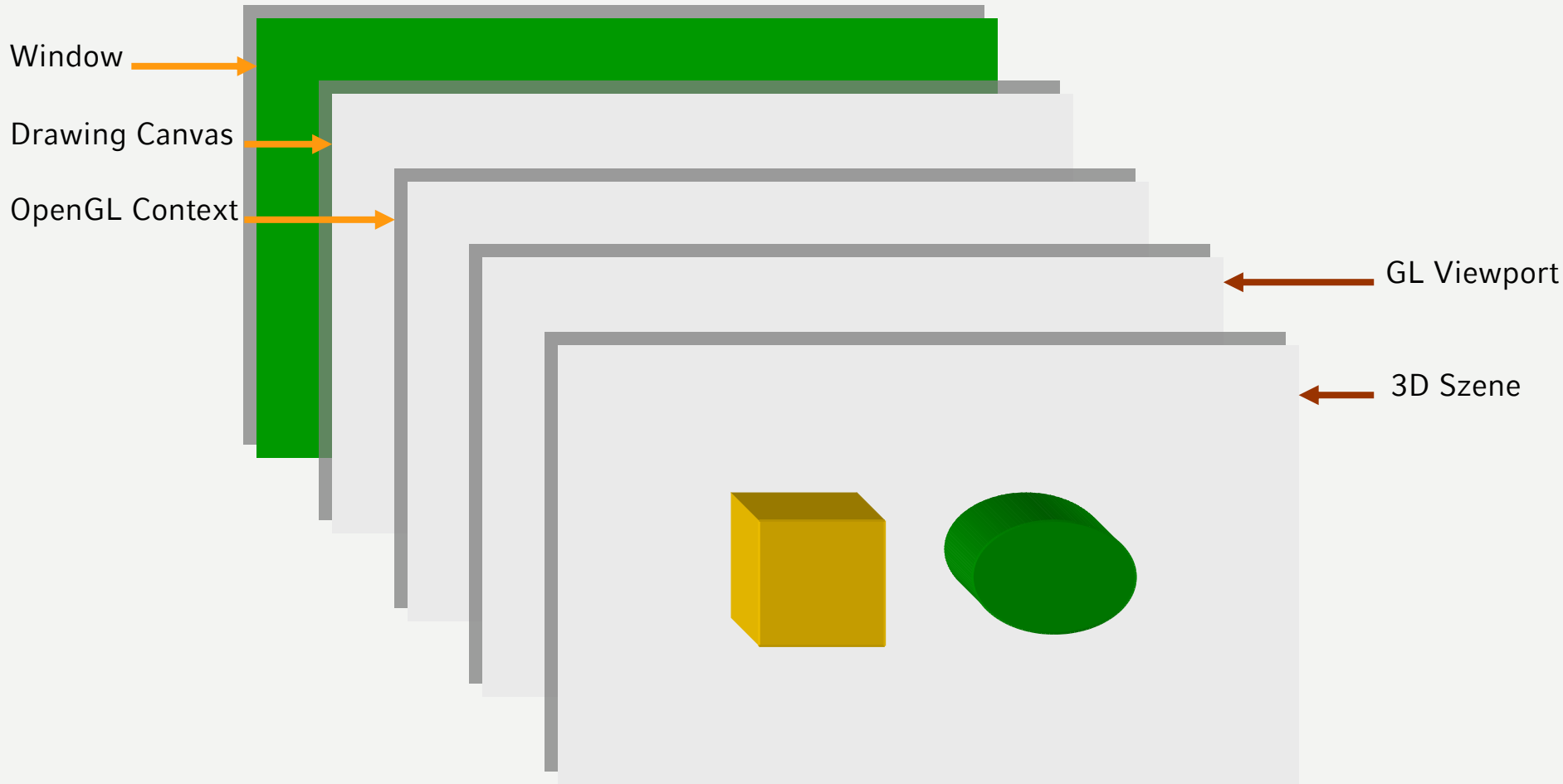
- Computergrafik: Echte Welt -> gerasterte Ausgabe





- Komplizierter Weg von der realen Szene zum gerasterten Bild
- Viele Arbeits(teil)stücke sind mathematisch komplex und Aufwendig (Clipping, Projektion, Culling, Rasterization)
- Grafikbibliotheken nehmen dem Programmierer viel Arbeit ab.
- Für uns sind zwei Arten von Toolkits Interessant:
  - Windowing Toolkits (Fenster, Dekorationen, Menus, Eingaben)
  - 3D Toolkits (Modellierung, Animation, Projektion, Rastern)







- OpenGL ist eine API zur 3D Programmierung
- Bietet Funktionen zum zeichnen von 3D-Szenen und Schnittstelle zur Grafik Hardware.
- Keinerlei Funktionalität für Zugriff auf/von OS (Fenster, Benutzereingaben usw.)
- → Erweiterungen des OpenGL Standards fügen diese Funktionalitäten hinzu.



- GLUT (*OpenGL Utility Toolkit*) Plattform unabhängiges Windowing Toolkit.
- Verwaltung von Fenstern für OpenGL-Anwendungen
- I/O-Ereignisverarbeitung durch Callbacks
- Bestandteil des OpenGL Standards
- Verwendung von GLUT:

```
#include <GL/glut.h>
```

```
// bindet auch GL/gl.h und GL/glu.h ein
```





- GLUT initialisieren:

```
void glutInit(int* argc, char** argv);
```

- GLUT-Fenster initialisieren:

```
void glutInitWindowSize(int w, int h);
```

```
void glutInitWindowPosition(int x, int y);
```

- Displaymodus (später mehr):

```
void glutInitDisplayMode(unsigned int mode);
```

- Fenster erzeugen:

```
int glutCreateWindow(char* title);
```



## ■ Fenster zerstören:

```
void glutDestroyWindow(int window);
```

## ■ Fenster neu zeichnen:

```
void glutPostRedisplay();
```

## ■ Bildschirmspeicher wechseln:

```
void glutSwapBuffers();
```

- GLUT zeichnet im Displaymodus `DOUBLE` jeweils in den Hintergrund-Buffer
- Am Ende des Zeichnens wird der aktuell dargestellte Buffer und der Hintergrund-Buffer gewechselt



- GLUT verwendet einige Callback-Funktionen, die durch Ereignisse aufgerufen werden
- Idle Callback:
  - Wird aufgerufen, wenn keine anderen Events aufgetreten sind → Zeichnet meistens die Szene neu

```
void glutIdleFunc(void (*func)(void));
```



## ■ Display Callback:

- Wird aufgerufen, wenn das **aktuelle** Fenster neu gezeichnet werden muss

```
void glutDisplayFunc(void (*func)(void));
```

## ■ Reshape Callback:

- Wird aufgerufen, wenn sich die Größe des **aktuellen** Fensters verändert

```
void glutReshapeFunc(void (*func)(int w, int h));
```



## ■ Keyboard Callback:

- Wird aufgerufen, wenn eine Standard-Taste (keine F-Tasten o.ä.) gedrückt wurde

```
void glutKeyboardFunc(void (*func)  
                      (unsigned char key, int x, int y));
```

## ■ Special Callback:

- Wird aufgerufen, wenn eine Sondertaste gedrückt wurde

```
void glutSpecialFunc(void (*func)  
                    (int key, int x, int y));
```



## ■ Mouse Callback:

- Wird aufgerufen, wenn eine Maus-Taste gedrückt wurde

```
void glutMouseFunc(void (*func)  
    (int button, int state, int x, int y));
```

## ■ Motion Callback:

- Wird aufgerufen, wenn die Maus bei gedrückter Maustaste bewegt wurde

```
void glutSpecialFunc(void (*func)  
    (int x, int y));
```



## ■ Passive Motion Callback:

- Wird aufgerufen, wenn die Maus ohne gedrückte Maustaste bewegt wurde

```
void glutPassiveMotionFunc(void (*func)  
    (int x, int y));
```

## ■ Starten der Ereignisverarbeitung:

- Wird nach der Initialisierung von GLUT und OpenGL aufgerufen und startet eine Endlosschleife

```
void glutMainLoop(void);
```



## ■ Beispiel:

```
void display(void) {  
    // some OpenGL drawing  
}  
  
int main(int argc, char** argv) {  
    glutInit(&argc, argv);  
    ...  
    glutDisplayFunc(display);  
    ...  
    glutMainLoop();  
    return 0;  
}
```





- GLUT unterstützt verschiedene OpenGL-Modi

- Mögliche Modi:

```
GLUT_RGBA      // RGBA mode window
GLUT_SINGLE    // Single buffering
GLUT_DOUBLE    // Double buffering
GLUT_ALPHA     // Window has alpha components
GLUT_DEPTH     // Window with depth buffer
```

- Beispiel:

```
glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);
```



## ■ Zwei Möglichkeiten, den Vollbildmodus zu aktivieren (Beispiele):

- Direkt bei der Erzeugung des Fensters:

```
glutGameModeString("640x480:16@60");  
glutEnterGameMode();
```

- Während der Ausführung:

```
// normal to fullscreen  
glutFullscreen();  
  
// fullscreen to normal  
glutReshapeWindow(640, 480);  
glutPositionWindow(100, 100);
```



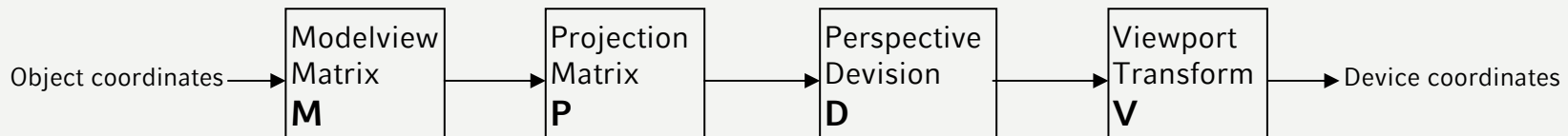
# OpenGL



- OpenGL ist eine State-Machine die das Zeichnen von 3D-Szenen und Animationen erlaubt
- Grafik Primitive werden zusammen gesetzt und manipuliert (Status Änderungen)
- Primitive:
  - Punkte
  - Linien
  - Polygone
  - Bitmaps
- Außerdem werden Lichtquellen und Kamera positioniert
- Anschließend wird die Szene ausgegeben
- Das ganze findet in einer Endlosschleife statt



1. Objekte im Raum anordnen und ausrichten (*Modelling Transformations*)
2. Kamera positionieren (*Viewing Transformations*)
  1. Defaultposition (0,0,1)
3. Ausschnitt aus der „Welt“ wählen (*Viewing Volume*)
4. Inhalt des Viewing Volumes in 2D-Ebene Projizieren (*Projection Transformations*)
5. Ausgabe des Resultats auf dem Bildschirm (*Viewport Transformations*)





- Alle 2D-Objekte liegen im Raum in einer Ebene und bestehen aus einer Menge von Punkten
- Mögliche 2D-Objekte:
  - Points (1D)
  - Lines (2D)
  - Line Strips (2D) → Verkettete Linien
  - Line Loops (2D) → geschlossener Linienzug
  - Polygons (2D) → geschlossener, gefüllter Linienzug



## ■ Beispiel:

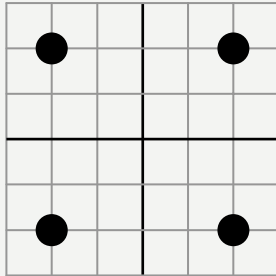
```
glBegin(GL_POLYGON);  
    glColor3f(1.0f, 0.0f, 0.0f); // red  
    glVertex3f(-1.0f, -1.0f, 0.0f);  
    glVertex3f(1.0f, -1.0f, 0.0f);  
    glColor3f(0.0f, 0.0f, 1.0f); // blue  
    glVertex3f(1.0f, 1.0f, 0.0f);  
    glVertex3f(-1.0f, 1.0f, 0.0f);  
glEnd();
```

## ■ Weitere Formen:

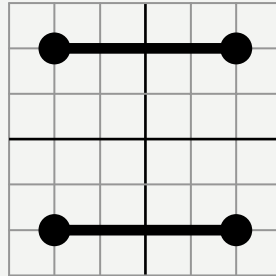
```
GL_POINTS, GL_LINES (je 2 Punkte verbunden)  
GL_LINE_STRIP, GL_LINE_LOOP  
GL_POLYGON, GL_QUADS, GL_TRIANGLES
```



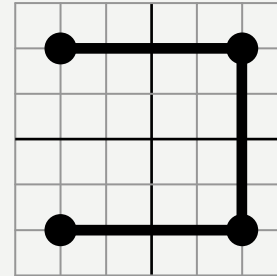
## Beispiele:



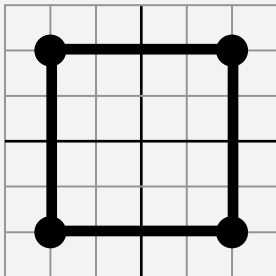
GL\_POINTS



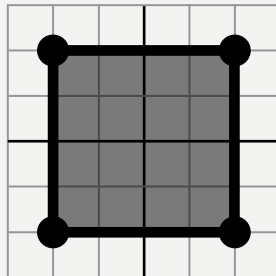
GL\_LINES



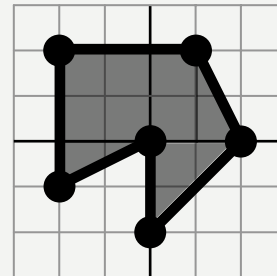
GL\_LINE\_STRIP



GL\_LINE\_LOOP



GL\_QUADS



GL\_POLYGON





- Werden aus 2D-Objekten erstellt
- Vertices einer Fläche unterscheiden sich in allen drei Koordinaten
- Beispiel:
  - Würfel aus 6 Quadraten
  - Würfel aus 12 Dreiecken
  - Kugel aus  $n$  Dreiecken
  - Tetraeder aus 4 Dreiecken
  - ...



## Die Einheitsmatrix:

```
glLoadIdentity();
```

- Lädt die Einheitsmatrix (um Speicher zu initialisieren)

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

## Die Translation:

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

```
glTranslated(x, y, z);
```

- Verschiebt den Ursprung an die neue Position

$$\begin{pmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



## Die Rotation:

```
glRotatef(a, x, y, z);
```

```
glRotated(a, x, y, z);
```

- Rotiert den Ursprung um den Vektor  $(x, y, z)$  entgegen dem Uhrzeigersinn

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha & 0 \\ 0 & \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
glRotate*( $\alpha$ , 1, 0, 0);
```

$$\begin{pmatrix} \cos \alpha & -\sin \alpha & 0 & 0 \\ \sin \alpha & \cos \alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
glRotate*( $\alpha$ , 0, 0, 1);
```

$$\begin{pmatrix} \cos \alpha & 0 & \sin \alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \alpha & 0 & \cos \alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
glRotate*( $\alpha$ , 0, 1, 0);
```



- Zunächst Löschen der Puffer:

```
glClear(GL_COLOR_BUFFER_BIT |  
        GL_DEPTH_BUFFER_BIT);
```

- Vor dem Zeichnen: Transformation des zu zeichnenden Objektes:

```
glLoadIdentity(); // Laden der Einheitsmatrix  
glTranslatef(GLfloat x, GLfloat y, GLfloat z);  
glRotatef(GLfloat angle, GLfloat x, GLfloat y,  
          GLfloat z);
```

- Nach dem Zeichnen der Objekte: Wechsel des Puffers

```
glutSwapBuffers();
```

OpenGL Reference Page:

<http://www.mevis.de/opengl/opengl.html>

GLUT Man Pages:

<http://www.cs.uccs.edu/~semwal/man.html>