Оглавление

[1. Перечень сокращений 2](#_Toc343467185)

[2. Введение 3](#_Toc343467186)

[3. Предпроектное исследование 4](#_Toc343467187)

[3.2. Подсистема анимации в RAO-studio 5](#_Toc343467188)

[3.3. Использование временных ресурсов в подсистеме анимации 6](#_Toc343467189)

[4. Формирование ТЗ 8](#_Toc343467190)

[4.1. Основания для разработки 8](#_Toc343467191)

[4.2. Общие сведения. 8](#_Toc343467192)

[4.3. Назначение и цели развития системы 8](#_Toc343467193)

[4.4. Характеристики объекта автоматизации 8](#_Toc343467194)

[4.5. Требования к системе 8](#_Toc343467195)

[4.5.1. Требования к функциональным характеристикам 8](#_Toc343467196)

[4.5.2. Требования к надежности 8](#_Toc343467197)

[4.5.3. Условия эксплуатации 8](#_Toc343467198)

[4.5.4. Требования к составу и параметрам технических средств 8](#_Toc343467199)

[4.5.5. Требования к информационной и программной совместимости 9](#_Toc343467200)

[4.5.6. Требования к маркировке и упаковке 9](#_Toc343467201)

[4.5.7. Требования к транспортированию и хранению 9](#_Toc343467202)

[4.5.8. Порядок контроля и приемки 9](#_Toc343467203)

[5. Концептуальный этап проектирования системы 10](#_Toc343467204)

[6. Технический этап проектирования системы 12](#_Toc343467205)

[6.1. Процедурный язык программирования в подсистеме анимации 12](#_Toc343467206)

[6.2. Доработка синтаксиса в подсистеме анимации 12](#_Toc343467207)

[6.2.1. Синтаксис получения массива ресурсов 12](#_Toc343467208)

[6.2.2. Синтаксис доступа к параметру элемента массива ресурсов 13](#_Toc343467209)

[6.2.3. Разработка архитектуры компонента rdo\_parser 13](#_Toc343467210)

[6.2.4. Разработка архитектуры компонента rdo\_runtime 14](#_Toc343467211)

[7. Рабочий этап проектирования системы 15](#_Toc343467212)

[7.1. Получение массива ресурсов 15](#_Toc343467213)

[7.1.1. Изменения в файлах синтаксического анализатора 15](#_Toc343467214)

[7.1.2. Разработка компонента rdo\_parser 16](#_Toc343467215)

[7.1.3. Разработка компонента rdo\_runtime 16](#_Toc343467216)

[7.2. Доступ к параметру ресурса – элемента массива 18](#_Toc343467217)

[7.2.1. Изменения в файлах синтаксического анализатора 18](#_Toc343467218)

[8. Апробирование разработанной системы для модельных условий 20](#_Toc343467219)

[9. Заключение 21](#_Toc343467220)

[10. Список используемых источников 22](#_Toc343467221)

[11. Приложение А. Код имитационной модели 23](#_Toc343467222)

1. **Перечень сокращений**

РП – Рабочий Проект

ТЗ – Техническое Задание

ТП – Технический Проект

ИМ – Имитационное Моделирование, Имитационная Модель

СДС – Сложная Дискретная Система

ЭВМ - Электронная Вычислительная Машина

ОЗУ - Оперативное Запоминающее Устройство

1. **Введение**

Имитационное моделирование (ИМ) на ЭВМ находит широкое применение при исследовании и управлении сложными дискретными системами (СДС) и процессами, в них протекающими. К таким системам можно отнести экономические и производственные объекты, морские порты, аэропорты, комплексы перекачки нефти и газа, ирригационные системы, программное обеспечение сложных систем управления, вычислительные сети и многие другие.

Интеллектуальное имитационное моделирование, характеризующиеся возможностью использования методов искусственного интеллекта и, прежде всего, знаний при принятии решений в процессе имитации, при управлении имитационным экспериментом, при реализации интерфейса пользователя, создании информационных банков ИМ, использовании нечетких данных, снимает часть проблем использования ИМ.

ИМ является эффективным, но и не лишенным недостатков, методом. Трудности использования ИМ, связаны с обеспечением адекватности описания системы, интерпретацией результатов, обеспечением стохастической сходимости процесса моделирования, решением проблемы размерности и т.п. К проблемам применения ИМ следует отнести также и большую трудоемкость данного метода.

Разработка интеллектуальной среды имитационного моделирования РДО – «Ресурсы, Действия, Операции», выполнена в Московском Государственном Техническом Университете им. Н.Э. Баумана на кафедре "Компьютерные системы автоматизации производства". Причинами создания РДО явились требования к универсальности ИМ относительно классов моделируемых систем и процессов, легкости модификации моделей, а также моделирования сложных систем управления совместно с управляемым объектом (включая использование ИМ в управлении в реальном масштабе времени). Таким образом, среда РДО стала решением указанных выше проблем ИМ и обеспечила исследователя и проектировщика новыми возможностями.

Программный комплекс RAO-studio предназначен для разработки и отладки имитационных моделей на языке РДО. Основные цели данного комплекса - обеспечение пользователя легким в обращении, но достаточно мощным средством разработки текстов моделей на языке РДО, обладающим большинством функций по работе с текстами программ, характерных для сред программирования, а также средствами проведения и обработки результатов имитационных экспериментов.

1. **Предпроектное исследование**
   1. **Основные положения языка РДО**

В основе системы РДО лежат следующие положения:

* Все элементы сложной дискретной системы (СДС) представлены как ресурсы, описываемые некоторыми параметрами.
* Состояние ресурса определяется вектором значений всех его параметров; состояние СДС – значением всех параметров всех ресурсов.
* Процесс, протекающий в СДС, описывается как последовательность целенаправленных действий и нерегулярных событий, изменяющих определенным образом состояния ресурсов; действия ограничены во времени двумя событиями: событиями начала и конца.
* Нерегулярные события описывают изменение состояния СДС, непредсказуемые в рамках продукционной модели системы (влияние внешних по отношению к СДС факторов либо факторов, внутренних по отношению к ресурсам СДС). Моменты наступления нерегулярных событий случайны.
* Действия описываются операциями, которые представляют собой модифицированные продукционные правила, учитывающие временные связи. Операция описывает предусловия, которым должно удовлетворять состояние участвующих в операции ресурсов, и правила изменения ресурсов в начале и конце соответствующего действия.

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

**Модель** – совокупность объектов РДО-языка, 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 СДС на языке РДО.

Объектами исходных данных являются:

* [типы ресурсов](http://rdo.rk9.bmstu.ru/help/help/rdo_lang_rus/html/rdo_base/rdo_base_res_types.htm) (с расширением .rtp);
* [ресурсы](http://rdo.rk9.bmstu.ru/help/help/rdo_lang_rus/html/rdo_base/rdo_base_resources.htm) (с расширением .rss);
* [события](http://rdo.rk9.bmstu.ru/help/help/rdo_lang_rus/html/rdo_obj/rdo_pat_type/rdo_pat_type_event.htm) (с расширением .evn);
* [образцы активностей](http://rdo.rk9.bmstu.ru/help/help/rdo_lang_rus/html/rdo_obj/rdo_obj_pat.htm) (с расширением .pat);
* [точки принятия решений](http://rdo.rk9.bmstu.ru/help/help/rdo_lang_rus/html/rdo_obj/rdo_obj_dpt.htm) и [процессы обслуживания](http://rdo.rk9.bmstu.ru/help/help/rdo_lang_rus/html/rdo_obj/rdo_obj_proc.htm) (с расширением .dpt);
* [константы, функции и последовательности](http://rdo.rk9.bmstu.ru/help/help/rdo_lang_rus/html/rdo_obj/rdo_obj_const_fun_sec.htm) (с расширением .fun);
* [кадры анимации](http://rdo.rk9.bmstu.ru/help/help/rdo_lang_rus/html/rdo_obj/rdo_obj_frm.htm) (с расширением .frm);
* [требуемая статистика](http://rdo.rk9.bmstu.ru/help/help/rdo_lang_rus/html/rdo_obj/rdo_obj_pmd.htm) (с расширением .pmd);
* [прогон](http://rdo.rk9.bmstu.ru/help/help/rdo_lang_rus/html/rdo_obj/rdo_obj_smr.htm) (с расширением .smr);
* [процессы обслуживания](http://rdo.rk9.bmstu.ru/help/help/rdo_lang_rus/html/rdo_obj/rdo_obj_proc.htm) (с расширением .prc).

Объекты, создаваемые РДО-имитатором при выполнении прогона:

* [результаты](http://rdo.rk9.bmstu.ru/help/help/rdo_lang_rus/html/rdo_res/rdo_res_pmv.htm) (с расширением .pmv);
* [трассировка](http://rdo.rk9.bmstu.ru/help/help/rdo_lang_rus/html/rdo_res/rdo_res_trc.htm) (с расширением .trc). [2]
  1. **Подсистема анимации в RAO-studio**

В рамках задачи, поставленной в данном курсовом проекте, рассматривается подсистема анимации, используемая в среде РДО. Стоит отдельно обратить внимание на то, что визуализация работы системы, отдельных ее элементов, а также их параметров, очень важна при разработке имитационных моделей, так как позволяет наглядно продемонстрировать работу системы, описываемой моделью, уже в процессе имитации.

В RAO-studio кадр представляет собой прямоугольную область экрана, в которой производится отображение. Он состоит из фоновой картинки и переменных элементов (элементов отображения), состав, форма, размеры и расположение которых определяются состоянием системы и, следовательно, могут изменяться во время просмотра кадра.

Описание кадра имеет следующий формат:

$Frame [<имя\_кадра>](qthelp://studio/doc/rdo_lang_rus/html/rdo_obj/rdo_obj_frm.htm#frm_name)

[ Show\_if [<условие\_показа\_кадра>](qthelp://studio/doc/rdo_lang_rus/html/rdo_obj/rdo_obj_frm.htm#show_cond) ]

$Back\_picture = [<описание\_фоновой\_картинки>](qthelp://studio/doc/rdo_lang_rus/html/rdo_obj/rdo_obj_frm.htm#back_desc)

[ [<описание\_элементов\_отображения>](qthelp://studio/doc/rdo_lang_rus/html/rdo_obj/rdo_obj_frm.htm#elem_desc) ]

$End

**имя\_кадра**

Имя кадра представляет собой [простое имя](qthelp://studio/doc/rdo_lang_rus/html/rdo_lex_div/rdo_lex_div_names.htm). Имена должны быть различными для всех кадров и не должны совпадать с ранее определенными именами.

**условие\_показа\_кадра**

Условие показа кадра используется при автоматическом переключении кадров и представляет собой [логическое выражение](qthelp://studio/doc/rdo_lang_rus/html/rdo_base/rdo_base_equations.htm). Это логическое выражение вычисляется при каждом событии. Если оно истинно, то кадр отображается, если ложно - то нет. Если при очередном событии значение выражения меняется, то кадр появляется на экране либо исчезает. Условие показа вместе с зарезервированным словом Show\_if может отсутствовать. Кадры с заданным условием показа называют условными. Если при некотором состоянии моделируемой системы выполняются условия показа нескольких условных кадров, то отображается тот из них, который в объекте описания кадров встречается раньше других.

**элемент\_отображения**

Элемент отображения имеет следующий формат:

[<тип\_элемента>](http://rdo.rk9.bmstu.ru/help/help/rdo_lang_rus/html/rdo_obj/rdo_obj_frm_elem.htm#elem_type) [ [<свойства\_элемента>](http://rdo.rk9.bmstu.ru/help/help/rdo_lang_rus/html/rdo_obj/rdo_obj_frm_elem.htm" \l "elem_props) ]

**тип\_элемента**

Тип элемента задают одним из следующих зарезервированных слов:

|  |  |
| --- | --- |
| Тип элемента | Описание |
| text | Текстовый элемент |
| bitmap | Битовая карта |
| rect | Прямоугольник |
| line | Отрезок прямой |
| circle | Окружность |
| ellipse | Эллипс |
| r\_rect | Прямоугольник со скругленными углами |
| triang | Треугольник |
| s\_bmp | Масштабируемая битовая карта |

**свойства\_элемента**

Порядок записи, количество и смысл свойств элемента зависят от типа элемента. Свойства элементов записываются в прямых скобках и разделяются запятыми.

* 1. **Использование временных ресурсов в подсистеме анимации**

**Временные ресурсы** в РДО – это вид ресурсов, которые во время прогона могут создаваться и уничтожаться при выполнении событий, активностей и процессов.

Анализ большого количества уже существующих моделей, разработанных с использованием среды РДО, а также анализ требований к вновь разрабатываемым моделям, показал, что при наличии большого количества временных ресурсов, использующихся при проектировании имитационных моделей, довольно часто существует необходимость в их визуализации. Например, путем передачи их параметров в качестве **свойств элемента** отображения РДО (см. **4.2**).

В подсистеме анимации при ее текущем состоянии функционал по взаимодействию с временными ресурсами крайне ограничен и представлен следующим набором методов:

|  |  |
| --- | --- |
| Имя функции | Значение |
| Exist | Если существует хотя бы один ресурс указанного типа, состояние которого удовлетворяет заданному логическому выражению, функция выдает значение ИСТИНА, в противном случае - ЛОЖЬ |
| Not\_Exist | Если не существует ни одного ресурса указанного типа, состояние которого удовлетворяет заданному логическому выражению, функция выдает значение ИСТИНА, в противном случае - ЛОЖЬ |
| For\_All | Если состояние всех ресурсов указанного типа удовлетворяет заданному логическому выражению, функция выдает значение ИСТИНА, в противном случае - ЛОЖЬ |
| Not\_For\_All | Если состояние не всех ресурсов указанного типа удовлетворяет заданному логическому выражению, функция выдает значение ИСТИНА, в противном случае - ЛОЖЬ |

Если вместо логического выражения указано зарезервированное слово NoCheck, то рассматриваются все ресурсы указанного типа. В этом случае функция Exist определяет, существует ли хотя бы один ресурс указанного типа, функция For\_All всегда выдает значение ИСТИНА (даже если нет ни одного ресурса указанного типа), а функция Not\_For\_All всегда выдает значение ЛОЖЬ.

В языке определена еще одна стандартная функция, позволяющая работать со списком ресурсов. Она позволяет отобрать только часть ресурсов указанного типа и проверить их состояния. Синтаксис:

Select( <имя\_типа\_ресурсов>:

(<логическое\_выражение> | NoCheck) ).<метод\_списка>

Список методов приведен в таблице:

|  |  |  |
| --- | --- | --- |
| Имя метода | Описание | Значение |
| Exist | Описание аналогично [приведённому](http://rdo.rk9.bmstu.ru/help/help/rdo_lang_rus/html/rdo_base/rdo_base_std_var_func.htm#exist), но проверяются не все, ресурсы, а только отобранные в список. Формат использования:  Exist( <логическое\_выражение> | NoCheck ) | Логическое |
| Not\_Exist | Описание аналогично [приведённому](http://rdo.rk9.bmstu.ru/help/help/rdo_lang_rus/html/rdo_base/rdo_base_std_var_func.htm#not_exist), но проверяются не все, ресурсы, а только отобранные в список. Формат использования:  Not\_Exist( <логическое\_выражение> | NoCheck ) | Логическое |
| For\_All | Описание аналогично приведённому, но проверяются не все, ресурсы, а только отобранные в список. Формат использования:  For\_All( <логическое\_выражение> | NoCheck ) | Логическое |
| Not\_For\_All | Описание аналогично приведённому, но проверяются не все, ресурсы, а только отобранные в список. Формат использования:  Not\_For\_All( <логическое\_выражение> | NoCheck ) | Логическое |
| Empty | Возвращает значение ИСТИНА, если список отобранных ресурсов пуст, в противном случае - ЛОЖЬ. Формат использования:  Empty() | Логическое |
| Size | Возвращает размер списка ресурсов, целое число. Если список пуст, то вовращается ноль. Формат использования:  Size() | Арифметич. |

Примеры:

Exist(Заявки : Заявки.Состояние = ожидает)

Not\_Exist(Тип\_1 : NoCheck)

Select(Заявка: Заявка.Время\_прихода >= 480 and

Заявка.Состояние = в\_очереди).For\_All(Заявка.Время\_в\_очереди > 20) [1]

* 1. **Постановка задачи**

Таким образом, можно сделать вывод, что для подсистемы анимации наиболее гибким и подходящим решением будет **прямой доступ** к параметрам временных ресурсов. Это позволит в значительной степени расширить возможности по визуализации моделей, что, в свою очередь, кроме придания модели наглядности и улучшения ее восприятия, откроет разработчикам дополнительные возможности в области отладки проектируемых ими систем (в тех случаях, когда уже доступный в RAO-studio функционал по трассировке окажется неудобен или попросту излишен).

1. **Формирование ТЗ**
   1. **Основания для разработки**

Задание на курсовой проект.

* 1. **Общие сведения.**

В системе РДО расширяются возможности работы с временными ресурсами в подсистеме анимации. Основной разработчик РДО – кафедра РК-9, МГТУ им. Н.Э. Баумана.

* 1. **Назначение и цели развития системы**

В подсистеме анимации среды РДО разработать механизм создания массива с временными ресурсами, отобранными по определенному критерию, а также механизм обращения к параметрам ресурсов – элементов данного массива.

* 1. **Характеристики объекта автоматизации**

РДО – язык имитационного моделирования, включающий все три основные подхода описания дискретных систем: процессный, событийный и сканирования активностей.

* 1. **Требования к системе**
     1. **Требования к функциональным характеристикам**
* обеспечение в подсистеме анимации РДО возможности создания массива временных ресурсов, отобранных по определенным условиям;
* обеспечение доступа к параметрам ресурсов, хранящихся в полученном массиве;
* демонстрация новых возможностей системы на примере простой имитационной модели, в полной мере задействующей эти возможности;
* дополнение документации новыми возможностями системы.
  + 1. **Требования к надежности**

Основное требование к надежности направлено на поддержание в исправном и работоспособном ЭВМ на которой происходит использование программного комплекса RAO-studio.

* + 1. **Условия эксплуатации**

Аппаратные средства должны эксплуатироваться в помещениях с выделенной розеточной электросетью 220В ±10%, 50 Гц с защитным заземлением.

* + 1. **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 с разрешением от 800\*600 и выше;
  + 1. **Требования к информационной и программной совместимости**

Данная система должна работать под управлением операционных систем Windows 2000, Windows XP, Windows Vista и Windows 7.

* + 1. **Требования к маркировке и упаковке**

Не предъявляются.

* + 1. **Требования к транспортированию и хранению**

Не предъявляются.

* + 1. **Порядок контроля и приемки**

Контроль и приемка передачи параметров должны осуществляться на тестовом примере модели.

1. **Концептуальный этап проектирования системы**

Наиболее логичной реализацией задачи прямого доступа к ресурсам и их параметрам, в нашем случае, будет такая, при которой в максимальной степени будут использоваться уже существующие возможности системы.

Анализ типов данных, доступных для использования в РДО, показал, что на данный момент в системе уже реализованы необходимые типы данных, способные обеспечить данную концепцию. Такими типами являются **массив** и сам **тип** **временного ресурса**.

Массив можно создать следующей командой:

array< *Тип\_данных\_языка\_РДО* >;

Где *Тип\_данных\_языка\_РДО* может быть любым из атомарных типов языка (***integer****,* ***real****,* ***bool****,* ***string***), но и также типом временного ресурса.

Следовательно, **первой задачей** является передача списка временных ресурсов, отобранных по какому-либо условию, в массив, принимающий объект ресурса в качестве своего элемента.

Наиболее удобной выглядит реализация данного функционала в виде нового метода к уже существующей функции работы с временными ресурсами **Select** (См. 3.3). Таким образом, новый метод, по аналогии с методом **Size**, вернет арифметическое выражение, типом которого будет массив, содержащий временные ресурсы в качестве элементов.

Далее можно будет получить доступ к любому из параметров ресурса по аналогии с существующим синтаксисом РДО (**вторая задача**).

* 1. **Диаграмма компонентов**

**N:\RDOKURSACH\Диаграммаразвертывания.emf**

***Рис. 1. Упрощенная диаграмма компонентов***

Базовый функционал представленных на диаграмме компонентов (Рис. 1)**Ошибка! Источник ссылки не найден.**:

**rdo\_kernel** реализует ядровые функции системы. Не изменяется при разработке системы.

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

**rdo\_repository** реализует управление потоками данных внутри системы и отвечает за хранение и получение информации о модели. Не изменяется при разработке системы.

**rdo\_mbuilder** реализует функционал, используемый для программного управления типами ресурсов и ресурсами модели. Не изменяется при разработке системы.

**rdo\_simulator** управляет процессом моделирования на всех его этапах. Он осуществляет координацию и управление компонентами rdo\_runtime и rdo\_parser. Не изменяется при разработке системы.

**rdo\_parser** производит лексический и синтаксический разбор исходных текстов модели, написанной на языке РДО. Модернизируется при разработке системы.

**rdo\_runtime** отвечает за непосредственное выполнение модели, управление базой данных и базой знаний. Модернизируется при разработке системы.

Объекты компонента **rdo\_runtime** инициализируются при разборе исходного текста модели компонентом **rdo\_parser**.

В дальнейшем компоненты **rdo\_parser** и **rdo\_runtime** описываются более детально.

1. **Технический этап проектирования системы**
   1. **Процедурный язык программирования в подсистеме анимации**

Предложенные на этапе концептуального проектирования задачи и идеи потребовали, в свою очередь, решения еще одной задачи – внедрения **процедурного языка** **программирования** в подсистему анимации среды РДО. Данная задача стала блокирующей для выполнения двух основных, а ее решение оказалось довольно трудоемким и требующим значительных архитектурных изменений в системе в рамках курсового проекта.

Эта задача была выполнена **руководителем курсового проекта**, сделав возможным создание локальных переменных в подсистеме анимации (например, необходимый массив временных ресурсов), и использование операторов процедурного программирования (if..else, for и других операторов, которые позволят достичь необходимой гибкости во взаимодействии с полученным массивом).

* 1. **Доработка синтаксиса в подсистеме анимации**

Для создания лексического анализатора в системе РДО используется генератор лексических анализаторов общего назначения *Bison,* который преобразует описание контекстно-свободной LALR(1) грамматики в программу на языке C++ для разбора этой грамматики. Для того чтобы Bison мог разобрать программу на каком-то языке, этот язык должен быть описан *контекстно-свободной грамматикой*. Это означает, необходимо определить одну или более *синтаксических групп* и задать правила их сборки из составных частей. Наиболее распространённой формальной системой для представления таких правил в удобном для человека виде является *форма Бэкуса-Наура* (БНФ, Backus-Naur Form, BNF), которая является контекстно-свободной грамматикой. Bison принимает на вход, в сущности, особый вид БНФ, адаптированный для машинной обработки, [7].

В правилах формальной грамматики языка каждый вид синтаксических единиц или групп называется *символом*. Те из них, которые формируются группировкой меньших конструкций в соответствии с правилами грамматики, называются *нетерминальными символами*, а те, что не могут разбиты -- *терминальными символами* или *типами лексем*.

* + 1. **Синтаксис получения массива ресурсов**

Так как на этапе концептуального проектирования было решено, что механизм получения массива временных ресурсов будет реализован через новый метод функции Select, то в результате синтаксис вызова должен будет выглядеть следующим образом:

Select(Тип\_ресурса: условия\_выборки | **NoCheck**).**getArray**();

*или*

Select(Тип\_ресурса: условия\_выборки | **NoCheck**).**get\_array**();

Данный метод вернет арифметическое выражение, представленное массивом с ресурсами, отобранными по определенным условиям, указанным в теле функции Select, или же со всеми ресурсами этого типа, если вместо условия будет написано ключевое слово **NoCheck** (См. 3.3). Это выражение можно будет присвоить переменной объявленного ранее массива.

C:\Users\kirill\Dropbox\backup\rdo курсач\select_arithm.emfC:\Users\kirill\Dropbox\backup\rdo курсач\select_arithm.emf

***Рисунок 2. Синтаксическая диаграмма арифметического выражения функции Select***

Соответствующие этому вызову синтаксические диаграммы более подробно представлены на 2-м листе курсового проекта.

* + 1. **Синтаксис доступа к параметру элемента массива ресурсов**

Механизм доступа к параметрам ресурсов, хранящихся в массиве, было решено сделать по аналогии с уже существующими механизмами вызова параметров и доступа к элементу массива, синтаксис которых, в свою очередь, аналогичен синтаксису в языке С++.

Таким образом, доступ к параметрам массива можно продемонстрировать следующим примером:

if (**k[i].состояние** == Погрузка and **k[i].погрузчик** != 0)

{

text( 40, 20 + **k[i].ширина**, 10, 12, white, black, > 'П');

text(50, 20 + **k[i].ширина**, 10, 12, white, black, < **k[i].погрузчик**);

}

Выражения типа **k[i].*имя\_параметра*** возвращают арифметические выражения того же типа, как и параметр ресурса, что позволяет использовать их в любых целях (в операторах сравнения, в том числе и для перечислимых типов enum, в арифметических операциях с числовыми значениями и т.д.).

Часть синтаксической диаграммы для арифметического выражения, относящаяся к параметру элемента массива ресурсов, представлена на иллюстрации ниже:

…

D:\SSD\Dropbox\backup\rdo курсач\12.emf



…

***Рисунок 3. Часть синтаксической диаграммы арифметического выражения***

Полная синтаксическая диаграмма, как и в случае с предыдущим пунктом, представлена на 2-м листе курсового проекта.

* + 1. **Разработка архитектуры компонента rdo\_parser**

Для возможности обработки новой конструкции в коде модели необходимо дополнить класс RDOFUNSelect, отвечающий за создание выражений, возвращаемых функцией Select, новым методом, задающим в компоненте rdo\_runtime вычисление массива.

Также потребуется внести изменения, соответствующие пунктам 6.2.1 и 6.2.2, в файлы лексического анализатора (**flex**) и файлы анализатора Bison, дополнив их новыми лексемами и соответствующим им кодом на языке С++.

* + 1. **Разработка архитектуры компонента rdo\_runtime**

В компоненте необходимо написать вычислитель (Calc), преобразовывающий искомый массив с временными ресурсами из внутреннего формата класса RDOFunCalcSelect, где данные хранятся в одном из атрибутов в «сыром виде», в формат контейнера для значений в системе РДО – RDOValue.

1. **Рабочий этап проектирования системы**

Для реализации в среде имитационного моделирования новых инструментов, разработанных на предыдущих этапах проектирования, в первую очередь, необходимо добавить новые терминальные символы в лексический анализатор РДО и нетерминальные символы в грамматический анализатор в соответствии с синтаксическими диаграммами, разработанными на техническом этапе проектирования, и соответствующий им код на языке С++. Также необходимо дополнить компоненты rdo\_parser и rdo\_runtime необходимыми классами и методами, обеспечивающими выполнение этого кода (в соответствии с пп. 6.2.3, 6.2.4).

* 1. **Получение массива ресурсов**
     1. **Изменения в файлах синтаксического анализатора**

В лексическом анализаторе (flex) был добавлен новый токен RDO\_SelectArray, который может быть записан следующими способами:

getArray return(RDO\_SelectArray);

get\_array return(RDO\_SelectArray);

Этот токен также необходимо также добавить в файлы грамматики генератора синтаксического анализатора (Bison):

%token RDO\_SelectArray

Далее необходимо добавить описание для вызова метода функции Select в соответствии с синтаксическими диаграммами:

fun\_select\_arithm

: fun\_select\_body '.' RDO\_Size '(' ')'

| fun\_select\_body '.' RDO\_Size error

| fun\_select\_body '.' RDO\_Size '(' error

| fun\_select\_body '.' RDO\_Select\_Array '(' ')'

{

LPRDOFUNSelect pSelect =

PARSER->stack().pop<RDOFUNSelect>($1);

ASSERT(pSelect);

pSelect->setSrcPos(@1, @5);

RDOParserSrcInfo arrayInfo(@3, @5, \_T("Array()"));

LPRDOFUNArithm pArithm =

pSelect->createFunSelectArray(arrayInfo);

ASSERT(pArithm);

$$ = PARSER->stack().push(pArithm);

}

| fun\_select\_body '.' RDO\_Select\_Array error

{

PARSER->error().error(@3, \_T("Ожидается

октрывающаяся скобка"));

}

| fun\_select\_body '.' RDO\_Select\_Array '(' error

{

PARSER->error().error(@4, \_T("Ожидается

закрывающаяся скобка"));

};

Данный код создает арифметическое выражение с массивом, вызывая метод createFunSelectArray класса RDOFUNSelect, который также является объектом проектирования (См. 6.2.3). Листинг данного метода представлен в следующем пункте, 7.1.2.

* + 1. **Разработка компонента rdo\_parser**

Для получения массива ресурсов в процессе компиляции имитационной модели было необходимо дополнить компонент rdo\_parser, а именно класс RDOFUNSelect, методом createFunSelectArray. Данный метод создает выражение (expression), предназначенное для вычисления массива. Ниже приведен листинг данного метода:

LPRDOFUNArithm RDOFUNSelect::createFunSelectArray(

CREF(RDOParserSrcInfo) array\_info)

{

setSrcText(src\_text() + \_T(".") + array\_info.src\_text());

RDOParser::s\_parser()->getFUNGroupStack().pop\_back();

end();

LPRDOArrayType pArrayType = rdo::Factory<RDOArrayType>::create(

rdo::Factory<TypeInfo>::create(getResType(),

array\_info), array\_info);

LPExpression pExpression = rdo::Factory<Expression>::create(

rdo::Factory<TypeInfo>::create(pArrayType, array\_info),

rdo::Factory<rdo::runtime::RDOFunCalcSelectArray>::create(

m\_pCalcSelect),

array\_info

);

ASSERT(pExpression);

LPRDOFUNArithm pArithm =

rdo::Factory<RDOFUNArithm>::create(pExpression);

ASSERT(pArithm);

pArithm->setSrcInfo(array\_info);

return pArithm;

}

Это выражение (pExpression) инициализируется специальным вычислителем (Calc), представляющим собой класс rdo::runtime::RDOFunCalcSelectArray и входящим в компонент rdo\_runtime, что видно из его пространства имен.

* + 1. **Разработка компонента rdo\_runtime**

Ключевым методом Calc‘а RDOFunCalcSelectArray является RDOFunCalcSelectArray::doCalc, код которого представлен ниже:

RDOValue RDOFunCalcSelectArray::doCalc(

CREF(LPRDORuntime) pRuntime)

{

m\_pSelect->prepare(pRuntime);

std::list<LPRDOResource>::iterator it =

m\_pSelect->res\_list.begin();

std::list<LPRDOResource>::iterator end =

m\_pSelect->res\_list.end ();

rdo::runtime::LPRDOArrayType pType =

rdo::Factory<rdo::runtime::RDOArrayType >::create(

m\_pSelect->getResType());

ASSERT(pType);

rdo::runtime::LPRDOArrayValue pValue =

rdo::Factory<rdo::runtime::RDOArrayValue>::create(pType);

ASSERT(pValue);

while (it != end)

{

pValue->push\_back(rdo::runtime::RDOValue(

m\_pSelect->getResType(), \*(it++)));

}

return RDOValue(pType, pValue);

}

Данный код позволяет увидеть, что сначала на основе типа ресурса, хранящегося в объекте уже упоминавшегося RDOFunCalcSelect, и получаемого методом getResType(). Стоит отметить, что для осуществления этой операции потребовался небольшой редизайн данного класса. Был добавлен новый частный атрибут, хранящий в себе объект с типом ресурса, соответствующего текущей функции Select, а также был изменен конструктор, позволяя проинициализировать этот объект:

...

**CREF(LPIResourceType) getResType();**

private:

RDOFunCalcSelect(**CREF(LPIResourceType) pResType**,

int nResType, CREF(LPRDOCalc) pCondition);

**LPIResourceType m\_pResType;**

...

Далее уже созданный массив заполняется найденными ресурсами из хранящегося в m\_pSelect списка res\_list или же остается пустым, если таковых не найдено (итератор res\_list.begin() равен итератору res\_list.end()).

Сформированный массив, хранящийся в pValue, заносится в универсальный контейнер для значений в системе РДО – RDOValue.

* 1. **Доступ к параметру ресурса – элемента массива**
     1. **Изменения в файлах синтаксического анализатора**

Для обеспечения доступа к параметру ресурса из их массива по индексу и идентификатору параметра описание арифметического выражения в файлах грамматики Bison было дополнено следующим пунктом (См. 6.2.2):

fun\_arithm

...

| RDO\_IDENTIF '[' fun\_arithm ']' '.' RDO\_IDENTIF

{

LPRDOValue pArrayValue = PARSER->stack().pop<RDOValue>($1);

ASSERT(pArrayValue);

LPRDOFUNArithm pArrayArithm =

RDOFUNArithm::generateByIdentificator(pArrayValue);

ASSERT(pArrayArithm);

LPRDOArrayType pArrayType = pArrayArithm->

typeInfo()->type().object\_dynamic\_cast<RDOArrayType>();

if (!pArrayType)

{

PARSER->error().error(@1, rdo::format(\_T("'%s'

не является массивом")

, pArrayValue->value().getIdentificator().c\_str())

);

}

LPRDORTPResType pResType = pArrayType->getItemType()->

type().object\_dynamic\_cast<RDORTPResType>();

if (!pResType)

{

PARSER->error().error(@1, rdo::format(\_T("'%s' не является

массивом ресурсов")

, pArrayValue->value().getIdentificator().c\_str())

);

}

LPRDOFUNArithm pArrayIndex = PARSER->

stack().pop<RDOFUNArithm>($3);

ASSERT(pArrayIndex);

LPRDOValue pParamName = PARSER->stack().pop<RDOValue>($6);

ASSERT(pParamName);

rsint paramIndex = pResType->getRTPParamNumber(

pParamName->value().getAsString());

if (paramIndex == RDORTPResType::UNDEFINED\_PARAM)

{

PARSER->error().error(@6, rdo::format(\_T("'%s' не является

параметром ресурса '%s'")

, pParamName->value().getAsString().c\_str()

, pResType->name().c\_str())

);

}

rdo::runtime::LPRDOCalc pArrayItem =

rdo::Factory<rdo::runtime::RDOCalcArrayItem>::create(

pArrayArithm->calc(),

pArrayIndex->calc()

);

ASSERT(pArrayItem);

rdo::runtime::LPRDOCalc pParamValue =

rdo::Factory<rdo::runtime::RDOCalcGetResourceParam>::create(

pArrayItem, paramIndex

);

ASSERT(pParamValue);

LPExpression pParamExpression =

rdo::Factory<Expression>::create(

pResType->getParams()[paramIndex]->getTypeInfo(),

pParamValue,

RDOParserSrcInfo(@6)

);

ASSERT(pParamExpression);

LPRDOFUNArithm pParamArithm =

rdo::Factory<RDOFUNArithm>::create(pParamExpression);

ASSERT(pParamArithm);

$$ = PARSER->stack().push(pParamArithm);

}

Этот код содержит несколько проверок (является ли переменная массивом содержит ли массив ресурсы содержит ли ресурс параметр с заданным идентификатором), позволяющих отследить ошибки в коде анимации на этапе компиляции.

Также удалось реализовать задачу доступа, не прибегая к написанию собственных классов в пространстве имен rdo\_runtime, а лишь воспользоваться уже существующими **rdo::runtime::RDOCalcArrayItem** и **rdo::runtime::RDOCalcGetResourceParam**, возвращающими элемент массива в контейнере RDOValue и получающими из него параметр ресурса по его индексу соответственно.

1. **Апробирование разработанной системы для модельных условий**

Специально для тестирования разработанного функционала подсистемы анимации была написана имитационная модель торгового судна, перевозящего контейнеры.

Отображение контейнеров, особенно при большом их числе, а также большом числе их параметров, могло бы стать весьма непростой задачей при реализации отображения модели через постоянные ресурсы. Однако, с другой стороны, это стало подходящей задачей для разработанного метода getArray.

Приведенный ниже отрывок из кода модели показывает, что таким образом можно исключить дублирование и, следовательно, уделить больше внимания визуальным качествам кадра анимации модели.

array<Контейнеры>A=Select(Контейнеры :

Контейнеры.состояние != Заказан).getArray();

for ( i = 0; i < A.Size; i++ )

{// -------------- список активных контейнеров ---------------

R = A[i].cR; G = A[i].cG; B = A[i].cB;

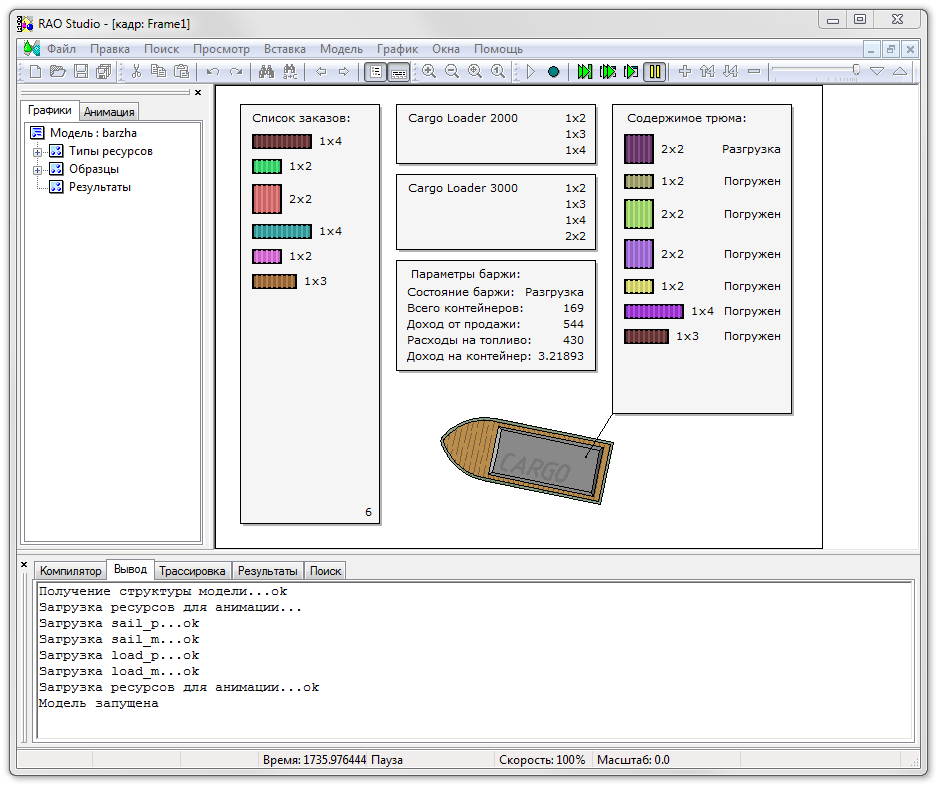
// код с отображением текстовой информации

// код для отрисовки самих контейнеров

curY+= 10 + A[i].ширина\*cSize ;

}// ----------------------------------------------------------

Таким образом, 8 условий на отображение каждого контейнера, находящегося в трюме, заменяются одним объявлением массива и одним циклом.



Более подробное описание преимуществ разработанного способа отображения находится на заключительном листе курсового проекта. Полный код имитационной модели представлен в **Приложении А**.

***Рисунок 4.***

1. **Заключение**

В рамках данного курсового проекта были получены следующие результаты:

* 1. Проведено предпроектное исследование системы имитационного моделирования РДО.
  2. На этапе концептуального проектирования системы выделены подзадачи, необходимые для выполнения основного задания.
  3. На этапе технического проектирования доработан синтаксис подсистемы анимации, который представлен на синтаксической диаграмме. С помощью диаграммы классов разработана архитектура новой системы.
  4. На этапе рабочего проектирования с помощью диаграммы активностей смоделирован механизм получения массива ресурсов и доступа к параметру ресурса как элемента массива. Написан программный код для реализации спроектированных ранее алгоритмов работы и архитектуры компонента rdo\_parser и rdo\_runtime системы РДО. Проведены отладка и тестирование нового функционала системы, в ходе которых исправлялись найденные ошибки.
  5. Для демонстрации новых возможностей системы была написана модель, позволяющая в полной мере показать реализованный функционал.
  6. Все внесенные в систему изменения отражены в справочной информации по системе РДО, что позволяет пользователям оперативно получать справку по новым функциям системы.

1. **С****писок используемых источников**
2. Справка по языку РДО [<http://rdo.rk9.bmstu.ru/help/>];
3. Справка по RAO-studio [<http://rdo.rk9.bmstu.ru/help/>];
4. Емельянов В. В., Ясиновский С. И. Имитационное моделирование систем: Учеб. Пособие – М.: Издательство МГТУ им. Н. Э. Баумана, 2009. – 584 с.: ил. (Информатика в техническом университете);
5. Мартин Р. Чистый код. Создание, анализ и рефакторинг / пер. с анг. Е. Матвеев – CПб.: Питер, 2010. – 464 стр.;
6. Единая система программной документации. Техническое задание. Требования к содержанию и оформлению. ГОСТ 19.201-78;
7. Б. Страуструп Язык программирования С++. Специальное издание / пер. с англ. – М.: ООО «»Бином-Пресс, 2006. – 1104 с.: ил.;
8. Чарльз Доннелли. Ричард Столлман. Bison. Генератор синтаксических анализаторов, совместимый с YACC. / пер. с анг. [<http://www.opennet.ru/docs/RUS/bison_yacc/>];
9. Единая система программной документации. Схемы алгоритмов, программ, данных и систем. ГОСТ 19.701-90. Условные обозначения и правила выполнения.

**Список использованного программного обеспечения**

* + - 1. RAO-Studio;
      2. ArgoUML v0.34;
      3. Autodesk®, Inc., AutoCAD® 2012;
      4. Inkscape 0.48.2;
      5. Microsoft®, Office Word 2010 SP1;
      6. TortoiseSVN;
      7. Microsoft®, Visual Studio 2008 SP1.

1. **Приложение А. Код имитационной модели**

**Вкладка RTP**

$Resource\_type Баржи: permanent

$Parameters

состояние : (Погрузка, Отплывает, Плавание, Разгрузка)

доход : real

количество\_контейнеров: integer

всего\_контейнеров : integer

грузится\_контейнеров : integer

количество\_плаваний : integer

расходы\_на\_топливо : integer

доход\_на\_контейнер : real

$End

$Resource\_type Контейнеры : temporary

$Parameters

состояние : (Заказан, Погрузка, Погружен, Разгрузка)

стоимость\_перевозки : real

ширина : integer

длина : integer

cR : integer

cG : integer

cB : integer

погрузчик : integer = 0

$End

$Resource\_type Погрузчики: permanent

$Parameters

состояние : ( Свободен, Занят ) = Свободен

длительность\_min : real

длительность\_max : real

тип\_контейнера : integer

погружено : integer

$End

**Вкладка RSS**

$Resources

Баржа : Баржи trace Погрузка 0 0 0 0 0 0 0

Погрузчик1 : Погрузчики trace \* 5 8 1 0

Погрузчик2 : Погрузчики trace \* 3 5 2 0

Разгрузчик : Погрузчики trace \* 1 2 0 0

$End

**Вкладка EVN**

$Pattern Заказ\_на\_контейнер : event

$Relevant\_resources

\_Баржа : Баржа Keep

\_Контейнер : Контейнеры Create

$Body

\_Баржа

Convert\_event

Заказ\_на\_контейнер.planning(time\_now + Интервал\_прихода(9));

\_Контейнер

Convert\_event trace

состояние = Заказан;

integer c = rand(0, 100);

if (c>75)

ширина = 2;

else

ширина = 1;

if( \_Контейнер.ширина == 2 )

длина = 2;

else

длина = rand(2, 5);

стоимость\_перевозки = \_Контейнер.ширина\*\_Контейнер.длина;

cR = 50\*int(rand(1, 5));

cG = 50\*int(rand(1, 5));

cB = 50\*int(rand(1, 5));

$End

**Вкладка PAT**

$Pattern Образец\_погрузки : operation

$Relevant\_resources

\_Баржа : Баржи Keep Keep

\_Погрузчик : Погрузчики Keep Keep

\_Контейнер : Контейнеры Keep Keep

$Time = Длительность\_обслуживания( \_Погрузчик.длительность\_min, \_Погрузчик.длительность\_max )

$Body

\_Баржа

Choice from \_Баржа.состояние == Погрузка and \_Баржа.грузится\_контейнеров + \_Баржа.количество\_контейнеров < 8

Convert\_begin

грузится\_контейнеров++;

Convert\_end

количество\_контейнеров++;

грузится\_контейнеров--;

if (\_Баржа.количество\_контейнеров >= 8)

{

состояние = Отплывает;

расходы\_на\_топливо+=10;

}

\_Погрузчик

Choice from \_Погрузчик.состояние == Свободен

with\_max ( \_Погрузчик.тип\_контейнера )

Convert\_begin

состояние = Занят;

Convert\_end

состояние = Свободен;

погружено++;

\_Контейнер

Choice from \_Контейнер.состояние == Заказан and \_Контейнер.ширина <= \_Погрузчик.тип\_контейнера

first

Convert\_begin

состояние = Погрузка;

погрузчик = \_Погрузчик.тип\_контейнера;

Convert\_end

состояние = Погружен;

погрузчик = 0;

$End

$Pattern Образец\_плавания : operation

$Relevant\_resources

\_Баржа : Баржи Keep Keep

$Time = Длительность\_обслуживания( 6, 8 )

$Body

\_Баржа

Choice from \_Баржа.состояние == Отплывает

Convert\_begin

состояние = Плавание;

Convert\_end

if (\_Баржа.количество\_контейнеров == 0)

состояние = Погрузка;

else

состояние = Разгрузка;

$End

$Pattern Образец\_разгрузки : operation

$Relevant\_resources

\_Баржа : Баржи NoChange Keep

\_Контейнер : Контейнеры Keep Erase

\_Разгрузчик : Погрузчики Keep Keep

$Time = Длительность\_обслуживания( 4, 5 )

$Body

\_Баржа

Choice from \_Баржа.состояние == Разгрузка

Convert\_end

количество\_контейнеров--;

доход+=\_Контейнер.стоимость\_перевозки;

всего\_контейнеров++;

if (\_Баржа.количество\_контейнеров == 0)

{

состояние = Отплывает;

расходы\_на\_топливо+=10;

}

доход\_на\_контейнер = \_Баржа.доход / \_Баржа.всего\_контейнеров;

\_Контейнер

Choice from \_Контейнер.состояние == Погружен

Convert\_begin

состояние = Разгрузка;

\_Разгрузчик

Choice from \_Разгрузчик.состояние == Свободен and \_Разгрузчик.тип\_контейнера == 0

Convert\_begin

состояние = Занят;

Convert\_end

состояние = Свободен;

погружено++;

$End

**Вкладка DPT**

$Decision\_point load : some

$Condition NoCheck

$Activities

Погрузка : Образец\_погрузки

$End

$Decision\_point sail : some

$Condition NoCheck

$Activities

Плавание : Образец\_плавания

$End

$Decision\_point unload : some

$Condition NoCheck

$Activities

Разгрузка : Образец\_разгрузки

$End

**Вкладка FRM**

$Frame Frame1

$Back\_picture = white 608 464

integer i,j;

integer bx = 214, by = 269; // условная точка отсчета отрисовки

integer curY = by - 250 + 8, cSize = 15, contX = bx + 183 + 12,

ordX = bx - 189 + 12, ordY = by - 250 + 8;

if(Баржа.состояние == Отплывает or Баржа.состояние == Плавание)

bitmap (bx - 25, by + 50, sail\_p, sail\_m);

else

bitmap (bx, by, load\_p, load\_m);

rect (bx + 186, by - 247, 180, 310, <180,180,180>, transparent);

rect (bx + 183, by - 250, 180, 310, <245,245,245>, black); // список активных контейнеров

text(bx + 183 + 15, curY, 160, 12, transparent, black, 'Содержимое трюма:');

curY+= 12 + 10;

rect (bx - 186, by - 247, 140, 420, <180,180,180>, transparent);

rect (bx - 189, by - 250, 140, 420, <245,245,245>, black); // список заказов

text(ordX, ordY, 160, 12, transparent, black, 'Список заказов:');

ordY+= 12 + 10;

rect (bx - 30, by - 247, 200, 60, <180,180,180>, transparent);

rect (bx - 33, by - 250, 200, 60, <245,245,245>, black); // погрузчик 1

text(bx - 21, by - 242, 163, 12, transparent, black, 'Cargo Loader 2000');

text(bx - 16, by - 242, 173, 12, transparent, black, > '1х2');

text(bx - 16, by - 226, 173, 12, transparent, black, > '1х3');

text(bx - 16, by - 210, 173, 12, transparent, black, > '1х4');

rect (bx - 30, by - 177, 200, 76, <180,180,180>, transparent);

rect (bx - 33, by - 180, 200, 76, <245,245,245>, black); // погрузчик 2

text(bx - 21, by - 172, 163, 12, transparent, black, 'Cargo Loader 3000');

text(bx - 16, by - 172, 173, 12, transparent, black, > '1х2');

text(bx - 16, by - 156, 173, 12, transparent, black, > '1х3');

text(bx - 16, by - 140, 173, 12, transparent, black, > '1х4');

text(bx - 16, by - 124, 173, 12, transparent, black, > '2х2');

rect (bx - 30, by - 91, 200, 111, <180,180,180>, transparent); // все параметры баржи

rect (bx - 33, by - 94, 200, 111, <245,245,245>, black);

text(bx - 18, by - 86, 173, 12, transparent, black, 'Параметры баржи:');

text(bx - 22, by - 68, 173, 12, transparent, black, 'Состояние баржи:');

text(bx - 28, by - 68, 183, 12, transparent, black, > Баржа.состояние);

text(bx - 22, by - 52, 173, 12, transparent, black, 'Всего контейнеров:');

text(bx - 28, by - 52, 183, 12, transparent, black, > Баржа.всего\_контейнеров);

text(bx - 22, by - 36, 173, 12, transparent, black, 'Доход от продажи:');

text(bx - 28, by - 36, 183, 12, transparent, black, > Баржа.доход);

text(bx - 22, by - 20, 173, 12, transparent, black, 'Расходы на топливо:');

text(bx - 28, by - 20, 183, 12, transparent, black, > Баржа.расходы\_на\_топливо);

text(bx - 22, by - 4, 173, 12, transparent, black, 'Доход на контейнер:');

text(bx - 28, by - 4, 183, 12, transparent, black, > Баржа.доход\_на\_контейнер);

integer R, G, B;

integer boundary\_flag = 1;

array<Контейнеры>A=Select(Контейнеры : Контейнеры.состояние != Заказан).getArray();

for ( i = 0; i < A.Size; i++ )

{// --------------- список активных контейнеров ------------------

R = A[i].cR; G = A[i].cG; B = A[i].cB;

text(contX + A[i].длина\*cSize + 4, curY + A[i].ширина\*0.5\*cSize - 6, 10,

12, transparent, black, > A[i].ширина);

text(contX + A[i].длина\*cSize + 12, curY + A[i].ширина\*0.5\*cSize - 6, 10,

12, transparent, black, > 'x');

text(contX + A[i].длина\*cSize + 20, curY + A[i].ширина\*0.5\*cSize - 6, 10,

12, transparent, black, > A[i].длина);

text(contX + 4, curY + A[i].ширина\*0.5\*cSize - 6, 153, 12, transparent,

black, > A[i].состояние);

rect (contX, curY, A[i].длина\*cSize, A[i].ширина\*cSize, <R, G, B>, black);

rect (contX + 1, curY + 1, A[i].длина\*cSize - 2, A[i].ширина\*cSize - 2,

<R, G, B>, black);

for ( j = 0; j < A[i].длина\*3 - 1; j++ )

rect(contX + 5\*(j + 1) - 1, curY + 1, 2, A[i].ширина\*cSize - 2,

transparent , <R + 30, G + 30, B + 30>);

curY+= 10 + A[i].ширина\*cSize ;

}// --------------------------------------------------------------

array<Контейнеры> Z=Select(Контейнеры : Контейнеры.состояние == Заказан or

Контейнеры.состояние == Погрузка).getArray();

for ( i = 0; i < Z.Size; i++ )

{

if (ordY + Z[i].ширина\*cSize < by + 163)

{//------------------- список заказов ------------------------

text(ordX + Z[i].длина\*cSize + 4, ordY + Z[i].ширина\*

\*0.5\*cSize - 6, 10, 12, transparent, black, > Z[i].ширина);

text(ordX + Z[i].длина\*cSize + 12, ordY + Z[i].ширина\*

\*0.5\*cSize - 6, 10, 12, transparent, black, > 'x');

text(ordX + Z[i].длина\*cSize + 20, ordY + Z[i].ширина\*

\*0.5\*cSize - 6, 10, 12, transparent, black, > Z[i].длина);

if (Z[i].погрузчик != 0)

{

text(ordX + 4, ordY + Z[i].ширина\*0.5\*cSize - 6,

104, 12, transparent, black, > 'П');

text(ordX + 108, ordY + Z[i].ширина\*0.5\*cSize - 6, 10, 12,

transparent, black, < Z[i].погрузчик);

}

R = Z[i].cR; G = Z[i].cG; B = Z[i].cB;

rect (ordX, ordY, Z[i].длина\*cSize, Z[i].ширина\*cSize, <R, G, B>, black);

rect (ordX + 1, ordY + 1, Z[i].длина\*cSize - 2, Z[i].ширина\*cSize - 2,

<R, G, B>, black);

for ( j = 0; j < Z[i].длина\*3 - 1; j++ )

rect(ordX + 5\*(j + 1) - 1, ordY + 1, 2, Z[i].ширина \*cSize

- 2, transparent ,<R + 30, G + 30, B + 30>);

ordY+= 10 + Z[i].ширина\*cSize ;

}// ----------------------------------------------------------

else

if (boundary\_flag == 1)

{//---------- выход за пределы списка заказов -----------

rect (ordX + 15, by + 162, 2, 2, black, black);

rect (ordX + 10, by + 162, 2, 2, black, black);

rect (ordX + 5, by + 162, 2, 2, black, black);

boundary\_flag = 0;

}// -----------------------------------------------------

}

text (ordX + 100, by + 152, 20, 12, transparent, black, > i);

array<Контейнеры> P=Select(Контейнеры :

Контейнеры.состояние == Погрузка).getArray();

for ( i = 0; i < P.Size; i++ )

{

R = P[i].cR; G = P[i].cG; B = P[i].cB;

if(P[i].состояние == Погрузка)

if (P[i].погрузчик == 1)

{//------------- состояние погрузчика 1 -----------------

text(bx - 19, by - 243 + 16\*(P[i].длина - 2), 153,

12, transparent, black, > 'o');

text(bx - 14, by - 217, 100, 12, transparent, black, 'Погрузка:');

rect (bx + 53, by - 218, P[i].длина\*cSize, 15, <R, G, B>, black);

rect (bx + 54, by - 217, P[i].длина\*cSize - 2, 13, <R, G, B>, black);

for ( j = 0; j < P[i].длина\*3 - 1; j++ )

rect(bx + 57 + 5\*j, by - 217, 2, P[i].ширина\*cSize - 2,

transparent ,<R + 30, G + 30, B + 30>);

}// -----------------------------------------------------

else

{//------------- состояние погрузчика 2 -----------------

text(bx - 19, by - 250 + 70 + 7 + 16\*(P[i].длина - 2 + (P[i].ширина

- 1)\*3), 153, 12, transparent, black, > 'o');

text(bx - 14, by - 245 + 70 + 28 + 8, 100, 12, transparent, black,

'Погрузка:');

rect (bx + 53, by - 133 - 0.5\*P[i].ширина\*cSize, P[i].длина\*cSize,

P[i].ширина\*cSize, <R, G, B>, black);

rect (bx + 54, by - 132 - 0.5\*P[i].ширина\*cSize, P[i].длина\*cSize –

2, P[i].ширина\*cSize - 2, <R, G, B>, black);

for ( j = 0; j < P[i].длина\*3 - 1; j++ )

rect(bx + 57 + 5\*j, by - 132 - 0.5\*P[i].ширина\*cSize, 2,

P[i].ширина\*cSize - 2, transparent ,<R + 30, G + 30, B + 30>);

}// -----------------------------------------------------

}

$End

**Вкладка FUN**

$Sequence Интервал\_прихода : real

$Type = exponential 123456789

$End

$Sequence Длительность\_обслуживания : real

$Type = uniform 123456789

$End

$Sequence rand : integer

$Type = uniform 12345678

$End

**Вкладка SMR**

Show\_mode = Animation

Frame\_number = 1

Show\_rate = 8000.0

Заказ\_на\_контейнер.planning( time\_now + 4 )

Terminate\_if Time\_now >= 12 \* 7 \* 6000