



Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии

ОТЧЕТ ПО УЧЕБНОМУ ПРАКТИКУМУ

Студент

подпись, дата

фамилия, и.о.

Ментор команды

подпись, дата

Жуков Д. М._____
фамилия, и.о.

Ментор команды

подпись, дата

Качалов А. Д._____
фамилия, и.о.

Руководитель практики

подпись, дата

Оленев А. А._____
фамилия, и.о.

2019г.

Оглавление

Оглавление	2
Введение	3
Аналитический раздел	3
Анализ существующих решений	3
Вывод и постановка задачи	7
Конструкторский раздел	8
Структура составляющих игры и элементов интерфейса	8
Система персонажей	9
Описание компонентов и их назначение	9
Система предметов	9
Описание компонентов и их назначение	9
Игровая физика	11
Классы заданий и скина персонажа	12
Класс менеджера карт	12
Диаграммы для интерфейсов	13
Главное меню	13
Меню создания персонажа	14
Задачи поставленные перед каждым участником	17
Технологический раздел	17
Python 3.7	17
Cocos2D	18
Методология разработки	18
Тестирование	18
Личный вклад	19
Организация рабочего процесса	19
Написание классов для системы игровых предметов	19
Написание вместе с остальными членами команды классов игрового ИИ	20
Заключение	21
Литература	21

Введение

Целью практики является получение навыков командной работы и закрепление знаний и умений, полученных в рамках курсов: «Программирование» и «Основы программной инженерии».

Команда состоит из пяти человек: Иван Аникин - разработчик, композитор; Коннов Константин - разработчик интерфейсов, дизайн уровней; Азимова Гахира - художник, разработчик интерфейсов; Елизарова Мария - разработчик, дизайн уровней; Куликов Георгий - руководитель команды, разработчик.

Задачей нашей команды было разработать игру в стиле «top-down shooter» со стилистикой «эстетики русских дворов». Игровой процесс представляет из себя выполнение основных и побочных заданий.

Для выполнения поставленной задачи был произведен анализ существующих решений.

Аналитический раздел

Анализ существующих решений

Анализ Hotline Miami:



Рис.1 - скриншот игры Hotline Miami

Hotline Miami — инди-игра, особенностями которой являются вид сверху, сюрреалистичный сюжет и стильный саундтрек. При работе над

исполнением игры разработчики вдохновлялись культурой восьмидесятых годов. Hotline Miami разделена на несколько глав, каждая из которых поделена на этапы. В начале каждой главы главный герой просыпается в своей квартире и прослушивает зашифрованные послания, присылаемые ему на автоответчик. В каждом сообщении протагониста просят проделать определённую, кажущуюся рутинной работу в указанном месте. Сообщения являются метафорой для реального задания — убить всех по упомянутому в сообщении адресу. Перед началом миссии герою предлагается надеть одну из множества звериных масок — каждая из них даёт игроку определённые особенности.

Плюсы:

1. Маски (главная фишка игры) — каждая маска даёт персонажу свои уникальные способности, что многократно разнообразивает тактику.
2. Разнообразие оружия — широкий выбор из различных видов рукопашного и огнестрельного оружия. Оружие нельзя перезаряжать, что побуждает к импровизации. Самое интересное, что найден баланс между холодным и огнестрельным.
3. Шикарный саундтрек.
4. Стильный визуал.
5. Интересный, но ненавязчивый сюжет.
6. Простор для тактики — бесшумное оружие, простреливаемые стены, нелинейность карт.

Спорное, специфика:

1. Система очков — за убийства, добивания и прочее вы получаете очки, как на аркадном автомате, в конце карты получаете оценку (А+, А, В тд).
2. Оружие не сохраняется между уровнями — из-за аркадного стиля каждая карта начинается с пустыми руками.
3. 1-выстрел-1-убийство — как вас, так и вашего противника можно убить с одного удара. С одной стороны это создаёт баланс между холодным и огнестрельным оружием, с другой стороны вы можете потерять всё из-за одного неверного шага.
4. Вы видите (сверху) сразу всю карту — противники не являются для вас неожиданностью.
5. 2 вида противников, не считая боссов — обычные противники и танки, выдерживающие по несколько выстрелов.
6. Весёлая кровавость.
7. На карте спавнится случайное оружие.

Минусы:

1. Плохой ИИ — противники просто патрулируют территорию, могут стоять на месте и разворота на 180 убить вас первой пулей.
2. Некоторые баги — кровь брызгает через стены, предметы, застревающие в дверях.
3. Длинные карты — неприятно, когда с другого конца коридора, который не помещается на экран, прилетает пуля.

Анализ Party Hard:

Party Hard - игра в стиле top-down с уклоном в игровую механику stealth. Состоит из десятка уровней. После выбора уровня некоторые его детали генерируются случайным образом. После загрузки игрок появляется у входа в здание либо комплекс зданий, в котором проходит вечеринка на 40-70 человек. Задача состоит в их полном уничтожении. Средства её выполнения: возможность главного героя орудовать ножом на близком расстоянии, большое количество ловушек (временного и мгновенного, зонального и точечного уничтожения). Препятствия: полицейские (вызываются участниками вечеринки при обнаружении трупа), охранники (имеют определённые траектории движения, при обнаружении игрока бегут за ним до конца их зоны и при поимке отправляют на перезагрузку), агенты ФБР (смесь охранников и полицейских, появляются только при убийстве полицейского, а также на последнем уровне игры).



рис.2 - скриншот из игры Party Hard

Плюсы:

1. Наличие 5 уникальных персонажей, которые помогают добавить разнообразия в игру
2. Хороший саундтрек
3. Множественные подходы к прохождению уровней

Спорное:

1. Stealth механика была весьма спорным решением для такого жанра, и хоть это добавляло разнообразия в тактику, динамику игры это убивало, из-за чего ты мог просидеть над одним уровнем очень много времени, без видимых на то причин.

2. Сюжет был хоть и не абсолютно неинтересным, но сказать, чтобы он затягивает тоже нельзя. Основная его часть состоит из диалогов детектива с неким Дариусом. Диалоги обычно идут о преступлениях и в конце нас ждет весьма очевидная для множества триллеров развязка, где Дариусом оказывается главный герой. Сюжет был явно не тем, на что разработчики сделали упор, однако многим он все-таки понравился и показался незаурядным.

Минусы:

1. Очень слабая пиксельная графика для игры 5-летней давности.
2. Плохой ИИ у полицейских, которые не понятно когда прекращают преследование, не понятно почему видят тебя сквозь стены и могут иногда арестовать безо всяких логических причин.
3. После первых двух - трех уровней играть становится очень скучно. Локации однообразны, всё то обилие персонажей, которое может предоставить игра, вы скорее всего не получите, потому что, пока вы откроете всех персонажей, игра вам надоест.

Анализ Synthetik: Legion Rising:

Synthetik: Legion Rising - игра в стиле top-down shooter с элементами roguelike в научно-фантастическом сеттинге. Персонаж отправляется покорять мир, захваченный машинами. В игре присутствует множество



рис.3 - скриншот игры Synthetik: Legion Rising

видов оружия и элементы «прокачки» персонажа. Сюжет игры таков: в 1985 году в производстве роботов лидирует Kaida Corporation. Сбросив оковы

многолетнего гнёта, ИИ корпорации создаёт "Легион машин" и начинает истреблять людей. Главным героем выступает андроид, наделённый человеческим сознанием, основной целью которого является спасти мир.

Плюсы:

1. Весьма яркая и красочная графика, проработанные локации, приятная стилистика.
2. Разнообразие оружия, механик, классов.
3. Неплохой саундтрек

Спорное:

1. Излишнее добавление сложности, путем усложнения процесса перезарядки у оружия, из-за чего ты чаще концентрируешься на том, как совершить перезарядку, чем на тактике и врагах. И хоть идея того, чтобы при перезарядке оставшиеся в обойме патроны должны пропадать, неплоха, однако превращать перезарядку в мини игру было через чур.
2. Система хитбоксов хоть и является неплохой, однако не предоставляет особого разнообразия и состоит из получения очков только за выстрелы в голову.

Минусы:

1. Абсолютная несбалансированность в игре. Одни боссы являются слишком слабыми, другие слишком сильными.
2. Малое количество боезапаса у некоторых видов оружия, что делает их абсолютно неиграбельными.
3. Множество бесполезных предметов
4. Произвольное появление оружия может сделать некоторые уровни просто непроходимыми, так как если вам выпало слабое по характеристикам оружие, то вы скорее всего не достигните успеха.
5. Несмотря на разнообразие классов, большое количество навыков являются не применимыми в реалиях игры.

Вывод и постановка задачи

Вдохновившись данными играми и увидев их минусы, наша команда решила, сохранив и объединив ряд несомненных достоинств данных игр, избавиться от недостатков.

Взяв из Hotline Miami динамичность и разнообразие игрового процесса, мы сделаем врагов более сложными для убийства и дадим возможность игроку сохранять оружие между картами.

Из Party Hard мы постараемся взять нелинейность прохождения игры, устранив недостатки, связанные с отсутствием разнообразия в игре.

Из Synthetik: Legion Rising мы постараемся сохранить множество подходов к игре, избавившись от несбалансированных противников.

Итак, основными недостатками игр являются: несбалансированность и отсутствие разнообразия в игровом процессе.

Поэтому наша команда посчитала, что необходимо создать сбалансированную игру, позаимствовав некоторые игровые механики из

жанра RPG и выбрав незаурядный сеттинг, дабы сделать игровой процесс более увлекательным.

Конструкторский раздел

Структура составляющих игры и элементов интерфейса

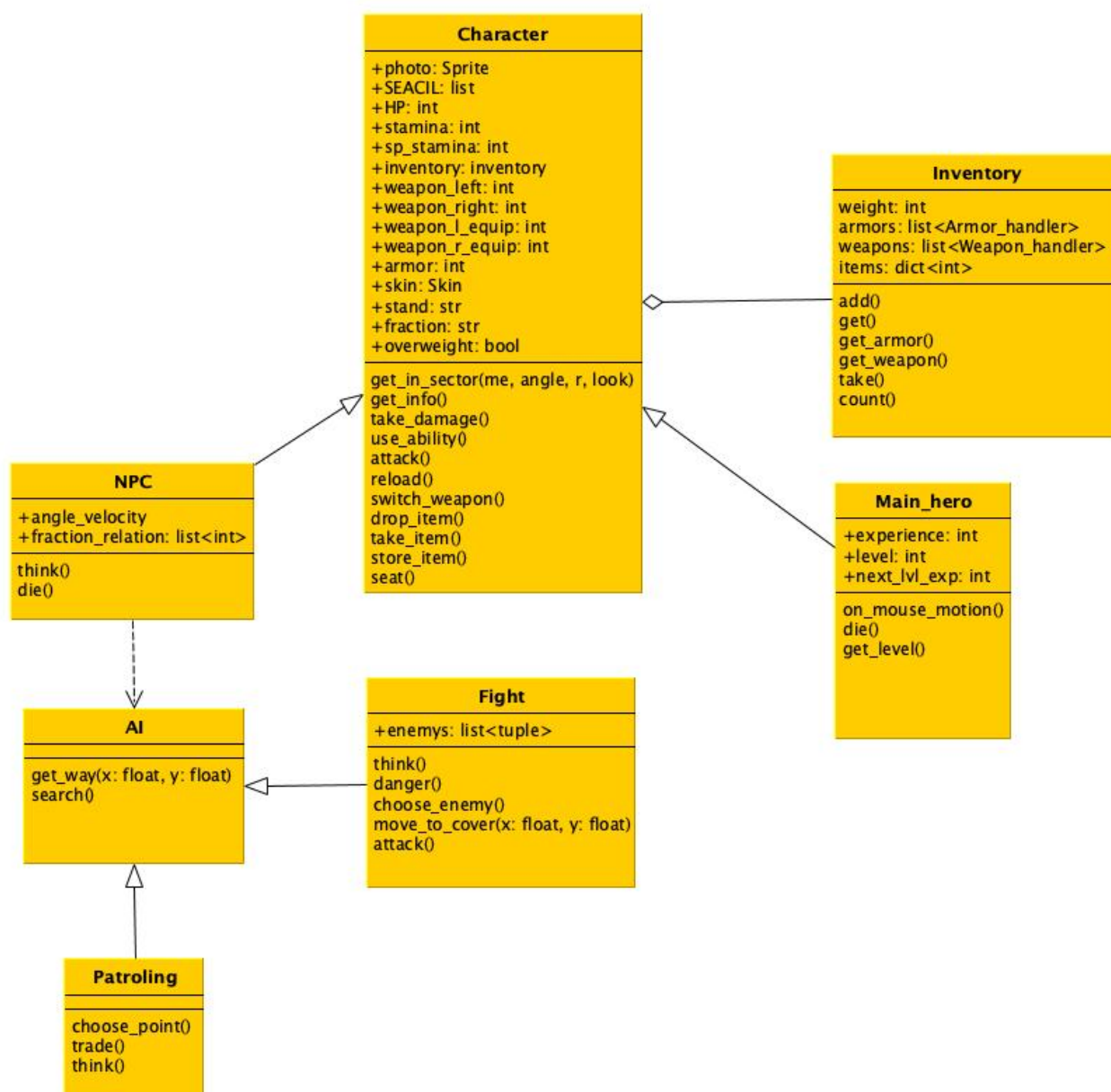


рис.4 - UML диаграмма персонажей)

Система персонажей

Описание компонентов и их назначение

Класс `character` является родительским классом для игрового и неигрового персонажей и создан для описания общих свойств этих двух типов персонажей. Включает в себя атрибуты для спрайта, массив характеристик персонажей, здоровье, выносливость, специальную выносливость для применения навыков, инвентарь, атрибуты, отвечающие за экипированное и спрятанное оружие и броню, скин персонажа, его положение, фракцию и флаг перевеса. Так же класс включает в себя методы получения информации об объектах, которые находятся в секторе видимости персонажа, получения урона, использования способности, перезарядки, смены оружия, выбрасывания оружия, поднятие оружия, хранения и смены положения.

Класс главного героя (`Main_hero`) наследуется от класса `character` и описывает свойства главного героя. Содержит в себе атрибуты, отвечающие за опыт, уровень и количество очков опыта для получения следующего уровня. И методы отвечающие, за смерть и получение уровня.

Класс неигрового персонажа наследуется от класса `character` и описывает свойств NPC, зависит от класса `AI`, который производит разделение состояний неигрового персонажа. У NPC может быть два состояния: патрулирование и бой. При патрулировании NPC сначала выбирает точку в зоне патруля, а затем идет к ней проверять окрестности, если обнаруживает персонажа, то проверяется взаимоотношения между персонажами и либо состояние сменяется на бой, либо продолжается патрулирование.

Класс `inventory` описывает инвентарь игрового и неигрового персонажей, содержит атрибуты отвечающие за вес, массив оружия и брони и массив обычных предметов, а так же содержит методы для взаимодействия с общими объектами инвентаря инвентаря.

Система предметов

Описание компонентов и их назначение

Система предметов в игре устроена по шаблону проектирования `Flyweight`, который мы решили использовать, дабы оптимизировать использование памяти, путем предотвращения создания экземпляров элементов, имеющих общую сущность. У нас присутствует общий класс предметов(`item`), от которого наследуются классы брони(`armor`), оружия(`weapon`) и используемых объектов(`usable_obj`). Данные классы характеризуют не конкретные объекты в инвентаре, а некую сущность, которая представляет из себя набор неизменных внутренних свойств и работы с ними. Классы же `armor_handler` и `weapon_handler` представляют из себя уже конкретные предметы, которые обладают рядом уникальных свойств.

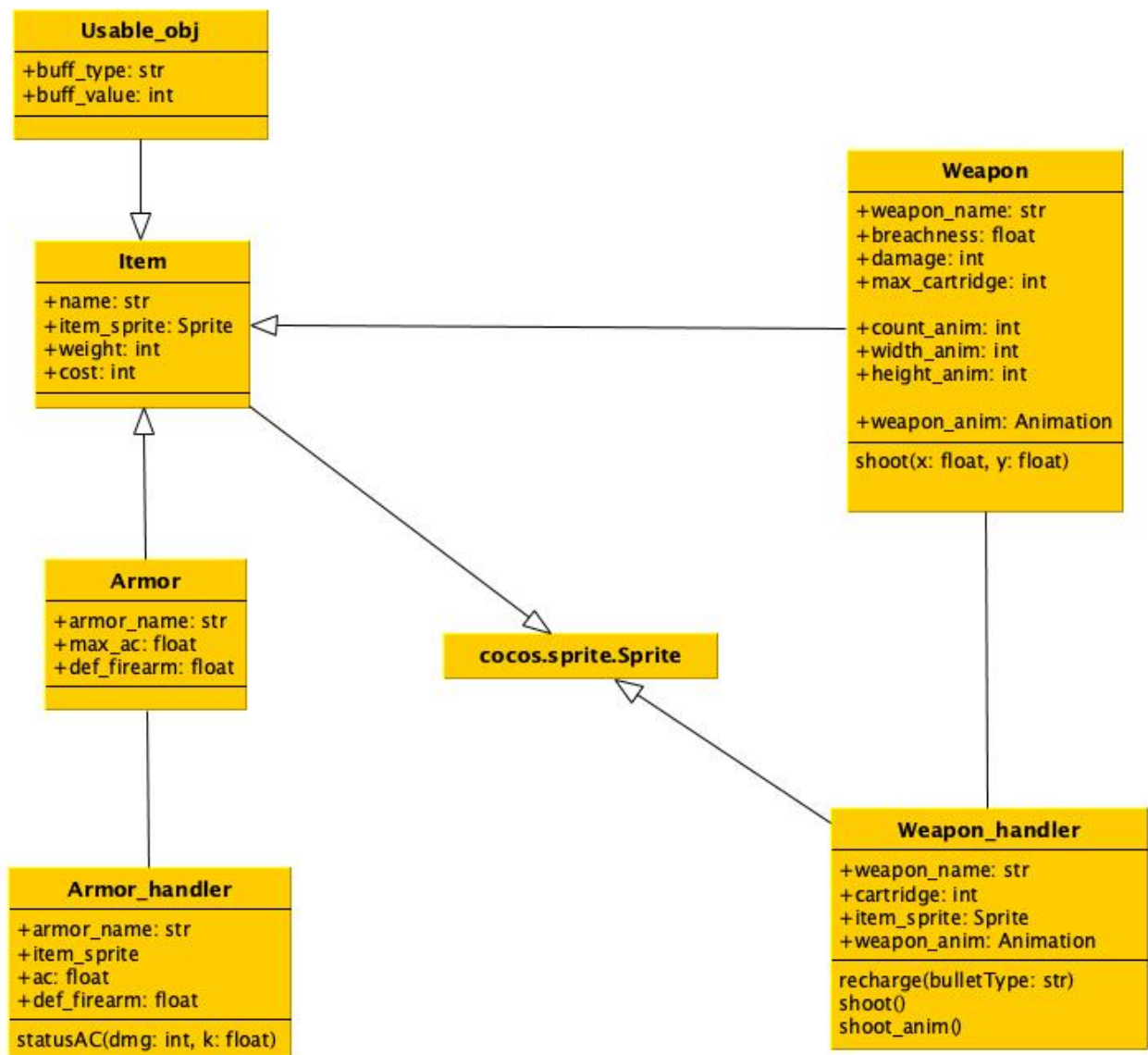


рис.5 - UML диаграмма системы предметов

Игровая физика

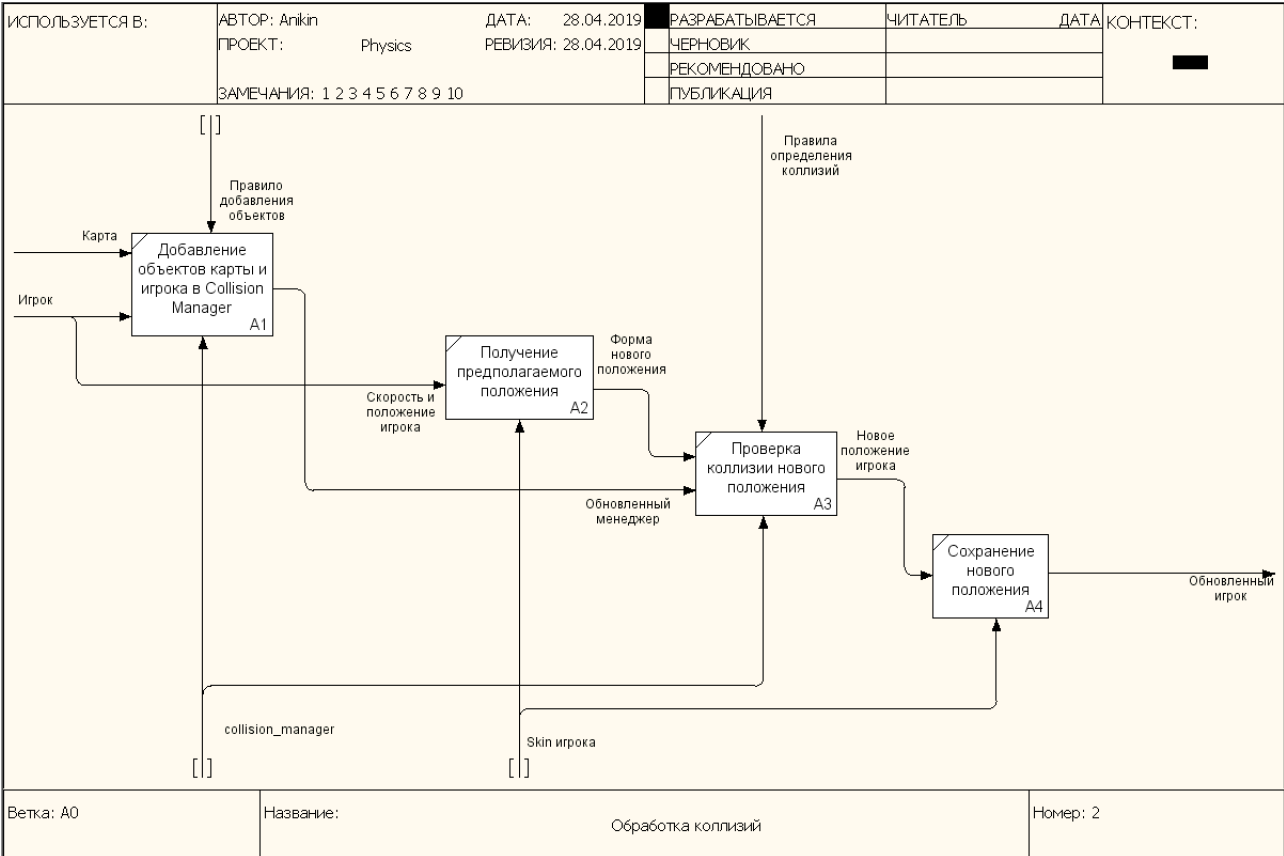
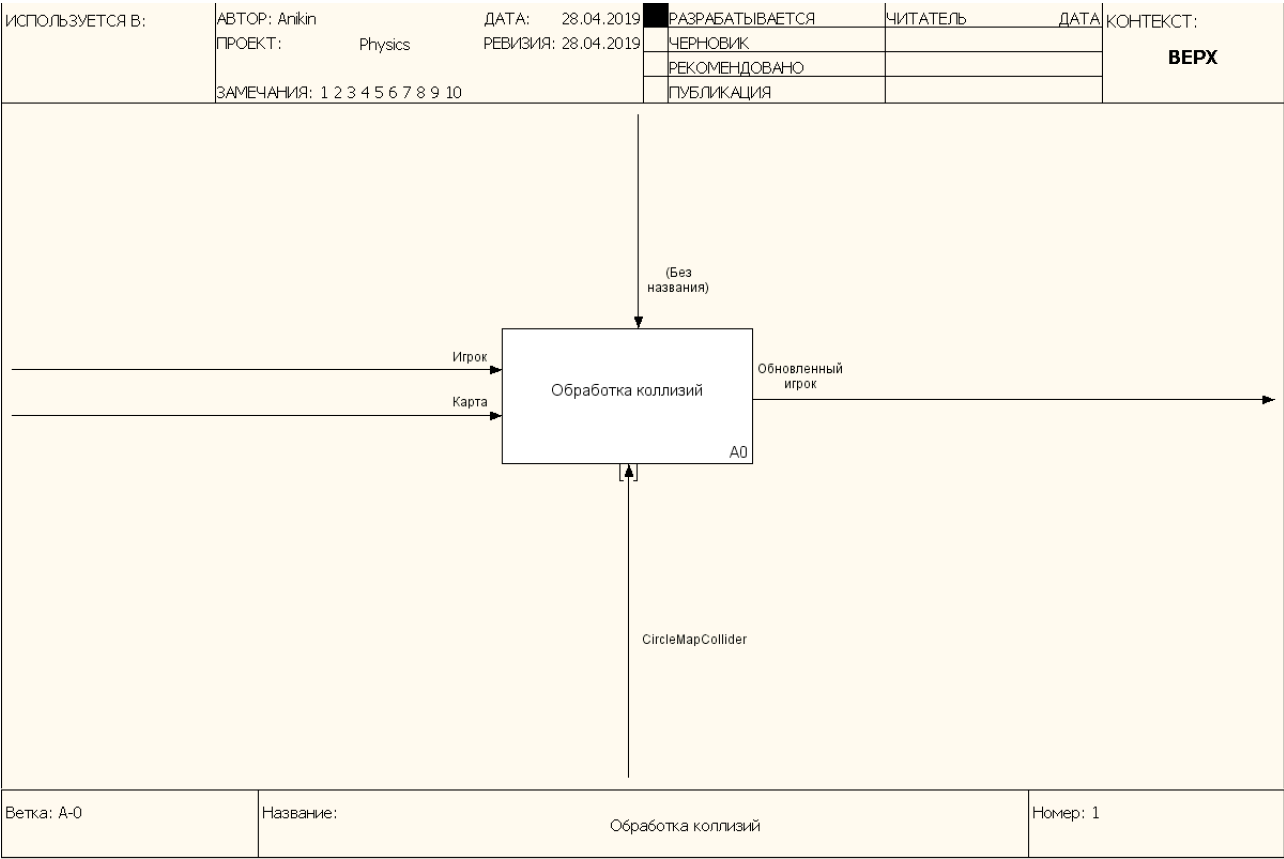


рис.6-7 - IDEF0 диаграмма игровой физики

Классы заданий и скина персонажа



рис.8 - UML диаграмма класса заданий

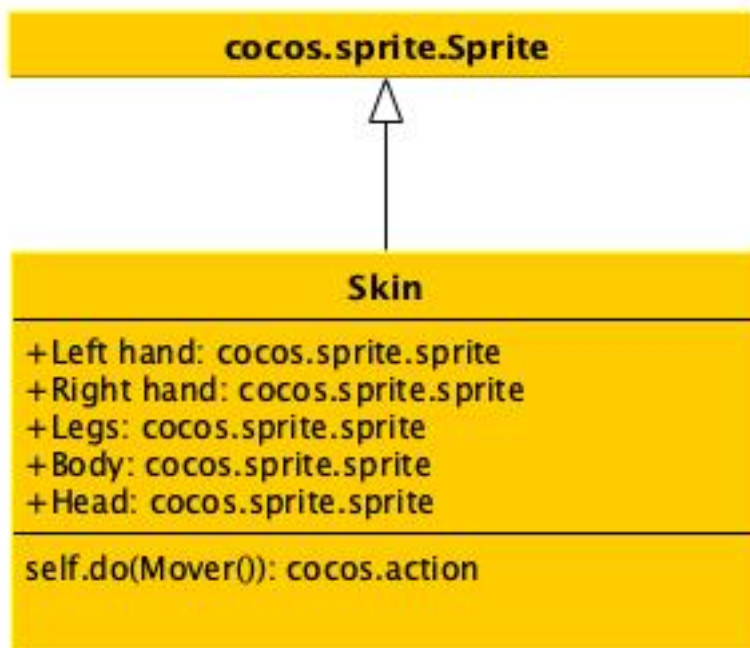


рис.9 - UML диаграмма класса скина

Класс менеджера карт

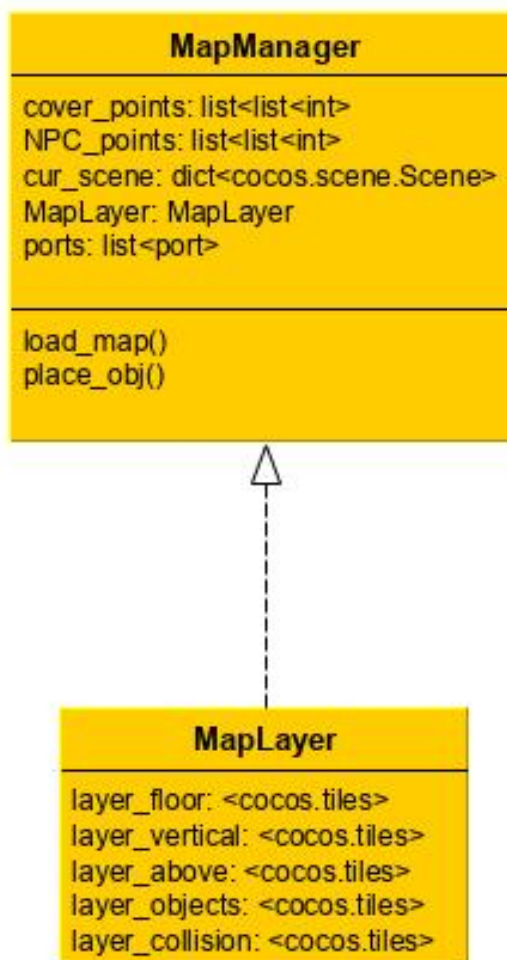
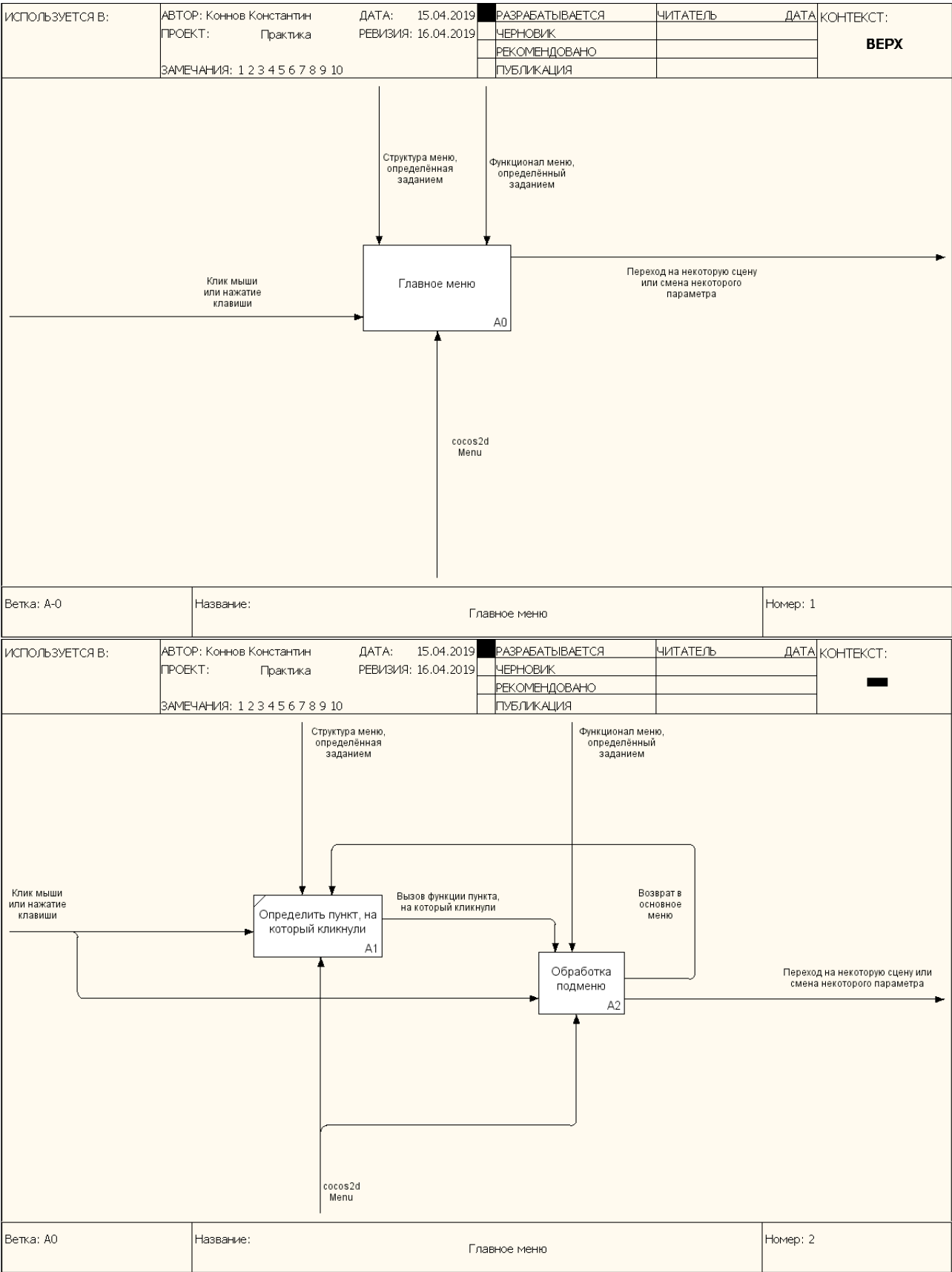


рис.10 - UML диаграмма класса менеджера карт

Диаграммы для интерфейсов

Главное меню



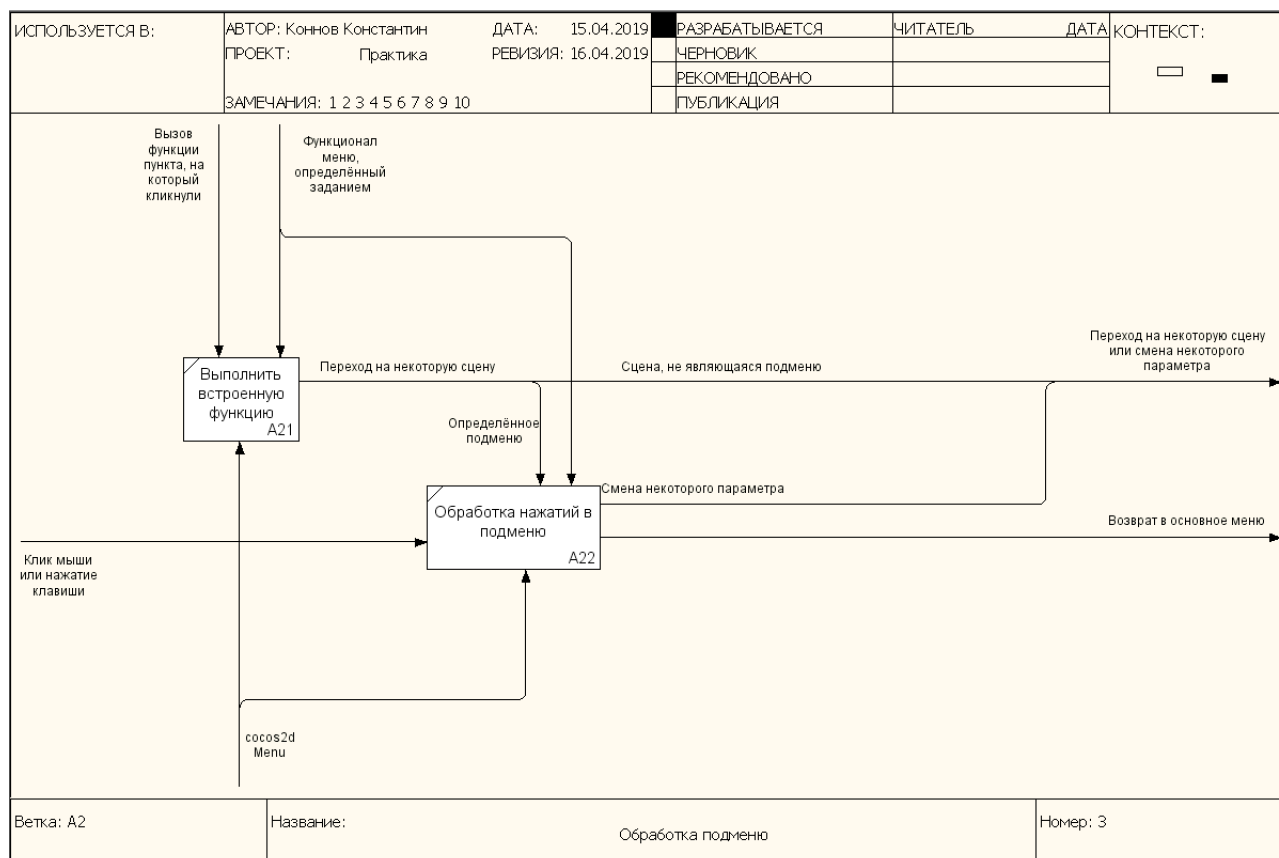
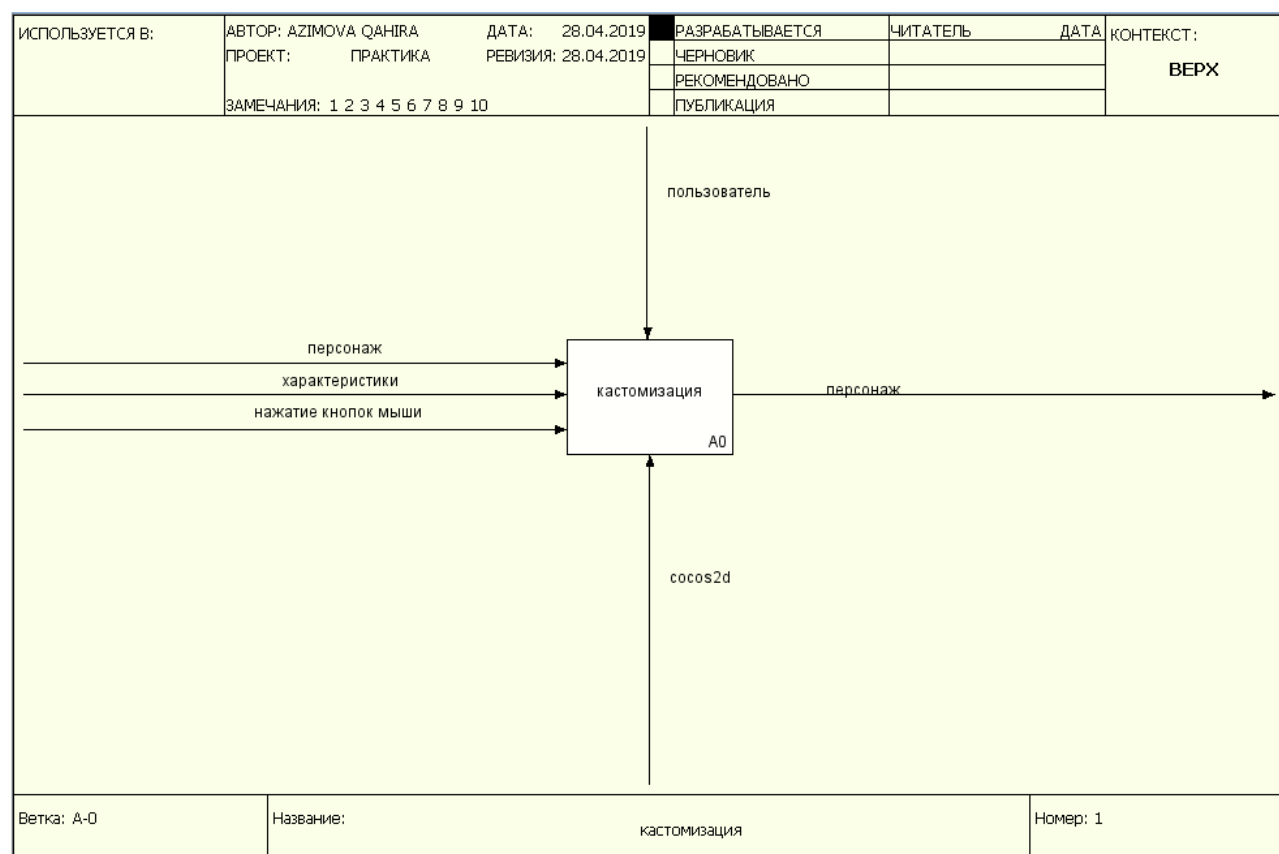
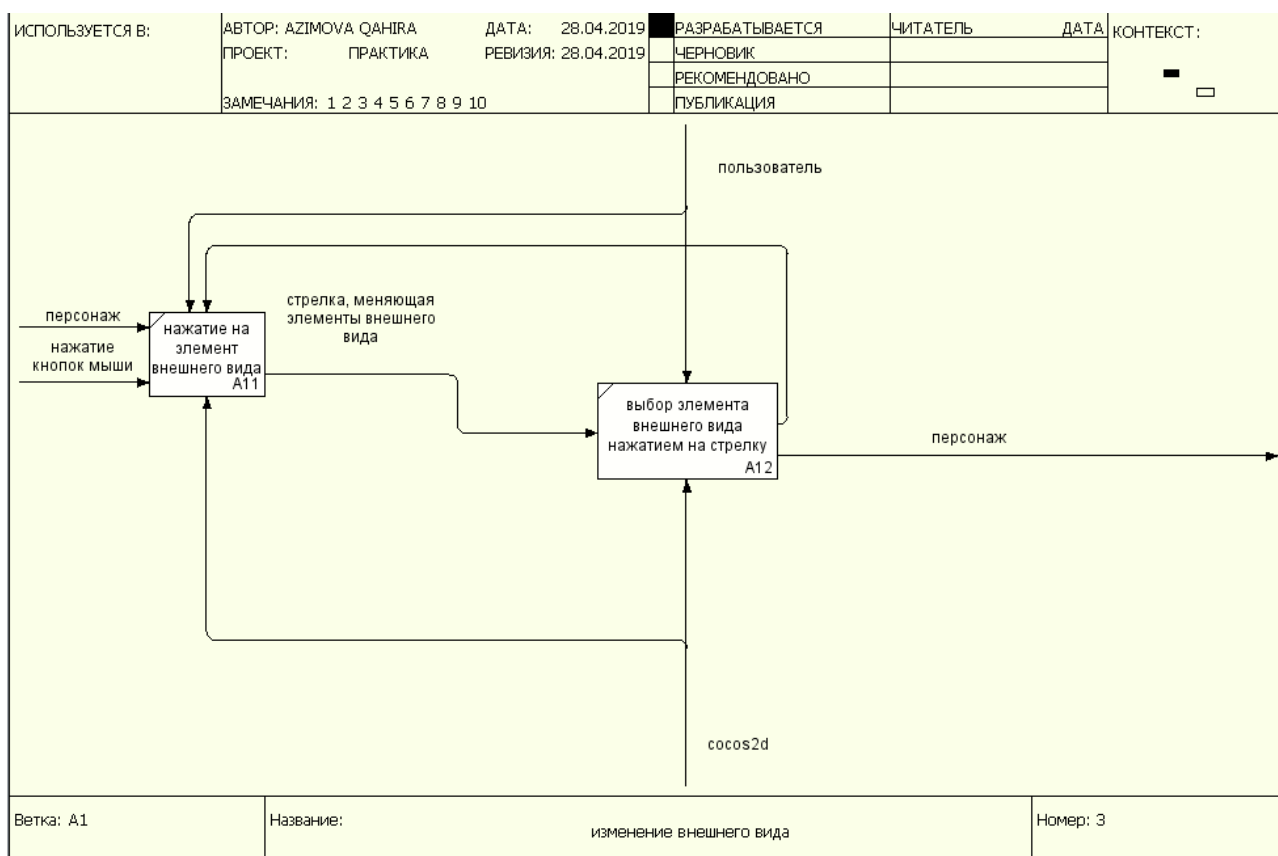
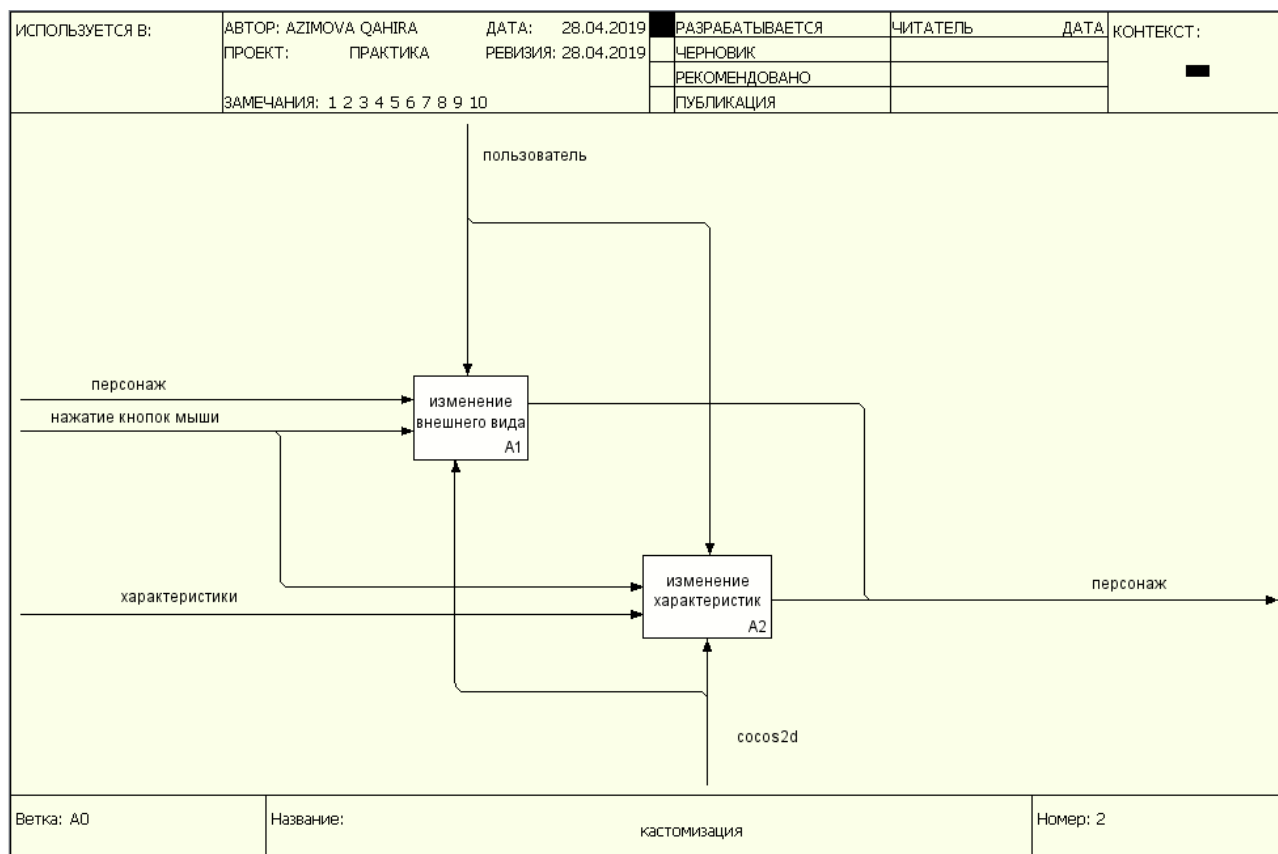


рис.11-13 - IDEF0 диаграмма главного меню

Меню создания персонажа





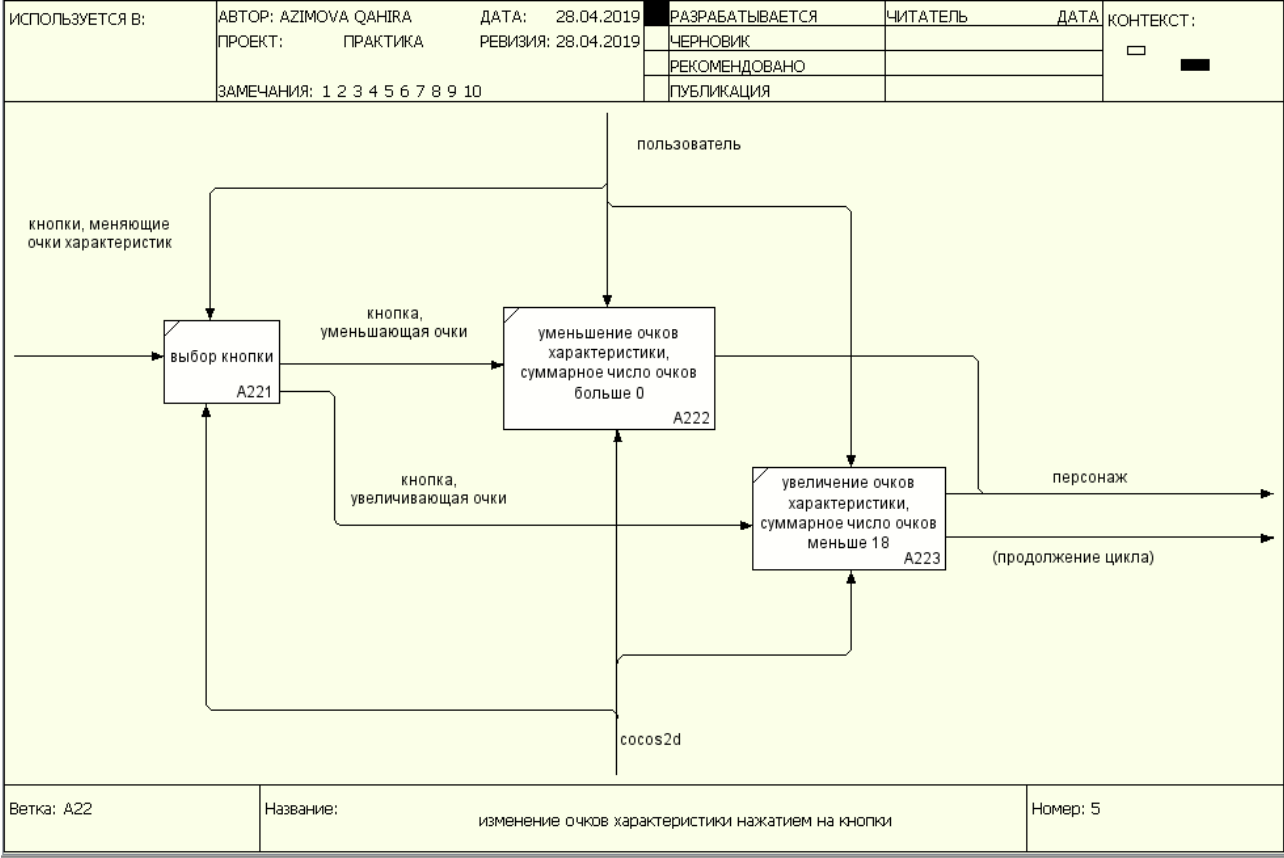
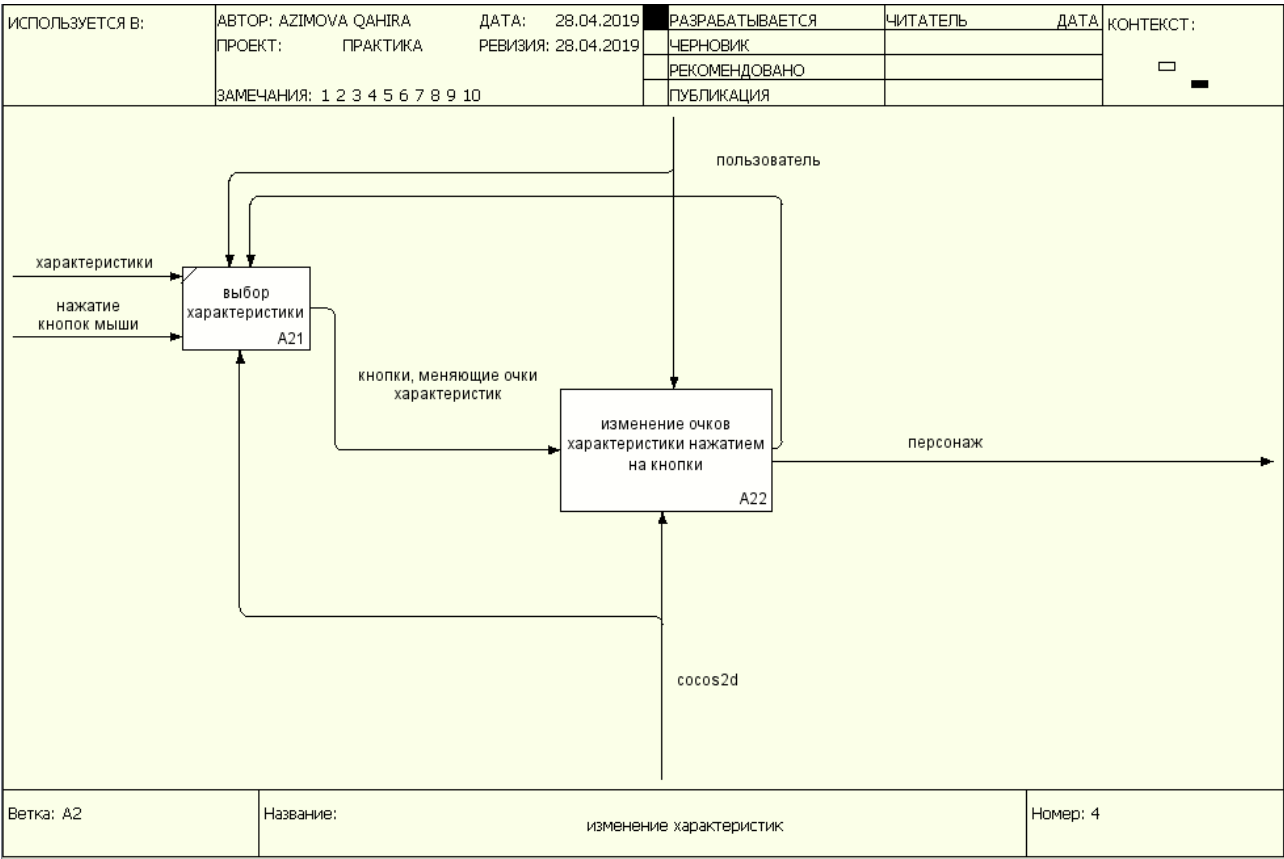


рис.14-18 - IDEF0 диаграмма меню создания персонажа

Задачи поставленные перед каждым участником

Коннов Константин: разработка классов персонажа, главного героя и интерфейса, разработка игрового ИИ

Аникин Иван: разработка игровой физики, разработка скина, написание саундтрека для игры, разработка игрового ИИ

Азимова Гахира: разработка меню создания персонажа, рисование моделей для игры

Елизарова Мария: разработка системы заданий, разработка систем перехода между картами, подгрузки карт и выставления объектов

Куликов Георгий: организация рабочего процесса в команде, разработка системы игровых предметов, разработка игрового ИИ

Технологический раздел

При реализации игры были использованы:

1. Язык программирования Python 3.7
2. Библиотека Cocos2D

Python 3.7

Python3 - высокоуровневый язык программирования с минималистичным синтаксисом, язык предоставляет возможность быстро писать код. Благодаря этому получилось в минимальные сроки начать реализацию проекта, используя уже знакомый язык программирования по курсу предыдущего семестра.

Cocos2D

В качестве игрового движка было решено использовать Cocos2d, основным качеством которого является его широкий функционал.

В Cocos2d присутствуют методы для описания столкновений, batch-отрисовки спрайтов (для оптимизации программы), работы с двухмерной анимацией, эффектов и переходов.

Также Cocos2d является кроссплатформенным движком, что позволяет собирать ПО для разных операционных систем. Не маловажным достоинством Cocos2d является и стабильность его API.

Cocos2d является открытым ПО с лицензированием MIT License, что означает, что данное программное обеспечение имеет открытый исходный код и распространяется с разрешительной лицензией, то есть позволяет программистам использовать лицензируемый код в закрытом ПО при условии, что текст лицензии предоставляется вместе с этим ПО.

Методология разработки

Методология разработки RAD была выбрана по следующим причинам:

1. Необходимость в сжатые сроки осуществить проект
2. На первых этапах проектирования, наличие идеи, но отсутствие четких требований к готовому продукту

Так же данная методология была выбрана из-за требования использовать в игровых проектах язык программирования Python и библиотеку cocos2D. Выбор данных технологий соответствует философии использования методологии RAD, так как одним из принципов является использование технологий, нацеленных на уменьшение времени разработки, а Python и cocos2D как раз такие технологии.

Работа была основана на постоянном создании прототипов, на основе которых итерационно строились следующие, стремясь удовлетворить условиям заказчика.

Тестирование

В ходе проведения практики производилось тестирование программного продукта. Было проведено ручное тестирование, методы и функции с особо сложными алгоритмами были покрыты тестированием черного и белого ящиков.

Ручное тестирование проводилось для выявления самых распространенных ошибок в играх: застревание в стене, прохождение

сквозь объекты и пр. В результате тестирования было выявлено несколько ошибок и впоследствии данные ошибки были устранены.

Методы добавления предметов в инвентарь, выбора случайной точки NPC, стрельбы и классы отвечающие за игровую физику были покрыты критическими тестами. В ходе тестирования были выявлены ошибки связанные, с неправильным расположением точки вылета пуль и выходом случайной точки для патрулирования за допустимы границы.

Личный вклад

В течении проекта в мои обязанности входило:

1. Организация рабочего процесса
2. Написание классов для системы игровых предметов
3. Написание вместе с остальными членами команды классов игрового ИИ

Организация рабочего процесса

В качестве руководителя команды мне необходимо было сплотить команду и корректно организовать работу, чтобы на каждого члена команды были распределены задачи. В качестве методологии разработки выбрал RAD. Так же я подготовил презентацию к защите первого этапа практики и составил общую часть итогового отчета.

Кроме этого я составлял и раздавал задания для каждого члена команды, подбирал теоретический материал и проверял сделанную работу. В случае малой эффективности какого-то члена команды, я старался выяснять, в чем проблема, и помогать человеку с её решением. Так же при необходимости, если член команды обращался за более подробным разъяснением задания, я проводил консультацию, где мы разбирали само задание и возможные способы решения.

По итогу могу сказать, что руководить командой так, чтобы все участники проекта работали эффективно, задача непростая, хоть и весьма интересная. Так как недостаточно сформировать четкое задание, обсудить тонкости и возможные решения, зная при всем при этом теоретическую базу. Ведь в данном аспекте процесса разработки речь идет о взаимодействии с людьми, поэтому необходимо понимать нужды и переживания человека, обладать склонностью к «психологизму» и быть терпеливым, не забывая быть нужной степени строгим.

Написание классов для системы игровых предметов

Одним из действительно важных аспектов любой игры является система предметов. Первым вопросом, который я задал себе на этапе проектирования, на что я должен сделать упор при создании такой системы. Ответом на этот вопрос было: экономия памяти. Ведь, если даже создать класс, который будет описывать лишь один предмет, то в таком случае, если предметов на карте будет 20, 30, 40 или больше, то столько же потребуется и экземпляров классов. Подобная перспектива не радовала и посоветовавшись с коллегами, я пришел к выводу, что необходимо сделать

классы, которые будут отображать общие внутренние свойства игрового предмета того или иного типа, то есть будут некой неизменной сущностью предмета. Так же необходимы классы, которые будут уже представлять конкретные предметы. Позже при совещании с менторами мы узнали, что данный шаблон называется Flyweight и направлен он именно на оптимизацию работы с памятью.

Когда примерный шаблон системы был намечен, была так же произведена классификация игровых предметов. Было выделено 4 типа предметов: предмет, не обладающий какими-либо особыми свойствами(квестовые предметы, трофеи, неиспользуемые предметы), используемый предметы(аптечки, предметы, увеличивающие какие-либо показатели героя), оружие и броня. Так как большую часть предметов составляют два последних типа, то паттерн проектирования Flyweight было решено применить только к ним.

Основной сложностью при создании данной системы было именно эффективное использование памяти. Данный вопрос был решен на этапе проектирования, на этапе же реализации проблем не возникало и с каждой новой версией лишь исправлялись некоторые ошибки в коде.

Подводя итоги проделанной работы можно сказать, что была создана эффективная система предметов, позволяющая оптимизировать работу с памятью, что и являлось первоначальной задачей.

Написание вместе с остальными членами команды классов игрового ИИ

Задачей было спроектировать и создать рабочий игровой искусственный интеллект. На этапе проектирования была произведена классификация двух основных состояний любого NPC: патрулирование и бой. Предполагалось, что дружелюбные к главному герою NPC на самом деле просто не враждебные и если совершить действия, которые ухудшат взаимоотношения между главным героем и NPC, то NPC может перейти в состояние боя и атаковать главного героя. Пока этого не происходит, то любой NPC просто исследует местность на наличие врага. Подобный подход можно назвать «паранойей искусственного интеллекта». То есть любой персонаж в игре, кроме главного героя, постоянно находится в поиске противника.

Ввиду ограничений по времени было решено, что точку, к которой будет направляться NPC для поиска врага будет выбираться случайно в четко обозначенной для каждого NPC зоне патрулирования.

Когда NPC находит враждебных персонажей, то он сначала оценивает, какого персонажа ему атаковать по принципу наиболее близкого по характеристикам к самому NPC, затем выбирает свободное укрытие, перемещается к нему и открывает огонь по противнику.

Для разделения данных состояний было решено на этапе проектирования сделать класс AI, который будет неким менеджером, контролирующим смену этих состояний.

На этапе реализации стало понятно, что данная задача является непосильной для одного человека, из-за чего было принято решение, что делать ИИ будут несколько человек по завершению всех остальных своих задач. Однако к концу проекта так и не получилось полностью разработать систему искусственного интеллекта и было сделано лишь патрулирование для NPC.

Тем не менее в процессе проектирования было прочитано множество материала про устройство игрового ИИ в различных играх. И хоть завершить эту часть проекта так и не удалось, было получено определенное количество опыта в разработке.

Заключение

В результате практики была создана практически вся техническая база для программного продукта. Для дальнейшей разработки было решено оставить декоративные задачи(сюжет, локации, квесты, диалоги и пр.) и систему боя у не игровых персонажей.

В ходе практики члены команды научились работать с UML диаграммами, освоили основы объектно-ориентированного программирования на языке Python 3.7, была изучена библиотека Cocos2D, были изучены различные паттерны проектирования и методологии разработки, были получены важные навыки командной разработки, навыки ведения проекта при помощи удаленного репозитория.

Литература

1. Документация библиотеки pygame // pygame.org URL: <https://pygame.readthedocs.io/en/pygame-1.3-maintenance/> (дата обращения: 30.03.2019).
2. Документация библиотеки cocos2D // python.cocos2d.org URL: <http://python.cocos2d.org/doc.html> (дата обращения: 28.03.2019).
3. Разработка игр с использованием Cocos2d на Python // habr URL: <https://habr.com/ru/post/120438/> (дата обращения: 28.03.2019).