Saint-Petersburg state university of
informational technologies, mechanics and optics
Department “Computer technologies”

D. Zakharov, S. Kosukhin

“One handed gangster” device simulation with *UniMod* tool

The project was created within the bounds of
“Open documentation movement”
http://is.ifmo.ru

Saint-Petersburg, 2006

Contents

Introduction

The task of developing software or its part is usually divided into three phases: analysis of the subject (we've got to answer the question "WHAT are we going to create?"), functional design ("HOW are we going to create this?") and implementation.

The first phase is the most important one, because mistakes made during its execution are of the highest cost. At this phase a particular supervision over the results of developer's actions is required. The second one plays a little less important role, and it is it, which presents the solution of the task, established at the first phase. It's necessary to describe in details what way the implementation is to be done. If functional design is really well done, then the third phase is only a translation of given description into source code.

SWITCH-technology [1–3] presents a new approach in functional design. It helps to forget for some time about source code and pay all the attention to essence of the task. For this they suggest to evolve controlled objects and event providers. And to describe behavior of the program, develop a system of interacting finite automata. For each automaton its states correspond to states of working program. In that way program's behavior is visually described. That's why this approach let us far easier avoid design mistakes, in the contrast to verbal project description.

Internet page http://is.ifmo.ru contains information about SWITCH-technology and projects built on its basis. *UniMod* tool helps to apply SWITCH-technology to construction of real program (http://unimod.sourceforge.net). This tool is a plug-in for *Eclipse* development framework (http://eclipse.org). It allows beginning the project with creation of link scheme, consisting of event providers, automata and controlled objects. Afterwards for each automaton a transition diagram is constructed.

There are initial and finite states at transition diagrams. For each of others ones there is a possibility to define transitions to another states at the income of definite event and fulfillment of definite conditions. During transitions and with entrance into states one can perform actions under controlled objects. These objects and event providers are implemented as *Java* classes, and their source code is written manually.

After link scheme and transition diagram are built we can immediately obtain working program with stubs. This can be done in two ways. With first one an *XML*-file, describing link scheme and transition diagrams is constructed automatically. This file is given as input to interpreter (one of *UniMod* components) altogether with manually written classes for event providers and controlled object (this is interpretive approach). The second way considers compilation of all the components into byte-code (compilation-based approach).

The approach, which *UniMod* is based on, let (in contrast to other open tools) build the program as a whole, but not its separate compenents.

The aim of this work is study and demonstration of *UniMod* tool possibilities with the example of simple program, simulator of playing machine "One handed gangster". The other simulator of this machine can be found in work http://is.ifmo.ru/projects/slotmachine/.

1. Description of the device's main nodes

Playing machine "One handed gangster", if implemented in real life, is autonomous and doesn't require any serving staff except for money excavation or adding.

Here is the description of virtual device, which is modeled in this work. The device consists of the following components:

- playing cylinders;
- monitor for showing the current state;
- panel with buttons for managing game process;
- coin mechanism;
- device, which manages rotation of cylinders;
- device for giving out a prize;

The device accepts coins of only one denomination, and they are considered to be identity elements of account. Any other tossed coin is considered to be false. A player can make a stake and rotate playing cylinders. After they stop the device counts the gain. Depending on combination at cylinders the player can either gain nothing (loose his money) or gain some sum. The amount of gained coins is defined as follows: with presence of any two similar digits it is equal to amount of stake, multiplied by 5. With the presence of three similar digits the stake is multiplied by 10.

After gain delivery the player can make a stake again and play until either he or bank is out of money.

User interface (fig. 1) presents the model of playing device.

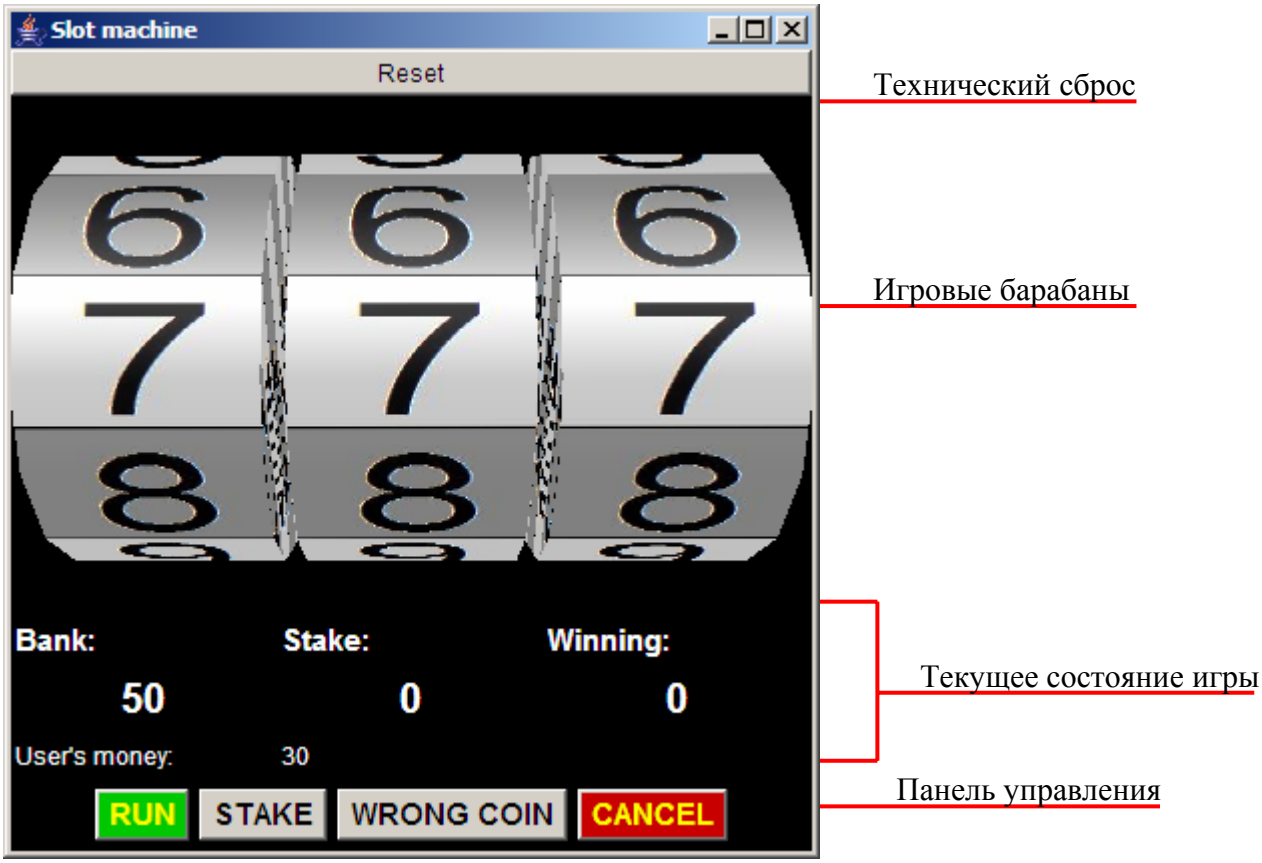

Fig. 1. User interface of application

In the middle of application's window there are playing cylinders, below them current state of the game is shown. Panel for game managing is located at the bottom of the window. At the beginning of the game it is assumed that the player has 30 coins, and the bank has 50. In case the bank is out of money there is possibility to start the game once more by pressing "Reset" button, which is located at the top of playing field.

The buttons at panel have the following meaning:

- "Play" – start playing (rotate cylinders);
- "Stake" – increment stake (toss true coin)
- "Bad coin" – toss false coin;
- "Return" – get back all coins from currently made stake.

There are messages for player about current game state and they are displayed at the bottom of the window, under the cylinders.

2. Behavior description

1. **The player makes stake.** Coin mechanism accepts all the coins, one by one, checking each for trueness. In case the coin is false, the mechanism tells the player about it and he can return all his coins by pressing “Return” button. In case of true coin the stake is incremented.
2. **The player rotates cylinders.** Cylinders rotate and stop one by one. Received combination of digits is used to calculate the gain. If it is zero, the device is ready for accepting next stake. In other case it gives the player his gain, one coin after another. If during this process the bank happens to be out of money, the player is told about that. After this one can return the device in initial state by hitting “Reset” button.

3. Design

3.1. Link scheme of automata

At the link scheme of automata (fig. 2) there are event providers, automata, controlled objects and links between them.

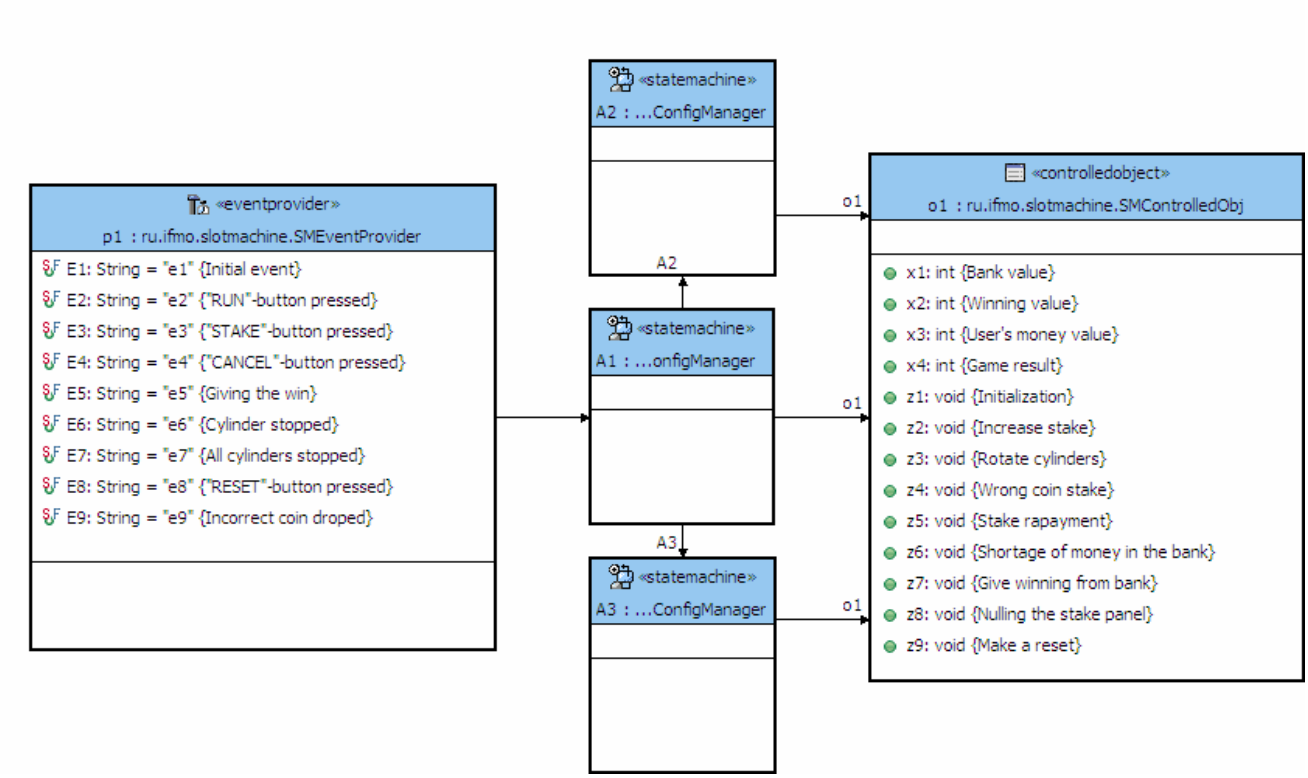

Fig. 2. Automata’s link scheme

At the left of the scheme there is event provider. This object will generate events during program execution.

At the middle of the scheme there are three automata. Behavior of each of them is defined by corresponding graph of transitions.

Generation of event means its translation to one of automata; in our case it is automaton A1 (the main one). For example, translation of `e1` event starts the main automaton, and event `e2` rotates cylinders.

At the right of the scheme there is controlled object. It encapsulates methods, which are called by automata. These methods can be of two types. Methods of the first type return information about object's state (they are denoted `x1`, `x2` etc.). They are used when describing conditions of transitions in automata. Methods of the second type allow changing state of the controlled object (they are denoted `z1`, `z2`, etc.).

3.2. Automata's transition graphs

3.2.1. The main automaton

The main automaton (fig. 3) unites the parts of the program (rotating the cylinders, acceptance of coins and the game process itself). It is it, which receives events from event provider and translate them to other two auxiliary automata if needed.

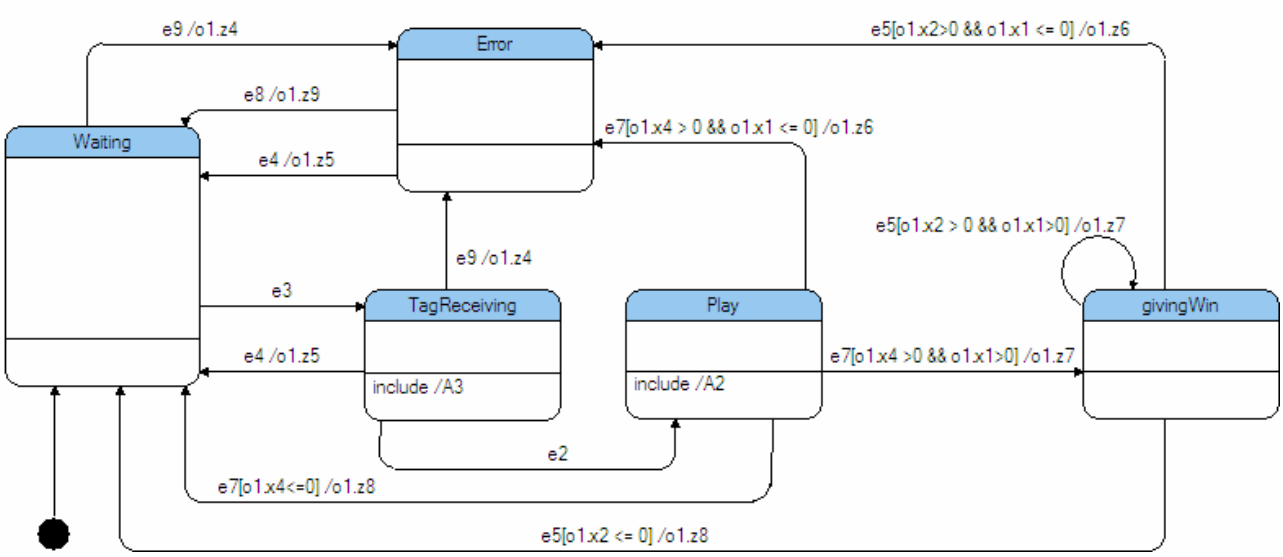

Fig. 3. The main automaton

The black circle denotes initial state of automaton. When automaton receives `e1` event, it changes its state to `Waiting`. Every time when receiving an event automaton can change its state. *UniMod* allows constructing conditions at transitions, which allow making transition iff given formula is true. Expressions at this formula are functions of controlled object, which return some value. For instance, the condition of transition

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

means, that the transition is made by event `e5`, but iff the controlled object has the result of cylinders rotation (it is got by `o1.x2` method) is greater than zero and amount of coins in bank (this is got by `o1.x2` method) is less or equal to zero.

The end of the condition `/o1.z6` means that with transition a method of controlled object `z6` will be called (it execute actions, needed with zero balance).

3.2.2. Automaton, responsible for cylinders rotation

Except for main automaton, there are two more, one of which is responsible for cylinders rotation, and another for coins acceptance.

Auxiliary automata are rather simple. First of them (fig. 4)

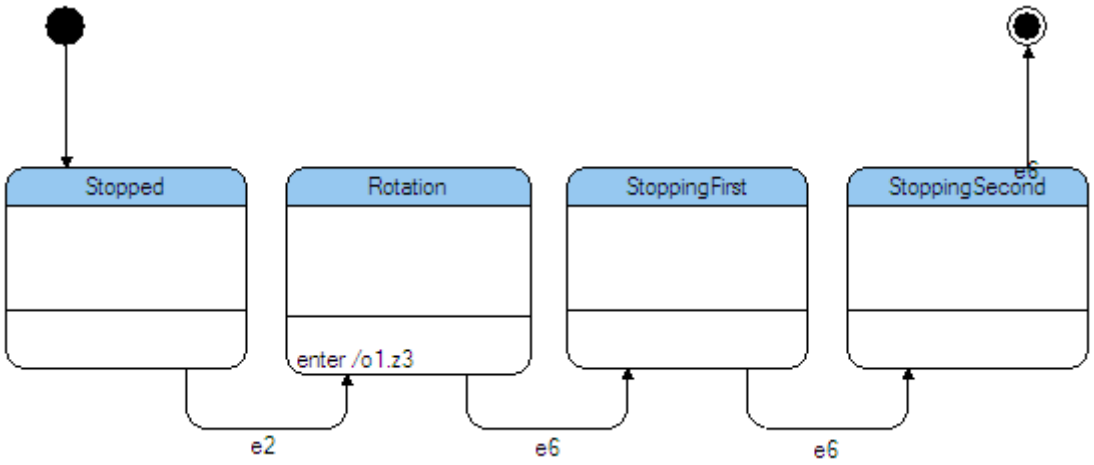

Fig. 4. Automaton, responsible for cylinders rotation

divides rotation of cylinders into 4 states. At this automaton action over controlled object are done when entering `Rotation` state. This is told by caption in the bottom of image `enter /o1.z3`. In this case method `z3` of controlled object is called, which starts rotation of cylinders.

3.2.3. Automaton, responsible for coins acceptance

Another automaton is responsible for coin acceptance (fig. 5). It stays in its main state **TrueCoinReceived** until the player has coins.

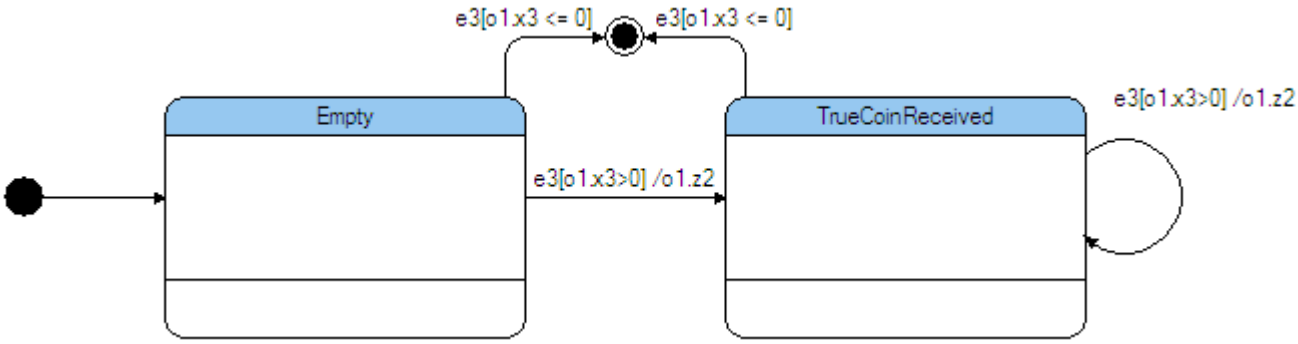

Fig. 5. Automaton, responsible for coins acceptance

4. Program execution

Pay attention that for launching the program DirectX 9.0c library is needed.

UniMod suggests two approaches for execution of constructed application: interpretive and compilation-based. Within the **interpretive** approach an *XML*-file is generated, which describes the link scheme of automata and transition graphs (see Enclosure 2), and *Java* classes of controlled objects and event providers are compiled into byte-code. Interpreter (one of *UniMod* components) uses *XML*-file to manage the program execution. To launch the application with usage of interpretive approach, enter in command line:

call *Java* -jar SM.jar A1.XML

This command launches *Java* application, which is included into **SM.jar** archive with **A1.XML** parameter. Here **A1.XML** is an automatically generated by *UniMod* tool file, which contains link scheme of automata and graphs of transitions.

Within the compilation-based approach *XML*-file is translated into *Java* code with the help of Velocity program (http://jakarta.apache.org/velocity) for creation of working application. To launch this application, created within compilation-based approach it is necessary just to run the *Java* application.

5. Statistics

When developing the application the part of source code, denoted within compilation-based approach by diagrams, is generated by *UniMod* tool automatically. At the diagram (fig. 6) proportion of manually written and automatically generated code is presented.

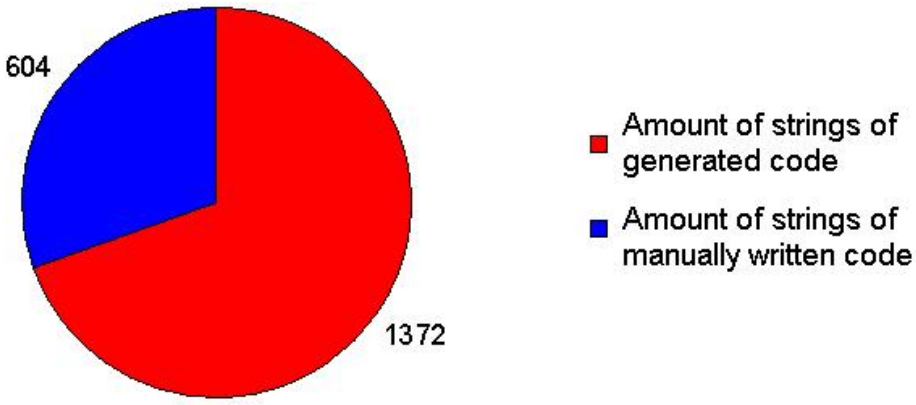

Fig. 6. Statistics of code strings written manually and generated automatically

Post amble

The amount of applications, developed with a SWITCH-technology as a base, is growing. *UniMod* is a convenient tool for projecting such applications. It allows to see clearly the structure of the program before it is written. In this work there is an example of application, developed with *UniMod* tool.

References

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/.

Enclosure 1. Source codes

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 Initial event
     */
    public static final String E1 = "e1";

    /**
     * @unimod.event.descr "RUN"-button pressed
     */
    public static final String E2 = "e2";

    /**
     * @unimod.event.descr "STAKE"-button pressed
     */
    public static final String E3 = "e3";

    /**
     * @unimod.event.descr "CANCEL"-button pressed
     */
    public static final String E4 = "e4";

    /**
     * @unimod.event.descr Giving the win
     */
    public static final String E5 = "e5";

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

    /**
     * @unimod.event.descr All cylinders stopped
     */
    public static final String E7 = "e7";

    /**
     * @unimod.event.descr "RESET"-button pressed
     */
    public static final String E8 = "e8";

    /**
     * @unimod.event.descr Incorrect coin droped
     */
    public static final String E9 = "e9";

    SMControlledObj SM;
```

```
    public void init(ModelEngine engine) throws CommonException {
        SM = (SMControlledObj) engine.getControlledObjectsManager()
                .getControlledObject("o1");// Getting of controlled object
        SM.setEngine(engine); //Setting engine for controlled object

        SM.z1(StateMachineContextImpl.create());
        engine.getEventManager().handle(new Event(E1),
                StateMachineContextImpl.create());// Generation of the initial
event

    }

    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.*;

import javax.vecmath.*;

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;// Pointer to managing engine

    private int stake;

    private int bank;

    private int winsize;

    private int usermoney;

    private TransformGroup[] objLocalRotAr;// Array of groups for cylinder
rotations

    private Label lStakeDyn;// Stake panel
```

```
    private Label lWinDyn;// Winning panel

    private Label lBankDyn;// Bank panel

    private Label lWalDyn;// User's money panel

    private Label lWalFalse;// Error panel

    public int[] current;// Array with last game result

    /*
     * Setting the managing engine
     */
    public void setEngine(ModelEngine engine) {
        this.engine = engine;
    }

    /*
     * Refresh of all panels
     */
    public void refreshLabels() {
        lBankDyn.setText("" + bank);
        lStakeDyn.setText("" + stake);
        lWinDyn.setText("" + winsize);
        lWalDyn.setText("" + usermoney);
    }

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

        // Setting initial values
        stake = 0;
        bank = 50;
        winsize = 0;
        usermoney = 30;

        // Creation of all panels
        lBankDyn = new Label();
        lStakeDyn = new Label();
        lWinDyn = new Label();
        lWalDyn = new Label();
        lWalFalse = new Label("");// No errors at the start
        lWalFalse.setForeground(new Color(255, 0, 0));// Setting attributes
                                                      // for error panel:
                                                      // red font
        refreshLabels();// Representation of initial values

        /*
         * 3D graphics
         */

        // Color constants
        final Color3f white = new Color3f(1.0f, 1.0f, 1.0f);// white color
        final Color3f grey = new Color3f(1.0f, 1.0f, 1.0f);// grey (70%) color

        // Root of the scene graph
        BranchGroup objRoot = new BranchGroup();

        // Light generation
        BoundingSphere bounds = new BoundingSphere();// Border of the light
spreading
```

```
        DirectionalLight light = new DirectionalLight(white, new Vector3f(1f,
                -1f, -1f));
        light.setInfluencingBounds(bounds);
        objRoot.addChild(light);

        AmbientLight ambientLightNode = new AmbientLight(grey);
        ambientLightNode.setInfluencingBounds(bounds);
        objRoot.addChild(ambientLightNode);

        // Texture loading for cylinders
        URL fileName = SMControlledObj.class.getResource("tex.jpg");
        TextureLoader loader = new TextureLoader(fileName, "RGBA",
                new Container());
        Texture texture = loader.getTexture();
        texture.setBoundaryModeS(Texture.WRAP);
        texture.setBoundaryModeT(Texture.WRAP);
        texture.setBoundaryColor(new Color4f(0.0f, 1.0f, 0.0f, 0.0f));
        TextureAttributes texAttr = new TextureAttributes();
        texAttr.setTextureMode(TextureAttributes.MODULATE);

        // Generating cylinders' appearance
        Appearance ap = new Appearance();
        ap.setTexture(texture);
        ap.setTextureAttributes(texAttr);

        ap.setMaterial(new Material(white, grey, white, white, 20.0f));
        int primflags = Primitive.GENERATE_NORMALS
                + Primitive.GENERATE_TEXTURE_COORDS;

        // Groups for rotation and translation of cylinders
        TransformGroup objGlobalRotate = new TransformGroup();// Group for global
rotation of cylinders
        Transform3D transGlobalRot = new Transform3D();
        transGlobalRot.setRotation(new AxisAngle4f(0.0f, 0.0f, -1.0f, 1.57f));
        objGlobalRotate.setTransform(transGlobalRot);
        objRoot.addChild(objGlobalRotate);

        TransformGroup[] objLocalTransAr = new TransformGroup[3];// Group for
translation of cylinders
        objLocalRotAr = new TransformGroup[3];//Groups for cyliners' rotation
        current = new int[3];// Last game results

        // Finishing the scene
        for (int i = 0; i < 3; i++) {
            objLocalTransAr[i] = new TransformGroup();

            Transform3D transLocalTrans = new Transform3D();
            transLocalTrans.setTranslation(new Vector3f(0.0f, -0.55f + i
                    * 0.55f, 0.0f));
            objLocalTransAr[i].setTransform(transLocalTrans);

            objGlobalRotate.addChild(objLocalTransAr[i]);

            objLocalRotAr[i] = new TransformGroup();

            Transform3D transLocalRot = new Transform3D();
            transLocalRot.setRotation(new AxisAngle4f(0.0f, 1.0f, 0.0f,
                    angleForDigit(7)));
            objLocalRotAr[i].setTransform(transLocalRot);

            objLocalRotAr[i]
                    .setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);//
Setting permission  for rotation
            objLocalTransAr[i].addChild(objLocalRotAr[i]);
```

```
            objLocalRotAr[i].addChild(new Cylinder(0.5f, 0.5f, primflags, ap));//
Creation of ith cylinder

            current[i] = 7;// Initial number is 7

        }


        /*
         * Creation of control elements
         */

        GraphicsConfiguration config = SimpleUniverse
                .getPreferredConfiguration();
        Canvas3D canvas = new Canvas3D(config);

        // Buttons' creation
        Button bRun = new Button("RUN");// Button for starting game
        bRun.setBackground(new Color(0, 200, 0));
        bRun.setFont(new Font("Dialog", Font.BOLD, 15));
        bRun.setForeground(new Color(255, 255, 0));

        Button bCancel = new Button("CANCEL");// Button for stake repayment
        bCancel.setBackground(new Color(200, 0, 0));
        bCancel.setFont(new Font("Dialog", Font.BOLD, 15));
        bCancel.setForeground(new Color(255, 255, 0));

        Button bStake = new Button("STAKE");// Button for making stake
        bStake.setFont(new Font("Dialog", Font.BOLD, 15));
        bStake.setForeground(new Color(0, 0, 0));

        Button bFalse = new Button("WRONG COIN");// button of dropping of
incorrect coin
        bFalse.setFont(new Font("Dialog", Font.BOLD, 15));
        bFalse.setForeground(new Color(0, 0, 0));

        Button bReset = new Button("Reset");// Button to make reset
        bReset.setForeground(new Color(0, 0, 0));

        // Reactions on user's actions
        bRun.addActionListener(new ActionListener() {
                    public void actionPerformed(ActionEvent e) {
                        ModelEngine engine = SMControlledObj.this.engine;
                        engine.getEventManager().handle(
                                new Event(SMEventProvider.E2),
                                StateMachineContextImpl.create());
                    }
                });
        bCancel.addActionListener(new ActionListener() {
                    public void actionPerformed(ActionEvent e) {
                        ModelEngine engine = SMControlledObj.this.engine;
                        engine.getEventManager().handle(
                                new Event(SMEventProvider.E4),
                                StateMachineContextImpl.create());
                    }
                });
        bStake.addActionListener(new ActionListener() {
                    public void actionPerformed(ActionEvent e) {
                        ModelEngine engine = SMControlledObj.this.engine;
                        engine.getEventManager().handle(
                                new Event(SMEventProvider.E3),
                                StateMachineContextImpl.create());
                    }
                });
```

```
bFalse.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                ModelEngine engine = SMControlledObj.this.engine;
                engine.getEventManager().handle(
                        new Event(SMEventProvider.E9),
                        StateMachineContextImpl.create());
            }
        });
bReset.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                ModelEngine engine = SMControlledObj.this.engine;
                engine.getEventManager().handle(
                        new Event(SMEventProvider.E8),
                        StateMachineContextImpl.create());
            }
        });


setLayout(new BorderLayout());
add("Center", canvas);
Panel panelHalf = new Panel();
panelHalf.setBackground(new Color(0, 0, 0));
panelHalf.setForeground(new Color(255, 255, 255));
Panel panelLab = new Panel();
panelLab.setLayout(new GridLayout());
Panel pBank = new Panel();
pBank.setLayout(new BorderLayout());
Label lBankStatic = new Label("Bank:");
lBankStatic.setFont(new Font("Dialog", Font.BOLD, 15));
lBankDyn.setFont(new Font("Dialog", Font.BOLD, 20));
lBankDyn.setAlignment(Label.CENTER);
pBank.add("North", lBankStatic);
pBank.add("Center", lBankDyn);
Panel pStake = new Panel();
pStake.setLayout(new BorderLayout());
Label lStakeStatic = new Label("Stake:");
lStakeStatic.setFont(new Font("Dialog", Font.BOLD, 15));
lStakeDyn.setFont(new Font("Dialog", Font.BOLD, 20));
lStakeDyn.setAlignment(Label.CENTER);
pStake.add("North", lStakeStatic);
pStake.add("Center", lStakeDyn);
Panel pWin = new Panel();
pWin.setLayout(new BorderLayout());
Label lWinStatic = new Label("Winning:");
lWinStatic.setFont(new Font("Dialog", Font.BOLD, 15));
lWinDyn.setFont(new Font("Dialog", Font.BOLD, 20));
lWinDyn.setAlignment(Label.CENTER);
pWin.add("North", lWinStatic);
pWin.add("Center", lWinDyn);
panelLab.add(pBank);
panelLab.add(pStake);
panelLab.add(pWin);
Panel panelBut = new Panel();
panelBut.setLayout(new FlowLayout());
panelBut.add(bRun);
panelBut.add(bStake);
panelBut.add(bFalse);
panelBut.add(bCancel);
Panel panelWal = new Panel();
panelWal.setLayout(new GridLayout());
Label lWalStat = new Label("User's money: ");
panelWal.add(lWalStat);
panelWal.add(lWalDyn);
panelWal.add(lWalFalse);
```

```
        panelHalf.setLayout(new BorderLayout());
        panelHalf.add("North", panelLab);
        panelHalf.add("Center", panelWal);
        panelHalf.add("South", panelBut);
        add("South", panelHalf);
        add("North", bReset);

        objRoot.compile();
        SimpleUniverse universe = new SimpleUniverse(canvas);
        universe.getViewingPlatform().setNominalViewingTransform();
        universe.addBranchGraph(objRoot);

        /*
         * Application window
         */

        MainFrame main = new MainFrame(this, 400, 400);
        main.setTitle("Slot machine");

    }

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

    public static float angleForDigit(int dig) {
        return 0.3f + 2 * 0.31415f * (5 - dig); // Computing the rotation angle
for digit
    }

    /**
     * @unimod.action.descr Rotate cylinders
     */
    public void z3(StateMachineContext context) {
        ModelEngine engine = SMControlledObj.this.engine;
        Date date = new Date();
        int seed = (int) date.getTime();//Getting the seed for random number
computation
        Random rand = new Random(seed);
        int[] rec = new int[3];
        for (int i = 0; i < 3; i++) {
            rec[i] = rand.nextInt();
            rec[i] = Math.abs(rec[i]) % 10;
        }

        Transform3D[] transLocalRot = new Transform3D[3];

        float[] angleStep = new float[3];

        for (int i = 0; i < 3; i++) {
            transLocalRot[i] = new Transform3D();
            transLocalRot[i].setRotation(new AxisAngle4f(0.0f, 1.0f, 0.0f,
                    angleForDigit(current[i])));
            objLocalRotAr[i].setTransform(transLocalRot[i]);
        }
        int rotationSteps = 500;// Number of iterations for first cylinder
        int rotationStepsDif = 200;//Difference between cylinders' iterations
```

```
        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);// Computing the
angle delta

                if (k == rotationSteps + i * rotationStepsDif) {
                    engine.getEventManager().handle(
                            new Event(SMEventProvider.E6),
                            StateMachineContextImpl.create());
                }
                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);// To show rotation we should stop for a while
            } catch (InterruptedException e) {
            }
        }

        for (int i = 0; i < 3; i++) {
            current[i] = rec[i];
        }
        engine.getEventManager().handle(new Event(SMEventProvider.E6),
                StateMachineContextImpl.create());

        engine.getEventManager().handle(new Event(SMEventProvider.E7),
                StateMachineContextImpl.create());
    }

    /**
     * @unimod.action.descr Wrong coin stake
     */
    public void z4(StateMachineContext context) {
        lWalFalse.setText("Wrong coin!");
    }

    /**
     * @unimod.action.descr Stake rapayment
     */
    public void z5(StateMachineContext context) {
        bank -= stake;
        usermoney += stake;
        stake = 0;
        winsize = 0;
        refreshLabels();
        lWalFalse.setText("");
    }

    /**
     * @unimod.action.descr Shortage of money in the bank
     */
    public void z6(StateMachineContext context) {
        lWalFalse.setText("Shortage of money!");
    }

    /**
```

```
     * @unimod.action.descr Give winning from bank
     */
    public void z7(StateMachineContext context) {
        bank--;
        winsize--;
        usermoney++;
        refreshLabels();

        try {
            Thread.sleep(200);
        } catch (InterruptedException e) {
        }

        ModelEngine engine = SMControlledObj.this.engine;
        engine.getEventManager().handle(new Event(SMEventProvider.E5),
                StateMachineContextImpl.create());

    }

    /**
     * @unimod.action.descr Nulling the stake panel
     */
    public void z8(StateMachineContext context) {
        stake = 0;
        refreshLabels();
    }

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

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

    /**
     * @unimod.action.descr User's money value
     */
    public int x3(StateMachineContext context) {
        return usermoney;
    }

    /**
     * @unimod.action.descr Game result
     */
    public int x4(StateMachineContext context) {
        if (current[0] == current[1] || current[1] == current[2]
                || current[0] == current[2]) {//Two equal numbers mean winning
(*)
            if (current[0] == current[1] && current[1] == current[2])//Three
equal numbers mean
                winsize = stake * 5;// winning 3 times
            else
                winsize = stake * 2;//(*) 2 times
            refreshLabels();

            return 1;// Winning
        } else
```

```
            return 0;// Losing
    }

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

Enclosure 2. *XML* description of automata

```
<?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>
```

Enclosure 3. Generated by *XML* description *Java* code

Model1EventProcessor.java

```
/**
  * This file was generated from model [Model1] on [Fri Oct 13 03:24:26 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;
    }

}
```

Enclosure 4. Example of program working protocol

```
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]