#include<GL/gl.h>

#include<GL/glut.h>

#include<windows.h>

#include<stdlib.h>

#include<stdio.h>

#include<math.h>

int frmcontent[3]={0,0,0},frames[3]={0,0,0}, temp[3],pages[9],curpage=0,pos[9],pgf=0,n;

char status[9];

int diffx=0,diffy=0,i=0;

void init()

{

glClearColor(0,0,0,0);

glClear(GL\_COLOR\_BUFFER\_BIT);

glMatrixMode(GL\_PROJECTION);

gluOrtho2D(0,1000,0,1000);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glFlush();

}

void drawText(char \*s,float x,float y,float z)

{

char \*c;

glRasterPos3f(x,y,z);

glColor3f(1,0,1);

for (c=s; \*c != '\0'; c++)

{

glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_24, \*c);

}

}

void clrsc()

{

if(strcmp(status,"HIT")==0)

{

glClearColor(0,0,0,0);

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(0.99,0.94,0.6);

drawText("LEAST RECENTLY USED ALGORITHM!!",-500,800,0);

glColor3f(0.75,0.7,1);

drawText("THE RESULTS ARE...",-330,650,0);

glColor3f(1,0,1);

drawText("PAGE FAULTS:",-300,550,0);

glRasterPos2f(50,550);

glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_24, pgf+'0');

drawText("PAGE HITS:",-250,450,0);

glRasterPos2f(30,450);

glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_24, (n-pgf)+'0');

glFlush();

}

else

{

glClearColor(0,0,0,0);

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(0.99,0.94,0.6);

drawText("LEAST RECENTLY USED ALGORITHM!!",70,1150,0);

glColor3f(0.75,0.7,1);

drawText("THE RESULTS ARE...",270,1000,0);

glColor3f(1,0,1);

drawText("PAGE FAULTS:",300,900,0);

glRasterPos2f(650,900);

glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_24, pgf+'0');

drawText("PAGE HITS:",350,800,0);

glRasterPos2f(630,800);

glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_24, (n-pgf)+'0');

glFlush();

}

}

int lru(int pages[9],int cpg) //returns position to fill page in

{

int m, n, position, k, l;

int a = 0, b = 0;

for(m = 0; m < 3; m++)

if(frames[m] == pages[cpg]) //page hit

{

a = 1;

b = 1;

return -1;

}

if(a == 0) //frame empty insert

{

for(m = 0; m < 3; m++)

if(frames[m] == 0)

{

frames[m] = pages[cpg];

b = 1;

pgf++;

return m;

}

}

if(b == 0) //page fault

{

for(m = 0; m < 3; m++)

temp[m] = 0;

for(k = cpg-1, l = 1; l <=2; l++, k--)

{

for(m = 0; m <3; m++)

if(frames[m] == pages[k])

temp[m] = 1;

}

for(m = 0; m <3; m++)

if(temp[m] == 0)

{

position = m;

break;

}

frames[position] = pages[cpg];

pgf++;

return position;

}

}

void boxpush(int x)

{

glBegin(GL\_POLYGON);

glColor3f(0.8,0.84,1); //initialize box with current page

glVertex2f(470, 770);

glVertex2f(530, 770);

glVertex2f(530, 830);

glVertex2f(470, 830);

glEnd();

glColor3f(0,0,0);

glRasterPos2f(492, 796);

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_12, x+'0');

Sleep(50);

diffy-=10;

if(pos[curpage]==0&&diffy>-200) //if curpage is inserted to frame 1

{

strcpy(status,"FAULT");

glutPostRedisplay(); //redisplay after each change in position

}

else if(pos[curpage]==1&&diffy>-300) //if curpage is inserted to frame 2

{

strcpy(status,"FAULT");

glutPostRedisplay();

}

else if(pos[curpage]==2&&diffy>-380) //if curpage is inserted to frame 3

{

strcpy(status,"FAULT");

glutPostRedisplay();

}

else if(pos[curpage]==-1&&diffx<550) //if curpage is hit

{

Sleep(50);

diffy=0;

diffx+=20;

strcpy(status,"HIT");

glutPostRedisplay();

}

else //move to next page

{

if(curpage<n-1) //if 8th page increment and display one final time

{

frmcontent[pos[curpage]]=pages[curpage]; //assign curpage value to chosen frame

curpage++;

diffy=0;

diffx=0; //reset positions

glutPostRedisplay();

}

else

{

clrsc(); //display result

}

}

}

void drawframes() //draws frames with current content

{

glBegin(GL\_POLYGON);

glColor3f(0.5,0.9,0.56);

glVertex2f(470, 570);

glVertex2f(530, 570);

glVertex2f(530, 630);

glVertex2f(470, 630);

glEnd();

glColor3f(0,0,0);

glRasterPos2f(492,592);

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_12, frmcontent[0]+'0');

glColor3f(0.5,0.9,0.56);

glBegin(GL\_POLYGON);

glVertex2f(470, 490);

glVertex2f(530, 490);

glVertex2f(530, 550);

glVertex2f(470, 550);

glEnd();

glColor3f(0,0,0);

glRasterPos2f(492,512);

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_12, frmcontent[1]+'0');

glColor3f(0.5,0.9,0.56);

glBegin(GL\_POLYGON);

glVertex2f(470, 410);

glVertex2f(530, 410);

glVertex2f(530, 470);

glVertex2f(470, 470);

glEnd();

glColor3f(0,0,0);

glRasterPos2f(492,432);

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_12, frmcontent[2]+'0');

}

void drawstatus() //draws status of each page

{

glColor3f(0.99,0.94,0.6);

drawText("LEAST RECENTLY USED ALGORITHM!!",60,880,0);

glColor3f(1,0,1);

drawText("INPUT",190,780,0);

glColor3f(1,0,1);

glRasterPos2f(380, 780);

glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_24, curpage+1+'0');

drawText("SLOT 1",200,580,0);

drawText("SLOT 2",200,500,0);

drawText("SLOT 3",200,420,0);

drawText("STATUS:",190,200,0);

drawText(status,450,200,0);

}

void disp()

{

glClearColor(0,0,0,0);

glClear(GL\_COLOR\_BUFFER\_BIT);

drawframes();

drawstatus();

glPushMatrix();

glTranslatef(diffx,diffy,0); //push box down/left

boxpush(pages[curpage]); //keep pushing till reach end

glPopMatrix();

glutSwapBuffers();

glFlush();

}

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

{

int i;

printf("Enter the number of pages (maximum 9):\n");

scanf("%d",&n);

printf("Enter the sequence of pages:\n");

for(i=0;i<n;i++)

scanf("%d",&pages[i]);

glutInit(&argc,argv);

glutInitWindowSize(500,500);

glutCreateWindow("LEAST RECENTLY USED ALGORITHM");

init();

glutDisplayFunc(disp);

printf("\nPAGE NO.\tINSERTED INTO FRAME NO.\t\tFRAME STATE\n");

for(i=0;i<n;i++)

{

pos[i]=lru(pages,i);

printf(" %d\t\t\t %d\t\t\t %d %d %d\n",pages[i],pos[i]+1,frames[0],frames[1],frames[2]);

}

printf("\nPAGE FAULTS=%d\n",pgf);

printf("PAGE HITS=%d\n",n-pgf);

glutMainLoop();

return 0;

}