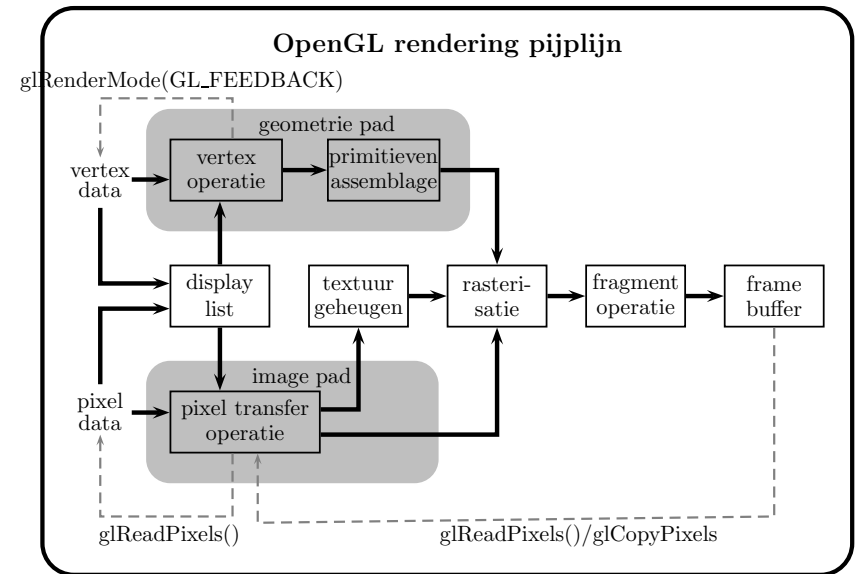


Computergrafieken

$$\begin{cases} \text{theorie :} & 8 \times 1.5u \\ \text{practica :} & 12 \times 1.5u \end{cases}$$

Evaluatie:

- computergrafieken, theorie: rekenopgave en project met examen in 2EP
- computergrafieken, practica: permanente evaluatie



OpenGL: introductie

- **GL**: low-level grafische bibliotheek van functies voor het specificeren van een 2D- of 3D-geometrie met behulp van een aantal **primitieven** en deze geometrie op het scherm te tonen (**rendering**)
- **GLU**: (utility library) combineren van low-level functies voor het opzetten van **transformaties** voor specifieke projecties; genereren en renderen van **quadrics**;
- **GLUT**: (utility toolkit) **venster-systeem onafhankelijke** bibliotheek onzichtbaar maken van allerlei complexe problemen van verschillende windows-API's voor de programmeur.

Programma's die GLUT gebruiken: include van **headerbestand** GL/glut.h (met daarin een include van GL/gl.h en GL/glu.h)

- correcte prototyping van alle OpenGL-functies
- definitie van alle OpenGL-constanten

Initialisatie


- `glutInit(&argc,argv)`: **initialisatie** van de GLUT-library; moet opgeroepen worden voor elke andere glut functie
- `glutInitDisplayMode(...)`: argument: **bit-OR** van GLUT-parameters:
 - definitie van **kleuren**: op basis van rood-groen-blauw waarden (GLUT_RGB) of met bijkomende α -waarde om de doorzichtigheid aan te geven (GLUT_RGBA): van totale **transparantie** ($\alpha = 0.0$) tot volledige ondoorzichtbaarheid ($\alpha = 1.0$);
 - **frame-buffering** (GLUT_SINGLE); bij animaties is het aan te raden om met dubbele buffering te werken (GLUT_DOUBLE);
 - bij 3D: GLUT_DEPTH: bij elke pixel in het frame wordt een **diepte** bijgehouden: objecten die zich veraf bevinden, worden bedekt door objecten die zich dichterbij bevinden
- `glutInitWindowSize(breedte, hoogte)`: grootte van het window
- `glutInitWindowPosition(x, y)` tov. linkerbovenhoek van het window

Window creatie en display

- glutCreateWindow(titel): maken van een **window** met bovenaan het argument als **titel**; maar dit window wordt niet getoond
- **callback-functies**: o.a display- en reshape-functie
- glutPostRedisplay(): **forceren** van een hertekening;
- glutMainLoop(): laatste functieoproep in main: daadwerkelijk **tonen** van alle windows die tijdens initialisatie gedefinieerd zijn door het oproepen van de display-functie; daarna een **oneindige lus**, waarbij bij elke **event** één of meerdere callback-functies opgeroepen worden.

Voorbeeld: lijn, woord, rechthoek

```
#include <stdio.h>
#include <GL/glut.h>
char ratio = 'n'; /* aangepaste viewport bij arg == 'k' */
GLdouble xmin = 0.0, xmax = 200.0, ymin = 0.0, ymax = 150.0;
GLdouble verhouding = xmax/ymax;
void init(void)
{
    glClearColor(0.9, 1.0, 0.9, 0.0);
}
void lijnSegment(void)
{
    glClear(GL_COLOR_BUFFER_BIT);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
```




Callback functies

een functie die uitgevoerd wordt, telkens een specifiek **event** zich voordoet activeren door een GLUT-functie met de naam van de functie als argument

- glutDisplayFunc(teken): de functie **teken** wordt uitgevoerd telkens het systeem het nodig vindt om het venster te hertekenen, bijv. bij het tijdelijk bedekken van het venster of geforceerd door het oproepen van glutPostRedisplay()
functie **teken**: alle routines om de tekening volledig op te bouwen
- glutReshapeFunc(herschaal): telkens een window verandert van **grootte** of van **positie** op het scherm
herschaal heeft twee argumenten: de nieuwe breedte en hoogte

```
glColor3f(1.0, 0.0, 0.0);
glBegin(GL_LINES);
    glVertex2i(180,15);
    glVertex2i(10,145);
glEnd();
glColor3f(0.0, 1.0, 0.0);
glRecti(115,90,165,140);
glColor3f(0.0, 0.0, 1.0);
glRasterPos2i(150,20);
glutBitmapCharacter(GLUT_BITMAP_9_BY_15, 'l');
glutBitmapCharacter(GLUT_BITMAP_8_BY_13, 'ijn');
glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_10, 'j');
glutBitmapCharacter(GLUT_BITMAP_HELVETICA_10, 'n');
glFlush();
}
```



```

void herschaal(GLint n_w, GLint n_h)
{
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(xmin, xmax, ymin, ymax);
    if ( ratio == 'k' )
    {
        if ( n_w <= n_h * verhouding )
            glViewport(0, 0, n_w, n_w/verhouding);
        else
            glViewport(0, 0, verhouding*n_h, n_h);
    }
    else
        glViewport(0, 0, n_w, n_h);
    fprintf(stderr, "nieuw_schaal_%d_%d\n", n_w, n_h);
}

```

```

int main( int argc, char * argv[])
{
    if ( argc > 1 )
        ratio = argv[1][0];
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowPosition(50, 100);
    glutInitWindowSize(400, 300);
    glutCreateWindow("een_lijnsegment");
    init();
    glutDisplayFunc(lijnSegment);
    glutReshapeFunc(herschaal);
    glutMainLoop();
    return 0;
}

```

Skeleton

- een aantal globale variabelen
- een init functie met eenmalige initialisaties;
- een *teken* callback functie waarin het volledige tafereel opgebouwd wordt, eventueel door het oproepen van andere functies;
- een *herschaa* callback functie waarin de wereld- naar schermcoördinatensysteem transformatie gedefinieerd wordt;
- de main functie die, op enkele details na, telkens integraal kan overgenomen worden.

```

GLdouble xmin = -5.0, xmax = 5.0, ymin = -0.3, ymax = 1.2;

1 void gebogen(void)
2 {
3     GLfloat x;
4
5     glClear(GL_COLOR_BUFFER_BIT);
6     glMatrixMode(GL_MODELVIEW);
7     glLoadIdentity();
8     if ( assen )
9     {
10         glColor3f(0.0, 1.0, 0.0);
11         glBegin(GL_LINES);
12             glVertex2f( xmin+0.6, 0.0 );
13             glVertex2f( xmax-0.6, 0.0 );
14             glVertex2f( 0.0, ymin+0.1);
15             glVertex2f( 0.0, ymax-0.1);
16         glEnd();

```

```
17     }  
18     glColor3f(0.0, 0.0, 1.0);  
19     glLineWidth(2.0);  
20     glPointSize(3.0);  
21     glBegin(GL_LINE_STRIP);  
22         for ( x = -4.0; x < 4.0; x += 0.1 )  
23         {  
24             glVertex2f( x, sin(M_PI*x)/(M_PI*x) );  
25         }  
26     glEnd();  
27     glFlush();  
}
```
