УДК 681.3: 658.56

В.В. Кожаев

Инсерционное моделирование сценариев ролевых игр

Предложена инсерционная модель однопоточной ролевой игры – первое применение инсерционного моделирования к разработке игр. На основании данной модели разработан декларативный язык описания сценариев.

A model of a signal-insertion role-playing game, which is the first application of modeling to the development of insertional games is suggested. On the basis of the given model a declarative language of scenarios description is developed.

Запропоновано інсерційну модель однопоточної рольової гри, що є першим застосуванням інсерційного моделювання до розробки ігор. На підставі поданої моделі розроблено декларативну мову опису сценаріїв.

Введение. Ролевыми играми будем называть такие, в которых персонажи, обладающие неким поведением, взаимодействуют между собой, персонажем игрока и средой. Под поведением, в общем случае, следует понимать обмен сообщениями между персонажами и средой и реакцию персонажа на сообщения: отправку новых сообщений, изменение свойств и решение как реагировать на следующие сообщения. Другими словами, персонажей можно считать актерами [1].

Правила поведения персонажа будем называть сценарием его поведения (далее – сценарий). В настоящее время сценаристы используют неформальные описания сценариев. Однако такие описания изобилуют неточностями, поэтому в практике разработки игр часто случается, что воплощенная в программный код игра существенно отличается от первоначального замысла сценариста. Таким образом, возникает задача описания сценариев ролевых игр с помощью некоторого формализма. Поскольку игру и персонажей можно рассматривать как среду с погруженными в нее агентами, для создания модели удобно использовать инсерционное исчисление [2]. До недавнего времени инсерционное моделирование использовалось в основном для верификации и анализа отказоустойчивых систем. Однако этот инструмент можно использовать и для моделирования систем, не требующих высокой отказоустойчивости, в частности игр, рассмотренных в [3].

Пример игры

В центре игрового поля расположен форт, на котором игрок может строить неподвижные башни различного типа. Кроме того, на территории форта располагаются роботы. Из-за пределов игрового поля к форту движутся роботы противника, стреляющие по роботам и башням игрока. Последний строит башни на территории форта, башни стреляют по противнику или создают роботов. Цель игры состоит в том, чтобы, найдя оптимальную стратегию постройки башен, продержаться против роботов противника в течение определенного времени.

Игровое поле. Прямоугольное игровое поле делится на две части: форт и поле. Роботы игрока могут перемещаться только по форту, роботы противника – только по полю. Роботы игрока не выходят за территорию форта, соответственно роботы противника не могут в форт попасть. Форт имеет вид односвязной фигуры (вид фигуры изменяется от уровня к уровню).

Персонажи. В игре участвуют два типа персонажей: башни (неподвижные) и роботы (подвижные). Вторые в свою очередь делятся на роботов игрока и роботов противника. У каждого персонажа имеется показатель, называемый «жизнь». При повреждениях значение показателя снижается. Если жизнь равна нулю или меньше, персонаж погибает.

Роботы игрока. Робот игрока предназначен для починки башен. Под починкой понимается продление жизни, происходящее в течение определенного интервала времени. Для починки робот должен подойти к башне и некоторое время находиться возле нее. В роботов можно стрелять так же, как и в башни.

Роботы противника. Появляются за пределами игрового поля, движутся по направлению к форту. Перед возникновением, робот противника выбирает цель. После того, как робот подойдет к цели на расстояние выстрела, он останавливается и начинает стрельбу. Выбор происходит с помощью оценочной функции, учи-

тывающей расстояние до цели, ее важность для жизнеобеспечения и пр. Если цель уничтожена (не обязательно данным персонажем), выбирается другая цель.

Башни. Неподвижны в отличие от роботов. Стреляют по роботам противника либо создают новых роботов. Выбор цели также происходит с помощью оценочной функции, и он возможен только из числа роботов, находящихся на расстоянии выстрела. Если цель уничтожена так же, как и роботы противника, башни выбирают другую цель.

Игрок может в любой момент времени вмешаться в течение игры и построить новую башню; это возможно, если игрок обладает достаточным количеством монет, а они начисляются за уничтожение башнями игрока роботов противника.

Время и пространство. Как время, так и пространство (игровое поле) дискретны. Единица дискретного времени называется *тиком*. Расстояние, на которое робот может переместиться на один тик, зависит от его типа.

Инсерционная модель игры

Персонажи игры рассматриваются как агенты, которые погружаются в среду игры. Игрок находится во внешней среде и отождествляется с ней.

Поведение игрока. Действия игрока: создание робота в точке x; – создание башни в точке x, W; наблюдение за игрой в течение одного тика. Поскольку о поведении игрока ничего не известно, его можно моделировать как недетерминированную систему с поведением, определяемым таким уравнением: $P = \sum_{x \in M} (R(x).P + T(x).P + W.P)$, где M – множество точек игрового поля.

Среда игры определяет для каждого персонажа его положение на игровом поле и показатель жизни. Кроме атрибутов агентов, состояние среды определяет число монет, заработанных игроком. Таким образом, общее состояние системы имеет вид $P[E[u]]$, где P – состояние игрока, $E[u]$ – состояние среды игры**,** $u = (u1, \ldots, u_m)$ – состояния персонажей игры.

Действия персонажей

Действия роботов игрока

show_start_moving(x) –начало движения;

show_stop_moving(x) – окончание движения;

select_new_tower(x) – выбирает новую башню;

before_agony(x) – начало агонии;

show_splash(x) – реакция на попадание (вспышку);

show_start_charging(x) – начало зарядки;

how_moving_process(x) – демонстрирует процесс ремонта башни (заряда жизненной силой);

remove_robot(x) – удаление робота с игрового поля;

show_charging_process(x) – процесс ремонта башни (заряда жизненной силой).

Действия роботов противника

shoot(x) – стреляет по противнику;

select_target(x) – выбирает новую цель;

move_to_shoot_distance(x) – движется к цели на расстояние выстрела;

show_splash(x) – вспышка от попадания по роботу.

Функции погружения внешней среды и среды игры

Создание нового робота:

$$(R(x).P)[E[u]] \xrightarrow{\tau} P[E'[u, v]],$$

где τ – пустое действие (оно не изменяет изображения состояния игры на экране), E' – новое состояние среды игры, v – новый робот игрока в начальном состоянии. Условие применения этого правила – достаточное количество монет у игрока. В новом состоянии робот v установлен в точке x.

Создание новой башни:

$$(R(x).P)[E[u]] \xrightarrow{\tau} P[E'[u, v]]$$

аналогично созданию нового робота.

Изменение состояния игры

$$(W.P)[E[a_1.u_1, \ldots, a_m.u_m]] \xrightarrow{\varphi(E, a_1, \ldots, a_m)} P[E'[u'_1, \ldots, u'_m]],$$

где $\varphi(E, a_1, \ldots, a_m)$ – демонстрация состояния игры на экране после выполнения всех действий. Новое состояние u'_i i-го персонажа совпадает с u_i', если действие этого персонажа выполнено успешно или равно Δ, если персонаж выведен из строя. Условие применимости этого правила состоит в том, что все действия персонажей, не выведенных из строя, могут быть выполнены успешно.

Опишем условия, при которых робот переходит в конечное состояние. Предположим, что ро-

бот и башня стреляют друг в друга. Поскольку условием существования персонажей принято положительное значение жизни, первым погибнет тот персонаж, у которого жизнь иссякнет раньше, причем возможна ситуация, когда по одному персонажу игрока стреляют несколько роботов противника и наоборот. Выстрелы осуществляются с определенным разлетом, зависящим от характеристик конкретного персонажа. Таким образом, возможна ситуация, когда при выстреле попадания не происходит. В игре имеется как оружие массового («граната»), так и индивидуального («снайперская винтовка») поражения. Оружие массового поражения стреляет в определенную точку и поражает всех персонажей, находящихся в радиусе поражения. Величина поражения обратно пропорциональна расстоянию от эпицентра взрыва. Для простоты принято, что если персонаж находится вне радиуса поражения, он не повреждается. Оружие индивидуального поражения может поразить только конкретного персонажа. Поскольку роботы противника и роботы игрока находятся на разных участках игрового поля, для простоты принимается, что выстрелы роботов противника поражают только персонажей игрока, соответственно выстрелы персонажей игрока поражают только противника. Дальность выстрела зависит от типа оружия. Каждый персонаж может владеть только одним типом оружия. Соответственно для выстрела робот противника должен приблизится к роботу противника на расстояние выстрела. Если робот, по которому стреляют, движется, то принимается, что меткость персонажа, стреляющего по нему из оружия индивидуального поражения не меняется. К оружию массового поражения понятие меткости выстрела по определенному персонажу не применимо, поскольку поражается группа персонажей, находящихся на расстоянии, меньшем, чем радиус поражения для данного оружия.

Поведение персонажей. В соответствии с описанием игры, у персонажа может быть достаточно сложное поведение. В частности, поскольку он сам выбирает следующую цель, ему должны быть доступны все данные о расположении персонажей на игровом поле. Если робот противника выбрал целью робот игрока, а этот робот движется, робот противника должен отслеживать движение цели и менять свои действия в соответствии с этим движением (погоня за целью).

Каждый персонаж имеет свою среду, которая локально ограничивает действия агента и пользуется информацией из состояния среды. Ввиду сказанного функционирование персонажа определяется уравнениями вида

$u = before.while.\sum_{i \in I} \alpha_i.after_i.v_i,\ while = y.while$, где

before и *while* – действия, производимые над агентом соответственно до входа в состояние из некоего другого состояния и действия, происходящие при переходе состояния самого в себя (петля); *afer* – действия, производимые при переходе из состояния α в υ.

Реализация инсерционной модели

Для управления персонажами предлагается использовать детерминированный конечный автомат, когда управление происходит в процессе переходов от состояния к состоянию с помощью управляющих воздействий. Под последними понимают функции, выполняемые над персонажами перед входом в данное состояние, перед выходом из него, при переходе из состояния *A* в состояние *B* и при переходе из определенного состояния в него же (петля) [1].

Ввиду того, что большинство платформ, используемых для разработки казуальных игр, однопоточны, предлагается реализовывать активную систему обработки событий. Состояние может измениться единожды за тик. Если это произошло, выполняются воздействия, приписанные к переходу из состояния в состояние, иначе – воздействия, выполняемые при переходе состояния в себя. Переход в данное состояние из текущего происходит, если справедливо логическое выражение, атомами которого служат идендификаторы функций, реализованные на императивном языке программирования, подходящем для данной платформы и возвращающих логическое значение. Управляющие воздействия также являются идентификаторами методов на императивном языке, но в отличие от первых, требование к возвращаемому значению не выдвигается.

Язык разметки сценариев. Для задания поведения персонажа создадим текстовый язык.

Программа на этом языке состоит из двух частей: раздела объявлений и собственно тела программы. В разделе объявлений указываются допустимые имена состояний управляющих воздействий и атомов логических выражений. Причем, имена состояний должны быть уникальными.

Каждое состояние состоит из списка управляющих воздействий и условий перехода. Условие перехода в свою очередь содержит логическое выражение, при выполнении которого происходит переход и управляющие воздействия, выполняемые при этом.

Формальное описание языка создания сценариев. Представим синтаксис языка в виде контекстно независимой грамматики.

Programm::=<states_definition> <athoms_definition> <methods_definition> <programm_body>
states_definition::=states:<list>;
athoms_definition::=atoms:<list>;
methods_definition::=methods:<list>;
list::={<list_body>};
list_body::=<lexeme>;
list_body::=<list_body> <lexeme>;
program_body_definitions::=<states_list>
states_list::=<state_definition>
states_list::=<states_list> <state_definition>
state_definition::=state:<lexeme>
methods_before:<list>
methods_after:<list>
methods_in_process:<list>
transitions_definition:<transition_list>
transitions_list::=<transition_definition>
transitions_list::=<transition_list> <transition_definition>
transition_definition::=state:<lexeme>
condition:<logical_expression>
methods:<list>

Программа робота игрока. Представим программу робота игрока, предлагаемую для инсерционного моделирования. Программы для роботов противника и башен аналогичны, как и программы для роботов персонажа и башен.

states:{*move_to_tower*,*fix_tower*,*wait*,*agony*,*death*, *not_move*}
//Состояния, атомы и методы объявляются в начале программы
//(как переменные в паскале)
atoms: {*is_life_more_null*,
is_there_tower_for_fixing,
is_selected_tower_dead,
is_life_more_then_null,
is_robot_near_tower,
before_agony,
remove_robot,
show_splash,
show_charging_process}
methods:{*show_start_moving*,
show_stop_moving,
select_new_tower,
before_agony,
show_splash,
show_start_charging,
how_moving_process,
remove_robot,
show_charging_process}
program_body:
state:*move_to_tower*
methods_before:{*show_start_moving*}
methods_after:{*show_stop_moving*}
methods_in_process:
{*show_moving_process*}
transitions:
state:*agony*
condition:*is_life_more_then_null*
state:*move_to_tower*
condition: *is_selected_tower_dead*
methods:{*select_new_tower*}
state:*fix_tower*
condition: !*is_selected_tower_dead* && *is_robot_near_tower*
methods:{*show_start_charging*}
state:*agony*
methods_before:{*before_agony*}
methods_after:{*remove_robot*}
methods_in_process:{*show_splash*}
transitions: {}//Нет переходов
state:*fix_tower*
methods_before:{}//Нет действий
methods_after:{}
methods_in_process:{*show_charging_process*}

Заключение. Предложенная инсерционная модель однопоточной ролевой игры – первое применение инсерционного моделирования к разработке игр. На основании описанной модели разработан декларативный язык описания сценариев. Научная новизна состоит в построении формального описания сценария игры. Автору не известно других попыток описать сценарий формально.

1. *Кожаев В.В*. Использование декларативного подхода к созданию каркаса двумерных игр // Компьютерная математика. – 2011. – № 1. – С. 79–85.
2. *Летичевский А.А*., *Капитонова Ю.В*. Инсерционное моделирование / Пр. міжнар. конф. «50 років Інституту кібернетики ім. В.М. Глушкова НАН України», Київ, 2008. – С. 293–301.
3. *Кожаев В.В*. Тестирование интереса к игре / Матеріали сьомої міжнар. наук.-практ. конф. з програмування. УкрПРОГ2010, Київ 2010. – С. 452–456.
4. *Летичевский А.А*. Инсерционное моделирование / Курс лекций – *http://schum.kiev.ua/let/*

Тел. для справок: +38 063 775-7467 (Киев)
E-mail: *vkozhaev@gmail.com*