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

ДОПУСТИТЬ К ЗАЩИТЕ

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

_____ Л.Б. Соколинский

«___»_____ 2024 г.

Разработка компьютерной игры «Дрифт» на платформе Unity

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 02.03.02.2024.308-306.ВКР

Научный руководитель,
ст. преподаватель кафедры СП
_____ П.Г. Верман

Автор работы,
студент группы КЭ-401
_____ А.Р. Павленко

Ученый секретарь
(нормоконтролер)
_____ И.Д. Володченко
«___»_____ 2024 г.

Челябинск, 2024 г.

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

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
29.01.2024 г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы бакалавра
студенту группы КЭ-401
Павленко Александру Романовичу,
обучающемуся по направлению
02.03.02 «Фундаментальная информатика и информационные технологии»

1. **Тема работы** (утверждена приказом ректора от 22.04.2024 г. № 764-13/12)
Разработка компьютерной игры «Дрифт» на платформе Unity.
2. **Срок сдачи студентом законченной работы:** 03.06.2024 г.
3. **Исходные данные к работе**
3.1. Роллингз Э., Моррис Д. Проектирование и архитектура игр. Вильямс, 2006. – 1035 с.
3.2. Хоккинг Д. Unity – в действии. Мультиплатформенная разработка на C# – СПб: Питер, 2016. – 336 с.
3.3. Unity User Manual. [Электронный ресурс] URL: https://docs.unity3d.com/Manual/index.html (дата обращения: 29.01.2024 г.).
4. **Перечень подлежащих разработке вопросов**
4.1. Выполнить анализ предметной области.
4.2. Спроектировать игровое приложение.
4.3. Реализовать игровое приложение.
4.4. Провести тестирование приложения.
5. **Дата выдачи задания:** 29.01.2024 г.

Научный руководитель,
ст. преподаватель кафедры СП П.Г. Верман

Задание принял к исполнению А.Р. Павленко

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность

Согласно исследованию компании Newzoo [1] гоночные игры пользуются большой популярностью среди игроков: почти четверть всех любителей игр называют их одним из своих любимых жанров. 30% консольных игроков называют гоночные игры своим любимым жанром, на мобильных платформах и персональных компьютерах этот показатель составляет 23% и 21% соответственно. При этом возраст 36% любителей гоночных игр имеют высокий доход, что намного выше, чем в среднем по всем игрокам. В исследовании подчеркивается, что любители гоночных игр являются уникальной категорией игроков, ведь они одновременно моложе других игроков и зарабатывают в среднем больше других, что делает рынок гоночных игр привлекательным для разработчиков компьютерных игр.

В современных компьютерных гоночных играх, а также в автосимуляторах дрифт является неотъемлемой частью игрового процесса, выступая в качестве самостоятельного режима. С учетом динамического развития интереса к гоночным симуляторам, включающим в себя элементы дрифта, все больше игроков ищут уникальный игровой опыт, где дрифт является не просто дополнительным режимом, а основой игрового процесса.

Постановка задачи

Целью выпускной квалификационной работы является разработка компьютерной игры «Дрифт» на платформе Unity. Для достижения поставленной цели необходимо решить следующие задачи:

1) выполнить анализ предметной области;
2) спроектировать игровое приложение;
3) реализовать игровое приложение;
4) провести тестирование игрового приложения.

Структура и содержание работы

Работа состоит из введения, четырех глав, заключения и списка литературы. Объем работы составляет 78 страниц, объем списка литературы – 28 источников.

В первой главе «Анализ предметной области» описывается предметная область проекта, приводится описание игровых жанров, подробно сравниваются аналоги и описываются существующие игровые механики.

Во второй главе «Проектирование системы» приводятся функциональные и нефункциональные требования к разрабатываемому игровому приложению, концепция игрового приложения, описываются варианты использования системы, приведены макеты игрового интерфейса и описаны другие подробности проектирования.

Третья глава «Реализация игрового приложения» посвящена подробностям реализации игрового приложения на платформе Unity.

В четвертой главе «Тестирование» представлено тестирование игрового приложения.

В заключении приводятся основные результаты работы и направление дальнейшей работы.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Описание предметной области

Дрифт представляет собой технику вождения под большим углом управляемого заноса без потери контроля над автомобилем, то есть в условиях устойчивого бокового заноса [2]. Занос представляет собой боковое скольжение задних колес при продолжающемся поступательном движении автомобиля [3]. При этом передние колеса во время управляемого заноса повернуты в противоположную повороту сторону.

Также дрифт – это вид автоспорта, смысл которого заключается в демонстрации наивысшего контроля над автомобилем и мастерства пилотирования. Соревнования проводятся на сухом асфальте небольшого участка трассы с большим количеством соединяющихся поворотов.

Помимо мастерства пилота, результат заезда зависит от технических характеристик автомобиля. Требуется заднеприводный автомобиль с большой мощностью двигателя. Также практически всегда автомобили облегчают и подвергают техническим улучшениям.

Оценка заезда водителя осуществляется по следующим критериям [4]:

1) траектория – существуют специально обозначенные точки, проезжая рядом с которыми, судьи начисляют больше очков, чем обычно;
2) угол прохождения поворота – чем больше угол, тем выше оценка;
3) скорость – чем выше скорость, чем выше оценка;
4) зрелищность – оценивается судьями по опасным маневрам, таким как легкое касание стены, количеству дыма покрышек и другим показателям.

Существуют два типа заездов: одиночные и парные. В одиночных заездах судьи начисляют водителю определенное количество очков. Количество очков не зависит от времени, за которое пилот преодолел трассу.

Судейство парных заездов осуществляется по этим же критериям, но участники оцениваются в паре. Парный заезд состоит из двух заездов.

Сначала первый участник лидирует, а второй должен проехать трассу на максимально возможном приближении к ведущему пилоту с углом заноса не меньшим, чем у ведущего пилота. Затем участники меняются местами. Победителем считается водитель, который наберет большее количество баллов.

1.2. Обзор жанра и специфики

Компьютерные игры про дрифт относятся к категории «спортивного автосимулятора» с элементами «аркады». Часто игровые механики, связанные с дрифтом, или режимы, основанные на данной механике, являются частью гоночных игр. Именно из режимов с данной механикой выделились компьютерные игры, в которых упор делался только на данный режим с различными соревновательными особенностями.

Жанр «спортивный симулятор» – это жанр компьютерных игр, который имитирует занятие спортом, в том числе автоспортом [5]. Спортивный симулятор может быть ориентирован на быстрый игровой процесс, при этом физическая модель мира может быть упрощена. Главной целью «спортивных симуляторов» является получение наибольшего количества очков, соревнуясь с настоящими или компьютерными соперниками. В то же время существуют игры, которые максимально точно воссоздают физику поведения и управления автомобиля. В таких играх можно сломать и разбить транспортное средство. Также спортивные симуляторы имеют возможность более точной и аккуратной настройки характеристик автомобиля.

Впервые режим дрифта начал появляться в гоночных играх, в которых дрифт представлял собой самостоятельную механику. Жанр «Гонки» [6] – это жанр компьютерных игр, в которых игрок принимает участие в соревнованиях на скоростном транспорте. К такому транспорту могут относиться не только автомобили, но и другие наземные, водные, воздушные и космические транспортные средства. Основная идея гоночных игр

заключается в соревновании с настоящими или компьютерными оппонентами в скорости преодоления расстояния или мастерства управления.

При этом часто под термином «Гоночные игры» подразумевают «Аркадные гоночные» [7] игры. В общем случае «аркада» [8] представляет собой жанр компьютерных игр, в которых игровой процесс сильно упрощен, становясь примитивным и нереалистичным, с которым любой игрок может разобраться в максимально короткие сроки.

Особенностью же аркадных гонок является упрощенная физическая модель. Таким образом, физика автомобиля сильно упрощается в угоду веселья игрока. В большинстве аркадных гоночных игр не учитываются параметры, необходимые для точной симуляции поведения транспортного средства. Также упрощается физика разрушения автомобилей и окружающих объектов. То есть игрок не может разбиться или пострадать, транспортное средство невозможно сломать и не оно не может получить серьезные повреждения. Чаще всего это связано с лицензированием настоящих автомобилей, так как автопроизводители накладывают определенные ограничения на разработчиков гоночных игр. Автопроизводители хотят, чтобы их автомобиль всегда выглядел безопасно.

Также аркадные гоночные игры характеризуются упрощенной системой улучшения технических характеристик автомобиля, при этом возможности авто становятся фантастическими и далекими от реальных возможностей настоящих автомобилей. Такие гоночные игры пользуются наибольшим спросом и задевают максимально большую аудиторию.

1.3. Обзор аналогов и существующих игровых механик

Для сравнительного обзора были выбраны проекты, игровой процесс которых построен вокруг дрифта, и в которых дрифт представляет собой отдельный игровой режим. Следовательно, дрифт в играх может быть основой игрового процесса, а может быть самостоятельным режимом в гоночной игре.

CarX Drift Racing Online

«CarX Drift Racing Online» – компьютерная игра в жанре спортивный автосимулятор, которая была выпущена в 2017 году [9]. В Steam игра заработала 15,7 миллионов долларов США и получила 95% положительных отзывов [10].

На рисунке 1 представлен скриншот игрового процесса «CarX Drift Racing Online».

Рисунок 1 – Игра «CarX Drift Racing Online»

Дрифтинг является основой игрового процесса «CarXDrift Racing Online». Главная задача игрока – набрать максимально возможное количество очков за заезд. Очки напрямую конвертируются в игровую валюту, то есть за каждое очко дрифта игрок получает один игровой доллар. Количество набирающихся очков зависит от скорости движения игрока и угла его заноса. Очки умножаются на специальный множитель, который зависит от количества времени, которое игрок провел в управляемом заносе. Также игрок может получить дополнительные очки во время дрифта, если начнет управляемый занос на большой скорости, будет долго удерживать боль-

шой угол заноса, без касания стен проедет большое расстояние, будет менять направление заноса.

Заработанную игровую валюту игрок может тратить на покупку новых автомобилей, усовершенствование имеющихся автомобилей, открытие новых трасс.

Torque Drift

«Torque Drift» [11] – компьютерная игра про дрифт, выпущенная в 2018 году на мобильные платформы, а в 2021 выпущенная на ПК.

На рисунке 2 представлен скриншот игрового процесса «Torque Drift».

Рисунок 2 – Скриншот игры «Torque Drift»

Основным режимом игры является фристайл (от англ. freestyle – свободный стиль), в котором игрок соревнуется с тремя соперниками, а победителем становится тот, кто наберет наибольшее количество очков. Количество очков зависит от скорости движения игрока, угла наклона автомобиля во время заноса и расстояния до объектов во время дрифта. Набирающиеся очки могут быть умножены на множитель, который можно накопить, проезжая в заносе через специальные зоны. Стандартные зоны уве-

личивают количество очков на определенное количество бонусных очков, и увеличивают множитель. Золотые зоны заноса увеличат общее число очков и принесут в 5 раз больше бонусных очков. Чем больше зон заноса игрок преодолеет за один занос, тем выше будет множитель очков.

За победу в заезде игрок получает различную награду. В качестве награды могут выступать игровая валюта, детали для автомобиля и даже сам автомобиль. Игровую валюту можно потратить на покупку новых автомобилей и на покупку запчастей к ним.

Серия игр Need for Speed

«Need for Speed» – популярная серия компьютерных гоночных игр, издателем которой является Electronic Arts. Первая игра серии «The Need for Speed» была выпущена в 1994 году. С тех пор было продано более 150 миллионов копий [12] игр серии, что делает «Need for Speed» одной из самых популярных и влиятельных гоночных игр в мире.

Дрифт в серии игр «Need for Speed» представляет собой самостоятельный тип заездов, смысл которого заключается в том, что игроку необходимо набрать как можно больше очков, находясь в управляемом заносе. При этом, чем дольше игрок находится в управляемом заносе, и чем больше его скорость, тем больше очков он получит. Также от того, насколько долго игрок находится в заносе, зависит множитель очков, на который умножаются получаемые игроком очки. При столкновении со стеной набранные очки обнуляются. Для победы в заезде игроку необходимо набрать больше очков, чем наберут соперники.

Победа в состязании награждается определенным количеством игровой валюты. Поражением считается любое место, кроме первого. Игровую валюту игрок может потратить на покупку новых автомобилей в автосалоне, техническую и визуальную модификацию своих автомобилей.

Первой игрой серии, в которой появляется режим дрифтинга, является «Need for Speed: Underground». Проект ждал большой коммерческий успех. В первую неделю декабря 2003 года «Underground» занимал первое

место в рейтинге самых продаваемых игр для консолей [13], а по состоянию на июнь 2015 года суммарные продажи игры составляют 15 миллионов экземпляров [14], что делает данную часть серии одной из самых популярных среди игроков.

На рисунке 3 представлен скриншот игрового процесса «Need for Speed: Underground».

Рисунок 3 – Скриншот игры «Need for Speed: Underground»

Особенностью дрифта в данной части серии является то, что игрок может получать больше очков, проезжая по призовым зонам, которые находятся у стен. При этом количество набирающихся очков также зависит от угла заноса. Сам угол может находиться в пределах от 0 до 90 градусов. Текущий угол заноса отображен на пользовательском интерфейсе в виде числового значения, что помогает игроку лучше контролировать угол во время заноса и набирать больше очков.

Далее дрифт появляется практически в каждой игре серии «Need for Speed» и является одним из самых любимых типов заездов у игроков. Серьезные изменения дрифт претерпел в «Need for Speed: Carbon». Данная

часть серии была выпущена в 2006 году, и по состоянию на 2019 год была продана тиражом 15,6 миллионов копий [15].

На рисунке 4 представлен скриншот игрового процесса «Need for Speed: Carbon».

Рисунок 4 – Скриншот игры «Need for Speed: Carbon»

Нововведением стало появление скоростного бонуса, который влиял на темп набирания очков игроком. Множитель очков при этом теперь не зависел от времени, проведенном в заносе. Множитель очков каждый раз увеличивался при смене направления движения в управляемом заносе игроком. К тому же разработчики убрали возможность отслеживать угол дрифта. Также в «Carbon» присутствовала система нитро-ускорения, которая помогала быстрее преодолеть прямые участки трассы и сохранить бонусы, накопленные игроком.

Совершенно новый режим, построенный вокруг дрифта, появился в последней «Need for Speed: Unbound» [16]. Unbound была выпущена в

2022 году. Только за первый месяц продаж в Steam игра заработала 14 миллионов долларов США [17].

На рисунке 5 представлен скриншот игрового процесса «Need for Speed: Unbound».

Рисунок 5 – Игра «Need for Speed: Unbound»

Takeover – тип соревнования, в котором игроку начисляются очки в зависимости от выполненной цепочки навыков. Самым основным навыком является дрифт, за который можно получить больше всего очков. Очки также можно получить за использование нитро-ускорения, прыжков, прохождение контрольных точек, разрушение специальных объектов. Но, помимо набора очков, игрок может потерять их, если будет разрушать и сбивать специально помеченные объекты или ударяться о стены. Множитель очков в этот раз зависит от того, насколько быстро игрок сможет выполнять различные цепочки трюков.

Также ключевым отличием «Unbound» от предыдущих частей серии является то, что награду в виде игровой валюты игрок получит даже в том случае, если не займет первого места. Награда зависит от места игрока в рейтинге заезда, то есть чем выше позиция игрока, тем больше будет его

награда. Накопленную валюту можно потратить на покупку новых автомобилей, так и на доработку уже купленных автомобилей.

1.4. Сравнение

Ключевыми механиками, выделенными в процессе обзора аналогов, являются: набор очков, множитель очков, призовые зоны, влияние скорости, угла движения, близости к стенам и ограждениям на темп набора очков, возможность получать дополнительные очки за выполнение трюков или цепочку трюков, покупка и продажа автомобилей, техническое усовершенствование и визуальное изменение автомобилей. Перечисленные механики и присутствие их в различных играх представлены в таблице 1.

Таблица 1 – Сравнение аналогов

Критерий	**CarX Drift Online**	**Torque Drift**	**NFS: Underground**	**NFS: Carbon**	**NFS: Unbound**
Цель – набор очков	Да	Да	Да	Да	Да
Наличие множителя очков	Да	Да	Да	Да	Да
Наличие призовых зон	Нет	Нет	Да	Да	Нет
Скорость влияет на набор очков	Да	Да	Да	Да	Да
Угол влияет на набор очков	Да	Да	Да	Да	Да
Игрок видит угол заноса	Да	Нет	Да	Нет	Нет
Расстояние до стен влияет на набор очков	Да	Да	Нет	Нет	Нет
Можно получить дополнительные очки	Да	Да	Нет	Нет	Да
Можно покупать и продавать автомобили	Да	Да	Да	Да	Да
Можно улучшать технические характеристики автомобилей	Да	Да	Да	Да	Да
Можно менять внешний облик автомобилей	Да	Да	Да	Да	Да

Основной механикой всех игр про дрифт является набор максимально возможного количества очков. При этом количество очков может зависеть от нескольких параметров. В большинстве проектов такими параметрами являются скорость и угол движения автомобиля. Но, помимо этого,

также в качестве параметра, влияющего на набор очков, может быть расстояние до стен и оград, как в компьютерной игре «Torque Drift».

Немаловажной игровой механикой во всех играх про дрифт является множитель очков. Множитель очков может зависеть как от времени, проведенного в заносе, так и от выполненной цепочки маневров, таких как смена направления заноса. В «Torque Drift» множитель увеличивается каждый раз, когда игрок проезжает по специальным зонам.

Также популярной игровой механикой является возможность набрать дополнительные очки за выполнение различных трюков. К таким трюкам могут относиться: высокая скорость движения, начало заноса на высокой скорости, удержание большого угла заноса, аккуратное вождение без касаний стен, но легкое касание стены автомобилем может служить в качестве одного из бонусов. В «Need for Speed: Unbound» дополнительные очки можно не только заработать, но и потерять, если сбивать объекты, за которые полагается штраф.

За участие и победу в компьютерных играх про дрифт игроку начисляется награда. Победой в заезде в «Torque Drift», «Need for Speed: Underground», «Need for Speed: Carbon» засчитывается первое место в итоговом зачете по количеству набранных очков среди всех соперников. За победу Награда может быть представлена в виде игровой валюты. Также в качестве награды игрок может получить автомобиль или запчасти для него, как в игре «Torque Drift». Награда в «CarX Drift Online» зависит от количества набранных очков за заезд, так как все очки напрямую конвертируются во игровую валюту. Награда в виде игровой валюты в «Need for Speed: Unbound» зависит от места игрока в итоговом зачете. Максимальную награду игрок получит за первое место, наименьшую за последнее место.

Существенной игровой механикой во всех играх про дрифт является тюнинг и стайлинг, то есть техническое и визуальное улучшение автомобиля игрока. Техническое улучшение автомобиля влияет на скорость и угол, при котором игрок не теряет контроль во время заноса. Таким обра-

зом, чем больше установлено улучшений на автомобиль игрока, тем лучше результаты его заездов.

Таким образом, процесс сравнения дал возможность подчеркнуть общие черты игр про дрифт.

Во-первых, основной целью игрока должен быть набор максимально возможного количества очков для победы в заезде.

Во-вторых, на набор очков должны влиять скорость и угол движения автомобиля игрока.

В-третьих, для разнообразия игрового процесса необходимо предоставить игроку возможность получать дополнительные очки.

Также немаловажным аспектом игры должны стать тюнинг и стайлинг, то есть технические и визуальные улучшения автомобилей. Помимо прочего игроку должна быть предоставлена возможность покупать новые и продавать свои старые автомобили.

Вывод по первой главе

В ходе анализа предметной области было описано, что представляет собой дрифт, был выполнен обзор жанров и специфики игр про дрифт, проведен обзор существующих решений и игровых механик на основе таких компьютерных игр, как Torque Drift, CarX Drift Online, серии игр Need for Speed. На основе обзора были выявлены основные моменты, которые необходимо учесть при разработке игрового приложения. Процесс сравнения дал возможность подчеркнуть общие черты игр про дрифт.

2. ПРОЕКТИРОВАНИЕ СИСТЕМЫ

В данной главе содержатся функциональные и нефункциональные требования, описывается концепция игрового приложения, приводятся диаграмма вариантов использования, диаграмма компонентов и макеты пользовательского интерфейса.

2.1. Функциональные и нефункциональные требования

Функциональные требования

Игровое приложение должно соответствовать следующим функциональным требованиям:

1) игроку должна быть предоставлена возможность выбирать заезд;

2) игроку должна быть предоставлена возможность смены автомобиля;

3) игроку должна быть предоставлена возможность продажи свои автомобили;

4) игроку должна быть предоставлена возможность модифицировать выбранный автомобиль, как технически, так и визуально;

5) игроку должна быть предоставлена возможность покупать новые автомобили;

6) игроку должна быть предоставлена возможность изменять настройки игры;

7) система должна сохранять прогресс игрока и настройки игры в файл и загружать данные из файлов во время запуска игрового приложения.

Нефункциональные требования

Игровое приложение должно соответствовать следующим нефункциональным требованиям:

1) игровое приложение должно быть написано на языке программирования C#;

2) игровое приложение должно быть разработано при помощи платформы Unity.

2.2. Концепция

«No Limits» – трехмерная компьютерная игра с видом от третьего лица в жанре «спортивный автосимулятор». Игроку предстоит участвовать в заездах на автомобиле, используя управляемый занос, то есть дрифт. Цель заездов состоит в том, что игроку необходимо набрать определенное количество очков, чтобы получить награду. Награду игрок может потратить на покупку новых автомобилей, усовершенствование технических характеристик текущего выбранного автомобиля, изменение внешнего вида текущего выбранного автомобиля. Также игроку предоставляется возможность продавать свои автомобили.

Цели игры

Целью игры является прохождение всех трасс игроком на максимально возможное количество очков. Для каждой трассы установлено определенное количество очков, которые необходимо набрать, чтобы пройти заезд на оценку «золото», «серебро», «бронза». Для того чтобы пройти трассу, игроку необходимо набрать достаточное для оценки «бронза» количество очков.

Начало игры

Перед началом игры игрок должен выбрать автомобиль, на котором желает принять участие в заезде, и трассу, которую хочет пройти.

Игровой процесс

Основной игровой механикой является дрифт. При введении автомобиля в занос игрок получает очки. Количество получаемых очков игроком зависит от скорости и угла скольжения. Чем дольше автомобиль игрока находится в заносе, тем больше очков может набрать игрок, так как в момент скольжения они умножаются на множитель заноса. Если угол заноса или скорость малы для того, чтобы засчитывать новые очки, то через опре-

деленное время они суммируются в общий результат заезда. После суммирования очков в общий результат набор очков начинается с 0. При столкновении игрока со статическими объектами на трассе, такими как стены, набранные очки обнуляются.

Коэффициент множителя заноса зависит от того, насколько долго игрок находится в управляемом заносе, на какой скорости движется, и какой в данный момент времени угол скольжения автомобиля. Диапазон множителя заноса может находиться в пределах от 1 до 4. Чем больше скорость движения или угол скольжения, тем быстрее увеличивается коэффициент множителя заноса. При снижении скорости движения и уменьшении угла скольжения пропорционально сокращается коэффициент множителя заноса. Коэффициент множителя заноса обнуляется при столкновении со статическими объектами.

Игрок может получить бонусные очки. Для этого ему необходимо сбивать специально помеченные объекты, такие как зеленые конусы. Бонусные очки суммируются с набирающимися очками. Благодаря сбору бонусных очков набор набирающихся очков продолжается, даже если скорость движения или угол скольжения автомобиля слишком малы.

Также игрок может получить штрафные очки. Штрафные очки начисляются, если игрок собьет специально помеченные объекты, такие как красные конусы. Штрафные очки вычитаются из набирающихся очков.

Игровая валюта

В зависимости от того, на какую оценку прошел игрок трассу, он получает определенное количество игровой валюты. Игровая валюта может быть использована для технического улучшения автомобиля, изменения его внешнего вида, покупки новых автомобилей.

Тюнинг и стайлинг

Тюнинг представляет собой техническое усовершенствование автомобиля путем модификации его составляющих. Игроку предоставляется возможность модифицировать двигатель и ходовую часть автомобиля.

Модификация двигателя влияет на ускорение и скорость автомобиля. Модификация ходовой части влияет на управляемость автомобиля.

Каждая модификация представлена в 3 вариантах: базовая, улучшенная, профессиональная. Каждая следующая модификация пропорционально улучшает характеристики автомобиля. При этом для покупки каждой следующей модификации требуется большее количество игровой валюты в соответствии с вариантом улучшения.

Стайлинг представляет собой создание индивидуального облика автомобиля путем изменения его внешнего вида. Для этого игрок может изменять цвет кузова автомобиля, менять колесные диски, менять цвет колесных дисков.

Покупка автомобилей

Игрок может приобретать новые автомобили в автосалоне. Покупка каждого автомобиля требует разного количества игровой валюты. Стоимость автомобиля зависит от его исходных характеристик.

Гараж

Купленные автомобили, а также стартовый автомобиль игрока, хранятся в его гараже. Игрок может по своему желанию менять используемый автомобиль на один из доступных в гараже.

Продажа автомобилей

Игрок может продавать свои автомобили в гараже. Цена продажи автомобиля зависит от первоначальной стоимости автомобиля и установленного уровня модификации на автомобиле.

2.3. Варианты использования системы

На рисунке 6 представлена диаграмма вариантов использования игрового приложения.

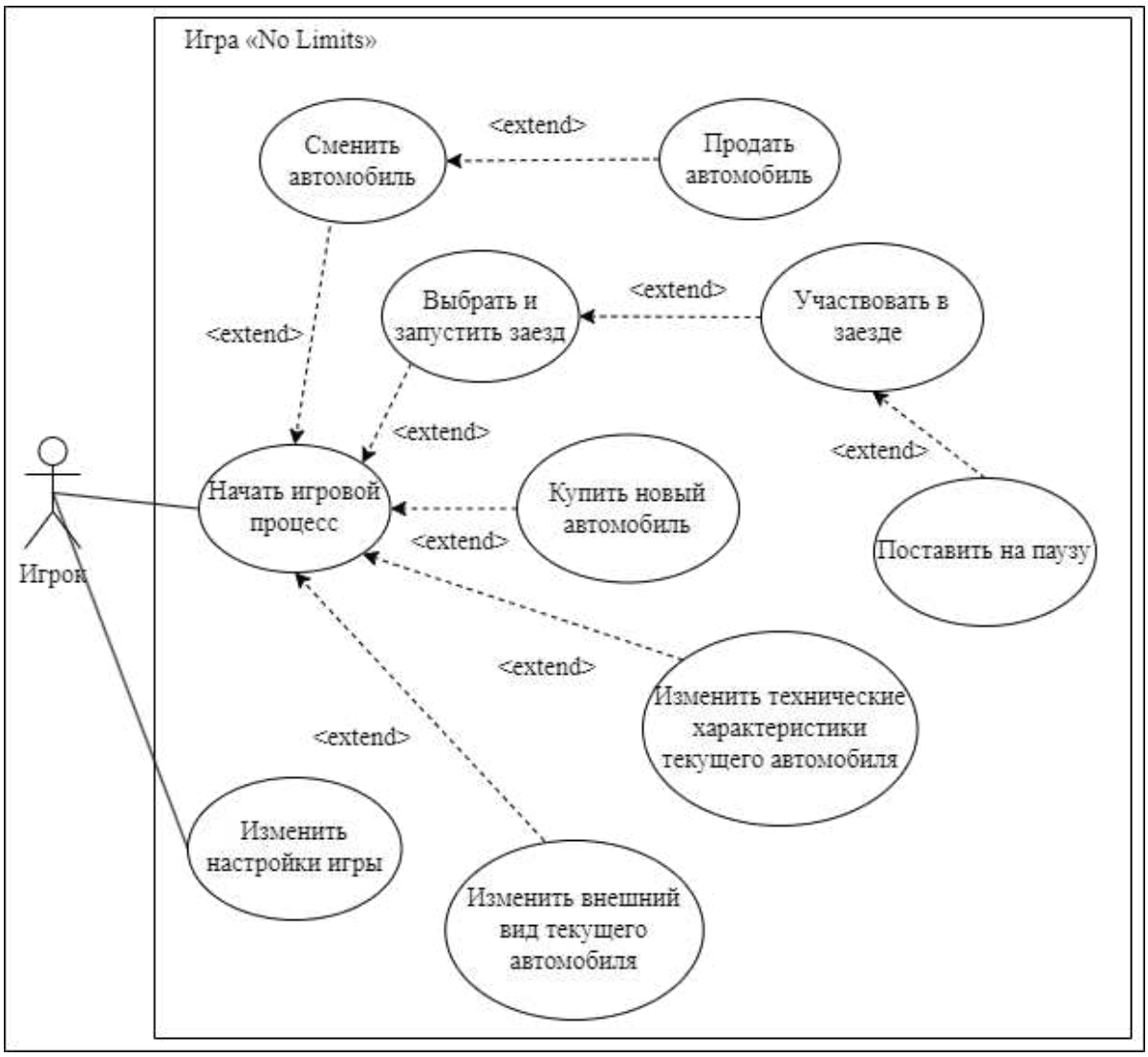

Рисунок 6 – Диаграмма вариантов использования игрового приложения

С игрой «No Limits» взаимодействует только один актер – игрок. Игрок – это человек, запустивший и использующий игровое приложение.

Игрок может изменить настройки игры. Игрок может изменить настройки графики: изменить разрешение экрана, выбрать качество графики, выбрать будет ли игра запущена в полноэкранном или оконном режиме, изменить настройки локализации игры: выбрать русский или английский язык, изменить громкость звука.

Игрок может начать игровой процесс, запустив игровое меню с выбором действия: выбрать заезд, сменить автомобиль, изменить технические характеристики автомобиля, изменить внешний вид автомобиля, купить новый автомобиль.

Игрок может сменить автомобиль на любой автомобиль из своего гаража. Выбранный автомобиль должен становиться текущим.

Игрок может изменить технические характеристики текущего автомобиля: купить или установить купленные улучшения ходовой части автомобиля, которые влияют на управляемость автомобиля, улучшения двигателя, которые влияют на скорость и ускорение автомобили. Покупка улучшений возможна при наличии достаточного количества игровой валюты.

Игрок может изменить внешний вид текущего автомобиля: купить или установить купленные колесные диски, изменить цвет колесных дисков, изменить цвет кузова автомобиля. Покупка улучшений возможна при наличии достаточного количества игровой валюты.

Игрок может продать автомобиль, который хранится у него в гараже. Цена продажи зависит от изначальной стоимости автомобиля и установленных улучшений. Игрок не может продать единственный в своем гараже автомобиль.

Игрок может купить новый автомобиль, представленный в автосалоне. Купленные автомобили должны отправляться в гараж игрока. При этом игрок может покупать неограниченное количество автомобилей. Игрок может купить только те автомобили, на которые у него хватает игровой валюты.

Игрок может выбрать и запустить заезд на сцене выбора заездов. Игрок может запустить только тот заезд, который доступен ему на данный момент.

Игрок может участвовать в заезде: запускается сцена с выбранным заездом. Система должна передать управление автомобилем игроку после трехсекундной отсрочки, чтобы он успел приготовиться к основному игровому процессу. После прохождения заезда на оценку, как минимум, «бронза», игроку должен открываться следующий заезд.

Игрок может поставить на паузу игру во время участия в заезде. Во время паузы игровое время должно останавливаться и на экране появляться меню паузы. После закрытия меню паузы игровое время должно замедляться на три секунды, чтобы игрок приготовился к продолжению заезда.

2.4. Компоненты системы

Архитектура разрабатываемого игрового приложения состоит из компонентов, представленных на рисунке 7 в виде диаграммы компонентов.

Рисунок 7 – Диаграмма компонентов игрового приложения

Основной компонент «пользовательские интерфейсы» отвечает за отображение игровой информации, содержит логику управления интерактивными элементами, такие как кнопки. Также компонент связывает визу-

альную и функциональную части игры, обеспечивая взаимодействие игрока с игрой.

Компонент «автосалон» представляет собой часть пользовательского интерфейса, которая отвечает за покупку новых автомобилей.

Компонент «гараж» представляет собой часть пользовательского интерфейса, которая отвечает за выбор и продажу автомобилей игрока.

Компонент «выбор заездов» представляет собой часть пользовательского интерфейса, которая отвечает за выбор и запуск заездов.

Компонент «основная игровая сцена» представляет собой часть пользовательского интерфейса, которая содержит логику участия игрока в выбранном заезде.

Компонент «тюнинг» представляет собой часть пользовательского интерфейса, которая отвечает за техническое улучшение выбранного автомобиля игрока.

Компонент «стайлинг» представляет собой часть пользовательского интерфейса, которая отвечает за изменение внешнего вида выбранного автомобиля игрока.

Компонент «настройки» представляет собой часть пользовательского интерфейса, которая отвечает за выбор различных игровых настроек игроком.

Компонент «прогресс» отвечает за хранение результатов прохождения игры. Прогресс хранит в себе числовое значение игровой валюты игрока, информацию о пройденных заездах, информацию об автомобилях игрока.

Компонент «валюта» содержит логику использования игровой валюты игроком, отвечает за начисление и снятие игровой валюты со счета игрока.

Компонент «автомобили» содержит логику, которая отвечает за хранение и обработку информации об автомобилях игрока, установленных технических и визуальных улучшениях.

Компонент «сохранение и загрузка» содержит логику загрузки из файла и сохранения в файл прогресса игрока.

Компонент «управление» отвечает за поведение автомобиля в зависимости от ввода игрока.

Компонент «игровые настройки» отвечает за установку выбранных игроком настроек игры и за сохранение их в файл и за загрузку их из файла.

Компонент «локализация» обеспечивает перевод всех игровых текстов на два языка: русский или английский, в зависимости от выбора игрока.

Компонент «звук и музыка» отвечает за воспроизведение различных звуков и музыки во время игрового процесса.

Компонент «дополнительные функции» предоставляет функционал для шины событий и локатора служб для взаимосвязи компонентов системы между собой.

2.5. Макеты пользовательского интерфейса

Пользовательский интерфейс состоит из главного меню, сцены игровых настроек, сцены гаража, сцены выбора автомобиля, сцены тюнинга, сцены стайлинга, сцены выбором заезда и основной игровой сцены.

Главное меню – это первая сцена, которую видит игрок после запуска игрового приложения. Главное меню игры «No Limits» представлено кнопками «Играть», «Настройки», «Выйти». При нажатии на кнопку «Играть» система запустит сцену с гаражом игрока. При нажатии на кнопку «Настройки» система переключиться на сцену, где игрок может настроить параметры игры, такие как графика, язык текста, громкость звука. При нажатии на кнопку «Выйти» игровое приложение будет закрыто. Макет интерфейса главного меню представлен на рисунке 8.

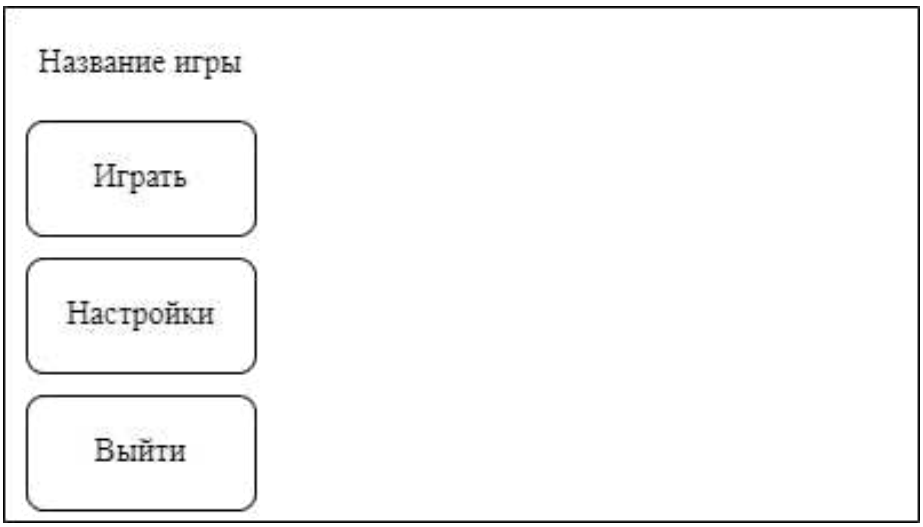

Рисунок 8 – Макет интерфейса главного меню

Сцена настроек состоит из текстовых полей, отражающих суть настраиваемых параметров и интерактивных элементов пользовательского интерфейса. Также интерфейс представлен кнопкой «Назад» и кнопкой «Сохранить». Макет интерфейса сцены настроек представлен на рисунке 9.

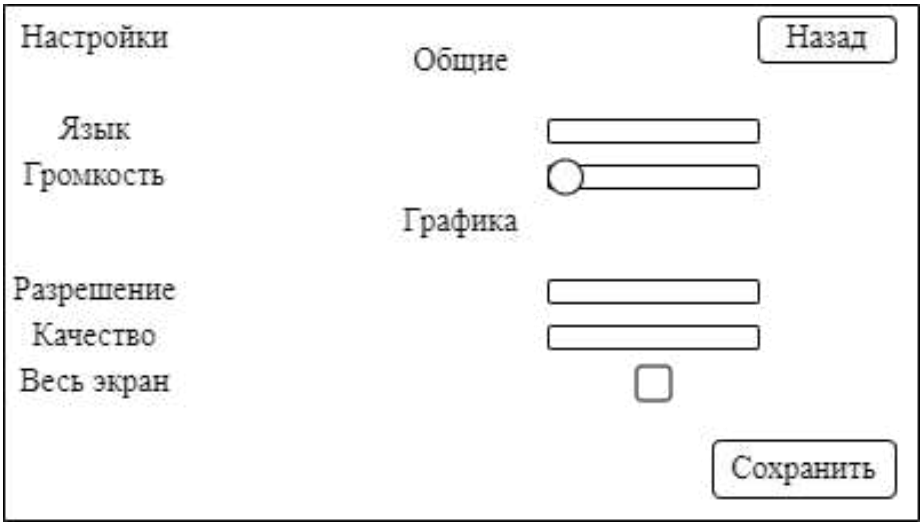

Рисунок 9 – Макет интерфейса сцены настроек

Раздел общих настроек содержит выпадающее меню для выбора языка, английского или русского, слайдер для настройки громкости игры. Раздел графических настроек состоит из выпадающего списка для выбора доступных разрешений экрана, выпадающего списка для выбора качества

изображения и флажка, который означает, будет ли игра запущена в полноэкранной или оконном режиме.

При нажатии на кнопку «Назад» игрок вернется в главное меню, а при нажатии на кнопку «Сохранить» система запишет новые настройки игры в файл и применит новые настройки.

Сцена гаража представляет собой сцену, на которую попадает игрок при нажатии на кнопку «Играть» в главном меню. Интерфейс сцены гаража составляет текстовое поле «Гараж», кнопка «Назад», возвращающая игрока в главное меню. Также внизу экрана расположены пять кнопок: «Заезды», «Мои авто», «Тюнинг», «Стайлинг», «Автосалон». Макет интерфейса сцены гаража представлен на рисунке 10.

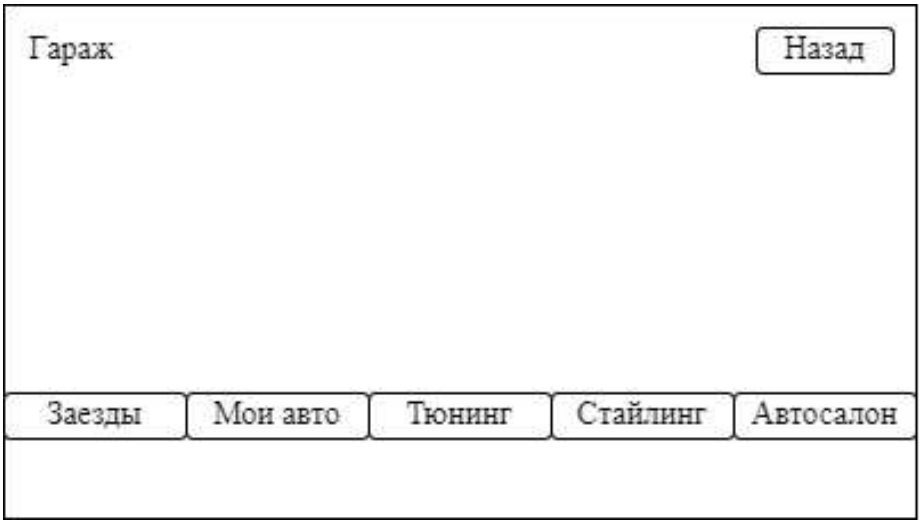

Рисунок 10 – Макет интерфейса сцены гаража

При нажатии на кнопку «Заезды» игрок перейдет на сцену выбора доступного заезда. При нажатии на кнопку «Мои авто» система запустит сцену с выбором имеющихся у игрока автомобилей. При нажатии на кнопку «Тюнинг» игрок попадет на сцену доработки технических характеристик выбранного автомобиля. А при нажатии на кнопку «Стайлинг» на сцену изменения внешнего вида выбранного автомобиля. При нажатии на кнопку «Автосалон» запустится сцена, на которой игрок может выбрать и купить новый автомобиль.

Сцена выбора автомобиля представлена текстовым полем «Мои авто», изображением выбираемого автомобиля, рядом с которым слева и справа расположены кнопки переключения, в виде стрелок, между автомобилями, кнопками «Выбрать», «Продать», «Назад», и полем с отображением характеристик выбираемого автомобиля. Макет интерфейса сцены выбора автомобиля представлен на рисунке 11.

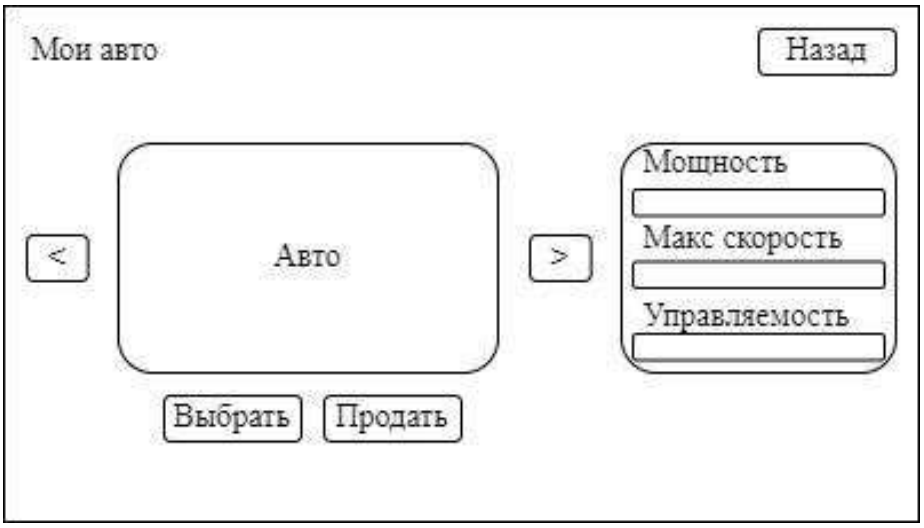

Рисунок 11 – Макет интерфейса сцены выбора автомобиля

При просмотре крайних автомобилей кнопки переключения становятся недоступны. При нажатии на кнопку «Назад» игрок вернется в меню гаража. При нажатии на кнопку «Выбрать» выбранный автомобиль станет текущим автомобилем игрока. При нажатии на кнопку «Продать» выбранный автомобиль будет продан за определенное количество игровой валюты. При этом, если у игрока в гараже остался единственный автомобиль, то кнопка «Продать» становится недоступной для нажатия. При нажатии на кнопку «Назад» игрок вернется на сцену с гаражом.

Сцена тюнинга выбранного автомобиля представлена изображением текущего выбранного автомобиля, полем с характеристиками автомобиля, кнопками «Ходовая часть», «Двигатель» и кнопкой «Назад». Также меню тюнинга представлено числовым значением текущего количества игровой валюты игрока, и также ценой улучшений, которые игрок может установить на свой автомобиль.

Макет интерфейса сцены тюнинга выбранного автомобиля представлен на рисунке 12.

Рисунок 12 – Макет интерфейса сцены тюнинга выбранного автомобиля

При нажатии на кнопки «Ходовая часть», «Двигатель», перед игроком откроется панель с выбором улучшений, которые можно установить на данный автомобиль. Изменения в технических характеристиках отображаются в поле технических характеристик. При нажатии на кнопку «Назад» игрок вернется на сцену с гаражом.

Сцена стайлинга представляет собой сцену, на которой игрок может изменить внешний вид выбранного автомобиля. Макет интерфейса сцены стайлинга выбранного автомобиля представлен на рисунке 13.

Сцена стайлинга выбранного автомобиля представлено изображением текущего модифицируемого автомобиля, кнопками «Цвет кузова», «Диски», «Цвет дисков», «Назад». Также меню стайлинга представлено числовым значением текущего количества игровой валюты игрока, и также ценой визуальных улучшений, которые игрок может установить на свой автомобиль. При нажатии на кнопки «Цвет кузова» и «Цвет дисков» перед игроком откроется палитра для выбора цвета соответственно кузова и дисков текущего автомобиля игрока. При нажатии на кнопку «Диски» перед игроком откроется панель для выбора колесных дисков. При нажатии на кнопку «Назад» игрок вернется в меню гаража.

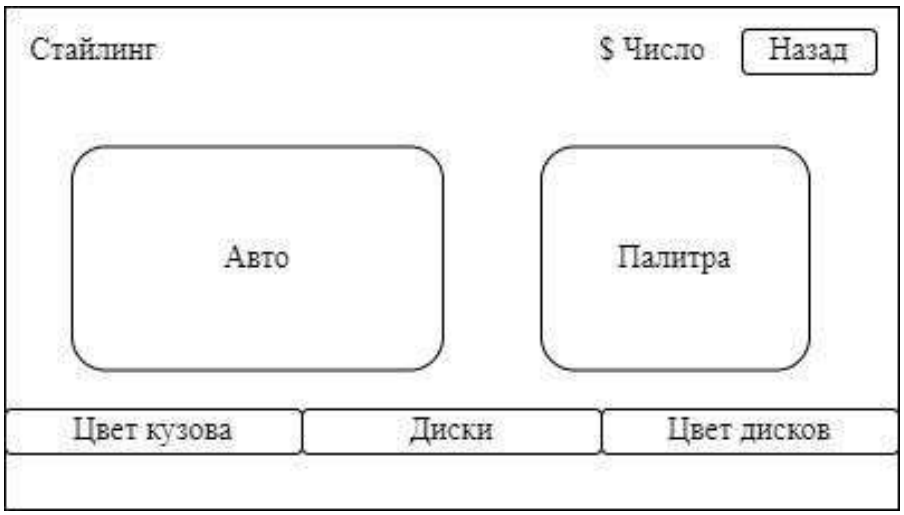

Рисунок 13 – Макет интерфейса сцены стайлинга выбранного автомобиля

Сцена автосалона представлена изображением выбираемого автомобиля, рядом с которым расположены кнопки для переключения между автомобилями, полем для отображения его технических характеристик, кнопками «Купить» и «Назад». Также сцена автосалона представлена числовым значением текущего количества игровой валюты игрока, и также ценой автомобилей, которые игрок может приобрести. Макет интерфейса сцены автосалона представлен на рисунке 14.

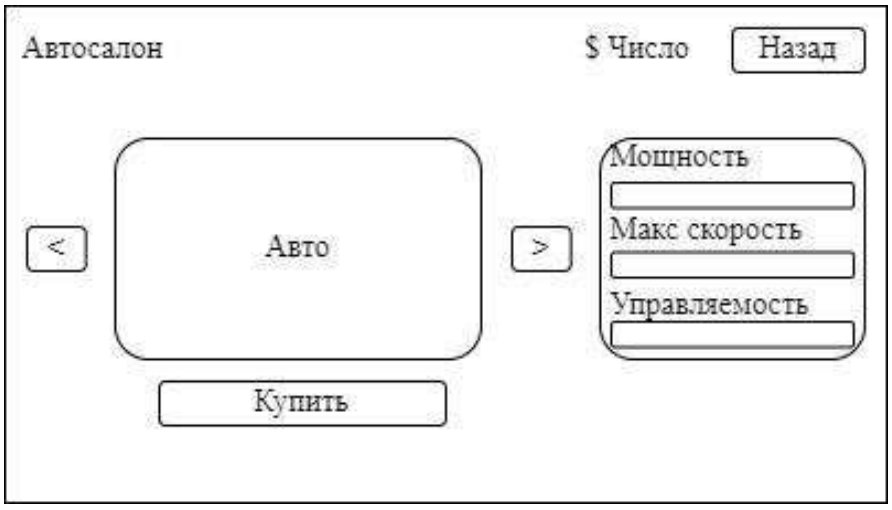

Рисунок 14 – Макет интерфейса сцены автосалона

При просмотре крайних автомобилей кнопки переключения становятся недоступны. При нажатии кнопку «Купить» выбранный автомобиль

будет доставлен в гараж игрока. При нажатии на кнопку «Назад» игрок вернется на сцену с гаражом.

Сцена выбора заезда состоит из панели текущего выбираемого заезда, на котором отображены его название в виде текстового поля, личного рекорда, отраженного числовым значением, и числовыми значениями в виде цели и наград, которые игрок может получить при прохождении данного заезда. Рядом с панелью с информацией о заезде располагаются кнопки выбора заезда. Также сцена представлена кнопками «Выбрать» и «Назад». Макет интерфейса меню выбора заезда представлен на рисунке 15.

Рисунок 15 – Макет интерфейса меню выбора заезда

При просмотре крайних заездов кнопки переключения становятся недоступны. При нажатии на кнопку «Выбрать» запустится выбранный игроком заезд. При нажатии на кнопку «Назад» игрок вернется в меню гаража.

Основная игровая сцена представлена следующими элементами: числовым полем, отражающее количество набранных очков игроком, числовое значение набирающихся очков на данный момент, текущий множитель, на который умножаются очки игрока, и числовым значением бонусных и штрафных очков, которые получил игрок в данный момент. В правом верхнем углу представлены цели заезда, чтобы игрок мог легко ориентироваться на тот результат, которого хочет достичь.

Помимо этого, экран основной игровой сцены представлен спидометром, который отражает в виде числового значения текущую скорость игрока. Макет интерфейса основной игровой сцены представлен на рисунке 16.

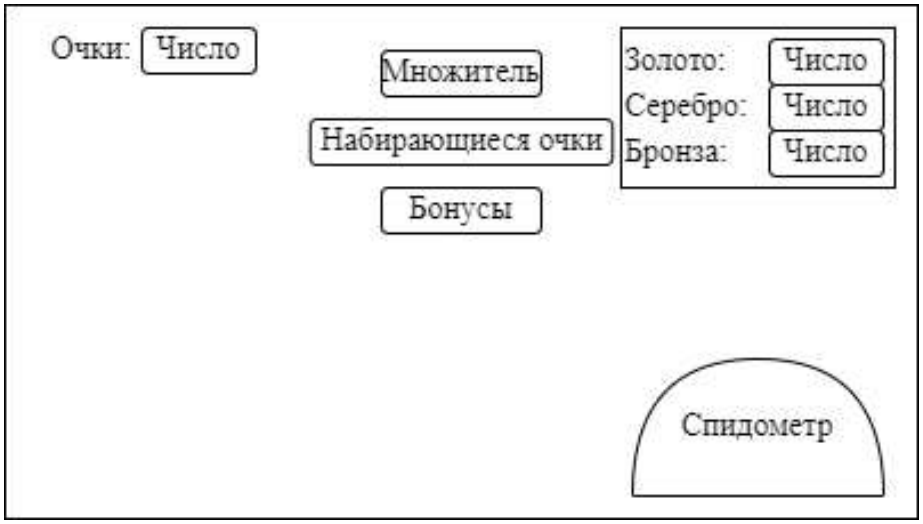

Рисунок 16 – Макет интерфейса основной игровой сцены

После того, как игрок набирает необходимое количество очков на определенную оценку, то цель на эту оценку подкрашивается в зеленый цвет, что символизирует игроку о прохождении заезда на эту оценку.

Вывод по второй главе

Во второй главе было спроектировано игровое приложение. Была разработана концепция игры «No Limits», были выявлены функциональные и нефункциональные требования к разрабатываемой системе. Также были приведены и описаны диаграммы использования системы и компонентов игрового приложения. Были созданы и описаны макеты пользовательских интерфейсов.

3. РЕАЛИЗАЦИЯ ИГРОВОГО ПРИЛОЖЕНИЯ

3.1. Средства реализации

В качестве средства реализации игрового приложения были выбраны межплатформенная среда разработки компьютерных игр Unity 2022.3.4f, интегрированная среда разработки программного обеспечения Visual Studio Community 2022 и язык программирования C#.

Основными преимуществами Unity являются наличие простого интерфейса, межплатформенная поддержка более 20 операционных систем для персональных компьютеров, мобильных устройств, интернет-приложения и других. Также преимуществом выступает популярность Unity, ведь найти полезную информацию, ускоряющую процесс разработки, очень просто.

Игровое приложение в Unity делится на сцены, которые представляют собой специальные файлы, содержащие игровые объекты, построенные из моделей, звуков, текстур, материалов, скриптов (сценариев).

Игровые объекты являются контейнерами для различным компонентов (Components) [18]. Компоненты Unity определяют поведение объектов в игре. Различные компоненты представлены конкретным классом. Программирование в Unity заключается в создании собственных компонентов. Пользовательские классы-компоненты должны наследоваться он встроенного класса `MonoBehaviour` [19]. Объекты таких классов должны создаваться с помощью метода Instantiate. Конструктор заменяют встроенные методы `Awake` и `Start`.

Модели автомобилей были взяты из сборника моделей STYLIZED: Complete Drift Cars из магазина Unity Asset Store [20].

Звуки для игры были взяты с сайта freesound.org [21], где звуки распространяются пользователями сайта бесплатно для их использования.

Музыка для игры была взята у музыканта Alexi Action, который распространяет свои произведения бесплатно для коммерческого использования.

3.2. Архитектура разработанной системы

Архитектура разработанной системы построена с использованием таких шаблонов проектирования как Entry Point, Service Locator и Event Bus.

Entry Point

Entry Point («Точка входа») [22], представляет собой шаблон проектирования, который решает проблему порядка инициализации компонентов системы. Большим недостатком Unity является порядок инициализации объектов при загрузке игры. Точный порядок инициализации объектов не предопределен [23], что создает проблемы зависимостей между объектами. Это проблема приводит к тому, что объект, который зависит от другого, не успевает инициализироваться до того, как потребуется ссылка на него. В итоге это приводит к некорректной работе системы и ошибкам, связанными с отсутствием ссылки на объект. Благодаря использованию связки Entry Point и Service Locator можно явно в коде определить порядок инициализации объектов.

В разработанной системе Entry Point представляет собой пустую сцену, с одним неуничтожаемым объектом на ней типа `EntryPoint`. При запуске сцены у объекта типа `EntryPoint` вызывается метод `Awake`, в котором происходит инициализация всех компонентов системы, каждому компоненту системы предоставляются необходимые ссылки. После инициализации всех компонентов происходит загрузка прогресса игрока. Когда все компоненты успешно проинициализированы, прогресс игрока загружен, система запускает сцену главного меню игры.

Service Locator

Service Locator («Локатор служб») представляет собой шаблон проектирования, суть которого заключается в том, чтобы иметь объект, который хранит ссылки на все сервисы, необходимые для работы приложения [24]. Сервис представляет собой комплекс определенных функций, направленных для решение определенной задачи или предоставления

определенных услуг. Все сервисы должны быть унаследованы от интерфейса `IService`, чтобы их можно было легко зарегистрировать в локаторе служб.

Локатор служб представлен классом `ServiceLocator`, который представляет собой синглтон, то есть обеспечивает существование единственного экземпляра класса и обеспечивает глобальный доступ к нему [25].

В качестве хранилища для экземпляров служб используется словарь, ключом каждой службы является имя типа службы, а значением сам экземпляр службы.

Для того, чтоб зарегистрировать экземпляр службы используется метод `Register`, который принимает обобщенный аргумент `TService`, который должен реализовать интерфейс `IService`. В случае, если служба ранее уже была зарегистрирована, то новый экземпляр зарегистрирован не будет.

Для того, чтобы получить доступ к службе используется метод `Get`, который извлекает из словаря нужный требуемый программистом экземпляр. Он также использует в качестве ключа для поиска службы имя типа самой службы. В случае, если требуемая служба не зарегистрирована в локаторе служб, возникает исключение `InvalidOperationException`.

Регистрация всех служб происходит при запуске игрового приложения, в точке входа, до момента, когда запустится сцена с главным игровым меню.

Event Bus

Event Bus («Шина событий») представляет собой объект-посредник, который принимает сигналы от отправителей и передает их получателям. Сигналы представляют собой события, которые могут содержать данные, полезные для получателей события. Получатели – это объекты, которые подписываются на сигналы, а отправители – это объекты, вызывающие эти события. Ключевой особенностью шины событий является то, что объекты могут подписываться на сигналы другого объекта, при этом отправитель

сигнала и получатель сигнала не знают о существовании друг друга, но оба знают о существовании шины событий.

Шина событий представлена классом `EventBus`. В качестве хранилища списка подписок на сигналы используется словарь, ключом которого является строка, представляющая тип сигнала, а значением список методов обратного вызова, ассоциированных с этим сигналом, с их приоритетами.

Метод `Subscribe` представляет функционал подписки на сигнал определенного типа. При этом сигнал должен представлять собой объект определенного класса, реализующий интерфейс `ISignal`. Данный подход является существенным минусом шины событий, ведь для каждого игрового события необходим свой объект-сигнал.

Метод `Unsubscribe` представляет функционал для отписки от сигнала. Отписка необходима для предотвращения `NullReferenceException`.

Метод `Invoke` предоставляет функционал вызова сигнала определенного типа. Метод вызывает все подписанные делегаты, ассоциированные с этим сигналом.

3.3. Реализация прогресса

Прогресс игрока представлен классом `Progress`, который хранит в себе информацию о текущем ходе прохождения игры игроком. Класс `Progress` содержит поля в виде числового значения игровой валюты, поля типа `Garage`, который хранит в себе всю информацию об автомобилях игрока. Также класс `Progress` хранит в себе информацию о пройденных заездах игрока, представленной в виде объекта класса `RaceProgress`.

Автомобили игрока

Класс `Garage` отвечает за хранение информации об автомобилях игрока. В качестве полей имеет целочисленное поле `CurrentCarId`, который хранит в себе текущий уникальный идентификатор автомобиля, и список, который хранит все автомобили типа `Car`. Класс предоставляет интерфейсные методы `AddNewCar` для добавления нового автомобиля в список

автомобилей, SelectCar для выбора нового текущего автомобиля, SellCar для удаления автомобиля из списка при его продаже, GetCurrentCar для получения автомобиля из списка по уникального идентификатору.

Класс Car представляет собой класс, который отвечает за хранение информации о конкретном автомобиле. Класс хранит целочисленный уникальный идентификатор автомобиля, строковое название автомобиля, цвета кузова и колесных дисков, представленные типом UnityEngine.Color, список купленных улучшений для двигателя и ходовой части автомобиля, список купленных колес, установленные улучшения для двигателя и ходовой части автомобиля, установленные колесные диски, и значения технических характеристик автомобиля, представленные мощностью, максимальной скоростью и уровнем сцепления с трассой. Также имеет целочисленное поле, которое хранит цену автомобиля.

Пройденные заезды

Класс RaceProgress отвечает за хранение информации о пройденных заездах игроком в виде списка объектов типа Race. Объект типа Race хранит в себе уникальный идентификатор заезда, лучший счет игрока и флаг, который показывает, пройден заезд или нет. Класс RaceProgress предоставляет интерфейсный метод UpdateRaceProgress для обновления прогресса. В случае, если игра запускается впервые и до этого не было сохранено никакого прогресса, то класс Progress через конструктор проинициализирует количество игровой валюты, создаст гараж игрока с одной стартовой машиной. Доступ к прогрессу игрока можно получить, используя службу ProgressService, который предоставляет прямой доступ к объекту класса Progress.

Внутриигровая валюта

За работу с валютой игрока отвечает класс Currency. Класс содержит закрытое поле _value, которое представляет собой количество игровой валюты игрока. С помощью свойства Value можно получить текущее значение этого поля. Для обработки совершения покупки и продажи отве-

чают соответственно методы `Buy(int price)` и `Sell(int price)`. Метод Buy позволяет совершить покупку только в том случае, если у пользователя достаточно средств для осуществления данной покупки. Проверка достаточно ли средств у игрока осуществляется в методе `IsEnough-Value(int price)`. В случае, если игрок располагает необходимым количеством игровой валюты, то значение цены покупки вычитается из поля `_value`. Метод `Sell` увеличивает значение `_value` на значение цены продажи. Помимо этого, класс `Currency` обновляет и устанавливает новый прогресс, передаваемый через объект класса `Progress`. Использование класса `Currency` возможно только через службу, реализуемую классом `CurrencyService`, который инкапсулирует внутри себя всю работу, связанную с классом `Currency`. Другие компоненты системы могут манипулировать валютой игрока через эту службу.

Сохранение и загрузка

За сохранение и загрузку игрового прогресса отвечает класс `Save-LoadService`. В данной классе определены методы сохранения и загрузки прогресса в файл формата JSON. Для сохранения и загрузки прогресса классу необходимо получить доступ к службе, которая реализует интерфейс `IProgressService`. В методе `Save` данные сериализуются автоматически посредством встроенного в Unity функционала, предоставляемого `JsonUtility.ToJson`. Метод `Load` проверяет наличие файла сохранения, и если файл существует, то содержимое файла десериализуется в прогресс игрока с помощью `JsonUtility.FromJson`, а в случае отсутствия файла сохранения в методе проинициализируется объект класса `Progress`, и при его создании проинициализируется стартовый прогресс.

С учетом реализации компонента прогресса файл сохранения имеет вид, представленный в листинге 1.

Листинг 1 – Файл сохранения игры «No Limits»

```
{ "Currency": 5000,
  "Garage": {
    "CurrentCarId": 242072325,
```

```
    "Cars": [{
        "Id": 242072325,
        "Name": "Roku",
        "Body": {
          "PrefabPath": "Prefabs/Cars/roku",
          "Color": {"r": 0.0,"g": 0.0,"b": 0.0,"a": 0.0}
        },
        "Wheel": {
          "PrefabPath": "Prefabs/Rims/roku_rim",
          "Color": {"r": 0.0,"g": 0.0,"b": 0.0,"a": 0.0}
        },
        "PurchasedWheels": ["Prefabs/Rims/roku_rim"],
        "InstalledEngineImprovement": "",
        "PurchasedEngineImprovements": [],
        "InstalledChassisImprovement": "",
        "PurchasedChassisImprovements": [],
        "Price": 1500,
        "Specs": {
          "StockEnginePower": 500.0,
          "StockMaxSpeed": 70.0,
          "BrakePower": 50000.0,
          "StockGrip": 0.30000001192092898,
          "CurrentEnginePower": 500.0,
          "CurrentMaxSpeed": 70.0,
          "CurrentGrip": 0.30000001192092898
   }}]},
  "RaceProgress": {"Races": [{"Id": 0,"BestScore": 0,"IsPassed": false}]}
}
```


3.4. Реализация музыки и звуков

За воспроизведение музыки и звуков в Unity отвечает встроенный компонент `AudioSource`, которому необходимо передать файл для воспроизведения. Для хранения информации об музыке был разработан класс, контейнер для данных, `MusicData`, который содержит информацию о исполнителе, названии и файле для воспроизведения. Пример объекта `MusicData` для музыки представлен на рисунке 17.

Рисунок 17 – Пример объекта `MusicData` для музыки

Для хранения всей музыки был разработан класс `Playlist`, который хранит объекты типа `MusicData` в виде массива. `MusicManager` воспроизводит случайную музыку из массива класса `Playlist`. Также `MusicManager` при воспроизведении трека передает информацию о проигрываемом треке классу `MusicInfo`, который выводит на экран пользователя информацию об исполнителе и название музыки.

За воспроизведение звуков отвечает класс `SoundManager`. В классе `SoundManager` через компонент `AudioSource` с помощью метода `PlayClipAtPoint` воспроизводит определенный звук. Методу `PlayClipAtPoint` необходимо передать `AudioClip`, то есть файл звука для воспроизведения и позицию воспроизведения. В классе `SoundManager` происходит подписка на сигналы шины событий. Сигналы передают информацию о позиции воспроизведения. Методы-обработчики сигналов определяют какие звуковые файлы должны воспроизводиться. Для того, чтобы класс не хранил информацию обо всех звуковых файлах, `AudioClip` создаются с помощью метода `Load` класса `Resources`, который позволяет загружать ресурсы во время игры. При этом загружаемые ресурсы должны храниться в папке `Resources`. Для загрузки ресурса методу `Load` необходимо получить путь к ресурсу, который необходимо загрузить.

3.5. Реализация пользовательских интерфейсов

При запуске игрового приложения игрока встречает главное меню игры, на котором расположены три кнопки: кнопка «Играть», кнопка «Настройки» и кнопка «Выйти из игры».

Из главного меню игрок может закрыть игровое приложение, нажав на кнопку «Выйти из игры». Также игрок может запустить меню с настройками игры, нажав на кнопку «Настройки». Помимо этого, игрок может перейти на сцену с гаражом игрока, нажав на кнопку «Играть». Главное меню игры представлено на рисунке 18.

Рисунок 18 – Главное меню игры «No Limits»

Игровые настройки

При нажатии кнопки «Настройки» система запускает сцену с настройками игрового приложения.

За сохранение игровых настроек отвечает класс `GameSettings`. Класс помечен атрибутом `System.Serializable`, для того чтобы значения настроек могли быть сохранены в файл и загружены из файла. Класс содержит поля, хранящие информацию о выбранном языке, громкости звука, уровне графики, разрешении экрана. В случае, если игра запускается впервые, то класс через конструктор установит язык на русский, громкость на максимальное значение, самое высокое качество графики, и разрешение экрана, соответствующее монитору игрока.

Для управления игровыми настройками была разработана служба `GameSettingsService`, которая обрабатывает сигналы от элементов пользовательского интерфейса. Методы `SetLocale`, `SetVolume`, `SetResolution`, `SetQuality`, `SetScreenMode`, `ApplySettings`, `LoadSettings`, `SaveSettings` используются для установки, сохранения и загрузки игровых настроек.

Реализованная сцена с настройками игры представлена на рисунке 19.

Рисунок 19 – Настройки игры «No Limits»

За переключение полноэкранного режима отвечает элемент пользовательского интерфейса Toggle, который посылает сигнал об изменении своего состояния. Обрабатывается сигнал методом `SetScreenMode` класса `GameSettingsService`.

За выбор качества изображения и разрешения экрана отвечают соответствующие компоненты Dropdown, изменения состояний которых обрабатывают методы `SetQuality` и `SetResolution` соответственно.

За настройку громкости отвечает компонент пользовательского интерфейса Slider, изменения которого обрабатывает метод `SetVolume`.

При нажатии на кнопку «Сохранить» `GameSettingService` с помощью методов `SaveSettings` и `ApplySettings` сохранит и применит новые настройки для игры.

За смену языка текста отвечает Dropdown, изменения которого обрабатываются методом `SetLocale`.

Локализация

Локализация выполнена с помощью встроенного механизма Localization [26] в движок Unity. С его помощью можно установить различное зна-

чение для строк, отображаемых на экране пользователя, в зависимости от выбранного языка.

Для начала работы необходимо создать настройки локализации и выбрать локали. Локаль определяет на какой язык локализуется приложение. Также необходимо выбрать локаль по умолчанию, то есть выбрать какой язык будет использовать приложение при первом запуске.

Для игры «No Limits» были выбраны два поддерживаемых языка: русский и английский. Причем, русский язык является языков по умолчанию. Далее необходимо создать коллекцию таблиц строк. Таблица представляет собой структуру данных с ключом и значениями, которые должны принимать строки, в зависимости от выбранной локали. Таблица строк для главного меню игры представлена на рисунке 20.

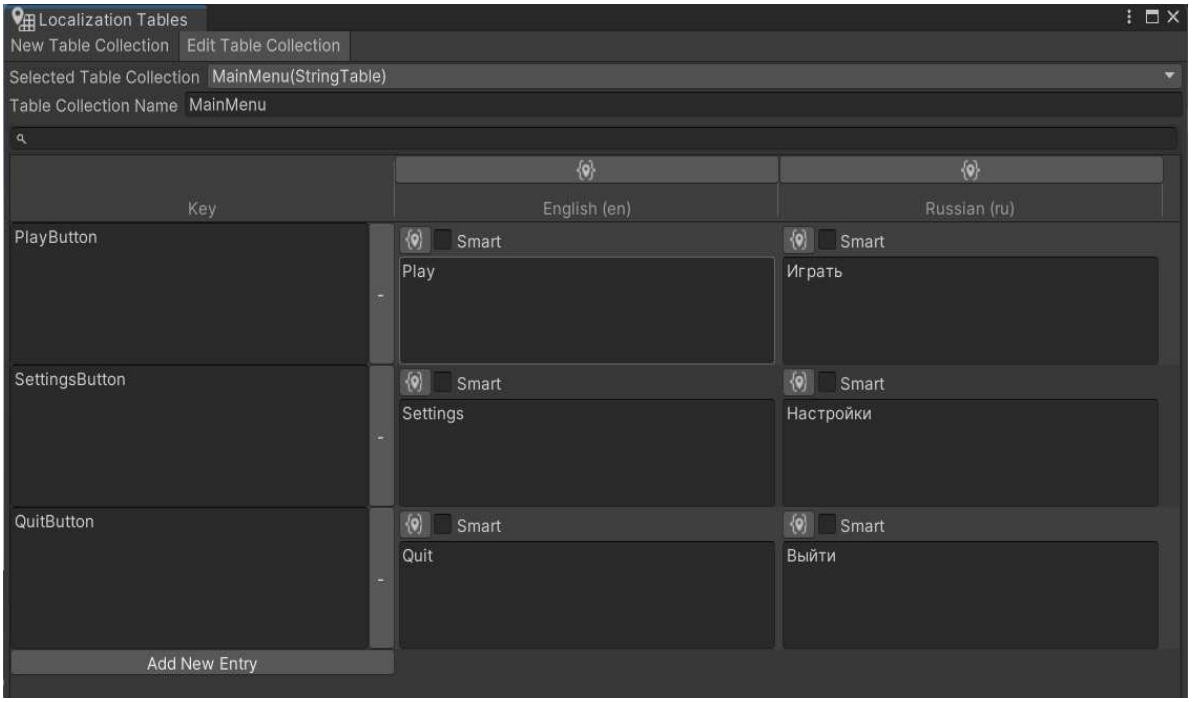

Рисунок 20 – Таблица строк для главного меню игры

С помощью компонента LocalizeStringEvent для компонентов Text можно сопоставить текст и ключ из таблицы для автоматического перевода текста на различные языки. На рисунке 21 представлена настройка текста кнопки «Играть».

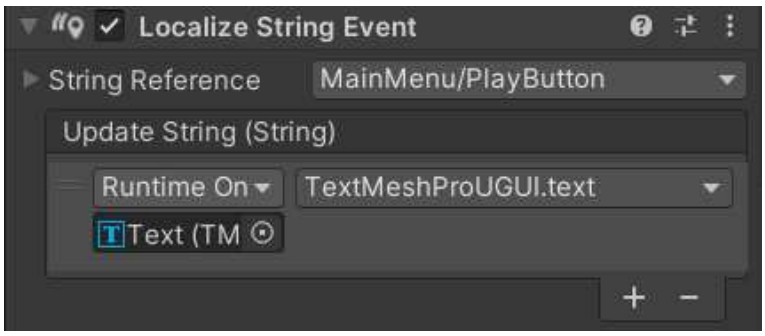

Рисунок 21 – Настройка локализации для кнопки «Играть» в главном меню

Гараж

При нажатии на кнопку «Играть» в главном меню система запустит сцену с гаражом. Посредине в окне представлена текущая машина игрока. Помимо автомобиля на сцене располагаются кнопки переключения на соответствующие сцены. При нажатии на кнопку «Назад» система вновь запустит сцену главного меню.

Реализованная сцена гаража представлена на рисунке 22.

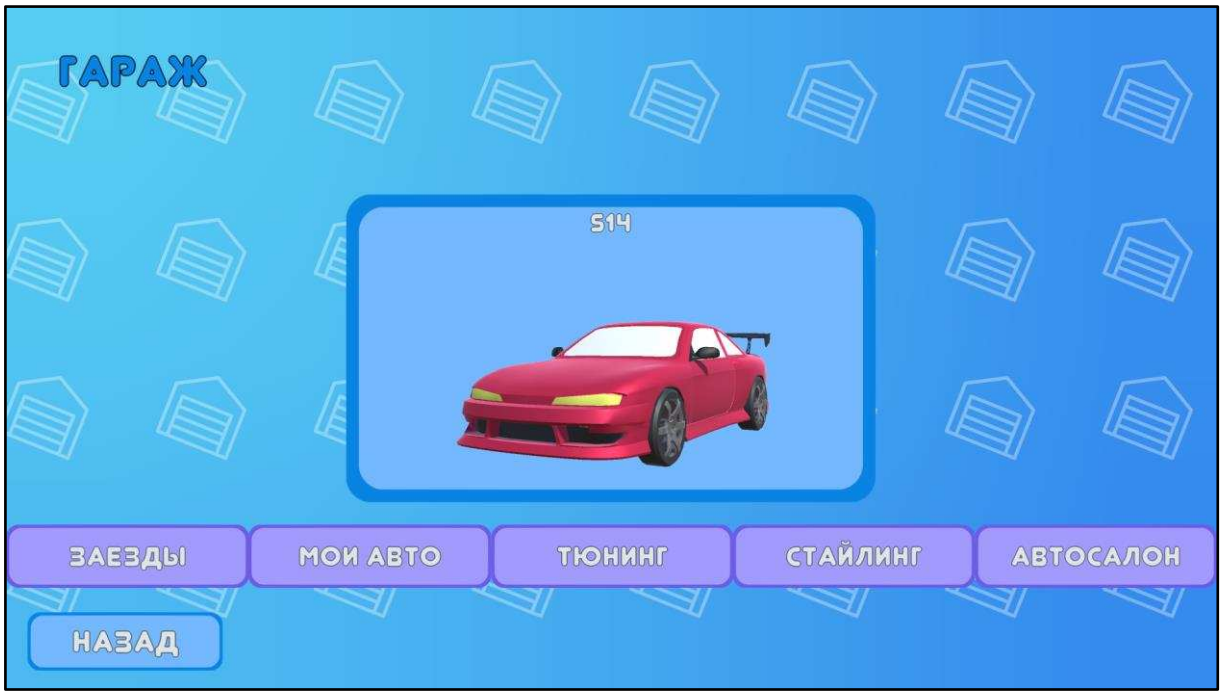

Рисунок 22 – Сцена «Гараж» игры «No Limits»

Для создания игрового объекта, который представляет собой текущий автомобиль игрока, используется информация о текущим автомобиле и интерфейсный метод `CreateCar` интерфейса `ICarFactory`, который реа-

лизует `CarFactory`. Метод принимает информацию об автомобиле, позицию, где должен быть создан автомобиль и флаг, который показывает, можно ли управлять автомобилем. Флаг управления выставляется только при создании автомобиля на сцене с заездом. Реализация метода представлена в листинге 2.

Листинг 2 – Метод `CreateCar`

```
public GameObject CreateCar(Car car, Transform container,
    bool isControlling = false)
{
    ICarBuilder carBuilder = new CarBuilder(_assets, container);
    carBuilder.SetBody(car.Body);
    carBuilder.SetWheels(car.Wheel);
    carBuilder.SetControl(isControlling);
    return carBuilder.Build();
}
```

Для удобства построения объекта был применен шаблон проектирования «Строитель» [27], который позволяет инкапсулировать создание объекта и позволяет разделить его на различные этапы.

Реализован шаблон с помощью интерфейса `ICarBuilder`, который с помощью интерфейсных методов `SetBody`, `SetWheels`, `SetControl` позволяет установить все необходимые параметры для создания автомобиля. Метод `Build` отвечает за создание объекта на сцене. Реализация метода `Build` представлена в листинге 3.

Листинг 3 – Метод `Build`

```
public GameObject Build()
{
    _gameCar = Object.Instantiate(
        _assets.Load<GameObject>(_body.PrefabPath), _container);
    if (_isControlling == true)
    {
        _gameCar.GetComponentInChildren<Rigidbody>().isKinematic = false;
    }
    BodyColor bodyColor = _gameCar.GetComponentInChildren<BodyColor>();
    bodyColor.SetCurrentColor(_body.Color);
    WheelColor[] wheelColors =
        _gameCar.GetComponentsInChildren<WheelColor>();
    foreach (WheelColor wheelColor in wheelColors)
    {
        wheelColor.SetCurrentColor(_wheel.Color);
    }
    var rims = Object.FindObjectsOfType<Rim>();
    foreach (Rim rim in rims)
    {
        Mesh newRimMesh = GetRimMeshForWheel(_wheel);
```

```
            if (rim.IsRigthSide)
            {
                rim.gameObject.transform.Rotate(0, 180, 0);
            }
            rim.SetCurrentMesh(newRimMesh);
        }
        _gameCar.transform.localPosition = Vector3.zero;
        _gameCar.transform.localRotation = Quaternion.identity;
        return _gameCar;
    }
```

Метод создает объект на сцене с помощью метода `Instantiate`, передав в метод `Instantiate` информацию об кузове автомобиля. Если флаг управления автомобилем выставлен как `true`, то через компонент `Rigidbody`, определяет, что на автомобиль влияет физика как на твердое тело. Получает компонент `BodyColor`, который отвечает за отображение цвета кузова, и устанавливает для него цвет кузова. Аналогично с компонентом `WheelColor`. Компонент `Rim` отвечает за отображение геометрии колесных диск.

Автосалон

При нажатии на кнопку «Автосалон» на сцене «Гараж», система запустит сцену, на которой игрок может приобрести новые автомобили. Реализованная сцена автосалона представлена на рисунке 23.

Рисунок 23 – Сцена «Автосалон» игры «No Limits»

Для вывода текущего значения игровой валюты у игрока используется класс `CurrencyService`, который обращается к классу `Currency` и получает значение игровой валюты.

Для хранения информации об автомобилях, представленных в автосалоне, был разработан класс `CarInShowroom`. Класс наследуется от `ScriptableObject` [28], то есть он представляет собой контейнер для данных. Он содержит название автомобиля, кузов и цвет кузова автомобиля, колесные диски и цвет диск, технические характеристики по умолчанию и цену автомобиля. Для хранения всех данных об автомобилях в автосалоне был разработан класс `Showroom`, который также является `ScriptableObject`. `Showroom` имеет одно поле, представляющее собой массив `CarInShowroom`. Пример заполнения `CarInShowroom` объекта представлен на рисунке 24.

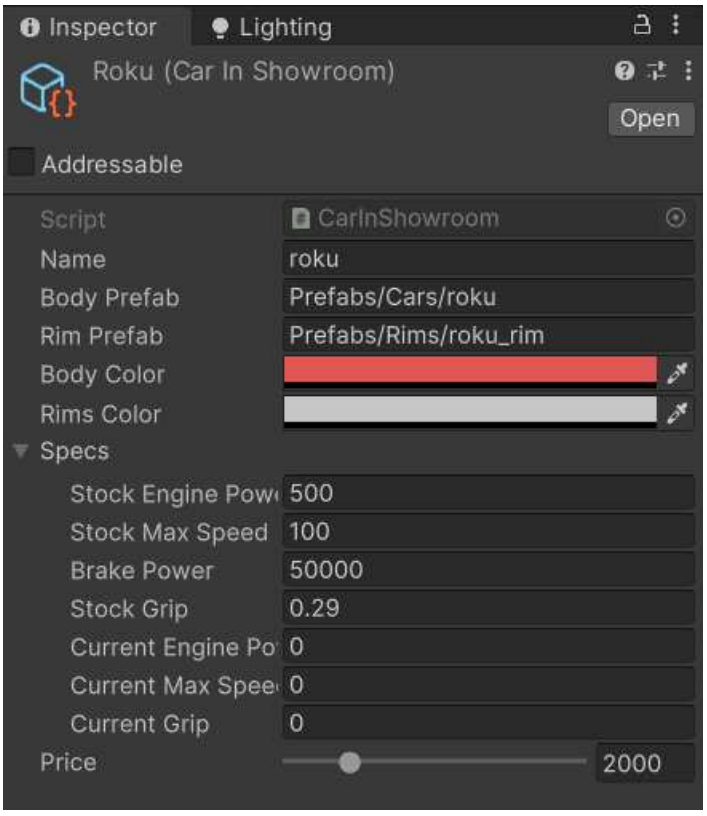

Рисунок 24 – Пример заполнения `CarInShowroom` для автомобиля «Roku»

Для вывода значений технических характеристик используется поле `fillAmount` компонента пользовательского интерфейса Image. Полю мож-

но задать значение от 0 до 1, где 0 – ничего не отображается, а 1 – изображение отображается полностью. Расчет значения для данного поля выполняется функцией `Mathf.InverseLerp` с параметрами `SpecsRange.MinPower`, `SpecsRange.MaxPower` и `CurrentEnginePower`.

В классе `SpecsRange` хранятся значения минимальных и максимальных значений технических характеристик автомобилей. В зависимости от значения технической характеристики относительно ограничений рассчитывается `fillAmount`.

Для обработки покупки автомобиля был разработан класс `ShowroomService`. Класс имеет свойство `Cars` типа `CarInShowroom`, которое представляет собой массив всех автомобилей, представленных в автосалоне, а свойство `CurrentCar` типа `CarInShowroom` указывает на текущий просматриваемый автомобиль. Метод `OnBuyingNewCar` обрабатывает сигнал о покупке нового автомобиля при нажатии на кнопку «Купить». При покупке автомобиль отправляется в гараж игрока и обновляется значение игровой валюты. Реализация метода представлена в листинге 4.

Листинг 4 – Реализация метода `OnBuyingNewCar`

```
private void OnBuyingNewCar(BuyingNewCarSignal signal)
{
    if (IsAvailableForPurchase(CurrentCar.Price))
    {
        _currency.Buy(CurrentCar.Price);

        _garage.AddNewCar(CurrentCar);

        _eventBus.Invoke(new CurrencyChangedSignal(_currency.Value()));
        _eventBus.Invoke(new AvailableForPurchaseSignal(
            IsAvailableForPurchase(CurrentCar.Price)));

        saveLoad.Save();
    }
}
```

Сначала с помощью метода `IsAvailableForPurchase` проверяется, достаточно ли у игрока игровой валюты для покупки просматриваемого автомобиля. В случае, если игроку хватает игровой валюты со счета игрока списывается сумма покупки, а в гараж отправляется купленный автомо-

биль. В случае, если игроку не хватает игровой валюты, то кнопка «Купить» становится недоступной для взаимодействия.

Также класс имеет метод `OnSelectedIndexChange`, который обрабатывает сигнал о смене просматриваемого автомобиля. Сигнал создается при нажатии на кнопки переключения автомобиля.

Автомобили игрока

При нажатии на кнопку «Мои авто» из сцены «Гараж» система запустит сцену с автомобилями игрока. Игрок может поменять текущий автомобиль, либо продать, который ему больше не нужен. Реализованная сцена «Мои авто» представлена на рисунке 25.

Рисунок 25 – Сцена «Мои авто» игры «No Limits»

При нажатии на кнопку «Продать» вызывается метод-обработчик, код которого представлен в листинге 5.

Листинг 5 – Реализация метода `OnSellCar`

```
public void OnSellCar(CarSellingSignal signal
{
    if (_garage.CurrentCarId == _garage.Cars[_viewedId].Id)
    {
        if (_viewedId == 0)
        {
            _garage.SelectCar(_garage.Cars[1].Id);
        }
```

```
            else
            {
                _garage.SelectCar(_garage.Cars[_viewedId - 1].Id);
            }
        }
        OnSell();

        if (_garage.Cars.Count == 1)
            _garage.SelectCar(_garage.Cars[0].Id);

        _garage.UpdateProgress(_progress.Progress);
        _saveLoad.Save();

        _eventBus.Invoke(new CarSoldSignal(_viewedId));
    }
```

В случае если игрок продает свой текущий выбранный автомобиль, то автоматически должен выбраться другой. В первом случае игрок продает первый автомобиль, и тогда его текущим становится следующий, расположенный в гараже. В остальных случаях выбранным становится предыдущий. Во время продажи автомобиля он удаляется из списка автомобилей игрока, а на счет игрока начисляется сумма продажи. При этом игрок не может продать единственный свой автомобиль. Если у игрока остался один автомобиль, то кнопка «Продать» становится недоступной для взаимодействия.

При нажатии на кнопку «Выбрать» вызывается метод-обработчик, который устанавливает в классе `Garage` новый `CurrentCarId`. После выбора автомобиля кнопка «Выбрать» сменяется на текст «Выбрано». Результат нажатия кнопки «Выбрать» представлен на рисунке 26.

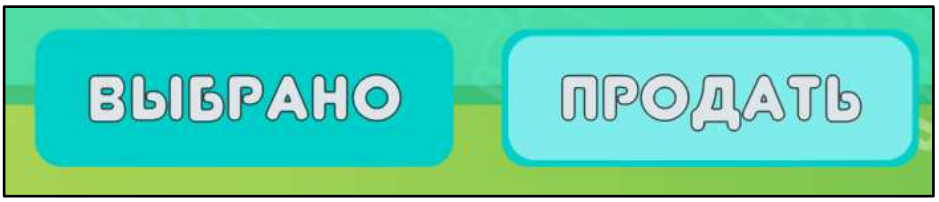

Рисунок 26 – Результат обработки нажатия кнопки «Выбрать»

Тюнинг

При нажатии на кнопку «Тюнинг» из сцены «Гараж» система запустит соответствующую сцену. Реализованная сцена «Тюнинг» представлена на рисунке 27.

Рисунок 27 – Сцена «Тюнинг» игры «No Limits»

Для удобной организации работы с тюнингом автомобиля был разработан класс `TuningService`, который оперирует конечным автоматом `TuningStateMachine`. Работа над отдельной частью технического аспекта автомобиля предоставлена специальному классу. При запуске сцены конечный автомат переходит в свое первое состояние `IdleState`. При нажатии на кнопку «Двигатель» конечный автомат переходит в состояние `ChangingEngineState`, которое отвечает за изменение мощности и максимальной скорости автомобиля. При нажатии на кнопку «Ходовая часть» конечный автомат переходит в состояние `ChangingChassisState`, которое отвечает за изменение сцепления с трассой. При нажатии на кнопку «Назад» конечный автомат возвращается в `IdleState`, а при выходе из сцены «Тюнинг» завершает свою работу.

При переходе в состояния `ChangingEngineState` или `ChangingChassisState` открывается панель с доступными улучшениями. При нажатии на элемент на панели улучшений игрок увидит изменение технических характеристик. Улучшения для двигателя и ходовой части представлены объектами `ScriptableObject`. Они хранят в себе название

улучшения, цену улучшения и множитель улучшения. Пример заполнения объекта для улучшения двигателя представлен на рисунке 28.

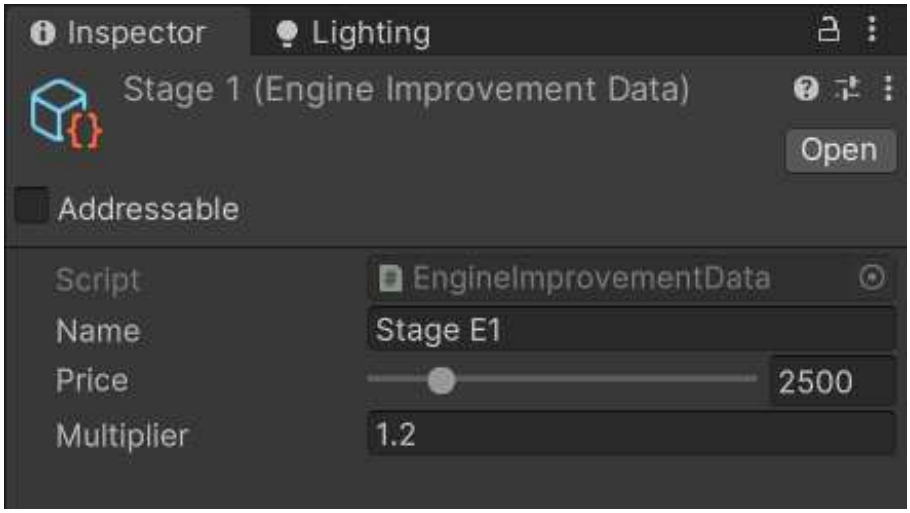

Рисунок 28 – Улучшение для двигателя

При переходе в состояние, отвечающее за улучшение, открывается панель с доступными улучшениями. Панель выбора улучшений для двигателя представлена на рисунке 29.

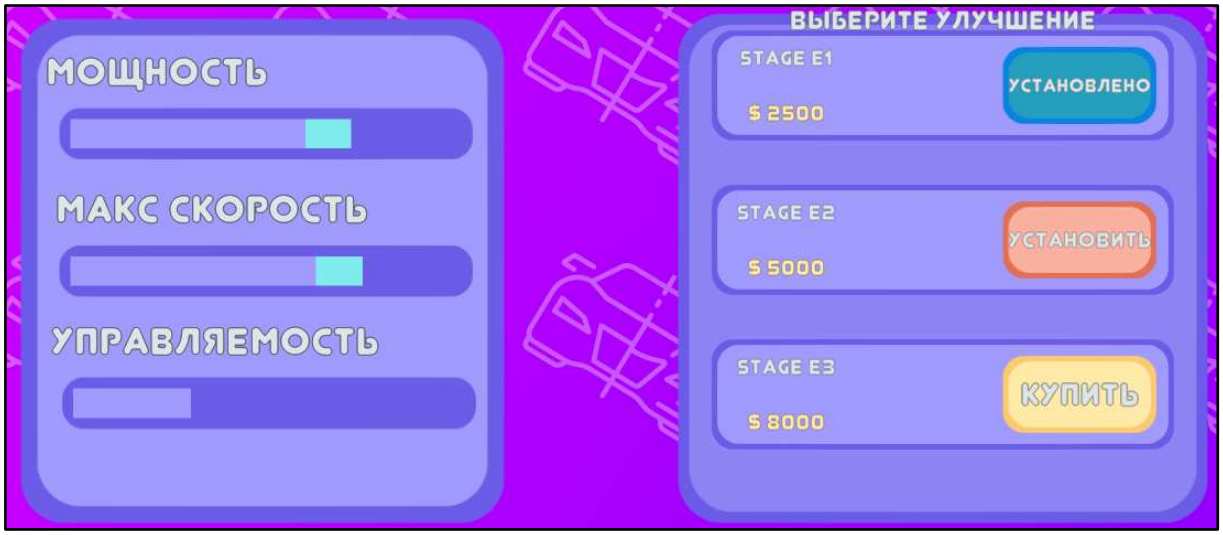

Рисунок 29 – Тюнинг двигателя

При выборе улучшения отображаются будущие изменения в технических характеристиках автомобиля. Если улучшение доступно для покупки, и у игрока хватает средств для его покупки, то он может купить и одновременно установить улучшение. Установленное улучшение помечено

текстом «Установлено». Купленное улучшение игрок может установить, нажав на кнопку «Установить».

Код, отвечающий за обработку нажатий кнопок «Купить» и «Установить» представлен в листинге 6.

Листинг 6 – Покупка и установка улучшений

```
Car currentCar =
_progress.Progress.Garage.GetCurrentCar(_progress.Progress.Garage.CurrentCa
rId);
currentCar.Specs = _specs;
currentCar.InstalledEngineImprovement = name;
if (!currentCar.PurchasedEngineImprovements.Contains(name)){
    currentCar.PurchasedEngineImprovements.Add(name);
}
currentCar.Specs.SetNewEnginePower(multiplier);
currentCar.Specs.SetNewMaxSpeed(multiplier);
```

При покупке или установке улучшений обновляется поле `Specs` класса `Car`, которое хранит технические характеристики автомобиля. Также необходимо обновить список купленных улучшений и обновить значение текущего установленного улучшения.

Стайлинг

При нажатии на кнопку «Стайлинг» из сцены «Гараж» система запустит соответствующую сцену, представленную на рисунке 30.

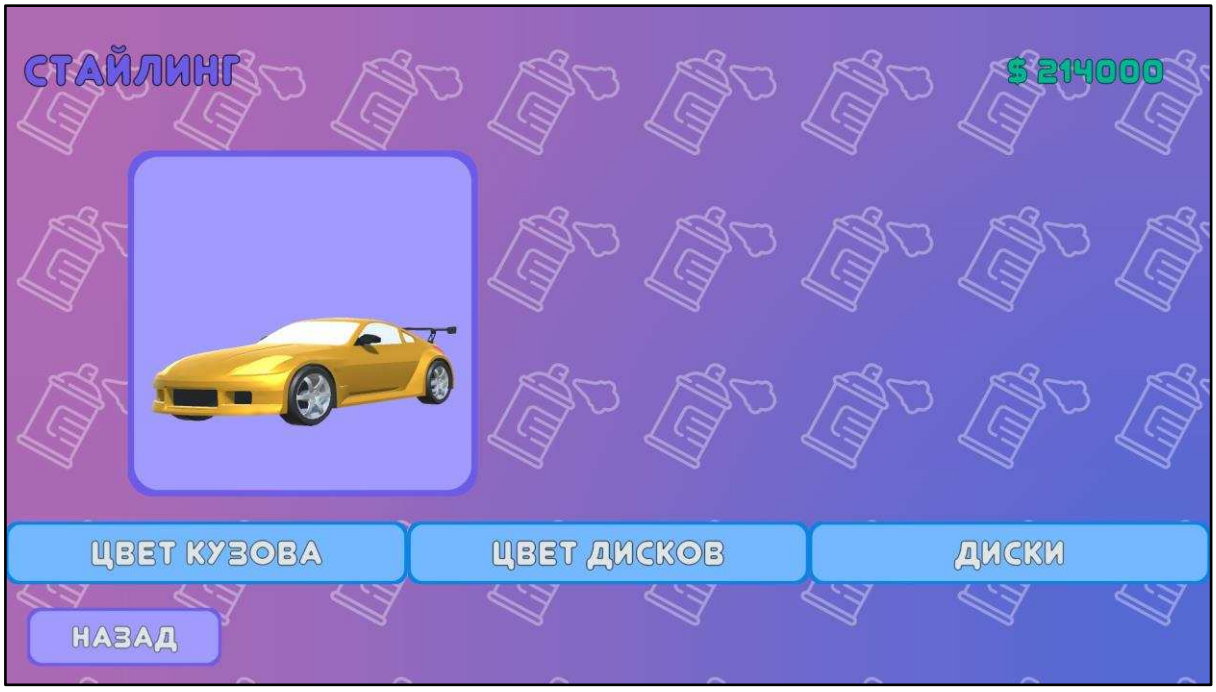

Рисунок 30 – Сцена «Стайлинг» игры «No Limits»

Изменение дисков реализовано аналогично покупки улучшений на сцене «Тюнинг». Отличия в реализации имеются при смене цвета кузова и смене цвета дисков.

При смене цвета кузова или дисков открывается палитра для выбора нового цвета. Игрок на палитре выбирает понравившийся ему цвет, все изменения отображаются на автомобиле игрока. Для отображения просматриваемого цвета при изменении цвета на палитре система получает компонент `BodyColor` автомобиля в случае изменения цвета кузова, и `RimColor` в случае изменения цвета дисков и устанавливает цвет материала. Если игрок отказался от покупки нового цвета и нажал на кнопку «Назад» цвет сбрасывается к установленному до применения изменений. При покупке нового цвета этот цвет сохраняется в прогрессе. При этом игрок не может купить уже установленный цвет, и он не может купить новый цвет, если ему не хватает количества игровой валюты. Изменение цвета кузова представлено на рисунке 31, на котором с помощью палитры выбирается новый цвет кузова для текущего автомобиля игрока.

Рисунок 31 – Изменение цвета кузова автомобиля

Покупка колесных дисков представлена на рисунке 32, на котором покупаются колеса «350Z» и «RX7», при этом устанавливаются на автомобиль игрока именно колесные диски «RX7».

Рисунок 32 – Изменение колесных дисков

Покупка нового цвета на колесных диск представлена на рисунке 33, на котором с помощью палитры меняется цвет колесных дисков с белого цвета на серый цвет.

Рисунок 33 – Изменение цвета колесных дисков автомобиля

Выбор заездов

При нажатии на кнопку «Заезда» из сцены «Гараж» система запустит соответствующую сцену. На сцене игрок может выбрать доступный ему заезд. Панель заезда отображает его название, продолжительность заезда, личный рекорд игрока, цели и награды. Реализованная сцена представлена на рисунке 34.

Рисунок 34 – Сцена «Заезды» игры «No Limits»

Для хранения данных о заездах был разработан `ScriptableObject` класс `Track`. На рисунке 35 представлен пример заполнения данных заезда: название, цель, структура ранжирования наград и длительность заезда.

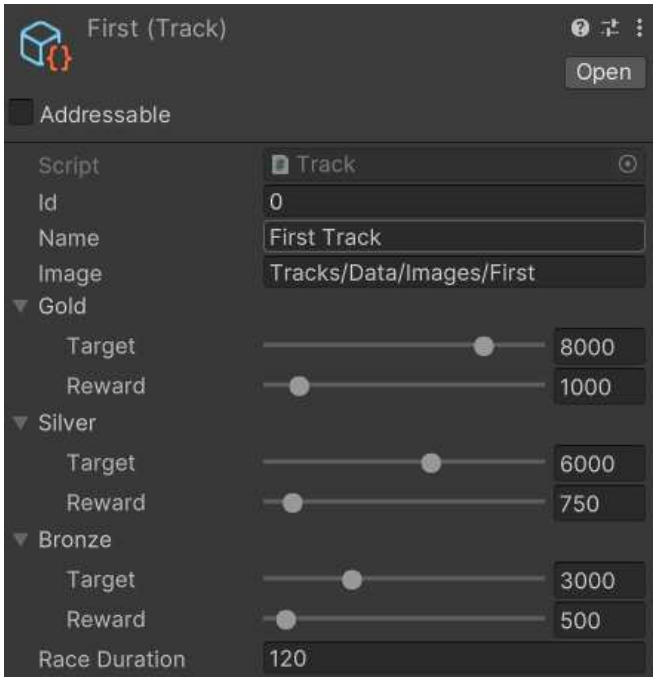

Рисунок 35 – `Track` для заезда «FirstTrack»

Для отображения информации о заезде был разработан класс `TrackWindow`, который получает значение из объекта класса `Track` и отображает значения на экране. Создание `TrackWindow` занимается класс `TrackFactory`, который принимает все объекты класса `Track` и получая данные о `RaceProgress` создает все панели. В случае, если заезд для игрока недоступен, на панели заезда будет отображено изображение замка, а кнопка запускается заезда будет недоступна. Недоступный заезд представлен на рисунке 36.

Рисунок 36 – Недоступный заезд

Основная игровая сцена

При нажатии на панель заезда запускается выбранный игроком заезд. Для организации и управления разными фазами заезда был разработан конечный автомат, который может находиться в нескольких состояниях: в стартовом состоянии, которое управляет началом заезда, в игровом состоянии, которое управляет основным игровым процессом, в состоянии паузы, которое управляет игрой во время паузы, в замедленном состоянии, которое управляет игрой после того, как игрок закроет меню паузы, и в состоянии результатов, которое отображает результат заезда игрока. Конечный автомат представлен классом `RaceStateMachine`. Диаграмма состояний разработанного конечного автомата представлена на рисунке 37.

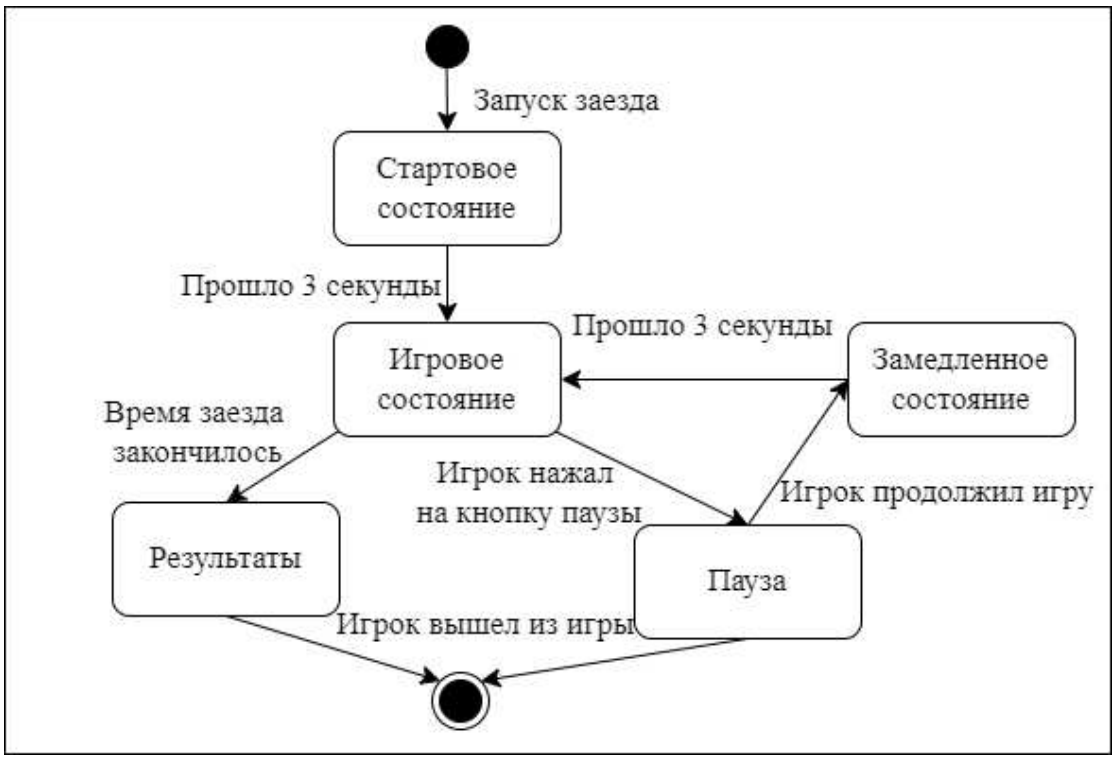

Рисунок 37 – Диаграмма состояний конечного автомата заезда

Стартовое состояние управляет обратным отсчетом до начала игры. По прошествии трех секунд конечный автомат переходит в новое состояние игры. Стартовое состояние представлено классом `StartState`. Начало заезда представлено на рисунке 38.

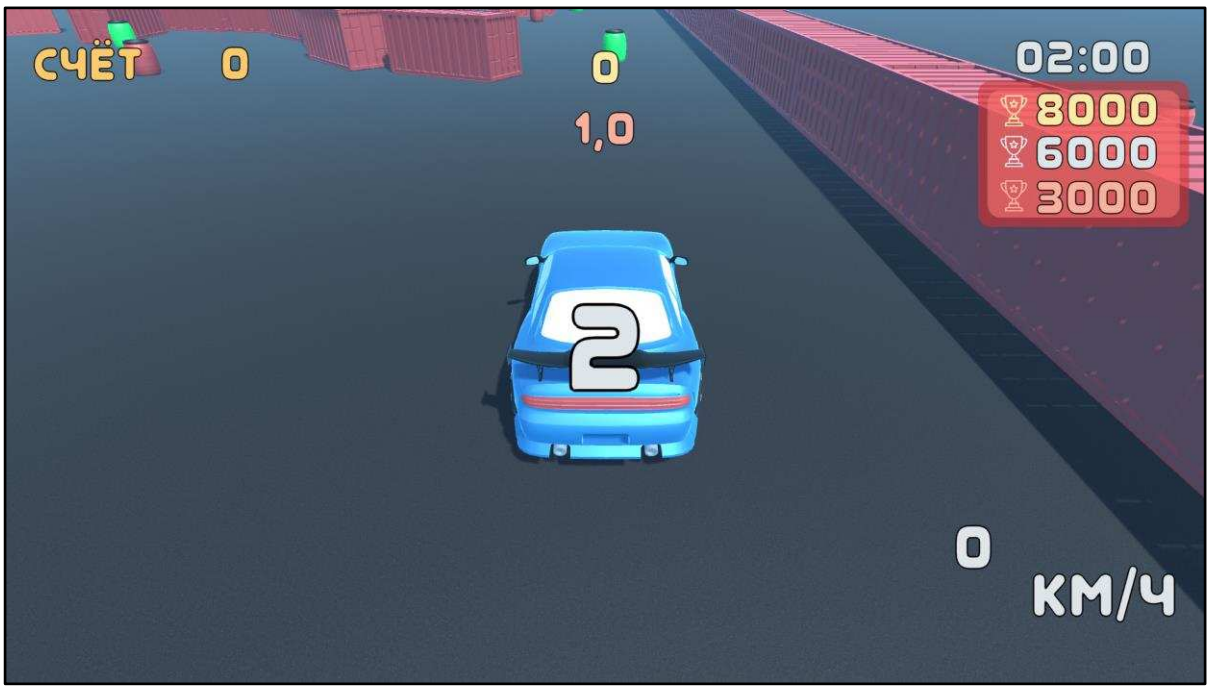

Рисунок 38 – Начало заезда

Игровое состояние обрабатывает ввод пользователя, рассчитывает угол дрифта, скорость движения игрока, отвечает за набор очков, обрабатывает столкновение игрока с объектами, управляет множителем очков. Из игрового состояния конечный автомат может перейти в состояние паузы или состояние результатов. Игровое состояние представлено классом `GameState`. Игровой процесс представлен на рисунке 39.

Рисунок 39 – Игровой процесс

На сцене заезда отображается текущий счет игрока, набирающиеся очки игрока, множитель заноса, бонусные и штрафные очки, оставшееся время, панель с целями для заезда и скорость игрока в километрах в час.

За счет очков и множителя очков отвечают классы `Score` и `Multiplier` соответственно. Класс `Score` реализует интерфейс `IScore` с которым работает `GameState`. Интерфейс предоставляет методы `Add`, `AddBonus`, `SubtractPenalty`, `UpdateNoScoreTime`. Реализация метода `Add` добавляет к общему счету заезда количество очков, набранных игроком за один дрифт. Методы `AddBonus` и `SubtractPenalty` вызываются соответственно при сбитии зеленого или красного конуса. Метод `UpdateNoScoreTime` отслеживает время, когда игрок больше не набирает очки. Класс `Multiplier`

реализует интерфейс `IMultiplier` с которым работает класс `GameState`. Метод `DecayMultiplier` уменьшает значение множителя очков в тот момент, когда игрок больше не набирает очков. Метод `UpdateMultiplier` считает множитель очков в зависимости от скорости и угла движения игрока.

Значение угла заноса рассчитывается как угол между вектором скорости автомобиля и вектором, который направлен вперед относительно автомобиля. Направления двух векторов представлено на рисунке 40.

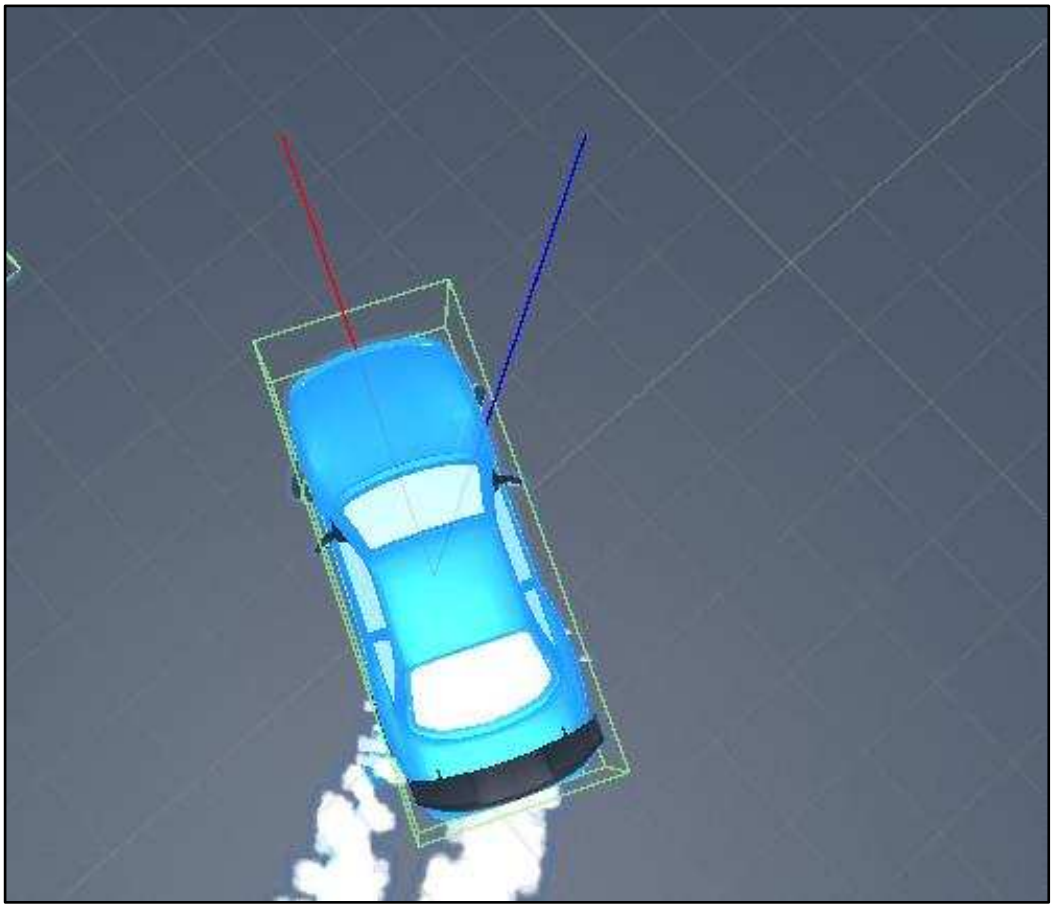

Рисунок 40 – Направление векторов для расчета угла заноса

Красный луч – направление вперед, относительно автомобиля. Синий луч – направление скорости движения автомобиля. Расчет угла заноса представлен в листинге 7.

Листинг 7 – Расчет угла заноса

```
Vector3 velocityDirection = _car.velocity.normalized;
Vector3 forwardDirection = _car.transform.forward;
float angle = Vector3.Angle(velocityDirection, forwardDirection);
```

Скорость движения объекта можно получить обращением к полю `velocity` компонента `Rigidbody`. Скорость движения необходимо перевести из м/с в км/ч, умножив значение `velocity` на 3,6.

Конечный автомат переходит в состояние паузы, если игрок нажал на клавишу «Esc» во время игрового состояния. Состояние останавливает игровое время и отвечает за показ меню паузы. Во время паузы игрок может выйти из игры, вернуться к игровому процессу или запустить заезд заново. Состояние паузы представлено классом `PauseState`. Пауза представлена на рисунке 41.

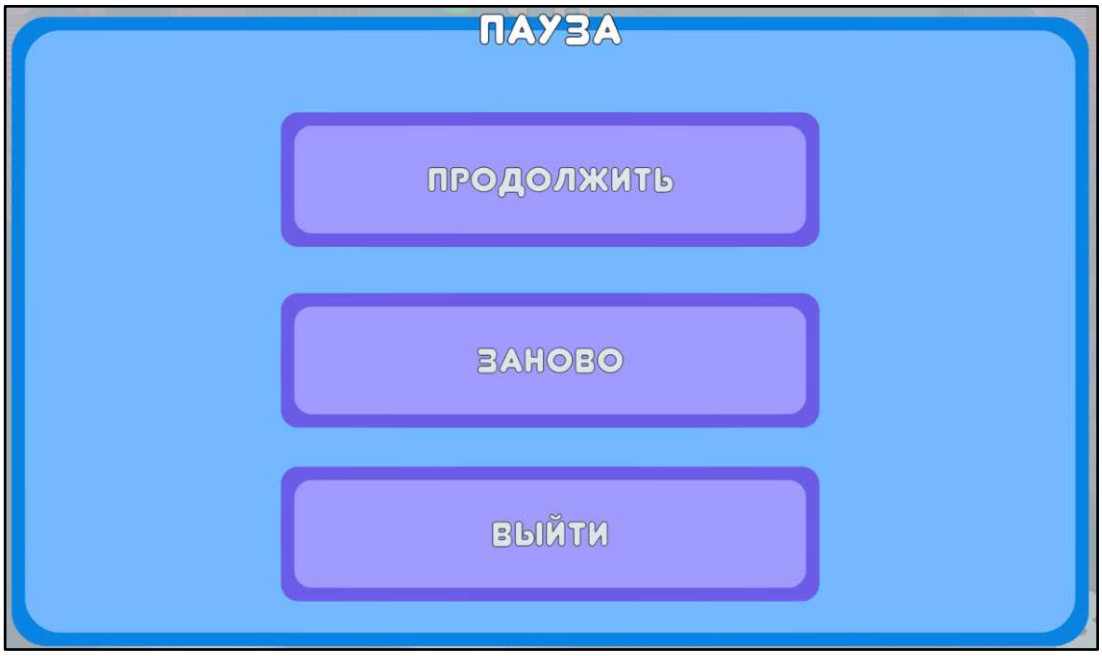

Рисунок 41 – Меню паузы

Конечный автомат переходит в состояние замедленной игры, при возвращении игрока к игровому процессу. Состояние предназначено для того, чтобы игрок успел приготовиться к игровому процессу после выхода из меню паузы. После трех секунд конечный автомат переходит вновь в игровое состояние. Состояние замедленной игры представлено классом `SlowMotionState`.

Конечный автомат переходит в состояние результатов заезда после того, как закончится время, отведенное на заезд. Состояние отвечает за сохранение результата заезда и отображение окна с результатами. Состояние

результатов заезда представлено классом `ResultsState`. Окно с результатами представлено на рисунке 42.

Рисунок 42 – Результаты заезда

3.6. Реализация управления

Для моделирования транспортных средств разработчики Unity встроили `WheelCollider` – специальный вид коллайдера, который имитирует пружину подвески и модель трения шин. Для управления колесом `WheelCollider` предоставляет свойство `motorTorque` для имитации ускорения автомобиля, свойство `brakeTorque` для имитации торможения, свойство `steerAngle` для имитации поворота. Важным параметром `WheelCollider` является `sidewaysFriction`, который хранит свойства трения шин в боковом направлении. Изменение параметров этого свойства позволяет настроить поведение автомобиля в заносе. Пример настройки `WheelCollider` представлен на рисунке 43.

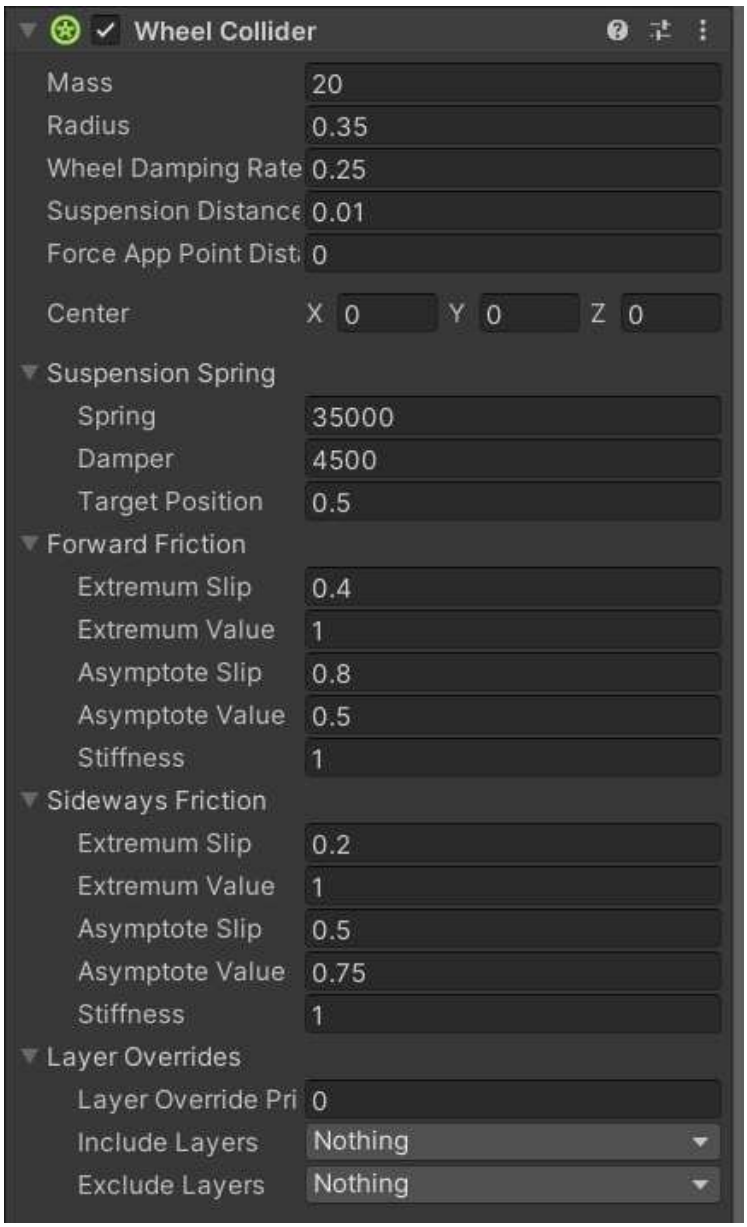

Рисунок 43 – Пример заполнения компонента `WheelCollider`

Для связи ввода пользователя и `WheelCollider` был разработан класс `CarController`, который получает компонент `WheelCollider` каждого из четырех колес автомобиля и использует их свойства `motorTorque`, `brakeTorque` и `steeringAngle` для управления автомобилем.

В листинге 8 представлено использование свойства `motorTorque` для имитации работы двигателя.

Листинг 8 – Использование свойства `motorTorque`

```
if (currentSpeed < _specs.CurrentMaxSpeed){
    _colliders.RRWheel.motorTorque = _specs.CurrentEnginePower *
        gasInput;
```

```
    _colliders.RLWheel.motorTorque = _specs.CurrentEnginePower *
        gasInput;
}
else
{
    _colliders.RRWheel.motorTorque = 0;
    _colliders.RLWheel.motorTorque = 0;
}
```

Если автомобиль игрока движется меньше максимальной возможной скорости для данного автомобиля, то в зависимости от ввода игрока устанавливается крутящий момент на задние колеса автомобиля. Если игрок движется на максимальной скорости, то крутящий момент устанавливается на 0, чтобы предотвратить дальнейшее ускорение автомобиля. Переменная `gasInput` может принимать значения от 0 до 1, в зависимости от ввода игрока.

Использование свойства `brakeTorque` для имитации торможения представлено в листинге 9.

Листинг 9 – Использование `brakeTorque`

```
_colliders.FRWheel.brakeTorque = brakeInput * _specs.BrakePower *
    _frontWheelsBrakeFactor;
_colliders.FLWheel.brakeTorque = brakeInput * _specs.BrakePower *
    _frontWheelsBrakeFactor;

_colliders.RRWheel.brakeTorque = brakeInput * _specs.BrakePower *
    _rearWheelsBrakeFactor;
_colliders.RLWheel.brakeTorque = brakeInput * _specs.BrakePower *
    _rearWheelsBrakeFactor;
```

Если игрок нажимает кнопку торможения, то для `WheelCollider` устанавливается новое значение свойства `brakeTorque`, вследствие чего происходит торможение автомобиля. При этом `brakeTorque` для передний колес устанавливается больше, чем для задних колес, чтобы сымитировать реалистичное торможение.

Использование свойства `steerAngle` представлено в листинге 10.

Листинг 10 – Использование свойства `steerAngle`

```
float steeringAngle = steeringInput * factor;

if (slipAngle < 120f)
 {
    steeringAngle += Vector3.SignedAngle(transform.forward, velocity +
        transform.forward, Vector3.up);
```

```
}

steeringAngle = Mathf.Clamp(steeringAngle, -90f, 90f);
_colliders.FRWheel.steerAngle = steeringAngle;
_colliders.FLWheel.steerAngle = steeringAngle;
```

Угол поворота колес определяется вводом пользователя и углом скольжения автомобиля. Основной проблемой использования `WheelCollider` является то, что автомобиль имеет тенденцию разворачиваться на 180 градусов при попытке поворота на небольшой скорости, поэтому угол поворота передних колес необходимо контролировать в зависимости от угла скольжения автомобиля, чтобы обеспечить более точное управление автомобилем. Ограничение угла поворота с помощью метода `Mathf.Clamp` предотвращает чрезмерное поворачивание колес.

Вывод по третьей главе

В соответствии с функциональными и нефункциональными требованиями было реализовано игровое приложение на платформе Unity. Были реализованы все компоненты игрового приложения и создан пользовательский интерфейс.

Разработанное игровое приложение полностью соответствует требованиям.

4. ТЕСТИРОВАНИЕ

Для тестирования игрового приложения проводилось тестирование, которое заключается в том, чтобы убедиться, что разработанное приложение работает в соответствии со сформулированными функциональными требованиями.

4.1. Функциональное тестирование

Тестирование покупки автомобилей

В ходе тестирования пользовательского интерфейса сцены «Автосалон» были протестированы кнопки «Купить» и кнопки переключения автомобилей. Поведение кнопок соответствует разработанным требованиям. Результаты функционального тестирования покупки автомобилей представлены в таблице 2.

Таблица 2 – Тестирование покупки автомобилей

№	Аспект работы	Действия	Результат	Тест пройден?
1	Покупка автомобиля, на который хватает игровой валюты	1. Перейти на сцену покупки автомобиля. 2. Купить автомобиль, на который хватает средств. 3. Перейти на сцену с выбором автомобилей.	В гараже игрока появился купленный автомобиль, количество игровой валюты корректно.	Да
2	Покупка автомобиля, на который не хватает игровой валюты	1. Перейти на сцену покупки автомобиля. 2. Попытаться купить автомобиль, на который не хватает средств.	Кнопка покупки автомобиля неактивна.	Да
3	Покупка двух одинаковых автомобилей	1. Перейти на сцену покупки автомобиля. 2. Купить автомобиль, на который хватает средств. 3. Купить второй автомобиль, на который хватает средств. 4. Перейти на сцену с выбором автомобилей.	В гараже игрока появились два купленных автомобиля, количество игровой валюты корректно.	Да

Тестирование выбора и продажи автомобилей

В ходе тестирования пользовательского интерфейса сцены «Гараж» были протестированы кнопки «Выбрать», «Продать» и кнопки переключения между автомобилями. Поведение кнопок соответствует разработанным требованиям. Результаты функционального тестирования выбора и продажи автомобилей представлены в таблице 3.

Таблица 3 – Тестирование выбора и продажи автомобилей

№	Аспект работы	Действия	Результат	Тест пройден?
1	Выбор автомобиля	1. Перейти на сцену выбора и продажи автомобилей. 2. Нажать кнопку «Выбрать».	Выбранный автомобиль стал текущим у игрока, кнопка «Выбрать» стала недоступной.	Да
2	Продажа выбранного автомобиля	1. Перейти на сцену выбора и продажи автомобилей. 2. Нажать кнопку «Выбрать». 3. Нажать кнопку «Продать»	Выбранный автомобиль пропал из гаража игрока, значение игровой валюты увеличилось на значение цены автомобиля, текущим автомобилем игрока стал следующий автомобиль в гараже.	Да
3	Продажа любого невыбранного автомобиля	1. Перейти на сцену выбора и продажи автомобилей. 2. Нажать кнопку «Выбрать». 3. Переключиться на следующий автомобиль. 4. Нажать кнопку «Продать».	Выбранный автомобиль пропал из гаража игрока, значение игровой валюты увеличилось на значение цены автомобиля.	Да
4	Продажа выбранного автомобиля, который находится на последней позиции в гараже	1. Перейти на сцену выбора и продажи автомобилей. 2. Переключиться на последний автомобиль 3. Нажать кнопку «Выбрать». 4. Нажать кнопку «Продать»	Выбранный автомобиль пропал из гаража игрока, значение игровой валюты увеличилось на значение цены автомобиля, текущим автомобилем игрока стал предыдущий автомобиль в гараже.	Да
5	Продажа единственного автомобиля	1. Перейти на сцену выбора и продажи автомобилей 2. Нажать кнопку «Продать»	Ничего не произошло, кнопка «Продать» недоступна.	Да

Тестирование выбора заездов

В ходе тестирования пользовательского интерфейса сцены «Выбор заездов» были протестированы кнопки запуска заезда и кнопки переключения между заездами. Результаты функционального тестирования выбора заездов представлены в таблице 4.

Таблица 4 – Тестирование выбора заездов

№	Аспект работы	Действия	Результат	Тест пройден?
1	Выбрать доступный заезд	1. Перейти на сцену выбора заездов. 2. Запустить первый заезд	Запустилась сцена с выбранным заездом.	Да
2	Выбрать недоступный заезд	1. Перейти на сцену выбора заездов. 2. Переключиться на первый недоступный заезд. 3. Запустить заезд.	Ничего не произошло. Заезд недоступен для запуска.	Да

Тестирование основного игрового процесса

В ходе тестирования пользовательского интерфейса сцены «Основная игровая сцена» были протестированы элементы пользовательского интерфейса, такие как, вывод очков на экран, вывод набирающихся очков, вывод множителя, вывод цели игрока, скорости автомобиля, оставшегося времени. Все отображаемые значения корректны. Результаты функционального тестирования основного игрового процесса представлены в таблице 5.

Таблица 5 – Тестирование основного игрового процесса

№	Аспект работы	Действия	Результат	Тест пройден?
1	Счет очков	1. Начать заезд. 2. Пустить автомобиль в занос.	Очки насчитываются в соответствии со скоростью и углом движения игрока.	Да
2	Счет множителя	1. Начать заезд. 2. Пустить автомобиль в занос.	Множитель насчитывается в соответствии со скоростью и углом движения игрока.	Да

Окончание таблицы 5

№	Аспект работы	Действия	Результат	Тест пройден?
3	Получение бонусных очков	1. Начать заезд. 2. Пустить автомобиль в занос. 3. Сбить зеленый конус.	К набирающимся очкам игрока прибавились бонусные очки.	Да
4	Получение штрафных очков	1. Начать заезд. 2. Пустить автомобиль в занос. 3. Сбить красный конус.	Из набирающихся очков отнялись штрафные очки.	Да
5	Запуск и начало заезда	1. Начать заезд.	Перед передачей управления игроку автомобилем таймер отсчитал 3 секунды.	Да
6	Постановка на паузу и снятие с паузы	1. Начать заезд. 2. Нажать клавишу «Ecs». 3. Выйти из паузы.	Во время паузы игровое время остановилось. После снятия паузы игровое время было замедленно в течение трех секунд.	Да
7	Сброс очков	1. Начать заезд. 2. Пустить автомобиль в занос. 3. Столкнуться со статическим объектом.	Набирающиеся очки обнулились.	Да

Тестирование тюнинга

В ходе тестирования пользовательского интерфейса сцены «Тюнинг» были протестированы кнопки «Ходовая часть», «Двигатель», панель для выбора новых улучшений. При нажатии на кнопки корректно открывается панель с доступными улучшениями. При покупке новых улучшений кнопка «Купить» сменяется кнопкой «Установить». При установке кнопка сменяется на «Установлено». Корректно отражаются текущие характеристики автомобиля, и корректно отображаются будущие характеристики автомобиля при покупке нового улучшения.

Результаты функционального тестирования тюнинга представлены в таблице 6.

Таблица 6 – Тестирование тюнинга

№	Аспект работы	Действия	Результат	Тест пройден?
1	Купить новое улучшение	1. Запустить сцену «Тюнинг». 2. Выбрать новое улучшение для автомобиля. 3. Нажать кнопку «Купить».	Новое улучшение установилось на автомобиль, значение характеристик автомобиля корректно, значение игровой валюты корректно.	Да
2	Установить ранее купленное улучшение	1. Запустить сцену «Тюнинг». 2. Выбрать ранее купленное улучшение для автомобиля. 3. Нажать кнопку «Установить».	Улучшение установлено на автомобиль, значение характеристик автомобиля корректно, значение игровой валюты корректно.	Да

Тестирование стайлинга

В ходе тестирования пользовательского интерфейса сцены «Стайлинг» были протестированы кнопки «Цвет кузова», «Цвет дисков», «Колесные диски», палитра для выбора цвета, панель для выбора новых колесных дисков. Результаты функционального тестирования стайлинга представлены в таблице 7.

Таблица 7 – Тестирование стайлинга

№	Аспект работы	Действия	Результат	Тест пройден?
1	Купить новый цвет для кузова	1. Перейти на сцену «Стайлинг». 2. Нажать кнопку «Цвет кузова». 3. Выбрать новый цвет на палитре. 4. Нажать кнопку «Купить».	Цвет автомобиля обновился в соответствии с выбранным цветом, значение игровой валюты корректно.	Да
2	Купить новый цвет для колесных дисков	1. Перейти на сцену «Стайлинг». 2. Нажать кнопку «Цвет дисков». 3. Выбрать новый цвет на палитре. 4. Нажать кнопку «Купить».	Цвет колесных дисков обновился в соответствии с выбранным цветом, значение игровой валюты корректно.	Да

Окончание таблицы 7

№	Аспект работы	Действия	Результат	Тест пройден?
3	Купить новые колесные диски	1. Перейти на сцену «Стайлинг». 2. Нажать кнопку «Колесные диски». 3. Выбрать новые колесные диски. 4. Нажать кнопку «Купить».	Вид колесных дисков обновился в соответствии с выбранными колесами, значение игровой валюты корректно.	Да

Тестирование изменения игровых настроек

В ходе тестирования пользовательского интерфейса сцены «Настройки» были протестированы кнопки «Применить», выпадающие меню для выбора настроек, слайдер для настройки звука и флаг для выбора режима экрана. Все элементы пользовательского интерфейса отображают корректные установленные значения настроек игры. Результаты функционального тестирования изменения игровых настроек представлены в таблице 8.

Таблица 8 – Тестирование изменения игровых настроек

№	Аспект работы	Действия	Результат	Тест пройден?
1	Изменить язык игры	1. Запустить сцену «Настройки». 2. Открыть выпадающее меню для выбора языка. 3. Сменить язык. 4. Нажать кнопку «Применить»	Текст пользовательского интерфейса поменялся на выбранный язык.	Да
2	Изменить громкость игры	1. Запустить сцену «Настройки». 2. Сменить значение уровня громкости. 3. Нажать кнопку «Применить»	Громкость игры корректна и соответствует установленному значению.	Да
3	Изменить качество графики	1. Запустить сцену «Настройки». 2. Открыть выпадающее меню для выбора. 3. Сменить качество графики. 4. Нажать кнопку «Применить».	Качество игры корректно и соответствует выбранному значению.	Да

Окончание таблицы 8

№	Аспект работы	Действия	Результат	Тест пройден?
4	Изменить разрешение экрана	1. Запустить сцену «Настройки». 2. Открыть выпадающее меню для выбора разрешения экрана. 3. Сменить разрешение экрана. 4. Нажать кнопку «Применить».	Разрешение игры корректно и соответствует выбранному значению.	Да
5	Переключить игру в оконный режим	1. Запустить сцену «Настройки». 2. Снять флаг «Полноэкранный режим». 3. Нажать кнопку «Применить».	Игра переключилась в оконный режим.	Да
6	Переключить игру в полноэкранный режим	1. Запустить сцену «Настройки». 2. Установить флаг «Полноэкранный режим». 3. Нажать кнопку «Применить».	Игра переключилась в полноэкранный режим.	Да

4.2. Юзабилити тестирование

В ходе юзабилити тестирования был изменен пользовательский интерфейс. Перед тестированием игрок не мог определить какой элемент улучшения стайлинга или тюнинга был установлен. Также при выборе нового автомобиле на сцене «Мои авто» игрок не мог определить, какой у него выбранный текущий автомобиль. После исправления кнопка «Установить» при нажатии становится недоступной, текст кнопки меняется на «Установлено» и меняется цвет кнопки. При выборе нового автомобиля кнопка «Выбрать» становится недоступной, и текст кнопки меняется на «Выбрано» с «Выбрать».

Помимо этого, важные изменения в пользовательском интерфейсе произошли на основной игровой сцене. Текст «осталось 10 секунд» был помещен по центру экрана. Был добавлен таймер, чтобы игрок мог ориентироваться, сколько у него осталось времени до конца заезда.

Также игровые объекты, такие как контейнеры, могли перекрывать элементы пользовательского интерфейса, например, текст, который отвечает за вывод скорости игрока. После исправления ошибок объекты больше не перекрывают пользовательский интерфейс.

Вдобавок было изменено положение кнопки «Назад» на всех сценах игры «No Limits». Изначально, согласно макетам, кнопка должна была располагаться в правом верхнем углу. После проведения тестирования было принято решение переместить ее в нижний левый угол, чтобы освободить место для вывода текущего баланса игровой валюты игрока.

Кроме изменений в пользовательском интерфейсе после юзабилити тестирования была переработана камера, которая следит за автомобилем игрока. Перед внесением изменений камера подергивалась, и портила впечатление от игрового процесса. Расположение камеры также зависело от скорости и угла движения автомобиля игрока, что приводило к тому, что камера слишком сильно отдалялась от автомобиля игрока. Все недостатки камеры были исправлены, что положительно повлияло на восприятие игрового процесса.

Вывод по четвертой главе

В ходе тестирования были протестированы все разработанные сцены игрового приложения. Помимо тестирования пользовательского интерфейса также был проведен ряд функциональных тестов для каждой из сцен. В общей сложности было разработано 28 тестов.

ЗАКЛЮЧЕНИЕ

В данной работе была разработана компьютерная игра «Дрифт» на платформе Unity. Код программы составил 5800 строк исходного кода на языке C#.

Основные результаты

В результате выполнения работы были получены следующие результаты.

1. Выполнен анализ предметной области.
2. Спроектировано игровое приложение.
3. Реализовано игровое приложение.
4. Протестировано игровое приложение.

Направление дальнейших исследований

Дальнейшим направлением работы над игровым приложением будет добавление нового наполнения, такого как добавление новых видов автомобилей, новых видов колесных дисков, новых игровых заездов.

Также планируется добавление новых игровых механик для тюнинга автомобилей игрока, новых игровых механик, связанных с дрифтом, например, выполнение различных трюков за бонусные очки.

Также направлением развития игрового приложения будет переработка модели движения автомобилей. Переход с встроенного механизма в Unity на собственную модель управления движением автомобиля.

ЛИТЕРАТУРА

1. Racing Player Consumer Insights: Demographics, Gaming Preferences, Behavior, Brand Attitudes, and More | Newzoo. [Электронный ресурс] URL: https://newzoo.com/resources/blog/racing-player-consumer-insights-demographics-gaming-preferences-behavior-brand-attitudes-and-more/ (дата обращения: 30.01.2024 г.).

2. On the Dynamics of Automobile Drifting. [Электронный ресурс] URL: https://saemobilus.sae.org/content/2006-01-1019/ (дата обращения: 30.01.2024 г.).

3. Боковое скольжение. Энциклопедия по машиностроению XXL. [Электронный ресурс] URL: https://mash-xxl.info/info/201865/ (дата обращения: 30.01.2024 г.).

4. Правила дрифта | drift.by. [Электронный ресурс] URL: https://drift.by/pravila-drifta/ (дата обращения: 30.01.2024 г.).

5. Apperley T.H. Genre and game studies: Toward a critical approach to video game genres. // Simulation & Gaming, 2006. – № 37(1). – 6–23 pp.

6. Что такое гонки | igrasan | ru. [Электронный ресурс] URL: https://igrasan.ru/gonki/ (дата обращения: 30.01.2024 г.).

7. На пределе скорости. История жанра гонки. [Электронный ресурс] URL: https://igroray.ru/journal/na-predele-skorosti-istoriya-zhanra-gonki/ (дата обращения: 30.01.2024 г.).

8. Что такое Аркады | igrasan | ru. [Электронный ресурс] URL: https://igrasan.ru/chto-takoe-arkady/ (дата обращения: 30.01.2024 г.).

9. CarX Drift Racing Online в Steam. [Электронный ресурс] URL: https://store.steampowered.com/app/635260/CarX_Drift_Racing_Online/ (дата обращения: 09.02.2024 г.).

10. CarX Drift Racing Online Steam stats – Video Game Insights. [Электронный ресурс] URL: https://vginsights.com/game/635260 (дата обращения: 09.02.2024 г.).

11. Torque Drift в Steam. [Электронный ресурс] URL: https://store.steampowered.com/app/1029550/Torque_Drift/ (дата обращения: 09.02.2024 г.).

12. How many copies did Need for Speed sell? – 2024 statistics | LEVVVEL. [Электронный ресурс] URL: https://levvvel.com/need-for-speed-statistics/ (дата обращения: 11.02.2024 г.).

13. Weekly Rentrak game rental chart: December 1-7 – GameSpot. [Электронный ресурс] URL: https://www.gamespot.com/articles/weekly-rentrak-game-rental-chart-december-1-7/1100-6085756/ (дата обращения: 11.02.2024 г.).

14. Need for Speed: Underground for All – Sales, Wiki, Release Dates, Review, Cheats, Walkthrough. [Электронный ресурс] URL: https://www.vgchartz.com/game/229665/need-for-speed-carbon/ (дата обращения: 11.02.2024 г.).

15. Need for Speed: Carbon for All – Sales, Wiki, Release Dates, Review, Cheats, Walkthrough. [Электронный ресурс] URL: https://www.vgchartz.com/game/229665/need-for-speed-carbon/ (дата обращения: 11.02.2024 г.).

16. Need for Speed Unbound – Official Site. [Электронный ресурс] URL: https://www.ea.com/games/need-for-speed/need-for-speed-unbound?isLocalized=true (дата обращения: 11.02.2024 г.).

17. Need for Speed Undound: Sales Soar to 14$ Million in First Month on Steam. [Электронный ресурс] URL: https://game-sensor.info/news/need_for_speed_unbound_sales (дата обращения: 11.02.2024 г.).

18. Использование компонентов – Unity Manual. [Электронный ресурс] URL: https://docs.unity3d.com/ru/2019.4/Manual/UsingComponents.html (дата обращения: 26.03.2024 г.).

19. Unity – Scripting API: MonoBehaviour. [Электронный ресурс] URL: https://docs.unity3d.com/ScriptReference/MonoBehaviour.html (дата обращения: 26.03.2024 г.).

20. STYLIZED: Complete Drift Cars | 3D Land | Unity Asset Store. [Электронный ресурс] URL: https://assetstore.unity.com/packages/3d/vehicles/land/stylized-complete-drift-cars-193256 (дата обращения: 26.03.2024 г.).

21. Freesound. [Электронный ресурс] URL: freesound.org (дата обращения: 26.03.2024 г.).

22. A Better Architecture for Unity Games | TheGamedev.Guru. [Электронный ресурс] URL: https://thegamedev.guru/unity-architecture/a-better-architecture-for-unity-projects/ (дата обращения: 30.03.2024 г.).

23. Unity – Manual: Order of execution for event functions. [Электронный ресурс] URL: https://docs.unity3d.com/Manual/ExecutionOrder.html (дата обращения: 30.03.2024 г.).

24. Service Locator. Decoupling Patterns. Game Programming Patterns. [Электронный ресурс] URL: https://gameprogrammingpatterns.com/service-locator.html (дата обращения: 30.03.2024 г.).

25. Singleton. Desing Patterns Revisited. Game Programming Patterns. [Электронный ресурс] URL: https://game-programmingpatterns.com/singleton.html (дата обращения 30.03.2024 г.).

26. Quick Start Guide | Localization | 1.0.5. [Электронный ресурс] URL: https://docs.unity3d.com/Packages/com.unity.localization@1.0/manual/QuickStartGuide.html (дата обращения: 11.04.2024 г.).

27. Builder Pattern | Object Oriented Design. [Электронный ресурс] URL: https://www.oodesign.com/builder-pattern (дата обращения: 26.04.2024 г.).

28. Unity – Manual: ScriptableObject. [Электронный ресурс] URL: https://docs.unity3d.com/Manual/class-ScriptableObject.html (дата обращения: 02.05.2024 г.).