МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет»
(национальный исследовательский университет)
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

РАБОТА ПРОВЕРЕНА

Рецензент начальник отдела
поддержки и обучения
пользователей ЛСМ ЮУрГУ

_____ Ф.М. Мелехин

“___”_____ 2017 г.

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой,
д.ф.-м.н., профессор

_____ Л.Б. Соколинский

“___”_____ 2017 г.

РАЗРАБОТКА ИГРОВОГО ПРИЛОЖЕНИЯ С НЕПРЯМЫМ УПРАВЛЕНИЕМ В ЖАНРЕ ТАКТИЧЕСКОЙ СТРАТЕГИИ

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 02.03.02.2017.308-066.ВКР

Научный руководитель
канд. техн. наук, доцент кафедры СП
_____Н.Ю. Долганина

Автор работы,
студент группы КЭ-402
_____А.А. Веретенников

Ученый секретарь
(нормоконтролер)

_____ О.Н. Иванова

“___”_____ 2017 г.

Челябинск-2017

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность темы

Игровая индустрия является одним из самых интенсивно развивающихся направлений рынка развлечений [1]. Современная игровая индустрия по показателю прибыльности намного опережает музыкальную и киноиндустрию. В период с 2010 до 2016 года доходы от продаж видеоигр выросли на 40 %. В 2015 году выручка от продаж традиционных видеоигр в России достигла 695 млн. долл. США, а в 2020 году ожидается, что данный показатель составит 842 млн. долл. США при прогнозируемых среднегодовых темпах роста на уровне 3,9 % [2].

Помимо прибыли данной индустрии, растет как количество игроков, так и их средний возраст. По данным исследования игрового рынка России, проведенного компанией Mail.ru Group, 87 % интернет аудитории нашей страны играет в игры чаще одного раза в месяц [4]. Также стоит обратить внимание на исследование CEO агентства Insight One, которое показало, что видеоигры популярны среди всех категорий населения. По его данным, средний возраст игроков в России составляет 30 лет, при этом 68 % из них – взрослые люди (старше 18 лет); 45 % игроков – женщины, что опровергает мнение о том, что геймерами в основном являются мужчины; 45 % игроков состоят в браке, а 40 % геймеров являются родителями, что противоречит стереотипу о том, что игры мешают развитию человеческих отношений [3].

Еще одним важным шагом в развитии игровой индустрии стало внесение Министерством спорта России киберспорта во второй раздел реестра (включает в себя виды спорта, развиваемые на общероссийском уровне, за которые даются разряды и звания) официально признанных видов спорта. Это показывает, что людям интересно не только играть, но и наблюдать за игрой других. К примеру, за крупными соревнованиями по таким играм, как Dota 2, League of Legends и Counter Strike наблюдают порядка 2 млн. зрителей.

Цель и задачи

Целью работы является разработка игрового приложения с непрямым управлением в жанре тактической стратегии.

Для достижения данной цели должны быть решены следующие задачи.

1. Исследовать существующие игровые приложения с непрямым управлением.

2. Спроектировать игровое приложение с непрямым управлением в жанре тактической стратегии.

3. Реализовать и протестировать игровое приложение.

Структура и объем работы

Работа состоит из введения, четырех глав, заключения, библиографического списка и двух приложений. Объем работы без приложений составляет 45 страниц, объем приложений составляет 7 страниц, объем библиографии – 27 источников.

Содержание работы

В первой главе «Анализ предметной области» проведен обзор программных средств разработки компьютерных игр и обзор существующих игровых приложений с непрямым управлением.

Вторая глава «Проектирование игрового приложения» посвящена определению требований к разрабатываемому игровому приложению. Также в этой главе рассматривается диаграмма вариантов использования и диаграмма компонентов.

В третьей главе «Реализация игрового приложения» описаны методы реализации, приведены скрипты приложения. В этой же главе рассмотрена файловая структура игрового приложения.

В четвертой главе «Тестирование игрового приложения» представлены результаты тестирования игрового приложения.

В заключении описываются основные результаты, полученные при выполнении работы.

В приложении 1 представлены скриншоты работающего игрового приложения.

В приложении 2 представлен отчет юзабилити-тестирования.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Обзор программных средств разработки компьютерных игр

Игровой движок – центральный программный компонент компьютерных и видеоигр или других интерактивных приложений с графикой, обрабатываемой в реальном времени. Он предоставляет основные технологии, упрощает разработку и обеспечивает игре кроссплатформенность – возможность работать более чем на одной аппаратной платформе или операционной системе. Как правило, основную функциональность обеспечивает игровой движок, включающий движок рендеринга, физический движок, звук, систему скриптов, искусственный интеллект, анимацию, сетевой код, управление памятью и многопоточность, другими словами, игровым движком является набор подсистем, которые упрощают наиболее часто используемые функции игр.

В сфере разработки игровых приложений наиболее мощными и вместе с тем доступными инструментами являются платформы Unity, Unreal Engine и Cry Engine.

Unity – кросс-платформенный игровой движок для разработки двухмерных и трехмерных приложений и игр под различные платформы, работающий под операционными системами Windows, Linux и OS X [7]. Unity играет важную роль на стремительно растущем рынке игр. Созданные на базе Unity приложения работают под операционными системами Windows, OS X, Windows Phone, Android, Apple iOS, Linux, а также на игровых приставках Wii, Playstation 3, Playstation 4, Xbox 360, Xbox One. Кроме того Unity является самой широко распространенной платформой разработки для VR (Virtual Reality), поддержка доступна для Oculus Rift, Gear VR, Playstation VR, Microsoft Hoto Lens, Stream VR/Vive и Google DayDream. На базе Unity сделано больше игр, чем с любой другой игровой технологией, в них играют все больше и больше игроков, а инструменты и сервисы Unity предпочитают все больше и больше разработчиков: 34 % из топ 1000 бесплатных мобильных игр разработаны на Unity, среди клиентов Unity

есть Coca-Cola, Disney, Electronic Arts, LEGO, Microsoft, NASA, Nexon, Nickelodeon, Square Enix, Ubisoft, Obsidian, Insomniac и Warner Bros. В игры, созданные с использованием движка Unity играют более 770 млн. пользователей по всему миру [9].

Технологии Unity позволяют создавать невероятно красивые и завораживающие игры и приложения с двухмерной и трехмерной графикой, а также виртуальной и дополненной реальностью, благодаря мощному графическому движку и полнофункциональному редактору. Unity имеет очень удобный Drag and Drop интерфейс. Движок поддерживает два языка программирования: C# и JavaScript. Расчеты физики в Unity 5 производит NVIDIA PhysX. Кроме того, движок имеет две очень важные особенности: Occlusion Culling и Level of Detail, которые позволяют сильно снизить нагрузку на центральный процессор, благодаря грамотной детализации, что позволяет выставить большие значения в детализации игры.

С помощью технологий Unity были созданы такие игры как Drakensang Online, Пароград, Lara Croft: Relic Pun, Lara Croft Go, Slender: The Arrival, Manifold Garden, Forgotton Anne, StarCrawlers, Rust и Hearthstone.

Достоинства: кроссплатформенность, удобный редактор, широкое сообщество разработчиков, регулярное обновление, поддерживает несколько языков программирования.

Недостатки: условно-бесплатный (в бесплатной версии доступна только часть возможностей).

Unreal Engine [10] – написанный на языке программирования C++, движок позволяет создавать игры для большинства операционных систем и платформ: Microsoft Windows, Linux, OS X; консолей Xbox, Xbox 360, Xbox One, Playstation 2, Playstation 3, Playstation 4, PSP, PS Vita, Wii, Dreamcast, CameCube, а также для различных портативных устройств. Для упрощения портирования движок использует модульную систему зависимых компонентов; поддерживает системы рендеринга: Direct3D, OpenGL и Pixomatic, воспроизведения звука: EAX, OpenAL, DirectSound3D, средства

голосового воспроизведения текста, распознавания речи, модули для работы с сетью и поддержки различных устройств ввода.

С марта 2015 года платформа Unreal Engine стала полностью бесплатной для использования, при условии, что прибыль от приложений, созданных с использованием движка, не превышает 3000$ за квартал, иначе необходимо будет отчислять 5 % о прибыли за данный промежуток времени.

На движке Unreal Engine были созданы Final Fantasy 7 Remake и Tekken 7.

Достоинства: кроссплатформенность, возможность создания алгоритмов внутри графического редактора, широкое сообщество разработчиков, регулярное обновление.

Недостатки: высокие аппаратные требования, неудобен для освоения, поддерживает лишь один язык программирования – C++.

Cry Engine [8] – игровой движок, разработанный немецкой компанией с ограниченной ответственностью Crytek. CryEngine 3 изначально является кроссплатформенным движком, ориентирован на IBM PC-совместимые компьютеры и игровые консоли Microsoft Xbox 360 и Sony PlayStation 3, а также на их последующие версии. Кроме того, движок обеспечивает также разработку массовых многопользовательских онлайновых игр. Движок CryEngine 3 является коммерческим, но есть и Free SDK версия.

Cry Engine один из самых популярных игровых движков, о чем свидетельствует количество известных игр, созданных на нем: Giant, Sniper II: Ghost Warrior, Cabal II, Far Cry, Ryse: Son of Rome и Aion: Tower of Eternity.

Достоинства: кроссплатформенность.

Недостатки: коммерческая лицензия, высокие аппаратные требования, неудобен для освоения, необходимо знание сразу нескольких языков программирования.

В качестве игрового движка был выбран Unity, благодаря наличию бесплатной версии, удобному интерфейсу и поддержке языка программирования C#, который было решено использовать для разработки игрового приложения [18, 23, 26].

1.2. Обзор существующих решений

Тактическая стратегия – популярный жанр видеоигр, в котором залогом достижения победы является планирование и стратегическое мышление. Смысл таких игр заключается в управлении определенными ресурсами, которые необходимо преобразовать в преимущество над противником при помощи оперативного плана, разрабатываемого с учетом меняющейся обстановки.

Непрямое управление не позволяет игроку отдавать прямые и точные указания игровым юнитам и изменяет перспективу взаимодействия игрок-мир в совершенно иное русло – косвенный контроль над ситуацией. Как правило, в играх с непрямым управлением пользователь может лишь выбирать расположение юнитов и задавать общее направление движения, все остальные действия юниты совершают самостоятельно и игрок не способен на них повлиять.

Одной из популярных серий игр в жанре стратегия с непрямым управлением, является ***The Settlers***, разработанная немецкой компанией Blue Byte Software (рис. 1) [15].

В данной серии игроку предстоит пройти путь развития своей империи. Действия игрока ограничены распоряжением ресурсами, которые он может тратить на расстановку зданий и производство боевых юнитов, и указанием цели движения боевым единицам. Добыча и переработка ресурсов, порядок строительства зданий, боевые действия происходят автоматически.

К плюсам данной серии игр можно отнести интересный сюжет, нестандартный игровой процесс, развивающий навыки планирования – некоторые виды ресурсов получаются путем объединения других ресурсов, что

стимулирует игрока к грамотному планированию расстановки зданий, для увеличения их эффективности. Минусами является медленное развитие игры – это потребует от игрока много времени и терпения, чтобы пройти игру, помимо этого, на каждом уровне игроку приходится выполнять однотипные действия.

Рис. 1. Скриншот из игры The Settlers 4

Slaves to Armok II: Dwarf Fortress – компьютерная игра, сочетающая в себе элементы экономической стратегии и элементы симулятора Бога с непрямым управлением, разработкой которой занимается американский программист Тарн Адамс (рис. 2) [14].

Игроку предстоит взять контроль над развитием поселения дворфов. Игрок может указать область для строительства пещер, добычи ресурсов, экипировать отряд и укомплектовать начальный отряд, однако дворфы управляются искусственным интеллектом и сами решают, что им делать.

Помимо задач, поставленных игроком, дворфы могут заниматься удовлетворением своих потребностей. К плюсам можно отнести интерес повторного прохождения игры за счет случайно генерируемых карт и событий и глубокую проработку игровых механик. Основным минусом игры является примитивная графика и сложность освоения в игре, в виду отсутствия хоть какого-то обучения.

Рис. 2. Скриншот из игры Slaves to Armok II: Dwarf Fortress

Еще одним ярким представителем стратегий с непрямым управлением является серия игр **Majesty**, разработанная Cyberlore Studios (рис. 3) [12].

Действие в серии игр Majesty происходит в стране Ардания, где игрок – суверен этой страны. Игровой процесс включает в себя строительство и наем героев для создания устойчивого королевства и выполнения целей сценария. Строительство зданий, наем героев и их улучшение не отличается от других RTS, однако юниты являются автономными единицами. У каждого из героев свой характер и манера поведения. Например, лучник предпочитает находиться вдали от цивилизации и охотится, может самостоятельно уйти в лес, найти стаю волков, перестрелять ее и вернуть-

ся, при этом игнорируя приказы игрока об атаке или защите. Отличный и необычный игровой процесс, изумительная атмосфера – вот главные плюс этой игровой серии. К минусам можно отнести затянутость миссий и большое количество программных недоработок.

Рис. 3. Скриншот из игры Majesty 2

Plants vs Zombies **–** двумерная компьютерная стратегическая игра с непрямым управлением, относящаяся к поджанру Tower Defense, разработанная и изданная компанией PopCap Games (рис. 4) [13].

Игрок защищает свой дом от атаки зомби, размещая на газоне рядом с домом различные растения. Для выращивания растений требуется определенное количество ресурса «солнце», производимого специальными видами растений, а также падающего с неба в светлое время суток. Другие растения используются для атаки против определенных видов зомби, для замедления продвижения противника и других целей. Виды зомби отличаются скоростью передвижения, возможностью преодоления разных препятствий и защитой от определенных типов атак. Действие основного режима игры происходит на трех локациях: на газоне перед домом, на заднем

дворе и на крыше дома. Задний двор отличается наличием бассейна, в котором могут расти дополнительные виды растений. Крыша отличается наклоном, не позволяющим использовать стреляющие по прямой растения на скате крыши. Помимо этого, действие может происходить днем и ночью, а также в тумане, что влияет на набор доступных для использования растений.

Рис. 4. Скриншот из игры Plants vs Zombies

При всей своей простоте – игра затягивает надолго, что, пожалуй, и является ее главным плюсом. К минусам можно отнести только однотипность игрового процесса.

Еще одна известная игра поджанра Tower Defense – ***King Rush***, разработанная Ironhide Game Studio и изданная Armor Games (рис. 5) [11].

Рис. 5. Скриншот из игры King Rush

Действие игры разворачивается в средневековом мире. Вдоль заранее установленного пути, по которому идет соперник, есть пустые слоты, где игрок может строить башни. В игре существует четыре типа башен – магическая, с лучниками, казармы и артиллерия, а также два типа заклинаний – вызов подкрепления и огненный дождь. В начале каждого уровня у игрока есть определенная сумма денег, чтобы купить первые башни. Цель игрока – уничтожить всех врагов до того, как они пройдут до конца, отмеченного синим щитом. Убивая врагов, игрок получает дополнительные деньги для постройки новых башен и улучшения уже имеющихся. За прохождение каждого уровня игрок получает звезды, которые можно использовать для покупки улучшений для башен и заклинаний. Плюсов у игры много: выверенная боевая механика, сюжетная подача, система героев и улучшений. К минусам можно отнести отсутствие русификации и сравнительно небольшое количество уровней.

2. ПРОЕКТИРОВАНИЕ ИГРОВОГО ПРИЛОЖЕНИЯ

2.1. Эскизный проект игрового приложения

Концепция игры

Robot Colony – игровое приложение, разрабатываемое в данной работе, представляет собой тактическую стратегию с непрямым управлением и относится к поджанру Tower Defense, для персональных компьютеров. Игроку предстоит защитить космическую базу роботов от вторжения пришельцев. Игровой процесс не требует от игрока большего количества действий, однако игрок должен тщательно просчитывать каждое из них [24].

Цели игры

Целью является прохождение двух уровней игры. На каждом уровне игроку необходимо уничтожить всех пришельцев, атакующих его базу.

Правила игры

В начале каждого уровня игроку дается определенное количество ресурсов, используя которые, он может расставлять на игровое поле роботов. Роботы могут добывать энергию для создания новых юнитов, либо атаковать пришельцев. Пришельцы движутся к базе игрока справа налево и могут атаковать роботов. Если пришелец дойдет до базы – уровень будет проигран, при уничтожении всех пришельцев игрок проходит уровень.

Игровые возможности

Игрок может собирать энергию, произведенную роботами, выбирать, какого робота поставить на поле боя, а также решать, куда размещать, выбранного робота.

Игроки

Игра предназначена для одного игрока. Многопользовательский режим не предусмотрен.

Атмосфера игры

Основной экран игры представляет собой двумерное изображение боевого поля, разбитого на клетки. В левой части экрана находится база

игрока. Камера не перемещается. Графическая составляющая реализуется с помощью анимированных спрайтов [16, 21].

Интерфейс

Интерфейс приложения включает в себя главное меню, игровые меню 1-3 и игровую панель.

Главное меню содержит кнопки «Start game» и «Quit». При нажатии кнопки «Start game» загружается первый уровень, а при нажатии «Quit» происходит выход из игры.

Игровое меню «1» открывается при нажатии клавиши «Esc» на сценах с игровыми уровнями, при этом игра будет поставлена на паузу. Игровое меню «1» содержит кнопки «Resume», «Restart» и «Exit to menu». При нажатии на кнопку «Resume» или при повторном нажатии клавиши «Esc» игра будет продолжена на том же моменте, на котором была остановлена. При нажатии на кнопку «Restart» сцена с текущим уровнем игры будет перезагружена, а при нажатии на «Exit to menu» игровой процесс прерывается и происходит возврат к главному меню.

Игровое меню «2» появляется, в случае поражения игрока и содержит кнопки «Restart Level» и «Exit to menu». При нажатии кнопки «Restart Level» – уровень будет перезапущен, а при нажатии «Exit to menu» игровой процесс прерывается и происходит возврат к главному меню.

Игровое меню «3» появляется, в случае прохождения уровня и содержит кнопки «Next Level» и «Exit to menu». При нажатии кнопки «Next Level» загружается следующий уровень, а при нажатии «Exit to menu» произойдет возврат к главному меню.

Игровая панель находится в верхней части экрана и включает в себя несколько объектов: счетчик энергии отображает, сколько энергии имеется у игрока; панель выбора роботов позволяет игроку выбирать, какого робота разместить на игровом поле; счетчик пришельцев показывает количество пришельцев, которое необходимо уничтожить для прохождения уровня.

Объекты игры

Объекты игры делятся на роботов, пришельцев и другие объекты, в которые входят стены, пули и энергия. Роботы в свою очередь делятся на добывающих ресурсы и обороняющих базу. Они отличаются внешним видом, количеством здоровья, стоимостью в ресурсах и частотой атаки. Пришельцы различаются внешним видом и показателями здоровья и скорости перемещения.

Игровые уровни

На первом уровне игроку доступен выбор из двух роботов – один способен добывать ресурсы, второй оборонять базу. Пришельцы также представлены двумя видами.

На втором уровне игроку открывается доступ к еще одному обороняющемуся роботу, также появляется новый вид пришельцев.

Кроме того, уровни различаются общим количеством пришельцев, которых необходимо уничтожить.

2.2. Диаграмма вариантов использования

Диаграмма вариантов использования приведена на рис. 6.

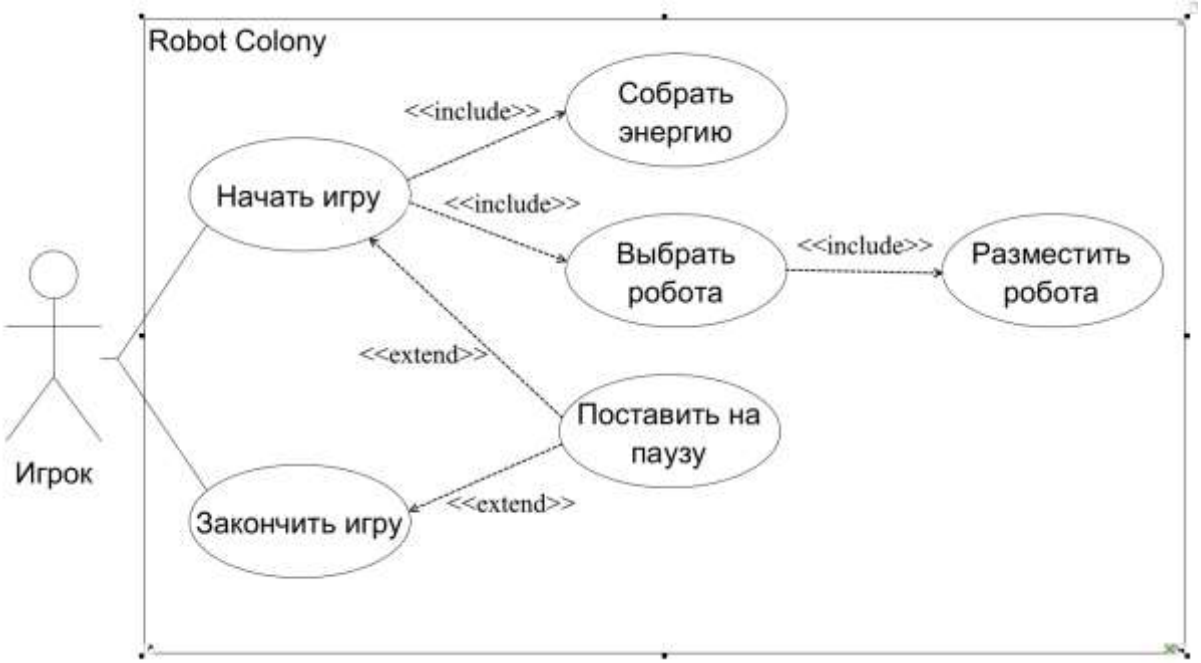

Рис. 6. Диаграмма вариантов использования

Для проектирования приложения был использован язык графического описания для объектного моделирования UML [17]. Была построена модель взаимодействия внешнего актера с игровым приложением в виде диаграммы вариантов использования.

В ходе проектирования был выделен актер ***игрок***. Игрок – неавторизованный пользователь приложения.

Начать игру – приступить к игровому процессу. Реализуется с помощью кнопки «Start game» главного меню, «Resume» игрового меню «1», «Next Level» игрового меню «3», «Restart» игровых меню «1», «2» и «3».

Собрать энергию – нажать левой кнопкой мыши на изображение энергии, вырабатываемой генератором.

Выбрать робота – нажать на изображения робота на игровой панели левой кнопкой мыши.

Разместить робота – нажать на клетке игрового поля левой кнопкой мыши после выбора робота.

Поставить на паузу – нажать клавишу «Esc» на сценах с уровнями «1» и «2».

Закончить игру – закончить игровой процесс. Реализуется с помощью кнопки «Quit» главного меню кнопки «Exit to menu» игровых меню «1», «2» и «3».

2.3. Диаграмма компонентов

Приложение состоит из четырех основных логических блоков.

1. User Scripts.
2. Menu.
3. Resources.
4. Levels.

Компонент User Scripts представлен в виде ряда артефактов – файлов с расширением *.cs, хранящих описание поведения приложения и игровых объектов, основными из которых являются: «GameController.cs», «SpawnRobot.cs», «RobotType.cs», «Allien.cs», «Projectile.cs» и «SunCollect.cs».

Компонент Menu содержит элементы главного меню игры и игровых меню.

Компонент Resources содержит файлы со шрифтами и графическим оформлением сцен.

Компонент Levels материализован в форме ряда артефактов – систем объектов, каждая из которых представляет собой отдельный уровень.

Диаграмма компонентов представлена на рис. 7.

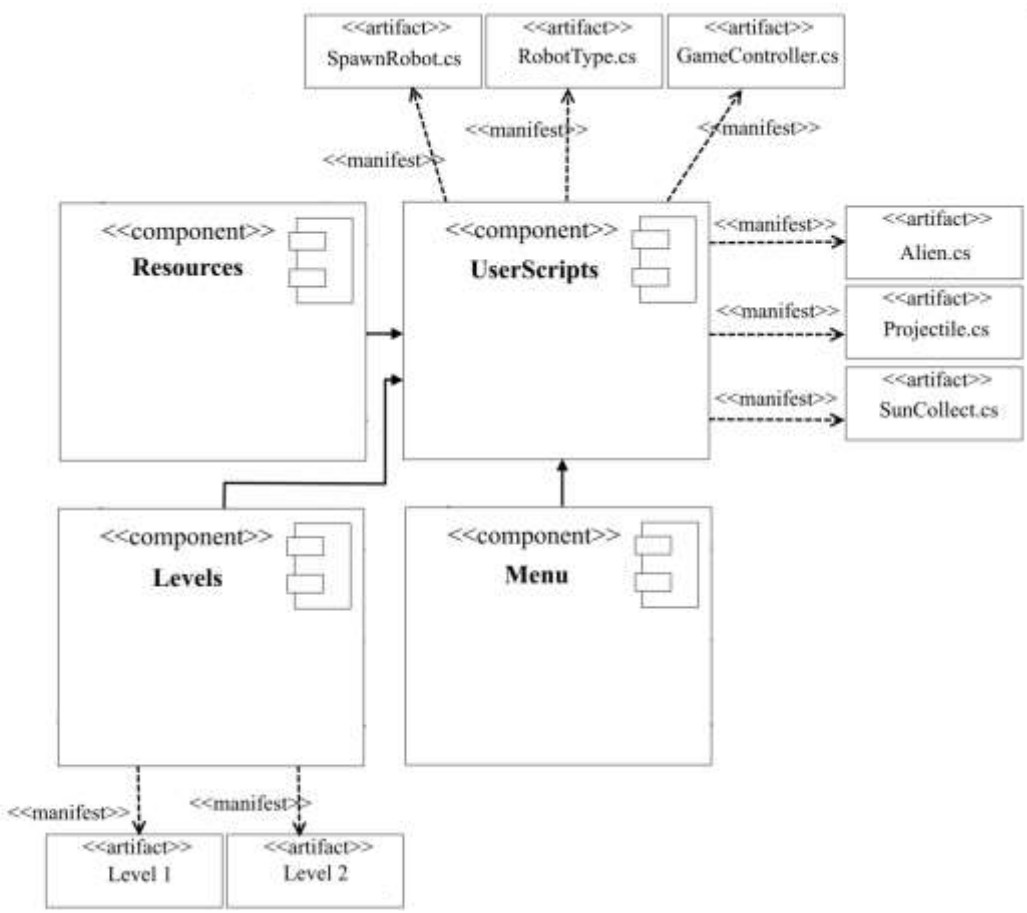

Рис. 7. Диаграмма компонентов

3. РЕАЛИЗАЦИЯ ИГРОВОГО ПРИЛОЖЕНИЯ

3.1. Файловая структура приложения

Разработанный проект включает в себя ряд каталогов, содержащих скрипты; шаблоны игровых объектов; графические файлы с изображениями спрайтов, фонов и элементов интерфейса; шрифты, используемые в игре; анимации игровых объектов; сцены. Файловая структура приложения представлена на рис. 8.

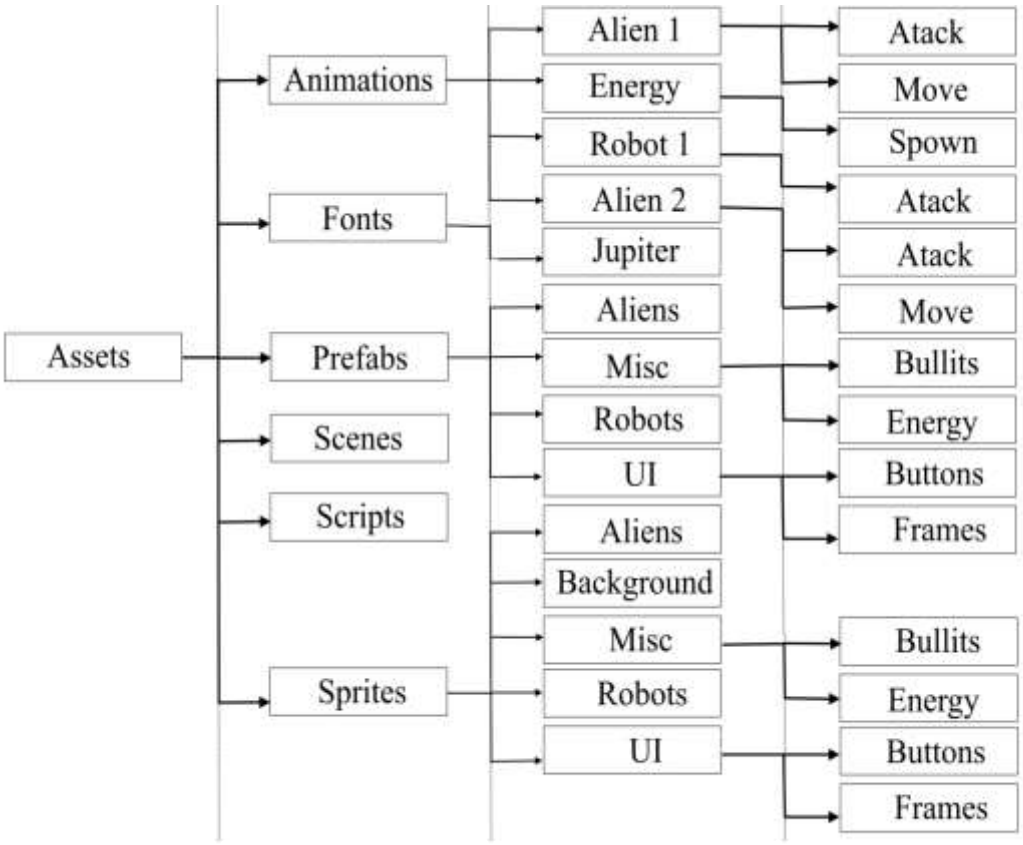

Рис. 8. Файловая структура приложения

3.2. Интерфейс

Главное меню

Главное меню является отдельной сценой и содержит кнопки «Start game» и «Quit» (рис. 1 приложения 1). При нажатии на кнопку «Start game» загружается сцена с первым уровнем, а при нажатии на кнопку«Quit» происходит выход из игры.

Игровые меню

Игровые меню является объектами сцен уровней игры и реализованы в виде объектов типа Canvas. Компонент Canvas представляет собой абстрактное пространство, в котором производится настройка и отрисовка UI (User Interface). Все UI-элементы должны быть потомками игровых объектов, к которым присоединен Canvas. Игровые меню «1», « 2» и «3» представлены на рис. 2-4 приложения 1. Скрипт, используемый игровыми меню, изображен на рис. 9.

```
public class InGameMenu : MonoBehaviour
{
    public GameObject gc;
    public void ResumeToGame()
    {
      GameObject.Find("InGameMenu").SetActive(false);
      GameOject.Find("GameController").GetComponent<GameController>().Pause
      Unpause();
    }
    public void RestartLevel()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().name);
        Time.timeScale = 1;
    }
    public void ExitToMenu()
    {
        SceneManager.LoadScene("MainMenu");
    }
    public void NextLevel(string levelname)
    {
        SceneManager.LoadScene(levelname);
    }
}
```

Рис. 9. Скрипт игровых меню

Кнопка «Resume» соответствует вызову процедуры «ResumeToGame» и отвечает за продолжение игрового процесса. Кнопка «Exit to menu» вызывает процедуру «ExittoMenu», что приводит к загрузке сцены «Главное меню». Кнопка «Restart», присутствующая на игровых меню «1» и «2», при нажатии обращается к процедуре «Restartlevel». Эта процедура заново загружает активную сцену. Кнопка «Next Level» игрового меню «3» связана с процедурой «NextLevel», в которой происходит загрузка сцены следующего уровня.

Игровая панель

Игровая панель представляет собой совокупность нескольких объектов: счетчиков энергии и инопланетян, которые состоят из спрайта и объекта GUI Text; панели выбора юнитов, представляющую собой набор объектов, реализованных в виде префабов и использующих скрипт «Robot Type», представленный на рис. 10. Данный скрипт отвечает за выбор роботов для размещения и хранит цену роботов. Игровая панель представлена на рис. 5 приложения 1.

```
public class RobotType : MonoBehaviour {
    public GameObject robot;
    public int price = 0;
    public Sprite nonSelected;
    public Sprite Selected;

    private void OnMouseDown()
    {
        if (GameObject.Find("Main Camera").GetComponent<SelectRobot>().Robot
!= gameObject)
        {
            if(GameObject.Find("Main Cam-
era").GetComponent<SelectRobot>().Robot != null)
            {
              GameObject.Find("Main Cam-
era").GetComponent<SelectRobot>().Robot.GetComponent<RobotType>().Deselect();
            }
            GameObject.Find("Main Camera").GetComponent<SelectRobot>().Robot
= gameObject;
            GetComponent<SpriteRenderer>().sprite = Selected;
        }
        else
        {
            GameObject.Find("Main Cam-
era").GetComponent<SelectRobot>().Robot.GetComponent<RobotType>().Deselect();
            GameObject.Find("Main Camera").GetComponent<SelectRobot>().Robot
= null;
        }
    }
    public void Deselect()
    {
        GetComponent<SpriteRenderer>().sprite = nonSelected;
    }
}
```

Рис. 10. Скрипт «Robot Type»

В процедуре «OnMouseDown» обрабатываются действия, происходящие при нажатии на объект. В первую очередь, происходит проверка: был ли уже выбран данный тип робота. Если это так, то изображение объекта меняется с «выделенного» на «невыделенное» и происходит удаление

типа робота из камеры. В противном случае, просматривается, выбран ли в данный момент хоть какой-нибудь робот. Если это так, то выделение у выбранного в данный момент объекта снимается (меняется изображение) и присваивается значение объекта, вызвавшего эту процедуру. Изображение данного объекта меняется на «выделенный». Процедура «Deselect» снимает выделение объекта.

3.3. Игровые уровни

Каждый уровень представляет собой сцену с игровыми объектами [20]. Объекты реализованы в виде префабов, которые состоят из спрайта и набора скриптов, отвечающих за их поведение. Префаб – это заранее созданный прототип объекта, который служит для создания экземпляров. Таким образом, для изменения группы объектов на всех игровых сценах достаточно лишь изменить префаб [27]. Интерфейс игрового уровня изображен на рис. 6 приложения 1. Скрипт, осуществляющий постановку игры на паузу и переход между уровнями, представлен на рис. 11-12.

```
public class GameController : MonoBehaviour {
    public GameObject menu;
    public GameObject roboFrame;
    public GameObject ground;
    public GameObject dynamicObjects;
    public GameObject MonstroFrame;
    public GameObject Wall;
    public GameObject menu1;
    public GameObject aliens;
    public GameObject design;
    public bool endgame=false;
    public static int BegResources = 100;
    public static  int CurResources = 0;
    public GameObject txt;
    bool paused = false;
void Start ()
    {
        Time.timeScale = 1;
        CurResources = BegResources;
        txt.GetComponent<GUIText>().text = GameControl-
ler.CurResources.ToString();
    }
  void Update () {
        if (Input.GetKeyDown(KeyCode.Escape) && (endgame == false))
        {
            menu.SetActive(!menu.active);
            PauseUnpause();
        }
    }
```

Рис. 11. Скрипт «Game Controller»

```
    public void PauseUnpause()
    {
        if(!paused)
        {
            Time.timeScale = 0;
        }
        else
        {
            Time.timeScale = 1;
        }
        ground.SetActive(paused);
        roboFrame.SetActive(paused);
        dynamicObjects.SetActive(paused);
        MonstroFrame.SetActive(paused);
        Wall.SetActive(paused);
        aliens.SetActive(paused);
        design.SetActive(paused);
        paused = !paused;
    }
    public void nextLevDial()
    {
        menu1.SetActive(true) ;
        PauseUnpause();
    }
}
```

Рис. 12. Скрипт «Game Controller»

3.4. Игровые объекты

Игровые объекты делятся на игровое поле, роботов, пришельцев и другие (стены, энергия и пули) и реализованы в виде префабов [22].

Игровое поле состоит из отдельных клеток, содержащих скрипт, описывающий механику размещения роботов на клетках. На каждой клетке может быть размещен только один робот. Скрипт размещения роботов на игровом поле представлен на рис. 13-14.

```
public class SpawnRobot : MonoBehaviour {
    public GameObject Robot;
    GameObject txt;
    bool placed = false;
    private void OnMouseDown()
    {
        if (!placed)
        {
            txt = GameObject.Find("EnergyText");

            if (GameObject.Find("Main Cam-
era").GetComponent<SelectRobot>().Robot != null)
            {
                Robot = GameObject.Find("Main Cam-
era").GetComponent<SelectRobot>().Robot.GetComponent<RobotType>().robot;
```

Рис. 13. Скрипт размещения роботов на игровом поле

```
                if ((Robot != null) && (GameController.CurResources -
GameObject.Find("Main Cam-
era").GetComponent<SelectRobot>().Robot.GetComponent<RobotType>().price >=
0))
                {
                    GameController.CurResources -= GameObject.Find("Main
Camera").GetComponent<SelectRobot>().Robot.GetComponent<RobotType>().price;
                    GameObject.Find("Main Cam-
era").GetComponent<SelectRobot>().Robot.GetComponent<RobotType>().Deselect(
);
                    txt.GetComponent<GUIText>().text = GameControl-
ler.CurResources.ToString();
                    Instantiate(Robot,
                            transform.position + Ro-
bot.GetComponent<UnitOffset>().offset,
                            Quaternion.identity, GameOb-
ject.Find("DynamicObjects").transform );
                    GameObject.Find("Main Cam-
era").GetComponent<SelectRobot>().Robot = null;
                    placed = true;
                }
            }
        }
    }
}
```

Рис. 14. Скрипт размещения роботов на игровом поле

В процедуре «OnMouseDown» обрабатывается нажатие левой кнопки мыши на клетки игрового поля. В первую очередь происходит проверка, размещен ли робот на данной клетке, если клетка пустая, то выполняется проверка, выбран ли какой-нибудь робот на игровой панели и хватает ли очков энергии для его размещения. При соблюдении этих условий количество очков энергии уменьшается на стоимость данного робота, снимается «выделение» у соответствующей иконки игровой панели и на данной клетке создается объект, который соответствует типу выбранного робота.

Роботы делятся на виды: производящие энергию и роботы для обороны. Роботы различаются по показателям здоровья, количества пуль за один выстрел и скорости атаки. В случае, если здоровье робота становится равно нулю, робот уничтожается и удаляется с игрового поля. Роботы не способны перемещаться по игровому полю и занимают позицию, заданную игроком. Для выработки энергии и стрельбы используется один скрипт, представленный на рис. 15. Скрипт для сбора энергии представлен на рис. 16. Виды роботов описаны в табл. 1.

Табл. 1. Виды роботов

№	Изображение	Здоровье, условные единицы	Количество пуль за один выстрел	Скорость атаки	Описание робота
1		3	-	-	Energy Generator производит энергию, необходимую для покупки роботов
2		5	1	1 выстрел в 5 секунд	Shooting Drone – оборонительный робот
3		5	3	3 выстрела в 5 секунд	UpdTurret – оборонительный робот

```
public class SpawnMisc : MonoBehaviour
{
        public GameObject prefab;
        public int times = 1;
        public Vector3 offset;
        GameObject DynObj;
        void Start()
        {
             DynObj = GameObject.Find("DynamicObjects");
             for (int i = 0; i < times; i++)
             {
                InvokeRepeating("Spawn", 5 + 0.5f * i, 5 );
             }
        }
        void Spawn()
        {
        if (DynObj != null)
        {
            Instantiate(prefab,
                    transform.position + offset,
                    Quaternion.identity, GameOb-
ject.Find("DynamicObjects").transform);
        }
        }

}
```

Рис. 15. Скрипт создания других объектов

Во время выполнения процедуры «Start» (рис. 15) осуществляется поиск ссылки на объект «DynamicObjects», дочерними объектами которого, являются энергия и пули. Процедура «InvokeRepeating» вызывает процедуру «Spawn» необходимое количество раз с заданной периодичностью. Процедура «Spawn» отвечает за создание различных объектов (пуль или энергии). В ней происходит проверка на существование объекта «DynamicObjects» и размещение объекта, выбранного типа, на игровом поле.

```
public class SunCollect : MonoBehaviour
{
    public GameObject GameContr;
    GameObject txt;
    public float timer;
    private void Start()
    {
        if (timer>0)
        {
            Invoke("Disappear", timer);
        }

    }
    void OnMouseDown()
    {
        txt =  GameObject.Find("EnergyText");;
        GameController.CurResources += 20;
        txt.GetComponent<GUIText>().text = GameControl-
ler.CurResources.ToString();
        Destroy(gameObject);
    }
    void Disappear()
    {
        Destroy(gameObject);
    }
}
```

Рис. 16. Скрипт сбора энергии

Процедура «Start» (рис. 16) вызывает процедуру «Disappear» в указанное время, что приводит к уничтожению объекта. Это необходимо для того, чтобы энергия, которую игрок не успел собрать, не загружала память компьютера. При нажатии на объект «энергия» левой кнопкой мыши, вызывается процедура «OnMouseDown», в которой наращивается количество энергии, после чего данный объект уничтожается.

Пришельцы атакуют базу игрока и, в случае, если пришелец дойдет до базы – уровень считается проигранным (рис. 3 приложения 1). Скрипт,

описывающий условие поражения, представлен на рис. 17. Пришельцы отличаются скоростью перемещения, показателем урона и количеством здоровья. Виды пришельцев описаны в табл. 2. В случае, если здоровье пришельца становится равно нулю, он уничтожается и удаляется с поля. Скрипт, реализующий здоровье, изображен на рис. 18. Все пришельцы перемещаются от края игрового поля к базе игрока, а также способны наносить урон роботам и разрушать их. Скрипт, реализующий поведения объектов вида пришельцы представлен на рис. 19-20.

Табл. 2. Виды пришельцев

№	Изображение	Здоровье, условные единицы	Атака	Скорость перемещения	Описание пришельца
1		5	1	Обычная	Slime – рядовой пришелец со слабыми характеристиками
2		5	1	Ускоренная	Alien – имеет стандартные характеристики здоровья и атаки, но отличается быстрой скоростью
3		7	2	Обычная	Alien Spider – самый сильный пришелец, имеет высокие показатели здоровья и атаки

Скрипт на рис. 17 относится к объекту «стена». При столкновении пришельца со стеной происходит вызов процедуры «OnCollisionEnter2D». В таком случае активируется игровое меню «2», игра останавливается, количество пришельцев обнуляется.

```
public class LoseCond : MonoBehaviour {

    public GameObject menu;
    public GameObject gamecontr;
    private void OnCollisionEnter2D(Collision2D collision)
    {
        if (collision.gameObject.CompareTag("Aliens"))
        {
            menu.SetActive(true);
            gamecontr.GetComponent<GameController>().endgame = true;
            GameOb-
ject.Find("GameController").GetComponent<GameController>().PauseUnpause();
            Alien.count = 0;
        }
    }
}
```

Рис. 17. Условие поражения

```
public class Health : MonoBehaviour
{
    public int cur = 5;
    public void doDamage(int n)
    {
        cur -= n;
        if (cur <= 0)
            Destroy(gameObject);
    }
}
```

Рис. 18. Скрипт, реализующий здоровье

Процедура «doDamage» (рис. 18) вычитает из здоровья объекта количество переданного урона. В случае, если здоровье опустилось до нуля, объект, к которому прикреплен скрипт уничтожается.

```
public class Alien : MonoBehaviour {
    public static int count = 0;
    private int damage = 1;
    float nextatack = 0;
    GameObject txt;
    Animator Anim;
    public int Damage
    {
        get {
            return damage;
        }
        set {
            damage = value;
        }
    }
    private void Start()
    {
        count++;
    }
```

Рис. 19. Скрипт, реализующий поведение объектов типа пришельцы

```
txt = GameObject.Find("MonsterCounter");
        txt.GetComponent<GUIText>().text = count.ToString();
        Anim = GetComponent<Animator>();
    }
private void OnCollisionEnter2D(Collision2D collision)
    {
        if (collision.gameObject.CompareTag("Robots"))
        {

                if (Anim != null)
                {
                    Anim.SetBool("Attacking", true);
                }
        }
    }
    private void OnCollisionStay2D(Collision2D collision)
    {
        if (collision.gameObject.CompareTag("Robots"))
        {
            nextatack -= Time.deltaTime;
            if (nextatack < 0)
            {
                colli-
sion.gameObject.GetComponent<Health>().doDamage(damage);
                nextatack = 1;
                if (Anim != null )
                {
                }
            }
        }
    }
    private void OnCollisionExit2D(Collision2D collision)
    {
            if (Anim != null)
            {
                Anim.SetBool("Attacking", false);
            }
    }
private void OnDestroy()
    {
        GameObject gc;
        txt = GameObject.Find("MonsterCounter");
        if (txt != null)
        {
            txt = GameObject.Find("MonsterCounter");
            count--;
            txt.GetComponent<GUIText>().text = count.ToString();
            if ( count == 0)
            {
                gc = GameObject.Find("GameController");
                if (gc != null)
                {
                    gc.GetComponent<GameController>().endgame = true;
                    gc.GetComponent<GameController>().nextLevDial();
                }
            }
        }
    }
}
```

Рис. 20. Скрипт, реализующий поведение объектов типа пришельцы

В процедуре «Start» (рис. 19-20) производится подсчет пришельцев. При столкновении пришельца с другим объектом вызывается процедура «OnCollisionEnter2d». Если столкновение произошло с роботом, переменная «Attacking» аниматора пришельца становится равна «True», что изменит анимацию пришельца с ходьбы на атаку. «OnCollisionStay2D» вызывает процедуру «doDamage» с заданной периодичностью, данная процедура наносит урон роботу. После уничтожения робота вызывается процедура «OnExitCollider2D», в которой переменная аниматора «Attacking» становится равна «False», что изменит анимацию пришельца на ходьбу [25]. При уничтожении пришельца счетчик пришельцев уменьшается на единицу, если значение счетчика стало равно нулю, то игра останавливается и отображается меню «3» (рис. 4 приложения 1).

К объектам типа ***другие*** относятся стены, энергия и пули. На очки энергии игрок может покупать и размещать роботов. Пули наносят урон пришельцам [5]. Скрипт нанесения урона представлен на рис. 21.

```
public class Projectile : MonoBehaviour
{
    public int damage = 1;
    void OnCollisionEnter2D(Collision2D collision)
    {
        if (collision.gameObject.CompareTag("Aliens"))
        {
            colli-
sion.gameObject.GetComponent<Health>().doDamage(damage);
            Destroy(gameObject);
        }
        else
        {
            if (collision.gameObject.CompareTag("Robots"))
            {
                GetComponent<Rigidbody2D>().position += new Vec-
tor2(collision.gameObject.GetComponent<BoxCollider2D>().size.x, 0);
             colli-
sion.gameObject.GetComponent<BoxCollider2D>().size.x;
            }
        }
    }
    private void Update()
    {
        if( gameObject.transform.position.x >= 6.5)
        {
            Destroy(gameObject);
        }
    }
}
```

Рис. 21. Скрипт, реализующий поведения объектов типа пули

В процедуре «OnCollisionEnter2D» выполняется проверка, с каким типом объекта произошло столкновение. Если столкновение произошло с пришельцем, то ему наносится урон, а пуля уничтожается. Если столкновение произошло с роботом, то пуля сдвигается на размер коллайдера робота по горизонтали. В процедуре «Update» осуществляется уничтожение пули, если она достигла края экрана.

4. ТЕСТИРОВАНИЕ ИГРОВОГО ПРИЛОЖЕНИЯ

4.1. Функциональное тестирование

Тестирование приложения производилось проверкой работы всех функций системы. Данный вид тестирования можно назвать функциональным тестированием. Функциональное тестирование – это тестирование ПО в целях проверки реализуемости функциональных требований, то есть способности ПО в определенных условиях решать задачи, нужные пользователям. Функциональные требования определяют, что именно делает ПО, какие задачи оно решает [6]. Результаты тестирования представлены в таблице 3.

Табл. 3. Функциональное тестирование

№	Название теста	Ожидаемый результат	Полученный результат	Тест пройден?
1	Отображение главного меню игры	На экране отображено главное меню игры с кнопками «Start game» и «Quit»	На экране отобразилось главное меню игры с кнопками «Start game» и «Quit»	Да
2	Работа кнопки «Start game»	При нажатии кнопки «Start game» загружается первый уровень	После нажатия кнопки «Start game» загрузился первый уровень	Да
3	Работа кнопки «Quit»	При нажатии кнопки «Quit» происходит выход из игры	После нажатия кнопки «Quit» произошел выход из игры	Да
4	Выбор робота для размещения	При нажатии на иконку робота на игровой панели, нажатая иконка	При нажатии на иконку робота на игровой панели, нажатая иконка	Да

Продолжение табл. 3

№	Название теста	Ожидаемый результат	Полученный результат	Тест пройден?
		выделяется красной рамкой, при повторном нажатии рамка исчезает	выделилась красной рамкой, при повторном нажатии рамка исчезла	
5	Размещение робота на игровом поле	Робот размещается на выбранной клетке игрового поля, на одну клетку игрового поля может быть размещен только один робот	Робот был размещен на выбранной клетке игрового поля, на клетку, занятую роботом, не удалось разместить другого робота	Да
6	При размещении робота общее количество энергии уменьшается на его стоимость	При размещении робота на выбранной клетке общее количество энергии уменьшается на стоимость робота	При размещении робота на выбранной клетке общее количество энергии уменьшилось на стоимость робота	Да
7	Работа робота-генератора	Генератор производит энергию в заданный интервал времени	Генератор производил энергию в заданный интервал времени	Да
8	Сбор энергии	Энергию можно собирать, нажав на объект «энергия» левой кнопкой	Энергия собирается при нажатии на объект «энергия» левой кнопкой	Да

Продолжение табл. 3

№	Название теста	Ожидаемый результат	Полученный результат	Тест пройден?
		мыши, при этом объект исчезает, а собранная энергия суммируется и отображается в счетчике энергии на игровой панели	мыши, при этом объект исчез, а собранная энергия суммировалась и отображалась в счетчике энергии на игровой панели	
9	Стрельба	Оборонительные роботы выпускают пули в заданный интервал времени	Оборонительные роботы выпускали пули в заданный интервал времени	Да
10	Нанесение урона пулями	Если пуля сталкивается с пришельцем, то наносит ему урон	Пуля, столкнувшись с пришельцем, нанесла ему урон	Да
11	Полет пуль	Пули летят до конца поля справа налево, после уничтожаются; пули не сталкиваются с роботами, а продолжают движение	Пули, достигнув конца поля справа налево, уничтожались; пули не сталкивались с роботами, а продолжали движение	Да
12	Перемещение пришельцев	Пришельцы перемещаются по игровому полю слева направо с заданной скоростью	Пришельцы перемещались по игровому полю слева направо с заданной скоростью	Да

Продолжение табл. 3

№	Название теста	Ожидаемый результат	Полученный результат	Тест пройден?
13	Пришельцы атакуют	Если пришелец сталкивается с роботом, то начинает атаковать его	Пришелец, столкнувшись с роботом, начал его атаковать	Да
14	Уничтожение роботов	Роботы уничтожаются, когда их здоровье становится равно нулю; клетка, на которой стоял робот, вновь активна для размещения	Когда здоровье роботов опускалось до нуля, они уничтожались, на клетку, на которой стоял робот, удалось разместить другого робота	Да
15	Уничтожение пришельцев	Пришельцы уничтожаются, когда их здоровье становится равно нулю	Когда здоровье пришельцев опускалось до нуля, они уничтожались	Да
16	Счетчик пришельцев	При смерти пришельца счетчик пришельцев на игровой панели уменьшается на единицу	При смерти пришельца счетчик пришельцев на игровой панели уменьшился на единицу	Да
17	Условие поражения	Если пришелец доходит до базы игрока, уровень считается проигранным –	Когда пришелец дошел до базы игрока отобразилось игровое меню 2	Да

Продолжение табл. 3

№	Название теста	Ожидаемый результат	Полученный результат	Тест пройден?
		отображается игровое меню 2		
18	Работа кнопки «Restart» игрового меню 2	При нажатии на кнопку «Restart» уровень начнется заново	После нажатия на кнопку «Restart» уровень начался заново	Да
19	Работа кнопки «Exit to menu» игрового меню 2	При нажатии на кнопку «Exit to menu» произойдет переход к «Главному меню»	После нажатия на кнопку «Exit to menu» произошел переход к «Главному меню»	Да
20	Прохождение первого уровня	При уничтожении всех пришельцев отображается игровое меню 3	Когда все пришельцы были уничтожены, отобразилось игровое меню 3	Да
21	Работа кнопки «Next Level» игрового меню 3	При нажатии на кнопку «Next Level» загружается второй уровень	После нажатия на кнопку «Next Level» загрузился второй уровень	Да
22	Работа кнопки «Exit to menu» игрового меню 3	При нажатии на кнопку «Exit to menu» произойдет переход к «Главному меню»	После нажатия на кнопку «Exit to menu» произошел переход к «Главному меню»	Да
24	Отображение игрового меню 1	При нажатии клавиши «Esc»	После нажатия клавиши «Esc»	Да

Продолжение табл. 3

№	Название теста	Ожидаемый результат	Полученный результат	Тест пройден?
		При нажатии клавиши «Esc»	После нажатия клавиши «Esc»	
25	Работа кнопки «Exit to menu» игрового меню 1	При нажатии на кнопку «Exit to menu» произойдет переход к «Главному меню»	После нажатия на кнопку «Exit to menu» произошел переход к «Главному меню»	Да

4.2. Юзабилити-тестирование

Юзабилити-тестирование – это метод тестирования, направленный на установление степени удобства использования, обучаемости, понятности и привлекательности для пользователей разрабатываемого продукта в контексте заданных условий [19].

Все задачи были решены всеми респондентами. Выявленные проблемы устранены. Отчет о результатах юзабилити-тестирования приведен в приложении 2.

ЗАКЛЮЧЕНИЕ

В ходе выпускной квалификационной работы бакалавра мною было реализовано игровое приложение с непрямым управлением в жанре тактическая стратегия и решены следующие задачи.

1. Были исследованы существующие игровые приложения с непрямым управлением.

2. Было спроектировано игровое приложение с непрямым управлением в жанре тактической стратегии.

3. Игровое приложение было реализовано и протестировано.

ЛИТЕРАТУРА

1. Зальцман М. Компьютерные игры. Как это делается. – М.: Издательство Логрус, 2000. – 530 с.

2. Исследование игровой индустрии в России. [Электронный ресурс] URL: https://www.pwc.ru/ru/entertainment-media/assets/e-media-outlook-2016.pdf (дата обращения: 16.03.2017).

3. Исследование игровой индустрии от CEO агентства Insight One. [Электронный ресурс] URL: https://vc.ru/p/game (дата обращения: 16.03.2017).

4. Исследование игровой индустрии от Mail.ru Group. [Электронный ресурс] URL: https://habrahabr.ru/company/mailru/blog/183248/ (дата обращения: 16.03.2017).

5. Конгер Д. Физика для разработчиков компьютерных игр. – М.: Бином. Лаборатория знаний, 2010. – 520 с.

6. Криспин Л., Грегори Д. Гибкое тестирование: практическое руководство для тестировщиков ПО и гибких команд. – М.: Вильямс, 2010. – 464 с.

7. Обзор игрового движка Unity3D. [Электронный ресурс] URL: http://devgam.com/polnyj-obzor-unity-5 (дата обращения: 28.03.2017).

8. Официальный сайт Cryengine. [Электронный ресурс] URL: http://cryengine.com/ (дата обращения 28.03.2017).

9. Официальный сайт Unity3D. [Электронный ресурс] URL: https://unity3d.com (дата обращения: 28.03.2017).

10. Официальный сайт Unreal Engine. [Электронный ресурс] URL: https://www.unrealengine.com/ (дата обращения: 28.03.2017).

11. Официальный сайт игры «Kingdom Rush». [Электронный ресурс] URL: https://www.kingdomrush.com/ (дата обращения: 12.04.2017).

12. Официальный сайт игры «Majesty 2». [Электронный ресурс] URL: http://majesty2.ru/ (дата обращения: 12.04.2017).

13.Официальный сайт игры «Plants vs Zombies». [Электронный ресурс] URL: http://www.popcap.com/plants-vs-zombies-1 (дата обращения: 12.04.2017).

14.Официальный сайт игры «Slaves to Armok II: Dwarf Fortress». [Электронный ресурс] URL: http://www.bay12games.com/dwarves/ (дата обращения: 12.04.2017).

15.Официальный сайт игры «The Settlers 7». [Электронный ресурс] URL: http://thesettlers.uk.ubi.com/the-settlers-7/ (дата обращения: 12.04.2017).

16.Пэрент Р. Компьютерная анимация. Теория и алгоритмы. – М.: КУДИЦ-ОБРАЗ, 2004. – 560 с.

17.Рамбо Д., Блаха М. UML 2.0. Объектно-ориентированное моделирование и разработка. – СПб.: Питер, 2007. – 544 с.

18.Стиллмен Э., Грин Д. Изучаем C#. – СПб.: Питер, 2016. – 816 с.

19.Тетерук И., Булат А., Антонов В. Тестирование удобства пользования. [Электронный ресурс] URL: http://www.protesting.ru/testing/types/usability.html (дата обращения: 24.05.2017).

20.Торн А. Искусство создания сценариев в Unity. – М.: ДМК, 2016. – 360 с.

21.Торн А. Основы анимации в Unity. – М.: ДМК, 2016. – 176 с.

22.Финни К. 3D-игры. Все о разработке. – М.: Бином. Лаборатория знаний, 2007. – 976 с.

23.Хокинг Д. Unity в действии. Мультиплатформенная разработка на C#. – СПб.: Питер, 2016. – 336 с.

24.Чиксентмихайи М. Поток. Психология оптимального переживания. – М.: Альпина нон-фикшн, 2017. – 464 с.

25.Шампандар А. Искусственный интеллект в компьютерных играх. Как обучить виртуальные персонажи реагировать на внешние воздействия. – М.: Вильямс, 2007. – 768 с.

26.Шилдт Г. C# 4.0. Полное руководство. – М.: Вильямс, 2015. – 1056 с.

27.Эхерн Л. Самоучитель по созданию компьютерных игр. – М.: ДМК, 2008. – 304 с.

ПРИЛОЖЕНИЯ

Приложение 1

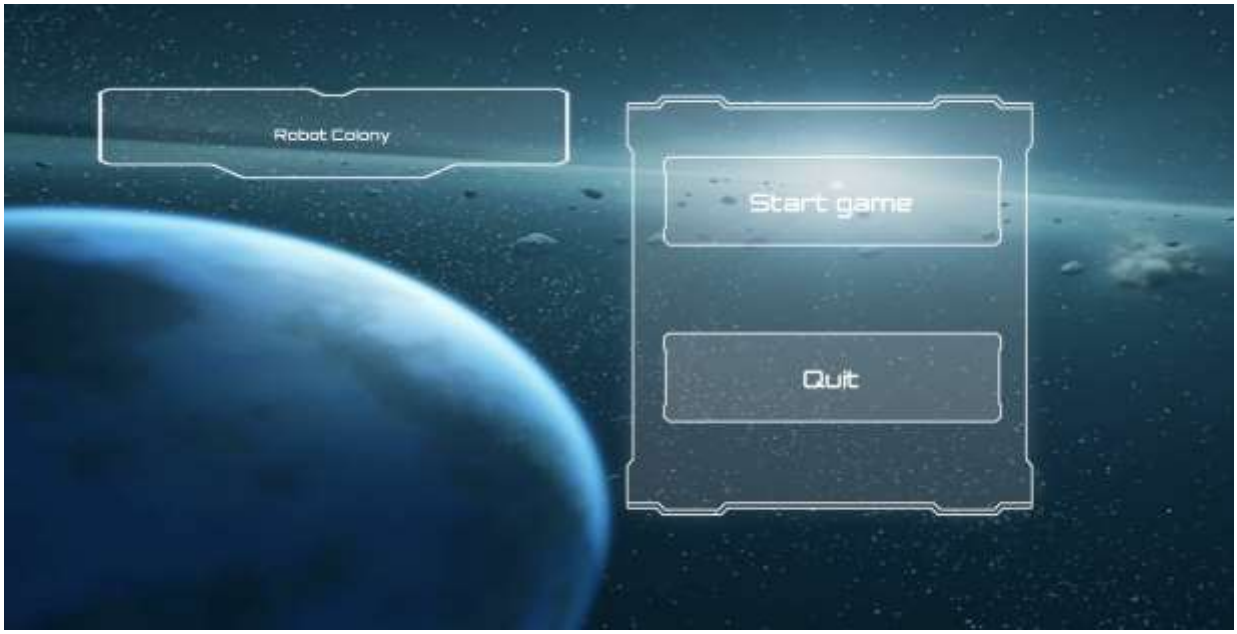

Рис. 1. Главное меню

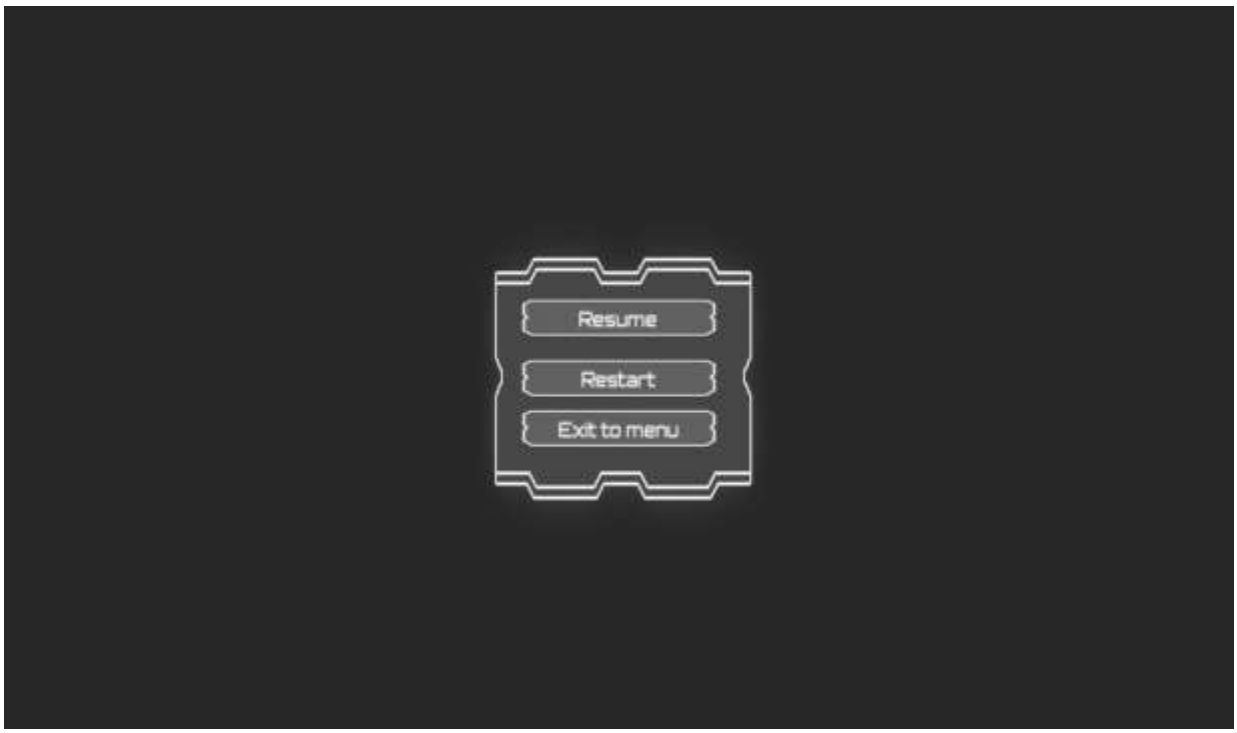

Рис. 2. Игровое меню 1

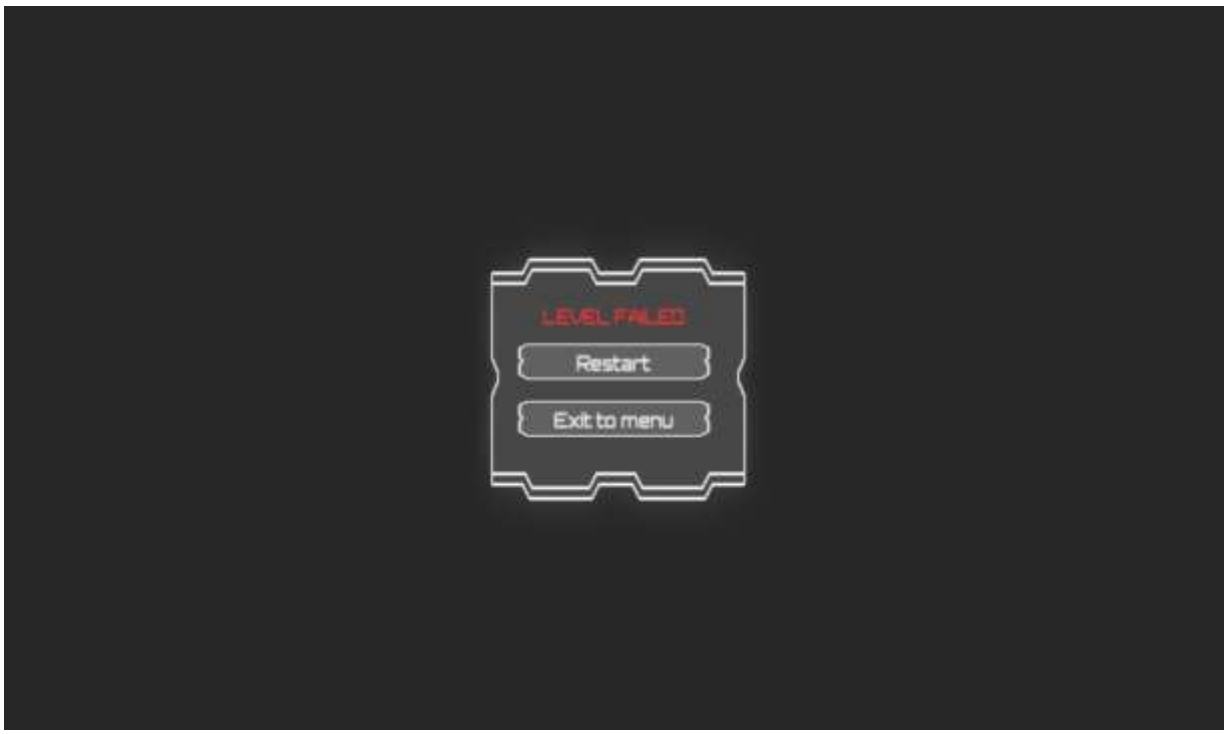

Рис. 3. Игровое меню 2

Рис. 4. Игровое меню 3

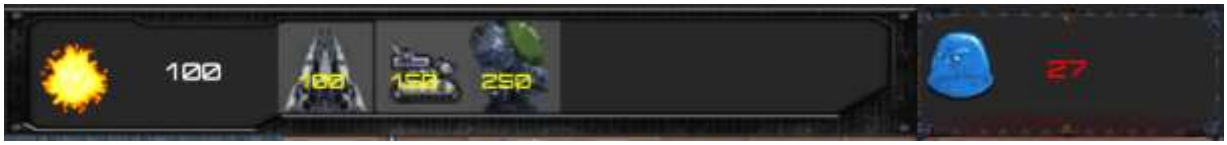

Рис. 5. Игровая панель

Рис. 6. Игровой уровень

Рис. 7. Игровой уровень

Рис. 8. Игровой уровень

Приложение 2

Отчет о юзабилити-тестировании

1. Объект исследования

Объектом исследования является игровое приложение Robot Colony.

2. Метод исследования

В качестве метода исследования был выбран метод юзабилити-тестирования «Мысли вслух».

3. План проведения тестирования

1. Планирование:
   1.1. Разработка заданий.
   1.2. Набор участников тестирования.
2. Проведение тестирования.
3. Анализ полученных данных.

4. Методика проведения тестирования

Для проведения исследования использовалось следующее оборудование и ПО:

1. ПК с операционной системой Windows 10;
2. Клавиатура и компьютерная мышь;
3. Встроенный в ПК микрофон.

Наблюдатель выдает список заданий участникам тестирования. Респонденты выполняют их последовательно и самостоятельно, при этом высказывая свои мысли и мнения, возникающие в процессе взаимодействия с приложением. Комментарии записываются через микрофон и затем представляются модератором в виде протокола тестирования.

5. Протокол заданий

1. Отправная точка: главное меню.
   Задание: начать новую игру.
2. Отправная точка: игровая сцена «Уровень 1».
   Задание: разместить «Energy Generator».
3. Отправная точка: игровая сцена «Уровень 1».

Задание: накопить 150 «энергии».

4. Отправная точка: игровая сцена «Уровень 1».

   Задание: разместить «Shooting Drone».

5. Отправная точка: игровая сцена «Уровень 1».

   Задание: открыть игровое меню 1.

6. Отправная точка: игровое меню 1.

   Задание: начать уровень заново.

7. Отправная точка: игровая сцена «Уровень 1».

   Задание: пройти первый уровень.

8. Отправная точка: игровое меню 3.

   Задание: перейти на второй уровень.

9. Отправная точка: игровая сцена «Уровень 2».

   Задание: проиграть.

10. Отправная точка: игровое меню 2.

    Задание: выйти из игры.

6. Респонденты

1. Авагян Геворг Артурикович, студент ЮУрГУ.
2. Ковалев Вячеслав Сергеевич, студент ЮУрГУ.
3. Елисеев Роман Сергеевич, студент ЮУрГУ.

7. Результаты тестирования

7.1. Метрики

Табл. 1. Результаты юзабилити-тестирования

Респонденты	**Доля выполненных задач**	**Доля правильно выполненных задач**	**Время на выполнение задач**	**Кол-во ошибок**
Авагян Геворг Артурикович	100%	100%	10 мин.	0
Ковалев Вячеслав Сергеевич	100%	100%	7 мин.	0
Елисеев Роман Сергеевич	100%	100%	12 мин.	1

7.2. Выявленные недостатки

Игровой процесс

Первый уровень был слишком труден для прохождения.

Дизайн

Шрифт в названии игры и на кнопках был слишком мелким.