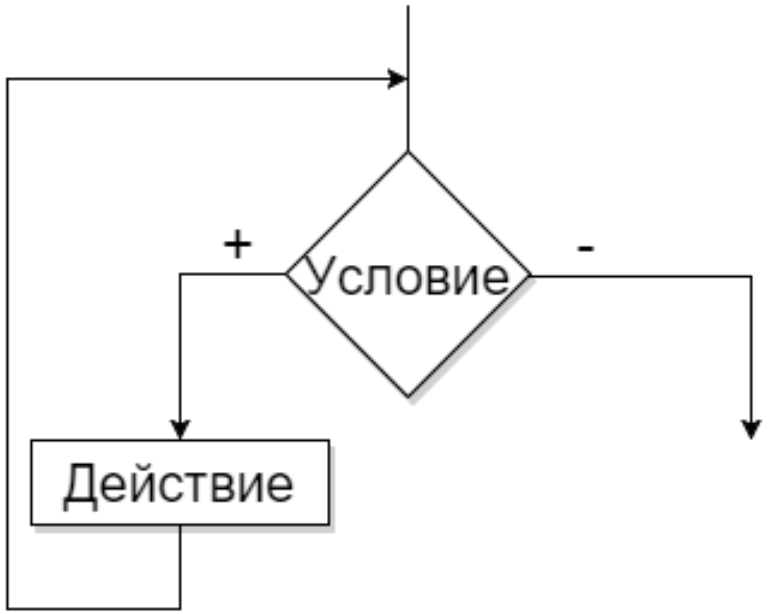
Основы программирования

ФИСТ 1 курс
Власенко
Олег
Федосович

Лекция 6
Цикл while. Enum.
Делаем простую игру - 2. Таймер.
Массив.

Цикл с предусловием while

```
while (Условие) {
        Действие;
}
```



Задача: ввести число. Найти сумму его цифр.

Ввод:

34

Вывод:

7

Ввод:

1023

Вывод:

6

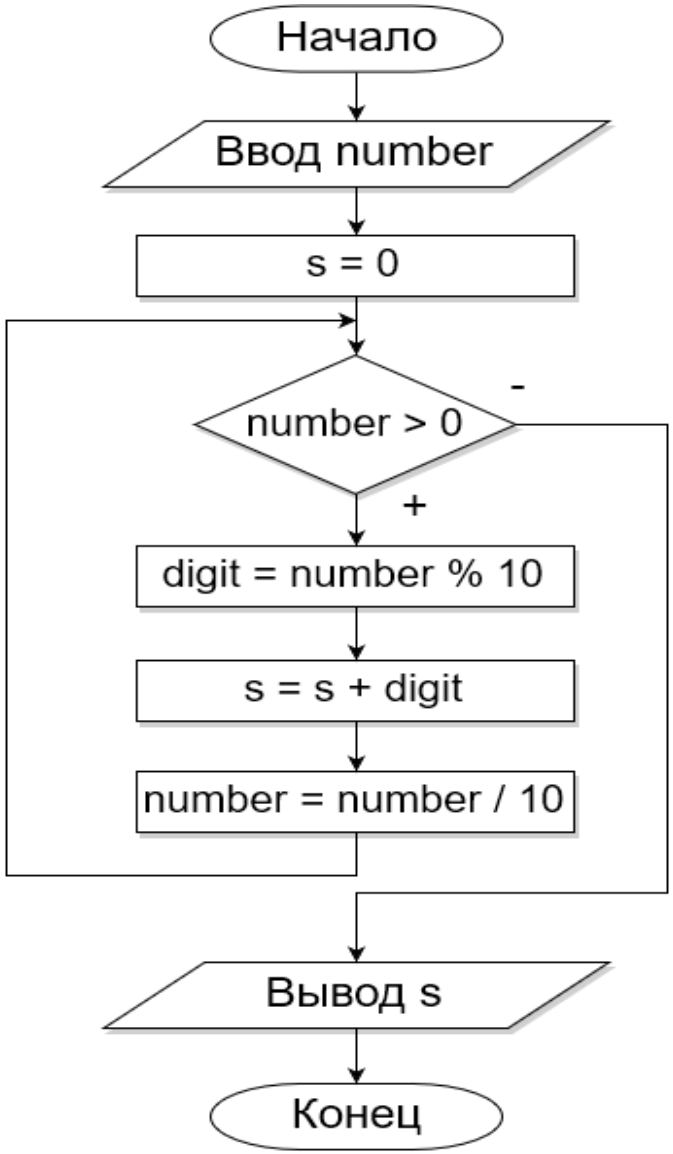
Ввод:

9876

Вывод:

30

Задача: ввести число. Найти сумму его цифр.



Задача: ввести число. Найти сумму его цифр.

```
void main() {
        int number;
        scanf("%d", &number);
        int s = 0;
        // Вычисляем сумму цифр
        while (number > 0) {
                // остаток от деления на 10 это последняя
                // цифра числа. Например: 129 % 10 = 9
                int digit = number % 10;
                // к сумме добавляем только что полученную цифру
                s = s + digit;
                // отбрасываем последнюю цифру числа
                // Например: 129 / 10 = 12
                number = number / 10;
        }
        printf("s=%d\n", s);
}
```

Enum: Перечисления (Перечисляемый тип)

Перечисление состоит из набора именованных целочисленных констант.

Синтаксис

enum identifier { enumerator-list }

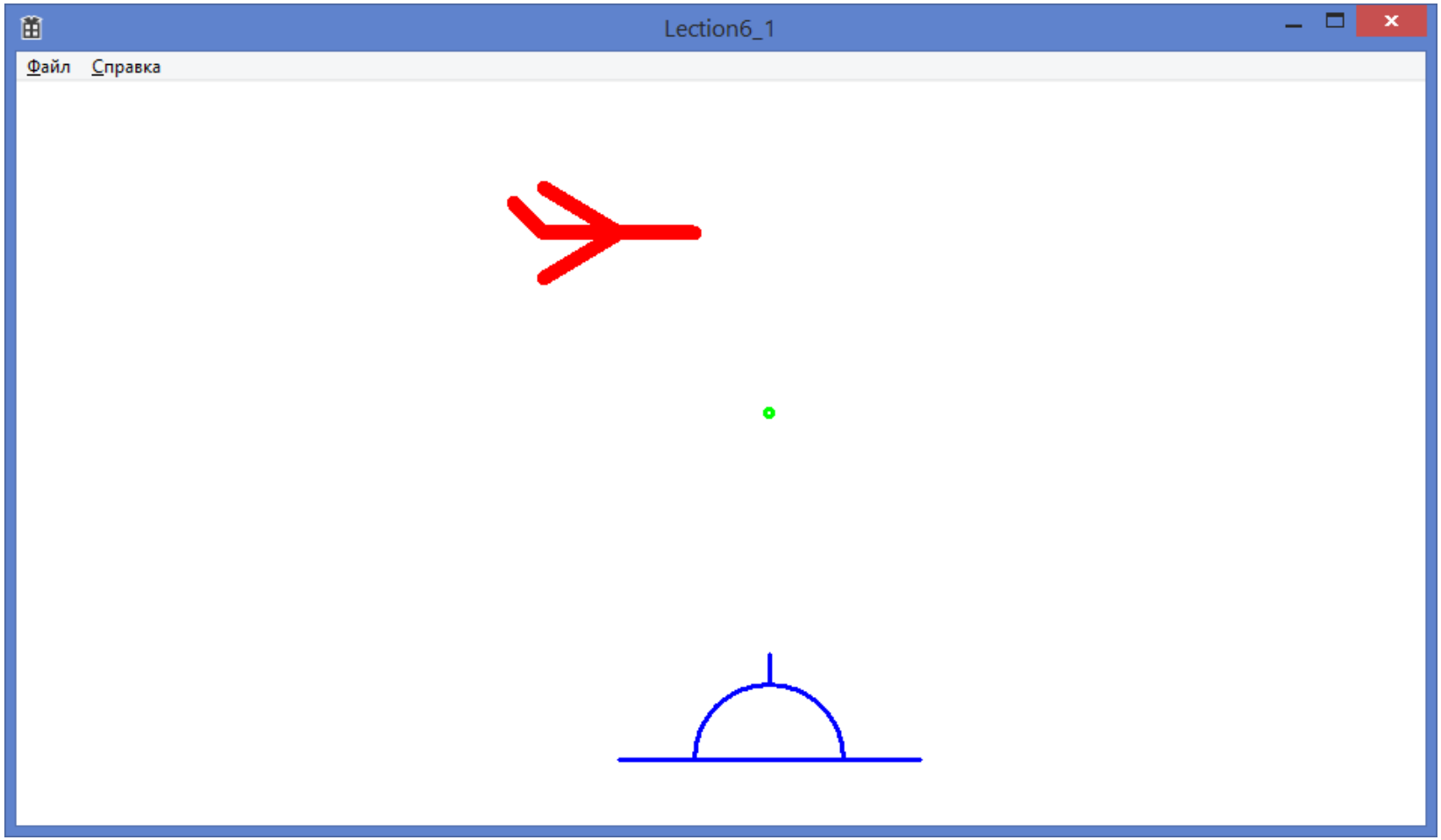
https://msdn.microsoft.com/ru-ru/library/whbyts4t.aspx

Пример

enum cardsuit { CLUBS, DIAMONDS, HEARTS, SPADES };

https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D1%87%D0%B8%D1%81%D0%BB%D1%8F%D0%B5%D0%BC%D1%8B%D0%B9_%D1%82%D0%B8%D0%BF

Делаем игру «Меткий зенитчик»



Константы и подключаемые файлы

```
// Lection6_1.cpp: определяет точку входа для приложения.
//
#include "stdafx.h"
#include "Lection6_1.h"

#define _USE_MATH_DEFINES
#include <math.h>

#define MAX_LOADSTRING 100

// Глобальные переменные:
HINSTANCE hInst;
WCHAR szTitle[MAX_LOADSTRING];
WCHAR szWindowClass[MAX_LOADSTRING];
```

Функция WndProc (1)

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT message,
WPARAM wParam, LPARAM lParam)
{
        switch (message)
        {
        …
        case WM_CREATE:
                SetTimer(hWnd, 1, 100, 0);
                break;

        case WM_TIMER:
                CheckContact();
                MovePlane();
                MoveBullet();
                InvalidateRect(hWnd, NULL, TRUE);
                break;
```

Функция WndProc (2)

```
case WM_PAINT:
{
        PAINTSTRUCT ps;
        HDC hdc = BeginPaint(hWnd, &ps);
        // TODO: Добавьте сюда любой код …

        DrawPlane(hdc);
        DrawCannon(hdc);
        DrawBullet(hdc);

        EndPaint(hWnd, &ps);
}
break;
```

Функция WndProc (3)

```
case WM_KEYDOWN:
        switch (wParam)
        {
        case VK_LEFT:
                MoveCannonLeft();
                InvalidateRect(hWnd, NULL, TRUE);
                break;
        case VK_RIGHT:
                MoveCannonRight();
                InvalidateRect(hWnd, NULL, TRUE);
                break;
        case VK_RETURN:
                ShotCannon();
                InvalidateRect(hWnd, NULL, TRUE);
                break;
        }
        break;
…
```

Модель (глобальные данные) (1)

// Глобальные переменные:

…

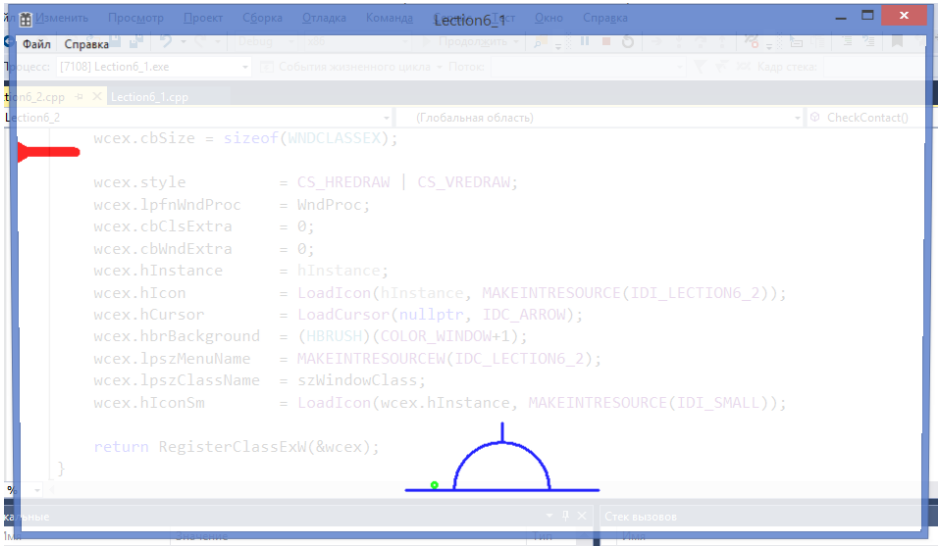
// самолет-мишень

int plane_x = 0;

int plane_y = 100;

int plane_vx = 10;

int plane_vy = 0;

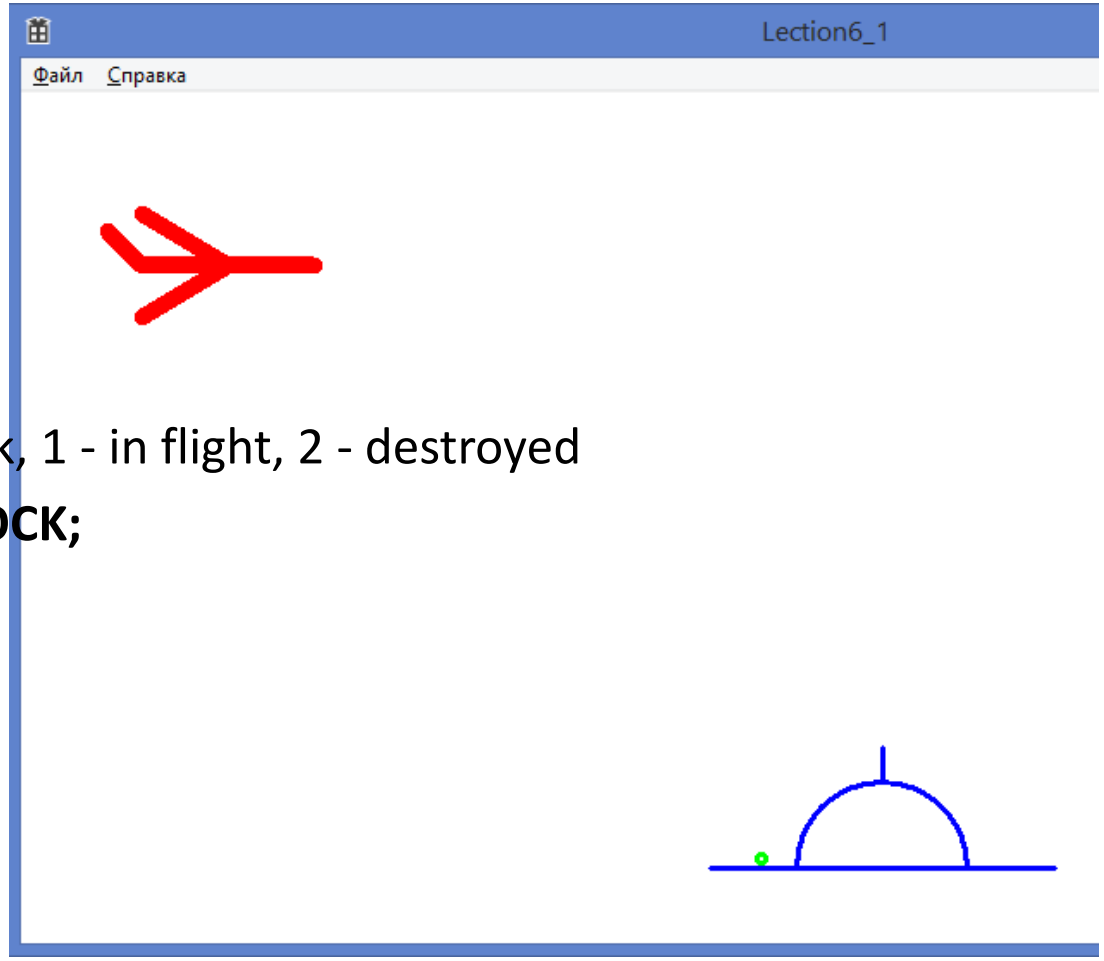


//int plane_state = 1; // 1 - in flight, 2 - destroyed

```
enum State {
        IN_STOCK,
        IN_FLIGHT,
        DESTROYED
};
State plane_state = IN_FLIGHT;
```

Модель (глобальные данные) (2)

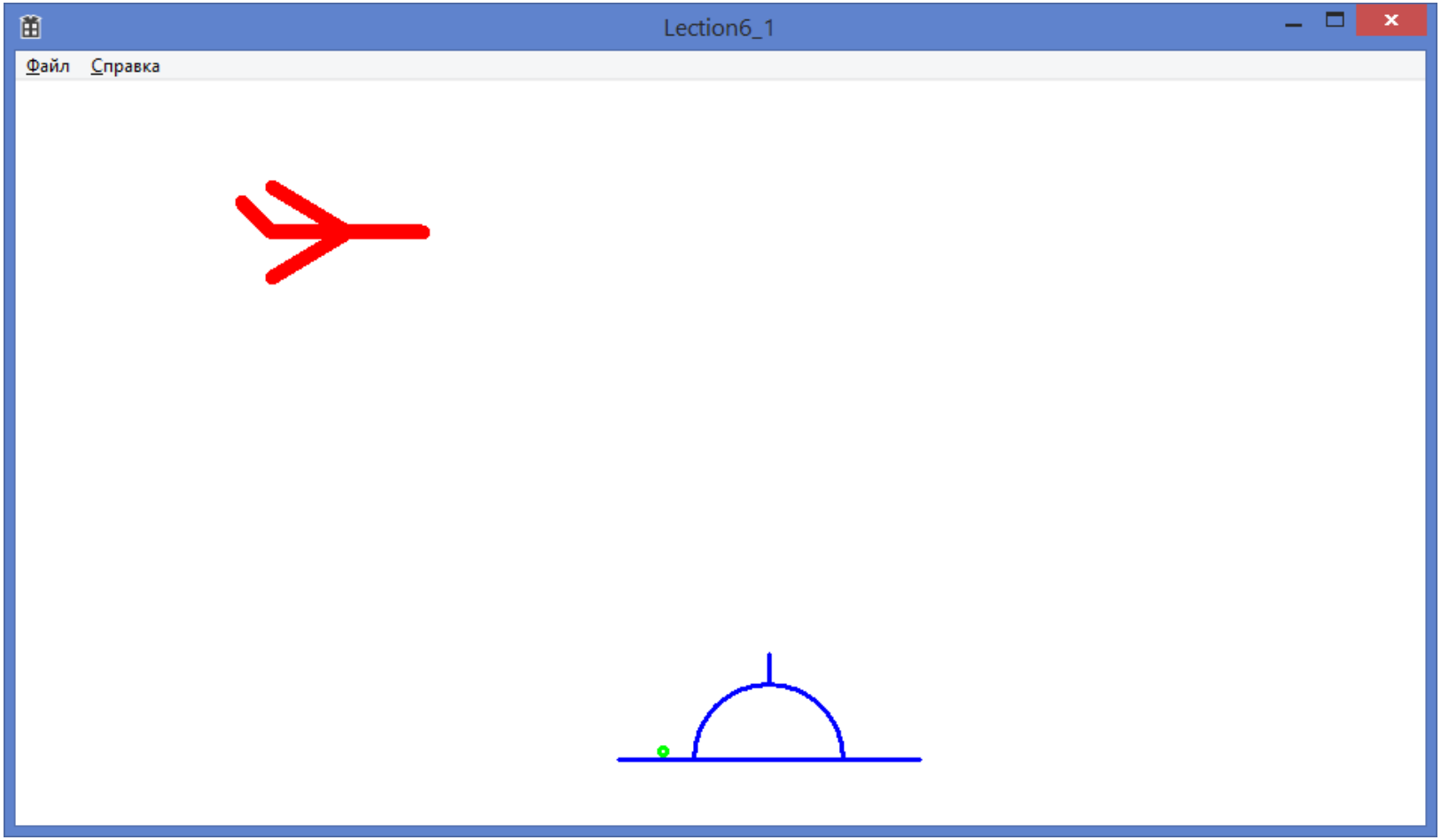
```
// пуля
int bullet_x = 430;
int bullet_y = 445;
int bullet_vx = 0;
int bullet_vy = 0;

//int state = 0; // 0 - in stock, 1 - in flight, 2 - destroyed
State bullet_state = IN_STOCK;
```



Модель (глобальные данные) (3)

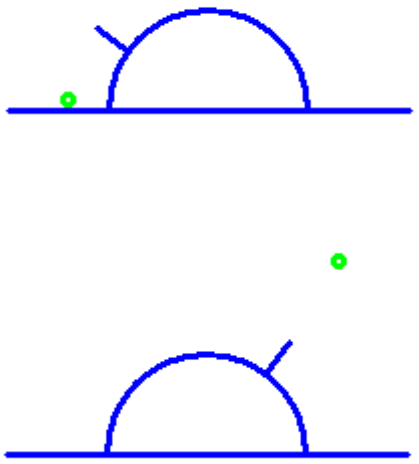
// зенитное орудие

double alpha = M_PI * 3.0 / 2.0; // направление стрельбы - СТРОГО ВВЕРХ



Зенитное орудие - отрисовка

```
void DrawCannon(HDC hdc) {
        HPEN hPen = CreatePen(PS_SOLID, 3, RGB(0, 0, 255));
        SelectObject(hdc, hPen);
        MoveToEx(hdc, 400, 450, NULL);
        LineTo(hdc, 600, 450);
        Arc(hdc, 450, 400,550, 500,      550,450, 450, 450);
        int r1 = 50;
        int r2 = 70;
        int x1 = 500 + (int)(cos(alpha) * r1);
        int x2 = 500 + (int)(cos(alpha) * r2);
        int y1 = 450 + (int)(sin(alpha) * r1);
        int y2 = 450 + (int)(sin(alpha) * r2);
        MoveToEx(hdc, x1, y1, NULL);
        LineTo(hdc, x2, y2);
}
```

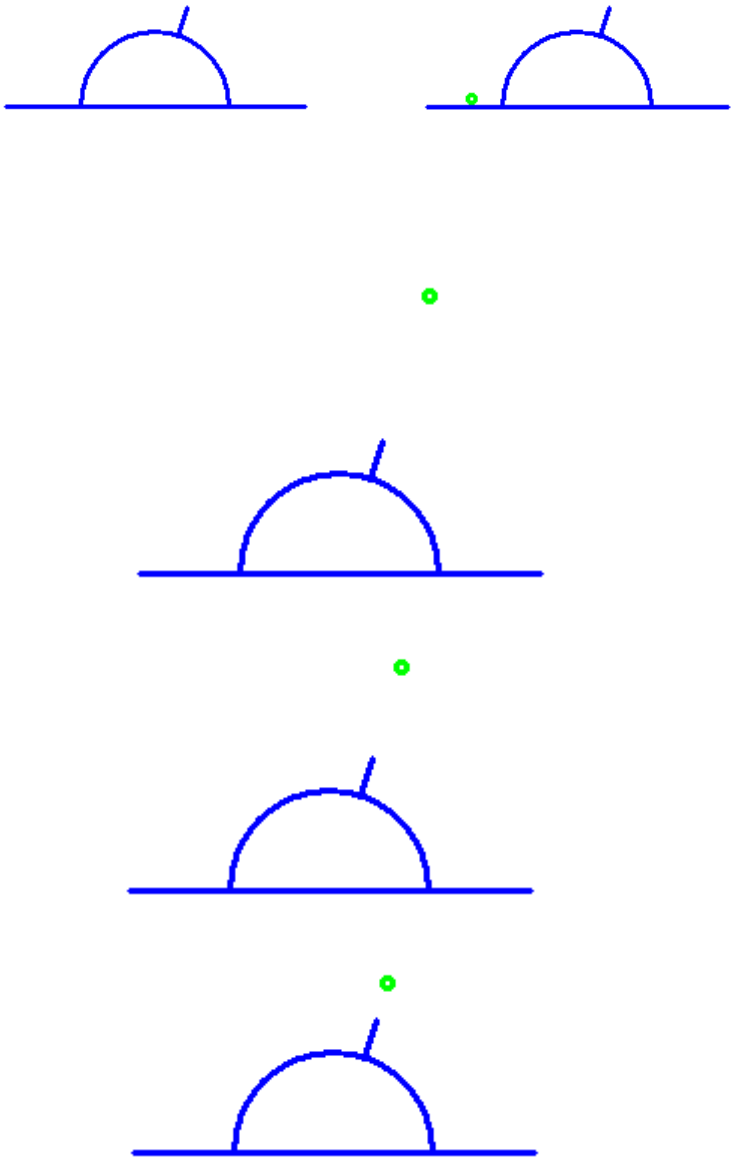


Зенитное орудие - управление

```
void MoveCannonRight() {
        if (alpha < M_PI * 1.95)
                alpha += M_PI / 20;
}

void MoveCannonLeft() {
        if (alpha > M_PI * 1.05)
                alpha -= M_PI / 20;
}
```

Зенитное орудие - выстрел

```
void ShotCannon() {
        if (bullet_state != IN_STOCK)
                return;
        int r2 = 70;
        int x2 = 500 + (int)(cos(alpha) * r2);
        int y2 = 450 + (int)(sin(alpha) * r2);

        int vr = 10;
        int vx = (int)(cos(alpha) * vr);
        int vy = (int)(sin(alpha) * vr);

        bullet_x = x2;
        bullet_y = y2;
        bullet_vx = vx;
        bullet_vy = vy;
        bullet_state = IN_FLIGHT;
}
```

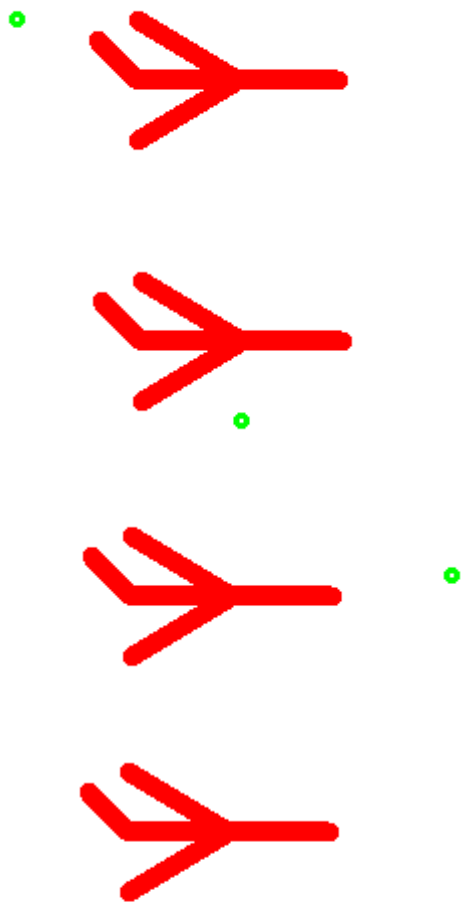


Самолет - отрисовка

```
void DrawPlane(HDC hdc) {
        HPEN hPen = NULL;
        if (plane_state == IN_FLIGHT)
                hPen = CreatePen(PS_SOLID, 10, RGB(255, 0, 0));
        if (plane_state == DESTROYED)
                hPen = CreatePen(PS_SOLID, 10, RGB(255, 255, 0));
        SelectObject(hdc, hPen);

        MoveToEx(hdc, plane_x + 50, plane_y, NULL);
        LineTo(hdc, plane_x - 50, plane_y);
        LineTo(hdc, plane_x - 70, plane_y - 20);

        MoveToEx(hdc, plane_x - 50, plane_y + 30, NULL);
        LineTo(hdc, plane_x, plane_y);
        LineTo(hdc, plane_x - 50, plane_y - 30);
}
```

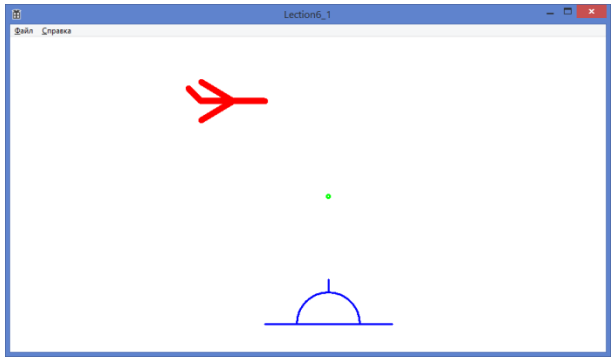
Самолет – проверка попадания в

```
int InsidePlane(int plane_x, int plane_y, int x, int y) {
        if (x < plane_x - 50)
                return 0;
        if (x > plane_x + 50)
                return 0;
        if (y < plane_y - 20)
                return 0;
        if (y > plane_y + 20)
                return 0;
        return 1;
}
```

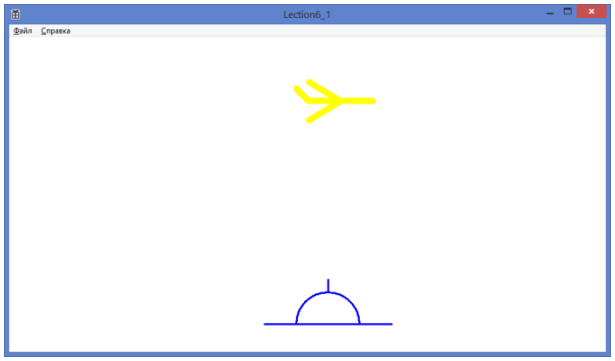


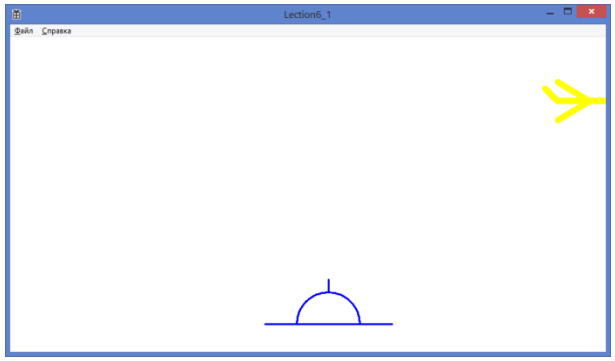
Самолет – перемещение и уничтожение

```
void MovePlane() {
        plane_x += plane_vx;
        plane_y += plane_vy;
}
```



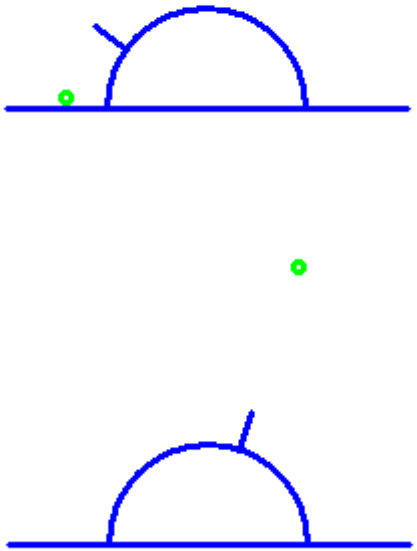

```
void DestroyPlane() {
        plane_state = DESTROYED;
}
```





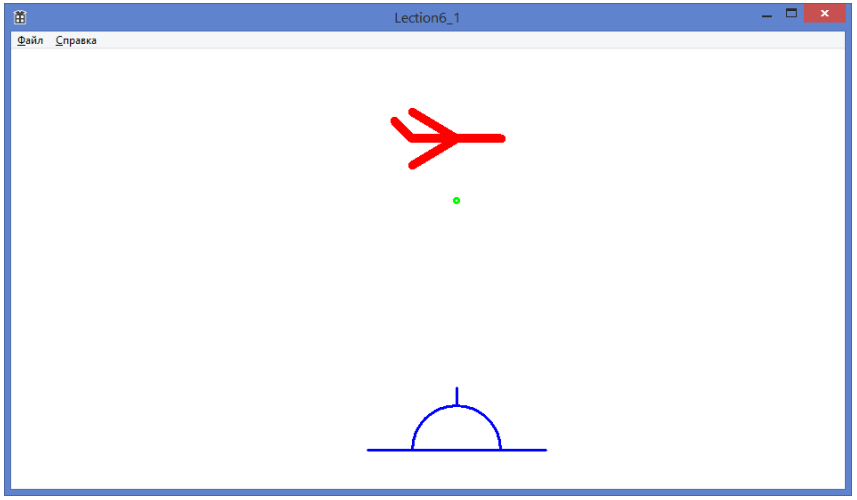
Пуля – отрисовка, перемещение и уничтожение

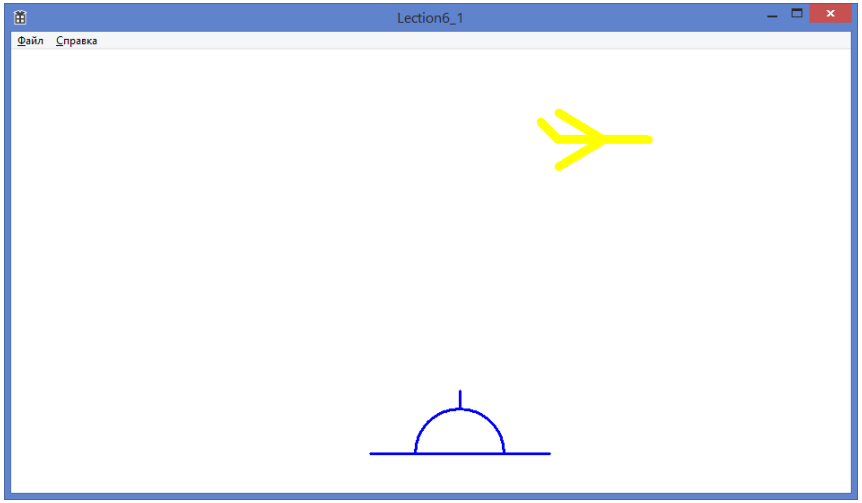
```
void DrawBullet(HDC hdc) {
        if (bullet_state == DESTROYED)
                return;

        HPEN hPen = CreatePen(PS_SOLID, 3, RGB(0, 255, 0));
        SelectObject(hdc, hPen);
        Ellipse(hdc, bullet_x - 3, bullet_y - 3, bullet_x + 3, bullet_y + 3);
}

void MoveBullet() {
        bullet_x += bullet_vx;
        bullet_y += bullet_vy;
}

void DestroyABullet() {
        bullet_state = DESTROYED;
}
```



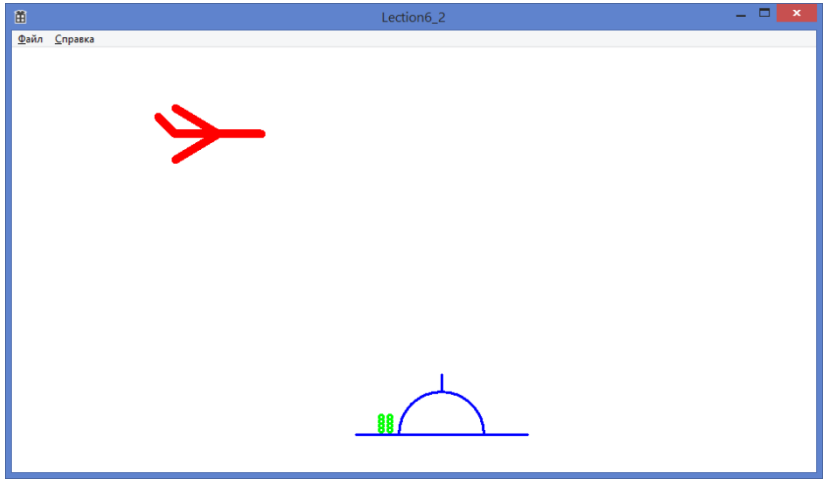
Пуля – проверка контакта пули и самолета

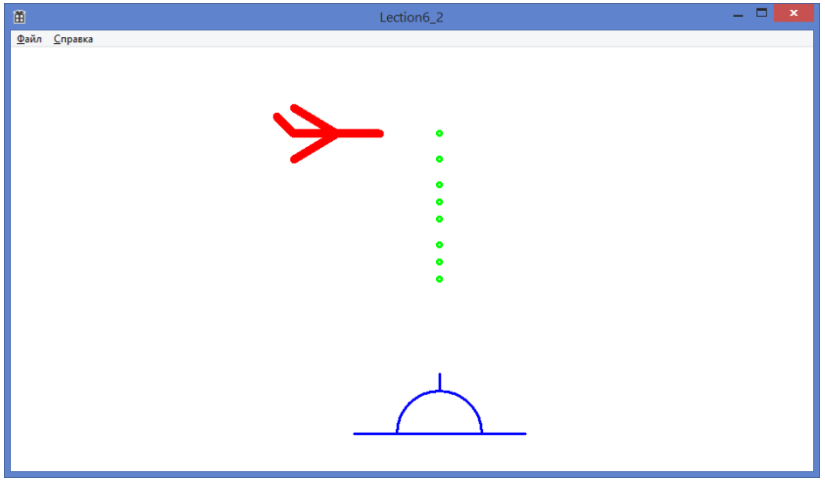
```
void CheckContact() {

        if (bullet_state == IN_FLIGHT
                && InsidePlane(plane_x, plane_y, bullet_x, bullet_y)) {

                DestroyPlane();
                DestroyABullet();

        }
}
```

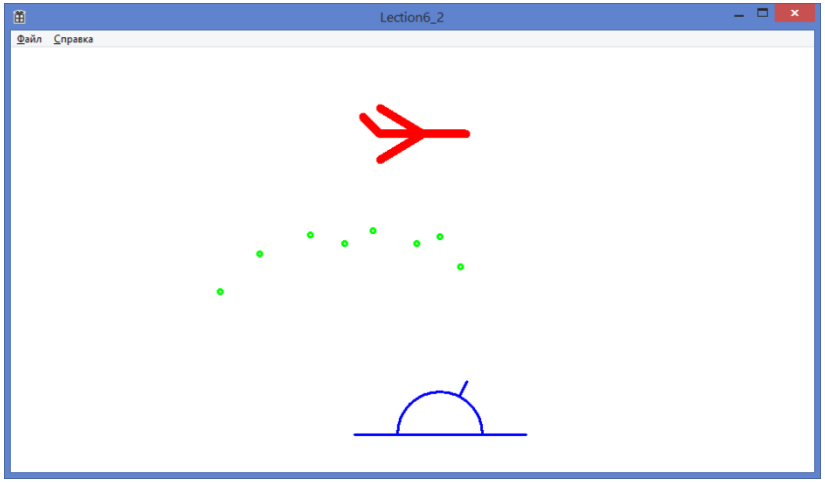


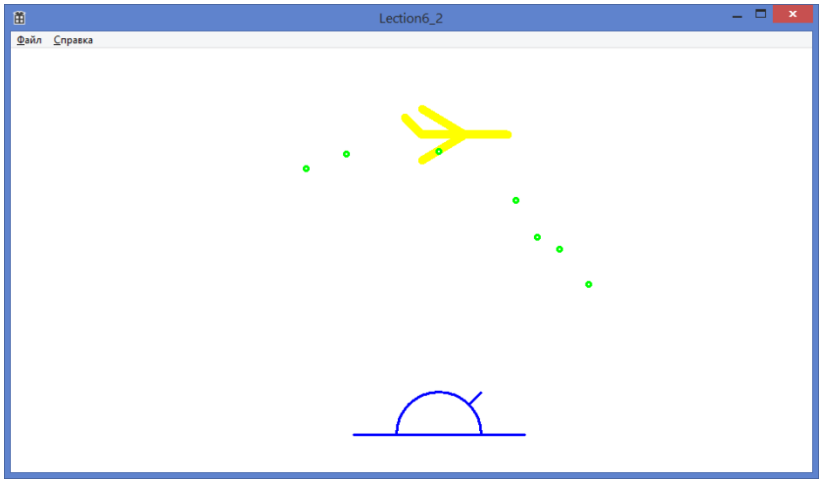


Версия 2.0: Стреляем очередями



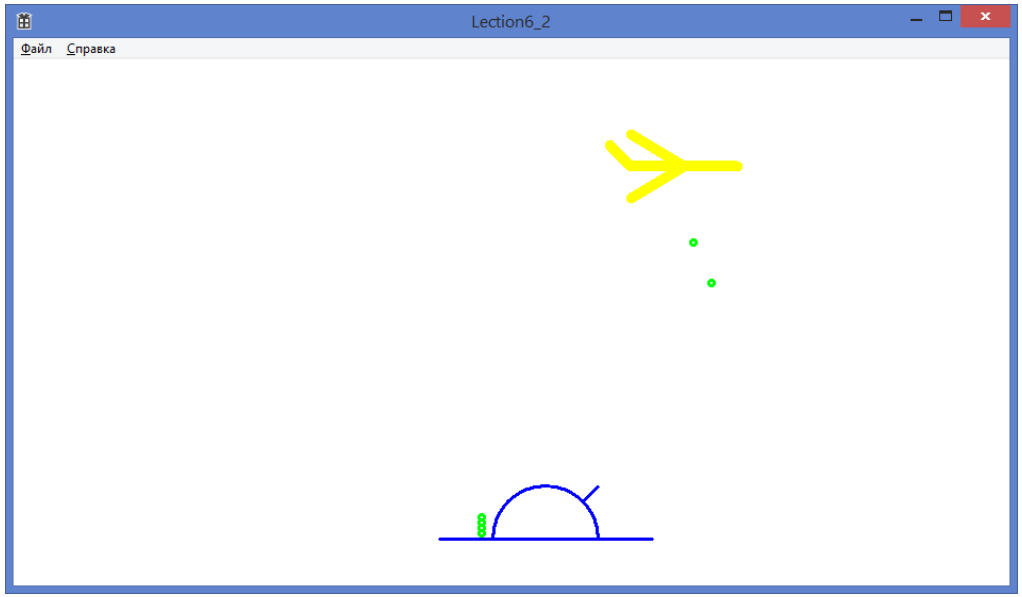






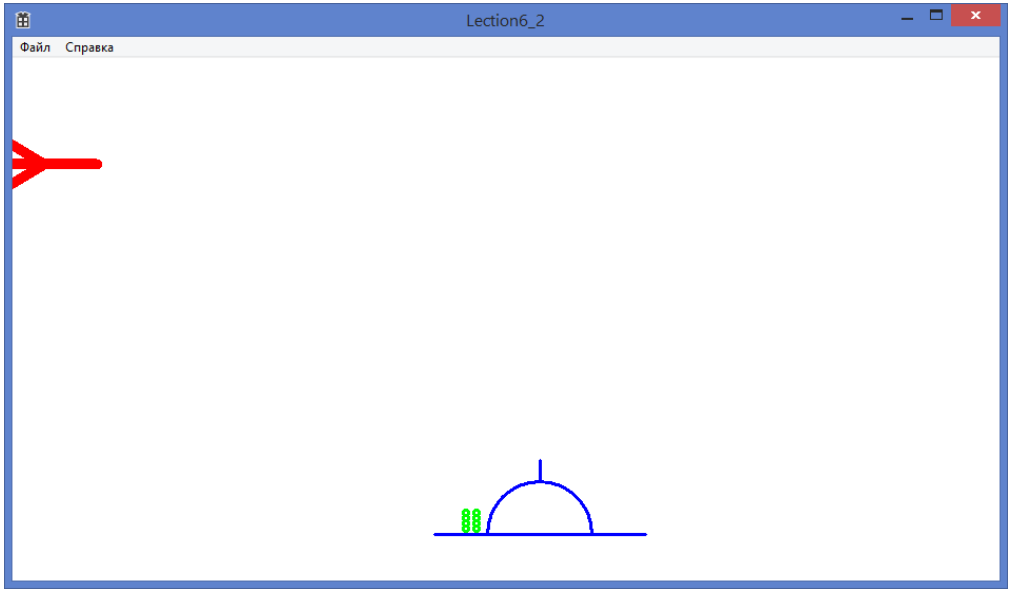
Версия 2.0: Стреляем очередями

- Что нужно изменить в программе, чтобы пуль было несколько?
- Как сделать, чтобы каждая пуля летела независимо от других?

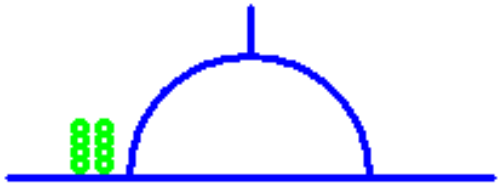


Модель (данные) (1)

```
// Глобальные переменные:
...
// самолет-мишень
int plane_x = 0;
int plane_y = 100;
int plane_vx = 10;
int plane_vy = 0;
 //int plane_state = 1; // 1 - in flight, 2 - destroyed
enum State {
        IN_STOCK,
        IN_FLIGHT,
        DESTROYED
};
State plane_state = IN_FLIGHT;
```



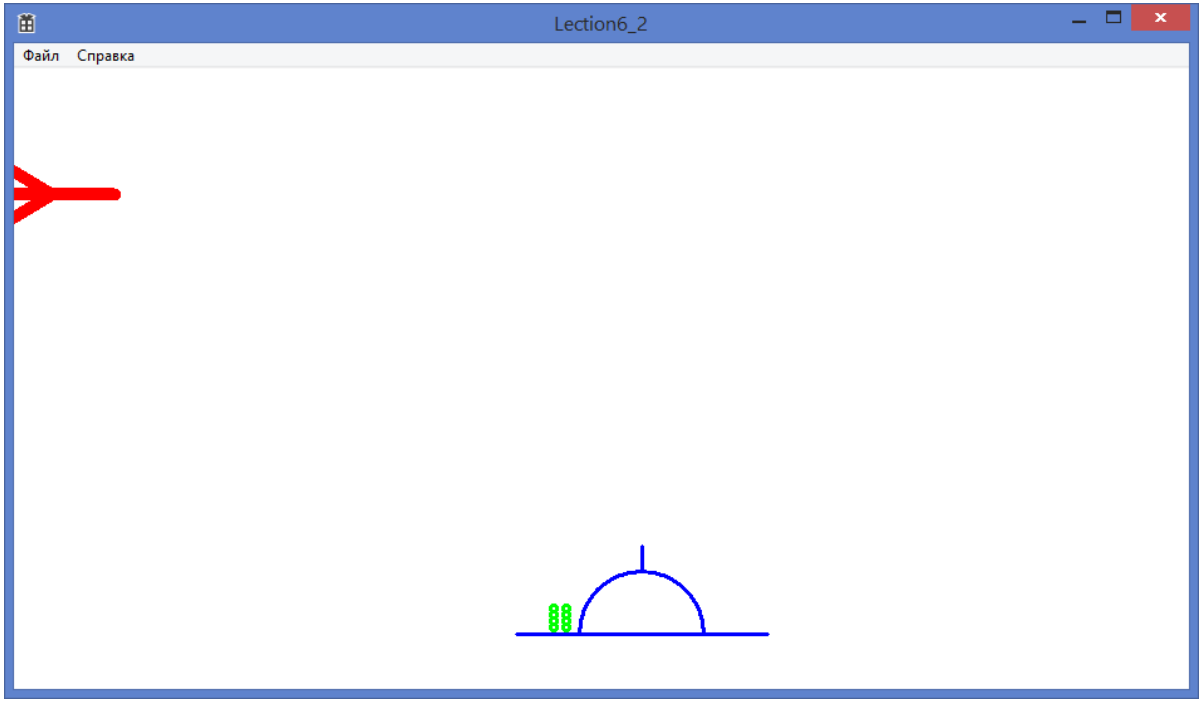
Модель (данные) (2)



```
// пуля
#define NUM_BULLET 8

int cnt_bullet = 0;
int bullet_x[NUM_BULLET] = { 430, 430, 430, 430,   440, 440, 440, 440 };
int bullet_y[NUM_BULLET] = { 430, 435, 440, 445,   430, 435, 440, 445 };

int bullet_vx[NUM_BULLET] = {0, 0, 0, 0, 0, 0, 0, 0};
int bullet_vy[NUM_BULLET] = {0, 0, 0, 0, 0, 0, 0, 0};

//int state = 0; // 0 - in stock, 1 - in flight, 2 - destroyed
State bullet_state[NUM_BULLET] = {
        IN_STOCK, IN_STOCK, IN_STOCK, IN_STOCK,
        IN_STOCK, IN_STOCK, IN_STOCK, IN_STOCK
};
```

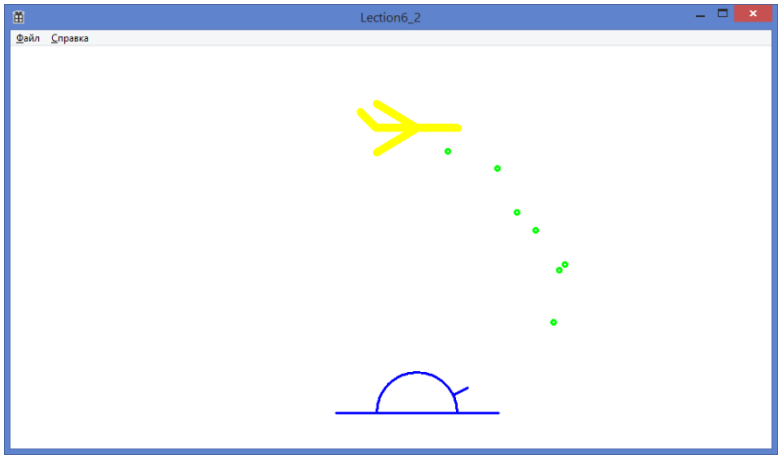
Модель (данные) (3)

```
// зенитное орудие
double alpha = M_PI * 3.0 / 2.0; // направление стрельбы - СТРОГО ВВЕРХ
```



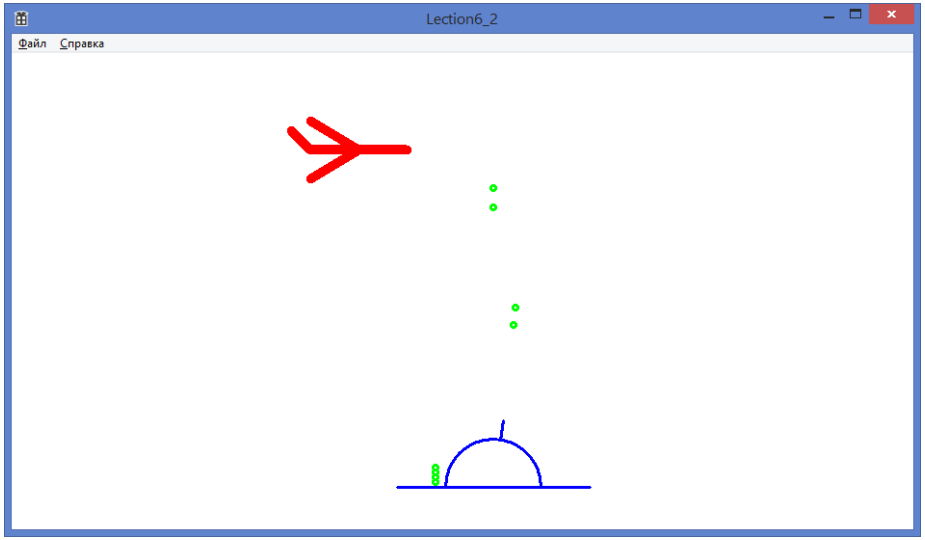
Много пуль: Отрисовка

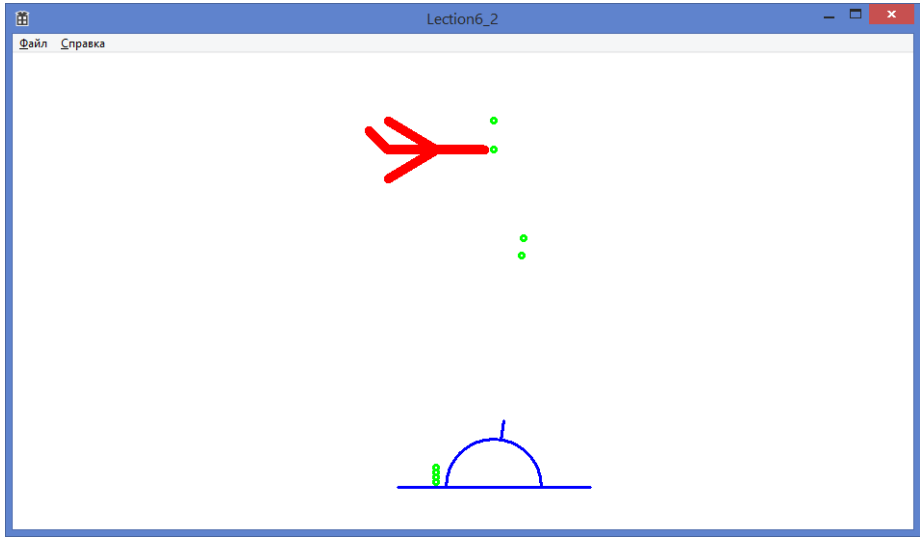
```
void DrawBullet(HDC hdc) {
        HPEN hPen = CreatePen(PS_SOLID, 3, RGB(0, 255, 0));
        SelectObject(hdc, hPen);

        int i = 0;
        while (i < NUM_BULLET) {
                if (bullet_state[i] != DESTROYED) {
                        Ellipse(hdc,
                                bullet_x[i] - 3, bullet_y[i] - 3,
                                bullet_x[i] + 3, bullet_y[i] + 3);
                }
                i++;
        }
}
```



Много пуль: перемещение всех пуль

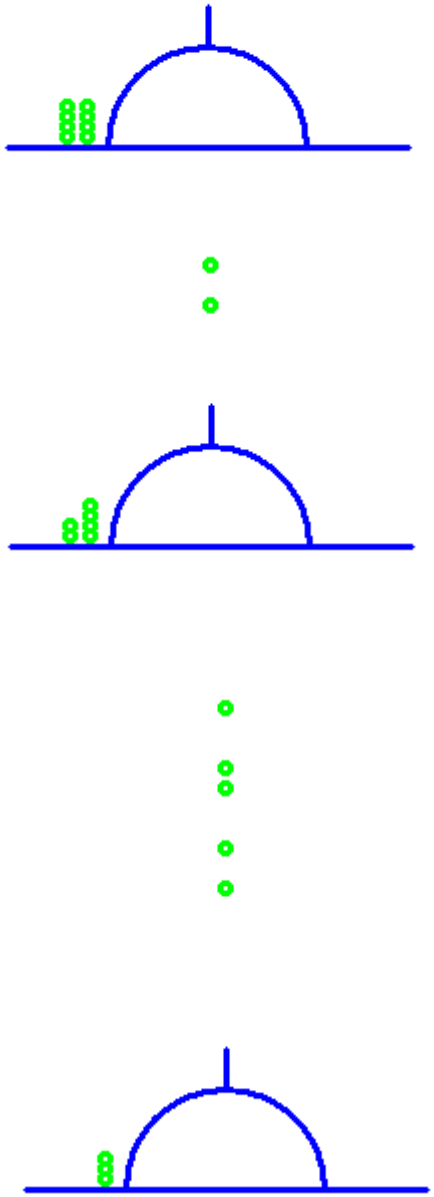
```
void MoveBullet() {
        int i = 0;
        while (i < NUM_BULLET) {
                bullet_x[i] += bullet_vx[i];
                bullet_y[i] += bullet_vy[i];
                i++;
        }
}
```





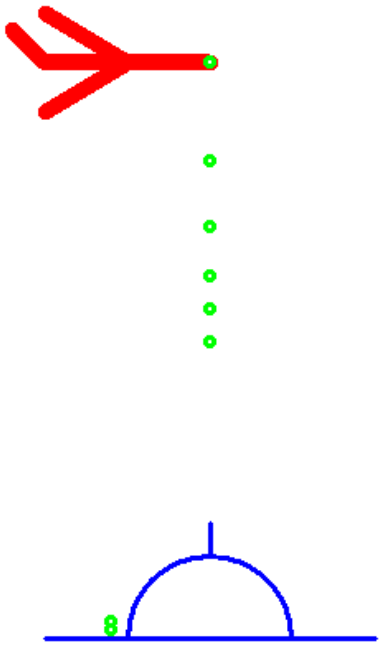
Много пуль: выстрел пушки

```
void ShotCannon() {
        if (cnt_bullet >= NUM_BULLET)
                return;
        int r2 = 70;
        int x2 = 500 + (int)(cos(alpha) * r2);
        int y2 = 450 + (int)(sin(alpha) * r2);
        int vr = 10;
        int vx = (int)(cos(alpha) * vr);
        int vy = (int)(sin(alpha) * vr);
        bullet_x[cnt_bullet] = x2;
        bullet_y[cnt_bullet] = y2;
        bullet_vx[cnt_bullet] = vx;
        bullet_vy[cnt_bullet] = vy;
        bullet_state[cnt_bullet] = IN_FLIGHT;
        cnt_bullet++;
}
```



Много пуль: проверка контакта

```
void DestroyABullet(int index_bullet) {
        bullet_state[index_bullet] = DESTROYED;
}
void CheckContact() {
        int i = 0;
        while (i < NUM_BULLET) {
                if (bullet_state[i] == IN_FLIGHT
                        &&
                        InsidePlane(plane_x, plane_y,
                                bullet_x[i], bullet_y[i])) {
                        DestroyPlane();
                        DestroyABullet(i);
                }
                i++;
        }
}
```



Демонстрация сборки проекта из кода презентации

Домашнее задание

1. ** Собрать игрушку из того, что есть в слайдах
2. *** Самолетов – много. Они добавляются постепенно.
3. **** Сделать игру со множеством объектов по собственному сценарию (Пример: сбор грибов – у каждого гриба свои координаты)

Источники информации

- Google.com