Содержание

|  |  |  |
| --- | --- | --- |
| 1 | Эскиз прогрессбара и анимации |  |
| 2 | Прогрессбар |  |
| 2.1 | Формулы для расчета параметров прогрессбара |  |
| 2.2 | Используемые типы данных для реализации прогрессбара |  |
| 2.3 | Результат работы прогрессбара |  |
| 2.4 | Исходный код прогрессбара |  |
| 3 | Анимация |  |
| 3.1 | Формулы для расчета параметров анимации |  |
| 3.2 | Используемые типы данных для реализации анимации |  |
| 3.3 | Результат работы анимации |  |
| 3.4 | Исходный код анимации |  |

1 Эскизы прогресс бара и анимации



Рис.1(Эскиз прогресс бара)

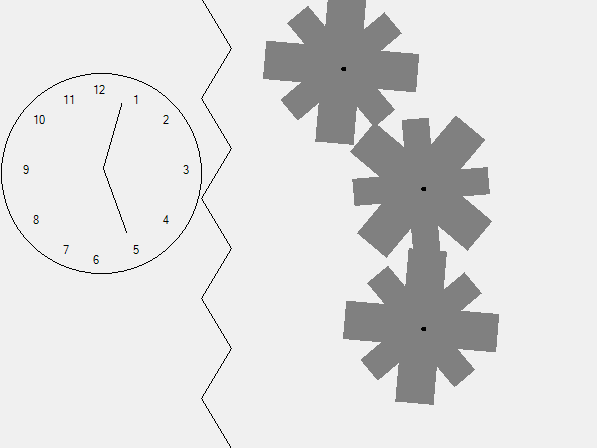


Рис.2(Эскиз анимации)

2 Прогрессбар

2.1 Формулы для расчетов параметров прогрессбара

Чтобы нарисовать прогрессбар я использовал готовые картинки которые отрисовывал с помощью класса Grafics на форме. Для отображения процентов в центре прогресс бара я использовал формулу нахождения 1 процента от текущего максимума (процент = (текущее число\*100) / максимум) ). Таким образом подставляя в максимум любые числа можно получить адекватное распределение процентов в зависимости от текущего значения.

2.2 Используемые типы данных для реализации прогрессбара

Для реализации пользовательского управления прогрессбаром, нужно на форму добавить несколько полей, поле с минимальным значением прогрессбара, поле с максимальным значением прогрессбара, поле с текущем значением прогрессбара и поле с выбором остановки прогрессбара.   
В самом коде необходимо создать несколько поле типа float. maxValue- максимальное значение прогрессбара, minValue-минимальное значение прогрессбара, tecValue- текущее значение прогрессбара. Все эти поля служат для хранения задаваемых пользователем значений из полей на форме.

В методе DrawBar вызывается метод DrawProgressBar который рисует прогрессбар.

2.3 Результат работы прогрессбара

В полях минимальное и максимальное значение зададим числа 0 и 100. В поле текущее значение пользователь может задать любое число, в случае если оно будет превышать максимальное или будет меньше минимально то соответственно текущему значению будет присваиваться либо максимальное если превышает либо минимальное если меньше.

Зададим текущее значение 35.

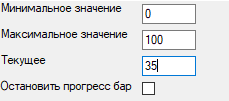


Рис 2.1(Изменение текущего значения на 35)

В прогресс баре соответственно отобразится нужное нам количество делений, точек снизу и цифра в процентах.

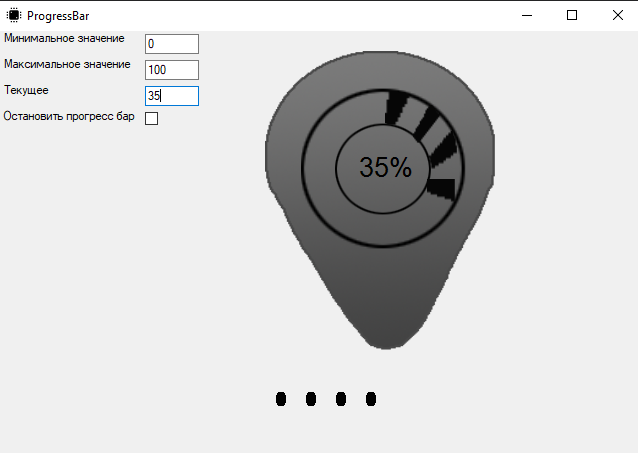


Рис 2.2(результат изменения текущего значения)

Так при нажатии на поле остановить прогресс бар, прогресс бар перестанет увеличивать текущее значение и просто остановится.

2.4 Исходный код прогрессбара

Файл ProgressBar.cs

using System;

using System.Drawing;

using System.Windows.Forms;

namespace TrushkovPractica

{

public partial class ProgressBar : Form

{

ProgreeBarMy prb = new ProgreeBarMy();

Timer tm = new Timer();

public int temptNumber;

public ProgressBar()

{

InitializeComponent();

tm.Interval = 100;

tm.Tick += UpdateForm;

}

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

{

tm.Start();

}

private void pictureBox1\_Click(object sender, EventArgs e)

{

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

try

{

int value = Convert.ToInt32(minTextBox.Text);

prb.minValue = value;

}

catch (Exception er)

{

int value = 0;

prb.minValue = value;

}

}

private void textBox2\_TextChanged(object sender, EventArgs e)

{

try

{

int value = Convert.ToInt32(maxTextBox.Text);

prb.maxValue = value;

}

catch(Exception er)

{

int value = 0;

prb.maxValue = value;

}

}

private void ChangeValue(object sender, EventArgs e)

{

try

{

int value = Convert.ToInt32(valueTecTextBox.Text);

prb.tecValue = value;

}

catch (Exception er)

{

}

}

private void UpdateForm(object sender, EventArgs e)

{

pictureBox1.Invalidate();

}

private void StopOrNot(object sender, EventArgs e)

{

if (stopCheckBox.Checked == true)

{

tm.Stop();

}

else

{

tm.Start();

}

}

private void DrawBar(object sender, PaintEventArgs e)

{

prb.DrawProgressBar(20, 20, 230, 310, e.Graphics);

}

}

public class ProgreeBarMy

{

public int tecValue;

public int minValue;

public int maxValue;

public Bitmap[] indick = new Bitmap[14];

int tec;

public int TecValue

{

get

{

return TecValue;

}

set

{

tecValue = value;

}

}

public void DrawProgressBar(int x, int y, int widht, int height, Graphics e)

{

tec++;

for (int i = 1; i <= 13; i++) {

indick[i - 1] = new Bitmap(@"C:\Users\Kangarion\Desktop\TrushkovPractica\Image\ProgressBar" + i + ".png");

}

RectangleF rec2 = new RectangleF(x \* 3, y \* 3.1f, tecValue \* 1.5f, tecValue \* 1.4f);

RectangleF rec = new RectangleF(0, 0, tecValue, tecValue);

GraphicsUnit units = GraphicsUnit.Pixel;

Bitmap border = new Bitmap(@"C:\Users\Kangarion\Desktop\TrushkovPractica\Image\BorderBar.png");

e.DrawImage(border, x, y, widht, height);

Bitmap cirle = new Bitmap(@"C:\Users\Kangarion\Desktop\TrushkovPractica\Image\Circle.png");

e.DrawImage(cirle, x \* 2.8f, y \* 2.5f, widht / 1.4f, height / 1.85f);

e.DrawImage(cirle, x \* 4.5f, y \* 4.4f, widht / 2.40f, height / 3.25f);

int procent = 0;

if (maxValue > 0)

{

procent = (tecValue \* 100) / maxValue;

}

e.DrawString(procent + "%", new Font("Arial", 20), Brushes.Black, x \* 6 - 10, y \* 6);

int temp = 100 /12;

#region[indicator]

if (procent > temp\*1)

{

e.DrawImage(indick[0], x \* 7, y \* 3, widht / 9, height / 9);

e.FillEllipse(Brushes.Black, x\*1.5f, y\*18, widht/20, height/20);

}

if (procent > temp\*2)

{

e.DrawImage(indick[1], x \* 8.4f, y \* 3.7f, widht / 8, height / 8);

e.FillEllipse(Brushes.Black, x\*3, y \* 18, widht / 20, height / 20);

}

if (procent > temp\*3)

{

e.DrawImage(indick[2], x \* 9.2f, y \* 5, widht / 8, height / 8);

e.FillEllipse(Brushes.Black, x\*4.5f, y \* 18, widht / 20, height / 20);

} if (procent > temp\*4)

{

e.DrawImage(indick[3], x \* 9.1f, y \* 7.4f, widht / 8, height / 13);

e.FillEllipse(Brushes.Black, x\*6, y \* 18, widht / 20, height / 20);

} if (procent > temp\*5)

{

e.DrawImage(indick[4], x \* 8.5f, y \* 8.5f, widht / 8, height / 10);

e.FillEllipse(Brushes.Black, x\*7.5f, y \* 18, widht / 20, height / 20);

} if (procent > temp\*6)

{

e.DrawImage(indick[5], x \* 7f, y \* 9, widht / 8, height / 10);

e.FillEllipse(Brushes.Black, x\*9, y \* 18, widht / 20, height / 20);

} if (procent > temp\*7)

{

e.DrawImage(indick[6], x \* 5.5f, y \* 9, widht / 12, height / 10);

e.FillEllipse(Brushes.Black, x\*10.5f, y \* 18, widht / 20, height / 20);

}

if (procent > temp\*8)

{

e.DrawImage(indick[7], x \* 3.8f, y \* 8, widht / 10, height / 10);

e.FillEllipse(Brushes.Black, x\*12, y \* 18, widht / 20, height / 20);

} if (procent > temp\*9)

{

e.DrawImage(indick[9], x \* 3f, y \* 6.7f, widht / 8, height / 13);

} if (procent > temp\*10)

{

e.DrawImage(indick[10], x \* 3.1f, y \* 5f, widht / 7, height / 13);

e.FillEllipse(Brushes.Black, x \* 13.5f, y \* 18, widht / 20, height / 20);

}

if (procent > temp\*11)

{

e.DrawImage(indick[11], x \* 3.8f, y \* 3.8f, widht / 8, height / 9);

}

if (procent > temp\*12)

{

e.DrawImage(indick[12], x \* 5.1f, y \*3.3f, widht / 8, height / 10);

e.FillEllipse(Brushes.Black, x \* 15, y \* 18, widht / 20, height / 20);

}

#endregion

}

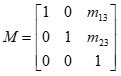
}

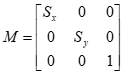
}

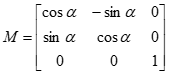
3 Анимация

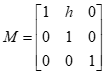
3.1 Формулы для расчетов параметров анимации

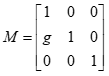
При создании анимации объектов на форме можно столкнуться с аффинными преобразованиями: перемещение, масштабирование относительно начала координат, поворот на заданный угол относительно начала координат, сдвиг по осям координат. Их с лёгкостью можно представить с помощью матрицы преобразований (обозначим её буквой M). Рассмотрим каждое из преобразований:

Перемещение: **,** где *m*13 – перемещение по оси *X*, а *m*23 – перемещение по оси *Y*.

Масштабирование относительно начала отсчета: , где *Sx* – масштабирующий коэффициент по оси *X*, а *Sy* – масштабирующий коэффициент по оси *Y*.

Поворот на заданный угол α относительно началакоординат: 

Сдвиг по оси *X:* , где h – величина сдвига по оси X.

Сдвиг по оси *Y:* *, где g – величина сдвига по оси.*

Для того, чтобы определить экранные координаты каждого объекта анимации, с определёнными параметрами при отображении я определил последовательность выполнения элементарных преобразований на плоскости для объектов:

1. Перенести центр объекта в начало координат;

2. Задать размеры объекта;

3. Повернуть объект относительно начала координат;

4. Перенести центр объекта в заданную точку (x, y);

5. Преобразовать координаты объекта из координат сцены в экранные координаты.

3.2 Используемые типы данных для реализации анимации

Для реализации анимации объекта было создано несколько классов в файле AnimationForm.cs. Класс Transformation - класс реализующий аффинные преобразования. Класс DrawObjects - класс реализующий рисование объектов на форме. В классе Transfomation реализованы 2 поля localCoordinates -координаты единичного объекта и sceneCoord - преобразованные координаты единичного объекта, это списки с типом данных PointF. Так же был создан файл Gear.cs в методах этого класса заданы координаты объектов в единичном квадрате. СlovePointReturn - координаты шестерни. Clock- координаты часов. Strelka - координаты стрелки часов. Все эти методы возвращают списки с типом данных PointF.

В классе Transfomation есть метод Number, данный метод выполняет аффинные преобразования. Матрица M0 - принимает координаты единичного объекта. Матрица М1 - принимает координаты центра объекта в единичном квадрате предварительно высчитав их по формуле: .

Матрица М2 - принимает высоту и ширину объекта которые задаются заранее. Матрица М3 принимает угол поворота объекта предварительно переведя градусы в радианы: , и найдя косинус и синус: .

Матрица М4 принимает координаты на сцене ,которые задаются предварительно.

Матрица М5 принимает параметры преобразования координат сцены в экранный, предварительно высчитывая соотношение сторон экрана:

Идет вставка изображения....

Дальше вычисляем параметры преобразования для х:

.

Для у:

.

В конце метода просто перемножаем все матрицы получая нужные нам значения.

В классе DrawObjects есть 4 метода. DrawGear - получает преобразованные точки шестерни и рисует ее на форме. DrawLine - рисует разделительную линию. DrawClock - получает преобразованные точки часов и рисует часы с циферблатом. DrawStrelka -получает преобразованные точки стрелки и рисует стрелки.

В классе AnimationForm инициализируется классы Transformation 4 раза для каждого объекта на форме, Gear для получения точек единичного объекта, Timer- для реализации метода который будет вызываться с определенным интервалом, DrawObjects - для отрисовки объектов.

## 3.3 Результаты работы анимации

В поля с