**Group 16**

**Mark Gachoki SCT212-0224/2017**

**Stanley Ngugi SCT212-0065/2017**

**Yvonne Kimani SCT212-0475/2017**

QUESTION 16

**PART 1**

*Question: Describe using a working examples, the relationship between the following coordinate systems in the graphics (OCS - object coordinate system WCS - world coordinate system VCS - viewing coordinate system CCS - clipping coordinate system NDCS - normalized device coordinate system DCS - device coordinate system)*

***b) Why do we need homogeneous coordinates ?***

**OCS.**

First, we can define the shapes of individual objects, such as trees or furniture, within a separate reference frame for each object.

These reference frames are called **modeling coordinates,** or sometimes **local coordinates** or **master coordinates.**

**WCS**

* Once the individual object shapes have been specified, we can construct (“model”) a scene by placing the objects into appropriate locations within a scene reference frame called **world coordinates.**
* This step involves the transformation of the individual modeling-coordinate frames to specified positions and orientations within the world-coordinate frame.
* As an example, we could construct a bicycle by defining each of its parts (wheels, frame, seat, handlebars, gears, chain, pedals) in a separate modeling coordinate

frame.

* Then, the component parts are fitted together in world coordinates.
* If both bicycle wheels are the same size, we need to describe only one wheel in a local-coordinate frame. Then the wheel description is fitted into the world-coordinate bicycle description in two places.
* Geometric descriptions in modeling coordinates and world coordinates can be given in any convenient floating-point or integer values, without regard for the constraints of a particular output device

.

* After all parts of a scene have been specified, the overall world-coordinate description is processed through various routines onto one or more output-device reference frames for display. This process is called the **viewing pipeline.**

**VCS**

World coordinate positions are first converted to *viewing coordinates* corresponding to the view we want of a scene, based on the position and orientation of a hypothetical camera.

**CCS**

Then object locations are transformed to a two-dimensional (2D) projection

of the scene, which corresponds to what we will see on the output device.

**NDCS**

The scene is then stored in **normalized coordinates,** where each coordinate value

is in the range from −1 to 1 or in the range from 0 to 1, depending on the system.

Normalized coordinates are also referred to as *normalized device coordinates,* since using this representation makes a graphics package independent of the coordinate range for any specific output device.

We also need to identify visible surfaces and eliminate picture parts outside the bounds for the view we want to show on the display device.

**DCS**

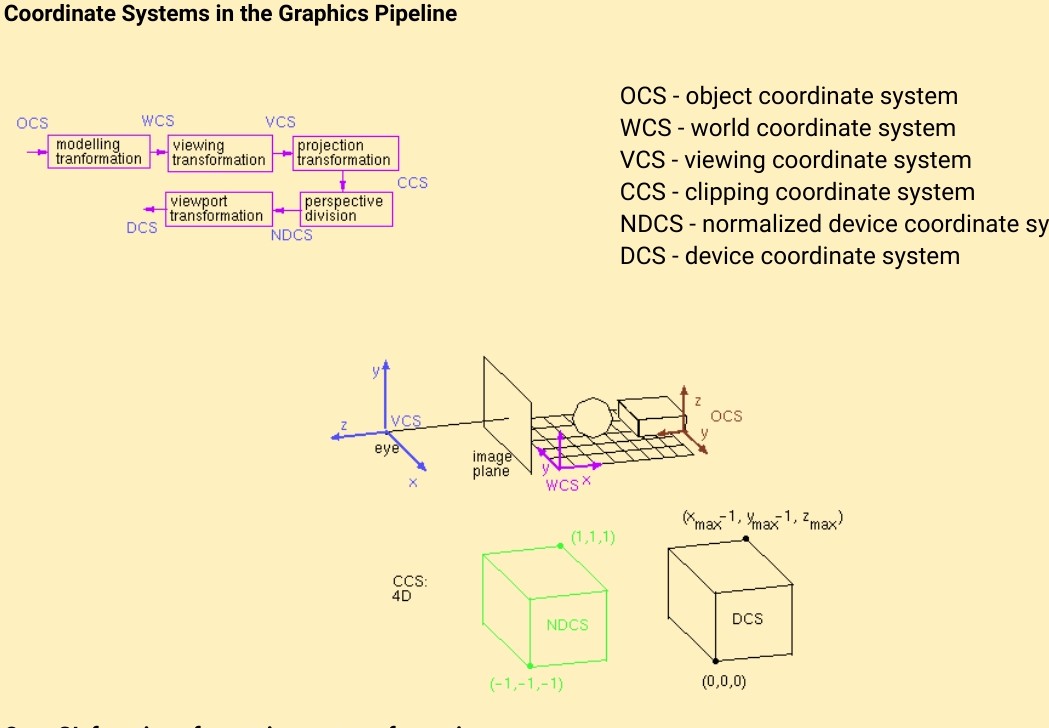
Finally, the picture is scan-converted into the refresh buffer of a raster system for display. The coordinate systems for display devices are generally called **device coordinates,** or **screen coordinates** in the case of a video monitor.

Often, both normalized coordinates and screen coordinates are specified in a left-handed coordinate reference frame so that increasing positive distances from the *xy* plane (the screen, or viewing plane) can be interpreted as being farther from the viewing position.

**PART 1b)**.

*Question: Why do we need homogeneous coordinates ?*

Homogenous coordinates are used because they allow common vector operations such as translation, rotation, scaling and perspective projection to be represented as a matrix by which the vector is multiplied.



**PART 2**

*Question: Our good Classrep Yvonne , is a fan of Chess. He has lost his chessboard. Write a program in OpenGL that implements a 8 by 8 chessboard. Kimani prefers brown and white color instead of black and white . Help Yvonne out!*

**Source Code:**

#include <stdio.h>

#include <GL/glut.h>

int x = 50, y = 50;

bool isBrown = true;

void whiteBox(int x, int y)

{

glBegin(GL\_LINE\_LOOP);

glVertex2i(x, y);

glVertex2i(x, y + 50);

glVertex2i(x + 50, y + 50);

glVertex2i(x + 50, y);

glEnd();

}

void brownBox(int x, int y)

{

glBegin(GL\_POLYGON);

glColor3f(0.5f, 0.35f, 0.05f);

glVertex2i(x, y);

glVertex2i(x, y + 50);

glVertex2i(x + 50, y + 50);

glVertex2i(x + 50, y);

glEnd();

}

void myDisplay(void)

{

// set display

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(0.0, 0.0, 0.0);

glPointSize(1.0);

//Loop horizontal to upper limit 8

for (int i = 0; i < 8; i++)

{

if (i % 2 == 0)

{

isBrown = true;

}

else

{

isBrown = false;

}

//Loop vertical to upper limit 8

for (int j = 0; j < 8; j++)

{

if (isBrown)

{

//draw brown

brownBox(x, y);

isBrown = false;

}

else

{

//draw white

whiteBox(x, y);

isBrown = true;

}

x += 50;

}

y += 50;

x = 50;

}

brownBox(100, 100);

whiteBox(150, 100);

glFlush();

}

void myInit(void)

{

glClearColor(1.0, 1.0, 1.0, 0.0);

glColor3f(0.0f, 0.0f, 0.0f);

glPointSize(4.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0, 640.0, 0.0, 480.0);

}

int main(int argc, char\*\* argv)

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(800, 600);

glutInitWindowPosition(100, 100);

glutCreateWindow("8 x 8 brown Chess Board");

glutDisplayFunc(myDisplay);

myInit();

glutMainLoop();

return 0;

}

**Output**

