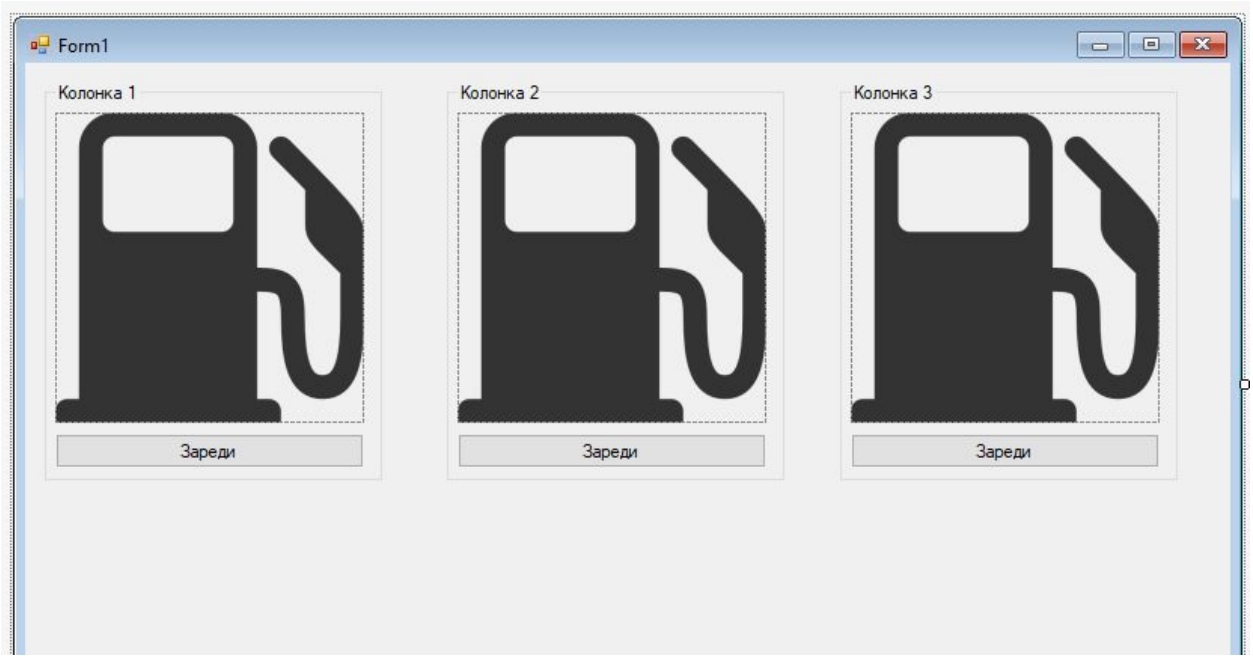
В настоящето упражнение ще направим малък визуален графичен интерфейс, който ще ни послужи за проект, който ще реализираме на по-късен етап.

Ще създадем Windows Forms приложение, в което ще имаме колонки за гориво. При активиране на зареждането ще анимиране колонката, така че тя да се запълва постепенно. При запълване можем да покажем съобщение, че колонката е готова или пък да рестартираме анимацията.

Искаме да имаме възможност да анимираме колонките едновременно, поради което всъщност ще трябва да използваме възможностите на многонишковото програмиране. Една особеност на Windows Forms Application-ите е, че те използват една обща нишка (UI thread) за визуалния интерфейс, което прави анимирането на даден обект в рамките на кода трудна задача. За щастие можем да използваме таймер компонента, но за това по-надолу.

Ще започнем със създаването на проекта и добавянето на съответните контроли. Тук ще използваме GroupBox контрола, който е характерен с това, че може да съдържа в себе си компоненти, което да направи лесно копирането/преместването на няколко компонента. Това ще ни е нужно по-късно в проекта.

Създайте Form, който да изглежда по аналогичен начин:



За да вмъкнете картинка, използвайте PictureBox. Картинката е качена като ресурс в електронното училище.

Сега, нека да преминем към следващия етап.

Ще направим така, че да анимираме картинката при натискане на бутона “Зареди”.

За да се справим с това, така че да не създадем проблем, свързан с факта, че нишката за графичния потребителски интерфейс е една ще трябва да използваме подходящи компоненти. За целта от Toolbox добавете 3 таймер компонента - всеки ще отговоаря за една от колонките.

Забележете, че таймерът не е като другите компоненти - той не се визуализира на самия форм а е “невидим”.

Visual Studio показва такъв тип компоненти на форма в долната лента под него:

Сега нека да преминем към писането на код.

Натиснете F7.

Ще създадем нов метод **Image RecolorImage(PictureBox pictureBox, double percent)**. Този метод ще има за задача да оцвети част от изображението - според параметъра percent.

Ще извикваме този метод последователно с различни проценти, за да постигнем желания от нас анимиран ефект.

Методът ще изглежда както следва:

```
private Image RecolorImage(PictureBox pictureBox, double percentAnimation)
{
    Bitmap original = new Bitmap(pictureBox.Image, pictureBox.Width, pictureBox.Height);
    using (Graphics g = Graphics.FromImage(original))
    {
        //Color Remap table - използваме това за да променим цвета
        ImageAttributes imageAttributes = CreateRecolorTable();
        Rectangle coloredRectangle = CreateColoredRectangle(pictureBox, percentAnimation);
        Bitmap croppedImage = this.Crop(original, coloredRectangle);

        g.DrawImage(
            croppedImage,
            coloredRectangle, //правоъгълника, който ще бъде нарисуван
            0, 0, //начална точка на рисуването
            coloredRectangle.Width, coloredRectangle.Height, //размер на нарисувания правоъгълник
            GraphicsUnit.Pixel,
            imageAttributes
        );

        return original;
    }
}
```

Нека да разгледаме кода:
Първо ще създадем обект от класа Bitmap - този обект ще се създаде на базата на оригиналното изображение с точно тези размери, с които е и picturebox-а, който ще ползваме.
След това ще ползваме обект от класа Graphics, който ще ни позволи да “рисуваме” върху оригиналното изображение.

Следващата стъпка е да повикаме метода CreateRecolorTable(); който ще връща таблицата за замяна на цветовете - в тази таблица се съдържа кода на стария цвят и кода на новия цвят. Този метод ще бъде описан малко по-надолу.

После създаваме правоъгълника, който ще представя частта от изображението, която ще бъде оцветена. Този правоъгълник ще има различни координати и размери, според желания процент на анимация.

Накрая изрязваме частта от оригиналното изображение, която искаме да оцветим. Ще използваме тази отрязка за метода за рисуване на изображение, където ще сменим цвета.

Накрая, извикваме метода DrawImage от Graphics. Този метод ще “нарисува” върху оригиналното изображение подаденото отрязано изображение, като отрязаното изображение ще бъде нарисувано в рамките на правоъгълника, който ще бъде оцветен. Оцветяването се случва на базата на imageAttributes обекта, който се подава към метода за рисуване. Към метода се подават и малко допълнителни параметри - началните

координати на рисуването, размерите на това, което предстои да нарисуваме, както и мерната единица, в която са зададени размерите.

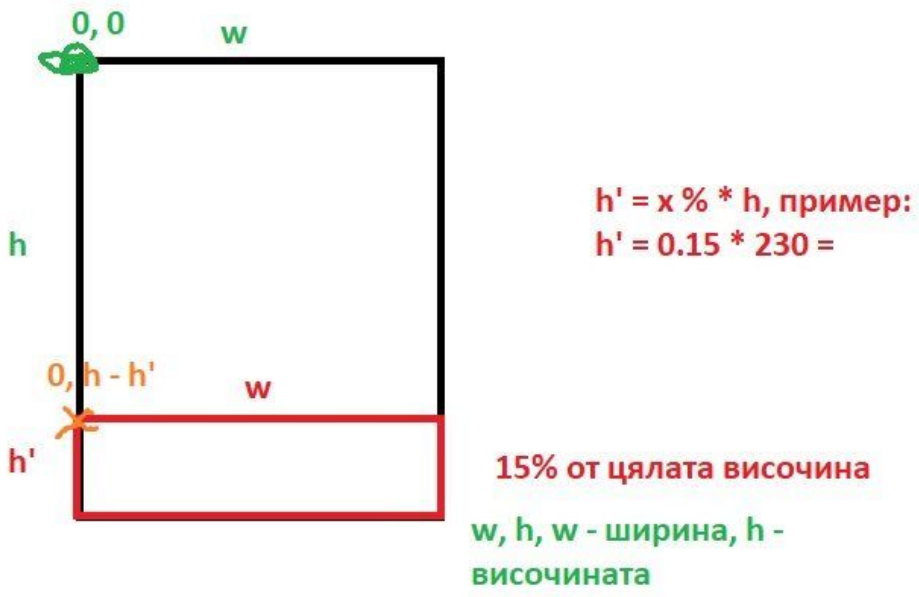
Сега нека да разгледаме метода за създаването на таблицата за смяна на цвета:

```
private static ImageAttributes CreateRecolorTable()
{
    ImageAttributes imageAttributes = new ImageAttributes();
    ColorMap colorMap = new ColorMap();
    colorMap.OldColor = Color.FromArgb(51, 51, 51);
    colorMap.NewColor = Color.FromArgb(0, 255, 0);
    ColorMap[] remapTable = { colorMap };

    imageAttributes.SetRemapTable(remapTable, ColorAdjustType.Bitmap);
    return imageAttributes;
}
```

Таблицата за смяна на цвета е базирана на класа ColorMap, а самият ColorMap се предава като параметър в рамките на атрибутите на изображението. Така по време на изрисуване на изображението, методът ще се съобрази с тези параметри и ще смени цвета.

В рамките на следващия метод трябва да изчислим и да създадем правоъгълника, който ще представя частта от изображението, която ще бъде оцветена



В примерната картинка по-горе в черния правоъгълник се намира цялостното изображение, а червеното представлява частта, която бихме оцветили. Тази част по отношение на височината ще е даден процент - този процент идва като параметър на метода и при всяко извикване може да бъде различен. За пример, нека да е 15%.

Идеята е, че трябва да изчислим височината на оцветения (червения) правоъгълник.

Това става като изчислим процента от оригиналната височина. Ширината се запазва същата. Това са и размерите на правоъгълника. Тъй като искаме анимацията да е такава, че колонката да се оцвети отдолу нагоре, то трябва да съобразим, че координатите на правоъгълника трябва да са такива, че да отидат долу. В този случай горния ляв ъгъл е с координати (0, 0) - ***не се бъркайте с Декартовата координатна система. Тук координатната система е различна!***

Координатите на горната лява точка на правоъгълника ще бъдат (0, h - h'), където h е оригиналната височина на кутийката с изорбажението, а h' е височината на оцветеното изображение.

Сега нека напишем всичко това като код:

```
1 reference
private static Rectangle CreateColoredRectangle(PictureBox pictureBox, double percentAnimation)
{
    int heightAnimation = (int)(percentAnimation * pictureBox.Height);

    Rectangle coloredRectangle = new Rectangle(
        new Point(0, pictureBox.Height - heightAnimation),
        new Size(pictureBox.Width, heightAnimation)
    );
    return coloredRectangle;
}
```

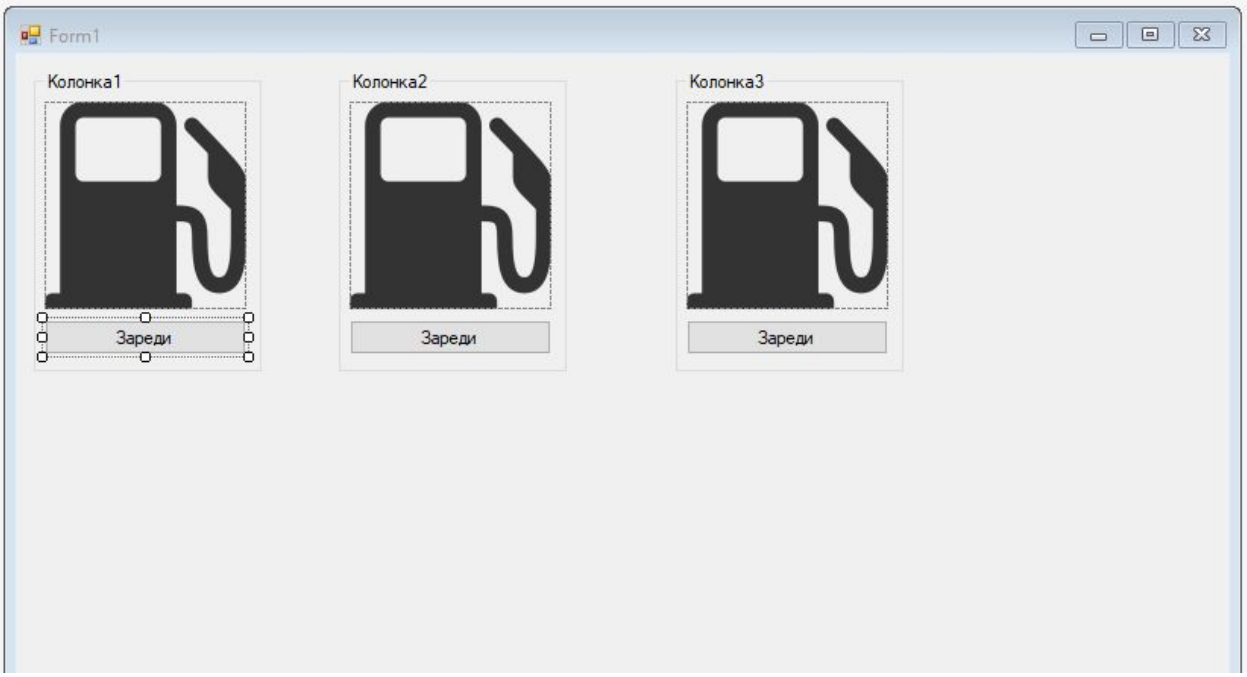
Доста малко код за толкова дълго обяснение, а? :)

Нека да реализираме и метода за отрязването:
Неговата логика е следната:
Той ще създаде рамка с размерите на новия оцветен правоъгълник.

В тази рамка трябва да остане само част от изображението. Можем да постигнем това по доста хитър начин. Ще нарисуваме оригиналното изображение в тази рамка, но задавайки координати отрицателни числа. Така ще "издърпаме" наляво и нагоре частта от изображението, която не трябва да влиза в рамката.

```
private Bitmap Crop(Bitmap original, Rectangle coloredRectangle)
{
    Bitmap cropped = new Bitmap(coloredRectangle.Width, coloredRectangle.Height);
    using (Graphics g = Graphics.FromImage(cropped))
    {
        g.DrawImage(original, -coloredRectangle.X, -coloredRectangle.Y);
        return cropped;
    }
}
```

Сега след като имаме всичко това, трябва само да викаме методите по подходящия начин. Нека да се върнем на form-а (Shift + F7). Цъкнете два пъти върху бутона “Зареди”.



В рамките на генерирания метод трябва да активираме таймера и да зададем метод, който да се изпълнява при всяко изтичане на таймера (например на всяка 1 секунда):

```
1 reference
private void btnLoad_Click(object sender, EventArgs e)
{
    timer1.Tick += Timer1_Tick;
    timer1.Interval = 1000;
    timer1.Start();
}
```

Преди да преминете нататък ще се нуждаем от две полета в класа:

```
4 references
public partial class Form1 : Form
{
    private double animationPercent1 = 0.05;
    private Image originalPumpImage = null;
```

Полето animationPercent1 ще показва в даден момент до какъв процент е достигнала анимацията за картинката на първата колонка. Такива полета ще създадем в последствие и за всяка от другите колонки.
Полето originalPumpImage трябва да съдържа оригиналната картинка. Ще искаме да я възвърнем след приключването на анимацията (зареждането). Задайте стойност за нея в рамките на конструктора:

```
public Form1()
{
    InitializeComponent();
    originalPumpImage = pictureBox1.Image;
}
```

Сега нека да се върнем към метода, който по-рано бе генериран за изпълнение на код при кликането върху бутона. В него се вика друг метод Timer1_Tick. Това е методът, който ще се изпълнява на всяка 1 секунда от таймера. Използвайте Visual Studio, за да генерирате този метод.

```
private void Timer1_Tick(object sender, EventArgs e)
{
    pictureBox1.Image = RecolorImage(pictureBox1, animationPercent1);
    if (animationPercent1 >= 1)
    {
        timer1.Stop();
        animationPercent1 = 0.05;
        pictureBox1.Image = originalPumpImage;
        MessageBox.Show("Готово - колонка 1!");
    }

    animationPercent1 += 0.05;
}
```

В рамките на този метод ще се вика RecolorImage метода. Освен това ще се увеличава стойността на animationPercent1 с 0.05 на всяка секунда. При достигане на стойност >= 1 таймерът ще се спира, при което променливата ще получи стойност 0.05, а в picturebox-а ще поставим оригиналното изображение. Накрая извеждаме съобщение, че зареждането е готово. Като алтернатива тука може да се възобнови анимацията - за целта не викайте timer1.Stop(), а за animationPercent1 задайте стойност 0. Не извеждайте и съобщение от MessageBox-а.

По подобен начин свържете и другите две колонки.