МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ**

**ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»**

**(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Расчетно-графическое задание №8

по дисциплине: Архитектура вычислительных систем

тема: «Разработка Windows приложения на Ассемблере»

Выполнил: ст. группы ПВ-31 Донцов А.А.

Проверил: Осипов О.В.

Белгород 2019 г.

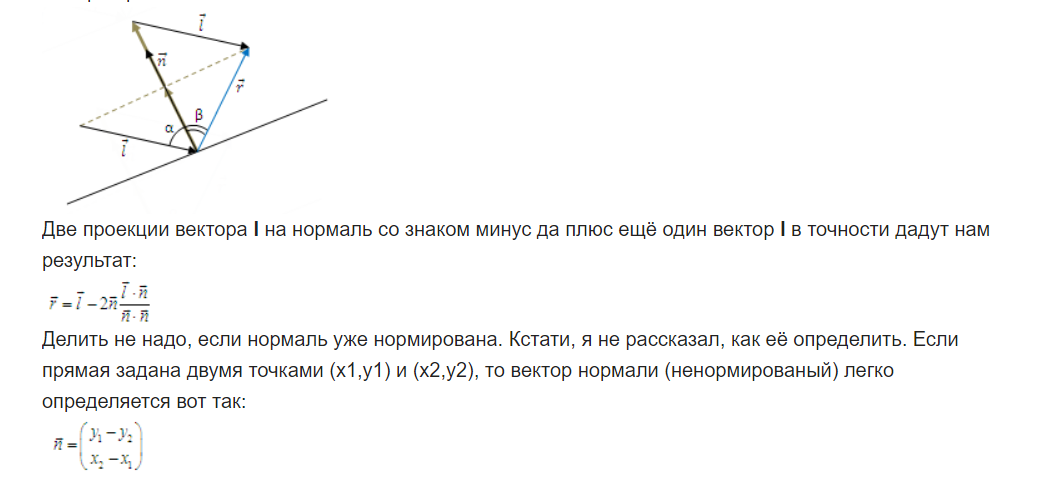
**Цель работы:** разработать Windows приложение для анимации движения абсолютно круглого шарика внутри ограниченной двумерной области.

**Постановка задачи:**

1. Создать на ассемблере оконное приложение для вывода на экран анимации следующего вида: (в автофигурах выбрать картинку)
2. Движение должно быть реализовано плавным и без мерцания. При необходимости использовать двойную буферизацию.
3. Раскрасить примитивы.

Алгоритм:

<https://habr.com/ru/post/105882/>



**Код программы:**

.386

.MODEL FLAT, STDCALL

OPTION CASEMAP: NONE

include c:\masm32\include\windows.inc

include c:\masm32\include\user32.inc

include c:\masm32\include\gdi32.inc

include c:\masm32\include\kernel32.inc

includelib c:\masm32\lib\user32.lib

includelib c:\masm32\lib\gdi32.lib

includelib c:\masm32\lib\kernel32.lib

.DATA?

hInstance DD ?

.DATA

className DB "MyAsmClass", 0

titleName DB "Balls", 0

startWidth DD 600

startHeight DD 600

clientWidth DD 800

clientHeight DD 800

backgroundColor DD 0FF670Ch

ballsColor DD 0FFFFFFh

ballstopX DD 0.0

ballstopY DD 0.0

ballnull DD 0.0

ballpart DD 2.0

ballX DD 400.0

ballY DD 400.0

ballRadius DD 50

ballVectorX DD 2.0

ballVectorY DD 5.0

min1 DD -1.0

NormVectX DD 0.0

NormVectY DD 0.0

BasVectX DD 0.0

BasVectY DD 0.0

cofX DD 0.0

cofY DD 0.0

;lenBC DD 0.0

;lenAB DD 0.0

;lenAC DD 0.0

x1 DD 0.0

y1 DD 0.0

;two DD 2.0

timer DD 1

;null2 DD 0.25

.CODE

moveBalls PROC

FLD ballX

FADD ballVectorX

FSTP ballX

FLD ballY

FADD ballVectorY

FSTP ballY

RET

moveBalls ENDP

handleWallCollisions PROC

FILD clientWidth

FSUB ballX

FILD ballRadius

FCOMPP

FSTSW AX

SAHF

JA collision\_1X

FLD ballX

FILD ballRadius

FCOMPP

FSTSW AX

SAHF

JBE collision\_2\_if

collision\_1X:

FLD ballVectorX

FCHS

FSTP ballVectorX

collision\_2\_if:

FILD clientHeight

FSUB ballY

FILD ballRadius

FCOMPP

FSTSW AX

SAHF

JA collision\_1Y

FLD ballY

FILD ballRadius

FCOMPP

FSTSW AX

SAHF

JBE collision\_circle

jmp collision\_1Y

collision\_circle:

FLD ballX

FLD ballX

FMULP ST(1), ST(0)

FLD ballY

FLD ballY

FMULP ST(1), ST(0)

FADDP ST(1), ST(0) ; расстояние между точками

FSQRT

FILD clientHeight

FLD ballpart

FDIVP ST(1), ST(0)

FIADD ballRadius

FCOMPP

FSTSW AX

SAHF

JAE collision\_circle2

JMP collision\_end

collision\_circle2:

;FLD ballX

;FMUL ballX

;FLD ballY

;FMUL ballY

;FADDP ST(1), ST(0)

;FLD clientHeight

;FMUL clientHeight

;FMUL null4

;FCOMPP

;FSTSW AX

;SAHF

;JAE collision\_circle2 ; >=

FLD ballX

FMUL min1

FSTP NormVectX

FLD ballY

FMUL min1

FSTP NormVectY

FLD NormVectX

FMUL NormVectX

FLD NormVectY

FMUL NormVectY

FADDP ST(1), ST(0)

FSQRT

FLD NormVectX

FDIV ST(0), ST(1)

FSTP NormVectX

FLD NormVectY

FDIVRP ST(1), ST(0)

FSTP NormVectY

FLD NormVectY

FMUL min1

FSTP BasVectX

FLD NormVectX

FSTP BasVectY

FLD NormVectX

FMUL ballVectorX

FLD NormVectY

FMUL ballVectorY

FADDP ST(1), ST(0)

FMUL min1

FSTP cofX

FLD BasVectX

FMUL ballVectorX

FLD ballVectorY

FMUL BasVectY

FADDP ST(1), ST(0)

FSTP cofY

FLD NormVectX

FMUL cofX

FSTP NormVectX

FLD NormVectY

FMUL cofX

FSTP NormVectY

FLD BasVectX

FMUL cofY

FSTP BasVectX

FLD BasVectY

FMUL cofY

FSTP BasVectY

FLD NormVectY

FLD BasVectY

FADDP ST(1), ST(0)

FSTP ballVectorY

FLD NormVectX

FLD BasVectX

FADDP ST(1), ST(0)

FSTP ballVectorX

;FLD ballX

;FLD ballVectorX

;FSUBP ST(1), ST(0) ; ST(0) = начало вектора (по х) х1

;FSTP x1

;FLD ballY

;FLD ballVectorY

;FSUBP ST(1), ST(0) ; ST(0) = начало вектора (по y) у1

;FSTP y1

;найти длину BC

;FLD ballVectorX

;FMUL ballVectorX

;FLD ballVectorY

;FMUL ballVectorY

;FADDP ST(1), ST(0) ; сумма в st(0)

;FSQRT ; корень в st(0)

;FSTP lenBC ; длина BC

;найти cos alpha

;FLD ballVectorX

;FMUL min1 ; st(0) = vectX \* (-1)

;FLD ballX ; ST(0) = ballX, ST(1) = vectX \* (-1)

;FMULP ST(1), ST(0) ; st(0)

;FLD ballVectorY ;st(0), st(1)

;FMUL min1 ; st(0) = vectY \* (-1)

;FLD ballY ; ST(0) = ballY, ST(1) = vectY \* (-1)

;FMULP ST(1), ST(0) ; st(0), st(1)

;FADDP ST(1), ST(0) ; st(0) = числитель;

;FLD ballVectorX ; st(0)= ballVectorX, st(1) = числитель

;FMUL ballVectorX ; в квадр

;FLD ballVectorY ; st(0)= ballVectorY, st(1)= ballVectorX, st(2) = числитель

;FMUL ballVectorY ; в квадр

;FADDP ST(1), ST(0) ; st(0), st(1)

;FSQRT ; корень в st(0)

; st(0), st(1)

;FLD ballX

;FMUL ballX ; в квадр

;FLD ballY

;FMUL ballY ; в квадр

;FADDP ST(1), ST(0)

;FSQRT ; корень в st(0)

;FMULP

;FDIVP ST(1),ST(0) ; st(0) = cos alpha

;найти AB (длинв нормали)

;FMUL lenBC

;FSTP lenAB ; длина AB

; найти длину АС

;FLD lenBC

;FMUL lenBC

;FLD lenAB

;FMUL lenAB

;FSUBP ST(1), ST(0)

;FSQRT

;FSTP lenAC

; ищем точку x2

;FLD lenAC

;FMUL lenAC ; st(0) = AC^2

;FLD x1

;FMUL x1 ; st(0)=x1^2, st(1) = AC^2

;FMUL min1

;FADDP ST(1), ST(0)

;FLD y1

;FMUL y1

;FMUL min1

;FADDP ST(1), ST(0)

;FLD lenAB

;FMUL lenAB

;FMUL min1

;FADDP ST(1), ST(0)

;FLD ballX

;FMUL ballX

;FADDP ST(1), ST(0)

;FLD ballY

;FMUL ballY

;FADDP ST(1), ST(0)

;FLD ballY

;FLD y1

;FADDP ST(1), ST(0)

;FMUL two

; ищем y2

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;FLD ballY

;FCOMPP

;FSTSW AX

;SAHF

;JA collision\_1Y;ballY >= ballX

;JBE collision\_1X

;FLD ballVectorX

;FCHS

;FLD ballVectorY

;FCHS

;FSTP ballVectorX

;FSTP ballVectorY

;FLD ballnull

;FST ballVectorX

;FSTP ballVectorY

jmp collision\_end

collision\_1Y:

FLD ballVectorY

FCHS

FSTP ballVectorY

collision\_end:

RET

handleWallCollisions ENDP

handleWindowMessage PROC hWindow:HWND, message:UINT, wParam:WPARAM, lParam:LPARAM

local windowRect:RECT

local hDC:HDC

local hCompatibleDC:HDC

local hBitmap:HBITMAP

local ps:PAINTSTRUCT

local backgroundBrush:HBRUSH

local ellipseBrush:HBRUSH

local ellipsePen:HPEN

local left:DWORD

local top:DWORD

local right:DWORD

local bottom:DWORD

MOV EBX, message

.IF message == WM\_CREATE

invoke SetTimer, hWindow, timer, 1, NULL

.ELSEIF message == WM\_DESTROY

invoke KillTimer, hWindow, timer

invoke PostQuitMessage, NULL

.ELSEIF message == WM\_PAINT

invoke GetClientRect, hWindow, ADDR windowRect

MOV EAX, windowRect.right

SUB EAX, windowRect.left

MOV clientWidth, EAX

MOV EAX, windowRect.bottom

SUB EAX, windowRect.top

MOV clientHeight, EAX

invoke BeginPaint, hWindow, ADDR ps

MOV hDC, EAX

invoke CreateCompatibleDC, hDC

MOV hCompatibleDC, EAX

invoke CreateCompatibleBitmap, hDC, clientWidth, clientHeight

MOV hBitmap, EAX

invoke SelectObject, hCompatibleDC, hBitmap

invoke CreateSolidBrush, backgroundColor

MOV backgroundBrush, EAX

invoke FillRect, hCompatibleDC, ADDR windowRect, backgroundBrush

invoke DeleteObject, backgroundBrush

invoke CreatePen, PS\_SOLID, 3, ballsColor

MOV ellipsePen, EAX

invoke CreateSolidBrush, ballsColor

MOV ellipseBrush, EAX

invoke SelectObject, hCompatibleDC, ellipsePen

invoke SelectObject, hCompatibleDC, ellipseBrush

FLD ballX

FISUB ballRadius

FISTP left

FLD ballY

FISUB ballRadius

FISTP top

FLD ballX

FIADD ballRadius

FISTP right

FLD ballY

FIADD ballRadius

FISTP bottom

invoke Ellipse, hCompatibleDC, left, top, right, bottom

FILD clientHeight

FLD ballpart

FDIVP ST(1), ST(0)

FLD ballstopX

FSUBRP ST(1), ST(0)

FISTP left

FILD clientHeight

FLD ballpart

FDIVP ST(1), ST(0)

FLD ballstopY

FSUBRP ST(1), ST(0)

FISTP top

FILD clientHeight

FLD ballpart

FDIVP ST(1), ST(0)

FLD ballstopX

FADDP ST(1), ST(0)

FISTP right

FILD clientHeight

FLD ballpart

FDIVP ST(1), ST(0)

FLD ballstopY

FADDP ST(1), ST(0)

FISTP bottom

invoke Ellipse, hCompatibleDC, left, top, right, bottom

invoke DeleteObject, ellipsePen

invoke DeleteObject, ellipseBrush

invoke SetStretchBltMode, hDC, COLORONCOLOR

invoke BitBlt, hDC, 0, 0, clientWidth, clientHeight, hCompatibleDC, 0, 0, SRCCOPY

invoke DeleteDC, hCompatibleDC

invoke DeleteObject, hBitmap

invoke EndPaint, hWindow, ADDR ps

.ELSEIF message == WM\_TIMER

invoke handleWallCollisions

; invoke handleBallsCollision

invoke moveBalls

invoke InvalidateRect, hWindow, NULL, FALSE

.ELSE

invoke DefWindowProc, hWindow, message, wParam, lParam

RET

.ENDIF

XOR EAX, EAX

RET

handleWindowMessage ENDP

registerClass PROC

local windowClass:WNDCLASSEX

MOV windowClass.cbSize, SIZEOF WNDCLASSEX

MOV windowClass.style, CS\_HREDRAW OR CS\_VREDRAW

MOV windowClass.lpfnWndProc, offset handleWindowMessage

MOV windowClass.cbClsExtra, NULL

MOV windowClass.cbWndExtra, NULL

PUSH hInstance

POP windowClass.hInstance

MOV windowClass.hbrBackground, COLOR\_WINDOW + 1

MOV windowClass.lpszMenuName, NULL

MOV windowClass.lpszClassName, offset className

invoke LoadIcon, hInstance, IDI\_APPLICATION

MOV windowClass.hIcon, EAX

MOV windowClass.hIconSm, EAX

invoke LoadCursor, NULL, IDC\_ARROW

MOV windowClass.hCursor, EAX

invoke RegisterClassEx, ADDR windowClass

RET

registerClass ENDP

mainWindow PROC nCommandShow:DWORD

local hWindow:HWND

local message:MSG

invoke registerClass

invoke CreateWindowEx, NULL, offset className, offset titleName,

WS\_OVERLAPPEDWINDOW OR WS\_VISIBLE, CW\_USEDEFAULT, CW\_USEDEFAULT,

startWidth, startHeight, NULL, NULL, hInstance, NULL

MOV hWindow, EAX

invoke ShowWindow, hWindow, nCommandShow

invoke UpdateWindow, hWindow

.WHILE TRUE

invoke GetMessage, ADDR message, NULL, 0, 0

.BREAK .IF (!EAX)

invoke TranslateMessage, ADDR message

invoke DispatchMessage, ADDR message

.ENDW

MOV EAX, message.wParam

RET

mainWindow ENDP

START:

invoke GetModuleHandle, NULL

MOV hInstance, EAX

invoke mainWindow, SW\_SHOWDEFAULT

invoke ExitProcess, NULL

END START

