Санкт-Петербургский государственный институт точной механики и оптики
(технический университет)

Кафедра «Компьютерных технологий»

М.А. Подтопельный, А.А. Чеботарева, А.А. Шалыто

Робот, ищущий выход из лабиринта

Объектно-ориентированное программирование с явным выделением состояний

Проектная документация

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

Санкт-Петербург
2003

В соответствии со строением вестибулярного аппарата человеку при ходьбе свойственно забирать влево. Поэтому путь от входа до кассы лежит против часовой стрелки через весь супермаркет.

Чтобы купить, человек должен остановиться и сфокусироваться на товаре. Психологи сделали «открытие»: остановиться человека заставляет поворот на 90 градусов.

Еженедельник «Петербург на Невском». 2003. №2

В 1950 году Клод Шеннон сконструировал механическую мышь (управляемую с помощью магнита и скрытой под полом сложной электрической схемы), способную находить выход из лабиринта.

Шеннон К.Э. Работы по теории информации и кибернетике.
М.: Изд-во иностр. лит., 1963

Введение

С тех пор, как впервые появилась проблема поиска выхода из лабиринта, было предложено множество способов ее решения. Для алгоритмизации и программирования решения этой задачи оказалось удобным применить SWITCH-технологию. Подробно ознакомиться с этой технологией и с конкретными примерами ее использования можно на сайтах http://is.ifmo.ru и http://www.softcraft.ru .

При реализации проекта была разработана среда, способная отображать клетчатое поле, на котором для формирования лабиринта могут быть расставлены стены.

Для прохода по лабиринту применяется робот, интеллект которого описывается графом переходов конечного автомата. Действия робота протоколируются *xml*-файлом и выводятся при достижении роботом точки выхода из лабиринта в формате *html*.

При использовании среды могут быть построены лабиринты, отвечающие следующим требованиям:

- лабиринт плоский (не имеет третьего измерения);
- все его коридоры параллельны или перпендикулярны друг другу;
- лабиринт не изменяется во время его прохода роботом.

Внешний вид среды в процессе прохода робота по лабиринту представлен на рис. 1.

Толщина стен и коридоров лабиринта считается одинаковой, так как это не влияет на решение задачи. Кроме того, лабиринт содержит перекрестки, в каждом из которых «встречаются» как минимум два коридора разных направлений.

Входные данные для среды задаются *xml*-файлом, в котором указаны:

- размеры лабиринта;
- координаты точки входа в лабиринт *S*;
- координаты точки выхода из лабиринта *F*;
- конфигурация лабиринта (список пар координат, задающих стены лабиринта).

Для реализации проекта были выбраны технология *Flash* и язык *Action Script*, как наиболее перспективные и удобные инструменты для визуализации проектов, работающих в сети

Internet. Использование этих средств, а также автоматного подхода, позволило обеспечить удобство разработки и наглядность проекта, а также резко сократить трудоемкость написания программы.

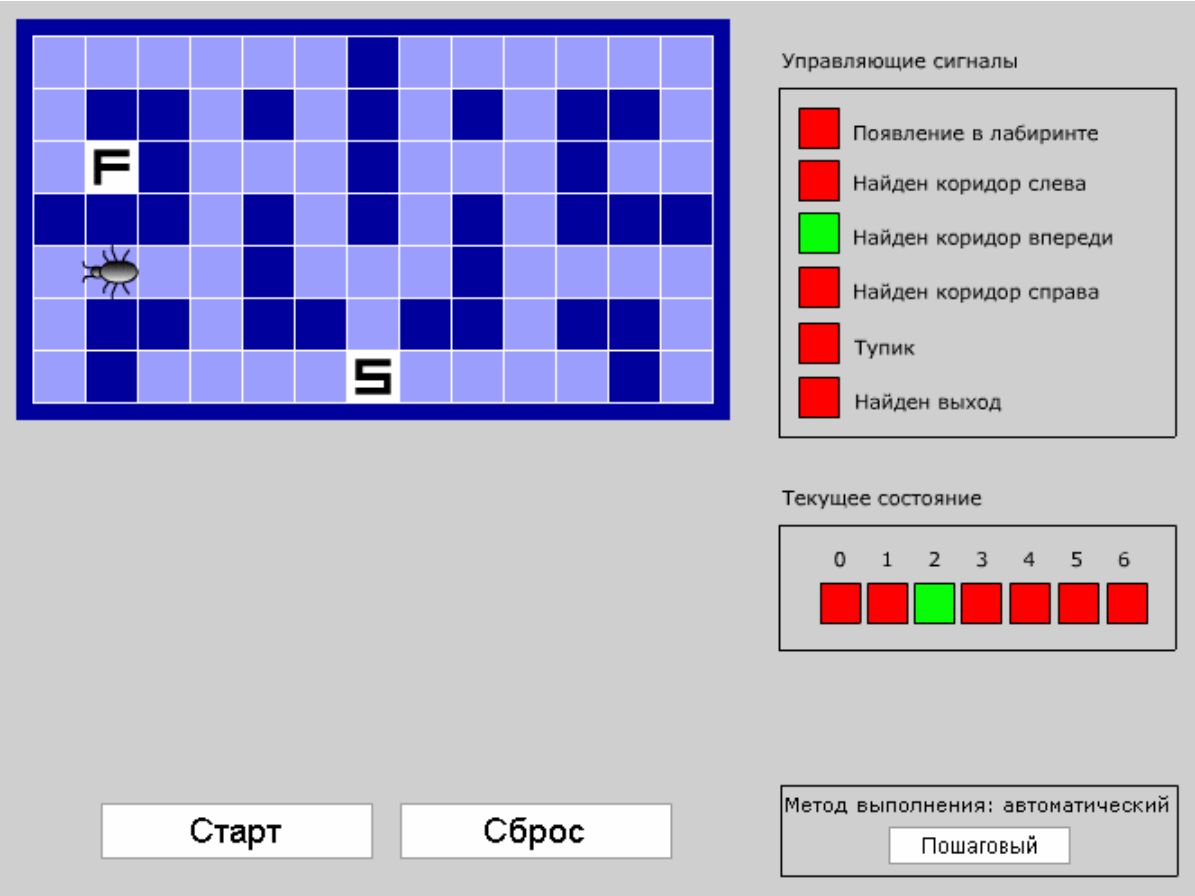

Рис. 1. Внешний вид среды исполнения

1. Постановка задачи

Целью настоящей работы является реализация алгоритма обхода лабиринта по правилу «левой руки».

2. Автомат «левой руки»

2.1. Словесное описание

Правило «левой руки» состоит в том, что робот идет, постоянно «держась рукой» за левую стену. При этом на каждом шаге выбирается самый левый из существующих коридоров.

Если найден коридор слева по ходу движения робота, то он выбирает этот коридор (поворачивает на 90^0 влево).

Если на этом шаге коридор слева отсутствует, но найден коридор впереди, робот идет вперед. Если нет коридоров слева и спереди, но есть коридор справа, робот поворачивает на 90^0 направо.

Если коридоров нет (робот оказался в тупике), то он поворачивает на 180^0 и идет в обратном направлении. Поворот на 180^0 в программе реализуется как два последовательных поворота налево на 90^0 каждый.

Для нахождения выхода из лабиринта точка входа в него не должна находиться на пути зацикленного маршрута. В этом случае, если на каждом шаге при выборе самого левого из возможных коридоров путь замыкается и робот возвращается в исходную точку, то он снова и снова будет идти этим маршрутом, и не сможет найти выход. Робот сам по себе не может попасть в такой цикл. Других ограничений в работе робота нет.

Робот выполняет следующие действия:

- инициализируется при входе;
- на каждом шаге «оглядывается по сторонам» в поисках коридоров;
- при необходимости совершает поворот налево или направо (в противном случае направление движения не меняется);
- совершает очередной шаг в выбранном направлении;
- протоколирует свое поведение, описывая каждый шаг.

2.2. Схема связей

Схема связей автомата «левой руки» представлена на рис. 2.

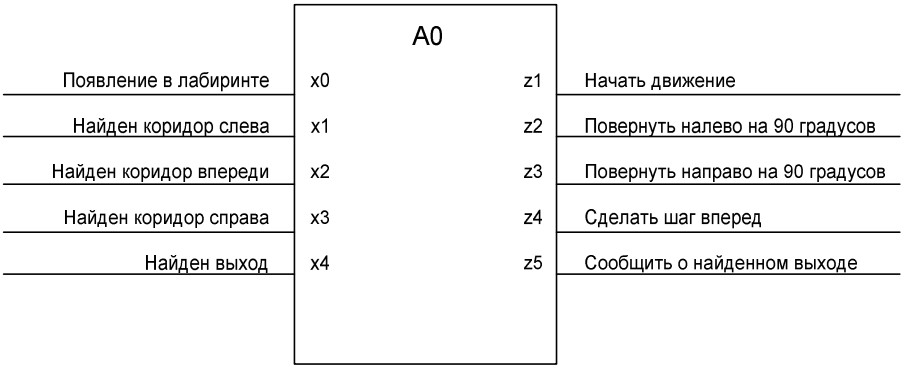

Рис.2. Схема связей автомата «левой руки»

2.3. Граф переходов

Граф переходов автомата «левой руки» представлен на рис. 3.

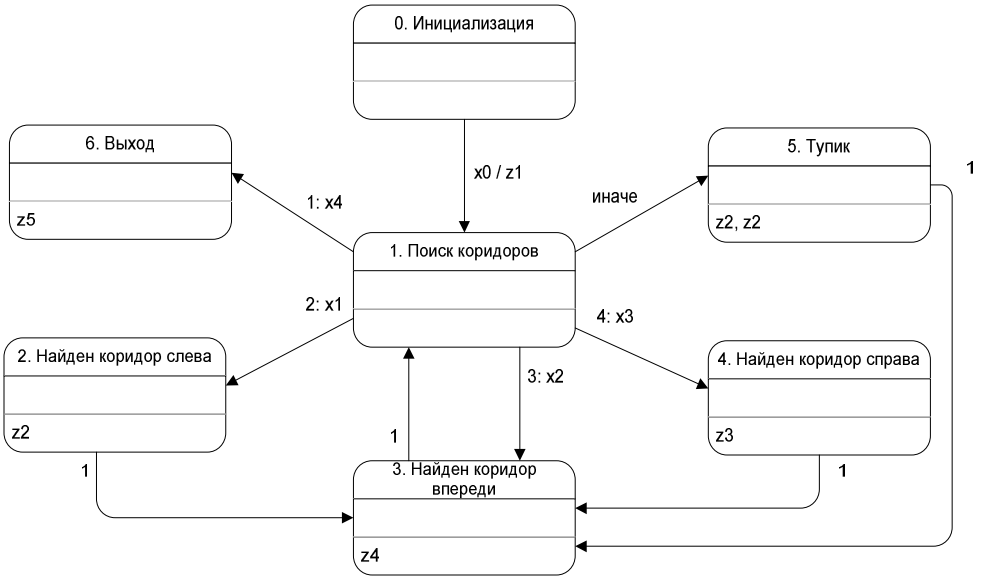

Рис.3. Граф переходов автомата «левой руки»

Обратим внимание, что состояния 2 - 5 могут быть устранены, если соответствующие им выходные функции вызывать на дугах (рис. 4).

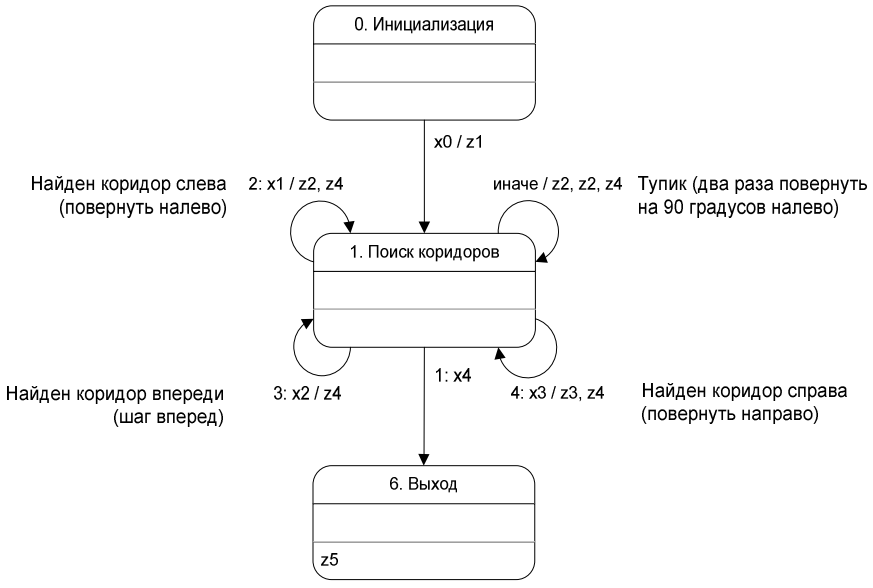

Рис. 4. Граф переходов автомата «левой руки». Минимизация состояний

Несмотря на то, что граф переходов на рис. 3 содержит большее количество вершин по сравнению с графом на рис. 4, в первом из них действие «шаг вперед» (z4) реализуется однократно, в то время как во втором – на каждой из четырех петель.
По указанной причине, а также ввиду того, что состояния удобно протоколировать, в программе реализуется граф с большим количеством вершин.

3. Протоколирование действий робота

3.1. Протоколирование

Во время работы программы можно непрерывно наблюдать за состоянием робота. Для этого разработана система индикаторов, которая позволяет в каждый момент времени наблюдать за управляющими сигналами и текущим состоянием робота. Внешний вид панели индикаторов представлен на рис. 5. Для удобства наблюдения можно переключить непрерывный (автоматический) режим выполнения на пошаговый.

Рис. 5. Внешний вид панели индикаторов

В процессе выполнения программы действия и состояния робота протоколируются в *xml*-файле. Когда робот находит выход из лабиринта, среда предлагает пользователю просмотреть полученный файл. Далее приведен пример такого файла для лабиринта, представленного на рис. 1.

3.2. Пример протокола

Номер шага: 0
Горизонтальная позиция: 6
Вертикальная позиция: 6
Состояние: 1
Направление движения: вверх
Есть ли коридор слева: true
Есть ли коридор впереди: true
Есть ли коридор справа: true

Номер шага: 2
Горизонтальная позиция: 6
Вертикальная позиция: 6
Состояние: 2
Направление движения: вверх
Есть ли коридор слева: true
Есть ли коридор впереди: true
Есть ли коридор справа: true

Номер шага: 3
Горизонтальная позиция: 5
Вертикальная позиция: 6
Состояние: 2
Направление движения: влево
Есть ли коридор слева: false
Есть ли коридор впереди: true
Есть ли коридор справа: false

Номер шага: 4
Горизонтальная позиция: 4
Вертикальная позиция: 6
Состояние: 2
Направление движения: влево
Есть ли коридор слева: false
Есть ли коридор впереди: true
Есть ли коридор справа: false

Номер шага: 5
Горизонтальная позиция: 3
Вертикальная позиция: 6
Состояние: 2
Направление движения: влево
Есть ли коридор слева: false
Есть ли коридор впереди: true
Есть ли коридор справа: true

Номер шага: 6
Горизонтальная позиция: 2
Вертикальная позиция: 6
Состояние: 2
Направление движения: влево
Есть ли коридор слева: false
Есть ли коридор впереди: false
Есть ли коридор справа: false

Номер шага: 7
Горизонтальная позиция: 3
Вертикальная позиция: 6
Состояние: 2
Направление движения: вправо
Есть ли коридор слева: true
Есть ли коридор впереди: true
Есть ли коридор справа: false

Номер шага: 8
Горизонтальная позиция: 3
Вертикальная позиция: 5
Состояние: 2
Направление движения: вверх
Есть ли коридор слева: false
Есть ли коридор впереди: true
Есть ли коридор справа: false

Номер шага: 9
Горизонтальная позиция: 3
Вертикальная позиция: 4
Состояние: 2
Направление движения: вверх
Есть ли коридор слева: true
Есть ли коридор впереди: true
Есть ли коридор справа: false

Номер шага: 10
Горизонтальная позиция: 2
Вертикальная позиция: 4
Состояние: 2
Направление движения: влево
Есть ли коридор слева: false
Есть ли коридор впереди: true
Есть ли коридор справа: false

Номер шага: 11
Горизонтальная позиция: 1
Вертикальная позиция: 4
Состояние: 2
Направление движения: влево
Есть ли коридор слева: false
Есть ли коридор впереди: true
Есть ли коридор справа: false

Номер шага: 12
Горизонтальная позиция: 0
Вертикальная позиция: 4
Состояние: 2
Направление движения: влево
Есть ли коридор слева: true
Есть ли коридор впереди: false
Есть ли коридор справа: false

Номер шага: 13
Горизонтальная позиция: 0
Вертикальная позиция: 5
Состояние: 2
Направление движения: вниз
Есть ли коридор слева: false
Есть ли коридор впереди: true
Есть ли коридор справа: false

Номер шага: 14
Горизонтальная позиция: 0
Вертикальная позиция: 6
Состояние: 2
Направление движения: вниз
Есть ли коридор слева: false
Есть ли коридор впереди: false
Есть ли коридор справа: false

Номер шага: 15
Горизонтальная позиция: 0
Вертикальная позиция: 5
Состояние: 2
Направление движения: вверх
Есть ли коридор слева: false
Есть ли коридор впереди: true
Есть ли коридор справа: false

Номер шага: 16
Горизонтальная позиция: 0
Вертикальная позиция: 4
Состояние: 2
Направление движения: вверх
Есть ли коридор слева: false
Есть ли коридор впереди: false
Есть ли коридор справа: true

Номер шага: 17
Горизонтальная позиция: 1
Вертикальная позиция: 4
Состояние: 2
Направление движения: вправо
Есть ли коридор слева: false
Есть ли коридор впереди: true
Есть ли коридор справа: false

Номер шага: 18
Горизонтальная позиция: 2
Вертикальная позиция: 4
Состояние: 2
Направление движения: вправо
Есть ли коридор слева: false
Есть ли коридор впереди: true
Есть ли коридор справа: false

Номер шага: 19
Горизонтальная позиция: 3
Вертикальная позиция: 4
Состояние: 2
Направление движения: вправо
Есть ли коридор слева: true
Есть ли коридор впереди: false
Есть ли коридор справа: true

Номер шага: 20
Горизонтальная позиция: 3
Вертикальная позиция: 3
Состояние: 2
Направление движения: вверх
Есть ли коридор слева: false
Есть ли коридор впереди: true
Есть ли коридор справа: false

Номер шага: 21
Горизонтальная позиция: 3
Вертикальная позиция: 2
Состояние: 2
Направление движения: вверх
Есть ли коридор слева: false
Есть ли коридор впереди: true
Есть ли коридор справа: true

Номер шага: 22
Горизонтальная позиция: 3
Вертикальная позиция: 1
Состояние: 2
Направление движения: вверх
Есть ли коридор слева: false
Есть ли коридор впереди: true
Есть ли коридор справа: false

Номер шага: 23
Горизонтальная позиция: 3
Вертикальная позиция: 0
Состояние: 2
Направление движения: вверх
Есть ли коридор слева: true
Есть ли коридор впереди: false
Есть ли коридор справа: true

Номер шага: 24
Горизонтальная позиция: 2
Вертикальная позиция: 0
Состояние: 2
Направление движения: влево
Есть ли коридор слева: false
Есть ли коридор впереди: true
Есть ли коридор справа: false

Номер шага: 25
Горизонтальная позиция: 1
Вертикальная позиция: 0
Состояние: 2
Направление движения: влево
Есть ли коридор слева: false
Есть ли коридор впереди: true
Есть ли коридор справа: false

Номер шага: 26
Горизонтальная позиция: 0
Вертикальная позиция: 0
Состояние: 2
Направление движения: влево
Есть ли коридор слева: true
Есть ли коридор впереди: false
Есть ли коридор справа: false

Номер шага: 27
Горизонтальная позиция: 0
Вертикальная позиция: 1
Состояние: 2
Направление движения: вниз
Есть ли коридор слева: false
Есть ли коридор впереди: true
Есть ли коридор справа: false

Номер шага: 28
Горизонтальная позиция: 0
Вертикальная позиция: 2
Состояние: 2
Направление движения: вниз
Есть ли коридор слева: true
Есть ли коридор впереди: false
Есть ли коридор справа: false

Номер шага: 29
Горизонтальная позиция: 1
Вертикальная позиция: 2
Состояние: 2
Направление движения: вправо
Есть ли коридор слева: false
Есть ли коридор впереди: false
Есть ли коридор справа: false

Номер шага: 30
Горизонтальная позиция: 1
Вертикальная позиция: 2
Состояние: 6
Направление движения: вправо
Есть ли коридор слева: false
Есть ли коридор впереди: false
Есть ли коридор справа: false

4. Заключение

Результатом проекта можно считать подтверждение целесообразности применения SWITCH-технологии для создания интерактивных *flash*-клипов, поскольку сама идеология *flash*-программирования схожа в основных принципах с автоматным программированием, так как, в частности, кадры *flash* могут быть сопоставлены с состояниями автомата.

Работающую версию проекта можно посмотреть в сети *Internet* по адресу http://rain.ifmo.ru/~podtop/fl1/.

Приложение. Текст программы

Код автомата (файл automat.as)

```
//Список входных переменных
var x0 = true;    //Робот находится в начальной позиции
var x1 = false;   //Найден коридор слева относительно направления
                  //движения робота
var x2 = false;   //Найден коридор впереди
var x3 = false;   //Найден коридор справа относительно направления
                  //движения робота
var x4 = false;   //Робот достиг конечной позиции

//Список внутренних переменных
var state = 0;       //Текущее состояние робота
var direction = 0;   //Направление движения - четыре варианта
var xpos = 0;        //Текущая горизонтальная позиция робота в лабиринте
var ypos = 0;        //Текущая вертикальная позиция робота в лабиринте
var xml_data = "";   //Строка, предназначенная для передачи отчетов
                     //о действиях робота
var left = false;    //Указывает, Есть ли коридор в клетке левее робота
                     //(в абсолютной системе координат)
var right = false;   //Указывает, Есть ли коридор в клетке правее робота
var up = false;      //Указывает, Есть ли коридор в клетке выше робота
var down = false;    //Указывает, Есть ли коридор в клетке ниже робота
var counter = 0;     //Счетчик шагов робота
```

//Выходные воздействия

```
//Функция, заставляющая робота начать движение по лабиринту
function z1() {
  LogStep();
  x0 = false;
}

//Функция, осуществляющая поворот робота на 90 градусов влево
function z2() {
  for (var i = 0; i <= 90; i += 5) {
    _rotation = direction * 90 - i;
    stop();
    play();
  }
  direction += 3;
  direction %= 4;
  cur_rotation = 91;
}


//Функция, осуществляющая поворот робота на 90 градусов вправо
function z3() {
  for (var i = 0; i <= 90; i += 5) {
    _rotation = direction * 90 + i;
    stop();
```

```
    play();
  }
  direction += 1;
  direction %= 4;
  cur_rotation = 91;
}

//Функция, осуществляющая шаг робота вперед в выбранном направлении
function z4() {
  for (var i = 1; i <= 20; i++) {
    if (direction == 0) _y -= (_root.sq_side+1)/20;
    if (direction == 1) _x += (_root.sq_side+1)/20;
    if (direction == 2) _y += (_root.sq_side+1)/20;
    if (direction == 3) _x -= (_root.sq_side+1)/20;
    stop();
    play();
  }
  if (direction == 0) ypos--;
  if (direction == 1) xpos++;
  if (direction == 2) ypos++;
  if (direction == 3) xpos--;
  cur_position = 0;
}

//Функция, заставляющая робота остановиться на финише
function z5() {
  LogStep();
  return;
}

//Функция протоколирования поведения робота
function LogStep() {
  xml_data = '<step><number>'+counter+'</number>';
  xml_data += '<x>'+xpos+'</x><y>'+ypos+'</y>
              <direction>'+direction+'</direction>';
  xml_data += '<state>'+state+'</state>';
  xml_data += '<left>'+x1+'</left><forward>'+x2+'</forward>
              <right>'+x3+'</right>';
  xml_data += '</step>';
  var eventlog = new XML(xml_data+"\n");
  eventlog.contentType = "text/xml";
  var x1 = new XML();
  x1.onLoad = function(success) {}
  eventlog.sendAndLoad("http://rain.ifmo.ru/~podtop/fl1/server.php",x1);
  counter++;
}

//Основная функция: заставляет робота сделать очередное действие
function Step() {
  if (counter == 0) {
    xpos = _root["maze"]["attributes"]["entrance_x"]-1+1;
    ypos = _root["maze"]["attributes"]["entrance_y"]-1+1;
  }
  //Сброс входных переменных, характеризующих наличие коридоров слева,
  //спереди и справа от робота
  x1 = false;
  x2 = false;
  x3 = false;
```

```
  //Вычисление переменной left для текущего положения
  if (xpos-1 < 0) left = false;
  else left = _root.maze["matrix"][xpos-1][ypos]["passage"];

  //Вычисление переменной right для текущего положения
  if (xpos+1 >= _root.maze["attributes"]["width"]) right = false;
  else right = _root.maze["matrix"][xpos+1][ypos]["passage"];

  //Вычисление переменной up для текущего положения
  if (ypos-1 < 0) up = false;
  else up = _root.maze["matrix"][xpos][ypos-1]["passage"];

  //Вычисление переменной down для текущего положения
  if (ypos+1 >= _root.maze["attributes"]["height"]) down = false;
  else down = _root.maze["matrix"][xpos][ypos+1]["passage"];

  //На основе вычисленных значений указанных переменных устанавливаются
  //соответствующие значения входных переменных автомата

  if (direction == 0) {
  //Если робот смотрит вверх
    x1 = left;
    x2 = up;
    x3 = right;

  } else if (direction == 1) {
  //Если робот смотрит вправо
    x1 = up;
    x2 = right;
    x3 = down;

  } else if (direction == 2) {
  //Если робот смотрит вниз
    x1 = right;
    x2 = down;
    x3 = left;

  } else if (direction == 3) {
  //Если робот смотрит влево
    x1 = down;
    x2 = left;
    x3 = up;
  }

  //Используется для индикации состояния тупика
  x = !x1 && !x2 && !x3;

  //Если робот пришел на финиш, то формируется значение переменной x4
  //по следующему правилу:
  x4 = (_root.maze["attributes"]["exit_x"] == xpos) &&
(_root.maze["attributes"]["exit_y"] == ypos);

  //Протоколирование совершенного шага
  LogStep();
```

```
  //Реализация автомата

  switch (state) {
  case 0:
    if (x0){                  z1();          state = 1};
    break;
  case 1:
    if (x4)                                  state = 6;
    else if (x1)                             state = 2;
    else if (x2)                             state = 3;
    else if (x3)                             state = 4;
    else                                     state = 5;
    break;
  case 2:
                              z2();          state = 3;
    break;
  case 3:
                              z4();          state = 1;
    break;
  case 4:
                              z3();          state = 3;
    break;
  case 5:
                              z2(); z2();    state = 3;
    break;
  case 6:                     z5();
    break;
  }
}
```

Код среды (файл frame1.as)

```
var maze = new Array();
_root.square._visible = false;
_root.start._visible = false;
_root.finish._visible = false;
_root.robot._visible = false;
_root.exit_found._visible = false;
_root.sq_side = 10;


//Функция, создающая лабиринт
function CreateMaze(width, height) {
  maze["attributes"] = new Array();
  maze["attributes"]["entrance_x"] = 0;
  maze["attributes"]["entrance_y"] = 0;
  maze["attributes"]["exit_x"] = width-1;
  maze["attributes"]["exit_y"] = height-1;
  maze["attributes"]["width"] = width;
  maze["attributes"]["height"] = height;
  maze["matrix"] = Array(width);
  for (var i = 0; i < width; i++) {
    maze["matrix"][i] = new Array(height);
    for (var j = 0; j < height; j++) {
```

```
      maze["matrix"][i][j] = new Array();
      maze["matrix"][i][j]["passage"] = true;
      maze["matrix"][i][j]["painted"] = 0;
    }
  }
}

//Функция, считывающая параметры лабиринта из xml-файла
function ReadMaze(maze_xml) {
  if (maze_xml.firstChild.nodeName == "maze") {
    var w = maze_xml.firstChild.attributes.width-1+1;
    var h = maze_xml.firstChild.attributes.height-1+1;
    createMaze(w, h);
    for (var i = 0; i < maze_xml.firstChild.childNodes.length; i++) {
      var tmp = maze_xml.firstChild.childNodes[i];
      if (tmp.nodeName != "") {
        if (tmp.nodeName == "checkpoints") {
          var x = tmp.firstChild.attributes.x;
          var y = tmp.firstChild.attributes.y;
          if ((x >= 0) && (x < w) && (y >= 0) && (y < h)) {
            maze["attributes"]["entrance_x"] = x;
            maze["attributes"]["entrance_y"] = y;
          }
          var x = tmp.lastChild.attributes.x;
          var y = tmp.lastChild.attributes.y;
          if ((x >= 0) && (x < w) && (y >= 0) && (y < h)) {
            maze["attributes"]["exit_x"] = x;
            maze["attributes"]["exit_y"] = y;
          }
        } else if (tmp.nodeName == "wall") {
          var x = tmp.attributes.x;
          var y = tmp.attributes.y;
          if ((x >= 0) && (x < w) && (y >= 0) && (y < h)) {
            maze["matrix"][x][y]["passage"] = false;
          }
        }
      }
    }
  } else {
    trace("wrong maze xml");
  }
}

//Функция, рисующая лабиринт на основе считанных из xml-файла данных
function RenderMaze() {
  var side = Math.max(maze["attributes"]["width"],
maze["attributes"]["height"]);
  sq_side = Math.floor(400/side)-2;
  var level = 0;
  with (_root["sq"]) {
    moveTo(-10, -10);
    beginFill(0x000099);
    lineStyle();
    lineTo((sq_side+1)*maze["attributes"]["width"] + 9, -10);
    lineTo((sq_side+1)*maze["attributes"]["width"] + 9,
(sq_side+1)*maze["attributes"]["height"] + 9);
    lineto(-10, (sq_side+1)*maze["attributes"]["height"] + 9);
    endFill();
```

```
    moveTo(-1, -1);
    beginFill(0xFFFFFF);
    lineStyle();
    lineTo((sq_side+1)*maze["attributes"]["width"], -1);
    lineTo((sq_side+1)*maze["attributes"]["width"],
(sq_side+1)*maze["attributes"]["height"]);
    lineto(-1, (sq_side+1)*maze["attributes"]["height"]);
    endFill();
  }
  duplicateMovieClip(_root.start, "start1",
maze["attributes"]["width"]*maze["attributes"]["height"]+10);
  duplicateMovieClip(_root.finish, "finish1",
maze["attributes"]["width"]*maze["attributes"]["height"]+11);
  duplicateMovieClip(_root.robot, "robot1",
maze["attributes"]["width"]*maze["attributes"]["height"]+12);
  duplicateMovieClip(_root.exit_found, "exit_found1",
maze["attributes"]["width"]*maze["attributes"]["height"]+13);
  with (_root["start1"]) {
    _width = sq_side;
    _height = sq_side;
    _x = (sq_side+1)*maze["attributes"]["entrance_x"]+20;
    _y = (sq_side+1)*maze["attributes"]["entrance_y"]+20;
    _visible = true;
  }
  with (_root["finish1"]) {
    _width = sq_side;
    _height = sq_side;
    _x = (sq_side+1)*maze["attributes"]["exit_x"]+20;
    _y = (sq_side+1)*maze["attributes"]["exit_y"]+20;
    _visible = true;
  }
  with (_root["robot1"]) {
    _x = (sq_side+1)*maze["attributes"]["entrance_x"]+20+sq_side/2;
    _y = (sq_side+1)*maze["attributes"]["entrance_y"]+20+sq_side/2;
    _visible = true;
  }
  with (_root["exit_found1"]) {
    _x = 350 - _width / 2;
    _y = 250 - _height / 2;
    _visible = false;
  }
  for (var i = 0; i < maze["attributes"]["width"]; i++) {
    for (var j = 0; j < maze["attributes"]["height"]; j++) {
      duplicateMovieClip(_root.square, "sq"+i+"_"+j, level++);
      with (_root["sq"+i+"_"+j]) {
        _x = (sq_side+1)*i+20;
        _y = (sq_side+1)*j+20;
        _width = sq_side;
        _height = sq_side;
        _visible = true;
        if (maze["matrix"][i][j]["passage"]) {
          wall._visible = false;
          twice_painted._visible = false;
          painted._visible = false;
          passage._visible = true;
        } else {
          wall._visible = true;
          twice_painted._visible = false;
```

```
          painted._visible = false;
          passage._visible = false;
        }
      }
    }
  }
}


function MazeOnLoad(success) {
  if (success) {                  //Если загрузка удалась, то
    ReadMaze(this);               //читаем данные лабиринта из xml-файла
                                  //во внутренние переменные

    RenderMaze();                 //Рисуем лабиринт

  } else {                        //Если загрузка не удалась, то
    trace("failed to open URL");  //выводим сообщение об ошибке
  }
}

//Создаем объект xml для чтения лабиринта с сервера
m = new XML();

//Определяем обработчик загрузки лабиринта
m.onLoad = MazeOnLoad;

//Отправляем запрос на загрузку лабиринта
m.load("http://rain.ifmo.ru/~podtop/fl1/test.xml");

//Останавливаем воспроизведение клипа flash
stop();

var paused = true;
var method = "continuous";
_root["st0"].gotoAndStop(2);
_root["x0"].gotoAndStop(2);
```

Описание лабиринта (файл test.xml)

```
<maze width="13" height="7">
  <checkpoints><entrance x="6" y="6" /><exit x="1" y="2"
/></checkpoints>
  <wall x="6" y="0" />
  <wall x="1" y="1" />
  <wall x="2" y="1" />
  <wall x="4" y="1" />
  <wall x="6" y="1" />
  <wall x="8" y="1" />
  <wall x="10" y="1" />
  <wall x="11" y="1" />
  <wall x="2" y="2" />
  <wall x="6" y="2" />
  <wall x="10" y="2" />
  <wall x="0" y="3" />
  <wall x="1" y="3" />
  <wall x="2" y="3" />
  <wall x="4" y="3" />
  <wall x="6" y="3" />
  <wall x="8" y="3" />
  <wall x="10" y="3" />
  <wall x="11" y="3" />
  <wall x="12" y="3" />
  <wall x="4" y="4" />
  <wall x="8" y="4" />
  <wall x="1" y="5" />
  <wall x="2" y="5" />
  <wall x="4" y="5" />
  <wall x="5" y="5" />
  <wall x="7" y="5" />
  <wall x="8" y="5" />
  <wall x="10" y="5" />
  <wall x="11" y="5" />
  <wall x="1" y="6" />
  <wall x="11" y="6" />
</maze>
```