Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Конструирование Программного Обеспечения (КПО)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему:

**ПРОГРАММНОЕ СРЕДСТВО «Slot car racing»**

БГУИР КП 1-40 01 01 423 ПЗ

Студент Мухин Д.С.

Руководитель Оношко Д.Е.

Минск 2024

СОДЕРЖАНИЕ

[Введение 5](#_Toc186115651)

[1 Анализ предметной области 6](#_Toc186115652)

[2 Проектирование прграммного средства 8](#_Toc186115655)

[2.1 Структура программы 8](#_Toc186115656)

[2.2 Проектирование интерфейса программного средства 8](#_Toc186115657)

[2.3 Проектирование функционала программного средства 10](#_Toc186115662)

[3 Разработка программного средства 16](#_Toc186115668)

[3.1 Основная механика 16](#_Toc186115669)

[3.2 Структура кнопки 19](#_Toc186115673)

[3.3 Структура кватерниона 21](#_Toc186115678)

[3.4 Работа с внешними источниками 22](#_Toc186115679)

[4 Тестирование программного средства 25](#_Toc186115682)

[5 Руковоство пользователя 27](#_Toc186115683)

[Заключение 33](#_Toc186115694)

[Список использованных источников 34](#_Toc186115695)

[Приложение А 35](#_Toc186115696)

# ВВЕДЕНИЕ

Slot Car Racing — это популярная форма развлечений, в которой небольшие игрушечные гоночные машинки перемещаются по специально созданным трекам с использованием электрического питания и направляющих дорожек (слотов). Эти игры появились в середине XX века и быстро завоевали популярность благодаря своей доступности, простоте и увлекательности. Управление машинками осуществляется игроками посредством пультов, регулирующих скорость движения, что делает Slot Car Racing сочетанием технической точности и соревновательного духа.

Особенность Slot Car Racing заключается в использовании физических ограничений трека, что добавляет стратегическую составляющую в игру. Игроки должны уметь правильно контролировать скорость, особенно на поворотах.

Сегодня, с развитием цифровых технологий, Slot Car Racing получили новую форму в виде виртуальных симуляторов. Компьютерные версии этих игр предлагают те же принципы управления и соревновательного процесса, но дополнены графическими и звуковыми эффектами, новыми игровыми механиками и возможностью многопользовательских режимов. Это позволяет не только сохранить классическую механику игры, но и обогатить её современными технологиями.

Виртуальные симуляторы Slot Car Racing становятся отличной платформой для изучения и реализации различных аспектов разработки программного обеспечения. Они включают в себя сложные расчёты физики движения, взаимодействие с графическим интерфейсом, а также разработку удобных инструментов для управления и обратной связи с пользователем. Такой проект предоставляет возможность объединить увлекательный игровой процесс с техническими навыками программирования, что делает его актуальным и востребованным в образовательной и профессиональной среде.

# АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## **Обзор аналогов**

В условиях растущего интереса к гоночным играм и симуляторам важно понимать существующие решения на рынке, их функциональность и особенности. Обзор аналогов позволит выявить сильные и слабые стороны уже доступных решений, а также определить, какие аспекты можно улучшить или адаптировать в новой игре.

Первым аналогом будущего программного средства является программное средство «Slot racing». Она позволяет игрокам управлять машинками по виртуальной рельсовой трассе с использованием сенсорного управления. Игра предлагает разнообразные трассы, несколько режимов, включая мультиплеер, что делает её доступной для широкой аудитории.

Игра очень увлекательная, так как она воспроизводит дух классических slot car гонок, но при этом остаётся простой для освоения даже для новичков. Однако высокие требования к качеству графики и физики в игре не позволяют насладится игрой на слабых устройствах.

Внешний вид данного программного средства представлен на рисунке 1.1.



Рисунок 1.1 – Программное средство «Slot racing»

В качестве второго программного средства рассмотрим «Virtual SlotCars». Программное средство предлагает широкий выбор трасс и машинок, а также возможность настроить различные параметры гонок. В игре доступен режим одиночного прохождения и мультиплеер, что делает её подходящей как для индивидуального развлечения, так и для коллективной игры. Кроме того, разработчики предусмотрели возможность создания собственных трасс с помощью встроенного редактора, что значительно расширяет потенциал игры и поддерживает интерес пользователей.

Однако, несмотря на множество плюсов, игра имеет и свои недостатки. Одним из них является довольно высокий порог вхождения из-за сложной физики движения, что может отпугнуть новичков.

Внешний вид программного средства представлен на рисунке 1.2.



Рисунок 1.2 – Программное средство «Solitaire»

Все эти примеры создают хорошую базу для дальнейшего развития гоночных игр, но хотелось бы увидеть более оптимизированные цифровые реализации. Интуитивно понятные и удобные версии могли бы объединить лучшие черты физических игр с преимуществами современных технологий, улучшая пользовательский опыт и привлекая более широкую аудиторию.

## **Постановка задачи**

Цель курсовой работы – разработка программного средства «Slot car racing» под платформу Windows. На основе проведённого анализа требуется реализовать следующие функции:

* Движение машины по гоночной трассе;
* Обработка вылета машины с трассы;
* Создание и отрисовка гоночной трассы;
* Подсчет количества пройденных кругов;
* Режим игры на время;
* Режим игры против компьютера.

При разработке программного средства будет использована среда разработки Visual Studio Code с использованием языка программирования Flat Assembler и средством отладки OllyDbg.

# ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

## **Структура программы**

Было принято решение, что при разработке данного программного средства будут использоваться 17 модуль:

* Drawing – модуль для рендера объектов;
* Audio – модуль для воспроизведения звуков;
* File – модуль для чтения файлов;
* Button – модуль для реализации работы кнопок;
* Object – модуль, которая отвечает за создание объектов;
* Matrix – модуль, содержащая подпрограммы для работы с матрицами;
* Vector – модуль, содержащая подпрограммы для работы с векторами;
* Spline – модуль для создания сплайнов;
* Physics – модуль для обработки физики объектов;
* Quaternion – модуль, содержащая реализацию основных функций для работы с кватернионами;
* UI – модуль для пользовательского интерфейса;
* Utils – модуль содержащая;
* BMP – модуль для загрузки содержимого BMP файла;
* Texture – модуль для создания текстур;
* OBJ – модуль для загрузки содержимого OBJ файла;
* VAO – модуль для создания Vertex Array Object объекта;
* VBO – модуль для создания Vertex Buffer Object объекта;

## **Проектирование интерфейса программного средства**

Для проектирования интерфейса программного средства использовался графический редактор «Paint».

### Окно главного меню

Окно будет содержать:

* заголовок названия игры;
* кнопка «START»;
* кнопка «SETTINGS»;
* кнопка «EXIT».

Предполагаемый интерфейс главного меню из программного средства представлен на рисунке 2.1.

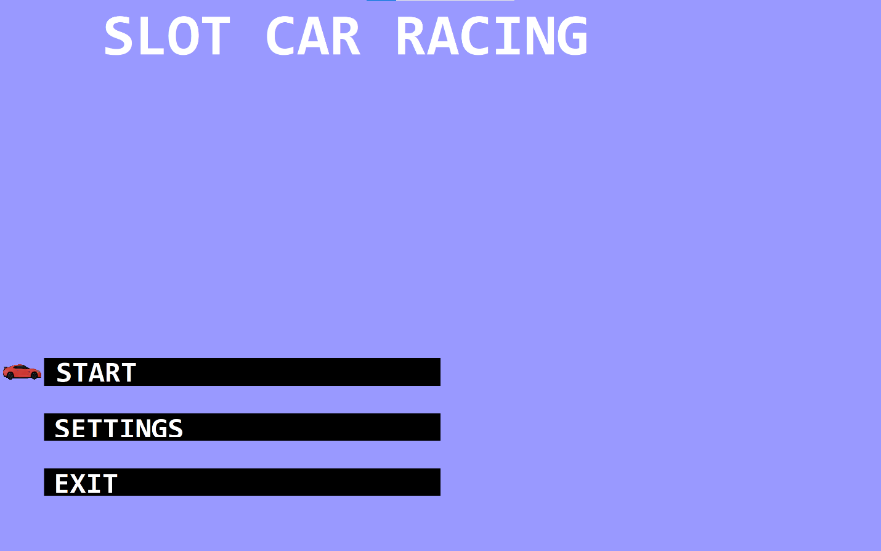


Рисунок 2.1 – Окно выхода из программного средства

### Окно выбора режима игры

Данное окно служит для выбора режима игры. Окно состоит из трех компонентов:

* Кнопка «TIME»;
* Кнопка «SETTINGS»;
* Кнопка «DUEL».

Дизайн окна загрузки находится на рисунке 2.2.

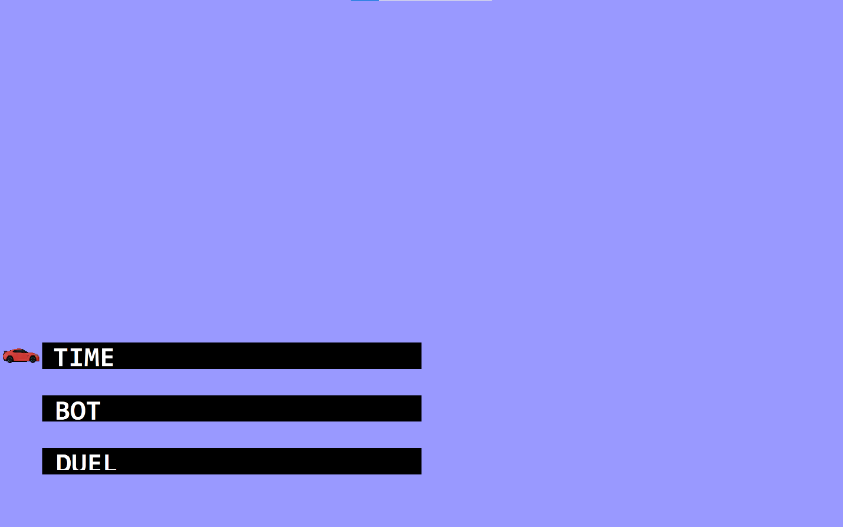


Рисунок 2.2 – Окно выбора режима игры

### Окно выбора стартового времени

Данное окно служит для выбора времени для игры. Окно состоит из трех компонентов:

* Кнопка «30»;
* Кнопка «60»;
* Кнопка «120».

Дизайн стартового окна находится на рисунке 2.3.

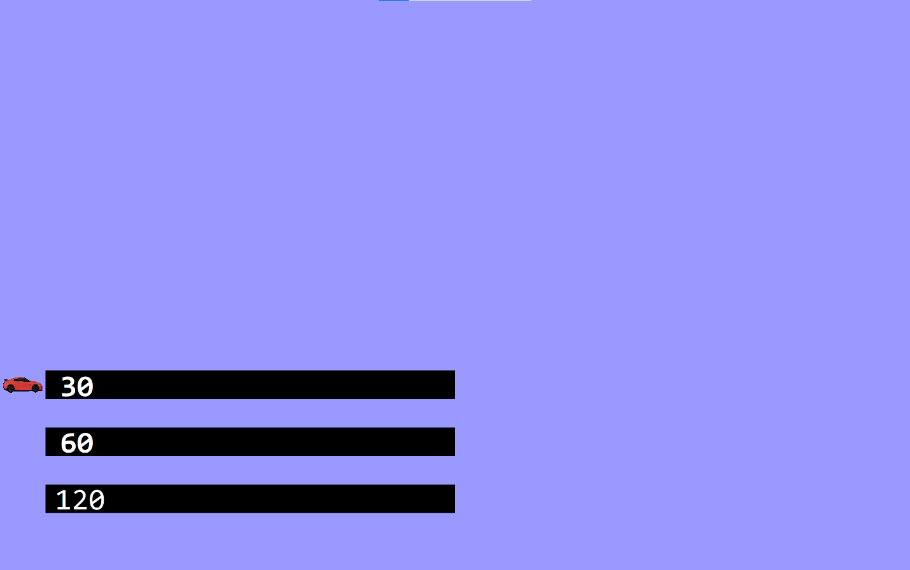


Рисунок 2.3 – Окно выбора стартового времени

## **Проектирование функционала программного средства**

Функционал программного средства – ключевая точка разработки программного средства, составляющей которого являются алгоритмы. В связи с этим, программное средство «Slot car racing» должно обладать некоторыми подпрограммами:

* умножение матриц четвертого порядка;
* отрисовка текста;
* чтение вершин из OBJ файла;
* поиск первой производной сплайна;
* перевод числа в строку.

### Умножение матриц четвертого порядка

Подпрограмма принимает 3 параметра: адрес первой матрицы, адрес второй матрицы, адрес по которому нужно записать результат.

Блок-схема алгоритма представлена на рисунке 2.4.



Рисунок 2.4 – Блок-схема подпрограммы умножения матриц четвертого порядка

Алгоритм умножения матриц заключается в вычислении каждого элемента выходной матрицы как суммы произведений соответствующих элементов строк первой матрицы и столбцов второй матрицы.

### Отрисовка текста

Данная подпрограмма реализует отрисовку текста. Подпрограмма принимает 5 параметров: объект для отрисовки, строку с текстом, координаты для отрисовки, цвет текста, отступ между буквами.

Алгоритм проходится по строке и для каждой буквы создает текстурные координаты, смещает квадрат на необходимый отступ и отрисовывает квадрат с текстурой буквы.

Рассмотреть блок-схему подпрограммы можно на рисунке 2.5.



Рисунок 2.5 – Блок-схема подпрограммы отрисовки текста

### Чтение вершин из Obj файла

Подпрограмма производит чтение вершин из Obj файла. В качестве параметров подпрограмма принимает буфер с данными, размер буфера и количество вершин. Вначале выделяется память под вершины. Далее, путем прохода по буферу, происходит поиск сочетания символов «v », которое обозначает начало строки с координатами вершины. Читаем 3 числа с плавающей точкой разделенных пробелом и добавляем их в итоговый буфер. Так продолжается до конца файла.

Рассмотреть блок-схему подпрограммы можно на рисунке 2.6.



Рисунок 2.6 – Блок-схема подпрограммы чтения вершин из Obj файла

### Поиск первой производной сплайна

Подпрограмма поиска первой производной сплайна находит по переданному сплайну и моменту времени его первую производную путем умножения матрицы кубического сплайна на дифференцированный вектор обычного кубического сплайна.

В конце подпрограммы значение переменной статуса присваивается переменной возврата. Рассмотреть блок-схему алгоритма можно на рисунке 2.7.



Рисунок 2.7 – Блок-схема подпрограммы поиска первой производной кубического сплайна

### Перевод числа в строку

Данная подпрограмма переводит число в строку. Принимает целое число, и возвращает указатель на Си строку. Алгоритм делит число на 10 и символ, полученный из остатка от деления, помещает на стек, далее возвращая из стека символ вставляет его в строку.

Увидеть блок-схему данной подпрограммы можно на рисунке 2.8.



Рисунок 2.8 – Блок-схема подпрограммы перевода числа в строку

1. **РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА**

Для разработки программного средства были использованы язык ассемблера FASM, библиотека OpenGL и язык описания шейдеров GLSL. С целью улучшения поддержки исходного кода в ходе разработки активно использовался комплект макросов, предлагаемый FASM.

## **Основная механика**

Ключевой составляющей программного средства является возможность движения по гоночной трассе и вылет с трассы при достижении максимальной скорости на данном участке трассы. Подпрограммы, которые позволяют реализуют подобный функционал, описаны ниже.

### Кубическая интреполяция

Подпрограмма реализует расчет точки на кубической кривой Безье с использованием математики сплайнов. Она четыре контрольные точки, вычисляет параметры для текущего значения времени f и применяет матрицу сплайнов для генерации итогового веса каждой точки. Затем эти веса используются для вычисления финальной позиции на кривой через линейное комбинирование контрольных точек.

;кубическая интерполяция

proc Spline.CurveBezier uses esi, p1, p2, p3, p4, result, f

locals

t dd ? ; Временной параметр t

tt dd ? ; Квадрат времени t

ttt dd ? ; Куб времени t

tempVector Vector4 ; Временный вектор для хранения параметров t

t1 Vector3 ; Промежуточный результат для p1

t2 Vector3 ; Промежуточный результат для p2

t3 Vector3 ; Промежуточный результат для p3

t4 Vector3 ; Промежуточный результат для p4

endl

; Вычисление параметров t, t^2 и t^3

fld [f] ; Загрузить f в стек FPU

fst [t] ; t = f

fmul [f] ; t^2

fst [tt] ; tt = f^2

fmul [f] ; t^3

fstp [ttt] ; ttt = f^3

; Подготовка вектора параметров (1, t, t^2, t^3)

lea esi, [tempVector]

fld1

fstp dword[esi + Vector4.x] ; tempVector.x = 1

fld [t]

fstp dword[esi + Vector4.y] ; tempVector.y = t

fld [tt]

fstp dword[esi + Vector4.z] ; tempVector.z = t^2

fld [ttt]

fstp dword[esi + Vector4.w] ; tempVector.w = t^3

; Умножение параметров на матрицу сплайна

stdcall Vector3.SMultiplyMatrix, esi, splineMatrix, splineVector

; Копирование контрольных точек в промежуточные векторы

lea eax, [t1]

stdcall Vector3.Copy, eax, [p1]

lea eax, [t2]

stdcall Vector3.Copy, eax, [p2]

lea eax, [t3]

stdcall Vector3.Copy, eax, [p3]

lea eax, [t4]

stdcall Vector3.Copy, eax, [p4]

; Вычисление вклада каждой точки с учетом весов из splineVector

lea eax, [t1]

stdcall Vector3.CrossValue, eax, [splineVector.x]

lea eax, [t2]

stdcall Vector3.CrossValue, eax, [splineVector.y]

lea eax, [t3]

stdcall Vector3.CrossValue, eax, [splineVector.z]

lea eax, [t4]

stdcall Vector3.CrossValue, eax, [splineVector.w]

; Суммирование вкладов контрольных точек для получения результата

mov esi, [result]

lea eax, [t1]

stdcall Vector3.Copy, esi, eax

lea eax, [t2]

stdcall Vector3.Add, esi, eax

lea eax, [t3]

stdcall Vector3.Add, esi, eax

lea eax, [t4]

stdcall Vector3.Add, esi, eax

ret

endp

### Получение вращения на сплайне

Чтобы найти угол поворота объекта на сплайне, нужно найти первую производную сплайна в этот момент времени и найти угол между ней и осью, задающей направление движения.

; получение вращения на сплайне

proc Spline.GetRotation uses esi edi,\

spline, objPtr, time

locals

point Vector3 ; Производная сплайна

toDegree dd 180.0 ; Константа для перевода радиан в градусы

angle dd 0.0 ; Угол вращения

endl

; Получение производной сплайна

lea eax, [point]

stdcall Spline.GetDerivative, [spline], eax, [time]

; Вычисление угла направления

lea esi, [point]

fld dword [esi]

fchs

fld dword [esi + 8]

fpatan

fmul [toDegree]

fldpi

fdivr st0, st1

fstp st1

fstp [angle]

; Сохранение угла и установка вращения

mov eax, [angle]

mov esi, [objPtr]

mov [esi + Object.tempAngle], eax

stdcall Quaternion.CreateFromAxisAngle, [objPtr], 0.0, 1.0, 0.0, [esi +

Object.tempAngle]

ret

endp

### Проверка количества пройденных кругов

Подпрограмма вычисляет, сколько полных кругов объект прошел вдоль сплайна, основываясь на текущей позиции объекта и длине сплайна. Она округляет результат до ближайшего целого числа и возвращает его.

; проверка количества пройденных кругов

proc Object.CheckLap uses esi, objPtr

locals

oldState dw ? ; Сохранение состояния FPU

newState dw ? ; Новое состояние FPU для округления

laps dd 0 ; Количество кругов

endl

; Сохранение текущего состояния FPU и настройка округления

mov esi, [objPtr]

fstcw [oldState]

mov ax, [oldState]

or ax, 0c00h

mov [newState], ax

fldcw [newState]

; Вычисление количества полных кругов

fld [esi + Object.positionOnSpline]

fdiv [cubeSpline.time]

fistp [laps]

; Восстановление исходного состояния FPU

fldcw [oldState]

; Возвращение количество кругов

mov eax, [laps]

ret

endp

## **Структура кнопки**

### Структура кнопки используется для представления состояния и поведения интерактивного элемента в пользовательском интерфейсе. Она обеспечивает эффективное и компактное описание свойств кнопки, таких как ее положение, размер, внешний вид, а также возможность обработки событий (например, нажатий). Основные подпрограммы работы со структурой кнопки представлены ниже:

### Создание кнопки

Подпрограмма создает и инициализирует структуру кнопки, задавая её позицию, внешний вид и текст. Код данной подпрограммы представлен ниже.

; подпрограмма создания кнопки

proc Button.Create uses esi, objPtr, x, y, message, buttonColor, textColor

locals

length dd ? ; Переменная для хранения длины текста

endl

; Создание прямоугольника для сообщения кнопки

mov esi, [objPtr]

add esi, Button.messageRect

stdcall Rectangle.Create, esi, 0.0, 0.0, 1.0, 1.0

stdcall Object.SetPosition, esi, [x], [y], 0.0

; Создание текстуры для текста кнопки

mov esi, [objPtr]

stdcall Texture.CreateRGBA, fontTexture, 512, 512

mov [esi + Button.messageRect.texture], eax

; Получение длины строки текста

stdcall GetStringLength, [message]

mov [length], eax

fild [length]

fstp [length]

; Создание прямоугольника для самой кнопки

mov esi, [objPtr]

add esi, Button.buttonRect

stdcall Rectangle.Create, esi, 0.0, 0.0, 9.0, 1.0

stdcall Object.SetPosition, esi, [x], [y], 0.0

; Создание текстуры для кнопки

mov esi, [objPtr]

stdcall Texture.Create, buttonTexture

mov [esi + Button.buttonRect.texture], eax

; Установка данных в структуру кнопки

mov eax, [message]

mov [esi + Button.message], eax

mov eax, [x]

mov [esi + Button.position.x], eax

mov eax, [y]

mov [esi + Button.position.y], eax

mov eax, [buttonColor]

mov [esi + Button.buttonColor], eax

mov eax, [textColor]

mov [esi + Button.textColor], eax

ret

endp

### Отрисовка прямоугольника

Процедура выполняет отрисовку объекта с текстурой и цветом в OpenGL. Она сначала создает модельную матрицу, затем устанавливает шейдерную программу и передает необходимые матрицы и цвет в шейдер. Далее привязываются текстура и вершинный массив, после чего выполняется рисование объекта, и очищаются привязки.

;отрисовка прямоугольника

proc Sprite.Draw uses esi, objPtr, color

; Создание модельной матрицы объекта

stdcall Matrix.CreateModel, [objPtr]

; Установка шейдерной программы

invoke glUseProgram, [program2D]

; Передача модельной и проекционной матриц в шейдер

invoke glUniformMatrix4fv, [modelLocation3], 1, GL\_FALSE, model

invoke glUniformMatrix4fv, [projectionLocation2D], 1, GL\_FALSE,

projection2D

; Передача цвета в шейдер

invoke glUniform3fv, [colorNameLocation], 1, [color]

; Привязка текстуры объекта

mov esi, [objPtr]

invoke glActiveTexture, GL\_TEXTURE0

invoke glBindTexture, GL\_TEXTURE\_2D, [esi + Object.texture]

; Привязка вершинного массива объекта

invoke glBindVertexArray, [esi + Object.VAO]

; Отрисовка объекта

invoke glDrawArrays, 6, 0, 6

invoke glBindVertexArray, 0

invoke glBindTexture, GL\_TEXTURE\_2D, 0

ret

endp

### Отрисовка кнопки

При вызове процедуры отрисовки кнопки последовательно вызываются процедуры отрисовки текста кнопки и самой прямоугольника кнопки. Подпрограмма данной логики представлена далее.

; алгоритм отрисовки кнопки

proc Button.Draw uses esi, objPtr

; Получение указателя на объект кнопки

mov esi, [objPtr]

; Отображение текста на кнопке

lea eax, [esi + Button.messageRect

stdcall Sprite.DrawText, eax, [esi + Button.message], [esi + Button.position.x],

[esi + Button.position.y], [esi + Button.textColor], 1.0

; Отображение фона кнопки

lea eax, [esi + Button.buttonRect

stdcall Sprite.Draw, eax, [esi + Button.buttonColor] ; Отображение фона кнопки

с указанным цветом

ret

endp

## **Структура кватерниона**

Структура кватерниона используется для представления вращений в 3D-пространстве, обеспечивая эффективное и компактное описание углов поворота. Она позволяет избежать проблем с углами Эйлера, таких как гимбал-лок (шарнирный замок). Основная подпрограмма для работы со структурой кнопки представлена ниже.

1. Создание кватерниона

Подпрограмма создает кватернион из оси вращения и угла (в градусах), представляя вращение в 3D-пространстве. Она вычисляет компоненты кватерниона на основе оси вращения и угла и сохраняет их в структуре объекта. В результате, кватернион может быть использован для выполнения вращений.

;создание кватерниона

proc Quaternion.CreateFromAxisAngle uses esi, objPtr, X, Y, Z, degree

locals

angle GLfloat ? ; угол в радианах

toRadian GLfloat 180.0 ; множитель для перевода градусов в радианы

two GLfloat 2.0 ; константа для деления

x GLfloat 0.0 ; компонент x кватерниона

y GLfloat 0.0 ; компонент y кватерниона

z GLfloat 0.0 ; компонент z кватерниона

w GLfloat 0.0 ; компонент w кватерниона

endl

; Перевод угла в радианы

fld [degree]

fdiv [toRadian] ; degree / 180

fldpi ; загружаем значение pi

fmulp ; умножаем на pi

fdiv [two] ; делим на 2

fsincos ; вычисляем синус и косинус

fxch ; меняем местами (синус и косинус)

fstp [w] ; сохраняем косинус в w

; Вычисление компонентов кватерниона

fst [angle] ; сохраняем синус в angle

fmul [X] ; angle \* X

fstp [x] ; сохраняем результат в x

fld [angle] ; загружаем angle

fmul [Y] ; angle \* Y

fstp [y] ; сохраняем результат в y

fld [angle] ; загружаем angle

fmul [Z] ; angle \* Z

fstp [z] ; сохраняем результат в z

; Сохранение компонентов кватерниона в объект

mov esi, [objPtr]

mov eax, [x]

mov [esi + Object.q.x], eax

mov eax, [y]

mov [esi + Object.q.y], eax

mov eax, [z]

mov [esi + Object.q.z], eax

mov eax, [w]

mov [esi + Object.q.w], eax

ret

endp

## **Работа с внешними источниками**

Работа с внешними источниками обеспечивает большую гибкость и упрощает управление данными. В программном средстве данные хранятся в файлах форматов Obj, Bmp, Glsl. Реализации подпрограмм работы этих форматов представлена ниже.

### Чтение BMP файла

Подпрограмма загружает содержимое BMP-файла в память. Она выделяет память для хранения данных изображения и копирует данные изображения из исходного файла в выделенный буфер. В конце возвращается указатель на загруженные данные. Код данной подпрограммы представлен ниже.

;чтение BMP файла

proc BMP.LoadFromFile uses edi esi ebx,\

fileName

locals

buffer dd 0

BMPBuffer dd 0

BMPBufferSize dd 0

h dd ?

w dd ?

endl

;чтение содержимого файла

stdcall File.LoadContent, [fileName]

mov ebx, eax

mov eax, [ebx + BMP.biHeight]

mov ecx, [ebx + BMP.biWidth]

xor edx, edx

mul ecx

xor edx, edx

mov ecx, 3

mul ecx

mov [BMPBufferSize], eax

;выделение памяти под буфер

malloc eax

mov [BMPBuffer], eax

;получение смещения данных в файле BMP

mov eax, [ebx + BMP.bfOffBits]

add ebx, eax

;копирование файлов в буфер

memcpy [BMPBuffer], ebx, [BMPBufferSize]

mov eax, [BMPBuffer]

mov ecx, [h]

mov edx, [w]

ret

endp

### Загрузка шейдеров из файла Glsl

Функция загружает исходный код шейдера из файла и создает шейдерный объект в OpenGL. Код функции представлен ниже.

proc Glext.LoadShader, fileName, shaderType

locals

buffer dd ?

shader dd ?

compiled dd ?

endl

;загружаем содержимое файла с исходным кодом шейдера

stdcall File.LoadContent, [fileName

mov [buffer], eax

;создаем шейдерный объект указанного типа

invoke glCreateShader, [shaderType]

mov [shader],

lea eax, [buffer

;передаем исходный код шейдера в объект шейдера

invoke glShaderSource, [shader], 1, eax, 0

;компилируем шейдер

invoke glCompileShader, [shader]

;освобождаем память, выделенную для исходного кода шейдера

invoke HeapFree, [hHeap], 0, [buffer]

; возвращаем шейдерный объект

mov eax, [shader]

ret

endp

Функция передает код шейдера в созданный объект, компилирует его. После этого освобождает память, выделенную для буфера с исходным кодом шейдера, и возвращает шейдерный объект.

1. **ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА**

В процессе разработки возникала проблема некорректной работы с вещественными числами. Проблема возникала из-за того, что при загрузке натуральных чисел в FPU с помощью команды fld они автоматически преобразовывались в вещественные числа, что могло привести к потерям точности. Это приводило к неожиданным погрешностям при выполнении арифметических операций с такими значениями. Решением проблемы стало использование инструкции fild.

Еще возникала проблема с неправильным вычислением смещения во время передачи в функцию указателя. Пример представлен ниже:

stdcall Sprite.Draw, esi + Button.buttonRect, [esi + Button.buttonColor]

Решением проблемы стало использование инструкции lea для правильной загрузки указателя со смещением в регистр.

Исправленный вариант представлен ниже:

lea eax, [esi + Button.buttonRect]

stdcall Sprite.Draw, eax, [esi + Button.buttonColor]

При рендеринге текстур с прозрачностью прозрачность не отображалась. Проблема была связана с выключенным альфа-тестом в OpenGL.

Решение проблемы показано ниже:

invoke glEnable, GL\_ALPHA\_TEST

Еще можно упомянуть проблему, связанную с неправильной работой теней. Неправильная работа теней была связана с некорректной работой процедуры по созданию матрицы ортографической проекции. Проблема заключалась в том матрица заполнялась единицами при каждом вызове процедуры. Исправленная версия процедуры представлена ниже:

proc Matrix.Ortho uses esi edi, \

matrix, left, right, bottom, top, zNear, zFar

locals

two dd 2.0

mtwo dd -2.0

mone dd -1.0

endl

; Инициализация указателя на матрицу

mov esi, [matrix]

; Вычисление m11

fld [right]

fsub [left]

fdivr [two]

fstp dword[esi + Matrix4x4.m11]

; Вычисление m22

fld [top]

fsub [bottom]

fdivr [two]

fstp dword[esi + Matrix4x4.m22]

; Вычисление m33

fld [zFar]

fsub [zNear]

fdivr [mtwo]

fstp dword[esi + Matrix4x4.m33]

; Установка m44 в 1

fld1

fstp dword[esi + Matrix4x4.m44]

; Вычисление m14

fld [right]

fadd [left]

fld [right]

fsub [left]

fdivp

fmul [mone]

fstp dword[esi + Matrix4x4.m14]

; Вычисление m24

fld [top]

fadd [bottom]

fld [top]

fsub [bottom]

fdivp

fmul [mone]

fstp dword[esi + Matrix4x4.m24]

; Вычисление m34

fld [zFar]

fadd [zNear]

fld [zFar]

fsub [zNear]

fdivp

fmul [mone]

fstp dword[esi + Matrix4x4.m34]

ret ; Возвращаем управление

endp

Исправленная версия правильно заполняет все элементы матрицы ортографической проекции.

1. **РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

## **Интерфейс программного средства**

### Окно загрузки

Окно загрузки служит для отображения процесса загрузки ресурсов программного средства.

Внешний вид окна загрузки изображён на рисунке 5.1.



Рисунок 5.1 – Внешний вид окна загрузки

### Главное меню

Главное меню – первое окно, с которым пользователь будет взаимодействовать. В данном окне расположены три управляющие кнопки:

* кнопка «START»;
* кнопка «SETTINGS»;
* кнопка «EXIT»;

Нажимая на кнопку, пользователь попадает на соответствующую кнопке страницу. При нажатии на кнопку «EXIT» пользователь выйдет из программы. При нажатии на кнопку «START» пользователь перейдет в меню выбора режима игры.

Внешний вид главного меню изображён на рисунке 5.2.



Рисунок 5.2 – Внешний вид главного меню

### Окно выбора режима игры

Окно выбора режима игры состоит из трех кнопок:

* кнопка «TIME»;
* кнопка «BOT»;
* кнопка «DUEL».

При нажатии на кнопку «TIME» пользователь выберет режим игры на время. При нажатии на кнопку «BOT» пользователь выберет режим игры против компьютера.

Увидеть интерфейс окна выбора режима игры 5.3.



Рисунок 5.3 – Интерфейс окна выбора режима игры

### Окно выбора времени игры

Окно выбора времени игры состоит из трех кнопок:

* кнопка «30»;
* кнопка «60»;
* кнопка «120».

При нажатии на кнопку пользователь выберет время для игры которое написано на кнопке: 30, 60 или 120 секунд.

Увидеть интерфейс окна выбора времени игры можно на рисунке 5.4.



Рисунок 5.4 – Интерфейс окна настроек

## **Управление программным средством**

Для правильного и комфортного пользования приложением пользователю необходим определённый набор возможностей. Наименьший возможный такой набор представлен далее.

### Знакомство с управлением

Первое, что рекомендуется сделать пользователю, ознакомится с управлением в игре. Для того, чтобы это сделать, необходимо, открыв главное меню, нажать на кнопку «SETTINGS».

После этого перед пользователем появится окно с описанием управления. Демонстрационный пример окна правил игры представлен на рисунке 5.5.



Рисунок 5.5 – Пример окна описания управления игры

После ознакомления с информацией необходимо нажать на клавишу клавиатуры «ESC», которая вернёт пользователя обратно на стартовую страницу.

### Начало игры

После ознакомления с управлением игры, можно начать играть. Для того, чтобы начать играть, нужно выбрать режим игры и количество времени для игры.

После попадания в игру у пользователя появляется возможность управления автомобилем. Для того чтобы придать ускорение автомобилю нужно нажать на клавишу клавиатуры «A», после отпускания клавиши автомобиль плавно начнет останавливаться. Если скорость будет слишком большой, то автомобиль вылетит с трассы.

Внешний вид ситуации с вылетом автомобиля с трассы показана на рисунке 5.6.

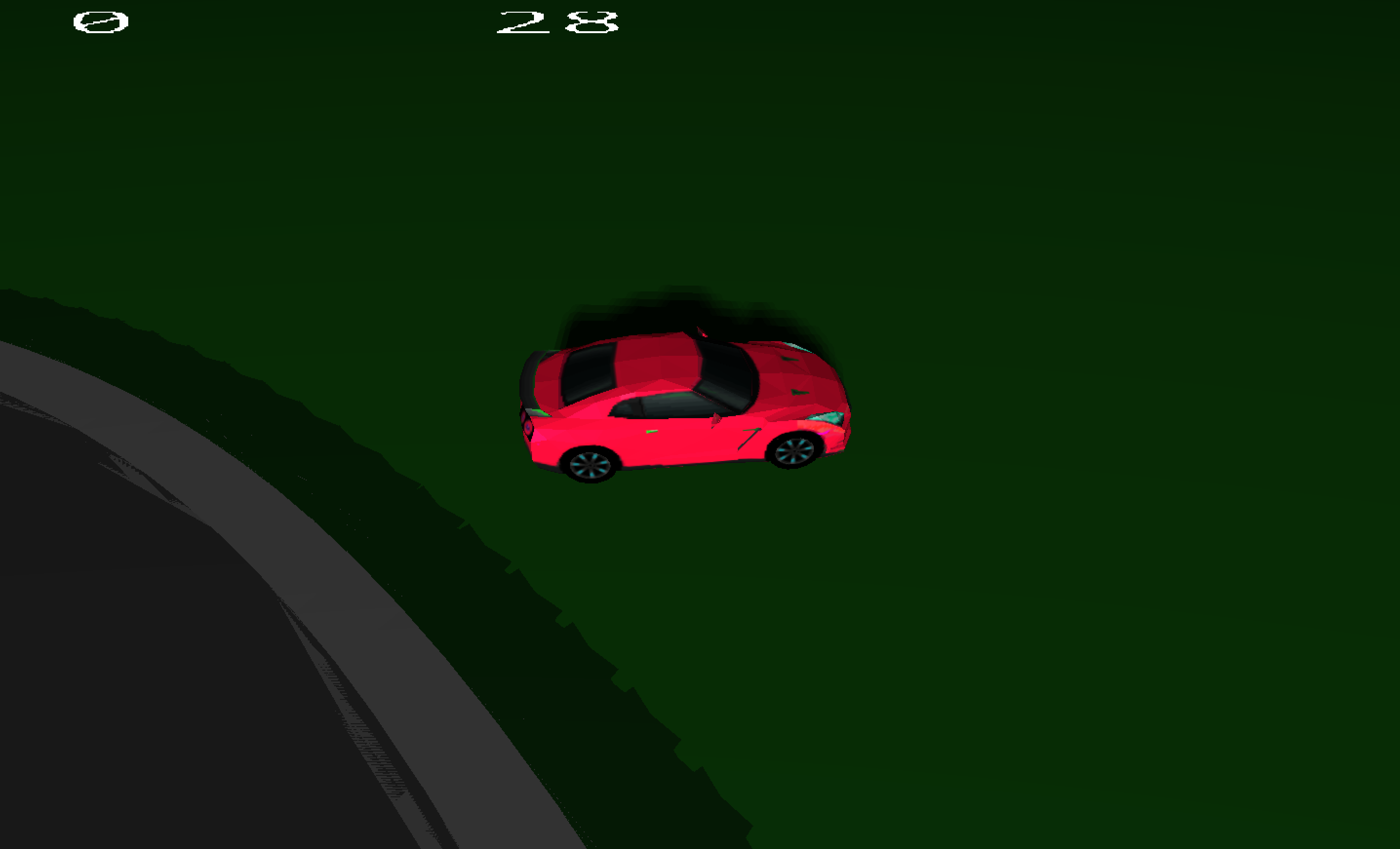


Рисунок 5.6 – Внешний вид ситуации с вылетом автомобиля с трассы

Для возвращения на трассу пользователю нужно нажать на клавишу клавиатуры «R», автомобиль вернется на последнее положение на трассе до вылета.

### Победа и поражение в игре

После завершения отсчета игрового таймера появляется окно с количеством пройденных кругов игроком, если был включен режим игры против компьютера, то выведется сообщение о том, кто победил.

Внешний вид этого окна показан на рисунке 5.7



Рисунок 5.7 – Внешний вид окна победы

### Выход из игры

Выйти из игры можно нажав на паузу во время игры и выйдя на стартовый экран нажать кнопку «EXIT выйти из игры.

Ещё одним способом выйти из игры нажатие клавиши клавиатуры «ESC» в главном меню.

# 

# ЗАКЛЮЧЕНИЕ

В данном курсовом проекте было разработано программное средство «Slot car racing». Игра создана под платформу Windows. При разработке успешно были реализованы следующие основные функции программного средства:

* Движение машины по гоночной трассе;
* Обработка вылета машины с трассы;
* Создание и отрисовка гоночной трассы;
* Подсчет количества пройденных кругов;
* Режим игры на время;
* Режим игры против компьютера.

Для достижения поставленной цели стало необходимым изучение основ программирования на FASM, знакомство с технологиями WinAPI, OpenGL, и Glsl, а также более глубокое изучение работы с внешними источниками и аудиовыходами.

Программное средство может быть дополнено режимом локального мультиплеера, игрой по сети с другими пользователями, разнообразным музыкальным сопровождением. Стоит рассмотреть возможность добавления большего количества гоночных трасс, изменения внешнего вида автомобиля.

Важным улучшением станет создание редактора трасс, позволяющего игрокам самостоятельно проектировать гоночные пути с различной степенью сложности и делиться своими проектами с другими пользователями. Такой функционал значительно увеличит реиграбельность и разнообразие игры.

Для более глубокого погружения в игровой процесс можно добавить систему погодных условий, которые будут влиять на управление машиной (например, дождь или снег). Это сделает гонки более динамичными и увлекательными.

Развитие таких возможностей позволит игре «Slot car racing» занять достойное место в жанре аркадных гоночных игр и привлечь как новых пользователей, так и удерживать существующих.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. OpenGL programming guide: The official guide to learning OpenGL [Электронный ресурс]. – Режим доступа: https://dl.acm.org/doi/abs/10.5555/1536767. – Дата доступа: 28.09.2024
2. Flat assembler [Электронный ресурс]. – Режим доступа: <https://flatassembler.net>. – Дата доступа: 01.10.2024
3. Intel 64 and IA-32 Architectures Software Developer’s Manual [Электронный ресурс]. – Режим доступа: <https://www.intel.com/content/www/us/en/developer/articles/technical/intel-sdm.html>. – Дата досутпа: 23.10.2024.
4. OpenGL API Documentation Overview [Электронный ресурс]. – Режим доступа: <https://www.opengl.org/Documentation/Specs.html>. – Дата доступа: 06.11.2024.
5. Стандарт предприятия. Дипломные проекты (работы). Общие требования : СТП 01−2017. – Введ. 01.01.2018. – Минск : БГУИР, 2017. – 169
6. The Book of Shaders [Электронный ресурс]. – Режим доступа: https://thebookofshaders.com/01/?lan=ru. – Дата доступа: 06.11.2024.

# ПРИЛОЖЕНИЕ А

Исходный код

proc Draw

stdcall Matrix.LookAt2, lightPosition, car.transform.position, upVector, view1

stdcall Matrix.Multiply4x4, view1, lightProjection, lightSpaceMatrix

invoke glViewport, 0, 0, SHADOW\_WIDTH, SHADOW\_HEIGHT

invoke glBindFramebuffer, GL\_FRAMEBUFFER, [depthMapFBO]

invoke glClear, GL\_DEPTH\_BUFFER\_BIT

invoke glUseProgram, [depthShader]

invoke glUniformMatrix4fv, [lightSpaceMatrixLocation2], 1, GL\_FALSE, lightSpaceMatrix

invoke glActiveTexture, GL\_TEXTURE0

invoke glBindTexture, GL\_TEXTURE\_2D, [depthMap]

stdcall RenderScene, [depthShader]

invoke glBindFramebuffer, GL\_FRAMEBUFFER, 0

invoke glClearColor, 0.6, 0.6, 1.0, 1.0

invoke glClear, GL\_COLOR\_BUFFER\_BIT or GL\_DEPTH\_BUFFER\_BIT

invoke glViewport, 0, 0, [clientRect.right], [clientRect.bottom]

stdcall Matrix.LookAt2, tempCameraVector, car.transform.position, upVector, view

invoke glUseProgram, [program]

invoke glUniformMatrix4fv, [projectionLocation], 1, GL\_FALSE, projection

invoke glUniformMatrix4fv, [viewLocation], 1, GL\_FALSE, view

invoke glUniformMatrix4fv, [lightSpaceMatrixLocation], 1, GL\_FALSE, lightSpaceMatrix

invoke glActiveTexture, GL\_TEXTURE0

invoke glBindTexture, GL\_TEXTURE\_2D, [depthMap]

stdcall RenderScene, [program]

;stdcall Sprite.Draw, R1, ORANGE\_COLOR

ret

endp

proc DrawHUD

invoke glUseProgram, [program2D]

invoke glEnable, GL\_ALPHA\_TEST

invoke glClear, GL\_DEPTH\_BUFFER\_BIT

invoke glUniformMatrix4fv, [projectionLocation2D], 1, GL\_FALSE, projection2D

stdcall Sprite.DrawText, R2, [lapCountStr], -9.0, 9.0, WHITE\_COLOR, 1.0

stdcall Sprite.DrawText, R2, [globalTimerStr], -3.0, 9.0, WHITE\_COLOR, 1.0

cmp [isBotGame], 2

jne @F

stdcall Sprite.DrawText, R2, [lapCountStr2], 8.0, 9.0, WHITE\_COLOR, 1.0

@@:

invoke glDisable, GL\_ALPHA\_TEST

ret

endp

proc DrawBotHUD

invoke glUseProgram, [program2D]

invoke glEnable, GL\_ALPHA\_TEST

invoke glClear, GL\_DEPTH\_BUFFER\_BIT

invoke glUniformMatrix4fv, [projectionLocation2D], 1, GL\_FALSE, projection2D

stdcall Sprite.DrawText, R2, [lapCountStr], -9.0, 9.0, WHITE\_COLOR, 1.0

stdcall Sprite.DrawText, R2, [globalTimerStr], -3.0, 9.0, WHITE\_COLOR, 1.0

stdcall Sprite.DrawText, R2, [lapCountStr2], 8.0, 9.0, WHITE\_COLOR, 1.0

invoke glDisable, GL\_ALPHA\_TEST

ret

endp

proc DrawSettings

invoke glUseProgram, [program2D]

invoke glEnable, GL\_ALPHA\_TEST

invoke glClear, GL\_DEPTH\_BUFFER\_BIT

invoke glUniformMatrix4fv, [projectionLocation2D], 1, GL\_FALSE, projection2D

stdcall Sprite.DrawText, R3, instructionMessage1, -8.0, 6.0, WHITE\_COLOR, 0.6

stdcall Sprite.DrawText, R3, instructionMessage2, -8.0, 3.0, WHITE\_COLOR, 0.6

stdcall Sprite.DrawText, R3, instructionMessage3, -8.0, 0.0, WHITE\_COLOR, 0.6

invoke glDisable, GL\_ALPHA\_TEST

stdcall Sprite.Draw, settingsRect, BLACK\_COLOR

ret

endp

proc DrawMenu uses esi, buttons

invoke glUseProgram, [program2D]

invoke glEnable, GL\_ALPHA\_TEST

invoke glClear, GL\_DEPTH\_BUFFER\_BIT

invoke glUniformMatrix4fv, [projectionLocation2D], 1, GL\_FALSE, projection2D

mov esi, [buttons]

stdcall Button.Draw, [esi]

stdcall Button.Draw, [esi + 4]

stdcall Button.Draw, [esi + 8]

stdcall Sprite.Draw2, carRect

invoke glDisable, GL\_ALPHA\_TEST

;stdcall Sprite.Draw, R1, ORANGE\_COLOR

ret

endp

proc DrawMainMenu

stdcall Matrix.LookAt2, lightPosition, car.transform.position, upVector, view1

stdcall Matrix.Multiply4x4, view1, lightProjection, lightSpaceMatrix

invoke glViewport, 0, 0, SHADOW\_WIDTH, SHADOW\_HEIGHT

invoke glBindFramebuffer, GL\_FRAMEBUFFER, [depthMapFBO]

invoke glClear, GL\_DEPTH\_BUFFER\_BIT

invoke glUseProgram, [depthShader]

invoke glUniformMatrix4fv, [lightSpaceMatrixLocation2], 1, GL\_FALSE, lightSpaceMatrix

invoke glActiveTexture, GL\_TEXTURE0

invoke glBindTexture, GL\_TEXTURE\_2D, [depthMap]

stdcall RenderScene, [depthShader]

invoke glBindFramebuffer, GL\_FRAMEBUFFER, 0

invoke glClearColor, 0.6, 0.6, 1.0, 1.0

invoke glClear, GL\_COLOR\_BUFFER\_BIT or GL\_DEPTH\_BUFFER\_BIT

invoke glViewport, 0, 0, [clientRect.right], [clientRect.bottom]

stdcall Matrix.LookAt2, cameraPosition, car.transform.position, upVector, view

invoke glUseProgram, [program]

invoke glUniformMatrix4fv, [projectionLocation], 1, GL\_FALSE, projection

invoke glUniformMatrix4fv, [viewLocation], 1, GL\_FALSE, view

invoke glUniformMatrix4fv, [lightSpaceMatrixLocation], 1, GL\_FALSE, lightSpaceMatrix

invoke glActiveTexture, GL\_TEXTURE0

invoke glBindTexture, GL\_TEXTURE\_2D, [depthMap]

stdcall Object.Draw, road, [shader]

stdcall Object.Draw, car, [shader]

stdcall Object.Draw, table, [shader]

stdcall DrawMenu, menuButtons

ret

endp

proc DrawEndGame

invoke glUseProgram, [program2D]

invoke glEnable, GL\_ALPHA\_TEST

invoke glClear, GL\_DEPTH\_BUFFER\_BIT

invoke glUniformMatrix4fv, [projectionLocation2D], 1, GL\_FALSE, projection2D

stdcall Sprite.DrawText, R2, finalMessage, -10.0, 0.0, WHITE\_COLOR, 1.0

stdcall Sprite.DrawText, R2, [lapCountStr], 3.5, 0.0, WHITE\_COLOR, 1.0

stdcall Sprite.DrawText, R3, restartMessage, -9.0, -9.0, WHITE\_COLOR, 0.7

invoke glDisable, GL\_ALPHA\_TEST

ret

endp

proc DrawEndBotGame

invoke glUseProgram, [program2D]

invoke glEnable, GL\_ALPHA\_TEST

invoke glClear, GL\_DEPTH\_BUFFER\_BIT

invoke glUniformMatrix4fv, [projectionLocation2D], 1, GL\_FALSE, projection2D

stdcall Sprite.DrawText, R2, [winnerMessage], -6.0, 5.0, WHITE\_COLOR, 1.0

stdcall Sprite.DrawText, R2, finalMessage, -10.0, 0.0, WHITE\_COLOR, 1.0

stdcall Sprite.DrawText, R2, [lapCountStr], 3.5, 0.0, WHITE\_COLOR, 1.0

stdcall Sprite.DrawText, R2, finalBotMessage, -10.0, -3.0, WHITE\_COLOR, 1.0

stdcall Sprite.DrawText, R2, [lapCountStr2], 3.5, -3.0, WHITE\_COLOR, 1.0

stdcall Sprite.DrawText, R3, restartMessage, -9.0, -9.0, WHITE\_COLOR, 0.7

invoke glDisable, GL\_ALPHA\_TEST

ret

endp

proc DrawSplashScreen

invoke glClearColor, 0.6, 0.6, 1.0, 1.0

invoke glUseProgram, [program2D]

invoke glEnable, GL\_ALPHA\_TEST

invoke glClear, GL\_DEPTH\_BUFFER\_BIT

invoke glUniformMatrix4fv, [projectionLocation2D], 1, GL\_FALSE, projection2D

stdcall Sprite.Draw, splashRect, splachColor

invoke glDisable, GL\_ALPHA\_TEST

ret

endp

proc RenderScene uses esi, shader

locals

dist dd 10.0

dist1 dd 10.0

toDegree dd 180.0

endl

;invoke CreateThread, 0, 0, ThreadWrapper, arg, 0, threadId

fldpi

fmul [tempAngle3]

fdiv [toDegree]

fstp [angle]

fld [car.transform.position.z]

fld [dist]

fsubp

fstp [tempCameraVector.z]

fld [car.transform.position.y]

fld [dist1]

faddp

fstp [tempCameraVector.y]

fld [car.transform.position.x]

fstp [tempCameraVector.x]

stdcall Object.Draw, road, [shader]

stdcall Object.Draw, car, [shader]

stdcall Object.Draw, car2, [shader]

stdcall Object.Draw, table, [shader]

stdcall Object.Draw, arch, [shader]

stdcall Object.Draw, grass, [shader]

ret

endp

proc Object.Draw uses esi, objPtr, shader

;memcpy tempMatrix, MVP, sizeof.Matrix4x4

stdcall Matrix.CreateModel, [objPtr]

mov eax, [shader]

cmp eax, [depthShader]

jne @F

invoke glUniformMatrix4fv, [modelLocation2D], 1, GL\_FALSE, model

stdcall DrawMesh, [objPtr], [drawingMode]

ret

@@:

invoke glUniformMatrix4fv, [modelLocation], 1, GL\_FALSE, model

stdcall DrawMesh, [objPtr], [drawingMode]

ret

endp

proc DrawMesh uses esi,\

mesh, flag

mov esi, [mesh]

cmp [esi + Object.useInstancing], 1

je @f

invoke glUniform1i, [useInstancingLocation], 0

invoke glActiveTexture, GL\_TEXTURE1

invoke glBindTexture, GL\_TEXTURE\_2D, [esi + Object.texture]

invoke glBindVertexArray, [esi + Object.VAO]

invoke glDrawArrays, [flag], 0, [esi + Mesh.verticesCount]

ret

@@:

invoke glUniform1i, [useInstancingLocation], 1

invoke glActiveTexture, GL\_TEXTURE1

invoke glBindTexture, GL\_TEXTURE\_2D, [esi + Object.texture]

invoke glBindVertexArray, [esi + Object.VAO]

invoke glDrawArraysInstanced, [flag], 0, [esi + Mesh.verticesCount], [instanseAmount]

ret

endp

proc Sprite.Draw uses esi, objPtr, color

stdcall Matrix.CreateModel, [objPtr]

invoke glUseProgram, [program2D]

invoke glUniformMatrix4fv, [modelLocation3], 1, GL\_FALSE, model

invoke glUniformMatrix4fv, [projectionLocation2D], 1, GL\_FALSE, projection2D

invoke glUniform3fv, [colorNameLocation], 1, [color]

mov esi, [objPtr]

invoke glActiveTexture, GL\_TEXTURE0

invoke glBindTexture, GL\_TEXTURE\_2D, [esi + Object.texture]

invoke glBindVertexArray, [esi + Object.VAO]

invoke glDrawArrays, 6, 0, 6

invoke glBindVertexArray, 0

invoke glBindTexture, GL\_TEXTURE\_2D, 0

ret

endp

proc Sprite.Draw2 uses esi, objPtr

stdcall Matrix.CreateModel, [objPtr]

invoke glUseProgram, [spriteShader]

invoke glUniformMatrix4fv, [modelLocationSprite], 1, GL\_FALSE, model

invoke glUniformMatrix4fv, [projectionLocation2DSprite], 1, GL\_FALSE, projection2D

mov esi, [objPtr]

invoke glActiveTexture, GL\_TEXTURE0

invoke glBindTexture, GL\_TEXTURE\_2D, [esi + Object.texture]

invoke glBindVertexArray, [esi + Object.VAO]

invoke glDrawArrays, 6, 0, 6

ret

endp

proc Sprite.DrawText uses esi edi ebx,\

objPtr, text, x, y, color, step

locals

tao dd ?

buffer dd ?

char db 0

charSize dd 0.0625

sixteen dd 16.0

x1 dd 0

y1 dd 4

left dd ?

right dd ?

top dd ?

bottom dd ?

xpos dd 0.0

endl

mov esi, [objPtr]

invoke glBindVertexArray, [esi + Object.VAO]

lea eax, [tao]

invoke glGenBuffers, 1, eax

invoke glBindBuffer, GL\_ARRAY\_BUFFER, [tao]

mov edi, [text]

.loop:

cmp byte[edi], 0

je .endl

xor eax, eax

mov al, byte[edi]

shr eax, 4

mov [y1], eax

mov al, byte[edi]

and eax, 1111b

mov [x1], eax

fild [x1]

fmul [charSize]

fst [left]

fadd [charSize]

fstp [right]

fild [y1]

fmul [charSize]

fst [top]

fadd [charSize]

fstp [bottom]

;0

mov ebx, textCoords

movss xmm0, [left]

movss xmm1, [bottom]

movss xmm2, [right]

movss xmm3, [top]

movups [ebx], xmm0

movups [ebx + 4], xmm1

movups [ebx + 8], xmm2

movups [ebx + 12], xmm1

movups [ebx + 16], xmm2

movups [ebx + 20], xmm3

movups [ebx + 24], xmm0

movups [ebx + 28], xmm3

invoke glBufferData, GL\_ARRAY\_BUFFER, 32, textCoords, GL\_DYNAMIC\_DRAW

invoke glEnableVertexAttribArray, 1

invoke glVertexAttribPointer, 1, 2, GL\_FLOAT, GL\_FALSE, 0, 0

stdcall Object.SetPosition, esi, [x], [y], 0.0

stdcall Matrix.CreateModel, esi

invoke glUniform3fv, [colorNameLocation], 1, [color]

invoke glUniformMatrix4fv, [modelLocation3], 1, GL\_FALSE, model

invoke glActiveTexture, GL\_TEXTURE0

invoke glBindTexture, GL\_TEXTURE\_2D, [esi + Object.texture]

invoke glDrawArrays, 6, 0, 4

inc edi

movss xmm0, dword [x]

addss xmm0, dword [step]

movss dword [x], xmm0

jmp .loop

.endl:

ret

endp

proc Vector3.Length uses esi,\

vector

locals

result dd ?

endl

mov esi, [vector]

fld [esi + Vector3.x]

fmul [esi + Vector3.x]

fld [esi + Vector3.y]

fmul [esi + Vector3.y]

fld [esi + Vector3.z]

fmul [esi + Vector3.z]

faddp

faddp

fsqrt

fstp [result]

mov eax, [result]

ret

endp

proc Vector3.Distance uses esi edi,\

v1, v2

locals

result dd ?

endl

mov esi, [v1]

mov edi, [v2]

fld [esi + Vector3.x]

fsub [edi + Vector3.x]

fmul st0, st0

fld [esi + Vector3.y]

fsub [edi + Vector3.y]

fmul st0, st0

fld [esi + Vector3.z]

fsub [edi + Vector3.z]

fmul st0, st0

faddp

faddp

fsqrt

fstp [result]

mov eax, [result]

ret

endp

proc Vector3.Normalize uses edi,\

vector

locals

l dd ?

endl

mov edi, [vector]

stdcall Vector3.Length, [vector]

mov [l], eax

fld [edi + Vector3.x]

fdiv [l]

fstp [edi + Vector3.x]

fld [edi + Vector3.y]

fdiv [l]

fstp [edi + Vector3.y]

fld [edi + Vector3.z]

fdiv [l]

fstp [edi + Vector3.z]

ret

endp

proc Texture.Create ,textPath

locals

temp dd 0

texture GLuint 0

endl

lea eax, [texture]

invoke glGenTextures, 1, eax

invoke glBindTexture, GL\_TEXTURE\_2D, [texture]

stdcall BMP.LoadFromFile, [textPath]

mov [temp], eax

invoke glTexImage2D, GL\_TEXTURE\_2D, 0, GL\_RGB, ecx, edx, 0, GL\_BGR, GL\_UNSIGNED\_BYTE, eax

invoke glGenerateMipmap, GL\_TEXTURE\_2D

invoke glTexParameteri, GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_REPEAT

invoke glTexParameteri, GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T, GL\_REPEAT

invoke glTexParameteri, GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR\_MIPMAP\_LINEAR

invoke glTexParameteri, GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR\_MIPMAP\_LINEAR

invoke HeapFree, [hHeap], 0, [temp]

invoke glBindTexture, GL\_TEXTURE\_2D, 0

mov eax, [texture]

ret

endp

proc Texture.CreateRGBA ,textPath, width, height

locals

temp dd 0

texture GLuint 0

endl

lea eax, [texture]

invoke glGenTextures, 1, eax

invoke glBindTexture, GL\_TEXTURE\_2D, [texture]

invoke glTexImage2D, GL\_TEXTURE\_2D, 0, GL\_RGBA, [width], [height], 0, GL\_RGBA,

GL\_UNSIGNED\_BYTE, [textPath]

invoke glTexParameteri, GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_CLAMP\_TO\_EDGE

invoke glTexParameteri, GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T, GL\_CLAMP\_TO\_EDGE

invoke glTexParameteri, GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR

invoke glTexParameteri, GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR

invoke glBindTexture, GL\_TEXTURE\_2D, 0

mov eax, [texture]

ret

endp

proc OBJ.ParseVertices uses edi esi,\

buffer, length, verticesCount

locals

resultVertices dd 0

endl

xor edx, edx

mov ecx, 12

mov eax, [verticesCount]

mul ecx

malloc eax

mov [resultVertices], eax

mov esi, [buffer]

mov ecx, [length]

mov edi, [resultVertices]

.cycle:

cmp byte[esi], 'v'

jne .skipLine

cmp byte [esi + 1], ' '

jne .skipLine

add esi, 2

sub ecx, 2

stdcall StrToFloat, esi

mov [edi], eax

add esi, 1

dec ecx

stdcall StrToFloat, esi

mov [edi + 4], eax

add esi, 1

dec ecx

stdcall StrToFloat, esi

mov [edi + 8], eax

add edi, 12

add esi, 1

dec ecx

loop .cycle

.skipLine:

cmp byte [esi], 10

je @F

cmp byte [esi], 0

je @F

dec ecx

jz .end

inc esi

jmp .skipLine

@@:

inc esi

loop .cycle

.end:

mov eax, [resultVertices]

ret

endp

proc OBJ.ParseIndices uses ebx edi esi ,\

buffer, length, indicesCount

locals

resultIndices dd 0

iCount dd 0

endl

xor edx, edx

mov ecx, 3 \* 4

mov eax, [indicesCount]

mul ecx

malloc eax ; eax

mov [resultIndices], eax

xor edx, edx

mov ecx, 3

mov eax, [indicesCount]

mul ecx

mov [iCount], eax

mov esi, [buffer]

mov ecx, [length]

mov edi, [resultIndices]

.cycle:

cmp byte[esi], 'f'

jne .skipLine

cmp byte [esi + 1], ' '

jne .skipLine

add esi, 2

dec ecx

dec ecx

stdcall StrToInt, esi

mov [edi], eax

dec[iCount]

stdcall SkipToSpace, esi

inc esi

dec ecx

stdcall StrToInt, esi

mov [edi + 4], eax

dec[iCount]

stdcall SkipToSpace, esi

inc esi

dec ecx

stdcall StrToInt, esi

mov [edi + 8], eax

dec[iCount]

add edi, 12

jmp .l1

.skipLine:

cmp byte [esi], 10

je @F

cmp byte [esi], 0

je @F

dec ecx

jz .end

inc esi

jmp .skipLine

@@:

inc esi

.l1:

cmp [iCount], 0

je .end

loop .cycle

.end:

mov eax, [resultIndices]

ret

endp

proc SkipToSpace pointer

mov esi, [pointer]

.cycle:

cmp byte[esi], ' '

je @F

inc esi

dec ecx

jmp .cycle

@@:

ret

endp

proc OBJ.CountObj uses esi ebx,\

buffer, length, verticesCount, indicesCount, textCoordCount

locals

vCount dd 0

iCount dd 0

tCount dd 0

endl

mov esi, [buffer]

mov ecx, [length]

.countCycle:

cmp byte[esi], 'v'

jne .l1

cmp byte[esi + 1], ' '

jne @F

inc [vCount]

jmp .newLine

@@:

cmp byte[esi + 1], 't'

jne .newLine

cmp byte[esi + 2], ' '

jne .newLine

inc [tCount]

.l1:

cmp byte[esi], 'f'

jne .newLine

cmp byte[esi + 1], ' '

jne .newLine

inc [iCount]

jmp .newLine

@@:

.newLine:

cmp byte [esi], 10

je @F

cmp byte[esi], 0

je .endl

inc esi

dec ecx

jmp .newLine

@@:

inc esi

loop .countCycle

.endl:

mov ebx, [verticesCount]

mov eax, [vCount]

mov [ebx], eax

mov ebx, [indicesCount]

mov eax, [iCount]

mov [ebx], eax

mov ebx, [textCoordCount]

mov eax, [tCount]

mov [ebx], eax

ret

endp

proc OBJ.ParseTextCoord uses edi esi,\

buffer, length, textCoordCount

locals

resultVertices dd 0

temp dd 0

ten dd 1.0

endl

xor edx, edx

mov ecx, 8

mov eax, [textCoordCount]

mul ecx

malloc eax

mov [resultVertices], eax

mov esi, [buffer]

mov ecx, [length]

mov edi, [resultVertices]

.cycle:

cmp byte[esi], 'v'

jne .skipLine

cmp byte [esi + 1], 't'

jne .skipLine

add esi, 2

sub ecx, 2

stdcall StrToFloat, esi

mov [temp], eax

fld [temp]

fmul [ten]

fstp [temp]

mov eax, [temp]

mov [edi], eax

add esi, 1

dec ecx

stdcall StrToFloat, esi

mov [temp], eax

fld [temp]

fmul [ten]

fstp [temp]

mov eax, [temp]

mov [edi + 4], eax

add esi, 1

dec ecx

add edi, 8

loop .cycle

.skipLine:

cmp byte [esi], 10

je @F

cmp byte [esi], 0

je @F

dec ecx

jz .end

inc esi

jmp .skipLine

@@:

inc esi

loop .cycle

.end:

mov eax, [resultVertices]

ret

endp

proc SkipToSlash pointer

mov esi, [pointer]

.cycle:

cmp byte[esi], '/'

je @F

inc esi

dec ecx

jmp .cycle

@@:

ret

endp

proc OBJ.ParseTextCoordI uses ebx edi esi ,\

buffer, length, indicesCount

locals

resultIndices dd 0

iCount dd 0

endl

xor edx, edx

mov ecx, 3 \* 4

mov eax, [indicesCount]

mul ecx

malloc eax

mov [resultIndices], eax

xor edx, edx

mov ecx, 3

mov eax, [indicesCount]

mul ecx

mov [iCount], eax

mov esi, [buffer]

mov ecx, [length]

mov edi, [resultIndices]

.cycle:

cmp byte[esi], 'f'

jne .skipLine

cmp byte [esi + 1], ' '

jne .skipLine

add esi, 2

sub ecx, 2

stdcall SkipToSlash, esi

inc esi

dec ecx

stdcall StrToInt, esi

mov [edi], eax

dec[iCount]

stdcall SkipToSpace, esi

stdcall SkipToSlash, esi

inc esi

dec ecx

stdcall StrToInt, esi

mov [edi + 4], eax

dec[iCount]

stdcall SkipToSpace, esi

stdcall SkipToSlash, esi

inc esi

dec ecx

stdcall StrToInt, esi

mov [edi + 8], eax

dec[iCount]

add edi, 12

jmp .l1

.skipLine:

cmp byte [esi], 10

je @F

cmp byte [esi], 0

je @F

dec ecx

jz .end

inc esi

jmp .skipLine

@@:

inc esi

.l1:

cmp [iCount], 0

je .end

loop .cycle

.end:

mov eax, [resultIndices]

ret

endp

proc VAO.Init uses edi esi ebx,\

buffer, mesh

locals

size dd ?

vertexVBO GLuint ?

normalsVBO GLuint ?

textVBO GLuint ?

endl

mov ebx, [buffer]

invoke glGenVertexArrays, 1, ebx

invoke glBindVertexArray, [ebx]

lea eax, [vertexVBO]

stdcall VBO.Create, eax, sizeof.Vertex, Mesh.vertices, [mesh]

invoke glEnableVertexAttribArray, 0

invoke glVertexAttribPointer, 0, 3, GL\_FLOAT, GL\_FALSE, 0, 0

lea eax, [normalsVBO]

stdcall VBO.Create, eax, sizeof.Vertex , Mesh.normals, [mesh]

invoke glEnableVertexAttribArray, 1

invoke glVertexAttribPointer, 1, 3, GL\_FLOAT, GL\_FALSE, 0, 0

lea eax, [textCoord]

stdcall VBO.Create, eax, sizeof.TextCoord, Mesh.texCoords, [mesh]

invoke glEnableVertexAttribArray, 2

invoke glVertexAttribPointer, 2, 2, GL\_FLOAT, GL\_FALSE, 0, 0

invoke glBindVertexArray, 0

ret

endp

proc VBO.Create uses esi edi ebx,\

buffer, sizeOfType, offset, mesh

locals

size dd ?

endl

mov esi , [mesh]

xor edx, edx

mov ecx, [sizeOfType]

mov eax, [esi + Mesh.verticesCount]

mul ecx

mov [size], eax

invoke glGenBuffers, 1, [buffer]

mov ebx, [buffer]

mov ebx, [ebx]

invoke glBindBuffer, GL\_ARRAY\_BUFFER, ebx

add esi, [offset]

invoke glBufferData, GL\_ARRAY\_BUFFER, [size], [esi], GL\_STATIC\_DRAW

ret

endp

proc WinMain

locals

msg MSG

endl

xor ebx, ebx

stdcall Init

lea esi, [msg]

.cycle:

invoke GetMessage, esi, ebx, ebx, ebx

invoke DispatchMessage, esi

jmp .cycle

endp

proc RenderThreadProc

ret

endp

proc WindowProc uses ebx,\

hWnd, uMsg, wParam, lParam

xor ebx, ebx

switch [uMsg]

case .Destroy, WM\_DESTROY

case .KeyDown, WM\_KEYDOWN

case .KeyUp, WM\_KEYUP

case .MouseWheel, WM\_MOUSEWHEEL

case .Paint, WM\_PAINT

invoke DefWindowProc, [hWnd], [uMsg], [wParam], [lParam]

jmp .Return

.KeyUp:

switch [wParam]

case .ResetAcceleration1, VK\_A

case .ResetAcceleration2, VK\_L

jmp .ReturnZero

.ResetAcceleration1:

stdcall Object.ResetAcceleration, car

jmp .ReturnZero

.ResetAcceleration2:

stdcall Object.ResetAcceleration, car2

jmp .ReturnZero

.Paint:

switch [GAME\_MODE]

case .RenderSettings, SETTINGS

case .RenderEnd2PGame, END2PGAME

case .RenderEndBotGame, ENDBOTGAME

case .RenderSplashScreen, SPLASHSCREEN

case .RenderMainMenu , MAINMENU

case .RenderMainMenu, SELECTMENU

case .RenderMainMenu, SELECTMODE

case .RenderMenu, PAUSEMENU

case .RenderGame, GAME

case .RenderEndGame, ENDGAME

.RenderSettings:

stdcall Draw

stdcall DrawSettings

jmp .RenderCommon

.RenderEnd2PGame:

jmp .ReturnZero

.RenderSplashScreen:

stdcall DrawSplashScreen

jmp .RenderCommon

.RenderGame:

stdcall Draw

stdcall DrawHUD

jmp .RenderCommon

.RenderMainMenu:

stdcall UI.DrawMainMenu, [activeMenu]

jmp .RenderCommon

.RenderMenu:

stdcall Draw

stdcall DrawMenu, [activeMenu]

jmp .RenderCommon

.RenderEndBotGame:

stdcall Draw

stdcall DrawEndBotGame

jmp .RenderCommon

.RenderEndGame:

stdcall Draw

stdcall DrawEndGame

;stdcall DrawMenu, [activeMenu]

.RenderCommon:

invoke SwapBuffers, [hdc]

stdcall Object.CheckLap, car

mov [car.lapCount], eax

stdcall Physics.Update, 0.07, cubeSpline, car

cmp [isBotGame], 2

jne .ReturnZero

stdcall BotPhysicsThread

jmp .ReturnZero

.KeyDown:

switch [wParam]

case .GoToMenu, VK\_ESCAPE

case .IncActiveButton, VK\_DOWN

case .DecActiveButton, VK\_UP

case .Speed, VK\_A

case .Respawn1, VK\_R

case .Enter, VK\_RETURN

jmp .ReturnZero

.Enter:

invoke PlaySound, musicPath1, NULL, 1

stdcall Button.Press, [activeButton]

stdcall UI.SetActiveButton, [activeMenu], 0

jmp .ReturnZero

.IncActiveButton:

invoke PlaySound, musicPath, NULL, 1

cmp [GAME\_MODE], SELECTMODE

je @F

cmp [GAME\_MODE], MAINMENU

je @F

cmp [GAME\_MODE], SELECTMENU

je @F

cmp [GAME\_MODE], PAUSEMENU

jne .ReturnZero

@@:

stdcall UI.SetDeactiveButton,[activeMenu], [activeButton]

cmp [activeButton], 1

ja @f

inc [activeButton]

stdcall UI.SetActiveButton, [activeMenu],[activeButton]

jmp .ReturnZero

@@:

mov [activeButton], 0

stdcall UI.SetActiveButton,[activeMenu], [activeButton]

jmp .ReturnZero

.DecActiveButton:

invoke PlaySound, musicPath, NULL, 1

cmp [GAME\_MODE], SELECTMODE

je @F

cmp [GAME\_MODE], MAINMENU

je @F

cmp [GAME\_MODE], SELECTMENU

je @F

cmp [GAME\_MODE], PAUSEMENU

jne .ReturnZero

@@:

stdcall UI.SetDeactiveButton, [activeMenu],[activeButton]

cmp [activeButton], 1

jb @f

dec [activeButton]

stdcall UI.SetActiveButton,[activeMenu], [activeButton]

jmp .ReturnZero

@@:

mov [activeButton], 2

stdcall UI.SetActiveButton, [activeMenu], [activeButton]

jmp .ReturnZero

.Respawn1:

switch [GAME\_MODE]

case .1, GAME

case .2, ENDGAME

case .2, ENDBOTGAME

.1:

stdcall Object.Respawn, car

jmp .ReturnZero

.2:

mov [GAME\_MODE], SELECTMENU

mov eax, selectModeButtons

mov [activeMenu], eax

stdcall Object.Respawn, car

jmp .ReturnZero

.Speed:

cmp [GAME\_MODE], ENDGAME

je .ReturnZero

cmp [GAME\_MODE], ENDBOTGAME

je .ReturnZero

stdcall Object.SetAcceleration, car

jmp .ReturnZero

.GoToMenu:

invoke PlaySound, musicPath1, NULL, 1

stdcall UI.SetDeactiveButton, [activeMenu],[activeButton]

stdcall UI.Pause

stdcall UI.SetActiveButton, [activeMenu], 0

jmp .ReturnZero

.Destroy:

invoke HeapDestroy, [hHeap]

invoke CloseHandle, [logFile]

invoke ExitProcess, ebx

mov [EXIT], 1

.ReturnZero:

xor eax, eax

.Return:

ret

endp

proc Quaternion.CreateFromAxisAngle uses esi, objPtr, X, Y, Z, degree

locals

angle GLfloat ?

toRadian GLfloat 180.0

two GLfloat 2.0

x GLfloat 0.0

y GLfloat 0.0

z GLfloat 0.0

w GLfloat 0.0

endl

fld [degree]

fdiv [toRadian]

fldpi

fmulp

fdiv [two]

fsincos

fxch

fstp [w]

fst [angle]

fmul [X]

fstp [x]

fld [angle]

fmul [Y]

fstp [y]

fld [angle]

fmul [Z]

fstp [z]

mov esi, [objPtr]

mov eax, [x]

mov [esi + Object.q.x], eax

mov eax, [y]

mov [esi + Object.q.y], eax

mov eax, [z]

mov [esi + Object.q.z], eax

mov eax, [w]

mov [esi + Object.q.w], eax

ret

endp

proc Quaternion.CreateMatrix uses esi edi,\

objPtr, matrix

locals

two GLfloat 2.0

one dd 1.0

x GLfloat 0.0

y GLfloat 0.0

z GLfloat 0.0

w GLfloat 0.0

endl

mov esi, [objPtr]

mov eax, [esi + Object.q.x]

mov [x], eax

mov eax, [esi + Object.q.y]

mov [y], eax

mov eax, [esi + Object.q.z]

mov [z], eax

mov eax, [esi + Object.q.w]

mov [w], eax

mov esi, [matrix]

fld [y] ;1 - 2(yy +zz)

fmul st0, st0

fld [z]

fmul st0, st0

faddp

fmul [two]

fsubr [one]

fstp [esi + Matrix4x4.m11]

fld [x] ;2(xy + zz)

fmul [y]

fld [z]

fmul [w]

faddp

fmul [two]

fstp [esi + Matrix4x4.m12]

fld [x] ; 2(x\*z - yw)

fmul [z]

fld [y]

fmul [w]

fsubp

fmul [two]

fstp [esi + Matrix4x4.m13]

mov [esi + Matrix4x4.m14], 0.0

fld [x]

fmul [y]

fld [z]

fmul [w]

fsubp

fmul [two]

fstp [esi + Matrix4x4.m21]

fld [x]

fmul st0, st0

fld [z]

fmul st0, st0

faddp

fmul [two]

fsubr [one]

fstp [esi + Matrix4x4.m22]

fld [z]

fmul [y]

fld [x]

fmul [w]

faddp

fmul [two]

fstp [esi + Matrix4x4.m23]

mov [esi + Matrix4x4.m24], 0.0

fld [x]

fmul [z]

fld [y]

fmul [w]

faddp

fmul [two]

fstp [esi + Matrix4x4.m31]

fld [y]

fmul [z]

fld [x]

fmul [w]

fsubp

fmul [two]

fstp [esi + Matrix4x4.m32]

fld [x]

fmul st0, st0

fld [y]

fmul st0, st0

faddp

fmul [two]

fsubr [one]

fstp [esi + Matrix4x4.m33]

mov [esi + Matrix4x4.m34], 0.0

mov [esi + Matrix4x4.m41], 0.0

mov [esi + Matrix4x4.m42], 0.0

mov [esi + Matrix4x4.m43], 0.0

mov [esi + Matrix4x4.m44], 1.0

ret

endp

proc File.LoadContent uses edi,\

fileName;, fileLength

locals

hFile dd ?

length dd ?

read dd ?

pBuffer dd ?

endl

invoke CreateFile, [fileName], GENERIC\_READ, ebx, ebx, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, ebx

mov [hFile], eax

invoke GetFileSize, [hFile], ebx

inc eax

mov [length], eax

;malloc 1, 8

invoke HeapAlloc, [hHeap], 8, [length]

mov [pBuffer], eax

lea edi, [read]

invoke ReadFile, [hFile], [pBuffer], [length], edi, ebx

invoke CloseHandle, [hFile]

mov eax, [pBuffer]

ret

endp

proc File.GetSize uses edi,\

fileName

locals

hFile dd ?

size dd ?

endl

invoke CreateFile, [fileName], GENERIC\_READ, ebx, ebx, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, ebx

mov [hFile], eax

invoke GetFileSize, [hFile], ebx

inc eax

mov [size], eax

invoke CloseHandle, [hFile]

mov eax, [size]

ret

endp

proc File.Create uses edi ,\

fileName

locals

hFile dd ?

endl

invoke CreateFile, [fileName], GENERIC\_WRITE, 0, 0, OPEN\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, 0

mov [hFile], eax

cmp eax, INVALID\_HANDLE\_VALUE

je .Error

stdcall Log, FILE\_CREATED\_SUCCESSFULLY, FILE\_CREATED\_SUCCESSFULLY.size

mov eax, [hFile]

ret

.Error:

stdcall Log, FILE\_CREATION\_FAILED , FILE\_CREATION\_FAILED.size

mov eax, [hFile]

ret

endp

proc File.WriteContent uses edi ,\

fileName, buffer, length

locals

hFile dd ?

pBuffer dd ?

endl

invoke CreateFile, [fileName], GENERIC\_READ, ebx, ebx, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, ebx

mov [hFile], eax

invoke WriteFile, [hFile], [buffer], [length], edi, ebx

ret

endp

proc Button.Press, index

stdcall UI.SetDeactiveButton, [activeMenu], [activeButton]

stdcall UI.SetActiveButton, [activeMenu], 0

mov [activeButton], 0

switch [index]

case .Start, 0

case .Settings, 1

case .Exit, 2

.Start:

switch [GAME\_MODE]

case .SettingsMode1, SETTINGS

case .SelectMenu, SELECTMENU

case .MainMenu, MAINMENU

case .PauseMenu, PAUSEMENU

case .SelestMode, SELECTMODE

case .GameMode, GAME

.SettingsMode1:

ret

.SelectMenu:

mov eax, modesButtons

mov [activeMenu], eax

mov [GAME\_MODE], SELECTMODE

ret

.MainMenu:

mov [GAME\_MODE], SELECTMENU

mov eax, selectModeButtons

mov [activeMenu], eax

ret

.PauseMenu:

mov [GAME\_MODE], GAME

ret

.SelestMode:

stdcall Button.selectMode, 30

.GameMode:

ret

.Settings:

switch [GAME\_MODE]

case .SettingsMode, SETTINGS

case .SelectMenu1, SELECTMENU

case .MainMenu1, MAINMENU

case .PauseMenu1, PAUSEMENU

case .SelestMode1, SELECTMODE

case .GameMode1, GAME

.SettingsMode:

ret

.SelectMenu1:

mov [isBotGame], 1

mov [GAME\_MODE], SELECTMODE

mov eax, modesButtons

mov [activeMenu], eax

ret

.MainMenu1:

mov [GAME\_MODE], SETTINGS

mov [fromMainMenu], 1

ret

.PauseMenu1:

mov [GAME\_MODE], SETTINGS

mov [fromMainMenu], 0

ret

.SelestMode1:

stdcall Button.selectMode, 60

ret

.GameMode1:

ret

ret

.Exit:

switch [GAME\_MODE]

case .SettingsMode2, SETTINGS

case .SelectMenu2, SELECTMENU

case .MainMenu2, MAINMENU

case .PauseMenu2, PAUSEMENU

case .SelestMode2, SELECTMODE

case .GameMode2, GAME

.SettingsMode2:

ret

.SelectMenu2:

ret

.MainMenu2:

invoke HeapDestroy, [hHeap]

invoke CloseHandle, [logFile]

invoke ExitProcess, ebx

.PauseMenu2:

mov [GAME\_MODE], MAINMENU

ret

.SelestMode2:

stdcall Button.selectMode, 120

ret

.GameMode2:

ret

ret

endp

proc Button.selectMode, time

mov [GAME\_MODE], GAME

mov eax, menuButtons

mov [activeMenu], eax

mov eax, [time]

mov [globalTimer], eax

mov [car.acceleration], 0.0

mov [car.speed], 0.0

mov [car.positionOnSpline], 0.0

mov [EXIT], 0

stdcall IntToStr, [car.lapCount]

mov [lapCountStr], eax

stdcall IntToStr, [car2.lapCount]

mov [lapCountStr2], eax

mov [car.lapCount], 0

mov [car2.lapCount], 0

invoke CreateThread, 0, 0, DecrementThread, 0, 0, 0

cmp [isBotGame], 1

jne @F

mov [isBotGame], 2

@@:

ret

endp

proc Button.Create uses esi, objPtr, x, y, message, buttonColor, textColor

locals

length dd ?

endl

mov esi, [objPtr]

add esi, Button.messageRect

stdcall Rectangle.Create, esi, 0.0, 0.0, 1.0, 1.0

stdcall Object.SetPosition, esi, [x], [y], 0.0

mov esi, [objPtr]

stdcall Texture.CreateRGBA, fontTexture, 512, 512

mov [esi + Button.messageRect.texture], eax

stdcall GetStringLength, [message]

mov [length], eax

fild [length]

fstp [length]

mov esi, [objPtr]

add esi, Button.buttonRect

stdcall Rectangle.Create, esi, 0.0, 0.0, 9.0, 1.0

stdcall Object.SetPosition, esi, [x], [y], 0.0

mov esi, [objPtr]

stdcall Texture.Create, buttonTexture

mov [esi + Button.buttonRect.texture], eax

mov eax, [message]

mov [esi + Button.message], eax

mov eax, [x]

mov [esi + Button.position.x], eax

mov eax, [y]

mov [esi + Button.position.y], eax

mov eax, [buttonColor]

mov [esi + Button.buttonColor], eax

mov eax, [textColor]

mov [esi + Button.textColor], eax

ret

endp

proc Button.Draw uses esi, objPtr

mov esi, [objPtr]

lea eax, [esi + Button.messageRect]

stdcall Sprite.DrawText, eax, [esi + Button.message], [esi + Button.position.x], [esi + Button.position.y], [esi + Button.textColor], 1.0

lea eax, [esi + Button.buttonRect]

stdcall Sprite.Draw, eax, [esi + Button.buttonColor]

ret

endp

proc Float.AddValue uses ebx, value, term

mov ebx, [value]

fld dword [ebx]

fadd [term]

fstp dword[ebx]

ret

endp

proc Float.SubValue uses ebx, value, term

mov ebx, [value]

fld dword [ebx]

fsub [term]

fstp dword[ebx]

ret

endp

proc StrToInt uses ebx edi,\

pointer

mov esi, [pointer]

xor eax, eax

@@:

cmp byte[esi], '/'

je @F

mov bl, byte[esi]

sub ebx, '0'

xor edx, edx

push ecx

mov ecx, 10

mul ecx

pop ecx

add eax, ebx

inc esi

dec ecx

jmp @B

@@:

dec eax

ret

endp

proc StrToFloat, pointer

locals

number dd 0.0

precision dd 0

sign db 0

ten dd 10.0

endl

mov esi, [pointer]

xor eax, eax

cmp byte[esi], '-'

jne @F

mov [sign], 1

xor eax, eax

inc esi

dec ecx

@@:

xor ebx, ebx

cmp byte[esi], '.'

je @F

mov bl, byte[esi]

sub ebx, '0'

push ecx

xor edx, edx

mov ecx, 10

mul ecx

pop ecx

add eax, ebx

inc esi

dec ecx

jmp @B

@@:

add eax, ebx

@@:

inc esi

dec ecx

xor ebx, ebx

cmp byte[esi], ' '

je @F

cmp byte[esi], 10

je @F

mov bl, byte[esi]

sub ebx, '0'

xor edx, edx

push ecx

mov ecx, 10

mul ecx

pop ecx

add eax, ebx

inc [precision]

cmp [precision], 9

;jg .precisionOF

jmp @B

.precisionOF:

inc esi

dec ecx

xor ebx, ebx

cmp byte[esi], ' '

je @F

cmp byte[esi], 10

je @F

jmp .precisionOF

@@:

mov [number], eax

mov eax, [precision]

fild [number]

@@:

cmp eax, 0

je @F

fdiv [ten]

dec eax

jmp @B

@@:

cmp [sign], 1

jne @F

fchs

@@:

fstp [number]

mov eax, [number]

ret

endp

proc IntToStr uses edi esi, \

value

locals

isNegative db 0

resString dd 0

numberLength dd 0

endl

stdcall GetNumberLength, [value]

mov [numberLength], eax

inc eax

cmp [value], 0

jg @f

mov [isNegative], 1

neg [value]

inc eax

@@:

malloc eax

mov [resString], eax

cmp [value], 0

jne @f

mov esi, [resString]

mov byte[esi], '0'

mov byte[esi + 1], 0

mov eax, esi

ret

@@:

mov ecx, 10

mov eax, [value]

.loop\_convert:

xor edx, edx

div ecx

add edx, '0'

push edx

cmp eax, 0

jg .loop\_convert

mov esi, [resString]

mov ecx, 0

cmp [isNegative], 1

jne .loop\_pop

mov byte[esi], '-'

inc esi

.loop\_pop:

pop eax

mov byte [esi + ecx], al

inc ecx

cmp ecx, [numberLength]

jne .loop\_pop

mov byte [esi + ecx], 0

mov eax, [resString]

ret

endp

proc IntToStr2 uses edi esi, \

value

locals

isNegative db 0

resString dd 0

numberLength dd 0

endl

stdcall GetNumberLength, [value]

mov [numberLength], eax

malloc eax

mov [resString], eax

cmp [value], 0

jne @f

mov esi, [resString]

mov byte[esi], '0'

mov eax, esi

ret

@@:

mov ecx, 10

mov eax, [value]

.loop\_convert:

xor edx, edx

div ecx

add edx, '0'

push edx

cmp eax, 0

jg .loop\_convert

mov esi, [resString]

mov ecx, 0

.loop\_pop:

pop eax

mov byte [esi + ecx], al

inc ecx

cmp ecx, [numberLength]

jne .loop\_pop

mov eax, [resString]

ret

endp

proc GetNumberLength uses edx,\

value

locals

length dd 0

endl

cmp [value], 0

jne @f

mov [length], 1

jmp @done

@@:

cmp [value], 0

jg @f

neg [value]

inc [length]

@@:

mov eax, [value]

mov ecx, 10

@@loop:

xor edx, edx

div ecx

inc [length]

cmp eax, 0

jne @@loop

@done:

mov eax, [length]

ret

endp

proc GetStringLength uses esi,\

string

xor ecx, ecx

mov esi, [string]

.loop:

cmp byte[esi], 0

je @f

inc esi

inc ecx

jmp .loop

@@:

mov eax, ecx

ret

endp

proc Time.Get

invoke GetSystemTime, systemTime

mov eax, [systemTime + SYSTEM\_TIME.wMinute]

ret

endp

proc FloatToStr uses esi, value, precision

locals

intValue dd ?

intString dd ?

intStringL dd ?

afterPointString dd ?

afterPointStringL dd ?

afterPointValue dd ?

ten dd 10.0

resultString dd ?

resultStringL dd ?

endl

;int3

fld [value]

fistp [intValue]

stdcall IntToStr2, [intValue]

mov [intString], eax

stdcall GetNumberLength, [intValue]

mov [intStringL], eax

fld [value]

fild [intValue]

fsubp

fstp [afterPointValue]

mov ecx, [precision]

fld1

.lp:

fmul [ten]

loop .lp

fmul [afterPointValue]

fistp [afterPointValue]

stdcall IntToStr, [afterPointValue]

mov [afterPointString], eax

stdcall GetNumberLength, [afterPointValue]

mov [afterPointStringL], eax

mov eax, [intStringL]

add eax, [afterPointStringL]

inc eax

mov [resultStringL], eax

inc eax

malloc eax

mov [resultString], eax

mov ecx, [intString]

memcpy eax, ecx, [intStringL]

mov esi, [resultString]

add esi, [intStringL]

mov byte[esi], '.'

inc esi

mov ecx, esi

mov eax, [afterPointString]

memcpy ecx, eax, [afterPointStringL]

mov esi, [resultString]

add esi, [resultStringL]

mov byte[esi], 0

mov eax, [resultString]

mov ecx, [resultStringL]

ret

endp

proc GetRandomNumber minValue, maxValue

push ebx ecx edx

mov eax, [seed]

or eax, eax

jnz @2

@1:

invoke GetTickCount

or eax, eax

jz @1

@2:

xor edx, edx

mov ebx, 127773

div ebx

push eax

mov eax, 16807

mul edx

pop edx

push eax

mov eax, 2836

mul edx

pop edx

sub edx, eax

mov eax, edx

mov [seed], edx

xor edx, edx

mov ebx, [maxValue]

sub ebx, [minValue]

inc ebx

div ebx

mov eax, edx

add eax, [minValue]

pop edx ecx ebx

ret

endp

proc UI.SetActiveButton uses esi, buttons, number

mov eax, [number]

shl eax, 2

mov esi, [buttons]

add esi, eax

mov eax, [esi]

mov esi, eax

mov [esi + Button.isActive], 1

mov [esi + Button.textColor], WHITE\_COLOR

stdcall Object.SetPosition, carRect, 0.0, [esi + Button.position.y], 0.0

;stdcall Rectangle.Create, carRect, -10.0, [esi + Button.position.y], 1.0, 1.0

ret

endp

proc UI.SetDeactiveButton uses esi, buttons, number

mov eax, [number]

shl eax, 2

mov esi, [buttons]

mov eax, [esi + eax]

mov esi, eax

mov [esi + Button.isActive], 0

mov [esi + Button.textColor], GRAY\_COLOR

ret

end

proc Rectangle.Create uses esi ,\

objPtr, x, y, width, heigth

locals

vao dd ?

vbo dd ?

tao dd ?

buffer dd ?

char db 0

charSize dd 0.0625

sixteen dd 16.0

x1 dd 0

y1 dd 4

left dd ?

right dd ?

top dd ?

bottom dd ?

endl

malloc 32

mov [buffer], eax

mov esi, eax

mov eax, [x]

mov dword[esi], eax

mov eax, [y]

mov dword[esi + 4], eax

add esi, 8

fld [x]

fadd [width]

fstp dword[esi]

add esi, 4

mov eax, [y]

mov dword[esi], eax

add esi, 4

fld [x]

fadd [width]

fstp dword[esi]

add esi, 4

fld [y]

fadd [heigth]

fstp dword[esi]

add esi, 4

mov eax, [x]

mov dword[esi], eax

add esi, 4

fld [y]

fadd [heigth]

fstp dword[esi]

lea eax, [vao]

invoke glGenVertexArrays, 1, eax

invoke glBindVertexArray, [vao]

lea eax, [vbo]

invoke glGenBuffers, 1, eax

invoke glBindBuffer, GL\_ARRAY\_BUFFER, [vbo]

invoke glBufferData, GL\_ARRAY\_BUFFER, 32, [buffer], GL\_DYNAMIC\_DRAW

invoke HeapFree, [hHeap], ebx, [buffer]

invoke glEnableVertexAttribArray, 0

invoke glVertexAttribPointer, 0, 2, GL\_FLOAT, GL\_FALSE, 0, 0

lea eax, [tao]

invoke glGenBuffers, 1, eax

invoke glBindBuffer, GL\_ARRAY\_BUFFER, [tao]

invoke glBufferData, GL\_ARRAY\_BUFFER, 32, textCoords, GL\_DYNAMIC\_DRAW

invoke glEnableVertexAttribArray, 1

invoke glVertexAttribPointer, 1, 2, GL\_FLOAT, GL\_FALSE, 0, 0

invoke glBindVertexArray, 0

mov esi, [objPtr]

mov eax, [vao]

mov [esi + Object.VAO], eax

;add esi, Object.transform

mov [esi + Object.transform.position.x], 0.0

mov [esi + Object.transform.position.y], 0.0

mov [esi + Object.transform.position.z], 0.0

mov [esi + Object.transform.rotation.x], 0.0

mov [esi + Object.transform.rotation.y], 0.0

mov [esi + Object.transform.rotation.z], 0.0

mov [esi + Object.transform.scale.x], 1.0

mov [esi + Object.transform.scale.y], 1.0

mov [esi + Object.transform.scale.z], 1.0

mov [esi + Object.q.x], 0.0

mov [esi + Object.q.y], 0.0

mov [esi + Object.q.z], 0.0

mov [esi + Object.q.w], 0.0

ret

endp