**Перечень принятых сокращений**

АРМ – автоматизированное рабочее место;

АСУ – автоматизированная система управления;

ВКО – воздушно-космическая оборона;

ИСР – интегрированная среда разработки;

КИТ – компьютерные информационные технологии;

КП – командный пункт;

КСА – комплекс средств автоматизации;

КСО – компьютерные средства обучения;

ЛБР – лицо (лица) боевого расчета;

ЛВС – локальная вычислительная сеть;

МСВС – мобильная система вооруженных сил;

НЖМД – накопитель на жестких магнитных дисках;

ОЗУ – оперативное запоминающее устройство;

ОС – операционная система;

ПС – программные средства;

ППП – пакет прикладных программ;

ПЭВМ – персональная электронно-вычислительная машина;

ПСПИ – программная система представления информации.

**Введение**

Успешное решение боевых задач, стоящих в современных условиях перед воинскими формированиями ВВС, во многом зависит не только от качества вооружения и военной техники и разработанных принципов и методов управления войсками, но и от оперативности и достоверности оценки поступающей информации.

Поступающая информация от различных источников (в большей степени от автоматизированных средств) представлена в цифровом виде, который необходимо перевести в более наглядный и информативный.

Анализ научных трудов и публикаций, посвященных поиску путей повышения эффективности восприятия и оперативной оценки поступающих данных, выявил в качестве одного из основных направлений решения обозначенной выше проблемы – внедрение в процесс боевой подготовки и учебный процесс программных средств построения трехмерных изображений данных из симметричных матричных форм.

В настоящее время существует малое количество ПСПИ. ПСПИ разрабатываются как в виде отдельных программных, так и в виде подсистем встроенных в системы проектирования. Многие из них, по заявлениям разработчиков, являются универсальными, однако ряд факторов, прежде всего программно-технического и экономического характера, не позволяют применить их для использования в ВВС. Возникает необходимость поиска нового решения, что обуславливает актуальность настоящей работы.

Цель проекта: разработать программную систему, обеспечивающую отображение и сравнение в трехмерном пространстве исходных данных из двух матричных форм.

Для достижения целевой установки в ходе работы решалась задача разработки ПС построения и сравнения трехмерных изображений по данным из двух матричных форм, интегрирующей в себе требуемые возможности аналогов и обеспечивающей получение решения за меньшее время при ограничениях на затраты.

В связи с этим была создана программа, в основу которой была взята раннее разработанная аналогичная программа. В нее внесены дополнения в виде добавления функции загрузки второй поверхности и функции сравнения поверхности образца с поверхностью с измененным рельефом. Так же появилась возможность изменения значений вершин матричной формы и сохранения этих изменений в текстовый файл.

Дипломный проект состоит из введения, 3 разделов и заключения.

В первом разделе проанализирован существующий подход к возможности отображения входной информации и определен путь повышения оперативности и достоверности. Проведен анализ возможности применения современных ПСПИ для вывода графической информации. Выделены их достоинства и недостатки. Определены факторы, препятствующих их применению. Выбран альтернативный вариант разработки. Определены требования, предъявляемые к ПСПИ, поставлена задача на их разработку.

Во втором разделе определены подходы к реализации функций ПСПИ, приведена их структура. Произведен выбор общего программного обеспечения и средств разработки. Разработана ПС построения трехмерного изображения из симметричных матричных форм.

В третьем разделе освещены вопросы особенностей эксплуатации и порядка работы с ПСПИ, проведена оценка эксплуатационно-технических характеристик, рассчитана экономическая эффективность.

1. **Общесистемный раздел**

**1.1 Анализ существующего подхода к построению трехмерного изображения данных из симметричных матричных форм и определение путей повышения уровня специализации на задаче**

**1.1.1 Анализ существующего подхода к построению трехмерного изображения данных из симметричных матричных форм**

Поступающая информация от различных источников (в большей степени от автоматизированных средств) представлена в цифровом виде, который необходимо перевести в более наглядный и информативный.

Процедура оценки поступающей информации является одним из основных элементов боевой подготовки ЛБР. В настоящее время в специализированных учебных центрах ВВС и на КП бригад ВКО оценка поступающей информации осуществляется посредством выводов графиков и таблиц.

Под выводом графиков и таблиц понимается ручной или автоматизированный пересчет поступающей информации для последующего вывода в удобочитаемом табличном виде либо в виде графиков.

Данный подход требует значительное усилие от ЛБР и затрату времени. Следовательно, данный процесс необходимо автоматизировать.

**1.1.2 Определение путей повышения достоверности оценки информации**

Существующий в настоящее время подход к представлению цифровой информации в графическом виде и табличной форме обладает низкой достоверностью получаемых результатов. Это обусловлено сложностью оценки информации из табличного вида и ограниченностью наглядности плоскостных графиков.

Исходя из этого, разработка и внедрение трехмерного представления информации, может выступить в качестве пути повышения достоверности оценки информации.

**1.2 Анализ возможности применения современных ПСПИ для оценки поступающей информации**

В отличие от традиционных способов представления, трехмерное обладает рядом преимуществ, позволяющих:

1. значительно сократить время, необходимое на обработку поступающей информации;
2. автоматизировать процесс представления;
3. повысить достоверность результата анализа графического представления информации.

Данные преимущества достигаются за счет применения трехмерного представления данных и реализации в различных программных средствах представления информации (ПСПИ).

Вместе с тем, конкретных программных решений промышленного производства, официально разрешенных для применения в настоящее время нет (тестирование проводится бланковым способом). Однако эту задачу можно рассматривать как частный случай автоматизированного представления данных, предполагающий использование специализированных программных средств представления изображения (ПСПИ).

Исходя из этого, необходимо рассмотреть существующие ПСПИ и проанализировать возможности их применения.

На данный момент существует небольшое количество ПСПИ, представленных как в виде подсистем в составе различных автоматизированных систем (АС), так и в виде автономных решений. Практически все они сильно ограничены для представления оперативной информации в силу следующих факторов:

1. функциональный фактор – ограниченные возможности по реализации функции автоматизированного представления данных;
2. программно-технический фактор – закрытые исходные коды программных продуктов не допускающие их внедрение в состав автоматизированных систем военного назначения;
3. технологический фактор – сложность в освоении и применении ПП;
4. экономический фактор – высокая стоимость ПП.

Вопросами создания ПСПИ в настоящее время занимаются как различные учебные заведения, коммерческие организации, так и отдельные программисты. Учитывая экономический фактор, для анализа существующих ПСПИ будем использовать только свободно-распространяемые (бесплатные или условно-бесплатные) решения. По причине того, что признанных лидеров среди программ такого класса нет, рассмотрим наиболее распространенные и цитируемые из них:

1. 3DMax;
2. Graphs (Sulaco);
3. Viewer (Колташов О.В.);
4. Uni3D (Товпенец Д.И.).

Приведем краткое описание функциональных возможностей и выделим достоинства и недостатки, присущие данным ПСПИ:

3DMax

Программа предназначена для создания трехмерных объектов также возможно отображение по входным матрицам.

Достоинства:

1. простой и понятный пользовательский интерфейс;
2. профессиональная программа с богатым набором инструментов.

Недостатки:

1. функциональная ограниченность работы с входными матрицами;
2. большие системные требования;
3. закрытый код программы.

Graphs

Данная программа является функционально-мощным средством автоматизированного представления трехмерного изображения, но имеет определенные недостатки.

Достоинства:

1. более специализированная на выводе из исходных матриц;
2. просмотр и печать результатов тестирования.

В качестве недостатка следует отметить существенные недоработки пользовательского интерфейса, закрытый код.

Viewer

Данный комплекс более специализирован на математических формулах, но так же имеется функция отображения из матриц.

Достоинства:

1. открытый код программы;
2. среда разработки Borland Delphi 7.0.

Недостатки:

1. длительная обработка данных;
2. большие системные требования;
3. плохо продуманный пользовательский интерфейс.

Uni3D

Программа обладает удобным пользовательским интерфейсом. Однако минимальная функциональность и отсутствие контроля входных данных, не дают возможность говорить о логической завершенности.

Достоинства:

удобный интерфейс в большинстве модулей комплекса.

Недостатки:

1. отсутствует контроль корректности входных данных;
2. возникают периодические сбои работы программы.

Сравнительная характеристика функциональных возможностей рассматриваемых ПСКТ представлена в таблице 1.1

Таблица 1.1 – Сравнительная характеристика функциональных возможностей программных средств компьютерного тестирования

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| №  п/п | Функциональная возможность | 3DMax | Graphs | Viewer | Uni3D |
| 1. | специализированность на входных матрицах | - | - | + | + |
| 2. | открытый код | - | - | + | + |
| 3. | контроль входных данных | + | + | + | - |
| 4. | устойчивость к сбоям | + | + | + | - |
| 5. | удобный интерфейс | + | - | - | + |
| 6. | функциональность | + | - | - | + |
| 7. | большие системные требования | + | + | - | - |
| 8. | быстрая обработка данных | + | + | - | - |
| 9. | сравнительно низкая стоимость | - | - | + | + |

Альтернативным вариантом реализации ПСПИ может быть самостоятельная разработка с использованием одного из языков программирования высокого уровня, интегрированных средств разработки (ИСР) в сочетании с использованием универсальных средств (текстовых редакторов, графических редакторов, редакторов электронных таблиц, редакторов презентаций, средств создания справочных систем).

**1.3 Анализ возможных вариантов реализации ПС представления информации**

В настоящее время ПСПИ реализуются с использованием двух основных подходов:

1. локальная архитектура;
2. клиент-серверная архитектура.

В данном проекте использована локальная архитектура, которая предполагает размещение всех компонентов (программных модулей) на одном компьютере. Такой подход применяется в случае невозможности организовать локальную вычислительную сеть (ЛВС) и распределение автоматизированных рабочих мест (АРМ), а так же в случае если программа не требует использования дополнительных АРМ. В данном случае этот вариант применим для решения задачи оценки информации в процессе боевой подготовки, так же в ходе и подготовки к несению боевого дежурства.

В соответствии с классификацией, определенной в ГОСТ 19.101-77 «Виды программ и программных документов»:

1. программный комплекс – это программа, состоящая из двух или более компонентов и (или) комплексов, выполняющих взаимосвязанные функции и применяемая самостоятельно или в составе другого комплекса;
2. программный компонент – это программа, рассматриваемая как единое целое, выполняющая законченную функцию и применяемая самостоятельно или в составе комплекса.

ПСПИ, разрабатываемая в настоящем проекте, состоит из одного компонента и, следовательно, является программным компонентом, применяемым самостоятельно либо в составе различных сред.

**1.4 Определение требований к ПС представления информации**

**1.4.1 Функциональные требования**

Исходя из целей, стоящих перед ПСПИ и задач, решаемых пользователями при работе с ними, основными процессами, подлежащими автоматизации, являются:

1. процесс обработки входной информации;
2. процесс взаимодействия пользователей с графической информацией.

Указанные процессы относятся к основным. Для их автоматизации в составе ПСПИ должен быть реализован следующий набор функций:

1. входной контроль данных;
2. настройка параметров представления.

Функция контроля входных данных предполагает:

1. проверка на соответствие входной информации с поставленными ограничениями;
2. остановка выполнения программы с вызовом сообщения об ошибке.

Проверка на соответствие входной информации с поставленными ограничениями предполагает сравнение поступающей информации с эталонными значениями входных данных.

В случае несоответствия входной информации поставленным ограничениям программа останавливает свое выполнение с вызовом сообщения об ошибке и ожидает загрузки корректных данных.

Функция перевода табличных данных в графический вид.

Данная функция осуществляет построение трехмерного изображения по данным входной симметричной матричной формы.

Функция сравнения графических изображений

Данная функция осуществляет сравнение формы образца с измененной матричной формой.

Функции, автоматизирующие процесс взаимодействия пользователей с графической информацией.

Взаимодействие пользователей с графической информацией обеспечивают функции:

1. настройка представления графической информации;
2. вращение трехмерного изображения;
3. определение значения точки на поверхности изображения.

Функция настройки представления графической информации выполняет следующие действия:

1. выбор текстуры поверхности изображения;
2. выбор масштаба представления.

Функция вращения трехмерного изображения позволяет вращать изображение в пространстве под любым углом и всесторонне воспринимать информацию.

Функция определения значения точки на поверхности изображения обеспечивает выдачу конкретного значения по выбранному элементу изображения.

**1.4.2 Требования к входной и выходной информации**

На основе анализа функциональных требований предъявляемых к ПСПИ можно определить для них перечень входной и выходной информации.

Входной информацией для ПС представления информации является симметричная матричная форма.

Выходной информацией для ПС представления информации является:

1. трехмерное представление входных данных;
2. трехмерное отображение результата сравнения;
3. цифровое значение отдельных областей изображения.

**1.4.3 Эргономические требования**

Для обеспечения комфортной работы пользователей с ПСПИ она должна соответствовать требованиям ГОСТ РВ 29.05.ХХХ – «Система стандартов эргономических требований и эргономического обеспечения. Общие эргономические требования».

**1.4.4 Технические требования**

Технические и программные средства, предназначенные для обеспечения функционирования разработанного комплекса ПО должны с одной стороны обеспечивать комфортную работу с программой и обеспечивать защиту, циркулирующей информации от несанкционированного доступа, а с другой – удовлетворять требованиям общего и специального программного обеспечения при заданной конфигурации вычислительных средств. Кроме того, технические требования и программные средства, предназначенные для функционирования ПС должны удовлетворять аппаратным и программным характеристикам вычислительных средств используемых в процессе боевой подготовки ЛБР и учебном процессе специализированных учебных центров.

Рекомендуемые технические характеристики ПЭВМ для функционирования ПС оценки уровня обученности ЛБР приведены в таблице 1.2.

Таблица 1.2 – Технические характеристики ПЭВМ

|  |  |
| --- | --- |
| Параметры технических средств | Значения параметров |
| Процессор | Intel Pentium II – 1600 МГц |
| ОЗУ | не менее 256 Мб |
| НЖМД | не менее 40 Гб |
| Разрешение монитора | 1024\*768 |

Кроме того, ПЭВМ должна быть оснащена клавиатурой, манипулятором типа «мышь» (или трекбол).

Для обеспечения функционирования ПС представления информации на ПЭВМ должны быть установлены ОС Windows NT, XP, Vista и пакет MS Office 2000 и выше.

**1.4.5 Требования к документации**

Документация на ПСПИ должна быть разработана в соответствии с требованиями приказа ГК ВВС 2000г. №029 «О введении в действие руководства по разработке программной продукции военного назначения в ВВС» и требованиями ГОСТ ЕСПД.

Документация на ПСПИ должна включать четыре документа:

1. постановка задачи;
2. алгоритм функционирования комплекса;
3. описание программы комплекса;
4. инструкция пользователю комплекса.

**1.5 Постановка задачи на разработку ПС представления информации**

Таким образом, в рамках дипломного проекта должны быть разработаны алгоритм и программные средства представления информации, отвечающие сформулированным выше требованиям:

Постановка задачи на создание ПСПИ разработана в соответствии с требованиями приказа ГК ВВС №029-2000г. и ГОСТ 19.201-78 и представлена в первом разделе документации на программную систему.

* 1. **Ограничения и допущения, принятые при разработке ПСПИ**

1. одновременно с программным обеспечением может работать только один пользователь;
2. для работы с ПС, пользователи должны владеть основными навыками работы с операционными системами семейства Microsoft Windows, а также с ППП Microsoft Office;
3. требования к техническим средствам приведены в пункте 1.4.4

**2. Специальный раздел**

**2.1 Определение подходов к реализации функций ПС представления информации**

Из описанных выше процессов, подлежащих автоматизации, ключевой момент составляют процессы обработки входной информации и процессы взаимодействия пользователей с графической информацией. Определим подходы к реализации функций, автоматизирующих их.

**2.1.1 Функции, автоматизирующие процесс обработки входной информации**

Для автоматизации процесса обработки входной информации в составе ПС представления информации должны быть реализованы функции:

1. контроль входных данных;
2. перевод табличных данных в графический вид.

Функция контроля входных данных предполагает:

1. проверка на соответствие входной информации с поставленными ограничениями;
2. остановка выполнения программы с вызовом сообщения об ошибке.

Проверка на соответствие входной информации с поставленными ограничениями предполагает сравнение поступающей информации с эталонными значениями входных данных.

В случае несоответствия входной информации поставленным ограничениям программа останавливает свое выполнение с вызовом сообщения об ошибке и ожидает загрузки корректных данных.

Функция перевода табличных данных в графический вид - осуществляет построение трехмерного изображения по данным входной симметричной матричной формы.

**2.1.2 Функции, автоматизирующие процесс взаимодействия пользователей с графической информацией**

Взаимодействие пользователей с графической информацией обеспечивают функции:

1. сравнение трехмерных изображений;
2. настройка представления графической информации;
3. вращение трехмерного изображения;
4. определение значения точки на поверхности изображения.

Функция сравнения трехмерных изображений позволяет проанализировать разницу, четко выраженную несколькими цветами.

Функция настройки представления графической информации выполняет следующие действия:

1. выбор текстуры поверхности изображения;
2. выбор масштаба представления.

Функция вращения трехмерного изображения позволяет вращать изображение в пространстве под любым углом и всесторонне воспринимать информацию.

Функция определения значения точки на поверхности изображения обеспечивает выдачу конкретного значения по выбранному элементу изображения.

**2.2 Структура ПС представления информации**

Исходя из задач, решаемых с помощью ПС представления информации, выполняемых ими функции и способов реализации этих функций, определим их структуру. Структура отражает состав основных программных и информационных компонентов, а также их связи друг с другом, пользователями и внешними программами. Вычленение программных компонентов производится по функциональному принципу. Информационные компоненты (массивы, блоки, файлы, фрагменты базы данных и т.д.) выделяются на основании их назначения, использования, форматов представления данных, способов доступа и других признаков.

Структурная схема разработанных программных средств приведена на рисунке 2.1



Рисунок 2.1 – Структурная схема

Основными структурными элементами ПС являются:

1. внешний источник данных – обеспечивает импорт входных данных;
2. программный модуль представления информации, который включает:
3. блок контроля и загрузки входных данных – предназначен для анализа поступающей информации и последующему импорту в программу;
4. блок перевода табличных данных в графический вид – предназначен для визуализации цифровых матричных форм;
5. блок сравнения двух матричных форм – предназначен для наложения на изображение - образец нового изображения и последующего анализа полученного результата;
6. блок настройки параметров представления информации – предназначен для установки параметров выводимого трехмерного изображения;
7. блок управления трехмерным изображением – предназначен для проведения различных действий с трехмерным изображением;
8. блок выбора определенного значения на поверхности трехмерного изображения – предназначен для уточнения значений отдельных элементов.

**2.3 Алгоритм ПС представления информации**

Описание указанных алгоритмов разработано в соответствии с требованиями ГОСТ 24.211-82 «Требования к содержанию документа» и «Описание алгоритма» приведено во второй части документации на задачу.

Условные графические обозначения в схемах алгоритмов выполнены в соответствии с требованиями ГОСТ 19.701-90 ЕСПД. «Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения в соответствии с ним».

**2.4 Выбор общего программного обеспечения и средств разработки**

В настоящий момент основным языком программирования при разработке программных средств военного назначения является С++. Однако несмотря на ряд преимуществ, присущих данному языку, он является достаточно сложным. В связи с этим, возникла необходимость выбора средства разработки, позволяющего в короткие сроки и максимально эффективно выполнить поставленную задачу. По результатам обзора современных средств быстрой разработки программного обеспечения (RAD), был сделан вывод, что в качестве такового может выступить интегрированная среда программирования Borland Delphi 7.0. Основанная на синтаксисе мощного и гибкого языка программирования Object Pascal и сочетающая в себе широкий набор функциональных возможностей данная ИСР предоставляет пользователю широкие возможности по разработке ПО.

В вооруженных силах Российской Федерации в интересах создания собственных средств общего программного обеспечения, в том числе и ОС, приказом Министра Обороны от 13 мая 2002 года № 190 принята на снабжение защищенная операционная система МСВС 3.0. Однако отсутствие подробного описания особенностей данной системы и необходимость использования ряда пакетов прикладных программ, разработанных в среде Windows, вызывает определенные затруднения при разработке ПО под данную ОС. Кроме того, на ПЭВМ, применяемых на КП бригад ВКО, установлены операционные системы семейства Windows. В связи с этим был сделан вывод о целесообразности разработки программы на основе операционной системы Microsoft Windows XP.

Исходя из вышеприведенных соображений, в качестве средства разработки ПСПИ, в рамках настоящего дипломного проекта, была определена ИСР Borland Delphi 7.0.

**2.5 Исходный текст программы**

Исходный текст программы разработан в соответствии с требованиями ГОСТ 19.401-78 и представлен в приложении 2.

**3. Эксплуатационный раздел**

**3.1 Оценка эксплуатационно-технических характеристик**

Анализ результатов оценки эксплуатационно-технических характеристик разработанной ПС построения трехмерного изображения данных из симметричных матричных форм показал, что они являются законченными программными продуктами высокого качества. Функциональные характеристики ПС соответствуют предъявляемым к ним требованиям.

Испытания ПС выявили полное соответствие его характеристик следующим стандартам:

1. ГОСТ-28195-95 “Оценка качества программных средств. Общие положения”;
2. ГОСТ Р ИСО/МЭК 9126-93 “Информационные технологии. Оценка программной продукции. Характеристики качества и руководства по их применению”.

**3.2 Оценка надежности и качества ПСПИ**

Качество любого изделия представляется набором показателей, отражающих его свойства и определяющих возможность и эффективность его применения по прямому назначению. Качество программных средств описывается совокупностью показателей-критериев, для каждого из которых должны быть определены метрики и методы их измерения. Эти критерии и метрики программ позволяют описывать их свойства как конечного продукта независимо от способа их достижения.

Имеется ряд стандартов и публикаций, определяющих основные понятия, номенклатуру критериев, методы измерения показателей и метрики качества программ. Основными стандартами в области оценки качества программных средств являются:

1. ISO 9126:1991;
2. ГОСТ 28190-89;
3. ГОСТ 28806-90.

В стандартах описываются до 20 – 30 показателей – основных критериев и приводятся определения их вычисления. Однако многие показатели имеют иллюстративный характер, и их значения определяются экспертно. В зависимости от класса и особенностей программы целесообразно выбирать различные наборы критериев, адекватные свойствам конкретного ПС.

Эти критерии должны наиболее полно отражать назначение и функциональные характеристики ПС при его применении, и их обычно не более десяти.

Среди показателей качества можно выделить две крупные группы и соответствующие им наборы критериев:

1. функциональные критерии (функциональная пригодность и удобство использования), отражающие специфику областей применения и степень соответствия программ их основному целевому назначению;
2. конструктивные критерии (надежность и эффективность использования ресурсов), отражающие эффективность использования программой ресурсов вычислительных средств (ВС), а также надежность функционирования программного обеспечения.

**3.2.1 Оценка функциональной пригодности**

Оценка функциональной пригодности включает в себя оценку:

1. функциональной корректности (корректность структуры, корректность выполнения функций, корректность взаимодействия компонент);
2. способности к взаимодействию;
3. мобильности.

Анализ исходного кода программы показал, что он имеет корректную структуру. Опытная эксплуатация ПСПИ показала, они выполняют все функции, определенные в постановке задачи в соответствии с предъявляемыми для них требованиями.

Исходный код программы требует минимального объема изменений, при дополнении или исключении отдельных функции. Следовательно, программные и информационные компоненты способны к взаимодействию.

Опытная эксплуатация проводилась на различных версиях ОС Windows и не требовала при переносе никаких дополнительных доработок программы. Этот факт свидетельствует о мобильности разработанной программы.

Из сказанного выше можно сделать вывод, что программа является функционально пригодной.

**3.2.2 Оценка удобства использования**

Удобство использования программных средств определяется по показателям понятности, обучаемости и комфортности эксплуатации.

Разработанная ПСПИ имеет четкую концепцию, обладает широкими демонстрационными возможностями и наглядностью представления возможных функций.

Подготовка пользователей к полноценной эксплуатации ПСПИ требует минимальных затрат времени.

Это обеспечивается за счет:

1. доступности и удобства использования руководств и инструкций по эксплуатации;
2. интуитивно понятного интерфейса.

Разработанная ПС является достаточно легкой в управлении. Некоторые параметры управления реализуются автоматически. Сообщения, выдаваемые пользователю в процессе работы, являются достаточно информативными. Выполнение всех функций предусмотренных ПС и регистрация результатов происходит практически мгновенно. Эргономические характеристики интерфейса соответствуют требованиям ГОСТ и обеспечивают комфортную работу пользователей.

Из сказанного выше следует, что разработанные ПС представления информации является удобной в использовании.

**3.2.3 Оценка надежности**

В соответствии с ГОСТ 13.377-75, надежность – свойства объекта выполнять заданные функции, сохраняя во времени значения установленных эксплуатационных показателей в заданных пределах, соответствующих заданным режимам и условиям использования, технического обслуживания, ремонта, хранения и транспортирования.

Надежность ПС – уровень, при котором программа удовлетворяет поставленным требованиям и пригодна для эксплуатации. Основными показателями надежности ПС являются устойчивость, восстанавливаемость.

Опытная эксплуатация ПСПИ показала её способность к безотказному функционированию после возникновения каких-либо сбоев. После перезапуска – рестарта функционирование восстанавливается полностью.

Это позволяет сделать заключение о надежности разработанной ПС представления информации.

**3.2.4 Оценка эффективности использования ресурсов**

Эффективность использования ресурсов определяется по следующим показателям:

1. временная экономичность (время реакции, пропускная способность);
2. ресурсная экономичность (занятость, используемость ресурсов).

Выполнение функций предусмотренных в ПСПИ и реакция на запросы пользователя осуществляется практически мгновенно. ПС обладают высокой пропускной способностью.

Функционирование ПСПИ сопровождается минимальной загрузкой ресурсов ПЭВМ (центрального процессора, оперативной, внешней и виртуальной памяти, каналов ввода-вывода). Таким образом, система эффективно использует ресурсы в процессе функционирования.

**3.2.5 Расчет обобщенного показателя качества ПО**

Обобщенный показатель качества ПО[[1]](#footnote-1)) (Пок) оценивается суммой показателей, входящих в него и определяется по формуле:

Пок=0,25ПЗ + 0,05Пэф + 0,15Пдок + 0,2Пн + 0,05Писп + 0,1Пэкспл + 0,15Пм + 0,05Пи, (3.1)

где:

Пз – показатель соответствия заданию на дипломный проект;

Пэф – показатель эффективности;

Пдок – показатель документированности;

Пн – показатель надежности;

Писп – показатель простоты использования;

Пэкспл – показатель удобства эксплуатации;

Пм – показатель мобильности;

Пи – показатель испытуемости.

Применительно к разработанным программным средствам оценки уровня обученности ЛБР Пок составляет:



Таким образом, исходя из сформулированных выше выводов и на основе полученного значения обобщенного показателя, можно сделать вывод о том, что разработанные в рамках дипломного проекта ПС являются качественными.

**3.3 Оценка экономической эффективности разработанных ПС представления информации**

Оценка экономической эффективности разработанной ПСПИ проводится для анализа и обоснования целесообразности их внедрения в специализированные учебные центры и воинские части.

Источниками экономической эффективности ПС является сокращение затрат и повышение эффективности процесса оценки поступающей информации.

Факторами экономической эффективности ПС являются средства реализации источников эффективности. К ним относят основные функции, выполняемые ПС.

Различают две группы методов оценки проектов, связанных с применением компьютерных информационных технологий:

1. простые (статические) методы;
2. методы дисконтирования.

В рамках настоящей работы рассматриваются простые методы оценки экономической эффективности применительно к решению задачи оперативной оценки информации с использованием ПСПИ.

Данные методы базируются на допущении равной значимости доходов и расходов в финансовой деятельности и не учитывают временной стоимости денег.

К данным методам относятся расчеты следующих показателей:

1. годового экономического эффекта Эг – основного показателя, представляющего собой всю прибыль, получаемую в результате автоматизации решения задачи.
2. годовой экономии Э – части прибыли, получаемой от снижения себестоимости документов при автоматизации процесса их разработки;
3. расчетного коэффициента эффективности капиталовложений Ер – прибыли, получаемой на один рубль, вложенный в приобретение ВТ для решения рассматриваемой задачи;
4. срока окупаемости капиталовложений Ток – временного периода за который окупятся затраты, связанные с приобретением ВТ для решения рассматриваемой задачи.

**3.4 Расчет годового экономического эффекта**

Годовой экономический эффект Эг определяется как разница затрат между базовым и оцениваемым вариантами решения задачи.

Под базовым вариантом будем понимать оценку оперативной информации традиционным способом.

Под оцениваемым вариантом будем понимать затраты, связанные с автоматизацией данной задачи.

Годовой экономический эффект рассчитывается по формуле:

Эг=Зб-Зоц (3.2)

где:

Зб – затраты по базовому варианту;

Зоц – затраты по оцениваемому варианту.

В общем случае затраты складываются из следующих компонентов:

Зрб – затраты ручного труда, связанные с работой ЛБР по оценке поступающих данных с использованием традиционного подхода;

Зр оц – затраты, связанные с частичным использованием традиционного подхода при автоматизированном решении задачи;

Зтек оц – текущие затраты, связанные с эксплуатацией задачи на ЭВМ, которые связаны с решением задачи на ВТ (заработная плата должностных лиц, обслуживающих компьютеры, электроэнергия, амортизация);

Зк оц – затраты на приобретение компьютерной техники, покупка программных продуктов для автоматизации задачи и обучение сотрудников.

С учетом вышеизложенного, формула для расчета годового экономического эффекта приобретает вид:

 (3.3)

Здесь:

 (3.4)

где:

Тр – трудоемкость одноразового решения задачи вручную;

кр – периодичность решения задачи в течение года;

tr – среднечасовая тарифная ставка должностного лица;

n – коэффициент, учитывающий премии и доклады;

R – коэффициент отчисления от фонда оплаты труда;

Ен – нормативный коэффициент эффективности капитальных затрат, принимается равным 0.15.

Тр=0.32 ч; кр = 500 опер.; tr = 107 руб.; n = 1,4; R =1,36.



 (3.5)

где:

Титр – трудоемкость одноразового решения задачи автоматизировано.



 (3.6)

где:

Тпр – продолжительность разработки программы в мес;

кл – количество человек, участвующих в разработке;

Fм – месячный фонд времени работы;

tr – среднечасовая тарифная ставка разработчика.

Тпр=1 мес.; кл = 1 чел.; Fм = 80 час. (при норм. усл.); tr = 100 руб.;



 (3.7)

где:

Тм – время машинной реализации задачи;

Сэвм – стоимость ЭВМ, программных продуктов и обучения

должностных лиц;

Fэвм – действительный фонд времени ЭВМ в течение года.

Тм=0.015 час; Сэвм=15000 руб.; Fэвм=2880 час;

Зк оц=(0.015 500 15000)/2880 ≈ 39 руб.

 (3.8)

где:

Зитр – заработная плата должностных лиц, обслуживающих

ЭВМ;

А – амортизация (A=0.12 Сэвм=1800 руб.);

Зэл – затраты на электроэнергию.

 (3.9)

где:

Fгод – годовой фонд заработной платы должностных лиц, обслуживающих ЭВМ;

Зитр=(216000/2880) 0.015 500 1.4 1.36 =1071 руб.

 (3.10)

где:

Кис – коэффициент использования энергоустановок по

мощности;

СУМэвм – суммарная установленная мощность ЭВМ;

Цэл – стоимость одного кВт/ч энергии.

Кис=0.9; СУМэвм=100; Цэл=2.5 руб;

Зэл=0.9 100 0.015 500 2.5=1688 руб;

Зтек оц=1071+1800+1688=4559 руб;

Основными затратами по автоматизации будут:

1. обучение должностных лиц обеспечивающих работу ЭВМ;
2. обслуживание компьютерной техники.

Затраты по базовому варианту:

Зрб=32600 руб.;

Затраты по оцениваемому варианту:

Зр оц = 1530 руб.

Зтек оц=4559 руб.;

Зк оц=39 руб.

Зпр оц =15232 руб.

Тогда годовой экономический эффект будет:

Эг=(Зрб-Зр оц-Зтек оц)-Ен (Зк оц+Зпр оц)=(32600-1530-4559)-0.15 (39+15232)=26511-2291=24220 руб.

Расчет годовой экономии.

 (3.11)

Расчет расчетного коэффициента эффективности капиталовложений.

 (3.12)

Внедрение КИТ является эффективным если Ер>Ен, т. е. расчетный коэффициент эффективности больше нормативного.

Расчет срока окупаемости капиталовложений.

 (3.13)

Таким образом, на основании результатов проведенной оценки можно сделать вывод, что внедрение разработанной в рамках дипломного проекта ПС представления информации в специализированные учебные центры и воинские части является экономически эффективным.

**3.4 Рекомендации по обеспечению безопасной жизнедеятельности при работе с ПСПИ**

Эксплуатация разработанной ПСПИ предполагается на ПЭВМ, которая является источником вредных факторов, негативно влияющих на безопасность жизнедеятельности пользователей. Для снижения воздействия этих факторов при организации эксплуатации системы необходимо руководствоваться описанными ниже рекомендациями.

**3.4.1 Рекомендации по обеспечению электробезопасности**

Источником питающего напряжения ПЭВМ должна быть сеть переменного тока с напряжением 220 В, на которую распространяется ГОСТ 25861-83.

Для предупреждения поражений электрическим током необходимо:

1. чётко и в полном объёме выполнять правила производства работ и правила технической эксплуатации;
2. исключить возможность доступа оператора к частям оборудования, работающим под опасным напряжением, неизолированным частям, предназначенным для работы при малом напряжении и не подключенным к защитному заземлению;
3. применять изоляцию, служащую для защиты от поражения электрическим током, выполненную с применением прочного сплошного или многослойного изоляционного материала, толщина которого обусловлена типом обеспечиваемой защиты;
4. подводить электропитание к ПЭВМ от розетки здания при помощи специальной вилки с заземляющим контактом;
5. защитить от перегрузок по току, рассчитывая на мощность, потребляемую от сети; а также защитить от короткого замыкания оборудование, встроенное в сеть здания;
6. надёжно подключить к заземляющим зажимам металлические части, доступные для оператора, которые в результате повреждения изоляции могут оказаться под опасным напряжением;
7. проверить, что защитный заземляющий проводник не имеет выключателей и предохранителей и надёжно изолирован.

**3.4.2 Рекомендации по обеспечению пожарной безопасности**

Пожарная безопасность помещений, имеющих электрические сети, регламентируется ГОСТ 12.1.033-81, ГОСТ 12.1.004-85. Работа оператора ЭВМ должна вестись в помещении, соответствующем категории Д. Огнестойкость здания должна соответствовать СНиП 2.01.02-85 .

В конструкции дисплеев должны использоваться специальные разъемы, уменьшающие переходное сопротивление, и, соответственно, нагрев. ЭВМ нельзя располагать вблизи источников тепла или термоизлучателей, на экраны дисплеев не должны падать прямые солнечные лучи. Устанавливать ЭВМ необходимо так, чтобы задняя и боковые стенки отстояли не менее чем на 0.2 м от других предметов. Для соблюдения теплового режима в корпусе ЭВМ должны быть предусмотрены вентиляционные отверстия и охлаждающий вентилятор. Внутренний монтаж должен быть выполнен проводом с повышенной теплостойкостью.

Пожарная безопасность объекта должна обеспечиваться:

1. системой предотвращения пожара;
2. системой противопожарной защиты;
3. организационно-техническими мероприятиями.

Предотвращение пожара в помещении может быть достигнуто минимальным количеством предметов из горючих материалов, их безопасным расположением, а также отсутствием легковоспламеняющихся материалов.

Противопожарная защита помещения может быть обеспечена применением автоматической установки пожарной сигнализации, наличием средств пожаротушения, применением основных строительных конструкций здания с регламентированными пределами огнестойкости.

Организационно-технические мероприятия должны включать организацию обучения служащих правилам пожарной безопасности.

**3.4.3 Рекомендации по обеспечению допустимого уровня шума и вибрации в помещениях**

Показатели уровня шумов в рабочих помещениях, где будет осуществляться эксплуатация программы должны соответствовать требованиям ГОСТ 12.1.003-83. Допустимый уровень шума при умственном труде, требующем сосредоточенности, 50дБ[[2]](#footnote-2)).

Для уменьшения шума и вибрации в помещении оборудование, аппараты и приборы должны устанавливаться на специальные фундаменты и амортизирующие прокладки. Если стены и потолки помещения являются источниками шумообразования, они должны быть облицованы звукопоглощающим материалом.

**Заключение**

Анализ существующего подхода к оперативной оценке поступающей информации выявил необходимость разработки и внедрения дополнительных ПС представления информации.

Анализ ПСПИ, разработанных в настоящее время выявил, функциональный, программно-технический, технологический и экономический факторы, сильно ограничивающие их применение для оценки информации.

На основе вышесказанного, определен альтернативный вариант реализации ПС представления информации – самостоятельная разработка с использованием ИСР Borland Delphi 7.0.

Исходя из практического применения различных вариантов реализации ПСПИ, выбран локальный вариант их разработки в виде программного компонента выполняющего функцию оперативного и достоверного показа информации в графическом виде.

В ходе работы были сформулированы функциональные, технические, информационные и эргономические требования к программным средствам. В соответствии с этими требованиями была разработана ПСПИ. Оценены эксплуатационно-технические характеристики, надежность и качество. Проведено экономическое обоснование внедрения ПС в специализированные учебные центры и воинские части.

Направлениями дальнейших исследований может быть повышение диапазона отображения и обработка более сложных матричных форм.

**Список использованной литературы**

1. Иваненко А. Ю. Оформление документации на программные средства. – М.: Издательство МГТУ им. Баумана, 2002;
2. Фаронов В. В. Delphi Программирование на языке высокого уровня. – СПБ: «Питер», 2003;
3. ГОСТ 19.101-77 ЕСПД «Виды программ и программных документов».
4. ГОСТ 19.402-78 ЕСПД «Описание программы».
5. ГОСТ 19.701-90 ЕСПД. «Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения в соответствии с ним».
6. ГОСТ 24.211-82 «Требования к содержанию документа «Описание алгоритма»».
7. Мамиконов А. Г. Модели и методы проектирования информационного обеспечения АСУ. – М.: Статистика, 1978.
8. ГОСТ ЕСКД, ГОСТ ЕСПД, ГОСТ КС и РД АСУ, ГОСТ СТД АСУ.
9. Иваненко А. Ю. Оформление документации на программные средства. – М.: Издательство МГТУ им. Баумана, 2002.
10. Фленов М. Е., DirectX и Delphi. Искусство программирования. – СПб.: БХВ-Петербург, 2006.
11. Краснов М. В., DirectX. Графика в проектах Delphi. – СПб.: БХВ-Петербург, 2005.
12. Архангельский А. Я. Программирование на Borland Delphi 7.0. – М: «Бином», 2004.
13. Актуальные задачи развития Вооруженных сил Российской Федерации // издание «Красная звезда» от 11.10.2003.
14. Аленичева Е. В., Монастырев П. В. Электронный учебник (проблемы создания и оценки качества) // Высшее образование в России, №1, 2001.
15. ISO 9126:1991 Информационная технология. Оценка программного продукта. Характеристики качества и руководство по их применению.
16. ГОСТ Р ИСО/МЭК 9126-93 “Информационные технологии. Оценка программной продукции. Характеристики качества и руководства по их применению”
17. ГОСТ 19.101-77 ЕСПД «Виды программ и программных документов»
18. ГОСТ 19.201-78 ЕСПД. Техническое задание. Требования к содержанию и оформлению.
19. ГОСТ 19.401-78 ЕСПД. «Текст программы. Требования к содержанию и оформлению».
20. ГОСТ 19.402-78 ЕСПД «Описание программы»
21. ГОСТ 19.701-90 ЕСПД. «Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения в соответствии с ним».
22. ГОСТ 24.211-82 «Требования к содержанию документа «Описание алгоритма»»
23. ГОСТ-28195-95 “Оценка качества программных средств. Общие положения”.
24. ГОСТ 12.1.004-85. ССБТ. Пожарная безопасность.

**Приложение А**

Алгоритм программы



**Приложение Б**

Текст программы

unit Unit1;

interface

uses

Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,

ExtCtrls, Menus, OpenGL, Buttons, ExtDlgs, ComCtrls, StdCtrls, Dialogs;

type

TRenderPanel = record

DC:HDC;

HRC:HGLRC;

ps:TPaintStruct;

end;

TMatrix = record

w:Integer; //размерность матрицы

vx:Array of Array of Extended;//массив вершин

nx:Array of Array of Array[1..3] of Extended;//массив нормалей

cx:Array of Array of Array[1..3] of GLfloat;//массив цветов

cc:Array of Array of Array[1..3] of GLfloat;//массив цветов

end;

PMatrix = ^TMatrix;

TMat = class(TForm)

Panel1: TPanel;

OpenPictureDialog1: TOpenPictureDialog;

Panel4: TPanel;

GroupBox1: TGroupBox;

Label1: TLabel;

Label2: TLabel;

Edit1: TEdit;

Edit2: TEdit;

UpDown1: TUpDown;

UpDown2: TUpDown;

Label3: TLabel;

Edit3: TEdit;

GroupBox2: TGroupBox;

Button2: TButton;

FoDialog: TOpenDialog;

Addons: TGroupBox;

SpeedButton1: TSpeedButton;

Button1: TButton;

Button3: TButton;

LBData: TListBox;

ListBox1: TListBox;

cb\_Surface: TCheckBox;

progress: TPanel;

Label4: TLabel;

bar: TProgressBar;

ComboBoxMatrix: TComboBox;

BitBtnSave: TBitBtn;

SaveDialogMain: TSaveDialog;

procedure CalcNormals(x1,y1,z1,x2,y2,z2,x3,y3,z3:Extended; var nx,ny,nz:Extended);

procedure GL(var Matrix:TMatrix); //прорисовка матрицы на экран

procedure Init();

procedure SelPos(var Matrix:TMatrix; xx:Integer;yy:Integer);

procedure FormCreate(Sender: TObject);

procedure FormDestroy(Sender: TObject);

//---------------------------------

function LoadMatrixFromBitmap(filename:string; var Matrix:TMatrix):boolean;

function LoadMatrixFromDtFile(filename:string; var Matrix:TMatrix):boolean;

function MakeAnalysMatrixData(Matrix01,Matrix02:TMatrix; var Matrix03:TMatrix):boolean;

//---------------------------------

procedure bmp1Click(Sender: TObject);

procedure Panel4MouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure Panel4MouseMove(Sender: TObject; Shift: TShiftState; X,

Y: Integer);

procedure Panel4MouseUp(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure Edit1KeyPress(Sender: TObject; var Key: Char);

procedure Edit1Change(Sender: TObject);

procedure Edit2KeyPress(Sender: TObject; var Key: Char);

procedure Edit2Change(Sender: TObject);

procedure Button1Click(Sender: TObject);

procedure Button2Click(Sender: TObject);

procedure Button3Click(Sender: TObject);

procedure Button4Click(Sender: TObject);

procedure cb\_SurfaceClick(Sender: TObject);

procedure ComboBoxMatrixChange(Sender: TObject);

procedure Edit3Change(Sender: TObject);

procedure BitBtnSaveClick(Sender: TObject);

private

MyPanels: TRenderPanel;

MCurrent: PMatrix;

myMatrix01:TMatrix; //первое изображение

myMatrix02:TMatrix; //второе изображение

myMatrix03:TMatrix; //наложение первого на второе

// w:Integer; //размерность матрицы

// vx:Array of Array of Extended;//массив вершин

// nx:Array of Array of Array[1..3] of Extended;//массив нормалей

// cx:Array of Array of Array[1..3] of GLfloat;//массив цветов

// cc:Array of Array of Array[1..3] of GLfloat;//массив цветов

mess:string;

bmp:TBitmap;

procedure InitOpenGL;

procedure SetDCPixelFormat(DC:HDC);

end;

var

yess: Boolean;

Mat: TMat;

MouseButton : Integer;

Xcoord, Ycoord, Zcoord : Integer;

YRot, XRot : integer;

Depth : integer;

LastCCX,LastCCY : Integer;

//-----------------------------------------------------------------------------

implementation

//-----------------------------------------------------------------------------

uses UAbout;

{$R \*.dfm}

procedure TMat.InitOpenGL;

begin

MyPanels.DC:=GetDC(Panel4.Handle);

SetDCPixelFormat(MyPanels.DC);

MyPanels.HRC:=wglCreateContext(MyPanels.DC);

wglMakeCurrent(MyPanels.DC,MyPanels.HRC);

glEnable(GL\_DEPTH\_TEST);

glClearColor(0,0,0,1);

end;

procedure TMat.SetDCPixelFormat(DC:HDC);

var

pfd:TPixelFormatDescriptor;

nPixelFormat:Integer;

begin

FillChar(pfd,SizeOf(pfd),0);

pfd.dwFlags:=PFD\_DOUBLEBUFFER or

PFD\_DRAW\_TO\_WINDOW or

PFD\_SUPPORT\_OPENGL;

nPixelFormat:=ChoosePixelFormat(DC,@pfd);

SetPixelFormat(DC,nPixelFormat,@pfd);

end;

procedure TMat.FormCreate(Sender: TObject);

begin

MouseButton :=0;

bmp:=TBitmap.Create;

InitOpenGL;

Left:=0;

Top:=0;

LastCCX:=0;

LastCCY:=0;

self.MCurrent := @self.myMatrix01;

self.ComboBoxMatrix.ItemIndex := 0;

// Width:=Screen.Width;

// Height:=Screen.Height;

// WindowState:=wsMaximized;

// Timer1.Enabled:=True;

end;

procedure TMat.FormDestroy(Sender: TObject);

begin

wglMakeCurrent(0,0);

bmp.Destroy;

Finalize(myMatrix01.vx);

Finalize(myMatrix01.cx);

Finalize(myMatrix01.cc);

Finalize(myMatrix01.nx);

Finalize(myMatrix02.vx);

Finalize(myMatrix02.cx);

Finalize(myMatrix02.cc);

Finalize(myMatrix02.nx);

Finalize(myMatrix03.vx);

Finalize(myMatrix03.cx);

Finalize(myMatrix03.cc);

Finalize(myMatrix03.nx);

wglDeleteContext(MyPanels.HRC);

ReleaseDC(MyPanels.DC,Panel4.Handle);

DeleteDC(MyPanels.DC);

end;

procedure TMat.GL(var Matrix:TMatrix);

var

j,k,dw,dv : Integer;

ps : TPaintStruct;

av : Integer;

stroka:string;

begin

av:=0;

dw:=0;

dv:=0;

if (Yess=true) then

begin

GroupBox1.Enabled:=True;

dw:=Matrix.w div 2;

dv:=25 div 2;

BeginPaint(Panel4.Handle,ps);

wglMakeCurrent(MyPanels.DC,MyPanels.HRC);

glViewport(0,0,Panel4.Width,Panel4.Height);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity;

gluPerspective(30,Panel4.Width/Panel4.Height,1,10000);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity;

glClear(GL\_COLOR\_BUFFER\_BIT or GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity;

glTranslatef(0,0,-Zcoord);

glRotatef(Xrot,1,0,0);

glRotatef(Yrot,0,1,0);

try

if (Matrix.w>5) then

If cb\_Surface.Checked then av:=2 else av:=1;

case av of

1 : begin

glDisable(GL\_LIGHTING);

glDisable(GL\_LIGHT0);

glColor3f(1,1,1);

glBegin(GL\_LINES);

for j:=0 to Matrix.w-2 do

for k:=0 to Matrix.w-2 do

begin

glColor3f(Matrix.cx[j,k,1],Matrix.cx[j,k,2],Matrix.cx[j,k,3]);

glVertex3f(j-dw,Matrix.vx[j,k]-dv,k-dw);

glColor3f(Matrix.cx[j,k+1,1],Matrix.cx[j,k+1,2],Matrix.cx[j,k+1,3]);

glVertex3f(j-dw,Matrix.vx[j,k+1]-dv,k+1-dw);

end;

for j:=0 to Matrix.w-2 do

for k:=0 to Matrix.w-2 do

begin

glColor3f(Matrix.cx[j,k,1],Matrix.cx[j,k,2],Matrix.cx[j,k,3]);

glVertex3f(j-dw,Matrix.vx[j,k]-dv,k-dw);

glColor3f(Matrix.cx[j+1,k,1],Matrix.cx[j+1,k,2],Matrix.cx[j+1,k,3]);

glVertex3f(j+1-dw,Matrix.vx[j+1,k]-dv,k-dw);

end;

glEnd;

end;

2 : begin

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(1,1,1);

glBegin(GL\_TRIANGLES);

for j:=0 to Matrix.w-2 do

for k:=0 to Matrix.w-2 do

begin

glColor3f(Matrix.cx[j,k,1],Matrix.cx[j,k,2],Matrix.cx[j,k,3]);

glNormal3f(Matrix.nx[j,k,1],Matrix.nx[j,k,2],Matrix.nx[j,k,3]);

glVertex3f(j-dw,Matrix.vx[j,k]-dv,k-dw);

glColor3f(Matrix.cx[j+1,k,1],Matrix.cx[j+1,k,2],Matrix.cx[j+1,k,3]);

glNormal3f(Matrix.nx[j+1,k,1],Matrix.nx[j+1,k,2],Matrix.nx[j+1,k,3]);

glVertex3f(j-dw+1,Matrix.vx[j+1,k]-dv,k-dw);

glColor3f(Matrix.cx[j+1,k+1,1],Matrix.cx[j+1,k+1,2],Matrix.cx[j+1,k+1,3]);

glNormal3f(Matrix.nx[j+1,k+1,1],Matrix.nx[j+1,k+1,2],Matrix.nx[j+1,k+1,3]);

glVertex3f(j-dw+1,Matrix.vx[j+1,k+1]-dv,k-dw+1);

end;

for j:=0 to Matrix.w-2 do

for k:=0 to Matrix.w-2 do

begin

glColor3f(Matrix.cx[j,k,1],Matrix.cx[j,k,2],Matrix.cx[j,k,3]);

glNormal3f(Matrix.nx[j,k,1],Matrix.nx[j,k,2],Matrix.nx[j,k,3]);

glVertex3f(j-dw,Matrix.vx[j,k]-dv,k-dw);

glColor3f(Matrix.cx[j,k+1,1],Matrix.cx[j,k+1,2],Matrix.cx[j,k+1,3]);

glNormal3f(Matrix.nx[j,k+1,1],Matrix.nx[j,k+1,2],Matrix.nx[j,k+1,3]);

glVertex3f(j-dw,Matrix.vx[j,k+1]-dv,k-dw+1);

glColor3f(Matrix.cx[j+1,k+1,1],Matrix.cx[j+1,k+1,2],Matrix.cx[j+1,k+1,3]);

glNormal3f(Matrix.nx[j+1,k+1,1],Matrix.nx[j+1,k+1,2],Matrix.nx[j+1,k+1,3]);

glVertex3f(j-dw+1,Matrix.vx[j+1,k+1]-dv,k-dw+1);

end;

glEnd;

glDisable(GL\_COLOR\_MATERIAL);

end;

end;

except

Matrix.w:=0;

MessageBox(Handle,'Ошибка при прорисовке изображения',

'Ошибка',MB\_OK or MB\_ICONERROR);

end;

EndPaint(Panel4.Handle,ps);

glRotatef(120,1.0,0.0,0.0); // Rotate on x

glRotatef(120,0.0,1.0,0.0); // Rotate on y

glRotatef(120,0.0,0.0,1.0); // Rotate on z

SwapBuffers(MyPanels.DC);

//временная защита кода

// stroka := 'Это демонстрационная версия!!!';

// TextOut(myPanels.DC,300,200,PChar(stroka),Length(stroka));

// stroka := 'Программа сделана на заказ!!!';

// TextOut(myPanels.DC,300,220,PChar(stroka),Length(stroka));

// stroka := 'лоалофв аофоа длфыв а лдо';

// TextOut(myPanels.DC,300,240,PChar(stroka),Length(stroka));

end;

end;

procedure TMat.SelPos(var Matrix:TMatrix; xx:Integer;yy:Integer);

var

fx:Integer;

s:string;

begin

if (Matrix.w>0) then

begin

for fx := 0 to Matrix.w-1 do

begin

Matrix.cx[LastCCx,fx,1]:=Matrix.cc[LastCCx,fx,1];

Matrix.cx[LastCCx,fx,2]:=Matrix.cc[LastCCx,fx,2];

Matrix.cx[LastCCx,fx,3]:=Matrix.cc[LastCCx,fx,3];

Matrix.cx[fx,LastCCy,1]:=Matrix.cc[fx,LastCCy,1];

Matrix.cx[fx,LastCCy,2]:=Matrix.cc[fx,LastCCy,2];

Matrix.cx[fx,LastCCy,3]:=Matrix.cc[fx,LastCCy,3];

Matrix.cx[xx,fx,1]:=1;

Matrix.cx[xx,fx,2]:=1;

Matrix.cx[xx,fx,3]:=1;

Matrix.cx[fx,yy,1]:=1;

Matrix.cx[fx,yy,2]:=1;

Matrix.cx[fx,yy,3]:=1;

end;

LastCCx:=xx;

LastCCy:=yy;

s:=FormatFloat('0.00', Matrix.vx[LastCCx,LastCCy]);

// if Matrix.vx[LastCCx,LastCCy] < 0 then s:= '-'+s;

// Edit3.Text:=FloatToStr(Round(Matrix.vx[LastCCx,LastCCy]\*100)/100);

Edit3.Text := s;

end; трехмерный изображение матричный графический

end;

//-----------------------------------------------------------------------------

function TMat.LoadMatrixFromBitmap(filename:string; var Matrix:TMatrix):boolean;

var

i,j :Integer;

ss : string;

begin

Result := false;

if not FileExists(filename) then exit;

with Matrix do

begin

bmp.Width:=0;

bmp.Height:=0;

bmp.LoadFromFile(OpenPictureDialog1.FileName);

w:=bmp.Width;

UpDown1.Max:=w;

UpDown2.Max:=w;

LastCCX:=w div 2;

LastCCY:=w div 2;

SetLength(vx,w);

SetLength(nx,w);

SetLength(cx,w);

SetLength(cc,w);

for i:=0 to w-1 do

begin

SetLength(vx[i],w);

SetLength(nx[i],w);

SetLength(cx[i],w);

SetLength(cc[i],w);

end;

ss:='';

ListBox1.Items.Clear;

for i:=0 to w-1 do

begin

for j:=0 to w-1 do

begin

vx[i,j]:=(GetRValue(bmp.Canvas.Pixels[i,j])+

GetGValue(bmp.Canvas.Pixels[i,j])+

GetBValue(bmp.Canvas.Pixels[i,j]))/50;

if vx[i,j]>10 then vx[i,j]:=9+(random(99)+1)/100;

ss:=ss+FormatFloat('0.00', vx[i,j])+' ';

cx[i,j,1]:=GetRValue(bmp.Canvas.Pixels[i,j])/255;

cx[i,j,2]:=GetGValue(bmp.Canvas.Pixels[i,j])/255;

cx[i,j,3]:=GetBValue(bmp.Canvas.Pixels[i,j])/255;

cc[i,j,1]:=GetRValue(bmp.Canvas.Pixels[i,j])/255;

cc[i,j,2]:=GetGValue(bmp.Canvas.Pixels[i,j])/255;

cc[i,j,3]:=GetBValue(bmp.Canvas.Pixels[i,j])/255;

end;

ListBox1.Items.Add(ss);

ss:='';

end;

Zcoord :=w\*2;

SelPos(Matrix, LastCCX, LastCCY);

UpDown1.Position:=LastCCX;

UpDown2.Position:=LastCCY;

end;

Result := true;

end;

function TMat.LoadMatrixFromDtFile(filename:string; var Matrix:TMatrix):boolean;

var

i,x,y,j,k,posp,posbar:Integer;

spr,sfl,ss,formfl:String;

Fres : TFloatRec;

Conv : Extended ;

coint :integer;

ValStr :Extended;

begin

Result := false;

if not FileExists(filename) then exit;

with Matrix do

begin

LBData.Items.Clear;

bar.Position:=0;

progress.Visible:=True;

progress.Update;

LBData.Items.LoadFromFile(FileName);

if LBData.Items.Count>5 then

begin

bar.Position:=5;

bar.Update;

w:=LBData.Items.Count;

UpDown1.Max:=w;

UpDown2.Max:=w;

LastCCX:=w div 2;

LastCCY:=w div 2;

SetLength(vx,w);

SetLength(nx,w);

SetLength(cx,w);

SetLength(cc,w);

for i:=0 to w-1 do

begin

SetLength(vx[i],w);

SetLength(nx[i],w);

SetLength(cx[i],w);

SetLength(cc[i],w);

for y :=0 to w-1 do

begin

vx[i,y]:=0;

nx[i,y,1]:=0;

nx[i,y,2]:=0;

nx[i,y,3]:=0;

cx[i,y,1]:=0;

cx[i,y,2]:=0;

cx[i,y,3]:=0;

cc[i,y,1]:=0;

cc[i,y,2]:=0;

cc[i,y,3]:=0;

end;

end;

yess:=True;

mess:='';

for y :=0 to w-1 do

begin

spr:=LBData.Items[y];

x:=0;

while (((pos(' ',spr)>0) or (Length(spr)>0)) and (Yess=True) and (x<w)) do

begin

posp:=pos(' ',spr);

If (posp>0) then

begin

sfl:=trim (copy(spr,0,posp));

delete(spr,1,posp);

ValStr:=strtofloatdef(sfl,-100);

If (ValStr=-100) then

begin

yess:=False;

if (Length(mess)=0) then mess:='Неверное значение'+#13#10+'строка '+ IntToStr(y+1)+#13#10+'позиция '+IntToStr(x+1)+#13#10+'['+sfl+']';

break;

end;

If ((ValStr<-10) or (ValStr>10)) then

begin

yess:=False;

if (Length(mess)=0) then mess:='Значение >10, либо <-10'+#13#10+'строка '+ IntToStr(y+1)+#13#10+'позиция '+IntToStr(x+1)+#13#10+'['+sfl+']';

break;

end else vx[x,y]:=ValStr;

end else

begin

spr:=Trim(spr);

ValStr:=strtofloatdef(spr,-100);

If (ValStr=-100) then

begin

yess:=False;

if (Length(mess)=0) then mess:='Неверное значение'+#13#10+'строка '+ IntToStr(y+1)+#13#10+'позиция '+IntToStr(x+1)+#13#10+'['+spr+']';

break;

end;

If ((ValStr<-10) or (ValStr>10)) then

begin

yess:=False;

if (Length(mess)=0) then mess:='Значение >10, либо <-10'+#13#10+'строка '+ IntToStr(y+1)+#13#10+'позиция '+IntToStr(x+1)+#13#10+'['+spr+']';

break;

end else vx[x,y]:=ValStr;

spr:='';

end;

inc(x);

end;

formfl := FormatFloat('0',70\*(((y+1)\*(x))/(w\*w)));

coint:=StrToInt(formfl);

bar.Position:=5+coint;

bar.Update;

// mat.Caption :=mat.Caption+inttostr(x)+' ';

if (x<w) then

begin

Yess:=false;

if (Length(mess)=0) then mess:='строка '+ IntToStr(y+1)+#13#10+'короткая, либо излишек строк в файле';

break;

end;

if (spr<>'') then

begin

Yess:=false;

if (Length(mess)=0) then mess:='строка '+ IntToStr(y+1)+#13#10+'длинная, либо недостаточно строк в файле';

break;

end;

end;

end else

begin

Yess:=false;

mess:='Форма должна иметь'+#13#10+'размер более чем 5х5';

end;

if Yess=true then

begin

bar.Position:=90;

bar.Update;

for i:=0 to w-1 do

begin

for j:=0 to w-1 do

begin

cx[i,j,1]:=(vx[i,j]+1)/9;

cx[i,j,2]:=1-vx[i,j+1]/9;

cx[i,j,3]:=0;

cc[i,j,1]:=(vx[i,j]+1)/9;

cc[i,j,2]:=1-vx[i,j+1]/9;

cc[i,j,3]:=0;

end;

end;

for i:=0 to w-1 do

for j:=0 to w-1 do

for k:=1 to 3 do

nx[i,j,k]:=1;

for i:=0 to w-2 do

for j:=0 to w-2 do

begin

CalcNormals(i,vx[i,j],j,

i+1,vx[i+1,j],j,

i+1,vx[i+1,j+1],j+1,

nx[i,j,1],nx[i,j,2],nx[i,j,3]);

end;

bar.Position:=100;

bar.Update;

Zcoord :=w\*2;

XRot:=90;

YRot:=0;

UpDown1.Position:=LastCCX;

UpDown2.Position:=LastCCY;

SelPos(Matrix,LastCCX, LastCCY);

progress.Hide;

Panel4.Show;

end;

end;

Result := Yess;

end;

//-----------------------------------------------------------------------------

procedure TMat.bmp1Click(Sender: TObject);

begin

try

if OpenPictureDialog1.Execute then

if FileExists(OpenPictureDialog1.FileName) then

begin

self.LoadMatrixFromBitmap(OpenPictureDialog1.FileName,self.MCurrent^);

self.GL(self.MCurrent^);

end else

MessageBox(Handle,

PAnsiChar('Файл '+OpenPictureDialog1.FileName+' не найден'),

'Ошибка',MB\_OK or MB\_ICONERROR);

except

MessageBox(Handle,

PAnsiChar('Ошибка во время загрузки файла '+

OpenPictureDialog1.FileName),

'Ошибка',MB\_OK or MB\_ICONERROR);

end;

end;

procedure TMat.Panel4MouseDown(Sender: TObject;

Button: TMouseButton; Shift: TShiftState; X, Y: Integer);

begin

if Button = mbLeft then

begin

MouseButton :=1;

Xcoord := X;

Ycoord := Y;

end;

if Button = mbRight then

begin

MouseButton :=2;

Zcoord := Y;

end;

end;

procedure TMat.Panel4MouseMove(Sender: TObject; Shift: TShiftState;

X, Y: Integer);

begin

if MouseButton = 1 then

begin

xRot := xRot + (Y - Ycoord) div 2; // moving up and down = rot around X-axis

yRot := yRot + (X - Xcoord)div 2;

Xcoord := X;

Ycoord := Y;

GL(self.MCurrent^);

end;

if MouseButton = 2 then

begin

Depth :=Depth - (Y-ZCoord) div 3;

Zcoord := Y;

GL(self.MCurrent^);

end;

// caption:=inttostr(xRot)+':'+inttostr(yRot);

end;

procedure TMat.Panel4MouseUp(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

begin

MouseButton :=0;

end;

procedure TMat.Edit1KeyPress(Sender: TObject; var Key: Char);

begin

if Ord(Key)<>8 then if ((key<'0') or (key>'9')) then Key:=#0;

end;

procedure TMat.Edit1Change(Sender: TObject);

var

x:Integer;

begin

If TryStrToInt(Edit1.Text,x)

then begin

if x>self.MCurrent^.w then Edit1.Text:=IntToStr(self.MCurrent^.w);

If x<2 then Edit1.Text:='1';

end

else begin

Edit1.Text:='1';

end;

SelPos(self.MCurrent^,UpDown1.Position-1,LastCCY);

GL(self.MCurrent^);

end;

procedure TMat.Edit2KeyPress(Sender: TObject; var Key: Char);

begin

if Ord(Key)<>8 then if ((key<'0') or (key>'9')) then Key:=#0;

end;

procedure TMat.Edit2Change(Sender: TObject);

var

x:Integer;

begin

If TryStrToInt(Edit2.Text,x)

then begin

if x>self.MCurrent^.w then Edit2.Text:=IntToStr(self.MCurrent^.w);

If x<2 then Edit2.Text:='1';

end

else begin

Edit2.Text:='1';

end;

SelPos(self.MCurrent^,LastCCX,UpDown2.Position-1);

GL(self.MCurrent^);

end;

procedure TMat.CalcNormals(x1,y1,z1,x2,y2,z2,x3,y3,z3:Extended; var nx,ny,nz:Extended);

var

wrki: Double;

vx1,vy1,vz1,vx2,vy2,vz2: Double;

begin

vx1:=x1-x2;

vy1:=y1-y2;

vz1:=z1-z2;

vx2:=x2-x3;

vy2:=y2-y3;

vz2:=z2-z3;

wrki:=sqrt(sqr(vy1\*vz2-vz1\*vy2)+sqr(vz1\*vx2-vx1\*vz2)+sqr(vx1\*vy2-vy1\*vx2));

nx:=-(vy1 \* vz2 - vz1 \* vy2)/wrki;

ny:=-(vz1 \* vx2 - vx1 \* vz2)/wrki;

nz:=-(vx1 \* vy2 - vy1 \* vx2)/wrki;

end;

procedure TMat.Button1Click(Sender: TObject);

begin

ListBox1.Items.SaveToFile(ChangeFileExt(Application.ExeName,'.txt'));

end;

procedure TMat.Init();

begin

Edit1.OnChange :=Edit1Change;

Edit1.OnKeyPress :=Edit1KeyPress;

Edit2.OnChange :=Edit2Change;

Edit2.OnKeyPress :=Edit2KeyPress;

Panel4.OnMouseDown :=Panel4MouseDown;

Panel4.OnMouseMove :=Panel4MouseMove;

Panel4.OnMouseUp :=Panel4MouseUp;

end;

procedure TMat.Button3Click(Sender: TObject);

begin

About.ShowModal;

end;

procedure TMat.Button4Click(Sender: TObject);

begin

Edit1.OnChange :=nil;

Edit1.OnKeyPress :=nil;

Edit2.OnChange :=nil;

Edit2.OnKeyPress :=nil;

Panel4.OnMouseDown :=nil;

Panel4.OnMouseMove :=nil;

Panel4.OnMouseUp :=nil;

Close;

end;

//проводим анализ данных, точки совпадения красным, ниже синим, выше зеленым

function TMat.MakeAnalysMatrixData(Matrix01,Matrix02:TMatrix; var Matrix03:TMatrix):boolean;

var

i,j,k,y:integer;

begin

Result := false;

//инициализация результ. матрицы

Matrix03.w := Matrix01.w;

with Matrix03 do

begin

SetLength(vx,w);

SetLength(nx,w);

SetLength(cx,w);

SetLength(cc,w);

for i:=0 to w-1 do

begin

SetLength(vx[i],w);

SetLength(nx[i],w);

SetLength(cx[i],w);

SetLength(cc[i],w);

for y :=0 to w-1 do

begin

vx[i,y]:=Matrix01.vx[i,y];

nx[i,y,1]:=Matrix01.nx[i,y,1];

nx[i,y,2]:=Matrix01.nx[i,y,2];

nx[i,y,3]:=Matrix01.nx[i,y,3];

cx[i,y,1]:=Matrix01.cx[i,y,1];

cx[i,y,2]:=Matrix01.cx[i,y,2];

cx[i,y,3]:=Matrix01.cx[i,y,3];

cc[i,y,1]:=Matrix01.cc[i,y,1];

cc[i,y,2]:=Matrix01.cc[i,y,2];

cc[i,y,3]:=Matrix01.cc[i,y,3];

cx[i,y,1]:=255;

cx[i,y,2]:=255;

cx[i,y,3]:=255;

//часть первого, которая не пересеклась со вторым

//окрашиваем в желтый цвет

if Matrix02.vx[i,y] = 0 then

begin

cx[i,y,1]:=(vx[i,y]+1)/6;

cx[i,y,2]:=(vx[i,y]+1)/6;

cx[i,y,3]:=0;

end;

//часть второго, которая не пересеклась с первой

//окрашиваем в красный цвет

if Matrix01.vx[i,y] = 0 then

begin

vx[i,y]:=Matrix02.vx[i,y];

cx[i,y,1]:=(vx[i,y]+1)/6;

cx[i,y,2]:=0;

cx[i,y,3]:=0;

end;

//если нет поверхностей => зеленый

if (Matrix01.vx[i,y] = 0)

and (Matrix02.vx[i,y] = 0)then

begin

cx[i,y,1]:=0;

cx[i,y,2]:=(vx[i,y]+1)/2;

cx[i,y,3]:=0;

end;

//совпадающие обозначае зеленым цветом

if (Matrix01.vx[i,y] = Matrix02.vx[i,y])

and (Matrix01.vx[i,y] <> 0)

and (Matrix02.vx[i,y] <> 0)then

begin

cx[i,y,1]:=0;

cx[i,y,2]:=(vx[i,y]+1)/2;

cx[i,y,3]:=0;

end;

//те, которые выше - делаем зеленым

if (Matrix01.vx[i,y] < Matrix02.vx[i,y])

and (Matrix01.vx[i,y] <> 0)

and (Matrix02.vx[i,y] <> 0)then

begin

vx[i,y]:=Matrix02.vx[i,y];

cx[i,y,1]:=0;

cx[i,y,2]:=(vx[i,y]+1)/2;;

cx[i,y,3]:=0;

end;

//те, которые ниже будут синим

if (Matrix01.vx[i,y] > Matrix02.vx[i,y])

and (Matrix01.vx[i,y] <> 0)

and (Matrix02.vx[i,y] <> 0)then

begin

cx[i,y,1]:=(vx[i,y]+1)/6;

cx[i,y,2]:=0;

cx[i,y,3]:=0;

end;

cc[i,y,1]:=cx[i,y,1];

cc[i,y,2]:=cx[i,y,2];

cc[i,y,3]:=cx[i,y,3];

end;

end;

end;

{

w:Integer; //размерность матрицы

vx:Array of Array of Extended;//массив вершин

nx:Array of Array of Array[1..3] of Extended;//массив нормалей

cx:Array of Array of Array[1..3] of GLfloat;//массив цветов

cc:Array of Array of Array[1..3] of GLfloat;//массив цветов

}

Result := true;

end;

procedure TMat.cb\_SurfaceClick(Sender: TObject);

begin

GL(self.MCurrent^);

end;

procedure TMat.Button2Click(Sender: TObject);

begin

//возможно, режим анализа поверхностей

if self.ComboBoxMatrix.ItemIndex = 2 then

begin

if not self.MakeAnalysMatrixData(self.myMatrix01, self.myMatrix02, self.myMatrix03) then

begin

ShowMessage('Не удалось провести анализ поверхностей!');

end;

self.GL(self.MCurrent^);

exit;

end;

Panel4.Hide;

FoDialog.InitialDir:=ExtractFilePath(Application.ExeName);

If FoDialog.Execute then

begin

if self.LoadMatrixFromDtFile(FoDialog.FileName,self.MCurrent^) then

begin

self.GL(self.MCurrent^);

end else //Yess=false

begin

progress.Hide;

MessageBox(Handle,PAnsiChar('Ошибка в файле данных!'+#13#10+self.mess),PAnsiChar('Ошибка'),MB\_OK or MB\_ICONINFORMATION);

Panel4.Hide;

// w:=0;

end;

end;

end;

procedure TMat.ComboBoxMatrixChange(Sender: TObject);

begin

if self.ComboBoxMatrix.ItemIndex = 0 then self.MCurrent := @self.myMatrix01;

if self.ComboBoxMatrix.ItemIndex = 1 then self.MCurrent := @self.myMatrix02;

if self.ComboBoxMatrix.ItemIndex = 2 then self.MCurrent := @self.myMatrix03;

self.Button2.Caption := 'Загрузить';

if self.ComboBoxMatrix.ItemIndex = 2 then self.Button2.Caption := 'Провести анализ';

self.GL(self.MCurrent^);

exit;

end;

procedure TMat.Edit3Change(Sender: TObject);

var

pos\_x:integer;

pos\_y:integer;

value:real;

begin

//изменение значения вершины

pos\_x := self.UpDown1.Position-1;

pos\_y := self.UpDown2.Position-1;

value := StrToFloatDef(self.Edit3.Text,-1000);

if value > -1000 then

self.MCurrent^.vx[pos\_x,pos\_y] := value;

// else

// self.Edit3.Text := FloatToStr(self.MCurrent^.vx[pos\_x,pos\_y]);

//теперь просчитываем цвета

With self.MCurrent^ do

begin

cx[pos\_x,pos\_y,1]:=(vx[pos\_x,pos\_y]+1)/9;

cx[pos\_x,pos\_y,2]:=1-vx[pos\_x,pos\_y+1]/9;

cx[pos\_x,pos\_y,3]:=0;

cc[pos\_x,pos\_y,1]:=(vx[pos\_x,pos\_y]+1)/9;

cc[pos\_x,pos\_y,2]:=1-vx[pos\_x,pos\_y+1]/9;

cc[pos\_x,pos\_y,3]:=0;

end;

//после изменений перерисовываем

self.GL(self.MCurrent^);

exit;

end;

procedure TMat.BitBtnSaveClick(Sender: TObject);

var

Spisok:TStringList;

stroka:string;

k,y:integer;

begin

//button "save" click

if self.MCurrent^.w = 0 then

begin

ShowMessage('Матрица не загружена!');

exit;

end;

if self.SaveDialogMain.FileName = '' then

self.SaveDialogMain.InitialDir := ExtractFileDir(ParamStr(0));

if not self.SaveDialogMain.Execute() then exit;

//---------------------------------------------

Spisok := TStringList.Create();

with self.MCurrent^ do

begin

for y:= 0 to w-1 do

begin

stroka := '';

for k:= 0 to w-1 do

begin

stroka := stroka + ' ' + FloatToStr(vx[k,y]);

continue;

end;

stroka := trim(stroka);

Spisok.Add(stroka);

end;

end;

Spisok.SaveToFile(self.SaveDialogMain.FileName);

Spisok.Free();

//---------------------------------------------

ShowMessage('Матрица была сохранена.');

exit;

end;

end.

);

var

Spisok:TStringList;

stroka:string;

k,y:integer;

begin

//button "save" click

if self.MCurrent^.w = 0 then

begin

ShowMessage('Матрица не загружена!');

exit;

end;

if self.SaveDialogMain.FileName = '' then

self.SaveDialogMain.InitialDir := ExtractFileDir(ParamStr(0));

if not self.SaveDialogMain.Execute() then exit;

//---------------------------------------------

Spisok := TStringList.Create();

with self.MCurrent^ do

begin

for y:= 0 to w-1 do

begin

stroka := '';

for k:= 0 to w-1 do

begin

stroka := stroka + ' ' + FloatToStr(vx[k,y]);

continue;

end;

stroka := trim(stroka);

Spisok.Add(stroka);

end;

end;

Spisok.SaveToFile(self.SaveDialogMain.FileName);

Spisok.Free();

//---------------------------------------------

ShowMessage('Матрица была сохранена.');

exit;

end;

end.

1. ) Расчет обобщенного показателя производится в соответствии с методикой оценки качества программного обеспечения, разработанной на кафедре оценки эффективности Военной академии воздушно-космической обороны. [↑](#footnote-ref-1)
2. ) в отдельных случаях эксплуатация программы допускается при превышении указанного значения [↑](#footnote-ref-2)