Статья опубликована в сборнике «Материалы II Международной научно-практической конференции «Объектные системы – 2010» (Зимняя сессия). Ростов–на–Дону: Южно-Российский государственный технический университет». 2010, с. 75–81. **http://www.objectsystems.ru/files/Sertificates2010_2/Object_Systems_2010_Winter_Session_Proceedings.pdf**

УДК 519.685.1

РАЗРАБОТКА ПЛАТФОРМЫ ДЛЯ АВТОМАТНОГО МОДЕЛИРОВАНИЯ И ПРОВЕДЕНИЯ СОРЕВНОВАНИЙ АВТОМАТНЫХ ИНТЕЛЛЕКТОВ С ТРЕХМЕРНОЙ ВИЗУАЛИЗАЦИЕЙ

Столяров Леонид Владимирович, ученик 10 класса, лицей «Вторая школа», Россия, Москва, lnd1212@rambler.ru.

Петряйкин Федор Алексеевич, ученик 10 класса, лицей «Вторая школа», Россия, Москва, feda.petraykin@gmail.com

Уваров Никита Сергеевич, ученик 8 класса, лицей «Вторая школа», Россия, Москва, dragonicthebest@gmail.com

1. Введение

В последнее время приобретают актуальность платформы, моделирующие поведение взаимодействующих интеллектуальных агентов [1] в виртуальной среде [2], которые осуществляют визуализацию происходящих процессов. Такого рода платформы могут использоваться как для исследования программных и интеллектуальных агентов, так и для проведения соревнований между ними.

В основе таких платформ лежит мультиагентная система [2, 3], представляющая собой совокупность множества интеллектуальных агентов и виртуальной среды. Агенты, управляемые искусственным интеллектом (ИИ) [3], существуют в виртуальной среде и взаимодействуют между собой и с окружением по законам этой среды. При этом законы виртуальной среды, как правило, близки к реальным физическим законам. Из основных свойств мультиагентной системы можно отметить:

1. Автономность агентов.
2. Ограниченность знаний агентов о состоянии системы.
3. Децентрализация – ни один агент не управляет всей системой.

Интеллектуальных агентов разделяют на две группы: обучаемые и необучаемые [1]. В данной работе рассматриваются необучаемых агенты. В большинстве случаев взаимодействие необучаемых интеллектуальных агентов с виртуальной средой описывается следующей схемой:

1. Получение доступной информации о состоянии системы.
2. Анализ полученной информации на основе записанных в агенте условий – его программы поведения («интеллекта»).
3. Выполнения действий, оказывающих воздействие на состояние системы.

2. Обзор существующих средств

В настоящее время существуют различные системы, позволяющие моделировать поведение интеллектуальных агентов в виртуальной среде. Примером такой системы является соревновательная платформа *Robocode* [4] – одна из самых популярных на данный момент. Участникам соревнований предлагается разработать интеллект для управления танком в виде программы на языке программирования *Java*, пользуясь определенными

функциями взаимодействия с виртуальной средой. При этом ставится задача уничтожить все танки других участников.

Однако эта платформа обладает рядом недостатков. В частности, отсутствует поддержка других языков и способов задания программы, реализующей интеллект, кроме языка *Java*, а также отсутствует возможность для проведения соревнований на базе других задач, кроме как сражения танков. К тому же, визуализация соревнований происходит в двумерном режиме, причем качеству графики не уделяется большого внимания. Платформа не поддерживает сетевой многопользовательский режим соревнований в реальном времени, в ходе которого каждый участник имеет возможность участвовать в соревнованиях удаленно, подключаясь со своего компьютера.

3. Применение автоматного программирования

Особенностью созданной в рамках данной работы платформы является использование автоматного программирования [5, 6] для задания интеллекта агентов [7]. Автоматное программирование – подход к созданию программ с использованием конечных автоматов [5]. Конечный автомат представляет собой конечное множество состояний, в которых он может находиться, и переходов между ними, совершаемых при выполнении определенных условий. Подход основан на создании графов переходов автоматов [5] с последующей их трансляцией во фрагменты исходного кода программы. Могут использоваться и другие варианты описания поведения автоматов, например, текстовая. С помощью автоматного подхода удобно и эффективно реализуются системы логического и событийного управления [8], к которым, в частности, относят программы управления интеллектуальными агентами.

Такой подход обладает рядом достоинств, существенных для данной области применения. Этим и обосновывается выбор автоматного способа задания интеллекта агентов. Важным достоинством автоматного подхода является возможность формально верифицировать программы управления агентами [5]. Кроме того, графы переходов, обладая большей наглядностью и простотой по сравнению с текстовыми языками программирования, эффективно описывают поведение агентов, что значительно облегчает его разработку и отладку. Графы переходов и их схемы связей могут использоваться в качестве проектной документации для спецификации поведения агента. По мнению авторов, автоматный подход естественен для описания интеллекта агентов поскольку различные состояния и переходы между ними вообще свойственны живым существам.

4. Аналоги платформы автоматного моделирования

В настоящее время существует аналог разработанной платформы – «виртуальная лаборатория для первоначального обучения проектированию программ» [9]. Этот проект образовательный. В «виртуальной лаборатории» предлагается ряд задач на управление различными одиночными агентами, которые следует решать с применением автоматного подхода. Поддерживается два способа задания управляющего автомата: при помощи построения графа переходов и специально разработанного текстового языка описания автоматов. Для редактирования графов переходов может использоваться редактор инструментального средства *UniMod* [10].

Однако эта платформа имеет такие недостатки, как отсутствие соревновательных многопользовательских режимов и недостаточная расширяемость. Двумерный режим визуализации среды, графика низкого качества, не вполне удобный пользовательский интерфейс ухудшают общее впечатление от платформы, особенно у пользователей, только начинающих изучать автоматное программирование.

В сложившейся ситуации необходима разработка новой платформы с учетом недостатков существующих аналогов и привлечением современного арсенала трехмерных графических средств.

5. Цель и этапы разработки

Цель проекта – разработка платформы автоматного моделирования и проведения соревнований автоматных ИИ с трехмерной визуализацией. Ее основное назначение – моделирование взаимодействия агентов, поведение каждого из которых задает конечный автомат. Платформа также может быть использована и для проведения соревнований автоматных ИИ. Ее основу составляет реалистичная виртуальная среда, в которой существуют агенты. Во время разработки и отладки автоматного ИИ пользователь должен иметь возможность наблюдать текущее состояние среды. Для этого и необходима ее трехмерная визуализация и воспроизведение звукового оформления. Платформа может быть использована в двух режимах: одиночном и сетевом. В первом из них пользователь работает непосредственно с клиентской программой-приложением, а во втором – необходимо наличие серверной части платформы.

Базовой частью платформы должна стать связка графической, физической и звуковой подсистем с виртуальной машиной для выполнения автоматных программ, а также скриптовая подсистема.

Разработка платформы была разделена на два этапа. Первый этап – прототипирование платформы с применением стороннего условно-бесплатного игрового движка *NeoAxis* [11] на платформе *.NET*. Цель этапа – «обкатка» концепции, принятой для построения платформы до начала ее эффективной реализации и выявление оптимальных требований к компонентам платформы.

После завершения первого этапа был создан рабочий прототип, реализующий базовую функциональность платформы. Однако в процессе разработки и отладки прототипа обнаружились трудности, которые не позволили продолжать разработку платформы на базе движка *NeoAxis*. Трудности в основном были связаны с недостатками этого движка: низкая производительность, низкая стабильность работы, неработоспособность на широком спектре аппаратных и программных конфигураций компьютеров, неработоспособность на маломощных компьютерах.

В связи с непригодностью *NeoAxis* для основы платформы был проведен анализ некоторых свободно распространяемых библиотек для графического и физического моделирования в реальном времени (так называемых графических и физических движков, обычно разрабатывающихся для реализации компьютерных игр). Большинство из них оказались неподходящими поскольку обладали рядом недостатков: сложностью или невозможностью встраивания технологий автоматного программирования и концепции мультиагентной системы, обладали недостаточно высокой производительностью и низкой стабильностью работы. Снижение производительности в проанализированных движках часто было следствием универсальности движка, что, как правило, вызывает сильное усложнение архитектуры. Движок, обладающий низкой производительностью, не может быть использован в качестве основы платформы, поскольку разработчики подобной среды не вправе требовать от всех пользователей наличия мощных компьютеров. Стабильность работы платформы и минимизация ошибок ПО играют крайне важную роль при использовании платформы для проведения соревнований.

Поскольку проанализированные свободно распространяемые движки оказались неподходящими для создания платформы на их основе на втором этапе выполнялась самостоятельная разработка базовой части платформы. В ходе этих работ были учтены особенности задачи и требования к производительности конечного продукта. При этом в основные требования, предъявляемые к базовой части платформы, были включены:

1. Встроенная поддержка технологии автоматного программирования.
2. Высокая производительность.
3. Высокая стабильность работы. Работоспособность на большом числе конфигураций компьютеров, в том числе, маломощных.

6. Архитектура платформы

Платформа работоспособна на операционных системах семейства *Windows*. В качестве основного языка программирования для разработки платформы и ее базовой части был выбран язык *C++* для платформы *Native Win32* [12] ввиду его высокой производительности и эффективности для данной задачи. С целью упрощения архитектуры базовой части и, как следствие, повышения производительности, было решено отказаться от поддержки нескольких операционных систем.

Общая архитектура базовой части платформы представляет собой связку ядра и набора модулей, объединенных в подсистемы (рис. 1).

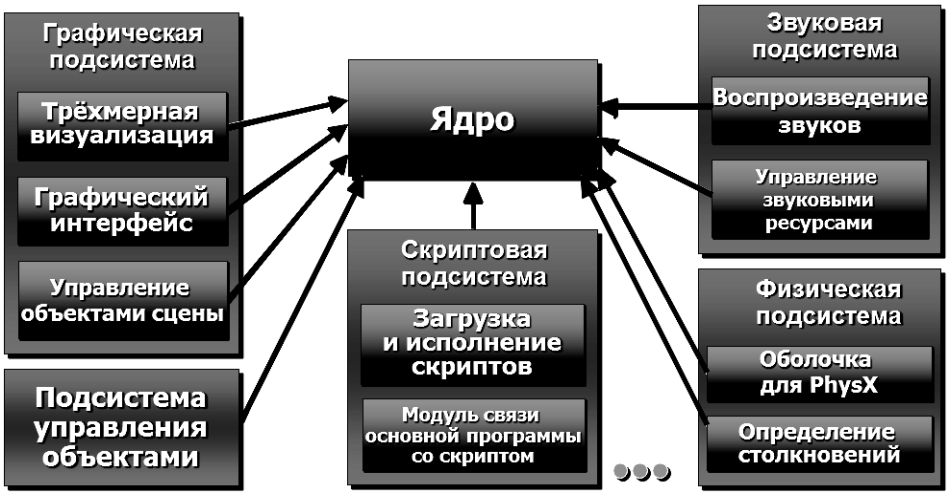

Рис. 1. Общая архитектура базовой части платформы

Модуль – структурная единица подсистемы, реализующая какую-либо часть функциональности базовой части платформы. Ядро синхронизирует работу всех запущенных модулей. Все модули регистрируются в ядре в момент инициализации программы. После этого ядро последовательно обновляет состояния всех модулей в установленном порядке в течение всего времени выполнения программы. Модулем является класс *C++*, имеющий ровно один экземпляр и реализующий интерфейс, в который входят функции инициализации, уничтожения и обновления состояния.

Подсистема – набор модулей, реализующий блок функциональности. Основу базовой части платформы составляют следующие подсистемы: графическая, физическая, звуковая, скриптовая, подсистема управления сущностями, подсистема взаимодействия с *Windows*, подсистема пользовательского графического интерфейса, подсистема управления ресурсами и т. п.

Архитектура базовой части платформы разработана с поддержкой внешних подключаемых модулей (*plug-ins*) в виде динамически компонуемых библиотек, что открывает широкие возможности для расширения функциональности платформы.

7. Подсистемы базовой части платформы

Одной из наиболее важных подсистем базовой части платформы является графическая подсистема, поскольку она необходима для визуализации виртуальной среды. Эта подсистема обеспечивает загрузку и визуализацию трехмерных объектов, а также различных графических эффектов, реализуемых с помощью вершинных и пиксельных шейдеров, таких как, например, туман, освещение, водная поверхность.

В качестве низкоуровневого графического *API* был использован *Microsoft DirectX 9* [12, 13], так как он весьма распространен, обладает высокой производительностью и позволяет эффективно визуализировать трехмерную среду в реальном времени на большом спектре оборудования. Графическая подсистема содержит модули отображения текста, графики, интерфейса пользователя, модули управления сценой (*scene management*) [14] и

графическими ресурсами. К графическим ресурсам относят трехмерные модели, описания материалов, текстуры, шейдеры. В задачи модуля управления сценой входит обработка объектов сцены, их сортировка и подготовка к визуализации. С целью повышения производительности графической подсистемы была применена технология *LOD* (*level of detail*, уровни детализации) [13].

Помимо статических объектов, составляющих виртуальную среду, существуют агенты, визуализация которых требует поддержки анимированных объектов. Для их визуализации был выбран метод скелетной анимации [15]. Для повышения качества анимации объектов использовался скиннинг (*skinning*) – процесс нахождения положения вершин анимированного объекта по заданному положению его костей и веса каждой кости по отношению к вершинам.

Не менее важной составляющей базовой части платформы является скриптовая подсистема. В ее основе лежит *Lua*-интерпретатор [16]. Язык *Lua* был выбран по нескольким причинам:

1. Этот язык является день одним из самых производительных.
2. Его функциональность позволяет эффективно встроить технологии автоматного программирования.
3. Простой синтаксис, обеспечивающий при этом большие выразительные возможности.
4. Большое число пользователей.
5. Подробная документация [16].

Эта подсистема используется для исполнения скриптов инициализации платформы и управления некоторыми ее частями. Также с ее помощью может быть значительно расширена функциональность платформы без внесения изменений в базовую часть.

Подсистема звука также играет важную роль, так как она способствует наблюдению процессов, происходящих в виртуальной среде. К этой подсистеме относятся модули, отвечающие за загрузку и воспроизведение звуковых файлов, а также за управление звуковыми ресурсами. В качестве низкоуровневого *API* для воспроизведения звука была использована библиотека *FMOD* [17].

Для реализации реалистичного взаимодействия объектов виртуальной среды друг с другом применяются библиотеки физического моделирования в реальном времени (физические движки). В связи со сложностью написания подобных систем было решено использовать движок *NVidia PhysX* [18], бесплатный для некоммерческого использования. Этот выбор был обусловлен тем, что он наиболее функционален и стабилен, имеет удобный интерфейс, а также аппаратно ускоряется на современных моделях видеокарт и обладает высокой производительностью.

8. Реализация поддержки автоматного программирования

В ходе разработки особое внимание было уделено рассмотрению поддержки платформой автоматного программирования. В настоящее время доступны два варианта задания автомата управления агентом: графом переходов и текстовой нотацией, применяемой в «виртуальной лаборатории» [9]. Поддержка различных вариантов представления управляющего автомата основывается на способности базовой части платформы исполнять код *Lua*. Для каждого формата описания автомата существует транслятор, генерирующий *Lua*-код управления агентом. Сгенерированный код исполняется скриптовой подсистемой и таким образом управляет поведением агента. Такая схема позволяет легко расширять функциональность системы, добавляя новые варианты описания интеллектов. Для реализации поддержки требуемого варианта описания необходимо создать соответствующий транслятор, генерирующий код на *Lua*.

Графы переходов создаются с помощью графического редактора *Microsoft Visio*. В состав платформы входит редактор текстового языка описания автоматов с синтаксической

подсветкой. Также в состав платформы входят два транслятора: *Visio2Auto* [19] и *Atxt2Xml*, которые реализованы на языке программирования *C#* для платформы *.NET*.

Результаты

В работе описана первая версия платформы для автоматного моделирования и проведения соревнований автоматных ИИ с трехмерной визуализацией. В настоящее время она поддерживает такие функциональные возможности, как загрузка и визуализация трехмерной виртуальной среды, пользовательское управление при помощи графического интерфейса, моделирование физических взаимодействий, воспроизведение звукового оформления виртуальной среды (рис. 2).

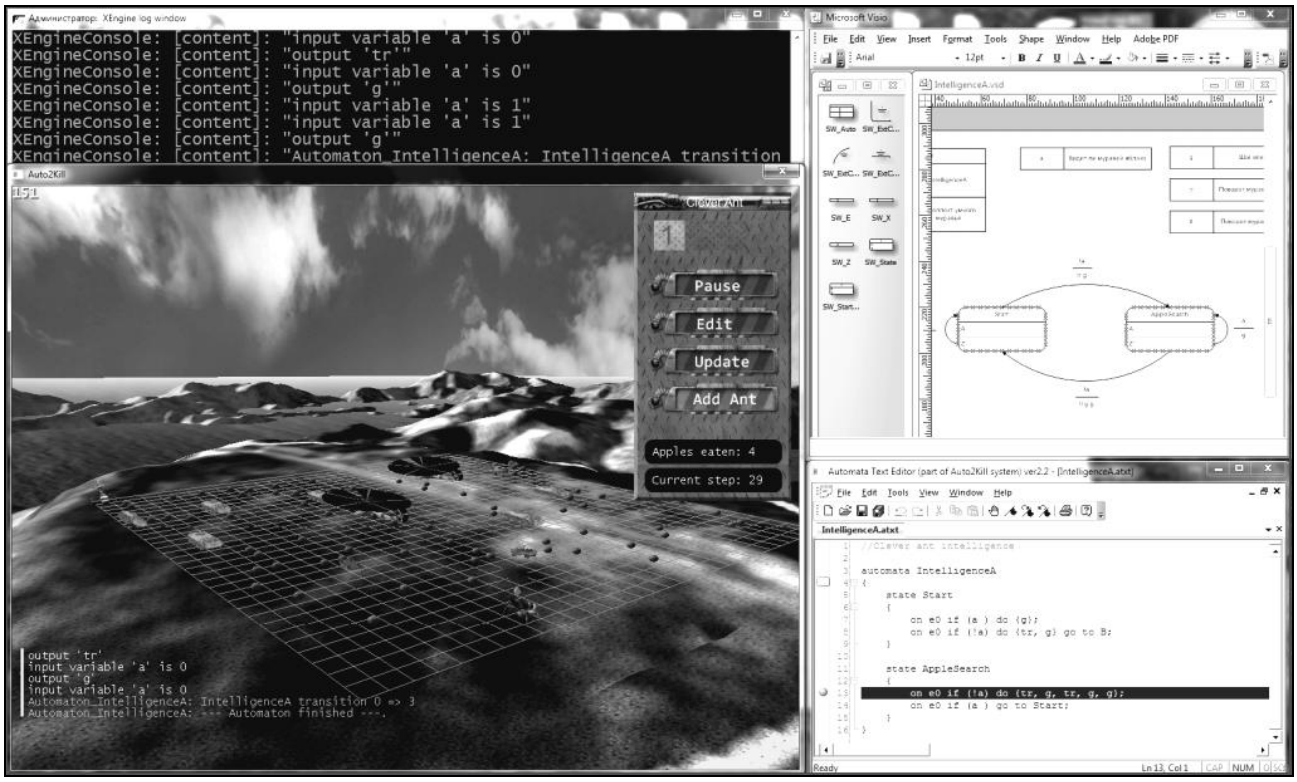

Рис. 2. Скриншот платформы во время сеанса моделирования

Платформа обладает высокой производительностью, стабильностью работы и работоспособна на значительном числе конфигураций компьютеров.

В качестве примера была реализована «задача об умном муравье» [20] и обеспечена возможность проведения соревнований автоматных ИИ «муравьев». Имеется возможность проведения соревнований в многопользовательском сетевом режиме. В этом случае все участники соревнований подключаются к заранее запущенному серверу для проведения соревнований.

Кроме графического представления автоматов, также был создан набор утилит, обеспечивающих работу с текстовым описанием автоматов. В этот набор входят транслятор текстового языка описания автоматов и редактор этого языка с подсветкой синтаксиса.

По мере разработки проводилось документирование исходного кода платформы, с использованием автоматической системы документирования кода *Doxygen* [21], что существенно упрощает модификацию и расширение платформы. Было составлено руководство по использованию платформы и проведения соревнований автоматных ИИ на ее основе.

Работа выполнена под руководством сотрудника кафедры информатики МФТИ, преподавателя программирования И. Р. Дединского.

Авторы благодарны профессорам СПбГУ ИТМО А. А. Шалыто и В. Г. Парфенову за постановку задачи и проявленный интерес к работе, а также старшему разработчику компании NVidia А. С. Татаринову за ценные консультации в ходе разработки.

Работа отмечена дипломом на Балтийском научно-инженерном конкурсе Baltic SEF 2009, проводящимся компанией Intel (http://baltic.contedu.ru).

Литература

1. *Рассел С., Норвиг П.* Искусственный интеллект. Современный подход. Вильямс, 2006.
2. *Wooldridge M.* An Introduction to MultiAgent Systems. John Wiley & Sons, 2002.
3. *Panait L., Luke S.* The State of the Art. Autonomous Agents and Multi-Agent Systems //Autonomous agents and multi-agent systems. 2005. 11(3).
4. *Robocode Wiki*. http://robowiki.net.
5. *Шалыто А. А.* Switch-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука, 1998.
6. *Поликарпова Н. И., Шалыто А. А.* Автоматное программирование. СПб.: Питер, 2009.
7. *Шалыто А. А., Наумов Л. А.* Методы объектно-ориентированной реализации реактивных агентов на основе конечных автоматов //Искусственный интеллект. 2004. № 4.
8. *Шалыто А. А., Туккель Н. И.* Реализация автоматов при программировании событийных систем //Программист. 2004. № 2.
9. *Красильников Н. Н., Парфенов В. Г., Царев Ф. Н., Шалыто А. А.* Виртуальная лаборатория для первоначального обучения проектированию программ //Компьютерные инструменты в образовании. 2007. № 5.
10. *UniMod*. http://unimod.sourceforge.net
11. *NeoAxis Game Engine*. http://www.neoaxisgroup.com
12. *Microsoft Software Developer Network*. http://msdn.microsoft.com
13. *Луна Ф. Д.* Введение в программирование трехмерных игр с DirectX 9.0. Wordware Publishing Inc. 2003.
14. *Снук Г.* 3D-ландшафты в реальном времени на C++ и DirectX 9. М.: Кудиц-Образ, 2007.
15. *Windows DirectX* Graphics Documentation. 2009.
16. *Справочное руководство* по языку Lua. http://www.lua.ru/doc
17. *FMOD* Interactive Audio Middleware. http://www.fmod.org
18. *Раздел PhysX* на сайте NVidia. http://www.nvidia.ru/object/physx_new_ru.html
19. *Столяров Л. В.* Трансляция описания автоматов, представленных в формате Microsoft Visio в исходный код на языке C //Компьютерные инструменты в образовании. 2009. № 5.
20. *О построении автоматов* с минимальным числом состояний для задачи об «умном муравье». СПбГУ ИТМО. 2007. http://is.ifmo.ru/download/ant_ga_min_number_of_state.pdf
21. *Официальный сайт* системы Doxygen. http://doxygen.org