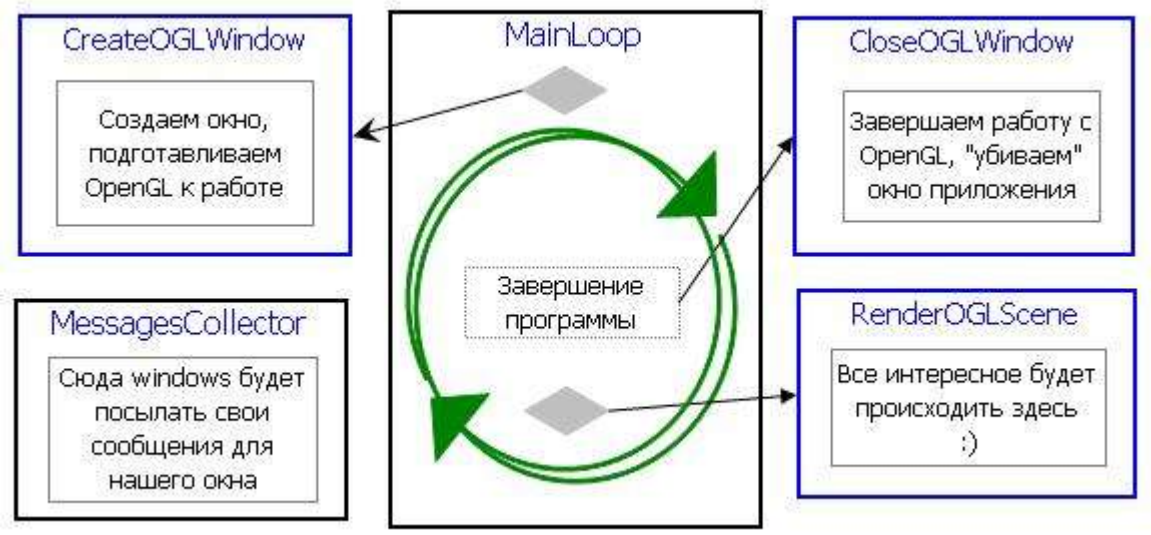
GLчатай, открой личико!

В предыдущей части (masm.opengl.chapters.introduction) мы занимались теорией, а сегодня пришло время практики. Давайте разбираться.



Скелет нашей первой программы.

Синими рамками обведены процедуры нашей программы, в которых мы будем обращаться к подпрограммам API OpenGL. В принципе больше комментировать здесь нечего. Если вы хорошо поработали над уроками Win32API от Iczelion'a, остальная часть схемы (подпрограммы в черных рамках) не должна вызвать недоумения. Если это не так, то работу над Win32API придется продолжить. Остальным слушателям предлагаю перейти к следующей главе.

Таможня дает "добро"

Давайте посмотрим на подпрограмму "CreateOGLWindow" повнимательнее. Ее название говорит само за себя - мы создаем окно, в

котором будут отображатся результаты нашего творчества в построении 3D виртуальной реальности. В следующей таблице расписаны шаги нашей подпрограммы, а также состояние OpenGL после каждого из них.

CreateOGLWindow	Состояние OpenGL
Создаем окно windows приложения	--неактивен--
Запрашиваем ОС Windows контекст устройства нашего окна	--неактивен--
Настраиваем контекст устройства нашего окна для работы с ним на OpenGL	--неактивен--
Сообщаем OpenGL о подготовленном контексте устройства	OpenGL создает контекст рендеринга (изображения), пригодный для работы с нашим окном (контекстом устройства)
Выбираем созданный OpenGL'ем контекст изображения (рендеринга) для построения 3D сцен командами API OpenGL	OpenGL готов к работе на выбранном контексте изображения - ожидает от нас команды для исполнения (путь в 3D мир открыт!)
MOV EAX, ECX (или любая другая)	OpenGL активен (ждет команды)

Сразу договоримся - мы будем смотреть на работу нашей программы с API OpenGL как на общение Клиента (наша программа) с Сервером (OpenGL). Представим на минутку, что мы пытаемся взломать секретный Сервер "OpenGL". Что нужно сделать сначала? Определить по какому сетевому протоколу нам удобнее приконектится к серверу! Допустим, мы выбрали протокол ABC_32бита. Это хорошо, но настоящие "кулхацкеры" не вламываются на секретные сервера со своего домашнего компьютера! Они вламываются с убитых PCишек (напомню, мы работаем в MASM) деревенских интернет-кафе.

Поддерживается ли на них нужный нам протокол ABC_32бита? Нужно обязательно проверить его наличие перед использованием. Если он не поддерживается, то выбираем другой протокол или отказываемся от идеи взлома. По-моему, пока все логично. На этот раз нам повезло – PCшка поддерживает протокол ABC_16бит (хуже, чем ABC_32бита, но хоть что-то). Мы посылаем команду серверу OpenGL с запросом на соединение по протоколу ABC_16бит и получаем сообщение, что сервер готов к соединению. Дальше дело

техники - сделать соединение активным. И всё, сервер OpenGL ожидает от нас команд для исполнения.

Не бойтесь контекстов устройств (device context) – они не кусаются. Вспомним наше детство: детский сад, строгую воспитательницу и неуемную страсть к рисованию. (на стенах, ватманах, на руках шариковой ручкой, на новеньком БМВ богатого дяди острым гвоздем). Но в детском саду у строгой воспитательницы всегда нужно было спрашивать ее разрешение перед тем, как расписывать что-нибудь – «Марья Ивановна (ОС Windows), можно я пририсую усы фломастером на вашей фотографии (в окне моей программы)?». А как иначе? За плохое поведение могут и из детского садика исключить («Ваша программа пыталась выполнить недопустимую операцию и будет закрыта» © WIndows), а там столько красивых игрушек!

Давайте договоримся, что под контекстом устройства (device context) мы будем понимать некий объект на котором нам не терпится что-либо нарисовать. В детском саду такими объектами (контекстами устройства) могли быть: лицо лучшего друга (зубной пастой во время «тихого часа»), стена в туалете, асфальтавая дорожка (цветными мелками – милое дело).

В мире Windows мы будем пользоваться 3мя такими объектами (контекстами устройств) – монитором, принтером и памятью компьютера. API OpenGL может отрисовывать наши картины не только на мониторе, но и на принтере (есть ограничения) и просто рисовать картинку в памяти компьютера (Off-screen rendering. Кстати, только в этом случае мы можем подправлять изображения OpenGL самостоятельно непосредственно в памяти.) Контекст изображения OpenGL (не путать с контекстом устройства) к Windows не имеет никакого отношения, это "внутренняя кухня" OGL.

Теперь, отягощенные знанием о контекстах устройсв Windows и контексте изображения OpenGL, еще раз изучите таблицу «CreateOGLWindow».

Что автор имел в виду под «протоколом», например «ABC_32бита»?. Структуру под симпатичным названием **PIXELFORMATDESCRIPTOR**.

```
; C
; © MSDN Library
typedef struct tagPIXELFORMATDESCRIPTOR { // pfd
    WORD  nSize;
    WORD  nVersion;
    DWORD dwFlags;
    BYTE  iPixelType;
    BYTE  cColorBits;
    BYTE  cRedBits;
    BYTE  cRedShift;
    BYTE  cGreenBits;
    BYTE  cGreenShift;
    BYTE  cBlueBits;
    BYTE  cBlueShift;
    BYTE  cAlphaBits;
    BYTE  cAlphaShift;
    BYTE  cAccumBits;
    BYTE  cAccumRedBits;
    BYTE  cAccumGreenBits;
    BYTE  cAccumBlueBits;
    BYTE  cAccumAlphaBits;
    BYTE  cDepthBits;
    BYTE  cStencilBits;
    BYTE  cAuxBuffers;
    BYTE  iLayerType;
    BYTE  bReserved;
    DWORD dwLayerMask;
    DWORD dwVisibleMask;
    DWORD dwDamageMask;
} PIXELFORMATDESCRIPTOR;
```

Я специально сделал фонт помельче, чтобы у вас не возникло желание с ним разбиратся прямо сейчас. На данном этапе просто насладитесь количеством разных параметров, которые мы сможем выбирать для работы с OpenGL.

Для чего конкретно нужна эта структура? **PIXELFORMATDESCRIPTOR** определяет формат пикселей графического изображения, например, сколько цветов оно может отображать (16бит или 32бита - вы неоднократно делали этот выбор, играя в компьютерные игры). Механизм работы со структурой предельно простой. Мы «заказываем» нужный нам формат пикселей с которыми хотели бы работать в 3D. Затем отправляем «заказ» ОС Windows. Система подбирает нам наиболее подходящий формат из списка тех, что поддерживаются «железом» на данном компьютере и возращает его порядковый номер(индекс). Мы сообщаем Windows, что будем работать в предложенном формате и настраиваем контекст устройства окна под него. Теперь мы уверены, что контекст устройства окна и контекст изображения OpenGL будут работать в одинаковых форматах пикселей, т.е. не будет ситуации, когда OpenGL пытается запихнуть 32бита туда, где могут поместится только 16бит. Логично?

MASM.OpenGL.Chapters – первая кровь

Вот как выглядит подпрограмма CreateOGLWindow.

```
;
;  MASMv7
;
;-----------------------------------------------------
        CreateOGLWindow proc
;-----------------------------------------------------
;  Резервируем место для индекса
;  формата пикселей, возращаемого Windows
LOCAL PixelFormat :DWORD

; Регистрируем класс нашего окна
invoke RegisterClassEx, ADDR wc
; Ошибка? – "Ошибка при регистрации окна"
; и выходим

; Создаем окно win32 приложения
; Для окна с OpenGL необходимы WS_CLIPSIBLINGS и WS_CLIPCHILDREN !
; Выставляем размеры окна 400 x 400
invoke CreateWindowEx,  NULL,
                        ADDR szWinClassName,
                        ADDR szWinName,
                        WS_CLIPSIBLINGS or WS_CLIPCHILDREN \
                        or WS_OVERLAPPEDWINDOW,
                        0, 0,
                        400,
                        400,
                        NULL,
                        NULL,
                        hInstance,
                        NULL
; Ошибка? –  "Ошибка при создании окна"
; и выходим
mov hWnd, eax

; Подпрграмма GetDC  (входит в Win32API) возвращает
; нам указатель на контекст устройства (КУ) нашего окна
; «Марьиванновна, можно мне порисовать на вашей фотографии?»
; Марьиванновна (с кислым лицом) – «Да на, рисуй. У меня еще есть».
invoke GetDC, hWnd
; Ошибка - "Не могу получить указатель КУ окна"
; и выходим
mov hDC, eax

; Подпрграмма ChoosePixelFormat входит в Win32API.
; В структуре pfd :PIXELFORMATDESCRIPTOR
; лежит наш «заказ» на пиксельный формат (ПФ),
; который мы хотели бы использовать в OpenGL.
; Просим Windows подобрать нам подходящий
; ПФ из поддерживаемых железом на выбранном
; контексте устройства - hDC
invoke ChoosePixelFormat, hDC, ADDR pfd
```

```
; Ошибка - "Нет подходящего пиксельного формата."
; и выходим
mov PixelFormat, eax

; Подпрграмма SetPixelFormat входит в Win32API.
; Настраиваем наш контекст устройства (hDC) на
; работу с подходящим пиксельным форматом, также
; сообщаем некоторую дополнительную информацию
; из нашей структуры pfd :PIXELFORMATDESCRIPTOR
; (windows добавит к выбранному ПФ «тонкие настройки»)
; "Соединение КУ с OpenGL возможно по протоколу ABC_XXbits»
invoke SetPixelFormat, hDC,PixelFormat, ADDR pfd
; Ошибка – "Установка ПФ не удалась."
; и выходим

; Подпрграмма wglCreateContext входит в API OpenGL (windows версия).
; Просим OpenGL создать у себя контекст изображения,
; который можно отобразить на контексте устройства нашего
; окна -- формат пикселей у КУ и КИ  одинаковый!
; "Подготовить OpenGL для коннекта по протоколу ABC_XXbits"
invoke wglCreateContext, hDC
; Ошибка – "Ошибка в создании контекста изображения."
; и выходим
mov hRC, eax

; Подпрграмма wglMakeCurrent входит в API OpenGL (windows версия).
; Делаем созданный котекст изображения (hRC) активным,
; сообщаем на каком контексте устройства (hDC) мы хотим видеть
; результаты выполнения наших комманд к OpenGL (сегодня мы рисуем
: на мониторе).  OpenGL готов к выполнению наших комманд, находится
: в режиме ожидания.
; Поздравляю, мы в OpenGLe – 3D открыт!
invoke wglMakeCurrent, hDC, hRC
; Ошибка - "Ошибка активизации КИ."
; и выходим

; Подпрграмма ShowWindow входит в Win32API.
; Являем OpenGL окно миру
invoke ShowWindow, hWnd, SW_SHOW

; Подпрграмма SetForegroundWindow входит в Win32API.
; Да не просто являем, а выпячиваем
invoke SetForegroundWindow, hWnd

; Подпрграмма SetFocus входит в Win32API.
; Да не просто являем и выпячиваем, а насильно
; приковываем к нему всеобщее внимание.
invoke SetFocus, hWnd

return TRUE
CreateGLWindow endp
```

Keep your country tidy! (англ.)

Как бы Вам понравилось, если на Вашей жилой площади совершенно чужие люди брали ваши кровные контекстные устройства и создавали разные контексты изображения, работали с ними в свое удовольствие, жутко при этом мусорили и, махнув ручкой на прощанье, убирались восвояси, не прибрав за собой? Вот и ОС Windows будет ужасно растроена, если мы не прибиремся.

Напомню, что, создавая наше первое OpenGL приложение, мы на время взяли для рисования контекст устройства окна. Затем по нашей просьбе добрый OpenGL выделил для программы частичку себя – контекст изображения. Настоящие джентельмены, поиграв в чужие игрушки, возвращают их хозяевам.

```
;  MASMv7
;
;-----------------------------------------------------
        CloseOGLWindow proc
;-----------------------------------------------------

; Есть ли в нашем распоряжении контекст изображения OpenGL?
.IF hRC != 0

  ; Подпрграмма wglMakeCurrent входит в API OpenGL (windows версия).
  ; Сообщаем OpenGL, что больше не хотим активного соединения с его
  ; сервером. OpenGL перестает быть активным, все наши комманды ему
  ; будут проигнорированы, хотя созданный для  программы контекст
  ; изображения все еще зарезервирован.
  invoke wglMakeCurrent, NULL, NULL

  ; Подпрграмма wglDeleteContext входит в API OpenGL (windows версия).
  ; Мы сообщаем OpenGL, что выделенный для нас контекст изображения
  ; нам больше не понадобится. Сервер приводит себя в порядок, а мы идем
  ; своей дорогой. «И разошлись как в море корабли»
  invoke wglDeleteContext, hRC
.ENDIF


; Подпрграмма ReleaseDC входит в Win32API.
; Возвращаем фотографию Марьивановне с глубокой благодарностью
invoke ReleaseDC, hWnd, hDC

; Окно нашей программе больше не нужно
invoke DestroyWindow, hWnd

; Убираем за собой
invoke UnregisterClass, ADDR szWinName, hInstance

; Рисуем поразительно реальный 3D образ Клаудии Шифер, в поный рост и
; совершенно голую.
ret
; Шутка.
KillOGLWindow endp
```

PFD_SUPPORT_OPENGL equ 020h

Мы узнали почти все, чтобы закончить этот отрывок из MASM.OpenGL.Chapters На сладкое осталась структура формата пикселей – pfd.

```
;
;  MASMv7
;
pfd PIXELFORMATDESCRIPTOR  { \
                    sizeof PIXELFORMATDESCRIPTOR,
                    1,
                    PFD_DRAW_TO_WINDOW or  PFD_SUPPORT_OPENGL,
                    PFD_TYPE_RGBA,
                    32,
                    0, 0, 0, 0, 0, 0,
                    0,
                    0,
                    0,
                    0, 0, 0, 0,
                    0,
                    0,
                    0,
                    0,
                    0,
                    0, 0, 0}
```

Нули нас пока не интересуют.

Единица это pfd.nVersion – версия структуры (да и такое бывает). nVersion всегда равна единице.

32 – это pfd.cColorBits, определяет кол-во битов на один пиксель (bbp), т.е. количество разнообразных цветов в которые можно этот пиксель покрасить. Я взял 32 (а мог бы 24 или 16, например), что даст мне возможность отобразить 4,294,967,295 цветов.

PFD_TYPE_RGBA (pfd. iPixelType)– сообщает Windows, что цвета пиксилей мы будем определять через Red Green Blue компоненты (подробнее в следующих главах)

И последним определяем элемент pdf. dwFlags:

PFD_DRAW_TO_WINDOW сообщает Windows о том, что мы будет рендерить (строить 3D сцены) в окно, а не в память, например.
PFD_SUPPORT_OPENGL – у меня нет информации.

В этой главе мы казалось бы ничего не сделали особенного. То, что получилось, Вы увидите, когда после **внимательнейшего** изучения исходника сами скомпилируете его, если надо – отловите ошибки, а затем запустите. Напоминаю, **СНАЧАЛА** изучаете исходник, **ПОТОМ** компилируете, выдерживаете одни сутки и в самую последнюю очередь запускаете. Так и только так.

Один мой хороший знакомый добился исключительных результатов за непостижимо короткий срок. Когда я попытался выяснить, в чем секрет и какими пособиями он пользовался, ZYtDHE грустно посмотрел в мою сторону и виновато промямлил «Рассыпался винчестер, а на старом не было компилятора». (!!!) Этот парень создавал программы, отлавливал свои ошибки в течении двух недель, так не разу и не запустив код на исполнение. «All errors are made by mistake» (c) ZYtDHE

Вы замечали, что первое впечатление от знакомства с чем-то новым - самое сильное? Если именно сейчас в эти первые минуты после прочтения текста вы постараетесь вникнуть в философию OpenGL, с самых азов, с вызова его первой подпрограммы, с инициализации его первой структуры, то остальной более продвинутый материал покажется вам логичным продолжением, вы увидите в хаосе нулей и единиц, байтов и двойных слов, в контекстах изображения удивительную красоту и гармонию. И тогда вы сможете импровизировать как хороший музыкант на любимом инструменте, а возможности API OpenGL будут ограниченны лишь Вашей фантазией.

Информация к размышлению

Так как исходник без комментариев и обработки возможных ошибок после вызова подпрограмм API (на их месте стоят мои комментарии «и выходим» ,смотри выше по тексту), предлагаю следующий конкурс для любознательных – доработайте исходник. Лучший будет опубликован в следующей главе)

Еще необходимо придупредить читателя о том, что откомпилируванный исходник может вообще не запуститься. Если это Ваш случай – отсается только порадоваться. Ведь потратив некоторое время на самостоятельный поиск и исправление ошибок вы лучше усвоите материал, и значит потратили время не зря. Вам может понадобится отладчик. К сожалению у меня сейчас при себе нет его намера телефона, а на email он не отвечает. Поищите его в интернете. Встретите Сафонову Асю, передавайте привет.

Автор работает в редакторе AsmEditor, который можно скачать отсюда (http://www.avtlab.ru/) Так же скачайте и установите базу подсветки синтаксиса, которую любезно предоставил vkim, разработчик VKDEBUG (входит в пакет MASMv7).

Основные возможности AsmEditor:

- подключение различных компиляторов (Ассемблер, Си и другие);
- настраиваемые схемы подсветки ключевых слов;
- быстрый переход между процедурами, функциями, метками;
- возможность подключения файлов помощи (например, win32.hlp) для вызова контекстной справки по выделенному в редакторе слову;
- работа с исходными кодами в DOS-кодировке без потери символов псевдографики;
- меню и дополнительные кнопки с функциями, назначаемыми пользователем;
- настраиваемые "горячие клавиши"; быстрый переход по номеру строки; "закладки"; сохранение позиции курсора и т.д.
- Не требует инсталляции, не вносит изменений в системные файлы и реестр.
- По умолчанию настроен на пакет MASM32

Дакже скачайте fraps (www.fraps.com) и испробуйте его на каких-нибудь программах.

Для развлечения, поиграйте с графическими режимами в своем любимом FPS, попереключайтесь с 16 на 32 цветность. Постарайтесь объяснить увиденное.

Просто вопросы на засыпку:

1) Если мы будем работать в режиме off-screen rendering'га (т.е. рисовать изображение в памяти компьютера, а не выводить на экран) сможем ли получить любой «заказанный» формат пикселей для роботы на этом контексте устройства (ограничений то по «железу» на память нет)?

2) Что такое wiggle?

3) Почему в нашей программе необходимо указывать WS_CLIPSIBLINGS и WS_CLIPCHILDREN при вызове CreateWindowEx?