МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное образовательное учреждения высшего образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт компьютерных технологий и информационной безопасности

Кафедра «Математического обеспечения и применения ЭВМ»

|  |  |  |
| --- | --- | --- |
| 1002262_584459121576624_356098510_n |  | logo МОПЭВМ |
|  |  |  |

**Лабораторная работа №1**

по курсу

компьютерная графика

Вариант№ 6

Выполнили:

студент группы КТбо3-9

Митина А.Г.

Ячменев М.И.

Проверил:

Селянкин В.В.

Таганрог, 2016

**Постановка задачи**

Задание заключается в разработке программы создания динамического изображения. Изображение включает в себя простейшую фигуру, определяемую в соответствии с вариантом задания. Заданием определяются и другие параметры изображения. Динамическое изображение создается последовательными операциями вывода фигуры на экран, стиранием его и новым выводом изображения, смещенного относительно предыдущего вывода. Изображение перемещается по экрану под заданным углом. При достижении границы экрана происходит отражение фигуры по закону отражения луча света. В процессе движения фигуры размеры или цвет изображения могут изменяться. Заданием определяется и способ стирания фигуры

**Вариант задания**

Тип фигуры: 2, разносторонний треугольник

Изменение цвета

Угол = 31 \* 6 + 45 = 231

Типы стирания (1,3,6):

1. Очистка экрана
2. Вывод спрайта изображения
3. Стирание изображения цветом фона

**Описание алгоритма**

Для начала введем координаты нашего треугольника, и проверим, что бы они не выходили за границы окна. Это осуществлено в функции IsRightData();

Объявим карандаши для рисования треугольника.

Вычислим факторы, которые позволят не накапливать ошибку. Для этого вычислим разницу между координатами по x и координатами по y и запомним ее. Осуществим сдвиг фигуры, для этого к одной точке прибавим шаг:

А к оставшимся точкам прибавим вычисленные в начале факторы. Проверим не выходит ли какая-либо точка за край поля, если выходит, то изменяем цвет нашей палитры на следующий и осуществляем отражение от края поля по заданному углу отражения.

Диапазон изменения цвета фигуры таков, что просто не используется цвет фона и при смене цвета учитывать его не надо.

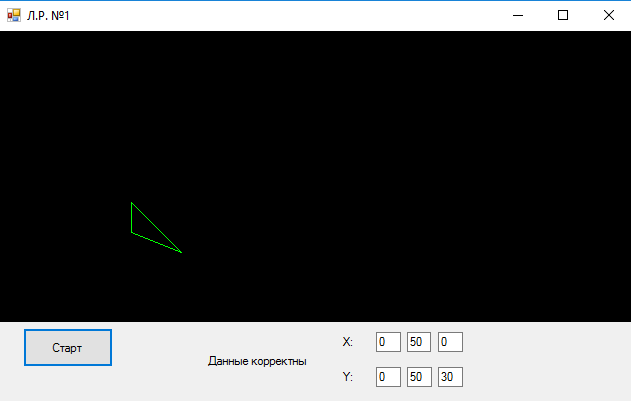
Нарисуем фигуру использую стандартные средства языка С# используя текущий цвет палитры, которые задается в конструкторе.

**Листинг**

1. using System;
2. using System.Collections.Generic;
3. using System.ComponentModel;
4. using System.Data;
5. using System.Drawing;
6. using System.Linq;
7. using System.Text;
8. using System.Threading.Tasks;
9. using System.Windows.Forms;


13. namespace WindowsFormsApplication1
14. {
16. public struct point
17. {
18. public float x;
19. public float y;
20. }
21. public partial class Form1 : Form
22. {
23. bool IsCorrectData = **false**;
24. int countPens;
25. int step;
26. double corner;
27. point[] points;
28. Graphics gr;       *//We declare an object - graphics, which will draw*
29. Pen[] p;            *//We declare an object - a pencil, which will draw the contour*
30. Pen penFon;
31. SolidBrush fon;    *//We declare an object - fill, to fill the background respectively*
32. point[] startPoints;
33. float firstXFactor = 0;
34. float firstYFactor = 0;
35. float secondXFactor = 0;
36. float secondYFactor = 0;
38. */\*\**
39. *\* Set Color for pen*
40. *\**
41. *\* @return void*
42. *\* \*/*
43. public void SetColorPen()
44. {
45. penFon = new Pen(Color.Black);
46. p = new Pen[3];
47. p[0] = new Pen(Color.Lime);
48. p[1] = new Pen(Color.Coral);
49. p[2] = new Pen(Color.FloralWhite);
50. }
52. */\*\**
53. *\* Construct*
54. *\**
55. *\* @return void*
56. *\* \*/*
57. public Form1()
58. {
59. countPens = 0;
60. InitializeComponent();
61. step = 7;
62. corner = 231;
64. startPoints = new point[3];
65. points = new point[3];
66. SetColorPen();
67. points = startPoints;
68. }
69. */\*\**
70. *\* Аngle calculation*
71. *\**
72. *\* @return double*
73. *\* \*/*
74. private double Corner(double cor)
75. {
76. return cor \* (Math.PI / 180);
77. }
78. */\*\**
79. *\* Draw a triangle*
80. *\**
81. *\* @return double*
82. *\* \*/*
83. private void DrawTriangleFon()
84. {
86. gr.DrawLine(penFon, points[0].x, points[0].y, points[1].x, points[1].y);
87. gr.DrawLine(penFon, points[0].x, points[0].y, points[2].x, points[2].y);
88. gr.DrawLine(penFon, points[1].x, points[1].y, points[2].x, points[2].y);
89. }
90. */\*\**
91. *\* Draw a triangle by color fon*
92. *\**
93. *\* @return double*
94. *\* \*/*
95. private void DrawTriangle()
96. {
97. gr.DrawLine(p[countPens % 3], points[0].x, points[0].y, points[1].x, points[1].y);
98. gr.DrawLine(p[countPens % 3], points[0].x, points[0].y, points[2].x, points[2].y);
99. gr.DrawLine(p[countPens % 3], points[1].x, points[1].y, points[2].x, points[2].y);
100. }
101. */\*\**
102. *\* Moving the pieces*
103. *\**
104. *\* @return void*
105. *\* \*/*
106. private void Shift()
107. {
108. point[] tempPoints = points;
109. int tempStep = step;
110. bool flag = **true**;
112. while (flag == **true**)
113. {
114. points[0].x = startPoints[0].x + step;
115. points[0].y = (startPoints[0].y + (float)Math.Tan(Corner(corner)) \* step);
116. points[1].x = points[0].x - firstXFactor;
117. points[1].y = points[0].y - firstYFactor;
118. points[2].x = points[0].x - secondXFactor;
119. points[2].y = points[0].y - secondYFactor;
120. flag = **false**;
121. for (int i = 0; i < 3; ++i)
122. {
123. if (points[i].x >= ClientSize.Width || points[i].x <= 0)
124. {
125. countPens = ++countPens % 3;
126. step \*= (-1);
127. corner \*= (-1);
128. flag = **true**;
129. **break**;
130. }
131. if (points[i].y >= ClientSize.Height - 92 || points[i].y <= 0)
132. {
133. countPens = ++countPens % 3;
134. corner \*= (-1);
135. flag = **true**;
136. **break**;
137. }
138. }
139. }
141. startPoints = tempPoints;
142. }
144. */\*\**
145. *\* Button click*
146. *\**
147. *\* @return void*
148. *\* \*/*
149. private void button1\_Click(object sender, EventArgs e)
150. {
151. if (!IsRightData())
152. {
153. AnswerLabel.Text = "Неверные данные";
154. IsCorrectData = **false**;
155. }
156. else
157. {
158. IsCorrectData = **true**;
159. AnswerLabel.Text = "Данные корректны";
160. firstXFactor = startPoints[0].x - startPoints[1].x;
161. firstYFactor = startPoints[0].y - startPoints[1].y;
162. secondXFactor = startPoints[0].x - startPoints[2].x;
163. secondYFactor = startPoints[0].y - startPoints[2].y;
165. gr = pictureBox1.CreateGraphics();  *//initialize an object of type Graphics*
166. *// tied to a PictureBox*
168. fon = new SolidBrush(Color.Black);
169. gr.FillRectangle(fon, 0, 0, pictureBox1.Width, pictureBox1.Height);
170. timer1.Enabled = **true**;
171. }

174. }
175. */\*\**
176. *\* Timer tick*
177. *\**
178. *\* @return void*
179. *\* \*/*
180. private void timer1\_Tick(object sender, EventArgs e)
181. {
182. if (IsCorrectData)
183. {
184. *//сначала будем очищать область рисования цветом фона(1)*
185. *//  gr.FillRectangle(fon, 0, 0, pictureBox1.Width, pictureBox1.Height);*
187. *// очищение фона(2)*
188. *// gr.Clear(Color.Black);*
190. *// очищение рисованием цвета фона(3)*
191. DrawTriangleFon();
193. Shift();
194. DrawTriangle();
195. };
196. }
197. */\*\**
198. *\* Check for correct points*
199. *\**
200. *\* @return bool*
201. *\* \*/*
202. private bool IsRightData()
203. {
204. int i = 0;
205. int j = 1;
206. foreach (TextBox c in Controls.OfType<TextBox>())
207. {
208. int result;
209. if (String.IsNullOrEmpty(c.Text) || !Int32.TryParse(c.Text, out result))
210. {
211. return **false**;
212. }
213. else
214. {
215. if (j % 2 == 1)
216. {
217. if (Int32.Parse(c.Text) < -9 || Int32.Parse(c.Text) >= 501)
218. return **false**;
219. else
220. startPoints[i].x = Int32.Parse(c.Text);
221. }
222. else
223. {
224. if (Int32.Parse(c.Text) < -2 || Int32.Parse(c.Text) >= 250)
225. return **false**;
226. else
227. startPoints[i].y = Int32.Parse(c.Text);
228. ++i;
229. }
230. ++j;
231. }
232. }
233. return **true**;
234. }
235. }
236. }

****