**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**
**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»**
**(БГТУ им. В.Г.Шухова)**

Расчетно-графическое задание по дисциплине «Архитектура вычислительных систем»

«Разработка Windows приложения на ассемблер»

Выполнил: студент группы ВТ-31 Макаров Д.С.

Проверил: Осипов О.В.

Белгород 2019

# Расчетно-графическое задание

## «Разработка Windows приложения на ассемблер»

**Цель работы:**разработать Windows приложение для анимации движения абсолютно упругого шарика внутри ограниченной двумерной области.

1. Создать на ассемблере оконное приложение для вывода на экран анимации.
2. Движение должно быть реализовано плавным и без мерцания, при необходимости использовать двойную буферизацию.
3. Раскрасить примитивы.

**Постановка задачи**: Шар представлен в виде координаты центра и радиуса, проекциями вектора скорости на оси Ox и OyОтскакивание шарика от стенок окна осуществляется инверсией знака одной из компонент скорости. При коллизии с другими примитивами инверсируются знаки обеих компонент скорости. Коллизия с линией проверяется по формуле функции прямой проходящей через 2 точки, а коллизия с окружностью по формуле принадлежности точки окружности.

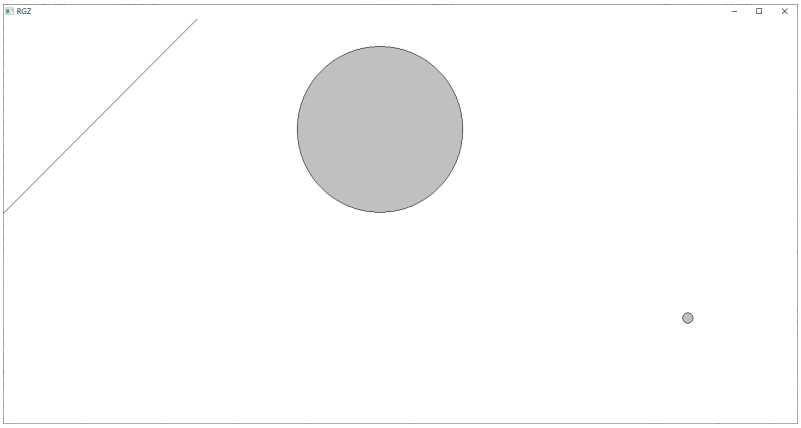


Рис. 1: Пример работы программы

# Приложение

## Содержимое файла rgz.asm

```
.386
.model flat,stdcall
option casemap:none
include C:\masm32\include\windows.inc
include C:\masm32\include\user32.inc
include C:\masm32\include\kernel32.inc
include C:\masm32\include\gdi32.inc
INCLUDE C:\masm32\include\msvcrt.inc

includelib C:\masm32\lib\msvcrt.lib
includelib C:\masm32\lib\user32.lib
includelib C:\masm32\lib\kernel32.lib
includelib C:\masm32\lib\gdi32.lib



WinMain proto :DWORD,:DWORD,:DWORD,:DWORD
.data?
hInstance dd ?
CommandLine dd ?
.data
ClassName db "SimpleWinClas",0
AppName db "RGZ",0
maxX SDWORD ?
maxY SDWORD ?

;Размеры окна
sx SDWORD ? ;по x
sy SDWORD ? ;по y

;Константы
c100 SDWORD 100  ;константа 100
c1 SDWORD 1      ;константа 1
c0 SDWORD 0      ;константа 0

;Временные переменные
temp SDWORD ?
t1 SDWORD ?
t2 SDWORD ?

;параметры шарика
ballX SDWORD 300 ;x
ballY SDWORD 300 ;y
ballR SDWORD 10  ;радиус шара
ballDX SDWORD 3  ;speed x
ballDY SDWORD 2  ;speed y
ballS SDWORD 2   ;speed t

;точки линии в углу экрана
left_top_t1_x SDWORD 350
left_top_t1_y SDWORD 0
left_top_t2_x SDWORD 0
left_top_t2_y SDWORD 350

;переменные для окружности
```

```
circle_x SDWORD 680
circle_y SDWORD 200
circle_r SDWORD 150

hdc SDWORD ?
memBit SDWORD ?
hBitmap SDWORD ?

eps SDWORD 0.001
ps        PAINTSTRUCT <?>
.code

;bool line_check(double x1,double y1,double x2,double y2,double x,double y)
;проверка принадлежности точки [x,y] прямой пересекающей точки [x1,y1],[x2,y2]
line_check proc
        pushad
                ;x1[esp+52]
                ;y1[esp+48]
                ;x2[esp+44]
                ;y2[esp+40]
                ;x[esp+36]
                ;y[esp+32]

                ;x-x1
                fild dword ptr [esp+40]
                fild dword ptr [esp+56]
                fsubp st(1),st(0)
        ;x2-x1
        fild dword ptr [esp+48]
        fild dword ptr [esp+56]
                fsubp st(1),st(0)
        fdivp st(1),st(0)

                ;y-y1
        fild dword ptr [esp+36]
                fild dword ptr [esp+52]
                fsubp st(1),st(0)
        ;y2-y1
        fild dword ptr [esp+44]
        fild dword ptr [esp+52]
                fsubp st(1),st(0)
        fdivp st(1),st(0)
        popad
        db 0dbh, 0f0h+1
        je check_eq
        check_neq:
            mov eax,0
            jmp check_end
        check_eq:
            mov eax,1
        check_end:
            FFREE st(0)
            FFREE st(1)
    ret 24
line_check endp

circle_check proc
        pushad
                ;a[esp+52]
                ;b[esp+48]
```

```
        ;r[esp+44]
                ;x[esp+40]
                ;y[esp+36]

                ;x-a
                fild dword ptr [esp+40]
                fild dword ptr [esp+52]
                fsubp st(1),st(0)
                fmul st(0),st(0)
        ;y-b
        fild dword ptr [esp+36]
        fild dword ptr [esp+48]
                fsubp st(1),st(0)
                fmul st(0),st(0)
        faddp st(1),st(0)

        fild dword ptr [esp+44]
                fmul st(0),st(0)
        popad
        db 0dbh, 0f0h+1
        jae check_eq
        check_neq:
            mov eax,0
            jmp check_end
        check_eq:
            mov eax,1
        check_end:
            FFREE st(0)
            FFREE st(1)
    ret 20
circle_check endp

start:
                invoke GetModuleHandle, NULL
                mov        hInstance,eax ;дескриптор экземпляра приложения
                ;Этот дескриптор содержит адрес начала кода программы в ее адресном
                ↪  пространстве.
                ;Дескриптор hInstance чаще всего требуется функциям, работающим с ресурсами
                ↪  программы.

                ;HINSTANCE hPrevInstance - дескриптор предыдущего экземпляра приложения.
                ;Этот дескриптор остался от старых версий Windows - скорее всего, вам он
                ↪  никогда не пригодится.
                ;HINSTANCE hPrevInstance=NULL в программе
                invoke GetCommandLine ;get
                mov        CommandLine,eax ;указатель на начало командной строки, введенной
                ↪  при запуске программы.
                ;int nCmdShow - это значение содержит желаемый вид окна (например, свернутый
                ↪  или развернутый)
                ;int nCmdShow=SW_SHOWDEFAULT
                invoke WinMain, hInstance, NULL, CommandLine, SW_SHOWDEFAULT
                invoke ExitProcess, eax

WinMain proc hInst:HINSTANCE, hPrevInst:HINSTANCE, CmdLine:LPSTR, CmdShow:DWORD
                LOCAL wc:WNDCLASSEX ;SS:0018FF68  Структура WNDCLASSEX содержит информацию о
                ↪  классе окна
                LOCAL msg:MSG ;SS:0018FF4C //структура сообщения
                LOCAL hwnd:HWND ;SS:0018FF78 Дескриптор - уникальный номер экземпляра окна
                ↪  программы
```

```
mov wc.cbSize, SIZEOF WNDCLASSEX ;Устанавливает размер этой структуры, в
↪  байтах
mov wc.style, CS_HREDRAW or CS_VREDRAW ;Устанавливает стиль(и) класса. Этот
↪  член структуры может быть любой комбинацией Стилей класса
mov wc.lpfnWndProc,OFFSET   WndProc ;Указатель на оконную процедуру.
mov wc.cbClsExtra,NULL ;Устанавливает число дополнительных байт, которые
↪  размещаются вслед за структурой класса окна
mov wc.cbWndExtra,NULL ;Устанавливает число дополнительных байтов, которые
↪  размещаются вслед за экземпляром окна
push hInstance ;Дескриптор экземпляра, который содержит оконную процедуру для
↪  класса.
pop wc.hInstance
mov wc.hbrBackground, COLOR_WINDOW+1;Дескриптор кисти фона класса.
mov wc.lpszMenuName ,NULL ;Указатель на символьную строку с символом конца
↪  строки ('\0')
mov wc.lpszClassName, OFFSET ClassName ;Указатель на символьную строку с нулем
↪  в конце или  атом

invoke LoadIcon,hInst,500
mov wc.hIcon, eax ;Дескриптор значка класса

mov wc.hIconSm,eax ;Дескриптор маленького значка, который связан с классом
↪  окна
invoke LoadCursor,NULL,IDC_ARROW
mov wc.hCursor,eax ;Дескриптор курсора класса
invoke RegisterClassEx,addr wc ; регистрирует класс окна для последующего
↪  использования
;CreateWindowEx создает перекрывающее, выпрыгивающее или дочернее окно с
↪  расширенным стилем

;DWORD dwExStyle,      // улучшенный стиль окна
;LPCTSTR lpClassName,  // указатель на зарегистрированное имя класса
;LPCTSTR lpWindowName, // указатель на имя окна
;DWORD dwStyle,        // стиль окна
;int x,                // горизонтальная позиция окна
;int y,                // вертикальная позиция окна
;int nWidth,           // ширина окна
;int nHeight,          // высота окна
;HWND hWndParent,      // дескриптор родительского или окна владельца
;HMENU hMenu,          // дескриптор меню или ID дочернего окна
;HINSTANCE hInstance,  // дескриптор экземпляра прикладной программы
;LPVOID lpParam        // указатель на данные создания окна

;Если функция завершается успешно, возвращаемое значение - дескриптор
↪  созданного окна.
INVOKE CreateWindowEx,NULL,ADDR ClassName,ADDR AppName,\
        13303808,CW_USEDEFAULT,\
        CW_USEDEFAULT,CW_USEDEFAULT,CW_USEDEFAULT,NULL,NULL,\
        hInst,NULL
mov hwnd,eax ;дескриптор созданного окна.
invoke LoadMenu,hInst,600 ;Загружает поименованный ресурс меню
invoke SetMenu,hwnd,eax ;Устанавливает и перерисовывает меню окна как меню,
↪  определенное параметром Menu

invoke ShowWindow,hwnd,SW_SHOWNORMAL ;hwnd-Идентификатор окна
;Отображает или прячет окно образом, указанным параметром CmdShow
invoke UpdateWindow,hwnd  ;обновляет рабочую область заданного окна hwnd
↪  -дескриптор окна
.WHILE TRUE ;запустить цикл обработки сообщений
```

```
                                ;ADDR msg указатель на структуру сообщения,
                                ;в которую GetMessage вернет результат.
                                ;2 параметр -описатель окна, от которого GetMessage примет
                                ↪  сообщение
                                ;(NULL означает, что GetMessage принимает сообщения от всех
                                ↪  окон, принадлежащих потоку
                                ;3 параметр UINT wMsgFilterMin - наименьший идентификатор
                                ↪  сообщения, которое примет GetMessage
                                ;4 параметр UINT wMsgFilterMax - наибольший идентификатор
                                ↪  сообщения, которое примет GetMessage
                                ;(если в значениях параметров wMsgFilterMin и wMsgFilterMax
                                ↪  передать 0, функция будет принимать ВСЕ сообщения)
                                invoke GetMessage,ADDR msg,NULL,0,0
                                ;в eax вернулась -переменная состояния
                                .BREAK .IF (!eax)
                                invoke TranslateMessage, ADDR msg ;разрешить use клавиатуры
                                invoke DispatchMessage, ADDR msg ;веонуть управление windows
                .ENDW
                mov eax,msg.wParam ;Определяет дополнительную информацию о сообщении
                ret
WinMain endp
        ;
        ; Работа программы
        ;


;                HWND hwnd,дескриптор оконной процедуры, которая получает сообщение
;                UINT uMsg,Определяет сообщение.
;                WPARAM wParam,Определяет дополнительную информацию о сообщении
;                LPARAM lParam Определяет дополнительную информацию о сообщении
WndProc proc hWnd:HWND, uMsg:UINT, wParam:WPARAM, lParam:LPARAM
                ;####################################close window
                .if uMsg==WM_DESTROY ;посылается когда необходимо уничтожить окно
                ;Функция DeleteObject удаляет логическое перо, кисть, шрифт, точечную картинку,
                ↪  регион или палитру,
                ;освобождая все системные ресурсы, связанные с объектом
                        invoke DeleteObject,memBit;дескриптор графического объекта
                ;функция PostQuitMessage указывает системе, что поток сделал запрос на то,
                ↪  чтобы завершить свою работу
                ;параметр Определяет код завершения прикладной программы.
                        invoke PostQuitMessage, NULL
                ;####################################create window
                .elseif uMsg==WM_CREATE ;осылается тогда, когда программа запрашивает, вызовом
                ↪  какой функции CreateWindowEx ,должно быть создано окно
                        ;set timer
                        fild c1
                        fild c100
                        fdiv
                        fst temp
                        ;HWND hWnd, Дескриптор окна, которое связано с таймером
                        ;UINT_PTR nIDEvent, Указывает идентификатор таймера отличный от нуля
                        ;UINT uElapse,Указывает значение времени простоя, в миллисекундах.
                        ;TIMERPROC lpTimerFun Указатель на функцию, которая уведомляет, когда
                        ↪  значение времени простоя истекает
                        invoke SetTimer,hWnd,1,c1,0



                        ; get max size window
                        invoke GetSystemMetrics, SM_CXSCREEN
```

```
mov maxX, eax
invoke GetSystemMetrics,SM_CYSCREEN;Ширина и высота экрана в пикселях
mov maxY, eax

; Получение контекста
;извлекает дескриптор дисплейного контекста устройства
invoke GetDC, hWnd ;hWnd дескриптор окна
mov hdc, eax
;Рисовать будем в памяти
;создает контекст устройства в памяти
invoke CreateCompatibleDC, hdc
mov memBit, eax
; CreateCompatibleBitmap  создает точечный рисунок
;HDC hdc,        // дескриптор DC
;int nWidth,     // ширина рисунка, в пикселях
;int nHeight     // высота рисунка, в пикселях
invoke CreateCompatibleBitmap, hdc, maxX, maxY
mov hBitmap, eax
;Если функция завершается успешно, возвращаемое значение - дескриптор
;совместимого точечного рисунка (аппаратно-зависимая точечная картинка
↪  (DDB)


;SelectObject выбирает объект в заданный контекст устройства (DC).
;Новый объект заменяет предыдущий объект того же самого тип
invoke SelectObject, memBit, hBitmap

; ######################################### Отрисовка
.elseif uMsg==WM_PAINT ; система или другое приложение делает запрос
↪  на закрашивание части окна приложения
        ; Отрисовка элипсов
        invoke PatBlt, memBit, 0, 0, maxX, maxY, WHITENESS
        ;BOOL PatBlt(
        ;HDC hdc,       // дескриптор контекста устройства (DC)
        ;int nXLeft,    // x-координата верхнего левого угла
        ↪  прямоугольника
        ;int nYLeft,    // y-координата верхнего левого угла
        ↪  прямоугольника
        ;int nWidth,    // ширина прямоугольника
        ;int nHeight,   // высота прямоугольника
        ;DWORD dwRop    // код растровой операции
        invoke GetStockObject, c1;извлекает дескриптор одного из
        ↪  предопределенных (стандартных) перьев, кистей, шрифтов или
        ↪  палитр
        invoke SelectObject, memBit, eax

        ;----------------------Отрисовка
        ↪  примитивов--------------------------------
                        fld circle_y
                        fld circle_r
                        fadd
                        fstp temp
                        push temp

                        fld circle_x
                        fld circle_r
                        fadd
                        fstp temp
                        push temp
```

```
                fld circle_y
                fld circle_r
                fsub
                fstp temp
                push temp

                fld circle_x
                fld circle_r
                fsub
                fstp temp
                push temp

                push memBit
call Ellipse

                push NULL
                push left_top_t1_x
                push left_top_t1_y
                push memBit
                call MoveToEx

                push left_top_t2_x
                push left_top_t2_y
                push memBit
call LineTo

                fld ballY
                fld ballR
                fsub
                fstp temp
                push temp

                fld ballX
                fld ballR
                fsub
                fstp temp
                push temp

                fld ballY
                fld ballR
                fadd
                fstp temp
                push temp

                fld ballX
                fld ballR
                fadd
                fstp temp
                push temp

                push memBit
call Ellipse

;HWND hwnd,            // дескриптор окна
;LPPAINTSTRUCT lpPaint // информация об окрашивании
invoke BeginPaint, hWnd, offset ps
mov hdc, eax
;BOOL BitBlt(
;HDC hdcDest, // дескриптор целевого DC
;int nXDest,  // x-коорд. левого верхнего угла целевого
↪  прямоугольника
```

```
        ;int nYDest,  // y-коорд. левого верхнего угла целевого
        ↪  прямоугольника
        ;int nWidth,  // ширина целевого прямоугольника
        ;int nHeight, // высота целевого прямоугольника
        ;HDC hdcSrc,  // дескриптор исходного DC
        ;int nXSrc,   // x-коорд. левого верхнего угла исходного
        ↪  прямоугольника
        ;int nYSrc,   // y-коорд. левого верхнего угла исходного
        ↪  прямоугольника
        ;DWORD dwRop  // код растровой операции
        invoke BitBlt, hdc, 0, 0, sx, sy, memBit, 0, 0, SRCCOPY
        invoke EndPaint, hWnd, offset ps ;отмечает конец окрашивания в
        ↪  заданном окне

.elseif uMsg==WM_SIZE ;посылается окну после того, как его размер
↪  изменился
        xor eax, eax
        mov ax, word ptr lParam
        mov sx, eax
        xor eax, eax
        mov eax, lParam
        shr eax, 16
        mov sy, eax

.elseif uMsg==WM_TIMER
;WM_TIMER является низкоприоритетным

;---------------------------- ПРОВЕРКА НА КОЛЛИЗИИ
↪  ------------------------------------
;Проверка на коллизии с горизонтальной границей окна c0 - верхняя
↪  граница, sx - нижняя
mov eax,ballX
.IF(c0>=eax || sx<=eax )
        fild ballDX
        FCHS
        fistp ballDX
.ENDIF

;Проверка на коллизии с вертикальной границей окна c0 - левая граница,
↪  sx - правая
mov eax,ballY
.IF(c0>=eax || sy<=eax )
        fild ballDY
        FCHS
        fistp ballDY
.ENDIF

;Проверка на коллизии с линией
xor eax,eax
push left_top_t1_x
push left_top_t1_y
push left_top_t2_x
push left_top_t2_y
push ballX
push ballY
call line_check
.IF(eax == 1)
        fild ballDX
        FCHS
        fistp ballDX
```

```
                        fild ballDY
                        FCHS
                        fistp ballDY
                .ENDIF

                ;Проверка на коллизии с окружностью
                xor eax,eax
                push circle_x
                push circle_y
                push circle_r
                push ballX
                push ballY
                call circle_check
                .IF(eax == 1)
                        fild ballDX
                        FCHS
                        fistp ballDX
                        fild ballDY
                        FCHS
                        fistp ballDY
                .ENDIF

                ;----------------------------- Перемещение шарика
                ↪  ----------------------------------------
                ; Прибавляем dx
                fild ballX
                fild ballDX
                fild ballS
                fmulp st(1),st
                faddp st(1),st
                FRNDINT
                fistp ballX
                ; Прибавляем dy
                fild ballY
                fild ballDY
                fild ballS
                fmulp st(1),st
                faddp st(1),st
                FRNDINT
                fistp ballY

                invoke InvalidateRect, hWnd, 0, 0  ;Указывает прямоугольник для
                ↪  перерисовки окна
                ;HWND hWnd,                // указатель на окно
                ;CONST RECT *lpRect,        // прямоугольник перерисовки
                ;BOOL bErase                // режим перерисовки
                .else
                        invoke DefWindowProc, hWnd, uMsg, wParam, lParam
                        ret
            .endif
        xor eax , eax
        ret
WndProc endp
end start
```