Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

**МЕХАНІКО-ТЕХНОЛОГІЧНИЙ ФАКУЛЬТЕТ**

Кафедра кібербезпеки та програмного забезпечення

**Звіт**

з виконаної лабораторної роботи № 3

Дисципліна: Комп'ютерна графіка

на тему

«Двовимірне векторне зображення. Списки елементів»

Виконав :

студент академічної групи КІ-15

Аннаєв А. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Перевірив :

Викладач

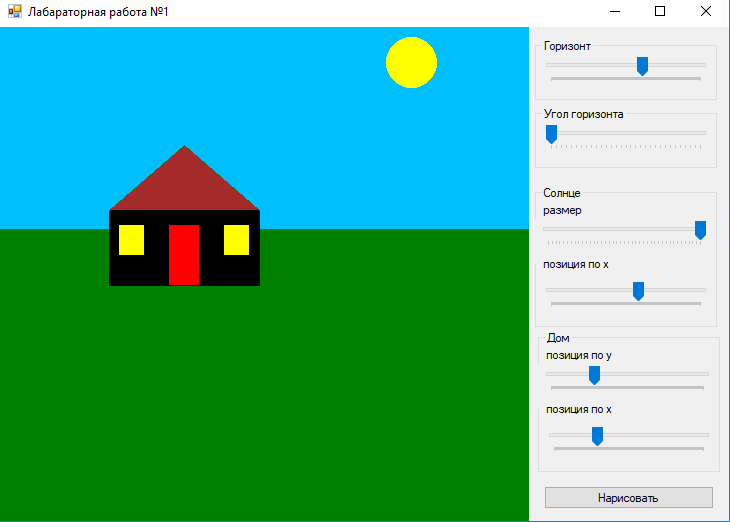
ЛадигiнаО.А.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

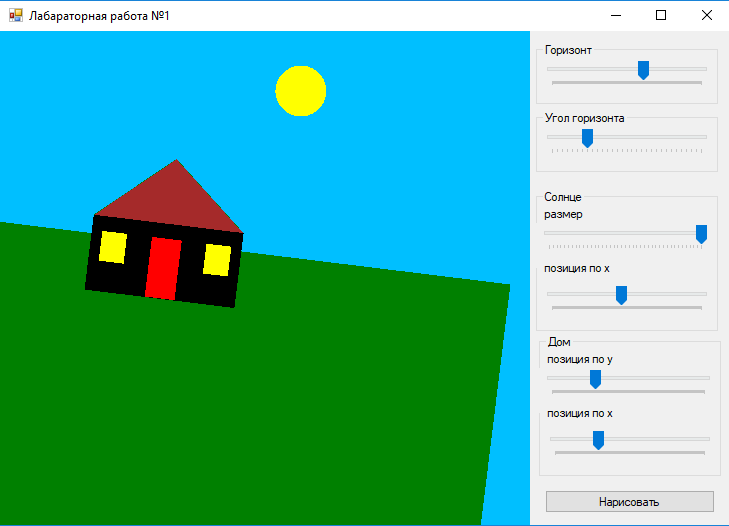
Кропивницкий- 2018

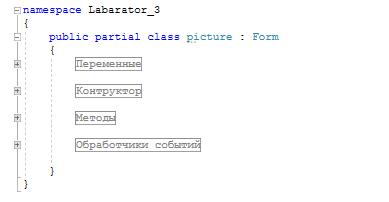
**Лабораторна робота №3**

**Тема**: Двовимірне векторне зображення. Списки елементів

**Мета**: Створити формат збереження векторного зображення. Додати рухомі елементи.





 #region Переменные

Graphics drawArea;

private int horizont;

private int height;

private int width;

private int curSunSize;

private int curSunPosition;

private int startX0 = 150;

private int startY0 = 150;

private int maxSunPosition;

private System.Timers.Timer timer;

#endregion

#region Контруктор

/// <summary>

/// Инициализация компонентов формы

/// </summary>

public picture() {

InitializeComponent();

Init();

}

#endregion

#region Методы

/// <summary>

/// Инициализация переменных

/// </summary>

private void Init() {

drawArea = drawingArea.CreateGraphics();

timer = new System.Timers.Timer(2000);

timer.Elapsed += OnTimerElapsed;

timer.Start();

height = drawingArea.Height;

width = drawingArea.Width;

horizontTrack.Minimum = 50;

horizontTrack.Maximum = height -50;

houseyTrack.Maximum = height - 100;

houseyTrack.Minimum = 100;

housexTrack.Maximum = width -150;

housexTrack.Minimum = 0;

maxSunPosition = width + 50;

sunPositionTrack.Minimum =-50;

sunPositionTrack.Maximum = maxSunPosition + 50;

sunSizeTrack.Minimum = 10;

sunSizeTrack.Maximum = 50;

}

/// <summary>

/// Сбор рисунков в правильной последовательности

/// </summary>

private void Compositor() {

try {

// прежде чем перерисовать очищаем

drawArea.Clear(Color.DeepSkyBlue);

DrawHorizon(drawArea);

DrawSun(drawArea);

DrawHouse(drawArea);

}

catch (Exception ex){

MessageBox.Show(ex.Message);

}

}

/// <summary>

/// Отрисовка горизонта

/// </summary>

private void DrawHorizon(Graphics area) {

using (SolidBrush rectBrush = new SolidBrush(Color.Green)) {

Rectangle rect = new Rectangle(0, (height - horizont), width, height);

area.DrawRectangle(new Pen(Color.Green, 3), rect);

area.FillRectangle(rectBrush, rect);

}

}

/// <summary>

/// Отрисовка солнца

/// </summary>

/// <param name="area"></param>

private void DrawSun(Graphics area) {

using (SolidBrush sunBrush = new SolidBrush(Color.Yellow)) {

Rectangle rect = new Rectangle(curSunPosition, 10, curSunSize, curSunSize);

area.DrawEllipse(new Pen(Color.Yellow), rect);

area.FillEllipse(sunBrush, rect);

}

}

/// <summary>

/// Отрисовка дома

/// </summary>

/// <param name="drawArea"></param>

private void DrawHouse(Graphics area) {

int houseWidth = 150;

int houseHeight = 75;

var winWidth = 25;

var winHeigth = 30;

// коробка дома

var houseRect = new Rectangle(startX0, startY0, houseWidth, houseHeight);

area.DrawRectangle(Pens.Black, houseRect);

area.FillRectangle(new SolidBrush(Color.Black), houseRect);

// левое окно

var windowLeftRect = new Rectangle(startX0 + 10, startY0 +15, winWidth, winHeigth);

area.DrawRectangle(Pens.Black, windowLeftRect);

area.FillRectangle(new SolidBrush(Color.Yellow), windowLeftRect);

// правое окно

var windwRigthRect = new Rectangle(startX0+ houseWidth - winWidth -10, startY0 + 15, winWidth, winHeigth);

area.DrawRectangle(Pens.Black, windwRigthRect);

area.FillRectangle(new SolidBrush(Color.Yellow), windwRigthRect);

// размеры двери

var doorWidth = 30;

var doorHeigth = 60;

var doorStartX0 = startX0 + (houseWidth / 2) - (doorWidth / 2);

// дверь

var doorRect = new Rectangle(doorStartX0, startY0 + 15, doorWidth, doorHeigth);

area.DrawRectangle(Pens.Black, doorRect);

area.FillRectangle(new SolidBrush(Color.Red), doorRect);

// точки для крыши

Point leftPoint = new Point(startX0, startY0);

Point rightPoint = new Point(startX0 + houseWidth, startY0);

// размеры и координаты крыши

var floorHeight = 65;

var middleX = startX0 + (houseWidth / 2);

var moddleY = startY0 - floorHeight;

Point middlePoint = new Point(middleX, moddleY);

// прорисовка по пути

GraphicsPath path = new GraphicsPath();

path.AddLine(leftPoint, middlePoint);

path.AddLine(middlePoint, rightPoint);

area.DrawPath(Pens.Green, path);

Brush br2 = new SolidBrush(Color.Brown);

area.FillPath(br2, path);

}

#endregion

#region Обработчики событий

/// <summary>

/// Нажатие кнопки отрисовки

/// </summary>

private void btnDraw\_Click(object sender, EventArgs e) {

Compositor();

}

/// <summary>

/// Размещение горизонта

/// </summary>

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

horizont = horizontTrack.Value;

Compositor();

}

/// <summary>

/// Размер солнца

/// </summary>

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

curSunSize = sunSizeTrack.Value;

Compositor();

}

/// <summary>

/// Положение солнца

/// </summary>

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

curSunPosition = sunPositionTrack.Value;

Compositor();

}

/// <summary>

/// Позиционирование дома по y

/// </summary>

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

startY0 = houseyTrack.Value;

Compositor();

}

/// <summary>

/// Позиционирование дома по x

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

startX0 = housexTrack.Value;

Compositor();

}

/// <summary>

/// Обработчик переполнения счетчика таймера

/// </summary>

private void OnTimerElapsed(object sender, ElapsedEventArgs e) {

if (maxSunPosition <= curSunPosition)

curSunPosition = 0;

else

curSunPosition += 10;

Compositor();

}

/// <summary>

/// Угол поворота рисунка горизонта

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void horizontRadTrack\_Scroll(object sender, EventArgs e)

{

corner = horizontRadTrack.Value;

Compositor();

}

#endregion