Оглавление

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

2. 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.

3. Предпроектное исследование

3.1. 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 (с расширением .rtp);
- ресурсы (с расширением .rss);
- события (с расширением .evn);
- образцы активностей (с расширением .pat);

- точки принятия решений и процессы обслуживания (с расширением .dpt);
- константы, функции и последовательности (с расширением .fun);
- кадры анимации (с расширением .frm);
- требуемая статистика (с расширением .pmd);
- прогон (с расширением .smr);
- процессы обслуживания (с расширением .prc).

Объекты, создаваемые РДО-имитатором при выполнении прогона:

- результаты (с расширением .pmv);
- трассировка (с расширением .trc). [2]

3.2. 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>
[ Show_if <условие_показа_кадра> ]
$Back_picture = <описание_фоновой_картинки>
[ <описание_элементов_отображения> ]
$End
```

имя_кадра

Имя кадра представляет собой простое имя. 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. 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> [ <свойства_элемента> ]
```

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.

3.3. 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, но проверяются не все, ресурсы, а только отобранные в список. Формат использования: Exist(<логическое_выражение> | NoCheck)	Логическое
Not_Exist	Описание аналогично приведённому, 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]

3.4. 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).

4. Формирование ТЗ

4.1. Основания для разработки

Задание на курсовой проект.

4.2. 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.

4.3. 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.

4.4. Характеристики объекта автоматизации

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

4.5. Требования к системе

4.5.1. 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.

4.5.2. Требования к надежности

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

4.5.3. Условия эксплуатации

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

4.5.4. 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 с разрешением от 800*600 и выше;

4.5.5. Требования к информационной и программной совместимости

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

4.5.6. Требования к маркировке и упаковке

Не предъявляются.

4.5.7. Требования к транспортированию и хранению

Не предъявляются.

4.5.8. Порядок контроля и приемки

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

5. 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**).

5.1. Диаграмма компонентов

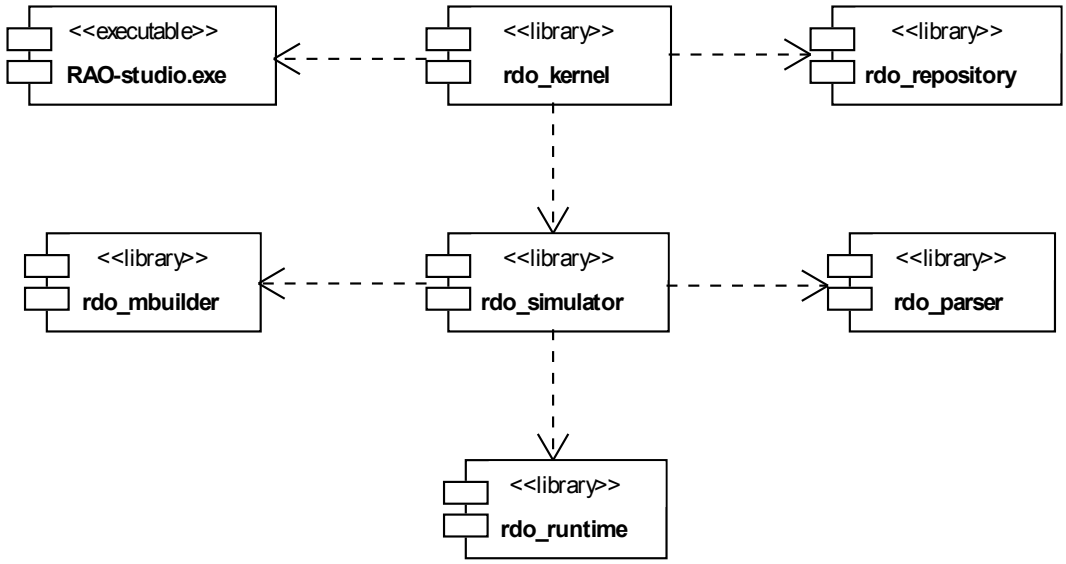

Рис. 1. Упрощенная диаграмма компонентов

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

rdo_kernel реализует ядровые функции системы. Не изменяется при разработке

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.

6. Технический этап проектирования системы

6.1. 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).

6.2. 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*.

6.2.1. 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.

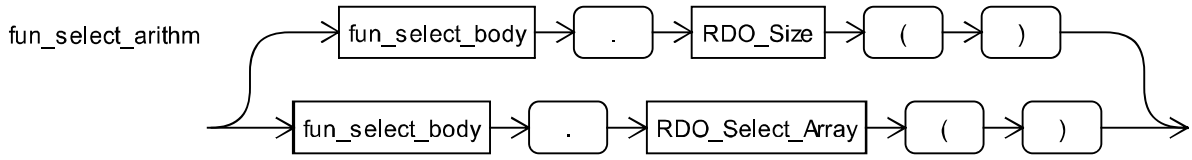

Рисунок 2. Синтаксическая диаграмма арифметического выражения функции Select

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

6.2.2. 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 C++.

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:

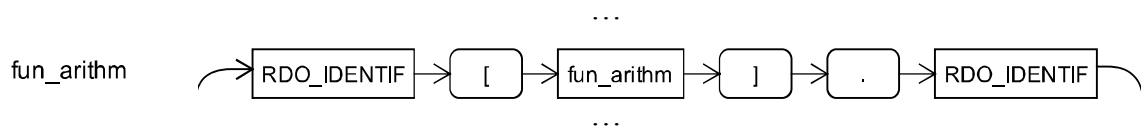

Рисунок 3. Часть синтаксической диаграммы арифметического выражения

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

6.2.3. 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 C++.

6.2.4. 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`.

7. 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 C++. 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).

7.1. Получение массива ресурсов

7.1.1. 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.

7.1.2. 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.

7.1.3. 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`.

7.2. Доступ к параметру ресурса – элемента массива

7.2.1. 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.

8. 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.

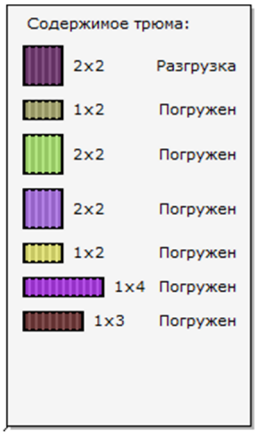

Рисунок 4.

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

9. 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.

10. Список используемых источников

1. Справка по языку РДО [http://rdo.rk9.bmstu.ru/help/];
2. Справка по RAO-studio [http://rdo.rk9.bmstu.ru/help/];
3. 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);
4. Мартин Р. Чистый код. Создание, анализ и рефакторинг / пер. с анг. Е. Матвеев – СПб.: Питер, 2010. – 464 стр.;
5. Единая система программной документации. Техническое задание. Требования к содержанию и оформлению. ГОСТ 19.201-78;
6. Б. Страуструп Язык программирования C++. Специальное издание / пер. с англ. – М.: ООО «»Бином-Пресс, 2006. – 1104 с.: ил.;
7. Чарльз Доннелли. Ричард Столлман. Bison. Генератор синтаксических анализаторов, совместимый с YACC. / пер. с анг. [http://www.opennet.ru/docs/RUS/bison_yacc/];
8. 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.

11. 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, > '1x2');
text(bx - 16, by - 226, 173, 12, transparent, black, > '1x3');
text(bx - 16, by - 210, 173, 12, transparent, black, > '1x4');
```

```
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, > '1x2');
text(bx - 16, by - 156, 173, 12, transparent, black, > '1x3');
text(bx - 16, by - 140, 173, 12, transparent, black, > '1x4');
text(bx - 16, by - 124, 173, 12, transparent, black, > '2x2');

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
```