Санкт-Петербургский государственный университет
информационных технологий, механики и оптики

Кафедра «Компьютерные технологии»

А.В. Хокканен, А.А. Шалыто

Имитатор игрового автомата класса «Однорукий бандит»

Объектно-ориентированное программирование
с явным выделением состояний

Проектная документация

Проект создан в рамках
«Движения за открытую проектную документацию»
http://is.ifmo.ru

Санкт-Петербург
2003

Содержание

Введение

Как показано в настоящей работе, SWITCH-технология, предложенная А.А. Шалыто [1] и развитая им совместно с Н.И. Туккелем [2], является, пожалуй, наиболее естественным решением для широкого класса задач управления событийными системами. Поэтому эта технология является оптимальной для решения задач построения имитаторов подобных систем.

Цель настоящей работы состоит в разработке имитатора игрового автомата класса "Однорукий бандит" на основе SWITCH-технологии.

Более подробно ознакомиться с этой технологией можно на сайтах http://is.ifmo.ru и http://www.softcraft.ru.

Программа создана с помощью системы разработки *Delphi 7.0*. Исходный код программы приведен в Приложении.

1. Постановка задачи

Целью данного проекта является создание имитатора игрового автомата класса "Однорукий бандит". Данный имитатор должен удовлетворять следующим требованиям.

1. Игра производится на барабанах с разметкой от 1 до 9.
2. В игре используются три барабана.
3. Имитатор должен обеспечивать современную защиту от взлома на повторение выигрышных комбинаций. Для этого каждый из барабанов должен иметь свой собственный таймер на остановку.
4. Выигрышные комбинации и размеры выплат в случае их выпадения определяются по следующим правилам:
   - три семерки – ставка умножается в 10 раз;
   - три других одинаковых числа – ставка умножается в 5 раз;
   - три подряд идущих числа – ставка умножается в 7 раз;
   - два одинаковых числа – ставка умножается в 4 раза;
   - первое число – семерка – ставка увеличивается на один жетон;
   - первое число – нечетное – один жетон.
5. В имитаторе должен учитываться факт того, что резервуар для хранения жетонов имеет ограниченный объем. Как следствие, соответствующим образом должна быть реализована обработка ситуации, когда у автомата не хватает жетонов на выдачу выигрыша.
6. Имитатор должен соответствующим образом обрабатывать ситуацию опускания в игровой автомат поддельного жетона.

На рис. 1 приведен пример окна работающей программы, которое разделено на три панели: игровую, ведения протокола и настройки ведения протокола.

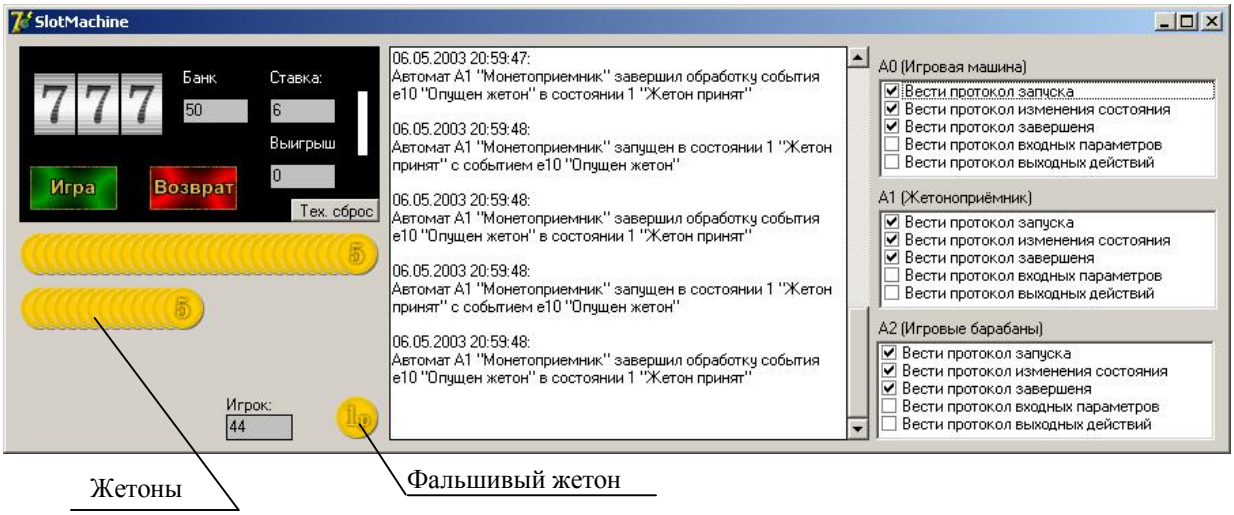

Рис. 1. Пример окна работающей программы

Работа с программой осуществляется следующим образом. Сначала в автомат опускаются жетоны, что имитируется с помощью щелчка мыши на кнопке с изображением жетона. В правом нижнем углу игровой панели находится изображение фальшивого жетона. При его опускании срабатывает защита автомата – он перестает принимать жетоны. При этом белая щель монетоприемника становится заштрихованной. В этом случае фальшивый жетон и опущенные ранее нормальные жетоны можно вернуть с помощью кнопки «Возврат»

После того, как игрок сделал ставку, можно начать игру с помощью кнопки «Игра». При этом барабаны начнут раскручиваться. При повторном нажатии на эту кнопку начнется процесс остановки барабанов.

В результате игрок либо выиграет, либо проиграет. В случае выигрыша начнется процесс выдачи жетонов из банка автомата, причем жетонов может не хватить. В реальных условиях в таком случае подразумевается, что игрок получит свой выигрыш в кассе. В том случае, если жетонов не хватит, для того чтобы игровой автомат смог продолжить работу, необходимо нажать кнопку «Тех. Сброс».

2. Диаграмма классов

В верхней части диаграммы (рис. 2) указаны классы, предоставляемые библиотекой *Borland VCL*. В нижней части приведены классы, созданные при проектировании программы.

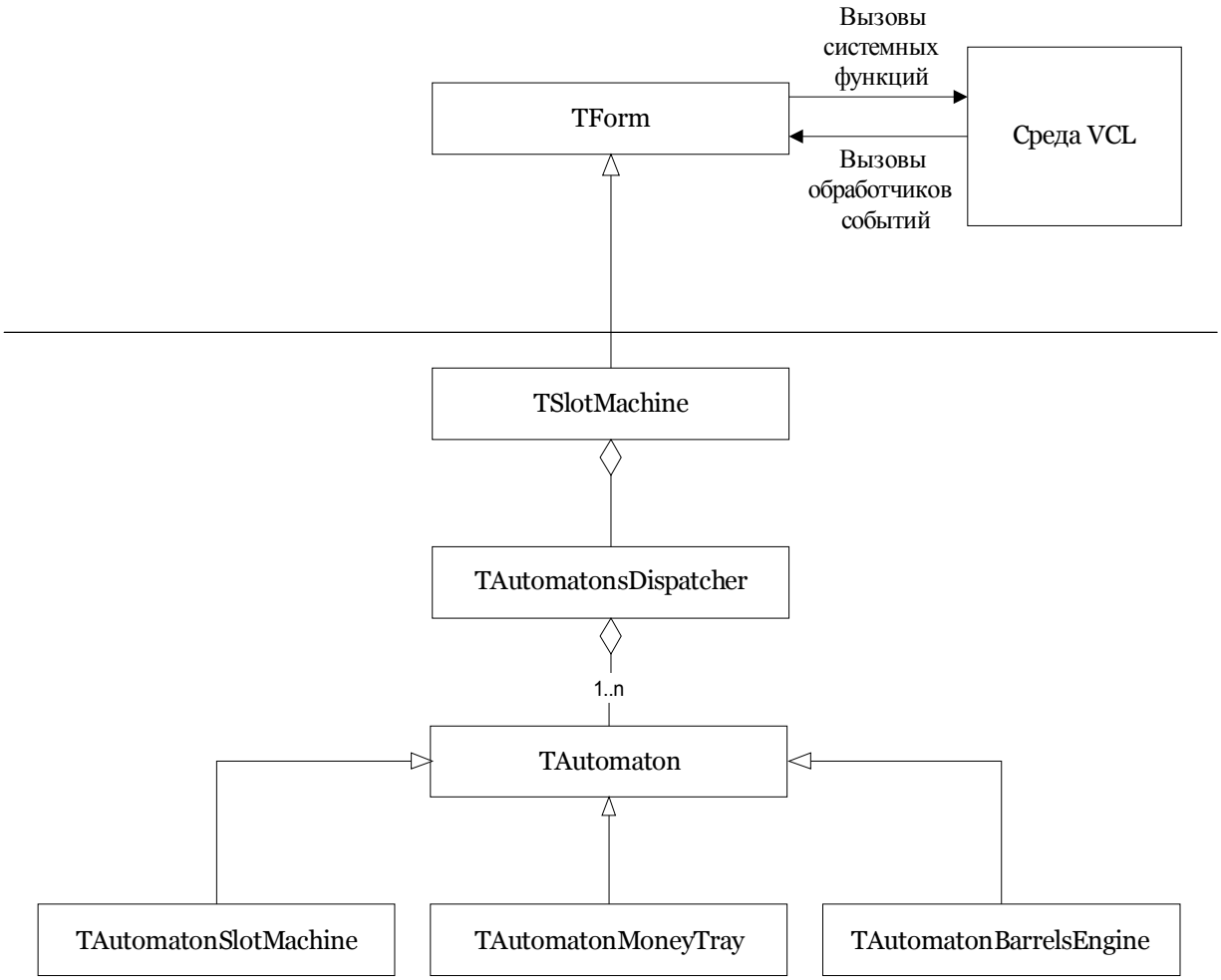

Рис. 2. Диаграмма классов

Приведем краткое описание классов, указанных на диаграмме:

- ***TForm*** – обеспечивает организацию визуального взаимодействия с пользователем;
- ***TSlotMachine*** **–** содержит обработчики событий, а также необходимый набор параметров для описания имитируемой игровой машины и набор методов, реализующих функции опроса и установки этих параметров;
- ***TAutomatonsDispatcher*** **–** содержит коллекцию объектов *TAutomaton*. Организует взаимодействие между автоматами;
- ***TAutomaton*** **–** базовый класс для реализации автоматов. Вызывает функцию записи в протокол и реализует проверку на реентерабельность. Под реентерабельностью понимается невозможность многократного запуска автомата в течение одного программного цикла;
- ***TAutomatonSlotMachine*** **–** содержит автомат «Игровая машина» (A0);
- ***TAutomatonMoneyTray*** **–** содержит автомат «Монетоприемник» (A1);
- ***TAutomatonBarrelsEngine*** **–** содержит автомат «Игровые барабаны» (A2).

3. Класс «Игровой автомат» (TSlotMachine)

3.1. Словесное описание

Класс *TSlotMachine* является наследником класса *TForm* и, с одной стороны, обеспечивает визуальный интерфейс с пользователем, а с другой – содержит в себе параметры, соответствующие имитируемому игровому автомату, например, количество жетонов в банке, текущую ставку и т. д.

3.2. Краткое описание методов

- *FormCreate* – конструктор окна приложения. Здесь производится инициализация параметров игрового автомата;
- *cbA0LogSettingsClickCheck* – обработчик событий изменения настроек протоколирования для автомата «Игровая машина» (A0);
- *cbA1LogSettingsClickCheck* – обработчик событий изменения настроек протоколирования для автомата «Монетоприемник» (A1);
- *cbA2LogSettingsClickCheck* – обработчик событий изменения настроек протоколирования для автомата «Игровые барабаны» (A2);
- *Image1Click* (e10) – обработчик события «Опущен жетон». Вызывает автомат A1 с событием e10;
- *IReturnClick* (e02) – обработчик события «Нажата кнопка “Возврат”». Вызывает автомат A0 с событием e02;
- *iPlayClick* (e01) – обработчик события «Нажата кнопка “Игра”». Вызывает автомат A0 с событием e01;
- *TimerSync1Timer* – обработчик неавтоматного события срабатывания системного таймера для отображения анимации крутящегося барабана N1;
- *TimerSync2Timer* – обработчик неавтоматного события срабатывания системного таймера для отображения анимации крутящегося барабана N2;
- *TimerSync3Timer* – обработчик неавтоматного события срабатывания системного таймера для отображения анимации крутящегося барабана N3;

- *TimerB1Timer* (e22) – обработчик события «Сработал таймер остановки первого барабана». Вызывает автомат A2 с событием e22;
- *TimerB2Timer* (e23) – обработчик события «Сработал таймер остановки второго барабана». Вызывает автомат A2 с событием e23;
- *TimerB3Timer* (e05) – обработчик события «Сработал таймер остановки третьего барабана». Вызывает автомат A0 с событием e05;
- *JackTimerTimer* (e06) – обработчик события «Сработал системный таймер выдачи очередного жетона в случае выигрыша игрока». Вызывает автомат A0 с событием e06;
- *iBadCoinClick* (e10) – обработчик события «Опущен фальшивый жетон». Вызывает автомат A1 с событием e10;
- *btnTechResetClick* (e03) – обработчик события «Нажата кнопка “Тех. сброс”». Вызывает автомат A0 с событием e03.

4. Класс «Диспетчер автоматов» (TAutomatonsDispatcher)

4.1. Словесное описание

Класс *TAutomatonsDispatcher* позволяет организовать защищенное взаимодействие между двумя автоматами. Например, он не позволяет программисту напрямую изменять состояние одного автомата из другого, минуя стандартные автоматные механизмы (вызовы автоматов с использованием событий). Для межавтоматного взаимодействия доступны два вида действий: опрос состояния автомата и запуск автомата с заданным событием.

4.2. Краткое описание методов

- *AddAutomaton* – производит добавление автомата в список управляемых диспетчером. Этому автомату диспетчер устанавливает ссылку на себя, для того, чтобы один автомат мог взаимодействовать с другим.

5. Класс «Автомат» (TAutomaton)

5.1. Словесное описание

Класс *TAutomaton* является базовым классом, на основе которого производится реализация автоматов. Этот класс осуществляет запуск, выполнение и завершение работы автомата. Для удобства и повышения эффективности программирования и отладки в базовом классе осуществляются такие действия, как проверка на реентерабельность автомата.

Проверка реентерабельности позволяет быстро обнаружить ошибки с зацикленными вызовами автоматов, а также реализовать базовые функции протоколирования работы этого автомата, включая протоколирование возможных ошибок. Функции вывода сообщений в протокол работы реализуются в потомках данного класса.

5.2. Краткое описание методов

- *Init* – конструктор автомата. Устанавливает его название и словесное описание;
- *Wake* – вызов автомата. В качестве параметра принимает номер события. Именно в этом методе осуществляется проверка на реентерабельность и осуществляется базовое протоколирование работы автомата. Данный метод последовательно вызывает реализуемые наследниками методы *HandleState* и *EnterState*;
- *HandleState* – метод, переопределяемый наследниками. В нем должен быть реализован первый оператор *switch* из шаблона для реализации автомата, предложенного в работе [2]. В данном методе реализуются действия, которые необходимо выполнить при получении автоматом события и определить его новое состояние;
- *EnterState* – метод, переопределяемый наследниками. В этом методе должен быть реализован второй оператор *switch* из указанного шаблона. В данном методе реализуются действия, которые необходимо произвести в случае, когда автомат после выполнения метода *HandleState* изменил свое состояние;
- *Log* – метод, переопределяемый наследниками. В этом методе наследники должны реализовать способ записи протокола (в файл, на экран и т.п.).

6. Автоматный класс «Игровой автомат» (TAutomatonSlotMachine)

6.1. Словесное описание

Класс *TAutomatonSlotMachine* является автоматным классом, порожденным от класса *TAutomaton*. Описание автомата «Игровая машина» (A0), который реализует данный класс, приводится ниже.

6.2. Автомат «Игровая машина» (A0)

6.2.1. Словесное описание

Данный автомат описывает поведение имитируемой игровой машины. Он обеспечивает получение жетонов, начало игры, выдачу выигрыша и работу в ситуации возникновения ошибки. Схема связей автомата «Игровая машина» (A0) приведена на рис. 3, а граф переходов — на рис. 4.

6.2.2. Схема связей

Рис. 3. Схема связей автомата «Игровая машина» (A0)

6.2.3. Граф переходов

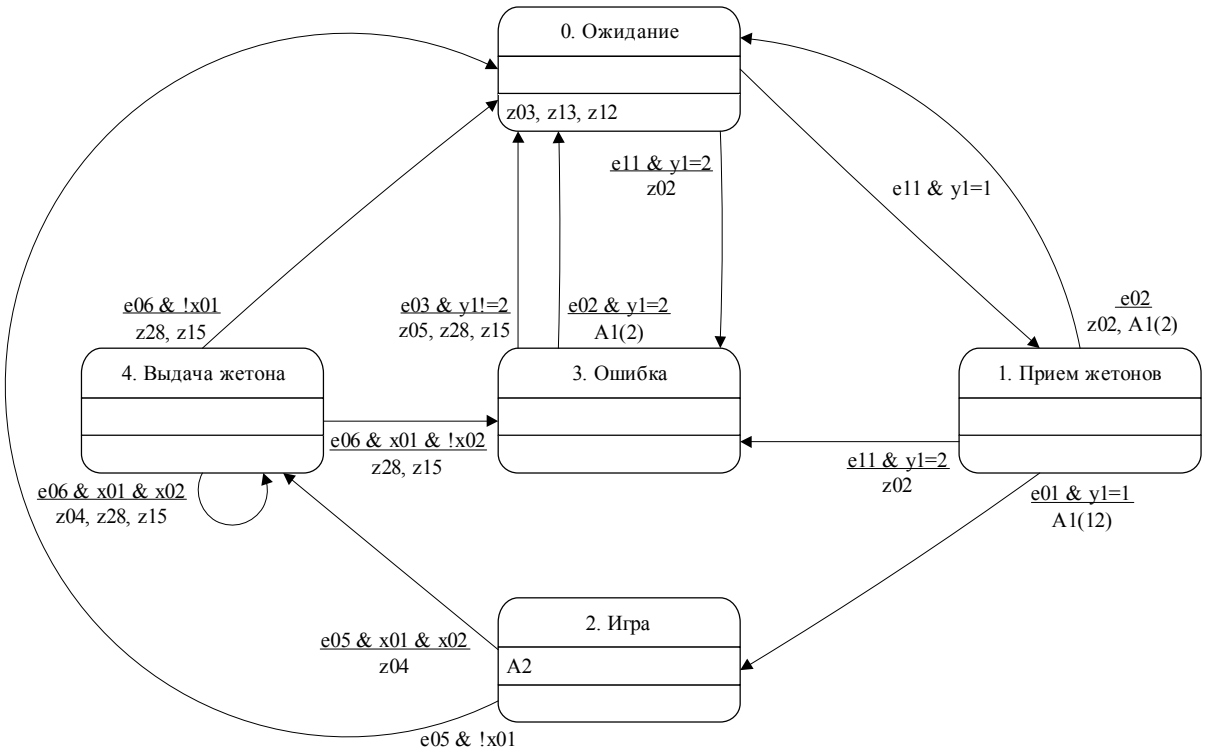

Рис. 4. Граф переходов автомата «Игровая машина» (A0)

7. Автоматный класс «Жетоноприемник» (TAutomatonMoneyTray)

7.1. Словесное описание

Класс *TAutomatonMoneyTray* является автоматным классом, порожденным от класса *TAutomaton*. Описание автомата «Монетоприемник» (A1), который реализует данный класс, приводится ниже.

7.2. Автомат «Монетоприемник» (A1)

7.2.1. Словесное описание

Данный автомат описывает поведение монетоприемника в имитируемой игровой машине. Он обеспечивает обработку получения жетонов, их распознавание, возврат жетонов по запросу пользователя, размещение жетонов из приемника в банк, а также сообщает автомату «Игровая машина» (A0) о получении очередного жетона. Схема

связей автомата «Монетоприемник» (A1) приведена на рис. 5, а граф переходов - на рис. 6.

7.2.2. Схема связей

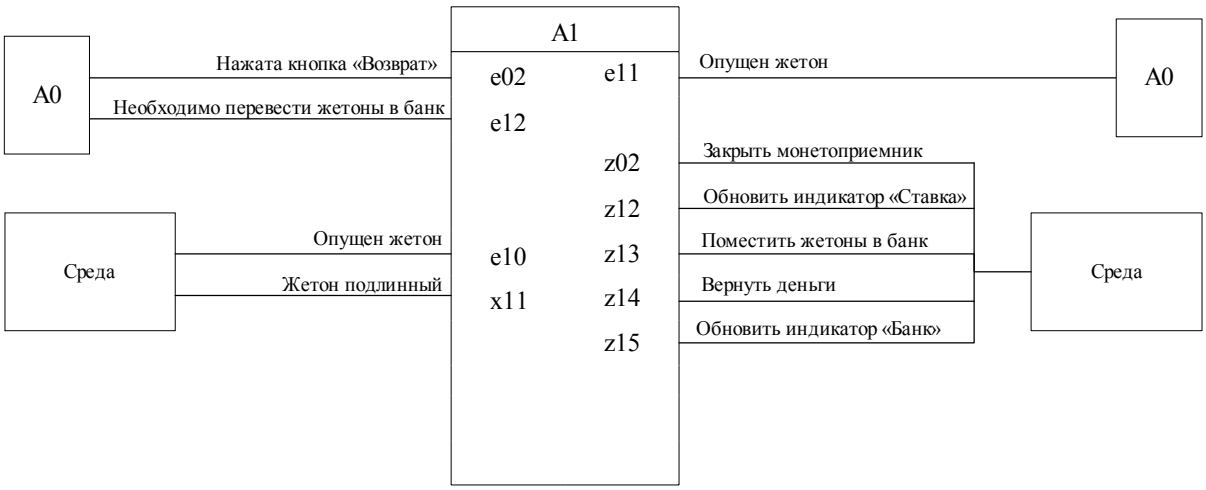

Рис. 5. Схема связей автомата «Монетоприемник» (A1)

7.2.3. Граф переходов

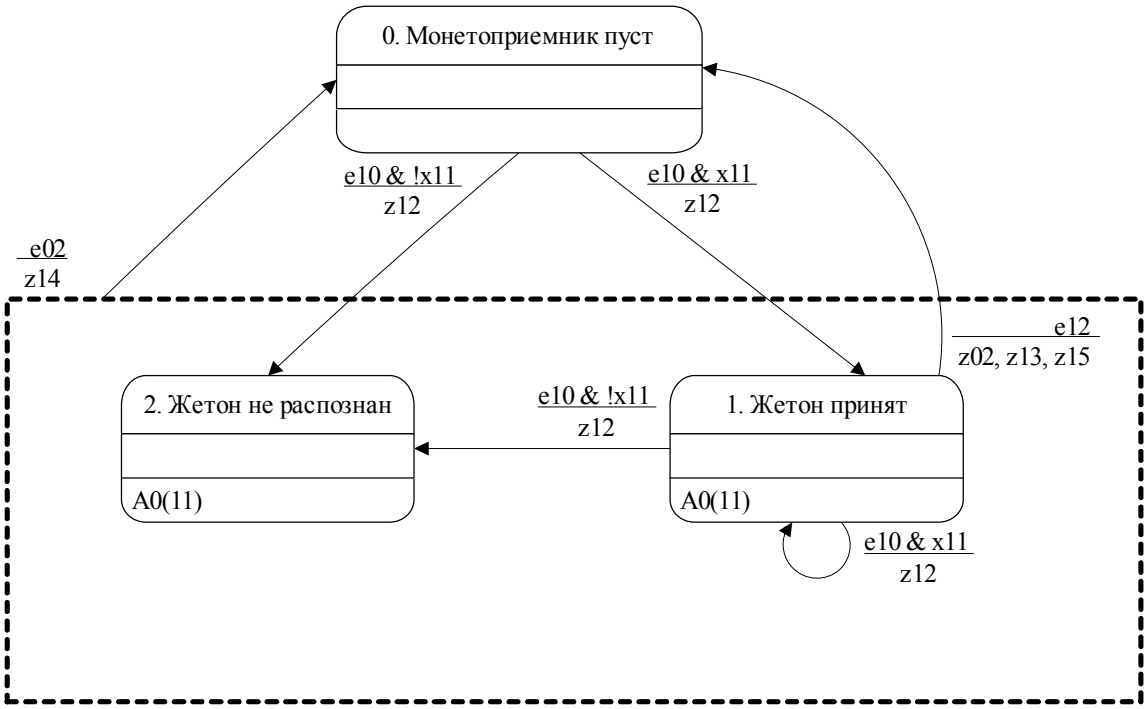

Рис. 6. Граф переходов автомата «Монетоприемник» (A1)

8. Автоматный класс «Барабаны» (TAutomatonBarrelsEngine)

8.1. Словесное описание

Класс *TAutomatonBarrelsEngine* является автоматным классом, порожденным от класса *TAutomaton*. Описание автомата «Игровые барабаны» (A2), который реализует данный класс, приводится ниже.

8.2. Автомат «Игровые барабаны» (A2)

8.2.1. Словесное описание

Данный автомат описывает поведение игровых барабанов в имитируемой игровой машине. Он обеспечивает установку, обработку срабатываний барабанных таймеров, расчет выигрышной суммы и уведомление автомата «Игровая машина» (A0) об окончании игры. Данный автомат является вложенным в автомат «Игровая машина» (A0). Схема связей автомата «Игровые барабаны» (A2) приведена на рис. 7, а граф переходов - на рис. 8.

8.2.2. Схема связей

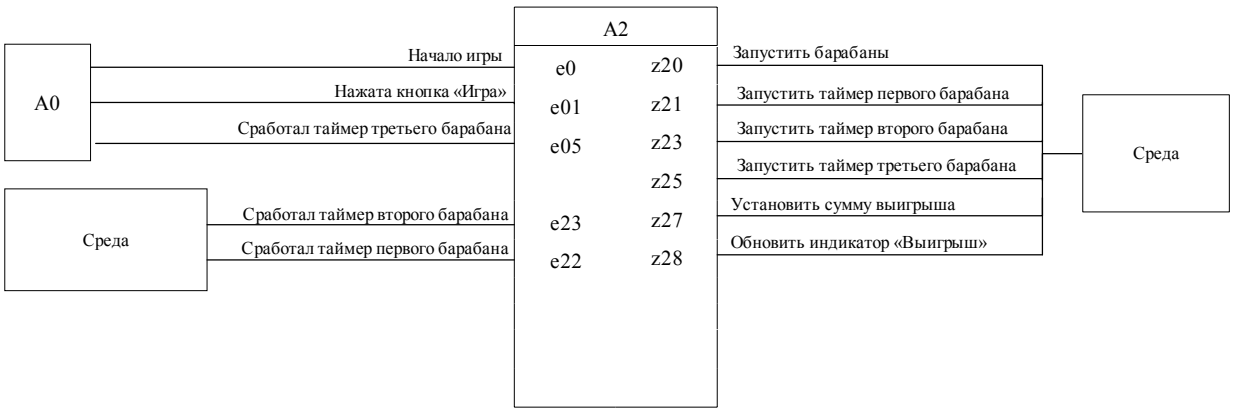

Рис. 7. Схема связей автомата «Игровые барабаны» (A2)

8.2.3. Граф переходов

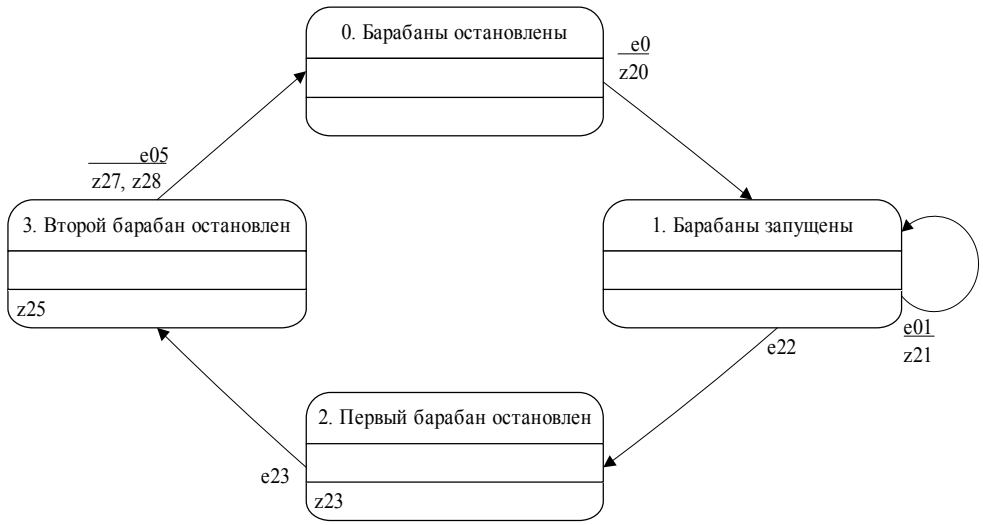

Рис. 8. Граф переходов автомата «Игровые барабаны» (A2)

9. Пример протокола

Ниже приводится протокол, отражающий описанный в постановке задачи цикл игры с опусканием одного «настоящего» жетона.

```
29.03.2003 16:56:18:
Автомат A1 "Монетоприемник" запущен в состоянии 0 "Монетоприемник пуст" с событием e10 "Опущен
жетон"

29.03.2003 16:56:18:
Автомат A1 "Монетоприемник" перешел из состояния 0 в состояние 1

29.03.2003 16:56:18:
Автомат A0 "Игровая машина" запущен в состоянии 0 "Ожидание" с событием e11 "(от автомата A1)
Опущен жетон"

29.03.2003 16:56:18:
Автомат A0 "Игровая машина" перешел из состояния 0 в состояние 1

29.03.2003 16:56:18:
Автомат A0 "Игровая машина" завершил обработку события e11 "(от автомата A1) Опущен жетон" в
состоянии 1 "Прием жетонов"

29.03.2003 16:56:18:
Автомат A1 "Монетоприемник" завершил обработку события e10 "Опущен жетон" в состоянии 1 "Жетон
принят"

29.03.2003 16:56:18:
Автомат A0 "Игровая машина" запущен в состоянии 1 "Прием жетонов" с событием e1 "Нажата кнопка
'Игра'"

29.03.2003 16:56:18:
Автомат A1 "Монетоприемник" запущен в состоянии 1 "Жетон принят" с событием e12 "(от автомата
A0) Перевести деньги в банк"

29.03.2003 16:56:18:
Автомат A1 "Монетоприемник" перешел из состояния 1 в состояние 0

29.03.2003 16:56:18:
Автомат A1 "Монетоприемник" завершил обработку события e12 "(от автомата A0) Перевести деньги в
банк" в состоянии 0 "Монетоприемник пуст"

29.03.2003 16:56:19:
Автомат A0 "Игровая машина" перешел из состояния 1 в состояние 2
```

```
29.03.2003 16:56:19:
Автомат A2 "Игровые барабаны" запущен в состоянии 0 "Барабаны остановлены" с событием e0 "(от
автомата A0): Начало игры"

29.03.2003 16:56:19:
Автомат A2 "Игровые барабаны" перешел из состояния 0 в состояние 1

29.03.2003 16:56:19:
Автомат A2 "Игровые барабаны" завершил обработку события e0 "(от автомата A0): Начало игры" в
состоянии 1 "Барабаны запущены"

29.03.2003 16:56:19:
Автомат A0 "Игровая машина" завершил обработку события e1 "Нажата кнопка 'Игра'" в состоянии 2
"Игра"

29.03.2003 16:56:19:
Автомат A0 "Игровая машина" запущен в состоянии 2 "Игра" с событием e1 "Нажата кнопка 'Игра'"

29.03.2003 16:56:19:
Автомат A2 "Игровые барабаны" запущен в состоянии 1 "Барабаны запущены" с событием e1 "Нажата
кнопка 'Игра'"

29.03.2003 16:56:19:
Автомат A2 "Игровые барабаны" завершил обработку события e1 "Нажата кнопка 'Игра'" в состоянии 1
"Барабаны запущены"

29.03.2003 16:56:19:
Автомат A0 "Игровая машина" завершил обработку события e1 "Нажата кнопка 'Игра'" в состоянии 2
"Игра"

29.03.2003 16:56:21:
Автомат A2 "Игровые барабаны" запущен в состоянии 1 "Барабаны запущены" с событием e22 "Сработал
таймер первого барабана"

29.03.2003 16:56:21:
Автомат A2 "Игровые барабаны" перешел из состояния 1 в состояние 2

29.03.2003 16:56:21:
Автомат A2 "Игровые барабаны" завершил обработку события e22 "Сработал таймер первого барабана"
в состоянии 2 "Первый барабан остановлен"

29.03.2003 16:56:22:
Автомат A2 "Игровые барабаны" запущен в состоянии 2 "Первый барабан остановлен" с событием e23
"Сработал таймер второго барабана"

29.03.2003 16:56:22:
Автомат A2 "Игровые барабаны" перешел из состояния 2 в состояние 3

29.03.2003 16:56:22:
Автомат A2 "Игровые барабаны" завершил обработку события e23 "Сработал таймер второго барабана"
в состоянии 3 "Второй барабан остановлен"

29.03.2003 16:56:24:
Автомат A0 "Игровая машина" запущен в состоянии 2 "Игра" с событием e5 "Сработал таймер третьего
барабана"

29.03.2003 16:56:24:
Автомат A2 "Игровые барабаны" запущен в состоянии 3 "Второй барабан остановлен" с событием e5
"Сработал таймер третьего барабана"

29.03.2003 16:56:24:
Автомат A2 "Игровые барабаны" перешел из состояния 3 в состояние 0

29.03.2003 16:56:24:
Автомат A2 "Игровые барабаны" завершил обработку события e5 "Сработал таймер третьего барабана"
в состоянии 0 "Барабаны остановлены"

29.03.2003 16:56:24:
Автомат A0 "Игровая машина" перешел из состояния 2 в состояние 4

29.03.2003 16:56:24:
Автомат A0 "Игровая машина" завершил обработку события e5 "Сработал таймер третьего барабана" в
состоянии 4 "Выдача жетона"

29.03.2003 16:56:24:
Автомат A0 "Игровая машина" запущен в состоянии 4 "Выдача жетона" с событием e6 "Выдан жетон из
банка"

29.03.2003 16:56:24:
```

```
Автомат A0 "Игровая машина" перешел из состояния 4 в состояние 0

29.03.2003 16:56:24:
Автомат A0 "Игровая машина" завершил обработку события e6 "Выдан жетон из банка" в состоянии 0
"Ожидание"
```

Выводы

Выполненная работа показала эффективность применения автоматов при решении задач построения имитаторов событийных систем. Также была продемонстрирована возможность совмещения объектно-ориентированного и автоматного подходов в программировании.

Работа, по нашему мнению, отвечает на вопрос, поставленный Александром Чижовым (chizh@irk.ru) в переписке на сайте http://cooler.it, состоящий в том, что «теорию конечных автоматов мы проходили, но при чем здесь программирование?».

Литература

1. *Шалыто А. А.* SWITCH-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука, 1998.
2. *Шалыто А.А., Туккель Н.И.* SWITCH-технология – автоматный подход к созданию программного обеспечения "реактивных" систем //Программирование. 2001. №5. http://is.ifmo.ru, раздел «Статьи».

Приложение. Листинг программы

Листинг модуля «Automaton.pas»

```
unit Automaton;

{ ************************************************************************** }
{ *                                                                        * }
{ *  ОБЪЯВЛЕНИЕ КЛАССОВ                                                    * }
{ *                                                                        * }
{ ************************************************************************** }

interface
uses
  Classes;
type
  TEvent = integer;       // Объявление типа для переменных "событие".
  TState = integer;       // Объявление типа для переменных "состояние автомата".
type

{ ************************************************************************** }
{ *                                                                        * }
{ *  Название класса: TAutomaton.                                          * }
{ *  Назначение:      Базовый класс для реализации автоматов.               * }
{ *                                                                        * }
{ ************************************************************************** }

  TAutomatonsDispatcher = class;
  TAutomaton = class
  private
    IsRunning:    boolean;     // Флаг, устанавливаемый на время работы автомата.
  protected
    Name:           integer;   // Номер автомата.
    Description:    string;    // Словесное описание автомата.
    State:          TState;    // Текущее состояние автомата.
  public
    LogInput:       boolean;   // Протоколирование входных воздействий.
    LogOutput:      boolean;   // Протоколирование выходных воздействий.
    LogWake:        boolean;   // Протоколирование начала работы автомата.
    LogStateChange: boolean;   // Протоколирование изменения состояния автомата.
    LogSleep:       boolean;   // Протоколирование окончания работы автомата.
    LogError:       boolean;   // Протоколирование ошибок в работе автомата.
  public
    property y: TState read State; // Публичное свойство для чтения состояния.
  public
    atmDispatcher: TAutomatonsDispatcher; // Ссылка на диспетчер.
```

```
  public
    constructor Init(AutomatonName: integer; AutomatonDescription: string);
    procedure   Wake(Event: TEvent);
  protected
    procedure HandleState(Event: TEvent);    virtual;
    procedure EnterState();                  virtual;
    procedure Log(AutomatonMessage: string); virtual;
  end;

{ ************************************************************************** }
{ *                                                                        * }
{ *  Название класса: TAutomatonsDispatcher.                               * }
{ *  Назначение:      Организует взаимодействие автоматов и хранит служебную * }
{ *                   информацию, например словесные описания событий,      * }
{ *                   названия состояний автоматов и т.п.                   * }
{ *                                                                        * }
{ ************************************************************************** }

  TAutomatonsDispatcher = class
  public
    Automatons : array of TAutomaton; // Список диспетчеризуемых автоматов.
    z: TStrings;                      // Словесое описание выходных воздействий.
    x: TStrings;                      // Словесое описание входных переменных.
    y: array of TStrings;             // Словесое описание состояний автоматов.
    e: TStrings;                      // Словесое описание автоматных событий.
    constructor Create();
    destructor  Free();
    procedure   AddAutomaton (A: TAutomaton);
  end;

{ ************************************************************************** }
{ *                                                                        * }
{ *  РЕАЛИЗАЦИЯ МЕТОДОВ                                                    * }
{ *                                                                        * }
{ ************************************************************************** }

implementation
uses
  SysUtils;

{ ************************************************************************** }
{ *                                                                        * }
{ *  Класс:           TAutomaton.                                          * }
{ *  Название метода: Init.                                                * }
{ *  Назначение:      Конструктор класса TAutomaton.                       * }
{ *                                                                        * }
{ *  Параметры:       AutomatonName - номер автомата.                      * }
```

```
{ *                    AutomatonDescription - название автомата.              * }
{ *                                                                           * }
{ ***************************************************************************** }

  constructor TAutomaton.Init
  (
    AutomatonName:        integer;
    AutomatonDescription: string
  );
  begin
    Name        := AutomatonName;
    Description := AutomatonDescription;
    State       := 0;
  end;

{ ***************************************************************************** }
{ *                                                                           * }
{ *  Класс:            TAutomaton.                                            * }
{ *  Название метода: Wake.                                                   * }
{ *  Назначение:       Производит вызов автомата с указываемым событием.      * }
{ *                                                                           * }
{ *  Параметры:        Event - событие, с которым необходимо вызвать автомат. * }
{ *                                                                           * }
{ ***************************************************************************** }

  procedure TAutomaton.Wake
  (
    Event: TEvent
  );
  var
    OldState: TState;
  begin
    if LogWake then
      Log('Автомат A' + IntToStr(Name) + ' "' + Description +
          '" запущен в состоянии ' + IntToStr(State) + ' "'
                                   + atmDispatcher.y[Name][State] +
          '" с событием e' + IntToStr(Event) + ' "' +
                              atmDispatcher.e[Event] + '"');
    if (IsRunning) then
      Log('Ошибка реентерабельности в автомате A' + IntToStr(Name) + ' "'
                                                  + Description +
          '" в состоянии ' + IntToStr(State) + ' "'
                           + atmDispatcher.y[Name][State]  + '"' +
          ' с событием e' + IntToStr(Event) + ' "'
                          + atmDispatcher.e[Event] + '"');
    IsRunning := true;
    OldState := State;
```

```
    HandleState(Event);
    if (OldState <> State) then
    begin
      if LogStateChange then
        Log('Автомат A' + IntToStr(Name)  + ' "' + Description +
            '" перешел из состояния ' + IntToStr(OldState) +
            ' в состояние ' + IntToStr(State));
      EnterState();
    end;
    IsRunning := false;
    if LogSleep then
      Log('Автомат A' + IntToStr(Name)  + ' "' + Description +
          '" завершил обработку события e' + IntToStr(Event)  + ' "'
                                           + atmDispatcher.e[Event] +
          '" в состоянии ' + IntToStr(State) + ' "'
                           + atmDispatcher.y[Name][State] + '"');
  end;

{ ************************************************************************** }
{ *                                                                        * }
{ *  Класс:           TAutomaton.                                          * }
{ *  Название метода: HandleState.                                         * }
{ *  Назначение:      Метод, перекрываемый в потомке. Производит действия,  * }
{ *                   которые необходимо выполнить для обработки пришедшего * }
{ *                   события. Соответствует первому оператору switch       * }
{ *                   шаблона для реализации автоматов                      * }
{ *                                                                        * }
{ *  Параметры:       Event - событие, с которым вызван автомат.            * }
{ *                                                                        * }
{ ************************************************************************** }

  procedure TAutomaton.HandleState(Event: TEvent);
  begin
  end;

{ ************************************************************************** }
{ *                                                                        * }
{ *  Класс:           TAutomaton.                                          * }
{ *  Название метода: EnterState.                                          * }
{ *  Назначение:      Метод, перекрываемый в потомке. Производит действия,  * }
{ *                   которые необходимо выполнить при переходе автомата в  * }
{ *                   новое состояние. Соответствует второму оператору      * }
{ *                   switch шаблона.                                       * }
{ *                                                                        * }
{ ************************************************************************** }

  procedure TAutomaton.EnterState();
```

```
  begin
  end;

{ ************************************************************************** }
{ *                                                                        * }
{ *  Класс:           TAutomaton.                                          * }
{ *  Название метода: Log.                                                 * }
{ *  Назначение:      Метод, перекрываемый в потомке. Осуществляет вывод    * }
{ *                   указываемой строки в протокол.                       * }
{ *                                                                        * }
{ *  Параметры:       AutomatonMessage - строка, которую необходимо внести в * }
{ *                                      протокол                          * }
{ *                                                                        * }
{ ************************************************************************** }

  procedure TAutomaton.Log(AutomatonMessage: string);
  begin
  end;

{ ************************************************************************** }
{ *                                                                        * }
{ *  Класс:           TAutomatonsDispatcher.                               * }
{ *  Название метода: Create.                                              * }
{ *  Назначение:      Конструктор класса TAutomatonsDispatcher.            * }
{ *                                                                        * }
{ ************************************************************************** }

  constructor TAutomatonsDispatcher.Create;
  begin
    x := TStringList.Create();
    z := TStringList.Create();
    e := TStringList.Create();
  end;

{ ************************************************************************** }
{ *                                                                        * }
{ *  Класс:           TAutomatonsDispatcher.                               * }
{ *  Название метода: Create.                                              * }
{ *  Назначение:      Деструктор класса TAutomatonsDispatcher.             * }
{ *                                                                        * }
{ ************************************************************************** }

  destructor TAutomatonsDispatcher.Free;
  var
   i: integer;
  begin
    x.Free;
```

```
    z.Free;
    e.Free;
    for i:=0 to High(y) do
      y[i].Free;
  end;

{ ************************************************************************** }
{ *                                                                        * }
{ *  Класс:           TAutomatonsDispatcher.                               * }
{ *  Название метода: AddAutomaton.                                        * }
{ *  Назначение:      Добавляет указываемый автомат в свой список.          * }
{ *                                                                        * }
{ *  Параметры:       A - автомат, который необходимо добавить в список     * }
{ *                       диспетчера                                       * }
{ *                                                                        * }
{ ************************************************************************** }

  procedure TAutomatonsDispatcher.AddAutomaton (A: TAutomaton);
  begin
    if Automatons = nil then
    begin
      SetLength(Automatons,1);
      SetLength(y,1)
    end else
    begin
      SetLength(Automatons, High(Automatons)+2);
      SetLength(y, High(y)+2);
    end;

    Automatons[High(Automatons)] := A;
    A.atmDispatcher := self;
    y[High(y)] :=  TStringList.Create;
  end;

{ ************************************************************************** }
{ *                                                                        * }
{ *  КОНЕЦ МОДУЛЯ "Automaton.pas"                                          * }
{ *                                                                        * }
{ ************************************************************************** }

end.
```

Листинг модуля «atmSlotMachine.pas»

```
unit atmSlotMachine;

{ ************************************************************************** }
{ *                                                                        * }
{ *  ОБЪЯВЛЕНИЕ КЛАССОВ                                                    * }
{ *                                                                        * }
{ ************************************************************************** }

interface
  uses
    Automaton;
type

{ ************************************************************************** }
{ *                                                                        * }
{ *  Название класса: TAutomatonSlotMachine.                               * }
{ *  Назначение:      Класс, содержащий автомат "Игровая машина" (A0).      * }
{ *                                                                        * }
{ ************************************************************************** }

  TAutomatonSlotMachine = class (TAutomaton)
  protected
    procedure HandleState(Event: TEvent); override;
    procedure EnterState();  override;
    procedure Log(AutomatonMessage: string); override;

    // Получение состояний других автоматов
    function  GetY1: TState;       // Возвращает состояние автомата A1.
    function  GetY2: TState;       // Возвращает состояние автомата A2.

    // Опрос входных переменных
    function  GetX01(): boolean;   // "Сумма выдачи выигрыша больше нуля"
    function  GetX02(): boolean;   // "В банке есть жетоны"

    // Выходные воздействия
    procedure z02();               // "Закрыть монетоприемник"
    procedure z03();               // "Открыть монетоприемник"
    procedure z04();               // "Выдать жетон из банка"
    procedure z05();               // "Сбросить сумму выигрыша"
    procedure z12();               // "Обновить индикатор <Ставка>"
    procedure z13();               // "Сбросить ставку"
    procedure z15();               // "Обновить индикатор <Банк>"
    procedure z28();               // "Обновить индикатор <Выигрыш>"
```

```
    // Вызовы автоматов
    procedure A1(Event:TEvent);    // Вызов автомата A1.
    procedure A2(Event:TEvent);    // Вызов автомата A2.
  Protected

    // Объявление свойств, «оборачивающих» функции получения состояния автоматов.
    property  y1: TState read GetY1;
    property  y2: TState read GetY2;
  public

    // Объявление свойств, «оборачивающих» функции получения входных параметров.
    property  x01: boolean read GetX01;
    property  x02: boolean read GetX02;
  end;

{ ************************************************************************** }
{ *                                                                        * }
{ *  Название класса: TAutomatonMoneyTray.                                 * }
{ *  Назначение:      Класс, содержащий автомат "Монетоприемник" (A1).      * }
{ *                                                                        * }
{ ************************************************************************** }

  TAutomatonMoneyTray = class (TAutomaton)
  protected
    procedure HandleState(Event: TEvent); override;
    procedure EnterState();  override;
    procedure Log(AutomatonMessage: string); override;

    // Опрос входных переменных
    function  GetX11(): boolean; // "Жетон подлинный"

    // Выходные воздействия
    procedure z02();             // "Закрыть монетоприемник"
    procedure z12();             // "Обновить индикатор <Ставка>"
    procedure z13();             // "Поместить жетоны в банк"
    procedure z14();             // "Вернуть деньги"
    procedure z15();             // "Обновить индикатор <Банк>"

    // Вызовы автоматов
    procedure A0(Event:TEvent);  // Вызов автомата A0.
  Public

    // Объяевление свойств, «оборачивающих» функции получения входных параметров.
    property  x11: boolean read GetX11;
  end;
```

```
{ ************************************************************************** }
{ *                                                                        * }
{ *  Название класса: TAutomatonBarrelsEngine.                             * }
{ *  Назначение:      Класс, содержащий автомат "Игровые барабаны" (A2).    * }
{ *                                                                        * }
{ ************************************************************************** }

  TAutomatonBarrelsEngine = class (TAutomaton)
  protected
    procedure HandleState(Event: TEvent); override;
    procedure EnterState();  override;
    procedure Log(AutomatonMessage: string); override;

    // Выходные воздействия
    procedure z20();              // "Запустить барабаны"
    procedure z21();              // "Запустить таймер первого барабана"
    procedure z23();              // "Запустить таймер второго барабана"
    procedure z25();              // "Запустить таймер третьего барабана"
    procedure z27();              // "Установить сумму выигрыша"
    procedure z28();              // "Обновить индикатор <Выигрыш>"
  end;

{ ************************************************************************** }
{ *                                                                        * }
{ *  РЕАЛИЗАЦИЯ МЕТОДОВ                                                    * }
{ *                                                                        * }
{ ************************************************************************** }

implementation
uses
  SysUtils,
  FormMain;

{ ************************************************************************** }
{ *                                                                        * }
{ *  Методы класса: TAutomatonSlotMachine.                                 * }
{ *                                                                        * }
{ ************************************************************************** }

  //
  // Автомат A0 Switch 1
  //

  procedure  TAutomatonSlotMachine.HandleState(Event: TEvent);
  begin
    case State of
```

```
0:  begin
      if ((Event=11) and (y1=1)) then
      begin
                                                State := 1;
      end
      else if ((Event=11) and (y1=2)) then
      begin
        z02();                                  State := 3;
      end;
    end;
1:  begin
      if ((Event=11) and (y1=2)) then
      begin
        z02();                                  State := 3;
      end
      else if ((Event=01) and (y1=1)) then
      begin
        A1(12);                                 State := 2;
      end
      else if (Event=02) then
      begin
        z02(); A1(2);                           State := 0;
      end;
    end;
2:  begin
      A2(Event);
      if ((Event = 05) and x01 and x02) then
      begin
        z04();                                  State := 4;
      end
      else if ((Event = 05) and not x01) then
      begin
                                                State := 0;
      end;
    end;
3:  begin
      if ((Event=02) and (y1=2)) then
      begin
        A1(2);                                  State := 0;
      end
      else if (Event=03) and (y1<>2) then
      begin
        z05(); z28(); z15();                    State := 0;
      end;
    end;
4:  begin
      if ((Event = 06) and x01 and x02) then
```

```
          begin
            z04(); z28(); z15();
          end
          else if ((Event = 06) and x01 and not x02) then
          begin
            z28(); z15();                          State := 3
          end
          else if ((Event = 06) and not x01) then
          begin
            z28(); z15();                          State := 0;
          end;
        end;
    else
      if LogError then
        Log('Ошибка в автомате A0 при обработке события e' + IntToStr(Event) +
            ': неизвестное состояние ' + IntToStr(State));
  end;
end;

//
// Автомат A0 Switch 2
//

procedure TAutomatonSlotMachine.EnterState();
begin
  case State of
    0: begin z03; z13(); z12(); end;
    2: A2(0);
  end;
end;

//
// Протоколирование
//

procedure TAutomatonSlotMachine.Log(AutomatonMessage: string);
begin
  SlotMachine.Log.Lines.Add(DateToStr(Date) + ' ' + TimeToStr(Time)+':');
  SlotMachine.Log.Lines.Add(AutomatonMessage);
  SlotMachine.Log.Lines.Add('');
end;

//
// Опрос входных переменных
//

function  TAutomatonSlotMachine.GetX01(): boolean;
```

```
  begin result := SlotMachine.JackPotCoinsCount > 0; end;

  function  TAutomatonSlotMachine.GetX02(): boolean;
  begin result := SlotMachine.BankCoinsCount > 0; end;

  //
  // Выходные воздействия
  //

  procedure TAutomatonSlotMachine.z02();
  begin SlotMachine.MoneyTrayEnable(false); end;

  procedure TAutomatonSlotMachine.z03();
  begin SlotMachine.MoneyTrayEnable(true); end;

  procedure TAutomatonSlotMachine.z04();
  begin SlotMachine.PayFromBank(); end;

  procedure TAutomatonSlotMachine.z05();
  begin Slotmachine.JackPotCoinsCount :=0 ; end;

  procedure TAutomatonSlotMachine.z12();
  begin SlotMachine.edStake.Text := IntToStr(SlotMachine.Stake); end;

  procedure TAutomatonSlotMachine.z13();
  begin SlotMachine.Stake := 0; end;

  procedure TAutomatonSlotMachine.z15();
  begin SlotMachine.edBank.Text := IntToStr(SlotMachine.BankCoinsCount); end;

  procedure TAutomatonSlotMachine.z28();
  begin SlotMachine.edJackPot.Text := IntToStr(SlotMachine.JackPotCoinsCount);
end;


  //
  // Вызовы автоматов
  //

  procedure TAutomatonSlotMachine.A1(Event: TEvent);
  begin atmDispatcher.Automatons[1].Wake(Event); end;

  procedure TAutomatonSlotMachine.A2(Event: TEvent);
  begin atmDispatcher.Automatons[2].Wake(Event); end;
```

```
  //
  // Получение состояний других автоматов
  //

  function  TAutomatonSlotMachine.GetY1: TState;
  begin result := atmDispatcher.Automatons[1].Y; end;

  function  TAutomatonSlotMachine.GetY2: TState;
  begin result := atmDispatcher.Automatons[2].Y; end;

{ ************************************************************************** }
{ *                                                                        * }
{ *  Методы класса: TAutomatonMoneyTray.                                   * }
{ *                                                                        * }
{ ************************************************************************** }

  //
  // Автомат A1 Switch 1
  //

  procedure TAutomatonMoneyTray.HandleState(Event: TEvent);
  begin
    case State of
      0:  begin
            if ((Event=10) and x11) then
            begin
              z12();                              State := 1;
            end
            else if ((Event=10) and not x11) then
            begin
              z12();                              State := 2;
            end;
          end;
      1:  begin
            if (Event=02) then
            begin
              z14();                              State := 0;
            end
            else if ((Event=10) and x11) then
            begin
              z12();
            end
            else if ((Event=10) and not x11) then
            begin
              z12();                              State := 2
            end
            else if (Event=12) then
```

```
          begin
            z02(); z13(); z15();                State := 0;
          end;
        end;
    2:  begin
          if (Event=02) then
          begin
            z14();                              State := 0;
          end;
       end;
    else
     if LogError then
       Log('Ошибка в автомате A1 при обработке события e' + IntToStr(Event) +
           ': неизвестное состояние ' + IntToStr(State));
  end;
end;

//
// Автомат A1 Switch 2
//

procedure TAutomatonMoneyTray.EnterState();
begin
  case State of
    1: A0(11);
    2: A0(11);
  end;
end;

//
// Протоколирование
//

procedure TAutomatonMoneyTray.Log(AutomatonMessage: string);
begin
  SlotMachine.Log.Lines.Add(DateToStr(Date) + ' ' + TimeToStr(Time)+':');
  SlotMachine.Log.Lines.Add(AutomatonMessage);
  SlotMachine.Log.Lines.Add('');
end;



//
// Опрос входных переменных
//

function  TAutomatonMoneyTray.GetX11(): boolean;
```

```
  begin result := SlotMachine.IsCoinOK(); end;

  //
  // Выходные воздействия
  //

  procedure TAutomatonMoneyTray.z02();
  begin SlotMachine.MoneyTrayEnable(false); end;

  procedure TAutomatonMoneyTray.z12();
  begin SlotMachine.edStake.Text := IntToStr(SlotMachine.Stake); end;

  procedure TAutomatonMoneyTray.z13();
  begin SlotMachine.RecieveMoney(); end;

  procedure TAutomatonMoneyTray.z14();
  begin SlotMachine.ReturnMoney(); end;

  procedure TAutomatonMoneyTray.z15();
  begin SlotMachine.edBank.Text := IntToStr(SlotMachine.BankCoinsCount); end;

  //
  // Вызовы автоматов
  //

  procedure TAutomatonMoneyTray.A0(Event:TEvent);
  begin atmDispatcher.Automatons[0].Wake(Event); end;

{ ************************************************************************** }
{ *                                                                        * }
{ *  Методы класса: TAutomatonBarrelsEngine.                               * }
{ *                                                                        * }
{ ************************************************************************** }

  //
  // Автомат A2 Switch 1
  //

  procedure TAutomatonBarrelsEngine.HandleState(Event: TEvent);
  begin
    case State of
      0:  begin
            if (Event=0) then
            begin
              z20();                                State := 1;
            end;
          end;
```

```
    1:  begin
          if (Event=22) then
          begin
                                                  State := 2
          end
          else if (Event=01) then
          begin
            z21();
          end;
        end;
    2:  begin
          if (Event=23) then
          begin
                                                  State := 3;
          end;
        end;
    3:  begin
          if (Event=05) then
            begin
              z27(); z28();                       State := 0;
            end;
        end;
    else
     if LogError then
       Log('Ошибка в автомате A2 при обработке события e' + IntToStr(Event) +
           ': неизвестное состояние ' + IntToStr(State));
  end;
end;

//
// Автомат A2 Switch 2
//

procedure TAutomatonBarrelsEngine.EnterState();
begin
  case State of
    2: z23();
    3: z25();
  end;
end;

//
// Протоколирование
//

procedure TAutomatonBarrelsEngine.Log(AutomatonMessage: string);
begin
```

```
    SlotMachine.Log.Lines.Add(DateToStr(Date) + ' ' + TimeToStr(Time)+':');
    SlotMachine.Log.Lines.Add(AutomatonMessage);
    SlotMachine.Log.Lines.Add('');
  end;

  //
  // Выходные воздействия
  //

  procedure TAutomatonBarrelsEngine.z20();
  begin
    SlotMachine.TimerSync1.Interval := 10;
    SlotMachine.TimerSync2.Interval := 1;
    SlotMachine.TimerSync3.Interval := 6;
  end;

  procedure TAutomatonBarrelsEngine.z21();
  begin
    if (SlotMachine.TimerB1.Interval=0) then
      SlotMachine.TimerB1.Interval := 1000 + Random(1000);
  end;

  procedure TAutomatonBarrelsEngine.z23();
  begin SlotMachine.TimerB2.Interval := 1000 + Random(1000) end;

  procedure TAutomatonBarrelsEngine.z25();
  begin SlotMachine.TimerB3.Interval := 1000 + Random(1000) end;

  procedure TAutomatonBarrelsEngine.z27();
  begin SlotMachine.SetJackPot(); end;

  procedure TAutomatonBarrelsEngine.z28();
  begin SlotMachine.edJackPot.Text := IntToStr(SlotMachine.JackPotCoinsCount);
end;

{ ************************************************************************** }
{ *                                                                        * }
{ *  КОНЕЦ МОДУЛЯ "atmSlotMachine.pas"                                     * }
{ *                                                                        * }
{ ************************************************************************** }

end.
```

Листинг модуля «FormMain.pas»

```
// Модуль, обеспечивающий реализацию оконного класса приложения (см. диаграмму классов на
// рис. 2).

unit FormMain;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ExtCtrls, StdCtrls, CheckLst, Automaton, atmSlotMachine,
  Buttons, ImgList, JvxAnimate, JvGIFCtrl;

type
  TSlotMachine = class(TForm)
    TimerSync1: TTimer;
    Panel1: TPanel;
    Barrel1: TJvGIFAnimator;
    Barrel2: TJvGIFAnimator;
    Barrel3: TJvGIFAnimator;
    Log: TMemo;
    cbA0LogSettings: TCheckListBox;
    TimerB1: TTimer;
    TimerB2: TTimer;
    TimerB3: TTimer;
    JackTimer: TTimer;
    Label1: TLabel;
    Label2: TLabel;
    cbA1LogSettings: TCheckListBox;
    Label3: TLabel;
    cbA2LogSettings: TCheckListBox;
    edBank: TEdit;
    Label4: TLabel;
    Image1: TImage;
    Image2: TImage;
    Image3: TImage;
    Image4: TImage;
    Image5: TImage;
    Image6: TImage;
    Image7: TImage;
    Image8: TImage;
    Image9: TImage;
    Image10: TImage;
    Image11: TImage;
    Image12: TImage;
    Image13: TImage;
    Image14: TImage;
    Image15: TImage;
    Image16: TImage;
    Image17: TImage;
    Image18: TImage;
    Image19: TImage;
    Image20: TImage;
    Image21: TImage;
    Image22: TImage;
```

```
    Image23: TImage;
    Image24: TImage;
    Image25: TImage;
    Image26: TImage;
    Image27: TImage;
    Image28: TImage;
    Image29: TImage;
    Image30: TImage;
    Image31: TImage;
    Image32: TImage;
    Image33: TImage;
    Image34: TImage;
    Image35: TImage;
    Image36: TImage;
    Image37: TImage;
    Image38: TImage;
    Image39: TImage;
    Image40: TImage;
    Image41: TImage;
    Image42: TImage;
    Image43: TImage;
    Image44: TImage;
    Image45: TImage;
    Image46: TImage;
    Image47: TImage;
    Image48: TImage;
    Image49: TImage;
    Image50: TImage;
    Image51: TImage;
    Image52: TImage;
    Image53: TImage;
    Image54: TImage;
    Image55: TImage;
    Image56: TImage;
    Image57: TImage;
    Image58: TImage;
    Image59: TImage;
    Image60: TImage;
    Image61: TImage;
    Image62: TImage;
    Image63: TImage;
    Image64: TImage;
    Image65: TImage;
    Image66: TImage;
    Image67: TImage;
    Image68: TImage;
    Image69: TImage;
    Image70: TImage;
    Image71: TImage;
    Image72: TImage;
    Image73: TImage;
    Image74: TImage;
    Image75: TImage;
```

```
Image76: TImage;
Image77: TImage;
Image78: TImage;
Image79: TImage;
Image80: TImage;
Image81: TImage;
Image82: TImage;
Image83: TImage;
Image84: TImage;
Image85: TImage;
Image86: TImage;
Image87: TImage;
Image88: TImage;
Image89: TImage;
Image90: TImage;
Image91: TImage;
Image92: TImage;
Image93: TImage;
Image94: TImage;
Image95: TImage;
Image96: TImage;
Image97: TImage;
Image98: TImage;
Image99: TImage;
Image100: TImage;
Shape1: TShape;
Label5: TLabel;
edStake: TEdit;
TimerSync2: TTimer;
TimerSync3: TTimer;
edJackPot: TEdit;
Label6: TLabel;
iBadCoin: TImage;
btnTechReset: TSpeedButton;
iPlay: TJvGIFAnimator;
iReturn: TJvGIFAnimator;
procedure FormCreate(Sender: TObject);
procedure cbA0LogSettingsClickCheck(Sender: TObject);
procedure cbA1LogSettingsClickCheck(Sender: TObject);
procedure cbA2LogSettingsClickCheck(Sender: TObject);
procedure Image1Click(Sender: TObject);
procedure iReturnClick(Sender: TObject);
procedure iPlayClick(Sender: TObject);
procedure TimerSync2Timer(Sender: TObject);
procedure TimerSync3Timer(Sender: TObject);
procedure TimerB1Timer(Sender: TObject);
procedure TimerB2Timer(Sender: TObject);
procedure TimerB3Timer(Sender: TObject);
procedure TimerSync1Timer(Sender: TObject);
procedure JackTimerTimer(Sender: TObject);
procedure iBadCoinClick(Sender: TObject);
procedure btnTechResetClick(Sender: TObject);
procedure FormDestroy(Sender: TObject);
```

```
  private
    AutomatonsDispatcher: TAutomatonsDispatcher;
    A0:                   TAutomatonSlotMachine;
    A1:                   TAutomatonMoneyTray;
    A2:                   TAutomatonBarrelsEngine;
  public
    BankCoinsCount:    integer;
    UserCoinsCount:    integer;
    JackPotCoinsCount: integer;
    Stake:             integer;
    IsTrayClosed:      boolean;
    CoinOK:            boolean;
  public
    procedure MoneyTrayEnable(enable: boolean);
    procedure PayFromBank();
    function  IsCoinOK(): boolean;
    procedure RecieveMoney();
    procedure ReturnMoney();
    procedure SetJackPot();
    procedure UpdateCoinImages();
  end;
var
  SlotMachine: TSlotMachine;
implementation
{$R *.dfm}
function GetBarrelNumber(Barrel: TJvGIFAnimator): integer;
begin
  if (Barrel.FrameIndex > 0) and (Barrel.FrameIndex < 4) or
     (Barrel.FrameIndex >= 57) and (Barrel.FrameIndex <= 60)
        then result := 1 else
  if (Barrel.FrameIndex >= 4) and (Barrel.FrameIndex < 11)
        then result := 2 else
  if (Barrel.FrameIndex >= 11) and (Barrel.FrameIndex <= 16)
        then result := 3 else
  if (Barrel.FrameIndex > 16) and (Barrel.FrameIndex < 24)
        then result := 4 else
  if (Barrel.FrameIndex >= 24) and (Barrel.FrameIndex <= 30)
        then result := 5 else
  if (Barrel.FrameIndex >=30) and (Barrel.FrameIndex <= 37)
        then result := 6 else
  if (Barrel.FrameIndex >37) and (Barrel.FrameIndex < 43)
        then result := 7 else
  if (Barrel.FrameIndex >43) and (Barrel.FrameIndex < 50)
        then result := 8
  else
        result := 9;
end;

procedure TSlotMachine.FormCreate(Sender: TObject);
var
  i:integer;
begin
  IsTrayClosed:=false;
```

```
  BankCoinsCount := 50;
  UserCoinsCount := 50;
  CoinOK := true;
  UpdateCoinImages;
  Randomize;
  AutomatonsDispatcher := TAutomatonsDispatcher.Create();
  A0 := TAutomatonSlotMachine.Init   (0, 'Игровая машина');
  A1 := TAutomatonMoneyTray.Init     (1, 'Монетоприемник');
  A2 := TAutomatonBarrelsEngine.Init (2, 'Игровые барабаны');
  Barrel1.FrameIndex := 40;
  Barrel2.FrameIndex := 40;
  Barrel3.FrameIndex := 40;
  cbA0LogSettings.Checked[0]:=true; A0.LogWake := true;
  cbA0LogSettings.Checked[1]:=true; A0.LogStateChange := true;
  cbA0LogSettings.Checked[2]:=true; A0.LogSleep:= true;
  cbA1LogSettings.Checked[0]:=true; A1.LogWake := true;
  cbA1LogSettings.Checked[1]:=true; A1.LogStateChange := true;
  cbA1LogSettings.Checked[2]:=true; A1.LogSleep:= true;
  cbA2LogSettings.Checked[0]:=true; A2.LogWake := true;
  cbA2LogSettings.Checked[1]:=true; A2.LogStateChange := true;
  cbA2LogSettings.Checked[2]:=true; A2.LogSleep:= true;
  AutomatonsDispatcher.AddAutomaton(A0);
  AutomatonsDispatcher.AddAutomaton(A1);
  AutomatonsDispatcher.AddAutomaton(A2);
  AutomatonsDispatcher.y[0].Add('Ожидание');
  AutomatonsDispatcher.y[0].Add('Прием жетонов');
  AutomatonsDispatcher.y[0].Add('Игра');
  AutomatonsDispatcher.y[0].Add('Ошибка');
  AutomatonsDispatcher.y[0].Add('Выдача жетона');
  AutomatonsDispatcher.y[1].Add('Монетоприемник пуст');
  AutomatonsDispatcher.y[1].Add('Жетон принят');
  AutomatonsDispatcher.y[1].Add('Жетон не распознан');
  AutomatonsDispatcher.y[2].Add('Барабаны остановлены');
  AutomatonsDispatcher.y[2].Add('Барабаны запущены');
  AutomatonsDispatcher.y[2].Add('Первый барабан остановлен');
  AutomatonsDispatcher.y[2].Add('Второй барабан остановлен');
  for i:=0 to 23 do
    AutomatonsDispatcher.e.Add('');
  AutomatonsDispatcher.e[0]  := 'от A0: Начало игры';
  AutomatonsDispatcher.e[22] := 'Сработал таймер первого барабана';
  AutomatonsDispatcher.e[23] := 'Сработал таймер второго барабана';
  AutomatonsDispatcher.e[5]  := 'Сработал таймер третьего барабана';
  AutomatonsDispatcher.e[1]  := 'Нажата кнопка "Старт"';
  AutomatonsDispatcher.e[2]  := 'Нажата кнопка "Возврат денег"';
  AutomatonsDispatcher.e[3]  := 'Нажата кнопка "Тех. Сброс"';
  AutomatonsDispatcher.e[6]  := 'Выдан жетон из банка';
  AutomatonsDispatcher.e[10] := 'Опущен жетон';
  AutomatonsDispatcher.e[11] := 'от A1: Опущен жетон';
  AutomatonsDispatcher.e[12] := 'от A0: Перевести деньги в банк';
  AutomatonsDispatcher.e[12] := 'от A0: Перевести деньги в банк';
end;

procedure TSlotMachine.MoneyTrayEnable(enable: boolean);
```

```
begin
  IsTrayClosed := not enable;
  if (IsTrayClosed) then
    Shape1.Brush.Style := bsDiagCross
  else
    Shape1.Brush.Style := bsSolid;
end;

procedure TSlotMachine.PayFromBank();
begin
    dec(BankCoinsCount);
    dec(JackPotCoinsCount);
    inc(UserCoinsCount);
    JackTimer.Interval := 250;
end;

function TSlotMachine.IsCoinOK(): boolean;
begin
  result := coinOK;
end;

procedure TSlotMachine.RecieveMoney();
begin
  BankCoinsCount := BankCoinsCount + Stake;
end;

procedure TSlotMachine.ReturnMoney();
begin
    UserCoinsCount := UserCoinsCount + Stake;
    Stake := 0;
    coinOK := true;
    UpdateCoinImages;
end;

procedure TSlotMachine.SetJackPot();
var
  b1: integer;
  b2: integer;
  b3: integer;
begin
  b1 := GetBarrelNumber(Barrel1);
  b2 := GetBarrelNumber(Barrel2);
  b3 := GetBarrelNumber(Barrel3);

  if (b1 = b2) and (b2 = b3) then
  begin
    if (b1 = 7) then JackPotCoinsCount := 10*Stake
                else JackPotCoinsCount := 5*Stake;
  end
  else if (b1+1 = b2) and (b2+1 = b3)           then JackPotCoinsCount := 7*Stake
  else if (b1 = b2) or (b3 = b2) or (b3 = b2)   then JackPotCoinsCount := 4*Stake
  else if (b1 = 7)                              then JackPotCoinsCount := stake + 1
  else if (b1 div 2 <> 0)                       then JackPotCoinsCount := 1
```

```
  else
    JackPotCoinsCount := 0;
end;

procedure TSlotMachine.cbA0LogSettingsClickCheck(Sender: TObject);
begin
  A0.LogWake        := cbA0LogSettings.Checked[0];
  A0.LogStateChange := cbA0LogSettings.Checked[1];
  A0.LogSleep       := cbA0LogSettings.Checked[2];
  A0.LogInput       := cbA0LogSettings.Checked[3];
  A0.LogOutput      := cbA0LogSettings.Checked[4];
end;

procedure TSlotMachine.cbA1LogSettingsClickCheck(Sender: TObject);
begin
  A1.LogWake        := cbA1LogSettings.Checked[0];
  A1.LogStateChange := cbA1LogSettings.Checked[1];
  A1.LogSleep       := cbA1LogSettings.Checked[2];
  A1.LogInput       := cbA1LogSettings.Checked[3];
  A1.LogOutput      := cbA1LogSettings.Checked[4];
end;

procedure TSlotMachine.cbA2LogSettingsClickCheck(Sender: TObject);
begin
  A2.LogWake        := cbA2LogSettings.Checked[0];
  A2.LogStateChange := cbA2LogSettings.Checked[1];
  A2.LogSleep       := cbA2LogSettings.Checked[2];
  A2.LogInput       := cbA2LogSettings.Checked[3];
  A2.LogOutput      := cbA2LogSettings.Checked[4];
end;

procedure TSlotMachine.Image1Click(Sender: TObject);
begin
  if (UserCoinsCount<>0) and not IsTrayClosed then
  begin
    coinOK := true;
    UserCoinsCount := UserCoinsCount - 1;
    Stake          := Stake + 1;
    A1.Wake(10);
    UpdateCoinImages;
  end;
end;

procedure TSlotMachine.UpdateCoinImages();
begin
  Image1.Visible := UserCoinsCount >= 1;
  Image2.Visible := UserCoinsCount >= 2;
  Image3.Visible := UserCoinsCount >= 3;
  Image4.Visible := UserCoinsCount >= 4;
  Image5.Visible := UserCoinsCount >= 5;
  Image6.Visible := UserCoinsCount >= 6;
  Image7.Visible := UserCoinsCount >= 7;
  Image8.Visible := UserCoinsCount >= 8;
```

```
Image9.Visible := UserCoinsCount >= 9;

Image10.Visible := UserCoinsCount >= 10;
Image11.Visible := UserCoinsCount >= 11;
Image12.Visible := UserCoinsCount >= 12;
Image13.Visible := UserCoinsCount >= 13;
Image14.Visible := UserCoinsCount >= 14;
Image15.Visible := UserCoinsCount >= 15;
Image16.Visible := UserCoinsCount >= 16;
Image17.Visible := UserCoinsCount >= 17;
Image18.Visible := UserCoinsCount >= 18;
Image19.Visible := UserCoinsCount >= 19;

Image20.Visible := UserCoinsCount >= 20;
Image21.Visible := UserCoinsCount >= 21;
Image22.Visible := UserCoinsCount >= 22;
Image23.Visible := UserCoinsCount >= 23;
Image24.Visible := UserCoinsCount >= 24;
Image25.Visible := UserCoinsCount >= 25;
Image26.Visible := UserCoinsCount >= 26;
Image27.Visible := UserCoinsCount >= 27;
Image28.Visible := UserCoinsCount >= 28;
Image29.Visible := UserCoinsCount >= 29;

Image30.Visible := UserCoinsCount >= 30;
Image31.Visible := UserCoinsCount >= 31;
Image32.Visible := UserCoinsCount >= 32;
Image33.Visible := UserCoinsCount >= 33;
Image34.Visible := UserCoinsCount >= 34;
Image35.Visible := UserCoinsCount >= 35;
Image36.Visible := UserCoinsCount >= 36;
Image37.Visible := UserCoinsCount >= 37;
Image38.Visible := UserCoinsCount >= 38;
Image39.Visible := UserCoinsCount >= 39;

Image40.Visible := UserCoinsCount >= 40;
Image41.Visible := UserCoinsCount >= 41;
Image42.Visible := UserCoinsCount >= 42;
Image43.Visible := UserCoinsCount >= 43;
Image44.Visible := UserCoinsCount >= 44;
Image45.Visible := UserCoinsCount >= 45;
Image46.Visible := UserCoinsCount >= 46;
Image47.Visible := UserCoinsCount >= 47;
Image48.Visible := UserCoinsCount >= 48;
Image49.Visible := UserCoinsCount >= 49;

Image50.Visible := UserCoinsCount >= 50;
Image51.Visible := UserCoinsCount >= 51;
Image52.Visible := UserCoinsCount >= 52;
Image53.Visible := UserCoinsCount >= 53;
Image54.Visible := UserCoinsCount >= 54;
Image55.Visible := UserCoinsCount >= 55;
Image56.Visible := UserCoinsCount >= 56;
```

```
  Image57.Visible := UserCoinsCount >= 57;
  Image58.Visible := UserCoinsCount >= 58;
  Image59.Visible := UserCoinsCount >= 59;

  Image60.Visible := UserCoinsCount >= 60;
  Image61.Visible := UserCoinsCount >= 61;
  Image62.Visible := UserCoinsCount >= 62;
  Image63.Visible := UserCoinsCount >= 63;
  Image64.Visible := UserCoinsCount >= 64;
  Image65.Visible := UserCoinsCount >= 65;
  Image66.Visible := UserCoinsCount >= 66;
  Image67.Visible := UserCoinsCount >= 67;
  Image68.Visible := UserCoinsCount >= 68;
  Image69.Visible := UserCoinsCount >= 69;

  Image70.Visible := UserCoinsCount >= 70;
  Image71.Visible := UserCoinsCount >= 71;
  Image72.Visible := UserCoinsCount >= 72;
  Image73.Visible := UserCoinsCount >= 73;
  Image74.Visible := UserCoinsCount >= 74;
  Image75.Visible := UserCoinsCount >= 75;
  Image76.Visible := UserCoinsCount >= 76;
  Image77.Visible := UserCoinsCount >= 77;
  Image78.Visible := UserCoinsCount >= 78;
  Image79.Visible := UserCoinsCount >= 79;

  Image80.Visible := UserCoinsCount >= 80;
  Image81.Visible := UserCoinsCount >= 81;
  Image82.Visible := UserCoinsCount >= 82;
  Image83.Visible := UserCoinsCount >= 83;
  Image84.Visible := UserCoinsCount >= 84;
  Image85.Visible := UserCoinsCount >= 85;
  Image86.Visible := UserCoinsCount >= 86;
  Image87.Visible := UserCoinsCount >= 87;
  Image88.Visible := UserCoinsCount >= 88;
  Image89.Visible := UserCoinsCount >= 89;

  Image90.Visible := UserCoinsCount >= 90;
  Image91.Visible := UserCoinsCount >= 91;
  Image92.Visible := UserCoinsCount >= 92;
  Image93.Visible := UserCoinsCount >= 93;
  Image94.Visible := UserCoinsCount >= 94;
  Image95.Visible := UserCoinsCount >= 95;
  Image96.Visible := UserCoinsCount >= 96;
  Image97.Visible := UserCoinsCount >= 97;
  Image98.Visible := UserCoinsCount >= 98;
  Image99.Visible := UserCoinsCount >= 99;

  Image100.Visible := UserCoinsCount >= 100;

  iBadCoin.Visible := coinOK;
end;
```

```
procedure TSlotMachine.iReturnClick(Sender: TObject);
begin
  A0.Wake(2);
  UpdateCoinImages;
end;

procedure TSlotMachine.iPlayClick(Sender: TObject);
begin
  A0.Wake(1);
end;

procedure TSlotMachine.TimerSync1Timer(Sender: TObject);
begin
 if (Barrel1.FrameIndex = 60)
    then Barrel1.FrameIndex := 1
  else
    Barrel1.FrameIndex := Barrel1.FrameIndex + 1;
end;

procedure TSlotMachine.TimerSync2Timer(Sender: TObject);
begin
  if (Barrel2.FrameIndex = 60)
    then Barrel2.FrameIndex := 1
  else
    Barrel2.FrameIndex := Barrel2.FrameIndex + 1;
end;

procedure TSlotMachine.TimerSync3Timer(Sender: TObject);
begin
  if (Barrel3.FrameIndex = 60)
    then Barrel3.FrameIndex := 1
  else
    Barrel3.FrameIndex := Barrel3.FrameIndex + 1;
end;

procedure TSlotMachine.TimerB1Timer(Sender: TObject);
begin
  TimerB1.Interval := 0;
  TimerSync1.Interval := 0;
  A2.Wake(22);
end;

procedure TSlotMachine.TimerB2Timer(Sender: TObject);
begin
  TimerB2.Interval := 0;
  TimerSync2.Interval := 0;
  A2.Wake(23);
end;

procedure TSlotMachine.TimerB3Timer(Sender: TObject);
begin
  TimerB3.Interval := 0;
  TimerSync3.Interval := 0;
```

```
    A0.Wake(5);
  end;

  procedure TSlotMachine.JackTimerTimer(Sender: TObject);
  begin
    if ((BankCoinsCount = 0) or (JackPotCoinsCount = 0))
      then
        JackTimer.Interval := 0;
    A0.Wake(6);
    UpdateCoinImages;
  end;

  procedure TSlotMachine.iBadCoinClick(Sender: TObject);
  begin
    if not IsTrayClosed then
    begin
      coinOK := false;
      A1.Wake(10);
      UpdateCoinImages;
    end;
  end;

  procedure TSlotMachine.btnTechResetClick(Sender: TObject);
  begin
    A0.Wake(3);
  end;

  procedure TSlotMachine.FormDestroy(Sender: TObject);
  begin
    AutomatonsDispatcher.Free;
  end;
end.
```