Практическая работа №20_1

Тема: Разработка приложений с анимацией

Цель работы: изучение приемов создания приложений Windows Forms с использованием графических методов, создания простейшей анимации движения

Задачи:

– изучение приемов создания приложений Windows Forms с графическими объектами;

– создание проектов с использованием графических методов, анимации.

Материально-техническое обеспечение:

Место проведения: Компьютерный класс.

Время на выполнение работы: 2 часа.

Оборудование: ПК

Средства обучения: операционная система, текстовый процессор MS Word, программные средства определенного вида

Исходные данные:

1. Конспект занятия.
2. Задание для практической работы.

Перечень справочной литературы:

1) Программирование на языке высокого уровня. Программирование на языке С++: учеб. пособие / Т. И. Немцова, С. Ю. Голова, А. И. Терентьев ; под ред. Л. Г. Гагариной. – М. : ИД «ФОРУМ» : ИНФРА-М, 2018. – 512 с. – (Среднее профессиональное образование).

Краткие теоретические сведения:

Работа с графикой GDI+

Для работы с графикой в среде .NET предназначен класс **Graphics** пространства имен **System.Drawing.** Для вывода графических примитивов (линий, геометрических фигур), текста, растровых изображений необходимо создать объект класса **Graphics**, например, методом **CreateGraphics**:

Graphics g = this.CreateGraphics();

После создания объекта типа **Graphics** можно применять его свойства и методы. Наиболее часто используются объекты и методы классов **Pen** (рисование линий и геометрических фигур), **Brush** (заполнение областей), **Font** (работа с текстом), **Color** (работа с цветом).

Для интерактивного управления свойствами графических объектов удобно использовать манипулятор мышь. События мыши **MouseDown, MouseUp, MouseMove** и другие работают в сочетании с делегатом **MouseEventHandler**.

Например, при регистрации события движения мыши по форме в методе **InitializeComponent()** (в файле Form1.Designer.cs) появляется строка:

MouseMove += new MouseEventHandler(Form1_MouseMove);

При этом в файле кода Form1.cs создается шаблон метода-обработчика, которому передаются два параметра: объект-источник события и объект класса **MouseEventArgs**, который содержит информацию о событии, например: **X** и **Y** – координаты указателя мыши; **Button** – нажатая кнопка (левая, правая); **Clicks** – количество нажатий и отпусканий кнопки мыши; **Delta** – счетчик (со знаком) щелчков поворота колесика.

Эту информацию можно использовать в обработчике, например, для вывода координат в заголовок формы:

```
public void Form1_MouseMove(object sender, MouseEventArgs e)
{       Text = string.Format("координаты: x={0}, y={1}", e.X, e.Y);
}
```

Для управления графическими объектами нередко используют и клавиатуру. События клавиатуры **KeyUp, KeyDown** работают в сочетании с делегатом **KeyEventHandler.**

Например, при регистрации события нажатия клавиши записывается строка

```
KeyDown += new KeyEventHandler(Form1_KeyUp);
```

Создается шаблон метода-обработчика, которому передаются: объект-источник события и объект класса **KeyEventArgs**, который содержит информацию о событии, например: **KeyCode** – код

клавиши для событий KeyDown или KeyUp; **Modifiers** – какие модифицирующие клавиши (Shift, Alt, Control) были нажаты; **Handled** – было ли событие полностью обработано.
Эту информацию можно использовать в обработчике, например:

```
private void Form1_KeyDown (object sender, KeyEventArgs e)
{       MessageBox.Show(e.KeyCode.ToString(), “клавиша нажата!");
}
```

Работа с таймером

Класс для работы с таймером (**Timer**) формирует в приложении повторяющиеся события. События повторяются с периодичностью, указанной в миллисекундах, в свойстве **Interval**. Установка свойства **Enabled** в значение **true** запускает таймер. Каждый тик таймера порождает событие **Tick**, обработчик которого обычно и создают в приложении. В этом обработчике могут изменяться каике либо величины, и вызываться принудительная перерисовка окна. Напоминаем, что вся отрисовка при создании анимации должна находиться в обработчике события Paint.
Младшее событие **timer.Tick** можно связать через обобщенный делегат **EventHandler** с **лямбда-выражением** (разновидность анонимной функции).
Тогда вместо объявления функции обработчика этого события мы можем использовать следующую конструкцию:

```
timer.Tick += new EventHandler((o, ev) =>
{
// обработка младшего события — составное лямбда-выражение
});
```

Здесь «=>» — лямбда-оператор, связывающий анонимный метод (функцию) с лямбда-выражением.

Создание анимации

Для создания простой анимации достаточно использовать таймер, при тике которого будут изменяться параметры изображения (например, координаты концов отрезка) и обработки события **Paint** для рисования по новым параметрам. При таком подходе не надо заботиться об удалении старого изображения, ведь оно создается в окне заново.

Ход работы:

Требования к содержанию отчета:

- Номер и название практической работы.
- Цель работы.
- По каждой заданию (задаче/примеру) экранные формы (при наличии) и листинг программного кода, показывающие порядок выполнения практической работы, и результаты, полученные в ходе её выполнения.
- Ответы на контрольные вопросы в тетради.

Порядок выполнения работы:

Все задания практической работы размещать в своей папке проектов в новой папке

Пр20_1_Фамилия

В начале каждого файла проекта установить комментарии: пр.р.№_____ (указать номер), свою Фамилию. Формулировку задания

Задание 1. Простейшая анимация движения.

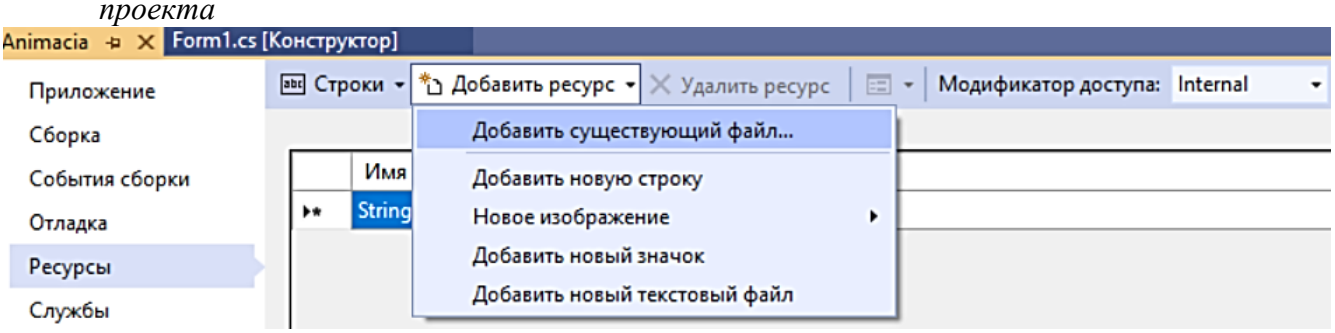
✓ Создайте новый проект **pr20_1.1_Фамилия** типа Windows Forms.
✓ Разместите на форме размером **580*200** две кнопки **Старт** и **Стоп**, а также элемент **PictureBox** размером **140*150**. Импортируйте в него изображение из файла **бег.gif** (рис. 2.1, *а*), не забудьте установить значение **StretchImage** для свойства **SizeMode**.

Примечание:

Для импорта изображения можно его сначала поместить в ресурсы проекта:

1) ***ПКМ** по имени проекта – **Свойства**, выбрать вкладку **Ресурсы** – развернуть список **Добавить ресурс** и выбрать **Добавить существующий файл…** и далее выбрать файлы изображения*
2) *при задании свойства **Image** элемента **PictureBox** файл выбрать из списка ресурсов*

проекта



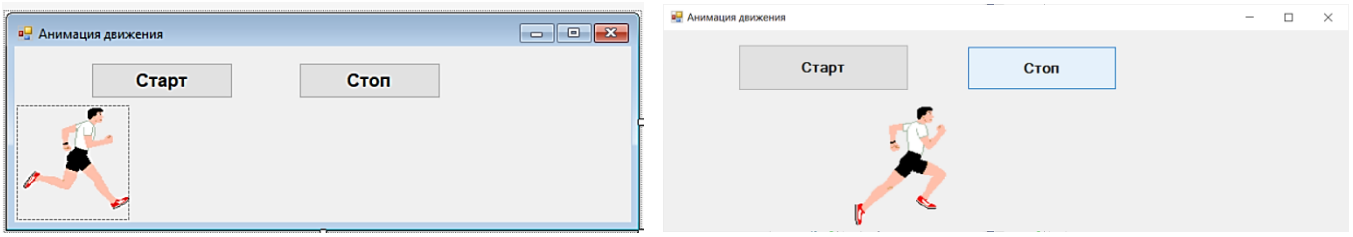


Рис. 2.1. Начальная (а) и промежуточная фазы анимации (б)

✓ Перетащите на форму **Timer** (свойства **Enabled = false, Interval = 20)**.

✓ Зарегистрируйте события **Tick** таймера, а также нажатий кнопок **Click**.

В шаблоны обработчиков введите коды:

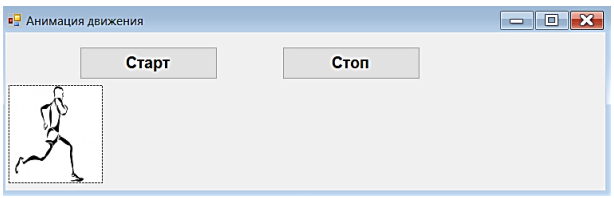
- перемещение по тикам таймера вправо на **2px**, если **Left <520**, иначе в начало

```
21          private void timer1_Tick(object sender, EventArgs e)
22          {
23              if (RunnerPictureBox.Left < 520) RunnerPictureBox.Left += 2;
24              else RunnerPictureBox.Left = 4;
25          }
            ссылка: 1
26          private void StartButton_Click(object sender, EventArgs e)
27          {
28              timer1.Enabled = true;  //старт
29          }
30
            ссылка: 1
31          private void StopButton_Click(object sender, EventArgs e)
32          {
33              timer1.Enabled = false; //стоп
34          }
35      }
```

✓ Протестируйте программу.

Примечание:

Если изображение имеет белый или какой-либо другой фон, для него необходимо установить прозрачный фон:



1 способ. *Например., имеем* *Для этого используется следующий программный код:*

```
13      public partial class Form1 : Form
14      {
15          Bitmap beg;     //определение объекта Bitmap
            ссылка: 1
16          public Form1()
17          {
18              InitializeComponent();
19
20              pictureBox1.BackColor = Color.Transparent; //определяем цвет фона изображения
21              beg = Properties.Resources.бег; // задаем ресурс на изображеие
22              beg.MakeTransparent();   //Сделать прозрачный цвет
23              pictureBox1.Image = beg;
24          }
```

2 способ. Для того, чтобы можно было использовать прозрачный фон для картинки (pictureBox), добавьте строку в класс Form1() (поместите ее обязательно после инициализации компонентов формы)

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();

        this.SetStyle(ControlStyles.SupportsTransparentBackColor, true);
    }
```

Примечание: для использования возможности прозрачного фона на картинке, необходимо чтобы графический файл имел формат ***PNG***

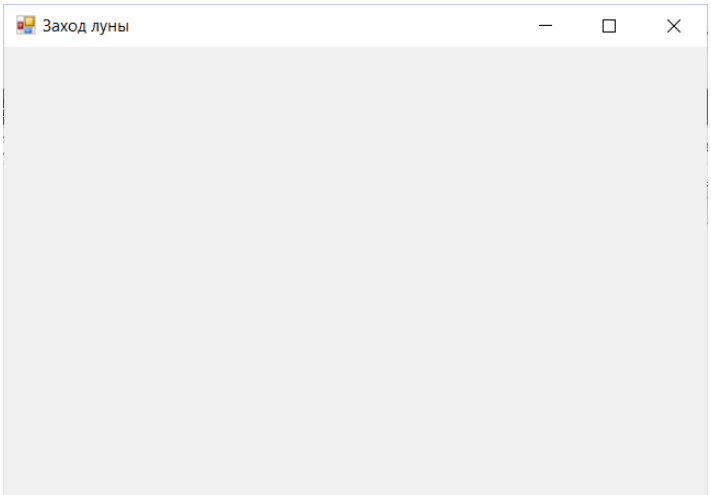
Задание 2. Простейшая анимация движения.

Создадим шуточную анимацию «Заход Луны». Добавим в класс Form1 метод, который будет вызываться при возникновении некоторого события, например, клика на форме. Тогда в течение некоторого времени в окне формы можно будет наблюдать заход Луны.

Идея алгоритма состоит в том, чтобы внутри обработчика **старшего события** (в нашем случае **клика на форме**) выполнялась анимация (заход Луны) в течение нескольких секунд. Для этого удобно использовать динамически создаваемый таймер, задающий частоту кадров. Каждое **срабатывание таймера (такт)** назовем **младшим событием**. Выключение таймера будет выполнено через заданное число тактов.

✓ Создайте новый проект **pr20_1.2_Фамилия** типа Windows Forms. Создайте форму по образцу

✓ Добавьте компонент **Timer**



Обработка старшего события (клика на форме) зададим следующим образом:

1) Создаем объект ***timer*** *через конструктор класса* ***Timer****. Задаем свойство* ***Interval = 40 мс*** *(25 кадров в секунду).*

2) Задаем счетчик перемещений и максимальное число кадров.

3) Задаем начальное положение и диаметр шара — Луны.

4) Задаем настройки графики: холст, его очистку, стиль и цвет кистей для рисования Луны и фона.

5) Рисуем шар в начальном положении.

6) Внутри скобок ***{ } :***

timer.Tick += new EventHandler((o, ev) =>

{

удаляем шар, изменяем его координаты, рисуем шар снова; считаем число перемещений, если оно максимальное, то останавливаем таймер.

}

);
7) Запускаем таймер.

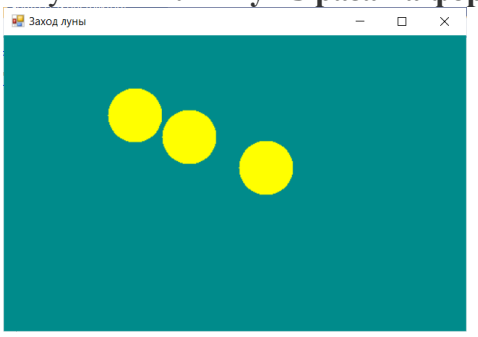
✓ Реализуйте программный код старшего события - **клика на форме**

Ниже приведен текст метода **Form1_Click()** с использованием связи через обобщенный делегат **EventHandler** с **лямбда-выражением**:

```
public partial class Form1 : Form
{
    ссылка: 1
    public Form1()
    {
        InitializeComponent();
    }

    ссылка: 1
    private void Form1_Click(object sender, EventArgs e)
    {
        Timer timer = new Timer();  // объект - таймер
        timer.Interval = 40;        // тик - каждые 40 миллисекунд
        int count = 0;              // счетчик перемещений
        int max = 150;              // их максимальное число
        int x = 20;                 // начальные
        int y = 20;                 // координаты
        int d = 60;                 // диаметр шара

        Graphics g = this.CreateGraphics();  // объект - на форме
        g.Clear(Color.DarkCyan);             // очистка цветом фона
        SolidBrush br = new SolidBrush(Color.Yellow);  // задание кисти
        SolidBrush brf = new SolidBrush(Color.DarkCyan);  // задание фона
        g.FillEllipse(br, x, y, d, d);       // заливка - начало

        // Каждые 40 мс будет удаляться и рисоваться новый шарик, остановка через 150 раз
        timer.Tick += new EventHandler((o, ev) =>
        {
            g.FillEllipse(brf, x, y, d, d);
            x += 5;
            y += 2;
            g.FillEllipse(br, x, y, d, d);
            count++;
            if (count == max)
                timer.Stop();
        }
        );
        timer.Start();   // запустили, остановится сам

    }
}
```

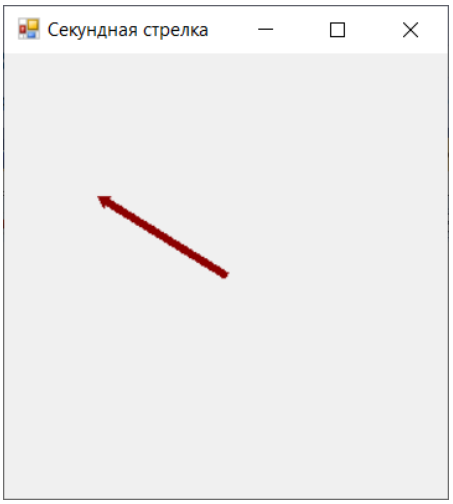
✓ Протестируйте программу. При необходимости откорректируйте код. Установите поясняющие комментарии в программном коде.

Результат: **Кликнув 3 раза на форме**, вы сможете наблюдать заход 3-х Лун.



Задание 3. Простейшая анимация движения. Секундная стрелка

1) Создайте новый проект **pr20_1.3_Фамилия** типа Windows Forms. Измените название формы.
2) Разместите на форме размером **400*400** компонент **Timer**



✓ выполните установку свойства **Enabled** в значение **true (** запускает таймер).

*Каждый тик таймера порождает событие **Tick**, обработчик которого обычно и создают в приложении. В этом обработчике могут изменяться каике либо величины, и вызываться принудительная перерисовка окна. Помните, что вся отрисовка при создании анимации должна находиться в обработчике события **Paint**.*

3) Создание анимации

*Для создания простой анимации достаточно использовать таймер, при тике которого будут изменяться параметры изображения (например, координаты концов отрезка) и обработки события **Paint** для рисования по новым параметрам. При таком подходе не надо заботиться об удалении старого изображения, ведь оно создается в окне заново.*

✓ Реализуйте обработчик события **Paint,** в котором будет рисоваться секундная стрелка соответствующего стиля

```
public partial class Form1 : Form
{
    //описываем переменные доступные в любом обработчике событий класса Form1
    private int x1, y1, x2, y2, r;


    private double a;
    private Pen pen = new Pen(Color.DarkRed, 2);

    public Form1()
    {
        InitializeComponent();
    }

    private void Form1_Paint(object sender, PaintEventArgs e)
    {
        Graphics g = e.Graphics;
        pen.Width = 5;
        pen.DashStyle = DashStyle.Solid;
        pen.EndCap = LineCap.ArrowAnchor;
        g.DrawLine(pen, x1, y1, x2, y2); //рисуем секундную стрелку
    }
```

✓ Реализуйте обработчики событий **Load** и **Tick,** в которых будут изменяться параметры изображения:

```
private void Form1_Load(object sender, EventArgs e)
{
    //определяем центр экрана
    x1 = ClientSize.Width / 2;
    y1 = ClientSize.Height / 2;
    r = 100; //задаем радиус
    a = 0;  //задаем угол поворота
    //определяем конец секундной стрелки с учетом центра экрана
    x2 = x1 + (int) (r * Math.Cos(a));
    y2 = y1 - (int)(r * Math.Sin(a));
}

ссылка: 1
private void timer1_Tick(object sender, EventArgs e)
{
    a -= 0.1;//уменьшаем угол на 0,1 радиану
    //определяем конец секундной стрелки с учетом центра экрана
    x2 = x1 + (int)(r * Math.Cos(a));
    y2 = y1 - (int)(r * Math.Sin(a));
    Invalidate(); //вынудительный вызов перерисовки (Paint)
}
```

✓ Протестируйте приложение. При необходимости измените свойства компонентов и код.

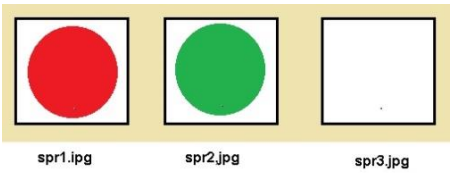
Задание 4. Спрайтовая анимация.

Спрайт (англ. Sprite — фея; эльф) — графический объект в компьютерной графике, чаще всего — растровое изображение, которое можно отобразить на экране.

Необходимо изобразить перемещение шара по некоторой траектории (например, по прямой) на холсте графического объекта

1) Создайте новый проект **pr20_1.4_Фамилия** типа Windows Forms. Измените название формы.

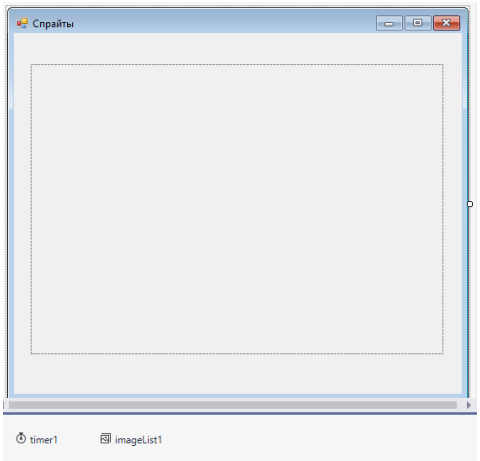
Предварительные действия. *При необходимости нужно создать Paint три изображения на белом фоне, каждый размером **100x100** пикселей. Нарисуем (без рамок) в первом квадрате красный круг, во втором — зеленый, третий квадрат оставим белым. Сохраним их в формате JPEG, присвоив имена файлов spr1.jpg, spr2.jpg, spr3.jpg соответственно.*



У вас уже есть готовые файлы работы (находятся в папке **к Пр20_1**)

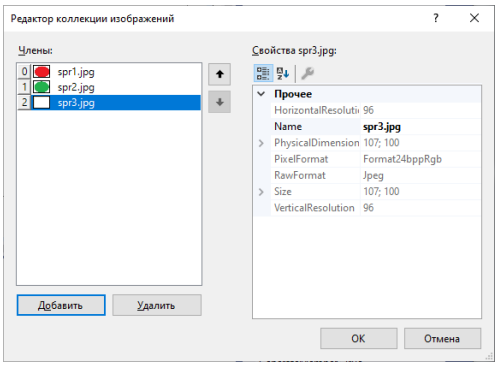
Часто бывает удобно использовать контейнеры, напирмер в виде **pictureBox1**, *вместо непосредственного рисования на форме. Для хранения трех спрайтов будем использовать невизуальный компонент* **ImageList1** *(список изображений), для управления анимацией будем использовать компонент* **timer1**.

2) Создайте форму по образцу



3) Добавьте на форму **pictureBox1**, а также невизуальные компоненты **timer1** и **ImageList1**. Пусть pictureBox1 занимает не все окно формы.

✓ В коллекцию изображений (свойство **Images**) добавьте через окно *Свойства* три рисунка с индексами ***0,1,2***.



✓ Для таймера свойство **Interval** установить значение **100**. Это будет означать, что

изменение положения спрайта-шарика будет происходить примерно ***14 раз в секунду***.

4) В описание класса **Form1** перед конструктором Form1() введите описание двух объектов и трех переменных класса:

```
public partial class Form1 : Form
{
    Graphics g;      //  графический объект – некий холст
    Bitmap buf;      //  буфер для Bitmap-изображения
    int stage = 0;       // индекс чередующегося шара (красный – 0, зеленый – 1, белый – 2)
    int x = 0, y = 0;    // координаты шара (левого верхнего угла квадрата)
```

5) В конструктор **Form1()** добавьте следующие операторы:

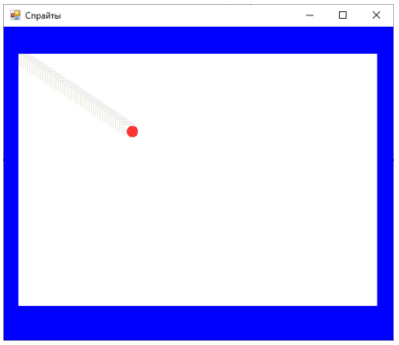
```
public Form1()
{
    InitializeComponent();
    this.BackColor = Color.Blue; // цвет формы
    buf = new Bitmap(pictureBox1.Width, pictureBox1.Height);
    // Создаем новый экземпляр класса Bitmap с заданным размером
    g = Graphics.FromImage(buf);
    // Создает новый графический объект из указанного рисунка
    SolidBrush bf = new SolidBrush(Color.White); // перекраска фона
    g.FillRectangle(bf, 0, 0, pictureBox1.Width, pictureBox1.Height);
    timer1.Enabled = true; // старт таймера
}
```

Первый изменяет цвет формы на синий, четыре следующих оператора готовят холст для рисования на pictureBox1, последний запускает в работу таймер (в конструкторе первоначально timer1.Enable=false). Переменные g и buf, объявленные в классе, в конструкторе Form1 конкретизируются: buf получает ссылку на pictureBox1, а g получает ссылку на новый графический объект. Теперь рисование спрайта-шара будет выполняться именно на поверхности pictureBox1. Координаты окна: (0,0) — (pictureBox1.Width, pictureBox1.Height) — левый верхний и правый нижний углы.

6) Рисование шара-спрайта реализуется в методе **timer1_Tick( )** с использованием переменной **stage** и метода **Draw(g, Point(x, y), stage)** объекта **imageList1**:

```
private void timer1_Tick(object sender, EventArgs e)
{
    if (stage < 1)
    {
        imageList1.Draw(g, new Point(x, y), 0); // вывести красный
        stage++;                                // сменить номер шара
    }
    else if (stage < 2)
    {
        imageList1.Draw(g, new Point(x, y), 1); // вывести зеленый
        stage++;                                // сменить номер шара
    }
    else if (stage < 3)
    {
        imageList1.Draw(g, new Point(x, y), 2); // вывести белый квадрат
        stage++;
    }
    else if (stage == 3)     // условие возврата красного шара - 0
    {
        stage = 0;
        x += 3; y += 2;    // траектория - прямая под углом вниз
    }
    pictureBox1.Image = buf;   // Показать шар
    if (x > pictureBox1.Width) // Возврат в начало
    { x = 0; y = 0; };
}
```

На холсте — будет происходить смена шаров в виде анимации движения по прямой от верхнего левого угла холста к правому нижнему!
Если закомментировать оператор (показать шар), то никаких изображений мы не увидим. Оно, конечно, формируется в паре объектов «g–buf» и в любой момент может появиться на холсте через вызов этого оператора. Последний оператор после прохождения шаром диагонали возвращает его в начальную позицию для продолжения. Поэкспериментируйте!



Задания для самостоятельной работы

1. Создайте приложение, в котором анимируется прямолинейное движение спутника **sputnik.png** на фоне звездного неба **sky.gif**.
2. Создайте приложение, в котором анимируется падение яблока **apple.png** с башни **tower.jpg**.
3. Создайте приложение, в котором по щелчку мышью по изображению совы **sova.png** оно начинает увеличиваться до достижения двукратного размера. Щелчок мыши по увеличенному изображению вызывает его уменьшение до первоначальных размеров.
4. Создайте приложение, в котором анимируется движение Луны **luna.png** вокруг Земли **zem.png** по эллиптической траектории.

Контрольные вопросы:

1) Опишите этапы выполнения простейшей анимации движения изображения.
2) Что называют старшим и младшим событиями? Приведите примеры.
3) Опишите способ реализации анимации путем связи через обобщенный делегат EventHandler с лямбда-выражением (приведите пример)
4) Что такое спрайт?
5) Опишите технологию создания спрайтовой анимации.
6) Опишите способы задания прозрачного фона для изображения