Санкт-Петербургский государственный университет
информационных технологий, механики и оптики
Факультет информационных технологий и программирования
Кафедра компьютерных технологий

А.А. Зезюкин, А.А. Шалыто

Карточная игра *Triple Triad*

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

Санкт-Петербург
2006

Оглавление

Введение

Автоматное программирование [1] предлагает рассматривать программу в общем случае как систему взаимодействующих конечных автоматов специального вида, аналогичных автоматам, используемым в логическом управлении. У каждого автомата имеются состояния и переходы. Он взаимодействует с *источниками событий* и *объектами управления*. Переходы между состояниями происходят только по событиям и, возможно, при выполнении некоторых условий, относящихся к объектам управления. Инструментальное средство *UniMod* позволяет визуально проектировать и реализовывать автоматные программы. Автоматы представляются в виде стандартных диаграмм *UML* [2]. Каждый автомат описывается диаграммой состояний, а взаимодействие автоматов, источников событий и объектов управления — диаграммой классов.

Может сложиться впечатление, что автоматное программирование позволяет эффективно проектировать только те программы, которые занимаются управлением какой-нибудь машиной (например, светофором) или технологическим процессом, или еще чем-нибудь в этом роде. Так как подобные задачи управления как раз и решаются в теории управления, достаточно перевести полученные решения на язык программиста — и дизайн функциональной части программы можно считать готовым. Однако автоматный подход применим и для «обычных» прикладных программ.

Классически при построении автоматов используются довольно громоздкие блоки выбора (`switch` в языке *Java*), ухудшающие читабельность программы. Однако, перейдя при помощи соответствующего рефакторинга от `switch` к виртуальным методам [3], можно поддерживать код объектно-ориентированным и вполне читабельным, в то же время реализуя автоматы произвольной сложности.

Цель настоящей работы — проектирование программы для карточной игры *Triple Triad* (правила игры описаны в Приложении 1) с использованием автоматного программирования. Компьютерная игра — типичная прикладная программа, поэтому ее проектирование представляет определенный интерес. В работе применялось инструментальное средство *UniMod*, среда разработки *Eclipse 3.1*. Программа написана на языке *Java* версии 1.4.

В работе использованы идея и графическое оформление из игры *Final Fantasy VIII* © *Square* Co., Ltd.

1. Описание программы

После запуска окно программы выглядит так, как показано на рис. 1.

Рис.1. Внешний вид программы

Меню «Программа» содержит единственный пункт «Выход», закрывающий программу. Меню «Игра» позволяет начинать игру, а во время игры приостанавливать и продолжать ее. Когда игра приостановлена, нажатия клавиш не вызывают никакого эффекта до тех пор, пока игра не продолжится.

Практически все управление в процессе игры выполняется с помощью клавиатуры: по умолчанию, клавиши со стрелками предназначены для выбора карт или перемещения курсора, клавиша “A” (латинская) — для подтверждения действия, а клавиша “Z” — для отмены действия.

Для начала игры следует выбрать пункт меню «Начать игру» из меню «Игра». Игрокам случайным образом будут выданы по пять карт, и случайным образом будет выбран один из двух игроков, который должен ходить первым. В игре установлен следующий набор правил: *SAME*, *COMBO*, *OPEN*, *RANDOM*.

Для того чтобы сделать ход, игроку следует сначала выбрать карту, пользуясь клавишами со стрелками «вверх» и «вниз» (край выбранной карты слегка выступает из ряда остальных карт, как это показано на рис. 2) и нажать клавишу “A”.

Рис.2. Идет игра. Ход синего игрока

Затем при помощи клавиш со стрелками следует выбрать свободное пустое место на игровом поле и нажать клавишу “A”, чтобы положить выбранную карту на выбранное место, или нажать “Z”, чтобы вернуться к выбору карты.

После того, как карта положена, автоматически будут выполнены все возможные (в рамках правил) перевороты карт с интервалом в полсекунды. Затем ход перейдет к другому игроку. После того как все игровое поле будет заполнено (конец игры), можно нажать клавишу “A” для того, чтобы сыграть еще раз.

2. Структура программы

Программа состоит из двух автоматов (рис.3).

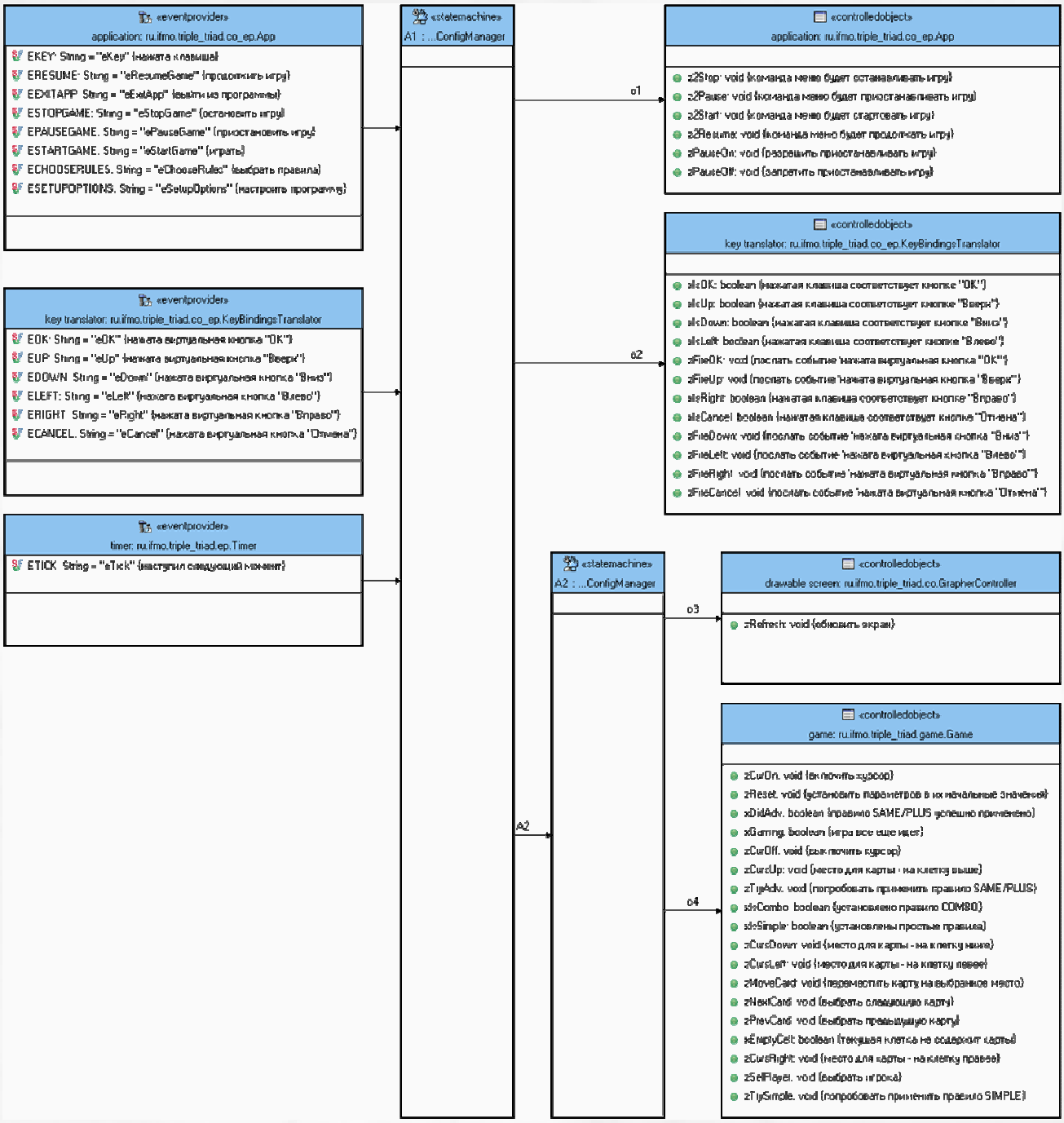

Рис.3. Диаграмма связей

Главный автомат *A1* отвечает за взаимодействие между пользователем и игрой, его состояния — это глобальные состояния игры и состояния интерфейса программы. Автомат *A2* отвечает за логику игры. Он запускается при достижении автоматом *A1* состояния «Идет игра».

У автомата *A1* есть два объекта управления, один из которых (объект класса `App`) также является и источником событий для этого автомата. События класса `App` — системные

события, такие как нажатия клавиш или выбор пунктов меню. *Выходные воздействия* (действия, которые могут быть выполнены классом при изменении состояния автомата) класса `App` служат для управления интерфейсом программы. Второй объект управления автомата *A1* (объект `KeyBindingsTranslator`) генерирует нажатия виртуальных клавиш, используемых автоматом *A2*.

У автомата *A2* два источника событий — объект `KeyBindingsTranslator` и объект `Timer`, посылающий единственное событие `eTick` каждые полсекунды. Объекты управления автомата *A2* — экран (объект `GrapherController`) с единственным выходным воздействием, обновляющим содержимое окна программы, и объект `Game`, содержащий всю информацию об игре. Процесс игры состоит в основном в управлении объектом `Game`, предоставляющим как входные воздействия, несущие информацию о правилах игры и стадиях игрового процесса, так и выходные воздействия, изменяющие эти стадии.

2.1. Интерфейс

На диаграмме состояний автомата *A1* (рис. 4) объект `App` фигурирует под именем `o1`, а объект `KeyBindingsTranslator` под именем `o2`.

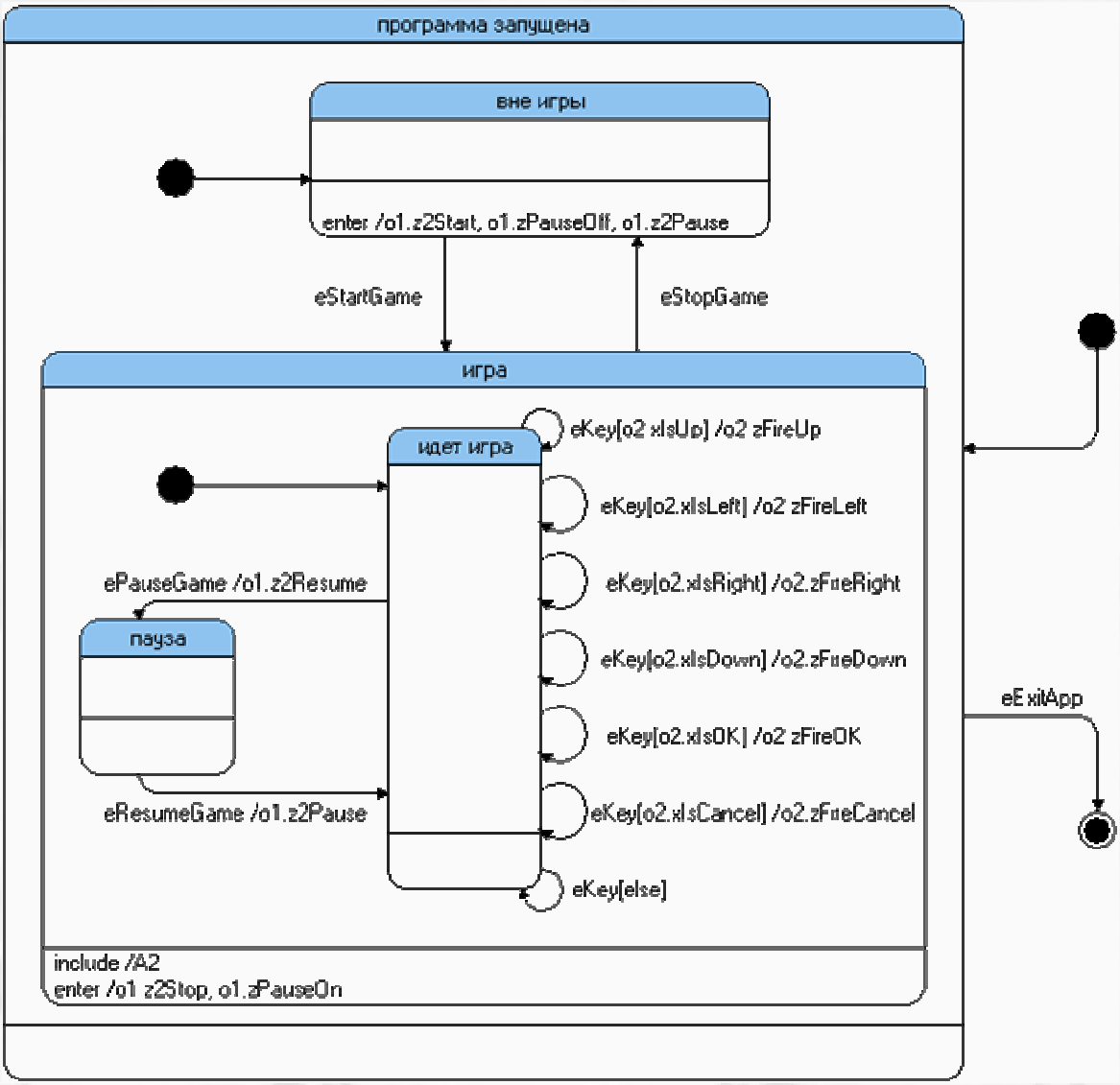

Рис.4. Граф переходов автомата *A1*

Граф переходов описывает состояния интерфейса программы. Его сложность связана с тем, что инструментальное средство *UniMod* не включает в себя редактор графического интерфейса пользователя, и привязка интерфейса к автомату происходит с некоторыми ухищрениями. Однако само по себе применение автоматного подхода в реализации пользовательского интерфейса [4] позволяет упростить многие аспекты разработки и реализации.

3.2. Логика

На диаграмме состояний автомата *A2* (рис.5) объект `GrapherController` фигурирует под именем `o3`, а объект `Game` — под именем `o4`.

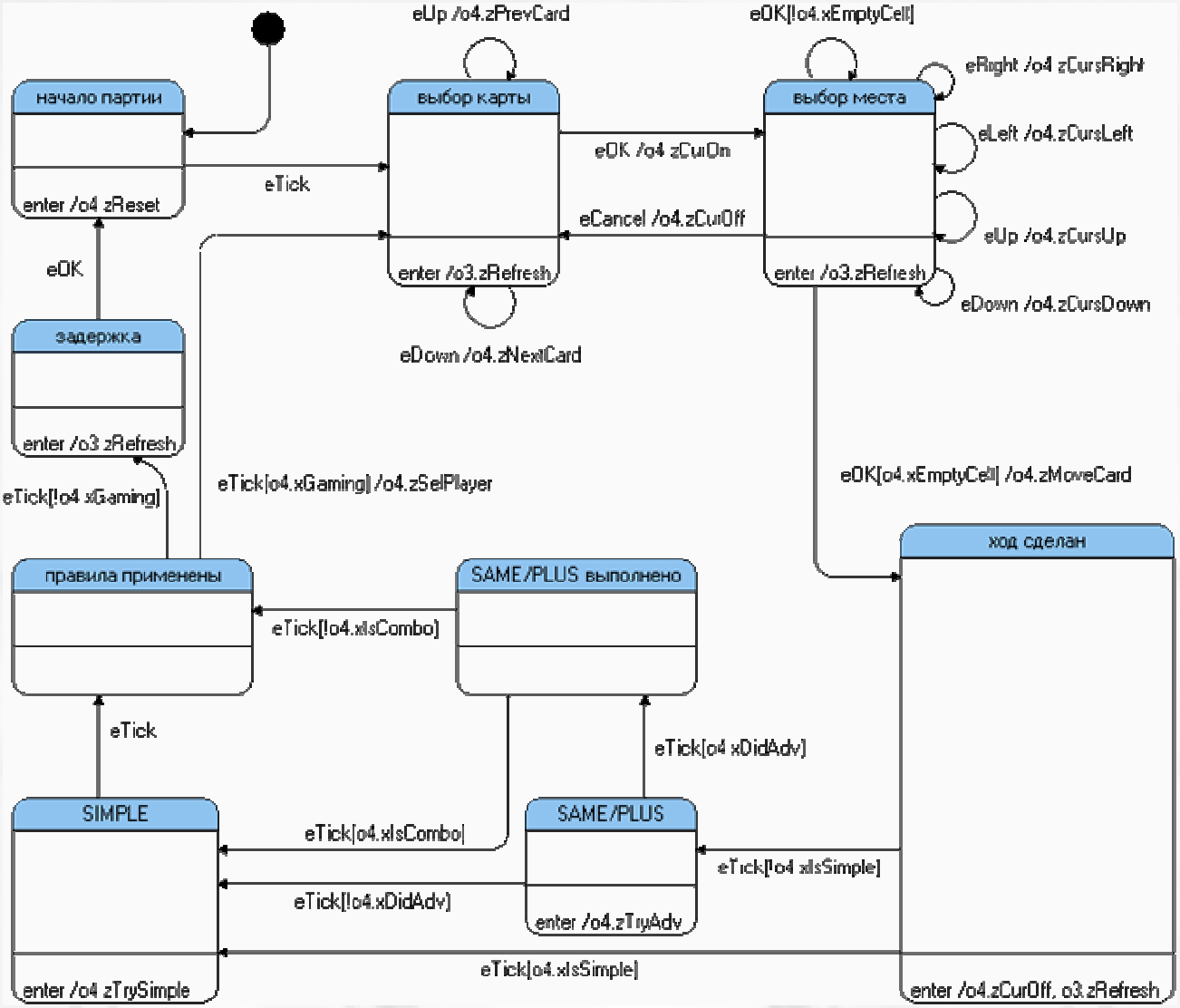

Рис.5. Граф переходов автомата *A2*

Этот граф переходов (и, соответственно, программа) описывает игровой процесс в упрощенном виде относительно правил, приведенных в Приложении 1 — без торговли и изменения правил, без поддержки правил *WALL*, *ELEMENTAL* и *SUDDEN DEATH*. Сложность графа переходов отражает исходную сложность правил игры, а сам граф делает их более наглядными.

3. Интерпретационный и компилятивный подходы

Существует два подхода к реализации кода автоматной программы — *интерпретационный* и *компилятивный*. Первый заключается в том, что логика поведения каждого автомата сохраняется в виде *XML*-описания, которое при выполнении программы интерпретируется специальным классом. При компилятивном же подходе логика поведения каждого автомата инкапсулируется в автоматически генерируемый *Java*-класс, и в процессе работы программы вместо интерпретации (как при интерпретационном подходе) происходит выполнение кода этого класса. Для данной программы выбор подхода не принципиален, тем более, последние версии *Unimod* автоматически создают все необходимые файлы для использования любого из указанных подходов на выбор.

Заключение

Автоматное программирование — удобная и мощная парадигма проектирования программ. Несмотря на некоторые недостатки существующих инструментальных средств, сам автоматный подход этих недостатков лишен и весьма перспективен.

Литература

1. *Сайт*, посвященный автоматному программированию. http://is.ifmo.ru/
2. *Ларман К.* Применение UML и шаблонов проектирования. М.: Вильямс, 2004.
3. *Шопырин Д.Г.* Объектно-ориентированная реализация конечных автоматов на основе виртуальных методов //Информационно-управляющие системы, 2005. №3, с. 36–40. http://is.ifmo.ru/works/runewstate/
4. *Корниенко А.А.* Автоматный подход к реализации графического пользовательского интерфейса. http://is.ifmo.ru/papers/kornienko/

Приложение 1. Полные правила игры

Основные правила игры

Игра *Triple Triad* присутствовала во многих играх серии *Final Fantasy*, но отличия между версиями состояли лишь в разных наборах карт и небольших вариациях правил. Полные правила игры описываются по версии игры *Triple Triad*, которая была приведена в игре *Final Fantasy VIII*.

В начале игры у каждого из игроков по пять карт. Каждая карта изображает какое-либо существо или персонажа из мира игры *Final Fantasy*, причем количество карт одного вида не ограничено. У всех карт в левом верхнем углу расположены четыре цифры, обозначающие «силу» карты по соответствующим четырем сторонам. Минимальная сила карты — “1”, максимальная — “A” (десять). У некоторых карт в правом верхнем углу расположен значок стихии, изменяющий при некоторых условиях силу карты.

Общую силу карты можно оценить по сумме сил карты по всем сторонам. Наиболее сильные карты могут обладать силой “A” сразу по двум сторонам (*Doomtrain*, *Rinoa*, *Edea*). При этом у наиболее слабых карт сила по каждой из сторон может быть меньше пяти (*Gayla*, *Caterchipillar*). Однако, в принципе, даже самая слабая карта может перевернуть сильную.

Игра ведется двумя игроками на поле 3x3. Игроки по очереди кладут карты на игровое поле, причем право первого хода случайным образом дается одному из игроков. Если положенная на текущем ходу карта соприкасается стороной с картой другого цвета, и сила положенной карты с этой стороны больше, то карта другого цвета «переворачивается», изменяя цвет и, соответственно, свою принадлежность. Выигрывает тот игрок, у которого в конце останется больше карт. После выигрыша проигравший игрок отдает часть своих карт победившему — происходит так называемая «торговля».

Дополнительные правила

В игре существует множество опциональных правил, дополняющих правила переворота карт, регулирующих правила торговли и бонусы к силе карт, или слегка изменяющих игровой процесс.

Правила переворота карт

- *SAME* — если положенная на этом ходу карта соприкасается с двумя или более картами другого цвета, причем на соприкасающихся сторонах силы равны, то положенная карта перевернет их.
- *PLUS* — если положенная на этом ходу карта соприкасается с двумя или более картами другого цвета, причем суммы сил на соприкасающихся сторонах одинаковы, то положенная карта перевернет их.
- *COMBO* — если карта, перевернутая на этом ходу по правилу *SAME* или *PLUS*, соприкасается с картой другого цвета, и сила перевернутой карты с этой стороны больше, то карта другого цвета переворачивается.
- *WALL* — при применении правил *SAME* или *PLUS* считается, что край игрового поля имеет силу “A” (максимальную). Таким образом, сторона карты, касающаяся края поля, как бы касается другой карты, имеющей с этой стороны силу “A”. Это расширяет количество случаев, при которых по правилам *SAME* и *PLUS* карты можно перевернуть.
- *ELEMENTAL* — при применении этого правила в некоторых клетках игрового поля расположены символы стихий (огня, льда, грома, земли, яда, ветра, воды или святости), но не больше одной стихии в клетке. Когда на клетку со стихией кладут карту той же стихии, сила карты увеличивается на единицу с каждой стороны. В противном случае сила карты с каждой стороны уменьшается на единицу. Эти изменения сил не учитываются при применении правил *SAME* или *PLUS*.

Правила торговли

- *ONE* — победитель забирает на выбор одну карту проигравшего.
- *DIRECT* — каждый забирает только карты своего цвета.
- *DIFF* — победитель забирает у проигравшего количество карт, равное разности в счете, или все карты, если разность в счете слишком велика.
- *ALL* — победитель забирает все карты проигравшего.

Другие правила

- *OPEN* — каждый видит карты, находящиеся на руках у противника.

- *RANDOM* — карты, участвующие в игре, выбираются случайным образом из общего запаса карт каждого из игроков.
- *SUDDEN DEATH* — если результатом игры будет ничья, то игра начинается заново, причем у игроков на руках будут те карты, которые были их цвета в конце игры.

Приложение 2. Исходные коды

Исходные коды состоят из файла *UniMod*-диаграмм `TripleTriad.unimod`, чье описание давалось выше, и *Java*-пакета `ru.ifmo.triple_triad`.

1. Файл co/GrapherController.java

```
package ru.ifmo.triple_triad.co;

import ru.ifmo.triple_triad.gui.Grapher;

import com.evelopers.unimod.runtime.ControlledObject;
import com.evelopers.unimod.runtime.context.StateMachineContext;

public class GrapherController implements ControlledObject {
    //output actions

    /**
     * @unimod.action.descr обновить экран
     */
    public void zRefresh(StateMachineContext context) {
        Grapher.getGrapher().repaint();
    }

}
```

2. Файл co_ep/App.java

```
package ru.ifmo.triple_triad.co_ep;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;

import javax.swing.JMenuItem;
import javax.swing.SwingUtilities;

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

import ru.ifmo.triple_triad.common.Messages;
import ru.ifmo.triple_triad.gui.AppFrame;

public class App implements ControlledObject, EventProvider {
//  events
    /**
     * @unimod.event.descr выбрать правила
     */
    public static final String ECHOOSERULES = "eChooseRules";

    /**
     * @unimod.event.descr выйти из программы
     */
    public static final String EEXITAPP = "eExitApp";

    /**
     * @unimod.event.descr настроить программу
     */
    public static final String ESETUPOPTIONS = "eSetupOptions";
    /**
     * @unimod.event.descr остановить игру
```

```
     */
    public static final String ESTOPGAME = "eStopGame";

    /**
     * @unimod.event.descr играть
     */
    public static final String ESTARTGAME = "eStartGame";

    /**
     * @unimod.event.descr приостановить игру
     */
    public static final String EPAUSEGAME = "ePauseGame";

    /**
     * @unimod.event.descr продолжить игру
     */
    public static final String ERESUME = "eResumeGame";

    /**
     * @unimod.event.descr нажата клавиша
     */
    public static final String EKEY = "eKey";

//  event provider implementations

    private AppFrame appFrame;
    private ModelEngine engine;

    private void initInterface() {
        appFrame.addKeyListener(new KeyTranslator());

        JMenuItem mi = appFrame.getExitMenuItem();
        mi.setActionCommand(EEXITAPP);
        mi.addActionListener(new ActionTranslator());

        mi = appFrame.getStartStopMenuItem();
        mi.setText(Messages.getString("Menu.startGame"));
        mi.setActionCommand(ESTARTGAME);
        mi.addActionListener(new ActionTranslator());

        mi = appFrame.getPauseResumeMenuItem();
        mi.setText(Messages.getString("Menu.pauseGame"));
        mi.setActionCommand(EPAUSEGAME);
        mi.addActionListener(new ActionTranslator());
        mi.setEnabled(false);

        appFrame.setVisible(true);
    }

    public void init(ModelEngine engine) throws CommonException {
        this.engine = engine;
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                appFrame = new AppFrame();
                initInterface();
            }
        });
    }

    public void dispose() {
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                appFrame.dispose();
            }
        });
    }

    public class ActionTranslator implements ActionListener {
        public void actionPerformed(ActionEvent ev) {
            if (engine == null)
                return;
            engine.getEventManager().handle(new Event(ev.getActionCommand()),
                                            StateMachineContextImpl.create());
        }
    }

    public class KeyTranslator implements KeyListener {
```

```
        public static final String KEY_EVENT = "Key Event";

        public void keyPressed(KeyEvent ev) {
            if (engine == null)
                return;
            EventManager em = engine.getEventManager();
            Event event = new Event(EKEY);
            StateMachineContext context = StateMachineContextImpl.create();
            context.getEventContext().setParameter(KEY_EVENT, ev);
            em.handle(event, context);
        }

        public void keyReleased(KeyEvent ev) {
        }

        public void keyTyped(KeyEvent ev) {
        }

    }

//  controlled object implementations

    /**
     * @unimod.action.descr команда меню будет стартовать игру
     */
    public void z2Start(StateMachineContext context) {
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                JMenuItem mi = appFrame.getStartStopMenuItem();
                mi.setText(Messages.getString("Menu.startGame"));
                mi.setActionCommand(ESTARTGAME);
            }
        });
    }

    /**
     * @unimod.action.descr команда меню будет приостанавливать игру
     */
    public void z2Pause(StateMachineContext context) {
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                JMenuItem mi = appFrame.getPauseResumeMenuItem();
                mi.setText(Messages.getString("Menu.pauseGame"));
                mi.setActionCommand(EPAUSEGAME);
            }
        });
    }

    /**
     * @unimod.action.descr команда меню будет продолжать игру
     */
    public void z2Resume(StateMachineContext context) {
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                JMenuItem mi = appFrame.getPauseResumeMenuItem();
                mi.setText(Messages.getString("Menu.resumeGame"));
                mi.setActionCommand(ERESUME);
            }
        });
    }

    /**
     * @unimod.action.descr команда меню будет останавливать игру
     */
    public void z2Stop(StateMachineContext context) {
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                JMenuItem mi = appFrame.getStartStopMenuItem();
                mi.setText(Messages.getString("Menu.stopGame"));
                mi.setActionCommand(ESTOPGAME);
            }
        });
    }

    /**
     * @unimod.action.descr разрешить приостанавливать игру
     */
    public void zPauseOn(StateMachineContext context) {
```

```
            SwingUtilities.invokeLater(new Runnable() {
                public void run() {
                    appFrame.getPauseResumeMenuItem().setEnabled(true);
                }
            });
        }

        /**
         * @unimod.action.descr запретить приостанавливать игру
         */
        public void zPauseOff(StateMachineContext context) {
            SwingUtilities.invokeLater(new Runnable() {
                public void run() {
                    appFrame.getPauseResumeMenuItem().setEnabled(false);
                }
            });
        }
    }
```

3. Файл co_ep/KeyBindingsTranslator.java

```
package ru.ifmo.triple_triad.co_ep;

import java.awt.event.KeyEvent;
import java.util.HashMap;


import com.evelopers.unimod.core.stateworks.Event;
import com.evelopers.unimod.runtime.ControlledObject;
import com.evelopers.unimod.runtime.EventProvider;
import com.evelopers.unimod.runtime.ModelEngine;
import com.evelopers.unimod.runtime.context.StateMachineContext;
import com.evelopers.unimod.runtime.context.StateMachineContextImpl;
import com.evelopers.unimod.runtime.context.StateMachineContext.Context;

public class KeyBindingsTranslator implements ControlledObject, EventProvider {
    private ModelEngine engine = null;
    private final HashMap mapping;

    public KeyBindingsTranslator() {
        mapping = new HashMap(6);
        mapping.put(EUP,     new Integer(KeyEvent.VK_UP));
        mapping.put(ELEFT,   new Integer(KeyEvent.VK_LEFT));
        mapping.put(ERIGHT,  new Integer(KeyEvent.VK_RIGHT));
        mapping.put(EDOWN,   new Integer(KeyEvent.VK_DOWN));
        mapping.put(EOK,     new Integer(KeyEvent.VK_A));
        mapping.put(ECANCEL, new Integer(KeyEvent.VK_Z));
    }

    public void init(ModelEngine e) {
        engine = e;
    }

    public void dispose() {
    }

    /**
     * проверяет, совпадает ли нажатая клавиша с клавишей, присвоенной
     * виртуальной кнопке key
     * @param map объект HashMap, хранящий соответствия клавиша-кнопка
     * @param key виртуальная кнопка
     * @param keyEvent объект, содержащий код нажатой клавиши
     * @return <code>true</code>, если в map есть пара (key, value),
     * где value - объект класса Integer, у которого
     * value.intValue() == keyEvent.getKeyCode()
     */
    private boolean consistWith(String key, KeyEvent keyEvent) {
        Integer value = (Integer) mapping.get(key);
        return value.intValue() == keyEvent.getKeyCode();
    }

    /**
     * посылает событие eventName без параметров, используя класс
     * @link com.evelopers.unimod.runtime.context.StateMachineContextImpl в
     * качестве реализации интерфейса @link StateMachineContext.
```

```
     * @param eventName событие
     */
    private void simpleFire(String eventName) {
        System.out.println(eventName + " fired");
        engine.getEventManager().handle(new Event(eventName),
                                        StateMachineContextImpl.create());
    }

    /**
     * возвращает последнее клавиатурное событие объекта @link AppConnector
     * @param context контекст автомата
     * @return клавиатурное событие
     */
    private KeyEvent getKeyEvent(StateMachineContext context) {
        Context ec = context.getEventContext();
        return (KeyEvent) ec.getParameter(App.KeyTranslator.KEY_EVENT);
    }

//output actions

    /**
     * @unimod.action.descr послать событие 'нажата виртуальная кнопка "Вверх"'
     */
    public void zFireUp(StateMachineContext context) {
        simpleFire(EUP);
    }

    /**
     * @unimod.action.descr послать событие 'нажата виртуальная кнопка "Влево"'
     */
    public void zFireLeft(StateMachineContext context) {
        simpleFire(ELEFT);
    }

    /**
     * @unimod.action.descr послать событие 'нажата виртуальная кнопка "Вправо"'
     */
    public void zFireRight(StateMachineContext context) {
        simpleFire(ERIGHT);
    }

    /**
     * @unimod.action.descr послать событие 'нажата виртуальная кнопка "Вниз"'
     */
    public void zFireDown(StateMachineContext context) {
        simpleFire(EDOWN);
    }

    /**
     * @unimod.action.descr послать событие 'нажата виртуальная кнопка "OK"'
     */
    public void zFireOK(StateMachineContext context) {
        simpleFire(EOK);
    }

    /**
     * @unimod.action.descr послать событие 'нажата виртуальная кнопка "Отмена"'
     */
    public void zFireCancel(StateMachineContext context) {
        simpleFire(ECANCEL);
    }

//input actions

    /**
     * @unimod.action.descr нажатая клавиша соответствует кнопке "Вверх"
     */
    public boolean xIsUp(StateMachineContext context) {
        return consistWith(EUP, getKeyEvent(context));
    }
    /**
     * @unimod.action.descr нажатая клавиша соответствует кнопке "Влево"
     */
    public boolean xIsLeft(StateMachineContext context) {
        return consistWith(ELEFT, getKeyEvent(context));
    }
    /**
     * @unimod.action.descr нажатая клавиша соответствует кнопке "Вправо"
```

```
     */
    public boolean xIsRight(StateMachineContext context) {
        return consistWith(ERIGHT, getKeyEvent(context));
    }
    /**
     * @unimod.action.descr нажатая клавиша соответствует кнопке "Вниз"
     */
    public boolean xIsDown(StateMachineContext context) {
        return consistWith(EDOWN, getKeyEvent(context));
    }
    /**
     * @unimod.action.descr нажатая клавиша соответствует кнопке "ОК"
     */
    public boolean xIsOK(StateMachineContext context) {
        return consistWith(EOK, getKeyEvent(context));
    }
    /**
     * @unimod.action.descr нажатая клавиша соответствует кнопке "Отмена"
     */
    public boolean xIsCancel(StateMachineContext context) {
        return consistWith(ECANCEL, getKeyEvent(context));
    }

//events

    /**
     * @unimod.event.descr нажата виртуальная кнопка "Вверх"
     */
    public static final String EUP = "eUp";
    /**
     * @unimod.event.descr нажата виртуальная кнопка "Влево"
     */
    public static final String ELEFT = "eLeft";
    /**
     * @unimod.event.descr нажата виртуальная кнопка "Вправо"
     */
    public static final String ERIGHT = "eRight";
    /**
     * @unimod.event.descr нажата виртуальная кнопка "Вниз"
     */
    public static final String EDOWN = "eDown";
    /**
     * @unimod.event.descr нажата виртуальная кнопка "ОК"
     */
    public static final String EOK = "eOK";
    /**
     * @unimod.event.descr нажата виртуальная кнопка "Отмена"
     */
    public static final String ECANCEL = "eCancel";
}
```

4. Файл common/ImageManager.java

```
package ru.ifmo.triple_triad.common;

import java.awt.Component;
import java.awt.Image;
import java.awt.MediaTracker;
import java.io.File;
import java.net.URL;
import java.net.URLClassLoader;
import java.util.Hashtable;

import javax.swing.ImageIcon;

import ru.ifmo.triple_triad.game.Game;
import ru.ifmo.triple_triad.game.GameCard;
import ru.ifmo.triple_triad.game.TTCard;
import ru.ifmo.triple_triad.gui.Grapher;

public class ImageManager extends Component {
    private static final String B_PREFIX = "(b)";
    private static final String R_PREFIX = "(r)";
    private Hashtable cache;
```

```
    private ImageManager() {
        cache = new Hashtable(TTCard.CARD_COUNT + 3);
        cache.put(Grapher.BG, getImageByName("_" + Grapher.BG));
        cache.put(GameCard.SHIRT, getImageByName("_" + GameCard.SHIRT));
        cache.put(Game.CURSOR, getImageByName("_" + Game.CURSOR));
        for (int l = 1; l <= TTCard.LEVELS_COUNT; l++)
            for (int i = 1; i <= TTCard.CARDS_PER_LEVEL; i++) {
                String name = TTCard.getCard(l, i).name;
                cache.put(B_PREFIX + name, getImageByName(name));
                cache.put(R_PREFIX + name, getImageByName("r" + name));
            }
    }

    private static ImageManager instance = null;

    public static ImageManager getInstance() {
        if (instance == null)
            instance = new ImageManager();
        return instance;
    }

    public ImageIcon getImage(String name) {
        if (cache != null)
            return (ImageIcon) cache.get(name);
        return null;
    }

    public ImageIcon getRImage(String name) {
        if (cache != null)
            return (ImageIcon) cache.get(R_PREFIX + name);
        return null;
    }

    public ImageIcon getBImage(String name) {
        if (cache != null)
            return (ImageIcon) cache.get(B_PREFIX + name);
        return null;
    }

    private String nameToFileName(String name) {
        return "images/" + name.replaceAll(" ", "").replaceAll(",", "")
               .toLowerCase() + ".jpg";
    }

    private ImageIcon getImageByName(String name) {
        String fileName = nameToFileName(name);
        URLClassLoader urlLoader = (URLClassLoader) ImageManager.class.getClassLoader();

        File f = new File(fileName);
        if (f.exists())
            System.out.println(f.getAbsolutePath());

        URL fileLoc = urlLoader.findResource(fileName);// */

        if (fileLoc == null)
            System.out.println("[" + fileName + "] not found");
        else {
            Image img = this.getToolkit().createImage(fileLoc);

            MediaTracker tracker = new MediaTracker(this);
            tracker.addImage(img, 0);
            try {
                tracker.waitForID(0);
                if (tracker.isErrorAny()) {
                    System.out.println("Error loading image " + fileName);
                }
            } catch (Exception ex) {
                System.out.println("Error" + ex.getClass().getName()
                        + " loading image " + fileName);
            }
            return new ImageIcon(img);
        }
        return null;
    }
}
```

5. Файл common/Messages.java

```
package ru.ifmo.triple_triad.common;

import java.util.HashMap;

public class Messages {

    private static HashMap strings;

    static {
        strings = new HashMap(4);

        strings.put("Menu.startGame" , "Начать игру");
        strings.put("Menu.pauseGame" , "Пауза");
        strings.put("Menu.resumeGame", "Продолжить");
        strings.put("Menu.stopGame"  , "Остановить игру");
    }

    private Messages() {
    }

    public static final String getString(String key) {
        String result = (String) strings.get(key);
        return result == null ? "[" + key + "]" : result;
    }
}
```

6. Файл ep/Timer.java

```
package ru.ifmo.triple_triad.ep;

import java.util.TimerTask;

import com.evelopers.unimod.core.stateworks.Event;
import com.evelopers.unimod.runtime.EventManager;
import com.evelopers.unimod.runtime.EventProvider;
import com.evelopers.unimod.runtime.ModelEngine;
import com.evelopers.unimod.runtime.context.StateMachineContextImpl;

public class Timer implements EventProvider {
    /**
     * @unimod.event.descr наступил следующий момент
     */
    public static final String ETICK = "eTick";

    private static final int INITIAL_DELAY = 100;
    private static final int DELAY = 500;
    private static final boolean IS_DAEMON = true;

    private static java.util.Timer t = new java.util.Timer(IS_DAEMON);
    private ModelEngine engine = null;

    public void init(ModelEngine engine) {
        this.engine = engine;

        t.schedule(new TimerTask() {
            public void run() {
                if (Timer.this.engine == null)
                    return;
                EventManager em = Timer.this.engine.getEventManager();
                em.handle(new Event(ETICK),
                          StateMachineContextImpl.create());
            }
        }, INITIAL_DELAY, DELAY);
    }

    public void dispose() {
        t.cancel();
    }
}
```

7. Файл game/Game.java

```
package ru.ifmo.triple_triad.game;

import java.awt.Component;
import java.awt.Graphics2D;
import java.util.ArrayList;
import java.util.Iterator;

import javax.swing.ImageIcon;

import ru.ifmo.triple_triad.common.ImageManager;

import com.evelopers.unimod.runtime.ControlledObject;
import com.evelopers.unimod.runtime.context.StateMachineContext;

public class Game implements ControlledObject {
    public final static String CURSOR = "Cursor";

    private Player redPlayer, bluePlayer;

    private ArrayList activeCards; //the card that has been put or
                                   //the cards that have been turned over
    private boolean isRedPlayer;
    private int xCurs, yCurs; //position of the cursor on the board (0..2)
    private int currCard; //current card in player's deck (0..4)

    private boolean advTrySucceeded;
    private int turns;
    private boolean showCursor;

    private GameCard[] bPlayer = new GameCard[5], //right column
                       rPlayer = new GameCard[5]; //left column
    private GameCard[][] table = new GameCard[3][3]; // board

    private ImageIcon cursor = ImageManager.getInstance().getImage(CURSOR);

    private static Game instance;
    public static Game getInstance() {
        return instance;
    }

    public final Rules rules = new Rules();

    public Game() {
        activeCards = new ArrayList();
        redPlayer = new Player("Красный");
        bluePlayer = new Player("Синий");

        if (instance == null)
            instance = this;
    }

//  ============================================================================
    private class XYPair {
        private final int x, y;
        private XYPair(int x, int y) {
            this.x = x;
            this.y = y;
        }
    }
//  ============================================================================
    public void draw(Component c, Graphics2D g) {
        for (int i = 0; i < 5; i++) {
            if (rPlayer[i] != null)
                rPlayer[i].draw(c, g);
            if (bPlayer[i] != null)
                bPlayer[i].draw(c, g);
        }

        for (int i = 0; i < 3; i++)
            for (int j = 0; j < 3; j++)
                if (table[i][j] != null)
                    table[i][j].draw(c, g);

        if (showCursor) {
            int x = GameCard.xDesk + GameCard.cw*xCurs;
```

```
            x += (int)Math.round((GameCard.cw - cursor.getIconWidth())*0.5);
            int y = GameCard.yDesk + GameCard.ch*yCurs;
            y += (int)Math.round((GameCard.ch - cursor.getIconHeight())*0.5);
            cursor.paintIcon(c, g, x, y);
        }
    }

    /**
     * @unimod.action.descr выключить курсор
     */
    public void zCurOff(StateMachineContext context) {
        showCursor = false;
    }
    /**
     * @unimod.action.descr включить курсор
     */
    public void zCurOn(StateMachineContext context) {
        showCursor = true;
    }

//  ============================================================================

    /**
     * @unimod.action.descr установлены простые правила
     */
    public boolean xIsSimple(StateMachineContext context) {
        return rules.isSimple();
    }

    /**
     * @unimod.action.descr установлено правило COMBO
     */
    public boolean xIsCombo(StateMachineContext context) {
        return rules.isCombo();
    }

    /**
     * @unimod.action.descr правило SAME/PLUS успешно применено
     */
    public boolean xDidAdv(StateMachineContext context) {
        return advTrySucceeded;
    }

    /**
     * @unimod.action.descr текущая клетка не содержит карты
     */
    public boolean xEmptyCell(StateMachineContext context) {
        return table[xCurs][yCurs] == null;
    }

    /**
     * @unimod.action.descr игра все еще идет
     */
    public boolean xGaming(StateMachineContext context) {
        return turns > 0;
    }

    /**
     * @unimod.action.descr выбрать предыдущую карту
     */
    public void zPrevCard(StateMachineContext context) {
        int newCurrCard = currCard - 1;
        while (newCurrCard >= 0)
            if (isRedPlayer)
                if (rPlayer[newCurrCard] != null) {
                    rPlayer[newCurrCard].setXY(GameCard.x0 + GameCard.dx,
                                               GameCard.y0 + GameCard.dy*newCurrCard);
                    rPlayer[currCard].setXY(GameCard.x0,
                                            GameCard.y0 + GameCard.dy*currCard);
                    currCard = newCurrCard;
                    return;
                } else
                    newCurrCard--;
            else
                if (bPlayer[newCurrCard] != null) {
                    bPlayer[newCurrCard].setXY(GameCard.x1 - GameCard.dx,
                                               GameCard.y0 + GameCard.dy*newCurrCard);
                    bPlayer[currCard].setXY(GameCard.x1,
```

```
                                        GameCard.y0 + GameCard.dy*currCard);
                currCard = newCurrCard;
                return;
            } else
                newCurrCard--;
    }

    /**
     * @unimod.action.descr выбрать следующую карту
     */
    public void zNextCard(StateMachineContext context) {
        int newCurrCard = currCard + 1;
        while (newCurrCard <= 4)
            if (isRedPlayer)
                if (rPlayer[newCurrCard] != null) {
                    rPlayer[newCurrCard].setXY(GameCard.x0 + GameCard.dx,
                                               GameCard.y0 + GameCard.dy*newCurrCard);
                    rPlayer[currCard].setXY(GameCard.x0,
                                            GameCard.y0 + GameCard.dy*currCard);
                    currCard = newCurrCard;
                    return;
                } else
                    newCurrCard++;
            else
                if (bPlayer[newCurrCard] != null) {
                    bPlayer[newCurrCard].setXY(GameCard.x1 - GameCard.dx,
                                               GameCard.y0 + GameCard.dy*newCurrCard);
                    bPlayer[currCard].setXY(GameCard.x1,
                                            GameCard.y0 + GameCard.dy*currCard);
                    currCard = newCurrCard;
                    return;
                } else
                    newCurrCard++;
    }

    /**
     * @unimod.action.descr место для карты - на клетку выше
     */
    public void zCursUp(StateMachineContext context) {
        if (yCurs > 0)
            yCurs--;
    }

    /**
     * @unimod.action.descr место для карты - на клетку ниже
     */
    public void zCursDown(StateMachineContext context) {
        if (yCurs < 2)
            yCurs++;
    }

    /**
     * @unimod.action.descr место для карты - на клетку левее
     */
    public void zCursLeft(StateMachineContext context) {
        if (xCurs > 0)
            xCurs--;
    }

    /**
     * @unimod.action.descr место для карты - на клетку правее
     */
    public void zCursRight(StateMachineContext context) {
        if (xCurs < 2)
            xCurs++;
    }

    /**
     * @unimod.action.descr переместить карту на выбранное место
     */
    public void zMoveCard(StateMachineContext context) {
        turns--;

        GameCard tmp;
        if (isRedPlayer) {
            tmp = rPlayer[currCard];
            rPlayer[currCard] = null;
        } else {
```

```
            tmp = bPlayer[currCard];
            bPlayer[currCard] = null;
        }
        activeCards.clear();
        activeCards.add(new XYPair(xCurs, yCurs));
        activeCards.trimToSize();
        table[xCurs][yCurs] = tmp;
        tmp.setXY(GameCard.xDesk + GameCard.cw*xCurs,
                  GameCard.yDesk + GameCard.ch*yCurs);

    }

    /**
     * @unimod.action.descr выбрать игрока
     */
    public void zSelPlayer(StateMachineContext context) {
        if (isRedPlayer) {
            if (rPlayer[currCard] != null)
                rPlayer[currCard].setXY(GameCard.x0,
                                        GameCard.y0 + GameCard.dy*currCard);
        } else {
            if (bPlayer[currCard] != null)
                bPlayer[currCard].setXY(GameCard.x1,
                                        GameCard.y0 + GameCard.dy*currCard);
        }

        isRedPlayer = !isRedPlayer;
        currCard = 0;

        if (isRedPlayer) {
            while (currCard < 4)
                if (rPlayer[currCard] != null)
                    break;
                else
                    currCard++;
            if (rPlayer[currCard] != null)
                rPlayer[currCard].setXY(GameCard.x0 + GameCard.dx,
                                        GameCard.y0 + GameCard.dy*currCard);
        } else {
            while (currCard < 4)
                if (bPlayer[currCard] != null)
                    break;
                else
                    currCard++;
            if (bPlayer[currCard] != null)
                bPlayer[currCard].setXY(GameCard.x1 - GameCard.dx,
                                        GameCard.y0 + GameCard.dy*currCard);
        }
    }

    /**
     * @unimod.action.descr попробовать применить правило SIMPLE
     */
    public void zTrySimple(StateMachineContext context) {
        Iterator it = activeCards.iterator();
        while(it.hasNext()) {
            XYPair pair = (XYPair) it.next();
            int x = pair.x;
            int y = pair.y;
            GameCard current = table[x][y];

            // попробуем перевернуть карту слева
            if (x > 0) {
                GameCard target = table[x - 1][y];
                if (target != null)
                    if ((target.isRed() != current.isRed()) &&
                        (current.getFace().left > target.getFace().right)) {
                        target.setRed(!target.isRed());
                    }
            }

            // попробуем перевернуть карту справа
            if (x < 2) {
                GameCard target = table[x + 1][y];
                if (target != null)
                    if ((target.isRed() != current.isRed()) &&
                        (current.getFace().right > target.getFace().left)) {
                        target.setRed(!target.isRed());
```

```
                    }
                }

                // попробуем перевернуть карту снизу
                if (y < 2) {
                    GameCard target = table[x][y + 1];
                    if (target != null)
                        if ((target.isRed() != current.isRed()) &&
                            (current.getFace().down > target.getFace().up)) {
                            target.setRed(!target.isRed());
                        }
                }

                // попробуем перевернуть карту сверху
                if (y > 0) {
                    GameCard target = table[x][y - 1];
                    if (target != null)
                        if ((target.isRed() != current.isRed()) &&
                            (current.getFace().up > target.getFace().down)) {
                            target.setRed(!target.isRed());
                        }
                }

                // перевернули все что могли
            }
        }

        /**
         * @unimod.action.descr попробовать применить правило SAME/PLUS
         */
        public void zTryAdv(StateMachineContext context) {
            advTrySucceeded = false;
            ArrayList turnedOver = new ArrayList();

            GameCard curr = table[xCurs][yCurs];
            if (rules.isPlusNotSame()) {
                // правило PLUS
                // попробуем перевернуть левую и верхнюю карты
                if (xCurs > 0 && yCurs > 0) {
                    GameCard tL = table[xCurs - 1][yCurs];
                    GameCard tU = table[xCurs][yCurs - 1];
                    if (tL != null && tU != null)
                        if ((tL.isRed() != curr.isRed()) && (tU.isRed() != curr.isRed()) &&
                            (curr.getFace().up + tU.getFace().down == curr.getFace().left +
tL.getFace().right)) {
                            advTrySucceeded = true;
                            turnedOver.add(new XYPair(xCurs - 1, yCurs));
                            turnedOver.add(new XYPair(xCurs, yCurs - 1));
                        }
                }

                // попробуем перевернуть левую и нижнюю карты
                if (xCurs > 0 && yCurs < 2) {
                    GameCard tL = table[xCurs - 1][yCurs];
                    GameCard tD = table[xCurs][yCurs + 1];
                    if (tL != null && tD != null)
                        if ((tL.isRed() != curr.isRed()) && (tD.isRed() != curr.isRed()) &&
                            (curr.getFace().down + tD.getFace().up == curr.getFace().left +
tL.getFace().right)) {
                            advTrySucceeded = true;
                            turnedOver.add(new XYPair(xCurs - 1, yCurs));
                            turnedOver.add(new XYPair(xCurs, yCurs + 1));
                        }
                }

                // попробуем перевернуть правую и верхнюю карты
                if (xCurs < 2 && yCurs > 0) {
                    GameCard tR = table[xCurs + 1][yCurs];
                    GameCard tU = table[xCurs][yCurs - 1];
                    if (tR != null && tU != null)
                        if ((tR.isRed() != curr.isRed()) && (tU.isRed() != curr.isRed()) &&
                            (curr.getFace().up + tU.getFace().down == curr.getFace().right +
tR.getFace().left)) {
                            advTrySucceeded = true;
                            turnedOver.add(new XYPair(xCurs + 1, yCurs));
                            turnedOver.add(new XYPair(xCurs, yCurs - 1));
                        }
                }
```

```
                    // попробуем перевернуть правую и нижнюю карты
                    if (xCurs < 2 && yCurs < 2) {
                        GameCard tR = table[xCurs + 1][yCurs];
                        GameCard tD = table[xCurs][yCurs + 1];
                        if (tR != null && tD != null)
                            if ((tR.isRed() != curr.isRed()) && (tD.isRed() != curr.isRed()) &&
                                (curr.getFace().down + tD.getFace().up == curr.getFace().right +
tR.getFace().left)) {
                                advTrySucceeded = true;
                                turnedOver.add(new XYPair(xCurs + 1, yCurs));
                                turnedOver.add(new XYPair(xCurs, yCurs + 1));
                        }
                    }

                    // перевернули все что могли
                } else {
                    // правило SAME
                    // попробуем перевернуть левую и верхнюю карты
                    if (xCurs > 0 && yCurs > 0) {
                        GameCard tL = table[xCurs - 1][yCurs];
                        GameCard tU = table[xCurs][yCurs - 1];
                        if (tL != null && tU != null)
                        if ((tL.isRed() != curr.isRed()) && (tU.isRed() != curr.isRed()) &&
                            (curr.getFace().up == tU.getFace().down) && (curr.getFace().left ==
tL.getFace().right)) {
                            advTrySucceeded = true;
                            turnedOver.add(new XYPair(xCurs - 1, yCurs));
                            turnedOver.add(new XYPair(xCurs, yCurs - 1));
                        }
                    }

                    // попробуем перевернуть левую и нижнюю карты
                    if (xCurs > 0 && yCurs < 2) {
                        GameCard tL = table[xCurs - 1][yCurs];
                        GameCard tD = table[xCurs][yCurs + 1];
                        if (tL != null && tD != null)
                            if ((tL.isRed() != curr.isRed()) && (tD.isRed() != curr.isRed()) &&
                                (curr.getFace().down == tD.getFace().up) && (curr.getFace().left ==
tL.getFace().right)) {
                                advTrySucceeded = true;
                                turnedOver.add(new XYPair(xCurs - 1, yCurs));
                                turnedOver.add(new XYPair(xCurs, yCurs + 1));
                            }
                    }

                    // попробуем перевернуть правую и верхнюю карты
                    if (xCurs < 2 && yCurs > 0) {
                        GameCard tR = table[xCurs + 1][yCurs];
                        GameCard tU = table[xCurs][yCurs - 1];
                        if (tR != null && tU != null)
                            if ((tR.isRed() != curr.isRed()) && (tU.isRed() != curr.isRed()) &&
                                (curr.getFace().up == tU.getFace().down) && (curr.getFace().right ==
tR.getFace().left)) {
                                advTrySucceeded = true;
                                turnedOver.add(new XYPair(xCurs + 1, yCurs));
                                turnedOver.add(new XYPair(xCurs, yCurs - 1));
                            }
                    }

                    // попробуем перевернуть правую и нижнюю карты
                    if (xCurs < 2 && yCurs < 2) {
                        GameCard tR = table[xCurs + 1][yCurs];
                        GameCard tD = table[xCurs][yCurs + 1];
                        if (tR != null && tD != null)
                            if ((tR.isRed() != curr.isRed()) && (tD.isRed() != curr.isRed()) &&
                                (curr.getFace().down == tD.getFace().up) && (curr.getFace().right ==
tR.getFace().left)) {
                                advTrySucceeded = true;
                                turnedOver.add(new XYPair(xCurs + 1, yCurs));
                                turnedOver.add(new XYPair(xCurs, yCurs + 1));
                        }
                    }

                    // перевернули все что могли
                }
                if (advTrySucceeded) {
                    activeCards = turnedOver;
```

```
            activeCards.trimToSize();
            Iterator it = activeCards.iterator();
            while(it.hasNext()) {
                XYPair pair = (XYPair) it.next();
                GameCard current = table[pair.x][pair.y];
                current.setRed(!current.isRed());
            }
            }
        }

        /**
         * @unimod.action.descr установить параметров в их начальные значения
         */
        public void zReset(StateMachineContext context) {
            advTrySucceeded = false;
            xCurs = 1; // курсор - в центр
            yCurs = 1; // курсор - в центр
            currCard = 0; // выбрана самая первая карта
            turns = 9; // 9 ходов

            showCursor = false;
            activeCards.clear();
            isRedPlayer = Math.random() < 0.5;

            for (int i = 0; i <= 2; i++)
                for (int j = 0; j <= 2; j++)
                    table[i][j] = null;

            TTCard[] cards = new TTCard[5];
            redPlayer.randomFill(cards);
            for (int i = 0; i < 5; i++)
                rPlayer[i] = new GameCard(cards[i], true, false,
                                          GameCard.x0, GameCard.y0 + GameCard.dy*i);

            bluePlayer.randomFill(cards);
            for (int i = 0; i < 5; i++)
                bPlayer[i] = new GameCard(cards[i], false, false,
                                          GameCard.x1, GameCard.y0 + GameCard.dy*i);

            if (isRedPlayer)
                rPlayer[0].setXY(GameCard.x0 + GameCard.dx, GameCard.y0);
            else
                bPlayer[0].setXY(GameCard.x1 - GameCard.dx, GameCard.y0);
        }
```

8. Файл game/GameCard.java

```
final package ru.ifmo.triple_triad.game;

import java.awt.Component;
import java.awt.Graphics2D;

import javax.swing.ImageIcon;

import ru.ifmo.triple_triad.common.ImageManager;


public class GameCard {
    public static final String SHIRT = "Shirt";

    public static final int sw = 576; // width of the screen
    public static final int sh = 432; // height of the screen
    public static final int cw = 100; // width of the card
    public static final int ch = 127; // height of the card
    public static final int pane = 144; // width of the pane with players' cards

    public static final int xDesk = 142;
    public static final int yDesk = 31;

    // coordinates of top left corner of the left column
    public static final int x0 = Math.round((pane - cw)/2.5f);
    public static final int y0 = x0;

    // coordinates of top left corner of the right column
    public static final int x1 = sw - cw - x0;
```

```
    public static final int y1 = y0;

    // horisontal gap between selected card and other cards
    public static final int dx = Math.round((pane - cw)/5f);

    // vertical gap between the closest cards
    public static final int dy = Math.round((sh - 2*y0 - ch)/4f);

    private TTCard card;
    private boolean red;
    private boolean hidden;
    private int x, y;
    private ImageIcon rImage, bImage, shirt;

    public GameCard(TTCard face,
                    boolean isRed, boolean isHidden,
                    int x0, int y0) {
        card = face;
        ImageManager im = ImageManager.getInstance();
        rImage = im.getRImage(card.name);
        bImage = im.getBImage(card.name);
        shirt = im.getImage(SHIRT);
        red = isRed;
        hidden = isHidden;
        x = x0;
        y = y0;
    }

    public void draw(Component c, Graphics2D g) {
        ImageIcon img;
        if (hidden)
            img = shirt;
        else
            if (red)
                img = rImage;
            else
                img = bImage;
        img.paintIcon(c, g, x, y);
    }

    public void setHidden(boolean hidden) {
        this.hidden = hidden;
    }

    public void setRed(boolean red) {
        this.red = red;
    }

    public boolean isRed() {
        return red;
    }

    public TTCard getFace() {
        return card;
    }

    public void setXY(int x, int y) {
        this.x = x;
        this.y = y;
    }

}
```

9. Файл game/Player.java

```
package ru.ifmo.triple_triad.game;

public class Player {
    private String name;

    public Player(String name) {
        this.name = name;
    }

    public String getName() {
```

```
            return name;
        }

        public void randomFill(TTCard[] cards) {
            for (int i = 0; i < 5; i++)
                cards[i] = TTCard.getCard((int)
Math.round(Math.floor(Math.random()*TTCard.LEVELS_COUNT)) + 1,
                                          (int)
Math.round(Math.floor(Math.random()*TTCard.CARDS_PER_LEVEL)) + 1);
        }

    }
```

10. Файл game/Rules.java

```
    package ru.ifmo.triple_triad.game;

    public class Rules {

        // trade rules
        public static final int ONE = 0;
        public static final int DIRECT = 1;
        public static final int DIFF = 2;
        public static final int ALL = 3;

        private int trade = ONE;

        // cards turning over rules
        private boolean simple = false;
        private boolean plusNotSame = false; // только при simple==false

        //additional rules
        private boolean combo = true; // только при simple==false
        private boolean wall = false;  // только при simple==false
        private boolean elemental = false;
        private boolean open = true;
        private boolean random = true;
        private boolean suddenDeath = false;

        public boolean isCombo() {
            return combo;
        }

        public void setCombo(boolean combo) {
            this.combo = combo;
        }

        public boolean isPlusNotSame() {
            return plusNotSame;
        }

        public void setPlusNotSame(boolean plusNotSame) {
            this.plusNotSame = plusNotSame;
        }

        public boolean isSimple() {
            return simple;
        }

        public void setSimple(boolean simple) {
            this.simple = simple;
        }

        public int getTrade() {
            return trade;
        }

        public void setTradeOne() {
            trade = ONE;
        }

        public void setTradeDirect() {
            trade = DIRECT;
        }
```

```
    public void setTradeDiff() {
        trade = DIFF;
    }

    public void setTradeAll() {
        trade = ALL;
    }

    public boolean isElemental() {
        return elemental; // TODO not implemented
    }

    public boolean isOpen() {
        return open; // TODO not implemented
    }

    public boolean isRandom() {
        return random; // TODO not implemented
    }

    public boolean isSuddenDeath() {
        return suddenDeath; // TODO not implemented
    }

    public boolean isWall() {
        return wall;
    }

}
```

11. Файл game/TTCard.java

```
package ru.ifmo.triple_triad.game;

public final class TTCard {
    public final static int LEVELS_COUNT = 3;
    public final static int CARDS_PER_LEVEL = 11;
    public final static int CARD_COUNT = LEVELS_COUNT*CARDS_PER_LEVEL;

    public static class Element {
        public final static Element ELEMENT_NONE    = new Element(0);
        public final static Element ELEMENT_FIRE    = new Element(1);
        public final static Element ELEMENT_ICE     = new Element(2);
        public final static Element ELEMENT_THUNDER = new Element(3);
        public final static Element ELEMENT_EARTH   = new Element(4);
        public final static Element ELEMENT_POISON  = new Element(5);
        public final static Element ELEMENT_WIND    = new Element(6);
        public final static Element ELEMENT_WATER   = new Element(7);
        public final static Element ELEMENT_HOLY    = new Element(8);

        private final int kind;

        public int getKind() {
            return kind;
        }

        private Element(int kind) {
            this.kind = kind;
        }

        public boolean equals(Object to) {
            if (to instanceof TTCard.Element)
                return ((TTCard.Element) to).kind == this.kind;
            return false;
        }
    }

    public final static int TYPE_MONSTER  = 0;
    public final static int TYPE_BOSS     = 1;
    public final static int TYPE_GUARDIAN = 2;
    public final static int TYPE_PLAYER   = 3;

    public final String name;
    public final int up, left, right, down;
    public final TTCard.Element element;
```

```
public final int type;

public int hashCode() {
    return ((((type*11 + up)*11 + left)*11 + right)*11 + down)*9 + element.kind;
}

private TTCard(String n,
                    int u,
               int l,      int r,
                    int d,
               TTCard.Element e, int t) {
    name = n;
    up = u;
    left = l;
    right = r;
    down = d;
    element = e;
    type = t;
}

private static final int A = 10;

public static TTCard getCard(int level, int number) {
    switch(level) {
        case 1:
            return l1[number - 1];
        case 2:
            return l2[number - 1];
        case 3:
            return l3[number - 1];
        case 4:
            return l4[number - 1];
        case 5:
            return l5[number - 1];
        case 6:
            return l6[number - 1];
        case 7:
            return l7[number - 1];
        case 8:
            return l8[number - 1];
        case 9:
            return l9[number - 1];
        case 10:
            return lA[number - 1];
    }
    return null;
}

public static TTCard getL1Card(int number) {
    return l1[number - 1];
}

public static TTCard getL2Card(int number) {
    return l2[number - 1];
}

public static TTCard getL3Card(int number) {
    return l3[number - 1];
}

public static TTCard getL4Card(int number) {
    return l4[number - 1];
}

public static TTCard getL5Card(int number) {
    return l5[number - 1];
}

public static TTCard getL6Card(int number) {
    return l6[number - 1];
}

public static TTCard getL7Card(int number) {
    return l7[number - 1];
}

public static TTCard getL8Card(int number) {
    return l8[number - 1];
```

```
}

public static TTCard getL9Card(int number) {
    return l9[number - 1];
}

public static TTCard getLACard(int number) {
    return lA[number - 1];
}

private static TTCard[] l1 = new TTCard[11];
static {
    l1[0] = new TTCard("Geezard",
                         1,
                       5,  4,
                         1,
                       Element.ELEMENT_NONE,
                       TYPE_MONSTER);
    l1[1] = new TTCard("Funguar",
                         5,
                       3,  1,
                         1,
                       Element.ELEMENT_NONE,
                       TYPE_MONSTER);
    l1[2] = new TTCard("Bite Bug",
                         1,
                       5,  3,
                         3,
                       Element.ELEMENT_NONE,
                       TYPE_MONSTER);
    l1[3] = new TTCard("Red Bat",
                         6,
                       2,  1,
                         1,
                       Element.ELEMENT_NONE,
                       TYPE_MONSTER);
    l1[4] = new TTCard("Blobra",
                         2,
                       5,  3,
                         1,
                       Element.ELEMENT_NONE,
                       TYPE_MONSTER);
    l1[5] = new TTCard("Gayla",
                         2,
                       4,  1,
                         4,
                       Element.ELEMENT_THUNDER,
                       TYPE_MONSTER);
    l1[6] = new TTCard("Gesper",
                         1,
                       1,  5,
                         4,
                       Element.ELEMENT_NONE,
                       TYPE_MONSTER);
    l1[7] = new TTCard("Fastitocalon-F",
                         3,
                       1,  5,
                         2,
                       Element.ELEMENT_EARTH,
                       TYPE_MONSTER);
    l1[8] = new TTCard("Blood Soul",
                         2,
                       1,  1,
                         6,
                       Element.ELEMENT_NONE,
                       TYPE_MONSTER);
    l1[9] = new TTCard("Caterchipillar",
                         4,
                       3,  2,
                         4,
                       Element.ELEMENT_NONE,
                       TYPE_MONSTER);
    l1[10] = new TTCard("Cockatrice",
                          2,
                        6,  1,
                          2,
                        Element.ELEMENT_THUNDER,
                        TYPE_MONSTER);
```

```
};

private static TTCard[] l2 = new TTCard[11];
static {
    l2[0] = new TTCard("Grat",
                         7,
                       1,  1,
                         3,
                       Element.ELEMENT_NONE,
                       TYPE_MONSTER);
    l2[1] = new TTCard("Buel",
                         6,
                       3,  2,
                         2,
                       Element.ELEMENT_NONE,
                       TYPE_MONSTER);
    l2[2] = new TTCard("Mesmerize",
                         5,
                       4,  3,
                         3,
                       Element.ELEMENT_NONE,
                       TYPE_MONSTER);
    l2[3] = new TTCard("Glacial Eye",
                         6,
                       3,  1,
                         4,
                       Element.ELEMENT_ICE,
                       TYPE_MONSTER);
    l2[4] = new TTCard("Belhelmel",
                         3,
                       3,  4,
                         5,
                       Element.ELEMENT_NONE,
                       TYPE_MONSTER);
    l2[5] = new TTCard("Thrustaevis",
                         5,
                       5,  3,
                         2,
                       Element.ELEMENT_WIND,
                       TYPE_MONSTER);
    l2[6] = new TTCard("Anacondaur",
                         5,
                       5,  1,
                         3,
                       Element.ELEMENT_POISON,
                       TYPE_MONSTER);
    l2[7] = new TTCard("Creeps",
                         5,
                       2,  2,
                         5,
                       Element.ELEMENT_THUNDER,
                       TYPE_MONSTER);
    l2[8] = new TTCard("Grendel",
                         4,
                       2,  4,
                         5,
                       Element.ELEMENT_THUNDER,
                       TYPE_MONSTER);
    l2[9] = new TTCard("Jell Eye",
                         3,
                       7,  2,
                         1,
                       Element.ELEMENT_NONE,
                       TYPE_MONSTER);
    l2[10] = new TTCard("Grand Mantis",
                          5,
                        3,  2,
                          5,
                        Element.ELEMENT_NONE,
                        TYPE_MONSTER);
};

private static TTCard[] l3 = new TTCard[11];
static {
    l3[0] = new TTCard("Forbidden",
                         6,
                       2,  6,
                         3,
```

```
                        Element.ELEMENT_NONE,
                        TYPE_MONSTER);
        l3[1] = new TTCard("Armadodo",
                          6,
                        6,  3,
                          1,
                        Element.ELEMENT_EARTH,
                        TYPE_MONSTER);
        l3[2] = new TTCard("Tri-Face",
                          3,
                        5,  5,
                          5,
                        Element.ELEMENT_POISON,
                        TYPE_MONSTER);
        l3[3] = new TTCard("Fastitocalon",
                          7,
                        3,  5,
                          1,
                        Element.ELEMENT_EARTH,
                        TYPE_MONSTER);
        l3[4] = new TTCard("Snow Lion",
                          7,
                        3,  1,
                          5,
                        Element.ELEMENT_ICE,
                        TYPE_MONSTER);
        l3[5] = new TTCard("Ochu",
                          5,
                        3,  6,
                          3,
                        Element.ELEMENT_NONE,
                        TYPE_MONSTER);
        l3[6] = new TTCard("SAM08G",
                          5,
                        4,  6,
                          2,
                        Element.ELEMENT_FIRE,
                        TYPE_MONSTER);
        l3[7] = new TTCard("Death Claw",
                          4,
                        2,  4,
                          7,
                        Element.ELEMENT_FIRE,
                        TYPE_MONSTER);
        l3[8] = new TTCard("Cactuar",
                          6,
                        3,  2,
                          6,
                        Element.ELEMENT_NONE,
                        TYPE_MONSTER);
        l3[9] = new TTCard("Tonberry",
                          3,
                        4,  6,
                          4,
                        Element.ELEMENT_NONE,
                        TYPE_MONSTER);
        l3[10] = new TTCard("Abyss Worm",
                           7,
                         5,  2,
                           3,
                         Element.ELEMENT_EARTH,
                         TYPE_MONSTER);
    };

    private static TTCard[] l4 = new TTCard[11];
    static {
        l4[0] = new TTCard("Turtapod",
                          2,
                        7,  3,
                          6,
                        Element.ELEMENT_NONE,
                        TYPE_MONSTER);
        l4[1] = new TTCard("Vysage",
                          6,
                        5,  5,
                          4,
                        Element.ELEMENT_NONE,
                        TYPE_MONSTER);
```

```
        l4[2] = new TTCard("T-Rexaur",
                             4,
                           7,  6,
                             2,
                           Element.ELEMENT_NONE,
                           TYPE_MONSTER);
        l4[3] = new TTCard("Bomb",
                             2,
                           3,  7,
                             6,
                           Element.ELEMENT_FIRE,
                           TYPE_MONSTER);
        l4[4] = new TTCard("Blitz",
                             1,
                           7,  6,
                             4,
                           Element.ELEMENT_THUNDER,
                           TYPE_MONSTER);
        l4[5] = new TTCard("Wendigo",
                             7,
                           6,  3,
                             1,
                           Element.ELEMENT_NONE,
                           TYPE_MONSTER);
        l4[6] = new TTCard("Torama",
                             7,
                           4,  4,
                             4,
                           Element.ELEMENT_NONE,
                           TYPE_MONSTER);
        l4[7] = new TTCard("Imp",
                             3,
                           6,  7,
                             3,
                           Element.ELEMENT_NONE,
                           TYPE_MONSTER);
        l4[8] = new TTCard("Blue Dragon",
                             6,
                           3,  2,
                             7,
                           Element.ELEMENT_POISON,
                           TYPE_MONSTER);
        l4[9] = new TTCard("Adamantoise",
                             4,
                           6,  5,
                             5,
                           Element.ELEMENT_EARTH,
                           TYPE_MONSTER);
        l4[10] = new TTCard("Hexadragon",
                              7,
                            3,  5,
                              4,
                            Element.ELEMENT_FIRE,
                            TYPE_MONSTER);
    };

    private static TTCard[] l5 = new TTCard[11];
    static {
        l5[0] = new TTCard("Iron Giant",
                             6,
                           5,  5,
                             6,
                           Element.ELEMENT_NONE,
                           TYPE_MONSTER);
        l5[1] = new TTCard("Behemoth",
                             3,
                           7,  6,
                             5,
                           Element.ELEMENT_NONE,
                           TYPE_MONSTER);
        l5[2] = new TTCard("Chimera",
                             7,
                           3,  6,
                             5,
                           Element.ELEMENT_WATER,
                           TYPE_MONSTER);
        l5[3] = new TTCard("Pupu",
                             3,
```

```
                        1,  A,
                          2,
                        Element.ELEMENT_NONE,
                        TYPE_MONSTER);
        l5[4] = new TTCard("Elastoid",
                          6,
                        7,  2,
                          6,
                        Element.ELEMENT_NONE,
                        TYPE_MONSTER);
        l5[5] = new TTCard("GIM47N",
                          5,
                        4,  5,
                          7,
                        Element.ELEMENT_NONE,
                        TYPE_MONSTER);
        l5[6] = new TTCard("Malboro",
                          7,
                        2,  7,
                          4,
                        Element.ELEMENT_POISON,
                        TYPE_MONSTER);
        l5[7] = new TTCard("Ruby Dragon",
                          7,
                        4,  2,
                          7,
                        Element.ELEMENT_FIRE,
                        TYPE_MONSTER);
        l5[8] = new TTCard("Elnoyle",
                          5,
                        6,  3,
                          7,
                        Element.ELEMENT_NONE,
                        TYPE_MONSTER);
        l5[9] = new TTCard("Tonberry King",
                          4,
                        4,  6,
                          7,
                        Element.ELEMENT_NONE,
                        TYPE_MONSTER);
        l5[10] = new TTCard("Wedge, Biggs",
                           6,
                         7,  6,
                           2,
                         Element.ELEMENT_NONE,
                         TYPE_MONSTER);
    };

    private static TTCard[] l6 = new TTCard[11];
    static {
        l6[0] = new TTCard("Fujin, Raijin",
                          2,
                        4,  8,
                          8,
                        Element.ELEMENT_NONE,
                        TYPE_BOSS);
        l6[1] = new TTCard("Elvoret",
                          7,
                        4,  8,
                          3,
                        Element.ELEMENT_WIND,
                        TYPE_BOSS);
        l6[2] = new TTCard("X-ATM092",
                          4,
                        3,  8,
                          7,
                        Element.ELEMENT_NONE,
                        TYPE_BOSS);
        l6[3] = new TTCard("Granaldo",
                          7,
                        5,  2,
                          8,
                        Element.ELEMENT_NONE,
                        TYPE_BOSS);
        l6[4] = new TTCard("Gerogero",
                          1,
                        3,  8,
                          8,
```

```
                        Element.ELEMENT_POISON,
                        TYPE_BOSS);
        l6[5] = new TTCard("Iguion",
                          8,
                        2,  2,
                          8,
                        Element.ELEMENT_NONE,
                        TYPE_BOSS);
        l6[6] = new TTCard("Abadon",
                          6,
                        5,  8,
                          4,
                        Element.ELEMENT_NONE,
                        TYPE_BOSS);
        l6[7] = new TTCard("Trauma",
                          4,
                        6,  8,
                          5,
                        Element.ELEMENT_NONE,
                        TYPE_BOSS);
        l6[8] = new TTCard("Oilboyle",
                          1,
                        8,  8,
                          4,
                        Element.ELEMENT_NONE,
                        TYPE_BOSS);
        l6[9] = new TTCard("Shumi Tribe",
                          6,
                        4,  5,
                          8,
                        Element.ELEMENT_NONE,
                        TYPE_BOSS);
        l6[10] = new TTCard("Krysta",
                           7,
                         1,  5,
                           8,
                         Element.ELEMENT_NONE,
                         TYPE_BOSS);
    };

    private static TTCard[] l7 = new TTCard[11];
    static {
        l7[0] = new TTCard("Propogator",
                          8,
                        8,  4,
                          4,
                        Element.ELEMENT_NONE,
                        TYPE_BOSS);
        l7[1] = new TTCard("Jumbo Cactuar",
                          8,
                        4,  8,
                          4,
                        Element.ELEMENT_NONE,
                        TYPE_BOSS);
        l7[2] = new TTCard("Tri-Point",
                          8,
                        8,  5,
                          2,
                        Element.ELEMENT_NONE,
                        TYPE_BOSS);
        l7[3] = new TTCard("Gargantua",
                          5,
                        8,  6,
                          6,
                        Element.ELEMENT_NONE,
                        TYPE_BOSS);
        l7[4] = new TTCard("Mobile Type 8",
                          8,
                        3,  6,
                          7,
                        Element.ELEMENT_NONE,
                        TYPE_BOSS);
        l7[5] = new TTCard("Sphinxara",
                          8,
                        8,  3,
                          5,
                        Element.ELEMENT_NONE,
                        TYPE_BOSS);
```

```
        l7[6] = new TTCard("Tiamat",
                             8,
                           4,  8,
                             5,
                           Element.ELEMENT_NONE,
                           TYPE_BOSS);
        l7[7] = new TTCard("BGH251F2",
                             5,
                           5,  7,
                             8,
                           Element.ELEMENT_NONE,
                           TYPE_BOSS);
        l7[8] = new TTCard("Red Giant",
                             6,
                           7,  8,
                             4,
                           Element.ELEMENT_NONE,
                           TYPE_BOSS);
        l7[9] = new TTCard("Catoblepas",
                             1,
                           7,  8,
                             7,
                           Element.ELEMENT_NONE,
                           TYPE_BOSS);
        l7[10] = new TTCard("Ultima Weapon",
                              7,
                            8,  7,
                              2,
                            Element.ELEMENT_NONE,
                            TYPE_BOSS);
    };

    private static TTCard[] l8 = new TTCard[11];
    static {
        l8[0] = new TTCard("Chubby Chocobo",
                             4,
                           9,  4,
                             8,
                           Element.ELEMENT_NONE,
                           TYPE_GUARDIAN);
        l8[1] = new TTCard("Angelo",
                             9,
                           3,  6,
                             7,
                           Element.ELEMENT_NONE,
                           TYPE_GUARDIAN);
        l8[2] = new TTCard("Gilgamesh",
                             3,
                           6,  7,
                             9,
                           Element.ELEMENT_NONE,
                           TYPE_GUARDIAN);
        l8[3] = new TTCard("Mini Mog",
                             9,
                           2,  3,
                             9,
                           Element.ELEMENT_NONE,
                           TYPE_GUARDIAN);
        l8[4] = new TTCard("Chicobo",
                             9,
                           4,  4,
                             8,
                           Element.ELEMENT_NONE,
                           TYPE_GUARDIAN);
        l8[5] = new TTCard("Quezacotl",
                             2,
                           4,  9,
                             9,
                           Element.ELEMENT_THUNDER,
                           TYPE_GUARDIAN);
        l8[6] = new TTCard("Shiva",
                             6,
                           9,  7,
                             4,
                           Element.ELEMENT_ICE,
                           TYPE_GUARDIAN);
        l8[7] = new TTCard("Ifrit",
                             9,
```

```
                        8,  6,
                          2,
                        Element.ELEMENT_FIRE,
                        TYPE_GUARDIAN);
    l8[8] = new TTCard("Siren",
                          8,
                        2,  9,
                          6,
                        Element.ELEMENT_NONE,
                        TYPE_GUARDIAN);
    l8[9] = new TTCard("Sacred",
                          5,
                        9,  1,
                          9,
                        Element.ELEMENT_EARTH,
                        TYPE_GUARDIAN);
    l8[10] = new TTCard("Minotaur",
                           9,
                         9,  5,
                           2,
                         Element.ELEMENT_EARTH,
                         TYPE_GUARDIAN);
};

private static TTCard[] l9 = new TTCard[11];
static {
    l9[0] = new TTCard("Carbuncle",
                          8,
                        4,  4,
                          A,
                        Element.ELEMENT_NONE,
                        TYPE_GUARDIAN);
    l9[1] = new TTCard("Diablos",
                          5,
                        3,  A,
                          8,
                        Element.ELEMENT_NONE,
                        TYPE_GUARDIAN);
    l9[2] = new TTCard("Leviathan",
                          7,
                        7,  A,
                          1,
                        Element.ELEMENT_WATER,
                        TYPE_GUARDIAN);
    l9[3] = new TTCard("Odin",
                          8,
                        5,  A,
                          3,
                        Element.ELEMENT_NONE,
                        TYPE_GUARDIAN);
    l9[4] = new TTCard("Pandemona",
                          A,
                        7,  1,
                          7,
                        Element.ELEMENT_WIND,
                        TYPE_GUARDIAN);
    l9[5] = new TTCard("Cerberus",
                          7,
                        A,  4,
                          6,
                        Element.ELEMENT_NONE,
                        TYPE_GUARDIAN);
    l9[6] = new TTCard("Alexander",
                          9,
                        2,  A,
                          4,
                        Element.ELEMENT_HOLY,
                        TYPE_GUARDIAN);
    l9[7] = new TTCard("Phoenix",
                          7,
                        A,  2,
                          7,
                        Element.ELEMENT_FIRE,
                        TYPE_GUARDIAN);
    l9[8] = new TTCard("Bahamut",
                          A,
                        6,  8,
                          2,
```

```
                        Element.ELEMENT_NONE,
                        TYPE_GUARDIAN);
        l9[9] = new TTCard("Doomtrain",
                          3,
                        A,  1,
                          A,
                        Element.ELEMENT_POISON,
                        TYPE_GUARDIAN);
        l9[10] = new TTCard("Eden",
                           4,
                         A,  4,
                           9,
                         Element.ELEMENT_NONE,
                         TYPE_GUARDIAN);
    };

    private static TTCard[] lA = new TTCard[11];
    static {
        lA[0] = new TTCard("Ward",
                           A,
                        8,  7,
                          2,
                        Element.ELEMENT_NONE,
                        TYPE_PLAYER);
        lA[1] = new TTCard("Kiros",
                          6,
                        A,  7,
                          6,
                        Element.ELEMENT_NONE,
                        TYPE_PLAYER);
        lA[2] = new TTCard("Laguna",
                          5,
                        9,  A,
                          3,
                        Element.ELEMENT_NONE,
                        TYPE_PLAYER);
        lA[3] = new TTCard("Selphie",
                          A,
                        4,  8,
                          6,
                        Element.ELEMENT_NONE,
                        TYPE_PLAYER);
        lA[4] = new TTCard("Quistis",
                          9,
                        2,  6,
                          A,
                        Element.ELEMENT_NONE,
                        TYPE_PLAYER);
        lA[5] = new TTCard("Irvine",
                          2,
                        A,  6,
                          9,
                        Element.ELEMENT_NONE,
                        TYPE_PLAYER);
        lA[6] = new TTCard("Zell",
                          8,
                        6,  5,
                          A,
                        Element.ELEMENT_NONE,
                        TYPE_PLAYER);
        lA[7] = new TTCard("Rinoa",
                          4,
                        A,  A,
                          2,
                        Element.ELEMENT_NONE,
                        TYPE_PLAYER);
        lA[8] = new TTCard("Edea",
                          A,
                        3,  A,
                          3,
                        Element.ELEMENT_NONE,
                        TYPE_PLAYER);
        lA[9] = new TTCard("Seifer",
                          6,
                        4,  9,
                          A,
                        Element.ELEMENT_NONE,
                        TYPE_PLAYER);
```

```
            lA[10] = new TTCard("Squall",
                                A,
                              9,  4,
                                6,
                              Element.ELEMENT_NONE,
                              TYPE_PLAYER);
        };
    }
```

12. Файл gui/AppFrame.java

```
package ru.ifmo.triple_triad.gui;

import java.awt.BorderLayout;
import java.awt.Dimension;

import javax.swing.JFrame;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.UIManager;
import javax.swing.WindowConstants;

 public class AppFrame extends JFrame {
    static {
        try {
            UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
        } catch (Exception e) {}
    }

    private Grapher jContentPane = null;
    private JMenuBar mainMenuBar = null;
    private JMenu programMenu = null;
    private JMenu gameMenu = null;
    private JMenuItem exitMenuItem = null;
    private JMenuItem startStopMenuItem = null;
    private JMenuItem pauseResumeMenuItem = null;

    public AppFrame() {
        super();
        initialize();
    }

    /**
     * This method initializes this
     * @return void
     */
    protected void initialize() {
        this.setContentPane(getJContentPane());
        this.setJMenuBar(getMainMenuBar());
        this.setTitle("Triple Triad");
        this.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
        this.setResizable(false);
        this.setSize(new java.awt.Dimension(582, 478));
    }

    /**
     * This method initializes jContentPane
     * @return javax.swing.JPanel
     */
    protected Grapher getJContentPane() {
        if (jContentPane == null) {
            jContentPane = new Grapher();
            jContentPane.setLayout(new BorderLayout());
            jContentPane.setPreferredSize(new Dimension(576, 432));
        }
        return jContentPane;
    }

    /**
     * This method initializes menuBar
     * @return javax.swing.JMenuBar
     */
    protected JMenuBar getMainMenuBar() {
        if (mainMenuBar == null) {
```

```
            mainMenuBar = new JMenuBar();
            mainMenuBar.add(getProgramMenu());
            mainMenuBar.add(getGameMenu());
        }
        return mainMenuBar;
    }

    /**
     * This method initializes programMenu
     * @return javax.swing.JMenu
     */
    protected JMenu getProgramMenu() {
        if (programMenu == null) {
            programMenu = new JMenu();
            programMenu.setText("Программа");
            programMenu.add(getExitMenuItem());
        }
        return programMenu;
    }

    /**
     * This method initializes exitMenuItem
     * @return javax.swing.JMenuItem
     */
    public JMenuItem getExitMenuItem() {
        if (exitMenuItem == null) {
            exitMenuItem = new JMenuItem();
            exitMenuItem.setText("Выход");
        }
        return exitMenuItem;
    }

    /**
     * This method initializes gameMenu
     * @return javax.swing.JMenu
     */
    protected JMenu getGameMenu() {
        if (gameMenu == null) {
            gameMenu = new JMenu();
            gameMenu.setText("Игра");
            gameMenu.add(getStartStopMenuItem());
            gameMenu.add(getPauseResumeMenuItem());
        }
        return gameMenu;
    }

    /**
     * This method initializes startStopMenuItem
     * @return javax.swing.JMenuItem
     */
    public JMenuItem getStartStopMenuItem() {
        if (startStopMenuItem == null) {
            startStopMenuItem = new JMenuItem();
        }
        return startStopMenuItem;
    }

    /**
     * This method initializes pauseResumeMenuItem
     *
     * @return javax.swing.JMenuItem
     */
    public JMenuItem getPauseResumeMenuItem() {
        if (pauseResumeMenuItem == null) {
            pauseResumeMenuItem = new JMenuItem();
        }
        return pauseResumeMenuItem;
    }
}
```

13. Файл gui/Grapher.java

```
package ru.ifmo.triple_triad.gui;

import java.awt.Graphics;
```

```java
import java.awt.Graphics2D;

import javax.swing.ImageIcon;
import javax.swing.JPanel;

import ru.ifmo.triple_triad.common.ImageManager;
import ru.ifmo.triple_triad.game.Game;

public class Grapher extends JPanel {

    public static final String BG = "Background";

    private ImageIcon back = ImageManager.getInstance().getImage(BG);

    private static Grapher grapher = null;

    public Grapher() {
        super(true);
        setDoubleBuffered(true);
        if (grapher == null)
            grapher = this;
    }

    public static Grapher getGrapher() {
        return grapher;
    }

    public void paintComponent(Graphics g) {
        //when doubleBuffered is set to true, g is Graphics2D
        back.paintIcon(this, g, 0, 0);

        Game game = Game.getInstance();
        if (game != null)
            game.draw(this, (Graphics2D) g);
    }

}
```