Computer Graphics (UCS505)

Project on

Visualization of OS Algorithms

Submitted By

Jasween Kaur Brar	102017187
Saurabh Thakur	102017183
Aurav S. Tomar	102067008

Group No. 5

B.E. Third Year – COPC

Submitted To:

Ms. Diksha Arora

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

Computer Science and Engineering Department

Thapar Institute of Engineering and Technology

Patiala – 147001

Table of Contents

Introduction to Project

This project aims to visualize two popular page replacement algorithms used in operating systems: First In, First Out (FIFO) and Least Recently Used (LRU). The primary objective is to help users understand the working principles of these algorithms and their implications on system performance.

First In First Out (FIFO) Algorithm: FIFO is a straightforward page replacement algorithm that replaces the oldest page in memory with the newly requested page. When a page fault occurs, the algorithm selects the page that has been in memory for the longest time and replaces it with the new page. This approach is simple to implement but may not be the most efficient, as it needs to consider the usage patterns of the memory pages.

Least Recently Used (LRU) Algorithm: LRU is a more sophisticated page replacement algorithm that aims to replace the page that has been used the least recently. The rationale behind this approach is that recently accessed pages are more likely to be reaccessed shortly. When a page fault occurs, the algorithm replaces the least recently accessed page with the new page. This method is generally more efficient than FIFO but is more complex to implement.

The visualization implemented in this project consists of an interactive application that enables users to visualize the process of each algorithm. The user inputs a page array of size 9, and the application displays the progress of the selected page replacement algorithm (FIFO or LRU) as the user interacts with it. The application also provides a graphical representation of the memory frame buffer, input page array, and page fault occurrences. Users can control the animation, background color, and tile color to better understand the algorithms' workings.

By providing a visual representation of these algorithms, users can gain a deeper understanding of the fundamental concepts behind page replacement strategies in operating systems and observe the performance implications of each method.

Computer Graphics concepts used

1. OpenGL: OpenGL (Open Graphics Library) is a cross-platform API (Application Programming Interface) for rendering 2D and 3D graphics. The code uses OpenGL functions and GLUT (OpenGL Utility Toolkit) to create graphical elements and manage the user interface.

2. Coordinate system: The code employs a 2D Cartesian coordinate system to position and transform graphical elements. The origin (0, 0) is located at the bottom-left corner of the window, and the coordinates are defined in pixels.

3. Geometric primitives: The code uses basic geometric primitives like polygons and line loops to create the visual elements, such as tiles and flags. These primitives are defined using their vertices, which are specified in the 2D coordinate system.

4. Colors: Colors are represented using the RGB (Red, Green, Blue) model. The code uses arrays of RGB color values to set the background and tile colors. glColor3f and glColor3fv functions are used to specify the color of the graphical elements being drawn.

5. Text rendering: The code uses the GLUT_BITMAP_* font family and glutBitmapCharacter function to render text on the screen. This allows for displaying information about the page replacement algorithms, such as the number of page faults.

6. Transformations: The code employs transformations like translation to position and animate the graphical elements on the screen. glPushMatrix and glPopMatrix functions are used to save and restore the current transformation matrix, while glTranslatef is used to apply translation transformations.

7. Animation: The code uses the idle function and the GLUT_IDLE_FUNC callback to update the animation. When the user clicks on the window, the animation is triggered, and the page replacement algorithm progresses to the next step.

8. User interaction: The code uses the GLUT mouse callback functions (glutMouseFunc) to detect user input (left mouse button clicks) and trigger the execution of the FIFO or LRU page replacement algorithm accordingly.

9. Menus: The code uses the GLUT menu functions (glutCreateMenu, glutAddMenuEntry, and glutAttachMenu) to create a context menu that allows the user to change the background and tile colors or quit the application.

User Defined Functions

It uses the OpenGL Utility Toolkit (GLUT) to create a graphical user interface for the visualization.

Global Variables:

1. request[], counter[], pages[], colour[], pagecolour[]: Integer arrays to store page requests, counters, pages, and colors.
2. faults, choice: Integers to store the number of page faults and user's algorithm choice.
3. fonts[]: Array of font styles for displaying text.
4. pageArray[], pageOfBuffer[], fault[]: Integer arrays to store page data and page faults.
5. pagePosition[]: Float array to store the positions of page elements.
6. hit, step, destination, startAnimation: Integers for animation control and to count hits.
7. res[]: Character array to store the result string.
8. backgroundColor[][], tileColor[][]: Float arrays to store the background and tile colors.
9. bcPointer, tcPointer: Integer pointers for background and tile colors.

User-Defined Functions:

1. void init(): Initializes the display settings and sets up the projection matrix.
2. void tile(float x, float y, char n): Draws a square (or tile) with a number inside it.
3. void draw flag(): Draws a flag to indicate page fault.
4. void setInput(): Sets up the input area displaying page requests and handles the animation of moving tiles.
5. void drawText(const char* string, float x, float y, float z): Draws a text string at the specified (x, y, z) position.
6. void setFrame(): Sets up the frame containing the pages in the buffer.
7. void display(): Main display function that sets up the visualization and calls other functions to draw elements.
8. void idle(): Idle function to update the animation.
9. void mouseFIFO(int btn, int state, int x, int y): Mouse function for handling the FIFO algorithm. It processes the left button clicks and animates the page replacement.
10. int getLRU(): Returns the index of the Least Recently Used (LRU) page in the buffer.
11. void mouseLRU(int btn, int state, int x, int y): Mouse function for handling the LRU algorithm. It processes the left button clicks and animates the page replacement.
12. void setBGColor(int action): Sets the background color based on user's choice.
13. void setTileColor(int action): Sets the tile color based on user's choice.
14. void menu(int action): Handles the main menu actions.
15. void addMenufifo(): Creates the right-click menu and submenus for the application.
16. void output(int x, int y, const char* string, int j): Outputs text at the specified (x, y) position with the given font style.
17. int main(int argc, char** argv): Main function that initializes GLUT, takes user input, and sets up the appropriate visualization based on the user's choice (FIFO or LRU).

Code

```
#include <stdlib.h>
#include <GL/glut.h>
#include <stdio.h>
#include <string.h>
#include <math.h>

// Integer arrays to store page requests, counters, pages, and colors.
int request[9] = { 0 }, counter[3] = { 0 }, pages[3] = { 0 };
int colour[9] = { 1, 1, 1, 1, 1, 1, 1, 1, 1 }, pagecolour[3] = { 1, 1, 1 };
// Integers to store the number of page faults and user's algorithm choice.
int faults = 0;
int choice;

void* fonts[] =
{
    GLUT_BITMAP_9_BY_15,
    GLUT_BITMAP_TIMES_ROMAN_10,
    GLUT_BITMAP_TIMES_ROMAN_24,
    GLUT_BITMAP_HELVETICA_12,
    GLUT_BITMAP_HELVETICA_10,
    GLUT_BITMAP_HELVETICA_18,
};
// Integer arrays to store page data and page faults.
int pageArray[9] = { 0 }, pageOfBuffer[3] = { 0 }, fault[9] = { 0 };
// Float array to store the positions of page elements.
float pagePosition[9] = { -5.5, -5.5, -5.5, -5.5, -5.5, -5.5, -5.5, -5.5, -5.5 };
// Integers for animation control and to count hits.
int hit = 0, step = -1, destination = 0, startAnimation = 0;
// Character array to store the result string.
char res[] = "No. of Page Faults are :  ";

// Float arrays to store the background and tile colors.
float backgroundColor[][3] = { {0, 0.3, 0.8}, {0, 0.8, 0.5}, {0.9, 0, 0} };
float tileColor[][3] = { {1, 1, 0.2}, {1, 0.7, 0.7}, {0.8, 0.8, 1} };
```

```
// Integer pointers for background and tile colors.
int bcPointer = 0, tcPointer = 0;

// This function initializes the OpenGL settings.
// It sets the color to black and configures the projection and modelview matrices.
void init()
{
    glColor3f(0, 0, 0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0, 800, 0, 600);
    glMatrixMode(GL_MODELVIEW);
}

// This function is responsible for drawing a square(or tile) with a number inside it.
// The input arguments are the x, y coordinates and the number to be displayed inside the
tile.
// It first draws a filled square with the current color, then a black border around it,
// and finally the number centered within the square.
void tile(float x, float y, char n)
{

    float size = 20;

    glBegin(GL_POLYGON);
    glVertex2f(x - size, y - size);
    glVertex2f(x + size, y - size);
    glVertex2f(x + size, y + size);
    glVertex2f(x - size, y + size);
    glEnd();

    glColor3f(0, 0, 0);
    glBegin(GL_LINE_LOOP);
    glVertex2f(x - size, y - size);
    glVertex2f(x + size, y - size);
```

```
    glVertex2f(x + size, y + size);
    glVertex2f(x - size, y + size);
    glEnd();
    glRasterPos2f(x - size / 2, y);
    glutBitmapCharacter(GLUT_BITMAP_9_BY_15, n);
}

// This function draws a small square (flag) to indicate a page fault.
void draw_flag()
{
    glColor3fv(backgroundColor[bcPointer]);
    glBegin(GL_POLYGON);
    glVertex2f(-10, -10);
    glVertex2f(10, -10);
    glVertex2f(10, 10);
    glVertex2f(-10, 10);
    glEnd();
}

// This function sets the input for the visualization by drawing the input memory pages
// And their corresponding positions, flags for page faults, and the moving animation.
void setInput()
{
    glColor3fv(backgroundColor[bcPointer]);

    glBegin(GL_POLYGON);
    glVertex2f(0, 0);
    glVertex2f(700, 0);
    glVertex2f(700, 100);
    glVertex2f(0, 100);
    glEnd();

    glPushMatrix();
    glTranslatef(-10, 40, 0);
    int i;
    for (i = 0; i < 9; i++)
```

```
    {
        glColor3fv(tileColor[tcPointer]);
        glTranslatef(70, 0, 0);
        glPushMatrix();

        if (pagePosition[i] > -4.5)
        {
            // Moving right
            glTranslatef(70 * step - 70 * i, 0, 0);
        }
        // Input positions
        glTranslatef(0, -250 - (45 * pagePosition[i]), 0);

        if ((int)pagePosition[i] == destination && pagePosition[i] >= 0)
            glColor3f(1, 0.3, 0.3);
        else
            glColor3fv(tileColor[tcPointer]);
        // glColor3f(0.1,0.5,0.1);

        tile(10, 10, pageArray[i] + '0');
        glPopMatrix();

        if (fault[i])
        {
            glPushMatrix();
            glTranslatef(0, -385, 0);
            draw_flag();
            glPopMatrix();
        }
    }
    glPopMatrix();
}

// This function is used to render text on the screen
// at the specified x, y, and z coordinates.
// It takes a string and the coordinates as input.
```

```
void drawText(const char* string, float x, float y, float z)
{
    const char* c;
    glRasterPos3f(x, y, z);
    for (c = string; *c != '\0'; c++)
    {
        if (*c == '\n')
            glRasterPos3f(x, y - 0.05, z);
        else
            glutBitmapCharacter(GLUT_BITMAP_9_BY_15, *c);
    }
}

// This function sets the frame buffer,
// which consists of three tiles for the memory pages.
void setFrame()
{
    glPushMatrix();

    tile(0, 0, pageOfBuffer[0] == 0 ? ' ' : pageOfBuffer[0] + '0');
    glTranslatef(0, -45, 0);

    tile(0, 0, pageOfBuffer[1] == 0 ? ' ' : pageOfBuffer[1] + '0');
    glTranslatef(0, -45, 0);

    tile(0, 0, pageOfBuffer[2] == 0 ? ' ' : pageOfBuffer[2] + '0');

    glPopMatrix();
}

// This function is called whenever the display needs to be updated.
//  It clears the screen, sets the identity matrix,
//  and then translates and renders the frame buffer and input memory pages.
// It also displays the number of page faults and the algorithm name.
void display()
{
```

```
glClear(GL_COLOR_BUFFER_BIT);
glLoadIdentity();

glPushMatrix();
// Frame Buffer position
glTranslatef(120 + (70 * step), 195, 0);
setFrame();
glPopMatrix();

glColor3f(1, 0, 0);
glPushMatrix();
// Input Red box position
glTranslatef(50, 400, 0);
setInput();
glPopMatrix();

glEnd();
if (step == 8)
{
    glColor3f(0, 0, 0);
    if (choice == 1)
    {
        res[24] = (9 - hit) + '0';
    }
    else
    {
        res[24] = (faults)+'0';
    }

    drawText(res, 50, 20, 0);
}
if (choice == 1)
{
    drawText("FIFO Page Replacement Algorithm", 250, 550, 0);
}
```

```
    else
    {
        drawText("LRU Page Replacement Algorithm", 250, 550, 0);
    }

    drawText("Fault->", 10, 50, 0);
    glFlush();
    glutSwapBuffers();
}

// This function is called during the idle time of the program.
// It updates the animation of the memory pages and checks if they have reached their
destination.
void idle()
{
    if (!startAnimation)
        return;
    printf("Idle called\n");
    printf("%d, %f\n", destination, pagePosition[step]);
    if (destination > pagePosition[step])
        pagePosition[step] += 0.10;

    if (destination < pagePosition[step]) // It has reached its location...So stop
animation
    {
        if (fault[step])
            pageOfBuffer[destination] = pageArray[step];

        startAnimation = 0;
        destination = -10;
    }
    display();
}

// These functions handle mouse events for the FIFO and LRU algorithms, respectively.
// They detect left mouse button clicks and update the visualization accordingly.
```

```
// They determine if there's a page fault, increment the hit counter, and set the animation
variables.
void mouseFIFO(int btn, int state, int x, int y)
{
    printf("Mouse function called\n");
    int n, i, j;

    if (startAnimation == 1)
    {
        printf("Animating");
        return;
    }
    if (btn == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
    {
        if (step < 9)
            step++;
        else
            printf("%d\n", hit);

        for (i = 0; i < 3; i++)
        {
            if (pageArray[step] == pageOfBuffer[i])
            {
                hit++;
                fault[step] = 0;
                startAnimation = 1;
                destination = -5;
                glutPostRedisplay();
                return;
            }
        }
        destination = ((step - hit) % 3);
        printf("%d\n", destination);
        startAnimation = 1;
        fault[step] = 1;
```

```
        // glutPostRedisplay();
    }
}

int getLRU()
{
    if (counter[0] >= counter[1] && counter[0] >= counter[2])
        return 0;
    if (counter[1] >= counter[0] && counter[1] >= counter[2])
        return 1;
    // if(counter[2]>=counter[1] && counter[2]>=counter[3]) return 2;
    return 2;
}

void mouseLRU(int btn, int state, int x, int y)
{
    int n, i, j;

    if (startAnimation == 1)
        return;
    if (btn == GLUT_LEFT_BUTTON && state == GLUT_DOWN && step < 8)
    {
        if (step < 9)
            step++;
        for (i = 0; i < 3; i++)
        {
            if (pageArray[step] == pageOfBuffer[i])
            {
                fault[step] = 0;
                counter[i] = 0;
                for (j = 0; j < 3; j++)
                    if (j != i)
                        counter[j]++;
                // dest=i;
                destination = -5;
                startAnimation = 1;
```

```
                colour[step] = 0;
                glutPostRedisplay();
                return;
            }
        }
        n = getLRU();
        counter[n] = 0;
        for (i = 0; i < 3; i++)
            if (i != n)
                counter[i]++;
        // pagecolour[n]=0;
        // assign[step]=n;
        destination = n;
        startAnimation = 1;
        fault[step] = 1;
        faults++;
    }

    glutPostRedisplay();
}

// These functions handle the options in the right-click menu,
// allowing users to change the background color, tile color, and exit the program.
void setBGColor(int action)
{
    bcPointer = action;
    glutPostRedisplay();
}

void setTileColor(int action)
{
    tcPointer = action;
    glutPostRedisplay();
}

void menu(int action)
```

```
{
    if (action == 0)
        exit(0);
}

// This function creates and attaches the right-click menu to the window.
void addMenufifo()
{
    int submenu1, submenu2;

    submenu1 = glutCreateMenu(setBGColor);
    glutAddMenuEntry("Blue", 0);
    glutAddMenuEntry("Green", 1);
    glutAddMenuEntry("Red", 2);

    submenu2 = glutCreateMenu(setTileColor);
    glutAddMenuEntry("Yellow", 0);
    glutAddMenuEntry("Light Red", 1);
    glutAddMenuEntry("Light Purple", 2);

    glutCreateMenu(menu);
    glutAddSubMenu("Background Colour", submenu1);
    glutAddSubMenu("Tile Colour", submenu2);
    glutAddMenuEntry("Quit", 0);
    glutAttachMenu(GLUT_RIGHT_BUTTON);
}

// This function is used to render text on the screen
// at the specified x and y coordinates using a specified font.
void output(int x, int y, const char* string, int j)
{
    int len, i;
    glColor3f(1.0f, 0.0f, 0.0f);
    glRasterPos2f(x, y);
    len = (int)strlen(string);
    for (i = 0; i < len; i++)
```

```
        glutBitmapCharacter(fonts[j], string[i]);
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);

    int i;
    printf("  ***********************************************\n");
    printf("          UCS505 Computer Graphics Project        \n");
    printf("       Topic: Visualisation of OS Algorithms      \n");
    printf("\n");
    printf("\n");
    printf("  Submitted By:\n");
    printf("  Jasween Kaur Brar, 102017187\n");
    printf("  Saurabh Thakur, 102017183\n");
    printf("  Aurav S. Tomar, 102067008\n");
    printf("  ***********************************************\n");
    printf("\n");
    printf("  Welcome to use our PROJECT!!\n");
    printf("  Note: Stack Size = 3 and Array Size = 9 is fixed\n\n");
    printf("  Page Replacement algorithms available:\n");
    printf("  1) FIFO {First In, First Out}\n");
    printf("  2) LRU  {Least Recently Used}\n");
    printf("  Enter your choice: ");
    scanf_s("%d", &choice);

    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
    // glutInitWindowSize(640,480);
    glutInitWindowSize(640, 480);
    if (choice == 1)
    {
        int j;
        printf("  Enter the page array of size 9\n");
        for (j = 0; j < 9; j++)
            scanf_s("%d", &pageArray[j]);
```

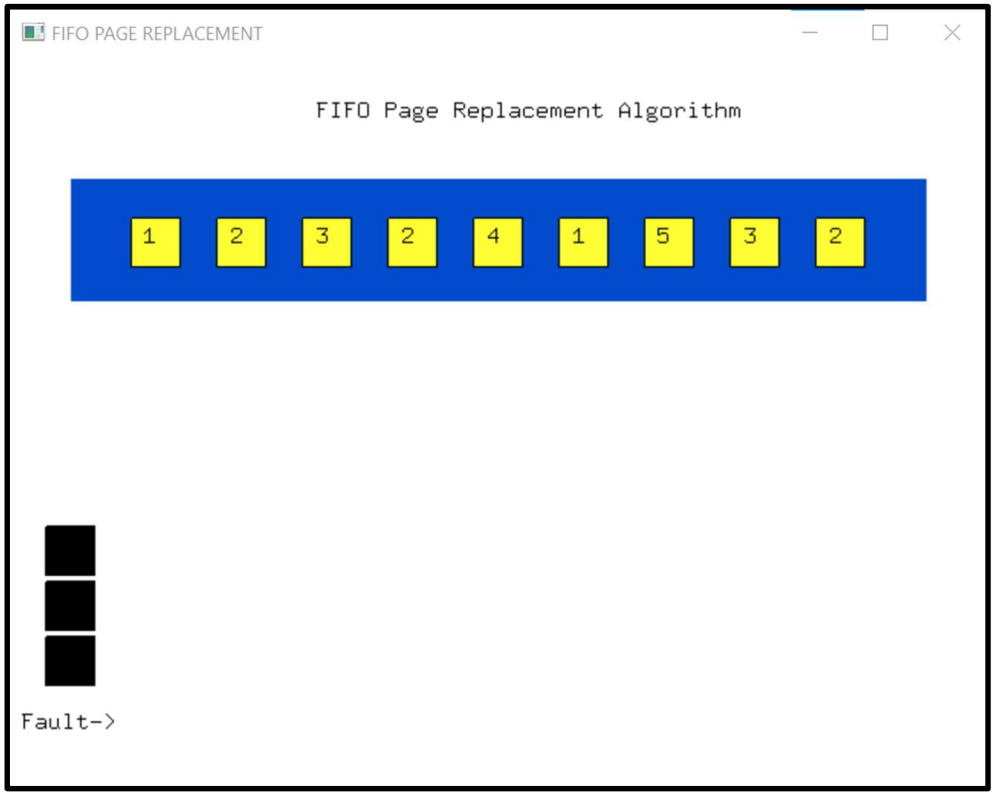
```
        glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
        glutInitWindowSize(640, 480);
        glutCreateWindow("FIFO PAGE REPLACEMENT");
        init();
        addMenufifo();
        glutDisplayFunc(display);
        glutMouseFunc(mouseFIFO);
        glutIdleFunc(idle);

        glClearColor(1, 1, 1, 1);
        glutMainLoop();
    }

    else
    {
        int j;
        printf("  Enter the page array of size 9\n");
        for (j = 0; j < 9; j++)
            scanf_s("%d", &pageArray[j]);

        glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
        glutInitWindowSize(640, 480);
        glutCreateWindow("LRU PAGE REPLACEMENT");
        init();
        addMenufifo();
        glutDisplayFunc(display);
        glutMouseFunc(mouseLRU);
        glutIdleFunc(idle);

        glClearColor(1, 1, 1, 1);
        glutMainLoop();
    }
    return 0;
}
```

Output/ Screenshots

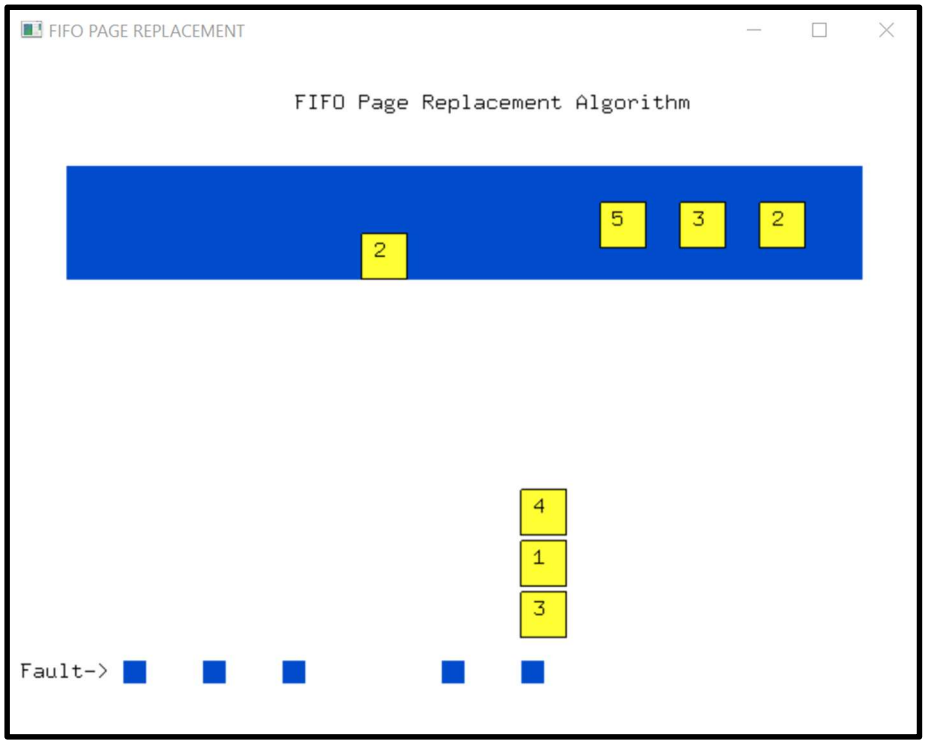
On running the code, we get this screen displaying necessary information and takes input from the user.

```
*********************************************
        UCS505 Computer Graphics Project
     Topic: Visualisation of OS Algorithms


 Submitted By:
 Jasween Kaur Brar, 102017187
 Saurabh Thakur, 102017183
 Aurav S. Tomar, 102067008
 *********************************************

 Welcome to use our PROJECT!!
 Note: Stack Size = 3 and Array Size = 9 is fixed

 Page Replacement algorithms available:
 1) FIFO {First In, First Out}
 2) LRU  {Least Recently Used}
 Enter your choice: 1
 Enter the page array of size 9
1 2 3 2 4 1 5 3 2
```

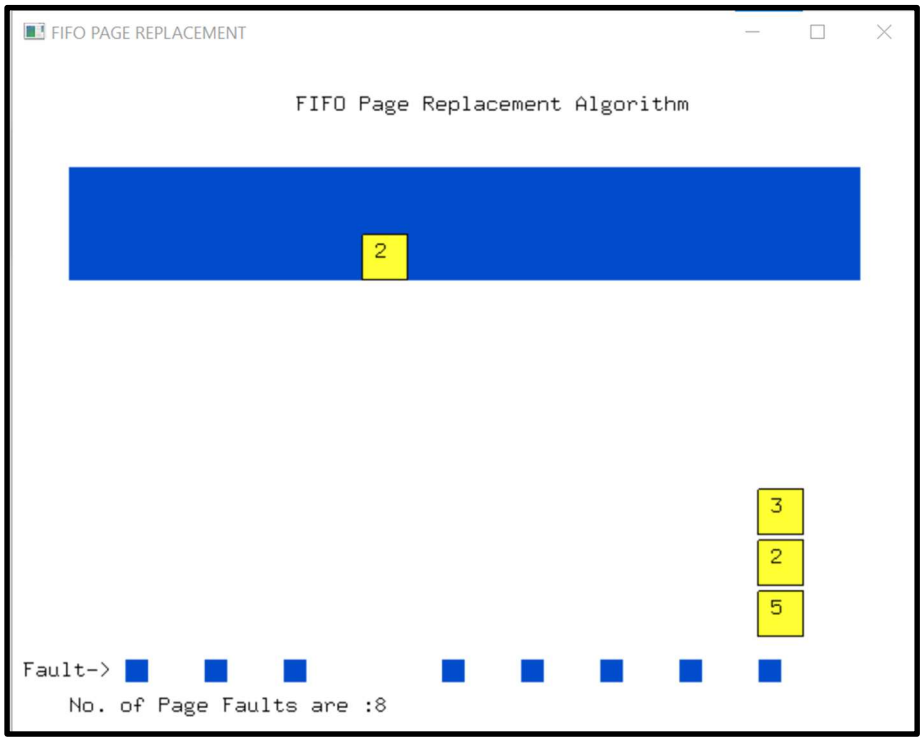
As the user chose the 1st option, a window opens providing the visualisation.



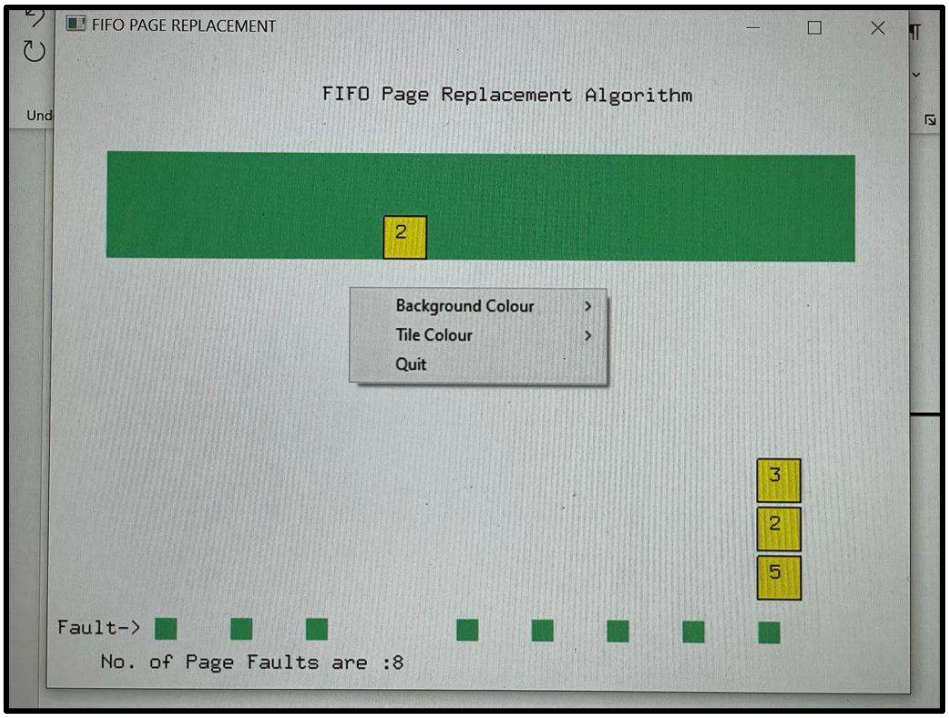
The code detects user input (left mouse button clicks) and trigger the execution of the FIFO or LRU page replacement algorithm accordingly through animation. The stack depicts memory buffer and page array is depicted on top. The page faults are shown with help of small squares.



This is the final output. Also, the total no. of page faults is calculated and shown at the bottom.



The code uses the GLUT menu functions that allows the user to change the background and tile colours or quit the application.



Similarly, when user chooses LRU Algorithm and provides input.

```
**********************************************
        UCS505 Computer Graphics Project
     Topic: Visualisation of OS Algorithms


 Submitted By:
 Jasween Kaur Brar, 102017187
 Saurabh Thakur, 102017183
 Aurav S. Tomar, 102067008
 **********************************************

 Welcome to use our PROJECT!!
 Note: Stack Size = 3 and Array Size = 9 is fixed

 Page Replacement algorithms available:
 1) FIFO {First In, First Out}
 2) LRU  {Least Recently Used}
 Enter your choice: 2
 Enter the page array of size 9
1 2 3 1 2 3 4 5 6
```

This is the corresponding output: