Практическая работа №19_2

Тема: Разработка игрового приложения

Цель работы: создание игрового приложения
Задачи:
– совершенствование приемов создания приложений Windows Forms с графическими объектами и анимацией;
– совершенствование приемов рефакторинга
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 для разработки приложений Visual Studio
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:
Элемент управления Windows Forms **PictureBox** используется для вывода изображений в формате точечных рисунков, GIF, JPEG, метафайлов и значков. Отображаемое изображение определяется свойством **Image**, которое можно задать во время выполнения или во время разработки. Свойство **SizeMode** определяет, каким образом изображение и элемент управления соответствуют друг другу.

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
Пр19_2_Фамилия

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

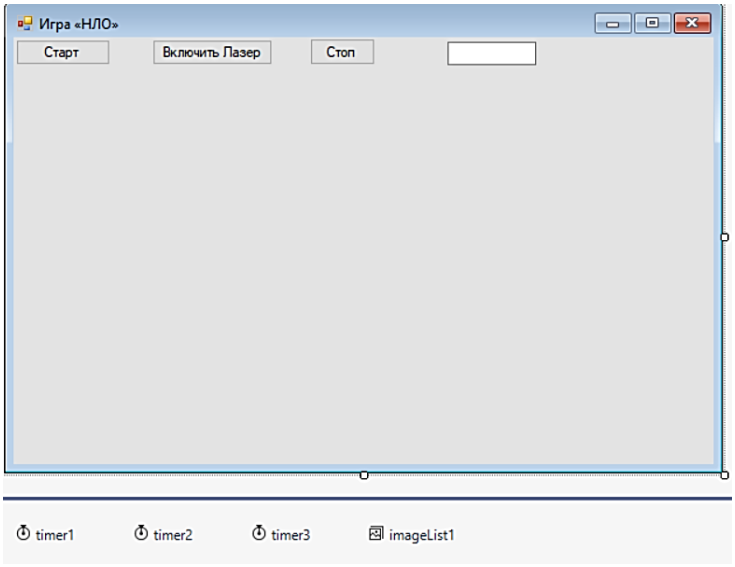
Задание 1. Создание игры **«НЛО»**

Начнем с создания достаточно простого шутера «НЛО», в котором придется отражать нашествие на Землю армады НЛО. Дальнейшее ее развитие вы сможете выполнить самостоятельно.
Проектирование шутера начинается с разработки сценария, после чего определяются классы (поля и методы), проектируется визуальный интерфейс игры, программируются методы этих классов.

Постановка задачи:
Возможный сценарий игры
Действующие персонажи: 1) противник (враги — Enemies); 2) НЛО (один из видов противника — Bugs-«жучки»), ограничимся пока одним видом врагов; 3) защитник Земли (игрок — Player).
Место действия. Сражение происходит в космическом пространстве в окрестностях Земли.

Противник. НЛО имеют традиционную форму тарелок, каждая из них отличается размерами и окраской. Над Землей они появляются сериями и планируют сверху вниз на Землю с различной скоростью, при этом возможно и их горизонтальное смещение. Интервал времени между появлениями серий может уменьшаться. Количество НЛО в небе одновременно не превышает некоторого максимального числа — количество НЛО в космическом флоте противника.
Игрок. Игрок, находясь в своем корабле, использует свое оружие с лазерным прицелом (blaster) и перемещается в околоземном пространстве, старается сбить НЛО противника. Для управления перемещением игрока используется мышь, стрельба — нажатием левой кнопки мыши.
Взаимодействие. При попадании в НЛО следует обозначить его подрыв и удалить его останки из околоземного пространства. В первом приближении будем считать, что физическое касание НЛО и корабля не приводит к потере корабля игрока (это будет следующая задача).
Цель игры — уничтожить максимальное количество НЛО противника (максимальный результат — 100%).

1) Создайте новый проект **pr19_2_Фамилия** типа Windows Forms.
2) Создайте форму (в том числе с не визуальными компонентами):

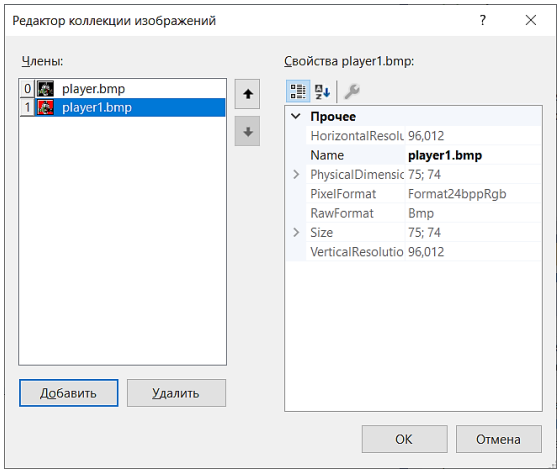


Для использования полного экрана задайте свойство Form1 **WindowState = Maximized**.
Примечание: Вместо кнопок можно задать меню из трех позиций: ***Старт, Лазер, Стоп****. Соответственно сменятся и названия методов, связанных с выбором пунктов меню.*
3) Создайте элементы для управления игрой:
   - Выберите на панели элементов 4 визуальных объекта:

три кнопки — button1 («Старт»), button2 («Лазер: включить/выключить»), button3 («Стоп»), задайте кнопкам соответствующие имена, например, **StartButton, LazerButton, StopButton**.
textBox1 для отображения счета игры (**name = ResultTextBox**)
   - Также добавьте 3 не визуальных объекта: **timer1, timer2 и imageList1**.

Объект imageList1 используется для хранения изображений игрока (player).
Таймер timer1 будет задавать частоту изменений (минимальный временной такт). Таймер timer2 будет использоваться для генерации серий НЛО. Активность игрока нужно связать, как указано в сценарии, с нажатием левой кнопки мыши и перемещением ее по экрану — событие **MouseClick**.
4) Выполните настройку объектов:
   - Для игрока понадобятся изображения размером 100x100 пикселей с именами **player.bmp** и **player1.bmp**, например так:


и добавьте их в папку **Resources** проекта.



   - Задайте свойство imageList1 **ImageSize = 100;100** и внесите в коллекцию **Images** эти два файла (члены 0 и 1)
   - Задайте button3.Enabled=false.

Изображения НЛО будет генерировать метод public void Form_bug().

- Настройки таймеров:
  timer1.Enable=false (выключен);
  timer2.Enable=false (выключен);
  timer1.Interval=400 (0,4с);
  timer2.interval=5000 (5с).

5) Проектирование классов

Обозначив в сценарии сущности, определим три класса:

- Enemies — противник;
- Bugs — НЛО;
- Player – игрок.

Автоматически создается класс

- Form1 – форма.

И понадобится вспомогательный класс

- BrushColor – кисти/цвета.

Класс Player

Поля класса:

```
public class Player
{
    //Поля класса Player
    public Point point;          // положение игрока в 2D-пространстве
    public Size size;            // размеры игрока
    public Region reg;           // занимаемая им область в пространстве
    public Pen laser_pen;        // свойство оружия
```

Методы класса:

public void New_player() // задать свойства (параметры) игрока
public void Show_player() // показать его на поле битвы

Содержание методов этого и последующих классов рассмотрены ниже.

Класс Bugs

Поля:

```
public class Bugs
{
    //Поля Класса Bugs
    public Point point;               // положение НЛО в 2D-пространстве
    public Size size;                    // размеры НЛО
    int veloX;                                 // скорость смещения по X
    int veloY;                                 // скорость_падения по Y
    public HatchBrush br;         // кисть для покраски НЛО
    public Region reg = new Region();   // занимаемая им область в пространстве
    public Boolean life = true;                  // НЛО жив (true) или мертв (false)
```

Методы:

public void New_bug() // задать свойства (параметры) НЛО
public void Form_bug() // задать форму НЛО, например, тарелку
public void Move_bug() // задать новое местоположение НЛО

Класс Enemies

Поля:

```
public class Enemies
{
    //Поля Класса Enemies
    // для генерации серий
    public int Delta_N;          // количество НЛО в серии
    public int N_generation;     // число генераций – серий
    public int k_generation;     // номер серии
    public int N;                // актуальное количество НЛО на экране
    // массив НЛО-объектов
    public Bugs[] bugs = new Bugs[Form1.N_max];
```

Примечание. Поле N_max задается константой в классе Form1 (см. ниже), а последнее поле bugs задает массив ссылок на объекты-НЛО.

Методы:

public void New_Enemies() // инициализация объектов НЛО
public void Show_bugs() // сдвинуть и показать «живые» НЛО
public void Enemy() // генерация одной серии НЛО
public void Killed_bugs() // определение сбитых НЛО
public int Select_bugs() // удаление сбитых НЛО

Вспомогательный класс BrushColor

Поля класса:

```
public class BrushColor
{
    // Поля Вспомогательный класс BrushColor
    public Color FonColor;              // цвет фона
    public Color LaserColor;           // цвет лазера
    public Color DashBug;               // цвет штриховки НЛО
    public Color KilledBug;             // цвет сбитого НЛО
```

Методы класса:

public BrushColor() // конструктор (настройка цветов)
public HatchBrush New_br(int rch) // кисть для задания цвета НЛО
public Color RandomColor(int rch) // генератор случайного цвета

Класс Form1

Поля:

```
public partial class Form1 : Form
{
    //Поля Класс Form1
    public const int N_max = 200;           // Максимальное количество НЛО на экране
    public Player player = new Player();     // Игрок, который сбивает НЛО (объект)
    public Boolean laser = false;            // Его оружие – бластер
    public Bitmap imageP;                    //  Изображения игрока
    public int Result = 0;                   // Количество сбитых НЛО (счет игры)
    public Graphics g;                       // холст для битвы
    public BrushColor bc = new BrushColor();       // набор кистей и цветов
    public Enemies nlo = new Enemies();            // Все НЛО
```

Как видно, задана всего одна константа (необходима для определения размерности массива ссылок, см. класс Enemies); три объекта классов Player, Enemies и BrushColor; поле laser (включен/выключен); поле imageP для хранения изображения игрока; поле Result для ведения счета подбитых НЛО; поле g — холст (графический контекст) для рисования.

Методы класса Form1 (реакции на события):

а) конструктор формы
public Form1()
б) при загрузке формы private void Form1_Load(object sender, EventArgs e)
в) старт игры private void button1_Click(object sender, EventArgs e)
г) включение/отключение лазера private void button2_Click(object sender, EventArgs e)
д) стоп игры, результат private void button3_Click(object sender, EventArgs e)
е) один временной такт игры private void timer1_Tick(object sender, EventArgs e)
ж) генерация серий private void timer2_Tick(object sender, EventArgs e)
з) попадание НЛО под вертикальный обстрел лазером
private void Form1_MouseClick(object sender, MouseEventArgs e)

6) Реализация методов **класса Player**:

а) Новый игрок

```
// новый игрок
public void New_player(Form1 F)
{
    size = F.imageP.Size;
    point.X = 0;
    point.Y = 0;
    Rectangle rec = new Rectangle(point, size);
    reg = new Region(rec);
    laser_pen = new Pen(new HatchBrush(HatchStyle.DashedUpwardDiagonal, F.bc.LaserColor, F.bc.LaserColor), 3);
}
```

Комментарии: Размер изображения определяется через размер рисунка (см. далее метод Form1_Load()). Левый верхний угол объекта имеет координаты (0,0). Область, занимаемая игроком, прямоугольная. Цвет луча лазера определяется свойством F.bc.LaserColor. Для доступа к объекту bc, расположенному на форме, параметр метода задаем как Form1 F.

б) показать игрока

```
// показать игрока
public void Show_player(Form1 F, int x, int y)
{
    F.g.ResetClip();
    F.g.FillRegion(new SolidBrush(F.BackColor), reg);
    point.X = x - size.Width / 2;
    point.Y = y;
    Rectangle rec = new Rectangle(point, size);
    reg = new Region(rec);
    F.g.DrawImage(F.imageP, point);
    F.g.ExcludeClip(reg);
}
```

Комментарии: Метод имеет параметры: Form1 F (для доступа к объекту g и свойству BackColor – цвету фона), int x, int y — новые координаты игрока (x – ось симметрии). Первый оператор снимает защиту с предыдущей области расположения игрока. Также определяется новая область reg. Метод DrawImage(F.imageP, point) обеспечивает рисование игрока, метод ExcludeClip(reg) обеспечивает защиту заданной области до следующего вызова метода.

Попробуйте убрать (закомментировать) первый и последний операторы и посмотрите на изменения в отображении игрока.

7) Реализация методов **класса Bugs**

а) генерация одного НЛО

```
// генерация одного НЛО
public void New_bug(Form1 F, int rch)
{
    Random rv = new Random(rch);
    point.X = rv.Next(10, Form1.ActiveForm.Width - 40);
    point.Y = rv.Next(10, Form1.ActiveForm.Height / 5);
    size.Width = rv.Next(20, 50);
    size.Height = size.Width * 2 / 3;
    veloX = rv.Next(7) - 3;
    veloY = rv.Next(3, 10);
    br = F.bc.New_br(rch);
    reg = Form_bug();
}
```

Комментарии: Метод формирует начальные координаты НЛО, его размеры, скорости смещения за 1 такт срабатывания таймера timer1, выбирает цвет кисти для его покраски. Везде используется генератор случайных чисел класса Random. Для задания области reg вызывается следующий метод.

б) задание формы НЛО

```
//задание формы НЛО
public Region Form_bug()
{
    Point pt = new Point();
    Size st = new Size();
    pt.X = point.X;
    pt.Y = point.Y + size.Height / 4;
    st.Width = size.Width;
    st.Height = size.Height / 2;
    Rectangle rec = new Rectangle(pt, st);
    GraphicsPath path1 = new GraphicsPath();
    path1.AddEllipse(rec);
    Region reg = new Region(path1);
    rec.X = point.X + size.Width / 4;
    rec.Y = point.Y;
    rec.Width = size.Width / 2;
    rec.Height = size.Height;
    path1.AddEllipse(rec);
    reg.Union(path1);
    return reg;
}
```

Комментарий: Использование объекта класса GraphicsPath и метода reg.Union(path1) — объединение двух эллипсов.

в) вертикальное падение НЛО с горизонтальными смещениями

```
// вертикальное падение НЛО с горизонтальными смещениями
public void Move_bug()
{
    point.X += veloX;
    point.Y += veloY;
    reg = Form_bug();
}
```

Комментарий: новая область размещения НЛО заполняется на каждом такте срабатывания таймера 1 после изменения координат через метод Form_bug().

8) Реализация методов **класса Enemies**

а) настройка серий и создание ссылок на объекты НЛО

```
//настройка серий и создание ссылок на объекты НЛО
public void New_Enemies(Form1 F)
{
    N_generation = 10;
    Delta_N = Form1.N_max / N_generation;
    k_generation = 0;
    N = 0;
    for (int j = 0; j < Form1.N_max; j++)
        bugs[j] = new Bugs();
}
```

Комментарии: В методе задается жестко число серий — 10. Вычисляется число НЛО в каждой серии. Обнуляется счетчик числа генераций (серий) и числа активных НЛО. Создаются ссылки на максимально возможное число объектов.

б) удаление сбитых НЛО

```
//  удаление сбитых НЛО
public int Select_bugs()
{
    int k = 0;
    for (int j = 0; j < N; j++)
    {
        if (!bugs[j].life)
            k++;
    }
    for (int i = 0; i < k; i++)
    {
        for (int j = 0; j < N; j++)
        {
            if (!bugs[j].life)
            {
                for (int j1 = j; j1 < (N - 1); j1++)
                    bugs[j1] = bugs[j1 + 1];
                break;
            }
        }
        N--;
    }
    return k;    // счетчик подбитых НЛО
}
```

Комментарии: Если в результате попадания НЛО под луч лазера его свойство life=false, то такой объект удаляется из массива bugs. Сначала определяется общее количество подбитых НЛО, затем в цикле они по очереди удаляются путем сдвига. Количество активных НЛО N также уменьшается. Метод возвращает k — число сбитых НЛО на данном такте.

в) смещение и отображение НЛО

```
// смещение и отображение НЛО
public void Show_bugs(Form1 F)
{
    for (int j = 0; j < N; j++)
    {
        bugs[j].Move_bug();
        F.g.FillRegion(bugs[j].br, bugs[j].reg);
    }
}
```

Комментарий: Метод в цикле для активных НЛО обеспечивает их смещение и отображение на экране.

г) одна серия НЛО

```
//одна серия НЛО
public void Enemy(Form1 F)
{
    int N0 = N;
    N = N + Delta_N;
    int rch;
    Random rnd = new Random();
    for (int j = N0; j < N; j++)
    {
        bugs[j] = new Bugs();
        rch = rnd.Next();
        bugs[j].New_bug(F, rch);
        F.g.FillRegion(bugs[j].br, bugs[j].reg);
    }
}
```

Комментарий: Метод добавляет новую серию НЛО, каждый из которых имеет свой цвет, размеры и начальное местоположение.

д) подбитые НЛО, выделены F.bc.KilledBug цветом

```
// подбитые НЛО, выделены F.bc.KilledBug цветом
public void Killed_bugs(Form1 F, int x, int y)
{
    for (int j = 0; j < N; j++)
    {
        Rectangle r = new Rectangle(x - bugs[j].size.Width / 2, 0, bugs[j].size.Width, y);
        if (bugs[j].reg.IsVisible(r, F.g) & F.laser)
        {
            bugs[j].br = new HatchBrush(HatchStyle.DarkHorizontal, F.bc.KilledBug, F.bc.KilledBug);
            F.g.FillRegion(bugs[j].br, bugs[j].reg);
            bugs[j].life = false;
        }
    }
}
```

Комментарий: проверяет попадание объектов под лазерный прицел, отмечает их в свойстве life как false, обеспечивает вспышку НЛО при подрыве.

9) Реализация методов **класса BrushColor**

а) кисть для задания цвета НЛО

```
// кисть для задания цвета НЛО
public HatchBrush New_br(int rch)
{
    return new HatchBrush(HatchStyle.DashedUpwardDiagonal, DashBug, RandomColor(rch));
}
```

Комментарий: Метод возвращает кисть класса HatchBrush (шаблон штриховки) из библиотеки System.Drawing.Drawing2D, состоит из одного оператора.

б) случайный цвет

```
// случайный цвет
public Color RandomColor(int rch)      // rch - случайное число
{
    int r, g, b;
    byte[] bytes1 = new byte[3];        // массив 3 цветов
    Random rnd1 = new Random(rch);
    rnd1.NextBytes(bytes1);             // генерация в массив
    r = Convert.ToInt16(bytes1[0]);
    g = Convert.ToInt16(bytes1[1]);
    b = Convert.ToInt16(bytes1[2]);
    return Color.FromArgb(r, g, b);     // возврат цвета
}
```

в) самостоятельно опишите структуру метода BrushColor(), в котором включите настройку цветов:

- ✓ черный цвет фона
- ✓ произвольный цвет лазера, например, желтый или красный
- ✓ произвольный цвет штриховки НЛО
- ✓ ярко-зеленой цвет для сбитого НЛО

10) Реализация методов **класса Form1**

а) конструктор формы

```
public Form1()
{
    InitializeComponent();
}
```

б) при загрузке формы

```
private void Form1_Load(object sender, EventArgs e)
{
    g = this.CreateGraphics();          // инициализация холста
    BackColor = bc.FonColor;            // цвет фона
    imageP = new Bitmap(imageList1.Images[0], 100, 100);
    player.New_player(this);            // инициализация игрока
    nlo = new Enemies();                // инициализация противника
    nlo.New_Enemies(this);              // инициализация НЛО как объектов
}
```

Комментарии: При загрузке формы задается холст, цвет фона, изображение игрока (прямоугольная область — первое изображение из коллекции imageList1.Images[]), инициализируются объекты imageP, player, nlo (все НЛО).

в) Старт игры

```
private void StartButton_Click(object sender, EventArgs e)
{
    nlo.k_generation = 0;
    nlo.Enemy(this);
    timer1.Start();
    timer2.Start();
    StopButton.Enabled = true;
    StartButton.Enabled = false;
}
```

Комментарии: Номер первой серии = 0. Генерация первой серии НЛО — nlo.Enemy(this). Запуск таймеров. Открыть доступ к кнопке «Стоп», закрыть доступ к кнопке «Старт».

г) Включение/отключение лазера игроком

```
private void LazerButton_Click(object sender, EventArgs e)
{
    if (laser)
    {
        laser = false;
        LazerButton.Text = "Включить Лазер";
    }
    else
    {
        laser = true;
        LazerButton.Text = "Отключить Лазер";
    }
}
```

д) Стоп. Результат

```
private void StopButton_Click(object sender, EventArgs e)
{
    timer1.Stop();
    timer2.Stop();
    imageP = new Bitmap(imageList1.Images[1], 100, 100);
    int procent = Result * 100 / (nlo.Delta_N * nlo.N_generation);
    string msg = "Подбито " + Result.ToString() + " НЛО, " + procent.ToString() + "% результат";
    MessageBox.Show(msg, "Ваш результат", MessageBoxButtons.OK);
    player.Show_player(this, 50, 50);
    nlo.N = 0;
    StartButton.Enabled = true;
    Result = 0;
    ResultTextBox.Text = Result.ToString();
}
```

Комментарии: Остановка таймеров (игры). Помещение игрока в «гараж» (на красном фоне, вывод второго изображения из коллекции imageList1.Images[]), расчет результата игры в

процентах, его вывод в окне MessageBox, обнуление числа активных объектов НЛО, открытие доступа к кнопке «Старт», обнуление результата. Игра снова может быть запущена без перезапуска приложения.

е) один временной такт

```
private void timer1_Tick(object sender, EventArgs e)
{
    g.Clear(BackColor);
    Result = Result + nlo.Select_bugs();
    nlo.Show_bugs(this);
    ResultTextBox.Text = Result.ToString();
}
```

Комментарии: При срабатывании 1-го таймера производится очистка фона кроме защищенной области под игроком (см. выше метод Show_player(Form1 F, int x, int y)), что позволяет избежать исчезновения игрока. К результату добавляется число сбитых на предыдущем такте НЛО. Отображаются активные (ещё «живые») НЛО и текущий счет игры.

ж) генерация серий

```
private void timer2_Tick(object sender, EventArgs e)
{
    nlo.k_generation++;
    timer2.Interval -= 100;
    if (nlo.k_generation < nlo.N_generation)
        nlo.Enemy(this);
    else
        timer2.Stop();
}
```

Комментарии: Увеличение номера серии на 1, сокращение на 100 мс интервала выпуска следующей серии, если реализованы все nlo.N_generation, то остановка генерации (timer2.Stop();), иначе генерация очередной серии.

з) попадание НЛО под вертикальный обстрел лазером

```
private void Form1_MouseClick(object sender, MouseEventArgs e)
{
    player.Show_player(this, e.X, e.Y);
    if (laser)
        g.DrawLine(player.laser_pen, player.point.X + player.size.Width / 2, player.point.Y, player.point.X + player.size.Width / 2, 0);
    nlo.Killed_bugs(this, e.X, e.Y);
}
```

Комментарии: Главный метод, вызываемый при клике на форме, обеспечивает показ игрока в заданном мышью положении, выстрел из оружия, отметка подбитых НЛО, вспышки при их подрыве.Событие MouseClick демонстрирует взаимодействие объектов трех классов: Form1, Player и Enemies, а опосредовано и всех пяти (+ Bugs и BrushColor).

11) Выполните элементарные приемы рефакторинга программного кода: удаление ненужных директив в проекте (**ПКМ - Удалить и отсортировать директивы usings**)
12) Развитие игры

Опираясь на пример игры, попробуйте самостоятельно ее развить:

- Например, чтобы столкновение игрока с НЛО приводило к уменьшению счета игры вплоть до ее остановки при ранениях, несовместимых с жизнью игрока.
- Второе направление — изменение траекторий движения, как игрока, так и противников.
- Третье направление — добавление нового вида (класса) противников, отличающихся от безобидных пока НЛО, после чего создайте для них общий класс, от которого они будут наследоваться (принцип наследования).

Не забывайте и про принцип полиморфизма, пусть методы родственных классов называются одинаково, а действуют по-разному.