Белгородский Государственный Технологический Университет им. В. Г. Шухова

Кафедра программного обеспечения вычислительной техники  
и автоматизированных систем

## Лабораторная работа №1 по теме: «Графические примитивы GDI»

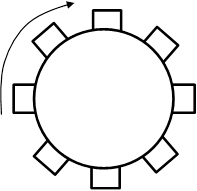
**Выполнил:**  
студент группы ПВ-41  
Адаменко И. И.

**Проверил:**  
старший преподаватель  
Осипов О. В.

Белгород  
2015

**Цель работы:** изучение графических 2D-примитивов с использованием GDI в среде Microsoft Visual Studio.

## Задание для варианта №1

Разработать алгоритм и составить программу для построения на экране следующего изображения:

Реализовать вращение фигуры и возможность установки количества зубцов во время исполнения программы.

# Ход работы

## Алгоритм работы

Сперва рассчитаем максимально возможный размер области отрисовки. Поскольку в центре фигуры окружность, то область отрисовки должна быть квадратом. Таким образом, размер области должен быть равен меньшему из значений длины и ширины окна.

Пусть размер стороны области отрисовки равен , тогда (т. к. по условию фигура должна иметь отступ в 10 пикселей от сторон окна) радиус вписанной в неё окружности равен: .

В задании точно не указано, но допустим, что ситуация, в которой зубцы на фигуре будут пересекаться друг с другом — недопустима. В таком случае, нам необходимо проверять их размер и, если это необходимо, уменьшать его. Для проверки достаточно сравнить длину будущей окружности (изображённой на фигуре) и периметр вписанного в неё многоугольника, с длиной стороны равной ширине зубца. Поскольку мы занимаем всю область отрисовки, то радиус окружности, изображённой на фигуре будет равен , где — ширина зубца.

Таким образом, если зубцов размера помещаются на окружности радиуса , то неравенство верно.

Допустим мы также желаем оставить между зубцами расстояние в h пикселей. Тогда условно формула изменится следующим образом: .

Поскольку зубцы перекрываются окружностью, то можно представить их равными прямоугольниками, у которых середина одной из сторон совпадает с центром окружности. Таким образом, мы можем поворачивать их относительно этой точки и получать новые зубцы на фигуре. Значит, если у нас есть определённый зубец с номером , то угол его поворота будет равен: , или, если рассматривать фигуру в какой-то определённый момент её вращения, в который она повёрнута на градусов относительно начального положения, то угол поворота зубца будет равен: .

## Исходный код программы

1. **using** System;
2. **using** System.Drawing;
3. **using** System.Windows.Forms;
4. **using** System.Drawing.Drawing2D;
6. **namespace** lab1
7. {
8. **public** partial **class** Main : Form
9. {
10. // угол поворота всех квадратов
11. **float** angle = 0;
12. // количество квадратов на экране
13. **int** squaresNum = 1;
15. **public** Main()
16. {
17. InitializeComponent();
19. **this**.SetStyle(
20. // вкл. двойную буферизацию == боремся с мерцанием
21. ControlStyles.DoubleBuffer |
22. // контрол отрисовывает себя сам, вместо ОС
23. ControlStyles.UserPaint |
24. // игнорирование события WM\_ERASEBKGND для уменьшения мерцания
25. ControlStyles.AllPaintingInWmPaint |
26. // перерисовывание при ресайзе окна
27. ControlStyles.ResizeRedraw,
28. **true**
29. );
30. **this**.UpdateStyles();
31. }
33. **protected** **override** **void** OnPaint(PaintEventArgs e)
34. {
35. Graphics g = e.Graphics;
36. RectangleF area = g.VisibleClipBounds;
38. // вычисляем максимально возможный радиус вписанной в area окружности
39. **float** d = (area.Width > area.Height) ? area.Height : area.Width;
40. // по условию: отступ от краёв по 10 пикселей
41. **float** radius = (d - 20) / 2;
43. // координаты центра окружности
44. var center = **new** PointF(area.Width / 2, area.Height / 2);
46. // задаём начальный размер квадратов, равный чуть меньше,  
     // чем половине диаметра
47. **int** squareSize = (**int**)radius / 2 - 1;
49. // уменьшаем размер квадратов, если они не влезают
50. **while** (! IsFit(squareSize, radius))
51. {
52. squareSize--;
53. }
55. // если окно или размер квадратов совсем уж маленькие,  
     // то ничего не выводить
56. **if** (radius < squareSize \* 2 || squareSize < 1) **return**;
58. // вычисляем реальные радиус и диаметр окружности
59. **float** roundRadius = (radius - squareSize);
60. **float** roundDiameter = roundRadius \* 2;
62. // находим область отрисовки круга
63. var roundArea = **new** RectangleF(
64. center.X - roundRadius,
65. center.Y - roundRadius,
66. roundDiameter,
67. roundDiameter
68. );
70. // отрисовываем фон, чтобы было видно отступы
71. var bg = **new** RectangleF(
72. center.X - radius,
73. center.Y - radius,
74. radius \* 2,
75. radius \* 2
76. );
78. g.FillRectangle(Brushes.Beige, bg);
80. // зададим левую верхнюю точку квадрата
81. var squareStartF = **new** PointF(center.X - squareSize / 2,  
     center.Y - radius);
82. // зададим размеры квадрата (на самом деле прямоугольника)
83. var squareSizeF = **new** SizeF(squareSize, radius);
84. // зададим область отрисовки квадрата
85. var square = **new** RectangleF(squareStartF, squareSizeF);
87. // инициализируем матрицу трансформаций
88. var m = **new** Matrix();
90. **for** (**int** i = 0; i < squaresNum; i++)
91. {
92. // поворачиваем область отрисовки относительно центра
93. m.RotateAt(i \* 360 / squaresNum + angle, center);
94. g.Transform = m;
96. // отрисовываем квадрат
97. g.FillRectangle(Brushes.Maroon, square);
99. // сбрасываем матрицу трансформации и поворот области отрисовки
100. m.Reset();
101. g.ResetTransform();
102. }
104. // отрисовываем сам круг
105. g.FillEllipse(Brushes.BurlyWood, roundArea);
107. **base**.OnPaint(e);
108. }
110. **private** **bool** IsFit(**int** squareSize, **float** radius)
111. {
112. // расстояние между квадратами
113. var spaceBetween = 5;
115. **int** p = squareSize \* squaresNum + spaceBetween \* (squaresNum + 1);
116. **double** l = 2 \* Math.PI \* (radius - squareSize);
118. **return** p <= l;
119. }
121. **private** **void** timer\_Tick(**object** sender, EventArgs e)
122. {
123. angle += 6;
125. // перерисовываем
126. Invalidate();
127. }
129. **private** **void** squaresNumUpDown\_ValueChanged(**object** sender, EventArgs e)
130. {
131. squaresNum = Convert.ToInt32((sender **as** NumericUpDown).Value);
132. }
133. }
134. }

## Пример работы программы

