**Содержание**

[Введение 4](#_Toc406422134)

[1 Раздел 1 5](#_Toc406422135)

[1.1 Подраздел 1 5](#_Toc406422136)

[1.2 Подраздел 2 5](#_Toc406422137)

[2 Раздел 2 7](#_Toc406422138)

[2.1 Подраздел 1 7](#_Toc406422139)

[2.2 Подраздел 2 7](#_Toc406422140)

[3 Раздел 3 9](#_Toc406422141)

[3.1 Подраздел 1 9](#_Toc406422142)

[3.2 Подраздел 2 9](#_Toc406422143)

[Заключение 10](#_Toc406422144)

[Список использованных источников 11](#_Toc406422145)

[Приложение А Задание на выполнение бакалаврской работы 12](#_Toc406422146)

[Приложение Б Руководство пользователя 14](#_Toc406422147)

[Приложение В Исходный код программы 15](#_Toc406422148)

Введение

Объем рынка игр растет с каждым годом (смотри рисунок Г.1 и Г.2). В первую очередь развитие игровых приложений обязано стремительному прогрессу, но также важная роль в разработке игр ложится на плечи программистов. Именно с их помощью реализуются идеи геймдизайнеров, именно они собирают воедино воображения художников с целью создания новой игры.

Программисту, в настоящее время, чтобы написать игру, что называется, «с нуля» необходимо позаботиться о многих важных аспектах: игровая платформа, графическая система, аудио система, системы моделирования игровой физики и виртуального интеллекта игры. Каждая из этих областей объемна и потребуется много времени, чтобы связать воедино все системы в соответствии с игровой логикой.

Для того, чтобы упростить и ускорить работу над созданием игры программисты написали множество библиотек для работы с графическими системами и для моделирования игровой физики. Такие библиотеки, как Lightweight Java Game Library, предоставляют инструменты для работы с графической и аудио системами, а библиотека box2d предоставляет инструменты для моделирования реалистичной физики в плоском пространстве. Однако для упрощения реализации виртуального интеллекта долгое время не было придумано, по большому счету, ничего. Над созданием искусственного интеллекта в игре программист трудился самостоятельно, так как этот процесс не был сложным или утомительным. В конечном итоге модуль виртуального интеллекта содержал в себе некоторое количество условных операторов.

Такое решение обнажило бы свою слабую сторону в играх с развитым поведением персонажей потому, что условных операторов становилось очень много, что затрудняло отладку и дальнейшую поддержку игры. Пример такой игры – Spore. В ней все игровые персонажи имели если не уникальное, то редко повторяющееся поведение, для создания которого в Spore использовался совершенно другой подход.

Только в последнее десятилетие (примерно с момента выхода игры Spore) виртуальный интеллект в играх стал развиваться и появились некоторые библиотеки AIEngine (Artificial Intelligence Engine), которые обобщали накопленные программистами знания об интеллекте в играх. Эти библиотеки предоставляют инструменты для создания конечных автоматов поведения, для обработки взаимодействия автономных объектов. Но конечные автоматы в общем виде сложны и часто запутаны. Чтобы стратегия объекта была ясной были созданы деревья поведения – конечные автоматы древовидной структуры, состоящие из вершин трех типов: вершины-действия, вершины-условия и управляющие вершины. Вершина-действие содержит в себе некоторое возможное действие объекта (бежать, искать, стрелять), вершина-условие содержит в себе некоторый предикат (есть патроны?, враг рядом?), в зависимости от которого выбирается следующее состояние, управляющая вершина организует порядок обхода дочерних вершин (параллельно, до первой успешной вершины, до первой неуспешной вершины).[[1]](#footnote-1)

Цель данной работы - проектирование библиотеки и реализация визуального средства для создания стратегий поведения виртуальных игровых персонажей на основе деревьев поведений.

1. Обзор существующих инструментов

## Общие сведения

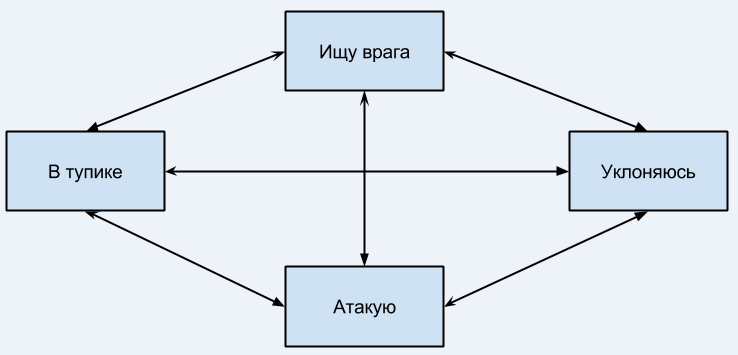
Для разработки виртуальных игровых объектов необходимо создать и поддерживать большой набор их поведений. Например, в военных играх виртуальному персонажу необходимо распознавать опасность и убегать в укрытие, когда уровень здоровья ниже 10%. От количества разнообразных вариантов действий, которые могут использовать игровые персонажи, зависит количество игровых ситуаций, которые могут быть распознаны и приняты во внимание. Чем больше различного поведения игроки будут встречать в играх даже от несущественных (фоновых) объектах[[2]](#footnote-2), тем интереснее будет игра. 

Рисунок 1.1 - Конечный автомат поведения игрового объекта в военных играх

Для создания поведения долгое время использовались конечные автоматы, где каждое поведение может быть представлено графически (рисунок 1.1). Одновременно автомат может находиться в одном состоянии, которое представляет поведение объекта. Каждое состояние имеет логику переходов с соответствующими проверками. Например, если игрок находится в состоянии «В тупике» и обнаружил врага, и здоровье игрока больше 50%, то следующим состоянием будет «Атакую».

Такой подход к реализации принятия решений игровыми объектами имеет ряд недостатков:

* Расширяемость: конечный автомат с большим количеством состояний теряет преимущество графического представления, со временем такое поведение станет невозможно понять.
* Изменяемость: при добавлении/удалении поведения (состояния) необходимо изменить все другие состояние, которые связаны с новым/старым поведением; большие изменения могут приводить к ошибкам логики поведения объекта в целом.
* Распараллеливание: запускать состояния автомата параллельно не представляется возможным.

В 2005 году был создан более эффективный способ принятия решения игровыми объектами по сравнению с конечными автоматами – деревья поведения.

Дерево поведения – это направленный связный ациклический[[3]](#footnote-3) граф, имеющий единственную вершину, в которую не входят ребра – корень дерева. Из пары вершин, соединенных ребром, та, из которой выходит ребро, называется родительской вершиной, а другая дочерней вершиной. Вершина, не имеющая дочерних, называются листом. Каждое поддерево дерева поведения определяет различное поведение. Вершины, находящиеся между корнем дерева и листами могут быть двух типов – декораторами или композитами. Корень дерева поведения периодично генерирует сигнал, который передает дочерним вершинам, заставляя их выполнять алгоритм, определенный типом вершины. Как только сигнал достигнет листа, то лист произведет некоторые вычисления и вернет одно из 4 состояний: «успешно» (success), «не успешно» (failure), «запущено» (running), «ошибка» (error). Возвращенное состояние передастся родительским вершинам, для принятия решений в соответствии с типом вершины. Процесс закончится тогда, когда корневая вершина вернет некоторое состояние.

## Типы вершин

Все вершины дерева поведения одного из трех типов: декоратор, композит, лист.

## Лист

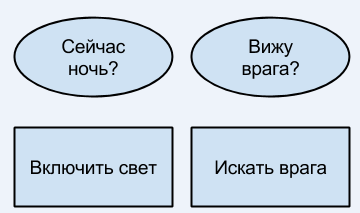
Лист – это не делимая часть дерева поведения. Эти вершины не имеют дочерних вершин, они принимают сигнал, производят некоторые вычисления и возвращают результат родительской вершине. 

Рисунок 1.2 - Графическое представление листовых вершин

Существует два вида листовых вершин: лист-условие и лист-действие. Лист-условие выполняет проверку некоторого условия и возвращает соответствующий результат (успешно, не успешно или ошибка). Лист-действие выполняет действие и возвращает результат успешно, запущено или ошибка.

Графически лист-условие изображается овалом, лист-действие прямоугольником (рисунок 1.2.1.1).

## Композит

Композит имеет одну или больше дочерних вершин. Он принимает и передает сигнал дочерним вершинам в некотором порядке, и также решает какое и когда вернуть состояние. Композит всегда возвращает одно из трех состояний: «успешно», «не успешно» или «ошибка». Все композитные вершины изображаются в виде квадрата со специальным символом внутри.

Существует три вида композитной вершины: композит-селектор, композит-последовательность и параллельный композит. Селектор обрабатывает дочерние вершины до тех пор, пока дочерняя вершина возвращает результат «не успешно», затем пробрасывает полученный результат родительской вершине и заканчивает выполнение. Если все дочерние вершины вернули результат «не успешно», то селектор вернет результат «не успешно». Специальный символ для селектора – знак вопроса.

Последовательность обрабатывает дочерние вершины до тех пор, пока они возвращают результат «успешно», затем пробрасывает полученный результат родительской вершине и заканчивает выполнение. Если все дочерние вершины вернули результат «успешно», то последовательность вернет результат «успешно». Специальный символ для последовательности – стрелка вправо.

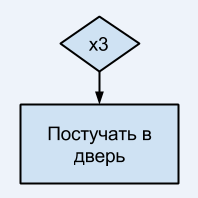
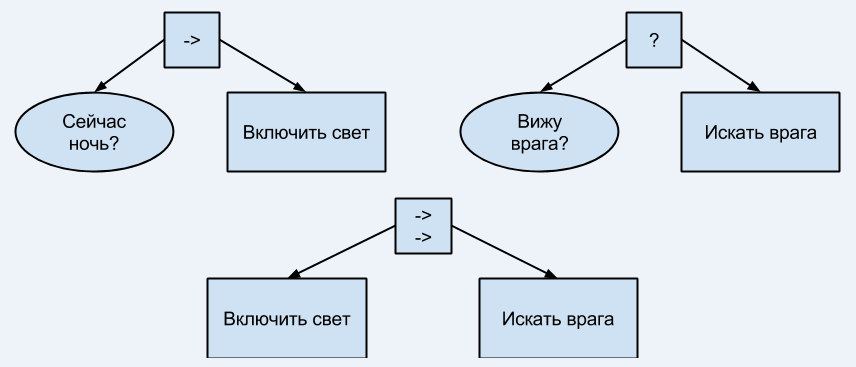
Параллельный композит обрабатывает все вершины одновременно, возвращает «успешно», если количество дочерних вершин с результатом «успешно» превышает некоторую константу S (которая может быть различна для разных параллельных композитов), возвращает результат «не успешно», если количество дочерних вершин с результатом «не успешно» превышает некоторую константу F, которая так же может быть определена для конкретного параллельного композита, иначе возвращает результат «запущено». Специальный символ для параллельного композита – две стрелки вправо.

Рисунок 1.4 – Пример вершины-декоратора

Рисунок 1.3 – Примеры композитных вершин

## Декоратор

Декоратор – это специальная вершина, которая имеет ровно одну дочернюю вершину. Цель, которую преследует декоратор, - изменить возвращаемое дочерней вершиной значение, или повлиять на частоту передаваемого сигнала дочерней вершины. Например, декоратор может делать инверсию возвращаемого значения, а может повторить сигнал, передаваемый дочерней вершине 3 раза. Декоратор изображается в виде ромба с пояснением внутри.

1. Раздел 2

## Подраздел 1

## Подраздел 2

Для точного определения числа математики используют формулу Валлиса:

|  |  |
| --- | --- |
| , | (3.2) |

где n – целое число.

Сам Валлис пришел к этой формуле, вычисляя площадь круга. Это произведение сходится крайне медленно, поэтому для практического вычисления числа формула Валлиса не пригодна. Однако она полезна в теоритических исследованиях, например при выводе формулы Муавра – Стирлинга.

Заключение

Заключение должно содержать краткую характеристику результатов выполненной работы (результатов решения поставленных задач), и рекомендации по улучшению системы.

Список использованных источников

http://www.json.ru/markets\_research/new\_research/

[https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%B4%D1%83%D1%81%D1%82%D1%80%D0%B8%D1%8F\_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D1%8B%D1%85\_%D0%B8%D0%B3%D1%80](https://ru.wikipedia.org/wiki/Индустрия_компьютерных_игр)

<http://www.lwjgl.org/>

<http://box2d.org/>



**Задание на выполнение бакалаврской работы**

Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

«Алтайский государственный технический университет им. И. И. Ползунова»

УТВЕРЖДАЮ

Заведующий кафедрой ПМ

\_\_\_\_\_\_\_\_\_ Кантор С.А.

*подпись ФИО*

**ЗАДАНИЕ №**

**НА ВЫПОЛНЕНИЕ БАКАЛАВРСКОЙ РАБОТЫ**

по направлению подготовки\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

по профилю\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

студенту группы Никитину Алексею Александровичу

*фамилия, имя, отчество*

**Тема**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Утверждена приказом ректора от \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_№ \_\_\_\_\_\_\_

Срок выполнения работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Задание принял к исполнению: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*подпись ФИО*

Барнаул 2014 г.

**1 Исходные данные**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**2. Содержание разделов работы**

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование разделов  работы и их содержание | Трудо-ёмкость,  % | Срок  выполнения | Консультант  (Ф.И.О., подпись) |
| 1 Расчетно-пояснительная записка |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 2 Графическая часть |  |  |  |
|  |  |  |  |
|  |  |  |  |

**3. Научно-библиографический поиск**

3.1. По научно-технической литературе просмотреть Реферативные журналы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ за последние \_\_\_\_ года и научно-технические журналы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ за последние \_\_\_\_ года.

3.2. По нормативной литературе просмотреть указатели государственных и отраслевых стандартов за последний год.

3.3. Патентный поиск провести за \_\_\_\_ лет по странам \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель работы: Старолетов С.М.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Ф.И.О. подпись*



**Руководство пользователя**



**Исходный код программы**

1. Объемы рынка игр

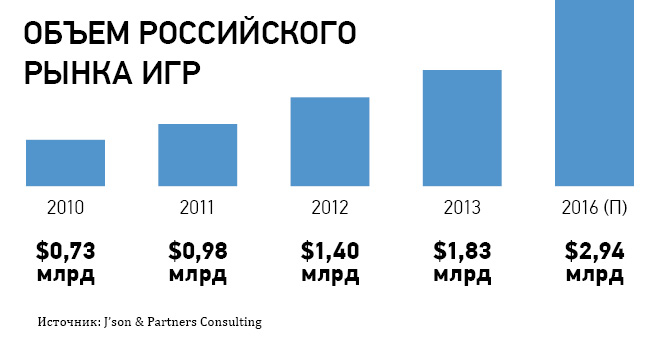
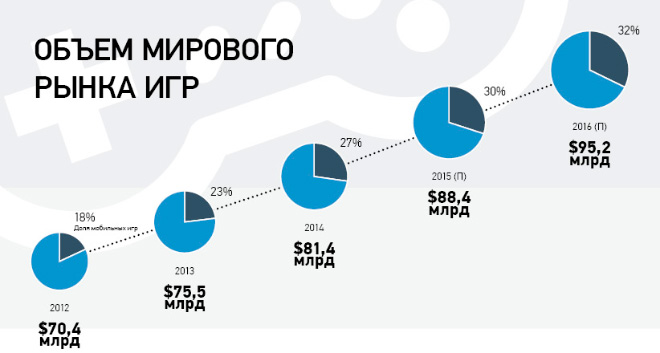
По оценке J’son & Partenrs Consulting, в 2015 году объем мирового рынка игр составит 88,4 миллиарда долларов, а объем российского рынка игр составит 1,83 миллиарда долларов. В целом рынок игр будет стабильно развиваться.

Рисунок Г.1 - Объем мирового рынка игр

Рисунок Г.2 - Объем российского рынка игр

1. Деревья поведения подробно описаны в главе 1. [↑](#footnote-ref-1)
2. Несущественные объекты – это игровые объекты, такие как массовка, животные, птицы, исключая ситуации, когда именно эти персонажи – главные герои в играх. Поведение несущественных объектов никак не влияет на ход игры. [↑](#footnote-ref-2)
3. В данном случае дерево поведения не должно содержать циклов даже если не учитывать направления ребер. [↑](#footnote-ref-3)