Ασκήσεις 3ου μαθήματος: 3D Shapes

Κώδικας προγράμματος #4.1

Ακολουθεί ο κώδικας του προγράμματος 3D Shapes

```
#include <windows.h>  // for MS Windows
#include <GL/glut.h>  // GLUT, include glu.h and gl.h

/* Global variables */
char title[] = "3D Shapes";

/* Initialize OpenGL Graphics */
void initGL() {
   glClearColor(0.0f, 0.0f, 0.0f, 1.0f); // Set background color to black and opaque
   glClearDepth(1.0f);                   // Set background depth to farthest
   glEnable(GL_DEPTH_TEST);   // Enable depth testing for z-culling
   glDepthFunc(GL_LEQUAL);    // Set the type of depth-test
   glShadeModel(GL_SMOOTH);   // Enable smooth shading
   glHint(GL_PERSPECTIVE_CORRECTION_HINT, GL_NICEST);  // Nice perspective corrections
}

/* Handler for window-repaint event. Called back when the window first appears and


  whenever the window needs to be re-painted. */
void display() {
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); // Clear color and depth buffers
   glMatrixMode(GL_MODELVIEW);     // To operate on model-view matrix

   // Render a color-cube consisting of 6 quads with different colors
   glLoadIdentity();                 // Reset the model-view matrix
   glTranslatef(1.5f, 0.0f, -7.0f);  // Move right and into the screen

   glBegin(GL_QUADS);                // Begin drawing the color cube with 6 quads
      // Top face (y = 1.0f)
      // Define vertices in counter-clockwise (CCW) order with normal pointing out
      glColor3f(0.0f, 1.0f, 0.0f);     // Green
      glVertex3f( 1.0f, 1.0f, -1.0f);
      glVertex3f(-1.0f, 1.0f, -1.0f);
      glVertex3f(-1.0f, 1.0f,  1.0f);
      glVertex3f( 1.0f, 1.0f,  1.0f);

      // Bottom face (y = -1.0f)
      glColor3f(1.0f, 0.5f, 0.0f);     // Orange
      glVertex3f( 1.0f, -1.0f,  1.0f);
      glVertex3f(-1.0f, -1.0f,  1.0f);
      glVertex3f(-1.0f, -1.0f, -1.0f);
      glVertex3f( 1.0f, -1.0f, -1.0f);

      // Front face  (z = 1.0f)
      glColor3f(1.0f, 0.0f, 0.0f);     // Red
      glVertex3f( 1.0f,  1.0f, 1.0f);
      glVertex3f(-1.0f,  1.0f, 1.0f);
      glVertex3f(-1.0f, -1.0f, 1.0f);
      glVertex3f( 1.0f, -1.0f, 1.0f);

      // Back face (z = -1.0f)
      glColor3f(1.0f, 1.0f, 0.0f);     // Yellow
      glVertex3f( 1.0f, -1.0f, -1.0f);
      glVertex3f(-1.0f, -1.0f, -1.0f);
      glVertex3f(-1.0f,  1.0f, -1.0f);
```

```
      glVertex3f( 1.0f,  1.0f, -1.0f);

      // Left face (x = -1.0f)
      glColor3f(0.0f, 0.0f, 1.0f);     // Blue
      glVertex3f(-1.0f,  1.0f,  1.0f);
      glVertex3f(-1.0f,  1.0f, -1.0f);
      glVertex3f(-1.0f, -1.0f, -1.0f);
      glVertex3f(-1.0f, -1.0f,  1.0f);

      // Right face (x = 1.0f)
      glColor3f(1.0f, 0.0f, 1.0f);     // Magenta
      glVertex3f(1.0f,  1.0f, -1.0f);
      glVertex3f(1.0f,  1.0f,  1.0f);
      glVertex3f(1.0f, -1.0f,  1.0f);
      glVertex3f(1.0f, -1.0f, -1.0f);
    glEnd();  // End of drawing color-cube

    // Render a pyramid consists of 4 triangles
    glLoadIdentity();                 // Reset the model-view matrix
    glTranslatef(-1.5f, 0.0f, -6.0f);  // Move left and into the screen

    glBegin(GL_TRIANGLES);           // Begin drawing the pyramid with 4 triangles
      // Front
      glColor3f(1.0f, 0.0f, 0.0f);     // Red
      glVertex3f( 0.0f, 1.0f, 0.0f);
      glColor3f(0.0f, 1.0f, 0.0f);     // Green
      glVertex3f(-1.0f, -1.0f, 1.0f);
      glColor3f(0.0f, 0.0f, 1.0f);     // Blue
      glVertex3f(1.0f, -1.0f, 1.0f);

      // Right
      glColor3f(1.0f, 0.0f, 0.0f);     // Red
      glVertex3f(0.0f, 1.0f, 0.0f);
      glColor3f(0.0f, 0.0f, 1.0f);     // Blue
      glVertex3f(1.0f, -1.0f, 1.0f);
      glColor3f(0.0f, 1.0f, 0.0f);     // Green
      glVertex3f(1.0f, -1.0f, -1.0f);

      // Back
      glColor3f(1.0f, 0.0f, 0.0f);     // Red
      glVertex3f(0.0f, 1.0f, 0.0f);
      glColor3f(0.0f, 1.0f, 0.0f);     // Green
      glVertex3f(1.0f, -1.0f, -1.0f);
      glColor3f(0.0f, 0.0f, 1.0f);     // Blue
      glVertex3f(-1.0f, -1.0f, -1.0f);

      // Left
      glColor3f(1.0f,0.0f,0.0f);       // Red
      glVertex3f( 0.0f, 1.0f, 0.0f);
      glColor3f(0.0f,0.0f,1.0f);       // Blue
      glVertex3f(-1.0f,-1.0f,-1.0f);
      glColor3f(0.0f,1.0f,0.0f);       // Green
      glVertex3f(-1.0f,-1.0f, 1.0f);
    glEnd();   // Done drawing the pyramid

    glutSwapBuffers();  // Swap the front and back frame buffers (double buffering)
}

/* Handler for window re-size event. Called back when the window first appears and whenever the
window is re-sized with its new width and height */
void reshape(GLsizei width, GLsizei height) {  // GLsizei for non-negative integer
    // Compute aspect ratio of the new window
    if (height == 0) height = 1;                // To prevent divide by 0
    GLfloat aspect = (GLfloat)width / (GLfloat)height;

    // Set the viewport to cover the new window
    glViewport(0, 0, width, height);

    // Set the aspect ratio of the clipping volume to match the viewport
    glMatrixMode(GL_PROJECTION);  // To operate on the Projection matrix
    glLoadIdentity();             // Reset
```

```
  // Enable perspective projection with fovy, aspect, zNear and zFar
  gluPerspective(45.0f, aspect, 0.1f, 100.0f);
}

/* Main function: GLUT runs as a console application starting at main() */
int main(int argc, char** argv) {
  glutInit(&argc, argv);           // Initialize GLUT
  glutInitDisplayMode(GLUT_DOUBLE); // Enable double buffered mode
  glutInitWindowSize(640, 480);   // Set the window's initial width & height
  glutInitWindowPosition(50, 50); // Position the window's initial top-left corner
  glutCreateWindow(title);          // Create window with the given title
  glutDisplayFunc(display);       // Register callback handler for window re-paint event
  glutReshapeFunc(reshape);       // Register callback handler for window re-size event
  initGL();                       // Our own OpenGL initialization
  glutMainLoop();                 // Enter the infinite event-processing loop
  return 0;
}
```

Κατανόηση προγράμματος #4.2

Στην εντολή glutDisplayFunc(display)

1. Ποιο είναι το συμβάν (event) που πυροδοτεί την glutDisplayFunc();
2. Σε ποιες περιπτώσεις εμφανίζεται το παραπάνω συμβάν;
3. Ποια είναι η συμπεριφορά της μηχανής (υποσύστημα γραφικών) κάθε φορά που εκδηλώνεται το συμβάν;

Κατανόηση προγράμματος #4.3

Τι κάνουν οι εντολές glClearColor(0.0f, 0.0f, 0.0f, 1.0f), glClearDepth(1.0f) και glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); Αναφερθείτε αναλυτικά τόσο στις εντολές όσο και στις παραμέτρους.

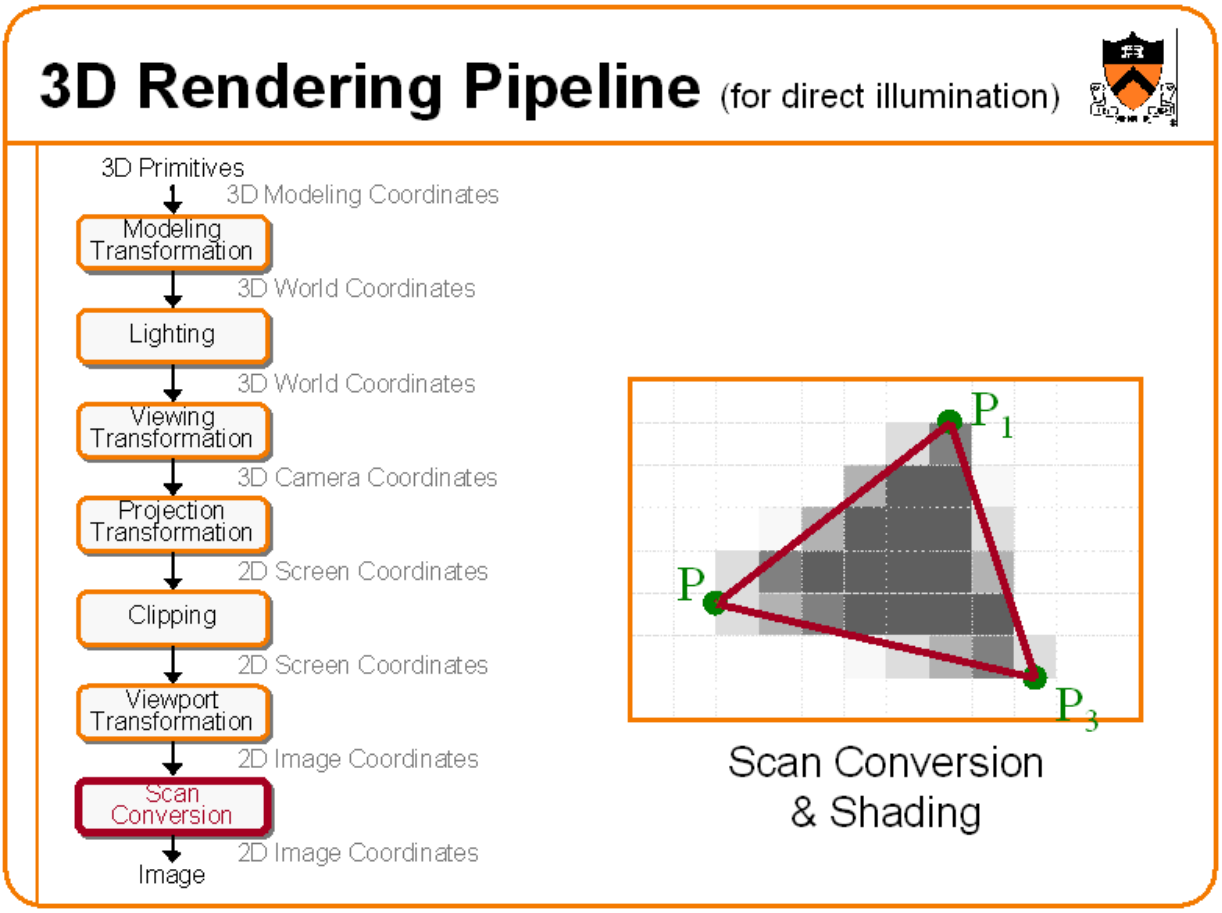
Κατανόηση προγράμματος #4.4

Τι κάνουν οι εντολές glEnable(GL_DEPTH_TEST) και glDepthFunc(GL_LEQUAL); Γιατί τις χρειάζεται η εντολή glClear(GL_DEPTH_BUFFER_BIT) - περιγράψτε με λεπτομέρεια.

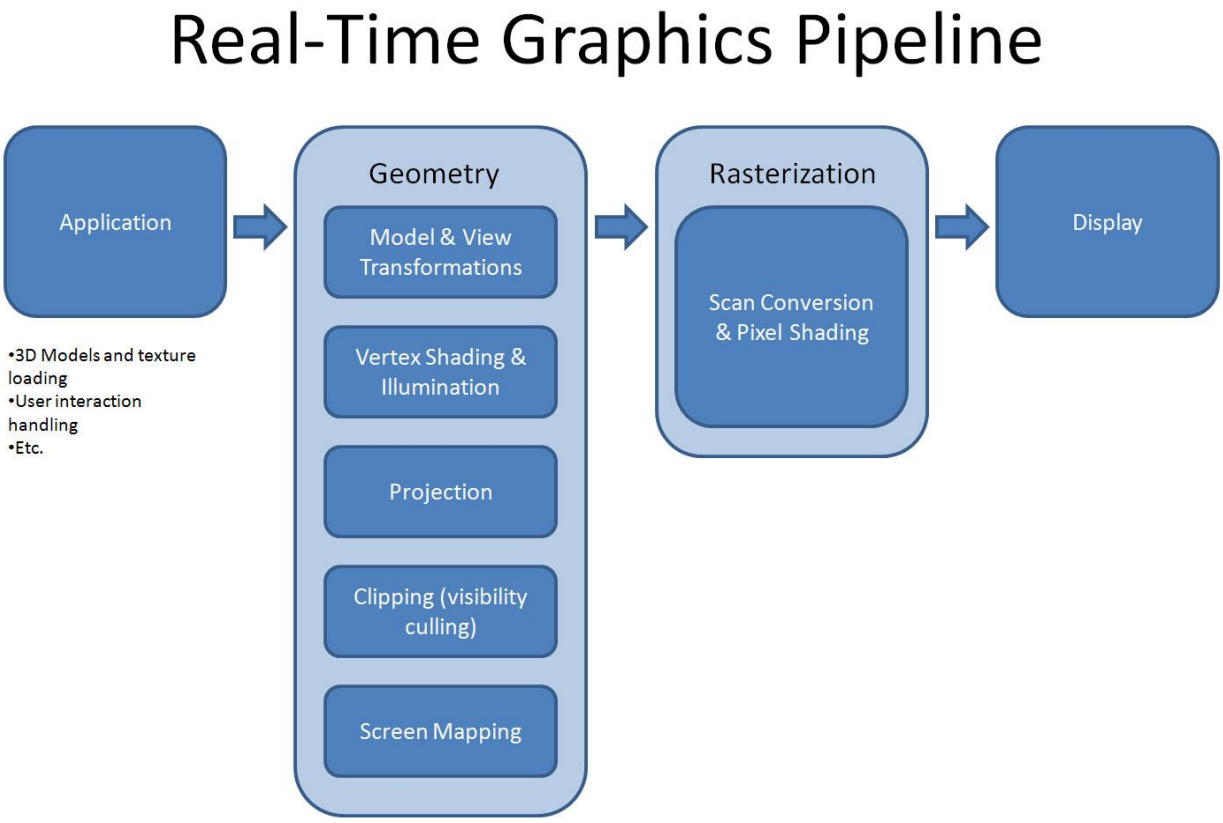
Κατανόηση προγράμματος #4.5

Περιγράψτε όσο το δυνατό πληρέστερα πώς αναπαρίσταται ένα 3D γραφικό στην οθόνη. Χρησιμοποιείστε

- το σχήμα (αγνοήστε τα επίπεδα Lighting και Scan Conversion).



- Το εναλλακτικό σχήμα



Ποιες εντολές του προγράμματος υλοποιούν τους επόμενους μετασχηματισμούς (οι οποίοι φυσικά αντιστοιχίζονται στα επίπεδα των παραπάνω σχημάτων).

- MODEL το WORLD μετασχηματισμό
- WORLD to VIEW μετασχηματισμό
- VIEW to PROJECTION μετασχηματισμό

Κατανόηση προγράμματος #4.6

Ποια είδη Projection γνωρίζετε; Αναζητήστε στο WEB σχήματα που βοηθούν στην κατανόησή τους.
Σε ποια περίπτωση και με ποιες εντολές χρησιμοποιούμε το κάθε είδος Projection;

Κατανόηση προγράμματος #4.7

Κάντε τις απαραίτητες προγραμματιστικές επεμβάσεις στο πρόγραμμα ώστε

1. το χρώμα φόντου να γίνει λευκό
2. οι διαστάσεις του παραθύρου να μεγαλώσουν (επιλέξτε εσείς το μέγεθος)
3. ο τίτλος του παραθύρου να γίνει "Γ' ΕΞΑΜΗΝΟ"
4. η πυραμίδα να μετακινηθεί ελαφρώς πιο αριστερά και ο κύβος ελαφρώς πιο δεξιά
5. οι έδρες της πυραμίδας να είναι μονόχρωμες και του κύβου gradient (μεταξύ χρωμάτων της επιλογής σας)
6. η πυραμίδα να περιστρέφεται (συνδυάστε με το προηγούμενο μάθημα - animation)