Санкт-Петербургский государственный университет
информационных технологий, механики и оптики
Факультет «Информационные технологии и программирование»
Кафедра «Компьютерные технологии»

Д.Д. Захаров, С.С. Косухин

Моделирование устройства класса «Однорукий бандит» на инструментальном средстве *UniMod*

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

Санкт-Петербург
2006

Оглавление

Введение

Задачу разработки программного продукта или его части обычно разбивают на три фазы: анализ предметной области (необходимо ответить на вопрос «ЧТО мы собираемся создать?»), функциональный дизайн («КАК мы собираемся это создать?») и программная реализация.

Первая фаза является самой важной, поскольку ошибки, совершенные во время ее выполнения, имеют самую большую цену. На данной фазе необходим особый контроль над результатами действий разработчиков. Вторая играет чуть менее важную роль, и именно она представляет собой решение задачи, поставленной во время выполнения первой фазы. Необходимо детально описать, каким образом будет произведена реализация. Если функциональный дизайн сделан действительно хорошо, то на третьей фазе необходимо будет просто преобразовать в программный код полученное детальное описание.

SWITCH-технология (автоматное программирование) [1–3] представляет собой новый подход к функциональному дизайну. Она помогает на время забыть о программном коде и все внимание посвятить сути задачи. Для этого предлагается выделить объекты управления и поставщики событий, а для того, чтобы описать поведение программы, разработать систему взаимодействующих конечных автоматов. В каждом автомате состояния будут соответствовать состояниям работающей программы. Таким образом, наглядно описывается поведение программы, и поэтому при таком подходе намного проще избежать ошибок проектирования, нежели при словесном описании проекта.

Сайт http://is.ifmo.ru содержит информацию о SWITCH-технологии и проекты, построенные на ее основе.

Применить SWITCH-технологию к построению реальной программы помогает инструментальное средство *UniMod* (http://unimod.sourceforge.net), которое является встраиваемым модулем (plug-in) для среды разработки *Eclipse* (http://eclipse.org). Он позволяет начать проектирование с создания схемы связей, состоящей из поставщиков событий, автоматов и объектов управления. После этого для каждого автомата строится его диаграмма переходов.

На диаграммах переходов присутствуют начальное и конечное состояния. Для каждого из остальных имеется возможность определить переходы в другие состояния при приходе определенного события и выполнении определенных условий. На переходах и при входе в состояния есть возможность выполнять действия над объектами управления. Объекты управления и поставщики событий реализуются как *Java*-классы, а программный код для них пишется вручную.

После того, как построены схема связей и диаграммы переходов, можно сразу получить работающую программу с заглушками. Получить такую программу возможно двумя способами. Первый из них заключается в том, что автоматически строится *XML*-файл, описывающий схему связей и диаграммы переходов, этот файл подается на вход интерпретатора (один из компонентов *UniMod*) вместе с написанными вручную классами для поставщиков событий и объектов управления (интерпретационный подход). Второй предполагает компиляцию всех компонент в байт-код и его реализацию (компиляционный подход).

Подход, на котором основан *UniMod*, позволяет, в отличие от других открытых средств, строить программу в целом, а не ее отдельные компоненты.

Целью данной работы является изучение и демонстрация возможностей инструментального средства *UniMod* на примере простой программы – имитатора игрового устройства класса «Однорукий бандит». Другой имитатор этого класса приведен в работе http://is.ifmo.ru/projects/slotmachine/.

1. Описание основных узлов устройства

Игровое устройство класса «Однорукий бандит», реализованное аппаратно, является автономным и требует вмешательства обслуживающего персонала только для изъятия накопленных или при нехватке текущих средств.

Ниже приводится описание виртуального устройства, которое моделируется в данной работе. Устройство состоит из следующих компонент:

- игровые барабаны;
- дисплей для показа текущего состояния игроку;
- панель с кнопками для управления процессом игры;
- монетный механизм;
- устройство, управляющее вращением барабанов;
- устройство выдачи выигрыша.

Устройством принимаются монеты только одного достоинства, и они являются единицами расчета. Любая другая опущенная монета считается фальшивой. Игрок может сделать ставку и запустить игровые барабаны. После их остановки происходит подсчет выигрыша. В зависимости от выпавшей комбинации чисел игрок может ничего не выиграть (потерять поставленные деньги) или выиграть некоторую сумму. Сумма выигрыша определяется следующим образом: при наличии любых двух одинаковых цифр выигрыш равен размеру ставки, увеличенной в пять раз; при наличии любых трех одинаковых цифр размер выигрыша равен размеру ставки, увеличенной в 10 раз.

После выдачи выигрыша игрок может снова делать ставку и играть до тех пор, пока либо у игрока, либо в банке не останется средства.

Интерфейс приложения (рис. 1) представляет собой модель игрового устройства.

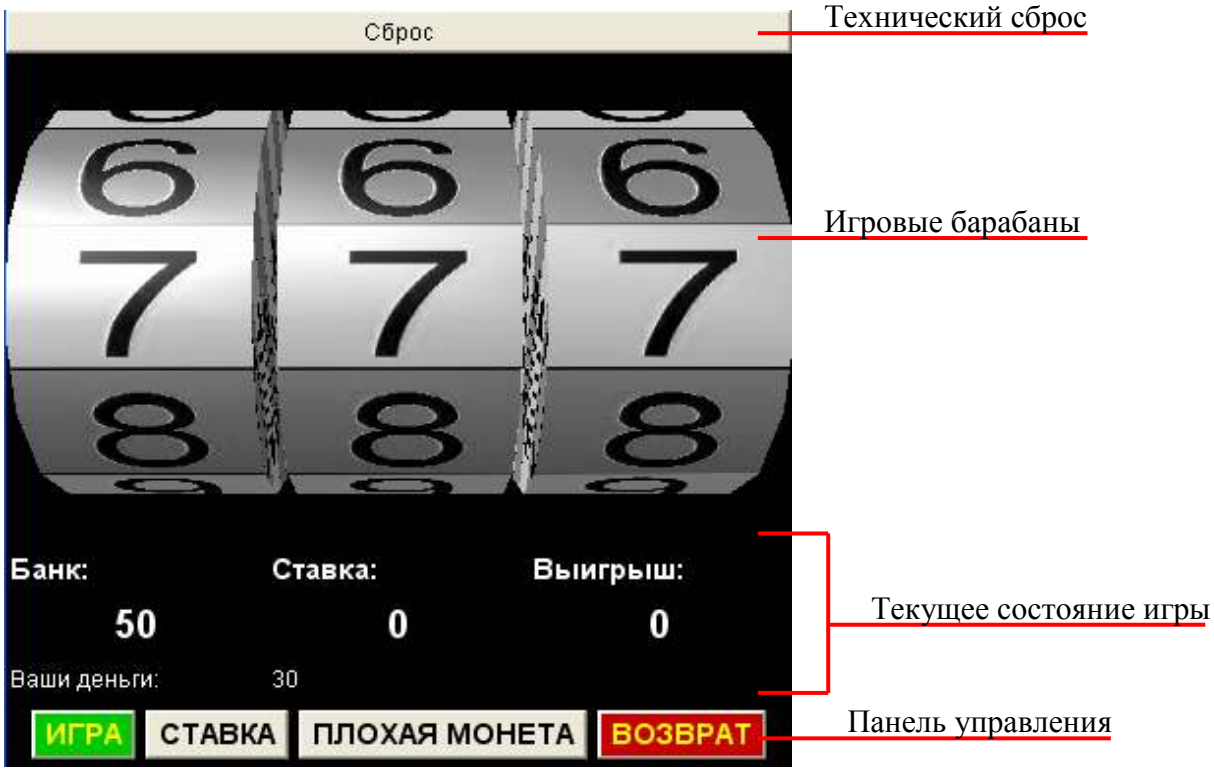

Рис. 1. Внешний вид приложения

В центре окна приложения размещены игровые барабаны, ниже отображается текущее состояние игры. Панель управления игрой расположена внизу окна. В начале игры предполагается, что у игрока есть 30 монет, а в банке – 50. В случае, если в банке заканчиваются деньги, то есть возможность начать игру заново, нажав кнопку «Сброс», расположенную вверху игрового поля.

Кнопки на панели управления означают следующее:

- «Игра» – начать игру (запустить барабаны);
- «Ставка» – опустить одну правильную монету (увеличить ставку);
- «Плохая монета» – опустить фальшивую монету;
- «Возврат» – вернуть все монеты из сделанной ставки.

Сообщения для игрока и текущее состояние игры отображаются в нижней части окна, расположенной под барабанами.

2. Описание поведения

1. **Игрок делает ставку.** Монетный механизм принимает все монеты по очереди, проверяя каждую из них на подлинность. В случае, если монета фальшивая, механизм сообщает об этом игроку, и тот может вернуть монеты, нажав кнопку «Возврат». В случае подлинной монеты ставка увеличивается на единицу.
2. **Игрок запускает барабаны.** Барабаны вращаются и останавливаются по очереди. Выпавшая комбинация используется для подсчета выигрыша игрока. Если выигрыш нулевой, то автомат готов к приему следующей ставки. Если выигрыш ненулевой, то он выдается игроку по одной монете. Если при выдаче выигрыша в какой-то момент в банке закончились средства, то игроку об этом сообщается. После этого вернуть автомат в начальное состояние можно, нажав кнопку «Сброс».

3. Проектирование

3.1. Схема связей автоматов

На схеме связей автоматов (рис. 2) изображены поставщики событий, автоматы, объекты управления и связи между ними.

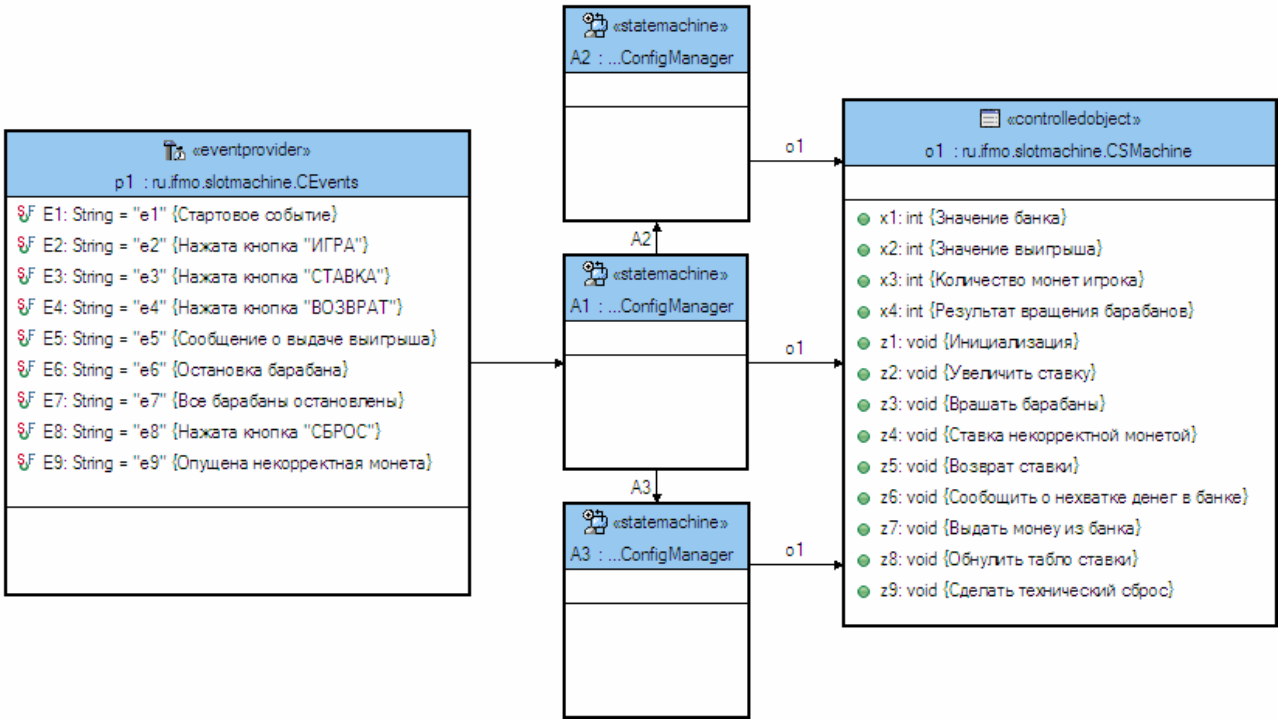

Рис. 2. Схема связей автоматов

Слева на схеме изображен поставщик событий. Этот объект будет генерировать события во время исполнения программы.

В центре схемы изображены три автомата, поведение каждого из которых определяется соответствующим графом переходов.

Генерация события означает его передачу на один из автоматов, в нашем случае в автомат A1 (основной автомат). Например, передача события `e1` запустит основной автомат, а событие `e2` запустит барабаны.

Справа на схеме расположен объект управления. В нем реализованы методы, вызываемые автоматами. Эти методы могут быть двух видов. Методы первого вида возвращают информацию о состоянии объекта управления (они обозначаются `x1`, `x2` и т.д.). Они используются при описании условий переходов в автоматах. Методы второго вида позволяют изменять состояние объекта управления (они обозначаются `z1`, `z2` и т. д.).

3.2. Графы переходов автоматов

3.2.1. Основной автомат

Основной автомат (рис. 3) объединяет части программы (вращение барабанов, прием монет и общий ход игры) в одно целое. Именно ему передаются события из поставщика событий, а он, в свою очередь, может направлять их двум вспомогательным автоматам по мере необходимости.

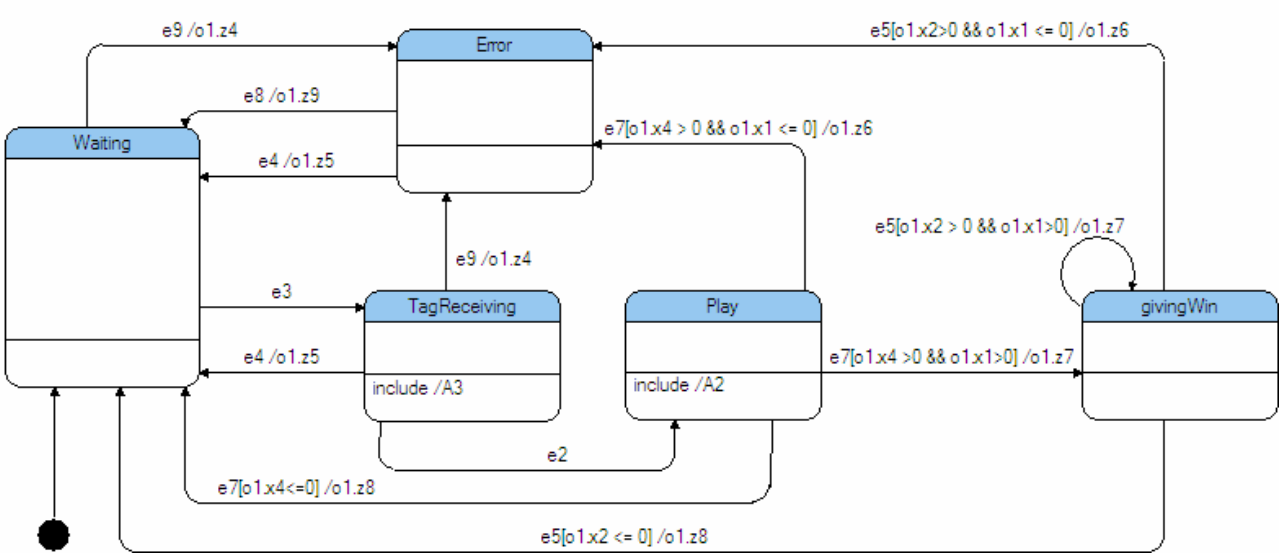

Рис. 3. Основной автомат

Черным кружком обозначено начальное состояние автомата. Когда автомат получает событие `e1`, он переходит в состояние `Waiting`. Каждый раз, получая событие, автомат может перейти в другое состояние. *UniMod* позволяет указывать условия на переходах, которые позволяют совершать переход тогда и только тогда, когда выполняется булева формула. В качестве выражений в булевой формуле могут участвовать методы объекта управления, которые возвращают некоторое значение. К примеру, условие перехода

e5 [o1.x2>0 && o1.x1 <= 0] / o1.z6

означает, что переход будет выполняться по событию `e5`, но только тогда, когда в объекте управления результат вращения барабанов (его возвращает функция `o1.x2`) больше нуля и количество монет в банке (его возвращает функция `o1.x1`) меньше либо равно нулю. Окончание условия `/o1.z6` означает, что при переходе будет вызвана функция объекта

управления `z6` (которая выполняет действия, необходимые при наступлении нулевого баланса в банке).

3.2.2. Автомат, отвечающий за вращение барабанов

Кроме главного автомата, в программе используется еще два, один из которых отвечает за вращение барабанов, а другой за прием монет.

Вспомогательные автоматы довольно просты. Первый из них (рис. 4) разбивает вращение барабанов на четыре состояния. В этом автомате действия над объектом управления выполняются при входе в состояние `Rotation`. Об этом свидетельствует надпись внизу изображения состояния `enter/o1.z3`. В данном случае вызывается функция `z3` объекта управления, которая начинает вращение барабанов.

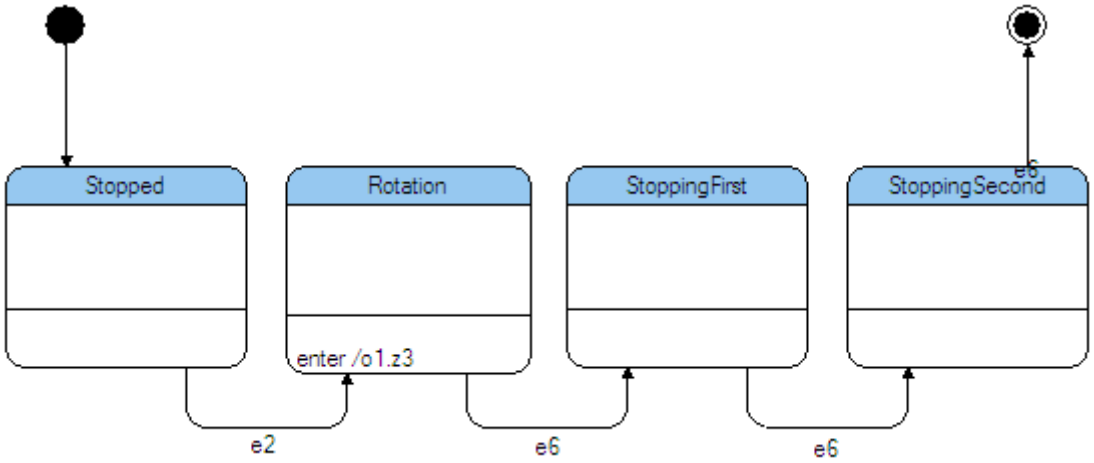

Рис. 4. Автомат, отвечающий за вращение барабанов

3.2.3. Автомат, отвечающий за прием монет

За прием монет отвечает отдельный автомат (рис. 5). Он находится в своем основном состоянии **TrueCoinReceived** до тех пор, пока у игрока не закончатся монеты.

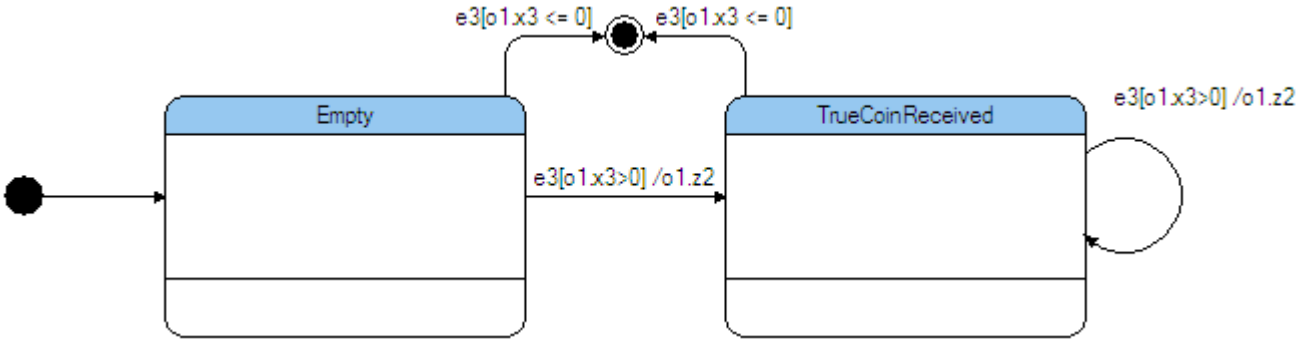

Рис. 5. Автомат, отвечающий за прием монет

4. Исполнение программы

Обратим внимание, что для запуска программы потребуется поддержка *OpenGL*.

UniMod предлагает два подхода к выполнению созданного приложения: нтерпретационный и компиляционный.

При **интерпретации** создается *XML*-файл, который описывает схему связей автоматов и графы переходов (Приложение 1), а *Java*-классы объектов управления и поставщиков событий, написанные вручную, компилируются в байт-код. *XML*-файл используется интерпретатором (одним из компонентов *UniMod*) для того, чтобы управлять выполнением программы.

Для запуска приложения с использованием интерпретационного подхода, введите в командной строке команду `call Java -jar SM.jar A1.XML`

Эта команда запускает *Java*-приложение, которое содержится в архиве `SM.jar` с параметром `A1.XML`. Этот параметр – файл, автоматически сгенерированный инструментальным средством *UniMod*. Файл содержит схему связей автоматов и графы переходов.

При **компиляции** для создания работающего приложения *XML*-файл транслируется в *Java*-код с помощью программы Velocity (http://jakarta.apache.org/velocity). Код программы также содержит написанные вручную классы объектов управления и поставщиков событий.

Для инициализации приложения необходимо запустить на выполнение файл `SM.jar`.

5. Статистика

При разработке приложения часть кода, задаваемая при компиляционном подходе диаграммами, генерируется инструментальным средством *UniMod* автоматически. На диаграмме (рис. 6) представлено соотношение кода, написанного вручную и сгенерированного автоматически.

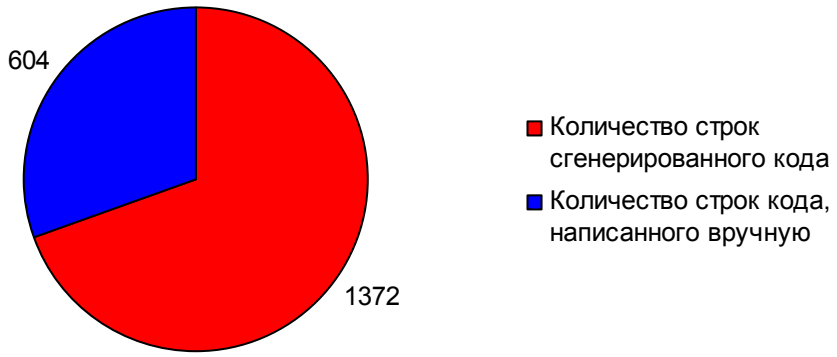

Рис. 6. Статистика по строкам кода, написанного вручную и сгенерированного автоматически

Заключение

Количество приложений, разрабатываемых на основе SWITCH-технологии, растет. Удобным инструментальным средством для их проектирования является *UniMod*. Оно позволяет четко увидеть структуру программы еще до того, как она будет написана. В данной работе приведен пример программы, реализованной с помощью инструментального средства *UniMod*.

Список литературы

1. *Шалыто А. А*. SWITCH-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука, 1998. http://is.ifmo.ru/books/switch/1/.
2. *Шалыто А. А., Туккель Н.И.* Танки и автоматы // BYTE/Россия. 2003. № 2, с. 69-73 http://is.ifmo.ru/works/tanks_new/.
3. *Шалыто А. А., Туккель Н.И.* SWITCH-технология - автоматный подход к созданию программного обеспечения «реактивных» систем // Программирование. 2001. № 5, с. 45-62. http://is.ifmo.ru/works/switch/1/.

Приложение 1. *XML*-описание автоматов

```
<?xml version="1.0" encoding="UTF-8" ?>
- <model name="Model1">
  <controlledObject class="ru.ifmo.slotmachine.SMControlledObj" name="o1" />
- <eventProvider class="ru.ifmo.slotmachine.SMEventProvider" name="p1">
  <association clientRole="p1" targetRef="A1" />
    </eventProvider>
- <rootStateMachine>
  <stateMachineRef name="A1" />
    </rootStateMachine>
- <stateMachine name="A2">
  <configStore class="com.evelopers.unimod.runtime.config.DistinguishConfigManager" />
  <association clientRole="A2" supplierRole="o1" targetRef="o1" />
- <state name="Top" type="NORMAL">
  <state name="s1" type="INITIAL" />
  <state name="s2" type="FINAL" />
  <state name="Stopped" type="NORMAL" />
- <state name="Rotation" type="NORMAL">
  <outputAction ident="o1.z3" />
    </state>
  <state name="StoppingFirst" type="NORMAL" />
  <state name="StoppingSecond" type="NORMAL" />
    </state>
  <transition sourceRef="s1" targetRef="Stopped" />
  <transition event="e2" sourceRef="Stopped" targetRef="Rotation" />
  <transition event="e6" sourceRef="Rotation" targetRef="StoppingFirst" />
  <transition event="e6" sourceRef="StoppingFirst" targetRef="StoppingSecond" />
  <transition event="e6" sourceRef="StoppingSecond" targetRef="s2" />
    </stateMachine>
- <stateMachine name="A1">
  <configStore class="com.evelopers.unimod.runtime.config.DistinguishConfigManager" />
  <association clientRole="A1" supplierRole="A2" targetRef="A2" />
  <association clientRole="A1" supplierRole="o1" targetRef="o1" />
  <association clientRole="A1" supplierRole="A3" targetRef="A3" />
- <state name="Top" type="NORMAL">
  <state name="Error" type="NORMAL" />
  <state name="Waiting" type="NORMAL" />
- <state name="TagReceiving" type="NORMAL">
  <stateMachineRef name="A3" />
    </state>
- <state name="Play" type="NORMAL">
  <stateMachineRef name="A2" />
    </state>
  <state name="givingWin" type="NORMAL" />
  <state name="s1" type="INITIAL" />
    </state>
- <transition event="e4" sourceRef="Error" targetRef="Waiting">
  <outputAction ident="o1.z5" />
    </transition>
- <transition event="e8" sourceRef="Error" targetRef="Waiting">
  <outputAction ident="o1.z9" />
    </transition>
- <transition event="e9" sourceRef="Waiting" targetRef="Error">
  <outputAction ident="o1.z4" />
    </transition>
```

```
<transition event="e3" sourceRef="Waiting" targetRef="TagReceiving" />
- <transition event="e9" sourceRef="TagReceiving" targetRef="Error">
<outputAction ident="o1.z4" />
</transition>
- <transition event="e4" sourceRef="TagReceiving" targetRef="Waiting">
<outputAction ident="o1.z5" />
</transition>
<transition event="e2" sourceRef="TagReceiving" targetRef="Play" />
- <transition event="e7" guard="o1.x4 > 0 && o1.x1 <= 0" sourceRef="Play"
targetRef="Error">
<outputAction ident="o1.z6" />
</transition>
- <transition event="e7" guard="o1.x4<=0" sourceRef="Play" targetRef="Waiting">
<outputAction ident="o1.z8" />
</transition>
- <transition event="e7" guard="o1.x4 >0 && o1.x1>0" sourceRef="Play"
targetRef="givingWin">
<outputAction ident="o1.z7" />
</transition>
- <transition event="e5" guard="o1.x2>0 && o1.x1 <= 0" sourceRef="givingWin"
targetRef="Error">
<outputAction ident="o1.z6" />
</transition>
- <transition event="e5" guard="o1.x2 <= 0" sourceRef="givingWin" targetRef="Waiting">
<outputAction ident="o1.z8" />
</transition>
- <transition event="e5" guard="o1.x2 > 0 && o1.x1>0" sourceRef="givingWin"
targetRef="givingWin">
<outputAction ident="o1.z7" />
</transition>
<transition sourceRef="s1" targetRef="Waiting" />
</stateMachine>
- <stateMachine name="A3">
<configStore class="com.evelopers.unimod.runtime.config.DistinguishConfigManager" />
<association clientRole="A3" supplierRole="o1" targetRef="o1" />
- <state name="Top" type="NORMAL">
<state name="s2" type="FINAL" />
<state name="Empty" type="NORMAL" />
<state name="TrueCoinReceived" type="NORMAL" />
<state name="s1" type="INITIAL" />
</state>
<transition event="e3" guard="o1.x3 <= 0" sourceRef="Empty" targetRef="s2" />
- <transition event="e3" guard="o1.x3>0" sourceRef="Empty" targetRef="TrueCoinReceived">
<outputAction ident="o1.z2" />
</transition>
<transition event="e3" guard="o1.x3 <= 0" sourceRef="TrueCoinReceived" targetRef="s2" />
- <transition event="e3" guard="o1.x3>0" sourceRef="TrueCoinReceived"
targetRef="TrueCoinReceived">
<outputAction ident="o1.z2" />
</transition>
<transition sourceRef="s1" targetRef="Empty" />
</stateMachine>
</model>
```

Приложение 2. Исходные коды

SMEventProvider.java

```
package ru.ifmo.slotmachine;

import com.evelopers.common.exception.CommonException;
import com.evelopers.unimod.core.stateworks.Event;
import com.evelopers.unimod.runtime.EventProvider;
import com.evelopers.unimod.runtime.ModelEngine;
import com.evelopers.unimod.runtime.context.StateMachineContextImpl;

public class SMEventProvider implements EventProvider {

    /**
     * @unimod.event.descr Стартовое событие
     */
    public static final String E1 = "e1";

    /**
     * @unimod.event.descr Нажата кнопка "ИГРА"
     */
    public static final String E2 = "e2";

    /**
     * @unimod.event.descr Нажата кнопка "СТАВКА"
     */
    public static final String E3 = "e3";

    /**
     * @unimod.event.descr Нажата кнопка "ВОЗВРАТ"
     */
    public static final String E4 = "e4";

    /**
     * @unimod.event.descr Сообщение о выдаче выигрыша
     */
    public static final String E5 = "e5";

    /**
     * @unimod.event.descr Остановка барабана
     */
    public static final String E6 = "e6";

    /**
     * @unimod.event.descr Все барабаны остановлены
     */
    public static final String E7 = "e7";

    /**
     * @unimod.event.descr Нажата кнопка "СБРОС"
     */
    public static final String E8 = "e8";

    /**
     * @unimod.event.descr Опущена некорректная монета
     */
    public static final String E9 = "e9";

    SMControlledObj SM;

    public void init(ModelEngine engine) throws CommonException {
        SM = (SMControlledObj) engine.getControlledObjectsManager()
                .getControlledObject("o1");// SM - объект управления для автомата
```

```
        SM.setEngine(engine);// Указание объекту управления на управляющий
автомат

        SM.z1(StateMachineContextImpl.create());
        engine.getEventManager().handle(new Event(E1),
                StateMachineContextImpl.create());// Генерация инициирующего
события для начала работы

    }

    public void dispose() {
    }

}
```

SMControlledObj.java

```
package ru.ifmo.slotmachine;

import java.applet.Applet;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.net.URL;
import java.util.Date;
import java.util.Random;

import javax.media.j3d.AmbientLight;
import javax.media.j3d.Appearance;
import javax.media.j3d.BoundingSphere;
import javax.media.j3d.BranchGroup;
import javax.media.j3d.Canvas3D;
import javax.media.j3d.DirectionalLight;
import javax.media.j3d.Material;
import javax.media.j3d.Texture;
import javax.media.j3d.TextureAttributes;
import javax.media.j3d.Transform3D;
import javax.media.j3d.TransformGroup;
import javax.vecmath.AxisAngle4f;
import javax.vecmath.Color3f;
import javax.vecmath.Color4f;
import javax.vecmath.Vector3f;

import com.evelopers.unimod.core.stateworks.Event;
import com.evelopers.unimod.runtime.ControlledObject;
import com.evelopers.unimod.runtime.ModelEngine;
import com.evelopers.unimod.runtime.context.StateMachineContext;
import com.evelopers.unimod.runtime.context.StateMachineContextImpl;
import com.sun.j3d.utils.applet.MainFrame;
import com.sun.j3d.utils.geometry.Cylinder;
import com.sun.j3d.utils.geometry.Primitive;
import com.sun.j3d.utils.image.TextureLoader;
import com.sun.j3d.utils.universe.SimpleUniverse;

public class SMControlledObj extends Applet implements ControlledObject {

    private static final long serialVersionUID = 8773383499438061758L;

    private ModelEngine engine;// Указатель на управляющий автомат

    private int stake;// Содержит значение ставки

    private int bank;// Содержит значение банка

    private int winsize;// Содержит значение выигрыша
```

```
    private int usermoney;// Содержит количесто денег игрока

    private TransformGroup[] objLocalRotAr;// Массив из трансформ-групп для
вращения барабанов

    private Label lStakeDyn;// Табло значения ставки

    private Label lWinDyn;// Табло значения выигрыша

    private Label lBankDyn;// Табло значения банка

    private Label lWalDyn;// Табло количества денег пользователя

    private Label lWalFalse;// Табло вывода ошибок

    public int[] current;// Массив с последней выпавшей на барабанах комбинацией

    /*
     * Указание объекту управления на управляющий автомат
     */
    public void setEngine(ModelEngine engine) {
        this.engine = engine;
    }

    /*
     * Обновление всех табло
     */
    public void refreshLabels() {
        lBankDyn.setText("" + bank);
        lStakeDyn.setText("" + stake);
        lWinDyn.setText("" + winsize);
        lWalDyn.setText("" + usermoney);
    }

    /**
     * @unimod.action.descr Инициализация
     */
    public void z1(StateMachineContext context) {

        // Выставление начальных значений
        stake = 0;
        bank = 50;
        winsize = 0;
        usermoney = 30;

        // Создание всех табло
        lBankDyn = new Label();
        lStakeDyn = new Label();
        lWinDyn = new Label();
        lWalDyn = new Label();
        lWalFalse = new Label("");// В начале выполнения ошибок нет
        lWalFalse.setForeground(new Color(255, 0, 0));// Установка атрибута для
табло ошибок: красный цвет текста
        refreshLabels();// Отображение начальных значений

        /*
         * Работа с 3D графикой
         */

        // Цветовые константы
        final Color3f white = new Color3f(1.0f, 1.0f, 1.0f);// белый цвет
        final Color3f grey = new Color3f(1.0f, 1.0f, 1.0f);// серый (70%) цвет

        // Создание корня графа сцены
        BranchGroup objRoot = new BranchGroup();
```

```
        // Работа со светом
        BoundingSphere bounds = new BoundingSphere();// Создание границы
освещенности
        DirectionalLight light = new DirectionalLight(white, new Vector3f(1f,
                -1f, -1f));// Создание направленного света
        light.setInfluencingBounds(bounds);// Установка границы освещенности для
направленного света
        objRoot.addChild(light);// Добавление направленного света к графу сцены

        AmbientLight ambientLightNode = new AmbientLight(grey);// Создание общей
освещенности
        ambientLightNode.setInfluencingBounds(bounds);// Установка границы для
общей освешенности
        objRoot.addChild(ambientLightNode);// Добавление общей освещённости

        // Загрузка текстуры барабанов
        URL fileName = SMControlledObj.class.getResource("tex.jpg");//
Определение полного пути к файлу с текстурой
        TextureLoader loader = new TextureLoader(fileName, "RGBA",
                new Container());// Загрузка файла текстуры
        Texture texture = loader.getTexture();// Создание текстуры
        texture.setBoundaryModeS(Texture.WRAP);// Задание параметров наложения
текстуры
        texture.setBoundaryModeT(Texture.WRAP);
        texture.setBoundaryColor(new Color4f(0.0f, 1.0f, 0.0f, 0.0f));// Задание
цвета для подсветки текстуры
        TextureAttributes texAttr = new TextureAttributes();// Создание
контейнера для атрибутов текстуры
        texAttr.setTextureMode(TextureAttributes.MODULATE);// Задание атрибутов

        // Задание внешнего вида и материала баробанов
        Appearance ap = new Appearance();// Создание контейнера внешнего вида
барабана
        ap.setTexture(texture);// Установка текстуры
        ap.setTextureAttributes(texAttr);// Установка атрибутов текстуры

        ap.setMaterial(new Material(white, grey, white, white, 20.0f));//
Установка материала с заданными параметрами
        int primflags = Primitive.GENERATE_NORMALS
                + Primitive.GENERATE_TEXTURE_COORDS;// указание на наличие
текстуры и отражающей поверхности

        // Формирование групп для вращения и смещения объектов
        TransformGroup objGlobalRotate = new TransformGroup();// Создание группы
трансформации для глобального поворота всех барабанов
        Transform3D transGlobalRot = new Transform3D();// Создание конейнера для
глобального  поворота
        transGlobalRot.setRotation(new AxisAngle4f(0.0f, 0.0f, -1.0f, 1.57f));//
Установка угла поворота
        objGlobalRotate.setTransform(transGlobalRot);// Выполнение поворота
        objRoot.addChild(objGlobalRotate);// Добавление группы в граф сцены

        TransformGroup[] objLocalTransAr = new TransformGroup[3];// Создание
массива групп трансформаций для смещения барабанов
        objLocalRotAr = new TransformGroup[3];// Создание массива групп
трансформаций для вращения барабанов
        current = new int[3];// Создание массива значений выпавших комбинаций

        // Достроение графа сцены
        for (int i = 0; i < 3; i++) {
            objLocalTransAr[i] = new TransformGroup();// Создание трансформации-
смещения для i-го барабана

            Transform3D transLocalTrans = new Transform3D();// Создание
контейнера трансформации для смещения
            transLocalTrans.setTranslation(new Vector3f(0.0f, -0.55f + i
                    * 0.55f, 0.0f));// Установка вектора смещения
```

```
            objLocalTransAr[i].setTransform(transLocalTrans);// Выполнение
            objGlobalRotate.addChild(objLocalTransAr[i]);// Добавление смещения
для i-го барабана

            objLocalRotAr[i] = new TransformGroup();// Создание трансформации-
вращения для i-го барабана

            objLocalRotAr[i]
                    .setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);//
Установка разрешение на вращение

            Transform3D transLocalRot = new Transform3D();// Создание контейнера
для вращения
            transLocalRot.setRotation(new AxisAngle4f(0.0f, 1.0f, 0.0f,
                    angleForDigit(7)));// Выполнение вращения барабана до его
начального значения
            objLocalRotAr[i].setTransform(transLocalRot);
            objLocalTransAr[i].addChild(objLocalRotAr[i]);// Добавление вращения
для i-го барабана

            objLocalRotAr[i].addChild(new Cylinder(0.5f, 0.5f, primflags, ap));//
Создание i-го барабана смещения
            current[i] = 7;// Выставление начального значения для i-го барабана
        }

        /*
         * Создание и расстановка элементов управления и холста 3D-сцены
         */

        GraphicsConfiguration config = SimpleUniverse
                .getPreferredConfiguration();// Создание конфигурации отображения
3D-сцены
        Canvas3D canvas = new Canvas3D(config);// Создание холста 3D-сцены

        // Создание кнопок
        Button bRun = new Button("ИГРА");// Создание кнопки вращения барабана
        bRun.setBackground(new Color(0, 200, 0));// Установка фонового цвета
кнопки, зеленый 78%
        bRun.setFont(new Font("Dialog", Font.BOLD, 15));// Установка параметров
шрифта, Dialog,жирный,15п
        bRun.setForeground(new Color(255, 255, 0));// Установка цвета шрифта
кнопки, желтый

        Button bCancel = new Button("ВОЗВРАТ");// Создание кнопки возврата ставки
        bCancel.setBackground(new Color(200, 0, 0));// Установка фонового цвета
кнопки, красный 78%
        bCancel.setFont(new Font("Dialog", Font.BOLD, 15));// Установка
параметров шрифта, Dialog,жирный,15п
        bCancel.setForeground(new Color(255, 255, 0));// Установка цвета шрифта
кнопки, желтый

        Button bStake = new Button("СТАВКА");// Создание кнопки увеличения ставки
        bStake.setFont(new Font("Dialog", Font.BOLD, 15));// Установка параметров
шрифта, Dialog,жирный,15п
        bStake.setForeground(new Color(0, 0, 0));// Установка цвета шрифта
кнопки, черный

        Button bFalse = new Button("ПЛОХАЯ МОНЕТА");// Создание кнопки опускания
некорректной монеты
        bFalse.setFont(new Font("Dialog", Font.BOLD, 15));// Установка параметров
шрифта, Dialog,жирный,15п
        bFalse.setForeground(new Color(0, 0, 0));// Установка цвета шрифта
кнопки, черный

        Button bReset = new Button("Сброс");
        bReset.setForeground(new Color(0, 0, 0));
```

```
        // Описание реакций на нажатие
        bRun.addActionListener(new ActionListener() {// Установка реакции на
нажатие кнопки вращения барабанов
                    public void actionPerformed(ActionEvent e) {
                        ModelEngine engine = SMControlledObj.this.engine;//
Установка автомата-получателя события
                        engine.getEventManager().handle(
                                new Event(SMEventProvider.E2),
                                StateMachineContextImpl.create());// Генерация
события "e2"
                    }
                });
        bCancel.addActionListener(new ActionListener() {// Установка реакции на
нажатие кнопки возврата ставки
                    public void actionPerformed(ActionEvent e) {
                        ModelEngine engine = SMControlledObj.this.engine;//
Установка автомата-получателя события
                        engine.getEventManager().handle(
                                new Event(SMEventProvider.E4),
                                StateMachineContextImpl.create());// Генерация
события "e4"
                    }
                });
        bStake.addActionListener(new ActionListener() {// Установка реакции на
нажатие кнопки увеличения ставки
                    public void actionPerformed(ActionEvent e) {
                        ModelEngine engine = SMControlledObj.this.engine;//
Установка автомата-получателя события
                        engine.getEventManager().handle(
                                new Event(SMEventProvider.E3),// Генерация
события "e3"
                                StateMachineContextImpl.create());
                    }
                });
        bFalse.addActionListener(new ActionListener() {// Установка реакции на
нажатие кнопки опускания плохой монеты
                    public void actionPerformed(ActionEvent e) {
                        ModelEngine engine = SMControlledObj.this.engine;//
Установка автомата-получателя события
                        engine.getEventManager().handle(
                                new Event(SMEventProvider.E9),// Генерация
события "e9"
                                StateMachineContextImpl.create());
                    }
                });
        bReset.addActionListener(new ActionListener() {// Установка реакции на
нажатие кнопки "Технический сброс"
                    public void actionPerformed(ActionEvent e) {
                        ModelEngine engine = SMControlledObj.this.engine;//
Установка автомата-получателя события
                        engine.getEventManager().handle(
                                new Event(SMEventProvider.E8),// Генерация
события "e9"
                                StateMachineContextImpl.create());
                    }
                });

        // Размещение объектов интерфейса
        setLayout(new BorderLayout());// Установка типа размещения в главном окне
- по краям
        add("Center", canvas);// Размещение холста 3D-сцены в центр окна
        Panel panelHalf = new Panel();// Создание панели с табло и кнопками
        panelHalf.setBackground(new Color(0, 0, 0));// Установка цвета панели,
черный
        panelHalf.setForeground(new Color(255, 255, 255));// Усановка цвета
текста панели
        Panel panelLab = new Panel();// Создание дочерней панели с табло
```

```
        panelLab.setLayout(new GridLayout());// Установка типа размещения в
панели с табло - по ячейкам
        Panel pBank = new Panel();// Создание панели с табло банка
        pBank.setLayout(new BorderLayout());// Установка типа размещения в панели
с табло банка - по краям
        Label lBankStatic = new Label("Банк:");// Создание подписи к табло банка
        lBankStatic.setFont(new Font("Dialog", Font.BOLD, 15));// Установка
параметров шрифта подписи, Dialog,жирный,15п
        lBankDyn.setFont(new Font("Dialog", Font.BOLD, 20));// Установка
параметров шрифта табло, Dialog,жирный,20п
        lBankDyn.setAlignment(Label.CENTER);// Установка выравнивания текста
табло банка - по центру
        pBank.add("North", lBankStatic);// Установка подписи над табло банка
        pBank.add("Center", lBankDyn);// Установка табло банка
        Panel pStake = new Panel();// Создание панели с табло ставки
        pStake.setLayout(new BorderLayout());// Установка типа размещения в
панели с табло ставки - по краям
        Label lStakeStatic = new Label("Ставка:");// Создание подписи к табло
ставка
        lStakeStatic.setFont(new Font("Dialog", Font.BOLD, 15));// Установка
параметров шрифта подписи, Dialog,жирный,15п
        lStakeDyn.setFont(new Font("Dialog", Font.BOLD, 20));// Установка
параметров шрифта табло, Dialog,жирный,20п
        lStakeDyn.setAlignment(Label.CENTER);// Установка выравнивания текста
табло ставки - по центру
        pStake.add("North", lStakeStatic);// Установка подписи над табло ставки
        pStake.add("Center", lStakeDyn);// Установка табло ставки
        Panel pWin = new Panel();// Создание панели с табло выигрыша
        pWin.setLayout(new BorderLayout());// Установка типа размещения в панели
с табло выигрыша - по краям
        Label lWinStatic = new Label("Выигрыш:");// Создание подписи к табло
выигрыша
        lWinStatic.setFont(new Font("Dialog", Font.BOLD, 15));// Установка
параметров шрифта подписи, Dialog,жирный,15п
        lWinDyn.setFont(new Font("Dialog", Font.BOLD, 20));// Установка
параметров шрифта табло, Dialog,жирный,20п
        lWinDyn.setAlignment(Label.CENTER);// Установка выравнивания текста табло
выигрыша - по центру
        pWin.add("North", lWinStatic);// Установка подписи над табло ставки
        pWin.add("Center", lWinDyn);// Установка табло ставки
        panelLab.add(pBank);// Добавление панели банка
        panelLab.add(pStake);// Добавление панели ставки
        panelLab.add(pWin);// Добавление панели выигрыша
        Panel panelBut = new Panel();// Создание панели с кнопками
        panelBut.setLayout(new FlowLayout());// Установка типа размещения в
панели с кнопками - плавающее
        panelBut.add(bRun);// Добавление кнопки начала вращения барабанов
        panelBut.add(bStake);// Добавление кнопки увеличения ставки
        panelBut.add(bFalse);// Добавление кнопки опускания некорректной монет
        panelBut.add(bCancel);// Добавление кнопки возврата ставки
        Panel panelWal = new Panel();// Создание панели с табло денег игрока и
табло ошибок
        panelWal.setLayout(new GridLayout());// Установка типа размещения в
панели - по ячейкам
        Label lWalStat = new Label("Ваши деньги: ");// Создание подписи к табло
денег игрока
        panelWal.add(lWalStat);// Добавление подписи к табло
        panelWal.add(lWalDyn);// Добавление абло денег игрока
        panelWal.add(lWalFalse);// Добавление табло ошибок
        panelHalf.setLayout(new BorderLayout());// Установка размещения в панели
табло и кнопок - по краям
        panelHalf.add("North", panelLab);// Добавление панели табло
        panelHalf.add("Center", panelWal);// Добавление панели денег игрока и
ошибок
        panelHalf.add("South", panelBut);// Добавление панели с кнопками
        add("South", panelHalf);// Добавление панели табло и кнопок
```

```
        add("North", bReset);// Добавление кнопки "Технический сброс" Вставка 3D-
сцены в холст
        objRoot.compile();// оптимизация сцены
        SimpleUniverse universe = new SimpleUniverse(canvas);// Создание
конейнера сцены и ассоциация его с холстом
        universe.getViewingPlatform().setNominalViewingTransform();// Установка
точки осмотра сцены
        universe.addBranchGraph(objRoot);// Добавление графа сцены в контейнер

        /*
         * Создания и установка параметров окна приложения
         */

        MainFrame main = new MainFrame(this, 400, 400);// Создание окна
приложения размером 400x400
        main.setTitle("Однорукий бандит");// Установка заголовка окна

    }

    /**
     * @unimod.action.descr Увеличить ставку
     */
    public void z2(StateMachineContext context) {
        usermoney--;
        stake++;
        bank++;
        this.refreshLabels();
    }

    public static float angleForDigit(int dig) {
        return 0.3f + 2 * 0.31415f * (5 - dig);// Вычисление угла поворота для
отображение заданной цифры
    }

    /**
     * @unimod.action.descr Врашать барабаны
     */
    public void z3(StateMachineContext context) {
        ModelEngine engine = SMControlledObj.this.engine;// Установка автомата-
получателя события
        Date date = new Date();
        int seed = (int) date.getTime();// Вычисление зерна для случайного числа
по дате
        Random rand = new Random(seed);// Создание "черного ящика" случайных
чисел
        int[] rec = new int[3];// Массив для сохранения результатов вращения
        for (int i = 0; i < 3; i++) {
            rec[i] = rand.nextInt();// Вычисление результата вращения
            rec[i] = Math.abs(rec[i]) % 10;// Результатом вращеня должно быть три
цифры
        }

        Transform3D[] transLocalRot = new Transform3D[3];// Создание конейнеров
трансформации для вращения барабанов

        float[] angleStep = new float[3];// Массив для поворота барабанов за одну
итерацию

        for (int i = 0; i < 3; i++) {
            transLocalRot[i] = new Transform3D();// Инициализация трансформации
            transLocalRot[i].setRotation(new AxisAngle4f(0.0f, 1.0f, 0.0f,
                    angleForDigit(current[i])));// Установка параметров вращения
            objLocalRotAr[i].setTransform(transLocalRot[i]);// Вращение барабанов
        }
        int rotationSteps = 500;// Количество итераций для вращения первого
барабана
```

```
        int rotationStepsDif = 200;// Количесво итераций, на которое каждый
следующий барабан вращается дольше

        for (int k = 0; k < rotationSteps + 2 * rotationStepsDif; k++) {
            for (int i = 0; i < 3; i++) {
                angleStep[i] = angleForDigit(rec[i])
                        - angleForDigit(current[i]);
                angleStep[i] = (angleStep[i] + 8 * 3.1415f)
                        / (rotationSteps + i * rotationStepsDif);// Вычисление
дельты угла поворота

                if (k == rotationSteps + i * rotationStepsDif) {
                    engine.getEventManager().handle(
                            new Event(SMEventProvider.E6),
                            StateMachineContextImpl.create());// Генерация
события "e6" после остановки барабана
                }
                if (k < rotationSteps + i * rotationStepsDif) {
                    transLocalRot[i]
                            .setRotation(new AxisAngle4f(0.0f, 1.0f, 0.0f,
                                    angleForDigit(current[i]) + angleStep[i]
                                            * k));
                    objLocalRotAr[i].setTransform(transLocalRot[i]);
                }
            }
            try {
                Thread.sleep(7);// Задержка между итерациями для видимости
вращения
            } catch (InterruptedException e) {
            }
        }

        for (int i = 0; i < 3; i++) {
            current[i] = rec[i];
        }
        engine.getEventManager().handle(new Event(SMEventProvider.E6),
                StateMachineContextImpl.create());// Генерация события "e6" после
остановки последнего барабана

        engine.getEventManager().handle(new Event(SMEventProvider.E7),
                StateMachineContextImpl.create());// Генерация события "e7" -
конец вращения

    }

    /**
     * @unimod.action.descr Ставка некорректной монетой
     */
    public void z4(StateMachineContext context) {
        lWalFalse.setText("Некорректная монета!");// Отображение текста ошибки в
табло ошибок
    }

    /**
     * @unimod.action.descr Возврат ставки
     */
    public void z5(StateMachineContext context) {
        bank -= stake;// Изъятие денег у банка
        usermoney += stake;// Возврат денег пользователю
        stake = 0;// Обнуление ставки
        winsize = 0;// Обнуление выигрыша
        refreshLabels();// Обновление табло
        lWalFalse.setText("");
    }

    /**
     * @unimod.action.descr Сообощить о нехватке денег в банке
```

```
     */
    public void z6(StateMachineContext context) {
        lWalFalse.setText("Денег нет!");// Отображение текста ошибки в табло
ошибок
    }

    /**
     * @unimod.action.descr Выдать монеу из банка
     */
    public void z7(StateMachineContext context) {
        bank--;
        winsize--;
        usermoney++;
        refreshLabels();

        try {
            Thread.sleep(200);// Пауза в выдаче монет
        } catch (InterruptedException e) {
        }

        ModelEngine engine = SMControlledObj.this.engine;
        engine.getEventManager().handle(new Event(SMEventProvider.E5),//
Генерация события "e5" - выдать следующую монету
                StateMachineContextImpl.create());

    }

    /**
     * @unimod.action.descr Обнулить табло ставки
     */
    public void z8(StateMachineContext context) {
        stake = 0;
        refreshLabels();
    }

    /**
     * @unimod.action.descr Значение банка
     */
    public int x1(StateMachineContext context) {
        return bank;
    }

    /**
     * @unimod.action.descr Значение выигрыша
     */
    public int x2(StateMachineContext context) {
        return winsize;
    }

    /**
     * @unimod.action.descr Количество монет игрока
     */
    public int x3(StateMachineContext context) {
        return usermoney;
    }

    /**
     * @unimod.action.descr Результат вращения барабанов
     */
    public int x4(StateMachineContext context) {
        if (current[0] == current[1] || current[1] == current[2]
                || current[0] == current[2]) {// Если две одинаковых цифры -
выигрыш
            if (current[0] == current[1] && current[1] == current[2])// Если все
три цифры одинаковы
                winsize = stake * 5;// ставка умножается на 5
            else
```

```
                winsize = stake * 2;// иначе ставка умножается на 2
            refreshLabels();

            return 1;// Выигрыш
        } else
            return 0;// Проигрыш
    }

    /**
     * @unimod.action.descr Сделать технический сброс
     */
    public void z9(StateMachineContext context) {
        usermoney += winsize;
        winsize = 0;
        bank = 50;
        stake = 0;
        lWalFalse.setText("");
        refreshLabels();
    }
}
```

Приложение 3. Сгенерированный по *XML*-описанию *Java*-код

Model1EventProcessor.java

```
/**
  * This file was generated from model [Model1] on [Fri Oct 13 03:21:13 MSD
2006].
  * Do not change content of this file.
  */

import java.io.IOException;
import java.util.*;

import org.apache.commons.lang.BooleanUtils;
import org.apache.commons.lang.math.NumberUtils;
import org.apache.commons.lang.StringUtils;
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;

import com.evelopers.common.exception.*;
import com.evelopers.unimod.core.stateworks.*;
import com.evelopers.unimod.debug.app.AppDebugger;
import com.evelopers.unimod.debug.protocol.JavaSpecificMessageCoder;
import com.evelopers.unimod.runtime.*;
import com.evelopers.unimod.runtime.context.*;
import com.evelopers.unimod.runtime.logger.SimpleLogger;


public class Model1EventProcessor extends AbstractEventProcessor {

    private ModelStructure modelStructure;

                private static final int A2 = 1;
            private static final int A1 = 2;
            private static final int A3 = 3;

    private int decodeStateMachine(String sm) {

        if ("A2".equals(sm)) {
            return A2;
                        } else
```

```
        if ("A1".equals(sm)) {
            return A1;
                        } else

        if ("A3".equals(sm)) {
            return A3;
            }

        return -1;
    }

            private A2EventProcessor _A2;
            private A1EventProcessor _A1;
            private A3EventProcessor _A3;

    public Model1EventProcessor() {
        modelStructure = new Model1ModelStructure();

                    _A2 = new A2EventProcessor();
                    _A1 = new A1EventProcessor();
                    _A3 = new A3EventProcessor();
            }

            public static void run(int debuggerPort, boolean debuggerSuspend)
throws
                                    InterruptedException,
EventProcessorException, CommonException,
                        IOException {

                /* Create runtime engine */
                ModelEngine engine = createModelEngine(true);

                /* Setup logger */
                final Log log = LogFactory.getLog(Model1EventProcessor.class);
                engine.getEventProcessor().addEventProcessorListener(new
SimpleLogger(log));

                /* Setup exception handler */
                engine.getEventProcessor().addExceptionHandler(new
ExceptionHandler() {
                    public void handleException(StateMachineContext context,
SystemException e) {
                                log.fatal(e.getChainedMessage(),
e.getRootException());
                    }
                });

                if (debuggerPort > 0) {
                        AppDebugger d = new AppDebugger(
                                        debuggerPort, debuggerSuspend,
                                        new JavaSpecificMessageCoder(), engine);
                                d.start();
                    }
                    engine.start();
            }

        public static void main(String[] args) throws Exception {
                int debuggerPort =
NumberUtils.stringToInt(System.getProperty("debugger.port"), -1);
                boolean debuggerSuspend =
BooleanUtils.toBoolean(System.getProperty("debugger.suspend"));
                Model1EventProcessor.run(debuggerPort, debuggerSuspend);
        }

        public static ModelEngine createModelEngine(boolean useEventQueue) throws
CommonException {
            ObjectsManager objectsManager = new ObjectsManager();
```

```
            return ModelEngine.createStandAlone(
                        useEventQueue ? (EventManager) new QueuedHandler() :
(EventManager) new StrictHandler(),
                        new Model1EventProcessor(),
                    objectsManager.getControlledObjectsManager(),
                    objectsManager.getEventProvidersManager());
        }

        public static class ObjectsManager {
            private ru.ifmo.slotmachine.SMControlledObj o1 = null;
            private ru.ifmo.slotmachine.SMEventProvider p1 = null;

            private ControlledObjectsManager controlledObjectsManager = new
ControlledObjectsManagerImpl();
            private EventProvidersManager eventProvidersManager = new
EventProvidersManagerImpl();

            public ControlledObjectsManager getControlledObjectsManager() {
                return controlledObjectsManager;
            }

            public EventProvidersManager getEventProvidersManager() {
                return eventProvidersManager;
            }

            private class ControlledObjectsManagerImpl implements
ControlledObjectsManager {
                public void init(ModelEngine engine) throws CommonException {}

                public void dispose() {}

                public ControlledObject getControlledObject(String coName) {
                if (StringUtils.equals(coName, "o1")) {
                            if (o1 == null) {
                                o1 = new ru.ifmo.slotmachine.SMControlledObj();
                            }
                            return o1;
                }
                    throw new IllegalArgumentException("Controlled object with
name [" + coName + "] wasn't found");
                }
            }

            private class EventProvidersManagerImpl implements
EventProvidersManager {
                private List nonameEventProviders = new ArrayList();

                public void init(ModelEngine engine) throws CommonException {
                    EventProvider ep;
                    ep = getEventProvider("p1");
                ep.init(engine);
                }

                public void dispose() {
                        EventProvider ep;
                        ep = getEventProvider("p1");
                ep.dispose();
                for (Iterator i = nonameEventProviders.iterator(); i.hasNext();)
{
                    ep = (EventProvider) i.next();
                    ep.dispose();
                }
            }

                public EventProvider getEventProvider(String epName) {
                    if (StringUtils.equals(epName, "p1")) {
                            if (p1 == null) {
```

```
                                p1 = new ru.ifmo.slotmachine.SMEventProvider();
                            }
                            return p1;
                }
                throw new IllegalArgumentException("Event provider with name [" +
epName + "] wasn't found");
                }
            }
        }

    public ModelStructure getModelStructure() {
        return modelStructure;
    }

    public void setControlledObjectsMap(ControlledObjectsMap
controlledObjectsMap) {
        super.setControlledObjectsMap(controlledObjectsMap);

                    _A2.init(controlledObjectsMap);
                    _A1.init(controlledObjectsMap);
                    _A3.init(controlledObjectsMap);
            }

    protected StateMachineConfig process(
            Event event, StateMachineContext context,
            StateMachinePath path, StateMachineConfig config) throws
SystemException {

        // get state machine from path
        int sm = decodeStateMachine(path.getStateMachine());

        try {
        switch (sm) {
                    case A2:
                return _A2.process(event, context, path, config);
                    case A1:
                return _A1.process(event, context, path, config);
                    case A3:
                return _A3.process(event, context, path, config);
                    default:
                throw new EventProcessorException("Unknown state machine [" +
path.getStateMachine() + "]");
        }
        } catch (Exception e) {
            if (e instanceof SystemException) {
                throw (SystemException)e;
            } else {
                throw new SystemException(e);
            }
        }
    }

    protected StateMachineConfig transiteToStableState(
            StateMachineContext context,
            StateMachinePath path, StateMachineConfig config) throws
SystemException {

        // get state machine from path
        int sm = decodeStateMachine(path.getStateMachine());

        try {
        switch (sm) {
                    case A2:
                return _A2.transiteToStableState(context, path, config);
                    case A1:
                return _A1.transiteToStableState(context, path, config);
                    case A3:
```

```
                return _A3.transiteToStableState(context, path, config);
                    default:
                throw new EventProcessorException("Unknown state machine [" +
path.getStateMachine() + "]");
        }
        } catch (Exception e) {
            if (e instanceof SystemException) {
                throw (SystemException)e;
            } else {
                throw new SystemException(e);
            }
        }
    }


private class Model1ModelStructure implements ModelStructure {
        private Map configManagers = new HashMap();

    private Model1ModelStructure() {
                configManagers.put("A2", new
com.evelopers.unimod.runtime.config.DistinguishConfigManager());
                configManagers.put("A1", new
com.evelopers.unimod.runtime.config.DistinguishConfigManager());
                configManagers.put("A3", new
com.evelopers.unimod.runtime.config.DistinguishConfigManager());
            }

        public StateMachinePath getRootPath()
                throws EventProcessorException {
                return new StateMachinePath("A1");
        }

        public StateMachineConfigManager getConfigManager(String stateMachine)
                throws EventProcessorException {
                return
(StateMachineConfigManager)configManagers.get(stateMachine);
        }

        public StateMachineConfig getTopConfig(String stateMachine)
                throws EventProcessorException {
                int sm = decodeStateMachine(stateMachine);

                switch (sm) {
                                case A2:
                                return new StateMachineConfig("Top");
                                case A1:
                                return new StateMachineConfig("Top");
                                case A3:
                                return new StateMachineConfig("Top");
                                default:
                                throw new EventProcessorException("Unknown state
machine [" + stateMachine + "]");
                }
        }

        public boolean isFinal(String stateMachine, StateMachineConfig config)
                throws EventProcessorException {
                /* Get state machine from path */
                int sm = decodeStateMachine(stateMachine);
                int state;
                        switch (sm) {
                                case A2:
                                state = _A2.decodeState(config.getActiveState());
                                switch (state) {

case A2EventProcessor.s2:
```

```
return true;
                                                        default:
                                                    return false;
                    }
                                case A1:
                                state = _A1.decodeState(config.getActiveState());
                                switch (state) {

default:
                                                    return false;
                    }
                                case A3:
                                state = _A3.decodeState(config.getActiveState());
                                switch (state) {

case A3EventProcessor.s2:

return true;
                                                        default:
                                                    return false;
                    }
                                default:
                                throw new EventProcessorException("Unknown state
machine [" + stateMachine + "]");
                }
        }
}


private class A2EventProcessor {

        // states
        private static final int Top = 1;
        private static final int s1 = 2;
        private static final int s2 = 3;
        private static final int Stopped = 4;
        private static final int Rotation = 5;
        private static final int StoppingFirst = 6;
        private static final int StoppingSecond = 7;

    private int decodeState(String state) {

        if ("Top".equals(state)) {
            return Top;
                        } else

        if ("s1".equals(state)) {
            return s1;
                        } else

        if ("s2".equals(state)) {
            return s2;
                        } else

        if ("Stopped".equals(state)) {
            return Stopped;
                        } else

        if ("Rotation".equals(state)) {
            return Rotation;
                        } else

        if ("StoppingFirst".equals(state)) {
            return StoppingFirst;
                        } else
```

```
        if ("StoppingSecond".equals(state)) {
            return StoppingSecond;
            }

        return -1;
    }

        // events
        private static final int e2 = 1;
        private static final int e6 = 2;

    private int decodeEvent(String event) {

        if ("e2".equals(event)) {
            return e2;
                        } else

        if ("e6".equals(event)) {
            return e6;
                }

        return -1;
    }

                              private ru.ifmo.slotmachine.SMControlledObj o1;

    private void init(ControlledObjectsMap controlledObjectsMap) {
                                        o1 =
(ru.ifmo.slotmachine.SMControlledObj)controlledObjectsMap.getControlledObject("o1
");
                          }

    private StateMachineConfig process(Event event, StateMachineContext context,
StateMachinePath path, StateMachineConfig config) throws Exception {
        config = lookForTransition(event, context, path, config);

        config = transiteToStableState(context, path, config);

        // execute included state machines
        executeSubmachines(event, context, path, config);

        return config;
    }

    private void executeSubmachines(Event event, StateMachineContext context,
StateMachinePath path, StateMachineConfig config) throws Exception {
        int state = decodeState(config.getActiveState());

        while (true) {
        switch (state) {
                                                                        case
s1:

                                                 return;

case s2:

                                                 return;

case Stopped:

                                                 return;

case Rotation:

                                                 return;
```

```
case StoppingFirst:

                                                return;

case StoppingSecond:

                                                return;
                                                                default:
                        throw new EventProcessorException("State with name [" +
config.getActiveState() + "] is unknown for state machine [A2]");
        }
        }
    }

    private StateMachineConfig transiteToStableState(StateMachineContext context,
StateMachinePath path, StateMachineConfig config) throws Exception {

        int s = decodeState(config.getActiveState());
        Event event;

        switch (s) {
                                        case Top:


                    fireComeToState(context, path, "s1");

                    // s1->Stopped [true]/
                    event = Event.NO_EVENT;
                        fireTransitionFound(context, path, "s1", event,
"s1#Stopped##true");


    fireComeToState(context, path, "Stopped");

    // Stopped []

                                                return new
StateMachineConfig("Stopped");

}

        return config;
    }

    private StateMachineConfig lookForTransition(Event event, StateMachineContext
context, StateMachinePath path, StateMachineConfig config) throws Exception {

                BitSet calculatedInputActions = new BitSet(0);

        int s = decodeState(config.getActiveState());
        int e = decodeEvent(event.getName());

        while (true) {
            switch (s) {

case Stopped:


                                                switch (e) {
                                                    case e2:

                            // Stopped->Rotation e2[true]/

            fireTransitionCandidate(context, path, "Stopped", event,
"Stopped#Rotation#e2#true");
```

```
                    fireTransitionFound(context, path, "Stopped", event,
"Stopped#Rotation#e2#true");


    fireComeToState(context, path, "Rotation");

    // Rotation [o1.z3]
                fireBeforeOutputActionExecution(context, path,
"Stopped#Rotation#e2#true", "o1.z3");

                o1.z3(context);

    fireAfterOutputActionExecution(context, path, "Stopped#Rotation#e2#true",
"o1.z3");
                    return new StateMachineConfig("Rotation");


                                            default:


                                        // transition not found
                                return config;
                                        }


case Rotation:


                                            switch (e) {
                                                case e6:

                        // Rotation->StoppingFirst e6[true]/

            fireTransitionCandidate(context, path, "Rotation", event,
"Rotation#StoppingFirst#e6#true");




                    fireTransitionFound(context, path, "Rotation", event,
"Rotation#StoppingFirst#e6#true");


    fireComeToState(context, path, "StoppingFirst");

    // StoppingFirst []
                    return new StateMachineConfig("StoppingFirst");


                                            default:


                                        // transition not found
                                return config;
                                        }


case StoppingFirst:


                                            switch (e) {
                                                case e6:
```

```
                        // StoppingFirst->StoppingSecond e6[true]/

            fireTransitionCandidate(context, path, "StoppingFirst", event,
"StoppingFirst#StoppingSecond#e6#true");




                    fireTransitionFound(context, path, "StoppingFirst", event,
"StoppingFirst#StoppingSecond#e6#true");


    fireComeToState(context, path, "StoppingSecond");

    // StoppingSecond []
                    return new StateMachineConfig("StoppingSecond");


                                                default:


                                        // transition not found
                                return config;
                                            }

case StoppingSecond:

                                                switch (e) {
                                                    case e6:

                        // StoppingSecond->s2 e6[true]/

            fireTransitionCandidate(context, path, "StoppingSecond", event,
"StoppingSecond#s2#e6#true");

                    fireTransitionFound(context, path, "StoppingSecond", event,
"StoppingSecond#s2#e6#true");

    fireComeToState(context, path, "s2");

    // s2 []
                    return new StateMachineConfig("s2");

                                                default:

                                        // transition not found
                                return config;
                                            }
                                                                    default:
                    throw new EventProcessorException("Incorrect stable state ["
+ config.getActiveState() + "] in state machine [A2]");
            }
        }
    }
}

private class A1EventProcessor {

        // states
        private static final int Top = 1;
        private static final int Error = 2;
        private static final int Waiting = 3;
        private static final int TagReceiving = 4;
        private static final int Play = 5;
        private static final int givingWin = 6;
```

```
    private static final int s1 = 7;

private int decodeState(String state) {

    if ("Top".equals(state)) {
        return Top;
                    } else

    if ("Error".equals(state)) {
        return Error;
                    } else

    if ("Waiting".equals(state)) {
        return Waiting;
                    } else

    if ("TagReceiving".equals(state)) {
        return TagReceiving;
                    } else

    if ("Play".equals(state)) {
        return Play;
                    } else

    if ("givingWin".equals(state)) {
        return givingWin;
                    } else

    if ("s1".equals(state)) {
        return s1;
        }

    return -1;
}

    // events
    private static final int e4 = 1;
    private static final int e2 = 2;
    private static final int e9 = 3;
    private static final int e7 = 4;
    private static final int e5 = 5;
    private static final int e3 = 6;
    private static final int e8 = 7;

private int decodeEvent(String event) {

    if ("e4".equals(event)) {
        return e4;
                    } else

    if ("e2".equals(event)) {
        return e2;
                    } else

    if ("e9".equals(event)) {
        return e9;
                    } else

    if ("e7".equals(event)) {
        return e7;
                    } else

    if ("e5".equals(event)) {
        return e5;
                    } else

    if ("e3".equals(event)) {
```

```
            return e3;
                        } else

        if ("e8".equals(event)) {
            return e8;
                }

        return -1;
    }

                                        private
ru.ifmo.slotmachine.SMControlledObj o1;

    private void init(ControlledObjectsMap controlledObjectsMap) {
                                                        o1 =
(ru.ifmo.slotmachine.SMControlledObj)controlledObjectsMap.getControlledObject("o1
");
                                            }

    private StateMachineConfig process(Event event, StateMachineContext context,
StateMachinePath path, StateMachineConfig config) throws Exception {
        config = lookForTransition(event, context, path, config);

        config = transiteToStableState(context, path, config);

        // execute included state machines
        executeSubmachines(event, context, path, config);

        return config;
    }

    private void executeSubmachines(Event event, StateMachineContext context,
StateMachinePath path, StateMachineConfig config) throws Exception {
        int state = decodeState(config.getActiveState());

        while (true) {
        switch (state) {
                                                                    case
Error:

                                            return;

case Waiting:

                                            return;

case TagReceiving:

// TagReceiving includes A3

    fireBeforeSubmachineExecution(context, event, path, "TagReceiving", "A3");

    Model1EventProcessor.this.process(event, context, new StateMachinePath(path,
        "TagReceiving", "A3"));

    fireAfterSubmachineExecution(context, event, path, "TagReceiving", "A3");

                                            return;

case Play:

// Play includes A2

    fireBeforeSubmachineExecution(context, event, path, "Play", "A2");

    Model1EventProcessor.this.process(event, context, new StateMachinePath(path,
        "Play", "A2"));
```

```
    fireAfterSubmachineExecution(context, event, path, "Play", "A2");

                                                return;

case givingWin:

                                                return;

case s1:

                                                return;
                                                                default:
                        throw new EventProcessorException("State with name [" +
config.getActiveState() + "] is unknown for state machine [A1]");
        }
        }
    }

    private StateMachineConfig transiteToStableState(StateMachineContext context,
StateMachinePath path, StateMachineConfig config) throws Exception {

        int s = decodeState(config.getActiveState());
        Event event;

        switch (s) {
                                        case Top:


                        fireComeToState(context, path, "s1");

                        // s1->Waiting [true]/
                        event = Event.NO_EVENT;
                            fireTransitionFound(context, path, "s1", event,
"s1#Waiting##true");


    fireComeToState(context, path, "Waiting");

    // Waiting []

                                                return new
StateMachineConfig("Waiting");

}

        return config;
    }

    private StateMachineConfig lookForTransition(Event event, StateMachineContext
context, StateMachinePath path, StateMachineConfig config) throws Exception {



            int

            o1_x1 = 0
                                        ,

            o1_x4 = 0
                                        ,

            o1_x2 = 0
                ;

                BitSet calculatedInputActions = new BitSet(3);
```

```
        int s = decodeState(config.getActiveState());
        int e = decodeEvent(event.getName());

        while (true) {
            switch (s) {


case Error:

        switch (e) {
                case e4:

                        // Error->Waiting e4[true]/o1.z5

            fireTransitionCandidate(context, path, "Error", event,
"Error#Waiting#e4#true");
                    fireTransitionFound(context, path, "Error", event,
"Error#Waiting#e4#true");

                fireBeforeOutputActionExecution(context, path,
"Error#Waiting#e4#true", "o1.z5");

                o1.z5(context);

    fireAfterOutputActionExecution(context, path, "Error#Waiting#e4#true",
"o1.z5");

    fireComeToState(context, path, "Waiting");

    // Waiting []
                    return new StateMachineConfig("Waiting");


                                                        case e8:

                        // Error->Waiting e8[true]/o1.z9

            fireTransitionCandidate(context, path, "Error", event,
"Error#Waiting#e8#true");
                    fireTransitionFound(context, path, "Error", event,
"Error#Waiting#e8#true");

                fireBeforeOutputActionExecution(context, path,
"Error#Waiting#e8#true", "o1.z9");

                o1.z9(context);

    fireAfterOutputActionExecution(context, path, "Error#Waiting#e8#true",
"o1.z9");

    fireComeToState(context, path, "Waiting");

    // Waiting []
                    return new StateMachineConfig("Waiting");

                                                    default:

                                        // transition not found
                                return config;
                                        }

case Waiting:
                                                switch (e) {
                                                    case e9:

                        // Waiting->Error e9[true]/o1.z4
```

```
            fireTransitionCandidate(context, path, "Waiting", event,
"Waiting#Error#e9#true");




                    fireTransitionFound(context, path, "Waiting", event,
"Waiting#Error#e9#true");

                fireBeforeOutputActionExecution(context, path,
"Waiting#Error#e9#true", "o1.z4");

                o1.z4(context);

    fireAfterOutputActionExecution(context, path, "Waiting#Error#e9#true",
"o1.z4");

    fireComeToState(context, path, "Error");

    // Error []
                    return new StateMachineConfig("Error");

                                                        case e3:

                        // Waiting->TagReceiving e3[true]/

            fireTransitionCandidate(context, path, "Waiting", event,
"Waiting#TagReceiving#e3#true");

                    fireTransitionFound(context, path, "Waiting", event,
"Waiting#TagReceiving#e3#true");


    fireComeToState(context, path, "TagReceiving");

    // TagReceiving []
                    return new StateMachineConfig("TagReceiving");

                                                    default:

                                    // transition not found
                                return config;
                                            }

case TagReceiving:

        switch (e) {
                case e4:
                        // TagReceiving->Waiting e4[true]/o1.z5

            fireTransitionCandidate(context, path, "TagReceiving", event,
"TagReceiving#Waiting#e4#true");

                    fireTransitionFound(context, path, "TagReceiving", event,
"TagReceiving#Waiting#e4#true");

                fireBeforeOutputActionExecution(context, path,
"TagReceiving#Waiting#e4#true", "o1.z5");

                o1.z5(context);

    fireAfterOutputActionExecution(context, path, "TagReceiving#Waiting#e4#true",
"o1.z5");

    fireComeToState(context, path, "Waiting");
```

```
    // Waiting []
                    return new StateMachineConfig("Waiting");

                                                        case e2:

                        // TagReceiving->Play e2[true]/

            fireTransitionCandidate(context, path, "TagReceiving", event,
"TagReceiving#Play#e2#true");

                    fireTransitionFound(context, path, "TagReceiving", event,
"TagReceiving#Play#e2#true");


    fireComeToState(context, path, "Play");

    // Play []
                    return new StateMachineConfig("Play");

                                                        case e9:
                        // TagReceiving->Error e9[true]/o1.z4

            fireTransitionCandidate(context, path, "TagReceiving", event,
"TagReceiving#Error#e9#true");
                    fireTransitionFound(context, path, "TagReceiving", event,
"TagReceiving#Error#e9#true");

                fireBeforeOutputActionExecution(context, path,
"TagReceiving#Error#e9#true", "o1.z4");

                o1.z4(context);

    fireAfterOutputActionExecution(context, path, "TagReceiving#Error#e9#true",
"o1.z4");

    fireComeToState(context, path, "Error");

    // Error []
                    return new StateMachineConfig("Error");

                                                    default:

                                        // transition not found
                                    return config;
                                                }
case Play:
        switch (e) {
                case e7:

                        // Play->Error e7[o1.x4 > 0 && o1.x1 <= 0]/o1.z6

            fireTransitionCandidate(context, path, "Play", event,
"Play#Error#e7#o1.x4 > 0 && o1.x1 <= 0");
                                                                    if
(!isInputActionCalculated(calculatedInputActions, _o1_x1)) {

        fireBeforeInputActionExecution(context, path, "Play#Error#e7#o1.x4 > 0 &&
o1.x1 <= 0", "o1.x1");

                    o1_x1 = o1.x1(context);

            fireAfterInputActionExecution(context, path, "Play#Error#e7#o1.x4 > 0
&& o1.x1 <= 0", "o1.x1", new Integer(o1_x1));
            }
                                                                  if
(!isInputActionCalculated(calculatedInputActions, _o1_x4)) {
```

```
        fireBeforeInputActionExecution(context, path, "Play#Error#e7#o1.x4 > 0 &&
o1.x1 <= 0", "o1.x4");

                    o1_x4 = o1.x4(context);

            fireAfterInputActionExecution(context, path, "Play#Error#e7#o1.x4 > 0
&& o1.x1 <= 0", "o1.x4", new Integer(o1_x4));
            }

if (o1_x4 > 0 && o1_x1 <= 0) {

                    fireTransitionFound(context, path, "Play", event,
"Play#Error#e7#o1.x4 > 0 && o1.x1 <= 0");

                fireBeforeOutputActionExecution(context, path,
"Play#Error#e7#o1.x4 > 0 && o1.x1 <= 0", "o1.z6");

                o1.z6(context);

    fireAfterOutputActionExecution(context, path, "Play#Error#e7#o1.x4 > 0 &&
o1.x1 <= 0", "o1.z6");

    fireComeToState(context, path, "Error");

    // Error []
                    return new StateMachineConfig("Error");

                                    }
                                                // Play->Waiting
e7[o1.x4<=0]/o1.z8

            fireTransitionCandidate(context, path, "Play", event,
"Play#Waiting#e7#o1.x4<=0");



                                                if (o1_x4 <= 0) {

                    fireTransitionFound(context, path, "Play", event,
"Play#Waiting#e7#o1.x4<=0");

                fireBeforeOutputActionExecution(context, path,
"Play#Waiting#e7#o1.x4<=0", "o1.z8");

                o1.z8(context);

    fireAfterOutputActionExecution(context, path, "Play#Waiting#e7#o1.x4<=0",
"o1.z8");

    fireComeToState(context, path, "Waiting");

    // Waiting []
                    return new StateMachineConfig("Waiting");

                                    }
                                                // Play->givingWin e7[o1.x4 >0 &&
o1.x1>0]/o1.z7

            fireTransitionCandidate(context, path, "Play", event,
"Play#givingWin#e7#o1.x4 >0 && o1.x1>0");



                                                if (o1_x4 > 0 && o1_x1 > 0) {
```

```
                    fireTransitionFound(context, path, "Play", event,
"Play#givingWin#e7#o1.x4 >0 && o1.x1>0");

                fireBeforeOutputActionExecution(context, path,
"Play#givingWin#e7#o1.x4 >0 && o1.x1>0", "o1.z7");

                o1.z7(context);

    fireAfterOutputActionExecution(context, path, "Play#givingWin#e7#o1.x4 >0 &&
o1.x1>0", "o1.z7");

    fireComeToState(context, path, "givingWin");

    // givingWin []
                    return new StateMachineConfig("givingWin");

                                    }

                                        // transition not found
                                return config;
                                                                    default:


                                        // transition not found
                                return config;
                                            }


case givingWin:


                                                switch (e) {
                                                    case e5:

                        // givingWin->Error e5[o1.x2>0 && o1.x1 <= 0]/o1.z6

            fireTransitionCandidate(context, path, "givingWin", event,
"givingWin#Error#e5#o1.x2>0 && o1.x1 <= 0");

                                                                        if
(!isInputActionCalculated(calculatedInputActions, _o1_x1)) {

        fireBeforeInputActionExecution(context, path, "givingWin#Error#e5#o1.x2>0
&& o1.x1 <= 0", "o1.x1");

                    o1_x1 = o1.x1(context);

            fireAfterInputActionExecution(context, path,
"givingWin#Error#e5#o1.x2>0 && o1.x1 <= 0", "o1.x1", new Integer(o1_x1));
            }
                                                                    if
(!isInputActionCalculated(calculatedInputActions, _o1_x2)) {

        fireBeforeInputActionExecution(context, path, "givingWin#Error#e5#o1.x2>0
&& o1.x1 <= 0", "o1.x2");

                    o1_x2 = o1.x2(context);

            fireAfterInputActionExecution(context, path,
"givingWin#Error#e5#o1.x2>0 && o1.x1 <= 0", "o1.x2", new Integer(o1_x2));
            }

if (o1_x2 > 0 && o1_x1 <= 0) {

                    fireTransitionFound(context, path, "givingWin", event,
"givingWin#Error#e5#o1.x2>0 && o1.x1 <= 0");
```

```
                fireBeforeOutputActionExecution(context, path,
"givingWin#Error#e5#o1.x2>0 && o1.x1 <= 0", "o1.z6");

                o1.z6(context);

    fireAfterOutputActionExecution(context, path, "givingWin#Error#e5#o1.x2>0 &&
o1.x1 <= 0", "o1.z6");

    fireComeToState(context, path, "Error");

    // Error []
                    return new StateMachineConfig("Error");
                                    }
                                                    // givingWin->Waiting e5[o1.x2 <=
0]/o1.z8

            fireTransitionCandidate(context, path, "givingWin", event,
"givingWin#Waiting#e5#o1.x2 <= 0");

if (o1_x2 <= 0) {

                    fireTransitionFound(context, path, "givingWin", event,
"givingWin#Waiting#e5#o1.x2 <= 0");

                fireBeforeOutputActionExecution(context, path,
"givingWin#Waiting#e5#o1.x2 <= 0", "o1.z8");

                o1.z8(context);

    fireAfterOutputActionExecution(context, path, "givingWin#Waiting#e5#o1.x2 <=
0", "o1.z8");

    fireComeToState(context, path, "Waiting");

    // Waiting []
                    return new StateMachineConfig("Waiting");

                                    }
                                                    // givingWin->givingWin e5[o1.x2
> 0 && o1.x1>0]/o1.z7

            fireTransitionCandidate(context, path, "givingWin", event,
"givingWin#givingWin#e5#o1.x2 > 0 && o1.x1>0");

if (o1_x2 > 0 && o1_x1 > 0) {

                    fireTransitionFound(context, path, "givingWin", event,
"givingWin#givingWin#e5#o1.x2 > 0 && o1.x1>0");

                fireBeforeOutputActionExecution(context, path,
"givingWin#givingWin#e5#o1.x2 > 0 && o1.x1>0", "o1.z7");

                o1.z7(context);

    fireAfterOutputActionExecution(context, path, "givingWin#givingWin#e5#o1.x2 >
0 && o1.x1>0", "o1.z7");

    fireComeToState(context, path, "givingWin");

    // givingWin []
                    return new StateMachineConfig("givingWin");
        }
                                    // transition not found
                            return config;
                                                            default:
                                    // transition not found
```

```
                                return config;
                                                }

default:
                    throw new EventProcessorException("Incorrect stable state ["
+ config.getActiveState() + "] in state machine [A1]");
            }
        }
    }

                //o1.x1
                private static final int _o1_x1 = 0;
            //o1.x4
                private static final int _o1_x4 = 1;
            //o1.x2
                private static final int _o1_x2 = 2;

}

private class A3EventProcessor {

        // states
        private static final int Top = 1;
        private static final int s2 = 2;
        private static final int Empty = 3;
        private static final int TrueCoinReceived = 4;
        private static final int s1 = 5;

    private int decodeState(String state) {

        if ("Top".equals(state)) {
            return Top;
                        } else

        if ("s2".equals(state)) {
            return s2;
                        } else

        if ("Empty".equals(state)) {
            return Empty;
                        } else

        if ("TrueCoinReceived".equals(state)) {
            return TrueCoinReceived;
                        } else

        if ("s1".equals(state)) {
            return s1;
            }

        return -1;
    }

        // events
        private static final int e3 = 1;

    private int decodeEvent(String event) {

        if ("e3".equals(event)) {
            return e3;
                }

        return -1;
    }

private ru.ifmo.slotmachine.SMControlledObj o1;
```

```
    private void init(ControlledObjectsMap controlledObjectsMap) {
                                        o1 =
(ru.ifmo.slotmachine.SMControlledObj)controlledObjectsMap.getControlledObject("o1
");
                            }

    private StateMachineConfig process(Event event, StateMachineContext context,
StateMachinePath path, StateMachineConfig config) throws Exception {
        config = lookForTransition(event, context, path, config);

        config = transiteToStableState(context, path, config);

        // execute included state machines
        executeSubmachines(event, context, path, config);

        return config;
    }

    private void executeSubmachines(Event event, StateMachineContext context,
StateMachinePath path, StateMachineConfig config) throws Exception {
        int state = decodeState(config.getActiveState());

        while (true) {
        switch (state) {
                                                                        case
s2:
                                            return;
case Empty:
                                            return;

case TrueCoinReceived:
                                            return;
case s1:
                                            return;
default:

        throw new EventProcessorException("State with name [" +
config.getActiveState() + "] is unknown for state machine [A3]");
        }
        }
    }

    private StateMachineConfig transiteToStableState(StateMachineContext context,
StateMachinePath path, StateMachineConfig config) throws Exception {

        int s = decodeState(config.getActiveState());
        Event event;

        switch (s) {
                                    case Top:
                    fireComeToState(context, path, "s1");

                    // s1->Empty [true]/
                    event = Event.NO_EVENT;
                        fireTransitionFound(context, path, "s1", event,
"s1#Empty##true");

    fireComeToState(context, path, "Empty");

    // Empty []
                                            return new
StateMachineConfig("Empty");
}
        return config;
    }
```

```
    private StateMachineConfig lookForTransition(Event event, StateMachineContext
context, StateMachinePath path, StateMachineConfig config) throws Exception {

            int

            o1_x3 = 0
                ;

                BitSet calculatedInputActions = new BitSet(1);

        int s = decodeState(config.getActiveState());
        int e = decodeEvent(event.getName());

        while (true) {
            switch (s) {

case Empty:
        switch (e) {
                case e3:

                        // Empty->s2 e3[o1.x3 <= 0]/
            fireTransitionCandidate(context, path, "Empty", event,
"Empty#s2#e3#o1.x3 <= 0");

if (!isInputActionCalculated(calculatedInputActions, _o1_x3)) {

        fireBeforeInputActionExecution(context, path, "Empty#s2#e3#o1.x3 <= 0",
"o1.x3");

                    o1_x3 = o1.x3(context);

            fireAfterInputActionExecution(context, path, "Empty#s2#e3#o1.x3 <=
0", "o1.x3", new Integer(o1_x3));
            }

if (o1_x3 <= 0) {

                    fireTransitionFound(context, path, "Empty", event,
"Empty#s2#e3#o1.x3 <= 0");


    fireComeToState(context, path, "s2");

    // s2 []
                    return new StateMachineConfig("s2");
                                }
                                                    // Empty->TrueCoinReceived
e3[o1.x3>0]/o1.z2

            fireTransitionCandidate(context, path, "Empty", event,
"Empty#TrueCoinReceived#e3#o1.x3>0");

if (o1_x3 > 0) {
                    fireTransitionFound(context, path, "Empty", event,
"Empty#TrueCoinReceived#e3#o1.x3>0");

                fireBeforeOutputActionExecution(context, path,
"Empty#TrueCoinReceived#e3#o1.x3>0", "o1.z2");

                o1.z2(context);

    fireAfterOutputActionExecution(context, path,
"Empty#TrueCoinReceived#e3#o1.x3>0", "o1.z2");

    fireComeToState(context, path, "TrueCoinReceived");

    // TrueCoinReceived []
```

```
                    return new StateMachineConfig("TrueCoinReceived");
                                }

                                        // transition not found
                                return config;
                                                                        default:
                                        // transition not found
                                return config;
                                            }


case TrueCoinReceived:
        switch (e) {
                case e3:

                        // TrueCoinReceived->s2 e3[o1.x3 <= 0]/

            fireTransitionCandidate(context, path, "TrueCoinReceived", event,
"TrueCoinReceived#s2#e3#o1.x3 <= 0");

                                                                    if
(!isInputActionCalculated(calculatedInputActions, _o1_x3)) {

        fireBeforeInputActionExecution(context, path,
"TrueCoinReceived#s2#e3#o1.x3 <= 0", "o1.x3");

                    o1_x3 = o1.x3(context);

            fireAfterInputActionExecution(context, path,
"TrueCoinReceived#s2#e3#o1.x3 <= 0", "o1.x3", new Integer(o1_x3));
            }

if (o1_x3 <= 0) {

                    fireTransitionFound(context, path, "TrueCoinReceived", event,
"TrueCoinReceived#s2#e3#o1.x3 <= 0");
    fireComeToState(context, path, "s2");

    // s2 []
                    return new StateMachineConfig("s2");
                                }
                                            // TrueCoinReceived-
>TrueCoinReceived e3[o1.x3>0]/o1.z2

            fireTransitionCandidate(context, path, "TrueCoinReceived", event,
"TrueCoinReceived#TrueCoinReceived#e3#o1.x3>0");

if (o1_x3 > 0) {

                    fireTransitionFound(context, path, "TrueCoinReceived", event,
"TrueCoinReceived#TrueCoinReceived#e3#o1.x3>0");

                fireBeforeOutputActionExecution(context, path,
"TrueCoinReceived#TrueCoinReceived#e3#o1.x3>0", "o1.z2");

                o1.z2(context);

    fireAfterOutputActionExecution(context, path,
"TrueCoinReceived#TrueCoinReceived#e3#o1.x3>0", "o1.z2");

    fireComeToState(context, path, "TrueCoinReceived");

    // TrueCoinReceived []
                    return new StateMachineConfig("TrueCoinReceived");
                                }
                                        // transition not found
                                return config;
```

```
                                                                    default:
                                                // transition not found
                                        return config;
                                                            }
default:
                        throw new EventProcessorException("Incorrect stable state ["
+ config.getActiveState() + "] in state machine [A3]");
                }
            }
        }

                //o1.x3
                private static final int _o1_x3 = 0;
}

    private static boolean isInputActionCalculated(BitSet calculatedInputActions,
int k) {
        boolean b = calculatedInputActions.get(k);

        if (!b) {
            calculatedInputActions.set(k);
        }

        return b;
    }

}
```

Приложение 4. Пример протокола работы программы

```
03:27:34,845 INFO  [Run] Start event [e1] processing. In state [/A1:Top]
03:27:34,845 INFO  [Run] Transition to go found [s1#Waiting##true]
03:27:34,845 INFO  [Run] Finish event [e1] processing. In state [/A1:Waiting]
03:27:37,048 INFO  [Run] Start event [e3] processing. In state [/A1:Waiting]
03:27:37,048 DEBUG [Run] Try transition [Waiting#TagReceiving#e3#true]
03:27:37,048 INFO  [Run] Transition to go found [Waiting#TagReceiving#e3#true]
03:27:37,048 INFO  [Run] Start event [e3] processing. In state [/A1:TagReceiving/A3:Top]
03:27:37,048 INFO  [Run] Transition to go found [s1#Empty##true]
03:27:37,048 DEBUG [Run] Try transition [Empty#s2#e3#o1.x3 <= 0]
03:27:37,048 INFO  [Run] Start input action [o1.x3] calculation
03:27:37,048 INFO  [Run] Finish input action [o1.x3] calculation. Its value is [30]
03:27:37,048 DEBUG [Run] Try transition [Empty#TrueCoinReceived#e3#o1.x3>0]
03:27:37,048 INFO  [Run] Transition to go found [Empty#TrueCoinReceived#e3#o1.x3>0]
03:27:37,048 INFO  [Run] Start output action [o1.z2] execution
03:27:37,048 INFO  [Run] Finish output action [o1.z2] execution
03:27:37,048 INFO  [Run] Finish event [e3] processing. In state
[/A1:TagReceiving/A3:TrueCoinReceived]
03:27:37,048 INFO  [Run] Finish event [e3] processing. In state [/A1:TagReceiving]
03:27:37,258 INFO  [Run] Start event [e3] processing. In state [/A1:TagReceiving]
03:27:37,258 INFO  [Run] Start event [e3] processing. In state
[/A1:TagReceiving/A3:TrueCoinReceived]
03:27:37,258 DEBUG [Run] Try transition [TrueCoinReceived#s2#e3#o1.x3 <= 0]
03:27:37,258 INFO  [Run] Start input action [o1.x3] calculation
03:27:37,258 INFO  [Run] Finish input action [o1.x3] calculation. Its value is [29]
03:27:37,258 DEBUG [Run] Try transition [TrueCoinReceived#TrueCoinReceived#e3#o1.x3>0]
03:27:37,258 INFO  [Run] Transition to go found
[TrueCoinReceived#TrueCoinReceived#e3#o1.x3>0]
03:27:37,258 INFO  [Run] Start output action [o1.z2] execution
03:27:37,258 INFO  [Run] Finish output action [o1.z2] execution
```

```
03:27:37,258 INFO  [Run] Finish event [e3] processing. In state
[/A1:TagReceiving/A3:TrueCoinReceived]
03:27:37,258 INFO  [Run] Finish event [e3] processing. In state [/A1:TagReceiving]
03:27:37,459 INFO  [Run] Start event [e3] processing. In state [/A1:TagReceiving]
03:27:37,459 INFO  [Run] Start event [e3] processing. In state
[/A1:TagReceiving/A3:TrueCoinReceived]
03:27:37,459 DEBUG [Run] Try transition [TrueCoinReceived#s2#e3#o1.x3 <= 0]
03:27:37,459 INFO  [Run] Start input action [o1.x3] calculation
03:27:37,459 INFO  [Run] Finish input action [o1.x3] calculation. Its value is [28]
03:27:37,459 DEBUG [Run] Try transition [TrueCoinReceived#TrueCoinReceived#e3#o1.x3>0]
03:27:37,459 INFO  [Run] Transition to go found
[TrueCoinReceived#TrueCoinReceived#e3#o1.x3>0]
03:27:37,459 INFO  [Run] Start output action [o1.z2] execution
03:27:37,459 INFO  [Run] Finish output action [o1.z2] execution
03:27:37,459 INFO  [Run] Finish event [e3] processing. In state
[/A1:TagReceiving/A3:TrueCoinReceived]
03:27:37,459 INFO  [Run] Finish event [e3] processing. In state [/A1:TagReceiving]
03:27:37,659 INFO  [Run] Start event [e3] processing. In state [/A1:TagReceiving]
03:27:37,669 INFO  [Run] Start event [e3] processing. In state
[/A1:TagReceiving/A3:TrueCoinReceived]
03:27:37,669 DEBUG [Run] Try transition [TrueCoinReceived#s2#e3#o1.x3 <= 0]
03:27:37,669 INFO  [Run] Start input action [o1.x3] calculation
03:27:37,669 INFO  [Run] Finish input action [o1.x3] calculation. Its value is [27]
03:27:37,669 DEBUG [Run] Try transition [TrueCoinReceived#TrueCoinReceived#e3#o1.x3>0]
03:27:37,669 INFO  [Run] Transition to go found
[TrueCoinReceived#TrueCoinReceived#e3#o1.x3>0]
03:27:37,669 INFO  [Run] Start output action [o1.z2] execution
03:27:37,669 INFO  [Run] Finish output action [o1.z2] execution
03:27:37,669 INFO  [Run] Finish event [e3] processing. In state
[/A1:TagReceiving/A3:TrueCoinReceived]
03:27:37,669 INFO  [Run] Finish event [e3] processing. In state [/A1:TagReceiving]
03:27:37,879 INFO  [Run] Start event [e3] processing. In state [/A1:TagReceiving]
03:27:37,879 INFO  [Run] Start event [e3] processing. In state
[/A1:TagReceiving/A3:TrueCoinReceived]
03:27:37,879 DEBUG [Run] Try transition [TrueCoinReceived#s2#e3#o1.x3 <= 0]
03:27:37,879 INFO  [Run] Start input action [o1.x3] calculation
03:27:37,879 INFO  [Run] Finish input action [o1.x3] calculation. Its value is [26]
03:27:37,879 DEBUG [Run] Try transition [TrueCoinReceived#TrueCoinReceived#e3#o1.x3>0]
03:27:37,879 INFO  [Run] Transition to go found
[TrueCoinReceived#TrueCoinReceived#e3#o1.x3>0]
03:27:37,879 INFO  [Run] Start output action [o1.z2] execution
03:27:37,879 INFO  [Run] Finish output action [o1.z2] execution
03:27:37,879 INFO  [Run] Finish event [e3] processing. In state
[/A1:TagReceiving/A3:TrueCoinReceived]
03:27:37,879 INFO  [Run] Finish event [e3] processing. In state [/A1:TagReceiving]
03:27:38,090 INFO  [Run] Start event [e3] processing. In state [/A1:TagReceiving]
03:27:38,090 INFO  [Run] Start event [e3] processing. In state
[/A1:TagReceiving/A3:TrueCoinReceived]
03:27:38,090 DEBUG [Run] Try transition [TrueCoinReceived#s2#e3#o1.x3 <= 0]
03:27:38,090 INFO  [Run] Start input action [o1.x3] calculation
03:27:38,090 INFO  [Run] Finish input action [o1.x3] calculation. Its value is [25]
03:27:38,090 DEBUG [Run] Try transition [TrueCoinReceived#TrueCoinReceived#e3#o1.x3>0]
03:27:38,090 INFO  [Run] Transition to go found
[TrueCoinReceived#TrueCoinReceived#e3#o1.x3>0]
03:27:38,090 INFO  [Run] Start output action [o1.z2] execution
```

```
03:27:38,090 INFO  [Run] Finish output action [o1.z2] execution
03:27:38,090 INFO  [Run] Finish event [e3] processing. In state
[/A1:TagReceiving/A3:TrueCoinReceived]
03:27:38,090 INFO  [Run] Finish event [e3] processing. In state [/A1:TagReceiving]
03:27:38,951 INFO  [Run] Start event [e2] processing. In state [/A1:TagReceiving]
03:27:38,951 DEBUG [Run] Try transition [TagReceiving#Play#e2#true]
03:27:38,951 INFO  [Run] Transition to go found [TagReceiving#Play#e2#true]
03:27:38,951 INFO  [Run] Start event [e2] processing. In state [/A1:Play/A2:Top]
03:27:38,951 INFO  [Run] Transition to go found [s1#Stopped##true]
03:27:38,951 DEBUG [Run] Try transition [Stopped#Rotation#e2#true]
03:27:38,951 INFO  [Run] Transition to go found [Stopped#Rotation#e2#true]
03:27:38,951 INFO  [Run] Start on-enter action [o1.z3] execution
03:27:45,560 INFO  [Run] Finish on-enter action [o1.z3] execution
03:27:45,560 INFO  [Run] Finish event [e2] processing. In state [/A1:Play/A2:Rotation]
03:27:45,560 INFO  [Run] Finish event [e2] processing. In state [/A1:Play]
03:27:45,560 INFO  [Run] Start event [e6] processing. In state [/A1:Play]
03:27:45,560 INFO  [Run] Start event [e6] processing. In state [/A1:Play/A2:Rotation]
03:27:45,560 DEBUG [Run] Try transition [Rotation#StoppingFirst#e6#true]
03:27:45,560 INFO  [Run] Transition to go found [Rotation#StoppingFirst#e6#true]
03:27:45,560 INFO  [Run] Finish event [e6] processing. In state [/A1:Play/A2:StoppingFirst]
03:27:45,560 INFO  [Run] Finish event [e6] processing. In state [/A1:Play]
03:27:45,560 INFO  [Run] Start event [e6] processing. In state [/A1:Play]
03:27:45,560 INFO  [Run] Start event [e6] processing. In state [/A1:Play/A2:StoppingFirst]
03:27:45,560 DEBUG [Run] Try transition [StoppingFirst#StoppingSecond#e6#true]
03:27:45,560 INFO  [Run] Transition to go found [StoppingFirst#StoppingSecond#e6#true]
03:27:45,560 INFO  [Run] Finish event [e6] processing. In state [/A1:Play/A2:StoppingSecond]
03:27:45,560 INFO  [Run] Finish event [e6] processing. In state [/A1:Play]
03:27:45,560 INFO  [Run] Start event [e6] processing. In state [/A1:Play]
03:27:45,560 INFO  [Run] Start event [e6] processing. In state [/A1:Play/A2:StoppingSecond]
03:27:45,560 DEBUG [Run] Try transition [StoppingSecond#s2#e6#true]
03:27:45,560 INFO  [Run] Transition to go found [StoppingSecond#s2#e6#true]
03:27:45,560 INFO  [Run] State machine came to final state [/A1:Play/A2:s2]
03:27:45,560 INFO  [Run] Finish event [e6] processing. In state [/A1:Play/A2:s2]
03:27:45,560 INFO  [Run] Finish event [e6] processing. In state [/A1:Play]
03:27:45,560 INFO  [Run] Start event [e7] processing. In state [/A1:Play]
03:27:45,560 DEBUG [Run] Try transition [Play#Error#e7#o1.x4 > 0 && o1.x1 <= 0]
03:27:45,560 INFO  [Run] Start input action [o1.x4] calculation
03:27:45,560 INFO  [Run] Finish input action [o1.x4] calculation. Its value is [0]
03:27:45,560 INFO  [Run] Start input action [o1.x1] calculation
03:27:45,560 INFO  [Run] Finish input action [o1.x1] calculation. Its value is [56]
03:27:45,560 DEBUG [Run] Try transition [Play#Waiting#e7#o1.x4<=0]
03:27:45,560 INFO  [Run] Transition to go found [Play#Waiting#e7#o1.x4<=0]
03:27:45,560 INFO  [Run] Start output action [o1.z8] execution
03:27:45,560 INFO  [Run] Finish output action [o1.z8] execution
03:27:45,560 INFO  [Run] Finish event [e7] processing. In state [/A1:Waiting]
03:27:46,882 INFO  [Run] Start event [e3] processing. In state [/A1:Waiting]
03:27:46,882 DEBUG [Run] Try transition [Waiting#TagReceiving#e3#true]
03:27:46,882 INFO  [Run] Transition to go found [Waiting#TagReceiving#e3#true]
03:27:46,882 INFO  [Run] Start event [e3] processing. In state
[/A1:TagReceiving/A3:TrueCoinReceived]
03:27:46,882 DEBUG [Run] Try transition [TrueCoinReceived#s2#e3#o1.x3 <= 0]
03:27:46,882 INFO  [Run] Start input action [o1.x3] calculation
03:27:46,882 INFO  [Run] Finish input action [o1.x3] calculation. Its value is [24]
03:27:46,882 DEBUG [Run] Try transition [TrueCoinReceived#TrueCoinReceived#e3#o1.x3>0]
```

```
03:27:46,882 INFO  [Run] Transition to go found
[TrueCoinReceived#TrueCoinReceived#e3#o1.x3>0]
03:27:46,882 INFO  [Run] Start output action [o1.z2] execution
03:27:46,882 INFO  [Run] Finish output action [o1.z2] execution
03:27:46,882 INFO  [Run] Finish event [e3] processing. In state
[/A1:TagReceiving/A3:TrueCoinReceived]
03:27:46,882 INFO  [Run] Finish event [e3] processing. In state [/A1:TagReceiving]
03:27:47,113 INFO  [Run] Start event [e3] processing. In state [/A1:TagReceiving]
03:27:47,113 INFO  [Run] Start event [e3] processing. In state
[/A1:TagReceiving/A3:TrueCoinReceived]
03:27:47,113 DEBUG [Run] Try transition [TrueCoinReceived#s2#e3#o1.x3 <= 0]
03:27:47,113 INFO  [Run] Start input action [o1.x3] calculation
03:27:47,113 INFO  [Run] Finish input action [o1.x3] calculation. Its value is [23]
03:27:47,113 DEBUG [Run] Try transition [TrueCoinReceived#TrueCoinReceived#e3#o1.x3>0]
03:27:47,113 INFO  [Run] Transition to go found
[TrueCoinReceived#TrueCoinReceived#e3#o1.x3>0]
03:27:47,113 INFO  [Run] Start output action [o1.z2] execution
03:27:47,113 INFO  [Run] Finish output action [o1.z2] execution
03:27:47,113 INFO  [Run] Finish event [e3] processing. In state
[/A1:TagReceiving/A3:TrueCoinReceived]
03:27:47,113 INFO  [Run] Finish event [e3] processing. In state [/A1:TagReceiving]
03:27:47,333 INFO  [Run] Start event [e3] processing. In state [/A1:TagReceiving]
03:27:47,333 INFO  [Run] Start event [e3] processing. In state
[/A1:TagReceiving/A3:TrueCoinReceived]
03:27:47,333 DEBUG [Run] Try transition [TrueCoinReceived#s2#e3#o1.x3 <= 0]
03:27:47,333 INFO  [Run] Start input action [o1.x3] calculation
03:27:47,333 INFO  [Run] Finish input action [o1.x3] calculation. Its value is [22]
03:27:47,333 DEBUG [Run] Try transition [TrueCoinReceived#TrueCoinReceived#e3#o1.x3>0]
03:27:47,333 INFO  [Run] Transition to go found
[TrueCoinReceived#TrueCoinReceived#e3#o1.x3>0]
03:27:47,333 INFO  [Run] Start output action [o1.z2] execution
03:27:47,333 INFO  [Run] Finish output action [o1.z2] execution
03:27:47,333 INFO  [Run] Finish event [e3] processing. In state
[/A1:TagReceiving/A3:TrueCoinReceived]
03:27:47,333 INFO  [Run] Finish event [e3] processing. In state [/A1:TagReceiving]
03:27:47,573 INFO  [Run] Start event [e3] processing. In state [/A1:TagReceiving]
03:27:47,573 INFO  [Run] Start event [e3] processing. In state
[/A1:TagReceiving/A3:TrueCoinReceived]
03:27:47,573 DEBUG [Run] Try transition [TrueCoinReceived#s2#e3#o1.x3 <= 0]
03:27:47,573 INFO  [Run] Start input action [o1.x3] calculation
03:27:47,573 INFO  [Run] Finish input action [o1.x3] calculation. Its value is [21]
03:27:47,573 DEBUG [Run] Try transition [TrueCoinReceived#TrueCoinReceived#e3#o1.x3>0]
03:27:47,573 INFO  [Run] Transition to go found
[TrueCoinReceived#TrueCoinReceived#e3#o1.x3>0]
03:27:47,573 INFO  [Run] Start output action [o1.z2] execution
03:27:47,573 INFO  [Run] Finish output action [o1.z2] execution
03:27:47,573 INFO  [Run] Finish event [e3] processing. In state
[/A1:TagReceiving/A3:TrueCoinReceived]
03:27:47,573 INFO  [Run] Finish event [e3] processing. In state [/A1:TagReceiving]
03:27:48,414 INFO  [Run] Start event [e2] processing. In state [/A1:TagReceiving]
03:27:48,414 DEBUG [Run] Try transition [TagReceiving#Play#e2#true]
03:27:48,414 INFO  [Run] Transition to go found [TagReceiving#Play#e2#true]
03:27:48,414 INFO  [Run] Start event [e2] processing. In state [/A1:Play/A2:Top]
03:27:48,414 INFO  [Run] Transition to go found [s1#Stopped##true]
03:27:48,414 DEBUG [Run] Try transition [Stopped#Rotation#e2#true]
```

```
03:27:48,414 INFO  [Run] Transition to go found [Stopped#Rotation#e2#true]
03:27:48,414 INFO  [Run] Start on-enter action [o1.z3] execution
03:27:54,784 INFO  [Run] Finish on-enter action [o1.z3] execution
03:27:54,784 INFO  [Run] Finish event [e2] processing. In state [/A1:Play/A2:Rotation]
03:27:54,784 INFO  [Run] Finish event [e2] processing. In state [/A1:Play]
03:27:54,784 INFO  [Run] Start event [e6] processing. In state [/A1:Play]
03:27:54,784 INFO  [Run] Start event [e6] processing. In state [/A1:Play/A2:Rotation]
03:27:54,784 DEBUG [Run] Try transition [Rotation#StoppingFirst#e6#true]
03:27:54,784 INFO  [Run] Transition to go found [Rotation#StoppingFirst#e6#true]
03:27:54,784 INFO  [Run] Finish event [e6] processing. In state [/A1:Play/A2:StoppingFirst]
03:27:54,784 INFO  [Run] Finish event [e6] processing. In state [/A1:Play]
03:27:54,784 INFO  [Run] Start event [e6] processing. In state [/A1:Play]
03:27:54,784 INFO  [Run] Start event [e6] processing. In state [/A1:Play/A2:StoppingFirst]
03:27:54,784 DEBUG [Run] Try transition [StoppingFirst#StoppingSecond#e6#true]
03:27:54,784 INFO  [Run] Transition to go found [StoppingFirst#StoppingSecond#e6#true]
03:27:54,784 INFO  [Run] Finish event [e6] processing. In state [/A1:Play/A2:StoppingSecond]
03:27:54,784 INFO  [Run] Finish event [e6] processing. In state [/A1:Play]
03:27:54,784 INFO  [Run] Start event [e6] processing. In state [/A1:Play]
03:27:54,784 INFO  [Run] Start event [e6] processing. In state [/A1:Play/A2:StoppingSecond]
03:27:54,784 DEBUG [Run] Try transition [StoppingSecond#s2#e6#true]
03:27:54,784 INFO  [Run] Transition to go found [StoppingSecond#s2#e6#true]
03:27:54,784 INFO  [Run] State machine came to final state [/A1:Play/A2:s2]
03:27:54,784 INFO  [Run] Finish event [e6] processing. In state [/A1:Play/A2:s2]
03:27:54,784 INFO  [Run] Finish event [e6] processing. In state [/A1:Play]
03:27:54,784 INFO  [Run] Start event [e7] processing. In state [/A1:Play]
03:27:54,784 DEBUG [Run] Try transition [Play#Error#e7#o1.x4 > 0 && o1.x1 <= 0]
03:27:54,784 INFO  [Run] Start input action [o1.x4] calculation
03:27:54,784 INFO  [Run] Finish input action [o1.x4] calculation. Its value is [0]
03:27:54,784 INFO  [Run] Start input action [o1.x1] calculation
03:27:54,784 INFO  [Run] Finish input action [o1.x1] calculation. Its value is [60]
03:27:54,784 DEBUG [Run] Try transition [Play#Waiting#e7#o1.x4<=0]
03:27:54,784 INFO  [Run] Transition to go found [Play#Waiting#e7#o1.x4<=0]
03:27:54,784 INFO  [Run] Start output action [o1.z8] execution
03:27:54,784 INFO  [Run] Finish output action [o1.z8] execution
03:27:54,784 INFO  [Run] Finish event [e7] processing. In state [/A1:Waiting]
03:27:56,596 INFO  [Run] Start event [e3] processing. In state [/A1:Waiting]
03:27:56,596 DEBUG [Run] Try transition [Waiting#TagReceiving#e3#true]
03:27:56,596 INFO  [Run] Transition to go found [Waiting#TagReceiving#e3#true]
03:27:56,596 INFO  [Run] Start event [e3] processing. In state
[/A1:TagReceiving/A3:TrueCoinReceived]
03:27:56,596 DEBUG [Run] Try transition [TrueCoinReceived#s2#e3#o1.x3 <= 0]
03:27:56,596 INFO  [Run] Start input action [o1.x3] calculation
03:27:56,596 INFO  [Run] Finish input action [o1.x3] calculation. Its value is [20]
03:27:56,596 DEBUG [Run] Try transition [TrueCoinReceived#TrueCoinReceived#e3#o1.x3>0]
03:27:56,596 INFO  [Run] Transition to go found
[TrueCoinReceived#TrueCoinReceived#e3#o1.x3>0]
03:27:56,596 INFO  [Run] Start output action [o1.z2] execution
03:27:56,596 INFO  [Run] Finish output action [o1.z2] execution
03:27:56,596 INFO  [Run] Finish event [e3] processing. In state
[/A1:TagReceiving/A3:TrueCoinReceived]
03:27:56,596 INFO  [Run] Finish event [e3] processing. In state [/A1:TagReceiving]
03:27:56,806 INFO  [Run] Start event [e3] processing. In state [/A1:TagReceiving]
03:27:56,806 INFO  [Run] Start event [e3] processing. In state
[/A1:TagReceiving/A3:TrueCoinReceived]
03:27:56,806 DEBUG [Run] Try transition [TrueCoinReceived#s2#e3#o1.x3 <= 0]
```

```
03:27:56,806 INFO  [Run] Start input action [o1.x3] calculation
03:27:56,806 INFO  [Run] Finish input action [o1.x3] calculation. Its value is [19]
03:27:56,806 DEBUG [Run] Try transition [TrueCoinReceived#TrueCoinReceived#e3#o1.x3>0]
03:27:56,806 INFO  [Run] Transition to go found
[TrueCoinReceived#TrueCoinReceived#e3#o1.x3>0]
03:27:56,806 INFO  [Run] Start output action [o1.z2] execution
03:27:56,816 INFO  [Run] Finish output action [o1.z2] execution
03:27:56,816 INFO  [Run] Finish event [e3] processing. In state
[/A1:TagReceiving/A3:TrueCoinReceived]
03:27:56,816 INFO  [Run] Finish event [e3] processing. In state [/A1:TagReceiving]
03:27:57,017 INFO  [Run] Start event [e3] processing. In state [/A1:TagReceiving]
03:27:57,017 INFO  [Run] Start event [e3] processing. In state
[/A1:TagReceiving/A3:TrueCoinReceived]
03:27:57,017 DEBUG [Run] Try transition [TrueCoinReceived#s2#e3#o1.x3 <= 0]
03:27:57,017 INFO  [Run] Start input action [o1.x3] calculation
03:27:57,017 INFO  [Run] Finish input action [o1.x3] calculation. Its value is [18]
03:27:57,017 DEBUG [Run] Try transition [TrueCoinReceived#TrueCoinReceived#e3#o1.x3>0]
03:27:57,017 INFO  [Run] Transition to go found
[TrueCoinReceived#TrueCoinReceived#e3#o1.x3>0]
03:27:57,017 INFO  [Run] Start output action [o1.z2] execution
03:27:57,017 INFO  [Run] Finish output action [o1.z2] execution
03:27:57,017 INFO  [Run] Finish event [e3] processing. In state
[/A1:TagReceiving/A3:TrueCoinReceived]
03:27:57,017 INFO  [Run] Finish event [e3] processing. In state [/A1:TagReceiving]
03:27:57,227 INFO  [Run] Start event [e3] processing. In state [/A1:TagReceiving]
03:27:57,227 INFO  [Run] Start event [e3] processing. In state
[/A1:TagReceiving/A3:TrueCoinReceived]
03:27:57,227 DEBUG [Run] Try transition [TrueCoinReceived#s2#e3#o1.x3 <= 0]
03:27:57,227 INFO  [Run] Start input action [o1.x3] calculation
03:27:57,227 INFO  [Run] Finish input action [o1.x3] calculation. Its value is [17]
03:27:57,227 DEBUG [Run] Try transition [TrueCoinReceived#TrueCoinReceived#e3#o1.x3>0]
03:27:57,227 INFO  [Run] Transition to go found
[TrueCoinReceived#TrueCoinReceived#e3#o1.x3>0]
03:27:57,227 INFO  [Run] Start output action [o1.z2] execution
03:27:57,227 INFO  [Run] Finish output action [o1.z2] execution
03:27:57,227 INFO  [Run] Finish event [e3] processing. In state
[/A1:TagReceiving/A3:TrueCoinReceived]
03:27:57,227 INFO  [Run] Finish event [e3] processing. In state [/A1:TagReceiving]
03:27:57,527 INFO  [Run] Start event [e3] processing. In state [/A1:TagReceiving]
03:27:57,527 INFO  [Run] Start event [e3] processing. In state
[/A1:TagReceiving/A3:TrueCoinReceived]
03:27:57,527 DEBUG [Run] Try transition [TrueCoinReceived#s2#e3#o1.x3 <= 0]
03:27:57,527 INFO  [Run] Start input action [o1.x3] calculation
03:27:57,527 INFO  [Run] Finish input action [o1.x3] calculation. Its value is [16]
03:27:57,527 DEBUG [Run] Try transition [TrueCoinReceived#TrueCoinReceived#e3#o1.x3>0]
03:27:57,527 INFO  [Run] Transition to go found
[TrueCoinReceived#TrueCoinReceived#e3#o1.x3>0]
03:27:57,527 INFO  [Run] Start output action [o1.z2] execution
03:27:57,527 INFO  [Run] Finish output action [o1.z2] execution
03:27:57,527 INFO  [Run] Finish event [e3] processing. In state
[/A1:TagReceiving/A3:TrueCoinReceived]
03:27:57,527 INFO  [Run] Finish event [e3] processing. In state [/A1:TagReceiving]
03:27:57,818 INFO  [Run] Start event [e3] processing. In state [/A1:TagReceiving]
03:27:57,818 INFO  [Run] Start event [e3] processing. In state
[/A1:TagReceiving/A3:TrueCoinReceived]
```

```
03:27:57,818 DEBUG [Run] Try transition [TrueCoinReceived#s2#e3#o1.x3 <= 0]
03:27:57,818 INFO  [Run] Start input action [o1.x3] calculation
03:27:57,818 INFO  [Run] Finish input action [o1.x3] calculation. Its value is [15]
03:27:57,818 DEBUG [Run] Try transition [TrueCoinReceived#TrueCoinReceived#e3#o1.x3>0]
03:27:57,818 INFO  [Run] Transition to go found
[TrueCoinReceived#TrueCoinReceived#e3#o1.x3>0]
03:27:57,818 INFO  [Run] Start output action [o1.z2] execution
03:27:57,818 INFO  [Run] Finish output action [o1.z2] execution
03:27:57,818 INFO  [Run] Finish event [e3] processing. In state
[/A1:TagReceiving/A3:TrueCoinReceived]
03:27:57,818 INFO  [Run] Finish event [e3] processing. In state [/A1:TagReceiving]
03:27:58,449 INFO  [Run] Start event [e2] processing. In state [/A1:TagReceiving]
03:27:58,449 DEBUG [Run] Try transition [TagReceiving#Play#e2#true]
03:27:58,449 INFO  [Run] Transition to go found [TagReceiving#Play#e2#true]
03:27:58,449 INFO  [Run] Start event [e2] processing. In state [/A1:Play/A2:Top]
03:27:58,449 INFO  [Run] Transition to go found [s1#Stopped##true]
03:27:58,449 DEBUG [Run] Try transition [Stopped#Rotation#e2#true]
03:27:58,449 INFO  [Run] Transition to go found [Stopped#Rotation#e2#true]
03:27:58,449 INFO  [Run] Start on-enter action [o1.z3] execution
03:28:04,788 INFO  [Run] Finish on-enter action [o1.z3] execution
03:28:04,788 INFO  [Run] Finish event [e2] processing. In state [/A1:Play/A2:Rotation]
03:28:04,788 INFO  [Run] Finish event [e2] processing. In state [/A1:Play]
03:28:04,788 INFO  [Run] Start event [e6] processing. In state [/A1:Play]
03:28:04,788 INFO  [Run] Start event [e6] processing. In state [/A1:Play/A2:Rotation]
03:28:04,788 DEBUG [Run] Try transition [Rotation#StoppingFirst#e6#true]
03:28:04,788 INFO  [Run] Transition to go found [Rotation#StoppingFirst#e6#true]
03:28:04,788 INFO  [Run] Finish event [e6] processing. In state [/A1:Play/A2:StoppingFirst]
03:28:04,788 INFO  [Run] Finish event [e6] processing. In state [/A1:Play]
03:28:04,788 INFO  [Run] Start event [e6] processing. In state [/A1:Play]
03:28:04,788 INFO  [Run] Start event [e6] processing. In state [/A1:Play/A2:StoppingFirst]
03:28:04,788 DEBUG [Run] Try transition [StoppingFirst#StoppingSecond#e6#true]
03:28:04,788 INFO  [Run] Transition to go found [StoppingFirst#StoppingSecond#e6#true]
03:28:04,788 INFO  [Run] Finish event [e6] processing. In state [/A1:Play/A2:StoppingSecond]
03:28:04,788 INFO  [Run] Finish event [e6] processing. In state [/A1:Play]
03:28:04,788 INFO  [Run] Start event [e6] processing. In state [/A1:Play]
03:28:04,788 INFO  [Run] Start event [e6] processing. In state [/A1:Play/A2:StoppingSecond]
03:28:04,788 DEBUG [Run] Try transition [StoppingSecond#s2#e6#true]
03:28:04,788 INFO  [Run] Transition to go found [StoppingSecond#s2#e6#true]
03:28:04,788 INFO  [Run] State machine came to final state [/A1:Play/A2:s2]
03:28:04,788 INFO  [Run] Finish event [e6] processing. In state [/A1:Play/A2:s2]
03:28:04,788 INFO  [Run] Finish event [e6] processing. In state [/A1:Play]
03:28:04,788 INFO  [Run] Start event [e7] processing. In state [/A1:Play]
03:28:04,788 DEBUG [Run] Try transition [Play#Error#e7#o1.x4 > 0 && o1.x1 <= 0]
03:28:04,788 INFO  [Run] Start input action [o1.x4] calculation
03:28:04,788 INFO  [Run] Finish input action [o1.x4] calculation. Its value is [1]
03:28:04,788 INFO  [Run] Start input action [o1.x1] calculation
03:28:04,788 INFO  [Run] Finish input action [o1.x1] calculation. Its value is [66]
03:28:04,788 DEBUG [Run] Try transition [Play#Waiting#e7#o1.x4<=0]
03:28:04,788 DEBUG [Run] Try transition [Play#givingWin#e7#o1.x4 >0 && o1.x1>0]
03:28:04,788 INFO  [Run] Transition to go found [Play#givingWin#e7#o1.x4 >0 && o1.x1>0]
03:28:04,788 INFO  [Run] Start output action [o1.z7] execution
03:28:04,988 INFO  [Run] Finish output action [o1.z7] execution
03:28:04,988 INFO  [Run] Finish event [e7] processing. In state [/A1:givingWin]
03:28:04,988 INFO  [Run] Start event [e5] processing. In state [/A1:givingWin]
03:28:04,988 DEBUG [Run] Try transition [givingWin#Error#e5#o1.x2>0 && o1.x1 <= 0]
```

```
03:28:04,988 INFO  [Run] Start input action [o1.x2] calculation
03:28:04,988 INFO  [Run] Finish input action [o1.x2] calculation. Its value is [11]
03:28:04,988 INFO  [Run] Start input action [o1.x1] calculation
03:28:04,988 INFO  [Run] Finish input action [o1.x1] calculation. Its value is [65]
03:28:04,988 DEBUG [Run] Try transition [givingWin#Waiting#e5#o1.x2 <= 0]
03:28:04,988 DEBUG [Run] Try transition [givingWin#givingWin#e5#o1.x2 > 0 && o1.x1>0]
03:28:04,988 INFO  [Run] Transition to go found [givingWin#givingWin#e5#o1.x2 > 0 &&
o1.x1>0]
03:28:04,988 INFO  [Run] Start output action [o1.z7] execution
03:28:05,188 INFO  [Run] Finish output action [o1.z7] execution
03:28:05,188 INFO  [Run] Finish event [e5] processing. In state [/A1:givingWin]
03:28:05,188 INFO  [Run] Start event [e5] processing. In state [/A1:givingWin]
03:28:05,188 DEBUG [Run] Try transition [givingWin#Error#e5#o1.x2>0 && o1.x1 <= 0]
03:28:05,188 INFO  [Run] Start input action [o1.x2] calculation
03:28:05,188 INFO  [Run] Finish input action [o1.x2] calculation. Its value is [10]
03:28:05,188 INFO  [Run] Start input action [o1.x1] calculation
03:28:05,188 INFO  [Run] Finish input action [o1.x1] calculation. Its value is [64]
03:28:05,188 DEBUG [Run] Try transition [givingWin#Waiting#e5#o1.x2 <= 0]
03:28:05,188 DEBUG [Run] Try transition [givingWin#givingWin#e5#o1.x2 > 0 && o1.x1>0]
03:28:05,188 INFO  [Run] Transition to go found [givingWin#givingWin#e5#o1.x2 > 0 &&
o1.x1>0]
03:28:05,188 INFO  [Run] Start output action [o1.z7] execution
03:28:05,389 INFO  [Run] Finish output action [o1.z7] execution
03:28:05,389 INFO  [Run] Finish event [e5] processing. In state [/A1:givingWin]
03:28:05,389 INFO  [Run] Start event [e5] processing. In state [/A1:givingWin]
03:28:05,389 DEBUG [Run] Try transition [givingWin#Error#e5#o1.x2>0 && o1.x1 <= 0]
03:28:05,389 INFO  [Run] Start input action [o1.x2] calculation
03:28:05,389 INFO  [Run] Finish input action [o1.x2] calculation. Its value is [9]
03:28:05,389 INFO  [Run] Start input action [o1.x1] calculation
03:28:05,389 INFO  [Run] Finish input action [o1.x1] calculation. Its value is [63]
03:28:05,389 DEBUG [Run] Try transition [givingWin#Waiting#e5#o1.x2 <= 0]
03:28:05,389 DEBUG [Run] Try transition [givingWin#givingWin#e5#o1.x2 > 0 && o1.x1>0]
03:28:05,389 INFO  [Run] Transition to go found [givingWin#givingWin#e5#o1.x2 > 0 &&
o1.x1>0]
03:28:05,389 INFO  [Run] Start output action [o1.z7] execution
03:28:05,589 INFO  [Run] Finish output action [o1.z7] execution
03:28:05,589 INFO  [Run] Finish event [e5] processing. In state [/A1:givingWin]
03:28:05,589 INFO  [Run] Start event [e5] processing. In state [/A1:givingWin]
03:28:05,589 DEBUG [Run] Try transition [givingWin#Error#e5#o1.x2>0 && o1.x1 <= 0]
03:28:05,589 INFO  [Run] Start input action [o1.x2] calculation
03:28:05,589 INFO  [Run] Finish input action [o1.x2] calculation. Its value is [8]
03:28:05,589 INFO  [Run] Start input action [o1.x1] calculation
03:28:05,589 INFO  [Run] Finish input action [o1.x1] calculation. Its value is [62]
03:28:05,589 DEBUG [Run] Try transition [givingWin#Waiting#e5#o1.x2 <= 0]
03:28:05,589 DEBUG [Run] Try transition [givingWin#givingWin#e5#o1.x2 > 0 && o1.x1>0]
03:28:05,589 INFO  [Run] Transition to go found [givingWin#givingWin#e5#o1.x2 > 0 &&
o1.x1>0]
03:28:05,589 INFO  [Run] Start output action [o1.z7] execution
03:28:05,789 INFO  [Run] Finish output action [o1.z7] execution
03:28:05,789 INFO  [Run] Finish event [e5] processing. In state [/A1:givingWin]
03:28:05,789 INFO  [Run] Start event [e5] processing. In state [/A1:givingWin]
03:28:05,789 DEBUG [Run] Try transition [givingWin#Error#e5#o1.x2>0 && o1.x1 <= 0]
03:28:05,789 INFO  [Run] Start input action [o1.x2] calculation
03:28:05,789 INFO  [Run] Finish input action [o1.x2] calculation. Its value is [7]
03:28:05,789 INFO  [Run] Start input action [o1.x1] calculation
```

```
03:28:05,789 INFO  [Run] Finish input action [o1.x1] calculation. Its value is [61]
03:28:05,789 DEBUG [Run] Try transition [givingWin#Waiting#e5#o1.x2 <= 0]
03:28:05,789 DEBUG [Run] Try transition [givingWin#givingWin#e5#o1.x2 > 0 && o1.x1>0]
03:28:05,789 INFO  [Run] Transition to go found [givingWin#givingWin#e5#o1.x2 > 0 &&
o1.x1>0]
03:28:05,789 INFO  [Run] Start output action [o1.z7] execution
03:28:05,990 INFO  [Run] Finish output action [o1.z7] execution
03:28:05,990 INFO  [Run] Finish event [e5] processing. In state [/A1:givingWin]
03:28:05,990 INFO  [Run] Start event [e5] processing. In state [/A1:givingWin]
03:28:05,990 DEBUG [Run] Try transition [givingWin#Error#e5#o1.x2>0 && o1.x1 <= 0]
03:28:05,990 INFO  [Run] Start input action [o1.x2] calculation
03:28:05,990 INFO  [Run] Finish input action [o1.x2] calculation. Its value is [6]
03:28:05,990 INFO  [Run] Start input action [o1.x1] calculation
03:28:05,990 INFO  [Run] Finish input action [o1.x1] calculation. Its value is [60]
03:28:05,990 DEBUG [Run] Try transition [givingWin#Waiting#e5#o1.x2 <= 0]
03:28:05,990 DEBUG [Run] Try transition [givingWin#givingWin#e5#o1.x2 > 0 && o1.x1>0]
03:28:05,990 INFO  [Run] Transition to go found [givingWin#givingWin#e5#o1.x2 > 0 &&
o1.x1>0]
03:28:05,990 INFO  [Run] Start output action [o1.z7] execution
03:28:06,190 INFO  [Run] Finish output action [o1.z7] execution
03:28:06,190 INFO  [Run] Finish event [e5] processing. In state [/A1:givingWin]
03:28:06,190 INFO  [Run] Start event [e5] processing. In state [/A1:givingWin]
03:28:06,190 DEBUG [Run] Try transition [givingWin#Error#e5#o1.x2>0 && o1.x1 <= 0]
03:28:06,190 INFO  [Run] Start input action [o1.x2] calculation
03:28:06,190 INFO  [Run] Finish input action [o1.x2] calculation. Its value is [5]
03:28:06,190 INFO  [Run] Start input action [o1.x1] calculation
03:28:06,190 INFO  [Run] Finish input action [o1.x1] calculation. Its value is [59]
03:28:06,190 DEBUG [Run] Try transition [givingWin#Waiting#e5#o1.x2 <= 0]
03:28:06,190 DEBUG [Run] Try transition [givingWin#givingWin#e5#o1.x2 > 0 && o1.x1>0]
03:28:06,190 INFO  [Run] Transition to go found [givingWin#givingWin#e5#o1.x2 > 0 &&
o1.x1>0]
03:28:06,190 INFO  [Run] Start output action [o1.z7] execution
03:28:06,390 INFO  [Run] Finish output action [o1.z7] execution
03:28:06,390 INFO  [Run] Finish event [e5] processing. In state [/A1:givingWin]
03:28:06,390 INFO  [Run] Start event [e5] processing. In state [/A1:givingWin]
03:28:06,390 DEBUG [Run] Try transition [givingWin#Error#e5#o1.x2>0 && o1.x1 <= 0]
03:28:06,390 INFO  [Run] Start input action [o1.x2] calculation
03:28:06,390 INFO  [Run] Finish input action [o1.x2] calculation. Its value is [4]
03:28:06,390 INFO  [Run] Start input action [o1.x1] calculation
03:28:06,390 INFO  [Run] Finish input action [o1.x1] calculation. Its value is [58]
03:28:06,390 DEBUG [Run] Try transition [givingWin#Waiting#e5#o1.x2 <= 0]
03:28:06,390 DEBUG [Run] Try transition [givingWin#givingWin#e5#o1.x2 > 0 && o1.x1>0]
03:28:06,390 INFO  [Run] Transition to go found [givingWin#givingWin#e5#o1.x2 > 0 &&
o1.x1>0]
03:28:06,390 INFO  [Run] Start output action [o1.z7] execution
03:28:06,591 INFO  [Run] Finish output action [o1.z7] execution
03:28:06,591 INFO  [Run] Finish event [e5] processing. In state [/A1:givingWin]
03:28:06,591 INFO  [Run] Start event [e5] processing. In state [/A1:givingWin]
03:28:06,591 DEBUG [Run] Try transition [givingWin#Error#e5#o1.x2>0 && o1.x1 <= 0]
03:28:06,591 INFO  [Run] Start input action [o1.x2] calculation
03:28:06,591 INFO  [Run] Finish input action [o1.x2] calculation. Its value is [3]
03:28:06,591 INFO  [Run] Start input action [o1.x1] calculation
03:28:06,591 INFO  [Run] Finish input action [o1.x1] calculation. Its value is [57]
03:28:06,591 DEBUG [Run] Try transition [givingWin#Waiting#e5#o1.x2 <= 0]
03:28:06,591 DEBUG [Run] Try transition [givingWin#givingWin#e5#o1.x2 > 0 && o1.x1>0]
```

```
03:28:06,591 INFO  [Run] Transition to go found [givingWin#givingWin#e5#o1.x2 > 0 &&
o1.x1>0]
03:28:06,591 INFO  [Run] Start output action [o1.z7] execution
03:28:06,791 INFO  [Run] Finish output action [o1.z7] execution
03:28:06,791 INFO  [Run] Finish event [e5] processing. In state [/A1:givingWin]
03:28:06,791 INFO  [Run] Start event [e5] processing. In state [/A1:givingWin]
03:28:06,791 DEBUG [Run] Try transition [givingWin#Error#e5#o1.x2>0 && o1.x1 <= 0]
03:28:06,791 INFO  [Run] Start input action [o1.x2] calculation
03:28:06,791 INFO  [Run] Finish input action [o1.x2] calculation. Its value is [2]
03:28:06,791 INFO  [Run] Start input action [o1.x1] calculation
03:28:06,791 INFO  [Run] Finish input action [o1.x1] calculation. Its value is [56]
03:28:06,791 DEBUG [Run] Try transition [givingWin#Waiting#e5#o1.x2 <= 0]
03:28:06,791 DEBUG [Run] Try transition [givingWin#givingWin#e5#o1.x2 > 0 && o1.x1>0]
03:28:06,791 INFO  [Run] Transition to go found [givingWin#givingWin#e5#o1.x2 > 0 &&
o1.x1>0]
03:28:06,791 INFO  [Run] Start output action [o1.z7] execution
03:28:06,991 INFO  [Run] Finish output action [o1.z7] execution
03:28:06,991 INFO  [Run] Finish event [e5] processing. In state [/A1:givingWin]
03:28:06,991 INFO  [Run] Start event [e5] processing. In state [/A1:givingWin]
03:28:06,991 DEBUG [Run] Try transition [givingWin#Error#e5#o1.x2>0 && o1.x1 <= 0]
03:28:06,991 INFO  [Run] Start input action [o1.x2] calculation
03:28:06,991 INFO  [Run] Finish input action [o1.x2] calculation. Its value is [1]
03:28:06,991 INFO  [Run] Start input action [o1.x1] calculation
03:28:06,991 INFO  [Run] Finish input action [o1.x1] calculation. Its value is [55]
03:28:06,991 DEBUG [Run] Try transition [givingWin#Waiting#e5#o1.x2 <= 0]
03:28:06,991 DEBUG [Run] Try transition [givingWin#givingWin#e5#o1.x2 > 0 && o1.x1>0]
03:28:06,991 INFO  [Run] Transition to go found [givingWin#givingWin#e5#o1.x2 > 0 &&
o1.x1>0]
03:28:06,991 INFO  [Run] Start output action [o1.z7] execution
03:28:07,191 INFO  [Run] Finish output action [o1.z7] execution
03:28:07,191 INFO  [Run] Finish event [e5] processing. In state [/A1:givingWin]
03:28:07,191 INFO  [Run] Start event [e5] processing. In state [/A1:givingWin]
03:28:07,191 DEBUG [Run] Try transition [givingWin#Error#e5#o1.x2>0 && o1.x1 <= 0]
03:28:07,191 INFO  [Run] Start input action [o1.x2] calculation
03:28:07,191 INFO  [Run] Finish input action [o1.x2] calculation. Its value is [0]
03:28:07,191 INFO  [Run] Start input action [o1.x1] calculation
03:28:07,191 INFO  [Run] Finish input action [o1.x1] calculation. Its value is [54]
03:28:07,191 DEBUG [Run] Try transition [givingWin#Waiting#e5#o1.x2 <= 0]
03:28:07,191 INFO  [Run] Transition to go found [givingWin#Waiting#e5#o1.x2 <= 0]
03:28:07,191 INFO  [Run] Start output action [o1.z8] execution
03:28:07,191 INFO  [Run] Finish output action [o1.z8] execution
03:28:07,191 INFO  [Run] Finish event [e5] processing. In state [/A1:Waiting]
03:28:08,954 INFO  [Run] Start event [e9] processing. In state [/A1:Waiting]
03:28:08,954 DEBUG [Run] Try transition [Waiting#Error#e9#true]
03:28:08,954 INFO  [Run] Transition to go found [Waiting#Error#e9#true]
03:28:08,954 INFO  [Run] Start output action [o1.z4] execution
03:28:08,954 INFO  [Run] Finish output action [o1.z4] execution
03:28:08,954 INFO  [Run] Finish event [e9] processing. In state [/A1:Error]
03:28:09,835 INFO  [Run] Start event [e4] processing. In state [/A1:Error]
03:28:09,835 DEBUG [Run] Try transition [Error#Waiting#e4#true]
03:28:09,835 INFO  [Run] Transition to go found [Error#Waiting#e4#true]
03:28:09,835 INFO  [Run] Start output action [o1.z5] execution
03:28:09,835 INFO  [Run] Finish output action [o1.z5] execution
03:28:09,835 INFO  [Run] Finish event [e4] processing. In state [/A1:Waiting]
03:28:10,636 INFO  [Run] Start event [e3] processing. In state [/A1:Waiting]
```

```
03:28:10,636 DEBUG [Run] Try transition [Waiting#TagReceiving#e3#true]
03:28:10,636 INFO  [Run] Transition to go found [Waiting#TagReceiving#e3#true]
03:28:10,636 INFO  [Run] Start event [e3] processing. In state
[/A1:TagReceiving/A3:TrueCoinReceived]
03:28:10,636 DEBUG [Run] Try transition [TrueCoinReceived#s2#e3#o1.x3 <= 0]
03:28:10,636 INFO  [Run] Start input action [o1.x3] calculation
03:28:10,636 INFO  [Run] Finish input action [o1.x3] calculation. Its value is [26]
03:28:10,636 DEBUG [Run] Try transition [TrueCoinReceived#TrueCoinReceived#e3#o1.x3>0]
03:28:10,636 INFO  [Run] Transition to go found
[TrueCoinReceived#TrueCoinReceived#e3#o1.x3>0]
03:28:10,636 INFO  [Run] Start output action [o1.z2] execution
03:28:10,636 INFO  [Run] Finish output action [o1.z2] execution
03:28:10,636 INFO  [Run] Finish event [e3] processing. In state
[/A1:TagReceiving/A3:TrueCoinReceived]
03:28:10,636 INFO  [Run] Finish event [e3] processing. In state [/A1:TagReceiving]
03:28:10,857 INFO  [Run] Start event [e3] processing. In state [/A1:TagReceiving]
03:28:10,857 INFO  [Run] Start event [e3] processing. In state
[/A1:TagReceiving/A3:TrueCoinReceived]
03:28:10,857 DEBUG [Run] Try transition [TrueCoinReceived#s2#e3#o1.x3 <= 0]
03:28:10,857 INFO  [Run] Start input action [o1.x3] calculation
03:28:10,857 INFO  [Run] Finish input action [o1.x3] calculation. Its value is [25]
03:28:10,857 DEBUG [Run] Try transition [TrueCoinReceived#TrueCoinReceived#e3#o1.x3>0]
03:28:10,857 INFO  [Run] Transition to go found
[TrueCoinReceived#TrueCoinReceived#e3#o1.x3>0]
03:28:10,857 INFO  [Run] Start output action [o1.z2] execution
03:28:10,857 INFO  [Run] Finish output action [o1.z2] execution
03:28:10,857 INFO  [Run] Finish event [e3] processing. In state
[/A1:TagReceiving/A3:TrueCoinReceived]
03:28:10,857 INFO  [Run] Finish event [e3] processing. In state [/A1:TagReceiving]
03:28:11,047 INFO  [Run] Start event [e3] processing. In state [/A1:TagReceiving]
03:28:11,047 INFO  [Run] Start event [e3] processing. In state
[/A1:TagReceiving/A3:TrueCoinReceived]
03:28:11,047 DEBUG [Run] Try transition [TrueCoinReceived#s2#e3#o1.x3 <= 0]
03:28:11,047 INFO  [Run] Start input action [o1.x3] calculation
03:28:11,047 INFO  [Run] Finish input action [o1.x3] calculation. Its value is [24]
03:28:11,047 DEBUG [Run] Try transition [TrueCoinReceived#TrueCoinReceived#e3#o1.x3>0]
03:28:11,047 INFO  [Run] Transition to go found
[TrueCoinReceived#TrueCoinReceived#e3#o1.x3>0]
03:28:11,047 INFO  [Run] Start output action [o1.z2] execution
03:28:11,047 INFO  [Run] Finish output action [o1.z2] execution
03:28:11,047 INFO  [Run] Finish event [e3] processing. In state
[/A1:TagReceiving/A3:TrueCoinReceived]
03:28:11,047 INFO  [Run] Finish event [e3] processing. In state [/A1:TagReceiving]
03:28:11,307 INFO  [Run] Start event [e3] processing. In state [/A1:TagReceiving]
03:28:11,307 INFO  [Run] Start event [e3] processing. In state
[/A1:TagReceiving/A3:TrueCoinReceived]
03:28:11,307 DEBUG [Run] Try transition [TrueCoinReceived#s2#e3#o1.x3 <= 0]
03:28:11,307 INFO  [Run] Start input action [o1.x3] calculation
03:28:11,307 INFO  [Run] Finish input action [o1.x3] calculation. Its value is [23]
03:28:11,307 DEBUG [Run] Try transition [TrueCoinReceived#TrueCoinReceived#e3#o1.x3>0]
03:28:11,307 INFO  [Run] Transition to go found
[TrueCoinReceived#TrueCoinReceived#e3#o1.x3>0]
03:28:11,307 INFO  [Run] Start output action [o1.z2] execution
03:28:11,307 INFO  [Run] Finish output action [o1.z2] execution
```

```
03:28:11,307 INFO  [Run] Finish event [e3] processing. In state
[/A1:TagReceiving/A3:TrueCoinReceived]
03:28:11,307 INFO  [Run] Finish event [e3] processing. In state [/A1:TagReceiving]
03:28:12,469 INFO  [Run] Start event [e4] processing. In state [/A1:TagReceiving]
03:28:12,469 DEBUG [Run] Try transition [TagReceiving#Waiting#e4#true]
03:28:12,469 INFO  [Run] Transition to go found [TagReceiving#Waiting#e4#true]
03:28:12,469 INFO  [Run] Start output action [o1.z5] execution
03:28:12,469 INFO  [Run] Finish output action [o1.z5] execution
03:28:12,469 INFO  [Run] Finish event [e4] processing. In state [/A1:Waiting]
```