Једноставне анимације

Координатни систем

Кад год се помери прозор у коме се црта или му се промени величина, оперативни систем обрађује тај сигнал. Програмер има могућност да обради и одреагује на такав сигнал рутином коју проследи функцији `glutReshapeFunc()`. Она се позива сваки пут када се добије овакав сигнал од оперативног система. Та функција мора поново да формира правоугаони регион који ће бити рендерован, као и да дефинише координатни систем у коме се објекти цртају.

На примеру испод се може видети функција `reshape(int w, int h)` која се прослеђује функцији `glutReshapeFunc(reshape)`.

```
void reshape(int w, int h)
{
	glViewport(0, 0, (GLsizei) w, (GLsizei) h);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(0.0, (GLdouble) w, 0.0, (GLdouble) h);
}
```

Функцију `reshape` креира програмер тако да мора да прима два параметра која представљају димензије у пикселима прозора који је променио величину или је само померен. Наредба `glViewport` подешава правоугаоник за цртање тако да се цртање врши на целом прозору. Наредне три наредбе (`glMatrixMode`, `glLoadIdentity` и `gluOrtho2D`) подешавају координатни систем за цртање тако да у доњем левом углу екрана буде тачка са координатама $(0,0)$, а у горњем левом углу тачка са координатама (w, h). Уопштено, не морају да буду такве координате доње леве и горње десне тачке екрана, што зависи од тога шта програмер жели да прикаже на екрану.

Анимација

Анимација се добија узастопним цртањем слика (фрејмова). Исцртавање фрејмова може да се извршава са неком задатом учестаношћу или пак на неки улазни догађај иницран од стране корисника.

Руковање улазним догађајима

За комуникацију апликације са оперативним системом користе се функције библиотеке ***GLUT***. Библиотека ***GLUT*** поседује методе којима се може одреаговати на сигнале које шаље оперативни систем када дође до догађаја који је иницран од стране корисника. Функција која се користи при промени величине прозора или померању прозора је већ поменута, међутим поред ње постоје и функције које препознају догађаје тастатуре или миша:

- **glutReshapeFunc(void(*func)(int w, int h))** – Указује на акцију која се извршава приликом промене величине прозора.
- **glutKeyboardFunc(void(*func)(unsigned char key, int x, int y))** – Указује на акцију која се извршава уколико се притисне неко дугме на тастатури.

- **glutMouseFunc(void(*func)(int button, int state, int x, int y))** – Указује на акцију која се извршава ако се притисне тастер миша.
- **glutMotionFunc(void(*func)(int x, int y))** – Регистровање рутине која се извршава приликом померања миша.

Програмер креира функцију која ће се извршити као одговор на сигнал који шаље оперативни систем. Та функција се прослеђује једној од наведених функција у зависности од тога на који сигнал програмер жели да одговори.

Да би апликација реаговала на промену стања тастера миша, метода коју креира корисник прослеђује се функцији

`glutMouseFunc(void(*func)(int button, int state, int x, int y)).`

Та метода мора да има четири целобројна параметра. Први параметар је константа која представља тастер миша које је иницирао сигнал и може бити `GLUT_LEFT_BUTTON`, `GLUT_RIGHT_BUTTON` или `GLUT_MIDDLE_BUTTON`. Други параметар указује на стање тастера миша које је настало приликом генерисања сигнала. Може бити `GLUT_DOWN` (притиснут тастер) или `GLUT_UP` (пуштен тастер).

Битно је напоменути да је у ***OpenGL***-у могуће задати функцију која ће се изнова извршавати увек када се не врши обрада неког сигнала оперативног система. То се ради прослеђивањем показивача на ту функцију методи `glutIdleFunc(void (*func)(void))`. Уколико се проследи NULL, онемогућава се извршавање претходно послате функције.

У примеру испод, када се притисне леви тастер миша почиње извршавање функције `ChangePosition` кад год се не обрађује неки догађај. Функција `ChangePosition` се извршава у континуитету, односно чим се заврши њено извршавање почиње поновно извршавање исте. Притиском не десни тастер миша методи `glutIdleFunc` се прослеђује `NULL` па извршавање фунцкије `ChangePosition` престаје.

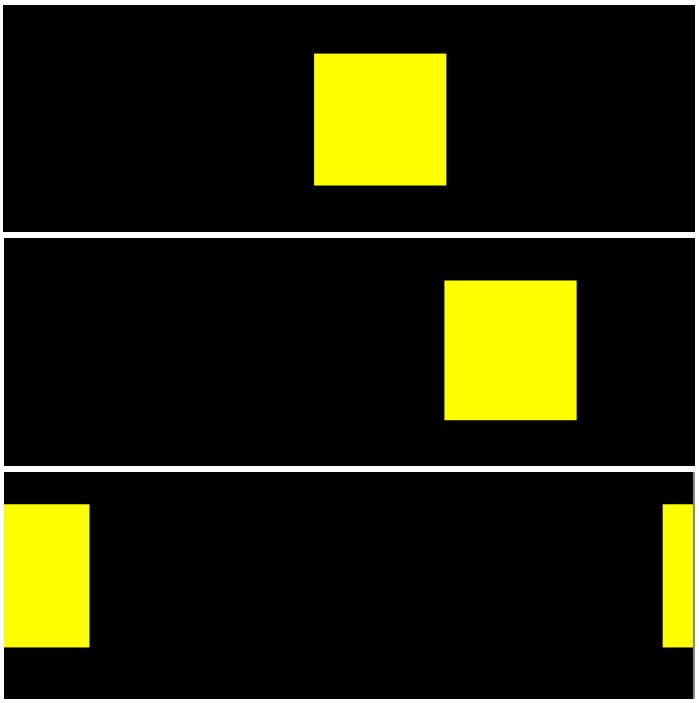
```
void mousePress(int button, int state, int x, int y)
{
    switch (button)
    {
    case GLUT_LEFT_BUTTON:
        if(state == GLUT_DOWN)
            glutIdleFunc(ChangePosition);
        break;
    case GLUT_RIGHT_BUTTON:
        if(state == GLUT_DOWN)
            glutIdleFunc(NULL);
        break;
    default:
        break;
    }
}
```

Остале функције за руковање улазним догађајима тастатуре и миша користе се на сличан начин као и већ објашњена `glutMouseFunc`. У оквиру наредних поглавља биће више речи о њима.

Задаци

1. Нацртати произвољан квадрат на произвољној позицији на екрану, и направити анимацију у којој ће тај квадрат непрестано мењати боје (црвена-зелена-плава-црвена-зелена-.....) и то највећом могућом брзином.

2. Нацртати произвољан квадрат на произвољној позицији на екрану, а потом направити анимацију где ће се тај квадрат праволинијски кретати удесно произвољном брзином. Када квадрат почне да нестаје са екрана са десне стране, треба да почне да се појављује са леве стране екрана (Слика 1). Анимација треба да се извршава само док је леви тастер миша притиснут.



Слика 1. Квадрат нестаје десно са екрана, а појављује се лево

Решење

1. На претходним часовима било је речи о цртању квадрата. Да би се мењале боје квадрата, довољно је функцији `glutIdleFunc()` проследити методу `display()` у којој се параметри за избор боје мењају тако да ако је била црвена, поставља се зелена, ако је била зелена поставља се плава, а ако је била плава поставља се црвена боја. Тиме ће се функција `display()` извршавати у континуитету једна за другом.

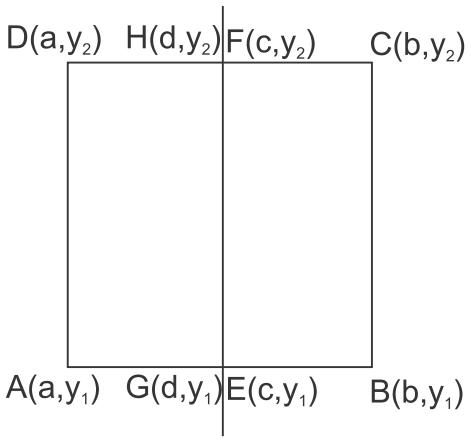
Непрестано мењање боја квадрата се може постићи и тако што се функцији `glutDisplayFunc()` проследи функција `display()`, на чијем крају се позива функција `glutPostRedisplay()`. Позив функције `glutPostRedisplay()` изазива тренутно исцртавање прозора, што у овом случају значи поновни позив функције `display()`.Оваква програмска конструкција за последицу има непрестано позивање функције `display()`, а самим тим и промену боја и исцртавање квадрата изнова.

```
void display()
{
    glClear (GL_COLOR_BUFFER_BIT);

    if(red == 1.0)
    {
        red = 0.0;
        green = 1.0;
    }
    else if(green == 1.0)
    {
        green = 0.0;
        blue = 1.0;
    }
    else if(blue == 1.0)
    {
        blue = 0.0;
        red = 1.0;
    }

    glColor3f (red, green, blue);
    glBegin(GL_POLYGON);
        glVertex3f (0.25, 0.25, 0.0);
        glVertex3f (0.75, 0.25, 0.0);
        glVertex3f (0.75, 0.75, 0.0);
        glVertex3f (0.25, 0.75, 0.0);
    glEnd();

    glFlush ();
    glutPostRedisplay();
}
```

2. У функцији `mousePress(int button, int state, int x, int y)` испод, када је леви тастер миша притиснут методи `glutIdleFunc()` се прослеђује функција `ChangePosition()` у којој се мењају координате објекта који се црта и у којој се иницира његово исцртавање. Уколико је притиснут леви тастер миша, функција `ChangePosition()` ће се извршавати кад год нема другог догађаја. Чим се пусти леви тастер миша, функцији `glutIdleFunc()` се прослеђује `NULL` што доводи до престанка извршавања функције `ChangePosition()`.

```
void mousePress(int button, int state, int x, int y)
{
    switch (button)
    {
    case GLUT_LEFT_BUTTON:
        if(state == GLUT_DOWN)
            glutIdleFunc(ChangePosition);
        else glutIdleFunc(NULL);
        break;
    default:
        break;
    }
}
```

При хоризонталном кретању објекта, y координата чворова ће остати иста, док ће се мењати x координата. Координате тих тачака се мењају у функцији `ChangePosition()`, и то при сваком поновном позиву, x координата свих тачака се мења за 0.05 у одговарајућем смеру. Међутим, тиме би се квадрат увек померао у десно, а потребно је када почне да се губи са екрана десно, да почне да се појављује лево, односно да кружи.Та два дела се морају цртати потпуно независно па се може закључити да се увек квадрат мора цртати из два дела која ће бити спојена осим у наведеном случају (Слика 2).



Слика 2. Квадрат се црта из два дела због потребе анимације

Дакле, цртају се два правоугаоника уместо један квадрат и то $AGHD$ и $EBCF$. Иницијално се координате a, b, c, d, y_1, y_2 постављају тако да четвороугао $ABCD$ буде квадрат.

Нека је координатни систем за цртање дефинисан функцијом

```
glOrtho(0.0, Y_AXES * aspectRatio, 0.0, Y_AXES, -1.0, 1.0)
```

односно доња лева тачка екрана има координате $(0.0,0.0)$, а горња десна тачка координате $(Y\_AXES \cdot aspectRatio, Y\_AXES)$ где су Y_AXES и $aspectRatio$ константне вредности.

Када је приликом анимације координата a већа од 0.0 икоорднината b мања од $Y\_AXES \cdot aspectRatio$ правоугаоници треба да буду спојени, односно да буде $c = d$. У супротном, потребно је цртати два одвојена правоугаоника, и то $AGHD$ до десне ивице екрана, а $EBCF$ од леве ивице екрана. У том случају треба да буде $c = 0.0$, а $d = Y\_AXES \cdot aspectRatio$.

Испод се може видети код функције `ChangePosition()` која на већ описан начин решава проблем промене координата квадрата приликом описаног кретања. Променљиве a, b, c, d и $flag\_cut$ су глобалне.

```
void ChangePosition(void)
{
    if(b > Y_AXES* aspectRatio || flag_cut == 1)
    {
        flag_cut = 1;

        if(b+0.05 > Y_AXES* aspectRatio)
        {
            b = 0.05;
            d = Y_AXES* aspectRatio;
        }
        else
            b += 0.05;

        a += 0.05;
    }

    if(a+0.05 > Y_AXES* aspectRatio && flag_cut == 1)
    {
        a = 0.0;
        b += 0.05;
        c += 0.05;
        d = c;
        flag_cut = 0;
    }

    if (flag_cut == 0)
    {
        a += 0.05;
        b += 0.05;
        c += 0.05;
        d = c;
    }

    glutPostRedisplay();
}
```

Програмски код

Пример 1.Мењање боје квадрата. Квадрат који се креће.

```
#include<freeglut.h>
#include <math.h>

int red = 1.0;
int green = 0.0;
int blue = 0.0;

double WINDOW_HIGHT = 500;
double Y_AXES = 2.0;

double aspectRatio = 4.0/3.0;

double a = 1.25;
double b = 1.75;
double c = 1.50;
double d = 1.50;

int flag_cut = 0;

void display()
{
    glClear (GL_COLOR_BUFFER_BIT);

    if(red == 1.0)
    {
        red = 0.0;
        green = 1.0;
    }
    else if(green == 1.0)
    {
        green = 0.0;
        blue = 1.0;
    }
    else if(blue == 1.0)
    {
        blue = 0.0;
        red = 1.0;
    }

    glColor3f (red, green, blue);
    glBegin(GL_POLYGON);
        glVertex3f (0.25, 0.25, 0.0);
        glVertex3f (0.75, 0.25, 0.0);
        glVertex3f (0.75, 0.75, 0.0);
        glVertex3f (0.25, 0.75, 0.0);
    glEnd();

    glColor3f(1.0, 1.0, 0.0);
```

```
    glBegin(GL_POLYGON);
        glVertex3f (a, 1.25, 0.0);
        glVertex3f (d, 1.25, 0.0);
        glVertex3f (d, 1.75, 0.0);
        glVertex3f (a, 1.75, 0.0);
    glEnd();

    glBegin(GL_POLYGON);
    glVertex3f (c, 1.25, 0.0);
    glVertex3f (b, 1.25, 0.0);
    glVertex3f (b, 1.75, 0.0);
    glVertex3f (c, 1.75, 0.0);
    glEnd();

    glFlush ();
    glutPostRedisplay();
}

void ChangePosition(void)
{
    if(b > Y_AXES* aspectRatio || flag_cut == 1)
    {
        flag_cut = 1;

        if(b+0.05 > Y_AXES* aspectRatio)
        {
            b = 0.05;
            c = 0.0;
            d = Y_AXES* aspectRatio;
        }
        else
        {
            b += 0.05;
            c = 0.0;
        }
        a += 0.05;
    }

    if(a+0.05 > Y_AXES* aspectRatio && flag_cut == 1)
    {
        a = 0.0;
        b += 0.05;
        c += 0.05;
        d = c;
        flag_cut = 0;
    }

    if (flag_cut == 0)
    {
        a += 0.05;
        b += 0.05;
        c += 0.05;
        d = c;
```

```
    }

    glutPostRedisplay();
}

void init (void)
{
    glClearColor (0.0, 0.0, 0.0, 0.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(0.0, Y_AXES * aspectRatio, 0.0, Y_AXES, -1.0, 1.0);
}

void reshape(int w, int h)
{
    if(w < h)
        WINDOW_HIGHT = w / aspectRatio;
    else
        WINDOW_HIGHT = h;

    glViewport(0,0,WINDOW_HIGHT*aspectRatio,WINDOW_HIGHT);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(0.0, Y_AXES * aspectRatio, 0.0, Y_AXES, -1.0, 1.0);
}

void mousePress(int button, int state, int x, int y)
{
    switch (button)
    {
    case GLUT_LEFT_BUTTON:
        if(state == GLUT_DOWN)
            glutIdleFunc(ChangePosition);
        else glutIdleFunc(NULL);
        break;
    default:
        break;
    }
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize (WINDOW_HIGHT*aspectRatio, WINDOW_HIGHT);
    glutInitWindowPosition (100, 100);
    glutCreateWindow ("hello");

    init ();

    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutMouseFunc(mousePress);
```

```
    glutMainLoop();
    return 0;
}
```

Домаћи задатак

За домаћи треба нацртати ветрењачу и сунце. Притом, елиса ветрењаче треба да се окреће све време. На клик миша треба да се помера сунце, које кретањем описује кружни лук. Анимација ради док је притиснут клик на мишу, у супротном не ради.

Рок за слање радова је среда, 24.10.2018. до 14h.

Радове слати на email: lazarkrstic94@gmail.com.