ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ДОПУСТИТЬ К ЗАЩИТЕ
Профессор с возложенными обязанностями заведующего Кафедрой информационных систем в искусстве и гуманитарных науках
(Борисов Н.В.)
"21" 05 2021 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Направление 09.03.03 «Прикладная информатика»
Уровень Бакалавриат
Основная образовательная программа
«Прикладная информатика в области искусств и гуманитарных наук»

«Создание мобильного приложения с аудиовизуальной инсталляцией»

Студентки *Дарьи Вячеславовны Зориной*

(подпись студента)

Руководитель к. *ф-м. н, доцент В. В. Захаркина*

(подпись руководителя)

Консультант *Ст. преподаватель И. А. Мбого*

Санкт-Петербург
2021

Оглавление

АННОТАЦИЯ

выпускной квалификационной работы

Зориной Дарьи Вячеславовны

«Создание мобильного приложения с аудиовизуальной инсталляцией»

Целью данной работы является создание аудиовизуальной инсталляции в мобильном приложении Android – «SILENTIO: Astral Relax».

ВКР состоит из 12 глав. В главах 1 и 2 проведен анализ предыдущих авторских работ с целью определения визуальной стилистики приложения, а также проведено исследование возможностей анимаций, встраиваемых в приложения. В главах 3—5 описано создание контента для приложения: иллюстраций, анимаций, музыкальной композиции. В главах 6—10 освещен процесс разработки приложения в среде Android Studio на языке программирования Java, в главе 11 – процесс публикации приложения на ресурсе Google Play и создание сопроводительных графических материалов, в главе 12 исследованы варианты дальнейшего развития приложения.

Прикладным результатом ВКР является мобильное приложение с интерактивной векторной анимацией и звуковым сопровождением. Дизайн, создание и анимация персонажей, исполнение, живая запись и синтез звука, программная реализация приложения выполнены автором ВКР.

В процессе работы использованы программы Adobe Illustrator, Animate, After Effects с расширением Bodymovin, FlStudio, Figma и среда разработки Android Studio.

Объем работы: 44 страницы текста, 20 рисунков и 4 источника литературы.

Ключевые слова: мобильное приложение, Android, Lottie, аудиовизуальная инсталляция, иллюстрация, анимация, музыкальная композиция, дизайн.

Автор работы _____ Зорина Дарья Вячеславовна
подпись (фамилия, имя, отчество)

Руководитель работы _____ Захаркина Валентина Валентиновна
подпись (фамилия, имя, отчество)

Введение

Индустрия мобильных игр – это стремительно развивающаяся область, значение которой легко недооценить. Однако свобода публикации, мобильная монетизация и высокая конкурентоспособность по количеству загрузок сделали ее одной из самых привлекательных областей для профессионалов: художников, аниматоров, разработчиков, геймдизайнеров, сценаристов.

Прежде всего, проект «SILENTIO» – аудиовизуальная цифровая инсталляция в мобильном приложении Android, разработанная на основе принципов эстетики и художественного смысла. Цель создания «SILENTIO» – переосмысление концепции мобильного приложения с точки зрения художника, иными словами – ставится вопрос, может ли художник создать цифровой интерактивный проект в пространстве мобильных приложений, выполняющих чаще всего какую-либо практическую функцию.

Кроме того, «SILENTIO» – это приложение для релаксации, которой пользователь достигает путем концентрации на медленных циклических интерактивных анимациях и прослушивания музыки определенного стиля.

Целью создания проекта также является самостоятельная оценка способности автора действовать и создавать в синтезе нескольких направлений, как то: создание художественного контента – иллюстраций и анимаций, программирование, создание музыки, оценка и применение современных технологий различной направленности.

Разработка мобильного приложения «SILENTIO», содержащего анимированные объекты, текст, музыкальное сопровождение, – сложный поэтапный процесс, требующий как классического, так и нестандартного подхода.

Создание столь разнообразного, но при этом уникального контента происходит зачастую при стековом использовании множества технологий, сочетание которых бывает неочевидным.

В данной работе решаются следующие задачи:

- Создание эскизов, дизайн прототипа в Figma;
- Запись звука, сведение, мастеринг;
- Создание анимаций на основе векторной графики прототипа;
- Разработка приложения на Android в среде Android Studio;
- Оценка функциональности приложения;
- Доработка функционала;
- Тестирование приложения;
- Публикация на Google Play;

Так как каждый дизайн в целом, каждый персонаж, кнопка или любой ассет в отдельности – неповторимое произведение, которое также может быть отчасти позаимствовано у коллег по цеху или у себя – стоит также целенаправленно произвести анализ предыдущих успешных работ, чтобы выявить некоторую тенденцию и выделить аспекты, применение которых будет положено в основу проекта «SILENTIO».

Используемые термины и определения

Маскот – практически любой узнаваемый персонаж, антропоморфный и не очень, олицетворяющий собой некий коллектив: школу, спортивную команду, сообщество, воинское подразделение, мероприятие или бренд.

Сплеш-скрин – (англ. Splash Screen, экран-заставка) – обычно первое, что видит пользователь на экране. Это изображение, которое появляется во время загрузки приложения или игры.

Туториал – это обучающий материал, инструкция, которая по шагам описывает процесс и рассказывает, как достичь конкретного результата (правила игры).

Геймплей (англ. gameplay), – компонент игры, отвечающий за интерактивное взаимодействие игры и игрока. Основной игровой экран приложения.

Ассет (англ. Game Asset) или Игровой ресурс – цифровой объект, преимущественно состоящий из однотипных данных, неделимая сущность, которая представляет часть игрового контента и обладает некими свойствами.

I. Опыт работы над дизайном мобильных приложений: анализ ранее реализованных интерфейсных решений

1. Предполагаемый персонаж

Так как мобильные игры – относительно новая область, универсальной теоретической базы по вопросам дизайна мобильного игрового интерфейса не было сформировано. В таком случае при создании художественной составляющей мобильных игр часто приходится опираться на собственное видение и опыт.

Одно из решений, успешно применяемых на практике – так называемый «маскот» в понимании игрового интерфейса. Это персонаж, который не участвует в процессе игры, но является ее олицетворением и осуществляет «личный контакт с пользователем» в традиционном понимании термина «маскот».

Так, персонаж игры «Mr. Lemos's Shell Game» – Лимон, который не участвует в процессе игры в наперстки (рис.2), но появляется в туториале и сплеш-скрине игры (рис.1), в финале игры (рис.3) и присутствует на рекламных материалах и материалах для публикации приложения. Таким образом, просматривая страницу игры в мобильном магазине, пользователь видит не обезличенные объекты, а вполне себе дружелюбного персонажа, который формирует личное отношение пользователя к игре еще до скачивания и непосредственного процесса игры.

Рисунок 1. Персонаж в туториале игры

Рисунок 2. Геймплей игры «Mr. Lemon's Shell Game"

Рисунок 3. Персонаж в финале игры

Второй пример – веселый садовник из детского приложения-игры «Найди пару». Традиционная механика всем известной игры оставляет не слишком большой простор для креатива, однако и здесь есть возможность подарить игре свое «лицо». Так, с этой целью был введен персонаж садовника, который появляется в меню игры, но в самом процессе игры также не участвует.

Рисунок 4. Меню игры «Найди пару»

Рисунок 5. Геймплей игры «Найди пару»

Еще один персонаж – детектив из игры «Стань детективом!». Здесь же «маскот» игры появляется исключительно на материалах для публикации, то есть на графике, предназначенной для Google Play Market: иконке, баннере и скриншотах.

Рисунок 6. Материалы для публикации игры «Стань детективом!»

2. Цветовые решения

Без сомнения, яркие, красочные цвета – залог привлекательности в той или иной мере, на первый взгляд или в долгосрочной перспективе, хоть яркие цвета и могут оказаться раздражающими для глаза. Привлекательность – в данном случае – понятие без эмоциональной окраски, так как первичная привлекательность не означает дальнейшую заинтересованность приложением, мобильной игрой или любым другим продуктом.

Умение управлять цветом – это умение также управлять настроением потенциального пользователя, а также управление его ассоциациями (рис. 7).

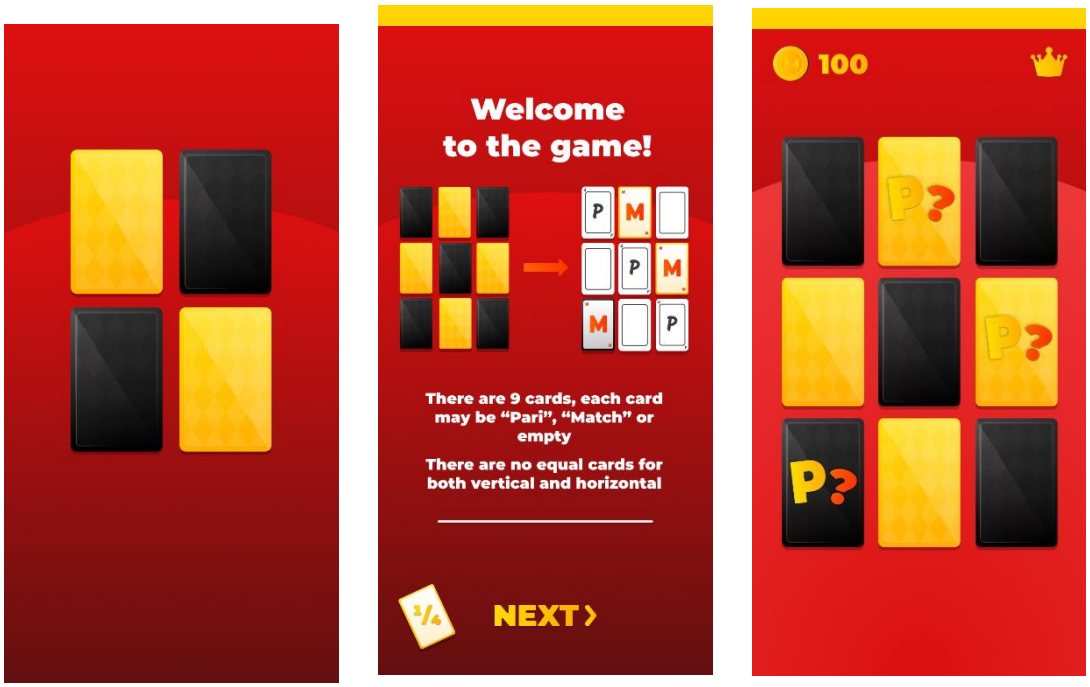


Рисунок 7. Игра "PariMatch" – рекламная игра, использующая фирменные цвета одноименного букмекера

Кроме классических цветовых сочетаний, таких, как контрастность (рис. 1 – 3) двух основных цветов, следует также выделить цветовые гаммы, базирующиеся на одном основном, при этом выразительном и ярком, цвете. Такова, например, цветовая гамма приложения на рисунке 6.

Чем же нужно руководствоваться при выборе цветовой гаммы дизайна игрового приложения? Разумеется, нет единственно верного ответа на этот вопрос, однако для каждого игрового приложения все-таки можно выделить заглавную цель, осознание смысла которой обычно помогает правильно расставить цветовые акценты. Цель эта может не зависеть, как в рекламных приложениях, или зависеть от функционала самой мобильной игры.

Так, цель приложения «SILENTIO» напрямую зависит от его функционала, а именно – расслабить пользователя с помощью циклически повторяющихся в медленном темпе анимированных картинок и соответствующей музыки.

Таким образом, очевидный в данном случае выбор – черно-белая цветовая гамма с черным фоном, как в вариантах интерфейса популярных приложений под названием «темная тема». Такой вариант цветовой гаммы призван снизить нагрузку на глаза, так как ограничивает яркость и количество света, исходящее от экрана устройства.

3. Художественная стилистика

Основной параметр художественной стилистики в мобильных играх, по мнению автора, – реалистичность графики. Двухмерная графика может кардинально отличаться от одной игры к другой – она может быть и совершенно «плоская», в стиле flat, либо же настолько объемная и реалистичная, что ее трудно будет отличить от фото или трехмерной графики. То же самое и со стилем персонажей: где-то совершенно неотличимые от изображений реальных людей, в ином случае они могут быть разве что антропоморфны.

Конечно, в определении художественного стиля для каждой конкретной игры можно руководствоваться теми же правилами, что и выше, в определении цветовой гаммы. Тут также основополагающим фактором является назначение приложения или игры.

Например, дизайн персонажей в игре «Bio Blast» на рисунке 8 неразрывно связан с дизайном персонажей игры-прототипа – «Candy Crush», так как целью первой игры является использование популярности второй для повышения интереса пользователей.

Рисунок 8. Персонажи игры «Bio Blast”

Инструменты, которые были использованы для достижения сходства, – «объемная» двухмерная векторная графика, созданная с помощью градиентов, векторных бликов, теней и рефлексов, определенный стиль глаз, а также округлая форма очертаний персонажей и похожие цвета.

Использование того или иного стиля влияет и на охват потенциальной аудитории игры. Будет ли стиль легким, детским, красочным или же сдержанным, строгим – все это, конечно, повлияет на восприятие игры пользователями, поэтому должно быть продумано заранее.

По авторской задумке, «SILENTIO» не столько игра, сколько приложение с игровым интерфейсом. Именно поэтому стиль графики для него заранее определяется скорее как строгий, нежели близкий к «детскому». Кроме того, в соответствии с ранее выбранной цветовой гаммой, понятно, что преобладающим фактором формирования стиля будет форма, а не цвет.

II. Анализ видов и возможностей анимаций, интегрируемых в приложения

GIF-анимации

GIF-анимации – один из самых распространенных видов анимаций, интегрируемых в приложения и веб-сайты. Простота разработки, универсальность в рамках воспроизведения, несложная интеграция – все это можно отнести к несомненным плюсам GIF-анимаций. Однако, так как GIF-анимации представляют собой последовательность кадров (изображений), имеют ограниченную цветовую палитру и, некоторым образом, сложны для рендеринга с прозрачным фоном, во многих случаях имеет смысл сделать выбор в пользу анимаций другого формата. Рассмотрим вышеперечисленные пункты подробнее.

1. Простота разработки.
   Получить GIF-анимацию одним из самых простых способов можно лишь сконвертировав ее из видеоформата любым удобным способом. Кроме того, экспорт в GIF поддерживается большинством программ для обработки видео и создания анимаций (Adobe After Effect + Media Encoder, Premiere Pro, Adobe Animate, Photoshop). Таким образом, несложно, например, создать анимированный туториал для IOS-приложений, имея в качестве исходника запись экрана устройства.

2. Универсальность в рамках воспроизведения и несложная интеграция.
   Воспроизведение GIF-анимаций поддерживается большинством сред и платформ. Интегрировать GIF-анимацию чаще всего не сложнее, чем интегрировать статичное изображение любого другого популярного формата.

3. Размер файла.
   Размер файла – следствие ранее определенного пункта: GIF-анимации представляют собой последовательность кадров (изображений). Данный фактор играет роль при оптимизации мобильного приложения ввиду того, что полноцветная GIF-анимация среднего размера с высокой частотой кадров (все вышеперечисленное можно отнести к описанию GIF-анимации высокого или среднего качества) может достигать размеров, скажем, 10 Мб, тогда как цельное приложение, не включающее данную гипотетическую GIF-анимацию, может и вовсе иметь меньший размер.

4. Ограниченная цветовая палитра.
   Цветовое ограничение GIF – 256 цветов. Разумеется, рендеринг GIF-анимаций из исходного изображения или видео может привести к искажению исходных цветов или появлению цветовых артефактов. Особенно данный фактор «мешает» при создании анимаций для мобильных приложений с одноцветным фоном, когда анимация, к примеру, должна быть круглой формы. Тогда, ввиду сложностей рендеринга GIF-анимации с прозрачным фоном, приходится иногда прибегать к созданию в GIF-анимации соответствующего фона, совпадающего с фоном приложения. Но, по причине ограниченности палитры, конечный цвет фона GIF-анимации после рендеринга может заметно отличаться от фона в приложении, из-за чего ее интеграция в приложение становится невозможной. Также большие трудности создают векторные исходники, градиенты, прочие элементы, включающие в себя большое количество цветов, так как GIF-анимация на основе подобных исходников будет сильно отличаться от статичного изображения, например, если анимация должна повторять, появляться после, накладываться на соответствующий векторный исходник – различия в качестве будут слишком заметны.

5. Сложность рендеринга анимаций с прозрачным фоном.
   При создании анимаций в After Effect одной из самых популярных программ для создания анимаций, в том числе, интегрируемых в приложения, отсутствует возможность прямого рендеринга GIF с прозрачным фоном даже при наличии Adobe Media Encoder. В таком случае, последовательность создания GIF с прозрачным фоном такова: анимирование в After Effect, экспорт анимации покадрово как последовательности png файлов с прозрачным фоном, сборка покадровой анимации в Photoshop, экспорт GIF с прозрачным фоном. Как видно, данная цепочка действий может вызывать затруднения, если, например, в готовую анимацию требуется внести изменения, так как весь цикл нужно было бы повторить заново. Кроме того, если в самой анимации, помимо прозрачного фона, изменяется прозрачность каких-либо элементов, задействованных в анимации, при экспорте GIF в Photoshop придется воспользоваться определенными настройками экспорта и так называемой «подложкой», без которой относительная прозрачность элементов, очевидно, не будет меняться. Подложка – избирательный фон для элементов, ее цвет можно мануально настроить, чтобы, например, придать ей цвет фона приложения, однако, как было выяснено в пункте 4, точный цвет при этом может исказиться.

Анимации в видеоформате

Проанализируем возможности анимаций в видеоформате на основе ранее выявленных возможностей GIF-анимаций. В данном контексте под видеоанимациями подразумеваются не видеоролики любого содержания, а классические анимации процессов для веб-сайтов и приложений, такие, как, например, анимация загрузки, анимация процесса очистки, сбора данных, анимированных персонажей и тому подобное.

Для сравнения, особенный интерес представляют последние два пункта анализа GIF-анимаций: цветовая палитра и рендеринг анимаций с прозрачным фоном. Первое, в случае анимаций в видеоформате, значительно улучшается: цветовая палитра расширяется, различные форматы предполагают практически не ограниченную цветовую палитру.

Что же касается рендеринга анимации с прозрачным фоном – в рамках классических программ для создания анимаций такими функциями обладает только один формат – QuickTime Mov, размер которого может также и превышать размер соответствующего файла GIF, что негативно сказывается на оптимизации приложения, когда анимация для сплеш-скрина может занимать больше памяти, чем целое готовое приложение.

Lottie-анимации

Lottie – это библиотека изображения для Android, iOS, React Native и Web, отображающая SVG-анимации из After Effects в реальном времени. Проще говоря, Lottie-анимации создаются традиционным способом в After Effect, поддерживают анимацию растровых и векторных файлов, а в последнем случае и сами являются векторными, то есть могут быть отмасштабированы в любом размере без потери качества. Традиционно Lottie-анимации рендерятся в файл json через расширение After Effect – Bodymovin, а в самом файле, подобно изображениям в формате SVG, находится закодированная с помощью тегов анимация.

Главные преимущества таких анимаций, без сомнения, векторность, а также предельно малый размер файла. Так, обычно размер одной анимации в формате json не превышает 1 Мб, а чаще всего он и вовсе составляет менее 100 Кб.

Не имеет смысла говорить о рендеринге анимации с прозрачным фоном или ограничениях цветовой палитры, более того, цвет в Lottie-анимации можно вообще поменять извне, просто обратившись к соответствующим тегам – само собой, при создании Lottie-анимаций таких ограничений нет.

Однако тут как раз важны иные ограничения – на данный момент список поддерживающих Lottie платформ и поддерживаемых Lottie эффектов при анимировании довольно короток. Так, у разработчиков нередко возникают сложности при интеграции Lottie в IOS приложения, так что в подобных случаях приходится делать выбор в пользу GIF, а в Unity Lottie-анимации пока что не поддерживаются совсем.

Что касается эффектов и возможностей самой анимации – пока что Lottie поддерживает только следующие из них: анимация формы (координат точек на кривых Безье) обводки и фигур, различные параметры и простейшие эффекты для анимации фигур и обводки (обрезка концов, закругленные углы, толщина обводки), градиенты для обводки и заливки и анимация сменяющихся цветов, ограниченные свойства масок, анимации текста, регулярных выражений и наложения фигур. Кроме того, Lottie не поддерживает рендеринг анимаций из вложенных композиций и большинство эффектов слоя, кроме размытия по Гауссу и тени.

К недостаткам Lottie-анимаций также можно отнести отсутствие свежей и разнообразной информации на русском языке, однако это относительно новая и пока еще непопулярная область, кроме того, практически на любой вопрос можно найти ответ в поисковой системе, если задать его по-английски.

Таким образом, ввиду всех преимуществ, а особенно малого размера файла на выходе, выбранный формат анимаций для приложения «SILENTIO» - Lottie.

III. Создание эскизов, дизайн прототипа в Figma

Итак, на рисунке 8 изображен эскиз проекта для проекта «SILENTIO», созданный в Photoshop в соответствии с ранее принятым решением о черно-белой цветовой гамме.

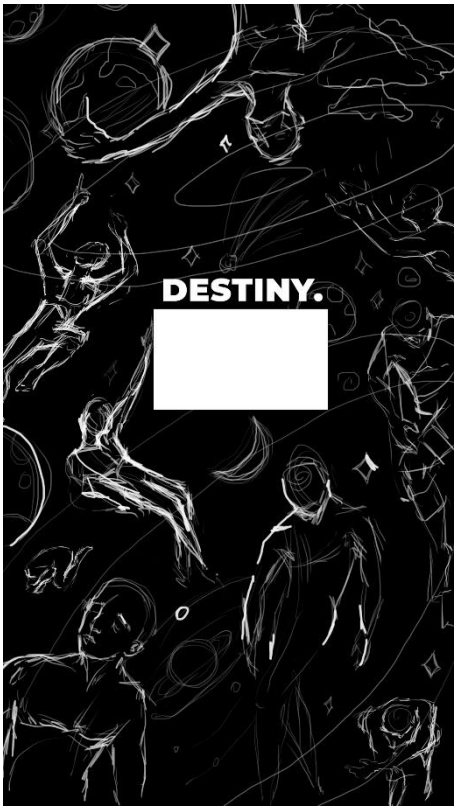


Рисунок 8. Эскиз проекта «Silentio»

Композиция эскиза – объекты располагаются по кругу, по рамке экрана, фон при этом также является функциональным (анимированным объектом), но менее выраженным, а текстовая составляющая располагается посередине экрана.

По замыслу приложения, пользователю предлагается запускать с помощью нажатия каждую отдельную анимацию персонажей, а целью является последующая релаксация пользователя под действием циклически повторяющихся анимаций в медленном темпе, можно предположить, что использование приложения может происходить в спокойной обстановке и в положении сидя, что сделает обе руки свободными для взаимодействия с элементами приложения и избавит от необходимости располагать функциональные элементы под конкретную руку.

Таким образом, данное композиционное решение, которое хоть и является больше художественным, нежели интерфейсным, можно считать достаточно эффективным для данной конкретной цели.

Кроме того, текст, окруженный функциональными элементами, но располагающийся в центре экрана, является композиционным центром, что также избавляет от необходимости наделять этим свойством какой-либо из функциональных элементов (анимированных персонажей), что было бы неверно согласно художественному замыслу.

Анимированные персонажи расположены на таком расстоянии друг от друга, чтобы обеспечить достаточную рамку, не перегружая пространство, но оставляя комфортные размеры иллюстрации для создания масштабных, относительно размеров статичных иллюстраций, анимаций.

Перед тем, как создать прототип приложения, нужно обозначить следующие ключевые моменты:

1. Определить «сюжет»;
2. Определить роли персонажей в сюжете;
3. Составить текстовые выражения для персонажей;
4. Определить функциональные особенности приложения.

Итак, сюжет приложения: на экране представлено 9 персонажей, каждый из которых является метафорой для какого-либо чувства, явления или переживания. В целом, по композиции и художественному значению, все они являются созвездиями.

Эти персонажи:

1. «Судьба» - персонаж, который располагается вниз головой с верхнего края экрана, держит в руках вращающуюся планету.
2. «Тянущийся к звездам» - персонаж, олицетворяющий погоню за мечтой и достижение ее. Плывя по воображаемому небосводу, тянется к звездам, касается их, после чего звезды мерцают.
3. «Лежащий» - персонаж, лежащий на невидимом фундаменте опыта и воспоминаний.
4. «Эмбрион» - персонаж, олицетворяющий детское в человеке.
5. «Оглядывающийся» - персонаж, который смотрит на пережитое.
6. «Шагающий» - персонаж, находящийся в непрерывном движении.
7. «Бессилие» - персонаж, охватывающий тело скованными от напряжения руками, вжимающий голову.
8. «Восстановление» - персонаж, снимающий старые бинты с заживших ран.
9. «Жажда» - персонаж, простирающий руки к дождю, символизирует стремление к новому.

Одним из анимированных объектов в приложении является фон, анимация которого будет запускаться автоматически после того, как будут запущены анимации всех девяти персонажей.

Таким образом, был создан следующий прототип:

Рисунок 9. Экраны прототипа из Figma.

Также определено рабочее название для проекта: «SILENTIO» - от лат. «тишина».
Анимация логотипа также будет осуществлена в соответствии с выбранными технологиями.

Следовательно, разработка прототипа происходила следующим образом:
1. Отрисовка в Animate по эскизу;
2. Редактирование в Illustrator (сокращение количества опорных точек для удобства создания последующей анимации);
3. Экспорт в svg;
4. Сборка прототипа в Figma;

Параллельно с созданием прототипа была осуществлена запись музыкальной композиции для приложения «SILENTIO».

IV. Запись звука, сведение, мастеринг

Музыкальная составляющая проекта – композиция, в составе которой ведущие гитарные и вспомогательные синтезаторные партии. В целях соответствия музыкальной композиции общей концепции и стилистике проекта было принято решение отказаться от партий ритм-секции: барабанов и баса, чтобы сохранить «плавность» звука и избавить его от ритмической градации.

Кроме того, композиция имеет схожие начало и окончание, также обработанные с помощью эффектов fade out\fade in, чтобы обеспечить возможность благозвучного циклического повторения композиции по ходу взаимодействия с приложением.

Для записи и обработки звука было использовано следующее оборудование:

1. Рабочая станция: Intel core i7-3770 3.40 GHz, 16 GB RAM, Windows 10 PRO 64-bit;
2. Музыкальный секвенсор FL Studio 10;
3. Звуковая карта Yamaha Steinberg UR12 (рис. 12);
4. Электрогитара Yamaha SG500 1978 г. (Япония) (рис. 10);
5. Полуакустическая электрогитара Cort TRG-1 1998 г. (Корея) (рис. 11);
6. Студийные наушники Sennheiser HD 215 (рис. 13);

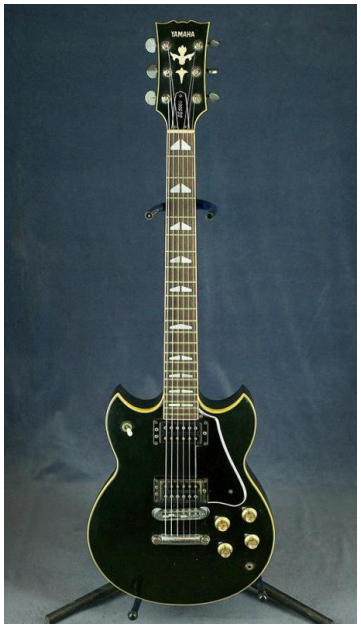


Рисунок 10. Yamaha SG500

Рисунок 11. Cort TRG-1

Рисунок 12. Yamaha Steinberg UR12

Рисунок 13. Sennheiser HD 215

В записи, как видно, были использованы две гитары по причине физических и акустических различий данных инструментов, иными словами, для разных отрезков композиции – арпеджио и игры аккордами – инструменты были удобны в разной степени. Так, имеющая большую мензуру и более широкий гриф электроакустическая гитара Cort больше удобна для игры арпеджио, а Yamaha – наоборот, для игры аккордами. Кроме того, акустические характеристики гитар также различны: Cort обеспечивает более глубокое звучание, богатое средними частотами за счёт некового хамбакера (звукоснимателя, расположенного у грифа инструмента), что смогло обеспечить отсутствие слишком звонких звуков, а Yamaha – наоборот, более частотно сбалансированное звучание при извлечении низких аккордов.

Так как звуковая карта получает на вход сигнал гитары, снятый звукоснимателем, а не записывает звук, воспринимаемый ухом при непосредственной игре в помещении, полученное таким образом звучание требовало последующей обработки внутри секвенсора, дабы приблизить его к натуральному звучанию акустического инструмента, что соответствовало бы концепции проекта.

Таким образом, концепция и стилистика гитарного звука такова: «космическое», плавное звучание, отсутствие перегруза, слишком звонких звуков и слишком низких частот.

Внутри секвенсора были использованы следующие плагины:

1. Guitar Rig 5: различные настройки для обработки звука гитар;
2. Steinberg Hypersonic 2: синтезатор Spacey Bell – партия синтезатора;
3. Edison: запись звука гитар;
4. Fruity Parametric EQ 2: эквалайзинг;
5. Maximus, Soundgoodizer, Limiter: мастеринг.

Рисунок 14. Графическое представление записанной композиции. Звуковые дорожки.

В результате работы над звуком продолжительностью более 2 месяцев было получено удовлетворяющее требованиям звучание.

Длина композиции составила 5 минут.
Темп: 110 ударов в минуту.
Композиция была экспортирована в формат mp3.
Количество промежуточных рендерингов: более 20.
Итоговый размер файла: 5,76 Мб.

V. Создание анимаций

Для создания анимаций в выбранном ключе необходимо, чтобы вся исходная графика была соответствующим образом подготовлена. Так как анимация будет осуществляться в Adobe After Effect с помощью, в основном, изменения координат точек на кривых Безье и изменения толщины обводки нужно, чтобы исходные векторные изображения состояли исключительно из кривых со свойствами обводки. Кроме того, дабы исключить всевозможные искажения и не усложнить задачу по созданию анимаций, количество опорных точек на кривых Безье для каждого изображения должно быть минимально возможным, но при этом достаточным, чтобы не нарушить художественную целостность каждого изображения.

Так, наиболее удачный способ для создания иллюстраций, удовлетворяющих вышеизложенным требованиям, – Adobe Animate + Adobe Illustrator. Да, возможно, программа Adobe Animate предназначена не совсем для создания иллюстраций, однако подход к упрощению кривых, сглаживанию и стиранию отрезков кривых в ней является наиболее удобным для данного конкретного проекта.

Adobe Illustrator в данном стеке выступает как удобный инструмент для регулирования количества опорных точек на кривых Безье, так как при использовании специального средства упрощения кривых в соответствующем окне содержится поле с непосредственным указанием количества опорных точек на кривых у конкретного изображения.

Обмен файлами (изображениями) между указанными программами осуществляется с помощью экспорта иллюстрации в формат SVG и, соответственно, последующего импорта.

Финальный этап в подготовке изображения для создания анимаций – сохранение его в формате Adobe Illustrator .ai. Это векторный формат, импорт которого непосредственно поддерживается программой Adobe After Effect.

Кроме экспорта, векторные слои в составе .ai файла могут быть также, с помощью одной команды, интерпретированы в векторные объекты внутри After Effect с сохранением соответствующих свойств цвета заливки, цвета и толщины обводки, положения на холсте и так далее для каждого слоя и объекта в составе него.

Таким образом, каждая подготовленная векторная иллюстрация без потерь открывается в After Effect и является готовой для создания анимации.

Рендеринг готовой анимации производится в формат Lottie-анимаций, json, с помощью встроенного в After Effect расширения Bodymovin. Также, для

большего удобства тестирования, производится дополнительный рендеринг в формат html, который воспроизводится в браузере и запускается как обычная веб-страница.

Следовательно, процесс создания каждой анимации выглядит так:

1. Подготовка иллюстрации, выполненная на этапе создания прототипа;
2. Импорт файла .ai в среду Adobe After Effect;
3. Создание анимации;
4. Рендеринг полученной анимации в файл json с превью в html с помощью расширения Bodymovin.

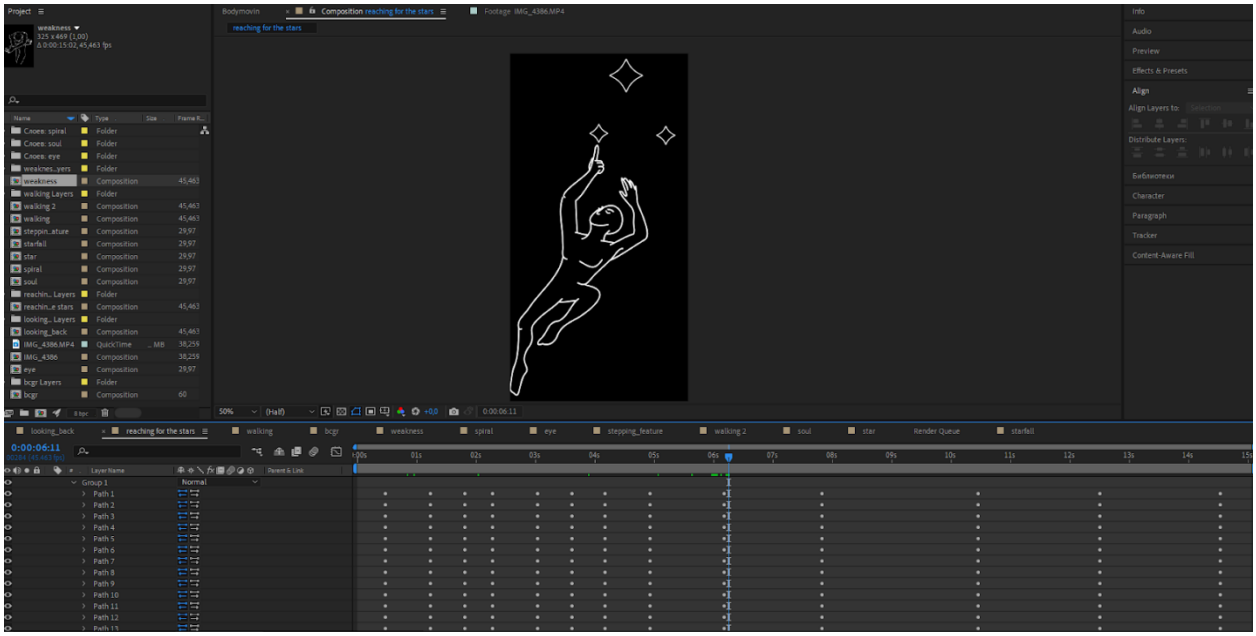


Рисунок 15. Создание анимации в After Effect.

Таким образом, был создан весь необходимый для приложения «SILENTIO» контент, что позволило перейти к следующему этапу – созданию разметки интерфейса в среде Android Studio.

VI. Разработка: создание разметки на основе прототипа

Когда создание художественного и музыкального контента для приложения считается завершенным, можно приступать к разработке приложения в среде Android Studio.

Первым этапом разработки будет создание разметки приложения в соответствии с созданным ранее прототипом.

Среда и принципы разметки Android Studio располагают некоторым количеством макетов. Все они хороши и удобны в различных специфических ситуациях, и выбор пал на современный макет под названием ConstraintLayout. Основная особенность макета ConstraintLayout состоит в том, что разметка создается с помощью так называемых Constrains – линий разметки, на основе которых располагаются объекты View внутри ConstraintLayout. Проще говоря, расположение объекта является относительным по отношению к другим объектам на экране, объекты «привязываются» друг к другу.

Так как прототип приложения по своей структуре не имеет ничего общего с табличным расположением объектов или иной структурой, легко поддающейся однозначной характеристике, а объекты, скорее расположены «хаотично», согласно лишь внутреннему художественному смысловому стержню – данный макет наиболее удачен для разметки в данном случае.

Итак, первый этап в создании разметки – импорт всех готовых анимаций в среду, определение фона в черном цвете.

Следующий – добавление анимаций на экран. И на этом этапе уже возникла некоторая трудность, так как какое-либо превью анимаций на экране в среде Android Studio не отображалось ввиду специфичности самого объекта, так что все объекты анимаций отобразились в качестве прямоугольников с заданными в xml-файле размерами (рисунок 8). Конечно, анимации отображались в нужном виде в процессе эмуляции и на физическом устройстве, и на эмуляторе устройства в Android Studio, однако, специфичность данных файлов анимаций также состоит в том, что анимации по своей сути векторные, а значит, изменение ширины и высоты всегда происходит пропорционально.

Иными словами, размер объекта анимации на экране подстраивался под размер определенного для него контейнера, а изменение ширины или высоты контейнера при верстке вело к практически непредсказуемому изменению

размеров самой анимации, которая, в свою очередь, согласно макету должна была иметь довольно строгие размеры.

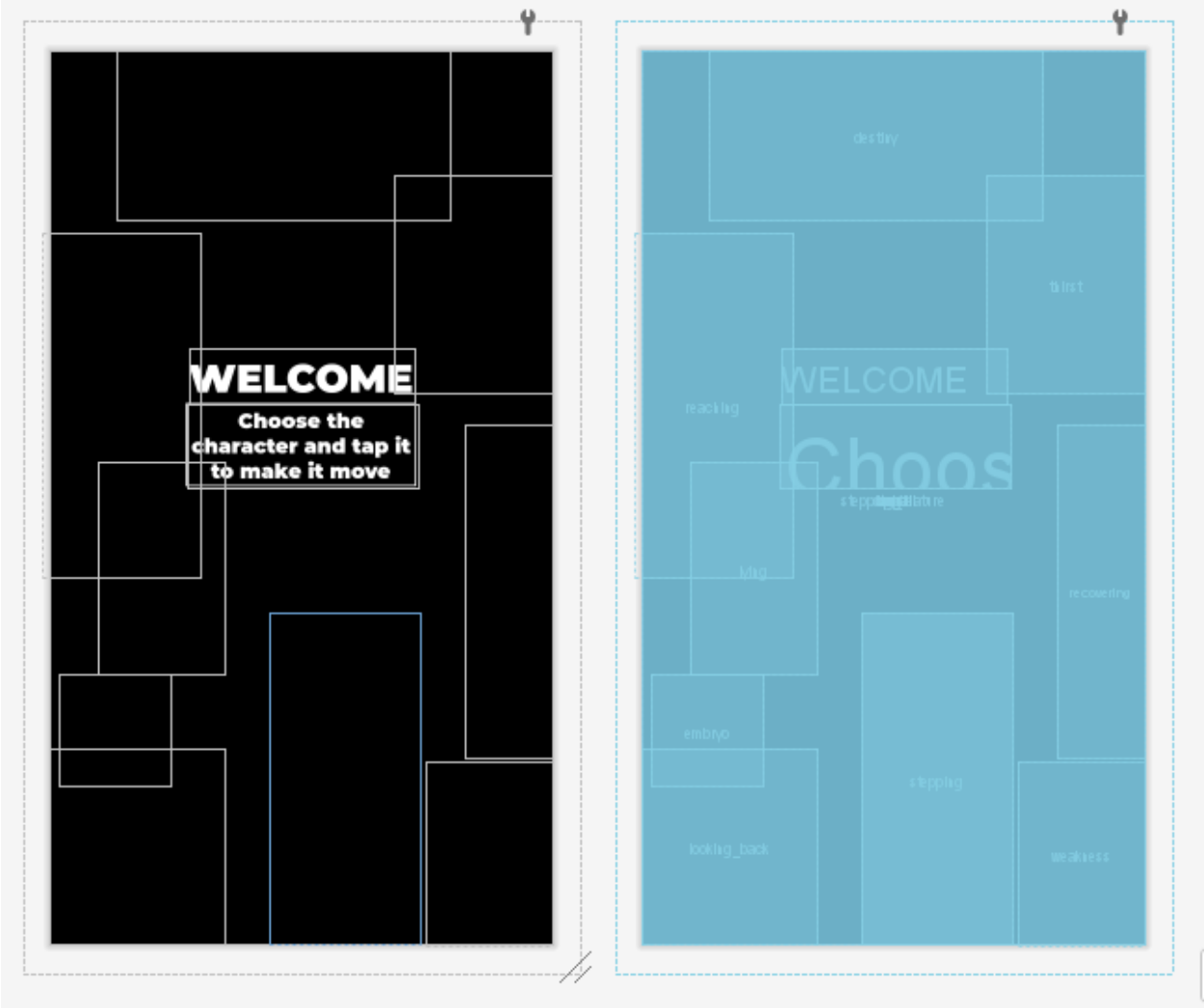


Рисунок 16. Отображение объектов анимаций внутри Android Studio.

Поэтому в начале работы с разметкой приходилось производить ее практически вслепую. К счастью, довольно скоро было изучено, что объектам при разметке можно передавать соотношение сторон, что спасло ситуацию и облегчило работу.

Соотношения сторон легко вычислялись, так как при открытии файла json с анимацией в текстовом редакторе в первой же строчке значится примерно следующее: "w":1498,"h":2623, что практически означает размеры высоты и ширины объекта.

Таким образом, этап импорта анимаций и настройки их размеров был завершен.

Следующий этап – настройка местоположения объектов на экране. Здесь, согласно прототипу, текст располагался в центре экрана с некоторым отклонением по вертикали (вверх), это довольно простая настройка. Настройки расположения объектов анимации тоже дались относительно легко с помощью удобного графического интерфейса среды даже без использования кода. Следующий этап – тестирование разметки на разных устройствах.

На этом, казалось бы, финальном этапе создания разметки и проявилась главная сложность всего процесса разметки. Вопреки ожиданиям, размеры объектов не менялись относительно размеров экрана и на малом экране выглядели слишком большими, а на крупных экранах с высоким разрешением – слишком маленькими.

Спустя некоторое количество времени, затраченного на поиск решения, оно было, наконец, найдено. А именно – использование библиотек Android SSP и SDP. Это библиотеки, которые используют новую единицу измерения отдельно для текста и объектов, которая обеспечивает более пропорциональное изменение размеров объектов и текста относительно размеров экрана устройства.

Подключив данные библиотеки, пришлось заново настроить размеры объектов и текста с использованием новых единиц измерения, однако результат не оставил сомнений в эффективности работы библиотек.

После подключения библиотек размеры анимаций и текста на экранах разных диагоналей и разрешения выглядели так, как и было задумано, и соответствовали прототипу.

Кроме непосредственной настройки местоположения анимаций на экране прямо в xml-файле всем анимациям были переданы некоторые ключевые свойства: `app:lottie_autoPlay="false"` и `app:lottie_loop="true"`, которые означают, что анимации не проигрываются по умолчанию, то есть, изначально имеют статус pause, и что воспроизведение анимаций зациклено.

Таким образом, разметка приложения была создана и прошла первый этап тестирования, что позволило перейти к следующему этапу – созданию программного кода.

VII. Разработка: определение механики взаимодействия

Итак, переходим к созданию программного кода, обеспечивающего взаимодействие пользователя и приложения, прежде определив схему, по которой оно будет осуществляться.

Механика или же сценарий взаимодействия был описан ранее тезисно, теперь определяем его окончательно:

- Пользователь запускает анимацию, выбирая персонаж.
- После нажатия на персонажа в центре экрана появляется его название и описание.
- После повторного нажатия анимация персонажа останавливается в текущем положении.
- Когда анимации всех девяти персонажей запущены, запускается анимация фона.

Схема реализации:

- Создается класс, поля которого - анимация, название (текст), описание (текст).
- Все объекты класса, по количеству соответствующие количеству анимаций, добавляются в список.
- Обрабатывается нажатие на анимации в цикле for по списку, проверяется, запущена анимация или остановлена, выполняются соответствующие действия: либо остановка в первом случае, либо запуск с установкой названия и описания во втором.

Таким образом, последовательность действий для разработки конкретно определена механикой взаимодействия и схемой реализации, и можно переходить непосредственно к созданию программного кода приложения на языке Java.

VIII. Разработка: Java

При разработке программного кода были использованы классические подходы, обеспечивающие возможность дальнейшего расширения функционала, добавления дополнительных элементов при необходимости.

Конечно, при таком сравнительно небольшом количестве элементов контента, как в приложении «SILENTIO» - 9 анимаций персонажей, 2 текстовых поля, 1 анимация фона, можно было бы обойтись написанием механики взаимодействия для каждого элемента в отдельности, и такой подход даже не повлек бы за собой значительного увеличения количества строк, однако большое число повторяющихся паттернов в коде – стилистически ошибочное решение.

Поэтому, как и определено в схеме реализации, при написании программного кода будет использоваться в том числе классовый подход для создания объектов механики взаимодействия.

Итак, этапы разработки по схеме:

1. Создан класс AnimationText с полями animation, title и description, целью которого является объединение одного файла анимации и соответствующих ему названия и описания в один объект, вследствие чего упрощается работа с взаимодействием по нажатию, так как при нажатии на анимацию название и описание «берется» прямиком из объекта, которому принадлежит данная конкретная анимация. Соответственно, классу добавлены методы getAnimation(), getTitle(), getDescription().

2. Создан метод initAnimationText(), в котором происходит инициализация анимаций и текста.

3. Создан метод addToList(), добавляющий объекты AnimationText в список.

4. Создан метод interactAnimation(), обрабатывающий нажатие на анимацию с помощью OnClickListener.

5. Создан метод onAnimationClicked(AnimationText animationText), содержащий действия по нажатию на анимацию, вызов метода добавлен в interactAnimation(). Метод проверяет, запущена ли данная анимация (в этом случае анимация приобретает статус pause) или же

остановлена (в этом случае анимация приобретает статус resume). Также проверяется условие для анимации фона.

6. После тестирования полученного результата также был добавлен метод setParameters(), который устанавливает параметры для некоторых анимаций, как то: скорость, стартовый кадр.

7. Создан класс MysicPlayer для воспроизведения музыки. Настройки воспроизведения выставлены так, чтобы музыка начинала проигрываться с запуском приложения (то есть, уже на этапе экрана загрузки, а не основного экрана), а останавливалась, когда приложение закрывается (onDestroy()).

Некоторые примечания:

Так как музыка в данном приложении является самостоятельной частью инсталляции, было принято решение не заканчивать воспроизведение при сворачивании приложения, а только при полном закрытии, как это делается, например, у приложений различных музыкальных плееров.

Таким образом, получился довольно лаконичный код, который можно полностью привести:

```
@Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);

        initAnimationText();
        addToList();
        setParameters();
        interactAnimation();
    }

    private void initAnimationText() {
        //init text
        title = findViewById(R.id.title);
        description = findViewById(R.id.description);
        //init main animations
```

```
        animation_Text_stepping = new
AnimationText(findViewById(R.id.stepping), R.string.stepping,
R.string.description_stepping);
        animation_Text_reaching = new
AnimationText(findViewById(R.id.reaching), R.string.reaching,
R.string.description_reaching);
        animation_Text_thirst = new
AnimationText(findViewById(R.id.thirst), R.string.thirst,
R.string.description_thirst);
        animation_Text_lying = new
AnimationText(findViewById(R.id.lying), R.string.lying,
R.string.description_lying);
        animation_Text_weakness = new
AnimationText(findViewById(R.id.weakness), R.string.weakness,
R.string.description_weakness);
        animation_Text_looking_back = new
AnimationText(findViewById(R.id.looking_back),
R.string.looking_back, R.string.description_looking_back);
        animation_Text_embryo = new
AnimationText(findViewById(R.id.embryo), R.string.embryo,
R.string.description_embryo);
        animation_Text_recovering = new
AnimationText(findViewById(R.id.recovering),
R.string.recovering, R.string.description_recovering);
        animation_Text_destiny = new
AnimationText(findViewById(R.id.destiny), R.string.destiny,
R.string.description_destiny);
        animation_bcgr = findViewById(R.id.bcgr);
     }

    private void addToList() {
        animationTexts = new ArrayList<>();
        animationTexts.add(animation_Text_stepping);
        animationTexts.add(animation_Text_reaching);
        animationTexts.add(animation_Text_thirst);
        animationTexts.add(animation_Text_lying);
        animationTexts.add(animation_Text_weakness);
        animationTexts.add(animation_Text_looking_back);
        animationTexts.add(animation_Text_embryo);
        animationTexts.add(animation_Text_recovering);
        animationTexts.add(animation_Text_destiny);
    }

    private void setParameters() {
        animation_Text_reaching.getAnimation().setFrame(310);
        animation_bcgr.setSpeed(0.35f);
        animation_Text_lying.getAnimation().setSpeed(1.2f);
    }

private void interactAnimation() {
        for (AnimationText animationText : animationTexts) {
            animationText.getAnimation().setOnClickListener(v ->
{
```

```
                onAnimationClicked(animationText);
            });
        }
 }

    private void onAnimationClicked(AnimationText animationText)
{
        if (!animationText.getAnimation().isAnimating()) {
            animationText.getAnimation().resumeAnimation();
            title.setText(animationText.getTitle());
          description.setText(animationText.getDescription());
      } else {
            animationText.getAnimation().pauseAnimation();
        }

        boolean allAnimated = true;

        for (AnimationText a : animationTexts)
            allAnimated = allAnimated &&
a.getAnimation().isAnimating();

        if (allAnimated)
            animation_bcgr.resumeAnimation();
        else
            animation_bcgr.pauseAnimation();
    }
}
```

Как видно, таким образом в методе onCreate удалось обойтись вызовом небольшого количества методов.

Работоспособность кода проверена: все работает исправно, как и было задумано.

Стоит также отметить, что параллельно по той же технологии был создан экран загрузки с анимацией логотипа, для которого были написаны свой класс и xml-файл: экран загрузки включает в себя две анимации – фоновую, такую же, как и на основном экране, но сразу в активном состоянии, и анимацию логотипа. Время отображения: 6000 мс.

Таким образом, разработка на Java в соответствии с ранее составленной схемой была завершена.

IX. Разработка: расширение функционала

На финальном этапе разработки «SILENTIO» было принято решение расширить функционал путем добавления “секретных” анимаций, появление которых было бы следствием активации основных персонажей в определенной последовательности.

Было реализовано 5 дополнительных анимаций и два метода, обеспечивающих их работу. Кроме того, вследствие неравномерного появления объектов на экране после загрузки был также написан метод, обеспечивающий плавное появление элементов композиции.

Вообще, расширение функционала концептуально готового приложения – процесс непростой, ведь, согласно изначальному плану, общая композиция уже выглядит целостно. Путем долгих умозрительных экспериментов было выявлено, что создание чего-либо мало-мальски сложного и наделенного особым художественным смыслом повлечет за особой разрушение выстроенной концепции. Однако и создание чего-либо слишком простого не впишется в концепт и только утяжелит его.

Поэтому и родилась идея создания технически более простых, коротких, однократно появляющихся анимаций, которые в то же время сюжетно вписываются в концепт. С точки зрения механики взаимодействия такой тип расширения функционала тоже отлично показал себя. Если такие дополнительные анимации будут скрыты в сюжете, это само по себе создаст новый игровой процесс внутри уже готовой инсталляции – пользователю будет предложено отыскать все спрятанные анимации. Обнаружение же самих анимаций в интерфейсе не обязательно специально обозначать, ведь согласно первоначальному и основному замыслу пользователь «включает» анимации персонажей в случайной последовательности, а значит, дополнительные анимации он тоже обнаружит случайно. Заметив одну дополнительную анимацию, пользователь обратит внимание на изменение на экране и будет настроен на поиск остальных. Таким образом, маленькая идея положила начало еще одному интерактивному процессу в приложении, который развился самостоятельно естественным образом. А значит, реализация данной идеи будет простой.

Итак, анимации получились следующие:

Рисунок 17. Дополнительные анимации.

1. Спираль. Появление анимации обусловлено приведением в активное состояние двух анимаций: «Бессилие» и «Оглядывающийся». В данном контексте смысл анимации можно трактовать как следствие двух отрицательных ощущений, поэтому спираль – «затягивание» отрицательных эмоций. Визуально такая анимация также оказывает гипнотический эффект.

2. Глаз. Появляется при условии активности анимаций «Судьба» и «Оглядывающийся». Таким образом, смысл – «взгляд в душу», око, обращенное внутрь.

3. Душа. Появляется при условии активности анимаций «Восстановление», «Бессилие», «Тянущийся к звездам». Визуально это силуэт человека, летящий вверх, раскидывающий руки в полете. Анимация символизирует прощение, очищение, когда человек отпускает свою боль. Буквально – катарсис.

4. Шагающий. Это вариация уже представленной в первоначальном концепте анимации. Однако, решение акцентировать на ней больше внимания не было бессмысленным. Движение вперед при любых обстоятельствах – основной принцип преодоления. Активируется, если активны анимации «Шагающий» и «Судьба» на основном экране.

5. Звездопад. Активируется при активном состоянии анимаций «Тянущийся к звездам» и «Жажда». Здесь идея более визуальна, и, наверное, более проста, логична – в анимации «Тянущийся к звездам» антропоморфное существо касается звезд, в анимации «Жажда» - ловит

падающие капли дождя. Таким образом, при слиянии этих двух анимаций появляются падающие звезды как символ сбывающихся мечтаний, загадывания желаний, удачи.

Что же касается программной реализации, получились следующие методы:

1. featureBindCheck(AnimationText... a) – буквально, «проверка связки». Данный метод проверяет, выполняется ли условие активности «нужных» анимаций и имеют ли «ненужные» анимации статус pause. Проверка происходит путем создания двух булевых переменных: needfulOn и uselessOff, а также копирования исходного списка объектов AnimationText в новый список под названием animationPaused.
   Далее в цикле for из списка animationPaused удаляются «нужные» анимации, переданные методу на вход, а также проверяется, все ли они активны - полученное в итоге значение передается переменной needfulOn.
   В следующем цикле for проверяется, все ли оставшиеся в списке animationPaused анимации имеют статус pause. Полученное значение передается переменной uselessOff.
   В результате метод выводит needfulOn & uselessOff, что как раз и является нужной проверкой – «все ли нужные включены, а ненужные выключены».
   Также можно заметить, что на вход методу можно передавать различное количество аргументов, а значит, его работа является довольно гибкой и подходит для различных творческих решений.

2. featureOn(boolean b, LottieAnimationView featureAnimation) – метод, который «включает» дополнительную анимацию. Он получился довольно коротким и содержит всего одну строку. На вход передается значение, полученное в предыдущем методе, а также сама дополнительная анимация, которую требуется активировать.

3. appearAll() – также довольно простой метод, который обеспечивает плавное появление объектов на экране после загрузки приложения. Внутри него производится анимация всех объектов на экране согласно отдельно созданному xml-файлу с анимацией (alpha).

```
private boolean featureBindCheck(AnimationText... a) {
        List<AnimationText> animationPaused = new
ArrayList<>(animationTexts);
        boolean needfulOn = true;
        boolean uselessOff = true;

        for (AnimationText animationText : a) {
            animationPaused.remove(animationText);
            needfulOn = needfulOn &
animationText.getAnimation().isAnimating();
        }

        for (AnimationText animation : animationPaused)
            uselessOff = uselessOff &
!animation.getAnimation().isAnimating();

        return needfulOn & uselessOff;

    }

    private void featureOn(boolean b, LottieAnimationView
featureAnimation) {
        if (b) featureAnimation.resumeAnimation();
    }
```

```
private void appearAll() {
        Animation alpha = AnimationUtils.loadAnimation(this,
R.anim.alpha);
        for (AnimationText animation : animationTexts) {
            animation.getAnimation().startAnimation(alpha);
        }
        title.startAnimation(alpha);
        description.startAnimation(alpha);
    }
```

Вызов данных методов добавлен в метод interactAnimation() и выглядит там следующим образом:

```
private void interactAnimation() {
        for (AnimationText animationText : animationTexts) {
            animationText.getAnimation().setOnClickListener(v ->
{
                onAnimationClicked(animationText);

featureOn(featureBindCheck(animation_Text_weakness,
animation_Text_looking_back), spiral);

featureOn(featureBindCheck(animation_Text_destiny,
animation_Text_looking_back), eye);

featureOn(featureBindCheck(animation_Text_stepping,
animation_Text_destiny), stepping_feature);

featureOn(featureBindCheck(animation_Text_recovering,
animation_Text_reaching, animation_Text_weakness), soul);

featureOn(featureBindCheck(animation_Text_thirst,
animation_Text_reaching), starfall);
            });
        }

    }
```

Таким образом, расширение функционала прошло «безболезненно» как для смысловой концепции приложения, так и для кода, который так и остался лаконичным и не был утяжелен.

X. Тестирование приложения

Во время тестирования приложения были выявлены и устранены все проблемы с разметкой, настроена подходящая анимация появления объектов. В конечном счете, готовое приложение работало исправно и выполняло все заявленные основные и дополнительные функции и было протестировано на различных физических и виртуальных устройствах.

В целом, характеристики готового приложения получились следующие:

- Суммарный вес анимаций: 1 Мб (17 файлов)
- Вес музыки: 5,76 Мб
- Вес готового приложения: 8,44 Мб

Рисунок 18. Сравнение эскиза, прототипа и экрана реального приложения.

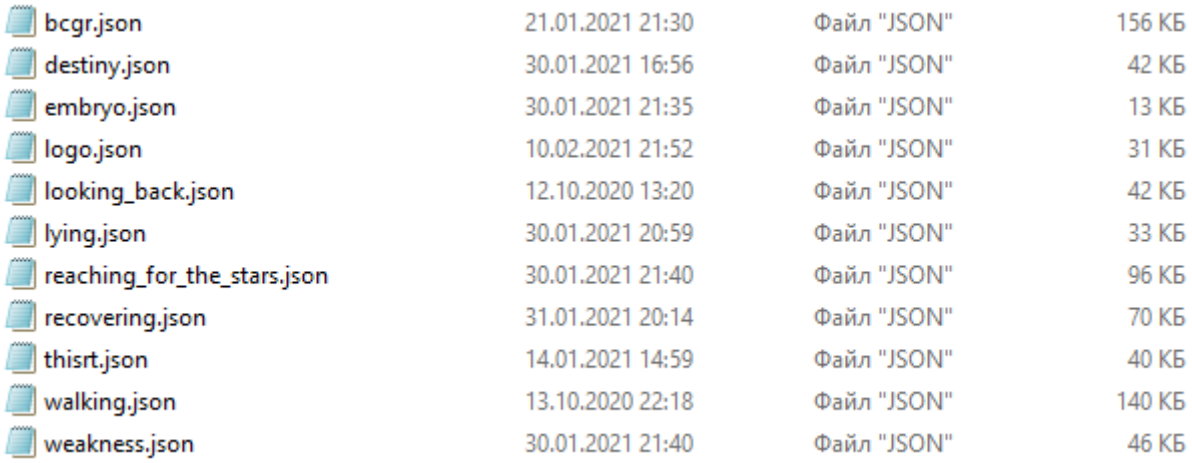
На рисунке 18 приведено сравнение эскиза, прототипа в Figma и экрана получившегося приложения. Как можно заметить, результат в целом соответствует ожиданиям, изменение же положения объектов на экране связано разве что с изменением соотношения сторон экрана на прототипе и физическом устройстве.

Готовое приложение имеет 2 экрана: экран загрузки и основной, 4 класса: для музыкального плеера, экрана загрузки, основного экрана и вспомогательный класс для анимации, ее названия и описания.

Важную роль при тестировании приложения перед публикацией играло получение обратной связи.

Пользователи, которым было предложено опробовать приложение, положительно оценили его, отдельно отметив музыку. Кроме того, пользователи обратили внимание на механику воспроизведения музыки, а именно на то, что воспроизведение не заканчивается при сворачивании приложения, и также нашли это интересным.

Положительным результатом, кроме прочего, является компактный размер готового приложения. Это достижение получено в основном благодаря технологии Lottie, ведь средний вес одной анимации составил 60 Кб.



bcgr.json	21.01.2021 21:30	Файл "JSON"	156 КБ
destiny.json	30.01.2021 16:56	Файл "JSON"	42 КБ
embryo.json	30.01.2021 21:35	Файл "JSON"	13 КБ
logo.json	10.02.2021 21:52	Файл "JSON"	31 КБ
looking_back.json	12.10.2020 13:20	Файл "JSON"	42 КБ
lying.json	30.01.2021 20:59	Файл "JSON"	33 КБ
reaching_for_the_stars.json	30.01.2021 21:40	Файл "JSON"	96 КБ
recovering.json	31.01.2021 20:14	Файл "JSON"	70 КБ
thisrt.json	14.01.2021 14:59	Файл "JSON"	40 КБ
walking.json	13.10.2020 22:18	Файл "JSON"	140 КБ
weakness.json	30.01.2021 21:40	Файл "JSON"	46 КБ

Рисунок 12. Размеры основных анимаций.

Таким образом, получилось мобильное приложение, суммарный размер всех ресурсов которого (анимаций, музыки) составил 6,76 Мб, что составляет 80% от общего размера приложения. Также приложение было подготовлено к дальнейшей публикации.

XI. Публикация на Google Play

Решение опубликовать приложение на Google Play было принято, когда разработка еще не была завершена. Основания для публикации – упрощение получения обратной связи, непосредственно получение опыта публикации, в дальнейшем также получение опыта продвижения приложения.

В настоящий момент необходимыми и достаточными условиями для публикации на Google Play является уплата членского взноса в размере 25$, наличие самого приложения и сопроводительных материалов к нему.

Сопроводительные материалы включают в себя пиктограмму приложения, скриншоты, которые могут быть оформлены различным образом с использованием надписей и графики, а также баннер приложения, который отображается в особых случаях: при действии «поделиться», при публикации приложения в топ Google Play. Все сопроводительные материалы должны быть созданы с учетом требований к публикации Google Play – в основном, должны иметь определенные размеры и формат.

В целом, создание сопроводительных материалов не вызвало затруднений, так как опыт их создания довольно значителен и включает более сотни уже опубликованных материалов для различных приложений.

Рисунок 19. Сопроводительные материалы для публикации приложения.

Непосредственно для публикации приложения создается аккаунт разработчика Google Play, а после уплаты членского взноса и подтверждения аккаунта становится доступна Google Play Console, в которой как раз и производятся все необходимые операции для публикации приложения. Эти операции включают в себя заполнение значительного количества анкет, например, для определения допустимого возраста пользователей, а также определение списка стран, для которых будет производиться публикация, и

загрузку сопроводительных материалов, самого приложения, ввода его названия и описания.

Как можно заметить, было принято решение использовать для надписей в сопроводительных материалах только английский язык, без локализаций для разных стран, так как само приложение тоже выполнено с использованием исключительно английского языка.

После выполнения всех необходимых действий и перед самой публикацией приложение проходит проверку, длительность которой в данном случае составила 10 дней.

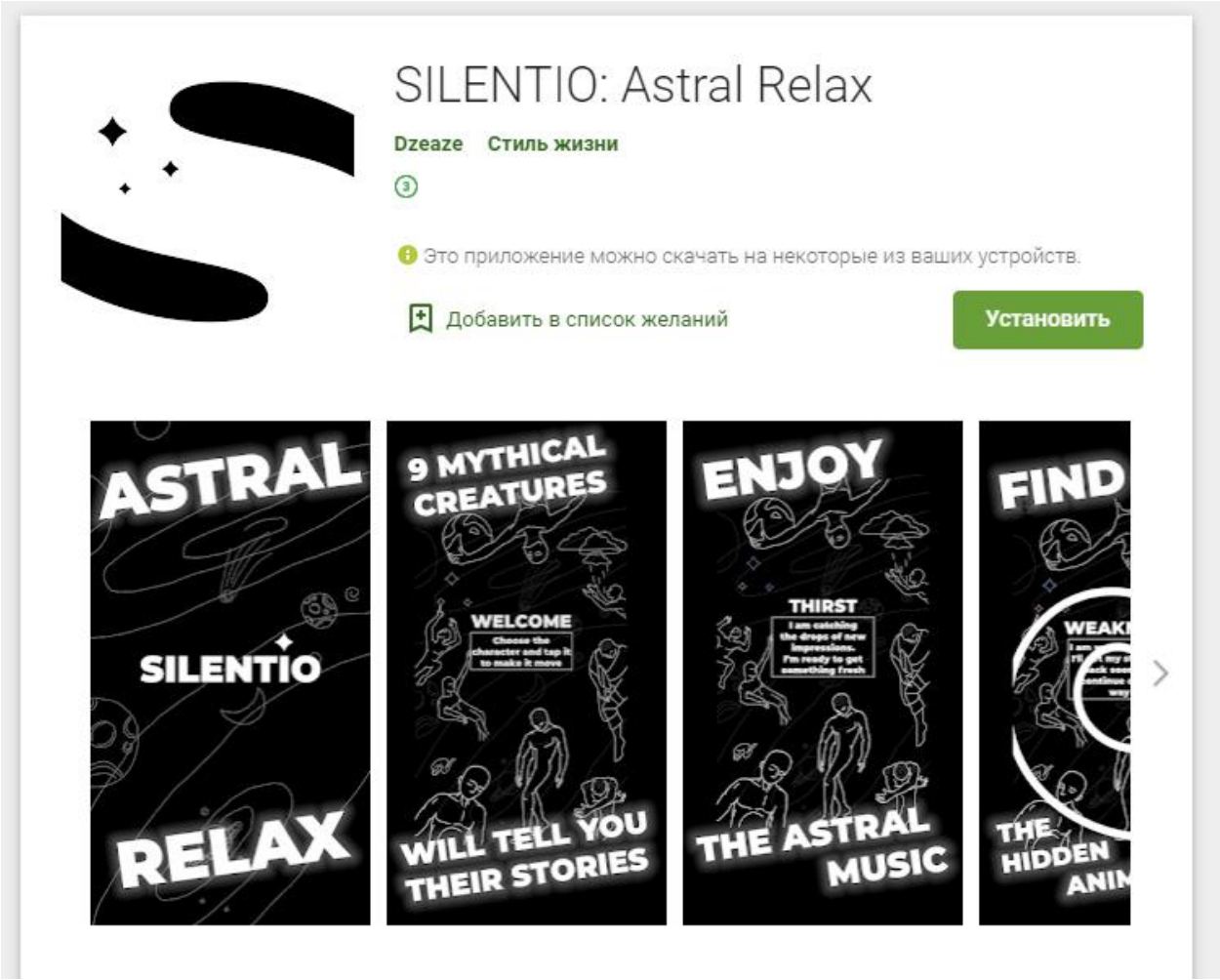


Рисунок 20. Скриншот страницы опубликованного приложения в Google Play.

Важным решением, принятым именно по причине публикации, является расширение названия: «SILENTIO: Astral Relax» вместо «SILENTIO». Решение было принято, чтобы увеличить количество попаданий приложения в поисковую выдачу, если производится поиск приложений для релаксации.

На наш взгляд, важным замечанием также является то, что публикация на Google Play, конечно, не гарантирует успех приложения и огромное количество скачиваний.

На сегодняшний день количество приложений, опубликованных на Google Play, составляет более 3 миллионов доступных единиц. Поэтому стратегия публикации и монетизации приложений в большинстве случаев включает грамотное продвижение, рекламу в различных источниках.

Однако стоит также отметить, что и востребованность тематики самого приложения играет немаловажную роль.

В настоящее время экспертами было выявлено, что скачивания приложений различной тематики неоднородны. Иными словами, появилась тенденция к скачиванию приложений узкого круга направленности. Данная тенденция логично объясняется тем, что пользователи, подобрав для себя оптимальный набор приложений-утилит (вспомогательных приложений для выполнения специализированных типовых задач, связанных с работой оборудования и операционной системы), социальных сетей, мессенджеров, развлекательных порталов и прочего, не стремятся заменять выбранные единицы и скачивать что-либо другое. Данный тезис как раз можно объяснить колоссальным количеством приложений на Google Play в целом, а значит, также и значительным количеством приложений-аналогов по каждой тематике. Кроме того, появилась также тенденция в разработке комплексных приложений, которые могут выполнять множество функций, «вшитых» в одну среду. Например, в приложении для заказа такси сейчас также можно заказать продукты, а обе эти услуги можно получить в приложении социальной сети.

Приложениями, количество скачиваний которых можно значительно увеличить за короткий срок, сейчас являются игры.

И, так как «SILENTIO» по авторскому определению является приложением с игровым интерфейсом, его продвижение может также увенчаться успехом.

XII. Варианты дальнейшего развития приложения

Приложение «SILENTIO» – концептуально целостная аудиовизуальная инсталляция, однако, что является, на наш взгляд, положительной характеристикой, подразумевает также некоторые пути дальнейшего развития:

- Создание нескольких различных экранов с подобной механикой взаимодействия, разные сюжеты.
- Расширение интерактивных возможностей текущего приложения.
- Расширение музыкальной библиотеки.
- Запуск приложения на других платформах.
- Доработка или переработка механики, визуала, музыки после анализа заинтересованности пользователей.

Создание нескольких различных экранов приложения с различными «сюжетами» (другими персонажами, анимациями, стилями) – логичный путь развития, который, кроме прочего, подчеркивает концептуальную завершенность настоящей версии приложения. Для данного варианта развития потребуется создание дополнительного экрана – меню для выбора «сюжета». В целом, приложение может приобрести вид некой галереи настроения, все «сюжеты» (экраны) будут иметь свой цвет-доминанту, свое музыкальное сопровождение, особенных персонажей, объединенных в специфическую метафору. Вопрос стилистики дополнительных экранов в данном случае остается открытым – будет ли это единый стиль, или же все экраны будут отличаться друг от друга – на текущем этапе покажет только время и обратная связь, полученная от пользователей.

Расширение интерактивных возможностей текущего приложения – решение неоднозначное, сложное по причинам, изложенным выше: такие возможности не должны будут разрушить выстроенную концепцию. Это значит, что анимации и персонажи не должны быть видоизменены. Кроме того, непросто даже конкретно определить, какого типа новшества можно внедрить, рассматривая текущее приложение как целостную мультимедийную единицу. Но, учитывая развитие современных технологий и разностороннюю направленность приложения, можно предположить, что новые интерактивные возможности будут предполагать взаимодействие не только с помощью нажатия, но и голосовые команды и айтрекинг, не только 2D отображение инсталляции, но и технологии дополненной реальности.

Идея расширения музыкальной библиотеки состоит не только в написании и добавлении новых авторских композиций, но и подключении музыкального плеера в традиционном понимании этого термина, то есть, позволение пользователю прослушивать внутри приложения свои музыкальные

подборки, хранящиеся на устройстве, либо же загрузка их из виртуального пространства. Однако здесь, несомненно, нужно будет соблюсти некий баланс, чтобы приложение целиком не превратилось в музыкальный плеер, и не сместить акценты аудиовизуальной инсталляции.
Интересным, хоть и технически сложным решением здесь является создание генератора музыки, который бы, возможно, алгоритмически создавал композиции, например, в соответствии с настроением пользователя.

Запуск приложения на других платформах – прежде всего IOS и web – решение, сложность которого определяется лишь сложностью освоения технологий программирования для выбранных платформ, так как весь графический контент не имеет особенностей встраивания и универсален для использования. Кроме того, все выполненные анимации – векторные, а это значит, что даже размеры диагоналей экранов устройств не имеют значения для грамотного отображения анимаций, поэтому «SILENTIO» можно реализовать даже как приложение для SMART TV. Однако для данного решения как раз имеет место переосмысление концепции приложения и его назначения, интерактивной механики, так как взаимодействие с устройствами на разных платформах может происходить в порядке, отличном от первоначально задуманного. Поэтому «SILENTIO» может выступать в роли, например, живых обоев или экрана ожидания.

Доработка или переработка механики, визуала, музыки после анализа заинтересованности пользователей – очевидный шаг. Сбор обратной связи – комплексный и долгий процесс. Как уже было обозначено выше, для увеличения количества скачиваний приложения необходимо продвижение, реклама, и эти процессы сами по себе уже могут поспорить по сложности с разработкой проекта. Конечно, приложение «SILENTIO» кроме всего прочего остается ориентированным на пользователя. И, разумеется, сделать приложение более удобным для пользователей могут помочь только сами пользователи.

Таким образом, несмотря на концептуальную целостность приложения «SILENTIO» остается доступным довольно большое количество различных вариантов дальнейшего развития, многие из которых возможны и не так сложны в реализации благодаря высокотехнологичным подходам к созданию контента и программного кода.

Заключение

В современном мире, где уровень стресса и тревожности постоянно растет, приложения для релаксации, конечно, набирают популярность. Поэтому созданная в рамках выпускной квалификационной работы аудиовизуальная инсталляция «SILENTIO» - потенциально востребованный продукт.

«SILENTIO» – это концептуально целостное мобильное приложение с игровым интерфейсом, разработка которого включала такие этапы, как написание и запись уникального музыкального произведения, создание векторного рисунка и его анимация, программирование на Java.

С технической стороны приложение отвечает всем обозначенным требованиям, оно работоспособно на различных устройствах, подразумевает некоторые пути дальнейшего развития и опубликовано на ресурсе Google Play, имеет малый размер файла.

Кроме того, «SILENTIO» в первую очередь остается уникальным художественным произведением, не имеющим визуальных аналогов, построенным по авторской концепции и прошедшим путь от идеи до публикации готового приложения.

При создании проекта получен опыт разработки мобильного приложения на Android, публикации приложения, решения нестандартных задач в рамках разработки.

Таким образом, получен утвердительный ответ на вопрос – может ли художник создать мобильное приложение – аудиовизуальную инсталляцию, может ли выпускник кафедры Информационных систем в области искусств и гуманитарных наук действовать в синтезе нескольких современных технологий – программирования, изобразительного искусства, дизайна, написания музыки в секвенсоре.

Список используемых источников

1. Airbnb Lottie [Электронный ресурс]. Режим доступа: https://airbnb.io/lottie, свободный. – Загл. с экрана. – Яз. рус., англ.

2. Анимация в мобильных приложениях: тестируем Lottie/Хабр [Электронный ресурс]. Режим доступа: https://habr.com/ru/post/451638/, свободный. – Загл. с экрана. – Яз. рус., англ.

3. LottieFiles - Free animation files built for Lottie [Электронный ресурс]. Режим доступа: https://lottiefiles.com/, свободный. – Загл. с экрана. – Яз. рус., англ.

4. Разработчик Александр Климов [Электронный ресурс]. Режим доступа: developer.alexanderklimov.ru/, свободный. – Загл. с экрана. – Яз. рус., англ.