Государственное образовательное учреждение высшего профессионального образования

«Московский государственный технический университет имени Н.Э. Баумана»

(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ____Робототехника и комплексная автоматизация_____

КАФЕДРА ____Компьютерные системы автоматизации производства___

_____

Р А С Ч Ё Т Н О - П О Я С Н И Т Е Л Ь Н А Я З А П И С К А

к дипломному проекту на тему:

_____Разработка подсистемы анимации в РДО_____

_____

_____

Студент _____Циферов А.А._____ _____ _____
(Подпись, дата) (И.О.Фамилия)

Руководитель дипломного проекта _____ _____ _____
(Подпись, дата) (И.О.Фамилия)

Консультант по
научно-исследовательской части _____ _____ _____
(Подпись, дата) (И.О.Фамилия)

Консультант по конструкторской части____ _____ _____
(Подпись, дата) (И.О.Фамилия)

Консультант по технологической части____ _____ _____
(Подпись, дата) (И.О.Фамилия)

Консультант по
организационно-экономической части __ _____ _____
(Подпись, дата) (И.О.Фамилия)

Консультант по охране труда и экологии ___ _____ _____
(Подпись, дата) (И.О.Фамилия)

Москва, 2012

Государственное образовательное учреждение высшего профессионального образования

«Московский государственный технический университет имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ

Заведующий кафедрой _____
(Индекс)

_____ _____
(И.О.Фамилия)

« ___ » _____ 20 __ г.

З А Д А Н И Е
на выполнение дипломного проекта

Студент _____Циферов Алексей Александрович_____
(Фамилия, имя, отчество)

_____Разработка подсистемы анимации в РДО_____ __ _____
(Тема дипломного проекта)

_____

_____

Источник тематики (НИР кафедры, заказ организаций и т.п.)_НИР кафедры_____

_____

_____

Тема дипломного проекта утверждена распоряжением по факультету № _____
от « _12_ » __марта_____ 2012 г.

1. Исходные данные

_____

_____

_____

_____

_____

_____

2. Технико-экономическое обоснование

_____

_____

_____

_____

_____

(обзор и анализ альтернативных решений; выбор вариантов для сравнения;
конкретные улучшаемые характеристики или параметры; возможный технико-экономический эффект и т.п.)

3. Научно-исследовательская часть

_____

_____

_____

_____

_____

Консультант _____ _____
(Подпись, дата) (И.О.Фамилия)

4. Проектно-конструкторская часть

_____

_____

_____

_____

_____

_____

_____

_____

Консультант _____ _____

(Подпись, дата) (И.О.Фамилия)

5. Технологическая часть

_____

_____

_____

_____

_____

Консультант _____ _____

(Подпись, дата) (И.О.Фамилия)

6. Организационно-экономическая часть

_____

_____

_____

Консультант _____ _____

(Подпись, дата) (И.О.Фамилия)

7. Охрана труда и экология

_____

_____

_____

Консультант _____ _____

(Подпись, дата) (И.О.Фамилия)

8. Оформление дипломного проекта

8.1. Расчетно-пояснительная записка на 121 листах формата А4.

8.2. Перечень графического материала (плакаты, схемы, чертежи и т.п.) _____

_____

_____

_____

Дата выдачи задания « ___ » _____ 20__ г.

В соответствии с учебным планом дипломный проект выполнить в полном объеме в срок до « ___ » _____ 20__ г.

Руководитель дипломного проекта _____ _____

(Подпись, дата) (И.О.Фамилия)

Студент _____ _____

(Подпись, дата) (И.О.Фамилия)

Примечание:

1. Задание оформляется в двух экземплярах; один выдаётся студенту, второй хранится на кафедре.

Государственное образовательное учреждение высшего профессионального образования

«Московский государственный технический университет имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ

Заведующий кафедрой _____
(Индекс)

_____ _____
(И.О.Фамилия)

« ___ » _____ 20 __ г.

К А Л Е Н Д А Р Н Ы Й П Л А Н
выполнения дипломного проекта

Студент _____Циферов Алексей Александрович_____
(Фамилия, имя, отчество)

_____Разработка подсистемы анимации в РДО_____
(Тема дипломного проекта)

№ п/п	Наименование этапов дипломного проекта	Выполнение этапов		Примечание
		Срок	Объем, %	

Руководитель дипломного проекта _____ _____
(Подпись, дата) (И.О.Фамилия)

Студент _____ _____
(Подпись, дата) (И.О.Фамилия)

П р е д с е д а т е л ю
Государственной Аттестационной Комиссии №_____

факультета _____ МГТУ им. Н.Э. Баумана

Направляется студент _____ на защиту дипломного проекта
(фамилия, инициалы)

_____
(наименование темы)

_____
_____
_____
_____

С п р а в к а о б у с п е в а е м о с т и

Студент _____ за время пребывания в МГТУ имени Н.Э. Баумана с 20___г. полностью выполнил учебный план специальности со следующими оценками:
отлично – _____%, хорошо – _____%, удовлетворительно – _____%.

Секретарь факультета_____
(фамилия, инициалы)
Декан факультета_____
(фамилия, инициалы)

Заключение руководителя дипломного проекта

Студент_____
_____
_____
_____
_____
_____
_____
_____

Руководитель «_____»_____20___г.

Заключение кафедры о дипломном проекте

Дипломный проект просмотрен и студент _____ может быть допущен
(фамилия, инициалы)
к защите проекта в Государственной Аттестационной комиссии.

Заведующий кафедрой _____
«_____»_____20____г.

ОГЛАВЛЕНИЕ

РЕФЕРАТ

Отчет 121 с., 11 рис., 9 табл., 15 источников, 3 прил.

ИМИТИЦИОННОЕ МОДЕЛИРОВАНИЕ, РЕСУРС-ДЕЙСТВИЕ-ОПЕРАЦИЯ, АНИМАЦИЯ, МОДЕЛЬ, ГРАФИЧЕСКИЙ РЕДАКТОР, ИМИТАТОР, ПРОЦЕСС, ТРАНЗАКТ.

Объектом разработки является подсистема анимации графического редактора системы РДО. Ресурс, действие, операция (РДО) - программный комплекс, предназначенный для имитационного моделирования сложных дискретных систем с целью проведения их анализа и синтеза.

Цель работы - наглядное представление имитации модели существующей системы в процессе моделирования, созданной в графическом редакторе РДО.

При создании подсистемы были проведены исследования, целью которых являлось установление оптимального алгоритма записи графической информации об имитационной модели в файл для последующей ее загрузки.

В результате работы произведено внедрение подсистемы анимации для имитационных моделей, созданных с помощью графических примитивов, которая позволяет отслеживать ход их выполнения в течение моделирования, увеличивает наглядность моделей и сокращает время на обучение системе новых пользователей.

Эффективность применения анимации при имитационном моделировании заключается в повышении адекватности моделей в среде РДО.

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ, СИМВОЛОВ И СПЕЦИАЛЬНЫХ ТЕРМИНОВ С ИХ ОПРЕДЕЛЕНИЕМ

ARIS	Architecture of Integrated Information Systems (методология и программный продукт для моделирования бизнес-процессов компании)
UML	Universal Modeling Language (универсальный язык моделирования)
XML	eXtensible Markup Language (расширяемый язык разметки)
ИМ	Имитационная модель
ОС	Операционная система
ПП	Программный продукт
ПО	Программное обеспечение
ПЭВМ	Персональная электронно-вычислительная машина
РДО	Ресурс Действие Операция – система имитационного моделирования
ТЗ	Техническое задание
ЭП	Эскизный проект

ВВЕДЕНИЕ

В мире информационных технологий имитационное моделирование переживает второе рождение. Интерес к этому виду компьютерного моделирования оживился в связи с существенным технологическим развитием систем моделирования, которые на сегодняшний день являются мощным аналитическим средством, вобравшим в себя весь арсенал новейших информационных технологий, включая развитые графические оболочки для целей конструирования моделей и интерпретации выходных результатов моделирования, мультимедийные средства и видео, поддерживающие анимацию в реальном масштабе времени, объектно-ориентированное программирование, Internet – решения и др. В силу своей привлекательности и доступности эти технологии имитационного моделирования с легкостью покинули академические стены и сегодня осваиваются IT - специалистами в бизнесе.

Благодаря естественному желанию людей найти и использовать некоторые общесистемные принципы и методы с целью обобщения накопленного опыта и результатов в различных сферах человеческой деятельности моделирование систем становится все более популярным. Именно этот общесистемный подход в перспективе должен стать той базой, которая позволит исследователю работать с любой сложной системой, независимо от ее физической сущности.

Широкое использование ИМ в задачах анализа и синтеза систем объясняется сложностью (а иногда и невозможностью) применения строгих методов оптимизации, которая обусловлена размерностью решаемых задач и неформализуемостью сложных систем.

Анимация на сегодняшний день рассматривается как обязательная компонента коммерческих имитационных систем. Многие системы имитационного моделирования, присутствующие на рынке, имеют встроенные средства анимации, позволяющие отобразить и сам процесс имитационного моделирования и его результаты. Но они не позволяют ре

шить многие задачи по визуализации исследований, поскольку имеют выраженную академическую направленность, уровень которой является недостаточным для выполнения конкретных задач [15]. Задача привлечения к моделированию более широкого круга пользователей может успешно решаться привлечением новых программных средств анимации, позволяющих перейти от обычных абстрактных объектов имитационного моделирования к анимационным картинкам, адекватно описывающим процессы реального мира.

Существует много прикладных пакетов для создания анимации имитационных моделей.

Одни представляют собой объектно-ориентированные коммерческие пакеты имитационного моделирования, основанные на принципах визуального моделирования. Из них можно выделить: Arena (разработка американской корпорации Rockwell Automation), AnyLogic (продукт российской компании XJ Technologies), GPSS World, AutoMod, Extend, ProModel, QUEST, SIMFACTORY II.5, SIMPLE++ (eM-Plant), Taylor ED.

Другие являются самостоятельными средствами для создания анимации имитационной модели на основании определенных входных данных. В этом плане представляет конкретный практический интерес в применении для исследований и в обучении пакет Proof Animation корпорации WolverineSoftware (1993г.). Proof Animation является универсальным инструментом для создания анимации на основе трассировочных файлов, представленных в ASCII.

В последнее время наблюдается тенденция по выводу в мировую сеть технологий имитационного моделирования на основе специализированных языков, подобных HTML. Так возник VRML (Virtual Reality Modeling Language) - Язык Моделирования Виртуальной Реальности. Этот формат файлов используется для описания трехмерных объектов и миров для сети Internet. Первая реализация VRML (VRML 1.0

спецификация) была создана компанией Silicon Graphics и представляла собой формат описания статических миров. Во второй реализации VRML (VRML 2.0 спецификация) добавились более сложные интерактивные возможности и анимация. Она была разработана компанией Silicon Graphics в сотрудничестве с компаниями Sony и Mitra. Слабая сторона VRML-моделей состоит в том, что вопросы взаимосвязи таких моделей с имитационными моделями разработаны очень слабо [14].

С другой стороны в РДО уже имеется возможность для анимации в процессе моделирования. Данная анимация описывается с помощью синтаксических конструкций языка РДО в текстовом редакторе системы. Для выполнения анимации какого-либо параметра моделируемой системы прежде всего нужно задать кадр (или кадры, если их несколько), который будет выводится и продукционные правила, обеспечивающие отображение кадра на экране. Таким образом, минусом существующей анимации является необходимость знания языка РДО и непосредственно написания условий вывода кадров анимации.

Таким образом, разработка подсистемы анимации для графического редактора системы РДО - актуальная задача в сфере изложенных выше фактов. Ее решение позволит улучшить восприятие модели пользователем (от него не будет требоваться знание специальных конструкций языка РДО для отображения анимации в имитационной модели) и повысит конкурентноспособность ПП.

1. ПРЕДПРОЕКТНОЕ ИССЛЕДОВАНИЕ

1.1. Назначение программного комплекса РДО

Задачи системного анализа и синтеза объектов различной природы и назначения часто решаются с использованием имитационных моделей. Эти модели позволяют исследовать динамические аспекты поведения сложных дискретных систем и процессов. Имитация, в частности, позволяет выполнить анализ функционирования объекта, прогнозирование, организационное управление, поддержать принятие решений при проектировании и управлении.

RDO-studio является средством имитационного моделирования, позволяющим воспроизводить на ПЭВМ динамику объекта, принятие решений сложной системой управления, и даже моделировать деятельность человека при принятии решений. В основе имитатора лежит РДО-метод формализации знаний о дискретных системах и процессах. Знания представляются в форме модифицированных продукционных правил, событий и процессов. При этом сохраняются такие достоинства продукционных систем, как универсальность, гибкость и наличие формальных механизмов логического вывода. Традиционные продукционные правила являются частным случаем модифицированных, поэтому в имитационную модель легко могут быть включены, например, экспертные системы.

Язык описания объектов, алгоритмов управления и задач в RDO-studio – это по существу язык представления знаний. Он требует от пользователя лишь знаний в предметной области, а не в программировании. Пользователь описывает ресурсы, правила функционирования, требуемые показатели и анимационные кадры непосредственно в терминах предметной области, не прибегая при этом к представлению своей системы в терминах какого-либо известного метода

(системы очередей, сети Петри, автоматы) или языка типа SLAM-II, ARENA, SIMPLE++ и других.

Основные элементы RDO-studio – это модифицированная продукционная система, аппарат событий, с помощью которых в системе реализованы событийный, процессный и подход сканирования активностей. События начала действий базы знаний и процессов инициируются механизмом логического вывода, а события, запланированные пользователем (в явном виде) или системой (в неявном виде – события окончания действий) – специальным событийным блоком. При имитации состояние системы изменяется в соответствии с описанием события, происходящим в данный момент модельного времени. После любого изменения состояния, т.е. при каждом событии, вызывается система вывода. Она просматривает в базе знаний и процессах продукционные правила и проверяет по предусловиям, могут ли начаться какие-либо действия. При нахождении таких действий инициируются события их начала.

Продукционная система, блок имитации событий совместно осуществляют построение имитационной модели системы. На основании анализа результатов имитации вычисляются требуемые показатели функционирования системы.

Система трассировки выводит подробную информацию о событиях в специальный файл, который затем обрабатывается для детального анализа работы модели. Система анимации отображает на экране во время имитации поведение моделируемого объекта.

RDO-studio может быть применена для создания имитационных моделей, систем планирования, игр и тренажеров, экспертных систем реального времени и гибридных систем, включающих экспертные системы, имитационные модели и алгоритмы оптимизации.

1.2. Функции программного комплекса РДО

При выполнении работ, связанных с созданием и использованием ИМ в среде РДО, пользователь оперирует следующими основными понятиями.

Модель – совокупность объектов языка РДО, описывающих какой-то реальный объект, собираемые в процессе имитации показатели, кадры анимации и графические элементы, используемые при анимации, результаты трассировки.

Прогон – это единая неделимая точка имитационного эксперимента. Он характеризуется совокупностью объектов, представляющих собой исходные данные и результаты, полученные при запуске имитатора с этими исходными данными.

Проект – один или более прогонов, объединенных какой-либо общей целью. Например, это может быть совокупность прогонов, которые направлены на исследование одного конкретного объекта или выполнение одного контракта на имитационные исследования по одному или нескольким объектам.

Объект – совокупность информации, предназначенной для определенных целей и имеющая смысл для имитационной программы. Состав объектов обусловлен РДО-методом, определяющим парадигму представления сложной системы на языке РДО.

Объектами исходных данных являются:

- типы ресурсов (с расширением .rtp);
- ресурсы (с расширением .rss);
- событий (с расширением .evn);
- образцы активностей (с расширением .pat);
- точки принятия решений (с расширением .dpt);
- процессы (с расширением .prc);
- константы, функции и последовательности(с расширением .fun);

- кадры анимации (с расширением .frm, .bmp);
- требуемая статистика (с расширением .pmd);
- прогон (с расширением .smr);
- проект (с расширением .rdox).

Объекты, создаваемые РДО-имитатором при выполнении прогона:

- результаты (с расширением .pmv);
- трассировка (с расширением .trc) [1].

На данный момент РДО реализует следующие основные функции, которые представлены на 2 листе дипломного проекта (Функциональная структура РДО. Нотация ARIS):

1) Создание модели на языке РДО:

- создание основных объектов (*.rtp, *.rss, *.evn, *.pat, *.dpt, *.prc, *.smr);
- создание объектов данных и функций (*.fun);
- создание объектов вывода (*.frm, *.pmd, *.bmp, а также .prcx, в котором хранится графическая информация модели).

2) Проведение экспериментов:

- изменение параметров системы в процессе моделирования;
- генерация случайных чисел.

3) Вывод результатов моделирования:

- анимация объектов модели;
- вывод необходимых показателей;
- трассировка изменений объектов модели.

1.3. Предпосылки создания подсистемы анимации в РДО

Программные средства имитационного моделирования, которые используются для разработки имитационных моделей производственных систем, можно разделить на следующие четыре группы [2]:

1. Программирование компьютерной модели с помощью универсальных языков (например, C++, Delphi, Pascal). Динамику системы описывают уравнениями, которые кодируют в программу, затем проводят расчет уравнений и устанавливают связь выходных величин с входными.

2. Программирование компьютерной модели с применением специализированных языков моделирования (например, GPSS, AnyLogic), написанных на универсальных языках. Динамика системы отображается взаимодействием элементов модели во времени и пространстве. Специализированные языки имитационного моделирования компактны и имеют широкий круг приложений, однако требуют специальной подготовки пользователя, который должен написать программу в терминах языка для конкретного объекта моделирования.

3. Построение компьютерных моделей и проведение имитационных экспериментов при помощи специализированных компьютерных сред (например, Arena, AnyLogic, GPSS World, VisSim). Имитационные среды не требуют программирования в виде последовательности команд. Вместо написания программы пользователи составляют модель из библиотечных графических модулей, и/или заполняют специальные формы. Как правило, имитационная среда обеспечивает возможность визуализации процесса имитации, позволяет проводить сценарный анализ и поиск оптимальных решений.

4. Включение средств имитационного моделирования в стандартные математические компьютерные системы (например, пакет Simulink системы Matlab, Mathcad, Mathematica). Это программные среды, предназначенные для выполнения разнообразных математических и технических расчетов, предоставляющие пользователю инструменты для работы с формулами, числами, графиками, текстом, включают в себя средства для управления переменными, вводом и выводом данных, а также снабжены графическим интерфейсом.

В программном комплексе РДО пользователь имеет возможность создать имитационную модель двумя способами: либо описав ее в текстовом редакторе благодаря использованию конструкций языка РДО, либо построив графическую модель моделируемой системы блок-схемой, то есть прибегнув к помощи графических примитивов (см. 1 лист дипломного проекта).

Исходя из приведенной выше классификации программных средств имитационного моделирования, систему RDO-studio можно отнести ко 2 и 3 группам. Но, к сожалению, в ней отсутствует визуализация и анимация графической модели. С другой стороны, имитационная модель, написанная в текстовом редакторе, может включать в себя инструкции, написанные самим пользователем, для отображения анимации.

В качестве примера модели, в которой используется анимация, созданная в текстовом редакторе, приведем модель "Гуляющий человек". Данная модель будет описывать человека, шагающего по экрану вперед-назад и совершающего прыжки, удары и остановки при нажатии клавиш и кнопки мыши [1]. Идея анимации движения человека состоит в том, чтобы в зависимости от направления движения, режима и стадии отображать различные битовые карты, соответствующие этим направлениям, режиму и стадии. Различают шесть стадий ходьбы, две стадии удара, три стадии прыжка и одну стадию остановки - всего 12 стадий. Соответствующие битовые карты для движения вперед приведены на рис.1.1. Кроме того, существуют еще столько же битовых карт для отображения человека при движении назад. Таким образом, описание анимации состоит из 24 условных множеств, в каждом из которых изображается одна из битовых карт.

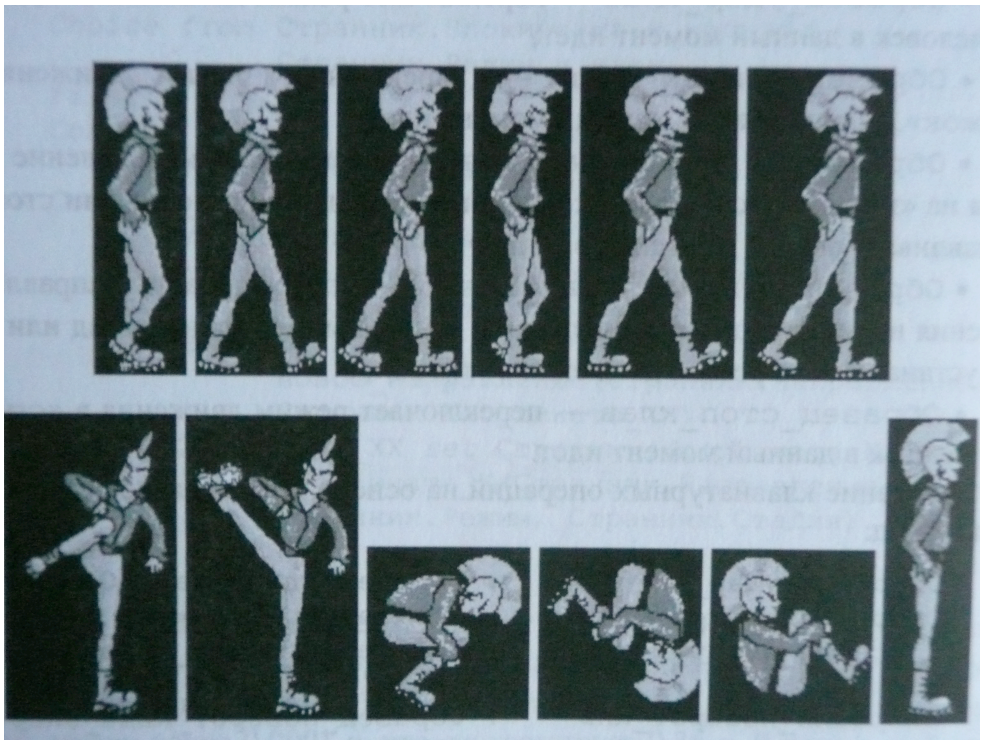

Рис.1.1. Битовые карты для отображения движения человека

Кадр анимации представлен на рис.1.2:

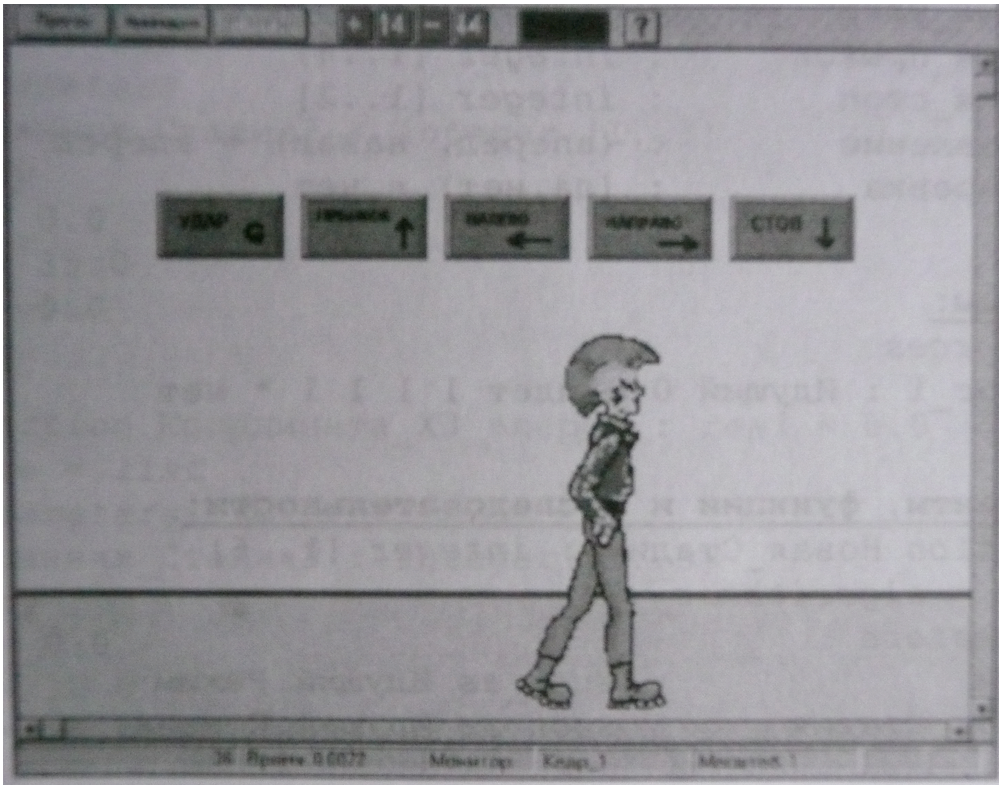

Рис.1.2. Анимация движения человека

Часть модели движущегося человека, отвечающая за отрисовку кадров, имеет следующий вид:

```
$Frame  Кадр_1
$Back_picture = <0 0 255> fon
Show
        bitmap [ 100,100, спор11, спор1m ]
        bitmap [ 200,100, спор12, спор1m ]
        bitmap [ 300,100, спор13, спор1m ]
        bitmap [ 400,100, спор14, спор1m ]
        bitmap [ 500,100, спор15, спор1m ]
        active Операция_удар_кл [ 100,100,100,50 ]
        active Операция_прыжок_кл [ 200,100,100,50 ]
        active Операция_поворот_клв [ 300,100,100,50 ]
        active Операция_поворот_кл [ 400,100,100,50 ]
        active Операция_стоп_кл [ 500,100,100,50 ]

        Show_if Ресурс_1.Режим = идет and Ресурс_1.Стадия = 1
                        and Ресурс_1.Направление = вперед
                bitmap [ Ресурс_1.Координата_XX,200, chag1, chag1m ]
        Show_if Ресурс_1.Режим = идет and Ресурс_1.Стадия = 2
                        and Ресурс_1.Направление = вперед
                bitmap [ Ресурс_1.Координата_XX,200, chag2, chag2m ]
        Show_if Ресурс_1.Режим = идет and Ресурс_1.Стадия = 3
                        and Ресурс_1.Направление = вперед
                bitmap [ Ресурс_1.Координата_XX,200, chag3, chag3m ]
        Show_if Ресурс_1.Режим = идет and Ресурс_1.Стадия = 4
                        and Ресурс_1.Направление = вперед
                bitmap [ Ресурс_1.Координата_XX,200, chag4, chag4m ]
        Show_if Ресурс_1.Режим = идет and Ресурс_1.Стадия = 5
                        and Ресурс_1.Направление = вперед
                bitmap [ Ресурс_1.Координата_XX,200, chag5, chag5m ]
        Show_if Ресурс_1.Режим = идет and Ресурс_1.Стадия = 6
                        and Ресурс_1.Направление = вперед
                bitmap [ Ресурс_1.Координата_XX,200, chag6, chag6m ]
        Show_if Ресурс_1.Режим = идет and Ресурс_1.Стадия = 1
                        and Ресурс_1.Направление = назад
                bitmap [ Ресурс_1.Координата_XX,200, chag11, chag11m ]
        Show_if Ресурс_1.Режим = идет and Ресурс_1.Стадия = 2
                        and Ресурс_1.Направление = назад
                bitmap [ Ресурс_1.Координата_XX,200, chag22, chag22m ]
        Show_if Ресурс_1.Режим = идет and Ресурс_1.Стадия = 3
                        and Ресурс_1.Направление = назад
                bitmap [ Ресурс_1.Координата_XX,200, chag33, chag33m ]
```

```
Show_if Ресурс_1.Режим = идет and Ресурс_1.Стадия = 4
                    and Ресурс_1.Направление = назад
          bitmap [ Ресурс_1.Координата_XX,200, chag44, chag44m ]
Show_if Ресурс_1.Режим = идет and Ресурс_1.Стадия = 5
                    and Ресурс_1.Направление = назад
          bitmap [ Ресурс_1.Координата_XX,200, chag55, chag55m ]
Show_if Ресурс_1.Режим = идет and Ресурс_1.Стадия = 6
                    and Ресурс_1.Направление = назад
          bitmap [ Ресурс_1.Координата_XX,200, chag66, chag66m ]
Show_if Ресурс_1.Режим = удар and
                    Ресурс_1.Стадия_удар = 1 and
                    Ресурс_1.Направление = вперед
          bitmap [ Ресурс_1.Координата_XX,200, cara1s, cara1m ]
.....
$End
```

Помимо самого кода, реализующего анимацию, как в примере выше, этот код должен быть неразделимо связан с остальной моделью системы. Как видно из примера, описание анимации подобным образом достаточно трудоемкий процесс и гибкость такого подхода вызывает сомнение.

Гораздо проще описать модель в графическом редакторе, но, к сожалению, без возможности анимации этой модели:

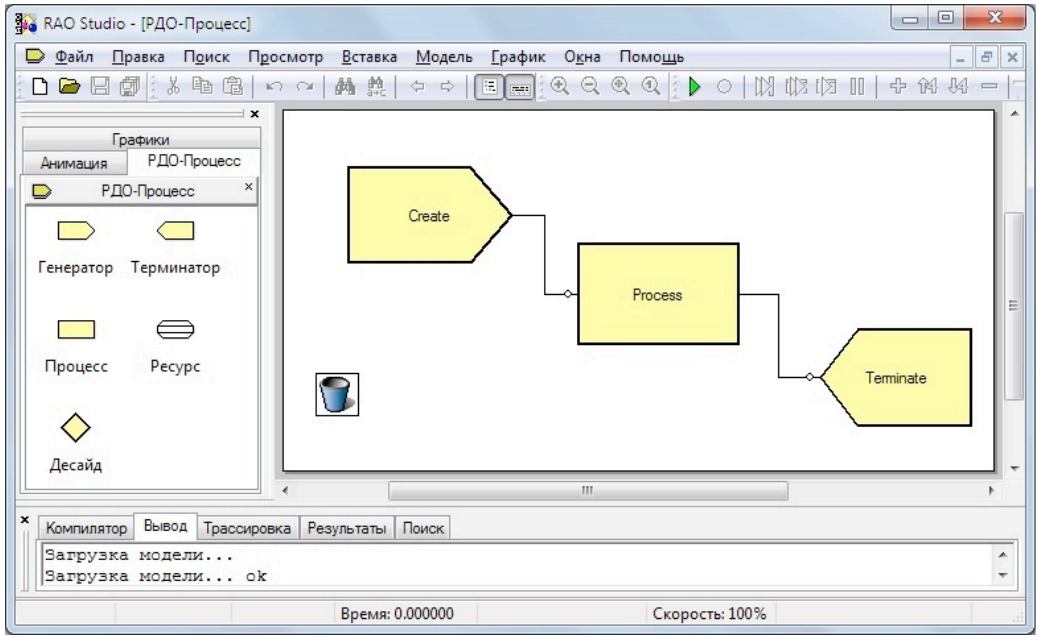

Рис.1.3. Модель системы массового обслуживания

1.4. Выводы по преддипломному этапу

Рассмотренная модель "Гуляющего человека", реализуемая конструкциями языка РДО в текстовом редакторе системы, показывает неудобство ее написания ввиду большого объема рукописного кода и возрастающей вместе с этим вероятности ошибок и, как следствие, возможном неправильном отображении сути модели в виде анимации на экране монитора.

С точки зрения пользователя, минусом такого подхода к отображению анимации является и тот факт, что разработчик модели должен знать язык РДО. В то время как сторонние производители имитационных систем предоставляют возможность пользователю своих систем не изучать новые для него специализированные языки моделирования и не программировать последовательность команд для работы анимации.

Существующая система анимации требует для работы анимации от пользователя дополнительных действий с его стороны. Ему необходимо приготовить битовые карты для каждого изменения состояния имитационной модели, продумать продукционные правила при выполнении которых стоит заменить одну битовую карту другой, отражающей в большей степени происходящие изменения в моделируемой системе. Все это занимает для сложных систем много времени и только замедляет процесс разработки имитационной модели.

2. КОНЦЕПТУАЛЬНОЕ ПРОЕКТИРОВАНИЕ

2.1. Цели разработки подсистемы

Основной целью данного дипломного проекта является **повышение адекватности в среде РДО**, которая состоит из трех подцелей:

- отслеживание выполнения модели в процессе моделирования;
- увеличение наглядности модели;
- сокращение времени на обучение системе новых пользователей.

В свою очередь, увеличение наглядности модели также декомпозируется на 2 подцели:

- визуализация модели;
- анимирование модели.

Благодаря введению подсистемы анимации в программный комплекс RDO-studio для имитационных моделей, описанных в графическом редакторе, пользователь может отслеживать выполнение модели в процессе моделирования, что позволит ускорить процесс отладки модели за счет своевременного обнаружения ошибок, сократит время на понимание сути имитационной модели, что может быть достигнуто увеличением наглядности модели. В отличие от анимации, которая существует в РДО, пользователю не надо беспокоится о создании кадров анимации, написании кода, который бы выводил их на экран при выполнении каких-либо условий, знать язык РДО. Также применяемый процессно-ориентированный подход при построении имитационных моделей с помощью графических примитивов менее ресурсоемок по сравнению с другими подходами имитационного моделирования в РДО, например, подхода сканирования активностей.

Из этого можно сделать вывод о том, что после разработки подсистемы анимации, все рассмотренные цели будут достигнуты и это приведет к повышению адекватности модели в среде РДО.

2.2. Функциональное моделирование подсистемы анимации

Используя методологию функционального моделирования IDEF0 (Integration Definition for Function Modeling), представим проектируемую подсистему анимации в виде набора взаимосвязанных функциональных блоков.

Сначала опишем контекстную диаграмму верхнего уровня, на которой объект моделирования представлен единственным блоком с граничными стрелками. Эта диаграмма называется A-0. Стрелки на этой диаграмме будут отображать связи объекта моделирования с окружающей средой. Диаграмма A-0 установит область моделирования подсистемы и ее границу. Далее рассмотрим по отдельности, с точки зрения разработчика модели (пользователя системы РДО), два подхода к созданию модели (в текстовом и графическом редакторе, см. лист 3 дипломного проекта).

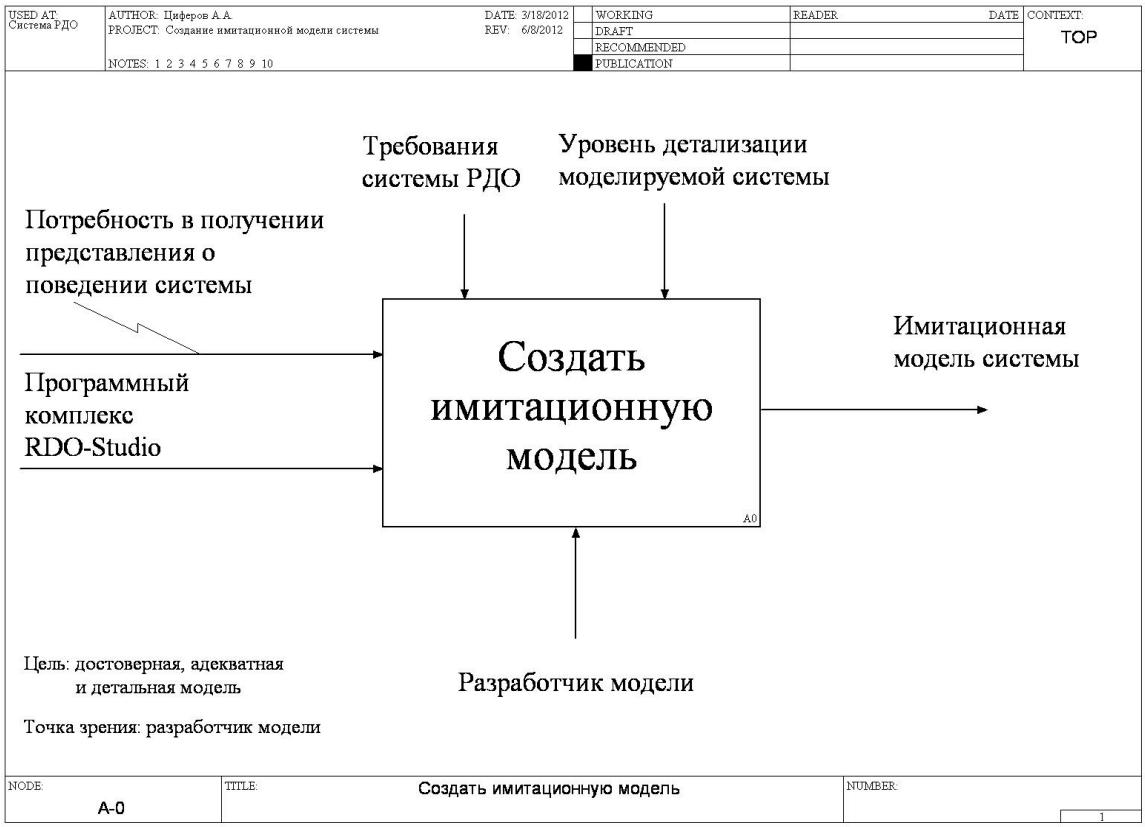

Рис.2.1. Диаграмма A-0

Общей последовательностью действий со стороны разработчика имитационной модели в системе РДО будь то использование им текстового или графического редактора, будут:

- описание модели в системе РДО;
- отладка написанной модели (проверка на предмет ошибок);
- проведение имитационных экспериментов;
- анализ созданной модели (соответствие требованиям, наложенными на модель);

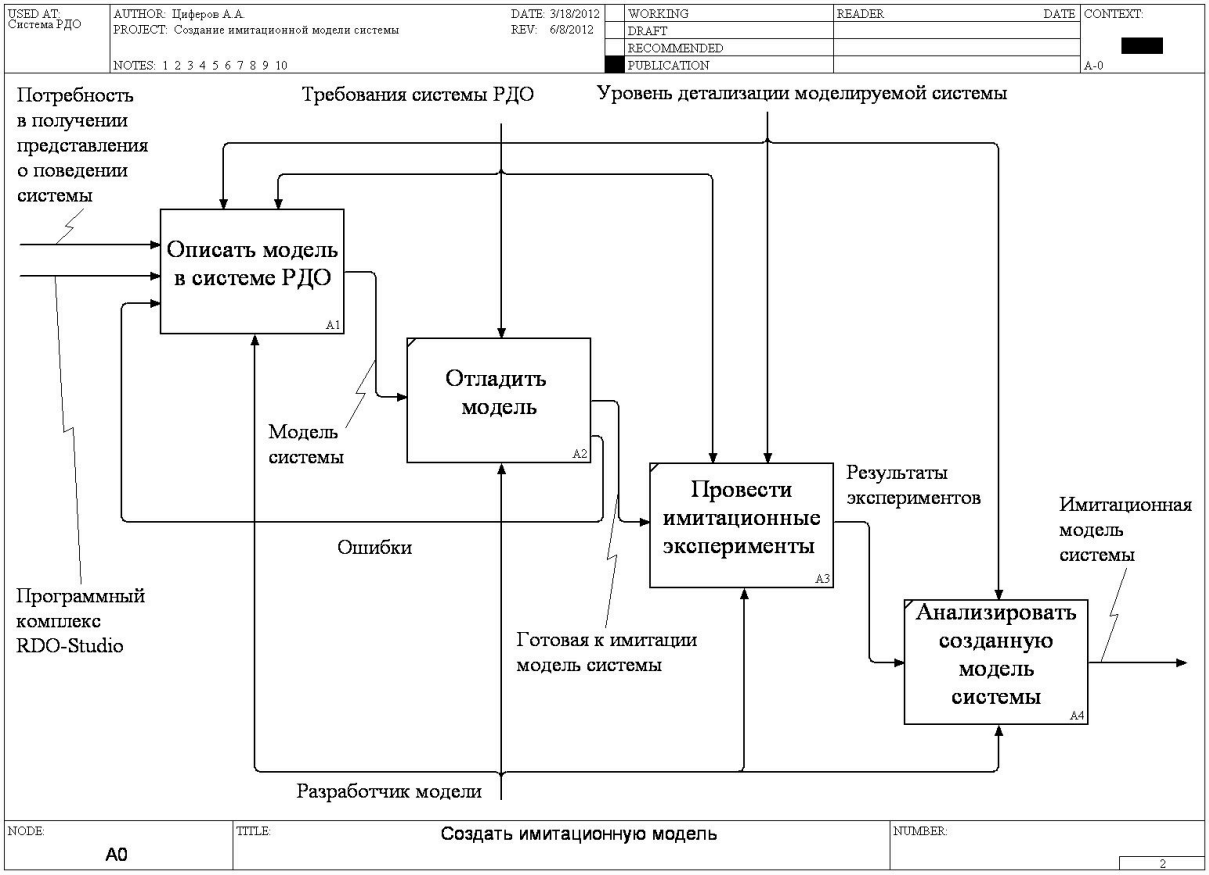

Рис.2.2. Диаграмма A0

Как видно из диаграммы на рис.2.2, пользователь, в случае некорректного описания модели в системе, может исправить ее в соответствии с требованиями системы РДО или же более детализировать, если результаты экспериментов, произведенных на ней, окажутся неадекватными.

Различия между двумя подходами создания модели начинаются с декомпозиции функционального блока A1 - "Описать модель в системе РДО". Как упоминалось выше, пользователь может набрать модель словами, придерживаясь, в частности, синтаксических правил языка РДО. Тем самым он изначально формализует моделируемую систему ключевыми словами и конструкциями внутреннего специализированного языка RDO-studio. Запрограммировав систему он получает "Модель системы". Такой способ требует знаний специализированного языка моделирования, в независимости от применяемого подхода к построению имитационной модели: дискретно-событийного, подхода сканирования активностей или процессно-ориентированного. В большинстве случаев построение моделей в текстовом редакторе с возможностью ее анимации есть трудоемкий процесс, поэтому обычно имитация модели состоит из следующих этапов: генерирование результатов имитации, вывод трассировки модели и возврат кода завершения имитации.

В противоположность созданию модели стандартным образом (используя текстовый редактор), описание моделируемой системы в графическом редакторе с помощью графических примитивов менее трудозатратно и не влечет за собой написания ни одной строчки кода пользователем, ему достаточно лишь набрать в соответствующие поля диалоговых окон графических примитивов параметры модели (см. рис.2.3).

Последовательность действий разработчика модели будет несколько иной. Во-первых, пользователь освобожден от знаний языка РДО, модель создается благодаря блок-схеме. Во-вторых, благодаря интеграции всех трех методологических подходов к построению модели, изложенных выше, пользователю нет необходимости задумываться о том, какой подход используется при интерпретации блок-схемы внутри системы. Данный аспект, безусловно, скрыт от него, но создание модели таким образом нисколько не влияет на результаты моделирования.

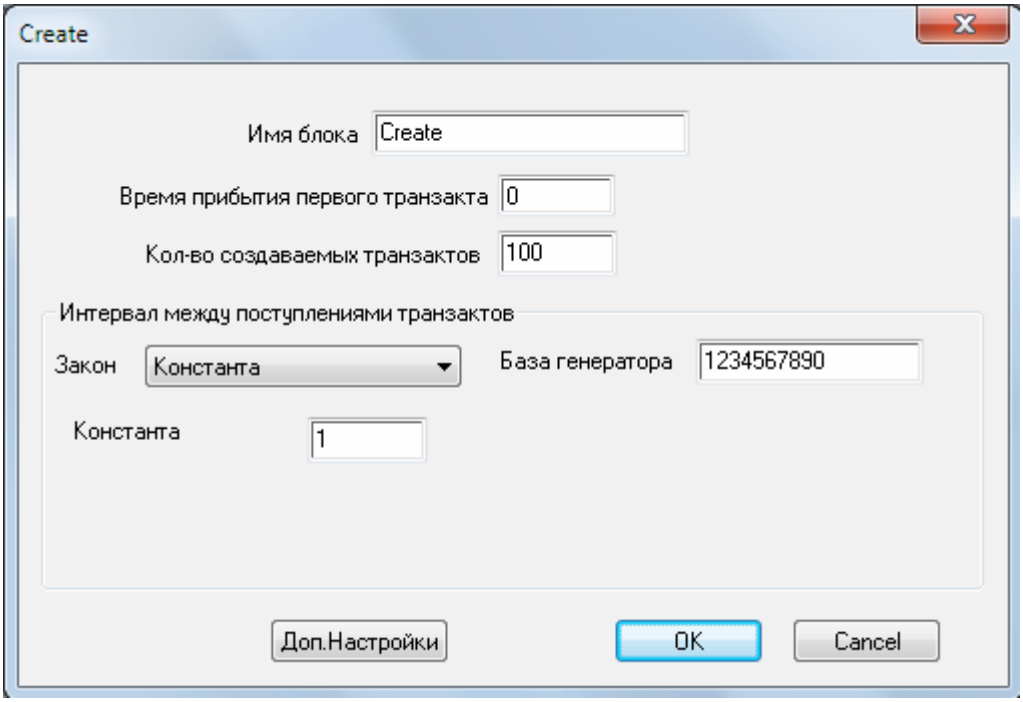

Рис.2.3. Диалоговое окно блока Create

Преследуя главную цель дипломного проекта, в декомпозиции функционального блока IDEF0 "Имитировать модель" вводится дополнительная функция, а именно, функция показа анимации в процессе моделирования графической имитационной модели. Это нововведение расширит возможность ПП RDO-Studio, облегчит восприятие имитационной модели пользователем и, в отличие от кадровой анимации, существующей в текстовом редакторе, будет автоматически анимировать модель.

2.3. Компоненты РДО

Диаграмма компонентов, изображенная на листе 5 дипломного проекта, описывает особенности физического представления подсистемы. Она позволяет определить архитектуру разрабатываемой подсистемы, установив зависимости между программными компонентами, в роли которых выступают библиотеки, папки, исполняемые файлы и исходный код системы.

Система имитационного моделирования РДО безусловно является сложной и статически, и динамически. На это указывает сложная иерархическая структура системы со множеством различных связей между компонентами и ее сложное поведение во времени.

Ярко выраженная иерархическая структура и модульность системы определяют направление изучения системы сверху вниз. Т.е. принцип декомпозиции применяется до тех пор, пока не будет достигнут уровень абстракции, представление на котором нужных объектов не нуждается в дальнейшей детализации для решения данной задачи.

Базовый функционал представленных на диаграмме компонентов [5]:

rdo_kernel реализует ядровые функции системы. Не изменяется при разработке подсистемы.

RAO-studio.exe реализует графический интерфейс пользователя. Не изменяется при разработки подсистемы.

rdo_repository реализует управление потоками данных внутри системы и отвечает за хранение и получение информации о модели. Не изменяется при разработке подсистемы. В него сохраняется файл графической модели расширением .prcx.

rdo_simulator управляет процессом моделирования на всех его этапах. Он осуществляет координацию и управление компонентами rdo_runtime и rdo_parser. Не изменяется при разработке подсистемы.

rdo_parser производит лексический и синтаксический разбор исходных текстов модели, написанной на языке РДО.

rdo_runtime отвечает за непосредственное выполнение модели, управление базой данных, базой знаний, планирование и выполнение событий, и работу процессов. Модернизируется при разработке подсистемы (файлы generate.cpp, advance.cpp, terminate.cpp модернизируются для посылки информации о изменении статистических данных от имитатора в графический редактор). Так же в библиотеку rdo_runtime входят файлы procgui.cpp и procgui_datablock.cpp, лежащие в папке компилятора процессно-ориентированного графического редактора.

rdo_studio отвечает за работу системы имитационного моделирования. В этой библиотеке содержатся методы управления средой имитационного моделирования (файл model.cpp).

rdo_studio_mfc отвечает за разработку графического интерфейса системы. В частности, в ней лежат файлы, которые декларируют методы отображения объектов на экране (rdoprocess_shape_create.cpp, rdoprocess_shape_process.cpp, rdoprocess_shape_decide.cpp, rdoprocess_shape_resource.cpp, rdoprocess_shape_terminate.cpp - описание графических примитивов, соответствующих процессно-ориентированной методологии).

Дополнительно подключена кроссплатформенная библиотека pugixml, с помощью которой производится запись информации о графической модели в формате xml в файл ".prcx" репозитария системы РДО.

Таким образом, основные изменения должны затронуть перечисленные библиотеки (кроме библиотеки pugixml) и файлы, включенный в них. На основании этих файлов будет построена подсистема анимации процессно-ориентированного подхода графической модели.

2.4. Разработка технического задания

Техническое задание разработано в соответствии с ГОСТ 19.201-78 (Единая система программной документации. Техническое задание. Требования к содержанию и оформлению) [8].

2.4.1. Введение

Разработка подсистемы анимации для графического редактора РДО имеет большое значение для системы. Ввод этой подсистемы в действие позволит повысить адекватность моделируемых системы, увеличит наглядность моделей, сократит время на обучение системе новых пользователей и создаст все условия для отслеживания процесса моделирования реальных систем. Разрабатываемая подсистема анимации является автоматической, то есть не требует никаких дополнительных действий для своей работы, в отличие от реализованной на данный момент кадровой анимации в системе РДО. И, как следствие, нет необходимости знать язык РДО для ее задания и запуска. Со временем в подсистему могут быть добавлены новые графические примитивы и возможность манипулирования ими в ходе моделирования.

2.4.2. Основания для разработки

1) Разработка ведется на основании следующих документов:

- Задание на выполнение дипломного проекта.
- Календарный план на выполнение дипломного проекта.

2) Документы утверждены « 12 » февраля 2012 года.

3) Тема дипломного проекта:

Разработка подсистемы анимации в РДО.

2.4.3. Назначение разработки

Функциональное и эксплуатационное назначение разработки подсистемы анимации в РДО состоит в создании инструмента, позволяющего в автоматическом режиме в процессе моделирования анимировать модель, в частности, визуализировать и анимировать статистические данные имитируемой системы.

2.4.4. Требования к программе или программному изделию

2.4.4.1. Требования к функциональным характеристикам

Требования к составу выполняемых функций реализуемой подсистемы заключаются в возможности анимации статистических данных, получаемых от имитатора системы РДО, в процессе моделирования. Организация входных данных построена на существующей системе имитатора, из которых в режиме имитации реальной системы берутся значения необходимых параметров модели. Выходные данные представляют собой отображаемые с помощью пользовательского интерфейса ОС Microsoft Windows значения статистических характеристик моделируемой системы.

2.4.4.2. Требования к надежности

Действия пользователя (разработчика модели) не должны приводить к сбоям в работе программы.

2.4.4.3. Условия эксплуатации

Аппаратные средства должны эксплуатироваться в помещениях с выделенной розеточной электросетью 220В ±10%, 50 Гц с защитным заземлением при следующих климатических условиях:

- температура окружающей среды – от 15 до 30 градусов С;
- относительная влажность воздуха - от 30% до 80%;
- атмосферное давление - от 630 мм. р.с. до 800 мм. р.с.

2.4.4.4. Требования к составу и параметрам технических средств

Программный продукт должен работать на компьютерах со следующими характеристиками:

- объем ОЗУ не менее 256 Мб;
- объем жесткого диска не менее 20 Гб;
- микропроцессор с тактовой частотой не менее 400 МГц;
- монитор не менее 15” с разрешением от 800*600 и выше.

2.4.4.5. Требования к информационной и программной совместимости

Данная система должна работать под управлением операционных систем семейства Microsoft Windows для рабочих станций:

- Windows XP,
- Windows Vista,
- Windows 7;

а также для серверов:

- Windows Server 2003,
- Windows Server 2008,

2.4.4.6. Требования к маркировке и упаковке

Не предъявляются.

2.4.4.7. Требования к транспортированию и хранению

Не предъявляются.

2.4.4.8. Требования к программной документации

Не предъявляются

2.4.5. Технико-экономические показатели

Расчет экономической эффективности разработанной подсистемы анимации не является целью дипломного проектирования, однако возможный экономический эффект может быть достигнут за счет следующих преимуществ подсистемы:

1). При разработке новых моделей разработчик сразу в режиме моделирования может видеть происходящие процессы в имитируемой системе. Этот факт сокращает время на обнаружение ошибок в системе и на обучение новых пользователей.

2). Разрабатываемая подсистема анимации автоматическая, не требующая со стороны пользователя никаких действий для своей работы. Ему не приходится заниматься ее построением так, как это делается на сегодняшний момент по отношению к анимации, описываемой в текстовом редакторе системы РДО.

3). Предлагаемая подсистема анимации не использует для своей работы сторонних продуктов, являясь неотъемлемой частью системы РДО, что позволяет не тратить средств на приобретение сторонних средств для поставленной цели и их интеграцию.

4). Подсистема анимации независимая, ее можно отключить. При этом на работу имитационной модели она не влияет.

2.4.6. Стадии и этапы разработки

Состав, содержание и сроки выполнения работ по созданию подсистемы в соответствии с календарным планом на выполнение дипломного проекта приведены ниже (см. Таблица 2.1).

Стадии	Этапы работ	Срок выполнения
Предпроектное исследование	Предпроектное исследование РДО с точки зрения перспективы внедрения подсистемы анимации графического редактора	25.02.12
Концептуальное проектирование	Разработка концепции реализации подсистемы	15.03.12
	Разработка технического задания на подсистему	20.03.12
Техническое проектирование	Разработка алгоритмов функционирования подсистемы, структуры программных средств	01.04.12
Рабочее проектирование	Программная реализация функционала подсистемы	05.05.12
	Разработка тестового примера модели, апробирование подсистемы	10.05.12
Организационно-экономическая часть	Расчет затрат на разработку подсистемы	15.05.12
Мероприятия по охране труда и технике безопасности	Требования безопасности при работе с подсистемой	20.05.12

Таблица 2.1. Стадии и этапы разработки подсистемы

2.4.7. Порядок контроля и приемки

Контроль и приемка подсистемы анимации должны осуществляться в процессе проверки функциональности системы имитационного моделирования на тестовом примере с включенным режимом анимации в соответствии с требованиями к функциональным характеристикам подсистемы.

2.4.8. Приложения

Документы, используемые при разработке приведены в списке использованных источников.

Используемое при разработке программное обеспечение:

- Операционная система Microsoft Windows 7 Professional;
- Среда разработки Microsoft Visual Studio 2008 Professional;
- Текстовый редактор Notepad++ 6.0;
- Библиотека для чтения и записи XML-файлов PugiXML 1.1;
- Централизованная система управления версиями Subversion;
- Система документирования исходных текстов Doxygen;
- Исходные коды системы РДО.

3. ТЕХНИЧЕСКОЕ ПРОЕКТИРОВАНИЕ

3.1. Проектирование прямого канала связи

На концептуальном этапе проектирования был сделан вывод о необходимости разработки подсистемы анимации в системе РДО, основной функцией которой будет анимирование моделей, созданных в графическом редакторе.

Для того, чтобы осуществить анимирование какого-либо параметра имитационной модели, требуется наладить как прямую так и обратную связь между двумя различными по своей природе и назначению объектами системы RDO-studio. А именно, речь идет о имитаторе РДО, который является математическим аппаратом системы, производящим

расчеты всевозможных параметров имитационной модели, и графическом редакторе, отвечающим за отображение этих параметров.

Роль посредника между двумя этими объектами играет так называемая "нить" (thread). Она позволяет общаться приложениям среды РДО по средствам отправки друг другу сообщений с помощью метода sendMessage. Сигнатура этого метода выглядит следующим образом [3]:

```
void sendMessage(PTR(RDOThread) to, RDOTreadMessage message, PTR(void) pParam = NULL)
```

Тип возвращаемого значения метода sendMessage - void, - что вполне логично, так как задача функции - пересылка данных. Внутренние параметры метода говорят сами за себя: PTR(RDOThread) to - имя той нити, которой адресуется сообщение; RDOThreadMessage message - сообщение, которое в дальнейшем обрабатывается указанной нитью, список возможных сообщений строго продекларирован и известен нити; PTR(void) pParam - как раз те самые данные, которые передаются от одного приложения системы РДО к другому.

Описанный выше подход к передаче сообщений между различными приложениями системы РДО находит свое применение и при компиляции графических примитивов модели. На листе 6 дипломного проекта ("Диаграмма последовательности") представлен процесс преобразования графического объекта, который пользователь расположил в окне графического редактора, в объект имитатора для дальнейшего математического расчета созданной разработчиком модели системы. Как было показано в преддипломной части настоящей пояснительной записки, разработчик имеет возможность задать параметры модели в диалоговых окнах соответствующих графических примитивов блок-схемы. Эти данные поступают в компилятор графической части системы, в нем создается объект, параметры которого определяются заданными пользователем в диалоговом окне значениями. Сообщение о созданном

таким образом объекте передается нити RDOThreadSimulator, которая, в свою очередь, преобразовывает поступившие к ней параметры к подходящему виду параметров имитатора. Затем процессный блок имитатора создает уже из этих данных создает объект имитатора и работает в последствие с ним.

Последовательность, приведенная выше, существовала в РДО. Необходимо только было передать вместе с остальными параметрами графического блока еще и указатель на этот блок. Для блока "Create", например, получение указателя на самого себя будет выглядеть так:

```
LPRPShapeCreateMJ pThis(this);
ASSERT(pThis);
```

3.2. Проектирование обратного канала связи

Передать параметры графической модели в блок имитатора системы РДО - это половина дела. Для отображения параметров модели на рабочем пространстве графического редактора, пусть даже простого изменения количества созданных транзактов блоком "Create", важно осуществить обратную связь блока имитатора с блоком графической части системы РДО.

Каждое изменение параметров моделируемой системы, источником которых является блок имитатора, должно корректно и своевременно отображаться в графическом приложении системы. Для решения поставленной задачи были введены два интерфейса, обеспечивающие связь между имитатором и графическим редактором. Как видно из листа № 7, интерфейс IInternalStatistics позволяет с помощью метода setTransCount сообщать графическому примитиву текущее количество созданных транзактов в системе. Связующим звеном между графическим компилятором системы и блоком имитатора служит интерфейс IInternalStatisticsManager, содержащий метод setStatistics.

Диаграмма классов поясняет логический уровень подсистемы анимации. Классы RPShpapeCreateMJ, RPShapeProcessMJ и RPShapeTerminate реализуют интерфейс IInternalStatistics. Данные классы отвечают за отрисовку графических объектов в окне графического редактора, но для имитационной модели этого недостаточно, нужно знать параметры модели. Эти параметры вводятся в системе в диалоговых окнах соответствующих графических блоках Create, Process и Terminate. Для сбора этих параметров и преобразования их в понятный системе язык и существует компилятор графической части, представленный на диаграмме классами RPShapeDataBlockCreate, RPShapeDataBlockProcess и RPShapeDataBlockTerminate. Полученные в режиме реального времени статистические данные (в нашем примере отображение количества созданных транзактов) благодаря методам setTransCount и drawCustom, определенных в блоках графической части, отображаются на блок-схеме модели. С другой стороны классы имитатора (RDOPROCGenerate, RDOPROCAdvance и RDOPROCTerminate) через интерфейс IInternalStatisticsManager и его метод setStatistics сообщают графическому редактору о факте изменения статистики. Например, при создании нового транзакта математической частью системы РДО эта информация транслируется непосредственно в графический редактор и, как следствие, происходит вывод и анимирование этой информации внизу блока графической модели.

3.3. Анимация статистических параметров

После того, как поднят канал общения между приложениями графической части и имитатора системы РДО, самое время отобразить, например, статистические данные о текущем количестве транзактов в системе, созданных в процессном блоке "Create". Для данной цели будет создан новый метод - `drawCustom(dc);` в качестве параметра которого выступает экземпляр класса CDC библиотеки Microsoft Foundation Classes, отвечающий за объекты контекста устройства.

Основной идеей отображения числа транзактов у блока "Create" является перерисовка окна и отображение графического примитива с уже новым значением количества транзактов в системе. За счет этого достигается эффетк анимации. За перерисовку отвечает существующий в системе метод:

```
void RPObjectFlowChart::update()
{
	if ( flowchart && can_update ) {
		flowchart->InvalidateRect( NULL );
		flowchart->UpdateWindow();
	}
}
```

4. РАБОЧЕЕ ПРОЕКТИРОВАНИЕ

4.1. Дополнение прямого канала связи

Как было показано на этапе технического проектирования, прямой канал связи графического редактора и имитатора системы РДО уже существует. Требуется лишь добавить к передаваемым параметрам со стороны графического редактора указатель на текущий графический блок. Код метода генерации, в который были внесены эти изменения, приведен для блока "Create":

```
void RPShapeCreateMJ::generate()
{
	rdo::compiler::gui::RPShapeDataBlock::zakonRaspr zakon;
	switch(gtype)
	{
		case 0: // константа
			zakon = rdo::compiler::gui::RPShapeDataBlock::Const;
			break;
		case 1: // нормальный
			zakon = rdo::compiler::gui::RPShapeDataBlock::Normal;
			break;
		case 2: // равномерный закон
			zakon = rdo::compiler::gui::RPShapeDataBlock::Uniform;
			break;
		case 3: // треугольный
			zakon = rdo::compiler::gui::RPShapeDataBlock::Triangular;
			break;
		case 4: // экспоненциальный
			zakon = rdo::compiler::gui::RPShapeDataBlock::Exp;
			break;
	}

	LPRPShapeCreateMJ pThis(this);
	ASSERT(pThis);

	pInternalStatistics =
pThis.interface_cast<rdo::runtime::IInternalStatistics>();
	ASSERT(pInternalStatistics);

	m_pParams =
rdo::Factory<rdo::compiler::gui::RPShapeDataBlockCreate>::create(zakon,
gname);
	m_pParams->setBase(base_gen);
	m_pParams->setAmount(gamount);
	m_pParams->setDisp(gdisp);
	m_pParams->setExp(gexp);
	m_pParams->setMax(gmax);
	m_pParams->setStatistics(pInternalStatistics);

	studioApp.m_pStudioGUI->sendMessage(kernel->simulator(),
RDOThread::RT_PROCGUI_BLOCK_CREATE, m_pParams.get());

	m_pParams = NULL;
}
```

Где метод setStatistics(pInternalStatistics) передает указатель на данный графический блок сначала нити, а затем и процессному блоку GENERATE, являющимся аналогом блока Create в имитаторе:

```
void
RPShapeDataBlockCreate::setStatistics(CREF(rdo::runtime::LPIInternalStatisti
cs) pStatistics)
{
	m_pStatistics = pStatistics;
}
```

4.2. Реализация обратного канала связи

Обратный канал связи фактически необходимо разрабатывать с нуля. Со стороны имитатора передача количества транзактов происходит при вызове метода `setTransCount:`

```
IBaseOperation::BOResult RDOPROCGenerate::onDoOperation(CREF(LPRDORuntime)
pRuntime)
{
	++m_TransCount;

	if (m_pStatistics)
		m_pStatistics->setTransCount(m_TransCount);

	LPRDOPROCTransact pTransact = m_pCreateAndGoOnTransactCalc-
>calcValue(pRuntime).getPointerSafety<RDOResourceTypeTransact>();
	ASSERT(pTransact);

	pTransact->setBlock(this);
	pTransact->next();

	PTR(RDOTrace) tracer = pRuntime->getTracer();
	if (!tracer->isNull())
	{
		tracer->getOStream() << pTransact->traceResourceState('\0',
pRuntime) << tracer->getEOL();
	}

	calcNextTimeInterval(pRuntime);
	return IBaseOperation::BOR_done;
}
```

А со стороны блока процесс принятия этого параметра выглядит следующим образом:

```
void RPShapeCreateMJ::setTransCount(ruint count)
{
```

```
	m_currentTransactCount = count;
	update();
}
```

4.3. Реализация анимации

Метод `drawCustom(dc)`, о котором шла речь ранее в на этапе технического проектирования, является виртаульным методом [4]. Вызывается он впервые после отрисовки на рабочем пространстве графического редактора соединительных линий и обводных прямоугольников фигур, а также после задания кисти, фона и прочих аттрибутов объекта контекстного устройства:

```
void RPShape::draw(REF(CDC) dc)
{
	RPObjectMatrix::draw(dc);

	// Перевод фигуры в глобальные координаты
	transformToGlobal();

	// Отрисовка полигона
	drawPolyline(dc);

	// Отрисовка доков для конекторов
	drawDocks(dc);

	LOGFONT lf;
	text_font.GetLogFont(&lf);
	lf.lfEscapement = static_cast<int>(getRotationGlobal()*10);

	CFont font;
	font.CreateFontIndirect(&lf);

	CFont* old_font = dc.SelectObject(&font);

	// Вывод имени
	if (text_show)
	{
		CRect calc(0, 0, 1, 1);
		dc.DrawText(name.c_str(), &calc, DT_CALCRECT | DT_SINGLELINE);
		rp::point center(-calc.Width()/2, -calc.Height()/2);
		center = globalMatrix(m_all & ~m_sc, m_all & ~m_sc)*center;
		dc.SetTextColor(text_color);
		dc.TextOut(static_cast<int>(center.x),
static_cast<int>(center.y), name.c_str());
	}

	drawCustom(dc);

	dc.SelectObject(old_font);
}
```

Реализация анимации окончательно происходит в переопределенном методе drawCustom(dc) на стороне фигуры (графического примитива):

```
void RPShapeCreateMJ::drawCustom(REF(CDC) dc)
{
	dc.SetTextColor(RGB(0x00, 0x64, 0x00));
	dc.TextOut((3*(this->pa_global.getMaxX()) + (this-
>pa_global.getMinX()))/4 - indent, this->pa_global.getMaxY() + indent,
rp::string::fromint(m_currentTransactCount).c_str());
}
```

5. ИССЛЕДОВАТЕЛЬСКАЯ ЧАСТЬ

5.1. Постановка задачи

Основной задачей дипломного проекта является разработка подсистемы анимации в РДО, которая анимировала бы графическую модель системы. Но, если не иметь возможности сохранять информацию о построенной графической модели в редакторе системы, то при следующем запуске программного комплекса этой информации неоткуда будет взяться. И все затраты на разработку подсистемы сойдут на нет.

В качестве средства для записи в файл информации в формате xml была применена кроссплатформенная библиотека под названием PugiXML. Эта библиотека предоставляет несколько различных функций для чтения (загрузки в парсер) данных длячтения из файлов, чтения из потоков и чтения из памяти. С помощью данной библиотеки осуществляется запись и последующая загрузка графической информации в формате XML (eXtensible Markup Language). Этот расширяемый язык разметки представляет из себя текстовый формат, предназначенный для хранения структурированных данных (например, взамен файлов баз данных), для обмена информацией между программами, а также для создания на его основе более специализированных языков разметки (XHTML как пример).

XML это описанные в текстовом формате иерархические данные. XML можно использовать для хранения любых данных. Визуально структура данных формата может быть представлена как дерево элементов, а сами элементы описываются тегами.

Задача, которую нужно исследовать с помощью библиотеки PugiXML, звучит так: найти оптимальный по структуре записываемого файла способ сохранения графической информации имитационной модели.

5.2. Альтернативные решения и их сравнение

Прежде чем сохранять в файл информацию о графической модели, будет не лишним оценить объем этой возможной информации. Помочь в этом может кроссплатформенная система документирования исходных текстов Doxygen. Эта система генерирует документацию на основе исходных текстов, а также может быть настроена на извлечение структуры программы из недокументированных исходных файлов. Пример извлечения информации:

RPObject Class Reference
Inheritance diagram for RPObject:

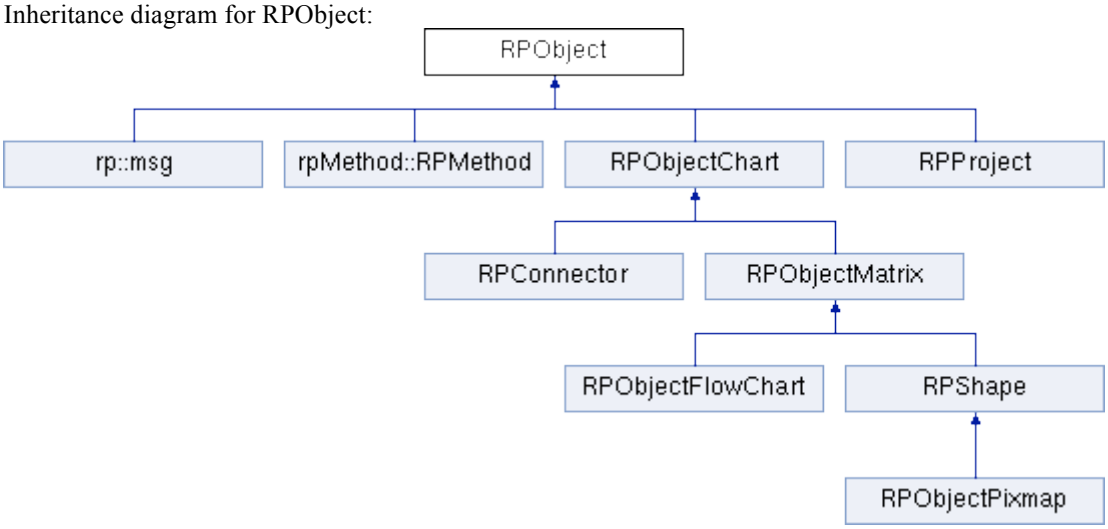

Рис.5.1. Пример построения иерархии doxygen

Таблица 5.1. Извлечение данных doxygen

Public Member Functions

	RPObject (RPObject *parent=NULL, const rp::string &name="object")
virtual rp::string	**getClassName** () const =0
const RPObjectClassInfo *	**getClassInfo** () const
virtual void	**load** (rp::RPXMLNode *node)
virtual rp::RPXMLNode *	**save** (rp::RPXMLNode *parent_node)
virtual void	**saveToXML** (REF(pugi::xml_node) parentNode) const =0
virtual void	**loadFromXML** (CREF(pugi::xml_node) node)=0
rbool	**hasChild** () const
std::list< RPObject * > ::const_iterator	**find_child** (const RPObject *object)
RPObject *	**find_child** (const rp::string &name) const
const std::list< RPObject * > &	**getChild** () const
void	**getAllChild** (std::list< RPObject * > &all_child, rbool recursive=false) const

void	**getAllChildByClass** (std::list< RPObject * > &all_child, const rp::string &class_name, rbool recursive=false) const
void	**getAllChildByClasses** (std::list< RPObject * > &all_child, const std::list< rp::string > &class_names, rbool recursive=false) const
void	**getAllChildNotByClass** (std::list< RPObject * > &all_child, const rp::string &class_name, rbool recursive=false) const
rp::string	**getName** () const
rp::string	**getFullName** () const
virtual rbool	**setName** (const rp::string &value)
void	**setCorrectName** (const rp::string &value)
rbool	**isSelected** () const
virtual void	**setSelected** (rbool value)
virtual rbool	**isChartObject** () const

Protected Member Functions

rbool	**isChildNameCorrect** (const RPObject *obj) const
void	**setCorrectChildName** (RPObject *obj)
std::list< RPObject * > ::const_iterator	**begin** () const
std::list< RPObject * > ::const_iterator	**end** () const
void	**clear** ()
void	**selectChildOff** (RPObject *withoutObj=NULL)
void	**save_child** (rp::RPXMLNode *node) const
virtual void	**notify** (RPObject *from, UINT message, void *param=NULL)
virtual void	**modify** ()
RPObject *	**get_this** ()
virtual rp::string	**get_xml_node_name** () const

Protected Attributes

RPObject *	**parent**
std::list< RPObject * >	**child**
rp::string	**name**
rbool	**selected**
rbool	**can_modify**

Friends

class	**rp::msg**

The documentation for this class was generated from the following files:

- rdoprocess_object.h
- rdoprocess_object.cpp

Более детально с иерархией потомков объекта RPObject можно ознакомиться, посмотрев на лист 8 дипломного проекта. Фукнцию сохранения несет в себе метод saveToXML, а функцию загрузки из файла xml – loadFromXML. Конечными потомками класса RPObject, в которых

переопределены данные виртуальные методы являются такие классы как RPShapeCreateMJ, RPShapeProcessMJ, RPShapeDecideMJ, RPShapeResourceMJ и RPShapeTerminateMJ. В них записываются и из них считываются непосредственные атрибуты самих графических блоков, которые необходимы для моделирования, например, максимальное количество транзактов у блока Create. Помимо этого, стоит отметить и класс RPConnector и RPObjectMatrix. В классе RPConnector производится запись параметров соединяющих блоки линий (конекторов), а в RPObjectMatrix – данные о местоположении блоков на рабочей области графического редактора (flowchart'e).

Первый вариант записи в файл заключается в последовательной записи всех объектов графического редактора. Записав информацию о фигуре, сохраняем данные о следующей за ней фигуре, а также соединяющей их линии и т.д.

Второй вариант записи отличается тем, что все фигуры пишутся в начале файла, а соединительные линии после. Это четче структурирует файл, он становится более читабельным, особенно, когда этот файл большой по количеству строк.

Алгоритм двух данных вариантов и результаты проведенного исследования приведены на листе 8 и 9 дипломного проекта.

Используемая модель, которую требуется сохранить:

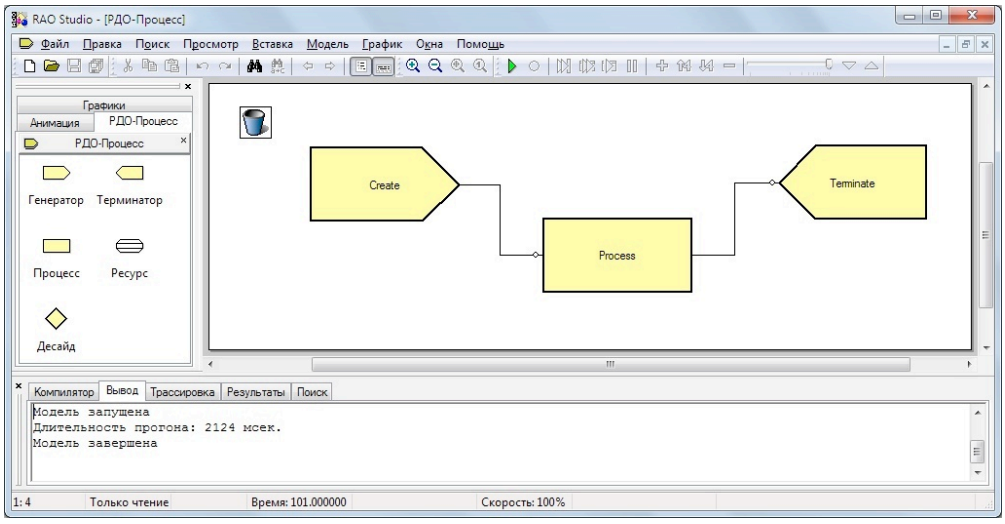

Рис.5.2. Модель для сохранения

Текст файла, полученный первым способом:

```
<?xml version="1.0"?>
<Model>
  <RPProjectMFC name="project">
    <RPObjectFlowChart_MJ name="РДО-Процесс">>
      <RPObjectPixmap />
      <RPShapeCreateMJ name="Create" amount="300" base_gen="1234567890"
exp="1" disp="0" zakon="0">
        <!--BaseClass-->
        <RPObjectMatrix X="112" Y="51.5" scale_X="1.43" scale_Y="1.54"
alpha="0" />
        <RPShape>
          <LOGBRUSH color="#FFFFA0" style="0" />
          <LOGFONT name="MS Shell Dlg" height="-23" color="black"
show="true" />
        </RPShape>
      </RPShapeCreateMJ>
      <RPShapeProcessMJ name="Process" base_gen="1234567890" exp="60"
disp="0" zakon="0" action="0" prior="1" queue="0">
        <!--BaseClass-->
        <RPObjectMatrix X="371.5" Y="111.5" scale_X="1.43" scale_Y="1.54"
alpha="0" />
        <RPShape>
          <LOGBRUSH color="#FFFFA0" style="0" />
          <LOGFONT name="MS Shell Dlg" height="-23" color="black"
show="true" />
        </RPShape>
      </RPShapeProcessMJ>
      <RPConnector obj_from="/project/ РДО-Процесс/Create" obj_to="/project/
РДО-Процесс/Process" index_from="0" index_to="0" />
      <RPShapeTerminateMJ name="Terminate" terminateCounter="5">
        <!--BaseClass-->
        <RPObjectMatrix X="639" Y="168" scale_X="1.43" scale_Y="1.54"
alpha="0" />
        <RPShape>
          <LOGBRUSH color="#FFFFA0" style="0" />
          <LOGFONT name="MS Shell Dlg" height="-23" color="black"
show="true" />
        </RPShape>
      </RPShapeTerminateMJ>
      <RPConnector obj_from="/project/ РДО-Процесс/Process"
obj_to="/project/ РДО-Процесс/Terminate" index_from="1" index_to="0" />
    </RPObjectFlowChart_MJ>
  </RPProjectMFC>1111
</Model>
```

Текст файла, полученный вторым методом:

```
<?xml version="1.0"?>
<Model>
  <RPProjectMFC name="project">
    <RPObjectFlowChart_MJ name="РДО-Процесс">>
      <RPObjectPixmap />
      <RPShapeCreateMJ name="Create" amount="300" base_gen="1234567890"
exp="1" disp="0" zakon="0">
        <!--BaseClass-->
        <RPObjectMatrix X="112" Y="51.5" scale_X="1.43" scale_Y="1.54"
alpha="0" />
        <RPShape>
          <LOGBRUSH color="#FFFFA0" style="0" />
          <LOGFONT name="MS Shell Dlg" height="-23" color="black"
show="true" />
        </RPShape>
```

```
      </RPShapeCreateMJ>
      <RPShapeProcessMJ name="Process" base_gen="1234567890" exp="60"
disp="0" zakon="0" action="0" prior="1" queue="0">
        <!--BaseClass-->
        <RPObjectMatrix X="371.5" Y="111.5" scale_X="1.43" scale_Y="1.54"
alpha="0" />
        <RPShape>
          <LOGBRUSH color="#FFFFA0" style="0" />
          <LOGFONT name="MS Shell Dlg" height="-23" color="black"
show="true" />
        </RPShape>
      </RPShapeProcessMJ>
      <RPShapeTerminateMJ name="Terminate" terminateCounter="5">
        <!--BaseClass-->
        <RPObjectMatrix X="639" Y="168" scale_X="1.43" scale_Y="1.54"
alpha="0" />
        <RPShape>
          <LOGBRUSH color="#FFFFA0" style="0" />
          <LOGFONT name="MS Shell Dlg" height="-23" color="black"
show="true" />
        </RPShape>
      </RPShapeTerminateMJ>
      <RPConnector obj_from="/project/ РДО-Процесс/Create" obj_to="/project/
РДО-Процесс/Process" index_from="0" index_to="0" />
      <RPConnector obj_from="/project/ РДО-Процесс/Process"
obj_to="/project/ РДО-Процесс/Terminate" index_from="1" index_to="0" />
    </RPObjectFlowChart_MJ>
  </RPProjectMFC>1111
</Model>
```

5.3. Выводы по результатам сравнения

Очевидно, что второй способ записи графической информации в файл предоставляет эту информацию в более структурированном виде, фигуры отделены от соединяющих их линий, что при больших объемах записываемой информации влияет на производительность системы вычитывания этих данных и построения по ним графической модели. Также структурированная форма записи во втором варианте способствует более быстрому восприятию информации человеком, читающим ее.

6. ОРГАНИЗАЦИОННО-ЭКОНОМИЧЕСКАЯ ЧАСТЬ

6.1. Расчет затрат на создание подсистемы анимации в РДО

В организационно-экономической части дипломного проекта **«Разработка подсистемы анимации в РДО»** будет произведен расчет затрат на создание подсистемы. Для этого будет произведены следующие расчёты:

- Оценка трудоёмкости разработки и внедрения программного обеспечения;
- Оценка состава и численности разработчиков программного обеспечения;
- Оценка трудоёмкости работ каждого участника проектной группы;
- Оценка стоимости разработки и внедрения системы.

Расчет действителен на 2-ой квартал 2012 года (цены на оборудование, расходные материалы, уровень заработной платы исполнителей и т.д.).

6.1.1. Оценка трудоёмкости разработки программного обеспечения

Перечень стадий и состав работ при разработке системы может быть определён при помощи ГОСТ 34.601-90. При этом следует отметить, что процесс разработки сложных информационных систем характеризуется совместной работой разработчиков постановки задач, разработчиков программного обеспечения и руководителя проекта.

Трудоемкость разработки программного продукта (ПП) зависит от ряда факторов, основными из которых являются следующие:

- степень новизны разрабатываемого программного продукта;

- сложность алгоритма его функционирования;
- объем используемой информации;
- вид представления и способ обработки информации;
- уровень используемого алгоритмического языка программирования.

Рассмотрим эти факторы более подробно и с их учетом порядок и особенности определения трудоемкости разработки ПП.

По **степени новизны** разрабатываемая подсистема анимации РДО относится к ***группе В***. Этот выбор обусловлен тем, что элементы разрабатываемой системы имеют реализованные аналоги.

По **степени сложности** алгоритма функционирования подсистемы анимации РДО относится к ***1 группе сложности***, так как она реализует моделирующие алгоритмы.

По **виду представления исходной информации и способу её контроля** разрабатываемая система относится к ***группе 11***, так как исходная информация хранится в информационной базе в виде документов, имеющих различную структуру, а также к ***группе 22***, т.к. требуется вывод информационных массивов на машинные носители. Таким образом, необходимо осуществлять контроль входной информации представленной в различной форме, учитывая её взаимовлияние.

6.1.2. Расчет трудоёмкости разработки подсистемы анимации РДО

Расчет трудоёмкости разработки системы ведётся по формуле:

$$\tau_{\text{СП}} = \tau_{ТЗ} + \tau_{ЭП} + \tau_{ТП} + \tau_{РП} + \tau_{В},$$

где:

$\tau_{ТЗ}$ -трудоёмкость разработки технического задания;

$\tau_{ЭП}$ -трудоёмкость разработки эскизного проекта системы;

$\tau_{ТП}$ -трудоёмкость разработки технического проекта системы;

$\tau_{РП}$-трудоёмкость разработки рабочего проекта системы;

$\tau_{В}$-трудоёмкость внедрения разработанной системы.

6.1.3. Расчёт трудоёмкости разработки технического задания

$\tau_{ТЗ} = T^{З}_{РЗ} + T^{З}_{РП}$, где

$T^{З}_{РЗ}$ - затраты времени разработчика постановки задач на разработку ТЗ;

$T^{З}_{РП}$ - затраты времени разработчика программного обеспечения на разработку ТЗ.

$T^{З}_{РЗ} = t_{З} \cdot K^{З}_{РЗ}$;

$T^{З}_{РП} = t_{З} \cdot K^{З}_{РП}$, где

$t_{З}$- норма времени на разработку ТЗ на программный продукт в зависимости от функционального назначения и степени новизны разрабатываемого продукта;

$K^{З}_{РЗ}$ - коэффициент, учитывающий удельный вес трудоемкости работ, выполняемых разработчиком постановки задач на стадии ТЗ;

$K^{З}_{РП}$ - коэффициент, учитывающий удельный вес трудоемкости работ, выполняемых разработчиком программного обеспечения на стадии ТЗ.

Для разрабатываемой системы:

$t_{З}$ = 47 чел.-дней, т.к. система относится к группе новизны В и решает задачи расчетного характера ([6], см. табл. 2):

$K^{З}_{РЗ}$ = 0,65 , $K^{З}_{РП}$ = 0,35 , так как разработчик постановки задач работает совместно с разработчиком ПО.

$\tau_{ТЗ}$ =47 * (0,65 + 0,35) = 47 чел.-дней.

6.1.4. Расчёт трудоёмкости разработки эскизного проекта

$\tau_{ЭП} = T^{Э}_{РЗ} + T^{Э}_{РП}$, где

$T^{Э}_{РЗ}$-затраты времени разработчика постановки задач на разработку ЭП;

$T^{Э}_{РП}$-затраты времени разработчика программного обеспечения на разработку ЭП.

$T^{Э}_{РЗ} = t_{Э} \cdot K^{Э}_{РЗ}$;

$T^{Э}_{РП} = t_{Э} \cdot K^{Э}_{РП}$, где

$t_{Э}$ - норма времени на разработку ЭП на программный продукт в зависимости от функционального назначения и степени новизны разрабатываемого продукта;

$K^{Э}_{РЗ}$ - коэффициент, учитывающий удельный вес трудоемкости работ, выполняемых разработчиком постановки задач на стадии ЭП;

$K^{Э}_{РП}$ - коэффициент, учитывающий удельный вес трудоемкости работ, выполняемых разработчиком программного обеспечения на стадии ЭП.

Для разрабатываемой системы:

$t_{Э}$= 67 чел.-дней, $K^{Э}_{РЗ}$= 0,7 , $K^{Э}_{РП}$= 0,3. Критерии выбора нормы и коэффициентов те же, что и для $t_{З}$.([6], см. табл. 3);

$\tau_{ЭП}$ = 67 * (0,7 + 0,3) = 67 чел.-дней.

6.1.5. Расчёт трудоёмкости разработки технического проекта

$\tau_{ТП} = (t^{Т}_{РЗ} + t^{Т}_{РП}) \cdot K_{В} \cdot K_{Р}$, где

$t^{Т}_{РЗ}$, $t^{Т}_{РП}$-нормы времени, затрачиваемого на разработку ТП разработчиком постановки задач и разработчиком программного обеспечения соответственно;

$K_{В}$ - коэффициент учёта вида используемой информации;

$K_{Р}$ - коэффициент учёта режима обработки информации.

$K_{В} = \dfrac{K_{П} n_{П} + K_{НС} n_{НС} + K_{Б} n_{Б}}{n_{Б} + n_{НС} + n_{Б}}$, где

$K_П, K_{НС}, K_Б$ - значения коэффициентов учёта вида используемой информации для переменной, нормативно-справочной информации и баз данных соответственно;

$n_Б, n_{НС}, n_Б$ - количество наборов переменной, нормативно-справочной информации и баз данных соответственно.

Для разрабатываемой системы:

$t^T_{РЗ}$ = 57 чел.-дней, $t^T_{РП}$ = 43 чел.-дней ([6], см. табл. 16), так как система решает задачи расчетного характера.

$K_Р$= 1,26, так как группа новизны - В, а обработка информации происходит в режиме реального времени. ([6], см. табл. 17):

Для группы новизны В:

$K_П$ =1,0, $K_{НС}$ =0,72, $K_Б$ =2,08, $n_Б$ =3, $n_{НС}$ =3, $n_Б$ =10. ([6], см. табл. 18);

$K_В$ = (1,0*3+0,72*3+2,08*10) / (3+3+10) = 25,96 / 16 = 1,62

$\tau_{ТП}$ =(57 + 43)*1,62*1,26 = 204 чел.-дней.

6.1.6 Расчёт трудоёмкости разработки рабочего проекта

$\tau_{РП} = K_К \cdot K_Р \cdot K_Я \cdot K_З \cdot K_{ИА} \cdot (t^P_{РЗ} + t^P_{РП})$, где

$t^P_{РЗ}$, $t^P_{РП}$- норма времени, затраченного на разработку рабочего проекта на алгоритмическом языке высокого уровня разработчиком постановки задач и разработчиком программного обеспечения соответственно;

$K_К$- коэффициент учета сложности контроля информации;

$K_Я$ - коэффициент учета уровня используемого алгоритмического языка программирования;

$K_З$ - коэффициент учета степени использования готовых программных модулей;

$K_{ИА}$ - коэффициент учета вида используемой информации и сложности алгоритма программного продукта.

$K_{ИА} = \frac{K_П' n_П + K_{НС}' n_{НС} + K_Б' n_Б}{n_П + n_{НС} + n_Б}$, где

$K_П', K_{НС}', K_Б'$ - значения коэффициентов учета сложности алгоритма программного продукта и вида используемой информации для переменной, нормативно-справочной информации и баз данных соответственно.

Для разрабатываемой системы:

$K_К$ = 1,07, так как степень сложности контроля входной информации - 11, а степень сложности контроля выходной информации - 22. ([6], см. табл. 19);

$K_Р$ = 1,32, так как группа новизны - В, а режим обработки информации — в реальном масштабе времени. ([6], см. табл. 17);

$K_Я$ = 1,0, так как С++ является алгоритмическим языком высокого уровня. ([6], см. табл. 20);

$K_З$ = 0,8, так как готовые программные модули практически не используются([6], см. табл. 21);

$K_П'$ = 1,2, $K_{НС}'$ = 0,65, $K_Б'$ = 0,54, так как сложность алгоритма - 3 группа и группа новизны - В. ([6], см. табл. 22);

$t_{РЗ}^{Р}$ = 35 чел.-дней, $t_{РП}^{Р}$ = 221 чел.-дней, ([6], см. табл. 35), так как система решает задачи расчетного характера;

$n_Б$ =3, $n_{НС}$ =3, $n_Б$ =10.

Таким образом получим:

$K_{ИА}$ = (1,2*3+0,65*3+0,54*10) / (3+3+10) = 10,95 / 16 = 0,68

$\tau_{РП}$ = 1,07*1,32*1*0,8*0,68*(35 + 221) = 196 чел.-дней.

6.1.7. Расчёт трудоемкости внедрения программного продукта

$\tau_B = (t_{P3}^B + t_{PП}^B) \cdot K_K \cdot K_P \cdot K_З$

$t_{P3}^B, t_{PП}^B$ - норма времени, затрачиваемого разработчиком постановки задач и разработчиком программного обеспечения соответственно на выполнение процедур внедрения программного продукта.

Для разрабатываемой системы:

t_{P3}^B = 8 чел.-дня, $t_{PП}^B$ = 24 чел.-дней, см. расчёт трудоёмкости РП. ([6], см. табл. 48);

τ_B = (8 + 24)*1,07*1,32*0,8 = 36 чел.-дней.

6.1.8. Суммарная трудоемкость

Суммарная трудоёмкость разработки и внедрения подсистемы анимации РДО:

$\tau_{СП} = \tau_{ТЗ} + \tau_{ЭП} + \tau_{ТП} + \tau_{РП} + \tau_B$ = 47 + 67 + 204 + 196 + 36 = 550 чел.-дней.

Табл.6.1. Трудоемкость этапов разработки программного продукта

№ пп	Стадия разработки	Трудоемкость чел.-дни
1	Техническое задание	47
2	Эскизный проект	67
3	Технический проект	204
4	Рабочий проект	196
5	Внедрение	36
	Всего	**550**

6.1.9. Оценка численности разработчиков программного обеспечения

Успешная реализация проекта по созданию систем автоматизации основывается на объективном планировании состава и численности коллектива разработчиков, а также сроков сдачи системы в эксплуатацию. Оценка состава и численности коллектива разработчиков в случае создания подсистемы анимации проводится на основании квалификационных требований к исполнителям каждого этапа проекта, а также наличия свободных трудовых ресурсов у фирмы-разработчика. Срок сдачи системы в эксплуатацию рассчитывается на основании трудоёмкости реализации каждого этапа проекта, состава участников и доли их участия в каждом этапе проекта.

Проведём расчёт срока сдачи системы в эксплуатацию $T_{экспл}$

$$\mathbf{T}_{экспл} = T_{тз} + T_{эп} + T_{тп} + T_{рп} + T_{в}$$

$$\mathbf{T} = \frac{\tau * Kд}{R * Kв},$$

τ - трудоёмкость этапа, чел.-дней;

$Kд$ - коэффициент дополнительных работ (= 1,1);

R - число исполнителей по этапам (см. табл.6.2);

$Kв$ - коэффициент выполнения норм (= 1,15).

Fn - фонд рабочего времени исполнителей этапа;

$Kб$ - коэффициент выполнения норм (= 1,15).

$$\mathbf{T}_{тз} = \frac{\tau * Kд}{R * Kв} = (47*1,1) / (1*1,15) = 45 \text{ дней};$$

$$\mathbf{T}_{эп} = \frac{\tau * Kд}{R * Kв} = (67*1,1) / (1,4*1,15) = 46 \text{ дней};$$

$$\mathbf{T}_{тп} = \frac{\tau * Kд}{R * Kв} = (204*1,1) / (1,5*1,15) = 130 \text{ дней};$$

$$\mathbf{T}_{рп} = \frac{\tau * Kд}{R * Kв} = (196*1,1) / (2,3*1,15) = 82 \text{ дня};$$

$$\mathbf{T}_{в} = \frac{\tau * Kд}{R * Kв} = (36*1,1) / (1,5*1,15) = 23 \text{ дня};$$

$\mathbf{T}_{\text{экспл}} = T_{\text{тз}} + T_{\text{эп}} + T_{\text{тп}} + T_{\text{рп}} + T_{\text{в}} = 45 + 46 + 130 + 82 + 23 = 326$ дней.

Табл.6.2. Количество работников на этапах разработки

№п п	**Стадия**	**Состав разработчиков**	**Число разработчиков**	**Занятость в проекте, %**	**Продолжительность этапа, дни**
1	Техническое задание	Системный аналитик	1	100	45
2	Эскизный проект	Системный аналитик	1	100	46
		Программист	1	40	
3	Технический проект	Системный аналитик	1	50	130
		Программист	1	100	
4	Рабочий проект	Системный аналитик	1	30	82
		Программист	2	100	
5	Внедрение	Системный аналитик	1	100	23
		Программист	1	50	

6.1.10. Оценка трудозатрат разработчиков подсистемы анимации РДО

Определим трудоемкость этапов разработки программного продукта для каждого исполнителя отдельно:

- Системный аналитик

$T_1 = T_{тз} *1 + T_{эп} *1 + T_{тп}*0,5 + T_{рп}*0,3 + T_{в}*1 = 45 + 46 + 65 + 25 + 23 = 204$ чел.-дней

- Программист

$T_2 = T_{эп} *0,4 + T_{тп}*1 + T_{рп}*1 + T_{в}*0,5 = 19 + 130 + 82 + 12 = 243$ чел.-дней

- Программист, подключаемый на этапе рабочего проектирования.

$T_3 = T_{рп} *1 = 82$ чел.-дней

Итог приведен в табл. 6.3.

Таблица 6.3. Трудоемкость разработки программного продукта для каждого исполнителя

№ пп	Исполнитель	Трудоемкость чел.-дни
1	Системный аналитик	204
2	Программист	243
3	Программист (рабочий проект)	82

6.2. Оценка стоимости разработки и внедрения подсистемы анимации РДО

6.2.1. Себестоимость разработки

Себестоимость продукции (работ, услуг) представляет собой стоимостную оценку используемых в процессе производства продукции (работ, услуг) природных ресурсов, сырья, материалов, топлива, энергии, основных фондов, трудовых ресурсов, а также других затрат на ее производство и реализацию.

Затраты, образующие себестоимость продукции (работ, услуг), группируются в соответствии с их экономическим содержанием по следующим элементам:

- материалы,
- специальное оборудование,
- основная заработная плата,
- дополнительная заработанная плата,
- отчисления на социальное страхование,
- производственные командировки,
- затраты на работы выполняемые сторонними организациями и предприятиями,
- накладные расходы,

В нашем случае примем, что расходы на производственные командировки и затраты на работы выполняемые сторонними организациями и предприятиями равны нулю, т.к. сотрудники в командировки не ездят, и разработкой системы занимается только одна организация.

6.2.2. Расчет амортизационных отчислений

Затраты на использование спецоборудование:

$$C_{CO} = \sum_i \frac{Ц_{Б_i} \cdot \alpha_i}{100 \cdot F_Д} \cdot t_i$$

$Ц_i$ -балансовая цена i-го вида оборудования, руб.;

α_i -норма годовых амортизационных отчислений для оборудования i-го вида, %

$F_Д$ -действительный годовой фонд времени, ч.;

$F_Д = F_p - F_p * 0{,}08$, где

F_p - годовой фонд рабочего времени сотрудника, ч.(2092 часа);

0,08 – коэффициент простоя оборудования на ремонт и профилактику, следовательно: $F_Д = 2092 - 2092 * 0{,}08 = 1925$ часов.

t_i -время использования i-го вида оборудования при выполнении данной разработки, ч.

Таблица 6.4. Специальное оборудование и ПО, используемое при разработке

№	**Наименование**	$Ц_i$, **руб.**	a_i **,%**	$F_Д$ **,ч.**	t_i **,чел.-дни.**
1	ПЭВМ и ПО системного аналитика	50 000	20	1925	204
2	ПЭВМ и ПО программиста	90 000	20	1925	325

Итого амортизационные затраты на специальное оборудование:

C_{CO} = 50 000*0,2*204*8/1925 + 90 000*0,2*325*8/1925 = 8480 + 24 300 = 32 780 рублей;

C_{CO} = 32 780 рублей.

6.2.3. Затраты на оплату труда

В данную статью включаются заработная плата всех исполнителей, непосредственно занятых разработкой данного программного продукта с учетом их должностных окладов и времени участия. Расчет проводится по формуле:

Сзо = Зi/d*ti, где

Зi - среднемесячный оклад i-го исполнителя, руб.

d - среднее количество рабочих дней в месяце, d = 21,8 дней.

Тi - трудоемкость работ, выполняемых i-м исполнителем, чел.дни.- определяются из табл. 6.2.

Расчет затраты на оплату труда каждого исполнителя.

- Системный аналитик

  Сзо $_2$ = 60 000 / 21*204 = 582 860 руб.

- Программист

  Сзо $_3$ = 40 000 / 21*243 = 462 860 руб.

- Программист (рабочий проект)

  Сзо $_3$ = 25 000 / 21*82 = 97 620 руб.

Общая заработная плата равна:

Сзо = 1 143 340 руб.

В данной статье также учитываются выплаты непосредственным исполнителям за время, не проработанное на производстве, в том числе: оплата очередных отпусков, компенсация за недоиспользованный отпуск, оплата льготных часов подросткам и др.

6.2.4. Отчисления в социальные фонды и на страхования от несчастного случая

В статье отражены обязательные отчисления по установленным законодательством нормам органам государственного социального страхования, Пенсионного фонда, государственного фонда занятости и медицинского страхования от затрат на оплату труда работников, включаемых в себестоимость продукции (работ, услуг) по элементу “Затраты на оплату труда” (кроме тех видов оплаты, на которые страховые взносы не начисляются).

Отчисления на единый социальный налог включают в себя отчисления собственно на социальное страхование, а также - на медицинское страхование, в фонд занятости и в пенсионный фонд, итого 31% от основной заработной платы.

Отчисления в социальные фонды и страхование от несчастного случая составляют:

Ссс = Сзо*31/100 = 1 143 340 * 0,31 = **354 435 руб.**

6.2.5. Накладные расходы

В данную статью входят другие затраты, входящие в состав себестоимости продукции, но не относящиеся к ранее перечисленным элементам затрат. Сюда можно отнести такие виды затрат как: ежемесячная арендная плата за помещение, оплата электроэнергии, оплата отопления, оплата телефона и других услуг связи и т.д.

Накладные расходы для высокотехнологичной компании-разработчика составляют 200% от суммы общей заработной платы:

Сн = Ан*Сзо, где

Ан - коэффициент накладных расходов Ан = 2

Сн = 2 * 354 435 = **708 870 руб.**

Результаты расчетов затрат на разработку программного продукта приведены в табл. 6.5.

Таблица 6.5. Смета затрат на разработку системы

№	Наименование статьи	Сметная себестоимость, руб.
1	Специальное оборудование	32 780
2	Заработанная плата	1 143 340
3	Отчисления в фонды	354 435
4	Накладные расходы	708 870
	Итого	2 239 425

6.2.6. Цена разработки и внедрения системы информационной поддержки

Так как ПП рассматривается и создается как продукция производственно-технического назначения, допускающая многократное тиражирование и отчуждение от непосредственных разработчиков, то ее цена определяется по формуле:

$$Ц = K \cdot C + П_P ,$$

где

С – затраты на разработку программной продукции (сметная себестоимость);

К – коэффициент учета затрат на изготовление опытного образца ПП как продукции производственно-технического назначения (К = 1.1…1.2);

$П_р$ – нормативная прибыль, рассчитываемая по формуле

$П_Р = C * \rho_Н / 100,$

Где, $\rho_Н$ – норматив рентабельности, $\rho_Н$ =30%;

$П_р = 2\ 239\ 425 * 0{,}3 = 671\ 830$ руб.

Ц = 1.1 * 2 239 425 + 671 830 = 3 135 200 руб.

6.3. Заключение организационно-экономической части

Произведенные расчеты показали следующее:

- Длительность разработки и внедрения подсистемы анимации РДО составляет 326 дней;
- В разработке участвуют следующие исполнители:
  - Системный аналитик;
  - Программист
  - Программист (рабочий проект)
- Суммарная трудоемкость создания программного продукта составляет 550 чел. – дня;
- Затраты на создание программного продукта составляют 2 239 425 рублей
- Цена программного продукта составляет 3 135 200 рублей.

7. ТРЕБОВАНИЯ БЕЗОПАСНОСТИ ПРИ РАЗРАБОТКЕ ПОДСИСТЕМЫ АНИМАЦИИ РДО

7.1. Введение

В данном разделе дипломного проекта рассматриваются и анализируются опасные и вредные факторы при использовании программного продукта «Подсистема анимации РДО» в рамках комплекса лабораторных занятий. Описываются мероприятия по обеспечению безопасности и безвредных условий труда, приводятся рекомендации по способам и методам ограничения действия вредных и исключению воздействия опасных факторов на студентов и преподавательский состав, проходящих и проводящих занятия в компьютерной сфере. При этом на человека, работающего в зоне действия ПЭВМ, влияет большое количество вредных факторов, состояние которых необходимо контролировать. Факторы, действующие на студентов и преподавательский состав [7]:

- уровень шума, источниками которого являются вентиляционные устройства ПЭВМ, устройства ввода-вывода, агрегаты кондиционирования и вентилирования воздуха, другие электрические приборы;
- уровни электростатического и электромагнитного излучения, источниками которого являются видеотерминалы;
- параметры микроклимата;
- освещенность в помещении и на рабочем месте.

Таблица 7.1. Опасные и вредные факторы

Тип	Опасные	Вредные
Физические	1. Поражение электротоком 2. Пожарная опасность	1. Повышенный уровень шума агрегата кондиционирования и вентилирования воздуха 2. Плохой микроклимат 3. Недостаточная освещенность рабочей зоны
Психофизиологические		4. Напряжение зрения 5. Напряжение внимания 6. Длительные статические нагрузки

Кроме того очевидна необходимость проведения мероприятий не только для улучшения состояния рабочего места по перечисленным вредным факторам, но и для удовлетворения требованиям эргономики, пожаро- и электробезопасности .

Ниже приведен анализ соответствия требованиям, определенным в СанПиН 2.2.2/2.4.1340-03 [9], к рабочим местам одной из лабораторных аудиторий МГТУ имени Н.Э. Баумана. Анализ проводился без использования специального измерительного оборудования, выполнен на основе наблюдений, качественного определения соответствия требованиям и соответствующих предположений. Выводы по результатам анализа носят рекомендательный характер.

7.2. Требований безопасности при отработке программного продукта

7.2.1. Требования к персональным электронно-вычислительным машинам

ПЭВМ должны соответствовать требованиям настоящих санитарных правил, и каждый их тип подлежит санитарно-эпидемиологической экспертизе с оценкой в испытательных лабораториях, аккредитованных в установленном порядке. Рассматриваются следующие требования.

Перечень продукции и контролируемых гигиенических параметров вредных и опасных факторов представлены.

Допустимые уровни звукового давления и уровней звука, создаваемых ПЭВМ.

Временные допустимые уровни электромагнитных полей (ЭМП), создаваемых ПЭВМ.

Допустимые визуальные параметры устройств отображения информации.

Концентрации вредных веществ, выделяемых ПЭВМ в воздух помещений, не должны превышать предельно допустимых концентраций (ПДК), установленных для атмосферного воздуха.

Мощность экспозиционной дозы мягкого рентгеновского излучения в любой точке на расстоянии 0,05 м от экрана и корпуса ВДТ (на электронно-лучевой трубке) при любых положениях регулировочных устройств не должна превышать 1 мкЗв/час (100 мкР/час).

Конструкция ПЭВМ должна обеспечивать возможность поворота корпуса в горизонтальной и вертикальной плоскости с фиксацией в заданном положении для обеспечения фронтального наблюдения экрана ВДТ. Дизайн ПЭВМ должен предусматривать окраску корпуса в спокойные мягкие тона с диффузным рассеиванием света. Корпус ПЭВМ, клавиатура и другие блоки и устройства ПЭВМ должны иметь матовую поверхность с коэффициентом отражения 0,4-0,6 и не иметь блестящих деталей, способных создавать блики.

Конструкция ВДТ должна предусматривать регулирование яркости и контрастности,

Документация на проектирование, изготовление и эксплуатацию ПЭВМ не должна противоречить требованиям настоящих Санитарных правил.

Результаты анализа:

- используется жидкокристаллическое средство отображения, которое при разрешении 1024x768 обеспечивает частоту обновления 75 Гц
- Конструкция ВДТ, дизайн соответствуют современным требованиям.
- Поворот корпуса ВДТ в горизонтальной и вертикальной плоскости соответствует нормам. Корпуса ВДТ и ПЭВМ окрашены в мягкие тона и имеют матовую поверхность с коэффициентом отражения, не превышающим норму, монитор имеет антибликовое и антистатическое покрытие, а также снабжен ручками регулировки яркости и контраста.
- Монитор не оборудован защитным фильтром на экране, т.к. в этом нет острой необходимости. Монитор соответствует стандарту MPR II, имеет низкую радиационную активность, уменьшает влияние электрических и магнитных полей на человека. В монитор также встроена система управления

энергопотребления, которая позволяет снижать потребляемую мощность компьютера во время простоя.

- Конструкция клавиатуры выполнена в виде отдельного устройства с возможностью свободного перемещения, имеет опорные устройства, позволяющие менять угол наклона клавиатуры. Основные параметры клавиатуры удовлетворяют нормативам.

7.2.2. Требования к помещению для работы с ПЭВМ

Эксплуатация ПЭВМ в помещениях без естественного освещения допускается только при наличии расчетов, обосновывающих соответствие нормам естественного освещения и безопасность их деятельности для здоровья работающих (пункт в редакции, введенной в действие с 1 июля 2007 года Изменением N 1 от 25 апреля 2007 года, - см. предыдущую редакцию).

Естественное и искусственное освещение должно соответствовать требованиям действующей нормативной документации. Окна в помещениях, где эксплуатируется вычислительная техника, преимущественно должны быть ориентированы на север и северо-восток.

Оконные проемы должны быть оборудованы регулируемыми устройствами типа: жалюзи, занавесей, внешних козырьков и др.

Площадь на одно рабочее место пользователей ПЭВМ с ВДТ на базе плоских дискретных экранов (жидкокристаллические, плазменные) должна составлять не менее 4,5 $м^2$.

Для внутренней отделки интерьера помещений, где расположены ПЭВМ, должны использоваться диффузно отражающие материалы с коэффициентом отражения для потолка - 0,7-0,8; для стен - 0,5-0,6; для пола - 0,3-0,5.

Полимерные материалы используются для внутренней отделки интерьера помещений с ПЭВМ при наличии санитарно-эпидемиологического заключения.

Помещения, где размещаются рабочие места с ПЭВМ, должны быть оборудованы защитным заземлением (занулением) в соответствии с техническими требованиями по эксплуатации.

Не следует размещать рабочие места с ПЭВМ вблизи силовых кабелей и вводов, высоковольтных трансформаторов, технологического оборудования, создающего помехи в работе ПЭВМ.

Результаты анализа:

- Помещение (см**.** рис.7.1) имеет искусственное комбинированное и естественное освещение, окна ориентированны на север. Искусственное освещение поддерживается 9х4 люминесцентными лампами ЛД-18. Площадь комнаты (8х6 метров) равна 48 $м^2$. Высота потолка 3,6 м. Число рабочих мест - 10. Площадь на одно рабочее место равна 4,8 $м^2$, что соответствует санитарным нормам: не менее 4,5 $м^2$ площади.
- Окна оснащены жалюзи.
- Комната имеет встроенные шкафы и полки для хранения одежды и вещей. Она так же оборудована системой центрального отопления и кондиционирования.
- Половое покрытие - линолеум.
- Помещение имеет один вход и не имеет смежных с ним помещений. Расположено на втором этаже здания.
- Помещение не граничит с помещениями с высоким уровнем шума и вибрации.
- Помещение оборудовано центральным защитным заземлением.

Стены помещения обклеены обоями бежевого тона.

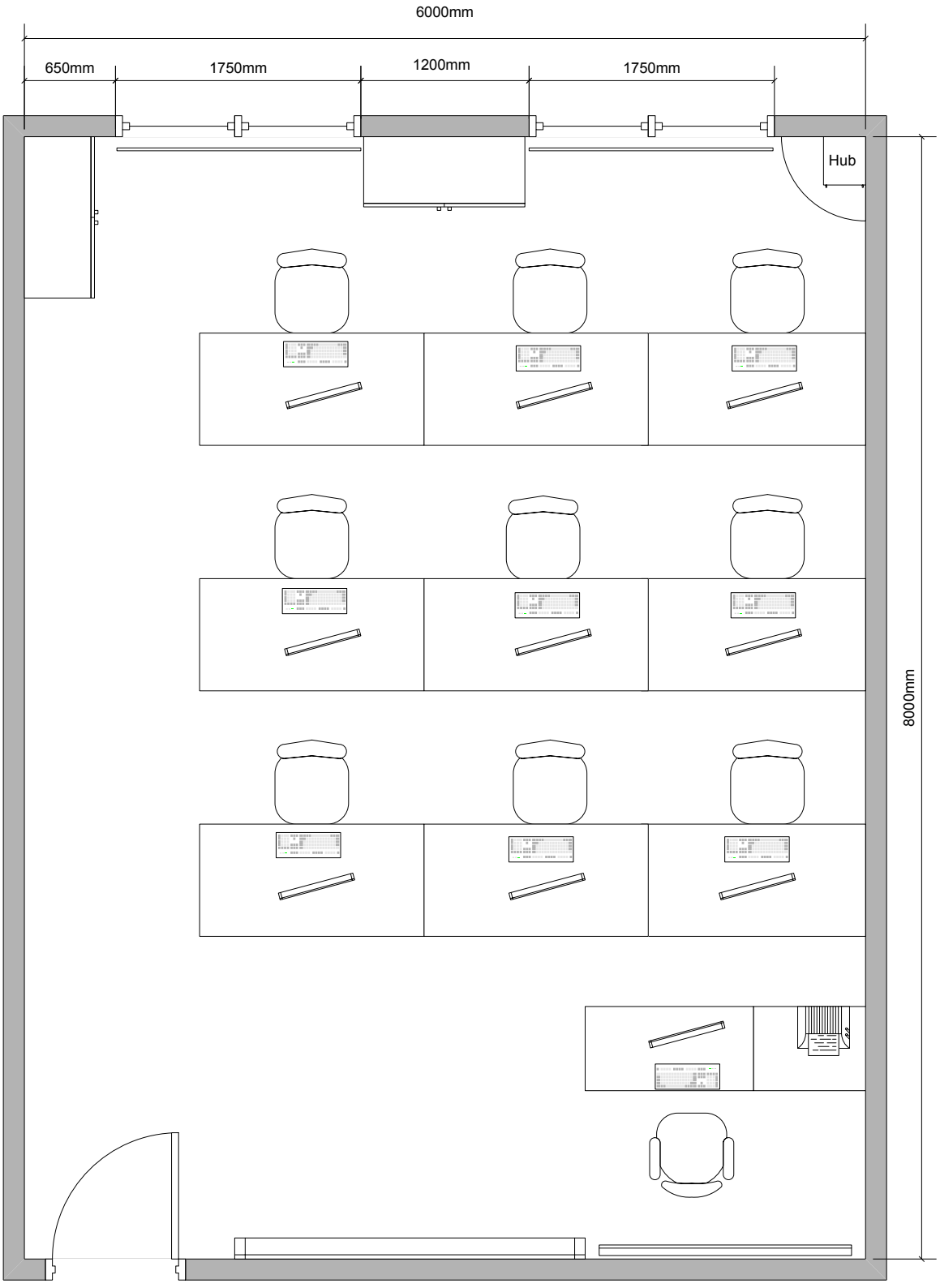

Рис.7.1. Планировка помещения

7.2.3. Требования к микроклимату, содержанию аэроионов и вредных химических веществ в воздухе на рабочих местах, оборудованных ПЭВМ

В помещениях всех типов образовательных учреждений, где расположены ПЭВМ, должны обеспечиваться оптимальные параметры микроклимата.

В помещениях, оборудованных ПЭВМ, проводится ежедневная влажная уборка и систематическое проветривание после каждого часа работы на ЭВМ.

Уровни положительных и отрицательных аэроионов в воздухе помещений, где расположены ПЭВМ, должны соответствовать действующим санитарно-эпидемиологическим нормативам.

Содержание вредных химических веществ в воздухе помещений, предназначенных для использования ПЭВМ во всех типах образовательных учреждений, не должно превышать предельно допустимых среднесуточных концентраций для атмосферного воздуха в соответствии с действующими санитарно-эпидемиологическими нормативами (ГН 2.1.6.1338-03).

Результаты анализа:

- В теплое время года в помещении температура воздуха составляет от 20 до 23 С. Подвижность воздуха – от 0 до 0.1 м/с. Влажность воздуха –55-60%.
- В холодное время года в помещении температура воздуха составляет 19-21 ºС, подвижность воздуха менее 0.1 м/с, влажность воздуха – 55-60%, что соответствует нормам.
- Содержание вредных химических веществ в воздухе соответствует содержанию этих веществ на улице, т.к. проводится периодическое проветривание помещений, однако, фильтры кондиционера предотвращают попадание пыли с улицы в помещение. Поскольку в Москве (особенно в центральной части города) уличный воздух не является

чистым, то воздух в нашем помещении не соответствует нормам.

- Содержание аэронов в воздухе помещения близко к оптимуму и равно соответственно: положительных – 1000-1200, а отрицательных – 1700-2000.

В помещении присутствует система центрального кондиционирования. Работы за ПЭВМ, выполняемые в аудитории, не влекут за собой выделения пыли, влаги, загрязняющих веществ. Запыленность низкая. Ежедневно проводятся влажные уборки.

7.2.4. Требования к уровням шума и вибрации на рабочих местах, оборудованных ПЭВМ

В помещениях всех образовательных учреждений, где расположены ПЭВМ, уровни шума не должны превышать допустимых значений, установленных для жилых и общественных зданий.

Согласно СН 2.2.4/2.1.8.562-96 «Шум на рабочих местах, в помещениях жилых, общественных зданий и на территории жилой застройки» для работы, связанной с обучением, уровень шума не должен превышать значения, указанного в приложении 3 (таблица 5):

Снизить уровень проникающего шума в помещениях с ВДТ и ПЭВМ при открытых окнах, где проходят занятия, возможно использованием звукопоглощающих материалов с максимальными коэффициентами звукопоглощения в области частот 63 - 8000 Гц (Допустимые уровни звука проникающего шума указаны в приложении 3 (таблица 5), согласно СН 2.2.4/2.1.8.562-96). Дополнительным звукопоглощением служат занавеси из плотной ткани. Ширина занавеси должна быть в 2 раза больше ширины окна.

В помещениях всех типов образовательных учреждений, в которых эксплуатируются ПЭВМ, уровень вибрации не должен превышать допустимых значений для жилых и общественных зданий в соответствии с действующими санитарно-эпидемиологическими нормативами ("Производственная вибрация, вибрация в помещениях жилых и общественных зданий" СН 2.2.4/2.1.8.566-96).

Для снижения вибрации в помещениях оборудование, аппараты и приборы необходимо устанавливать на амортизирующие прокладки. Также могут быть использованы средства индивидуальной защиты.

Шумящее оборудование (печатающие устройства, серверы и т.п.), уровни шума которого превышают нормативные, должно размещаться вне помещений с ПЭВМ.

Результаты анализа:

- Уровень шума в аудитории может превышать 50 дБА, т.к. ведутся лекционные занятия и проводится разъяснительная работа со студентами. На потолке установлено звукопоглощающее покрытие. Дополнительно, в качестве средства звукопоглощения в помещении, можно отметить, матерчатые жалюзи на окнах, однако их ширина не превосходит ширину окна в два раза;
- Вибрация на рабочих местах отсутствует. Выявлен источник вибрации в помещении - система кондиционирования. Типовой расчет виброизоляции для системы кондиционирования приведен в пункте 7.3.
- Уровень вибрации в помещении от оборудования не значителен (мало ощутим), поэтому амортизирующие прокладки или средства индивидуальной защиты не используются;
- Печатающие устройства – лазерные принтеры практически бесшумны и размещены в помещении.

7.2.5. Требования к освещению на рабочих местах, оборудованных ПЭВМ

Рабочие столы следует размещать таким образом, чтобы видеодисплейные терминалы были ориентированы боковой стороной к световым проемам, чтобы естественный свет падал преимущественно слева.

Искусственное освещение в помещениях для эксплуатации ПЭВМ должно осуществляться системой общего равномерного освещения. В производственных и административно-общественных помещениях, в случаях преимущественной работы с документами, следует применять системы комбинированного освещения (к общему освещению дополнительно устанавливаются светильники местного освещения, предназначенные для освещения зоны расположения документов).

Освещенность на поверхности стола в зоне размещения рабочего документа должна быть 300-500 лк. Освещение не должно создавать бликов на поверхности экрана. Освещенность поверхности экрана не должна быть более 300 лк.

Следует ограничивать прямую блесткость от источников освещения, при этом яркость светящихся поверхностей (окна, светильники и др.), находящихся в поле зрения, должна быть не более 200 кд/м^2.

Следует ограничивать отраженную блесткость на рабочих поверхностях (экран, стол, клавиатура и др.) за счет правильного выбора типов светильников и расположения рабочих мест по отношению к источникам естественного и искусственного освещения, при этом яркость бликов на экране ПЭВМ не должна превышать 40 кд/м^2 и яркость потолка не должна превышать 200 кд/м^2.

Показатель ослепленности для источников общего искусственного освещения в производственных помещениях должен быть не более 20. Показатель дискомфорта в административно-общественных помещениях не более 40, в учебных помещениях не более 15.

Яркость светильников общего освещения в зоне углов излучения от 50 до 90° с вертикалью в продольной и поперечной плоскостях должна составлять не более 200 кд/м^2, защитный угол светильников должен быть не менее 40°.

Светильники местного освещения должны иметь непросвечивающий отражатель с защитным углом не менее 40°.

Следует ограничивать неравномерность распределения яркости в поле зрения пользователя ПЭВМ, при этом соотношение яркости между рабочими поверхностями не должно превышать 3:1-5:1, а между рабочими поверхностями и поверхностями стен и оборудования 10:1.

В качестве источников света при искусственном освещении следует применять преимущественно люминесцентные лампы типа ЛБ и компактные люминесцентные лампы (КЛЛ). При устройстве отраженного освещения в производственных и административно-общественных помещениях допускается применение металлогалогенных ламп. В светильниках местного освещения допускается применение ламп накаливания, в том числе галогенных.

Для освещения помещений с ПЭВМ следует применять светильники с зеркальными параболическими решетками, укомплектованными электронными пуско-регулирующими аппаратами (ЭПРА). Допускается использование многоламповых светильников с ЭПРА, состоящими из равного числа опережающих и отстающих ветвей.

Применение светильников без рассеивателей и экранирующих решеток не допускается.

При отсутствии светильников с ЭПРА лампы многоламповых светильников или рядом расположенные светильники общего освещения следует включать на разные фазы трехфазной сети.

Общее освещение при использовании люминесцентных светильников следует выполнять в виде сплошных или прерывистых линий светильников, расположенных сбоку от рабочих мест, параллельно линии зрения пользователя при рядном расположении видеодисплейных

терминалов. При периметральном расположении компьютеров линии светильников должны располагаться локализованно над рабочим столом ближе к его переднему краю, обращенному к оператору.

Коэффициент запаса (Кз) для осветительных установок общего освещения должен приниматься равным 1,4.

Коэффициент пульсации не должен превышать 5%.

Для обеспечения нормируемых значений освещенности в помещениях для использования ПЭВМ следует проводить чистку стекол оконных рам и светильников не реже двух раз в год и проводить своевременную замену перегоревших ламп.

Результаты анализа:

- Требования к естественному освещению не нарушены. Расположение рабочего места по отношению к световым проемам является допустимым: мониторы ориентированны боковой стороной к световым проемам.
- Бликов от освещения на поверхности мониторов не наблюдается.
- Светильники местного освещения отсутствуют.
- Искусственное освещение поддерживается 9х4 люминесцентными лампами ЛД-18, которые расположены на потолке помещения. Светильники оснащены зеркальными параболическими решетками.
- Общее освещение выполнено в виде 3 прерывистых линий. Столы, по возможности, ориентированы так, чтобы свет подал на край, обращенный к оператору.
- Производится своевременная замена перегоревших ламп в помещении.
- В помещении не реже двух раз в год проводится чистка стекол оконных рам и светильников.

7.2.6. Требования к уровням электромагнитных полей на рабочих местах, оборудованных ПЭВМ

Временные допустимые уровни ЭМП, создаваемых ПЭВМ на рабочих местах пользователей, а также в помещениях образовательных учреждений.

Методика проведения инструментального контроля уровней ЭМП на рабочих местах пользователей ПЭВМ.

Результаты анализа:

- Поскольку рабочие места внедряемой системы (места в лаборатории) оборудованы жидкокристаллическими мониторами, то установка поглощающих или отражающих экранов не требуется.

7.2.7. Требования к визуальным параметрам ВДТ, контролируемым на рабочих местах

Предельно допустимые значения визуальных параметров ВДТ, контролируемые на рабочих местах.

Результаты анализа:

- Поскольку рабочие места внедряемой системы (места в лаборатории) оборудованы жидкокристаллическими мониторами с настраиваемыми визуальными параметрами, то все соответствуют указанным нормам.

7.2.8. Общие требования к организации рабочих мест пользователей ПЭВМ

При размещении рабочих мест с ПЭВМ расстояние между рабочими столами с видеомониторами (в направлении тыла поверхности одного видеомонитора и экрана другого видеомонитора), должно быть не менее 2,0 м, а расстояние между боковыми поверхностями видеомониторов - не менее 1,2 м.

Рабочие места с ПЭВМ в помещениях с источниками вредных производственных факторов должны размещаться в изолированных кабинах с организованным воздухообменом.

Рабочие места с ПЭВМ при выполнении творческой работы, требующей значительного умственного напряжения или высокой концентрации внимания, рекомендуется изолировать друг от друга перегородками высотой 1,5-2,0 м.

Экран видеомонитора должен находиться от глаз пользователя на расстоянии 600-700 мм, но не ближе 500 мм с учетом размеров алфавитно-цифровых знаков и символов.

Конструкция рабочего стола должна обеспечивать оптимальное размещение на рабочей поверхности используемого оборудования с учетом его количества и конструктивных особенностей, характера выполняемой работы. При этом допускается использование рабочих столов различных конструкций, отвечающих современным требованиям эргономики. Поверхность рабочего стола должна иметь коэффициент отражения 0,5-0,7.

Конструкция рабочего стула (кресла) должна обеспечивать поддержание рациональной рабочей позы при работе на ПЭВМ, позволять изменять позу с целью снижения статического напряжения мышц шейно-плечевой области и спины для предупреждения развития утомления. Тип рабочего стула (кресла) следует выбирать с учетом роста пользователя, характера и продолжительности работы с ПЭВМ.

Рабочий стул (кресло) должен быть подъемно-поворотным, регулируемым по высоте и углам наклона сиденья и спинки, а также расстоянию спинки от переднего края сиденья, при этом регулировка каждого параметра должна быть независимой, легко осуществляемой и иметь надежную фиксацию.

Поверхность сиденья, спинки и других элементов стула (кресла) должна быть полумягкой, с нескользящим, слабо электризующимся и

воздухопроницаемым покрытием, обеспечивающим легкую очистку от загрязнений.

Результаты анализа:

- расстояние между рабочими столами с видеомониторами (в направлении тыла поверхности одного видеомонитора и экрана другого видеомонитора), составляет 1,6м. Расстояние между боковыми поверхностями видеомониторов - равно 1,4 м, что в неполной мере соответствует нормам.
- Экран видеомонитора находиться от глаз пользователя на расстоянии 600 - 700 мм
- При работе с ПЭВМ используется вариант офисного стула, конструкция которого не предполагает наличие поворотных и наклонных регулируемых механизмов фиксации положения тела оператора, что не соответствует указанным нормам.
- Поверхность сиденья, спинки и других элементов кресла является полумягкой.

7.2.9. Требования к организации и оборудованию рабочих мест с ПЭВМ для обучающихся в общеобразовательных учреждениях и учреждениях начального и высшего профессионального образования

Помещения для занятий оборудуются одноместными столами, предназначенными для работы с ПЭВМ.

Конструкция одноместного стола для работы с ПЭВМ должна предусматривать:

- две раздельные поверхности: одна горизонтальная для размещения ПЭВМ с плавной регулировкой по высоте в пределах 520-760 мм и вторая - для клавиатуры с плавной регулировкой по высоте и углу наклона от 0 до 15° с надежной фиксацией в оптимальном рабочем положении (12-15°);

- ширину поверхностей для ВДТ и клавиатуры не менее 750 мм (ширина обеих поверхностей должна быть одинаковой) и глубину не менее 550 мм;

- опору поверхностей для ПЭВМ или ВДТ и для клавиатуры на стояк, в котором должны находиться провода электропитания и кабель локальной сети. Основание стояка следует совмещать с подставкой для ног;

- отсутствие ящиков;

- увеличение ширины поверхностей до 1200 мм при оснащении рабочего места принтером.

Высота края стола, обращенного к работающему с ПЭВМ, и высота пространства для ног должны соответствовать росту обучающихся в обуви.

При наличии высокого стола и стула, не соответствующего росту обучающихся, следует использовать регулируемую по высоте подставку для ног.

Линия взора должна быть перпендикулярна центру экрана и оптимальное ее отклонение от перпендикуляра, проходящего через центр экрана в вертикальной плоскости, не должно превышать ±5°, допустимое ±10°.

Рабочее место с ПЭВМ оборудуют стулом, основные размеры которого должны соответствовать росту обучающихся в обуви.

Результаты анализа:

- Рабочий стол представляет собой парту со следующими параметрами:
    - высота рабочей поверхности стола составляет 800 мм, пространство для ног высотой 780 мм, шириной - 1400 мм, глубиной на уровне колен – 800 мм и на уровне вытянутых ног - 850 мм.
    - Разделение поверхностей отсутствует.
    - Ящики отсутствуют.

- Стояк для электропитания отсутствует.

Что не соответствует нормам.

- Конструкция рабочего кресла предусматривает:
  - ширину 400 мм и глубину 400 мм поверхности сиденья
  - высота сиденья составляет 450 мм.
  - поверхность сиденья с закругленным передним краем
  - высота нижнего края спинки над сиденьем – 170 мм.
  - высоту опорной поверхности спинки 200 мм, ширину - 400 мм и радиус кривизны горизонтальной плоскости - 300 мм
  - угол наклона сиденья – 1-5 градусов.
  - угол наклона спинки в вертикальной плоскости составляет 5-10 градусов

  Что, в целом, соответствует нормам.
- Рабочее место пользователя ПЭВМ не оборудовано подставкой для ног.
- Клавиатура расположена на поверхности стола на расстоянии 100-200 мм от края, обращенного к пользователю.

7.2.10. Требования к электроснабжению, электробезопасности на рабочих местах, оборудованных ПЭВМ

Классификация помещений по опасности поражения электрическим током:

Помещения I класса. Без повышенной опасности.

- нет условий повышенной и особой опасности.

Помещения II класса. Помещения повышенной опасности поражения электрическим током.

- повышенная температура воздуха (t >= + 35 C);
- повышенная влажность (> 75 %);
- наличие токопроводящей пыли;

- наличие токопроводящих полов (металлические, земляные, железобетонные, кирпичные);
- наличие электрических установок (заземленных) — возможности прикосновения одновременно и к электрической установке и к заземлению или к двум электрическим установкам одновременно.

Помещения III класса. Особо опасные помещения.

- 100 % влажность;
- наличие активной среды;
- наличие одновременно двух и более условий повышенной опасности.

Наше помещение относится к помещению I класса опасности и требует заземления или зануления.

Питание устройств ПЭВМ осуществляется от однофазной сети переменного тока с частотой 50 Гц, и напряжением 220 В.

В целях обеспечения необходимой электробезопасности при проведении работ в помещениях с ПЭВМ, необходимо выполнять следующие требования:

- для обеспечения работы операторов ПЭВМ необходимо исключить возможность случайного соприкосновения людей с токонесущими частями оборудования. Это достигается путем изоляции токоведущих частей ЭВМ и приборов и размещения их в недоступных зонах;
- не оставлять ЭВМ и другое оборудование под напряжением без наблюдения.

Результаты анализа:

- Имеется автоматический выключатель;
- Все электрооборудование имеет рабочую изоляцию;
- К электрооборудованию допускаются лица не моложе 18 лет и прошедшие инструктаж по техники безопасности;
- Регулярно проводится технический осмотр оборудования.

- ПЭВМ подключены к розеткам, имеющим заземление.

7.2.11. Пожаробезопасность

Помещение (аудитория), где функционирует разработанная подсистема, относится к категории В — пожароопасные помещения, в котором имеются твердые сгораемые вещества, способные только гореть, но не взрываться при контакте с кислородом воздуха. Наиболее вероятной причиной пожара является неисправность электрооборудования и электросетей. При эксплуатации ЭВМ возможны возникновения следующих аварийных ситуаций: короткие замыкания, перегрузки, повышение переходных сопротивлений в электрических контактах, перенапряжение, возникновение токов утечки. При возникновении аварийных ситуаций происходит резкое выделение тепловой энергии, которая может явиться причиной возникновения пожара. Требования к пожаробезопасности зданий и сооружений определяются согласно СНиП 21.01-97.

Для снижения вероятности возникновения пожара необходимо проводить различные профилактические мероприятия:

- Организационные – правильная эксплуатация электрооборудования, правильное содержание зданий и помещений;
- Технические — соблюдение противопожарных правил и норм, норм при проектировании зданий, при устройстве отопления, вентиляции освещения, правильное размещение оборудования;

Для снижения вероятности возникновения и распространения пожара на ранней стадии необходимо:

- установить пожарную сигнализацию с системой оповещения работников, дежурного по объекту и, желательно, автоматическое оповещение противопожарных служб;

- иметь в наличии несколько ручных углекислотных огнетушителей (например, огнетушитель марки ОУ-3).

Результаты анализа:

- В помещении предусмотрена противопожарная сигнализация и индивидуальные средства пожаротушения: ОУ-3.

7.3. Типовой расчет виброизоляции для системы кондиционирования

В ходе обследования помещения, где происходит эксплуатация рассматриваемой системы, было обнаружено оборудование, производящее вибрации – кондиционер с частотой вращения вентилятора 720 оборотов в минуту. Проведем расчет жесткости виброизоляторов и их количества.

Оптимальное соотношение частоты вращения вентилятора в кондиционере и его собственной частоты определяется формулой (согласно ГОСТ 12.4.093-80): $\frac{f}{f_0} = 3$

Частота вращения вентилятора, рассчитывается по формуле: $f = \frac{n}{60} = \frac{720}{60} = 12 Гц$

Таким образом, собственная частота: $f_0 = \frac{f}{3} = \frac{12}{3} = 4 Гц$

С другой стороны, собственная частота рассчитывается по формуле: $f_0 = \frac{1}{2 \cdot \pi} \cdot \sqrt{\frac{q_1 \cdot N}{m}}$, где

q_1 - вертикальная жесткость виброизолятора, Н/см

N - число виброизоляторов, штук

m - масса виброизолятора, кг

Выразим и рассчитаем соотношение $\frac{q_1 \cdot N}{m}$:

$$\frac{q_1 \cdot N}{m} = (2 \cdot \pi \cdot f_0)^2 = (2 \cdot 3.1415 \cdot 4)^2 = 631.66$$

Выбираем виброизолятор ДО-38:

Таблица 7.2. Параметры виброизолятора ДО-38

Обозначение	Нагрузка Р, Н		Вертикальная жесткость, Н/см	Высота в свободном состоянии	Осадка пружины под нагрузкой, мм		Число рабочих витков	Масса, кг
	Рабочая (Рраб.)	Предельная (Рпр.)			Рраб.	Рпр.		
ДО38	122	152	45	72	27	33,7	5,6	0,3

Для 4 опор-виброизоляторов, согласно таблице 7.2, выражение $\frac{q_1 \cdot N}{m} = \frac{45 \cdot 4}{0,3} = 600$, собственная частота $f_0 = \frac{1}{2 \cdot \pi} \cdot \sqrt{\frac{q_1 \cdot N}{m}} = \frac{1}{2 \cdot 3.14} \cdot \sqrt{600} = 3.89 Гц$

Таким образом, соотношение частоты вращения вентилятора в кондиционере и его собственной частоты будет равно:

$$\frac{f}{f_0} = \frac{12}{3.89} = 3.08$$

Это соотношение оптимально.

Таким образом, для устранения в помещении вибраций, создаваемых кондиционером, необходимо установить его на 4 опоры-виброизоляторы марки ДО-38.

7.4. Утилизация носителей информации

Утилизация носителей информации осуществляется в соответствии с ГОСТ Р ИСО/МЭК 17799-2005. Национальный стандарт Российской Федерации. Информационная технология. Практические правила управления информационной безопасностью (утвержден и введен в действие Приказом Федерального агентства по техническому регулированию и метрологии от 29 декабря 2005 г. N 447-ст, настоящий стандарт идентичен международному стандарту ИСО/МЭК 17799:2000 “Информационная технология. Практические правила управления информационной безопасностью” (ISO/IEC 17799:2000 “Information technology. Code of practice for security management”).

Диски до дробления:

Рис.7.2. Диски до утилизации

и после:

Рис.7.3. Диски после дробления

Извлечение: пункт 8.6.2 ГОСТ: «Утилизация носителей информации»

«Носители информации по окончании использования следует надежно и безопасно утилизировать. Важная информация может попасть в руки посторонних лиц из-за небрежной утилизации носителей данных. Чтобы свести к минимуму такой риск, должны быть установлены формализованные процедуры безопасной утилизации носителей информации. Для этого необходимо предусматривать следующие мероприятия:

а) носители, содержащие важную информацию, следует хранить и утилизировать надежно и безопасно (например, посредством сжигания/измельчения). Если носители планируется использовать в пределах организации для других задач, то информация на них должна быть уничтожена;

б) ниже приведен перечень объектов, в отношении которых может потребоваться безопасная утилизация:

бумажные документы;

речевые или другие записи;

копировальная бумага;

выводимые отчеты;

одноразовые ленты для принтеров;

магнитные ленты;

сменные диски или кассеты;

оптические носители данных (все разновидности, в том числе носители, содержащие программное обеспечение, поставляемое производителями);

тексты программ;

тестовые данные;

системная документация;

в) может оказаться проще принимать меры безопасной утилизации в отношении всех носителей информации, чем пытаться сортировать носители по степени важности;

г) многие организации предлагают услуги по сбору и утилизации бумаги, оборудования и носителей информации. Следует тщательно выбирать подходящего подрядчика с учетом имеющегося у него опыта и обеспечения необходимого уровня информационной безопасности;

д) по возможности следует регистрировать утилизацию важных объектов с целью последующего аудита.

ЗАКЛЮЧЕНИЕ

В рамках данного дипломного проекта были получены следующие результаты.

Проведено предпроектное исследование системы имитационного моделирования РДО и сформулированы предпосылки создания подсистемы анимации графического редактора. Был произведен сравнительный анализ существующих решений в этой области и анализ системы анимации текстового редактора.

На этапе концептуального проектирования подсистемы с помощью диаграмм прецедентов, дерева целей и функциональной структуры РДО были выявлены основные задачи и цели разрабатываемой подсистемы, а также место, которое должна занять эта подсистема в системе RDO-studio. Диаграмма компонентов сделала ясным физическое устройство РДО и выделала те компоненты, на которых должна быть построена подсистема анимации.

На этапе технического проектирования разработан прямой и обратный каналы связи между приложениями графического редактора и имитатора системы РДО, позволяющий передавать параметры системы для дальнейшего их отображения в виде анимации. С помощью диаграммы классов показана структура принятого решения по внедрению подсистемы анимации. Разработана анимация статистических данных имитационной модели.

На этапе рабочего проектирования написан программный код для реализации спроектированных раннее алгоритма вывода на экран статистических данных и их анимирования. Проведены отладка и тестирование новой подсистемы. Работоспособность подсистемы была проверена на модели простейшей системы массового обслуживания. Автономность разработанного решения проверена с помощью сравнения

выходных файлов работы системы РДО без режима анимации, на примере текстовой модели системы массового обслуживания, а затем с помощью той же модели, написанной в графическом редакторе.

Таким образом, поставленная цель дипломного проекта достигнута в полном объеме.

СПИСОК ЛИТЕРАТУРЫ

1. Емельянов В.В., Ясиновский С.И. "Введение в интелектуальное имитационное моделирование сложных дискретных систем и процессов. Язык РДО". М.: МГТУ, 2009. – 584с.

2. Аверилл М. Лоу, В. Дэвид Кельтон. Имитационное моделирование. М.: Питер, 2004. – 848с.

3. Бьерн Страуструп. Язык программирования С++. Специальное издание. Пер. с англ. – М.:Издательство Бином, 2011. – 1136с.

4. Герберт Шилдт. Справочник программистов по С/С++, 3-е изд.:Пер с англ. – М.:ООО ”И.Д.Вильямс”, 2006. – 432с.

5. Фаулер М. UML. Основы, 3-е изд.:Пер. с англ. – Спб.:Символ-Плюс, 2004. – 192с.

6. Арсеньев В.В., Сажин Ю.Б. Методические указания к выполнению организационно-экономической части дипломных проектов по созданию программной продукции. М.: МГТУ, 1994. – 52с.

7. Белов С.В., Ильницкая А.В. Безопасность жизнедеятельности. М.: Высш.шк., 1999. – 448с.

8. ГОСТ. Комплекс стандартов и руководящих документов на автоматизированные системы. (ГОСТ 34.201-90, ГОСТ 34.201-90, РД 50-34.698-90 и др.), - М.: Госстандарт, 1991 г.

9. СанПин 2.2.2/2.4.1340-03 «Гигиенические требования к персональным электронно-вычислительным машинам и организации работы».

10. СанПиН 2.2.4.548-96 «Гигиенические требования к микроклимату производственных помещений».

11. СанПиН 2.2.2.542-96 «Санитарно-гигиенические нормы допустимых уровней ионизации воздуха».

12. СН 2.2.4/2.1.8.562-96 «Шум на рабочих местах, в помещениях жилых, общественных зданий и на территории жилой застройки».

13. СН 2.2.4/2.1.8.566-96 "«Производственная вибрация, вибрация в помещениях жилых и общественных зданий».

14. Технологии создания приложений в модельной среде. Источник: http://www.gpss.ru/paper/giniyatullin/index_w.html. 2010

15. Современные технологии имитационного моделирования и их применение в информационных бизнес-системах. Источник: //nit.miem.edu.ru/2006/sb/setions0/9.htm. 2006

ПРИЛОЖЕНИЕ 1. Листинг файлов графических примитивов

Часть файла rdoprocess_shape_create.cpp

```
#include "app/rdo_studio_mfc/rdo_process/proc2rdo/stdafx.h"
#include
"app/rdo_studio_mfc/rdo_process/proc2rdo/rdoprocess_shape_create.h"
#include
"app/rdo_studio_mfc/rdo_process/proc2rdo/rdoprocess_shape_create_dlg1.h"
#include
"app/rdo_studio_mfc/rdo_process/proc2rdo/rdoprocess_method_proc2rdo.h"
#include "app/rdo_studio_mfc/rdo_process/rp_method/rdoprocess_shape.h"
#include "app/rdo_studio_mfc/src/application.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

RPShapeCreateMJ::RPShapeCreateMJ(PTR(RPObject) _parent)
      : RPShape_MJ(_parent, _T("Create"))
      , m_currentTransactCount(0)
{
      pa_src.push_back( rp::point(-50, -25) );
      pa_src.push_back( rp::point(25, -25) );
      pa_src.push_back( rp::point(50, 0) );
      pa_src.push_back( rp::point(25, 25) );
      pa_src.push_back( rp::point(-50, 25) );
      pa_src.push_back( rp::point(-50, -25) );

      docks.push_back( new RPConnectorDockOne( this, RPConnectorDock::out,
rp::point(  50, 0 ), 0, "transact" ) );

      // инициализация параметров для генерирования
      gname=_T("Create"); // имя
      gfirst=0; // время первого
      gamount=100; // кол-во создаваемых
      gtype=0; // закон прибытия
      base_gen=1234567890;
```

```
	//атрибуты законов
	gexp=0;
	gdisp=0;
	gmax=0;

	//второе окно
	inf=1000000; // бесконечноть
	gID=1; // ID группы
	gpar1=0;
	gpar2=0;
	gpar3=0;

	indent = 5;
}

RPShapeCreateMJ::~RPShapeCreateMJ()
{
}

rp::RPXMLNode* RPShapeCreateMJ::save( rp::RPXMLNode* parent_node )
{
	rp::RPXMLNode* obj_node = RPShape_MJ::save( parent_node );
	return obj_node;
}

void RPShapeCreateMJ::saveToXML(REF(pugi::xml_node) parentNode) const
{
	// Записываем узел <RPShapeCreateMJ/>:
	pugi::xml_node node =
parentNode.append_child(getClassName().c_str());
	// Сохраняем атрибуты объекта:
	// 1) Атрибуты графики
	RPObjectMatrix::saveToXML(node);
	RPShape::	saveToXML(node);
	// 2) Атрибуты симулятора
	node.append_attribute(_T("name"))	.set_value(getName().c_str());
	node.append_attribute(_T("amount"))	.set_value(gamount		);
	node.append_attribute(_T("base_gen"))	.set_value(base_gen		);
	node.append_attribute(_T("exp"))	.set_value(gexp			);
	node.append_attribute(_T("disp"))	.set_value(gdisp		);
	node.append_attribute(_T("zakon"))	.set_value(gtype		);
}

void RPShapeCreateMJ::loadFromXML(CREF(pugi::xml_node) node)
```

```
{
	// Считываем атрибуты для загрузки сохраненного блока "Create":
	for (pugi::xml_attribute attr = node.first_attribute(); attr; attr =
attr.next_attribute())
	{
		// Присваиваем сохраненные в xml-файле параметры
		// Для симулятора (диалоговые окна)
		tstring attrName = attr.name();
		if (attrName == _T("name"))
		{
			setName(attr.value());
		}
		else if (attrName == _T("amount"))
		{
			gamount = attr.as_int();
		}
		else if (attrName == _T("base_gen"))
		{
			base_gen = attr.as_int();
		}
		else if (attrName == _T("exp"))
		{
			gexp = attr.as_double();
		}
		else if (attrName == _T("disp"))
		{
			gdisp = attr.as_double();
		}
		else if (attrName == _T("zakon"))
		{
			gtype = attr.as_int();
		}
	}
	// Отображения объекта на Flowchart'e
	RPObjectMatrix::loadFromXML(node);
	RPShape::	loadFromXML(node);
}

RPObject* RPShapeCreateMJ::newObject( RPObject* parent )
{
	return new RPShapeCreateMJ( parent );
}
```

```
void RPShapeCreateMJ::onLButtonDblClk( UINT nFlags, CPoint
global_chart_pos )
{
	UNUSED(nFlags          );
	UNUSED(global_chart_pos);

	RPShapeCreateDlg1_MJ dlg( AfxGetMainWnd(), this );
	dlg.DoModal();
}

void RPShapeCreateMJ::generate()
{
	rdo::compiler::gui::RPShapeDataBlock::zakonRaspr zakon;
	switch(gtype)
	{
		case 0: // константа
			zakon = rdo::compiler::gui::RPShapeDataBlock::Const;
			break;
		case 1: // нормальный
			zakon = rdo::compiler::gui::RPShapeDataBlock::Normal;
			break;
		case 2: // равномерный закон
			zakon = rdo::compiler::gui::RPShapeDataBlock::Uniform;
			break;
		case 3: // треугольный
			zakon = rdo::compiler::gui::RPShapeDataBlock::Triangular;
			break;
		case 4: // экспоненциальный
			zakon = rdo::compiler::gui::RPShapeDataBlock::Exp;
			break;
	}

	LPRPShapeCreateMJ pThis(this);
	ASSERT(pThis);

	pInternalStatistics =
pThis.interface_cast<rdo::runtime::IInternalStatistics>();
	ASSERT(pInternalStatistics);

	m_pParams =
rdo::Factory<rdo::compiler::gui::RPShapeDataBlockCreate>::create(zakon,
gname);
	m_pParams->setBase(base_gen);
	m_pParams->setAmount(gamount);
```

```
	m_pParams->setDisp(gdisp);
	m_pParams->setExp(gexp);
	m_pParams->setMax(gmax);
	m_pParams->setStatistics(pInternalStatistics);

	studioApp.m_pStudioGUI->sendMessage(kernel->simulator(),
RDOThread::RT_PROCGUI_BLOCK_CREATE, m_pParams.get());

	m_pParams = NULL;
}

void RPShapeCreateMJ::setTransCount(ruint count)
{
	m_currentTransactCount = count;
	update();
}

void RPShapeCreateMJ::drawCustom(REF(CDC) dc)
{
	dc.SetTextColor(RGB(0x00, 0x64, 0x00));
	dc.TextOut((3*(this->pa_global.getMaxX()) + (this-
>pa_global.getMinX()))/4 - indent, this->pa_global.getMaxY() + indent,
rp::string::fromint(m_currentTransactCount).c_str());
}
```

Часть файла rdoprocess_shape_process.cpp

```
#include "app/rdo_studio_mfc/rdo_process/proc2rdo/stdafx.h"
#include <list>
#include
"app/rdo_studio_mfc/rdo_process/proc2rdo/rdoprocess_shape_process.h"
#include
"app/rdo_studio_mfc/rdo_process/proc2rdo/rdoprocess_shape_process_dlg1.h"
#include
"app/rdo_studio_mfc/rdo_process/proc2rdo/rdoprocess_method_proc2rdo.h"
#include "app/rdo_studio_mfc/src/application.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif
```

```
RPShapeProcessMJ::RPShapeProcessMJ( RPObject* _parent )
	: RPShape_MJ( _parent, _T("Process") )
	, m_currentTransactCountProc(0)
{

	gname = _T("Process"); // имя

	gtype    = 0;
	base_gen = 1234567;

	//атрибуты законов
	gexp  = 0;
	gdisp = 0;
	gmax  = 0;
	gmin  = 0;

	action = 0; // тип действия по отношению к ресурсу
	prior  = 0;
	queue  = 0;
	parameter = 888; // ПОКА НЕ ТРОГАЛ ЭТОТ ПАРАМЕТР
	indent = 5;

		// инициализация типа блока
	type ="block";

	pa_src.push_back( rp::point(-50, -25) );
	pa_src.push_back( rp::point( 50, -25) );
	pa_src.push_back( rp::point( 50,  25) );
	pa_src.push_back( rp::point(-50,  25) );
	pa_src.push_back( rp::point(-50, -25) );

	docks.push_back( new RPConnectorDock   (this, RPConnectorDock::in,
rp::point(-50,  0), 180, "transact"));
	docks.push_back( new RPConnectorDockOne(this, RPConnectorDock::out,
rp::point( 50,  0),   0, "transact"));
	docks.push_back( new RPConnectorDock   (this, RPConnectorDock::in,
rp::point(  0, 25), 270, "resource"));
}

RPShapeProcessMJ::~RPShapeProcessMJ()
{
}

RPObject* RPShapeProcessMJ::newObject( RPObject* parent )
```

```
{
	return new RPShapeProcessMJ( parent );
}

void RPShapeProcessMJ::onLButtonDblClk( UINT nFlags, CPoint
global_chart_pos )
{
	UNUSED(nFlags          );
	UNUSED(global_chart_pos);

	RPShapeProcessDlg1_MJ dlg( AfxGetMainWnd(), this );
	dlg.DoModal();
}

void RPShapeProcessMJ::generate()
{
	rdo::compiler::gui::RPShapeDataBlock::zakonRaspr zakon;
	switch(gtype)
	{
		case 0: // константа
			zakon = rdo::compiler::gui::RPShapeDataBlock::Const;
			break;
		case 1: // нормальный
			zakon = rdo::compiler::gui::RPShapeDataBlock::Normal;
			break;
		case 2: // равномерный закон
			zakon = rdo::compiler::gui::RPShapeDataBlock::Uniform;
			break;
		case 3: // треугольный
			zakon = rdo::compiler::gui::RPShapeDataBlock::Triangular;
			break;
		case 4: // экспоненциальный
			zakon = rdo::compiler::gui::RPShapeDataBlock::Exp;
			break;
	}

	LPRPShapeProcessMJ pThis(this);
	ASSERT(pThis);

	pInternalStatistics =
pThis.interface_cast<rdo::runtime::IInternalStatistics>();
	ASSERT(pInternalStatistics);
```

```
	m_pParams =
rdo::Factory<rdo::compiler::gui::RPShapeDataBlockProcess>::create(zakon,
gname);
	ASSERT(m_pParams);
	m_pParams->setBase(base_gen);
	m_pParams->setDisp(gdisp);
	m_pParams->setExp(gexp);
	m_pParams->setMax(gmax);
	m_pParams->setStatistics(pInternalStatistics);

	switch(action)
	{
		case 0://advance
			m_pParams-
>addAction(rdo::compiler::gui::RPShapeDataBlockProcess::A_ADVANCE);
			break;
		case 1://sieze,advance,release
			m_pParams-
>addAction(rdo::compiler::gui::RPShapeDataBlockProcess::A_SEIZE  );
			m_pParams-
>addAction(rdo::compiler::gui::RPShapeDataBlockProcess::A_ADVANCE);
			m_pParams-
>addAction(rdo::compiler::gui::RPShapeDataBlockProcess::A_RELEASE);
			break;
		case 2://seize,advance
			m_pParams-
>addAction(rdo::compiler::gui::RPShapeDataBlockProcess::A_SEIZE  );
			m_pParams-
>addAction(rdo::compiler::gui::RPShapeDataBlockProcess::A_ADVANCE);
			break;
		case 3://advance,release
			m_pParams-
>addAction(rdo::compiler::gui::RPShapeDataBlockProcess::A_ADVANCE);
			m_pParams-
>addAction(rdo::compiler::gui::RPShapeDataBlockProcess::A_RELEASE);
			break;
	}

	std::list<CString>::iterator it = list_resource_procMJ.begin();
	while( it != list_resource_procMJ.end() )
	{
		m_pParams->addRes(static_cast<tstring>(*it));
		it++;
	}
```

```
	studioApp.m_pStudioGUI->sendMessage(kernel->simulator(),
RDOThread::RT_PROCGUI_BLOCK_PROCESS, m_pParams.get());

	m_pParams = NULL;
}

void RPShapeProcessMJ::setTransCount(ruint count)
{
	m_currentTransactCountProc = count;
	update();
}

void RPShapeProcessMJ::drawCustom(REF(CDC) dc)
{
	dc.SetTextColor(RGB(0x00, 0x64, 0x00));
	dc.TextOut(this->pa_global.getMaxX() - 2*indent, this-
>pa_global.getMaxY() + indent,
rp::string::fromint(m_currentTransactCountProc).c_str());
}

void RPShapeProcessMJ::saveToXML(REF(pugi::xml_node) parentNode) const
{
	// Записываем узел <RPShapeProcessMJ/>:
	pugi::xml_node node =
parentNode.append_child(getClassName().c_str());
	// Сохраняем атрибуты объекта:
	// 1) Атрибуты графики
	RPObjectMatrix::saveToXML(node);
	RPShape::	saveToXML(node);
	// 2) Атрибуты симулятора
	node.append_attribute(_T("name"))
.set_value(getName().c_str());
	node.append_attribute(_T("base_gen"))	.set_value(base_gen
);
	node.append_attribute(_T("exp"))	.set_value(gexp
);
	node.append_attribute(_T("disp"))	.set_value(gdisp
);
	node.append_attribute(_T("zakon"))	.set_value(gtype
);
	node.append_attribute(_T("action"))	.set_value(action
);
```

```
	node.append_attribute(_T("prior"))	.set_value(prior
);
	node.append_attribute(_T("queue"))	.set_value(queue
);
}

void RPShapeProcessMJ::loadFromXML(CREF(pugi::xml_node) node)
{
	// Считываем атрибуты для загрузки сохраненного блока "Process":
	for(pugi::xml_attribute attr = node.first_attribute(); attr; attr =
attr.next_attribute())
	{
		// Присваиваем сохраненные в xml-файле параметры:
		// Для симулятора (диалоговые окна)
		tstring attrName = attr.name();
		if (attrName == _T("name"))
		{
			setName(attr.value());
		}
		else if (attrName == _T("base_gen"))
		{
			base_gen = attr.as_int();
		}
		else if (attrName == _T("exp"))
		{
			gexp = attr.as_double();
		}
		else if (attrName == _T("disp"))
		{
			gdisp = attr.as_double();
		}
		else if (attrName == _T("zakon"))
		{
			gtype = attr.as_int();
		}
		else if (attrName == _T("action"))
		{
			action = attr.as_int();
		}
		else if (attrName == _T("prior"))
		{
			prior = attr.as_int();
		}
		else if (attrName == _T("queue"))
```

```
            {
                  queue = attr.as_int();
            }
      }
      // Для отображения объекта на Flowchart'e
      RPObjectMatrix::loadFromXML(node);
      RPShape::       loadFromXML(node);
}
```

Часть файла rdoprocess_shape_terminate.cpp

```
#include "app/rdo_studio_mfc/rdo_process/proc2rdo/stdafx.h"
#include
"app/rdo_studio_mfc/rdo_process/proc2rdo/rdoprocess_shape_terminate.h"
#include
"app/rdo_studio_mfc/rdo_process/proc2rdo/rdoprocess_shape_terminate_dlg1.h
"
#include
"app/rdo_studio_mfc/rdo_process/proc2rdo/rdoprocess_method_proc2rdo.h"
#include "app/rdo_studio_mfc/src/application.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

RPShapeTerminateMJ::RPShapeTerminateMJ( RPObject* _parent )
      : RPShape_MJ(_parent, _T("Terminate"))
      , m_currentTransactCountDel(0)
{
      m_term_inc = 1;
      m_name     = _T("Terminate");

      // инициализация типа блока
      type = "block";

      pa_src.push_back( rp::point(-50,   0) );
      pa_src.push_back( rp::point(-25, -25) );
      pa_src.push_back( rp::point( 50, -25) );
      pa_src.push_back( rp::point( 50,  25) );
      pa_src.push_back( rp::point(-25,  25) );
      pa_src.push_back( rp::point(-50,   0) );
```

```
	docks.push_back( new RPConnectorDock( this, RPConnectorDock::in,
rp::point( -50, 0 ), 180, "transact" ) );

	indent = 5;
}

RPShapeTerminateMJ::~RPShapeTerminateMJ()
{
}

RPObject* RPShapeTerminateMJ::newObject( RPObject* parent )
{
	return new RPShapeTerminateMJ( parent );
}

void RPShapeTerminateMJ::onLButtonDblClk( UINT nFlags, CPoint
global_chart_pos )
{
	UNUSED(nFlags          );
	UNUSED(global_chart_pos);

	RPShapeTerminateDlg1_MJ dlg( AfxGetMainWnd(), this );
	dlg.DoModal();
}

void RPShapeTerminateMJ::generate()
{
	LPRPShapeTerminateMJ pThis(this);
	ASSERT(pThis);

	pInternalStatistics =
pThis.interface_cast<rdo::runtime::IInternalStatistics>();
	ASSERT(pInternalStatistics);

	m_pParams =
rdo::Factory<rdo::compiler::gui::RPShapeDataBlockTerminate>::create(m_name
);
	m_pParams->setTermInc   (m_term_inc         );
	m_pParams->setStatistics(pInternalStatistics);

	studioApp.m_pStudioGUI->sendMessage(kernel->simulator(),
RDOThread::RT_PROCGUI_BLOCK_TERMINATE, m_pParams.get());
```

```
	m_pParams = NULL;
}

void RPShapeTerminateMJ::saveToXML(REF(pugi::xml_node) parentNode) const
{
	// Записываем узел <RShapeTerminateMJ/>:
	pugi::xml_node node =
parentNode.append_child(getClassName().c_str());
	// Сохраняем атрибуты объекта:
	// 1) Атрибуты графики
	RPObjectMatrix::saveToXML(node);
	RPShape::	saveToXML(node);
	// 2) Атрибуты симулятора
	node.append_attribute(_T("name"))
.set_value(getName().c_str());
	node.append_attribute(_T("terminateCounter"))
.set_value(m_term_inc	);
}

void RPShapeTerminateMJ::loadFromXML(CREF(pugi::xml_node) node)
{
	// Считываем атрибуты для загрузки сохраненного блока "Terminate":
	for (pugi::xml_attribute attr = node.first_attribute(); attr; attr =
attr.next_attribute())
	{
		// Присваиваем сохраненные в xml-файле параметры:
		// Для симулятора (диалоговые окна)
		tstring attrName = attr.name();
		if (attrName == _T("name"))
		{
			setName(attr.value());
		}
		else if (attrName == _T("terminateCounter"))
		{
			m_term_inc = attr.as_int();
		}
	}
	// Отображения объекта на Flowchart'e
	RPObjectMatrix::loadFromXML(node);
	RPShape::	loadFromXML(node);
}

void RPShapeTerminateMJ::setTransCount(ruint count)
{
```

```
	m_currentTransactCountDel = count;
	update();
}

void RPShapeTerminateMJ::drawCustom(REF(CDC) dc)
{
	dc.SetTextColor(RGB(0x00, 0x64, 0x00));
	dc.TextOut(this->pa_global.getMaxX() - 2*indent, this-
>pa_global.getMaxY() + indent,
rp::string::fromint(m_currentTransactCountDel).c_str());
}
```

Часть файла rdoprocess_shape_decide.cpp

```
#include "app/rdo_studio_mfc/rdo_process/proc2rdo/stdafx.h"
#include
"app/rdo_studio_mfc/rdo_process/proc2rdo/rdoprocess_shape_decide.h"
#include
"app/rdo_studio_mfc/rdo_process/proc2rdo/rdoprocess_shape_decide_dlg1.h"
#include
"app/rdo_studio_mfc/rdo_process/proc2rdo/rdoprocess_method_proc2rdo.h"
#include "app/rdo_studio_mfc/rdo_process/rp_method/rdoprocess_shape.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

class RPConnectorDockTrue: public RPConnectorDock
{
public:
	RPConnectorDockTrue( RPShape* _parent, Type _type, const rp::point&
_point, double _norm, const rp::string& type = "" ): RPConnectorDock(
_parent, _type, _point, _norm, type ) {};
	virtual ~RPConnectorDockTrue() {};

	virtual rbool can_connect( RPConnectorDock* dock = NULL ) const {
		if ( !RPConnectorDock::can_connect( dock ) ) return false;
		return connectors.empty();
	}
};
class RPConnectorDockFalse: public RPConnectorDock
```

```
{
public:
	RPConnectorDockFalse( RPShape* _parent, Type _type, const rp::point&
_point, double _norm, const rp::string& type = "" ): RPConnectorDock(
_parent, _type, _point, _norm, type ) {};
	virtual ~RPConnectorDockFalse() {};

	virtual rbool can_connect( RPConnectorDock* dock = NULL ) const {
		if ( !RPConnectorDock::can_connect( dock ) ) return false;
		return connectors.empty();
	}
};
RPShapeDecide::RPShapeDecide( RPObject* _parent ):
	RPShape_MJ( _parent, _T("Decide") )
{

	pa_src.push_back(rp::point(-55, 0));
	pa_src.push_back(rp::point(0, -55));
	pa_src.push_back( rp::point(55, 0) );
	pa_src.push_back( rp::point(0, 55) );
	pa_src.push_back( rp::point(-55, 0) );

	docks.push_back( new RPConnectorDock( this, RPConnectorDock::in,
rp::point(  -55, 0 ), 180, "transact" ) );
	docks.push_back( new RPConnectorDockOne( this, RPConnectorDock::out,
rp::point( 55, 0 ), 0, "transact" ) );
	docks.push_back( new RPConnectorDockOne( this, RPConnectorDock::out,
rp::point(  0, 55 ), 270, "transact" ) );
// инициализация параметров для генерирования
	 ptrue  = 0.9;
	 pfalse = 0.1;
}

RPShapeDecide::~RPShapeDecide()
{
}

rp::RPXMLNode* RPShapeDecide::save( rp::RPXMLNode* parent_node )
{
	rp::RPXMLNode* obj_node = RPShape_MJ::save( parent_node );
	return obj_node;
}

void RPShapeDecide::saveToXML(REF(pugi::xml_node) parentNode) const
```

```
{
	// Записываем узел <RPShapeDecide/>:
	pugi::xml_node node =
parentNode.append_child(getClassName().c_str());
	// Сохраняем атрибуты объекта:
	// 1) Атрибуты графики
	RPObjectMatrix::saveToXML(node);
	RPShape::	saveToXML(node);
	// 2) Атрибуты симулятора
	node.append_attribute(_T("name"))	.set_value(getName().c_str());
	node.append_attribute(_T("true"))	.set_value(ptrue		);
	node.append_attribute(_T("false"))	.set_value(pfalse		);
}

void RPShapeDecide::loadFromXML(CREF(pugi::xml_node) node)
{
	// Считываем атрибуты для загрузки сохраненного блока "Decide":
	for(pugi::xml_attribute attr = node.first_attribute(); attr; attr =
attr.next_attribute())
	{
		// Присваиваем сохраненные в xml-файле параметры:
		// Для симулятора (диалоговые окна)
		tstring attrName = attr.name();
		if (attrName == _T("name"))
		{
			setName(attr.value());
		}
		else if (attrName == _T("true"))
		{
			ptrue = attr.as_double();
		}
		else if (attrName == _T("false"))
		{
			pfalse = attr.as_double();
		}
	}
	// Отображения объекта на Flowchart'e
	RPObjectMatrix::loadFromXML(node);
	RPShape::	loadFromXML(node);
}

RPObject* RPShapeDecide::newObject( RPObject* parent )
{
	return new RPShapeDecide( parent );
```

```
}

void RPShapeDecide::onLButtonDblClk( UINT nFlags, CPoint global_chart_pos
)
{
	UNUSED(nFlags          );
	UNUSED(global_chart_pos);

	RPShapeDecideDlg1 dlg( AfxGetMainWnd(), this );
	dlg.DoModal();
}
```

Часть файла rdoprocess_shape_resource.cpp

```
#include "app/rdo_studio_mfc/rdo_process/proc2rdo/stdafx.h"
#include
"app/rdo_studio_mfc/rdo_process/proc2rdo/rdoprocess_shape_resource.h"
#include
"app/rdo_studio_mfc/rdo_process/proc2rdo/rdoprocess_shape_resource_DLG1.h"
#include
"app/rdo_studio_mfc/rdo_process/proc2rdo/rdoprocess_method_proc2rdo.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

// -------------------- RPConnectorDockResourceOut
class RPConnectorDockResourceOut: public RPConnectorDock
{
public:
	RPConnectorDockResourceOut( RPShape* _parent, Type _type, const
rp::point& _point, double _norm, const rp::string& type = "" ):
RPConnectorDock( _parent, _type, _point, _norm, type ) {};
	virtual ~RPConnectorDockResourceOut() {};

	virtual RPConnector* make_connector( RPObject* _parent ) {
		RPConnector* conn = RPConnectorDock::make_connector( _parent );
		conn->setPen( CPen(PS_DOT, 1, RGB(0x00, 0x00, 0x80)) );
		return conn;
	}
};
```

```
void RPShapeResource_MJ::saveToXML(REF(pugi::xml_node) parentNode) const
{
	// Записываем узел <RPShapeResource_MJ/>:
	pugi::xml_node node =
parentNode.append_child(getClassName().c_str());
	// Сохраняем атрибуты объекта:
	// 1) Атрибуты графики
	RPObjectMatrix::saveToXML(node);
	RPShape::	saveToXML(node);
	// 2) Атрибуты симулятора
	node.append_attribute(_T("name"))	.set_value(getName().c_str());
	node.append_attribute(_T("amount"))	.set_value(gamount		);
}

void RPShapeResource_MJ::loadFromXML(CREF(pugi::xml_node) node)
{
	// Считываем атрибуты для загрузки сохраненного блока "Resource":
	for(pugi::xml_attribute attr = node.first_attribute(); attr; attr =
attr.next_attribute())
	{
		// Присваиваем сохраненные в xml-файле параметры:
		// Для симулятора (диалоговые окна)
		tstring attrName = attr.name();
		if (attrName == _T("name"))
		{
			setName(attr.value());
		}
		else if (attrName == _T("amount"))
		{
			gamount = attr.as_int();
		}
	}
	// Отображения объекта на Flowchart'е
	RPObjectMatrix::loadFromXML(node);
	RPShape::	loadFromXML(node);
}
```

Часть файла rdoprocess_connector.cpp

```
#include "app/rdo_studio_mfc/rdo_process/rp_method/stdafx.h"
#include "app/rdo_studio_mfc/rdo_process/rp_method/rdoprocess_connector.h"
```

```
#include
"app/rdo_studio_mfc/rdo_process/rp_method/rdoprocess_object_flowchart.h"
#include "app/rdo_studio_mfc/rdo_process/rp_method/rdoprocess_shape.h"
#include "app/rdo_studio_mfc/rdo_process/rp_misc/rdoprocess_xml.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

// -------------------- RPConnector
RPConnector::RPConnector( RPObject* _parent, const rp::string& _name ):
      RPObjectChart( _parent, _name ),
      dock_begin( NULL ),
      dock_end( NULL ),
      recursive( 0 )
{
      main_pen_width = 1;
      main_pen_default.DeleteObject();
      main_pen_default.CreatePen( PS_SOLID, 1, RGB(0x00, 0x00, 0x00) );
      can_update = false;
      setPen( main_pen_default );
      can_update = true;
}

RPConnector::~RPConnector()
{
      if ( dock_begin ) dock_begin->connectors.remove( this );
      if ( dock_end )   dock_end->connectors.remove( this );
}

void RPConnector::load( rp::RPXMLNode* node )
{
      RPObjectChart::load( node );
      RPObject* obj_from = rpMethod::project->findObject( node-
>getAttribute( "obj_from" ) );
      RPObject* obj_to   = rpMethod::project->findObject( node-
>getAttribute( "obj_to" ) );
      if ( obj_from && obj_to && obj_from->getClassInfo()->isKindOf(
"RPShape" ) && obj_to->getClassInfo()->isKindOf( "RPShape" ) ) {
            dock_begin = static_cast<RPShape*>(obj_from)->getDock( node-
>getAttributeInt( "dock_index_from" ) );
```

```
            dock_end   = static_cast<RPShape*>(obj_to)->getDock( node-
>getAttributeInt( "dock_index_to" ) );
      }
}

rp::RPXMLNode* RPConnector::save( rp::RPXMLNode* parent_node )
{
      rp::RPXMLNode* obj_node = RPObjectChart::save( parent_node );
      obj_node->insertAttribute( "obj_from", dock_begin-
>object().getFullName() );
      obj_node->insertAttribute( "obj_to", dock_end->object().getFullName()
);
      obj_node->insertAttribute( "dock_index_from", dock_begin->getIndex()
);
      obj_node->insertAttribute( "dock_index_to", dock_end->getIndex() );
      return obj_node;
}

void RPConnector::saveToXML(REF(pugi::xml_node) parentNode) const
{
      // Записываем узел <RPConnector/>:
      pugi::xml_node node =
parentNode.append_child(getClassName().c_str());
      // Сохраняем атрибуты объекта:
      node.append_attribute("obj_from")        .set_value(dock_begin-
>object().getFullName().c_str());
      node.append_attribute("obj_to")          .set_value(dock_end->
object().getFullName().c_str());
      node.append_attribute("index_from")      .set_value(dock_begin-
>getIndex()                    );
      node.append_attribute("index_to")        .set_value(dock_end->
getIndex()                    );
}

void RPConnector::loadFromXML(CREF(pugi::xml_node) node)
{
      PTR(RPObject) pObjFrom = NULL;
      PTR(RPObject) pObjTo   = NULL;
      unsigned int iFrom     = 0;
      unsigned int iTo       = 0;

      // Считываем атрибуты для загрузки сохраненного блока "Connector":
      for (pugi::xml_attribute attr = node.first_attribute(); attr; attr =
attr.next_attribute())
```

```
	{
		// Присваиваем сохраненные в xml-файле параметры:
		// Для отображения объекта на Flowchart'e
		tstring attrName = attr.name();
		if (attrName == _T("obj_from"))
		{
			pObjFrom = rpMethod::project->findObject(attr.value());
			// Для блока Resource восстанавливаем его коннектор
(отличен от других):
			if (pObjFrom->getClassInfo()-
>isKindOf("RPShapeResource_MJ"))
			{
				main_pen_width = 1;
				main_pen_default.DeleteObject();
				main_pen_default.CreatePen( PS_DASHDOTDOT, 1,
RGB(0x00, 0x00, 0xFF) );
				can_update = false;
				setPen( main_pen_default );
				can_update = true;
			}
		}
		else if (attrName == _T("obj_to"))
		{
			pObjTo = rpMethod::project->findObject(attr.value());
		}
		else if (attrName == _T("index_from"))
		{
			iFrom = attr.as_uint();
		}
		else if (attrName == _T("index_to"))
		{
			iTo = attr.as_uint();
		}
	}
	if (pObjFrom && pObjTo && pObjFrom->getClassInfo()-
>isKindOf("RPShape") && pObjTo->getClassInfo()->isKindOf("RPShape"))
	{
		dock_begin = static_cast<RPShape*>(pObjFrom)->getDock(iFrom);
		dock_end   = static_cast<RPShape*>(pObjTo)  ->getDock(iTo);
		dock_begin->connectors.push_back(this);
		dock_end  ->connectors.push_back(this);
	}
}
```

```
void RPConnector::registerObject()
{
	rpMethod::factory->insertFactory( new RPObjectClassInfo(
"RPConnector", "RPObjectChart", RPConnector::newObject ) );
}

RPObject* RPConnector::newObject( RPObject* parent )
{
	return new RPConnector( parent );
}
```

ПРИЛОЖЕНИЕ 2. Листинг файлов имитатора

Часть файла generate.cpp

```
#include "simulator/runtime/pch.h"
#include "utils/rdotypes.h"
#include "utils/rdomacros.h"
#include "simulator/runtime/process/generate.h"
#include "simulator/runtime/calc/calc_base.h"
OPEN_RDO_RUNTIME_NAMESPACE

// -------------------- RDOPROCGenerate

IBaseOperation::BOResult RDOPROCGenerate::onDoOperation(CREF(LPRDORuntime)
pRuntime)
{
	++m_TransCount;

	if (m_pStatistics)
		m_pStatistics->setTransCount(m_TransCount);

	LPRDOPROCTransact pTransact = m_pCreateAndGoOnTransactCalc-
>calcValue(pRuntime).getPointerSafety<RDOResourceTypeTransact>();
	ASSERT(pTransact);

	pTransact->setBlock(this);
	pTransact->next();

	PTR(RDOTrace) tracer = pRuntime->getTracer();
	if (!tracer->isNull())
	{
		tracer->getOStream() << pTransact->traceResourceState('\0',
pRuntime) << tracer->getEOL();
	}

	calcNextTimeInterval(pRuntime);
	return IBaseOperation::BOR_done;
}


void RDOPROCGenerate::setStatistics(CREF(LPIInternalStatistics)
pStatistics)
```

```
{
	m_pStatistics = pStatistics;
}

CLOSE_RDO_RUNTIME_NAMESPACE
```

ПРИЛОЖЕНИЕ 3. Листинг файлов компилятора графической части системы РДО

Часть файла procgui_datablock.cpp

```
#include "simulator/compiler/procgui/pch.h"
#include <algorithm>
#include "simulator/compiler/procgui/procgui_datablock.h"

OPEN_COMPILER_GUI_NAMESPACE

// -------------------- RPShapeDataBlock
RPShapeDataBlock::RPShapeDataBlock(RPShapeDataBlock::zakonRaspr zakon,
CREF(tstring) name)
	: m_zakon(zakon)
	, m_name (name )
{}

void RPShapeDataBlock::setDisp(double disp)
{
	m_disp=disp;
}

void RPShapeDataBlock::setExp(double exp)
{
	m_exp=exp;
}

void RPShapeDataBlock::setBase(int base)
{
	m_base=base;
}

void RPShapeDataBlock::setMax(double max)
{
```

```
	m_max=max;
}

// -------------------- RPShapeDataBlockCreate
RPShapeDataBlockCreate::RPShapeDataBlockCreate(RPShapeDataBlock::zakonRasp
r zakon, CREF(tstring) name)
	: RPShapeDataBlock(zakon, name)
{}

RPShapeDataBlockCreate::~RPShapeDataBlockCreate()
{}

void RPShapeDataBlockCreate::setAmount(int amount)
{
	m_amount = amount;
}

void
RPShapeDataBlockCreate::setStatistics(CREF(rdo::runtime::LPIInternalStatis
tics) pStatistics)
{
	m_pStatistics = pStatistics;
}
// -------------------- RPShapeDataBlockTerminate
RPShapeDataBlockTerminate::RPShapeDataBlockTerminate(CREF(tstring) name)
	:m_name(name)
{}

RPShapeDataBlockTerminate::~RPShapeDataBlockTerminate()
{}

void RPShapeDataBlockTerminate::setTermInc(int term_inc)
{
	m_term_inc = term_inc;
}

void
RPShapeDataBlockTerminate::setStatistics(CREF(rdo::runtime::LPIInternalSta
tistics) pStatistics)
{
	m_pStatistics = pStatistics;
}

// -------------------- RPShapeDataBlockProcess
```

```
RPShapeDataBlockProcess::RPShapeDataBlockProcess(RPShapeDataBlock::zakonRa
spr zakon, CREF(tstring) name)
	: RPShapeDataBlock(zakon,name)
{}

RPShapeDataBlockProcess::~RPShapeDataBlockProcess()
{}

void RPShapeDataBlockProcess::addAction(RPShapeDataBlockProcess::Action
action)
{
	m_actionList.push_back(action);
}

void RPShapeDataBlockProcess::addRes(CREF(tstring) resName)
{
	ASSERT(std::find(m_resNameList.begin(), m_resNameList.end(), resName)
== m_resNameList.end());

	m_resNameList.push_back(resName);
}

CREF(RPShapeDataBlockProcess::ActionList)
RPShapeDataBlockProcess::getActionList() const
{
	return m_actionList;
}

CREF(RPShapeDataBlockProcess::ResNameList)
RPShapeDataBlockProcess::getResNameList() const
{
	return m_resNameList;
}

void
RPShapeDataBlockProcess::setStatistics(CREF(rdo::runtime::LPIInternalStati
stics) pStatistics)
{
	m_pStatistics = pStatistics;
}

CLOSE_COMPILER_GUI_NAMESPACE
```