Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Институт естественных и точных наук
Факультет математики, механики и компьютерных технологий
Кафедра прикладной математики и программирования
Направление подготовки Программная инженерия

РАБОТА ПРОВЕРЕНА
Рецензент,
_____
_____
«____»_____2018г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой, д.ф.-м.н.,
доцент
_____/А.А.Замышляева
«____»_____2018 г.

Интерактивное приложение «PhysiLabs»
ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ–09.04.04.2018.191 ПЗ ВКР

Руководитель работы д.ф.-м.н.,
профессор
_____/ Н.Д. Зюляркина
«____»_____2018 г.

Автор работы
студент группы ЕТ – 223
_____/ Е.Д. Васильев
«____»_____2018 г.

Нормоконтролер, к.э.н., доцент
_____/ Д.А. Дрозин
«____»_____2018 г.

Челябинск 2018

АННОТАЦИЯ

Васильев Е.Д. Интерактивное приложение «PhysiLabs». – Челябинск: ЮУрГУ, ЕТ-223, 124 с. 27 ил., 4 табл., библиогр. список – 30 наим., 2 прил.

Выпускная квалификационная работа посвящена разработке интерактивного приложения, которое позволит в игровой форме изучить и закрепить теоретические знания из различных областей естественных наук.

Основная цель проекта – дать возможность пользователям увидеть принципы действия и последствия тех или иных явлений и законов физики, механики, робототехники и т.п. На данный момент аналоги такого приложения являются узко специализируемыми, сложными в освоении или попросту развлекательными. Данная разработка объединяет в себе как теоретическою информацию, так и соответствующие интерактивные визуальные элементы.

Для достижения поставленных задач была изучена предметная область и множество аналогов, разработана концепция, на основе которой, спроектировано и реализовано приложение. Для защиты пользовательский данных от фальсификации, а также ресурсов системы от изъятия была разработана и реализована криптографическая модель, ориентированная на использование с инструментом разработки.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Внедрение мультимедиа-технологий в учебный процесс является перспективным направлением представления учебного материала в новой форме. Наряду с широкими возможностями современных цифровых устройств, стоит простота доступа к информационным ресурсам, их разнообразие и качество. Познавать окружающий нас мир становится всё проще и увлекательней и этому способствует множество программных продуктов.

Игровая форма познания является естественной, потому что человек с раннего детства воспринимает и изучает окружающий мир в игре, моделируя различные ситуации сам этого не замечая. Мы создаём игровую ситуацию как естественную форму существования, которая является комфортной для пользователя, и несёт в себе познавательную нагрузку.

Создание обучающих компьютерных игр представляет собой одно из важных направлений в компьютеризации обучения. Соединение эмоциональной привлекательности игры с аудиовизуальными, вычислительными, информационными и другими возможностями персональных компьютеров несет в себе большой потенциал.

В виртуальном пространстве можно моделировать множество различных явлений из окружающего мира. Они могут быть представлены в реалистичном виде или в виде некой абстракции, которая даст новое восприятие изучаемого объекта. Интерактивность игры позволяет изменять различные параметры модели и сразу же оценивать результаты выполненных действий. Мы можем комбинировать различные задачи, чтобы показать меж предметные связи.

Целью выпускной квалификационной работы является разработка интерактивного приложения, направленное на повышение уровня подготовки по дисциплинам естественных наук.

ГЛАВА 1. ИСПОЛЬЗОВАНИЕ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ В ОБРАЗОВАТЕЛЬНОМ ПРОЦЕССЕ

Мультимедиа-технологии – это одно из перспективных направлений информатизации учебного процесса. В совершенствовании программного и методического обеспечения, материальной базы, а также в обязательном повышении квалификации преподавательского состава видится перспектива успешного применения современных информационных технологий в образовании [1].

Информационные технологии в традиционном процессе обучения носят дополнительный характер и используются для организации самостоятельной работы учащихся, чаще всего ограниченной написанием рефератов, проведения контрольных и других видов работ.

Так же на уроках и лекциях при наличии средств проецирования изображения могут использоваться иллюстрации и фрагменты из состава электронных учебников или учебных пособий. На занятиях могут демонстрироваться эксперименты, входящие в состав лабораторных практикумов, проводиться тестирования по изученному материалу, просматриваться и создаваться видеоматериалы. На сегодняшний день активно начинают использоваться мультимедиа технологии. Учебные мультимедиа материалы имеют классификацию по уровням [2]:

1) текст;

2) аудио- и видеоинформация, двухмерная графика;

3) трехмерная компьютерная графика, компьютерные игры, интерфейсы виртуальной реальности.

Средства новых информационных технологий, используемые на занятиях, сами по себе являются стимулом к изучению предмета. Формирование и поддержание положительной мотивации к учению нельзя представить себе без использования специально разработанных и реализованных в средстве обучения методических приемов.

Широкое разнообразие и сферы распространения информационных технологий привели к тому, что университеты различных стран обратили внимание на возможность использования компьютерных и телекоммуникационных технологий для организации дистанционного обучения. Эта форма занятий позволяет организовать образовательный процесс с использованием необходимых учебно-методических материалов, устанавливать двустороннюю связь между преподавателем и обучающимся с обменом информацией внутри системы дистанционного обучения.

1.1. Использования компьютерных игр в образовательном процессе

Известно, что игра, как метод обучения, существует с древних времен, и использовалась достаточно широко для передачи опыта от старшего поколения младшему. В настоящее время при достаточно большом уплотнении материала, активном и интенсивном учебном процессе игровая деятельность может использоваться в следующих случаях:

– в качестве самостоятельных технологий для освоения понятия, темы или раздела учебного предмета;

– как элемента более обширной технологии;

– в качестве урока (занятия) или его части (например, введения, объяснения, закрепления, контроля).

В зависимости от того, как наставник понимает функции игры и классифицирует их, сочетаются элементы игры и ученья, от этого также зависит место и роль игровой технологии в учебном процессе.

Компьютерная игра – это программное обеспечение, направленное на организацию игрового процесса. Игра может служить как для связи игроков, так и выступать сама в качестве партнёра. Для осуществления игрового процесса используются различные устройства ввода, такие как: компьютерная мышь, клавиатура, камера, джойстик и т.д. [4].

Некоторые игры могут быть признаны предметом искусства, что позволяет им конкурировать с кино, а некоторые коммерческие организации используют их в качестве тренажёров для различных видов профессий.

Обучающая компьютерная игра – это форма учебно-воспитательной деятельности, имитирующая те или иные практические ситуации, являющиеся одним из средств активизации учебного процесса и способствующая умственному развитию. Такие игры по всем признакам соответствуют определению дидактической игры, которыми они по своей сути являются, только организованными на более высоком уровне.

Обучающей компьютерной игре свойственна двуплановость: с одной стороны, играющий выполняет реальную деятельность, осуществление которой требует действий, связанных с решением вполне конкретных, часто нестандартных задач; с другой – ряд моментов этой деятельности носит условный характер, позволяющий отвлечься от реальной ситуации с ее ответственностью и многочисленными сопутствующими обстоятельствами. Посредством визуализации и одновременным воздействием на различные органы чувств, «вживания в образ» и другими методами, она облегчает усвоение материала, активизирует познавательную деятельность. Компьютер предоставляет безграничные возможности организовать обучение в игровой форме.

Анализ игровых программ учебного назначения показывает, что больший интерес проявляется в том случае, когда обучающая программа выступает не столько в роли строго учителя, оценивающего каждый шаг своего ученика, сколько в роли доброжелательного и неназойливого помощника.

Помощь рассматривается в различных видах. Например:

– иерархии подцелей для успешного выполнения игрового задания;

– рекомендаций по ключевым вопросам;

– ответов на вопросы ученика в интерактивной форме при возникновении такой необходимости;

– традиционной справочной системы о программе с возможностью поиска информации по ключевым терминам, по разделам, по наиболее часто задаваемым вопросам и т. д.;

– эмоциональной оценки выполняемых действий в фоновом режиме.

1.2. Классификация компьютерных игр

Компьютерные игры можно классифицировать по объекту решаемой задачи:

– развивающие игры – программы «открытого» типа, направлены на обучение и экспериментирование. Такие игры предназначены для формирования и развития общих умственных способностей, целеполагания, способности мысленно соотносить свои действия по управлению игрой с создающимися изображениями, для развития фантазии, воображения, эмоционального и нравственного развития. В них нет явно заданной цели – они являются инструментами для творчества, для самовыражения;

– обучающие игры – это программы закрытого типа, в которых в игровой форме предлагается решить одну или несколько дидактических задач. К этому классу относятся игры, связанные с формированием математических представлений; с обучением азбуке, письму через чтение и чтению через письмо, родному и иностранным языкам; с формированием динамических представлений по ориентации на плоскости и в пространстве; с эстетическим, нравственным воспитанием; экологическим воспитанием; с основами систематизации и классификации, синтеза и анализа понятий;

– игры экспериментирования имеют не явно заданные цели и правила, они скрыты в сюжете или способе управления игрой. Чтобы добиться успеха в решении игровой задачи, игрок должен путем поисковых действий прийти к осознанию цели и способа действия, что и является ключом достижения общего решения игровой задачи;

– игры-забавы – это игры, в которых не содержатся в явном виде игровые задачи или задачи развития. Они просто предоставляют возможность развлечься, осуществить поисковые действия и увидеть на экране результат в виде какого-либо видеофрагмента;

При проведении классификации по параметру «жанр игры» выделяют следующие типы игр:

– приключенческая игра или квест (англ. adventure) – игра с продуманным и обычно линейным сюжетом. В этом направлении широко используют различные головоломки;

– боевик (англ. action) – игра, требующая от игрока постоянных действий, насыщенная боевыми сценами, драками и перестрелками;

– ролевая игра (RPG – англ. Role Playing Game) – игра, отличительной особенностью которой является наличие у персонажей определённых навыков и характеристик, которые можно обрести, а впоследствии развивать, выполняя какие-либо действия. К этому жанру относятся и многопользовательские ролевые игры,

которые, в отличие от однопользовательских, не имеют ни конечной цели, ни законченного сюжета;

– стратегическая игра (англ. strategy) – игра, представляющая собой управление глобальными процессами, как например, развитие экономики, создание армии, строительство парков и т. д. В данном жанре обычно используется два режима игры: в реальном времени или пошаговом режиме;

– компьютерный симулятор (англ. simulator) – игра, созданная с целью симуляции реальных действий или явлений из реального мира, например, управление автомобилем или инфраструктурой целого города;

– головоломка (англ. puzzle) – название жанра компьютерных игр, целью которых является решение логических задач, требующих от игрока задействования логики, стратегии и интуиции.

1.3. Игровые предпочтения пользователей

Чтобы определиться с жанром и направлением развития проекта следует проанализировать предпочтения пользователей. Обратимся к данным опроса проведённым фондом «Общественное Мнение» (ФОМ) в мае 2016 года [5].

По данным опроса ФОМ, в компьютерные игры играют 44% интернет-пользователей. В частности, 39% играют в компьютерные игры на планшетах, смартфонах и приставках чаще, чем раз в месяц: 12% – почти каждый день, 18% – несколько раз в неделю и 9% – несколько раз в месяц. В онлайн-игры играют только 18% интернет-пользователей.

Среди всех опрошенных можно определить предпочитаемые и наиболее популярные жанры игр. Стоит отметить, что многие пользователи отдают предпочтения нескольким жанрам.

74% пользователей выбирают казуальные игры, например, «сапер», пасьянсы, Angry Birds. 42% играют в игры-симуляторы, такие как гонки или футбол. 33% предпочитают боевики, аркады, например, Counter-Strike, Call of Duty, Mortal Kombat. 27% выбирают стратегии, такие как Heroes of Might and Magic, StarCraft, SimCity. 22% играют в приключенческие игры, квесты, например, «Петька и Василий Иванович», Broken Sword, Amnesia. 21% выбирает ролевые игры, такие как Diablo, World of Warcraft.

На рисунке 1.1 показано общее соотношение популярности игровых жанров по числу опрошенных.

Респонденты ФОМ отмечают, что увлечение компьютерными играми мешает образованию и развитию (3%), указывают на бесполезность и даже вредность такого времяпрепровождения (2%), а также обеспокоены агрессией в современных играх (2%).

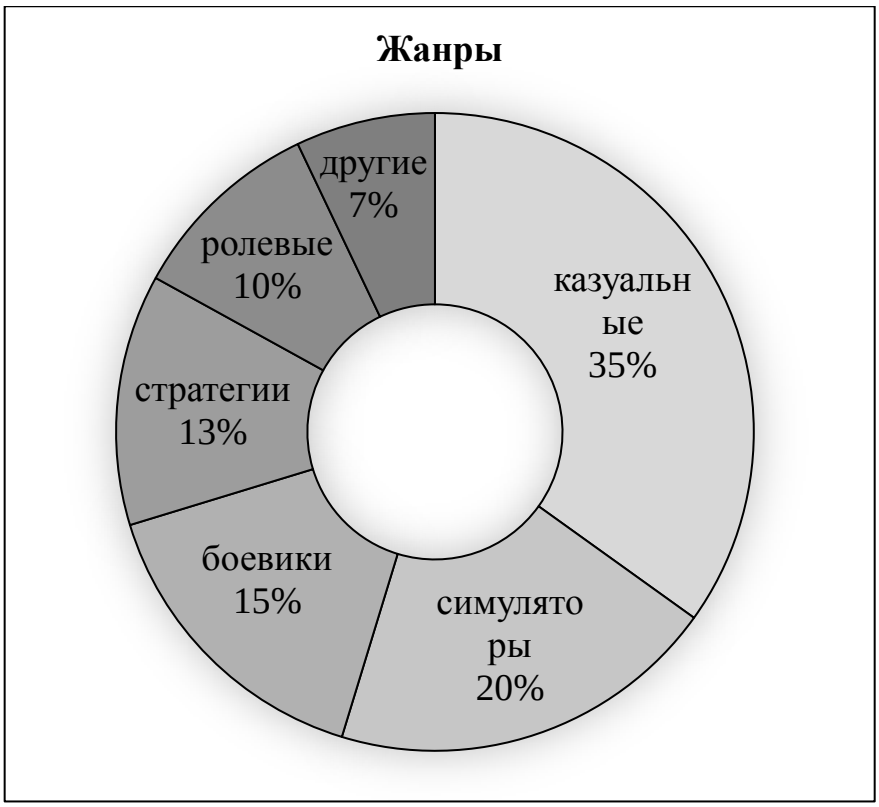

Рис. 1.1. Соотношение популярности жанров

По данным проведённого опроса можно сделать вывод, что почти половина опрошенных играют в игры. С учётом роста популярности данной области информационных технологий можно ожидать увеличения количества игроков и расширения возрастных границ. Однако, многие отмечали, что игры не приносят им пользы, поэтому вопрос о контенте в данной разновидности программного продукта является крайне актуальным.

1.4. Обзор аналогов

Развивающие мультимедиа-технологии уже успели занять свою нишу на рынке программных продуктов, распространившись на множество платформ. В качестве аналогов будут рассмотрены образовательные игры и приложения для изучения физических явлений с различным уровнем научной и развлекательной составляющих.

1.4.1. Snapshots of the Universe

Snapshots of the Universe [6] – приложение для iOS, выпущенное компанией Random House совместно со Стивеном Хокингом. На рисунке 1.2 изображена обучающая подсказка в начале игрового уровня.

Это приложение включает в себя несложную игру и интерактивные лекции. Игра состоит из восьми экспериментов, которые дают пользователям возможность получить базовые знания по физике, и познакомиться с принципами, управляющими нашей Вселенной.

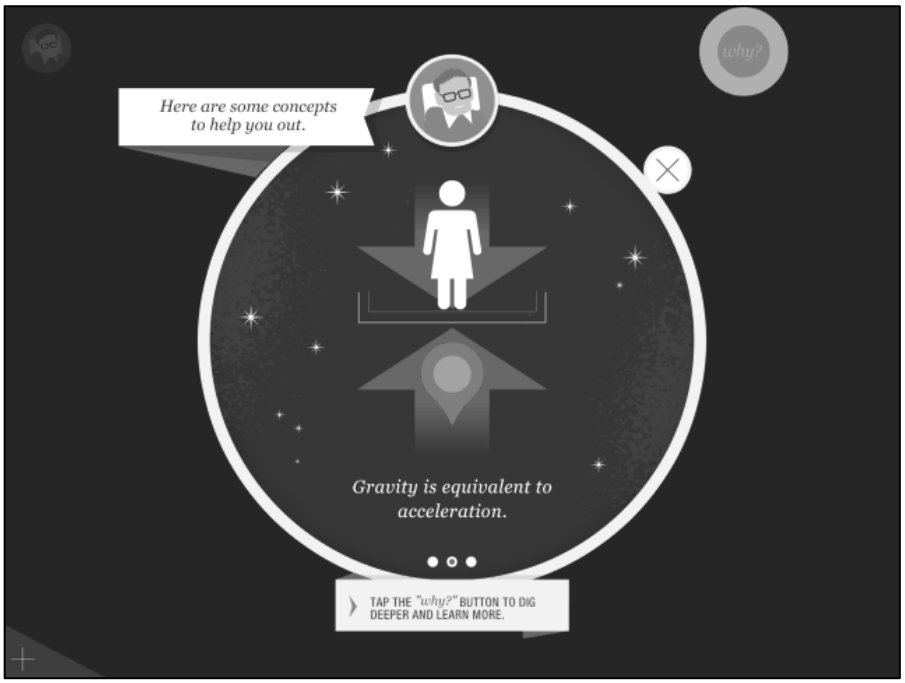

Рис. 1.2. Snapshots of the Universe

В рамках предложенных экспериментов игроки могут отправлять ракеты в открытый космос, собирать собственные звёздные системы, искать и изучать чёрные дыры. Каждый эксперимент можно проводить бесчисленное количество раз, изменяя физические параметры и наблюдая за появляющимися эффектами. Чтобы лучше понять эксперименты, можно зайти в раздел объяснения результатов и посмотреть видео. Приложение доступно на iTunes, стоимость его составляет $4,99.

К интересным моментам данного приложения можно отнести формат представления информации, где текст сопровождается анимацией, а само информационное поле не перегружено.

1.4.2. Particulars

Particulars [7] – это игра с сочетанием особенностей аркады и головоломки, место действия – мир субатомных частиц. На рисунке 1.3 изображена заставка при запуске игры. Взяв под контроль одного из кварков, игрок должен вести переговоры с фундаментальными силами Вселенной. Другие частицы будут притягиваться и отталкиваться, соединяться и изменять полярность, задача несчастного кварка – не терять контроль и избегать разрушения.

На сайте производителя представлена бесплатная демо-версия, стоимость полной версии составляет от $5 до $10 – в зависимости от системы.

Данное приложение выделяется на фоне других изобилием красок в игровом процессе и сильной абстракцией от реальных явлений, что позволяет представить невидимые элементы в виде условных игровых объектов.

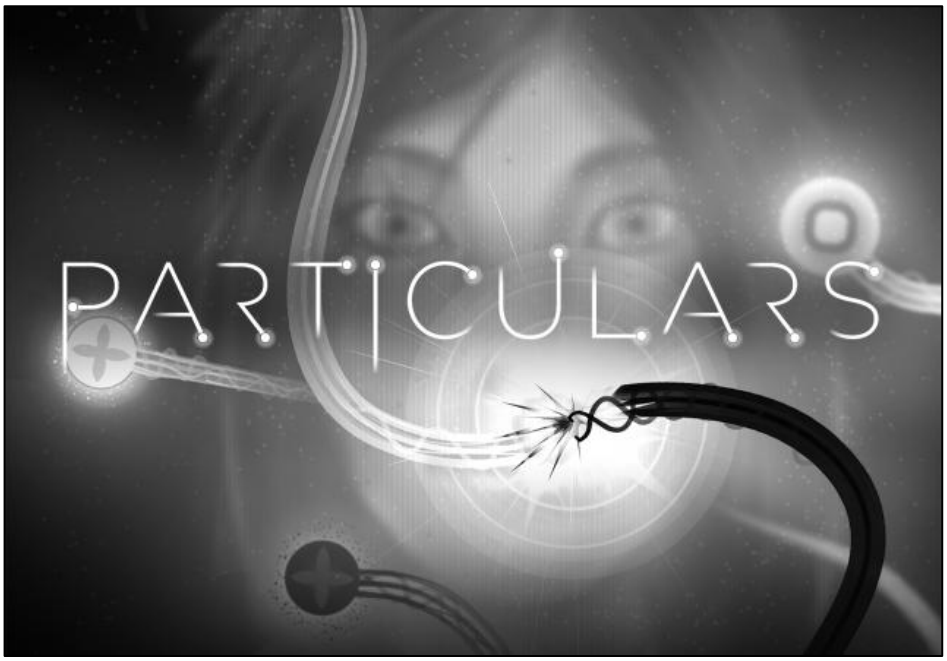

Рис. 1.3. Particulars

1.4.3. Power Toy

Power Toy [8] – приложение для изучения влияния давления, температуры на процессы бесчисленного количества взаимодействий между различными веществами. В поле внимания игроков попадают ядерные реакции, строительство и уничтожение атомных электростанций и т.д. На рисунке 1.4 представлен один из вариантов выбора параметров. Игра доступна бесплатно для PC, Mac и Linux.

Отличительной чертой данного приложения является детальность и точность проработки физических процессов, что позволяет варьировать достаточным набором параметров, чтобы рассмотреть все возможные сценарии поведения системы.

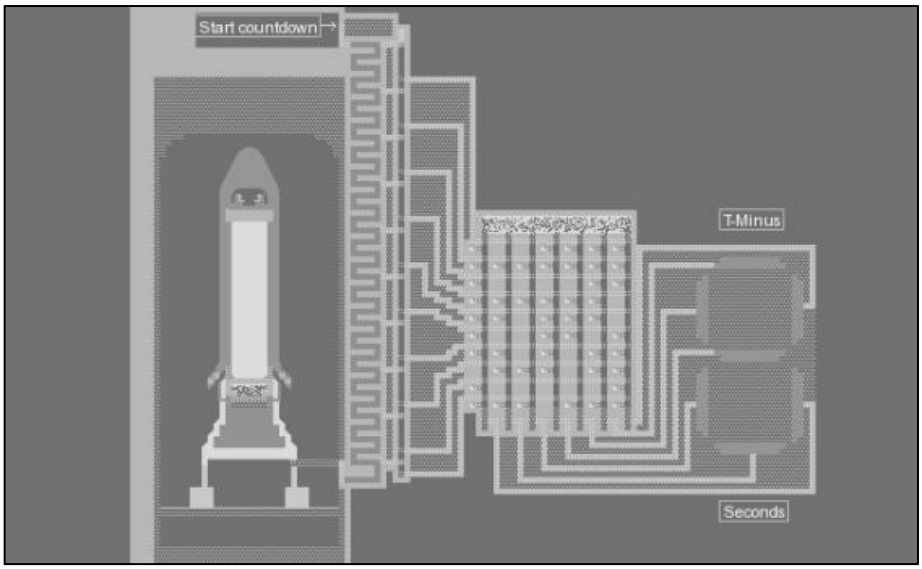

Рис. 1.4. Power Toy

1.4.4. A Slower Speed of Light

A Slower Speed of Light [9] – игра от первого лица, разработанная лабораторией игр Массачусетского технологического института (MIT), даёт возможность игрокам познакомиться с восприятием пространства на околосветовых скоростях и понять теорию относительности. Задача игрока – перемещаться по 3D-пространству в вымышленной деревне сказочных существ и гигантских грибов, изображённой на рисунке 1.5 и собирать сферические объекты, которые замедляют скорость света на фиксированные значения. Это даёт возможность наблюдать за различными визуальными эффектами эйнштейновской теории. Приложение распространяется бесплатно.

Аналогично Particulars, данное приложение выделяется своей нацеленностью на абстрактную визуализацию явлений, которые не доступны в обычных условиях или в пределах нашей планеты.

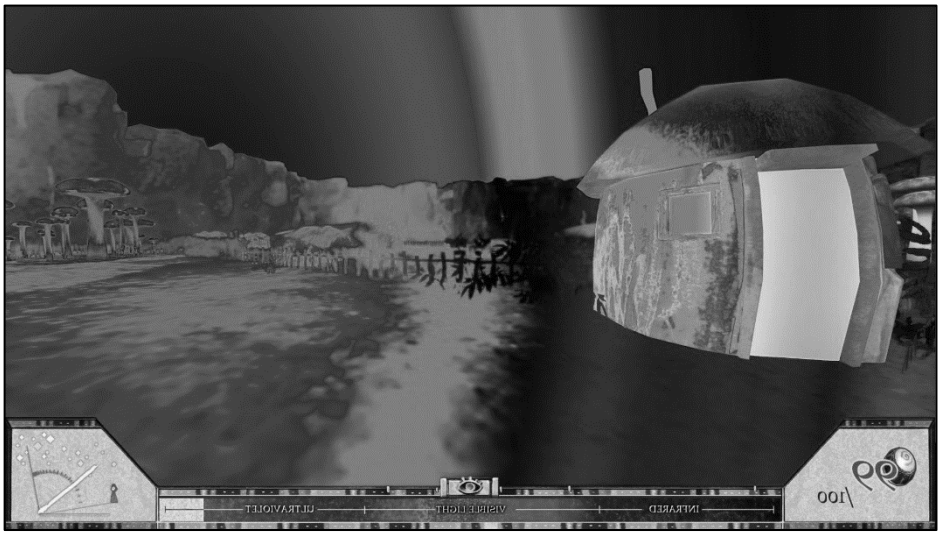

Рис. 1.5. A Slower Speed of Light

1.4.5. ALGODOO

ALGODOO [10] – Компьютерная игра-симулятор физики в виде "песочницы", представляет собой графический анимационный редактор, изображённый на рисунке 1.6. Особенностью приложения является возможность создавать механические и физические объекты на сцене во время симуляции, и они сразу начинают подчиняться законам физики. Приложение доступно в интернет-магазине Steam за $10.

Приложение выделяется уровнем свободы действий предоставляемым пользователю. В отличии от Power Toy, где пользователю предоставляются готовые блоки системы, ALGODOO позволяет создавать собственные использую примитивы более низкого уровня. Однако, как и с Power Toy это приложение имеет ограничение в виде двухмерного пространства.

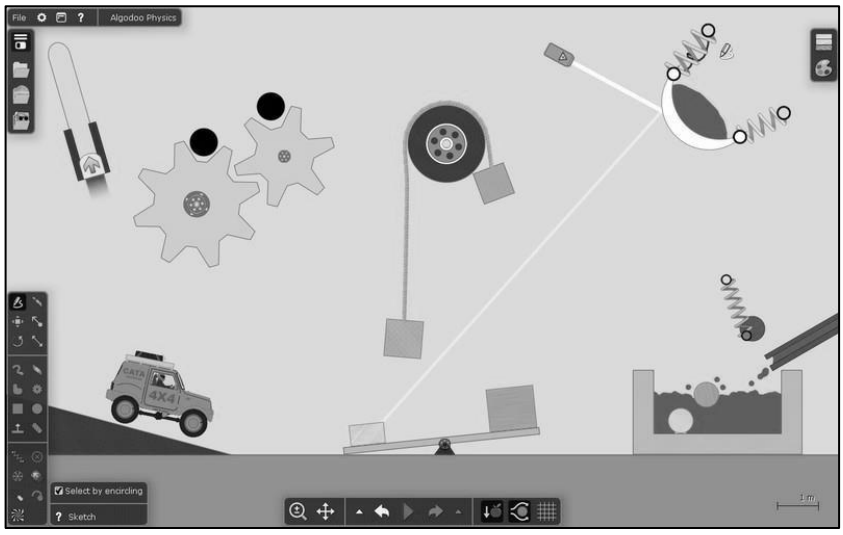

Рис. 1.6. Окно редактора в ALGODOO

Таким образом, на данный момент практически нет приложений, которые объединяли бы множество предметных областей и имели какую-либо контрольную статистику по решению задач. В большинстве случаев в игре рассматривается одно определённое явление, или она носит сугубо развлекательный характер.

Однако такие игры наделены интересным оформлением и сюжетом, в отличие от простых приложений моделирования, что делает прохождение таковых довольно увлекательным и приятным занятием.

1.5. Выводы по разделу

На данном этапе работы был проведён анализ предметной области, определена значимость информационных технологий в образовательном процессе, а также место компьютерных игр в форме представления образовательного контента. Выполнен обзор аналогов и выделены положительные стороны каждой разработки, что позволило выстроить общий вектор дальнейшего развития проекта.

ГЛАВА 2. РАЗРАБОТКА ИНТЕРАКТИВНОГО ПРИЛОЖЕНИЯ «PHYSILABS»

2.1. Концепция

По сценарию игрок перемещается по игровым локациям и решает задачи, предлагаемые согласно сюжетной задумке, способ выполнения которых, также завязан на сценарии игры. Каждое успешно решенное задание открывает доступ к новой игровой локации. Сочетания игровых элементов и задач достаточно велико, чтобы заставить игрока задуматься. По мере прохождения игрок познаёт новые разделы естественных наук, что позволяет комбинировать новые задачи с уже ранее решёнными.

Игроку предстоит взаимодействовать с различными игровыми элементами, которые могут представлять собой элементы пользовательского интерфейса, подобные компонентам Windows интерфейса, выполненные в виде кнопок, ползунков, текстовых полей и т.д., а также трёхмерные объекты игровой сцены.

Сам игровой процесс выполнен в жанре головоломки от первого лица [11], а место действия – помещения научной лаборатории на Луне с названием PhysiLabs, образованное от словосочетания «physic» (англ.) – естественный, «lab» (англ.) – лаборатория. Смежно с PhysiLabs располагается лаборатория робототехники и искусственного интеллекта GlobeLabs, продуктом деятельности которой являются различные роботизированные системы, такие как главный герой игры – робот серии PHY, от лица которого играет пользователь, сферические роботы GLOBE и камеры наблюдения GCam. Экземпляры последних трёх представлены

на рисунке 2.1–2.3.

Рис. 2.1. Камера наблюдения GCam;

Рис. 2.2. Сферический летающий робот GLOBE

Рис 2.3. Робот PHY

Отличительной чертой этих разработок является наличие искусственного интеллекта в разной степени развитости. Все разработки лаборатории GlobeLabs имеют совместимый интерфейс управления, что позволяет подключиться роботам друг к другу и выполнять определённые команды.

Главный герой имеет аккумуляторную батарею, которая в процессе игры разряжается, что позволяет делать игроку, вынужденный перерыв на отдых, а родителям ограничить время провождения ребёнка за игрой. Настраивать интенсивность разрядки батареи можно в меню, указывая желаемое время.

Существует различные модификации сферических роботов, которые различаются способом передвижения и функционалом:

– наземные (нейтральные) – автономные, передвигаются как самостоятельно по маршруту, так и при подключении по совместимому интерфейсу. При встрече может поприветствовать или вступить в краткий диалог;

– наземные (охранники) – автономный и агрессивный, неохотно вступает в диалог, находиться на охраняемой позиции и не подпускает других роботов отталкивая их при помощи воздушной пушки;

– воздушные (грузоперевозчики) – могут передвигаться по воздуху при помощи несущих винтов в автоматическом режиме по заданному маршруту, имеют подвеску, что позволяет брать с собой груз ограниченной массы.

Камеры наблюдения имеют особые навыки активировать элементы на сцене располагаясь на расстоянии в поле зрения таковых. Разнообразие роботов позволяет разбавлять решение предметных задач решение второстепенных головоломок, что делает игровой процесс интересней и содержательней.

Многие познавательные элементы в игре, например, стенды или портреты ключевых учёных данной предметной области, имеют подробное описание с изображениями. За прочтение таких информационных «карточек» игрок будет получать бонусные очки, так же он может прочитать более подробную статью из источника в сети, нажав соответствующею кнопку.

2.2. Архитектура разрабатываемой системы

На схеме вариантов использования, изображённой на рисунке 2.4, определены основные отношения между актёрами и прецедентами, описывающими систему на концептуальном уровне.

Исходя из вариантов использования, выделена совокупность составляющих компонентов системы, изображенных на рисунке 2.4, которая содержит:

– клиентское приложение, включающее сюжетную композицию, конструктор уровней и комнаты экспериментов;

– серверную часть с информационным сайтом и модулями управления учетными записями игроков;

– облачное хранилище для сохранения результатов достижений игроков и каталога новых заданий.

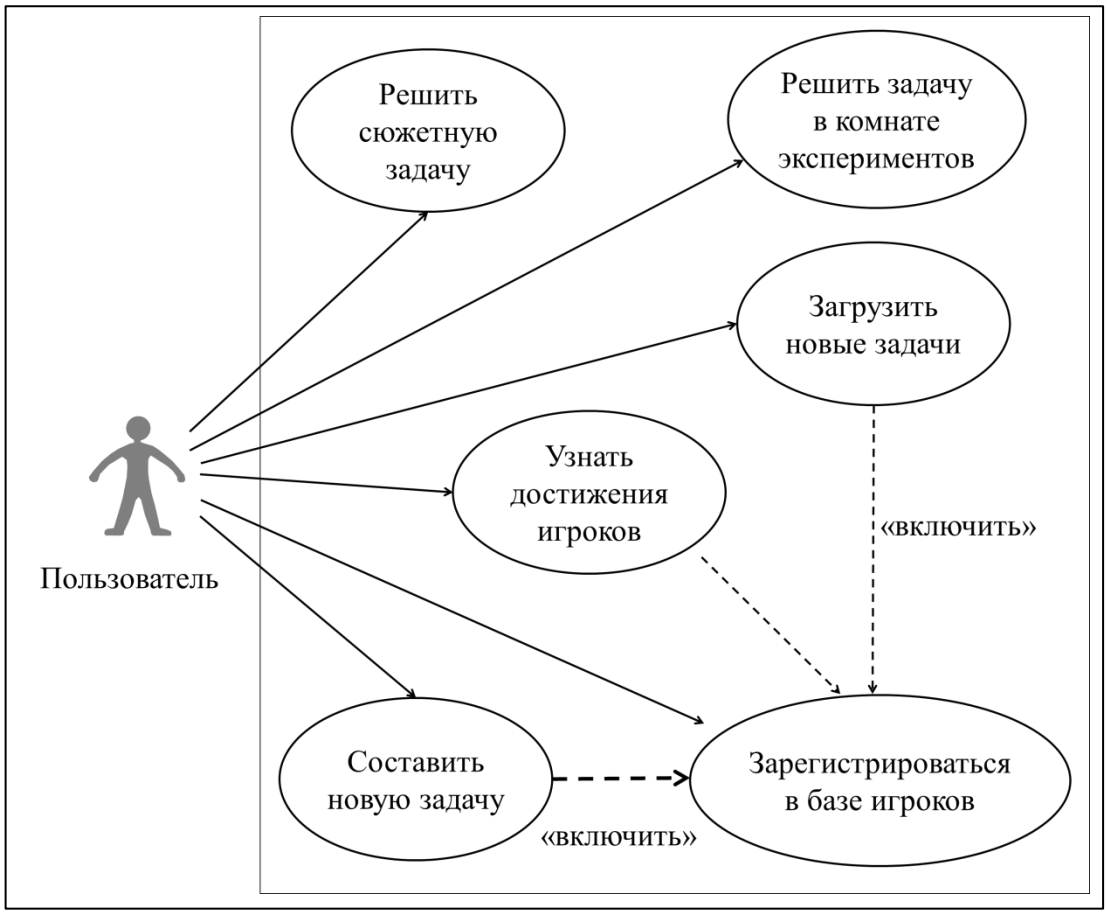

Рис. 2.4. Схема вариантов использования

Стрелками на рисунке 2.5 обозначены взаимосвязи отдельных модулей, где направление стрелки указывает на направление передачи основных пакетов, данных между модулями.

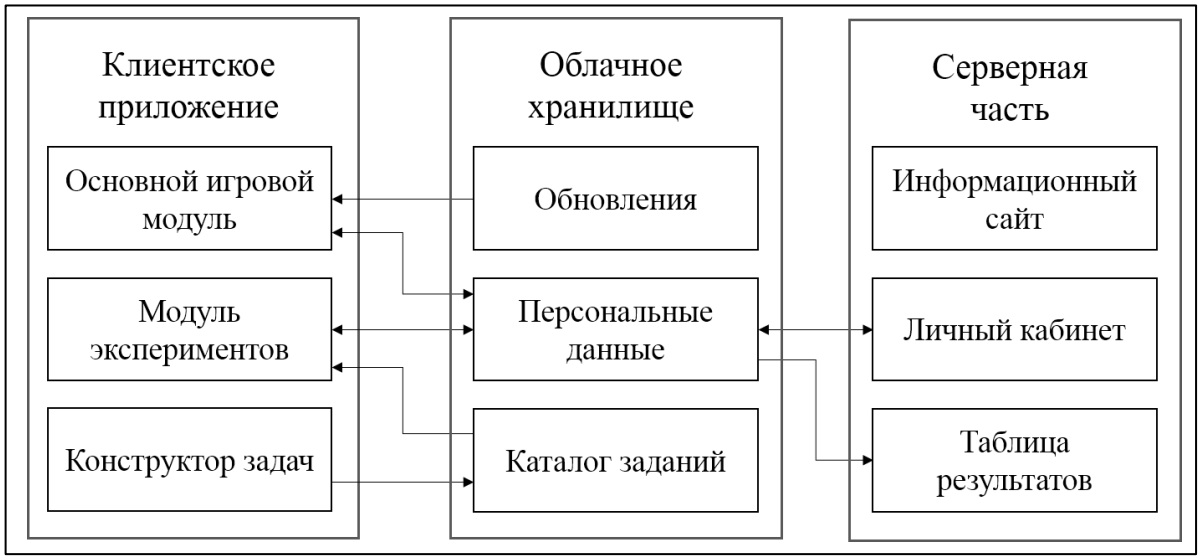

Рис. 2.5. Общая схема проекта

2.2.1. Клиентское приложение

Сюжетную композицию предоставляет разработчик системы, это задачи, которые завязаны на сценарии игры с промежуточными кат-сценами. Кат-сцены представляют собой «фильм» в игре, где игрок выступает в роли наблюдателя [12]. Такие сцены предназначены для развития сюжетной истории и используются в промежутках между выполнением заданий, чтобы ввести пользователя в курс дела.

Чтобы не нарушать сюжетную целостность и стабильность системы, локации, на которых происходят действия сюжетной линии, невозможно изменять извне. Сюжетные задачи специфичны и составляются под определённую локацию.

Дополнительно к сюжетной композиции пользователю предоставляются комнаты экспериментов. Это простые универсальные игровые локации, предназначенные для более подробного рассмотрения задач из выделенной предметной области. Задачи в таких комнатах не имеют сюжетной составляющей и могут составляться при помощи конструктора уровней.

Конструктор задач для комнат с экспериментами, позволяет составлять индивидуальные задачи в произвольной комнате. Игровые объекты доступные из конструктора соответствуют набору из сюжетной композиции с возможностью настройки взаимосвязей.

2.2.2. Серверная часть

Сайт позволяет отображать актуальную информация о приложении, и о результатах прохождения других игроков, вести статистику прохождения заданий в виде таблицы с результирующими очками, а также предоставлять доступ для скачивания актуальной версии приложения или обновлений.

Модуль доступа к данным облачного хранилища выполняет работу с клиентским приложением, позволяя: регистрировать новых игроков, передавать результаты достижений в виде итоговой таблицы или списка клиентскому приложению, выполнять управление данными учётной записи с приложения клиента, собирать общую и частную статистику по достижениям для отображения у пользователя.

2.2.3. Облачное хранилище

Облачное хранилище представляет собой обычный FTP сервер. Новые задания для комнат экспериментов хранятся в специальном каталоге, позволяя пользователю в удобное время загружать их на клиентское приложение. Формат хранения заданий представляют собой файл с описанием игровой сцены созданной в конструкторе уровней.

2.3. Взаимодействие игровых объектов

Отталкиваясь от особенностей построения игр, можно считать, что игровой мир представляет собой некоторый набор объектов произвольной природы (игровых объектов) и правил взаимодействия таковых. Игрок – также является одним из этих объектов, при этом всегда можно указать набор таких элементов, поведение которых зависит от действий игрока в противоположность независимо дейст-

вующим элементам. Это позволяет выделить основные типы объектов, различающихся по принципу взаимодействия:

– неактивные – не проявляющие самостоятельной деятельности вне воздействия на них;

– активные – проявляют самостоятельную деятельность вне воздействия на них;

– статические – неподвижные объекты сцены.

В данном аспекте у разработчика есть большое поле для реализации различных возможностей, так как виртуальное пространство позволяет не только моделировать элементы естественного мира, но и составлять произвольные абстракции или интерпретации, для представления невидимых не вооружённым глазом явлений.

В игровом пространстве существует следующие виды взаимодействий с объектами [13]:

1) метод трассировки лучей – это метод, где от какой-нибудь точки выпускается луч на определённое расстояние и с помощью геометрических алгоритмов определяется объект его пересечения;

2) область вхождения «триггер» – невидимая геометрическая фигура, которая реагируют на пересечение с геометрией другого объекта, что вызывает предусмотренные разработчиком действия;

3) соприкосновение объектов «коллизия» – геометрическая фигура, которая при контакте с другим объектом генерирует событие контакта. Для отслеживания таких соприкосновений используются специальные программные механизмы игрового движка.

На основании классификации определим используемые в разрабатываемой системе основные игровые объекты.

Активные:

– Globe – сферический мобильный робот помощник, выполняющий различные игровые функции;

– GlobeCam – автоматизированная камера наблюдения, ведущая слежку за объектом попадающим в область зрения (игрок или сферический робот);

– персона игровой лаборатории – люди, занимающиеся своими делами.

Неактивные объекты:

– активаторы – запускают определённые сценарии;

– игровой персонаж – робот-андроид по имени PHY;

– лифты – поднимают или опускают игрока на определённый этаж;

– панель управления – элемент игрового пространства с интерактивным пользовательским интерфейсом;

– двери – автоматически открываются при приближении, могут активироваться игроком, сферическим роботом и активаторами;

– люки – открываются или закрываются в зависимости от активации управляющего устройства;

– устройство создания электромагнитного поля – электромагнитная катушка высокой мощности, контролируемая с панели управления или кнопки;

– кнопки (для игрока и грузовые) – напольная платформа, активирующаяся постановкой на неё сферического робота или игрока.

На рисунке 2.6 представлена схема взаимодействия выделенных объектов, что позволит явно выделить основные внутренние игровые элементы и их отношения.

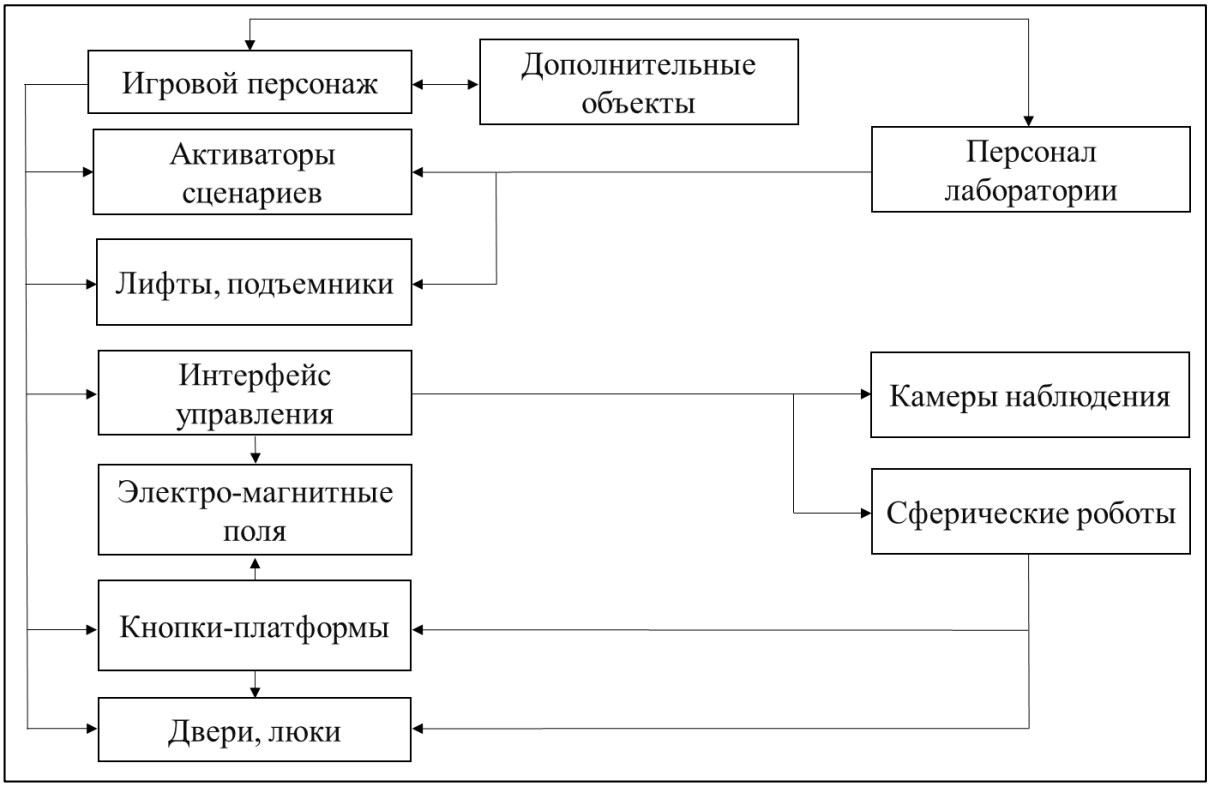

Рис. 2.6. Объектная схема проекта

2.4. Выбор инструментальных средств

Разработка данного приложения потребовало использования множества программных продуктов, так как количество обрабатываемых данных и используемых форматов файлов относительно велико. В приложении задействованы такие данные, как текстовая информация, изображения, аудиодорожки, трёхмерные модели, анимации и т.д., которые требуют разных инструментов обработки.

2.4.1. Центральная система разработки

Чтобы выбрать центральную систему разработки обратимся к статистике [14] популярности игровых движков, представленную в виде гистограммы на рисунке 2.7, среди разработчиков игр в небольших компаниях. Среди всего множества можно выделить таких фаворитов как Unreal Engine и Unity.

В качестве лучшей платформы для разработки данного проекта использован Unity в виду того, что данный инструмент обладает следующими преимуществами:

1) лёгкая интеграция с другими инструментами;

2) широкие возможности импорта, поддержка формата объектов FBX;

3) кроссплатформенность;

4) наличие бесплатных модулей расширения;

5) поддержку языка программирования C#;

6) возможность отладки кода в Visual Studio, которая предоставляется бесплатно вместе с Unity.

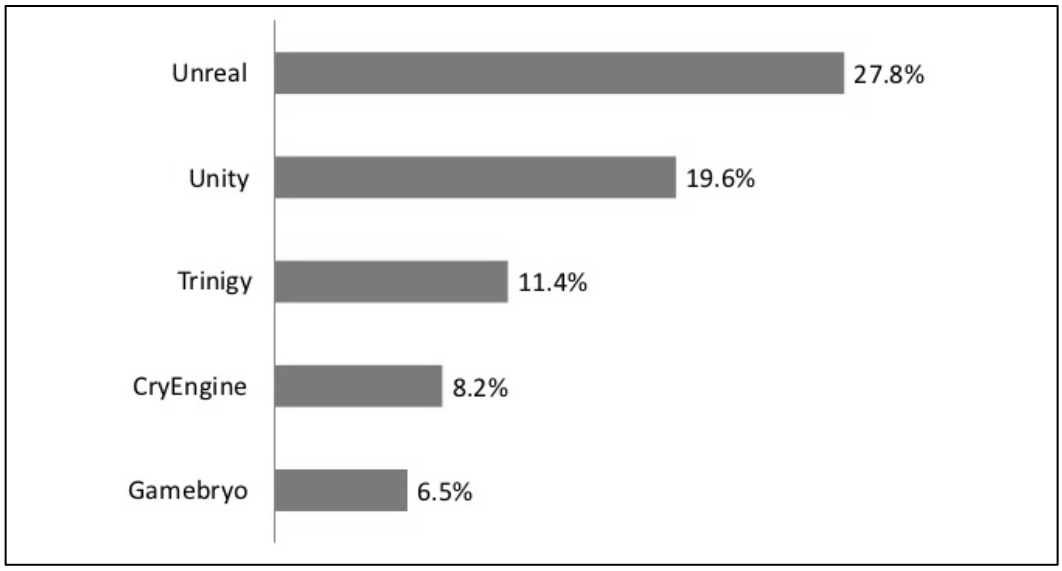

Рис. 2.7. Статистика популярности игровых движков

Unity – это инструмент для разработки двух- и трёхмерных приложений, и игр, работающий под операционными системами Windows, OS X. Созданные с помощью 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. Есть возможность создавать приложения для запуска в браузерах с помощью специального подключаемого модуля Unity (Unity Web Player), а также с помощью реализации технологии WebGL. Поддерживает программный код на языках программирования C#, Java и Boo.

2.4.2. Язык программирования

C# был выбран в качестве основного, так как это язык программирования, поддерживаемый Unity с полным доступом к базовому API. Unity поддерживает возможности .NET Framework (2.0) для всех платформ и до 4.5 для приложений Windows. C# относится к семье языков с C-подобным синтаксисом. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ.

2.4.3. Среда разработки

Visual Studio – продукт компании Майкрософт, включающий интегрированную среду разработки программного обеспечения и ряд других инструментальных средств, которая входит в бесплатный пакет, поставляемый вместе с Unity 5. Специальный компонент отладки для Unity – UnityVS позволяет контролировать поток работы программы.

2.4.4. 3D редактор

3ds Max – программная система от Autodesk для создания и редактирования трёхмерных моделей и их составляющих компонент. Работает в операционных системах Windows и Windows NT. Имеет возможность экспортировать модели в FBX – технология и формат файлов, использующийся для обеспечения совместимости различных программ трехмерной графики поддерживаемая Unity. 3ds Max имеет две лицензионные версии:

- студенческая – бесплатная (при наличии регистрации на сайте Autodesk);
- полная – коммерческая, стоимостью в $250.

2.4.5. Графический редактор

Paint.NET – бесплатный растровый графический редактор для Windows, разработанный на платформе .NET Framework, обладающий широкими возможностями. Данный продукт имеет ряд преимуществ:

1) бесплатный для распространения и использования;

2) обладает всем необходимым функционалом для создания и редактирования изображений;

3) поддерживает работу со слоями;

4) позволяет работать с несколькими документами одновременно.

2.4.6. Инструмент для обработки звука

GoldWave – бесплатный инструмент для работы с аудио. GoldWave обладает основными возможностями для обработки и редактирования звука. Поддерживает основные форматы аудиофайлов совместимые с Unity: WAV, MP3, AU, OGG и т.д. Имеет возможность записи звука с внешних устройств через аудио-карты или микрофон. Имеет модуль Expression Evaluator, который позволяет, используя стандартные математические функции, генерировать звуковой сигнал или эффект-процессор.

2.4.7. Облачное хранилище

Облачное хранилище представляет собой обычный FTP сервер. Новые задания для комнат экспериментов хранятся в специальном каталоге, позволяя пользователю в удобное время загружать их на клиентское приложение. Формат хранения заданий представляют собой файл с описанием игровой сцены созданной в конструкторе уровней.

В качестве платформы для хранилища результатов прохождения игроков и развёртывания сайта был выбран сервис Hostinger.ru, который предоставляет услуги по разным тарифным планам. Для реализации системы был выбран бесплатный тариф.

К особенностям данного сервиса при выбранном тарифном плане можно отнести:

- бесплатный домен;

- поддержка языка PHP;
- поддержка базы данных MySQL;
- 2GB места на диске;
- 100 GB трафика в день;
- бесплатный конструктор сайтов;
- средства администрирование phpMyAdmin;
- доступ по FTP протоколу;
- сервис POP3/IMAP.
- нет всплывающих окон, баннеров и рекламы от поставщика услуг.

Для доступа к различным сервисам и конфигурациям предусмотрена панель управления.

2.5. Выводы по разделу

На данном этапе работы была определена концепция приложения, разработана архитектура системы, выделены основные компоненты, определены игровые объекты и методы их взаимодействия, определена структура представления данных. Также исходя из формата приложения и направления его развития были выбраны инструменты разработки.

ГЛАВА 3. СПОСОБЫ ВЗЛОМА UNITY ПРИЛОЖЕНИЙ

Самым распространённым явлениями взлома приложения являются: пиратство, фальсификация результатов, вмешательство в процесс игры и извлечение ресурсов из приложения. Рассмотрим основные проблемы защиты приложений, разработанных на Unity.

Unity не имеет открытого кода в свободном доступе и скомпилированные проект работает как «черный ящик» [15]. Единственный способ понять, что происходит внутри стороннего приложения – это изучение конечной версии, используя специальные средства.

Однако, как показала практика Unity не имеет встроенного механизма шифрования или подписи файлов, что позволяет с лёгкостью извлекать интересуемые активы и использовать их на своё усмотрение. Такой сценарий не благоприятен для разработчика, так как другие не добросовестные разработчики могу использовать контент в свою пользу.

При сборке игры все объекты, текстуры и аудиодорожки будут запакованы в файлы sharedassets(n).assets, где (n) это номер сцены или дополнительные файлы с расширением (m).resources, где (m) имя ресурса.

Приложение PhysiLabs имеет в себе большой объём ресурсов, как заложенных самим разработчиком, так и сгенерированных пользователем. Получение злоумышленниками доступа к исходным ресурсам может повлечь нарушение авторского права или компрометацию приложения. Как говорилось ранее, Unity не обладает встроенными средствами защиты, поэтому разработка унифицированного средства данного назначения является хорошим подспорьем, как для проекта PhysiLabs, так и для других разработчиков.

3.1. Извлечение текстур и шейдеров

Шейдер – это подпрограмма, предназначенная для исполнения процессорами видеокарты, моделирующая оптические свойства заданных материалов [16].

Популярным на сегодняшний день инструментом для просмотра и извлечения файлов текстур и шейдеров из Unity является Unity Assets Explorer, изображённый на рисунке 3.1. Полученные файлы текстур будут иметь формат DDS, который можно «прочитать» с помощью Windows Texture Viewer.

Шейдеры извлекаются в скомпилированном виде и решений для автоматической трансляции в удобочитаемый формат пока не существует. Тем не менее, это не мешает импортировать и использовать полученные шейдеры в другом проекте в исходном виде.

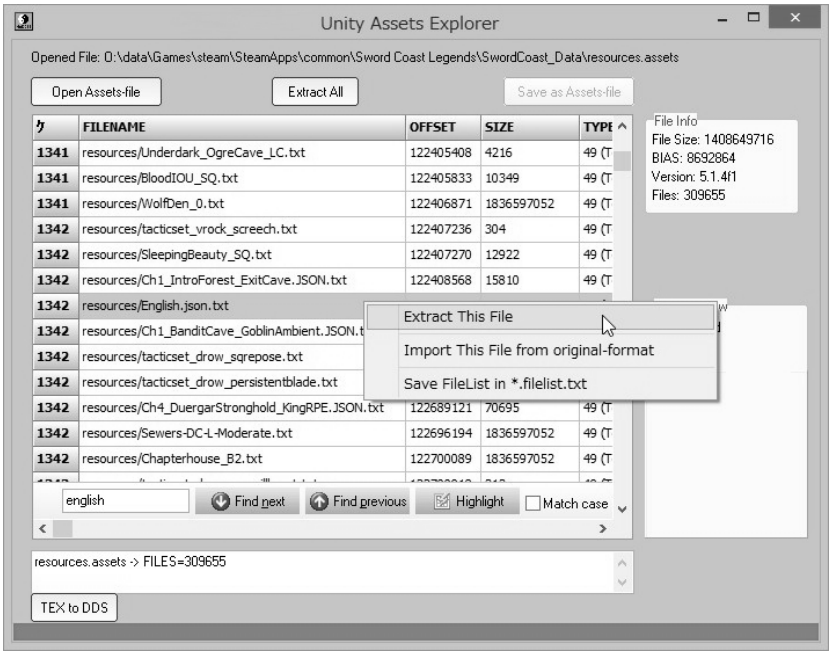

Рис. 3.1. Unity Assets Explorer

3.2. Извлечение 3D-моделей

Трёхмерные модели как правило распределены по различным ресурсам, а некоторые могут генерироваться во время выполнения. Разбор файлов в данном случае затруднителен, но существует альтернативный способ получить данные о геометрии, прочитать их из памяти графического ускорителя. Когда приложение запущена, вся информация о текстурах и моделях текущей сцены, загружены в память видеокарты. С помощью утилиты 3D RIPPER, изображённой на рисунке 3.2 можно извлечь эту информацию и сохранить в формате, который можно открыть 3D-редактором, например, 3DS Max.

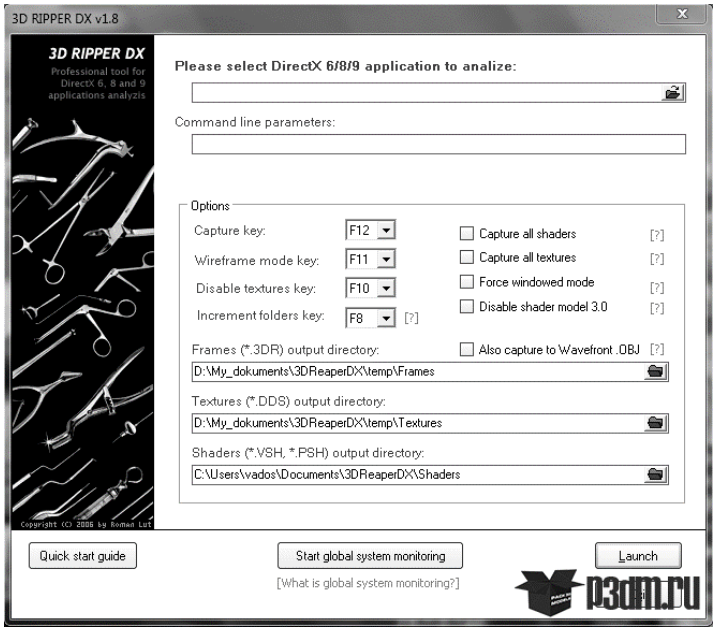

Рис. 3.2. 3D RIPPER

3.3. Модификация сохранённых данных пользователя

Стандартным и универсальным средством для сохранения и чтения данных пользователя на текущем компьютере в Unity является класс PlayerPrefs. Перечень данных, определённый разработчиком сохраняется между игровыми сеансами в реестре. Часто используется для хранения различных настроек, достижений, прогресса игрока и другой информации о состоянии игры.

В ОС Windows данные PlayerPrefs хранятся в реестре в разделе HKEY_CURRENT_USER\Software\[название компании]\[имя продукта], где имена компаний и продуктов - это имена, заданные в настройках проекта, как показано на рисунке 3.4.

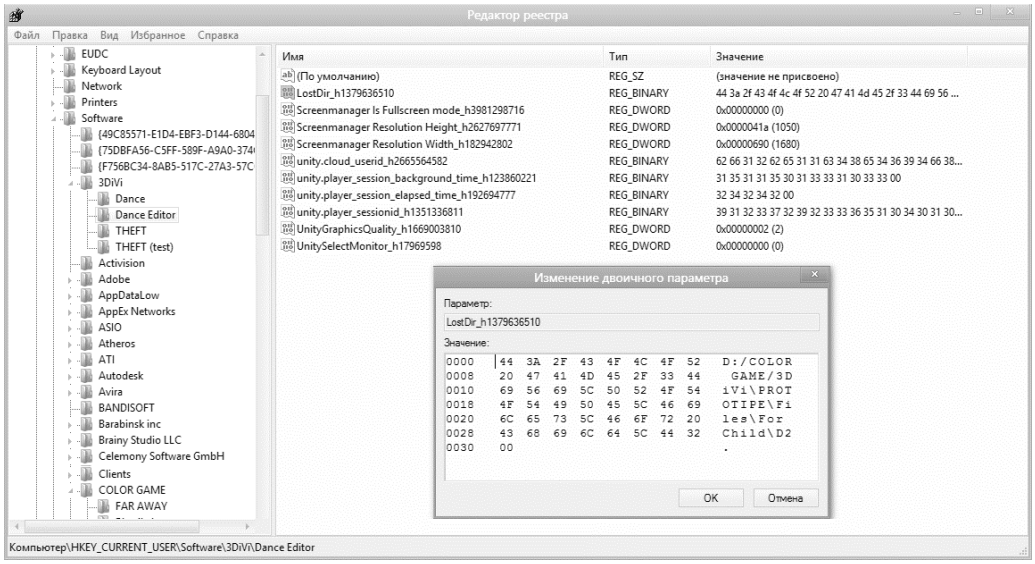

Рис. 3.4. Данные приложения в реестре

Не смотря на удобство, сохранённые данные полностью открыты и доступ к ним не представляет труда. С помощью стандартной утилиты Windows regedit можно модифицировать значения PlayerPrefs, изменяя тем самым конфигурацию и статус игры выполняя мошеннические действия в приложении.

3.4. Извлечение исходного кода

Для Windows-сборок Unity компилирует и сохраняет исходный код всех игровых скриптов в директорию приложения Managed, в файлах библиотек: Assembly-CSharp.dll, Assembly-CSharp-firstpass.dll и Assembly-UnityScript.dll.

Для декомпиляции и просмотра managed-кода .NET библиотек существуют удобные и бесплатные утилиты: IlSpy, как на рисунке 3.5 и dotPeek.

Данных подход особенно эффективен, так как Unity не оптимизирует исходный код игровых скриптов, практически не изменяя его структуру, а также не скрывает названия переменных. Это позволяет с легкостью читать и понимать код после декомпиляции.

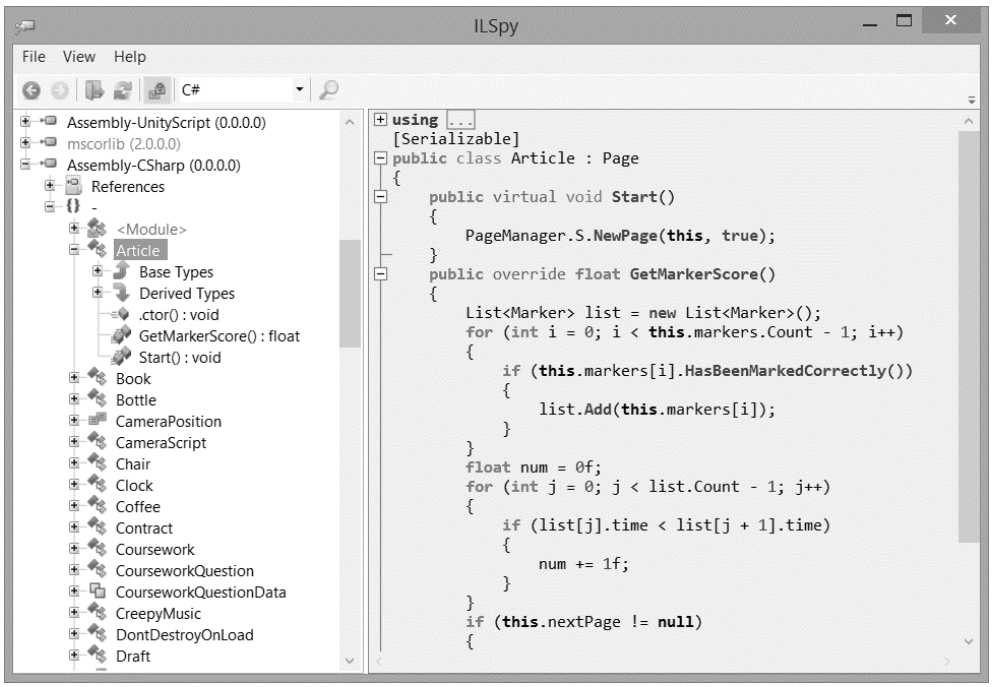

Рис. 3.5. Код после декомпиляции в редакторе IlSpy

3.5. Чтение памяти игры

Любое приложение, запущенное на ОС Windows получает в своё распоряжение область оперативной памяти для хранения ресурсов и выполняемого кода. Cheat Engine – программа, изображённая на рисунке 3.6, находит область оперативной памяти, принадлежащей процессу запущенного приложения, позволяя произвольно её изменять.

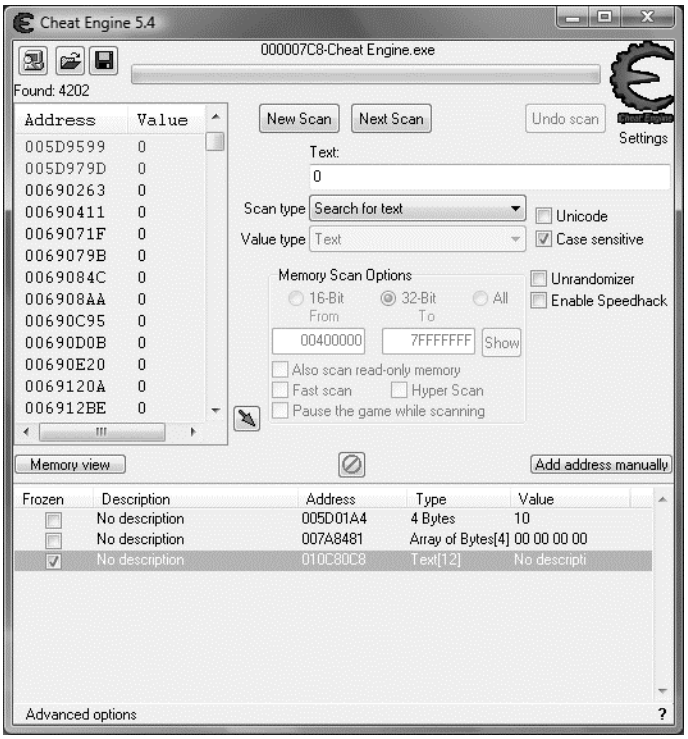

Рис. 3.6. Cheat Engine

Программа пользуется тем фактом, что разработчики редко защищают значения переменных. Например, имеется некая переменная имеющая значение money = 100 монет; используя Cheat Engine, можно выполнить поиск участков памяти, которые хранят значение «100». После траты одной монеты значение переменной составит 99 единиц, повторным сканированием ищется участок хранящий значение «99». После нескольких подобных итераций можно обнаружить расположение большинства переменных игры и произвольно их изменять, тем самым нарушая адекватность данных в приложении.

3.6. Классификация данных по релевантности угроз

С учётом всех выявленных способов мошеннических действий над приложением можно выделить основные типы данных в приложении целостность которых [17]:

1) Не влечёт угрозу мошеннической операции. К данному типу можно отнести большинство внутренних переменных, изменение которых лишь вызовет неадекватное поведение игровых объектов, сделав невозможным перемещение на следующую локацию.

2) Влечёт угрозу мошеннической операции, но есть возможность восстановить исходные значения. Подобные значения легко восстановить повторным вычислением на стороне клиента или получение последней копии с резервного сохранения или со стороны сервера. Это могут быть значения различных коэффициентов, большинство данных пользователя, касающихся прогресса игры.

3) Влечёт угрозу мошеннической операции, восстановление исходных значений не возможно, но возможна верификация. В основном это данные передаваемые на сторону сервера, значения сеансовых ключей. В случае, если верификация не пройдена, то данные не учитываются.

4) Влечёт угрозу мошеннической операции, и верификация затруднительна или невозможна. К этому типу относятся ресурсы приложения и исходный код, извлечение которого не повлияет на работоспособность приложения, но может нарушить авторское право.

3.7. Выводы по разделу

В данном разделе была определены основные угрозы для приложений разработанных на Unity, также выделены основные направления по решению данных проблем. Защита файлов и данных от фальсификации и передачи третьем лицам является актуальной проблемой на всех платформах. Особенно это актуально для приложений, разработанных на Unity, так как этот продукт не предусматривает встроенных систем защиты ресурсов и данных, складывая все полномочия на разработчика. Однако, модульность и кроссплатформенность данного инструмента позволяет разрабатывать универсальные модули, которые можно будет внедрять и в другие проекты.

ГЛАВА 4. КРИПТОГРАФИЧЕСКАЯ МОДЕЛЬ НА ОСНОВЕ CRYPTOAPI .NET FRAMEWORK

Платформа .NET Framework предоставляет реализации многих стандартных алгоритмов шифрования. Эти алгоритмы просты в использовании и имеют максимально безопасные свойства по умолчанию. Кроме того, криптографическая модель наследования объектов, поточно-ориентированная структура и конфигурация платформы .NET Framework являются расширяемыми.

4.1. Наследование объектов

Система безопасности .NET Framework реализует расширяемую модель наследования производных классов [18]. Иерархия имеет представленный ниже вид.

– Класс типа алгоритма, например, SymmetricAlgorithm, AsymmetricAlgorithm или HashAlgorithm. Этот уровень является абстрактным.

– Класс алгоритма, наследующий от класса типа алгоритма, например, Aes, RC2 или ECDiffieHellman. Этот уровень является абстрактным.

– Реализация класса алгоритма, наследующего от класса алгоритма, например, AesManaged, RC2CryptoServiceProvider или ECDiffieHellmanCng. Этот уровень полностью реализован.

При помощи этой системы производных классов можно легко добавить новый алгоритм или новую реализацию существующего алгоритма. Например, чтобы создать новый алгоритм открытого ключа, можно унаследовать собственный класс от класса AsymmetricAlgorithm. Чтобы создать новую реализацию определенного алгоритма, необходимо создать неабстрактный производный класс этого алгоритма.

4.2. Выбор алгоритма

Алгоритм можно выбирать, исходя из различных причин, например, для обеспечения целостности данных, для обеспечения конфиденциальности данных или для создания ключа. Ниже приведен список рекомендуемых алгоритмов в зависимости от приложения.

Конфиденциальность данных:

– Aes.

Целостность данных:

– HMACSHA256.

– HMACSHA512.

Цифровая подпись:

– ECDsa.

– RSA.

Обмен ключами:

– ECDiffieHellman.

– RSA.

Генерация случайных чисел:

- RNGCryptoServiceProvider

Формирование ключа из пароля:

- Rfc2898DeriveBytes

В случае с асимметричными алгоритмами используется публичный ключ для шифрования и приватный для расшифровки. Если шифрованные сообщения расшифровываются только одним из участников, то закрытый ключ можно хранить только в одном месте. В случае с симметричным шифрованием закрытый ключ приходится делать известным всем участникам обмена сообщениями.

Асимметричные алгоритмы шифрования приблизительно в 100-1000 раз медленнее симметричных. Поэтому большие объёмы данных ими шифровать может оказаться накладно. Асимметричное шифрование может быть использовано вместе с симметричным. С помощью асимметричного шифрования генерируется сессионный ключ, который используется для шифрования данных. В .NET реализованы следующие основные алгоритмы:

- RSA;
- DSA (только для цифровых подписей);
- ECDiffieHellman.

Для RSA используют ключи длиной 1024, 2048, 4096 битов. Доказано, что RSA на ключе длиной 1024 — ненадёжен, так что следует использовать более длинные ключи.

Для защиты ресурсов от изъятия могут подойти классы ProtectedData или ProtectedMemory, однако этот класс предоставляет доступ к API защиты данных (DPAPI), доступному в операционных системах Microsoft Windows 2000 и более поздних версий, что не подходит для кроссплатформенного решения.

Лучшим выбором будет использование асимметричного шифрования, который лучше использовать для шифрования малых объемов данных, например, для шифрования симметричного ключа и вектора инициализации. Таким образом мы избавляемся от резкого возрастания сложности операции шифрования при увеличения размера исходных файлов. Для данных целей платформа .NET Framework предоставляет класс RSACryptoServiceProvider.

4.3. Шифрование данных

Общая среда исполнения использует дизайн ориентированный на поток для криптографии. Ядром этого дизайна является CryptoStream. Любые криптографические объекты, реализующие CryptoStream, могут быть соединены вместе с любыми объектами, которые реализуют Stream, поэтому потоковый вывод одного объекта может быть подан во вход другого объекта [19].

Пакет шифрования использует следующий формат:

- Длина ключа, байты 0–3.
- Длина вектора инициализации, байты 4–7.
- Зашифрованный ключ.

– Вектор инициализации.
– Зашифрованный текст

Общая схема шифрования данных изображена на рисунке 4.1, этап инициализации ключа приведён в листинге 1.

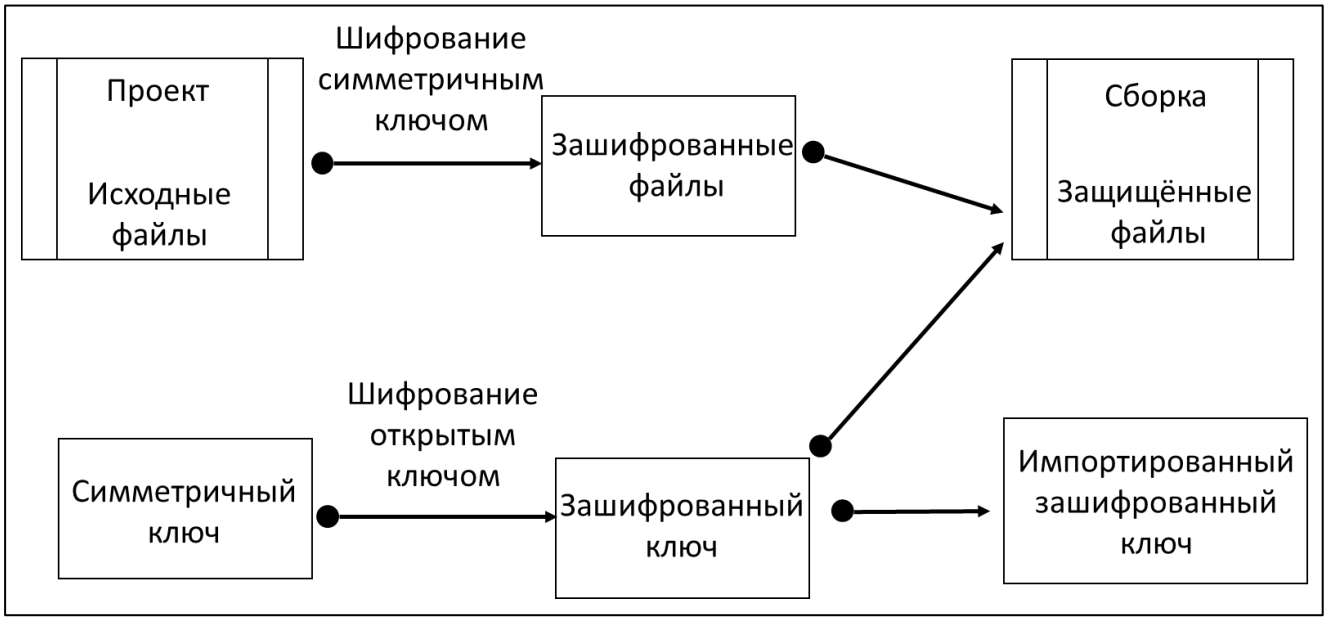

Рис. 4.1. Шифрование ресурсов при компиляции

Листинг 1. Инициализация ключа шифрования

```
CspParameters cspp = new CspParameters();
RSACryptoServiceProvider rsa;

cspp.KeyContainerName = keyName;
rsa = new RSACryptoServiceProvider(cspp);
rsa.PersistKeyInCsp = true;
```

Шифрование выполняется в следующей последовательности:

1) Инициализируется симметричный алгоритм RijndaelManaged для шифрования содержимого.

2) Создается объект RSACryptoServiceProvider для шифрования ключа RijndaelManaged.

3) Объект CryptoStream использует для чтения и шифрования исходного файла FileStream блоками байтов в конечный объект FileStream для зашифрованного файла.

4) Определяется длина зашифрованного ключа и вектора инициализации и создаются массивы байтов со значениями их длин.

5) Записываются ключ, вектор инициализации и значения их длин в зашифрованный пакет.

Класс RijndaelManaged является расширение Aes и относится в .NET Framework к пространству имен: System.Security.Cryptography, поддерживает

длину ключа 128, 192 или 256 бит; Имеет реализации на: .NET Core (2.1 RC1, 2.0), .NET Framework (4.7.2, 4.5), Стандарт .NET (2,0) Xamarin.Android (7,1), Xamarin.iOS (10.8), Xamarin.Mac(3.0). Реализация шифрования с использованиеи данного класса приведено в листинге 2.

Класс RSACryptoServiceProvider является расширением реализации RSA в CAPI, только RSACryptoServiceProvider отменяет порядок зашифрованного массива байтов после шифрования и до расшифровки. По умолчанию данные, зашифрованные классом RSACryptoServiceProvider, не могут быть расшифрованы функцией CAPI, а данные, зашифрованные методом CAPI, не могут быть расшифрованы классом RSACryptoServiceProvider.

Листинг 2. Шифрование с использованием сгенерированного ключа

```
RijndaelManaged rjndl = new RijndaelManaged();
rjndl.KeySize = 256;
rjndl.BlockSize = 256;
rjndl.Mode = CipherMode.CBC;

ICryptoTransform transform = rjndl.CreateEncryptor();

byte[] keyEncrypted = rsa.Encrypt(rjndl.Key, false);
byte[] LenK = new byte[4];
byte[] LenIV = new byte[4];

int lKey = keyEncrypted.Length;
LenK = BitConverter.GetBytes(lKey);
int lIV = rjndl.IV.Length;
LenIV = BitConverter.GetBytes(lIV);

using (FileStream outFs = new FileStream(outFile, FileMode.Create))
{
     outFs.Write(LenK, 0, 4);
     outFs.Write(LenIV, 0, 4);
     outFs.Write(keyEncrypted, 0, lKey);
     outFs.Write(rjndl.IV, 0, lIV);

    using (CryptoStream outStreamEncrypted = new CryptoStream(outFs,
      transform, CryptoStreamMode.Write))
    {
       int count = 0;
       int blockSizeBytes = rjndl.BlockSize / 8;
       byte[] data = new byte[blockSizeBytes];
       int bytesRead = 0;

       using (FileStream inFs = new FileStream(inFile, FileMode.Open))
       {
            do
            {
                 count = inFs.Read(data, 0, blockSizeBytes);
                 outStreamEncrypted.Write(data, 0, count);
                 bytesRead += blockSizeBytes;
            }
            while (count > 0);
            inFs.Close();
         }
```

```
            outStreamEncrypted.FlushFinalBlock();
            outStreamEncrypted.Close();
        }
        outFs.Close();
    }
```

4.4. Расшифровка ресурсов

Расшифровка ресурсов выполняется при обращении к ним из менеджера ресурсов Unity по идентификатору, указанному при компиляции. Для того чтобы не нарушать общую логику взаимодействия Unity с ресурсами, вместо зашифрованного файла в таблицу ассетов при компиляции отправляется файл-заглушка [20]. Данный файл имеет заголовок, определяющий его как ресурс для Unity, и ссылку на модуль расшифровки. Для Unity данный файл воспринимается как обычный, процесс подмены на расшифрованный происходит при предварительном обращении обработчика, при подготовке ресурса к загрузке. Расшифрованные файлы не хранятся на диске, а сразу конвертируются в требуемый ресурс и передаются в обработчик Unity. Данная схема добавляет в общий размер сборки 2 кб на каждый файл-заглушку и до 5% к размеру каждого зашифрованного файла, по сравнению с исходным.

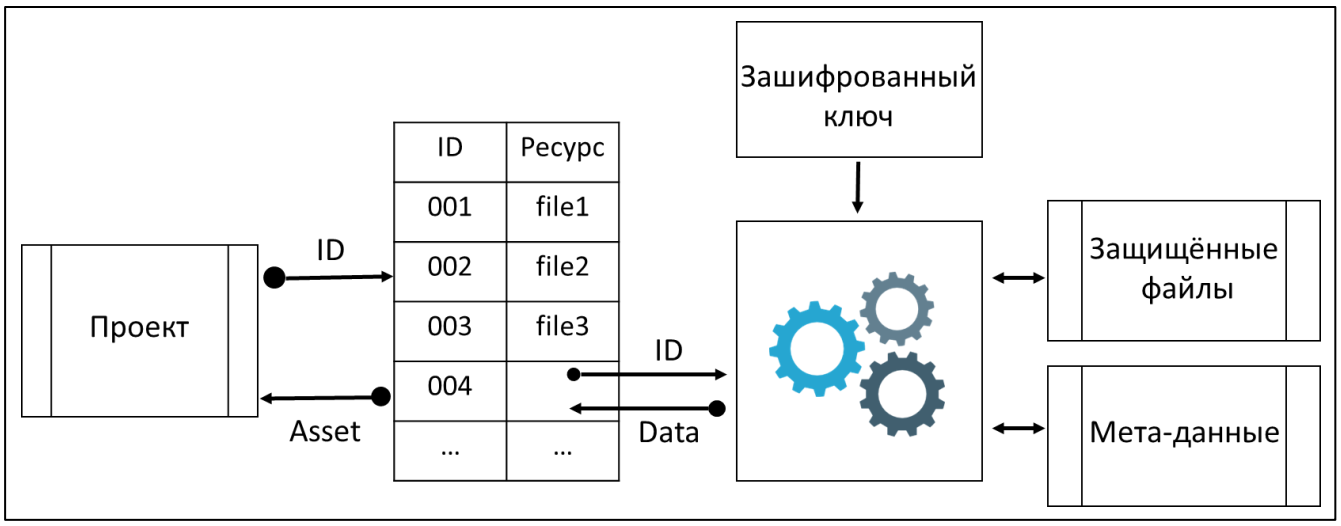

Рис. 4.2. Шифрование ресурсов при компиляции

При расшифровке, которая приведена в листинге 3, выполняется следующая последовательность действий:

1) Инициализируется симметричный алгоритм RijndaelManaged для расшифровки содержимого.

2) Считывается первые восемь байтов FileStream зашифрованного пакета в массивы байтов, чтобы изъять значения длин зашифрованного ключа и вектора инициализации.

3) Извлекается ключ и вектор инициализации из пакета шифрования в байтовые массивы.

4) Создается объект RSACryptoServiceProvider для расшифровки ключа RijndaelManaged.

5) Используется объект CryptoStream для чтения и расшифровки зашифрованного текста пакета шифрования FileStream блоками байтов в объект FileStream для расшифрованного файла. После завершения этой операции расшифровка считается завершённой.

Листинг 3. Расшифровка

```
RijndaelManaged rjndl = new RijndaelManaged();
rjndl.KeySize = 256;
rjndl.BlockSize = 256;
rjndl.Mode = CipherMode.CBC;

byte[] LenK = new byte[4];
byte[] LenIV = new byte[4];

string outFile = DecrFolder + inFile.Substring(0, inFile.LastIndexOf(".")) + ".txt";

using (FileStream inFs = new FileStream(EncrFolder + inFile, FileMode.Open))
{
    inFs.Seek(0, SeekOrigin.Begin);
    inFs.Seek(0, SeekOrigin.Begin);
    inFs.Read(LenK, 0, 3);
    inFs.Seek(4, SeekOrigin.Begin);
    inFs.Read(LenIV, 0, 3);

    int lenK = BitConverter.ToInt32(LenK, 0);
    int lenIV = BitConverter.ToInt32(LenIV, 0);

    int startC = lenK + lenIV + 8;
    int lenC = (int)inFs.Length - startC;

    byte[] KeyEncrypted = new byte[lenK];
    byte[] IV = new byte[lenIV];

    inFs.Seek(8, SeekOrigin.Begin);
    inFs.Read(KeyEncrypted, 0, lenK);
    inFs.Seek(8 + lenK, SeekOrigin.Begin);
    inFs.Read(IV, 0, lenIV);
    Directory.CreateDirectory(DecrFolder);

    byte[] KeyDecrypted = rsa.Decrypt(KeyEncrypted, false);

    ICryptoTransform transform = rjndl.CreateDecryptor(KeyDecrypted, IV);

    using (FileStream outFs = new FileStream(outFile, FileMode.Create))
    {
        int count = 0;
        int offset = 0;
        int blockSizeBytes = rjndl.BlockSize / 8;
        byte[] data = new byte[blockSizeBytes];
        inFs.Seek(startC, SeekOrigin.Begin);

        using (CryptoStream outStreamDecrypted = new CryptoStream(outFs,
        transform, CryptoStreamMode.Write))
        {
            do
            {
                count = inFs.Read(data, 0, blockSizeBytes);
```

```
                offset += count;
                outStreamDecrypted.Write(data, 0, count);
            }
            while (count > 0);

            outStreamDecrypted.FlushFinalBlock();
            outStreamDecrypted.Close();
        }
        outFs.Close();
    }
    inFs.Close();
}
```

4.5. Конвертирование ресурсов приложения

Одним из способов защита контента в Unity, является использование типа TextAsset для хранения данных в виде массива байт. Можно зашифровать нужные файлы, сохранив их с расширением .bytes, которое Unity будет воспринимать как тип TextAsset. После импортирования в редактор, данные файлы будут включены в пакет AssetBundles как TextAssets. Затем в конечном приложении из AssetBundle, будут извлечены необходимые пользователю данные [21]. При таком подходе можно применить алгоритм шифрования не AssetBundles, а к файлам типа TextAsset, находящиеся в AssetBundles. Это позволяет предварительно зашифровать файлы, а при загрузке сцены расшифровать и восстановить в требуемый ассет.

Например, для преобразования исходного изображения в байтовое представление необходимо выполнить всего одну операцию, которая приведена в листинге 4.

Листинг 4. Преобразования изображения

```
Texture2D sourceTexture;  //исходная текстура (изображение)
byte[] outData = sourceTexture.GetRawTextureData();
```

Обратное преобразование произвольного массива байт в изображение будет выглядеть в Unity как показано в листинге 5:

Листинг 5. Восстановление изображения

```
byte[] sourceByte; //входной массив байт, представляющий изображение
int weight = 480;
int height = 270;

Texture2D texture = new Texture2D(weight, height, TextureFormat.ARGB32, false);
texture.LoadRawTextureData(sourceByte); //заполнение текстуры пикселями из sourceByte
texture.Apply();  //применение внесённых значений

Sprite outImage = Sprite.Create(texture, new Rect(0, 0, weight, height), Vector2.one);
```

Дальнейшие преобразования над байтами могут иметь произвольный характер, так как нет жёсткой привязки к формату данных. Также данный сценарий открывает возможность последовательного представления доступа к уровням приложения на системном уровне, выдавая приложению ключи расшифровки каждой локации по определённому условию. Даже если злоумышленнику получится нарушить прогресс прохождения сюжетной композиции, то следующие локации не будут загружены из-за некорректного формата ресурсов.

4.5. Конвертирование пользовательских данных

Для данных пользователя можно использовать аналогичное шифрование пакета данных, что и для ресурсов приложения: скомпоновать все сохраняемы данные пользователя в обобщённый сериализуемый класс, и конвертировать в массив байт или Json формат. Далее с данными в таком виде можно выполнять операции шифрования и расшифровки. Хранение в реестре станет безопасным, так как пользователь не сможет восстановить исходный формат и произвести в нём изменения. Для увеличения надёжности системы выдачу одноразовых ключей следует выполнять со стороны сервера.

Метод-расширение сериализации в байты для класса RoomPack будет выглядеть следующим, как показано в листинге 6. Обратная операция – десериализация, приведена в листинге 7.

Листинг 6. Метод-расширение для сериализации класса RoomPack

```
public static byte[] ToByte(this RoomPack data)
{
    BinaryFormatter formatter = new BinaryFormatter();

    using (MemoryStream ms = new MemoryStream())
    {
        formatter.Serialize(ms, data);
        return ms.ToArray();
    }
}
```

Листинг 7. Метод-расширение для десериализация класса RoomPack

```
public static RoomPack ToRoomPack(this byte[] data)
{
    MemoryStream ms = new MemoryStream();
    BinaryFormatter formatter = new BinaryFormatter();

    ms.Write(data, 0, data.Length);
    ms.Seek(0, SeekOrigin.Begin);

    RoomPack outData = (RoomPack)formatter.Deserialize(ms);

    return outData;
}
```

4.6. Выводы по разделу

В данном разделе была разработана криптографическая модель защиты ресурсов приложения и пользовательских файлов, определены алгоритмы шифрования и расшифровка данных, а также способы преобразования исходных ресурсов. Разработанная на основе CryptoAPI модель разделима на специальные модули, компоненты которых будут включены как в скомпилированный проект, так и в среду редактора.

ГЛАВА 5. РЕАЛИЗАЦИЯ СИСТЕМЫ «PHYSILABS»

5.1. Структура пользовательских данных на локальном устройстве

Приложение предусматривает сохранение параметров прошлого сеанса и основных данных игрока, поэтому для этих целей используется внешний файл с именем «Prusse.rtm», который хранится в корневой папке с игрой. Формат хранения представляет собой XML структуру, а сам файл дополнительно шифруется, чтобы исключить умышленное редактирование некоторых параметров.

Данные записываемые в файл:

- ID – (строка) идентификатор пользователя в базе игроков в облачном хранилище (является уникальным и автоматически генерируется серверной частью при регистрации в базе);
- NickName – (строка) кличка игрока, выбирается самим игроком при регистрации в базе игроков и является уникальной, используется для отображения таблицы прохождения;
- Name – (строка) имя пользователя (не является обязательным для заполнения), используется для отображения у друзей пользователя;
- Surname – (строка) фамилия пользователя (не является обязательным для заполнения), используется для отображения у друзей пользователя;
- DOPInfarmation – (строка) дополнительная информация по игроку, которая может включать класс или группу пользователя (не является обязательным для заполнения);
- HaveServerConnect – логическая переменная, определяющая, использование соединения с серверной частью для получения и отправления данных по прохождению уровней;
- TimeLastPlay – (дата в полном формате) содержит время последнего сеанса, используется для определения перерыва и отдыха от игрового процесса.
- TimeLimits – (число) значение в минутах максимальной продолжительности одного игрового сеанса, по истечении которого игроку будет предложено сделать перерыв и отдохнуть;
- Password – (строка) зашифрованный пароль при установленном родительском контроле, который организует доступ с паролем к некоторым настройкам игрового процесса;
- TimeInGame – (число) содержит общее время в минутах проведённое в игре;
- LastLevel – (число) номер последнего достигнутого уровня;
- LevelList – список пройденных уровней с статистическими данными.

Файл считывается при запуске, или при переходе в основное меню, данные записываются при достижении контрольной точки сохранения, изменении настроек и при выходе.

Остальные данные хранятся непосредственно в самих классах, а некоторые поля являются сериализуемый и значения присваиваются через инспектор игрового движка при разработке [22]. Эти данные могут включать в себя игровые объ-

екты, файлы анимации, аудио дорожки, материалы, текстуры, строки, числовые значения, ссылки на другие объекты и т.д.

5.2. Структура файлов задач

5.2.1. Полная структура файла

Весь файла задачи разделен на блоки сгруппированные по признаку принадлежности описания выделенных данных: MainInfo (Таблица 1), который содержит общую информация о текущей задаче; Objects (Таблица 2), являющийся массивом с перечислением всех объектов на сцене и их взаимосвязей с заданными для них параметрами; UIModule (Таблица 3) представляющий собой массив с кортежами всех универсальных интерфейсных модулей для управления заданными пользователем объектами.

Таблица 1. Блок MainInfo

Название поля	Описание
Id	Уникальный идентификатор задачи, который генерируется и присваивается сервером при загрузке задачи в магазин
MinVersionApp	Минимальная версия приложения, в которой может быть развёрнута данная задача, данные берутся от версии в которой задача была составлена
Name	Название задачи, которое задаётся автором
ScreenImage	Небольшое изображение (320x180), для представления в магазине и в приложении, создаётся автором из конструктора по нажатию соответствующей загрузки
ShotInfo	Краткое описание задачи и предметной области
TimeComplate	Среднее время на решение задачи, если задача решена быстрее указанного времени, пользователь получит дополнительные очки
RoomID	Номер одной из комнат-заготовок для развёртывания задачи
AreaGame	Область, рассматриваемая в задаче: физика (динамика), физика (статика), механика, робототехника и т.д

Таблица 2. Блок Objects

Название поля	Описание
Id	Уникальный идентификатор объекта, который генерируется в конструкторе при составлении задачи
NameObj	Имя объекта, которое может отображаться в менеджере целей
Type	Тип объекта, задаёт текущий объект
Vector3	Позиционирование объекта, позиция в мировом пространстве
Quaternion	Поворот объекта в мировом пространстве
NameParent	Родитель текущего объекта к которому прикреплёт текущий объект (позиция и вращение будут влиять в порядке иерархии)
ActID [Array]	Список идентификаторов объектов, действие которых будет выполнено при активации текущего объекта
CanID [Array]	Список идентификаторов объектов, действие которых будет выполнено при деактивации текущего объекта
Info	Дополнительная информация о объекте (может содержать подсказки или сведения, которые отображаются в менеджере целей)

Таблица 3. Блок UIModule

Название поля	Описание
<UI – parentTr – Tr>	Элемент интерфейса \ родитель, к которому прикреплён данный элемент \ позиция в мировом пространстве
<UI – action – toObject>	Элемент интерфейса \ действие, которое пользователь может совершить с данным элементом для его активации \ целевой объект, которому будет передана простая команда вида (активен \ неактивен). Данным образом устанавливается связь между интерфейсом и игровыми объектами
<UI – text – params>	Элемент интерфейса \ отображаемый текст \ дополнительные параметры

<Object – toObject>	Последовательность, объект действие которого должно активировать целевой объект

Для компоновки задачи используется пакет, который содержит в себе следующую информацию:

- localIDAutor (локальный идентификатор автора)
- localKey (локальный ключ лицензии)
- globalKey (глобальный ключ лицензии)
- levelFile (файл задачи)

Глобальный ключ лицензии генерируется на этапе загрузки задачи в облачное хранилище. При загрузке задачи пользователем с облака на основе глобального ключа и уникальном идентификаторе пользователя генерируется локальный ключ, конкретно для данного пользователя. Файл задачи шифруется с использованием обоих ключей [23].

5.2.2. Разделённая структура файла задачи

Данные о задаче можно разделить на открытые и закрытые. К первому типу можно отнести блок MainInfo, в котором содержится обще доступная информация, отображаемая в приложении при просмотре комнат с экспериментами. Ко второму относятся блоки Objects и Goals, которые являются наполнением задачи, описывая объекты и их состояния.

Перед загрузкой в облачное хранилище задача разделяется на два файла: пред просмотра и содержания; имеющие типы «.rpp» (room package preview) и «.rpd» (room package data) соответственно. Оба файла являются бинарными [24]. Ко второму применяется шифрование, чтобы избежать использование другими пользователями без закреплённой лицензии.

Полностью задача хранится на компьютере пользователя если он является её автором или задача была скачена с облачного хранилища. Формат файла «.rpf» (room package full), по сути является упорядоченным представлением «.rpp» и «.rpd».

5.3. Публикация и загрузка задач

На рисунке 5.1 изображена условная последовательность действий загрузки задачи в облачное хранилище.

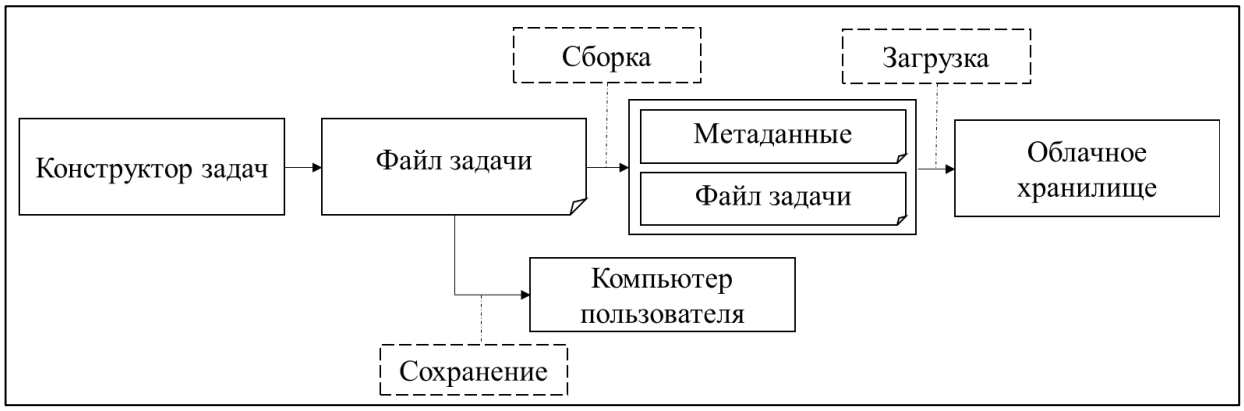

Рис. 5.1. Загрузки задачи в облачное хранилище

- Сборка – это процесс компоновки составленной задачи в универсальный файл с перечислением всех требуемых параметров.
- Загрузка – помещение файла на диск облачного хранилища.
- Сохранение – запись файла на диск пользовательского ПК, если разработка задачи не была завершена.

Последовательность действий при загрузке новой задачи с облака изображена на рисунке 5.2

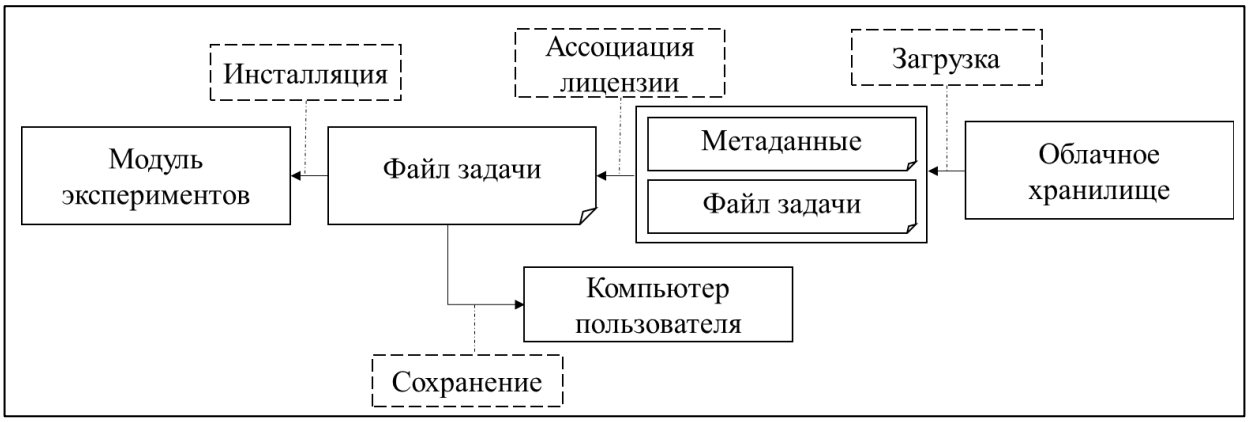

Рис. 5.2. Загрузки задачи из магазина

- Загрузка – отображение задачи в меню приложения.
- Ассоциация лицензии – закрепление загружаемого контента за конкретным пользователем для исключения передачи загруженного файла третьим лицам.
- Инсталляция – распаковка и загрузка задачи в приложение.

5.4. Структура пользовательских данных в облачном хранилище

Основной информацией, которая передаётся между пользователем и сервером является данные игрока и его успехи решения задач. Для хранения такой информации была создана таблица scores, которая размещена в облачном хранилище, и имеет следующие поля:

– ID – (строка) идентификатор пользователя в базе игроков (является уникальным и автоматически генерируется серверной частью при регистрации в базе);

– NickName – (строка) кличка игрока, выбирается самим игроком при регистрации в базе игроков и является уникальной;

– Name – (строка) имя пользователя (не является обязательным для заполнения);

– Surname – (строка) фамилия пользователя (не является обязательным для заполнения);

– DOPInfarmation – (строка) дополнительная информация по игроку, которая может включать класс или группу пользователя (не является обязательным для заполнения);

– Score – (число) условные очки, начисляемые при прохождения задач;

– LevelList – (строка) названия уровней и время прохождения соответствующих. Строка представления включает в себя сокращённое обозначение уровня и время в секундах, заключённое в квадратные скобки. Между собой записи разделяются символом «~».

Следует отметить, что для обозрения результатов прохождения другими игроками используется только NickName и Score, что позволяет упростить восприятие достижения других игроков и сохранить относительную анонимность, так как данные также отображаются на официальном сайте приложения. В близком кругу игроки будут узнавать друг друга, но постороннему пользователю будут неизвестны настоящие имена игроков [25]. Также наличие двух уникальных полей Nickname и ID, которые генерируются отдельно, повышает надёжность системы от попытки подмены подлинного игрока или от несанкционированного доступа извне приложения. Пример нескольких записей об игроках в таблице scores представлен на рисунке 5.3.

id	nickname	name	surname	doinfo	score	LevelList
id_12_57781e84dd8b6	Gopata	Азат	Акулин	ЮУрГУ ФТТ-490	1500	onL[113]~twl[50]~thl[300]~fol[-]
id_10_5727459ec114a	Man	Артём	Счатликов	ЮУрГУ ФТТ-490	1350	onL[126]~twl[59]~thl[340]~fol[-]
id_2_56f8144d0730f	VED	Ольга	Рыбкина	УГАТУ ФТП-120	980	onL[120]~twl[40]~thl[-]
id_13_5778f3436d05f	One	Тамара	Прегожина	ЮУрГУ ФТТ-260	850	onL[140]~twl[34]~thl[-]
id_1_56f8130a98577	Jon	Игорь	Воронов	МГУ ФКО - 102	500	onL[140]~twl[-]
id_0_56f8107aa38bd	Djocker	Роман	Годников	ЮУрГУ ФТТ-490	200	onL[164]~twl[-]
id_9_56fbc9980aef1	Pol	Азат	Акулин	ЮУрГУ ФТТ-490	120	onL[-]

Рис. 5.3. Таблица scores

Для хранения пользовательских задач в облачном хранилище предусмотрена папка RoomPacks, которая содержит подпапки Preview и MainData. Папка Preview содержит файлы с метаданными о задачах, которые соответствуют файлам формата «.rpp» на локальном устройстве и загружаются в модуле экспериментов для предоставлении общей информации о задаче. Папка MainData хранит связанные с мета-данными основное содержимое задач которое соответствует файлам форма-

та «.rpd» на локальном устройстве. Данные о связке задач с конкретным пользователем хранятся как в самих файлах, так и в таблице RoomPack, которая размещена в облачном хранилище, и имеет следующие поля:

– ID_ROOM – (строка) идентификатор задачи в базе (является уникальным и автоматически генерируется серверной частью при публикации новой задачи);
– URL_VIEW – (строка) путь к файлу с мета-данными задачи;
– URL_MAIN– (строка) путь к защищённому файлу с основным содеримым;
– ID_AUTOR – (строка) идентификатор пользователя, который является автором задачи;
– AREA – (строка) область, которую рассматривает данная задача;
– ID_ SUBSCRIBERS – (строка) включает в себя сокращённое представление идентификаторов пользователей установивших себе данную задачу. Между собой записи разделяются символом «~».

Пример нескольких записей о задачах в таблице RoomPacks представлен на рисунке 4.4.

ID_ROOM	URL_VIEW	URL_MAIN	ID_AUTOR	Area
room_1_5aeb67db7f156	http://physilabs.hol.es/RoomPack/Preview/room_	http://physilabs.hol.es/RoomPack/Base/room_1_	id_10_5a92e9936d3eb	3
room_2_5b084b565b609	http://physilabs.hol.es/RoomPack/Preview/room_	http://physilabs.hol.es/RoomPack/Base/room_2_	id_11_5aeb7100d6482	3
room_0_5aeb64321e631	http://physilabs.hol.es/RoomPack/Preview/room_	http://physilabs.hol.es/RoomPack/Base/room_0_	id_10_5a92e9936d3eb	3

Рис. 5.4. Таблица RoomPacks

5.5. Выделение объектных конструкций

Особенностью Unity является наследование классов, отвечающих за внутреннее игровое поведение от «MonoBehaviour», который является базовым классом и является связующим звеном между игровым движком и собственными классами. Каждый подобный класс сохраняется в отдельном файле, формируя скрипты, которые закрепляются за игровым объектом и описывают его поведение в игровом пространстве [26].

Следует понимать, что у Unity нет единой точки входа для выполнения кода, фактически это набор сценариев с привязкой к игровому пространству, выполнение которых зависит от происхождения какого-либо системного или пользовательского события. Определим стандартные для Unity методы, которые могут быть переопределены в скриптах.

Первая загрузка сцены:

– Awake() – этот метод всегда вызывается до начала любых функций, а также сразу после инициализации нового объекта в игровом пространстве;
– OnEnable() – (вызывается, если объект является активным): Эта функция вызывается только после того, как объект будет включен.

До первого обновления кадра:

– Start() – вызывается перед прорисовкой первого кадра, только если сценарий определён.

В промежутке между кадрами:

– OnApplicationPause() – это событие вызывается в конце кадра, когда обнаружена пауза, фактически между обычными обновлениями кадров. После OnApplicationPause() прорисовывается один дополнительный кадр для того, чтобы показать окно, которое отображается во время паузы.

Порядок обновления:

– FixedUpdate() – не зависит от Update(), и вызывается внутренним таймером;

– Update() – вызывается один раз за кадр;

– LateUpdate() – вызывается один раз в кадре, после завершения Update(). Любые расчеты, которые осуществляются в Update() будет завершены, при вызове LateUpdate().

Механические операции (триггеры):

– OnTriggerEnter(Collider collider) – вызывается при входе игрового объекта в зону триггера. В качестве аргумента принимает описание объекта, вошедшего в указанную зону;

– OnTriggerExit(Collider collider) – вызывается при выходе игрового объекта из зоны триггера;

– OnTriggerStay(Collider collider) – вызывается постоянно, пока игровой объект находится внутри зоны триггера.

Механические операции (коллайдеры):

– OnCollisionEnter()– вызывается каждый раз при контакте одного физического тела с другим физическим телом;

– OnCollisionExit()– вызывается каждый раз при прекращении контакта одного физического тела с другим физическим телом.

Стоит отметить, что Unity помимо множества предустановленных компонентов, например, визуальных эффектов постобработки, позволяет пользователю встраивать в систему собственные или сторонние модули [27]. Общая структура компонентов изображена на рисунке 5.6.

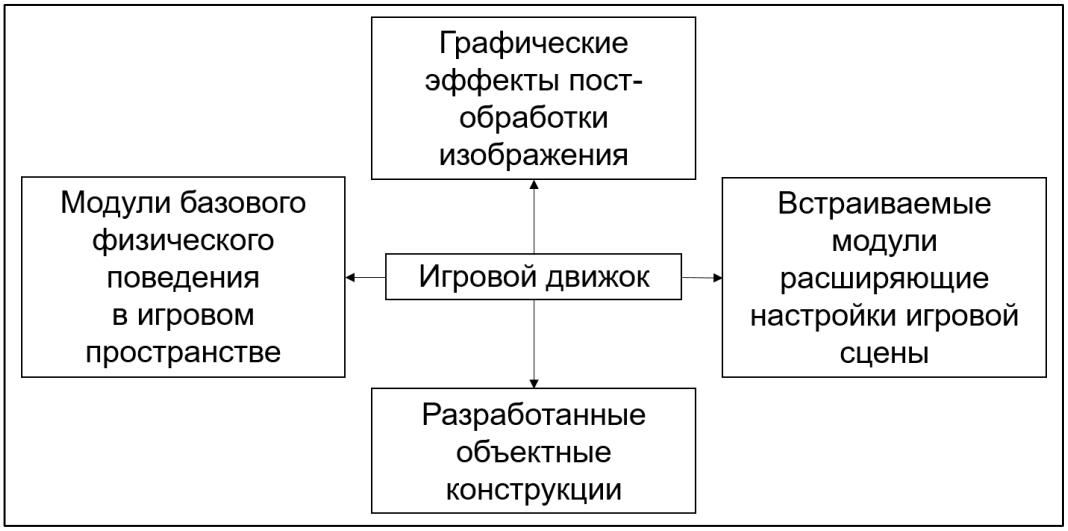

Рис. 5.6. Компоненты системы

Структуру разработанных классов в системе можно разделить следующим образом:

– интерфейс игрока;
– элементы управления;
– активные объекты;
– неактивные объекты;
– системные обработчики.

5.6. Модули взаимодействия с облачных хранилищем

Разработка сайта велась в визуальном конструкторе Zyro с использованием шаблона разметки. Конструктор позволяет создавать и настраивать основное графические элементы без использования HTML разметки. Также есть возможность дополнять содержимое страниц собственными фрагментами разметки или скриптами, которые помещаются в специальные блоки. Такие блоки корректно встраиваются в общую разметку и отображаются в браузере пользователя [28].

Концепция графических элементов представляет собой упрощённое представление дизайна самого приложения. Упрощение сделано для того, чтобы уменьшить размер страниц и ускорить загрузку в браузере. Также перед использованием все изображения были переведены в формат jpg, что позволило сократить объём данных хранимых в памяти сервера.

Доступ к данным в облачном хранилище осуществляется через PHP скрипты, которые выполняют SQL запросы. Скрипты хранятся в папке public_html на сервере и имеют следующий состав:

– Regestration.php – выполняет регистрацию пользователя в базе данных;

– display.php – выводит список лучших 20-ти игроков для отображения в приложении;

– WWWdisplay.php – формирует таблицу в виде html-разметки для отображения на сайте 20-ти лучших игроков;

– addscore.php – добавляет указанное количество очков к общей сумме за достижения;

– setdataprof.php – передаёт основные данные игрока при обновлении, так как полное заполнение при регистрации не обязательно;

– getDataLevel.php – получает информацию с метаданными о доступных задачах.

– roomPublicationINC.php – инициализирует загрузку новой задачи, генерирует ключи для шифрования, выделяет место в облачном хранилище;

– roomCollectionGet.php – получает список с общими данными всех доступных для загрузке задач. Опционально может задаваться ограничение на количество возвращаемых задач;

– FileUpwnload.php – загружает содержимое задачи, связывая мета-данные с получателем, выделяет ключ для расшифровки.

Ниже фрагмент скрипта, в котором инициализируется загрузку новой задачи:

```
<?php
	$db = mysqli_connect('mysql.hostinger.ru') or die(mysqli_error($db));
	mysqli_select_db($db, 'u295677728_phla') or die(mysqli_error($db));

	$id_user = $_GET['id'];
	$nickname = $_GET['nickname'];
	$area = $_GET['area'];
	$hash = $_GET['hash'];

	$secretKey="PhysiLabsAPP";
	$real_hash = md5($id_user . $nickname . $secretKey);

	if($real_hash == $hash)
	{
		$result = mysqli_query($db, $query) or die(mysqli_error($db));
		$nums = mysqli_num_rows($result);

		if($nums == 0)
		{
			echo "No autentificate";
		}
		else
		{
		    try
		    {
			$result = mysqli_query($db, $query) or die(mysqli_error($db));
			$row = mysqli_fetch_row($result);
			$prefix = "room_";

			if(count($row) > 0)
				$prefix = $prefix . $row[0] . "_";
			else
				$prefix = $prefix . "0_";

			$id_room = uniqid($prefix);
			$conn_id = ftp_connect('ftp.physilabs.hol.es');
			$filePath = " /RoomPack/Base/" . $id_room . ".rpd";
			$filePreviewPath = "/RoomPack/Preview/" . $id_room . ".rpp";

			mysqli_query($db, $query) or die(mysqli_error($db));

			echo $id_room;
		    }
		    catch (Exception $e)
```

```
                {
                    echo "PublicationError";
                }
            }
        }
?>
```

5.7. Разработка интерфейса пользователя

Концепция дизайна элементов интерфейса заключается в лёгкости и информативности, сочетая современный минимализм и идеи скевоморфизма. Элементы дизайна в данном подходе скопированы с формы другого реального объекта, но изготовлены из других материалов или имеют другую природу. Также объекты интерфейса могут встраиваться в игровое пространство, что добавляет новое измерение в использовании привычных графических элементов [29]. Основным шрифтом приложения является Lucida Console, так как он подходит по сюжету игры, которая насыщенна роботами.

Общая концепция оформления кнопок преследуется во всех однородных элементах. Кнопки выполнены в форме прямоугольника со скруглёнными вертикальными сторонами, радиус которых образует полукруг, как изображено на рисунке 5.7. В левой части находится пиктограмма на цветном фоне с эффектом тени и двухконтурной окантовкой, посередине располагается текст. Пиктограмма может располагаться в пределах самой кнопки или располагаться отдельно.

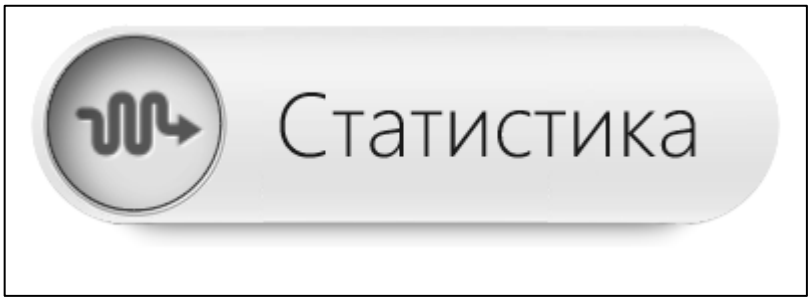

Рисунок 5.7. Оформление кнопки

Элемент пиктограммы, в сочетании с настройкой цвета подложки, позволяет улучшить визуально восприятие кнопки. Использование кнопок в игровом пространстве как, например, в основном меню, позволяет разделить пиктограмму на слои, и создать эффект глубины при перемещении кнопки относительно камеры.

В некоторых элементах интерфейса используется набор связанных кнопок, имитирующих тумблеры, при активации одной из кнопок, другие деактивируются. Такой элемент может использоваться для имитации переключения между вкладками или выбора одной из предлагаемых опций, количество которых не ограничено. На рисунке 5.8 видно, как активна кнопка «Опция 3». Концепция оформления идентична предыдущей, за исключением отсутствия пиктограмм и

наличием разделительной вертикальной линии. Цвет активной кнопки настраивается индивидуально.

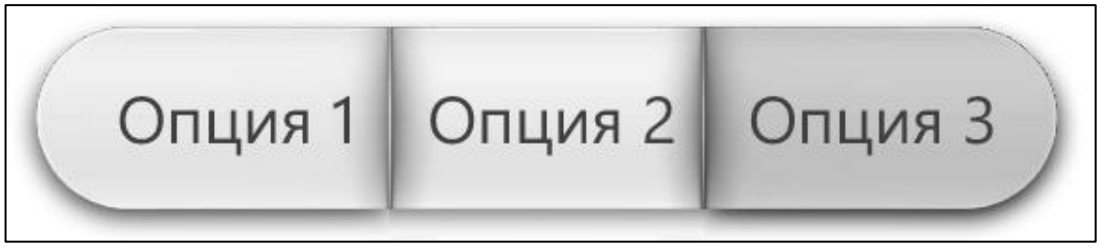

Рисунок 5.8. Оформление набора связанных кнопок

5.8. Тестирование системы

На этапе альфа-тестирования проверялись различные возможности совершения системой ошибок путём различных неадекватных действий в игре. Обнаруженные проблемы были исправлены, и игра приобрела товарный вид.

На этапе бета-тестирования определённой группе лиц была выдана первая бета-версия приложения для ознакомления и запуска на различных персональных компьютерах. Выполнялась проверка производительности системы на экранах с различными разрешениями и компьютерах с различными вычислительными мощностями. Главным критерием тестирования была выбрана средняя частота кадров в секунду, так как этот показатель влияет на комфорт игрового процесса [30]. Результаты тестирования приведены в таблице 4.

Таблица 4. Результаты тестирования

Процессор	Видеоадаптер	Оперативная память	Разрешение экрана	Средняя частота кадров
Intel Core i3 (4 ядра)	NVidia GEFORCE 840m	4 Гб	800x600	160
			1366x768	100
AMD Sempron 2800 (2 ядра)	ATI Radeon HD 3850 512 Mb	2 Гб	800x600	105
			1280x1024	72
AMD Sempron 2800 (2 ядра)	ATI Radeon HD 3850 512 Mb	1 Гб	800x600	107
			1280x1024	71
AMD Turion RM-70 (2 ядра)	ATI Radeon HD 3470 - x2 1 Gb	3 Гб	800x600	78
			1280x800	65
AMD Turion RM-70 (2 ядра)	ATI Radeon HD 3470 - x2 1 Gb	1 Гб	800x600	73
			1280x800	62

Intel Pentium inside	Встроенный адаптер Intel 450	2 Гб	800x600	35
			1366x768	18-25

По результатам тестирования видно, что для системы имеет большое значение наличие дискретного видеоадаптера, но для современных персональных компьютеров это не является проблемой. Производительность системы не зависит от объёма оперативной памяти, так как все сцены имеют небольшой размер.

Стоит отметить, что приложение может работать на маломощных персональных компьютерах, ввиду внутренней оптимизации и графических настроек, что соответствует предъявленным минимальным системным требованиям.

5.9. Выводы по разделу

В данной главе было реализовано приложение, выделена структура пользовательских данных, определена структура файлов задач. Были выделены объектные конструкции, разработаны модули взаимодействия с облачных хранилищем, реализована криптографическая модель. Приложение прошло альфа- и бета-тестирование, а демо-версия опубликована на официальном сайте www.physilabs.hol.es в разделе «Загрузки».

ЗАКЛЮЧЕНИЕ

В данной выпускной квалификационной работе было разработано интерактивного приложения, которое позволяет в игровой форме изучить и закрепить теоретические знания из различных областей естественных наук.

Изучена предметная область и проанализированы существующие программные продукты. Определён игровой формат, выделены основные игровые объекты и методы их взаимодействия. Разработана архитектура системы, выбраны инструменты разработки, с использованием которых реализована описанная система. Для увеличения интереса к изучению естественных наук приложение представляет собой интерактивную игру с сюжетной историей, конструктором задач и комнатами экспериментов.

Система предоставляет пользователям возможность увидеть принципы действия и последствия тех или иных явлений и законов физики, механики, робототехники и т.п. приложение объединяет в себе как теоретическою информацию, так и соответствующие интерактивные визуальные элементы. Система включает в себя модули клиентского приложения и модулей облачного хранилища.

В рамках данной работы, также были рассмотрены основные угрозы для приложение разработанных на Unity. Защита файлов и данных от фальсификации и передачи третьем лицам является актуальной проблемой на всех платформах. Особенно это актуально для приложений, разработанных на Unity, так как этот продукт не предусматривает встроенных систем защиты ресурсов и данных, складывая все полномочия на разработчика. Однако, модульность и кроссплатформенность данного инструмента позволяет разрабатывать универсальные модули, которые можно будет внедрять и в другие проекты.

Универсальность разработанной модели позволит другим разработчикам интегрировать и использовать её в других проектах на Unity. Также модель поддерживает концепцию разделяемого контента, когда доступ к определённым ресурсам возможен только по заданному условию.

Для защиты приложения была разработана и реализованная криптографическая модель, ориентированная на использование с Unity. Была разработана структура файлов задач и модули конструктора задач и экспериментов, адаптированных на разработанную модель. На основе модели был реализован модуль защиты файлов, в который входит компоненты, используемые в скомпилированном проекте и в редакторе Unity. Проведено альфа- и бета-тестирование системы группой лиц, в которую входили: 6 студентов, преподаватель по программированию и преподаватель по физике. После тестирования была опубликована первая демоверсия игры.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Соколова, Е.В. Мультимедиа-технологии в преподавании информатики / Е.В.Соколова // «Информационные технологии в образовании» – 2003: сб.науч.ст. – Челябинск: Изд-во ЮУрГУ, 2003. – с. 80-81.
2 Латыпов, Р. Интернет: к новому типу образования / Р. Латыпов // Высшее образование в России. – 1997. – № 3. – С. 98–103.
3 Технические средства обучения в современной школе: пособие для учителей и директора школы/ А.А. Журин, Е.А. Бондаренко, И.А Милютина; под редакцией А.А.Журина. – М: Юнвес, 2004. – 403с.
4 Понятие компьютерной игры [Электронный ресурс]. – Режим доступа: https://goo.gl/bDlqGi, свободный – Дата обращения: 04.09.2016
5 О компьютерных играх. Во что играют россияне [Электронный ресурс]. – Режим доступа: http://fom.ru/Kultura-i-dosug/10991, свободный – Дата обращения: 12.10.2016
6 Компьютерное приложение «Stephen Hawking's Snapshots of the Universe» [Электронный ресурс]. – Режим доступа: https://itunes.apple.com/ru/app/stephen-hawkings-snapshots/id714306520?mt=8, свободный – Дата обращения: 06.02.2017
7 Компьютерное приложение «Particulars» [Электронный ресурс]. – Режим доступа: http://particularsgame.com, свободный – Дата обращения: 15.02.2017
8 Компьютерное приложение «The Powder Toy» [Электронный ресурс]. – Режим доступа: http://powdertoy.co.uk, свободный – Дата обращения: 17.03.2017
9 Компьютерное приложение «A Slower Speed of Light» [Электронный ресурс]. – Режим доступа: http://gamelab.mit.edu/games/a-slower-speed-of-light, свободный – Дата обращения: 23.03.2017
10 Компьютерное приложение «ALGODOO» [Электронный ресурс]. – Режим доступа: http://www.algodoo.com, свободный – Дата обращения: 03.04.2017
11 Geig M. A.И. Unity Game Development / Mike Geig. // Sams Teach Yourself. – 2015. – pp. 186–201.
12 Lander, J. A.И. Game Engine Unity/ Jeff Lander // A K Peters. – 2009. – 864 p.
13 Hocking, J. Unity in Action: Multiplatform Game Development in C# / Joe Hocking. // Basic Books. – 2013. – pp. 125–142.
14 Статистика популярности игровых движков. – http://www.slideshare.net/markdeloura/game- engines-and-middleware-2011
15 Goldstone, W. Unity Game Development Essentials / Will Goldstone. // Packt Publishing. – 2010. – pp. 398–436.
16 Zucconi. A. Unity 5.x Shaders and Effects / Alan Zucconi. // Feral House. – 2016. – pp. 181–212.
17 David, R. Classical Cryptography / Richard David. // Creative Commons. – 2008. – pp. 108–117.

18 Ferguson, N. Cryptography Engineering: Design Principles and Practical Applications / Niels Ferguson. // Tor Books. – 2007. – pp. 126–140.
19 Lindell, Y. Introduction to Modern Cryptography: Principles and Protocols / Yehuda Lindell. // HarperCollins. – 2012. – pp. 27–52.
20 Gregory, J. Game Engine Architecture / Jason Gregory. // Gannett Company. – 2009. – pp. 687–717.
21 Sapio, F. Unity UI Cookbook / Francesco Sapio. // Tokyopop. – 2015. – 128 p.
22 Lintrami, T. Unity 2017 Game Development Essentials / Tommaso Lintrami // Packt Publishing. – 2018. – pp. 253–282.
23 Stallings, W. Cryptography and Network Security Principles and Practices / William Stallings. // Prentice Hall. – 2005. – pp. 345–380.
24 Wiley, J. App Cryptography / John Wiley. // Creative Commons. – 1996. – 784 p.
25 Gibson, J. Introduction to Game Design, Prototyping, and Development / Jeremy Gibson. // Orbit Books. – 2014. – pp. 125–148.
26 Thorn, A. Unity 5.x by Example / Alan Thorn. // Berlitz. – 2016. – 273 p.
27 Роллингз, Э. Проектирование и архитектура игр / Эндрю Роллингз, Дайв Моррисс; пер. с англ. А. Чекатков. – М.: Вильямс, 2005. – pp. 522–558.
28 Wenbo, M. Modern Cryptography: Theory and Practice / Yehuda Lindell. // Prentice Hall PTR. – 2003. – pp. 85–112. p.
29 Barrera, R. Unity UI Game Programming / Ray Barrera. // Apress. – 2015. – pp. 32–61
30 Akenine-Moller, T. Real-Time Rendering, Third Edition / Tomas Akenine-Moller. // A K Peters/CRC Press. – 2008. – pp. 845–952.

ПРИЛОЖЕНИЕ 1

РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

П1.1. Установка и настройка приложения

Чтобы установить приложение, необходимо предварительно скачать его с официального сайта www.physilabs.hol.es на странице «Загрузки». Скаченный архив необходимо распаковать в любую удобную папку.

Для запуска приложение требуется выполнить двойной щелчок левой кнопки мыши по файлу PhysiLabs.exe.

При запуске приложения откроется окно с графическими настройками и пояснением действий клавиш управления. При первом запуске рекомендуется выбрать минимальные настройки, чтобы проверить быстродействие системы. В последующем эти установки можно повысить в соответствии с вычислительными возможностями персонального компьютера.

П1.2. Краткие обозначения

В дальнейшем описании игровых сцен будут использоваться следующие цветовые и символьные обозначения.

Цветовые обозначения:

- темно-серый – недосягаемая область (стены помещения);
- белый – свободная область для перемещения;
- серый (наклонная штриховка) – видимая, но недосягаемая область (помещения за стеклом или дверью);
- светло-серый – небольшая возвышенность (ступень);
- серый – возвышенность, взобраться на которую невозможно, или возможно только прыжком.

Символьные обозначения:

- 1, 2… – этапы прохождения по игровой сцене;
- I – элемент с информационной «карточкой»;
- D – автоматическая дверь (открывается при приближении игрока);
- Do – управляемая дверь (для открытия или закрытия требуется активация привязанного устройства);
- Dr – закрытая дверь;
- C – камера наблюдения с отмеченной зоной обзора;
- Pm – мастер-отладчик;
- Pt – персонал лаборатории (работает на компьютере за столом);
- Ps – персонал лаборатории (работает на компьютере за стойкой терминала);
- Pr – персонал лаборатории (отдыхает на стуле, читает теорию «Большого взрыва» на планшете);

– Go – сферический робот с «зелёной» индикацией и отмеченным маршрутом перемещения;
– Go – сферический робот с жёлтой индикацией;
– Gr – сферический робот с красной индикацией;
– Ui – область с доступом к интерфейсу управления оборудованием;
– Ut – область, помещение в которую игрока или иного отмеченного объекта, вызывает активацию «привязанного» устройства.

П1.3. Локация «Первое включение»

История персонажа начинается на первой локации, схема которой изображена на рисунке П.1.1 (вид сверху). На данной локации происходит первое включение робота PHY для проведения теста его новой модификации на различных задачах. Проверкой основных систем занимается мастер-отладчик в специальной комнате (позиция 1 на рис. П.1.1).

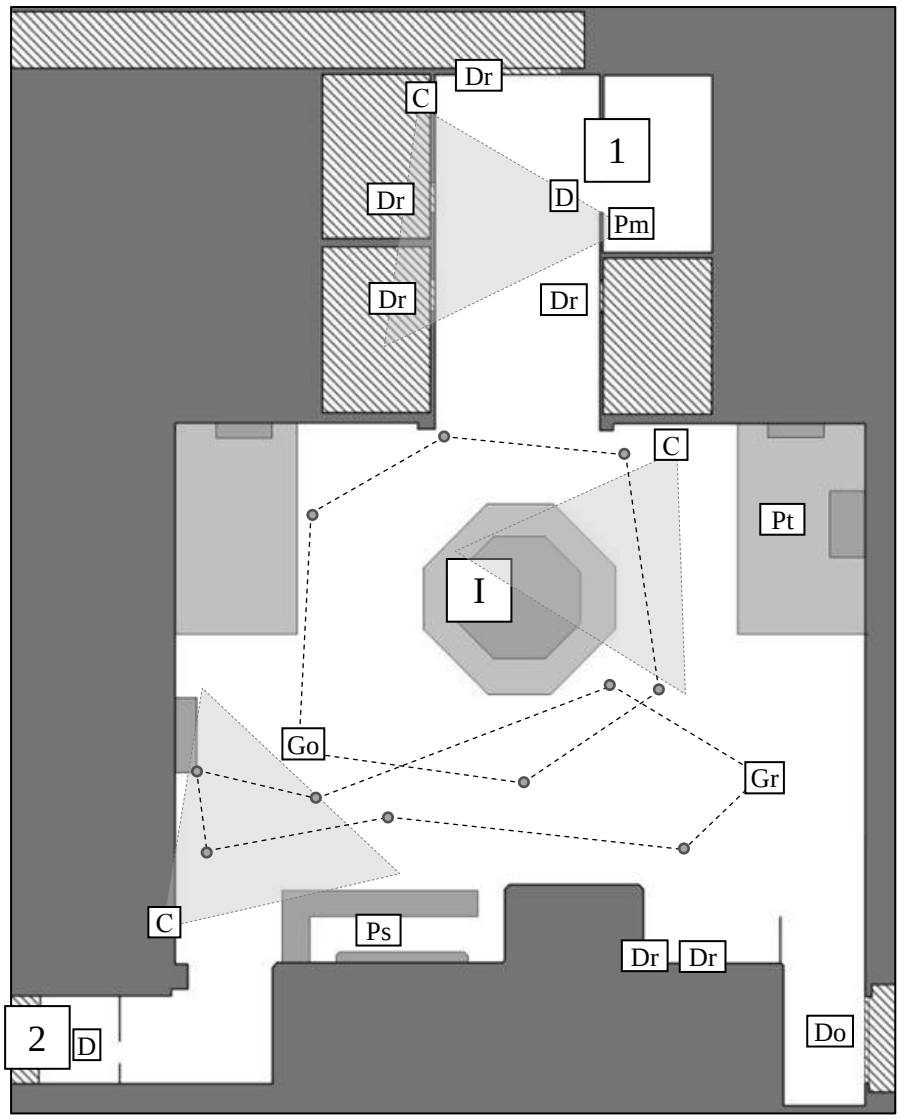

Рис. П.1.1. Схема первой локации

Первая кат сцена сопровождается монологом мастера со следующим текстом:

«Так…. Первое включение… Ум... Речевой синтезатор выдал ошибку, и батарея старая… Значит будешь у нас не разговорчивым… Фи, ты меня слышишь? Помаши головой…»

После того как игрок воспользуется мышью, чтобы осмотреться, мастер- отладчик продолжит монолог:

«Ага… Отлично… Мониторинг завершён, я тебя отпускаю, пока можешь осваиваться. Сейчас тебе необходимо проследовать в соседний отсек, там ты получишь новую батарею, а после этого мы перейдём к первому обучению. Ступай, я буду связываться с тобой по беспроводной сети.»

После проведения всех необходимых работ мастер отключает PHY от диагностического устройства и управление роботом передаётся игроку. На первом уровне игрок знакомится с элементами управления через интерактивные подсказки и осматривает окружающую обстановку. Нашему главному герою требуется получить новый модуль питания, так как изначально установленный не развивает требуемой мощности, что не позволяет PHY прыгать и быстро передвигаться. Модуль питания он сможет получить в соседнем отсеке, перейти в который можно через стыковочный узел (позиция 2 на рис. П.1.1).

Пока игрок продвигается по локации, он может познакомиться с другими разработками GlobeLabs (элементы G1, G2 и C на рис. П.1.1) и даже попробовать к ним подключиться.

В центре игровой сцены находится постамент (элемент I на рис. П.1.1), изображенный на рисунке П.1.2, с движущейся моделью «Земля-Луна», сведения о которой игрок может получить, используя информационную «карточку».

Рис. П.1.2. Постамент «Модель Земля-Луна»

После того как игрок достигнет конечного пункта (позиция 2 на рис. П.1.2), начнётся загрузка следующей локации, в это время на экране будет отображаться заставка с прогрессом загрузки.

П1.4. Локация «За запчастями»

На второй локации, изображённой на рисунке П.1.3, игрок начинает свой путь с позиции (позиция 1 на рис. П.1.3), где ему предстоит забрать модуль питания со стеллажа (позиция 2 на рис. П.1.3).

Когда игрок установит новый модуль питания, он услышит сообщение от мастера со следующим текстом:

«Всё, готово. Приступим к испытаниям, возвращайся в предыдущий отсек и оттуда переходи по туннелю в соседнее крыло. Там встретимся.»

После установки нового модуля робот сможет прыгать и перемещаться «бегом», о чём будет проинформировано специальными подсказками. Игроку необходимо вернуться на предыдущую локацию через стыковочный узел (позиция 1 на рис. П.1.3).

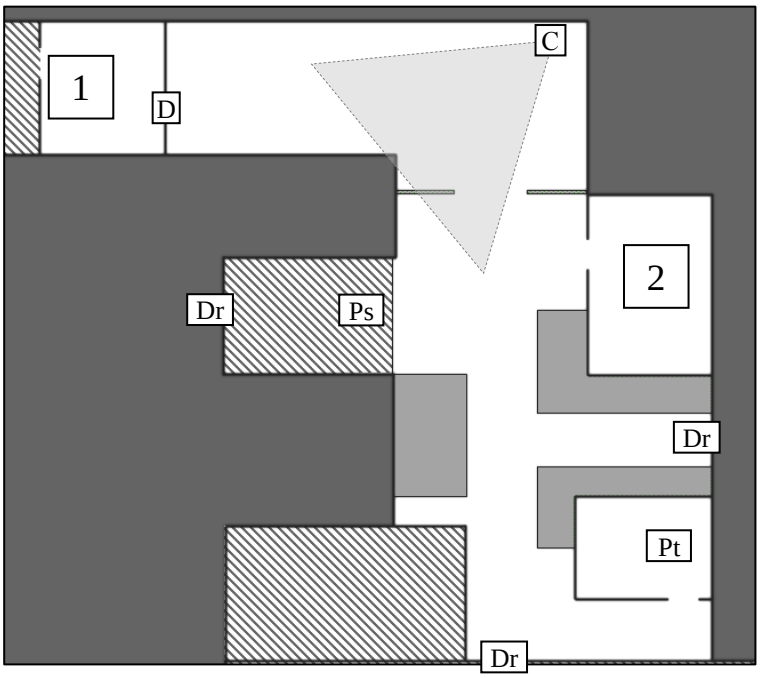

Рис. П.1.3. Схема второй локации

П1.5. Локация «Через галерею»

После возвращения на предыдущую локацию, дополнительная часть которой изображена на рисунке П.1.4, откроется дверь для перехода на нижний этаж через смотровую галерею (позиция 1 на рис. П.1.4), откуда игрок сможет увидеть Землю на горизонте Луны.

На нижнем этаже находится галерея космонавтики с портретами знаменитых людей (элементы I на рис. П.1.4), сделавших ключевой вклад в историю покорения космоса, а также различные космические аппараты. Закончив осмотр, игрок переходит на следующую локацию к своему первому испытанию через стыковочный узел (позиция 2 на рис. П.1.4).

При передвижении по игровым локациям в громкоговорители через разные промежутки времени будет воспроизводиться сообщение для персонала лабора-

торий об утечке хладагента, что обуславливает наличие масок у всех работников, текст сообщения, следующий:

«Просьба всех работников Лунного центра исследований надеть защитные маски, зарегистрирована утечка хладагента. Маски вы можете найти на своих рабочих местах.»

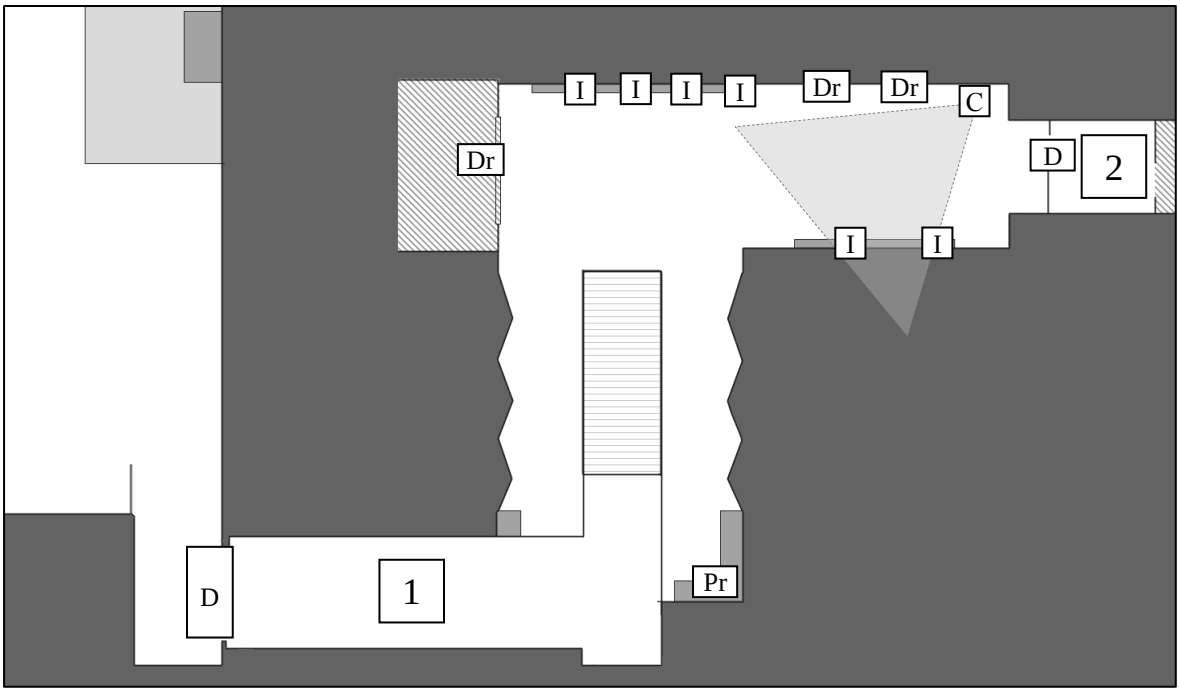

Рис. П.1.4. Схема третьей локации

П1.6. Локация «Притянутый за уши»

Перейдя в четвёртую локацию, схема которой изображена на рисунке П.1.5, игроку предстоит решить первую задачу, которая связана с электромагнетизмом.

В центре комнаты цилиндрической формы, изображённой на рисунке П.1.6, находится мощный электромагнит и прямо под ним колба со сферическим роботом внутри. Колба, выполненная из металла и стекла, может перемещаться только в вертикальном направлении под действием электромагнита до момента срабатывания упоров.

Игрок начинает прохождение локации из стыковочного узла (позиция 1 на рис. П.1.5). Задачей, которую требуется решить, является извлечение робота (элемент Go на рис. П.1.5) из колбы, чтобы установить его на активатор (элемент Ut на рис. П.1.5) для открытия двери (элемент Do на рис. П.1.5). Управление над установкой игрок может получить, встав в область с доступом к интерфейсу управления оборудованием (элемент Ui на рис. П.1.5). После того, как игрок настроит правильно входные параметры для работы электромагнита (так чтобы колба была извлечена, но при этом обмотка электромагнита не перегревалась), он сможет получить доступ к сферическому роботу (элемент Go на рис. П.1.5) и выполнить подключение по смежному интерфейсу.

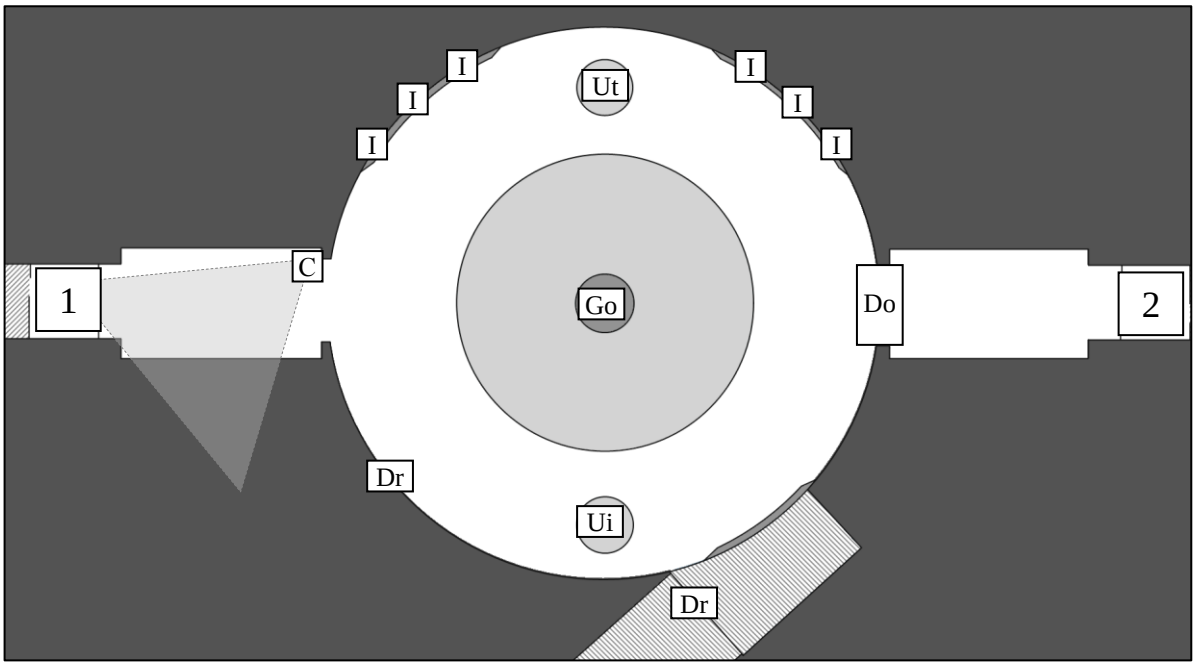

Рис. П.1.5. Схема четвёртой локации

После открытия двери (элемент Do на рис. П.1.5), игрок отключится от робота (элемент Go на рис. П.1.5) и проследует на следующее испытание через стыковочный узел (позиция 2 на рис. П.1.5). Также в комнате имеются портреты учёных (элемент I на рис. П.1.5), сделавших ключевые открытия в области электродинамики и электростатики.

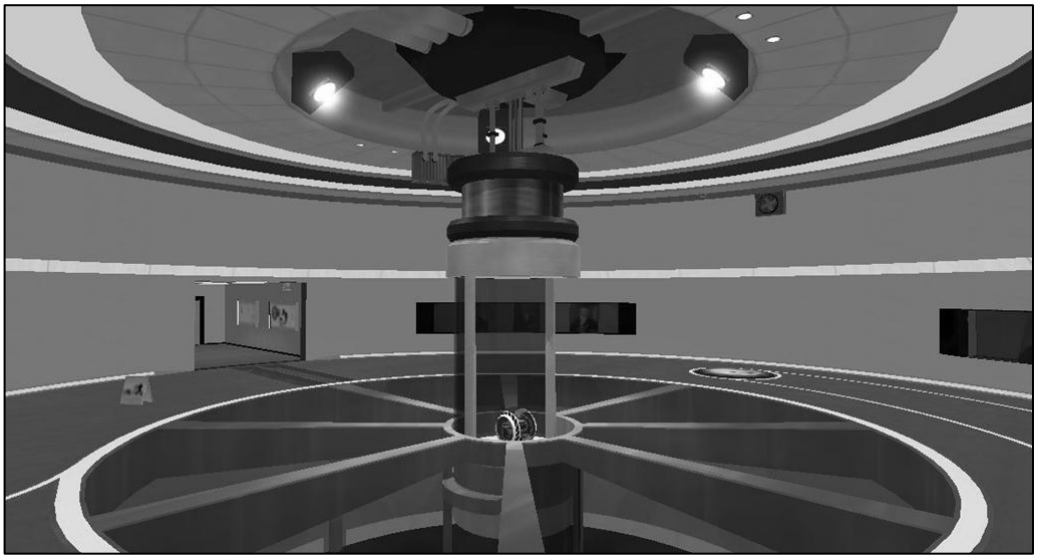

Рис. П.1.6. Электромагнит и колба с роботом внутри

Четвёртая локация на данный момент является последней, дальнейшие задания будут разрабатываться по мере развития проекта.

Как отмечалось ранее, все роботизированные объекты в игре могут обмениваться аудиосообщениями, которые разнятся в зависимости от источника, вызвавшего это событие.

Сообщения, когда сферический робот достигает определённой точки своего маршрута, имеют следующий текст:

- *Поехали!*
- *Я так обожаю приключения!*
- *Мы едем, едем, едем в далёкие края…*
- *Хьюстон, у нас проблемы…*
- *Едем, едем на дискотеку, в соседнее село…*
- *Ну, давай! Кто тут Чак Норрис?*
- *Сейчас я буду раздавать подарки, будет больно, но хватит всем…*

Сообщения при столкновении сферического робота с главным героем:

- *Эй! Полегче!*
- *Ну ладно, ты победил...*
- *Привет…*

Когда камера наблюдения замечает сферического робота, воспроизводится аудиодорожка:

- *Эй, шарообразный!*
- *Кто-то потерял шар для боулинга…*

П1.7. Управление персонажем

Управление персонажем осуществляется посредством клавиатуры для перемещения и прыжка, и мышью для обзора.

Стандартные назначения клавиш:

- W – игрок перемещается вперёд;
- S – игрок перемещается назад;
- A – игрок перемещается влево (для сферического робота поворот налево);
- D – игрок перемещается вправо (для сферического робота поворот направо);
- Пробел – прыжок;
- Shift – режим бега;
- M – масштаб мини-карты;
- Esc – поставить игру на паузу (в режиме игры).

Кнопкам мыши соответствуют следующие действия:

- левая кнопка мыши – прочитать информационную «карточку» при наведении на объект с данной опцией;
- правая кнопка мыши – подключиться или отключиться по совместимому интерфейсу.

П1.8. Основное меню

Основное меню выполнено в виде руки робота, где к каждому пальцу привязан определённый элемент пользовательского интерфейса, а именно кнопка. При наведении на одну из кнопок, её фон подсвечивается, а палец, находящийся под ней

поднимается вместе с кнопкой. Например, на рисунке П.1.7 приподнялся безымянный палец. При нажатии воспроизведётся анимация поворота камеры и отобразится выбранное подменю.

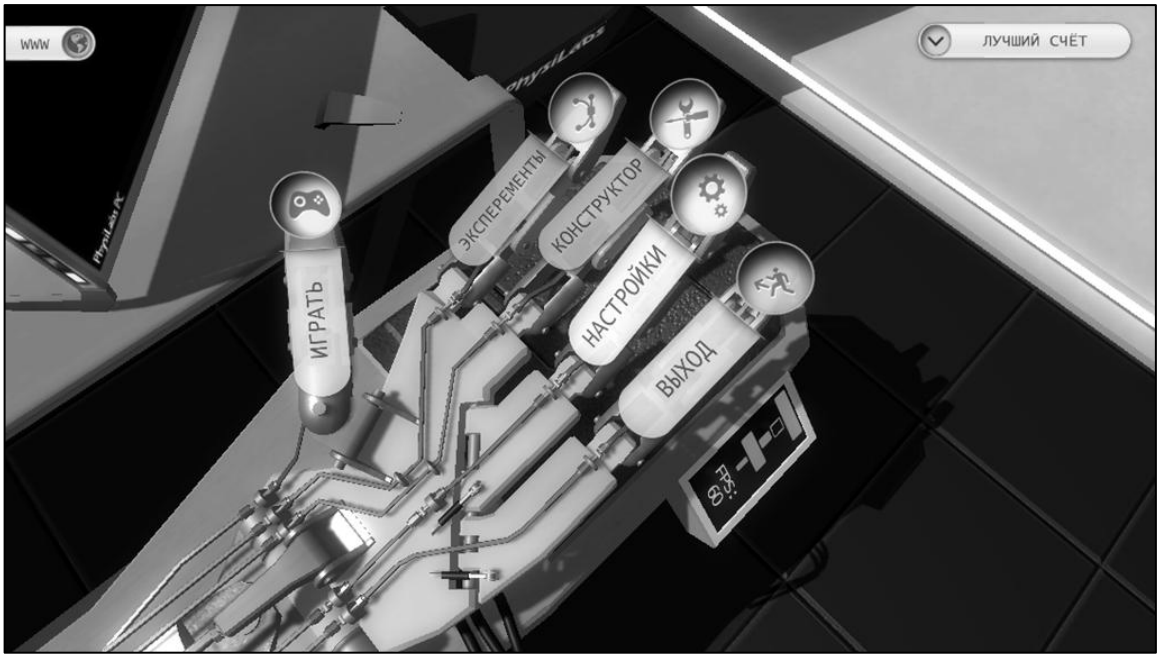

Рис. П.1.7. Оформление кнопки

Также на экране в верхней левой части располагается кнопка «WWW», по нажатию на которую, в браузере откроется главная страница сайта приложения www.physilabs.hol.es.

В правой части экрана имеется кнопка для раскрытия или сокрытия списка 20-ти лучших игроков с отображением никнеймов и суммы заработанных очков, как показано на рисунке П.1.8. Карточка пользователя подсвечивается оранжевым цветом.

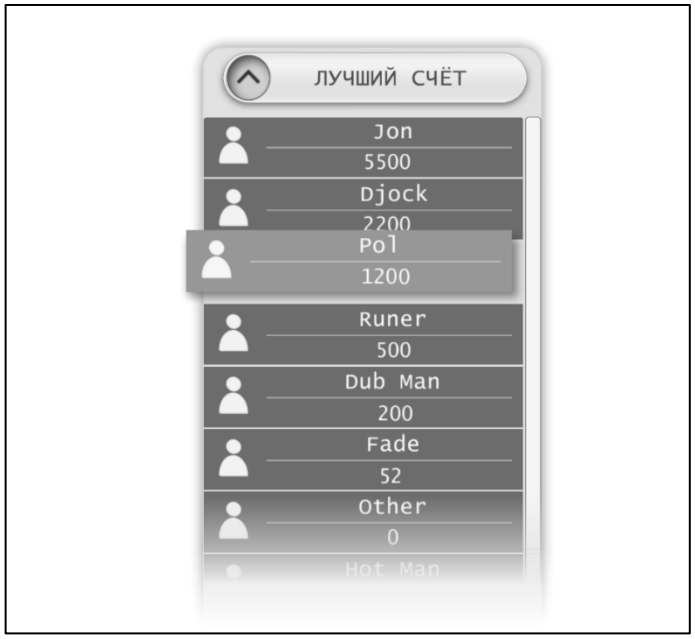

Рис. П.1.8. Список лучших игроков

П1.9. Подменю «Играть»

Любое подменю разделено на: заголовок, контент и функциональные кнопки. В верхней части располагается заголовок с пиктограммой и цветовой подложкой, которая может менять свой цвет при переходе в друге подменю. В центральной части экрана располагается содержимое, которое различно для каждого подменю. В нижней части находятся функциональные кнопки, которые отвечают за переход назад в меню или вызывают события, оговоренные содержимым.

Из подменю «Играть», изображённого на рисунке П.1.9, игрок может загрузить любой пройденный уровень кнопкой «Загрузить» или сразу перейти к прохождению последней достигнутой локации при помощи кнопки «Продолжить».

Содержимое подменю состоит из «карточек» уровней. «Карточки» содержат в левой части изображение локации, в центральной – информацию о дате и времени прохождения, итоговом времени прохождения и название самой локации. Если уровень не пройден до конца, то вместо итогового времени отображается соответствующее сообщение. В правой части имеется цветовая подсветка, которая показывает какой уровень для загрузки выбран в данный момент. Выбрав необходимый уровень, игрок может перейти к его повторному прохождению.

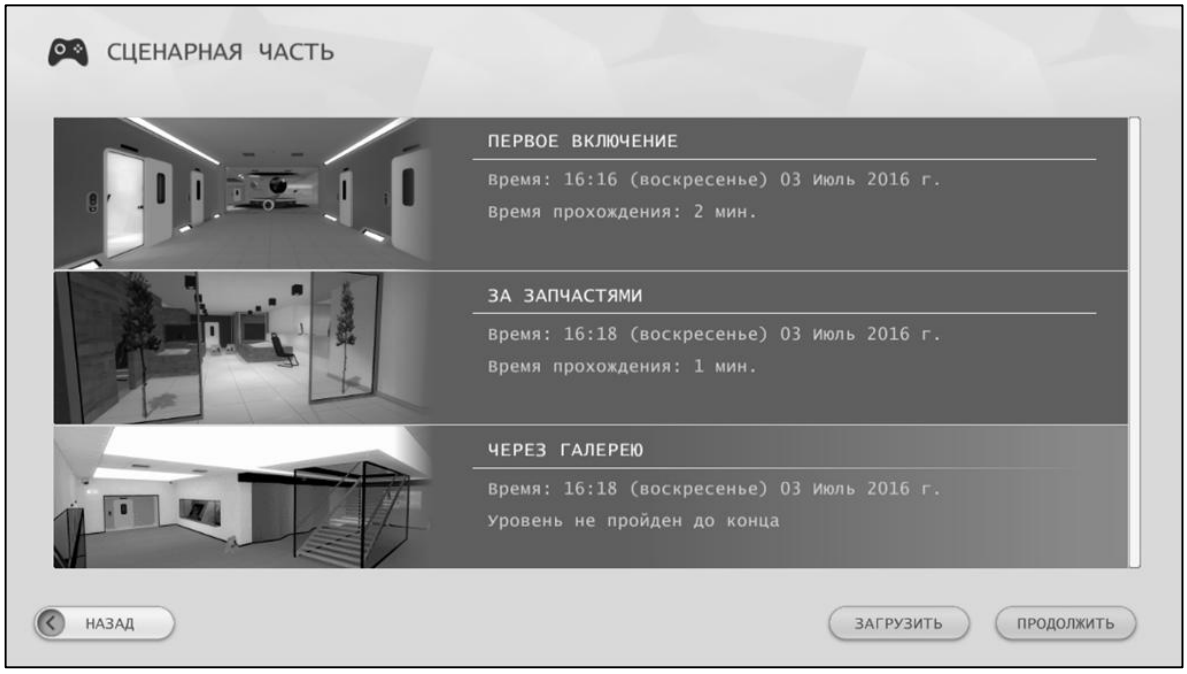

Рис. П.1.9. Подменю «Играть»

П1.10. Подменю «Настройки»

Данное подменю состоит из 4-х вкладок: «Профиль», «Видео», «Звук» и «Управление». Переключение по данным закладкам выполняется специальной разновидностью кнопок. Последние две вкладки находятся на стадии разработки.

На вкладке «Профиль», изображённой на рисунке П.1.10, пользователь может увидеть всю сохранённую информацию по своему профилю, а также изменить настройки или свои данные.

В левой части экрана отображается информация о проведённом времени в игре, дате последнего посещения приложения и фото пользователя.

Рис. П.1.10. Настройки профиля игрока

Настройки профиля условно разделены на настройки личной информации и настройки родительского контроля. В верхнем блоке пользователь может ввести личные данные и выбрать режим подключения к серверу для получения данных прохождения других игроков. При изменении данных, все изменения будут передаваться на сторону сервера, о чём сигнализирует информационное табло в нижней части экрана. Для обмена данными с серверной частью, требуется регистрация, которая заключается в выборе никнейма, как показано на рисунке П.1.11.

Ранее было отмечено, что при регистрации игроку необходимо указать уникальный «никнейм» и в случае совпадения система попросит ввести другой вариант (см. Рис. П.1.11).

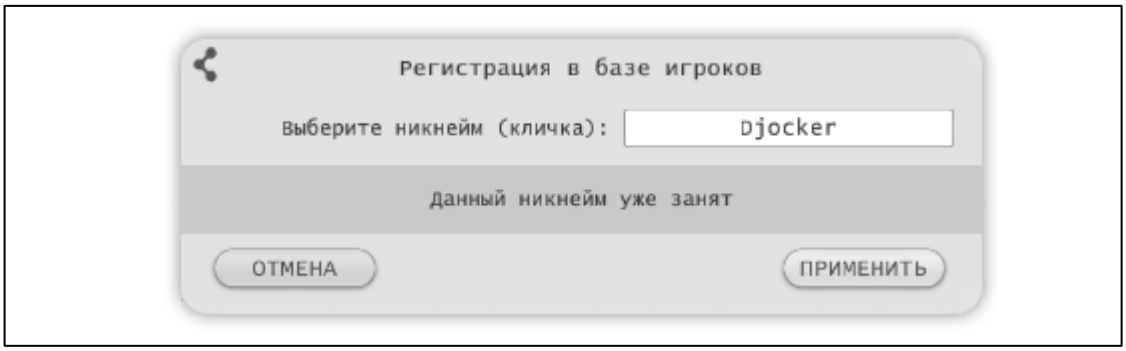

Рис. П.1.11. Сообщение о том, что «никнейм» уже используется

Если нет соединения с сетью, то выводится соответствующие сообщение пользователю, например, как показано на рисунке П.1.12.

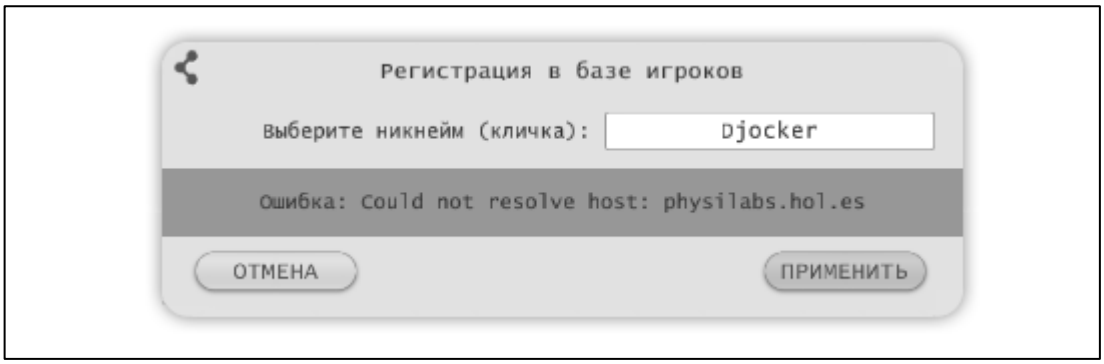

Рис. П.1.12. Сообщение о ошибке подключения

Подобные сообщения об ошибках имеются и в других диалоговых окнах приложения, также в зависимости от того, является прерывание операции ошибкой или логическим несоответствием, используются различные цветовые решения:

- красный – ошибка;
- желтый – предупреждение о несоответствии;
- зелёный – успешная операция.

При возникновении ошибки также выводится соответствующий текст, который позволяет легко отследить причину её возникновения.

В нижнем блоке настроек профиля игрока можно настроить максимальное время одного сеанса игры и установить пароль на изменение настроек, при помощи кнопки «Родительский контроль». Все слайдеры в пользовательском интерфейсе имеют фон, который изменяется в зависимости от заданного градиента и установленного на нём значения, что позволяет визуально подчеркнуть выбираемую величину.

При установке пароля требуется ввести дважды секретное «слово» в форму, изображённую на рисунке П.1.13, это требуется для избежание набора пароля случайным образом.

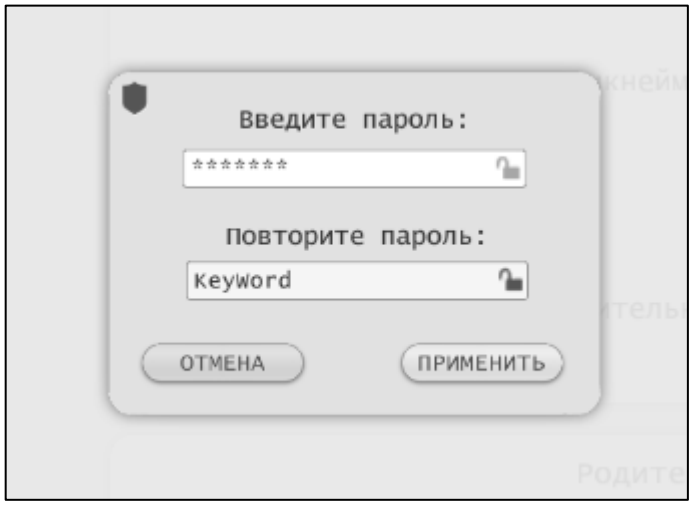

Рис. П.1.13. Форма установки пароля

Поле ввода пароля имеет кнопку для подглядывания, то есть временного отображения скрытого введённого пароля по нажатии на соответствующую пикто-

грамму, а также подсвечивание поля красным цветом при неправильном введённом пароле.

После установки пароля, оба блока настроек профиля заблокируются, как показано на рисунке П.1.14, а возможность выбирать подключение к серверу и отключение родительского контроля останутся активными.

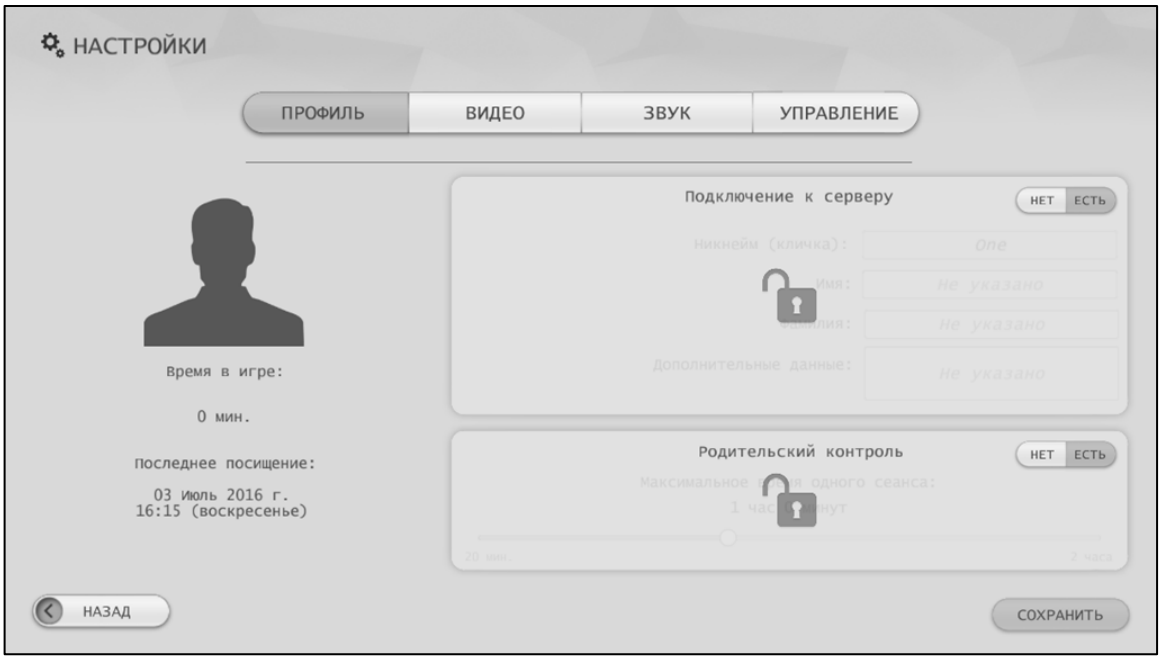

Рис. П.1.14. Настройки профиля игрока

Чтобы получить доступ к настройкам интересующего блока или отключить «Родительский контроль» достаточно щёлкнуть по пиктограмме «замка» или кнопке соответственно, и ввести пароль в форму, как показано на рисунке П.1.15.

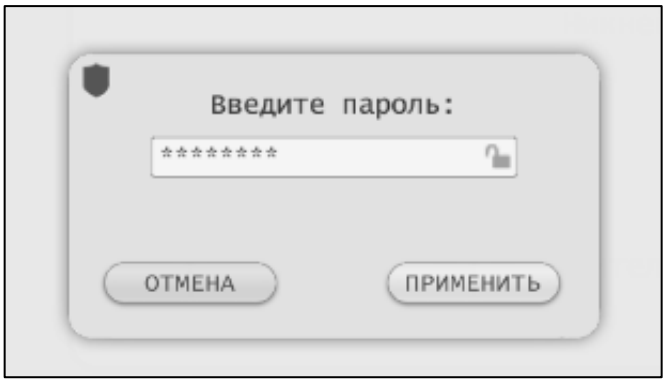

Рис. П.1.15. Форма ввода пароля

На вкладке «Видео», изображённой на рисунке П.1.16, игрок может выбрать качество изображения, разрешение экрана и режим «в окне» или «на весь экран». Качество изображения влияет в первую очередь на дальность прорисовки, дальность и качество теней, метод сглаживания, разрешение текстур, максимальное количество источников света, количество высоко-полигональных объектов на одной локации и некоторые визуальные эффекты пост-обработки.

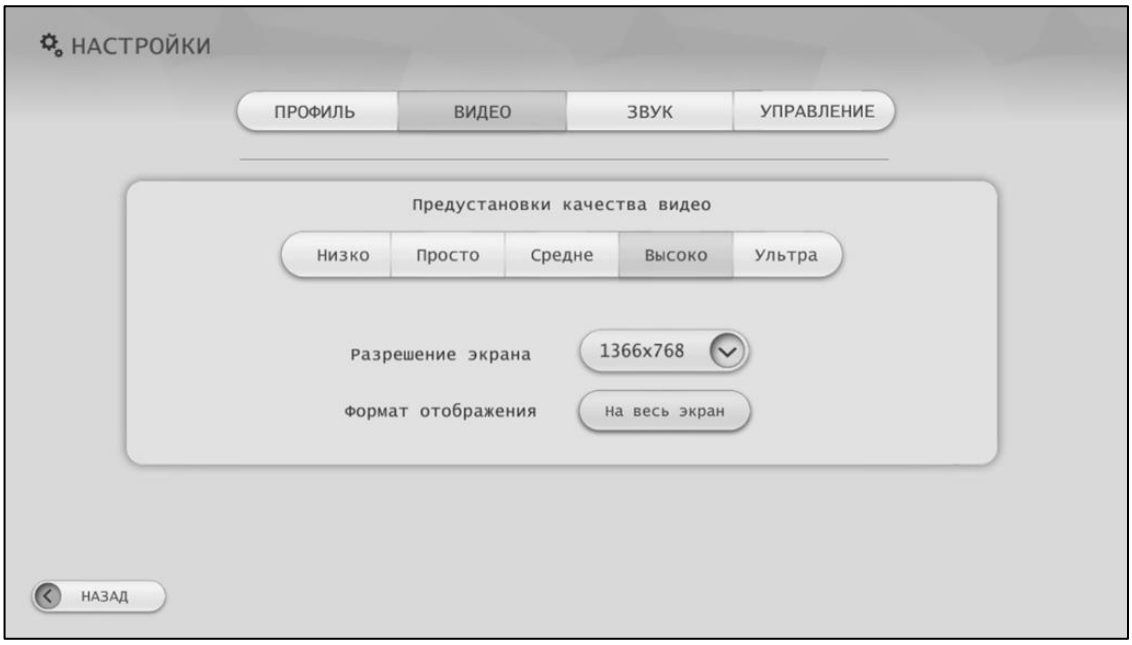

Рис. П.1.16. Настройки видео

При изменении любых настроек появляется форма, изображённая на рисунке П.1.17, и игроку даётся 10 секунд на то, чтобы подтвердить изменения, иначе приложение автоматически вернёт предыдущие значения. Это необходимо, если новые параметры вызвали сбой в выводе изображения и выбор других параметров невозможен или затруднителен. Такое может случаться на устаревших персональных компьютерах и отсутствие данной опцией вызвало бы невозможность дальнейшего использования приложения.

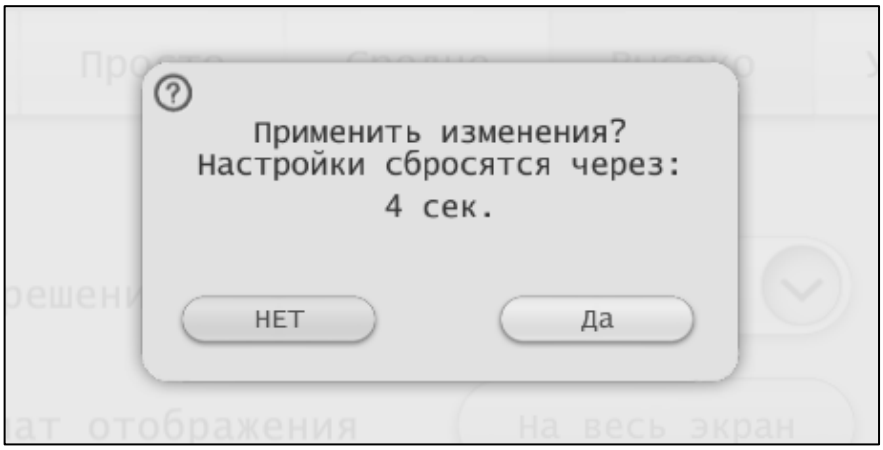

Рис. П.1.17. Форма подтверждения настроек

Выбор разрешения экрана выполнен в виде выпадающего списка, как показано на рисунке П.1.18, с перечислением всех возможных вариантов. Установленное в данный момент разрешение помечено специальной пиктограммой. Список автоматически заполняется данными, предоставляемыми видеодрайвером, начиная с наибольшего разрешения до наименьшего доступного.

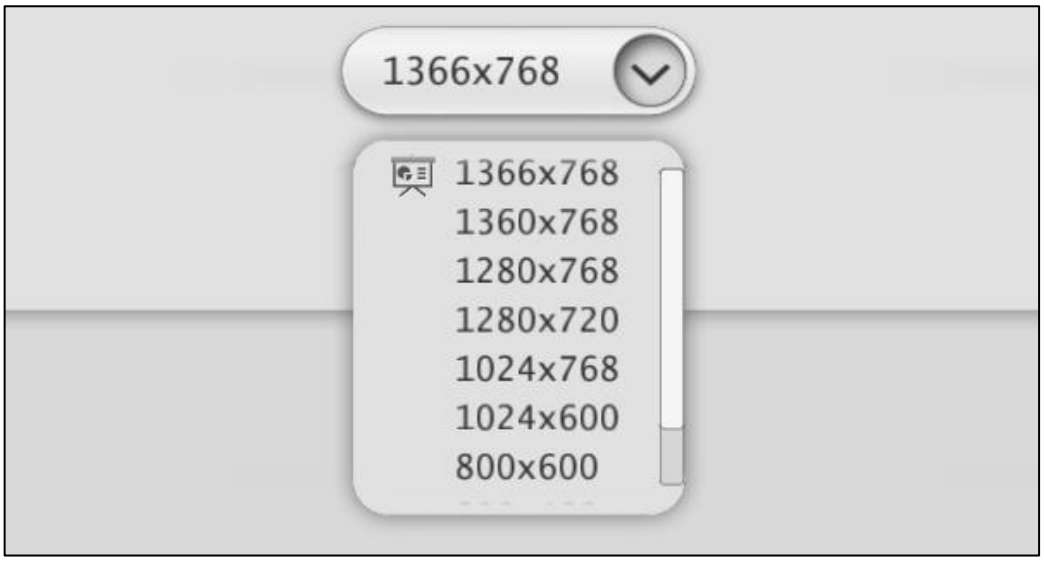

Рис. П.1.18. Список доступных разрешений экрана

П1.11. Меню паузы

Во время игрового процесса иногда требуется приостановить игру для того, чтобы отвлечься от игрового процесса. Для этого в приложении предусмотрено меню «Пауза», изображенное на рисунке П.1.20.

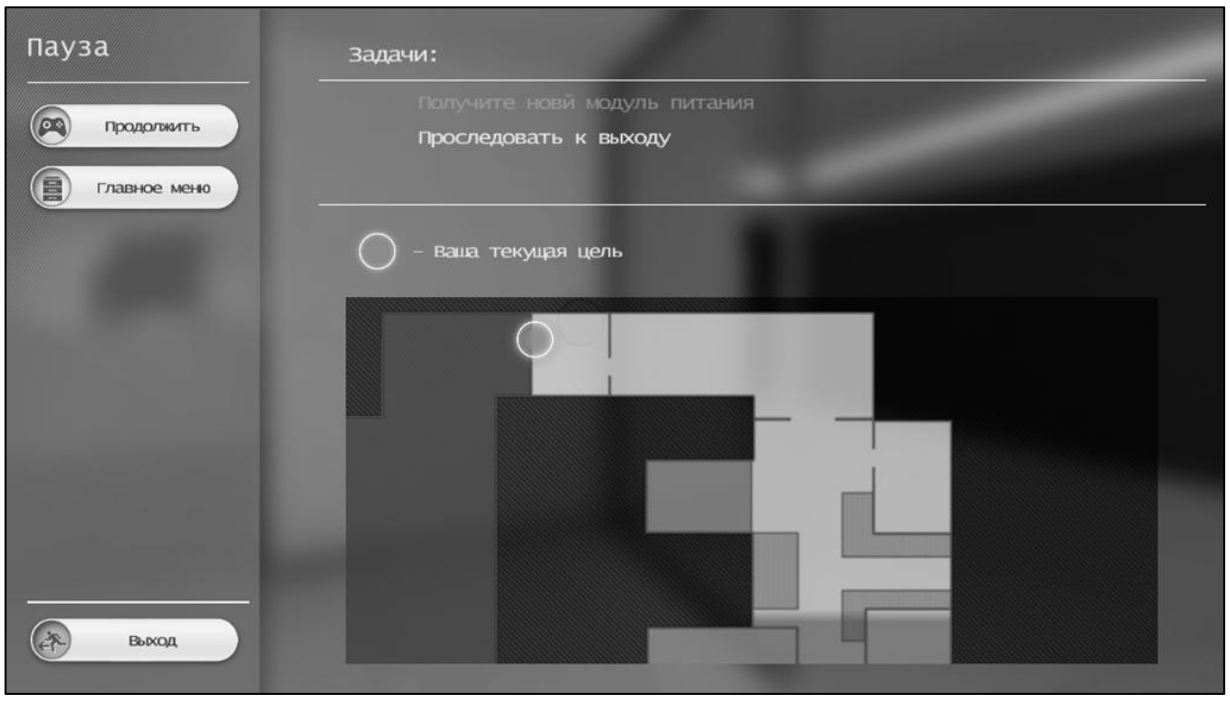

Рис. П.1.20. Меню «пауза»

При вызове данного меню, экран мутнеет и заменяется, а игровой процесс приостанавливается. Экран меню содержит в левой части кнопки для продолжения игры, перехода в главное меню и выхода, а в центральной перечисление текущих целей, мини-карту уровня с возможностью её прокручивать. Если игрок забыл, что ему требуется сделать в данный момент, то текущую цель можно найти в меню «паузы».

П1.12. Обучающие подсказки

В игре присутствуют моменты, когда игрок знакомится с элементами управления персонажем. Чтобы помочь пользователю быстрее освоиться с расположением управляющих клавиш, в графическом интерфейсе предусмотрены подсказки. На рисунке П.1.21 продемонстрированы подсказки клавиш перемещения персонажа и установки игры на паузу. Данная подсказка отображается в начале игры, когда приложение передаёт управление главным героем игроку и скрывается после нажатия одной из указанных клавиш клавиатуры. Для упрощения адаптации игрока знакомство с управлением разделено на этапы, переходя от основных элементов, таких как движения мышью и клавиши перемещения, к дополнительным, как, например, клавиши масштаба мини-карты, бега и прыжка.

Помимо подсказок, отображаемых при изучении управления, в игре присутствуют «подсказки действия», которые сообщают игроку о доступных действиях над тем или иным объектом. Например, на рисунке П.1.22 пользователь видит подсказку доступных действий для сферического робота, которая отображается если посмотреть на интересующий объект.

Рис. П.1.21. Подсказки клавиш управления

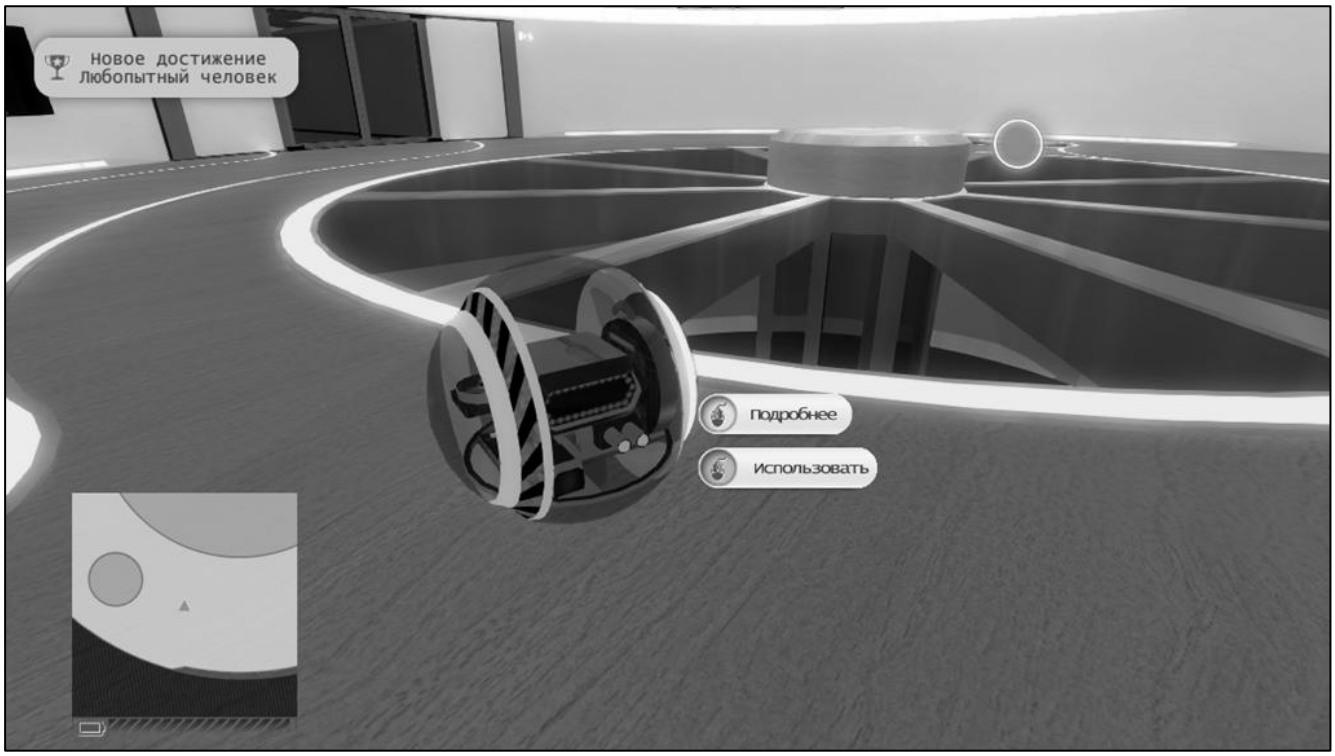

Рис. П.1.22. Подсказки действия и индикация игрока

П1.13. Индикация игрока

Дополнительные графические элементы интерфейса, отображаемые во время игры, называются индикацией игрока. Изображённая на рисунке П.1.22, индикация содержит мини-карту локации и заряда аккумуляторной батареи робота. Эти элементы расположены в нижней левой части рисунка П.1.22. Также, на экране отображается метка текущей цели, которая проецируется на цель в игровом пространстве (показано в верхней левой части рисунка П.1.22). После прочтении информационной «карточки» игрок получит бонусные очки и о чём его информирует сообщение с пиктограммой в верхнем левом углу. Обновление текущей цели также сопровождается сообщением и звуковым сигналом.

П1.14. Информационные «карточки»

Данный элемент пользовательского интерфейса представляет собой диалоговое окно, изображённое на рисунке П.1.23. Окно содержит заголовок просматриваемой темы, тематическое изображение, подробное текстовое описание, кнопку для перехода по ссылке на источник в сети и кнопку закрытия окна. Текстовое поле с вертикальной прокруткой автоматически подстраивается под длину текста. После просмотра «карточки» игрок получает бонусные очки.

Рис. П.1.23. Информационная «карточка» о Королёве Сергее Павловиче

П1.15. Ограничение времени одного игрового сеанса

В игре предусмотрено ограничение максимальной продолжительности по времени одного игрового сеанса. По истечению выделенного времени приложение сохранит все данные и игроку будет показано сообщение, как на рисунке П.1.24, с предложением сделать перерыв. После нажатия на кнопку «Хорошо», приложение закроется, и игрок сможет вернуться в игровой процесс после 10-ти минутного перерыва.

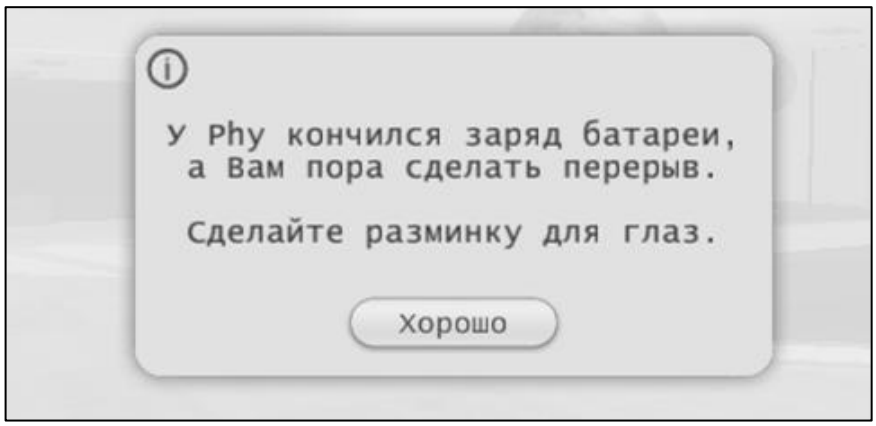

Рис. П.1.24. Диалоговое окно

П1.16. Управление оборудованием

Управлением оборудованием для экспериментов выполняется через специальный интерфейс, изображённый на рисунке П.1.25. При подключении главного героя, камера берёт заранее установленный удобный ракурс и в левой части экрана

отображается панель с кнопками «Назад» для возвращения в игру, и кнопка с обучающей подсказкой по данной предметной области. Для управления теми или иными входными значениями разных параметром или их отображение, используются различные ползунки, кнопки и другие элементы. Набор таких элементов интерфейса может варьироваться для разных задач.

Рис. П.1.25. Интерфейс управления оборудованием

ПРИЛОЖЕНИЕ 2

ТЕКСТ ПРОГРАММЫ

Файл «CelLabel.cs»

Выполняет отображение в проекции графической метки текущей цели на экране игрока. Если цель не находится в прямой видимости, то метка отображается полупрозрачной.

```
using UnityEngine;
using Configuration;
using UnityEngine.UI;
using UnityStandardAssets.ImageEffects;

[AddComponentMenu("MyScripts/Interface/CelLabel")]
public class CelLabel : MonoBehaviour
{
    public delegate void CellDelegate();
    public event CellDelegate Check;

    [Header("Целевой объект")]
    public GameObject Cel;
    [Header("Следующая цель [не обязательно]")]
    public CelLabel nextCel;

    RectTransform Full;
    [Header("Метка целиуказателя")]
    public RectTransform Target;
    [Header("Подложка целиуказателя")]
    public Image _up;
    Image _base;
    float _widthSCale;
    float _heightScale;
    public float MaxDist = 200;

    Camera FPC;
    [Header("Объекты связанные с целью")]
    public GameObject[] children;
    AudioSource Pick;

    [Header("Цвет текста неактивной метки")]
    public Color Disablet = new Color(0.5f,0.5f,0.5f,0.7f);
    public Text inform;

    [Header("Анимация управления инфорационным сообщением")]
    public Animator ShowNewCell;
    Vector3 vec;
    RaycastHit rcf;
    bool Used = false;

    /// <summary>
    /// Инициализация основных компонент и элементов интерфеса
    /// </summary>
    void Start()
    {
        if (Cel == null)
        {
            gameObject.tag = "Cell";
```

```
            Cel = gameObject;
        }

        Full = FindObjectOfType<Pause>().GetComponent<RectTransform>();
        FPC = FindObjectOfType<BlurOptimized>().GetComponent<Camera>();
        Pick = Full.GetComponent<AudioSource>();

        _widthSCale = Full.rect.width * Full.GetComponent<Canvas>().scaleFactor;
        _heightScale = Full.rect.height * Full.GetComponent<Canvas>().scaleFactor;
        _base = Target.GetComponent<Image>();
    }

    /// <summary>
    /// Игрок вошел в зону активации при достижении цели
    /// </summary>
    void OnTriggerEnter(Collider obj)
    {
        if (obj.tag != "Player" || Used)
            return;
        Enable_Disable(false);
        if (Check != null)
            Check();
        if (nextCel != null && !nextCel.enabled)
            nextCel.Enable_Disable(true);
        else
            Pick.Play();
        Used = true;
    }

    /// <summary>
    /// Построение луча для проецирования метки целеуказателя на экране пользова-
теля
    /// </summary>
    void Update()
    {
        if (!MenuConfig.OnPause)
        {
            vec = FPC.WorldToScreenPoint(Cel.transform.position);
            float x=0;
            float y=0;

            if (vec.z > 0)
            {
                x = Mathf.Clamp(vec.x, 0, _widthSCale);
                y = Mathf.Clamp(vec.y, 0, _heightScale);

                if (Physics.Raycast(new Ray(FPC.transform.position,
this.transform.position - FPC.transform.position), out rcf, MaxDist))
                {
                    bool selectCell = rcf.collider.tag == "Cell";
                    _base.enabled = !selectCell;
                    _up.enabled = selectCell;
                }
            }
            else
            {
                _base.enabled = true;
                _up.enabled = false;

                if (vec.x <= _widthSCale && vec.x >= 0)
                {
                    x = Mathf.Clamp(_widthSCale - vec.x, 0, _widthSCale);
                    y = 0;
```

```
                }
                else
                {
                    x = ((vec.x > (_widthSCale) / 2) ? 0 : _widthSCale);
                    y = Mathf.Clamp(vec.z, 0, _heightScale);
                }
            }
            Target.transform.position = new Vector3(x, y, 0);
        }
    }

    /// <summary>
    /// Включение или отключение целеуказателя
    /// </summary>
    public void Enable_Disable(bool option)
    {
        enabled = option;
        if (Full == null)
            Start();

        Target.gameObject.SetActive(option);
        for (int q = 0; q < children.Length; children[q].SetActive(option), q++) ;
        if (option == false)
            inform.color = Disablet;
        else
        {
            ShowNewCell.Play("ShowCellUpdate", 0);
            Pick.Play();
            inform.gameObject.SetActive(option);
        }
    }
}
```

Файл «DetectInfoPlane.cs»

Отслеживает объекты, которые попадают в область зрения игрока, и если объект обладает специальным тегом, то отображается подсказка для активации информационной панели описываемой классом InformationCard. При активации отображается «карточка» с текстовой и графической информацией об объекте.

InformationCard – содержит заголовок просматриваемой темы, тематическое изображение, подробное текстовое описание, ссылку на первоисточник и количество бонусных очков за просмотр.

```
using UnityEngine;
using System;
using UnityEngine.UI;
using UnityStandardAssets.Characters.FirstPerson;
using Configuration;

[AddComponentMenu("MyScripts/Interface/DetectInfoPlane")]
public class DetectInfoPlane : MonoBehaviour
{
    Vector3 vec;
    [Header("Контроллер и камера игрока")]
    public FirstPersonController FPC;
    public Camera MC;
    [Header("Подсказка 'подробнее'")]
    public GameObject ToolTip;
```

```
    [Header("Подсказка 'подключиться'")]
    public GameObject ToolTipControlled;
    [Header("Подсказка 'отключиться' [не обязательно]")]
    public GameObject ToolTipDisControlled;
    [Header("Полотно информационной 'карточки'")]
    public GameObject InfoPlane;
    AudioSource Pick;
    Client_Server cs;
    public Animator Dostij;
    [Header("Заголовок \ Изображение \ Содержимое")]
    public Text HEAD;
    public Image IMA;
    public Text CONT;
    InformationCard localCard;
    [Header("Ассет с содержимым")]
    public ContentManager DataContent;
    RaycastHit hit;
    Ray sel;
    int n;
    ParamsContent nameSelectObj;
    public bool I_Read = false;
    bool ILock = false;
    [Header("Набор ключевых тегов")]
    public localTags ContrilTag = new localTags(Tags.InfoObj);
    public localTags GlobeTag = new localTags(Tags.Globe);
    public localTags GlobePlayerTag = new localTags(Tags.GlobePlayer);
    Animator ani;
    GlobeControll GCon;
    [Header("Максимальная дистанция до отображения подсказки")]
    public int MaxDist = 20;

    /// <summary>
    /// Инициализация основных компонент, направляющего вектора и элементов интер-
феса
    /// </summary>
    void Awake()
    {
        vec = new Vector3(Screen.width / 2, Screen.height / 2, 0);
        if (MenuConfig.GetProfile.HaveServerConnect)
        {
            Pick = FindObjectOfType<Pause>().GetComponent<AudioSource>();
            cs = FindObjectOfType<Pause>().GetComponent<Client_Server>();
        }

        GlobePlayerAniTranslate tmp = FindObjectOfType<GlobePlayerAniTranslate>();
        if (tmp!=null)
            ani = tmp.GetComponent<Animator>();
    }

    /// <summary>
    /// Построение луча для определения объекта внимания игрока
    /// </summary>
    void Update ()
    {
        sel = MC.ScreenPointToRay(vec);
        if (!ILock)
        {
            ToolTipOff();

            if (!I_Read && Physics.Raycast(sel, out hit, MaxDist) &&
                  MenuConfig.OnPause == false)
            {
                GameObject tmp = hit.collider.gameObject;
```

```
            if (tmp.tag == ContrilTag || tmp.tag == GlobeTag)
                ShowInfoPlane(tmp);
            else if (tmp.tag == GlobePlayerTag)
            {
                ShowInfoPlane(tmp);
                ShowControl();
            }
            else
                ToolTipOff();
        }
    }
    else
    {
        if (Input.GetKeyDown(KeyCode.Mouse1))
        {
            ILock = false;
            ani.SetTrigger("event_Conect");
        }
    }
}

/// <summary>
/// Отображение подсказки 'подключиться' и обработка действия
/// </summary>
private void ShowControl()
{
    if (ToolTipControlled!=null)
        ToolTipControlled.SetActive(true);

    if (Input.GetKeyDown(KeyCode.Mouse1))
    {
        ani.SetTrigger("event_Conect");
        ILock = true;
        FPC.On_off(false);
    }
}

/// <summary>
/// Подключение или отключение от управления сферическим роботом
/// </summary>
public void CanControlGlobe()
{
    if (ILock)
    {
        transform.localRotation = new Quaternion(0, 0, 0, 0);
        GCon = hit.collider.gameObject.GetComponent<GlobeControll>();
        GCon.getPlayerControl(true, transform.parent);
        ToolTipControlled.SetActive(false);
        ToolTip.SetActive(false);
        ToolTipDisControlled.SetActive(true);
    }
    else
    {
        FPC.On_off(true);
        GCon.getPlayerControl(false, null);
        ToolTipDisControlled.SetActive(false);
    }
}

/// <summary>
/// Отображение информационной карточки
/// </summary>
private void ShowInfoPlane(GameObject g_o)
```

```
    {
        ToolTip.SetActive(true);
        if (Input.GetKeyDown(KeyCode.Mouse0))
        {
            nameSelectObj = g_o.GetComponent<ParamsContent>();
            n = nameSelectObj.Number;
            localCard = DataContent.GetCard(n);
            if (localCard == null)
            {
                Debug.LogError("Не найдены данние по id = " + n);
                return;
            }
            HEAD.text = localCard.head;
            IMA.sprite = localCard.images;

            CONT.text = localCard.infos;
            CONT.rectTransform.sizeDelta = new Vector2(0, localCard.setHigth);

            InfoPlane.SetActive(true);
            ToolTip.SetActive(false);
            I_Read = true;
            FPC.HaveLook = FPC.HaveMove = false;
            Cursor.visible = true;
        }
    }

    /// <summary>
    /// Скрытие подсказок
    /// </summary>
    private void ToolTipOff()
    {
        if (ToolTipControlled!=null)
            ToolTipControlled.SetActive(false);
        if (ToolTip != null)
            ToolTip.SetActive(false);
    }

    /// <summary>
    /// Закрытие информационной карточки с определением бнусных очков
    /// </summary>
    public void HIDH()
    {
        FPC.HaveLook = FPC.HaveMove = true;
        InfoPlane.gameObject.SetActive(false);
        I_Read = false;
        Cursor.visible = false;

        if (localCard.Score > 0 && MenuConfig.GetProfile.HaveServerConnect &&
MenuConfig.DonT_haveDoubleLook(nameSelectObj._tag))
        {
            cs.AddScor(localCard.Score);
            Dostij.Play("ShowCellUpdate", 0);
            Pick.Play();
        }
    }

    /// <summary>
    /// ОТкрыть ссылкку на внешний источник
    /// </summary>
    public void OpenURL()
    {
        Application.OpenURL(localCard.WikiURL);
    }
```

```
}

/// <summary>
/// Представление информационной карточки
/// </summary>
[Serializable]
public class InformationCard
{
    public string head;
    public Sprite images;
    [TextArea(5, 100)]
    public string infos;
    public int setHigth;
    public string WikiURL;
    public int Score = 50;
}
```

Файл «GradientSlider.cs»

```
using UnityEngine;
using UnityEngine.UI;

[AddComponentMenu("MyScripts/Menu/GradientSlider")]
[RequireComponent(typeof(Slider))]
public class GradientSlider : MonoBehaviour
{
    Slider sl;
    Image background;
    [Header("Градиент окраски")]
    public Gradient colors;

    /// <summary>
    /// Инициализация элементов интерфеса
    /// </summary>
    void Awake ()
    {
        sl = GetComponent<Slider>();
        background = sl.fillRect.GetComponent<Image>();
    }

    /// <summary>
    /// Окраска подложки слайдера, в зависимости от указанной им величины
    /// </summary>
    void Update ()
    {
        background.color = colors.Evaluate((sl.value - sl.minValue) /
            (sl.maxValue - sl.minValue));
      }
}
```

Файл «ElectroMagnetic.cs»

Выполняет управление электромагнитной катушкой по смежному интерфейсу. Описывает физической поведение работающей катушки при заданных значениях силы тока, напряжения, сопротивления обмотки и рабочей площади электромагнита.

```
using UnityEngine;
using System.Collections;
```

```
using UnityEngine.UI;
using UnityStandardAssets.Characters.FirstPerson;

[AddComponentMenu("MyScripts/Scene/ElectroMagnetic")]
public class ElectroMagnetic : MonoBehaviour
{
    [Header ("Управление колбой")]
    public Transform Gilz;
    public Transform ColliderGilz;
    Vector3 deltaPosition;
    float deltaY;
    Rigidbody GilzRB;
    [Header("Управление игроком")]
    public Transform Position_And_Rotate;
    [Range (0.05f, 10f)] public float SmoothTime = 2f;
    float smooth;
    FirstPersonController FPC;
    Transform CameraPlayer;
    bool CanControl = false;
    bool onArea = false;
    [Header("Управление электромагнитом")]
    public Transform Magnit;
    [Range(1, 200)] public int KoefEM = 2;
    [Range(0, 200)] public int minGC = 26;
    [Range(0, 200)] public int maxGC = 200;
    [Range(0.001f, 0.5f)] public float deltaDist = 0.05f;
    [Range(0.001f, 5f)] public float deltaR = 1.2f;
    Vector3 UpPoint;
    Vector3 DownPoint;
    Vector3 needPoint;
    float step;
    public float r=0;
    float delta = 0;
    public Vector3 vec;
    [Header("Управление интерфейсом")]
    public GameObject Econnect;
    public GameObject EM_UI;
    public Slider PositionMagnet;
    public Slider IntensiveMagnet;
    public Slider Tempreture;
    public Text TempretureVal;
    public string gragCels = " C";
    public localTags id = new localTags(Tags.Player);
    int tempGC = 26;

    /// <summary>
    /// Инициализация основных компонент и элементов интерфеса
    /// </summary>
      void Awake ()
    {
        smooth = SmoothTime * Time.deltaTime;
        FPC = FindObjectOfType<FirstPersonController>();
        CameraPlayer = FPC.m_Camera.transform;
        GilzRB = Gilz.GetComponent<Rigidbody>();
        deltaY = Gilz.position.y - ColliderGilz.position.y;
        deltaPosition = new Vector3(ColliderGilz.position.x, Gilz.position.y +
            deltaY, ColliderGilz.position.z);

        Tempreture.maxValue = maxGC + 20;
        Tempreture.minValue = minGC - 20;
        UpPoint = Magnit.position;
        needPoint = UpPoint;
        DownPoint = Gilz.position;
```

```
        step = (UpPoint.y - DownPoint.y) / PositionMagnet.maxValue;
      }

    /// <summary>
    /// Игрок вошел в зону активации
    /// </summary>
    void OnTriggerEnter(Collider obj)
    {
        if (obj.gameObject.tag == id)
        {
            Econnect.SetActive(true);
            onArea = true;
        }

    }

    /// <summary>
    /// Игрок вышел из зоны активации
    /// </summary>
    void OnTriggerExit(Collider obj)
    {
        if (obj.gameObject.tag == id)
        {
            Econnect.SetActive(false);
            onArea = false;
        }
    }

    /// <summary>
    /// Управление електромагнитом
    /// </summary>
      void Update ()
    {
        deltaPosition.y = Gilz.position.y - deltaY;
        ColliderGilz.position = deltaPosition;

        if (onArea && !CanControl && Input.GetAxis("Fire2") > 0)
            Connect_Disconnect(true);

        if (CanControl)
        {
            FPC.transform.position = Vector3.Lerp(FPC.transform.position,
                  Position_And_Rotate.position, smooth);
            CameraPlayer.localRotation = Quaterni-
on.Lerp(CameraPlayer.localRotation,
                  Quaternion.Euler(Position_And_Rotate.eulerAngles.x, 0, 0),
            smooth);

            FPC.transform.rotation = Quaternion.Lerp(FPC.transform.rotation,
                  Quaternion.Euler(0, Position_And_Rotate.eulerAngles.y, 0),
            smooth);

            Tempreture.value = tempGC;
            TempretureVal.text = tempGC + gragCels;
            r = (Magnit.position.y - Gilz.position.y);
            r = 1 + r * r;
            vec = (Vector3.up * IntensiveMagnet.value * KoefEM) / r;
            if (vec.y < GilzRB.mass*10)
            {
                GilzRB.isKinematic = false;
                GilzRB.AddForce(vec);
            }
            else
```

```
                if (r < deltaR)
                {
                    GilzRB.isKinematic = true;
                    Gilz.position = new Vector3(Gilz.position.x,
                        Magnit.position.y, Gilz.position.z);
                }
                else
                {
                    GilzRB.isKinematic = false;
                    GilzRB.AddForce(vec);
                }

                needPoint.y = UpPoint.y - (step * PositionMagnet.value);
                Magnit.position = Vector3.LerpUnclamped(Magnit.position,
                  needPoint, deltaDist);
            }
        }
    /// <summary>
    /// Подключение или отключение игрока по смежному интерфейсу
    /// </summary>
    public void Connect_Disconnect(bool oper)
    {
        FPC.HaveLook = FPC.HaveMove = !oper;
        CanControl = oper;
        Econnect.SetActive(!oper);
        EM_UI.SetActive(oper);
        Cursor.visible = oper;
    }
}
```

Файл «GetAKB.cs»

```
using UnityEngine;
using UnityStandardAssets.Characters.FirstPerson;

[AddComponentMenu("MyScripts/Scene/GetAKB")]
public class GetAKB : MonoBehaviour
{
    [Header("Целевой объект")]
    public CelLabel cl;
    [Header("Контроллер анимации игрока")]
    public Animator ani;
    [Header("Контроллер игрока")]
    public FirstPersonController FPC;
    [Header("Контроллер анимации интерфейса")]
    public Animator canvas;
    [Header("Следующая цель")]
    public CelLabel nextCel;
    [Header("Контроллер загрузки перехода на новую локацию")]
    public DetectAndLoadNext DetectAndLoad;

    /// <summary>
    /// Инициализация элементов интерфеса
    /// </summary>
    void Awake()
    {
        cl.Check += new CelLabel.CellDelegate(Detect);
    }

    /// <summary>
    /// Игрок вошел в зону активации анимации
    /// </summary>
```

```
    void Detect()
    {
        ani.enabled = FPC.CanJump = FPC.CanShift = true;
        ani.Play("GetAKB", 0);
    }

    /// <summary>
    /// Уничтожение объекта "батарея" со сцены
    /// </summary>
    public void KillAKB()
    {
        Destroy(cl.gameObject);
    }

    /// <summary>
    /// Отображение сообщения
    /// </summary>
    public void ShowInfoPlane()
    {
        canvas.gameObject.SetActive(true);
        nextCel.Enable_Disable(true);
        DetectAndLoad.Active = true;
    }
}
```

Файл «MusicPlay.cs»

```
using UnityEngine;
using System.Collections;

[AddComponentMenu("MyScripts/Generic/MusicPlay")]
[RequireComponent(typeof(BoxCollider))]
[RequireComponent(typeof(AudioSource))]
public class MusicPlay : MonoBehaviour
{
    AudioSource Player;

    /// <summary>
    /// Инициализация основных компонент
    /// </summary>
    void Awake()
    {
        Player = GetComponent<AudioSource>();
    }

    /// <summary>
    /// Игрок вошел в зону воспроизведения аудио-дорожки
    /// </summary>
    void OnTriggerEnter(Collider col)
    {
        if (col.tag == "Player")
        {
            GetComponent<BoxCollider>().enabled = false;
            Player.Play();
        }
    }
}
```

Файл «MenuConfig.cs»

Класс содержит основные данные о приложении, служебные переменные и названия всех сцен, а также методы чтения и записи в файл сохранений.

```
using System;
using System.IO;
using System.Xml.Serialization;

namespace Configuration
{
    public static class MenuConfig
    {
        #region "GameInfo"
        public static float VersionGame = 2.5f;
        public static string NameGame = "PHYSI LABS";
        public static string GameAutor = "Васильев Евгений";
        #endregion

        public static int LoadSceneName = 1;
        public static int MainMenuSceneName = 2;
        public static int OneLevelSceneName = 3;
        public static int OtherSceneSceneName = 4;
        public static int ThreeLevelSceneName = 5;
        public static int FourLevel = 6;
        public static int TimeBank = 0;
        public static int TimeOversouce = 0;
        static int LoadNext = -1;
        static string[] NameLevels = new string[] { "ПЕРВОЕ ВКЛЮЧЕНИЕ",
                  "ЗА ЗАПЧАСТЯМИ", "ЧЕРЕЗ ГАЛЕРЕЮ", "ПРИТЯНУТЫЙ ЗА УШИ" };

        public static bool OnPause = false;
        static string path = AppDomain.CurrentDomain.BaseDirectory + "Prusse.rtm";
        public static ProfileINFO Profile;

        /// <summary>
        /// Получить или задать следуюшию загужаемую локацию
        /// </summary>
        public static int NextLevel
        {
            get
            {
                return LoadNext;
            }
            set
            {
                LoadNext = value;
                if (value > GetProfile.LastLevel)
                    Profile.LastLevel = value;
            }
        }

        /// <summary>
        /// Обновить данные в файле сохранений
        /// </summary>
        public static void UpdateFile()
        {
            if (!new FileInfo(path).Exists)
            {
                Profile = new ProfileINFO();
                FileStream FS = new FileStream(path, FileMode.Create);
                FS.Close();
            }
```

```
            XmlSerializer Whriter = new XmlSerializer(typeof(ProfileINFO));

            using (FileStream fs = new FileStream(path, FileMode.Create))
            {
                Whriter.Serialize(fs, Profile);
            }
        }

        /// <summary>
        /// Считать данные из файла сохранений
        /// </summary>
        public static void ReadFromFile()
        {
            if (!new FileInfo(path).Exists)
                UpdateFile();
            else
            {
                XmlSerializer Whriter = new XmlSerializer(typeof(ProfileINFO));
                using (FileStream fs = new FileStream(path, FileMode.Open))
                {
                    Profile = (ProfileINFO)Whriter.Deserialize(fs);
                }
            }
        }
        /// <summary>
        /// Получить представление профиля игрока
        /// </summary>
        public static ProfileINFO GetProfile
        {
            get
            {
                if (Profile != null)
                    return Profile;
                else
                {
                    ReadFromFile();
                    return Profile;
                }
            }
        }

        /// <summary>
        /// Завершили прохождение уровня
        /// </summary>
        /// <param name="levelId">Идентификатор уровня</param>
        public static void LevelComplate(int levelId)
        {
            if (Profile == null)
                ReadFromFile();
            if (Profile.LevelList.Length < levelId - MainMenuSceneName)
            {
                LevelsComplate[] tmp = Profile.LevelList.Clone() as
LevelsComplate[];
                Profile.LevelList = new
LevelsComplate[Profile.LevelList.Length+1];
                tmp.CopyTo(Profile.LevelList, 0);
            }
            Profile.LevelList[levelId - OneLevelSceneName] = new LevelsComplate
                (NameLevels[levelId - OneLevelSceneName],
                (TimeOversouce - TimeBank)/60);

            UpdateFile();
        }
```

```
        /// <summary>
        /// Говорим, что начали проходить уровень
        /// </summary>
        /// <param name="levelId">Идентификатор уровня</param>
        public static void LevelStart(int levelId)
        {
            if (Profile == null)
                ReadFromFile();

            if (Profile.LevelList.Length < levelId - MainMenuSceneName)
            {
                LevelsComplate[] tmp = Profile.LevelList.Clone() as
LevelsComplate[];
                Profile.LevelList = new LevelsComplate[Profile.LevelList.Length +
1];
                tmp.CopyTo(Profile.LevelList, 0);
            }

            Profile.LevelList[Profile.LevelList.Length-1] = new LevelsComplate
                (NameLevels[levelId - OneLevelSceneName], -1);

            UpdateFile();
        }
        /// <summary>
        /// Проверка на повторное прохождение уровня
        /// </summary>
        /// <param name="levelId">Идентификатор уровня</param>
        /// <returns>Истина, если прохождение является первым</returns>
        public static bool DonT_haveDoubleRun(int levelId)
        {
            return Profile.LevelList[levelId - OneLevelSceneName]._timeRun == -1;
        }

        /// <summary>
        /// Проверка на повторный просмотр информационного табло
        /// </summary>
        /// <param name="nameObj">Название объекта</param>
        /// <returns>Истина, если просмотр является первым</returns>
        public static bool DonT_haveDoubleLook(string nameObj)
        {
            bool ok = true;
            for (int q = 0; q < Profile.LookObj.Length && ok; q++)
            {
                if (Profile.LookObj[q] == nameObj)
                    ok = false;
            }

            if (ok)
            {
                string[] tmp = Profile.LookObj.Clone() as string[];
                Profile.LookObj = new string[Profile.LookObj.Length + 1];
                tmp.CopyTo(Profile.LookObj, 0);
                Profile.LookObj[Profile.LookObj.Length - 1] = nameObj;
                UpdateFile();
            }
            return ok;
        }
    }
```

```
[Serializable]
public class ProfileINFO
{
    public string ID;
    public string NickName;
    public string Name;
    public string Surname;
    public string DOPInfarmation;
    public bool HaveServerConnect;
    public DateTime TimeLastPlay;
    public int TimeLimits;
    public string Passford;
    public int TimeInGame;
    public int LastLevel;
    public LevelsComplate[] LevelList;
    public string[] LookObj;

    /// <summary>
    /// Инициализировать объект
    /// </summary>
    public ProfileINFO()
    {
        ID = "";
        NickName = "";
        Name = "";
        Surname = "";
        DOPInfarmation = "";

        HaveServerConnect = false;
        LastLevel = 2;
        TimeLastPlay = DateTime.Now;
        TimeLimits = 60;
        Passford = GetSecretPass("");
        TimeInGame = 0;
        LevelList = new LevelsComplate[0];
        LookObj = new string[0];
    }

    /// <summary>
    /// Задать новый пароль
    /// </summary>
    /// <param name="NewPassford">Новый пароль</param>
    public void SetPass(string NewPassford)
    {
        Passford = GetSecretPass(NewPassford);
    }

    string GetSecretPass(string NoSecretPass)
    {
        return (M6.Md5Sum(MenuConfig.GameAutor + NoSecretPass +
              MenuConfig.NameGame + MenuConfig.VersionGame.ToString()));
    }

    /// <summary>
    /// Проверка корректности введённого пароля
    /// </summary>
    /// <param name="inpunPass">Введённый пароль</param>
    /// <returns>Истина, если парль достоверен</returns>
    public bool HaveCorrectPass(string inpunPass)
    {
        return GetSecretPass(inpunPass) == Passford;
    }
```

```
        /// <summary>
        /// Проверка на наличие пароля
        /// </summary>
        public bool havePass
        {
            get
            {
                  return Passford != null && Passford != GetSecretPass("");
            }
        }

        /// <summary>
        /// Проверка на пройденную регистрацию
        /// </summary>
        public bool haveRegestration
        {
            get{return ID != null && ID != "" && NickName != null && NickName !=
"";}
        }
    }

    [Serializable]
    public class LevelsComplate
    {
        public string _nameLevel;
        public string _dateRun;
        public int _timeRun;

        /// <summary>
        /// Инициализация экземпляра не пройденного уровня
        /// </summary>
        public LevelsComplate()
        {
            _nameLevel = "";
            _timeRun = -1;
            _dateRun = "";
        }

        /// <summary>
        /// Инициализация экземпляра Пройденного уровня
        /// </summary>
        /// <param name="nameLevel">название уровня</param>
        /// <param name="timeRun"> время прохождения</param>
        public LevelsComplate(string nameLevel, int timeRun)
        {
            _nameLevel = nameLevel;
            _timeRun = timeRun;
            _dateRun = DateTime.Now.ToString("H:mm (dddd) dd MMMM yyyy г.",
                  new System.Globalization.CultureInfo("ru-RU"));
        }
    }
}
```

Файл «DoorControl.cs»

Данный класс описывает поведение дверей и люков, которые имеют одинаковый подход, ввиду схожести большинства свойств и методов.

```
using UnityEngine;
using System;
```

```
public enum Tags{PlaneRun, Globe, Cell, InfoObj, Player, ContrilTag, GlobePlayer};

[Serializable]
public class localTags
{
    public Tags tag;
    bool used = false;

    public localTags(Tags _tag)
    {
        tag = _tag;
    }

    /// <summary>
    /// Свойство задаёт \ возвращает активность элемента
    /// </summary>
    public bool set_get_used
    {
        get { return used;}
        set { used = value; }
    }

    /// <summary>
    /// Получение строкового представления локального тега
    /// </summary>
    /// <param name="lt">Локальный тег</param>
    /// <returns>Строка</returns>
    static public implicit operator string(localTags lt)
    {
        return lt.tag.ToString();
    }

    /// <summary>
    /// Возвращает количество активных элеметов
    /// </summary>
    /// <param name="lt">Массив локальных тегов</param>
    /// <returns>Число вхождений</returns>
    static public int GetNumsTrue(localTags[] lt)
    {
        int nTrue = 0;
        for (int q = 0; q < lt.Length; q++)
            if (lt[q].set_get_used)
                nTrue++;
        return nTrue;
    }

    /// <summary>
    /// Проверка на наличие входящего соответствия по Тэгу
    /// </summary>
    /// <param name="lt">Массив локальных тегов</param>
    /// <param name="shablon">Шаблон для сравнения</param>
    /// <returns>Ислина, если вхождение обнаружено</returns>
    static public bool HaveNameTag(localTags[] lt, string shablon)
    {
        for (int q = 0; q < lt.Length; q++)
            if (lt[q] == shablon)
                return true;

        return false;
    }
}
```

```
[AddComponentMenu("MyScripts/Door/DoorController")]
public class DoorControl : MonoBehaviour
{
    [Header("Активность двери")]
    public bool Active = true;
    [Header("Контроллер анимации двери")]
    public Animator ani;

    [Header("Набор ключевых тегов")]
    public localTags[] lTag = new localTags[3]
    {
        new localTags(Tags.Globe),
        new localTags(Tags.GlobePlayer),
        new localTags(Tags.Player)
    };

    [Header("Источник звука")]
    public AudioSource shum;

    /// <summary>
    /// Игрок вошел в зону активации
    /// </summary>
    void OnTriggerEnter(Collider obj)
    {
        detect(obj.tag, 1, true);
    }

    /// <summary>
    /// Игрок вышел из зоны активации
    /// </summary>
    void OnTriggerExit(Collider obj)
    {
        detect(obj.tag, 0, false);
    }

    /// <summary>
    /// Определение инициатора и действия открытия или закрытия
    /// </summary>
    void detect(string srcTag, int count, bool oper)
    {
        if (!Active)

            return;
        for (int q = 0; q < lTag.Length; q++)
            if (srcTag == lTag[q].tag.ToString())
                lTag[q].set_get_used = oper;

        if (localTags.GetNumsTrue(lTag) == count)
        {
            ani.SetBool("Open", oper);
            shum.Play();
        }
    }
```

Файл «DetectAndLoadNext.cs»

Определяет, что игрок вошёл в область для загрузки следующей локации, блокирует управление и вызывает метод «затемнения» в ExtendetLoadControl, записывает все данные по пройденному уровню в MenuConfig. При активном подключении к серверной части пакет данных передаётся в Client_Server. После всех

операций загружает сцену для загрузки новой локации и передаёт управление LoadControl.

```
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityStandardAssets.Characters.FirstPerson;
using Configuration;

[AddComponentMenu("MyScripts/Phy/DetectAndLoadNext")]
public class DetectAndLoadNext : MonoBehaviour
{
    [Header("Элемент управляющий затемнением экрана")]
    public ExtendetLoadControl elc;
    Client_Server cs;
    [Header("Очки за прохождение")]
    public int Score = 500;
    [Header("Активность области")]
    public bool Active = true;

    /// <summary>
    /// Инициализация основных компонент
    /// </summary>
    void Start()
    {
        MenuConfig.LevelStart(SceneManager.GetActiveScene().buildIndex);
        cs = GameObject.FindObjectOfType<Client_Server>();
    }

    /// <summary>
    /// Игрок вошел в зону активации при достижении цели
    /// </summary>
    void OnTriggerEnter(Collider obj)
    {
        if (Active && obj.tag == "Player")
        {
            int id = SceneManager.GetActiveScene().buildIndex;
            if (MenuConfig.GetProfile.HaveServerConnect &&
                  MenuConfig.DonT_haveDoubleRun(id))
                cs.AddScor(Score);

            MenuConfig.NextLevel = id + 1;
            MenuConfig.LevelComplate(id);
            obj.GetComponent<FirstPersonController>().enabled = false;
            elc.function = ExtendetLoadControl.Functions.loadinglevel;
            elc.Show();
        }
    }
}
```

Файл «ExtendetLoadControl.cs»

Используется для эффекта «выцветание» при переходе к загрузке другой сцены и в некоторых элементах пользовательского интерфейса.

```
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
using Configuration;

[AddComponentMenu("MyScripts/Menu/ExtendetLoadControl")]
```

```
public class ExtendetLoadControl : MonoBehaviour
{
    [Header("Подложка фона")]
    public Image im;
    public enum Lstate { on, none, off};

    [Header("Состояние процесса")]
    public Lstate state = Lstate.none;
    public enum Functions { none, exit, loadinglevel};
    [Header("Конечная функция")]
    public Functions function = Functions.none;
    [Header("Кривая интенсивности")]
    public AnimationCurve Logic;
    [Header("Шаг интенсивности")]
    [Range(0.001f, 0.5f)] public float step = 0.1f;
    float times = 0;
    [Header("Набор цветов Старт / Выход / Окончание")]
    public Color StartColor;
    public Color ExitColor;
    public Color EndColor;
    [Header("Звуковые источники")]
    public AudioSource[] Sounds;
    [Header("Громкость звуковых источников")]
    public float[] OldVol;

    /// <summary>
    /// Инициализация основных компонент
    /// </summary>
    void Awake()
    {
        OldVol = new float[Sounds.Length];
        for (int q = 0; q < Sounds.Length; OldVol[q] = Sounds[q].volume, q++) ;
        im.color = StartColor;
    }

    /// <summary>
    /// Выполнение функциональной части и управнений подложкой
    /// </summary>
    void FixedUpdate()
    {
        if (state == Lstate.off)
        {
            float el = 1f - Logic.Evaluate(times += step);
            im.color = new Color(im.color.r, im.color.g, im.color.b, el);
            if (Sounds!=null)
                for (int q = 0; q < Sounds.Length;
                        Sounds[q].volume = OldVol[q] * (1f-el), q++) ;
            if (times >= 1)
            {
                state = Lstate.none;
                times = 0;
            }
        }
        else if (state == Lstate.on)
        {
            float el = Logic.Evaluate(times += step);
            im.color = new Color(im.color.r, im.color.g, im.color.b, el);
            if (Sounds != null)
                for (int q = 0; q < Sounds.Length;
                  Sounds[q].volume = OldVol[q] * (1f-el), q++) ;

            if (times >= 1)
            {
```

```
                Cursor.visible = false;
                state = Lstate.none;
                switch (function)
                {
                    case Functions.exit:
                        MenuConfig.Profile.TimeLastPlay = System.DateTime.Now;
                        MenuConfig.Profile.TimeInGame +=
                              MenuConfig.Profile.TimeLimits*60 -
                            MenuConfig.TimeBank;
                        MenuConfig.UpdateFile();
                        Application.Quit();
                        break;
                    case Functions.loadinglevel:
                        SceneManager.LoadScene(MenuConfig.LoadSceneName);
                        break;
                }
            }
        }
    }

    /// <summary>
    /// Инициализировать прояснение
    /// </summary>
    public void Show()
    {
        state = Lstate.on;
        HowFunc();
    }

    /// <summary>
    /// Определитьель цвета в зависимости от функции
    /// </summary>
    private void HowFunc()
    {
        if (function == Functions.exit)
            im.color = ExitColor;
        else
            im.color = EndColor;
    }

    /// <summary>
    /// Инициализировать затемнение
    /// </summary>
    public void Hide()
    {
        state = Lstate.off;
        HowFunc();
    }
}
```

Файл «LoadControl.cs»

Управляет асинхронной загрузкой новой сцены в игре и используется для устранения блокировки системы. По завершению загрузки управление переда-ёт ExtendetLoadControl.

```
using UnityEngine;
using System.Collections;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
using Configuration;
```

```
[AddComponentMenu("MyScripts/LoadControl")]
public class LoadControl : MonoBehaviour
{
    [Header("Подложка фона")]
    public Image im;
    public enum Lstate { on, wait, off };
    Lstate state = Lstate.on;
    [Header("Кривая интенсивности")]
    public AnimationCurve Logic;
    [Range (0.001f, 0.5f)]
    public float step = 0.1f;
    float times = 0;
    [Header("Набор цветов Старт / Окончание")]
    public Color StartColor;
    public Color EndColor;
    [Header("Изображение скрина")]
    public Image LoadScreen;
    public Sprite[] imageForLoadScreen;
    AsyncOperation async;

    /// <summary>
    /// Инициализация основных компонентов
    /// </summary>
    void Awake()
    {
        Cursor.visible = false;
        im.color = StartColor;
        LoadScreen.sprite = imageForLoadScreen[MenuConfig.NextLevel -
            MenuConfig.MainMenuSceneName];
    }

    /// <summary>
    /// Выполнение функциональной части и управнений подложкой
    /// </summary>
      void FixedUpdate ()
    {
        if (state == Lstate.on)
        {
            im.color = new Color(im.color.r, im.color.g, im.color.b,
                  1f - Logic.Evaluate(times += step));
            if (times >= 1)
            {
                state = Lstate.wait;
                im.color = EndColor;
                times = 0;

                if (MenuConfig.NextLevel != -1 &&
                  MenuConfig.NextLevel !=
            SceneManager.GetActiveScene().buildIndex)
                {
                    CoruntineLoad();
                }
            }
        }
        if (state == Lstate.off)
        {
            im.color = new Color(im.color.r, im.color.g, im.color.b,
                  Logic.Evaluate(times += step));


            if (times >= 1)
                Complate();
```

```
        }
      }

    /// <summary>
    /// Начало загрузки
    /// </summary>
    void CoruntineLoad()
    {
        async = SceneManager.LoadSceneAsync(MenuConfig.NextLevel);
        async.allowSceneActivation = false;
        StartCoroutine(LoadLevelCoroutine(async));
    }

    /// <summary>
    /// Ассинхронная загрузка
    /// </summary>
    /// <param name="async"></param>
    /// <returns></returns>
    IEnumerator LoadLevelCoroutine(AsyncOperation async)
    {
        while (!async.isDone)
        {
            if (async.progress >= 0.9f)
                break;
            yield return null;
        }
        state = Lstate.off;
        yield return async;
    }

    /// <summary>
    /// Окончание загрузки
    /// </summary>
    void Complate()
    {
        async.allowSceneActivation = true;
    }
}
```

Файл «ItemManager.cs»

Представляет элемент пользовательского интерфейса для отображения «карточек» с информацией о пройденном уровне в игре. В атрибутах содержит изображение заставки уровня, название, дату и время прохождения. Карточка уровня может выделяться в случае, если игрок решил пройти его повторно.

```
using UnityEngine;
using UnityEngine.UI;

[AddComponentMenu("MyScripts/Menu/ItemMenenger")]
public class ItemManager: MonoBehaviour
{
    public Image _Image;
    public Text _Name;
    public Text _Date;
    public Text _TimeRun;
    public Toggle tg;
    public int _NumLevel;

    /// <summary>
```

```
    /// Инициализация карточки уровня
    /// </summary>
    /// <param name="imageSource">Изображение</param>
    /// <param name="nameLevel">Низвание</param>
    /// <param name="dateComplate">Дата прохождения</param>
    /// <param name="numLevel">Номер уровня</param>
    /// <param name="timeRun">Время прохождения</param>
    public void SetVals(Sprite imageSource, string nameLevel,
      string dateComplate, int numLevel, int timeRun)
    {
        _Image.sprite = imageSource;
        _Name.text = nameLevel;
        _Date.text = "Время: " + dateComplate;
        _NumLevel = numLevel;
        if(timeRun==-1)
            _TimeRun.text = "Уровень не пройден до конца";
        else
            _TimeRun.text = "Время прохождения: " + timeRun + " мин.";
    }
}
```

Файл «MenuControl.cs»

Координирует действия в основном меню по отображению экранов загрузки, настроек и управление анимацией игровых объектов на сцене. Также выполняет запрос на чтение данных пользователя из файла.

```
using UnityEngine;
using UnityEngine.UI;
using Configuration;

[AddComponentMenu("MyScripts/Menu/MenuControl")]
public class MenuControl : MonoBehaviour
{
    [Header("Контроллер анимации руки")]
    public Animator ani;
    [Header("Элемент управляющий затемнением экрана")]
    public ExtendetLoadControl elc;
    [Header("Кривая интенсивности")]
    public RectTransform Loaded;
    public RectTransform Setter;
    public RectTransform Profile;
    public Text uka;
    string mode = "none";
    [Header("Контроллер анимации камеры")]
    public Animator Cam_ani;

    /// <summary>
    /// Инициализация основных компонентов
    /// </summary>
    void Awake()
    {
        Cursor.visible = true;
        uka.text = MenuConfig.GetProfile.LastLevel >
            MenuConfig.MainMenuSceneName ? "ПРОДОЛЖИТЬ" : "НАЧАТЬ ИГРУ";
    }

    [Header("Набор идентификаторов пальцев")]
    public string[] singers = new string[] { "Big", "Uka", "Fuc", "Bez", "Miz"};

    /// <summary>
```

```
/// Курсор навёлся на палец
/// </summary>
/// <param name="Finger">Идентификатор пальца</param>
public void SetMouseEnter(string Finger)
{
    foreach (string f in singers)
    {
        if (Finger == f)
            ani.SetBool(f, true);
        else
            ani.SetBool(f, false);
    }
}

/// <summary>
/// Действие 'Старт' или 'Загрузить'
/// </summary>
/// <param name="select">Выбранный уровень</param>
public void StartLoadSelectLevel(MenuLoadControll select)
{
    MenuConfig.NextLevel = select.selectI;

    elc.function = ExtendetLoadControl.Functions.loadinglevel;
    elc.Show();
}

/// <summary>
/// Действие 'Продолжить'
/// </summary>
public void StartContinueClick()
{
    if (MenuConfig.GetProfile.LastLevel <= MenuConfig.MainMenuSceneName)
    {
        MenuConfig.Profile.LastLevel = MenuConfig.OneLevelSceneName;
        MenuConfig.UpdateFile();
    }

    MenuConfig.NextLevel = MenuConfig.GetProfile.LastLevel;
    elc.function = ExtendetLoadControl.Functions.loadinglevel;
    elc.Show();
}

/// <summary>
/// Выход
/// </summary>
public void ExitClick()
{
    elc.function = ExtendetLoadControl.Functions.exit;
    ani.SetBool("Exit", true);
    elc.Show();
}

/// <summary>
/// Открыть 'Згрузки' или 'настройки'
/// </summary>
public void LoadOrSetClick(string _mode)
{
    mode = _mode;
    Cam_ani.SetBool("CamInLS", true);
}

/// <summary>
/// Вернуться назад
```

```
    /// </summary>
    public void BackClick(){ Cam_ani.SetBool("CamInLS", false); }

    /// <summary>
    /// Показать или скрыть элементы подменю
    /// </summary>
    public void ShowOrHideMenuItems()
    {
        if (mode == "Load")
            Loaded.gameObject.SetActive(!Loaded.gameObject.activeInHierarchy);
        else if (mode == "Setting")
            Setter.gameObject.SetActive(!Setter.gameObject.activeInHierarchy);
        else if (mode == "Profile")
            Profile.gameObject.SetActive(!Profile.gameObject.activeInHierarchy);
    }

    /// <summary>
    /// Задать уровень анимации
    /// </summary>
    /// <param name="val">Уровень</param>
    public void SetWight(float val)
    {
        ani.SetLayerWeight(0, val);
    }
}
```

Файл «MiniMap.cs»

Данный класс управляет отображением мини карты в активном режиме игры (не в меню паузы) с указанием позиции игрока. Также под схематичным изображением уровня располагается индикатор заряда батареи игрового персонажа. Напомним, что значение заряда пропорционально максимальной продолжительности одного сеанса в игре, который устанавливается в настройках родительского контроля.

```
using UnityEngine;
using UnityEngine.UI;
using Configuration;

[AddComponentMenu("MyScripts/Interface/MiniMap and HUD")]
public class MiniMap : MonoBehaviour
{
    [Header("Камера мини-карты")]
    public Camera MC;
    [Header("Ограничивающий прямоугольник")]
    public Rect rect;
    [Header("Набор величин масштаба карты")]
    public float[] MapScate = new float[] { 60, 40, 20 };
    int numSelect = 1;
    [Range (0.05f, 10f)]
    public float speedScate = 0.5f;
    [Header("Основная область экрана")]
    public RectTransform Full;
    [Header("Область заряда батареи")]
    public RectTransform Other;
    [Header("Полотно экрана")]
    public Canvas HUDCanvas;
    Slider _OtherSlider;
    [Header("Оставшееся время")]
    public long TimeOut = 0;
    public Pause _pause;
```

```
    /// <summary>
    /// Инициализация основных компонентов
    /// </summary>
    void Start ()
    {
#if UNITY_EDITOR
        MenuConfig.TimeBank = MenuConfig.GetProfile.TimeLimits*60;
#endif
        MenuConfig.TimeOversouce = MenuConfig.TimeBank;
        MC.rect = new Rect(rect.x, rect.y, ((float)Screen.height) /
            Screen.width * rect.width, rect.height);

        Other.transform.position = new Vector3(Full.rect.width * rect.x *
      HUDCanvas.scaleFactor, Full.rect.height * rect.y * HUDCanvas.scaleFactor,
0);
        Other.sizeDelta = new Vector2(Full.rect.height * rect.height, 32);

        _OtherSlider = Other.GetComponent<Slider>();
        _OtherSlider.maxValue = MenuConfig.GetProfile.TimeLimits*60;
        _OtherSlider.value = MenuConfig.TimeBank;
        TimeOut = (int)_OtherSlider.value * 50;
      }

    /// <summary>
    /// Отсчёт времени
    /// </summary>
    void FixedUpdate()
    {
        if (Time.timeScale == 0)
            return;

        if (TimeOut > 0)
            TimeOut--;
        else
            _pause.ComleteGame();
    }

    void Update()
    {
        _OtherSlider.value = TimeOut / 50;
        MenuConfig.TimeBank = (int)TimeOut / 50;
        if (Input.GetButtonDown("ScaleMap"))
        {
            numSelect = numSelect+1 >= MapScate.Length ? 0 : numSelect+1;
        }
        MC.orthographicSize = Mathf.Lerp(MC.orthographicSize,
            MapScate[numSelect], speedScate * Time.deltaTime);
    }
}
```

Файл «MenuLoadControll.cs»

Выполняет чтение из файла сохранённых данных о пройденных уровнях и отображает их во вкладке загрузок. Данные формируются в экземпляры класса ItemMenenger и выводятся списком «карточек». Также имеется кнопка для быстрой загрузки последнего сохранения.

```
using UnityEngine;
using Configuration;
using UnityEngine.UI;

[AddComponentMenu("MyScripts/Menu/MenuLoadControll")]
```

```
public class MenuLoadControll : MonoBehaviour
{
    public Sprite[] _imagesLevels;
    public ItemMenenger IM;
    public RectTransform ContentRect;
    public Button Loader;
    public int selectI = -1;

    void Start ()
    {
        int num = MenuConfig.GetProfile.LevelList.Length;
        if (num <= 0)
            return;
        int fullnumD = MenuConfig.OneLevelSceneName;
        float wight = IM.GetComponent<RectTransform>().rect.height;
        LevelsComplate[] lc = MenuConfig.GetProfile.LevelList;
        Loader.interactable = true;

        for (int q = 0; q < num; q++)
        {
            GameObject go = Instantiate<GameObject>(IM.gameObject);
            RectTransform goRT = go.GetComponent<RectTransform>();
            go.transform.parent = ContentRect.transform;
            goRT.localScale = new Vector3(1, 1, 1);
            goRT.sizeDelta = new Vector2(0, wight);
            goRT.localPosition = new Vector3(0, -q * (wight+2), 0);
            ItemMenenger tmpIM = go.GetComponent<ItemMenenger>();
            tmpIM.SetVals(_imagesLevels[q], lc[q]._nameLevel, lc[q]._dateRun,
                q + fullnumD, lc[q]._timeRun);

            if (q + 1 == num)
                tmpIM.tg.isOn = true;
            go.SetActive(true);
        }
        ContentRect.sizeDelta = new Vector2(0, num * (wight + 2));
    }


    public void SetLoadLevel(ItemMenenger _im)
    {
        selectI = _im._NumLevel;
    }
}
```

Файл «MenuProfileControl.cs»

Данный класс отвечает за управление формой с настройками профиля игрока.

```
using UnityEngine;
using UnityEngine.UI;
using Configuration;

[AddComponentMenu("MyScripts/Menu/MenuProfileControl")]
public class MenuProfileControl : MonoBehaviour
{
    public Gradient ColorDetect;
    public Slider TimeLim;
    public Image FillRect;
    public Text TimeLimINFO;
    [Header("Диалоговые окна")]
    public GameObject Lock;
    public GameObject Lock2;
    public GameObject SinglePassEnter;
    public GameObject DoublePassEnter;
```

```
    public GameObject RegestrationArea;
    [Header("Данные игрока")]
    public Text NickName;
    public InputField NameField;
    public InputField SurNameField;
    public InputField DOPInfoField;
    public Text TimeInGame;
    public Text LastTimeInGame;
    [Header("Поля ввода паролей")]
    public InputField OnePassford;
    public InputField TwoPassford;
    public InputField Dtetect;
    [Header("Кнопки и переключатели")]
    public Button Saver;
    public Toggle[] no_yes;
    public Toggle[] no_yes_Server;
    public int mode = 0;
    [Header("Клиент - сервер")]
    public Client_Server clientServerConnecter;
    public InputField NickNameField;
    public GameObject AnimationBlock;
    public GameObject ConnectButton;
    public Button CancelButton;
    public Image MessBlock;
    public Text Mess;
    public ScoresPanelControl spc;
    public Image PlaneInfo;
    public Text TextInfo;
    public Color ErrorColor;
    public Color WarningColor;
    public Color StateColor;
    public Color GoodColor;

    void Awake()
    {
        clientServerConnecter.DataSet +=
            new Client_Server.SimpleDelegate(DataSetRes);
    }

    void OnEnable()
    {
        TimeInGame.text = (MenuConfig.GetProfile.TimeInGame/60).ToString() + "
мин.";

        LastTimeInGame.text = MenuConfig.Profile.TimeLastPlay
.ToString("dd MMMM yyyy г.\nH:mm (dddd)", new Sys-
tem.Globalization.CultureInfo("ru-RU"));
        TimeLim.value = MenuConfig.Profile.TimeLimits;
        if(MenuConfig.Profile.haveRegestration)
            NickName.text = MenuConfig.Profile.NickName;

        NameField.text = MenuConfig.Profile.Name;
        SurNameField.text = MenuConfig.Profile.Surname;
        DOPInfoField.text = MenuConfig.Profile.DOPInfarmation;
        Lock.SetActive(MenuConfig.Profile.havePass);
        Lock2.SetActive(MenuConfig.Profile.havePass);
        Saver.interactable = false;
        clientServerConnecter.Result
            += new Client_Server.SimpleDelegate(ResultRegEv);

        PlaneInfo.gameObject.SetActive(false);
        DialogBack();
        DialogBackServer();
```

```
    }

    void Update()
    {
        FillRect.color = ColorDetect.Evaluate( (float)(TimeLim.value - 20) /
100f);
        string timesOut = "";

        if ((int)TimeLim.value / 60 == 1)
            timesOut += (int)TimeLim.value / 60 + " час ";
        if ((int)TimeLim.value / 60 < 2)
        {
            int minutes = (int)TimeLim.value % 60;
            timesOut += minutes + " ";
            if (minutes % 10 == 1)
                timesOut += "минута";
            else if (minutes % 10 > 1 && minutes % 10 < 5)
                timesOut += "минуты";
            else
                timesOut += "минут";
        }
        else
            timesOut += TimeLim.value / 60 + " часа";
        TimeLimINFO.text = timesOut;
    }

    public void Save()
    {
        spc.Enble();
        MenuConfig.Profile.Name = NameField.text;
        MenuConfig.Profile.Surname = SurNameField.text;
        MenuConfig.Profile.DOPInfarmation = DOPInfoField.text;
        MenuConfig.Profile.TimeLimits = (int)TimeLim.value;
        MenuConfig.UpdateFile();

        if(MenuConfig.Profile.HaveServerConnect)
        {
            clientServerConnecter.UpdateProfileData();
        }
    }

    public void DataSetRes(string _message, Client_Server.messages m)
    {
        PlaneInfo.gameObject.SetActive(true);
        TextInfo.text = _message;
        switch (m)
        {
            case Client_Server.messages.error:
                Saver.interactable = true;
                PlaneInfo.color = ErrorColor;
                break;
            case Client_Server.messages.nickname:
                Saver.interactable = true;
                PlaneInfo.color = WarningColor;
                break;
            case Client_Server.messages.OK:
                PlaneInfo.color = GoodColor;
                break;
        }
    }


    public void SetPassford()
```

```
{
    if (MenuConfig.Profile.HaveCorrectPass(Dtetect.text))
    {
        if (mode == 1)
            Lock.SetActive(false);
        if (mode == 11)
            Lock2.SetActive(false);
        else if (mode == -1)
        {
            Lock.SetActive(false);
            Lock2.SetActive(false);
            MenuConfig.Profile.SetPass("");
            MenuConfig.UpdateFile();
        }
        DialogBack();
    }
    else
        Detect.GetComponent<PassfordLook>().Error();
}
public void DialogBack()
{
    no_yes[0].isOn = !MenuConfig.GetProfile.havePass;
    no_yes[1].isOn = MenuConfig.GetProfile.havePass;
    no_yes[0].interactable = MenuConfig.GetProfile.havePass;
    no_yes[1].interactable = !MenuConfig.GetProfile.havePass;

    SinglePassEnter.SetActive(false);
    DoublePassEnter.SetActive(false);
    OnePassford.text = "";
    TwoPassford.text = "";
    Dtetect.text = "";
    mode = 0;
}

public void DialogBackServer()
{
    no_yes_Server[1].isOn = MenuConfig.Profile.HaveServerConnect;
    no_yes_Server[0].isOn = !MenuConfig.Profile.HaveServerConnect;

    no_yes_Server[0].enabled = true;
    no_yes_Server[1].enabled = true;
    SelectServer();
}

public void SelectServer()
{
    no_yes_Server[0].interactable = MenuConfig.Profile.HaveServerConnect;
    no_yes_Server[1].interactable = !MenuConfig.Profile.HaveServerConnect;
}

public void UpdatePassford()
{
    if (OnePassford.text != "" && TwoPassford.text != "" &&
        OnePassford.text == TwoPassford.text)
    {
        MenuConfig.Profile.SetPass(TwoPassford.text);
        MenuConfig.UpdateFile();
        DialogBack();
        Lock.SetActive(true);
        Lock2.SetActive(true);
    }

    if (OnePassford.text == "" || OnePassford.text == "" ||
```

```
            OnePassford.text != TwoPassford.text)
        {
            OnePassford.GetComponent<PassfordLook>().Error();
            TwoPassford.GetComponent<PassfordLook>().Error();
        }
    }

    public void Connect(bool val)
    {
        if (!val)
            return;
        if (!MenuConfig.GetProfile.haveRegestration)
            RegestrationArea.SetActive(true);
        else
        {
            MenuConfig.Profile.HaveServerConnect = true;
            Saver.interactable = true;
            SelectServer();
        }
    }
    public void Disconnect(bool val)
    {
        if (!val)return;
        MenuConfig.Profile.HaveServerConnect = false;
        Saver.interactable = true;
        SelectServer();
    }

    public void ResultRegEv(string _message, Client_Server.messages res)
    {
        CancelButton.interactable = true;
        ConnectButton.SetActive(true);
        AnimationBlock.SetActive(false);
        switch (res)
        {
            case Client_Server.messages.error:
                Mess.gameObject.SetActive(true);
                MessBlock.color = ErrorColor;
                Mess.text = _message;
                break;
            case Client_Server.messages.nickname:
                Mess.gameObject.SetActive(true);
                MessBlock.color = WarningColor;
                Mess.text = _message;
                break;
            case Client_Server.messages.OK:
                MenuConfig.Profile.HaveServerConnect = true;
                RegestrationArea.SetActive(false);
                Save();
                OnEnable();
                break;
        }
    }

    public void Regestration_Click()
    {
        Mess.gameObject.SetActive(false);
        MessBlock.color = StateColor;
        CancelButton.interactable = false;
        clientServerConnecter.Regestration(NickNameField.text);
    }
}
```

Файл «MenuSettingControl.cs»

Предназначен для управления формой с настройками игры и переключением между вкладками соответствующих разделов.

```
using UnityEngine;
using System.Collections;
using UnityEngine.UI;

[AddComponentMenu("MyScripts/Menu/MenuSettongControl")]
public class MenuSettongControl : MonoBehaviour
{
    public Dropdown dpScreenSize;
    Resolution[] rs;
    int timer = 250;
    bool timerWork = false;
    bool fullScr;
    int bpVal;

    public Text SecInfo;
    public GameObject DialogPanel;
    public Toggle tg;
    public Toggle[] TgRes;
    public Image SoftUpFon;
    public Color[] VideoSoundControlColor;
    int indexCollor = 3;
    [Range(0.01f, 1f)]
    public float speedColorLerp = 1f;

    void Awake ()
    {
        rs = Screen.resolutions;
        fullScr = Screen.fullScreen;
        tg.isOn = fullScr;

        for (int q = rs.Length-1; q >= 0; q--)
        {
            dpScreenSize.options.Add(new
Dropdown.OptionData(rs[q].width.ToString() +
             "x" + rs[q].height.ToString()));

            if (Screen.width == rs[q].width && Screen.height == rs[q].height)
                dpScreenSize.value = bpVal = (rs.Length) - q;
        }
        DialogPanelHide(true);
        TgRes[QualitySettings.GetQualityLevel()].isOn = true;
    }

    void FixedUpdate()
    {
        if (timerWork)
        {
            if (timer <= 0)
                DialogPanelHide(false);

            timer--;
            SecInfo.text = (timer/50).ToString() + " сек.";
        }
        SoftUpFon.color = Color.Lerp(SoftUpFon.color,
            VideoSoundControlColor[indexCollor], speedColorLerp);
    }
```

```
    public void SetSelectMenu(int n)
    {
        indexCollor = n;
    }

    #region VIDEO
    public void ChangeValue()
    {
        DialogPanelShow();
        UpdateResolution(Screen.fullScreen);
    }

    public void SetFullScreen(bool val)
    {
        fullScr = Screen.fullScreen;
        UpdateResolution(val);
        DialogPanelShow();
    }

    public void SetQuality(int Level)
    {
        QualitySettings.SetQualityLevel(Level);
    }
    public void DialogPanelShow()
    {
        timer = 250;
        timerWork = true;
        DialogPanel.SetActive(true);
    }

    public void DialogPanelHide(bool yes_no)
    {
        if (!yes_no)
        {
            tg.isOn = fullScr;
            UpdateResolution(tg.isOn);
            dpScreenSize.value = bpVal;
        }

        timerWork = false;
        bpVal = dpScreenSize.value;
        fullScr = Screen.fullScreen;
        DialogPanel.SetActive(false);
    }

    void UpdateResolution(bool _fullScr)
    {
        Screen.SetResolution(rs[(rs.Length - 1) - dpScreenSize.value].width,
            rs[(rs.Length - 1) - dpScreenSize.value].height, _fullScr);
    }
    #endregion
}
```

Файл «PasswordLook.cs»

Управляет работой области для ввода пароля в диалоговых окнах. Позволяет реализовать систему «подглядывания», то есть временного отображения скрытого пароля, по нажатию на соответствующую пиктограмму, а также подсвечивание поля при неправильном введённом пароле.

```
using UnityEngine;
using UnityEngine.UI;
```

```
[AddComponentMenu("MyScripts/Menu/PassfordLook")]
public class PasswordLook: MonoBehaviour
{
    public InputField passfordIn;
    public Text textLook;
    public Image back;
    public Gradient ErrorGrad;
    bool HaveError = false;
    [Range (0.005f, 0.5f)]
    public float speedEr = 0.03f;
    float delta = 0f;

    void Update ()
    {
        if(textLook!= null)
            textLook.text = passfordIn.text;
        if (HaveError)
        {
            back.color = ErrorGrad.Evaluate(delta += speedEr);
            if (delta >= 1)
            {
                delta = 0;
                HaveError = false;
            }
        }
    }
}
```

Файл «Pause.cs»

Управляет пользовательским интерфейсом при установке игры на паузу. Отображает меню, которое содержит перечисление текущих целей, мини карты уровня и кнопки для продолжения игры, перехода в главное меню и выхода.

```
using UnityEngine;
using Confinguration;
using UnityStandardAssets.Characters.FirstPerson;
using UnityStandardAssets.ImageEffects;

[AddComponentMenu("MyScripts/Menu/Pause")]
public class Pause : MonoBehaviour
{
    public GameObject[] Disable_Enablet;
    public GameObject UIPause;
    public GameObject CompleDialog;
    ExtendetLoadControl elc;
    FirstPersonController FPC;
    BlurOptimized blured;
    AudioSource[] SoundOnPause;
    bool[] playing;
    public bool _enable = true;

    void Start()
    {
       Cursor.visible = false;
       SoundOnPause = FindObjectsOfType<AudioSource>();
       elc = FindObjectOfType<ExtendetLoadControl>();
       FPC = FindObjectOfType<FirstPersonController>();
       blured = FindObjectOfType<BlurOptimized>();
    }
```

```
void Update()
{
    if (Input.GetButtonDown("Cancel") && _enable)
    {
        Rebreake();
    }
}

void Rebreake()
{
    if (Time.timeScale > 0)
    {
        playing = new bool[SoundOnPause.Length];
        for (int q = 0; q < SoundOnPause.Length; q++)
        {
            if (SoundOnPause[q] != null)
            {
                playing[q] = SoundOnPause[q].isPlaying;
                SoundOnPause[q].Pause();
            }
        }
        Cursor.visible = true;
        MenuConfig.OnPause = true;
    }
    else
    {
        Cursor.visible = false;
        MenuConfig.OnPause = false;

        for (int q = 0; q < SoundOnPause.Length; q++)
        {
            if (SoundOnPause[q] != null && playing[q])
                SoundOnPause[q].Play();
        }
    }

    blured.enabled = !blured.enabled;
    UIPause.SetActive(!UIPause.activeInHierarchy);
    Time.timeScale = 1 - (int)Time.timeScale;
    for (int q = 0; q < Disable_Enablet.Length;
        Disable_Enablet[q].SetActive(!Disable_Enablet[q].activeHierarchy),
  q++);
}

public void ComleteGame()
{
    playing = new bool[SoundOnPause.Length];

    for (int q = 0; q < SoundOnPause.Length;
        playing[q] = SoundOnPause[q].isPlaying, SoundOnPause[q].Pause(), q++);
    Cursor.visible = true;

    Time.timeScale = 0;
    for (int q = 0; q < Disable_Enablet.Length;
        Disable_Enablet[q].SetActive(!Disable_Enablet[q].activeHierarchy),
  q++);

    UIPause.SetActive(false);
    CompleDialog.SetActive(true);
    enabled = _enable = false;
}
```

```
    public void Resume()
    {
        Rebreake();
    }

    public void InMenu()
    {
        _enable = FPC.enabled = false;
        Time.timeScale = 1;

        MenuConfig.NextLevel = MenuConfig.MainMenuSceneName;
        elc.function = ExtendetLoadControl.Functions.loadinglevel;
        elc.Show();
    }

    public void Exit()
    {
        _enable = FPC.enabled = false;
        Time.timeScale = 1;
        elc.function = ExtendetLoadControl.Functions.exit;
        elc.Show();
    }
}
```

Файл «Client_Server.cs»

Выполняет обмен данными с серверной частью совместно с шифрованием передаваемых данных, а также обработку получение ответных команд и данных от серверной части. Обращение к данному классу выполняется, если установлена соответствующая опция в меню настроек игры.

```
using UnityEngine;
using System.Collections;
using Configuration;
using System.Security.Cryptography;
using System.Text;

[AddComponentMenu("MyScripts/Menu/Client_Server")]
public class Client_Server : MonoBehaviour
{
    public enum messages { error, nickname, OK };
    public delegate void SimpleDelegate(string message, messages mess);
    public event SimpleDelegate Result;
    public event SimpleDelegate GetScor;
    public event SimpleDelegate SetScor;
    public event SimpleDelegate DataSet;
    private string secretKey = "PhysiLabsAPP";
    public string addScoreUrl = "http://physilabs.hol.es/addscore.php";
    public string highscoreUrl = "http://physilabs.hol.es/display.php";
    public string regestrationUrl = "http://physilabs.hol.es/regestration.php";
    public string setdataprofUrl = "http://physilabs.hol.es/setdataprof.php";

    IEnumerator postScore(string _score)
    {
        string hash = M6.Md5Sum(MenuConfig.Profile.ID +
            MenuConfig.Profile.NickName + _score + secretKey);
        string highscore_url = addScoreUrl +
            "?id=" + WWW.EscapeURL(MenuConfig.Profile.ID) +
            "&nickname=" + WWW.EscapeURL(MenuConfig.Profile.NickName) +
            "&score=" + _score + "&hash=" + hash;
```

```
    WWW hs_post = new WWW(highscore_url);
    yield return hs_post;

    if (SetScor != null)
    {
        if (hs_post.error != null)
            SetScor("Ошибка: " + hs_post.error, messages.error);
        else
        {
            if (hs_post.text == "ProfileError")
                SetScor("Ошибка: несоответствие профиля", messages.nickname);
            else if (hs_post.text == "DataUpdate")
                SetScor("Данные обновлены", messages.OK);
        }
    }
    hs_post.Dispose();
}

IEnumerator ProfileDataSet()
{
    string hash = M6.Md5Sum(MenuConfig.Profile.ID +
        MenuConfig.Profile.NickName + MenuConfig.Profile.Name +
        MenuConfig.Profile.Surname + MenuConfig.Profile.DOPInfarmation +
        secretKey);

    string highscore_url = setdataprofUrl +
        "?id=" + WWW.EscapeURL(MenuConfig.Profile.ID) +
        "&nickname=" + WWW.EscapeURL(MenuConfig.Profile.NickName) +
        "&name=" + WWW.EscapeURL(MenuConfig.Profile.Name) +
        "&surname=" + WWW.EscapeURL(MenuConfig.Profile.Surname) +
        "&doinfo=" + WWW.EscapeURL(MenuConfig.Profile.DOPInfarmation) +
        "&hash=" + hash;

    WWW ds_post = new WWW(highscore_url);
    yield return ds_post;

    if (DataSet != null)
    {
        if (ds_post.error != null)
            DataSet("Ошибка: " + ds_post.error, messages.error);
        else
        {
            if (ds_post.text == "ProfileError")
                DataSet("Ошибка: несоответствие профиля", messages.nickname);
            else if (ds_post.text == "DataUpdate")
                DataSet("Данные обновлены", messages.OK);
        }
  ds_post.Dispose();
}

IEnumerator getScores()
{
    WWW hs_get = new WWW(highscoreUrl);
    yield return hs_get;

    if (GetScor != null)
    {
        if (hs_get.error != null)
            GetScor("Ошибка: " + hs_get.error, messages.error);
        else
            GetScor(hs_get.text, messages.OK);
    }
```

```
        hs_get.Dispose();
    }

    IEnumerator regestration(string _nickname)
    {
        string hash = M6.Md5Sum(_nickname + secretKey);
        string reg_url = regestrationUrl +
            "?nickname=" + WWW.EscapeURL(_nickname) +
            "&hash=" + hash;
        WWW hs_reg = new WWW(reg_url);

        yield return hs_reg;

        if (hs_reg.error != null)
        {
            if (Result != null)
                Result("Ошибка: " + hs_reg.error, messages.error);
        }
        else if(Result != null)
        {
            if (hs_reg.text == "RebreackNickname")
                Result("Данный никнейм уже занят", messages.nickname);
            else if (hs_reg.text == "RegError")
            {
             Result("Системная ошибка в базе, попробуйте позже.", messages.error);
            }
            else
            {
                MenuConfig.Profile.NickName = _nickname;
                MenuConfig.Profile.ID = hs_reg.text;
                MenuConfig.UpdateFile();
                Result(null, messages.OK);
            }
        }
        hs_reg.Dispose();
    }

    public void Regestration(string NickName)
    {
        StartCoroutine(regestration(NickName));
    }

    public void GetScores()
    {
        StartCoroutine(getScores());
    }

    public void AddScor(int val)
    {
        StartCoroutine(postScore(val.ToString()));
    }

    public void UpdateProfileData()
    {
        StartCoroutine(ProfileDataSet());
    }
}

static class M6
{
    public static string Md5Sum(string strToEncrypt)
    {
        UTF8Encoding encoding = new UTF8Encoding();
```

```
        var bytes = encoding.GetBytes(strToEncrypt);
        MD5CryptoServiceProvider md5 = new MD5CryptoServiceProvider();
        byte[] hashBytes = md5.ComputeHash(bytes);
        var hashString = "";

        for (var i = 0; i < hashBytes.Length; i++)
        {
            hashString +=
                  System.Convert.ToString(hashBytes[i], 16).PadLeft(2, "0"[0]);
        }
        return hashString.PadLeft(32, "0"[0]);
    }
}
```

Файл «ScoresPanelControl.cs»

Формирует список из 20-ти лучших игроков и выводит на панели достижений в виде «карточек» с перечислением никнеймов и результирующих очков. Панель достижений доступна из главного меню при активном соединении с сервером.

```
using UnityEngine;
using Configuration;
using UnityEngine.UI;

[AddComponentMenu("MyScripts/Menu/ScoresPanelControl")]
public class ScoresPanelControl : MonoBehaviour
{
    public GameObject Main;
    public GameObject Panel;
    public GameObject DownArrow;
    public GameObject UpArrow;
    public GameObject AnimanionBlock;
    public Image MessBlock;
    public Text Messager;
    public Client_Server clientServer;
    public Color ErrorColor;
    public Color StateColor;
    public InemClient Base;
    public RectTransform ContentRect;
    float wight;
    public InemClient[] ic;
    public Color Icolor;

    void Awake()
    {
        Enble();
        clientServer.GetScor += new Client_Server.SimpleDelegate(SetScores);
        wight = Base.GetComponent<RectTransform>().rect.height;
    }

    public void Enble()
    {
        Main.SetActive(MenuConfig.GetProfile.HaveServerConnect &&
            MenuConfig.GetProfile.haveRegestration);
    }

    void SetScores(string message, Client_Server.messages m)
    {
        if (m == Client_Server.messages.error)
        {
            AnimanionBlock.SetActive(false);
```

```
            Messager.text = message;
            MessBlock.color = ErrorColor;
        }
        else
        {
            AnimanionBlock.SetActive(false);
            MessBlock.gameObject.SetActive(false);
            string[] res = message.Split(new char[] { '\n' },
                  System.StringSplitOptions.RemoveEmptyEntries);
            for (int q = 0; q < ic.Length; Destroy(ic[q].gameObject), q++) ;
            ic = new InemClient[res.Length];
            for (int q = 0; q < res.Length; q++)
            {
                string[] nameScor = res[q].Split(new char[] { '\t' },
                  System.StringSplitOptions.RemoveEmptyEntries);
                GameObject go = Instantiate<GameObject>(Base.gameObject);
                if (MenuConfig.GetProfile.NickName == nameScor[0])
                    go.GetComponent<Image>().color = Icolor;
                RectTransform goRT = go.GetComponent<RectTransform>();
                go.transform.parent = ContentRect.transform;
                goRT.localScale = new Vector3(1, 1, 1);
                goRT.sizeDelta = new Vector2(0, wight);
                goRT.localPosition = new Vector3(0, -q * (wight + 2), 0);
                ic[q] = go.GetComponent<InemClient>();
                ic[q].SetVal(nameScor[0], nameScor[1]);
                go.SetActive(true);
            }
            ContentRect.sizeDelta = new Vector2(0, res.Length * (wight + 2));
        }
    }

    public void ShowHidh()
    {
        DownArrow.SetActive(!DownArrow.activeInHierarchy);
        UpArrow.SetActive(!UpArrow.activeInHierarchy);
        Panel.SetActive(!Panel.activeInHierarchy);
        if (Panel.activeInHierarchy)
        {
            MessBlock.color = StateColor;
            MessBlock.gameObject.SetActive(true);
            AnimanionBlock.SetActive(true);
            Messager.text = "";
            clientServer.GetScores();
        }
    }

    public void OpenSite()
    {
        Application.OpenURL("http://physilabs.hol.es");
    }
}
```

Файл «GlobeCameraControll.cs»

Данный класс описывает поведение камеры наблюдения и её взаимодействие с игроком и сферическим роботом. Выполняет функцию «слежения» за объектом и имеет ограниченную зону обзора.

```
using UnityEngine;
using System.Collections;
```

```
[AddComponentMenu("MyScripts/Globe/CameraController")]
public class GlobeCameraControll : MonoBehaviour
{
    public enum Ihave { none, player, globe };
    public Ihave have = Ihave.none;

    [Range (0.005f, 4f)]public float SpeedRot = 0.1f;
    public Transform ZeroQuartition;
    public Transform Target;
    public Transform Ramka;
    public Transform Cam;
    public AudioSource Speaker;
    public AudioClip[] HelloGlobe;
    public AudioClip[] HelloPhy;
    [Range(0.1f, 1f)] public float Phello = 0.5f;

    void OnTriggerEnter(Collider obj)
    {
        if (obj.tag == "Player")
        {
            have = Ihave.player;
            Target = obj.gameObject.transform;
            if (!Speaker.isPlaying && Random.Range(0f, 1f) < Phello)
            {
                Speaker.clip = HelloPhy[Random.Range(0, HelloPhy.Length)];
                Speaker.Play();
            }
        }
        else if (have == Ihave.none && obj.tag == "Globe")
        {
            have = Ihave.globe;
            Target = obj.gameObject.transform;

            if (!Speaker.isPlaying)
            {
                Speaker.clip = HelloGlobe[Random.Range(0, HelloGlobe.Length)];
                Speaker.Play();
            }
        }
    }

    void OnTriggerExit(Collider obj)
    {
        if (obj.tag == "Player" || (have == Ihave.globe && obj.tag == "Globe"))
        {
            have = Ihave.none;
            Target = null;
        }
    }

    void FixedUpdate ()
    {
        if (have != Ihave.none)
            Cam.rotation = Quaternion.Slerp(Cam.rotation,
                  Quaternion.LookRotation(transform.position - Target.position),
                  Time.deltaTime * SpeedRot);
        else
            Cam.rotation = Quaternion.Slerp(Cam.rotation, ZeroQuartition.rotation,
                  Time.deltaTime * SpeedRot);
        Ramka.rotation = Quaternion.Euler(0, Cam.rotation.eulerAngles.y, 0);
      }
}
```

Файл « GlobeControll.cs»

Данный класс выполняет управление сферическим роботом, как в автономном режиме, так и при подключении игрока.

OnDrawGizmos() – используется только в редакторе для визуальной обозначения маршрута. Изображается тонкими линиями через указанные точки в последовательности заданной pointMove.

```
using UnityEngine;
using System.Collections;

[AddComponentMenu("MyScripts/Globe/Controller")]
public class GlobeControll : MonoBehaviour
{
    [Header("Позиционирование")]
    public Vector3 CenterMass = new Vector3(0, -6, 0);
    public Transform Glass;
    public Transform FrontMoover;
    public enum State {run, speake, lissen, empty };
    public enum StateRun { start, run, end };
    public State MyState = State.run;
    public StateRun MyStateRun = StateRun.start;
    public GlobeControll brother;
    public GameObject ColliderPerfab;
    int i_point_move = 0;
    [Header("Настройки движения")]
    [Range (0.005f, 4f)] public float speed_raotate = 1f;
    Rigidbody body;
    public Transform[] pointMove = new Transform[0];
    public AnimationCurve graf;
    public float Aceleration = 0;
    [Range(0.005f, 0.1f)] public float AcelerationStep = 0.01f;
    float frontMoverX = 0;
    [Range (0.1f, 150f)] public float EndAcelerateDist = 10;
    [Range(0.005f, 1f)] public float MinAceleration = 0.1f;
    public bool haveContact = true;
    RaycastHit objCollider;
    float hightGlobe;
    [Range(0.005f, 0.1f)] public float StepSountAceleration = 0.01f;
    [Header("Настройки звука")]
    public AudioSource ShumRun;
    public AudioClip[] Hello;
    public AudioClip ForLight;
    public AudioClip[] GlobeBla;
    public AudioSource BlaBla;
    [Range(1, 10000)] public int timeLissenSpeake = 400;
    int timeLim = 0;
    [Header("Настройки управления игроком")]
    public bool CanControll = false;
    public bool PlayerControlled = false;
    public Quaternion PlayerControlRotation;
    public Transform ZeroCamPosition;

    void Start ()
    {
        timeLim = timeLissenSpeake;
        body = this.GetComponent<Rigidbody>();
        body.centerOfMass = CenterMass;
        frontMoverX = FrontMoover.localPosition.x;
        hightGlobe = GetComponent<SphereCollider>().radius + 0.1f;
    }
```

```
void FixedUpdate ()
{
    Physics.Raycast(new Ray(transform.position, Vector3.down),
        out objCollider, hightGlobe);
    haveContact = objCollider.collider == null ? false : true;
    if (PlayerControlled)
    {
        float horizontal = Input.GetAxis("Horizontal") * 2;
        float vertical = Input.GetAxis("Vertical");
        float yRot = Input.GetAxis("Mouse X");
        yRot = Mathf.Abs(horizontal) > Mathf.Abs(yRot) ? horizontal : yRot;

        PlayerControlRotation *= Quaternion.Euler(0f, yRot, 0f);

        transform.localRotation = Quaternion.Slerp(transform.localRotation,
              PlayerControlRotation, Time.deltaTime);
        ZeroCamPosition.position = Vector3.Lerp(ZeroCamPosition.position,
              transform.position, Time.deltaTime*5);
        ZeroCamPosition.rotation = Quaternion.Slerp(ZeroCamPosition.rotation,
              transform.rotation, Time.deltaTime * 2);

        if (!haveContact)
        {
            ZeroShum();
            return;
        }
        ShumRun.pitch = Mathf.Lerp(ShumRun.pitch, vertical,
              StepSountAceleration);
        if (vertical > float.Epsilon)
        {
            FrontMoover.localPosition = new Vector3(frontMoverX *
              graf.Evaluate(vertical), 0, 0);
            body.AddForce(FrontMoover.position - body.transform.position);
        }
        else if (vertical < -float.Epsilon)
        {
            FrontMoover.localPosition = new Vector3(frontMoverX *
              graf.Evaluate(-vertical), 0, 0);
            body.AddForce(-(FrontMoover.position - body.transform.position));
        }
    }

    if (!haveContact || (MyState == State.empty && !PlayerControlled))
    {
        ZeroShum();
        return;
    }

    if (pointMove != null && pointMove.Length != 0)
    {
        if (MyState == State.run)
        {
            timeLim = timeLim <
              timeLissenSpeake ? timeLim + 1 : timeLissenSpeake;
            Rotater(pointMove[i_point_move]);

            switch (MyStateRun)
            {
                case StateRun.start:
                    if (Aceleration < 1)
                        Aceleration += AcelerationStep;
                    else
```

```
                            MyStateRun = StateRun.run;
                        break;
                    case StateRun.run:
                        if (Vector3.Distance(transform.position,
                              pointMove[i_point_move].position) < EndAcelerateDist)
                            MyStateRun = StateRun.end;
                        break;
                    case StateRun.end:
                        Aceleration = Aceleration >
                  MinAceleration ? Aceleration - AcelerationStep : MinAceleration;
                        break;
                }

                FrontMoover.localPosition = new Vector3(frontMoverX *
                   graf.Evaluate(Aceleration), 0, 0);
                body.AddForce(FrontMoover.position - body.transform.position);
                ShumRun.pitch = Mathf.Lerp(ShumRun.pitch, Aceleration,
                  StepSountAceleration);
            }
            else if (MyState == State.speake || MyState == State.lissen)
            {
                Rotater(brother.transform);
                ZeroShum();
                if (timeLim++ >= timeLissenSpeake)
                {
                    MyState = State.run;
                    brother = null;
                    timeLim = 0;
                }
            }
        }

        if (body.velocity.magnitude < 0.1f)
            return;
        if (Vector3.Angle(FrontMoover.position - body.position, body.velocity) >
90)
            Glass.localRotation = Quaternion.AngleAxis(
      Glass.localRotation.eulerAngles.z + body.velocity.magnitude, Vec-
tor3.forward);
        else
            Glass.localRotation = Quaternion.AngleAxis(
      Glass.localRotation.eulerAngles.z - body.velocity.magnitude, Vec-
tor3.forward);
      }

    void Rotater(Transform from)
    {
        Vector3 tmpV = from.position;
        tmpV.y = transform.position.y;
        tmpV -= transform.position;
        transform.rotation = Quaternion.LerpUnclamped(transform.rotation,
            Quaternion.LookRotation(tmpV), Time.deltaTime * speed_raotate);
    }

    private void ZeroShum()
    {
        ShumRun.pitch = Mathf.Lerp(ShumRun.pitch, 0, StepSountAceleration * 2);
    }

    void OnCollisionEnter(Collision col)
    {
        if (col.gameObject.tag == "Player" || col.gameObject.tag == "Globe")
            SpeakeRand(ForLight, 0.7f);
```

```
    }

    public void LocalColliderEnter(int i)
    {
        if (i == i_point_move)
        {
            Aceleration = 0;
            MyStateRun = StateRun.start;
            i_point_move = i_point_move >= pointMove.Length - 1 ? 0 :
i_point_move+1;
            SpeakeRand(GlobeBla[Random.Range(0, GlobeBla.Length)], 0.5f);
        }
    }

    void OnTriggerEnter(Collider col)
    {
        if (MyState != State.empty && col.tag == "Globe" &&
            timeLim >= timeLissenSpeake &&
            brother == null && col.GetComponent<GlobeControll>().brother == null
&&
            col.GetComponent<GlobeControll>().MyState != State.empty)
        {
            SetState = State.speake;
            brother = col.GetComponent<GlobeControll>();
            col.GetComponent<GlobeControll>().brother = this;
            col.GetComponent<GlobeControll>().SetState = State.lissen;
        }
        else if (col.tag == "Player")
            SpeakeRand(Hello[Random.Range(0, Hello.Length)], 0.5f);
    }

    public void SpeakeRand(AudioClip clip, float P)
    {
        if (!BlaBla.isPlaying && Random.Range(0f, 1f) > P)
        {
            BlaBla.clip = clip;
            BlaBla.Play();
        }
    }

    public State SetState
    {
        set
        {
            MyState = value;
            timeLim = 0;
            if (value == State.speake && !BlaBla.isPlaying)
            {
                BlaBla.clip = Hello[Random.Range(0, Hello.Length)];
                BlaBla.Play();
            }
        }
    }

    public void getPlayerControl(bool oper, Transform Phy)
    {
        PlayerControlRotation = Quaternion.Euler(0,
            transform.localRotation.eulerAngles.y, 0);
        PlayerControlled = oper;
        MyState = !oper && pointMove.Length > 0 ? State.run : State.empty;
        ZeroCamPosition.transform.parent = oper ? null : transform;

        if (oper)
```

```
        {
            Transform PhyPosition = ZeroCamPosition.GetChild(0);
            Phy.parent = PhyPosition;
            Phy.localPosition = Vector3.zero;
            Phy.localRotation = new Quaternion(0, 0, 0, 0);
        }
    }

#if UNITY_EDITOR
    void OnDrawGizmosSelected()
    {
        GizmosAdd(true);
    }

    void OnDrawGizmos()
    {
        GizmosAdd(false);
    }

    void GizmosAdd(bool isSelect)
    {
        if (pointMove.Length <= 0)
            return;

        Gizmos.color = isSelect ? Color.red : Color.green;
        int m_1 = pointMove.Length - 1;

        for (int q = m_1; q < pointMove.Length * 2; q++)
        {
            Gizmos.DrawLine(pointMove[q % pointMove.Length].position,
                  pointMove[(q + 1) % pointMove.Length].position);
        }
    }

    [ContextMenu("AddPoint")]
    void AddPoint()
    {
        Transform[] tmp = new Transform[pointMove.Length];
        pointMove.CopyTo(tmp, 0);
        pointMove = new Transform[pointMove.Length + 1];
        tmp.CopyTo(pointMove, 0);

        GameObject GO = Instantiate(ColliderPerfab);
        GO.GetComponent<ColliderChecker>().TagDetect = this.name;
        GO.GetComponent<ColliderChecker>().MyIndex = pointMove.Length-1;
        GO.transform.position = this.transform.position;
        GO.name = this.name + "_p" + tmp.Length;
        pointMove.SetValue(GO.transform, pointMove.Length - 1);
    }

    [ContextMenu("CLEAR")]
    void ClearPoint()
    {
        for (int q = 0; q < pointMove.Length;  q++)
        {
            if (pointMove[q] != null)
                DestroyImmediate(pointMove[q].gameObject);
        }
        pointMove = new Transform[0];
    }
#endif
}
```

Файл «Booms.cs»

Определяет столкновение указанного объекта и воспроизводит определённый звук удара в зависимости от материала и силы столкновения.

```
using UnityEngine;
using System.Collections;

[AddComponentMenu("MyScripts/Objects/Booms")]
[RequireComponent(typeof(AudioSource))]
public class Booms : MonoBehaviour
{
    public AudioClip[] BoomSound;
    AudioSource _aus;
    Rigidbody _body;

    [Range (0.5f, 200f)]
    public float MinVelocy = 5f;

    void Awake ()
    {
        _aus = gameObject.GetComponent<AudioSource>();
        _body = GetComponent<Rigidbody>();
    }

    void OnCollisionEnter(Collision obj)
    {
        Rigidbody objCol = obj.gameObject.GetComponent<Rigidbody>();
        if (_body.velocity.magnitude > MinVelocy
            || objCol != null && objCol.velocity.magnitude > MinVelocy)
        {
            _aus.PlayOneShot(BoomSound[Random.Range(0, BoomSound.Length - 1)]);
        }
    }
}
```

Файл «PersonControl.cs»

Управляет персоналом научной лаборатории, запуская и переключая анимации. Имеет возможность из редактора указать стартовую анимацию.

```
using UnityEngine;
using System.Collections;

[AddComponentMenu("MyScripts/Person/PersonControl")]
public class PersonControl : MonoBehaviour
{
    public enum PersMode { SeatAndRead = 0, SeatAdnWorkPC = 1,
            StateAndWorkPC = 2, Seat = 3, StartPersonalAni = 4};
    public PersMode mode = PersMode.SeatAndRead;
    public Animator aniPers;

    void Awake ()
    {
        aniPers.SetInteger("select", (int)mode);
    }
}
```

Файл «addscore.php»

```
<?php
	$db = mysqli_connect('mysql.hostinger.ru', 'u295677728_plapp', 'пароль')
		or die(mysqli_error($db));
	mysqli_select_db($db, 'u295677728_phla') or die(mysqli_error($db));

	$id = $_GET['id'];
	$nickname = $_GET['nickname'];
	$score = $_GET['score'];
	$hash = $_GET['hash'];
	$secretKey = "PhysiLabsAPP";
	$real_hash = md5($id . $nickname . $score . $secretKey);
	if($real_hash == $hash) {
		$query = "SELECT nickname FROM `scores` WHERE id = '$id'
			AND nickname = '$nickname';";
		$result = mysqli_query($db, $query) or die(mysqli_error($db));
		$nums = mysqli_num_rows($result);

		if($nums == 1){
			$query = "UPDATE scores SET score = score + '$score'
				WHERE id = '$id';";
			$result = mysqli_query($db, $query) or die(mysqli_error($db));
			echo "DataUpdate";
		}
		else
			echo "ProfileError";
	}
?>
```

Файл «setdataprof.php»

```
<?php
	$db = mysqli_connect('mysql.hostinger.ru', 'u295677728_plapp', 'пароль')
		or die(mysqli_error($db));
	mysqli_select_db($db, 'u295677728_phla') or die(mysqli_error($db));

	$id = $_GET['id'];
	$nickname = $_GET['nickname'];
	$name = $_GET['name'];
	$surname = $_GET['surname'];
	$doinfo = $_GET['doinfo'];
	$hash = $_GET['hash'];
	$secretKey = "PhysiLabsAPP";
	$real_hash = md5($id . $nickname . $name . $surname . $doinfo . $secretKey);

	if($real_hash == $hash){
		$query = "SELECT nickname FROM `scores`
			WHERE id = '$id' AND nickname = '$nickname';";
		$result = mysqli_query($db, $query) or die(mysqli_error($db));
		$nums = mysqli_num_rows($result);

		if($nums == 1){
			$query = "UPDATE scores
			SET  name = '$name',  surname = '$surname', doinfo = '$doinfo'
				WHERE id = '$id';";
			$result = mysqli_query($db, $query) or die(mysqli_error($db));
			echo "DataUpdate";
		}
```

```
            else
                  echo "ProfileError";
      }
?>
```

Файл «display.php»

```
<?php
      $db = mysqli_connect('mysql.hostinger.ru', 'u295677728_plapp', 'пароль')
            or die(mysqli_error($db));
      mysqli_select_db($db, 'u295677728_phla') or die(mysqli_error($db));

    $query = "SELECT nickname, score FROM `scores` ORDER by `score` DESC LIMIT
20";
    $result = mysqli_query($db, $query) or die(mysqli_error($db));

    $num_results = mysqli_num_rows($result);
    for($i = 0; $i < $num_results; $i++){
         $row = mysqli_fetch_array($result);
         echo $row['nickname'] . "\t" . $row['score'] . "\n";
    }
?>
```

Файл «regestration.php»

```
<?php
      $db = mysqli_connect('mysql.hostinger.ru', 'u295677728_plapp', 'пароль') or
            die(mysqli_error($db));
      mysqli_select_db($db, 'u295677728_phla') or die(mysqli_error($db));
      $nickname = $_GET['nickname'];
      $name = $_GET['name'];
      $hash = $_GET['hash'];
      $secretKey="PhysiLabsAPP";
      $real_hash = md5($name . $nickname . $secretKey);

      if($real_hash == $hash)
      {
            $query = "SELECT nickname FROM `scores` WHERE nickname = '$nickname'";
            $result = mysqli_query($db, $query) or die(mysqli_error($db));
            $nums = mysqli_num_rows($result);

            if($nums > 0)
                  echo "RebreackNickname";
            else
            {
                  try{
                        $query = "SELECT COUNT(id) FROM `scores`";
                        $result = mysqli_query($db, $query)
                              or die(mysqli_error($db));
                        $row = mysqli_fetch_row($result);
                        $prefix = "id_" . $row[0] . "_";
                        $id_generate = uniqid($prefix);
                        $query = "INSERT INTO scores
                              VALUES('$id_generate', '$nickname', '$name', 0);";
                        mysqli_query($db, $query) or die(mysqli_error($db));
                        echo $id_generate;
                  }
                  catch (Exception $e) {echo "RegError";}
            }
      }
?>
```

Файл «WWWdisplay.php»

```
<style>
	table {
		width: 100%;
		height: 100%;
		color: white;
		border-spacing: 3px;
		font-size: 12pt;
	}
	td, th{
		padding: 5px;
		background: black;
		opacity: 0.6;
		filter: alpha(Opacity=60);
	}
</style>
<table border="2">
	<col width="20%">
	<col width="50%">
	<col width="30%">
	<tr bgcolor = "gray">
		<td align="center"> Позиция </td>
		<td align="center"> Никнейм </td>
		<td align="center"> Очки </td>
	</tr>
	<?php
		$db = mysqli_connect('mysql.hostinger.ru', 'u295677728_plapp',
	'пароль')
			or die(mysqli_error($db));
		mysqli_select_db($db, 'u295677728_phla') or die(mysqli_error($db));

		$query = "SELECT nickname, score FROM `scores`
			ORDER by `score` DESC LIMIT 20";
		$result = mysqli_query($db, $query) or die(mysqli_error($db));
		$num_results = mysqli_num_rows($result);

		for($i = 0; $i < $num_results; $i++){
			echo "<tr> <td align=\"center\">" . ($i+1) . "</td>";
			$row = mysqli_fetch_array($result);
			echo "<td align=\"center\">" . $row['nickname'] . "</td>
				<td align=\"center\">" . $row['score'] . "</td>";
			echo "</tr>";
		}
	?>
</table>
```