|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации Мытищинский филиал**  **Федерального государственного бюджетного образовательного учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МФ МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ \_\_\_\_\_\_космический\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_\_\_\_К-3\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**отчет**

***К ЛАБОРАТОРНОЙ РАБОТЕ***

***№ 5***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

# *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_* по ДИСЦИПЛИНЕ

**«компьютерная графика»**

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

Студент \_\_\_\_\_\_К3-51Б\_\_\_\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Бортников М.Д.

(Группа) (Подпись, дата) (И.О.Фамилия)

Преподаватель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_**В.В. Афанасьева**\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

*2022 г.*

**Задание к лабораторной работе № 5**

Разработать и реализовать задачу о 2D движении кривошипно-шатунного механизма, разобранного в задаче 2\_1 (см. приложенный файл «задача\_2.1.pdf»).

Здесь точка О - неподвижная, точка В совершает движения только вдоль прямой Ох, в точке А-шарнир, ОА и АВ – стержни. Движение должно начинаться при t=0, т.е. из положения, когда угол фи=0.

В окне формы:

1. должна быть возможность ввода длины ОА,
2. должна быть возможность ввода длины АВ,
3. должна быть возможность ввода длины АМ (точка М находится на стержне АВ),
4. должна быть возможность отображения траектории точки М (можно сделать чекбокс (отображать траекторию или нет)),
5. должно быть отображено 2D движение всего механизма,
6. надо на форме задавать время перерисовки, чтобы можно было ускорить или замедлить движение.

**Код программы**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

// для работы с библиотекой OpenGL

using Tao.OpenGl;

// для работы с библиотекой FreeGLUT

using Tao.FreeGlut;

// для работы с элементом управления SimpleOpenGLControl

using Tao.Platform.Windows;

namespace Laba\_2

{

public partial class Form1 : Form

{

private float OA, AB, AM;

private float ScreenH = 475, ScreenW = 620;

private float xA, yA, xM, yM, xB;

private float angle;

private float[,] GraphValuesArray;

private int elements\_count = 0;

private bool not\_calculate;

private int check = 0;

public Form1()

{

InitializeComponent();

AnT.InitializeContexts();

}

private void Form1\_Load(object sender, EventArgs e)

{

// инициализация Glut

Glut.glutInit();

Glut.glutInitDisplayMode(Glut.GLUT\_RGB | Glut.GLUT\_SINGLE);

// очистка окна

Gl.glClearColor(255, 255, 255, 1);

// установка порта вывода в соответствии с размерами элемента anT

Gl.glViewport(0, 0, AnT.Width, AnT.Height);

// настройка проекции

Gl.glMatrixMode(Gl.GL\_PROJECTION);

Gl.glLoadIdentity();

// теперь необходимо корректно настроить 2D ортогональную проекцию

// в зависимости от того, какая сторона больше

// мы немного варьируем то, как будет сконфигурированный настройки проекции

if ((float)AnT.Width <= (float)AnT.Height)

{

Glu.gluOrtho2D(0.0, 30.0 \* (float)AnT.Height / (float)AnT.Width, 0.0, 30.0);

}

else

{

Glu.gluOrtho2D(0.0, 30.0 \* (float)AnT.Width / (float)AnT.Height, 0.0, 30.0);

}

Gl.glMatrixMode(Gl.GL\_MODELVIEW);

Gl.glLoadIdentity();

}

private void new\_parametrs()

{

OA = trackBar1.Value;

AB = trackBar2.Value;

AM = trackBar3.Value;

ScreenW = (OA + AB) \* 2.2f \* (float)AnT.Width / (float)AnT.Height;

ScreenH = (OA + AB) \* 2.2f;

}

private void Graph()

{

Gl.glBegin(Gl.GL\_LINE\_STRIP);

Gl.glVertex2d(-ScreenW / 2, 0);

Gl.glVertex2d(ScreenW / 2, 0);

Gl.glEnd();

Gl.glBegin(Gl.GL\_LINE\_STRIP);

Gl.glVertex2d(0, -ScreenH / 2);

Gl.glVertex2d(0, ScreenH / 2);

Gl.glEnd();

Gl.glPointSize(5);

Gl.glBegin(Gl.GL\_POINTS);

Gl.glColor3f(255, 0, 0);

Gl.glVertex2d(0, 0);

Gl.glVertex2d(xA, yA);

Gl.glVertex2d(xB, 0);

Gl.glVertex2d(xM, yM);

Gl.glColor3f(0, 0, 0);

Gl.glEnd();

Gl.glPointSize(1);

PrintText2D((OA + AB) / 20, (OA + AB) / 20, "O");

PrintText2D(xA, yA + (OA + AB) / 20, "A");

PrintText2D(xM, yM - (OA + AB) / 15, "M");

PrintText2D(xB, (OA + AB) / 20, "B");

Gl.glLineWidth(2);

Gl.glBegin(Gl.GL\_LINE\_STRIP);

Gl.glVertex2d(0, 0);

Gl.glVertex2d(xA, yA);

Gl.glVertex2d(xB, 0);

Gl.glEnd();

Gl.glLineWidth(1);

}

private void Move\_var1()

{

xM = (float)((OA + AM) \* Math.Cos(Radian(angle)));

yM = (float)((OA - AM) \* Math.Sin(Radian(angle)));

if (angle <= 360)

angle += 0.5f;

else

angle = 0.5f;

xA = (float)((OA) \* Math.Cos(Radian(angle)));

yA = (float)((OA) \* Math.Sin(Radian(angle)));

xB = (float)((OA + AB) \* Math.Cos(Radian(angle)));

}

private void Move\_var2()

{

switch (check)

{

case 0:

{

if (yA < AB)

{

xA = (float)(OA \* Math.Cos(Radian(angle)));

yA = (float)((OA) \* Math.Sin(Radian(angle)));

if (angle <= Math.Asin(AB / OA) \* 180 / Math.PI)

{

xB = (float)(OA \* Math.Cos(Radian(angle)) + Math.Sqrt(AB \* AB - (OA \* OA) \* Math.Sin(Radian(angle)) \* Math.Sin(Radian(angle))));

xM = (float)(xB - Math.Sqrt(AB \* AB - (OA \* OA) \* Math.Sin(Radian(angle)) \* Math.Sin(Radian(angle))) / AB \* (AB - AM));

yM = (float)((AB - AM) \* Math.Sqrt(1 - Math.Sqrt(AB \* AB - (OA \* OA) \* Math.Sin(Radian(angle)) \* Math.Sin(Radian(angle))) / AB

\* Math.Sqrt(AB \* AB - (OA \* OA) \* Math.Sin(Radian(angle)) \* Math.Sin(Radian(angle))) / AB));

}

else

{

xB = (float)(xA);

xM = xB;

yM = yA - AM;

}

angle += 0.5f;

}

else

check = 1;

break;

}

case 1:

{

if (-yA <= AB)

{

angle -= 0.5f;

if (angle >= Math.Asin(AB / OA) \* 180 / Math.PI)

angle -= 0.5f;

xB = (float)(xA - Math.Sqrt(AB \* AB - (OA \* OA) \* Math.Sin(Radian(angle)) \* Math.Sin(Radian(angle))));

xA = (float)(OA \* Math.Cos(Radian(angle)));

yA = (float)((OA) \* Math.Sin(Radian(angle)));

if (angle >= 0)

{

yM = (float)((AB - AM) \* Math.Sqrt(1 - Math.Sqrt(AB \* AB - (OA \* OA) \* Math.Sin(Radian(angle)) \* Math.Sin(Radian(angle))) / AB

\* Math.Sqrt(AB \* AB - (OA \* OA) \* Math.Sin(Radian(angle)) \* Math.Sin(Radian(angle))) / AB));

}

else

{

yM = (float)(-(AB - AM) \* Math.Sqrt(1 - Math.Sqrt(AB \* AB - (OA \* OA) \* Math.Sin(Radian(angle)) \* Math.Sin(Radian(angle))) / AB

\* Math.Sqrt(AB \* AB - (OA \* OA) \* Math.Sin(Radian(angle)) \* Math.Sin(Radian(angle))) / AB));

}

xM = (float)(xB + Math.Sqrt(AB \* AB - (OA \* OA) \* Math.Sin(Radian(angle)) \* Math.Sin(Radian(angle))) / AB \* (AB - AM));

if (angle <= -Math.Asin(AB / OA) \* 180 / Math.PI)

{

xB = xA;

xM = xA;

yM = yA + AM;

}

}

else

{

check = 3;

}

break;

}

case 3:

{

if (angle < 0)

{

angle += 0.5f;

xA = (float)(OA \* Math.Cos(Radian(angle)));

yA = (float)((OA) \* Math.Sin(Radian(angle)));

xB = (float)(xA + Math.Sqrt(AB \* AB - (OA \* OA) \* Math.Sin(Radian(angle)) \* Math.Sin(Radian(angle))));

xM = (float)(xB - Math.Sqrt(AB \* AB - (OA \* OA) \* Math.Sin(Radian(angle)) \* Math.Sin(Radian(angle))) / AB \* (AB - AM));

yM = (float)(-(AB - AM) \* Math.Sqrt(1 - Math.Sqrt(AB \* AB - (OA \* OA) \* Math.Sin(Radian(angle)) \* Math.Sin(Radian(angle))) / AB

\* Math.Sqrt(AB \* AB - (OA \* OA) \* Math.Sin(Radian(angle)) \* Math.Sin(Radian(angle))) / AB));

}

else

{

check = 0;

}

break;

}

}

}

private void Move\_var3()

{

xA = (float)(OA \* Math.Cos(Radian(angle)));

yA = (float)((OA) \* Math.Sin(Radian(angle)));

xB = (float)(xA + Math.Sqrt(AB \* AB - (OA \* OA) \* Math.Sin(Radian(angle)) \* Math.Sin(Radian(angle))));

xM = (float)(xB - Math.Sqrt(AB \* AB - (OA \* OA) \* Math.Sin(Radian(angle)) \* Math.Sin(Radian(angle))) / AB \* (AB - AM));

if (angle <= 180)

yM = (float)((AB - AM) \* Math.Sqrt(1 - Math.Sqrt(AB \* AB - (OA \* OA) \* Math.Sin(Radian(angle)) \* Math.Sin(Radian(angle))) / AB

\* Math.Sqrt(AB \* AB - (OA \* OA) \* Math.Sin(Radian(angle)) \* Math.Sin(Radian(angle))) / AB));

else

yM = (float)(-(AB - AM) \* Math.Sqrt(1 - Math.Sqrt(AB \* AB - (OA \* OA) \* Math.Sin(Radian(angle)) \* Math.Sin(Radian(angle))) / AB

\* Math.Sqrt(AB \* AB - (OA \* OA) \* Math.Sin(Radian(angle)) \* Math.Sin(Radian(angle))) / AB));

if (angle < 360)

angle += 0.5f;

else

angle = 0;

}

private void button2\_Click(object sender, EventArgs e)

{

if (button2.Text == "Пауза")

{

timer1.Enabled = false;

button2.Text = "Продолжить";

}

else

{

timer1.Enabled = true;

button2.Text = "Пауза";

}

}

private void Draw\_Way()

{

Gl.glBegin(Gl.GL\_LINE\_STRIP);

Gl.glColor3f(255, 0, 0);

for (int ax = 0; ax < elements\_count; ax += 1)

{

Gl.glVertex2d(GraphValuesArray[ax, 0], GraphValuesArray[ax, 1]);

}

Gl.glColor3f(0, 0, 0);

Gl.glEnd();

}

private void trackBar4\_Scroll(object sender, EventArgs e) { }

private void button3\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void Draw()

{

Gl.glClear(Gl.GL\_COLOR\_BUFFER\_BIT | Gl.GL\_DEPTH\_BUFFER\_BIT);

Gl.glLoadIdentity();

Gl.glColor3f(0, 0, 0);

Gl.glPushMatrix();

Gl.glTranslated(ScreenW / 2, ScreenH / 2, 0);

Graph();

if (OA == AB)

Move\_var1();

else if (OA > AB)

Move\_var2();

else

Move\_var3();

Way();

Gl.glPopMatrix();

Gl.glFlush();

AnT.Invalidate();

}

private void button1\_Click(object sender, EventArgs e)

{

GraphValuesArray = new float[100000, 2];

if (trackBar3.Value > trackBar2.Value)

{

trackBar3.Value = trackBar2.Value;

}

elements\_count = 0;

Init(AnT);

new\_parametrs();

ScreenW = (OA + AB) \* 2.2f \* (float)AnT.Width / (float)AnT.Height;

ScreenH = (OA + AB) \* 2.2f;

xA = OA; yA = 0;

xM = OA + AM; yM = 0;

xB = OA + AB;

angle = 0;

check = 0;

not\_calculate = true;

elements\_count = 0;

Glu.gluOrtho2D(0.0, ScreenW, 0.0, ScreenH);

Gl.glMatrixMode(Gl.GL\_MODELVIEW);

timer1.Enabled = true;

}

private float Radian(float angle) {

return (float)((Math.PI / 180.0) \* angle);

}

private void Init(SimpleOpenGlControl x)

{

Gl.glViewport(0, 0, x.Width, x.Height);

Gl.glMatrixMode(Gl.GL\_PROJECTION);

Gl.glLoadIdentity();

Gl.glClear(Gl.GL\_COLOR\_BUFFER\_BIT | Gl.GL\_DEPTH\_BUFFER\_BIT);

Gl.glLoadIdentity();

Gl.glColor3f(0, 0, 0);

}

private void PrintText2D(float x, float y, string text)

{

Gl.glRasterPos2f(x, y);

foreach (char char\_for\_draw in text)

{

Glut.glutBitmapCharacter(Glut.GLUT\_BITMAP\_9\_BY\_15, char\_for\_draw);

}

}

private void Way()

{

if (not\_calculate && angle < 360)

{

GraphValuesArray[elements\_count, 0] = xM;

GraphValuesArray[elements\_count, 1] = yM;

elements\_count++;

}

else

not\_calculate = false;

if (checkBox1.Checked)

Draw\_Way();

}

private void timer1\_Tick(object sender, EventArgs e)

{

Draw();

timer1.Interval = trackBar4.Value;

}

}

}

**Результаты работы приложения**

