Статья опубликована в журнале "Компьютерные инструменты в образовании". 2005. № 6, 72-82.

Петрошенко Павел Александрович,
Корнеев Георгий Александрович,
Шалыто Анатолий Абрамович

РЕАЛИЗАЦИЯ ИГРЫ “МОРСКОЙ БОЙ” НА ОСНОВЕ АВТОМАТНОГО ПОДХОДА

“*Морской бой* – вооруженное противоборство соединений (групп), частей и кораблей с противником…”[1]

Введение

В настоящее время развивается автоматный подход для написания программ, который называется также *автоматное программирование* [1]. Этот подход может быть эффективно использован при построении различных классов программ, например, визуализаторов алгоритмов [2], программ обучения [3] и игр [4].

Однако в этих работах при описании поведения каждого из продуктов применялся только один автомат. Цель настоящей работы на примере проктирования известной игры *Морской Бой* показать, как использовать систему взаимодействующих конечных автоматов и описывать структуру классов в объектно-ориентированных программах, как это было предложено в работе [5].

1. Реализация

В программе автоматы реализованы в виде методов классов. Всего классов в программе более сорока, из них пять автоматных, каждый из которых содержит по одному автомату. Четыре автомата являются вложенными – их вызов осуществляется из состояний других автоматов. Автоматы взаимодействуют между собой по вложенности и с помощью обмена номерами состояний, а с внешней средой – при обработке событий, которых всего два: “Нажатие кнопки мыши” (*MouseEvent*) и “Нажатие кнопки “Начать заново”. Источниками событий (*EventProvider*), таким образом, являются, обработчик событий мыши на игровом поле и соответствующая кнопка (*ButtonEventProvider*).

Для обработки событий используется очередь необработанных событий (*EventStorage*). Каждый автомат, находясь в состоянии, в котором ожидается соответствующее событие, обращается к очереди необработанных событий, и, если в ней есть событие нужного типа, обрабатывает его, удаляя из очереди. Если такого события в очереди нет, то автомат ждет до тех пор, пока оно не произойдет и не поступит в очередь для обработки. Все автоматы используют единую очередь необработанных событий.

Для реализации данного проекта использован объектно-ориентированный язык *Java* и среда разработки *IntelliJ IDEA 4*.

При проектировании программы в части пользовательского интерфейса применялась стандартная архитектура “Модель – Вид – Контроллер” (*Model – View – Controller*). Одно из применений данной схемы – реализация игрового поля. Класс, представляющий модель поля, реализует внутреннюю структуру: физические характеристики и состояния ячеек

[1] Поленин В.И. Морской бой, применение сил в морском бою или тактическая операция? // Военная мысль. 2003, № 10.

игрового поля. В свою очередь, класс “Представление Игрового Поля” (*Field*) для этой модели определяет и реализует все функциональные особенности и детали визуализации. Изменения в текущие состояния полей вносят автоматные классы.

Схема “Модель – Вид – Контроллер” использована также и в механизме протоколирования состояний автоматов. Класс, реализующий модель системы протоколирования (*LoggingModel*), определяет ее функциональность и структуру. Средства отображения протоколов представлены классами, реализующими специальный интерфейс – представление протоколов (*LogView*).

В программе использованы следующие паттерны объектно-ориентированного проектирования: Одиночка (*Singleton*) и Наблюдатель (*Observer*).

При отладке программы применялись протоколы, являющиеся неотъемлемой частью использованной технологии. Отдельно поддерживается включение в протокол информации об объектах, автоматах, входных переменных и выходных воздействиях. При протоколировании входных переменных и выходных воздействий в протокол также добавляется информация об обрабатываемых автоматами событиях. Строки, содержащие информацию, относящеюся к событию, начинаются со знака“#”.

Независимо ведется протоколирование основных этапов работы программы для логического разбиения всего протокола на отдельные части, такие, как “Расстановка кораблей Пользователя”, “Расстановка кораблей Робота”, “Сражение”. Протоколирование осуществляется при изменении состояния хотя бы одного автомата. При наличии полного протокола можно восстановить ход работы программы. Полный протокол является “самописцем” внутреннего жизненного цикла программы, однако имеющейся в нем информации для восстановления хронологии боя недостаточно.

2. Описание правил игры “Морской бой”

На игровой площадке размером 10 на 10 клеток *Пользователь* расставляет один корабль из четырех клеток, два корабля из трех клеток, три корабля из двух клеток и четыре корабля размером в одну клетку. При этом корабль представляет собой последовательность соседних клеток, стоящих на одной вертикали или на одной горизонтали. Соседние корабли не должны иметь общих точек.

Противником *Пользователя* является *Робот* (Компьютер), который автоматически расставляет корабли на своем поле по указанным выше правилам.

После расстановки начинается бой. Он представляет собой поочередные выстрелы *Пользователя* и *Робота*. При попадании в корабль противника участник боя получает возможность проведения внеочередного выстрела. Игра заканчивается при уничтожении одним из участников всех кораблей противника.

3. Постановка задачи

Целью настоящей работы является апробирование объектно-ориентированного программирования с явным выделением состояний при создании игры – классического варианта игры “Морской бой”, которая имеет пользовательский интерфейс, представленный на рис. 1.

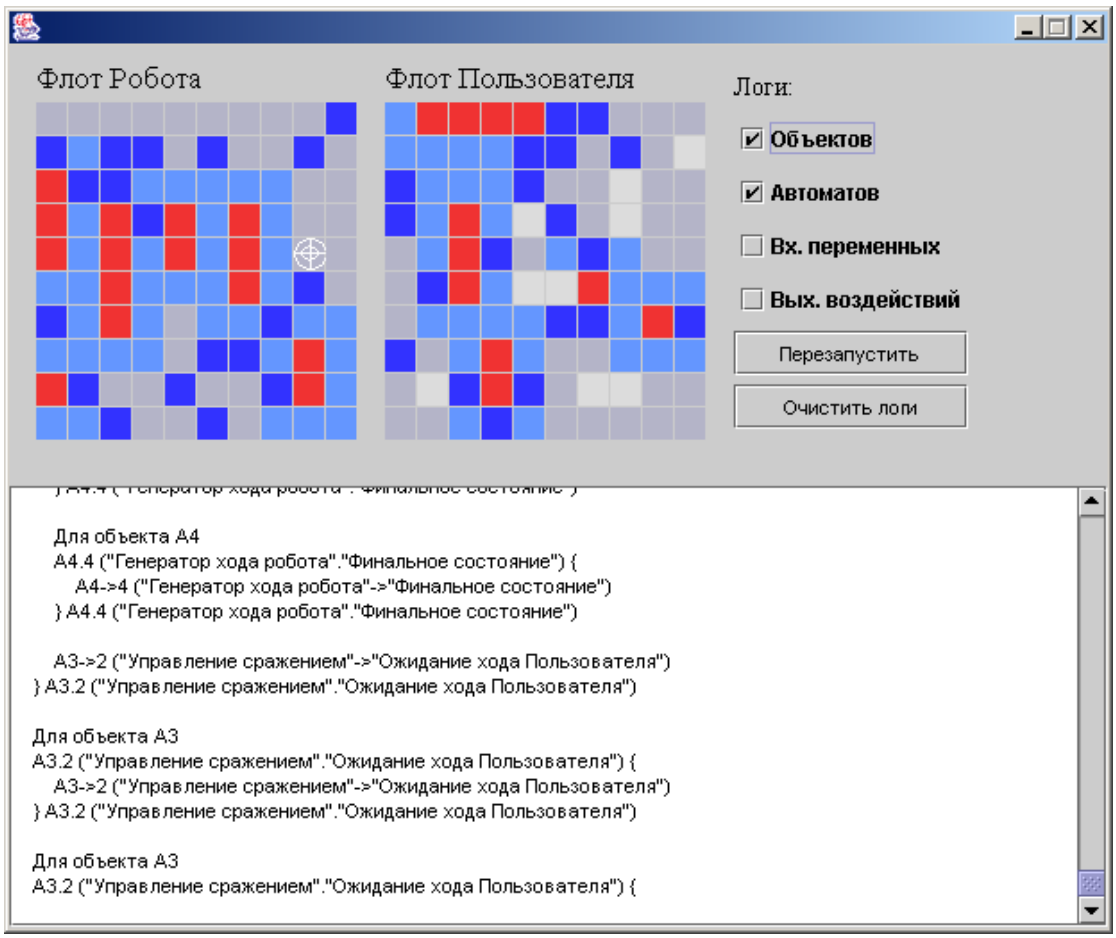

Рис. 1. Внешний вид программы

Отметим, что при создании игры не ставилась цель выбора оптимального алгоритма игры *Роботом*.

Основная идея алгоритма выбора роботом клетки для удара достаточна проста. Сначала ищется самая длинная вертикальная или горизонтальная последовательность непростреленных клеток. Затем из нее случайным образом выбирается клетка для удара.

4. Диаграмма автоматных классов

Диаграмма автоматных классов, с которыми работает аплет, преведена по адресу http://is.ifmo.ru/projects/seabattle/.

Диаграмма иллюстрирует отношение автоматных классов. При этом объект центрального автомата *A0* содержит в себе и зависит от объектов трех автоматных классов *A1*, *A2*, *A3*. Принадлежность автоматных объектов *A1*, *A2* и *A3* объекту класса *A0* продиктована вложенностью автоматов.

Связка классов *Automaton→Enumerable* реализует паттерн объектно-ориентированного программирования *Enum*. Это гарантирует, что для каждого автоматного класса будет создан единственный объект. Класс *Automaton* является базовым автоматным классом. В нем реализована общая для всех автоматов функциональность.

5. Класс *Основной Процесс* (*MainProcess*)

Словесное описание

Класс *Основной Процесс* является главным автоматным классом. Он реализует втомат *A0*, в состояния которого вложены остальные автоматы. Для взаимодействия с внешней средой и фиксации действий *Пользователя* предусмотрены несколько вспомогательных методов.

Структурная схема класса

Структурная схема класса приведена на.рис. 2.

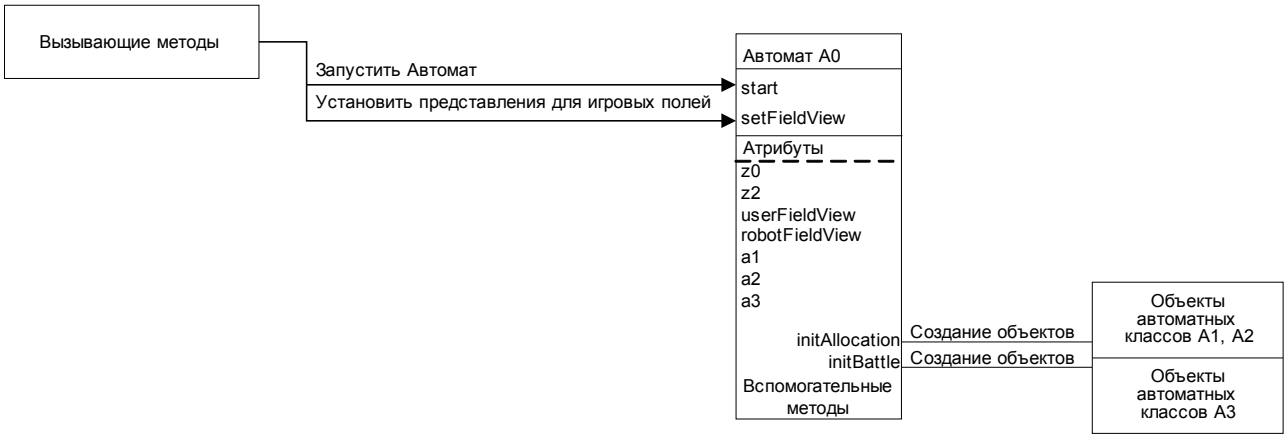

Рис. 2. Структурная схема автоматного класса A0

Главный автомат (*A0*)

Словесное описание

В состояниях главного автомата *A0* вложены автоматы расстановки кораблей *Пользователя*, расстановки кораблей *Робота* и игрового процесса.

Схема связей

На рис. 3 представлена схема связей автомата *A0*.

Рис. 3. Схема связей главного автомата *A0*

Граф переходов

На рис. 4 представлен граф переходов автомата *A0*.

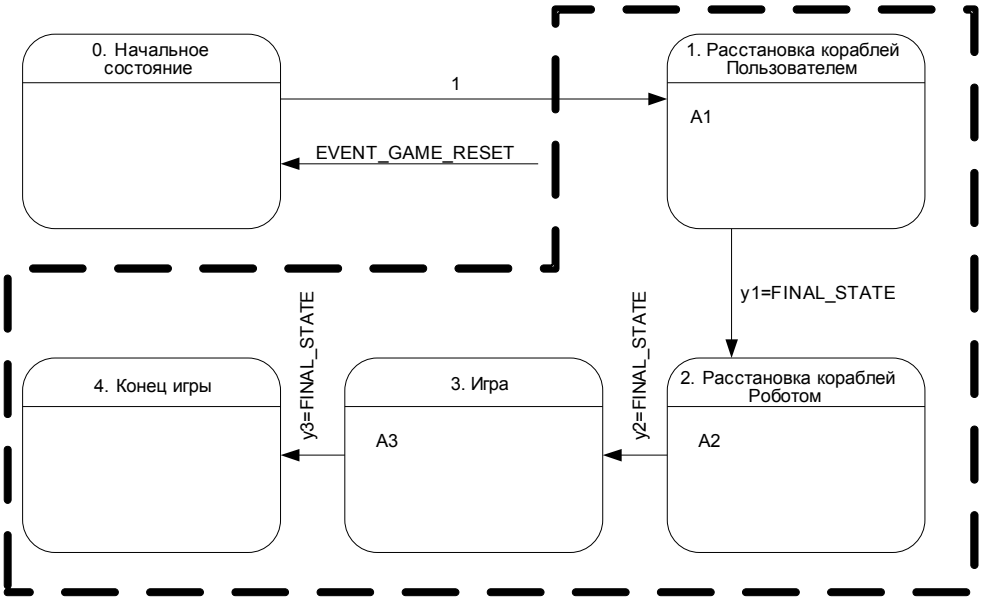

Рис. 4. Граф переходов главного автомата *A0*

6. Класс *Размещение Кораблей Пользователем* (*UserAlloc*)

Словесное описание

Класс *Размещение Кораблей Пользователем* содержит автомат расстановки кораблей *Пользователя* (*A1*).

Структурная схема класса

Структурная схема класса представлена на рис. 5.

Рис. 5. Структурная схема класса *A1*

Автомат *Расстановка кораблей Пользователя* (*A1*)

Словесное описание

Этот автомат выполняет контроль расстановки *Пользователем* своих кораблей. Автомат производит выбор типа устанавливаемого корабля, проверку корректности выбора позиции и направления корабля. При успешном завершении проверки производится установка корабля.

Схема связей

На рис. 6 представлена схема связей этого автомата *A1*.

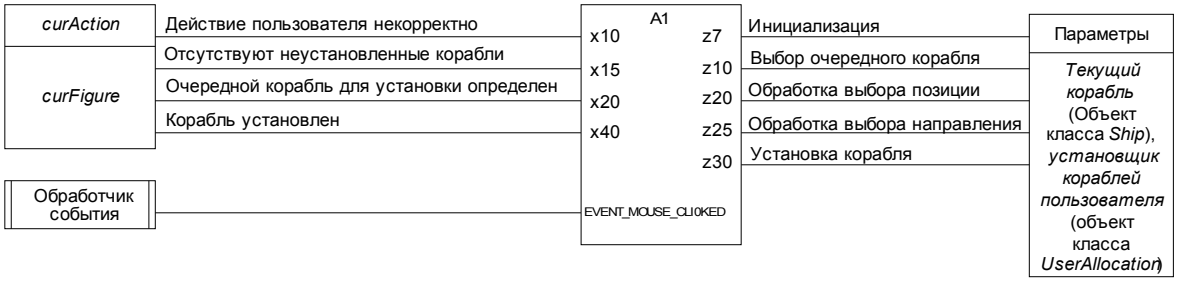

Рис. 6. Схема связей автомата *Расстановка кораблей Пользователя A1*

Граф переходов

На рис. 7 представлен граф переходов автомата *A1*.

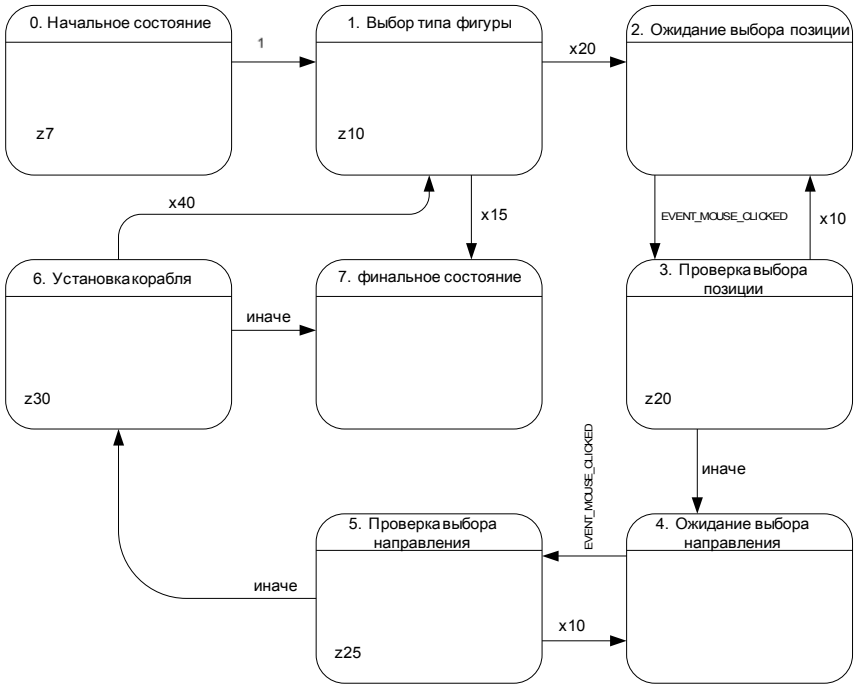

Рис. 7. Граф переходов автомата *Расстановка кораблей Пользователя A1*

7. Класс *Размещение кораблей Роботом* (*RobotAlloc*)

Словесное описание

Класс *Расстановка кораблей Роботом* функционально “похож” на класс *Расстановка кораблей Пользователем (A2)*. Процесс установки очередного корабля реализован в автомате расстановки кораблей *Робота*.

Структурная схема класса

Структурная схема класса представлена на рис. 8.

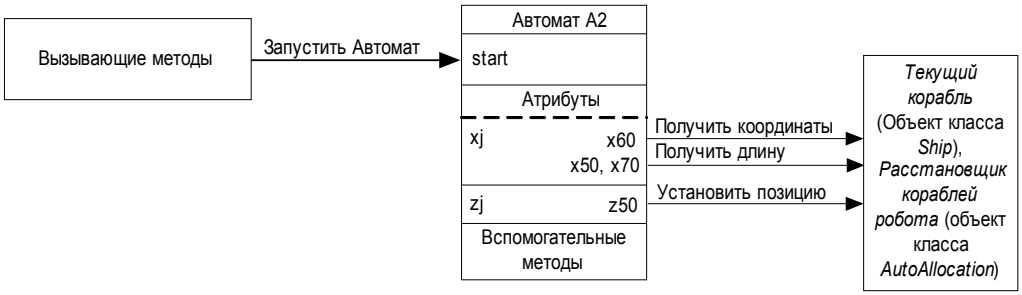

Рис. 8. Структурная схема класса *A2*

Автомат *Расстановка кораблей Робота* (*A2*)

Словесное описание

Автомат расстановки кораблей *Роботом* автоматически расставляет корабли на игровой площадке. Автомат производит выбор типа очередного устанавливаемого корабля, выбор позиции и направления установки, а также осуществляет саму установку. Данный автомат вызывается из главного автомата программы.

Схема связей

На рис. 9 представлена схема связей автомата *A2*.

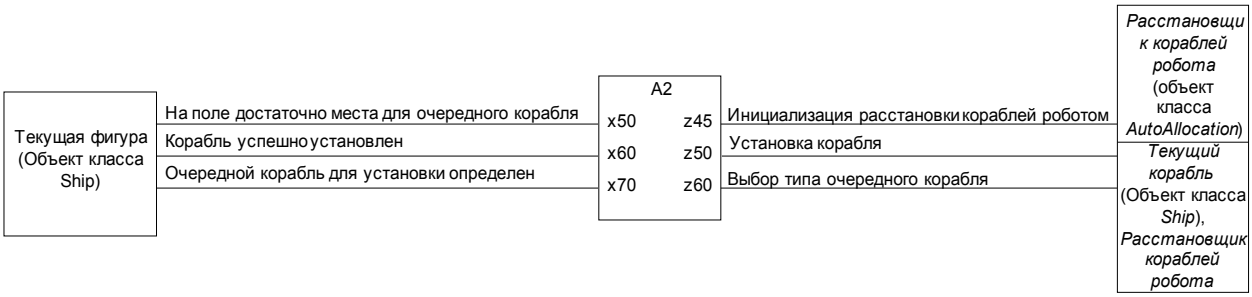

Рис. 9. Схема связей автомата *Расстановка кораблей Робота A2*

Граф переходов

На рис. 10 представлен граф переходов автомата A2.

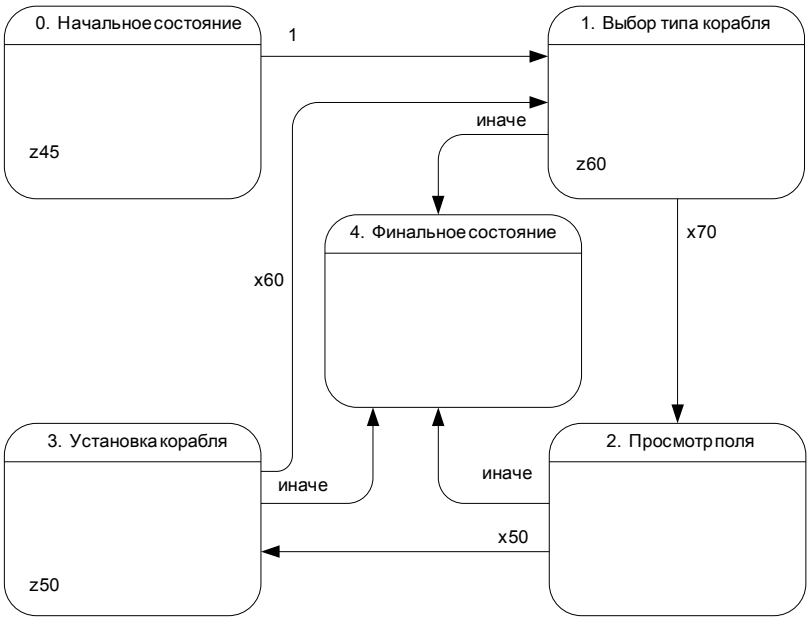

Рис. 10. Граф переходов автомата *Расстановка кораблей Робота A2*

8. Класс *Процесс игры* (*GameProcess*)

Словесное описание

Класс *Процесс Игры* реализует автомат управления игрой A3, обеспечивающий управление боем *Пользователя* и *Робота*.

Структурная схема класса

Структурная схема класса представлена на рис. 11.

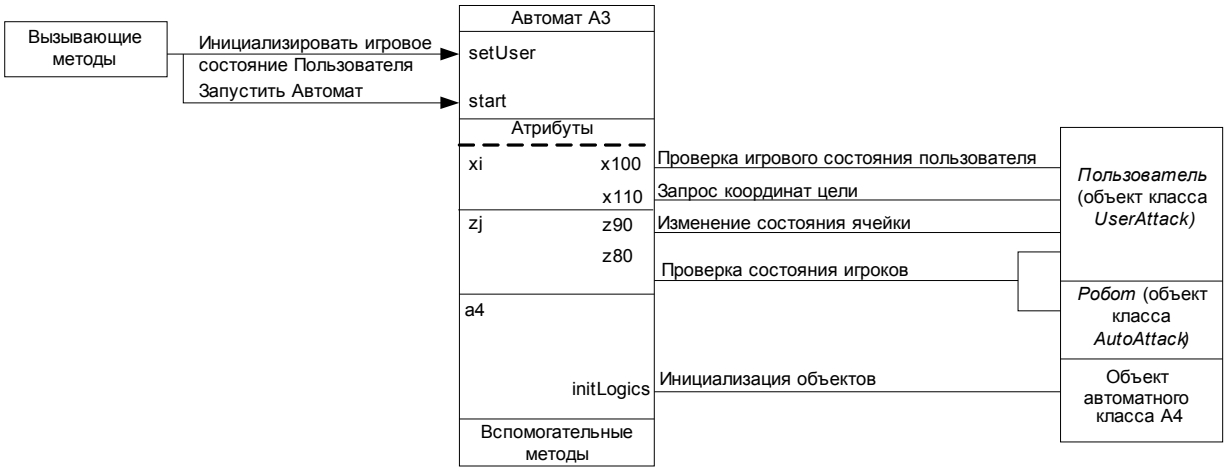

Рис. 2. Структурная схема класса *А3*

Автомат *Управление игрой* (*А3*)

Словесное описание

Автомат управления игрой реализует один такт игры, в который входит выстрел *Пользователя* и выстрел *Робота*. Процесс выполнения выстрела *Роботом* (один ход) реализован во вложенном в данный автомат автомате *А4*. При успешном выстреле *Пользователя* (при поражении им цели *Робота*) вложенный автомат *А4* не вызывается. Этим обеспечивается предоставление *Пользователю* внеочередного хода. Данный автомат вызывается из главного автомата программы.

Схема связей

На рис. 12 представлена схема связей переходов автомата *А3*.

Рис. 3. Схема связей автомата *Управление игрой А3*

Граф переходов

На рис. 13 представлен граф переходов этого автомата.

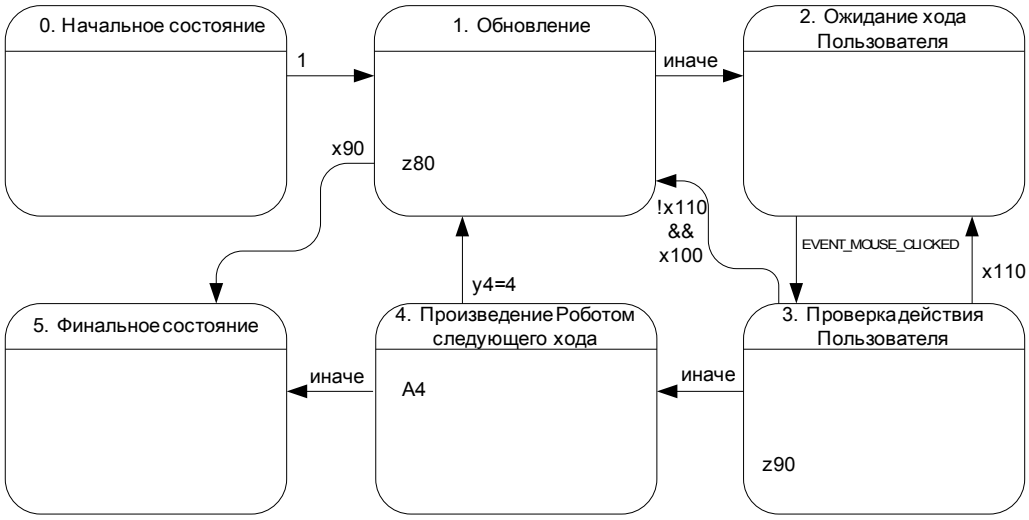

Рис. 4. Граф переходов автомата *Управление игрой A3*

9. Класс *Атака Робота* (*RobotAttack*)

Словесное описание

Класс *Атака Робота* реализует автомат, обеспечивающий выбор очередного хода *Робота*. Этот класс не принимает никаких внешних воздействий и использует только локальные методы, не воздействуя ни на какие другие объекты, кроме объекта класса *Робот (AutoAttack)*. Структурная схема класса представлена на рис. 14.

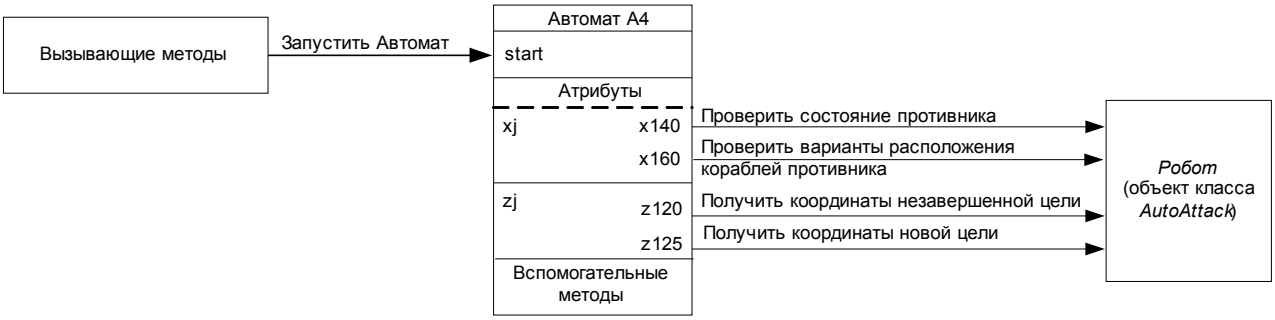

Рис. 5. Структурная схема класса *A4*

Автомат *Очередной ход Робота* (*A4*)

Словесное описание

Автомат выполнения очередного хода обеспечивает выбор поля для удара. При этом производится завершение незаконченной атаки. В случае поражения цели производится внеочередной выстрел. Автомат вызывается из автомата управления игрой.

Схема связей

Схема связей данного автомата представлена на рис. 15.

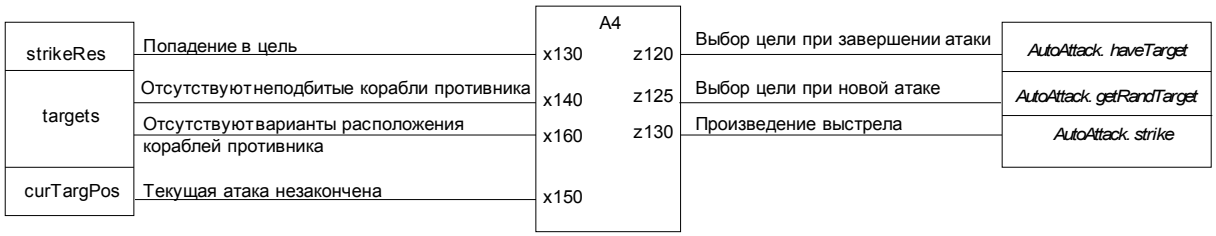

Рис. 6. Схема связей автомата *Очередной ход Робота A4*

Граф переходов

Схема граф переходов данного автомата представлен на рис. 16.

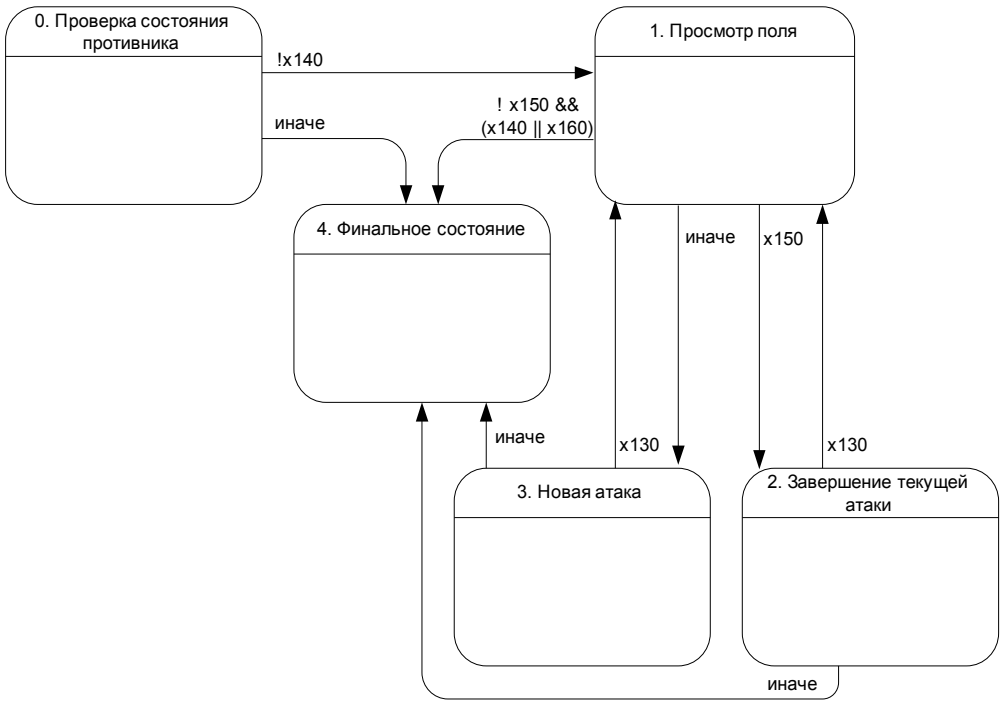

Рис. 7. Граф переходов автомата *Очередной ход Робота A4*

Заключение

Использование автоматного подхода значительно упростило построение логики программы, отладку и реализацию разработанных алгоритмов. Представление логики в виде системы взаимосвязанных графов переходов весьма наглядно и удобно для формального изоморфного перехода к тексту программ.

Апплет, приложение и исходные коды программы приведены на сайте http://is.ifmo.ru/projects/seabattle/.

Литература

1. Шалыто А.А. Технология автоматного программирования // Мир ПК. 2003. №10, с.74 – 78, http://is.ifmo.ru/works/tech_aut_prog
2. Казаков М.А., Шалыто А.А. Использование автоматного программирования для реализации визуализаторов // Компьютерные инструменты в образовании. 2004. № 2, с. 19 – 33, http://is.ifmo.ru/works/art_vis/

3. Мазин М.А., Парфенов В.Г., Шалыто А.А. Анимация. FLASH- технология. Автоматы // Компьютерные инструменты в образовании. 2003. № 4, с. 39 – 47, http://is.ifmo.ru/projects/flash/
4. Беляев А.В., Суясов Д.И., Шалыто А.А. Компьютерная игра «Космонавт». Проектирование и реализация // Компьютерные инструменты в образовании. 2004. № 4, с. 75 – 84, http://is.ifmo.ru/works/_cosmo_article.pdf
5. Туккель Н.И., Шалыто А.А. Автоматы и танки // BYTE/Россия. 2003. № 2, с.69–73, http://is.ifmo.ru/works/tanks_new/

Об авторах

Петрошенко Павел Александрович, магистрант кафедры «Компьютерные технологии» Санкт-Петербургского государственного университета информационных технологий, механики и оптики (СПбГУ ИТМО).

Корнеев Георгий Александрович, аспирант кафедры «Компьютерные технологии» СПбГУ ИТМО.

Шалыто Анатолий Абрамович, докт. техн. наук, профессор, заведующий кафедрой «Технологий программирования» СПбГУ ИТМО.