МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВОЛГОГРАДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

КАФЕДРА КОМПЬЮТЕРНЫХ НАУК И ЭКСПЕРИМЕНТАЛЬНОЙ МАТЕМАТИКИ

«Допустить к защите» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Заведующий кафедрой КНЭМ д.ф.-м.н., доцент  
**Клячин Владимир Александрович** «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021г.

Выпускная квалификационная работа по направлению подготовки магистров

«Математическое обеспечение и администрирование информационных

систем»

«Разработка учебно-игрового комплекса по алгоритмам компьютерной графики»

Выполнил: студент гр. МОСм-191

**Бударин Игорь Александрович**

**\_**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Научный руководитель:  
к.ф.-м.н., доцент

**Григорьева Елена Геннадьевна**

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

Волгоград 2021

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ……………………………………………………………………….3

1. АНАЛИЗ ОБЛАСТИ ИГРОВОГО ДИЗАЙНА……………………………….4

1.1. ЭЛЕМЕНТНАЯ ТЕТРАДА……………………………………………5

1.2. ИГРОЦЕНТРИЧЕСКИЙ ПОДХОД К РАЗРАБОТКЕ……………….9

2. АНАЛИЗ ПОХОЖИХ ПРИЛОЖЕНИЙ…………………………………..…11

2.1. МЕХАНИКИ ОБУЧЕНИЯ………………………………………….13

2.1.1. ОБУЧЕНИЕ В LINGODEER……………………………….13

2.1.2. ОБУЧЕНИЕ В DUOLINGO………………………………...15

2.1.3. ОБУЧЕНИЕ В INFLUENT………………………………….16

2.2. МЕХАНИКИ ВОВЛЕЧЕНИЯ В ОБУЧЕНИЕ……………………...17

2.2.1. ВОВЛЕЧЕНИЕ В ОБУЧЕНИЕ В LINGODEER…………..17

2.2.2. ВОВЛЕЧЕНИЕ В ОБУЧЕНИЕ В DUOLINGO……………18

2.2.3. ВОВЛЕЧЕНИЕ В ОБУЧЕНИЕ В INFLUENT……………..19

2.3. МЕХАНИКИ ВОЗВРАЩЕНИЯ ИГРОКА В ПРИЛОЖЕНИЕ……20

2.4. РЕЗУЛЬТАТ ИССЛЕДОВАНИЯ ПОХОЖИХ ПРИЛОЖЕНИЙ…..21

3. АНАЛИЗ СТАРОЙ ВЕРСИИ ПРИЛОЖЕНИЯ С ПРИМЕНЕНИЕМ ЭЛЕМЕНТНОЙ ТЕТРАДЫ……………………………………………………..22

4. ИЗМЕНЕНИЯ В ПРИЛОЖЕНИИ НА ОСНОВЕ АНАЛИЗА……………...23

4.1. ГЛАВНАЯ ТЕМА…………………………………………………….24

4.2. ИСТОРИЯ……………………………………………………………..25

4.3. ЭСТЕТИКА…………………………………………………………...26

4.4. МЕХАНИКА………………………………………………………….27

4.5.ТЕХНОЛОГИЯ………………………………………………………..28

5. ДОПОЛНИТЕЛЬНЫЕ РАЗРАБОТКИ……………………………………….32

5.1 ПРИЛОЖЕНИЕ PIXELSTATS………………………………………32

ЗАКЛЮЧЕНИЕ…………………………………………………………………….33

СПИСОК ЛИТЕРАТУРЫ………………………………………………………….34

ПРИЛОЖЕНИЕ…………………………………………………………………….35

**ВВЕДЕНИЕ**

В учебном процессе, связанном с изучением алгоритмов, в частности алгоритмов компьютерной графики, наблюдается проблема, заключающаяся в недостаточной наглядности описания работы алгоритмов. Одним из методов решения данной проблемы является подача описания алгоритмов в игровой форме, с помощью игрового приложения, чему и посвящена данная работа. Актуальность работы заключается в практически полном отсутствии игровых приложений, связанных с изучением компьютерной графики, а так же в неиспользовании игрового подхода при изучении алгоритмов студентами. Цель работы — разработать учебно-игровой комплекс приложений, обучающих студентов алгоритмам игровой графики в игровой форме.

В связи с целью сформированы следующие задачи исследования:

1. Изучить предметную область игрового дизайна, ознакомиться с принципами построения игр, методологиями разработки и принципами создания механик, мотивирующих игрока играть
2. Изучить существующие игровые приложения, используемые для обучения — конкретно обучения иностранным языкам
3. На основе теории игрового дизайна, проанализировать существующую версию созданного ранее приложения для выявления необходимых улучшений
4. Применить данные, полученные в ходе анализа приложения, для его улучшения
5. При необходимости, реализовать дополнительные приложения для поддержки основного приложения

**1. АНАЛИЗ ОБЛАСТИ ИГРОВОГО ДИЗАЙНА**

Для того, чтобы понять, на основе каких принципов строить приложение, необходимо изучить область игрового дизайна. В качестве материала для исследования были использованы книги, написанные игровыми дизайнерами с многолетним опытом работы над играми, а также в данный момент преподающих курсы по игровому дизайну в различных университетах. Цель изучения области игрового дизайна — изучить принципы построения игр, в которые игроку будет интересно играть, методологии разработки, используемых игровыми дизайнерами в реальных проектах и механики, которые будут удерживать внимание игрока и позволяет ему легко обучаться. В результате исследования были рассмотрены понятие элементной тетрады как принципов построения игр, связь элементов с основной темой игры, а так же игроцентрическая методология разработки.

**1. 1. ЭЛЕМЕНТНАЯ ТЕТРАДА**

Элементная тетрада — тетрада из элементов, которые образуют основу дизайна игры. Каждый из этих элементов отвечает за отдельную часть дизайна и напрямую влияет на игровой опыт, получаемый игроком. Наличие всех элементов необязательно, однако тщательная проработка каждого элемента увеличивает вероятность игрока получить более приятный и интересный игровой опыт. Тетрада состоит из следующих элементов — история, эстетика, механика и технология. Существует несколько трактовок каждого из элементов, однако их всех объединяет их влияние друг на друга и их связь с темой игры. Все элементы должны быть напрямую связаны с темой игры и работать на ее раскрытие (исключением может являться технология, но именно она задает ограничения и направляет другие элементы тетрады). В этой работе будет использоваться трактовка элементов, которую придерживается игровой дизайнер Джесси Шелл:

1) История - последовательность событий, разворачивающихся в игре. Она может быть линейной и последовательной или возникать и меняться во время игрового процесса.

2) Эстетика - то как игра выглядит, звучит, «пахнет», «чувствуется на вкус» и ощущается. Этот элемент наиболее очевиден для игроков и способствует погружению в вымышленный мир игры.

3) Механика - это процедуры и правила игры. Механика описывает цель игры, и то как игроки могут или же не могут её достичь, и что происходит, когда они пытаются.

4) Технология - любые материалы и взаимодействия, которые делают игру возможной, от бумаги и карандашей до компьютеризированного оборудования. Этот элемент устанавливает ограничения для других элементов, а также должен направлять процесс проектирования таким образом, чтобы он синхронизировался с этими ограничениями.

Схема взаимодействия между элементами тетрады выглядит следующим образом:

 Как видно на схеме, все элементы связаны между собой и влияют друг на друга. Некоторые из элементов более заметны для игрока, чем остальные, однако это не значит, что остальные элементы менее важны. Каждый элемент имеет большое значение в построении опыта игрока, и изменение в одном элементе может затронуть остальные и кардинально изменить игровой опыт.

Как пример влияния элементов друг на друга можно взять известную игру «Морской бой» и попытаться разбить её на составляющие, которые будут коррелировать с элементами тетрады. В классическом варианте игры — игра двух человек в одном помещении на бумаге — эстетикой будет именно формат игры на бумаге, квадратные обозначения кораблей, нарисованные ручкой или карандашом, обозначения полей и команды «мимо», «ранил» или «убил». История в данном случае будет таковой — игроки являются адмиралами своих флотов и сходятся в бою в открытом море. Их задача — победить противника, не зная его конкретного местоположения, делая выстрелы, основываясь на информации от предыдущих выстрелов, смекалки и случайности. Механика — игроки выбирают на поле 10 на 10 клеток место, где они расставят свои «корабли» - комбинации клеток длиной от 1 до 4: 4 клетки по одной, 3 клетки по две подряд, 2 по три подряд и 1 по четыре подряд. Комбинации клеток не должны пересекаться, так же, как корабли в реальной жизни не могут проходить друг через друга. Также они не могут быть расположены на соседних клетках, так же как в реальной жизни боевые корабли имеют достаточно большие размеры и нуждаются в пространстве, чтобы находиться друг рядом с другом. Каждой клетке присваивается координата — горизонтали помечаются цифрами, вертикали буквами. Затем игроки по очереди называют одну координату, в которую они делают «выстрел», и противник должен ответить тремя возможными вариантами - «мимо», «ранил» или «убил», которые означают, что игрок либо не попал ни в один корабль, либо уничтожил часть корабля, но не весь, либо он уничтожил корабль полностью соответственно. Если игрок не попадает в корабль, то ход переходит к другому игроку. Если же он попадает в часть корабля или уничтожает его, то он делает еще один выстрел. Игра заканчивается, когда один из игроков уничтожает все корабли противника.

Технологией в таком случае являются бумага и ручка или карандаш.

Допустим, мы решим изменить то, на чем играть в эту игру и изменим технологию на настольный вариант, с приготовленным игровым полем и фигурками кораблей. Это хоть и не изменит истории или механики, это напрямую повлияет на эстетику — теперь это уже не игра, в которую можно играть где угодно, имея под рукой лишь ручку и бумагу — теперь это настольная игра, которой нужно пространство, и которая имеет свои эстетические особенности в виде игрового поля и фигурок кораблей. Теперь допустим мы хотим поменять историю таким образом, чтобы игроки не являлись адмиралами, а стали обычными моряками. В таком случае необходимо будет изменить и механики, т. к. адмиралы отдают приказы об атаке, а подчиненные моряки исполняют их, а значит предыдущий вариант механики будет не работать с точки зрения истории. Или, к примеру, мы захотим добавить новое правило - игроки смогут менять местоположение неподбитых кораблей в обмен на то, что они не смогут стрелять на том же самом ходу. Это напрямую влияет на технологию, т. к. в классическом варианте игры на бумаге это будет работать только если игроки играют с карандашами в руках, в противном случае придется этот вариант невозможен без дополнительных листов бумаги для каждой партии. К тому же это повлияет и на уже существующие правила - это сильно усложнит жизнь второму игроку, т. к. ни одно из полей, в которые он до этого стрелял, больше не может служить гарантией того, что там ничего не находится. Смена эстетики может повлиять на все элементы сразу — если, например, данная игра из развлечения для двоих с ручкой и бумагой про морские бои превратится в игру про космические путешествия, не меняя всего остального, это полностью изменит механики, историю и, потенциально, технологию.

Таким образом можно сделать вывод, что эти четыре элемента, напрямую связанные с темой игры, тесно связаны между собой и образуют ту основу, которая должна быть заложена в игру, если разработчик желает, чтобы она приносила игроку интересный опыт, удовольствие и возбуждала в нем желание играть снова.

**1.2. ИГРОЦЕНТРИЧЕСКИЙ ПОДХОД К РАЗРАБОТКЕ**

В ходе изучения области игрового дизайна был изучен игроцентрический подход к разработке. Согласно игровому дизайнеру Трейси Фуллертон, главным движущим фактором разработки игры должны быть плей-тесты — игровые сессии, во время которых прототип игры отдается на тестирование обычным игрокам, которые играют в прототип и составляют свои впечатления о том, во что они поиграли. Основной цикл разработки состоит из следующих этапов:

1) Выполнение изменений (добавление нового функционала или изменение существующего)

2) Сборка прототипа с изменениями

3) Тестирование прототипа игроками

4) Опрос игроков на предмет их впечатлений о прототипе

5) Анализ отзывов игроков

6) Если в отзывах игроков найдены моменты, требующие изменения, доработки или удаления — производятся необходимые поправки, затем переход к пункту 2. Если отзывы игроков соответствуют ожидаемым или превосходят их, то изменения проекта сохраняются, и цикл начинается заново.

Таким образом через постоянные тестирования достигается контроль над тем, какой опыт получает игрок, и производятся изменения, необходимые для сохранения приятного и интересного опыта у игрока.

При выполнении данной работы в качестве тестировщиков выступили игроки из сообщества Cybervalhalla – игрового СМИ, игроки сообщества «Кодзима Гений», игроки сообщества Guardian.FM, а также несколько выпускников Института Математики и Информационных технологий Волгоградского Государственного университета. Основные моменты, протестированные игроками — механики приложения, простота понимания игроком приложения. Тестирование помогло выявить неочевидность работы механик некоторых игровых режимов, а также необходимость упрощения подсказок для того, чтобы облегчить игроку понимание того, что ему нужно делать в каждом из игровых режимов. Помимо этого, тестировщики помогли обнаружить несколько технических ошибок, которые были впоследствии исправлены.

**2. АНАЛИЗ ПОХОЖИХ ПРИЛОЖЕНИЙ**

В ходе исследования были изучены два приложения и одна игра — приложения LingoDeer и DuoLingo и игра Influent. Все вышеперечисленные программные продукты используются для изучения языков, и в достижении цели обучить пользователя иностранному языку используют игровой подход, однако каждое приложение подходит к этому по-разному.

Первым было изучено приложение для изучения языков под названием LingoDeer. Приложение позволяет пользователю изучать языки, постепенно обучая его новым словам, выражениям и конструкциям выбранного языка.

Большинство языков разделены на несколько уровней сложности, соответствующих уровням знания языка (A1, A2, B1, B2 и подобные).

Каждый уровень состоит из отдельных наборов заданий, посвященных отдельным темам. После определенного количества тем предлагается задание-тест, которое позволяет проверить усвоение уже изученных знаний и, если знания были усвоены, открыть следующий набор тем для изучения. Каждая тема содержит в себе несколько вещей:

1) Текстовая справка

2) Мини-игры на запоминание

3) Практика чтения и разговора

Текстовая справка содержит описание языковых конструкций, слов и выражений, которые будут изучаться в текущей теме, а также правила их употребления в некотором контексте.

Мини-игры присутствуют в каждой теме в количестве от 1 до 3 и состоят из нескольких подтипов мини-игр, описание которых будет ниже.

Практика чтения состоит из озвученных на изучаемом языке слайдов, на которых демонстрируются фразы и языковые конструкции, изучаемые в теме, в некотором контексте. На слайдах также присутствуют люди, которые являются действующими лицами этих слайдов. После нескольких слайдов пользователь должен прочесть фразу о том, что он увидел на слайдах и ответить, является ли эта фраза правдой или ложью. После практики чтения игра дает выбор: продолжить изучение других тем или практиковать разговоры. Практика разговоров состоит из записи каждой фразы, использованной в практике чтения, на микрофон устройства, на котором запущено приложение, а затем заменой уже записанных фраз на слайдах фразами, которые записал пользователь. После этого записанные реплики можно отправить на сервер приложения, где другие пользователи смогут прослушать их и оценить.

Следующим было оценено приложение DuoLingo. Можно сказать, что на данный момент DuoLingo является самым популярным приложением для изучения иностранных языков. Оно сочетает в себе простоту использования и методики обучения через мини-игры, которые дают положительный эффект при обучении новым понятиям в языке. Приложение во многом похоже на LingoDeer, однако имеет существенные отличия в некоторых аспектах, которые будут рассмотрены в работе далее. Структурно приложение также предлагает набор тем, которые нужно изучать, и также предлагает возможность повторить уже изученные материалы (режим Тренировка). Игра Influent также посвящена изучению иностранных языков, однако ее подход к изучению языков значительно отличается от подходов вышеупомянутых приложений. Главным мотиватором для изучения языка здесь выступает сюжет — главный герой (игрок) должен помочь ученому восстановить машину, которая помогает изучать языки, для этого он должен будет с помощью небольшого прототипа этой машины сканировать объекты внутри игрового мира и считывать данные о них. После считывания предмета выводится его перевод на изучаемый язык, а само слово добавляется во внутриигровой словарь.

**2.1. МЕХАНИКИ ОБУЧЕНИЯ**

**2.1.1. ОБУЧЕНИЕ В LINGODEER**

Механики обучения реализованы в виде мини-игр, в которых игрок должен давать правильные ответы, чтобы продвинуться дальше или получить очки. Мини игры бывают двух видов — на изучение нового материала и на проверку уже усвоенного. Мини-игры на изучение нового материала представлены в нескольких вариантах:

1. Вставка подходящего по смыслу слова
2. Выбор правильного перевода слова или предложения на изучаемый язык
3. Удаление лишних слов из предложения
4. Построение предложения вставкой недостающих слов в нужные позиции
5. Соединение слов с их переводом на изучаемый язык
6. Построение перевода предложения с нуля из представленных слов
7. Построение перевода услышанного слова/предложения с нуля из представленных букв/слов
8. Построение перевода предложения с нуля из представленных букв и знаков пунктуации (включая пробелы)
9. Выбор логичного по смыслу ответа на заданный вопрос

Мини-игры на повторение большинством такие же, как мини-игры на изучение нового, но в некоторых ситуациях усложнены (отсутствием подсказок, какие символы вводить), а также добавлен таймер. Если успеть ответить на все вопросы раньше, чем время истечет, то игра закончится. Помимо этого, существуют мини-игры тесты, которые открывают доступ к новому списку тем после прохождения предыдущих, однако при желании любой тест в курсе можно пройти, не завершив темы из раздела, усвоение которого этот тест проверяет. Мини-игры внутри тестов немного усложнены (отсутствуют подсказки с переводом на язык пользователя), и к тому же имеется ограничение на количество ошибок — после трех неверных ответов тест считается проваленным, и чтобы получить доступ к следующим темам, нужно пройти его заново. Выучить вопросы не получится, так как с каждым запуском они генерируются случайно (как в большинстве обычных мини-игр).

**2.1.2. ОБУЧЕНИЕ В DUOLINGO**

Большинство мини-игр для обучения совпадают с мини-играми в LingoDeer, однако некоторые мини-игры менее прозрачны для понимания. Из новых мини-игр существует произнесение фразы на языке в микрофон устройства, на котором запущено приложение.

Однако в этом приложении по-другому устроена структура обучения. Каждая тема имеет в себе пять уровней обучения. Для того, чтобы пройти на новую тему, требуется успешно пройти новую тему один раз, то есть пройти все включенные в нее уроки (в среднем четыре), однако для усвоения материала требуется пройти ее еще четыре раза.

Однако ключевым отличием от предыдущего приложения будет ограниченное количество ошибок на каждый урок — в LingoDeer ошибки во время мини-игр влияли на количество опыта, которое получал игрок, но не ограничивали его в получении всех заданий в рамках мини-игры. DuoLinguo, однако, дает игроку 5 «жизней» - возможностей ошибиться. После 5 ошибок приложение не дает пользователю продолжить обучение, а возвращает его на главный экран. После этого либо вы можете купить дополнительные жизни за алмазы (валюта внутри приложения, начисляется за прохождение уроков), либо пройти тренировку и повторить материалы за 1 новую жизнь, либо ждать 5 часов реального времени, пока жизни восстановятся. Кроме этого есть вариант купить подписку DuoLingo Plus, которая избавит пользователя от необходимости следить за жизнями вообще — они станут бесконечными.

**2.1.3. ОБУЧЕНИЕ В INFLUENT**

Основной механикой обучения является сканирование предметов вокруг главного героя. Игра поделена на уровни, которые соответствуют разным местам в повседневной жизни человека, и, соответственно, разным темам языка — личная комната, дом, улица, транспорт, магазин, работа, школа и так далее. Предметы, которые можно сканировать, также зависят от уровня, на котором находится игрок — это могут быть предметы быта, продукты в магазине, школьные принадлежности и так далее. При этом сами предметы могут быть любыми, отличаться друг от друга по размеру и предназначению, но все они объединены общей темой, которую выражает собой игровой уровень. Игроку позволяется сканировать любой предмет, который он видит перед собой, и после сканирования на экран выдается перевод, а само слово записывается во внутриигровой словарь — память устройства. При этом само устройство можно настраивать — в частности, настраивается количество слов, после которого запускается режим проверки, который по своей сути является обратным процессом для сканирования новых предметов — игра выдает слово из уже изученных, и игрок должен сканировать предмет, который представляет собой то слово, которое было выведено на экран. Минус этой механики в переусложненном меню настроек, которое поначалу скорее мешает понять свои возможности, чем настроить режим проверки под себя.

**2.2. МЕХАНИКИ ВОВЛЕЧЕНИЯ В ОБУЧЕНИЕ**

**2.2.1. ВОВЛЕЧЕНИЕ В ОБУЧЕНИЕ В LINGODEER**

Приложение вовлекает пользователя в обучение с помощью нескольких инструментов. Главный инструмент — видимый прогресс. У пользователя есть уровень, который можно повышать, изучая новые темы. Каждая мини-игра приносит от 0 до 10 единиц опыта. Кроме этого, игрок может поставить себе ежедневную цель — количество опыта, которое нужно заработать в день. Ежедневная цель варьируется от 20 до 60 единиц опыта, и за выполнение цели приложение начисляет дополнительный опыт в профиль пользователя. Помимо этого после запуска приложения 2-3 дня подряд, оно начинает присылать уведомления о том, что нужно повторить слабые места в изученном материале языка. Слабые места формируются из ошибок, допущенных в мини-играх во время предыдущих сессий запуска приложения. В дополнение к этому за определенные действия приложение награждает игрока достижениями, которые можно увидеть в его профиле. Достижения выдаются за прохождение уровней знания определенных языков и за определенное количество часов, проведенных за изучением конкретного языка.

**2.2.2. ВОВЛЕЧЕНИЕ В ОБУЧЕНИЕ В DUOLINGO**

Процесс вовлечения в обучение в DuoLingo схож с процессом вовлечения в LingoDeer – у пользователя есть прогресс в виде опыта, который зарабатывается за выполнение заданий, и есть достижения, которые выдаются за выполнение определенных действий в приложении. Игрок так же, как и в первом приложении, может выставить себе ежедневную цель. Как и LingoDeer, DuoLingo присылает уведомления о занятиях, при этом оно может присылать уведомления и тогда, когда происходит что-то другое. К примеру, в обоих приложениях можно подписываться на других людей, которые используют это приложение и следить за их прогрессом. Однако в DuoLingo если оба пользователя изучают один и тот же язык, то при ситуации, когда тот, на кого подписались, превзойдет результат подписчика в определенной теме, то подписчику придет уведомление об этом с предложением побить рекорд. Также по-другому работает режим повтора уже изученных материалов (здесь он называется тренировкой) — он не ограничен по времени, однако если пользователь допускает ошибки, то прогресс в тренировке откатывается назад. Количество ошибок также не ограничено, однако откат прогресса является очень неприятным моментом, который заставляет пользователя скорее бросить повторение, чем стараться лучше. К тому же режим тренировки здесь нужен для получения жизней — дополнительных прав на ошибку в обычных заданиях по изучению нового материала, и такой метод ограничения продвижения пользователя за его ошибки только мешает ему изучать новый материал, искусственно растягивая время, потраченное на повтор знаний, в обмен на очень маленький выигрыш. С другой стороны, такой метод может заставить пользователя лучше запоминать то, что он повторяет, так как пока он не ответит на вопрос правильно, он не будет продвигаться вперед. Впрочем, в любой момент из режима тренировки можно выйти, однако это не дает пользователю преимущества, но при этом и не заставляет это тратить на приложение больше времени, чем он сам хочет.

**2.2.3. ВОВЛЕЧЕНИЕ В ОБУЧЕНИЕ В INFLUENT**

Основной механикой вовлечения в обучение является сюжет игры, который дает игроку конкретную цель — помочь ученому собрать информацию для его машины. В самом процессе изучения тем (уровней игры) дополнительным мотиватором становится желание отсканировать все, что присутствует на уровне, чтобы количество отсканированных предметов было равно 100%. Кроме этого, сканирование предметов дает возможность после определенного количества слов получить улучшение сканера — вместо того, чтобы самостоятельно ходить по уровню и сканировать все в непосредственной близости к объекту, игрок получает в свое распоряжение подобие дрона, который быстрее перемещается по комнате, он проще в управлении и позволяет легче отсканировать труднодоступные предметы.

**2.3. МЕХАНИКИ ВОЗВРАЩЕНИЯ ИГРОКА В ПРИЛОЖЕНИЕ**

Помимо механик обучения и удержания внимания игрока приложения используют и некоторые механики по возвращению игрока в приложение. Одна из наиболее частых механик — уведомления. Условия использования уведомлений различаются — к примеру, приложение LingoDeer рассылает уведомления от 1 до 2-3 раз в день, если пользователь был активен в последние 2-3 дня с предложением изучить новый материал. Приложение Duolingo же, хотя и так же присылает уведомления, если пользователь был активен в течение последних 2-3 дней, ограничивается одним уведомлением в день. Оба приложения после нескольких уведомлений, оставшихся без внимания, перестают их присылать. В отличие от вышеупомянутых приложений, Influent не имеет подобных механик. Причина, возможно, кроется в методах монетизации — Influent является игрой, покупаемой один раз за полную цену, а Duolingo и LingoDeer являются условно-бесплатными приложениями, монетизирующимися за счет подписки, которая предлагается пользователям во время их использования, соответственно этим приложениям выгоднее возвращать пользователей и мотивировать их учиться дальше, и, как следствие, увидеть возможность приобрести подписку, которая дает преимущества на фоне бесплатного варианта обучения — отсутствие рекламы, полный доступ к курсам, отсутствие искусственных ограничений на количество ошибок (пример — сердца в Duolingo) и так далее.

**2.4. РЕЗУЛЬТАТЫ ИССЛЕДОВАНИЯ ПОХОЖИХ ПРИЛОЖЕНИЙ**

Таким образом, можно сделать такие выводы: внедрение игровой формы может помочь в обучении, если дать игроку достаточную мотивацию для того, чтобы изучать материал. Этого можно добиться введением какой-то абстрактной цели для игрока, к которой приведет постепенное освоение материала. Помимо этого игрока нужно поощрять за изучение материала либо видимыми наградами (достижениями), либо внутриигровыми ресурсами (опытом или чем-то другим), которые будут отражает его личный прогресс в обучении и давать конкретную промежуточную цель, которой он может достичь выполнением представленных заданий (набрать новый уровень, получить новый предмет и так далее). Кроме того, можно разделить процессы изучения и повторения уже изученного материала на отдельные разделы, которые, однако, связаны друг с другом, и поощрять игрока и за тот, и за другой процесс. Сами процессы должны быть выстроены так, чтобы игрок, не знакомый с тем, что от него могут потребовать, мог очень быстро понять принципы работы приложения/игры и начать выполнять задания как можно скорее. Участвующие в этих процессах мини-игры также должны быть максимально простыми для понимания, однако не обязательно одинаково простыми между собой, даже наоборот — максимально эффективно будет выстроить их порядок таким образом, чтобы игрок, начиная с самых легких игр, пришел к самым сложным, постепенно изучая материал. Кроме того, для каждой изучаемой темы должен присутствовать справочный материал, который поможет пользователю вникнуть в контекст и более успешно выполнять задания. Однако кроме этого игрока стоит ограничивать в продвижении, если он недостаточно успешно выполняет задания либо низкими результатами после выполнения заданий, либо отказывая в доступе к следующим темам. Тем не менее, не стоит ограничивать его слишком сильно, чтобы у игрока сохранилось желание изучать материал и преодолевать трудности, вместо того, чтобы все бросить и больше не продолжать.

**3. АНАЛИЗ СТАРОЙ ВЕРСИИ ПРИЛОЖЕНИЯ С ИСПОЛЬЗОВАНИЕМ ЭЛЕМЕНТНОЙ ТЕТРАДЫ**

Для того, чтобы определить, каким образом можно улучшить существующее приложение, необходимо было проанализировать его с использованием вышеупомянутой элементной тетрады, разобрать приложение на составляющие и оценить то, как реализованы те или иные элементы из тетрады, и, как следствие, какие из этих элементов требуют добавления, изменения или доработки для улучшения игрового опыта пользователя и улучшения качества его обучения.

Старая версия приложения состояла из 6 игровых режимов, основанных на алгоритмах компьютерной графики:

1. Алгоритм Брезенхема для рисования отрезка
2. Алгоритм Безье для отрисовки кривых
3. Алгоритм Сазерленда-Коэна для отсечения пикселей за пределами экрана
4. Алгоритм черепашки

Один из неназванных режимов — это повторение режима алгоритма Брезенхема с отрезками, образующими прямоугольники, и режим подбора похожего цвета по компонентам RGB.

**3.1. РЕЖИМ BREZENHEIM** Механика режима Brezenheim основана на алгоритме Брезенхема для отрисовки отрезка на экране. Суть алгоритма в том, что зная координаты начальных и конечных точек, на каждом шаге алгоритма вычисляется угол наклона отрезка, и в зависимости от значения угла, закрашивается либо пиксель на той же горизонтали/вертикали, что и текущий, либо на соседней, что ближе к конечной точке. Работа алгоритма продолжается, пока не будет достигнута конечная точка отрезка.  
 В игровом режиме сначала генерируются данные об отрезках — с помоью алгоритма Брезенхема создаются наборы точек, которые будут отрисовываться, а также углы наклона для каждой из точек. Задача игрока в данном режиме следующая: на табло выводится информация о текущем угле наклона, и в зависимости от этой информации игрок должен выбрать точку на игровом поле, которая будет закрашена. Если игрок указал правильно — точка будет закрашена. Игрок выиграет, если закрасит все точки всех линий за отведенное время.

Также помимо режима Brezenheim в проекте присутствует режим MultipleBrezenheim. Оба они опираются на алгоритм Брезенхема, но разница между ними в способе генерации отрезков — обычный Brezenheim генерирует случайные отрезки, а Multiple – генерирует их таким образом, чтобы из отрезков получился многоугольник.

**3.2. РЕЖИМ BEZIER**

Режим Bezier основан на алгоритме Безье для отрисовки кривых.

Перед началом игры с помощью алгоритма Безье генерируются опорные точки кривой, и сама кривая изображается на игровом поле. Игроку требуется найти все опорные точки кривой и нажать на них, включая первую и последнюю точки кривой. Игрок побеждает, если за отведенное время нажмет на все опорные точки заданной кривой.

**3.3. РЕЖИМ SOUTHCOHEN** Режим SouthCohen основан на алгоритме Сазерленда-Коэна, который отсекает части отрезков за пределами заданной области. Суть алгоритма в том, что область экрана делится прямыми на 9 зон, каждая из которых имеет свой четырехбитный идентификатор и затем проверяются концы отрезков. Если оба конца отрезков попадают в заданную область, то отрезок отрисовывается. Если один из концов отрезка попадает в область, а другой нет, то находится ближайшая точка отрезка, которая пересекается с прямой, которая разделяет точки исходного отрезка, и отрисовывается отрезок от найденной точки до точки внутри области. Если обе точки находятся за пределами области, но пересекают ее, то также находятся точки, в которых этот отрезок пересекает прямые, образующие зоны, и отрисовывается отрезок между этими точками. Соответственно, если обе точки лежат вне области и не пересекают ее, то этот отрезок не отрисовывается.

В начале работы режима создается некоторая область игрового поля, которая будет играть роль той части экрана, которая будет отрисовываться, а также генерируются несколько отрезков, причем таким образом, чтобы отрезок пересекал вышеупомянутую область как минимум один раз. Задача игрока нажимать на те части отрезков, которые выходят за область отрисовки. Игрок победит, если за отведенное время сможет убрать все части отрезков, которые не будут отрисовываться.

**3.4. РЕЖИМ TURTLE**

Режим Turtle основан на алгоритме выполнения команд «черепашка». «Черепашка» умеет только двигаться вперед в направлении ее взгляда и поворачиваться влево и вправо на 90 градусов. С помощью списка команд, соответствующим возможностям черепашки, можно задавать ей некоторый маршрут.

Перед началом игры генерируются строки маршрутов, в которых символ ‘F’ означает движение вперед, ‘+’ - поворот налево , ‘-’ - поворот направо. Во время игры на табло выводится маршрут. Игрок должен, нажимая на соответствующие кнопки, управлять черепашкой так, чтобы она повторяла маршрут на табло. Игрок победит, если за отведенное время сможет повторить все созданные ранее маршруты.

**3.5. РЕЖИМ COLORPICKER** Режим ColorPicker не основан на каком-либо алгоритме, он представляет собой режим пониженной сложности для ознакомления с RGB-компонентами, из которых состоят цвета. Перед началом игры генерируется цвет, и выводится на одно из двух пространств на экране. На втором пространстве выводится цвет, RGB-компоненты которого игрок может изменять, при этом сами компоненты тоже выводятся на экран. Задача игрока — с помощью слайдера изменить голубой цветовой компонент цвета на втором пространстве так, чтобы он был максимально похож на первый цвет. После того, как игрок найдет максимально похожий цвет, в первом пространстве сгенерируется новый цвет. В этом режиме нет ограничения по времени, и игрок побеждает, когда сможет найти наиболее похожий цвет на тот, что изображен на первом пространстве.

**3.6. АНАЛИЗ ПРИЛОЖЕНИЯ С ПОМОЩЬЮ ЭЛЕМЕНТОВ ТЕТРАДЫ**

История в старой версии приложения полностью отсутствовала. Эстетика представляла собой пиксельную графику, т. к. главной темой являлись компьютерные алгоритмы 2D-графики. Механики представляли собой простые головоломки, которые решались выбором нужного пикселя на экране или отсечением ненужных пикселей в зависимости от текущих условий работы алгоритма. Технологией было использование игрового движка Unity, однако без какой-либо выраженной архитектуры приложения.

Таким образом была очевидна необходимость добавления в приложение истории, которая будет актуальная для целевой аудитории игры и изменения технологии для того, чтобы упростить изменение приложения и добавление нового функционала, а так же решение технических проблем.

**4. ИЗМЕНЕНИЯ В ПРИЛОЖЕНИИ НА ОСНОВЕ АНАЛИЗА**

Для того, чтобы улучшить приложение и игровой опыт игроков, после анализа старой версии приложения с помощью элементной тетрады, необходимо было проанализировать каждый из элементов, то, как он представлен в приложении и выявить необходимые действия для улучшения. Как было сказано выше, основными изменениями было добавление истории и изменение технологии, однако небольшие изменения коснулись также механик и эстетики. Помимо этого, прежде чем проводить изменения, была еще раз проанализирована главная тема приложения, и уже исходя из нее, выбирались изменения для элементов приложения.

**4.1. ГЛАВНАЯ ТЕМА**

Главная тема приложения осталась без изменения — приложение является игрой, посвященной изучению алгоритмов компьютерной графики. Однако для эффективного использования элементной тетрады это означало, что все элементы должны коррелировать с этой темой и усиливать ее для создания лучшего игрового опыта для игрока, и как следствие, она влияла на то, какую форму примут все элементы тетрады.

**4.2. ИСТОРИЯ**

Для того, чтобы добавить игроку мотивацию играть, необходима была история, или хотя бы простое описание процесса игры, цель для игрока, причина изучать игру и находиться в ней. Кроме того, история должна быть связана с темой игры и быть актуальной для целевой аудитории игры — студентов. Это в свою очередь дает несколько критериев для истории:

1) История должна быть связана с алгоритмами компьютерной графики

2) История должна быть актуальной для студентов технического института, знакомых со сферой информационных технологий

3) История должна мотивировать их на изучение алгоритмов компьютерной графики

4) Так как целевая аудитория — студенты, и приложение является еще и обучающим, то история должна быть актуальной для их учебного процесса и даже быть напрямую связана с ним

Исходя из этих критериев, была придумана следующая небольшая история:

Студент в университете (игрок) связывается с продвинутой операционной системой DekanOS для того, чтобы помочь восстановить работоспособность графических алгоритмов и получить за это знания и баллы.

Таким образом, история актуальна для целевой аудитории — студентов технического института, изучающих компьютерную графику, и мотивирует их играть в игру, так как это поможет им изучить алгоритмы компьютерной графики и получить за это баллы в реальной жизни.

**4.3. ЭСТЕТИКА**

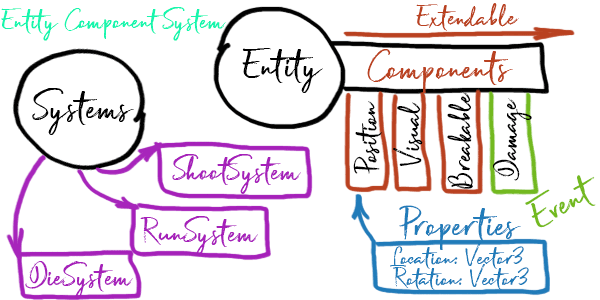
Эстетика приложения в основном осталась без изменений — эстетика пиксельной графики идеально подходит для приложения по 2D-алгоритмам компьютерной графики, так как алгоритмы напрямую работают с пикселями, которые напрямую используются в эстетике приложения. Единственные изменения, которые были сделаны — шрифты были заменены с «пиксельных» на более читаемые, так как, несмотря на следование эстетике, они ухудшали игровой опыт, так как игроки не всегда могли прочитать текст, написанный стилизованными под пиксели шрифтами.

**4.4. МЕХАНИКА**

Механики приложения также остались без изменений — игровой процесс разделен на простые головоломки, суть которых заключается в выборе нужного пикселя на экране или удалении лишних пикселей в зависимости от текущих условий алгоритма. Основная причина, по которым механики остались без изменений — такой тип механик следует тем же принципам обучения, которым следуют изученные в работе приложения для обучения иностранным языкам — простые головоломки, принципы работы которых достаточно легко понять, и дополнительные подсказки для более подробного объяснения принципов работы алгоритмов. Изменения коснулись подсказок, так как по отзывам тестеров, они являлись запутанными и не объясняли принципов работы алгоритмов и игровых режимов, поэтому подсказки были заменены на более понятные.

**4.5. ТЕХНОЛОГИЯ**

После анализа технологии стало очевидно, что приложению требуется масштабный рефакторинг. Помимо проблемы с огромным количеством дублирования кода, приложение использовало статический менеджер событий для управления игровым процессом. Это привело к тому, что класс, подписывающийся на определенные события, должен был либо отписываться и подписываться на события в реальном времени, либо иметь дополнительный код на случаи, когда реакция на событие должна была измениться или не происходить вовсе. Это привело к еще большему дублированию кода, добавлению лишнего когда и хаотичной системе управления игровым процессом, в которой было очень сложно выявить последовательность событий, а также это привело к хаотичным связям между разными системами, которые влияли друг на друга, хотя должны были работать независимо друг от друга. Поэтому было решено выбрать такую архитектуру, которая позволит легко отслеживать порядок выполнения действий в приложении, легко добавлять новый функционал и изменять старый, не ломая существующий код, а также использовать разные системы независимо друг от друга. Архитектурным паттерном, решающим все эти проблемы, является паттерн ECS (Entity-Component-System). Суть паттерна в разделении всего кода на три типа объектов — компоненты (чистые данные, никакой логики), сущности (контейнеры для компонентов) и системы (классы, содержащие логику обработки сущностей, содержащих те или иные компоненты). Простой пример разбиения небольшого количества игровой логики представлен ниже:



Как указано на картинке, сущности представляют собой только контейнеры для компонентов, а каждый компонент хранит в себе только данные. В этом конкретном примере компонент Position хранит себе поле Location – позицию сущности в игровом мире и поле Rotation с данными о том, насколько эта сущность повернута в пространстве. Компонент Visual содержит в себе визуальные данные, которые используются для отрисовки сущности на экране, компонент Breakable – компонент-флаг, обозначающий, что сущность можно сломать, а компонент Damage – урон, который нанесли этой сущности (события в паттерне ECS тоже являются компонентами). Кроме этого каждый кадр игровой мир, то есть набор сущностей, обрабатывается списком систем, причем каждая система обрабатывает только те сущности, которые имеют компоненты, которые ей нужны (проще говоря — фильтрует их). В данном системе система ShootSystem будет обрабатывать все сущности, которые в этот кадр сделали выстрел(имеют компонент-событие выстрела), RunSystem будет менять позицию всех сущностей, у которых есть компонент с позицией в игровом мире и у которых есть компонент-событие перемещения с новыми координатами или значениями, на которые будет изменяться текущая позиция сущностей, а DieSystem – обрабатывать существ, которые имеют компонент-событие о смерти или имеют компонент со здоровьем и это здоровье упало ниже нуля.

В качестве используемой реализации паттерна ECS был использован фреймворк LeoECS, так как он имел легкую интеграцию с движком Unity, в том числе и с его подсистемой событий пользовательского интерфейса, позволял настраивать порядок выполнения систем и при необходимости поддерживал возможность поддержки параллельного выполнения. К тому же LeoECS оптимизирует выполнение путем группировки сущностей по компонентам для более быстрой фильтрации, использует пулы для сущностей и компонентов для экономии памяти, позволяет не выделять память под компоненты-флаги, но при этом использовать их для фильтрации сущностей, а также автоматически удалять компоненты, которые используются только один кадр, такие, как, например, компоненты-события. Помимо LeoECS существовали и другие варианты, такие, как фреймворк Entitas, или свое решение паттерна ECS от разработчиков Unity. Однако на фоне LeoECS, Entitas выглядел громоздким и непростым для изучения, а решение от Unity хотя и направлено на параллельные вычисления для лучшей производительности, не имеет возможности управлять порядком выполнения систем. Кроме того, движок Unity изначально имеет в своей основе подобие паттерна ECS, однако в данном случае поведение хранится не в отдельных системах, а в самих компонентах. Плюсами такого подхода можно назвать то, что каждый компонент знает, как с собой работать и код компонента инкапсулирован в нем самом, а движок только вызывает нужный метод в тот момент, когда этот метод вызывается на всех сущностях, у которых есть компоненты с реализацией этого метода, что очень похоже на метод работы систем (методы Start, FixedUpdate, Update, LateUpdate и OnDelete в Unity, которые также вызываются каждый кадр в определенном порядке для всех сущностей с компонентами, которые содержат реализацию этих методов). Минусами такого подхода можно назвать то, что возможность управления порядком выполнения систем также остается не за разработчиком, к тому же поиск всех объектов с компонентами, которые имеют реализацию того или иного метода, занимает какое-то время, что сказывается на производительности.

Таким образом, технология новой версии приложения остается в основе на движке Unity, однако теперь весь код игровых режимов работает на паттерне ECS с использованием фреймворка LeoECS.

**5. ДОПОЛНИТЕЛЬНЫЕ РАЗРАБОТКИ**

**5.1. ПРИЛОЖЕНИЕ PIXELSTATS**

Для того, чтобы увеличить мотивацию игрока стремиться к лучшим результатам, необходимо было дать ему возможность отслеживать свои успехи и видеть свой прогресс. Для этой цели было разработано приложение PixelStats – спутник основного приложения. PixelStats – это приложение-сервис для добавления и просмотра статистики игрока по всем его игровым сессиям. Приложения позволяет смотреть историю игровых сессий конкретного пользователя, смотреть результаты каждой конкретной сессии, а также его лучшие результаты за все время игры. Дополнительно приложение имеет простой API для передачи и отправки данных по игровым сессиям через веб-запросы типов GET и POST с данными запросов в виде объектов JSON. API позволяет получить список игр пользователя по его имени, а также добавить данные об игровой сессии в список игр пользователя. Именно через API происходит взаимодействие основного приложения и PixelStats — в конце каждой игровой сессии подсчитываются ее результаты, сериализуются в JSON-объект и отправляются по адресу веб-приложения в виде POST-запроса с вышеупомянутым JSON-объектом в качестве данных. Веб-приложение получает этот запрос, десериализует JSON-объект в модель данных и добавляет полученную информацию об игровой сессии в базу данных. Приложение написано на языке C#, используя платформу ASP .NET Core, в основе которой лежит паттерн MVC (Model-View-Controller), база данных — Microsoft SQL Server 2016. Хостинг, на котором находится приложение — Microsoft Azure.

**ЗАКЛЮЧЕНИЕ**

В ходе написания диссертации была изучена предметная область игрового дизайна с целью найти основы, на которых должен строиться дизайн игр, методологии разработки, используемые в реальных проектах, а также механики, позволяющие обучить игрока, улучшить игровой опыт и удерживать игрока в игре. В итоге было изучено понятие элементной тетрады и ее применение в дизайне игр, а также была исследована и использована игроцентрическая методология разработки. Помимо этого были исследованы приложения для изучения иностранных языков на предмет механик, вовлекающих игроков в обучение, удерживающих его внимание и возвращающих в приложение, если игрок на какое-то время перестал его использовать. На основе полученных данных был проведен анализ старой версии уже созданного приложения с целью выяснить, какие из элементов требовали добавления или улучшения. В результате на основе этого анализа в приложение были добавлены история, изменена технология, проведен масштабный рефакторинг и изменение механик, основания для которых были взяты из проведенных плей-тестов как часть игроцентрической методологии разработки. Кроме этого было разработано веб-приложение для добавления, хранения и просмотра статистики игроков. Таким образом можно заключить, что данный игровой комплекс обладает достаточным набором функций, чтобы быть использованным в реальном учебном процессе, если не как официальная часть учебного плана по курсам, связанным с компьютерной графикой, то как экспериментальный метод обучения с целью улучшения как самого комплекса с потенциалом использования его в будущем, так и учебного процесса.

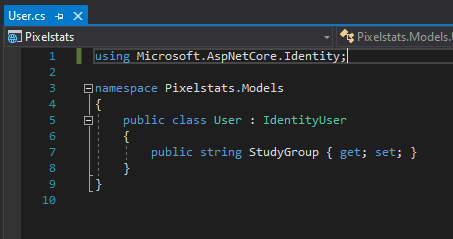
**СПИСОК ЛИТЕРАТУРЫ**

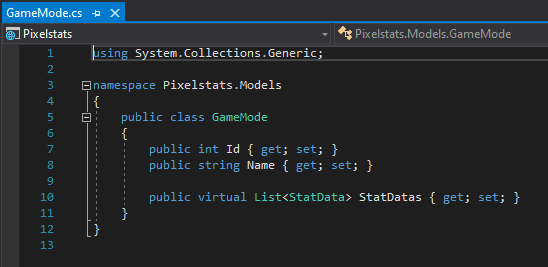
1. Хокинг Дж. Unity в действии. Мультиплатформенная разработка на C#. – Санкт-Петербург: Питер, 2016. – 336, с.:ил – (Серия «Для профессионалов»).
2. Торн А. Искусство создания сценариев в Unity / пер. с англ. Р. Н. Рагимова. – Москва: ДМК Пресс, 2016. – 360 с.: ил.
3. Григорьева Е.Г. Компьютерная графика. Краткий конспект лекций. Ч. 1. / Григорьева Е.Г. – Волгоград: Издательство ВолГУ, 2011. – 52 с.
4. Григорьева Е.Г. Компьютерная графика. Краткий конспект лекций. Часть 2. / Григорьева Е.Г. – Волгоград: Издательство ВолГУ, 2012. – 37 с.
5. Клячин В.А. Математические методы компьютерной графики. Монография. / Клячин В.А. – Волгоград: Издательство Волгоградский государственный университет, 2008. – 130 с.
6. Шелл Дж. Геймдизайн. Как создать игру, в которую будут играть все / пер. с англ. А.Лысенко. - Москва, ООО «Альпина Паблишер», 2019. - 820 с.
7. Fullerton, T. Game Design Workshop: A Playcentric Approach to Creating Innovative Games. - Берлингтон, Morgan Kauffman Publishers, 2008. - 535 с.
8. Бонд Дж. Гибсон. Unity и C#. Геймдев от идеи до реализации. - Санкт-Петербург: Питер, 2019. - 928 с.
9. <https://play.google.com/store/apps/details?id=com.lingodeer&hl=ru> 10.07.20
10. <https://play.google.com/store/apps/details?id=com.duolingo&hl=ru> 10.07.20
11. <https://store.steampowered.com/app/274980/Influent/> 10.07.20

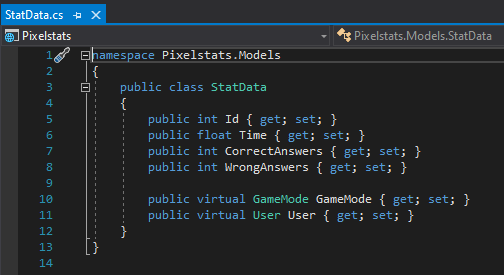
**ПРИЛОЖЕНИЕ**

Скриншоты кода приложения для сбора статистики PixelStats:

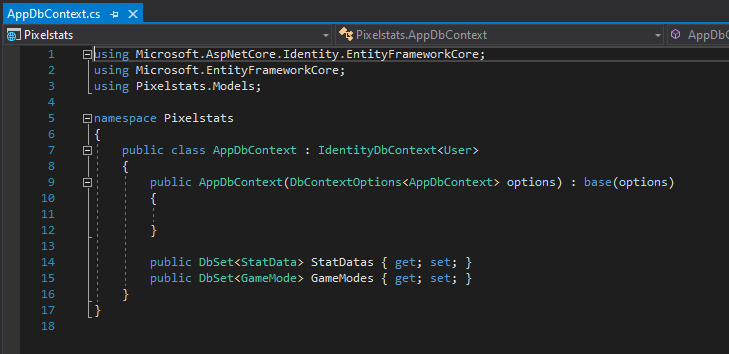
Модели:

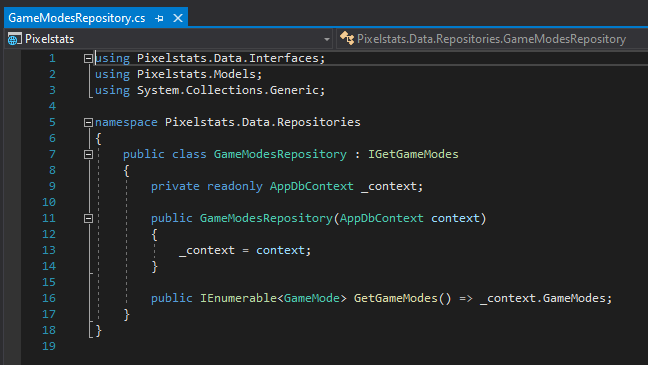


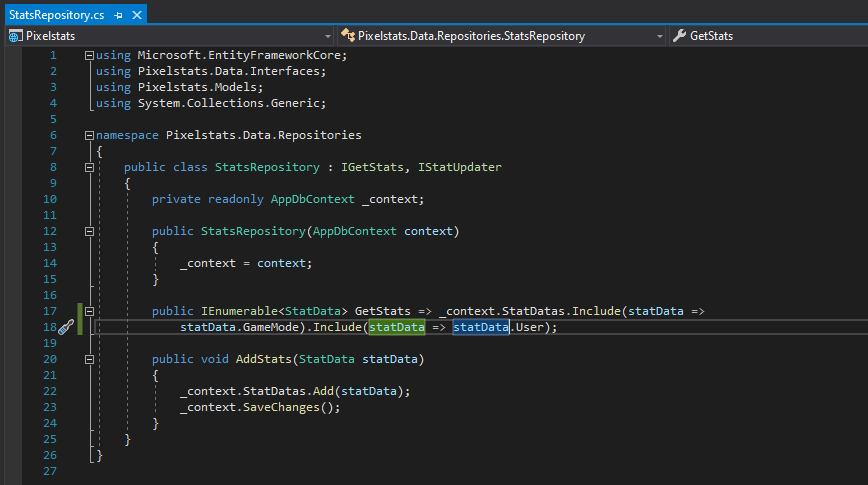


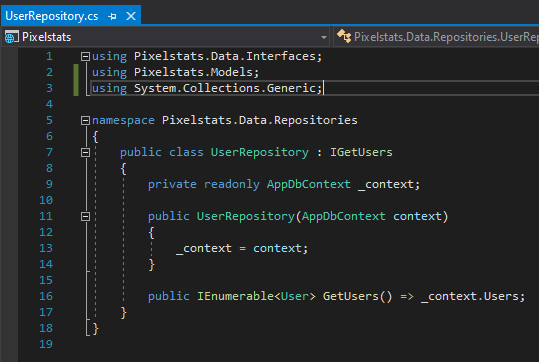


Класс-контекст для подключения к базе данных:

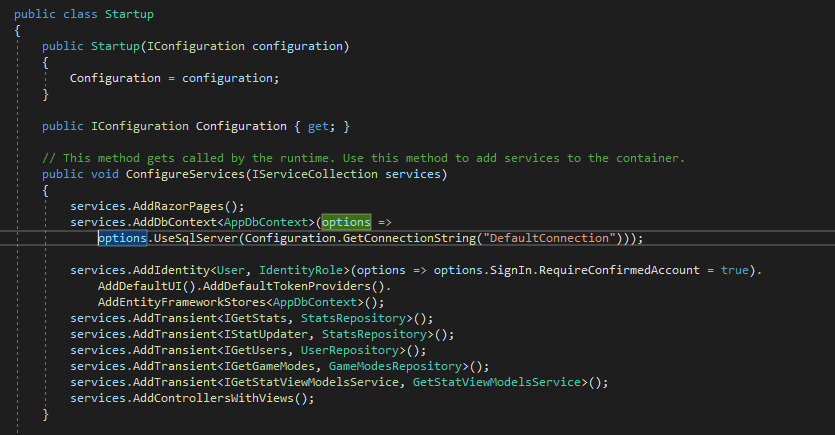
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
 Классы-получатели данных из базы данных:



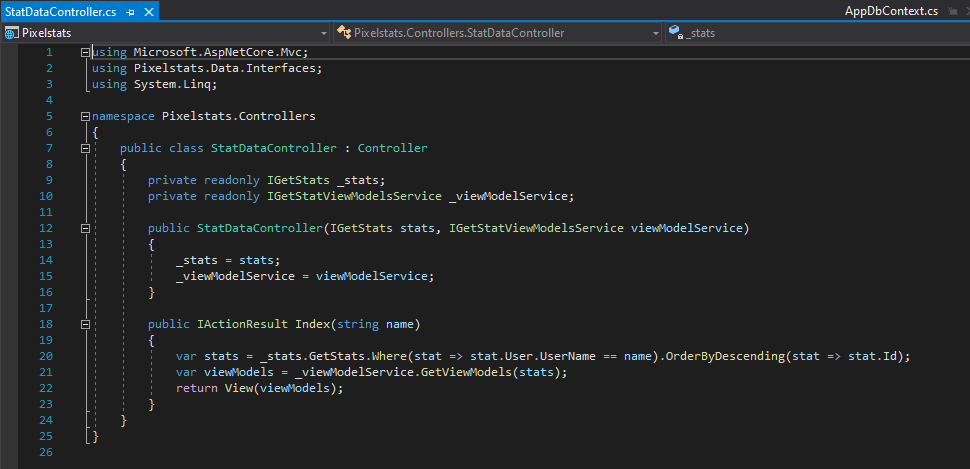


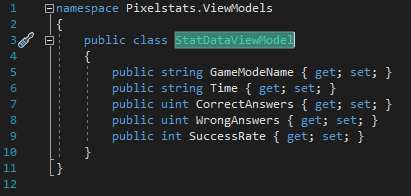


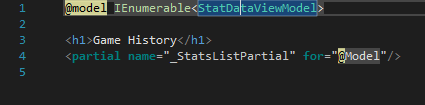
Класс-конфигуратор приложения:

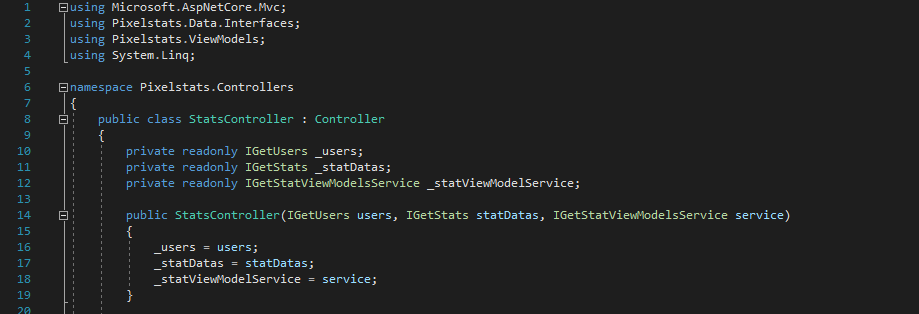


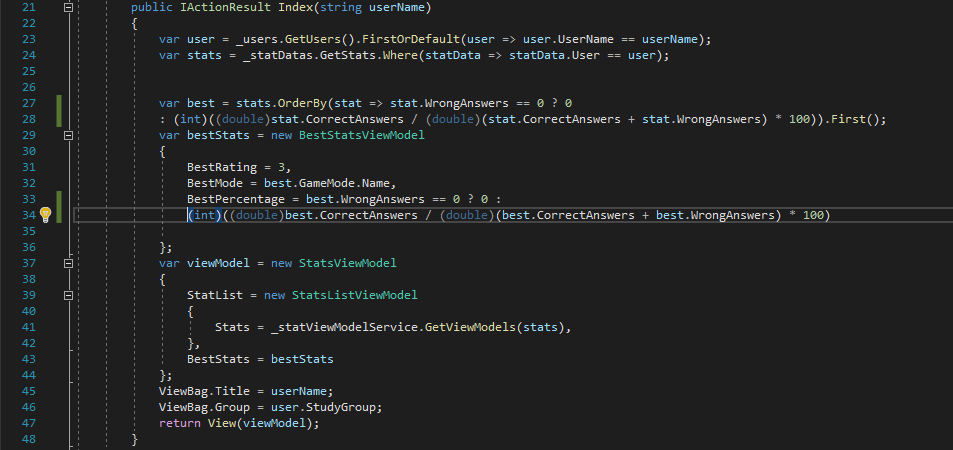
Контроллеры для обработки веб-запросов:

  
 Модели видов для передачи данных для отображения в веб-страницах:

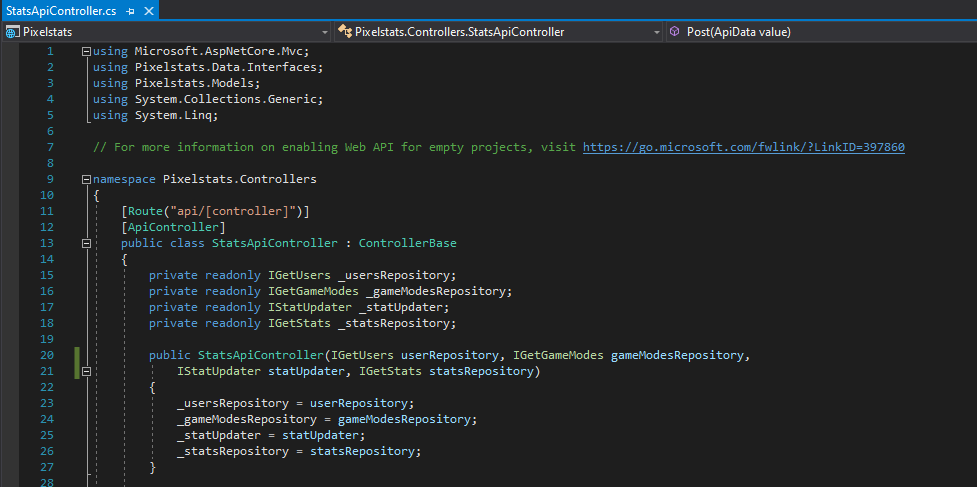


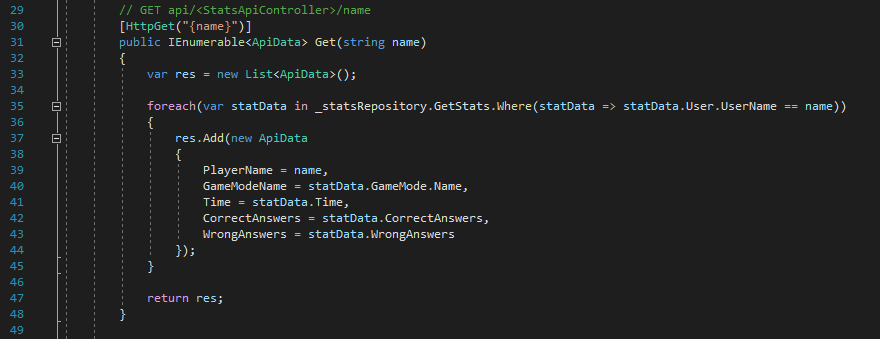


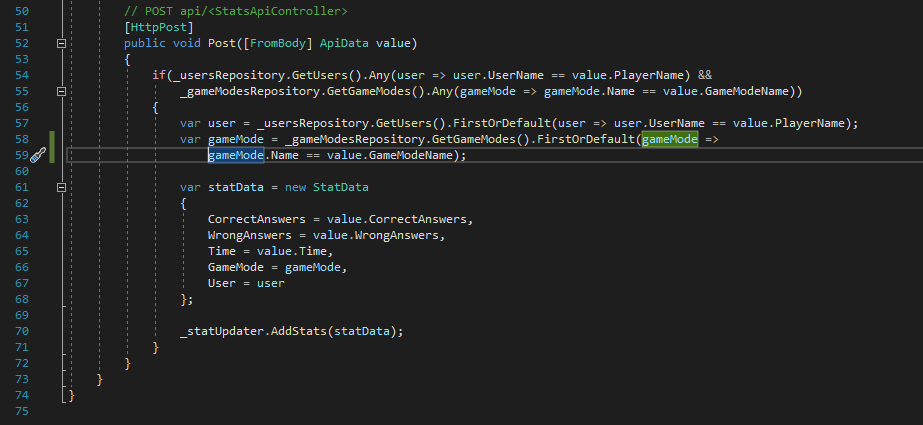




API-контроллер, принимающий и возвращающий данные об игровых раундах:







Примеры использования приложения:

