Федеральное агентство по образованию

Государственное образовательное учреждение высшего профессионального образования

Ижевский государственный технический университет

#### Кафедра АСОИУ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

по курсу «Системное программное обеспечение»

на тему «Проектирование трехмерной сцены в нестандартном графическом режиме»

Выполнила студентка Ворсина Н.А.

Руководитель проекта Ложкин А.Г.

Ижевск, 2009

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ

1. ОБЗОР СУЩЕСТВУЮЩИХ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ

1.1 Обзор языков программирования Pascal, Delphi

1.2 Обзор языка программирования Assembler

1.3 Обзор языка программирования С

1.4 Обоснование выбора языка программирования

2. МАТЕМАТИЧЕСКАЯ МОДЕЛЬ

2.1 Формулы поверхностей

2.2 Матрицы основных преобразований

2.3 Матрицы перспективных преобразований

3. ОПИСАНИЕ АЛГОРИТМА РАБОТЫ ПРОГРАММЫ

ЗАКЛЮЧЕНИЕ

СПИСОК ЛИТЕРАТУРЫ

ПРИЛОЖЕНИЯ

**ВВЕДЕНИЕ**

Традиционно программное обеспечение подразделяют на два класса:

1. Системное программное обеспечение (СПО);

2. Прикладное (пользовательское) программное обеспечение (ППО).

Прикладные программы предназначены для решения функциональных задач, они выполняют обработку информации различных предметных областей. Это самый многочисленный класс программных продуктов.

Системное программное обеспечение (System Software) - совокупность программ и программных комплексов для обеспечения работы компьютера и сетей ЭВМ.

СПО управляет ресурсами компьютерной системы и позволяет пользователям программировать в более выразительных языках, чем машинных язык компьютера. Состав СПО мало зависит от характера решаемых задач пользователя. Системное программное обеспечение предназначено для:

* создания операционной среды функционирования других программ (другими словами, для организации выполнения программ);
* автоматизации разработки (создания) новых программ;
* обеспечения надежной и эффективной работы самого компьютера и вычислительной сети;
* проведения диагностики и профилактики аппаратуры компьютера и вычислительных сетей;
* выполнения вспомогательных технологических процессов (копирование, архивирование, восстановление файлов программ и баз данных и т.д.).

В основном в дисциплине «Системное программное обеспечение» должно уделяться внимание операционным системам, средам и системам программирования.

В англоязычной технической литературе термин System Software (системное программное обеспечение) означает программы и комплексы программ, являющимися общими для всех, кто совместно использует технические средства компьютера, и применяемые как для автоматизации разработки (создания) новых программ, так и для организации выполнения программ существующих. [1; стр.17]

Системное программное обеспечение предназначено для разработки различных программ и для поддержки работы всего программного обеспечения.

Системное программное обеспечение делится на два основных класса:

* операционные системы;
* языки программирования.

Операционной системой называется программный продукт, позволяющий использовать прикладное программное обеспечение.

Операционные системы делится:

1. По числу пользователей: на однопользовательские и многопользовательские;
2. По интерфейсу: графические и текстовые;
3. По количеству одновременно решаемых задач: однозадачные и мультизадачные.

**1. ОБЗОР СУЩЕСТВУЮЩИХ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ**

Языки программирования делятся: на языки высокого уровня и низкого уровня; процедурные, функциональные и логические языки программирования.

Процедурные языки наиболее распространенные и наиболее используемые для системного обеспечения.

Логические языки программирования в отличие от процедурных используются для создания систем искусственного интеллекта. Главной особенностью данной группы языков является возможность определения собственных операций над собственными типами данных. Как известно, в процедурных языках, можно только определить собственный тип данных.

Функциональные языки работают с различными функциональными данными. Главный используемый тип данных в них списки. Основной особенностью данной группы языков является широкое использование рекурсии и переопределение текста программы в процессе ее выполнения. То есть исходный текст программы может быть интерпретирован как некоторый список данных.

**1.1 Обзор языков программирования Pascal, Delphi**

Существует ряд объективных причин, обусловивших выдающийся успех языка Pascal. Среди них в первую очередь необходимо указать следующие:

Язык в естественной и элегантной форме отразил важнейшие современные концепции технологии разработки программ:

* развитая система типов,
* ориентация на принципы структурного программирования,
* поддержка процесса пошаговой разработки.

Благодаря своей компактности, концептуальной целостности и ортогональности понятий, а также удачному первоначальному описанию, предложенному автором языка, Pascal оказался весьма легок для изучения и освоения. В противоположность громоздким многотомным описаниям таких языков, как PL/l, Cobol, FORTRAN, достаточно полное описание языка Pascal занимает около 30 страниц текста, а его синтаксические правила можно разместить на одной странице.

Несмотря на относительную простоту языка, он оказался пригоден для весьма широкого спектра приложений, в том числе для разработки очень больших и сложных программ, например, операционных систем.

Pascal весьма технологичен для реализации практически для всех, в том числе и нетрадиционных, машинных архитектур. Утверждается, что разработка Pascal-транслятора «почти не превышает по трудоемкости хорошую дипломную работу выпускника вуза». Благодаря этому для многих ЭВМ существует несколько различных реализаций языка, отражающих те или иные практические потребности программистов.

Язык Pascal стандартизован во многих странах. В 1983 году был принят международный стандарт (ISO 7185:1983) [2]

При описании достоинств Паскаля нужно сказать, что он является прототипом для языков нового поколения (среди которых следует отметить языка Ada и Modula-II). Паскаль дает очень многое в понимании сущности программирования, он прививает хороший стиль программирования, тщательную разработку алгоритма. Преимущества этого языка особенно ощутимы при написании достаточно сложных и мобильных (т.е. легко переносимых на другие РС) программ.[3]

Несколько лет назад рядовому программисту оставалось только мечтать о создании собственных программ, работающих в среде Windows, т. к. единственным средством разработки был Borland C++ for Windows, явно ориентированный на профессионалов, обладающих серьезными знаниями и опытом.

Бурное развитие вычислительной техники, потребность в эффективных средствах разработки программного обеспечения привели к появлению систем программирования, ориентированных на так называемую «быструю разработку», среди которых можно выделить Borland Delphi и Microsoft Visual Basic. В основе систем быстрой разработки (RAD-систем, Rapid Application Development — среда быстрой разработки приложений) лежит технология визуального проектирования и событийного программирования, суть которой заключается в том, что среда разработки берет на себя большую часть рутинной работы, оставляя программисту работу по конструированию диалоговых окон и функций обработки событий. Производительность программиста при использовании RAD-систем -фантастическая!

Delphi – это среда быстрой разработки, в которой в качестве языка программирования используется язык Delphi. Язык Delphi – строго типизированный объектно-ориентированный язык, в основе которого лежит хорошо знакомый программистам Object Pascal.

**1.2 Обзор языка программирования Assembler**

системный программный трехмерный математический

Язык ассемблера фактически представляет собой машинный язык (язык процессора), где коды команд заменены именами. Человек лучше ориентируется в именах, чем в числах, поэтому язык ассемблера проще для понимания, чем машинный язык. Другим упрощением языка ассемблера по отношению к машинному является использование меток вместо конкретных адресов. Это значительно упрощает работу, т.к. не нужно думать, по какому адресу расположена та или иная команда или данные. Вот два основных момента, которые определяют язык ассемблера и отличают его от машинного языка.

Дальнейшее развитие ассемблера шло по пути совершенствования макросредств. Ассемблер, имеющий в своем распоряжении макросредства, называют макроассемблером.[4; стр.12]

Когда-то ассемблер был языком, без знания которого нельзя было заставить компьютер сделать что-либо полезное. Постепенно ситуация менялась. Появлялись более удобные средства общения с компьютером. Но, в отличие от других языков, ассемблер не умирал, более того он не мог сделать этого в принципе. Почему? В поисках ответа попытаемся понять, что такое язык ассемблера вообще.

Если коротко, то язык ассемблера – это символическое представление машинного языка. Все процессы в машине на самом низком, аппаратном уровне приводятся в действие только командами (инструкциями) машинного языка. Отсюда понятно, что, несмотря на общее название, язык ассемблера для каждого типа компьютера свой. Это касается и внешнего вида программ, написанных на ассемблере, и идей, отражением которых этот язык является.

По-настоящему решить проблемы, связанные с аппаратурой (или даже, более того, зависящие от аппаратуры как, к примеру, повышение быстродействия программы), невозможно без знания ассемблера.

Программист или любой другой пользователь может использовать любые высокоуровневые средства, вплоть до программ построения виртуальных миров и, возможно, даже не подозревать, что на самом деле компьютер выполняет не команды языка, на котором написана его программа, а их трансформированное представление в форме скучной и унылой последовательности команд совсем другого языка — машинного.

А теперь представим, что у такого пользователя возникла нестандартная проблема или просто что-то не заладилось. К примеру, его программа должна работать с некоторым необычным устройством или выполнять другие действия, требующие знания принципов работы аппаратуры компьютера. И вот здесь-то и начинается совсем другая история.... Каким бы умным ни был программист, каким бы хорошим ни был язык, на котором он написал свою чудную программу, без знания ассемблера ему не обойтись. И не случайно практически все компиляторы языков высокого уровня содержат средства связи своих модулей с модулями на ассемблере либо поддерживают выход на ассемблерный уровень программирования.

Этот язык точно отражает все особенности машинного языка. Именно поэтому, в отличие от языков высокого уровня, язык ассемблера для каждого типа компьютера свой. Из всего вышесказанного можно сделать вывод, что, так как язык ассемблера для компьютера «родной», то и самая эффективная программа может быть написана только на нем (при условии, что ее пишет квалифицированный программист). Здесь есть одно маленькое «но»: это очень трудоемкий, требующий большого внимания и практического опыта процесс. Поэтому реально на ассемблере пишут в основном программы, которые должны обеспечить эффективную работу с аппаратной частью. Иногда на ассемблере пишутся критичные по времени выполнения или расходованию памяти участки программы. Впоследствии они оформляются в виде подпрограмм и совмещаются с кодом на языке высокого уровня.

**1.3 Обзор языка программирования С**

Сотрудник фирмы Bell Labs Деис Ритчи создал язык С в 1972 году во время совместной работы с Кеноном Томпсоном, как инструментальное средство для реализации операционной системы Unix, однако популярность этого языка быстро переросла рамки конкретной операционной системы и рамки конкретных задач системного программирования. В настоящее время любая инструментальная операционная система не может считаться полной, если в ее состав не входит компилятор языка С.

Ритчи не выдумал С просто из головы – прообразом служил язык Би разработанный Томпсоном. Язык программирования С был разработан как инструмент для программистов-практиков. В связи с этим целью его автора было создание удобного и полезного во всех отношениях языка.

С является орудием системного программиста и позволяет глубоко влезать а самые тонкие механизмы обработки информации на ЭВМ. Хотя язык требует от программиста высокой дисциплины, он не строг в формальных претензиях и допускает краткие формулировки.

С – современный язык. Он включает в себя те управляющие конструкции, которые рекомендованы теорией и практикой программирования. Его структура побуждает программиста использовать в своей работе нисходящее проектирование, структурное программирование и пошаговую разработку модулей.

С – эффективный язык его структура позволяет наилучшим образом использовать возможности современных ПЭВМ. Программирование на этом языке отличается компактностью и быстротой исполнения. С – переносимый или мобильный язык. Это означает, что программа написанная на этом языке для одной вычислительной системы, может быть перенесена с минимальными изменениями на другую.

С – мощный и гибкий язык. Большая часть операционной системы Unix, компиляторы и интерпретаторы языков фортран, Паскаль, Лисп, и Бейсик написаны именно с его помощью. С – удобный язык. Он достаточно структурирован, чтобы поддерживать хороший стиль программирования и вместе с тем не связан жесткими ограничениями.

В некотором смысле С – самый универсальный, т.к. кроме набора средств присущих современным языкам программирования высокого уровня ( Структурность, модульность, определенные типы данных), в него включены средства программирования практически на уровне ассемблера. Большой набор операторов и средств требуют от программиста осторожности, аккуратности и хорошего знания языка со всеми его преимуществами и недостатками.

Язык С++ появился в начале 80-х годов. Созданный Бьерном Страуструпом с первоначальной целью избавить себя и своих друзей от программирования на ассемблере, С или различных других языках высокого уровня. Очевидно, что больше всего С++, позаимствовал из Языка С, а также непосредственного его предшественника BCPL. Эти заимствования обеспечили С++ мощными средствами низкого уровня, позволяющие решать сложные задачи системного программирования. Но что в первую очередь отличает С++ от С – это разная степень внимания к типам и структурам данных. Это связано с появлением понятий класса, производного класса и виртуальной функции, перенятых в свою очередь из языка Симула 67. Это делает в С++ более эффективные возможности для контроля типов и обеспечивает модульность программы.

По мнению автора языка, различие между идеологией С и С++ заключается примерно в следующем: программа на С отражает «способ мышления» процессора, а С++ – способ мышления программиста. Отвечая требованиям современного программирования, С++ делает акцент на разработке новых типов данных наиболее полно соответствующих концепциям выбранной области знаний и задачам приложения. Класс является ключевым понятием С++. Описание класса содержит описание данных, требующихся для представления объектов этого типа и набор операций для работы с подобными объектами.

В отличие от традиционных структур С и Паскаля, членами класса являются не только данные, но и функции. Функции – члены класса имеют привилегированный доступ к данным внутри объектов этого класса и обеспечивают интерфейс между этими объектами и остальной программой. При дальнейшей работе совершенно не важно помнить о внутренней структуре класса и механизме работы встроенных функций. В этом смысле класс подобен электрическому прибору – мало кто знает о его устройстве, но все знают как им пользоваться.

Язык С++ является средством объектного проектирования и реализации программ, которая в текущем десятилетии, скорее всего, заменит традиционное процедурное программирование. Главной целью создателя языка было оснащения языка С++ конструкциями, позволяющими увеличить производительность труда программистов и облегчить процесс овладения большими программными продуктами.

Абстракция, реализация, наследование и полиморфизм являются необходимыми свойствами которыми обладает язык С++, благодаря чему он не только универсален, как и язык С, но и является объектным языком.[3]

В последнее время С и C++ становятся наиболее используемыми языками при разработке коммерческих и бизнеc-приложений. Эти языки устраивают многих разработчиков, но в действительности не обеспечивают должной продуктивности разработки. К примеру, процесс написания приложения на C++ часто занимает гораздо больше времени, чем разработка эквивалентного приложения на Visual Basic. Сейчас существуют языки, увеличивающие продуктивность разработки за счет потери в гибкости, которая так привычна и необходима программистам на C/C++. Подобные решения весьма неудобны для разработчиков и нередко предлагают значительно меньшие возможности. Эти языки также не ориентированы на взаимодействие с появляющимися сегодня системами и очень часто не соответствуют существующей практике программирования для Web. Многие разработчики хотели бы использовать современный язык, который позволял бы писать, читать и сопровождать программы с простотой Visual Basic и в то же время давал мощь и гибкость C++, обеспечивал доступ ко всем функциональным возможностям системы, взаимодействовал с существующими программами и легко работал с возникающими Web-стандартами. Учитывая все подобные пожелания, Microsoft разработала новый язык – С#. Он имеет массу преимуществ: простота, объектная ориентированность, типовая защищенность, «сборка мусора», поддержка совместимости версий и многое другое. Данные возможности позволяют быстро и легко разрабатывать приложения. При создании С# его авторы учитывали достижения многих других языков программирования: C++, С, Java, Visual Basic и т.д. Надо заметить, что поскольку С# разрабатывался что называется «с нуля», у его авторов была возможность не переносить в него все неудачные особенности любого из предшествующих языков. В результате получился действительно простой, удобный и современный язык, который по мощности не уступает C++, но существенно повышает продуктивность разработок.[5, стр. 13]

**1.4 Обоснование выбора языка программирования**

Таким образом, выбирая язык программирования, нужно соотнести возможности языка и сложность разработки на нем программы. Рассмотрев 3 языка программирования, я остановила свой выбор на языке Turbo Pascal. Язык Pascal является наилучшим выбором для этой курсовой работы, так как он достаточно прост и может обеспечить достаточную скорость выполнения программы. Конечно, программа на чистом ассемблере работала бы быстрее, но на ее разработку пришлось бы потратить во много раз больше времени, к тому же эту разницу в скорости было бы не так-то просто заметить. Именно поэтому язык Pascal – наилучший выбор в данном случае.

**2. МАТЕМАТИЧЕСКАЯ МОДЕЛЬ**

**2.1 Формулы поверхностей**

Для всех точек цилиндра справедливо[6]:

x = r\*cos(t),

y = r\*sin(t),

z = z,

где x, y, z ∈ ℝ – координатные оси,

t ∈ ℝ –угол к оси X,

r ∈ ℝ + – радиус основания цилиндра.

Для всех точек сегмента шара справедливо[6]:

x = (r-a)\*cos(t),

y = (r-a)\*sin(t),

z = r\*sin(γ),

где x, y, z ∈ ℝ – координатные оси, z ∉ ℝ - ,

t, γ ∈ ℝ –углы к координатным осям , r ∈ ℝ + – радиус шара,

a ∈ [0..r) – расстояние от центра шара до основания сегмента шара.

Формула боковой поверхности 7-гранной пирамиды имеет вид[6]:

Ai\*x + Bi\*y + Ci\*z + Di =0,

где Ai, Bi, Ci, Di, x, y, z ∈ ℝ,

i=6 – номер точки основания.

Основание 7-гранной пирамиды можно задать следующим образом[6]:

x=r\*cos(t),

y=r\*sin(t),

z=z0,

где x, y, z ∈ ℝ – координатные оси,

t ∈ ℝ – угол в оси Х, t меняется с шагом π/3.

r ∈ ℝ + – радиус описанной окружности.

**2.2 Матрицы основных преобразований**

Матрица преобразований переноса на вектор Т(x0,y0,z0)[7]:

,

где x0, y0, z0 ∈ ℝ – начало координатных осей.

Матрица преобразования поворота относительно оси X на угол α[7]:

,

где α ∈ ℝ – угол к оси Х.

Матрица преобразования поворота относительно оси У на угол β[7]:

,

где β ∈ ℝ – угол к оси Y.

Матрица преобразования поворота относительно оси Z на угол γ[7]:

,

где γ ∈ ℝ – угол к оси Z.

Матрица преобразования масштаба на вектор Е(ex,ey,ez)[7]:

,

где ex,ey,ez ∈ ℝ+ – координаты радиус-вектора.

**2.3 Матрицы перспективных преобразований**

Матрицы преобразования перспективного проектирования с одной точкой схода[7]:

,

,

,

где lx, ly, lz∈ℝ –расстояние по оси X, Y, Z соответственно, на котором находится точка проецирования.

Матрицы преобразования перспективного проектирования с двумя точками схода[7]:

,

,

,

где lx, ly, lz∈ℝ –расстояние по оси X, Y, Z соответственно, на котором находится точка проецирования.

Матрица преобразования перспективного проектирования в произвольном месте пространства[7]:

 ,

где lx, ly, lz∈ℝ –расстояние по оси X, Y, Z соответственно, на котором находится точка проецирования.

**3. ОПИСАНИЕ АЛГОРИТМА РАБОТЫ ПРОГРАММЫ**

На рис. 3.1 представлена блок-схема головной программы.

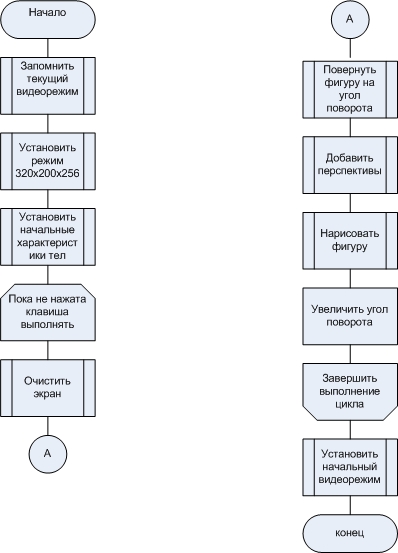


Рис. 3.1 Блок-схема головной программы

С самого начала программа производит подготовительные действия (такие как запоминание текущего видеорежима, установка нового видеорежима). Затем происходит установка начальных характеристик тел (7-гранной пирамиды, шарового сегмента и цилиндра). То есть заполняются массивы точек и линий. После этого происходит циклическое выполнение следующих действий: очистка всего экрана, поворот всей фигуры на угол поворота, добавление перспективы (все эти действия меняют вышеупомянутые массивы), рисование полученной фигуры (вывод точек, из которых состоит фигура производится непосредственно в видеопамять с помощью библиотеки DOS Turbo Pascal), увеличение угла поворота. Когда пользователь нажмет любую клавишу на клавиатуре, программа установит первоначальный графический режим и завершит свою работу.

Рассмотрим процедуру отрисовки линии. Для рисования линии использовался алгоритм Брезенхема (рис. 3.2).



Рис 3.2 Блок-схема процедуры отрисовки линии

На рис.3.3 изображена блок-схема поворота всей фигуры на угол поворота. Сначала фигура поворачивается на нулевой угол относительно оси X, затем на 15 градусов относительно оси Y, затем на 45 градусов относительно Z.





Рис. 3.3 Блок-схема процедуры поворота всей фигуры





Рис 3.3

**ЗАКЛЮЧЕНИЕ**

Целью данной курсовой работы было проектирование динамической трехмерной сцены в нестандартном графическом режиме 320х200 точек на 256 цветов. Необходимо было проделать немалую работу (ознакомиться с проективными преобразованиями, проекциями на плоскость, с преобразованиями углами поворота; составить математическую модель).

Данный проект представляется автору весьма важным, так как может найти свое применение в образовании (он может быть использован для моделирования трехмерных фигур и демонстрации их студентам). К тому же он может быть интересен людям, интересующимся компьютерной графикой и системами автоматизированного проектирования.

**СПИСОК ЛИТЕРАТУРЫ**

1. А.Е.Гордеев, Ю.В. Молчанов. Системное программное обеспечение. СПб.: – Питер, 2003. –736 стр.
2. Общие характеристики основных языков программирования [Электронный ресурс]: – Режим доступа: http://sapr.mgsu.ru/biblio/ibm/contents/jaziki.htm
3. В.Ю.Пирогов. ASSEMBLER.: – Москва, изд. Нолидж, 2001. – 848 стр.
4. Лабор В.В. Си Шарп: Создание приложений для Windows– Мн.: Харвест, 2003. –384 стр.
5. Аналитическая геометрия / В.А.Ильин, Э.Г. Позняк. М.:Наука,1981 г.-232с.
6. Математика и САПР: В 2-х кн. Кн. 1./ Шенен П., Коснар М., Гардан И. и др.-М.: Мир, 1988.-204 стр.

**ПРИЛОЖЕНИЯ**

**Приложение 1**

**ТЕКСТ ПРОГРАММЫ**

uses dos,crt;

type

tpoint = record

x,y,z:real;

end;

tline = record

point1,point2:integer;

end;

var

points:array[0..200] of tpoint;

rpoints:array[0..200] of tpoint;

lines:array[0..320] of tline;

fakeptr:pointer;

fakeseg:word;

const

numpoints:integer=0;

numlines:integer=0;

yfx=0.8;

procedure clearscreen;

begin

fillchar(mem[fakeseg:0],64000,0);

end;

procedure showscreen;

begin

move(mem[fakeseg:0],mem[$a000:0],64000);

end;

procedure rotate(xalpha,yalpha,zalpha:real; temppoints:array of tpoint; var rpoints:array of tpoint);

var

i:integer;

begin

xalpha:=0;

yalpha:=yalpha/90;

zalpha:=-45;

if xalpha<>0 then

for i:=0 to numpoints do

begin

rpoints[i].x:=temppoints[i].x;

rpoints[i].y:=temppoints[i].y\*cos(xalpha)-temppoints[i].z\*sin(xalpha);

rpoints[i].z:=temppoints[i].y\*sin(xalpha)+temppoints[i].z\*cos(xalpha);

end;

if yalpha<>0 then

begin

if xalpha<>0 then move(rpoints,temppoints,sizeof(points));

for i:=0 to numpoints do

begin

rpoints[i].x:=temppoints[i].x\*cos(yalpha)-temppoints[i].z\*sin(yalpha);

rpoints[i].y:=temppoints[i].y;

rpoints[i].z:=temppoints[i].x\*sin(yalpha)+temppoints[i].z\*cos(yalpha);

end;

end;

if zalpha<>0 then

begin

if (xalpha<>0) or (yalpha<>0) then move(rpoints,temppoints,sizeof(points));

for i:=0 to numpoints do

begin

rpoints[i].x:=temppoints[i].x\*cos(zalpha)-temppoints[i].y\*sin(zalpha);

rpoints[i].y:=temppoints[i].x\*sin(zalpha)+temppoints[i].y\*cos(zalpha);

rpoints[i].z:=temppoints[i].z;

end;

end;

end;

procedure addperspective(v:real);

var

i:integer;

begin

for i:=0 to numpoints do

begin

rpoints[i].x:=rpoints[i].x/(1-rpoints[i].z/v);

rpoints[i].y:=rpoints[i].y/(1-rpoints[i].z/v);

end;

end;

procedure initgraph;

var

r: registers;

begin

SwapVectors;

r.ah:=$00;

r.ax:=$13;

intr($10,r);

SwapVectors;

end;

procedure closegraph;

var

r: registers;

begin

SwapVectors;

r.ah:=$00;

r.ax:=$03;

intr($10,r);

SwapVectors;

end;

procedure drawp(x,y:integer;color:byte);

begin

if (x>=0)and(x<320)and(y>=0)and(y<200)then

mem[fakeseg:y\*320+x]:=color;

end;

procedure swap(var a,b:integer);

var

c:byte;

begin

c:=a;

a:=b;

b:=c;

end;

procedure line(x1,y1,x2,y2:integer;color:byte);

var

i:integer;

xs,ys:real;

begin

if abs(x2-x1)>abs(y2-y1) then

begin

if x2-x1<>0 then ys:=(y2-y1)/(x2-x1) else ys:=0;

if x1>x2 then

begin

swap(y1,y2);swap(x1,x2);

end;

for i:=0 to abs(x2-x1) do

Drawp(x1+i,y1+round(i\*ys),color)

end

else

begin

if y2-y1<>0 then xs:=(x2-x1)/(y2-y1) else xs:=0;

if y1>y2 then

begin

swap(y1,y2);swap(x1,x2);

end;

for i:=0 to abs(y2-y1) do

Drawp(x1+round(i\*xs),y1+i,color);

end;

end;

procedure init(nums,radius:integer);

var

i,y,z:integer;

begin

numpoints:=160;

{---cilindr--------------------------------}

for i:=0 to 19 do

begin

points[i].x:=-50;

points[i].y:=radius\*cos(pi\*i/nums);

points[i].z:=radius\*sin(pi\*i/nums);

points[i+20].x:=50;

points[i+20].y:=radius\*cos(pi\*i/nums);

points[i+20].z:=radius\*sin(pi\*i/nums);

end;

{piramida}

for i:=0 to 6 do

begin

points[143+i].x:=-50;

points[143+i].y:=(radius-10)\*cos(pi\*i\*3/nums);

points[143+i].z:=(radius-10)\*sin(pi\*i\*3/nums);

end;

points[150].x:=15; points[150].y:=0; points[150].z:=0;

{segment}

y:=0;

for z:=0 to 19 do

begin

points[y\*20+z+40].x:=cos(pi\*(2)/nums)\*35+45;

points[y\*20+z+40].y:=cos(pi\*z/nums)\*sin(pi\*(y)/nums)\*40;

points[y\*20+z+40].z:=sin(pi\*z/nums)\*sin(pi\*(y)/nums)\*40;

end;

for y:=0 to 4 do

for z:=0 to 19 do

begin

points[y\*20+z+40].x:=cos(pi\*(y)/nums)\*35+45;

points[y\*20+z+40].y:=cos(pi\*z/nums)\*sin(pi\*(y)/nums)\*40;

points[y\*20+z+40].z:=sin(pi\*z/nums)\*sin(pi\*(y)/nums)\*40;

end;

numlines:=320;

{cilindr}

for i:=0 to 19 do

begin

lines[i].point1:=i;

lines[i].point2:=i+1;

lines[20+i].point1:=20+i;

lines[20+i].point2:=20+i+1;

lines[40+i].point1:=i;

lines[40+i].point2:=20+i;

end;

lines[19].point1:=19;

lines[19].point2:=0;

lines[39].point1:=39;

lines[39].point2:=20;

lines[80].point1:=19;

lines[80].point2:=39;

{pyramid}

{ lines[279].point1:=149; lines[279].point2:=143;

lines[300].point1:=159; lines[300].point2:=0;}

for i:=0 to 6 do

begin

lines[260+i].point1:=143+i; lines[260+i].point2:=144+i;

lines[280+i].point1:=143+i; lines[280+i].point2:=3\*i;

lines[300+i].point1:=143+i; lines[300+i].point2:=150;

end;

{segment}

for y:=0 to 4 do

begin

lines[79+20\*y].point1:=79+20\*y;

lines[79+20\*y].point2:=60+20\*y;

for z:=0 to 18 do

begin

lines[60+20\*y+z].point1:=40+20\*y+z;

lines[60+20\*y+z].point2:=40+20\*y+z+1;

end;

end;

for y:=0 to 3 do

for z:=0 to 19 do

begin

lines[159+20\*y+z].point1:=40+20\*y+z;

lines[159+20\*y+z].point2:=40+20\*y+z+20;

end;

for i:=0 to 19 do

begin

lines[240+i].point1:=120+i;

lines[240+i].point2:=20+i;

end;

end;

procedure draw(xs,ys:integer);

var

i:integer;

begin

for i:=0 to numlines-1 do

begin

line(xs+round(rpoints[lines[i].point1].x),

ys+round(rpoints[lines[i].point1].y\*yfx),

xs+round(rpoints[lines[i].point2].x),

ys+round(rpoints[lines[i].point2].y\*yfx),15);

end;

end;

var

alpha:integer;

begin

alpha:=0;

initgraph;

init(9,50);

getmem(fakeptr,64000);

fakeseg:=seg(fakeptr^);

while not keypressed do

begin

showscreen;

clearscreen;

rotate(alpha,alpha\*2,alpha\*0.3,points,rpoints);

addperspective(200);

draw(160,100);

inc(alpha);

end;

closegraph;

end.

**Приложение 2**

**РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ**

На рис. П.2.1 и П.2.2 представлена исходная фигура под разными углами.

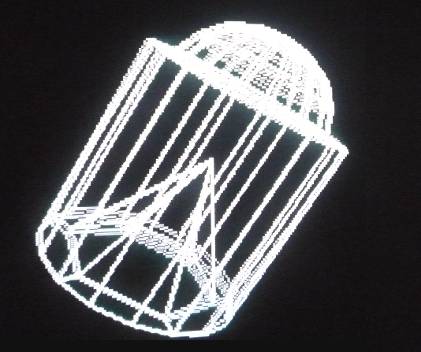
****

Рис. П.2.1

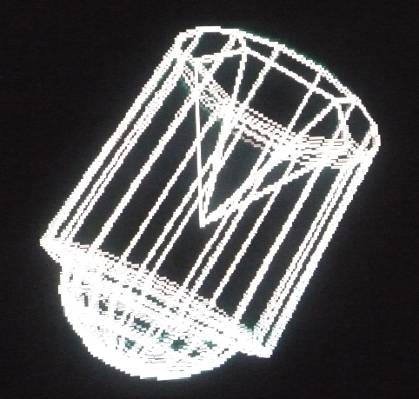


Рис. П. 2.2

Размещено на Allbest.ru