ОТЧЕТ

о проектной работе

по теме: «Обучение Астрономии»

по дисциплине: Проектный практикум

Команда: ПУШКА

Екатеринбург

2021

**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 4](#_Toc74699819)

[1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 5](#_Toc74699820)

[1.1 Жанр и аудитория 5](#_Toc74699821)

[1.2 Анализ аналогичных проектов 5](#_Toc74699822)

[1.4 Обзор стека для разработки 8](#_Toc74699823)

[1.5 Календарный план проекта 9](#_Toc74699824)

[1.6 Определение проблемы 11](#_Toc74699825)

[1.7 Подходы к решению проблемы 12](#_Toc74699826)

[2. ПРОЕКТИРОВАНИЕ СИСТЕМЫ 12](#_Toc74699827)

[2.1 Общие сведения 12](#_Toc74699828)

[2.2 Основная концепция игры 12](#_Toc74699829)

[2.3 Цель игры 13](#_Toc74699830)

[2.4 Игровые возможности 13](#_Toc74699831)

[2.5 Варианты использования игрового приложения(use-case) 13](#_Toc74699832)

[2.6 Макет игры 14](#_Toc74699833)

[3. РАЗРАБОТКА ИГРЫ 16](#_Toc74699834)

[3.1 Требования к продукту MVP 16](#_Toc74699835)

[3.2 Техническая реализация 16](#_Toc74699836)

[3.3 Анимирование изображений 25](#_Toc74699837)

[ЗАКЛЮЧЕНИЕ 26](#_Toc74699838)

[СПИСОК ЛИТЕРАТУРЫ 27](#_Toc74699839)

[ПРИЛОЖЕНИЯ 28](#_Toc74699843)

[Приложение А 28](#_Toc74699844)

[Приложение Б 33](#_Toc74699845)

[Приложение В 34](#_Toc74699846)

[Приложение Г 35](#_Toc74699847)

[Приложение Д 36](#_Toc74699848)

[Приложение Е 37](#_Toc74699849)

[Приложение Ж 38](#_Toc74699850)

[Приложение К 39](#_Toc74699851)

**Введение**

**Актуальность**

Игра в жанре головоломка – игра, созданная с целью развлечения посредством трудностей, которые разрешаются при помощи находчивости или настойчивых попыток.

Головоломка - название жанра компьютерных игр, целью которых является решение логических задач, требующих от игрока задействования логики, стратегии и интуиции. Существует множество разновидностей головоломок, например: загадки, лабиринты, задачи на нестандартное мышление, пространственную логику, распознавание образов.

Больше всего головоломки распространены на игровых приставках, мобильных телефонах, аркадных автоматах.

Головоломки подразумевают понятность в правилах и управлении. Игрок должен понимать серию возможных действий в игровом процессе. Важна простота использования интерфейса. Так же необходимо поощрять навык игрока при успешно решенном задании.

Таким образом, актуальной является задача разработки игры в жанре головоломка.

**Цель и задачи**

Целью данной работы является разработка образовательной игры по теме «Астрономия» в жанре «Голов­оломка» на платформе U­nity для ОС Android.

Для достижения поставленной цели мы поставили перед собой ряд задач:

**1.**Анализ предметной области

**а)** Обзор существующих решений

**б)** Обзор стека для разработки

**2.** Проектирование игры

**а)** Продумывание игровых циклов, фан составляющей.

**б)** Продумывание стиля игры, подбор референсов

**в)** Создание кликабельного прототипа игры в Figma

**3.** Разработка:­

**а)** Реализация core-механики игры

**б)** Создание анимации созвездия первого и второго уровня

**в)** Внедрение визуальной составляющей

**г)** Реализация всплывающих окон (подсказки, обучение, пауза, похвала)

**д)** Реализация экрана настроек

**4.** Подведение итогов и составление отчета

1. **АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**
   1. **Жанр и аудитория**

Игра «Собери созвездия» относится к жанру головоломки.

Сегмент ЦА: midcore/casual.

Пол: м/ж.

Возраст: 9–12 лет.

Целевой рынок: Россия.

Поведение: Игра будет интересна тем, кто интересуется астрономией, любит решать головоломки и играть в игры, основанные на визуальной составляющей.

* 1. **Анализ аналогичных проектов**

**Прямые конкуренты:**

Игра *Constellation Builder* в жанре казуал. В ней игроку дается набор точек, соединенных линиями, и его задачей является расположить их в правильном порядке, получая созвездие. К плюсам можно отнести начисление очков за собранное созвездие. К минусам можно отнести отсутствие дополнительной информации по созвездию (нет теории, нет красочной картинки персонажа, в честь которого названо созвездие).

Изображение выглядит как текст, платформа, наружный объект, ночное небо

Автоматически созданное описание 

Рисунок 1 - Скриншоты из игры Constellation Builder

Игра *Starpaths*: в жанре головоломки. Игроку необходимо прочертить линию к каждой точке и получить рисунок созвездия. Если игрок чертит линию, которой не должно быть, она удаляется, иначе линия остается на сцене игры. Плюсами данной игры можно считать наличие картинки животного после отрисовки всех линий и наличие бонусов за прохождение уровней. Так же присутствует экран подсказки с рисунком созвездия. Недостатком данной игры является то, что все точки на экране выглядят одинаково и сложно найти те, которые требуется соединить, даже с использованием экрана подсказок. Кроме того, к минусам можно отнести большой блок текста с информацией о созвездии.

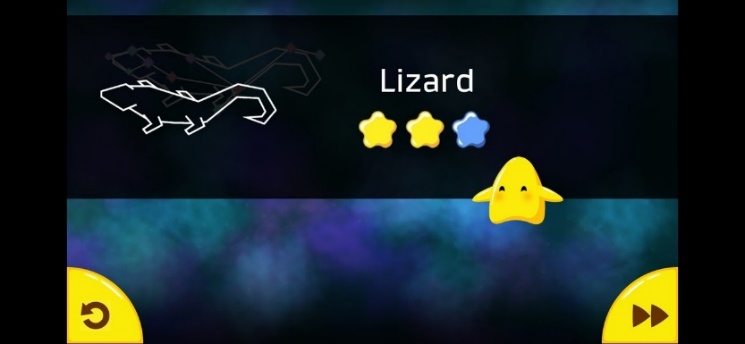
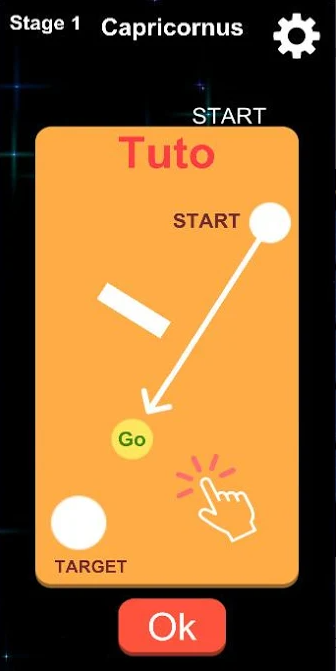
 

Рисунок 2 – игра *Starpaths*

Игра *Star Link: Constellation* в жанре казуал. Её суть заключается в том, что игроку необходимо чертить линию к каждой точке нажатием на экран, проходя при этом через препятствие, в виде движущейся платформы. Плюсами данной игры можно назвать наличие картинки животного после завершения отрисовки линий, и наличие теории по созвездию. К минусам данной игры можно отнести теорию, прописанную текстом. Так как большой блок информации, который требуется прочитать, портит впечатление об игре.

 Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3 - Скриншоты из игры *Star Link: Constellation*

**Косвенные конкуренты:**

*Звездный атлас 💫 Астрономия детям от Star Walk* в жанре Education. Энциклопедия для детей, в которой игрок изучает созвездия. На каждом уровне можно прослушать теорию по созвездию и посмотреть мультфильм о нем. К плюсам данной игры можно отнести интересное изложение фактов о созвездиях в виде озвучки и мультфильмов, так же красивый визуал. К минусам игры можно отнести недостаток интерактива, игрок только смотрит, слушает, ему не нужно дополнительно взаимодействовать с игрой.

Изображение выглядит как текст, монитор, электроника, компьютер

Автоматически созданное описаниеИзображение выглядит как текст, знак, векторная графика

Автоматически созданное описание

Рисунок 4 - *Звездный атлас💫 Астрономия детям от Star Walk*

* 1. **Референсы**

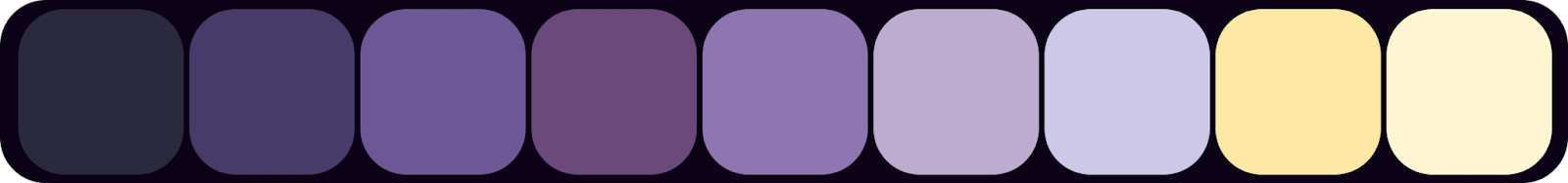


Рисунок 5 - Референс цветовой гаммы игры

**Изображение выглядит как текст

Автоматически созданное описание**

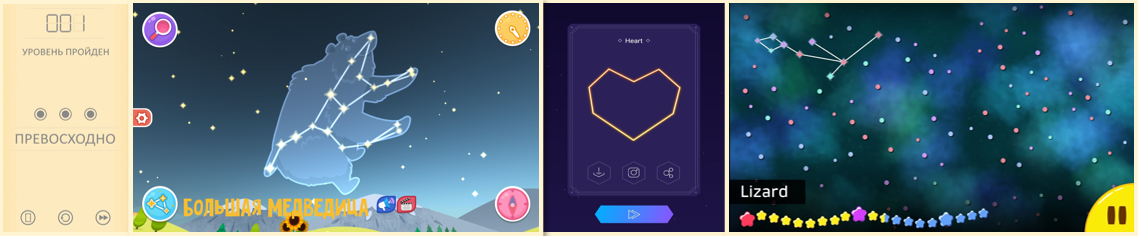
****

Рисунок 6 - Референсы стиля игры

Референс на геймплей и механику - Игра *Starpaths* (См. Рисунок 2)

* 1. **Обзор стека для разработки**

**Unity**

Это современная платформа разработки для создания игр и приложений. Данная среда разработки позволяет создавать приложения и игры не только для компьютеров, работающих на операционных системах: Microsoft Windows и Mac OS, но и для мобильных устройств на базе: iOS, Android, Windows Phone, игровых консолях: Sony PlayStation, Xbox, Wii, и других девайсов. В Unity используются языки программирования: C# и JavaScript. Огромной плюс данной платформы – это простота разработки приложений и игр для мобильных устройств, как от самых простых в разработке, так и до самых сложных.

Преимущества:

1. Много обучающего материала: статей, разборов, видеоуроков;
2. Огромное количество инструментов для разработки игр;
3. Присутствует внутренний магазин, где можно найти много полезного для создания своего проекта;
4. Большое количество библиотек, которые позволяют расширить функционал платформы;

Недостатки:

1) у разработчиков нет доступа к исходному коду платформы на системном уровне;

2) условно-бесплатный…

**Figma –** графический редактор, использующийся для разработки интерфейсов и прототипирования с возможностью совместной работы в режиме реального времени.

**Adobe Photoshop –** многофункциональный графический редактор для работы с фото.

**Dragon Bones -** бесплатный китайский аналог Spine, предоставляющий понятный интерфейс и возможности скелетной и mesh-анимации.

* 1. **Календарный план проекта**

«Таблица 1»

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **Название** | **Ответственный** | **Неделя** | | | | | | | | | | | | | | | | | | | | | | | | |
| **1** | **2** | | **3** | | **4** | | **5** | **6** | | **7** | **8** | **9** | **10** | **11** | **12** | | **13** | | **14** | | **15** | | **16** | |
| ***Анализ*** | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| *1.1* | *определение проблемы* | Серебрякова Е. Ю. |  |  | |  | |  | |  |  | |  |  |  |  |  |  | |  | |  | |  | |  | |
| *1.2* | *выявление целевой аудитории* | Серебрякова Е. Ю. |  |  | |  | |  | |  |  | |  |  |  |  |  |  | |  | |  | |  | |  | |
| *1.3* | *конкретизация проблемы* | Серебрякова Е. Ю. |  |  | |  | |  | |  |  | |  |  |  |  |  |  | |  | |  | |  | |  | |
| *1.4* | *анализ аналогов* | Серебрякова Е. Ю. |  |  | |  | |  | |  |  | |  |  |  |  |  |  | |  | |  | |  | |  | |
| *1.6* | *формулирование требований к MVP продукта* | Серебрякова Е. Ю. |  |  | |  | |  | |  |  | |  |  |  |  |  |  | |  | |  | |  | |  | |
| *1.7* | *определение платформы и стека для MVP* | Серебрякова Е. Ю. |  |  | |  | |  | |  |  | |  |  |  |  |  |  | |  | |  | |  | |  | |
| *1.8* | *формулировка цели* | Серебрякова Е. Ю. |  |  | |  | |  | |  |  | |  |  |  |  |  |  | |  | |  | |  | |  | |
| *1.9* | *определение задач* | Серебрякова Е. Ю. |  |  | |  | |  | |  |  | |  |  |  |  |  |  | |  | |  | |  | |  | |
| *1.10* | *Продумывание игровых циклов* | Серебрякова Е. Ю. |  |  | |  | |  | |  |  | |  |  |  |  |  |  | |  | |  | |  | |  | |
| ***Проектирование*** | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| *2.1* | *разработка сценариев использования системы* | Серебрякова Е. Ю. |  |  | |  | |  | |  |  | |  |  |  |  |  |  | |  | |  | |  | |  | |
| *2.2* | *прототипы интерфейсов* | Загуменова М.В. |  |  | |  | |  | |  |  | |  |  |  |  |  |  | |  | |  | |  | |  | |
| *2.3* | *формирование мудборта* | Шеметова А.С. |  |  | |  | |  | |  |  | |  |  |  |  |  |  | |  | |  | |  | |  | |
| *2.4* | *отрисовка и создание визуального оформления игры* | Шеметова А.С. |  |  | |  | |  | |  |  | |  |  |  |  |  |  | |  | |  | |  | |  | |
| *2.5* | *сбор теории для уровней* | Серебрякова Е. Ю. |  |  | |  | |  | |  |  | |  |  |  |  |  |  | |  | |  | |  | |  | |
| ***Разработка*** | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| *3.1* | *реализация простых атомарных механик, из которых состоит core-геймплей игры* | Романов Л. А. |  |  | |  | |  | |  |  | |  |  |  |  |  |  | |  | |  | |  | |  | |
| *3.1* | *создание прототипа игрового проекта (без визуала, на основе UI|UX прототипа интерфейса)* | Загуменова М.В. |  |  | |  | |  | |  |  | |  |  |  |  |  |  | |  | |  | |  | |  | |
| *3.2* | *тестирование приложения* | Серебрякова Е. Ю. и Романов Л. А. и  Соколов М. И. |  |  | |  | |  | |  |  | |  |  |  |  |  |  | |  | |  | |  | |  | |
| *3.3* | *интеграция игровой графики. Верстка и нарезка переданной Художником графики внутри Unity.* | Романов Л. А.+  Соколов М. И. |  |  | |  | |  | |  |  | |  |  |  |  |  |  | |  | |  | |  | |  | |
| *3.4* | *создание анимации* | Соколов М. И. и  Шеметова А.С. |  |  | |  | | |  |  | |  |  |  |  |  |  | |  | |  | |  | |  | |  |
| *3.5* | *интеграция анимации* | Романов Л. А.  Соколов М. И. |  |  | |  | | |  |  | |  |  |  |  |  |  | |  | |  | |  | |  | |  |
| *3.5* | *интеграция звуков при нажатии кнопок, фоновой музыки* | Романов Л. А. Соколов М. И. |  |  | |  | | |  |  | |  |  |  |  |  |  | |  | |  | |  | |  | |  |
| *3.6* | *озвучка теории* |  |  |  | |  | | |  |  | |  |  |  |  |  |  | |  | |  | |  | |  | |  |
| *3.7* | *внедрение озвученной теории* | Романов Л. А. Соколов М. И. |  |  | |  | | |  |  | |  |  |  |  |  |  | |  | |  | |  | |  | |  |
| ***Внедрение*** | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| *4.1* | *оформление MVP* | Романов Л. А. +  Соколов М. И. |  | |  | |  | |  |  | |  |  |  |  |  |  | |  | |  | |  | |  | |  |
| *4.2* | *написание отчета* | Серебрякова Е. Ю. |  | |  | |  | |  |  | |  |  |  |  |  |  | |  | |  | |  | |  | |  |
| *4.3* | *оформление презентации* | Загуменова М.В. и Серебрякова Е. Ю. |  | |  | |  | |  |  | |  |  |  |  |  |  | |  | |  | |  | |  | |  |
| *4.4* | *подготовка спикера к презентации* | Серебрякова Е. Ю. |  | |  | |  | |  |  | |  |  |  |  |  |  | |  | |  | |  | |  | |  |
|  | *защита проекта* |  |  | |  | |  | |  |  | |  |  |  |  |  |  | |  | |  | |  | |  | |  |

* 1. **Определение проблемы**

Большинство игр, направленных на изучение созвездий, выполняет роль справочника, показывая картинку созвездия и статью с фактами о нем, или являются картами звездного неба. Такие справочники могут не подойти пользователям, желающим закрепить полученные знания на практике и более активно взаимодействовать с приложением.

Для получения обратной связи мы провели опрос среди сверстников, у которых есть знакомы дети (братья/сестры) возраста 9–12 лет. Результаты опроса можно увидеть на Рисунке 7.



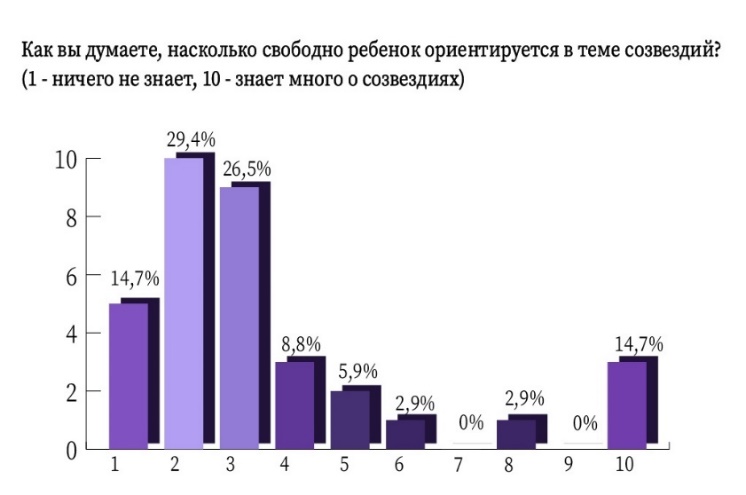


Рисунок 7 – Результаты опроса

По графикам видно, что почти половина опрошенных считает, что ребенок не знает, как выглядит рисунок созвездия.

**1.7 Подходы к решению проблемы**

После анализа конкурентов мы приняли решение создать игру с несколькими уровнями, в которой потребуется нарисовать созвездие самостоятельно, после чего оживет персонаж, в честь которого названо созвездие. Кроме того, игрок по желанию сможет прослушать факты о созвездии.

**2. Проектирование системы**

**2.1 Общие сведения**

Название данной игры - "Собери созвездия", где игрок изучает рисунок созвездий и тренирует память. Игрок должен запомнить расположение линий на картинке и самостоятельно провести их по памяти. Чтобы успешно пройти уровень нужно, соединив все точки, получить рисунок созвездия.

**2.2 Основная концепция игры**

Основная задача игрока – соединить точки линиями и получить рисунок созвездия.

Если игрок соединяет точки неверно, то они меняют цвет на красный.

Как только игрок получает правильный рисунок, появляется анимация животного, в честь которого названо созвездие.

**2.3 Цель игры**

Цель игры – получить рисунок созвездия.

**2.4 Игровые возможности**

Игрок может вести линию от одной звезды к любой другой. Соединять любые звезды. Удалять любую линию. Может ставить игру на паузу, проходить уровень сначала и выходить в главное меню.

**2.5 Варианты использования игрового приложения(use-case)**

На основе концепции игры, были разработана use-case диаграммы вариантов использования:

1. Игрового меню, где игрок может выбрать уровень и начать игру, зайти в настройки.
2. Игровой сцены.

Варианты использования представлены на рисунках 8, 9.

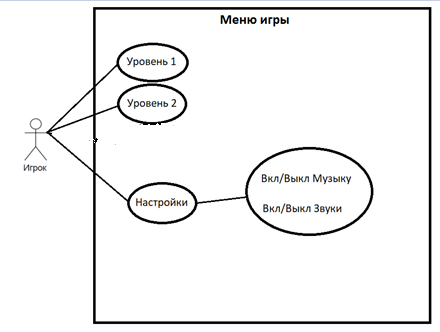


Рисунок 8. Диаграмма вариантов использования меню

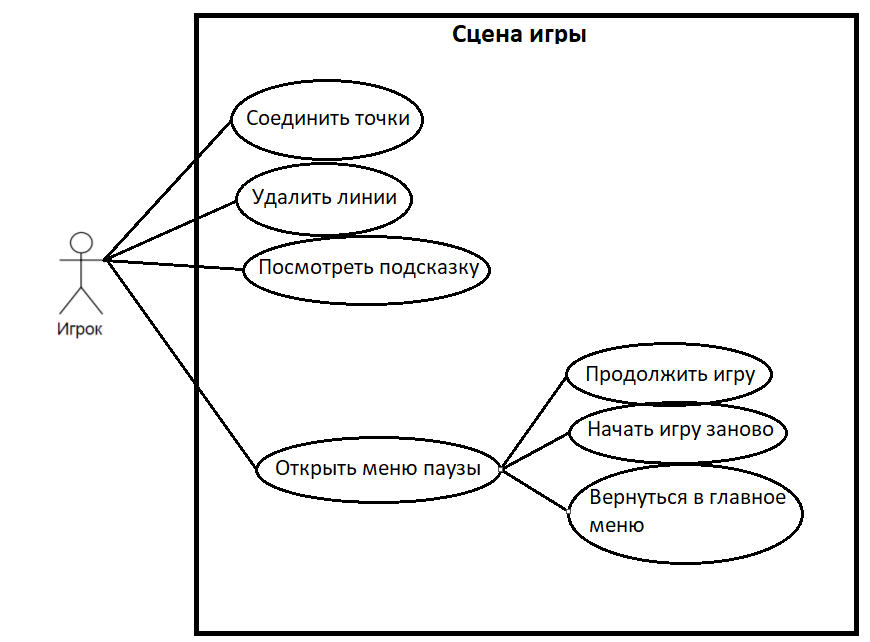


Рисунок 9 - Диаграмма вариантов использования в игровой сцене

Каждый уровень – это одна итерация игрового цикла. В которой игрок соединяет звезды, видит ожившего персонажа и получает похвалу за успешно пройденный уровень. Схему цикла можно увидеть на Рисунке 10.

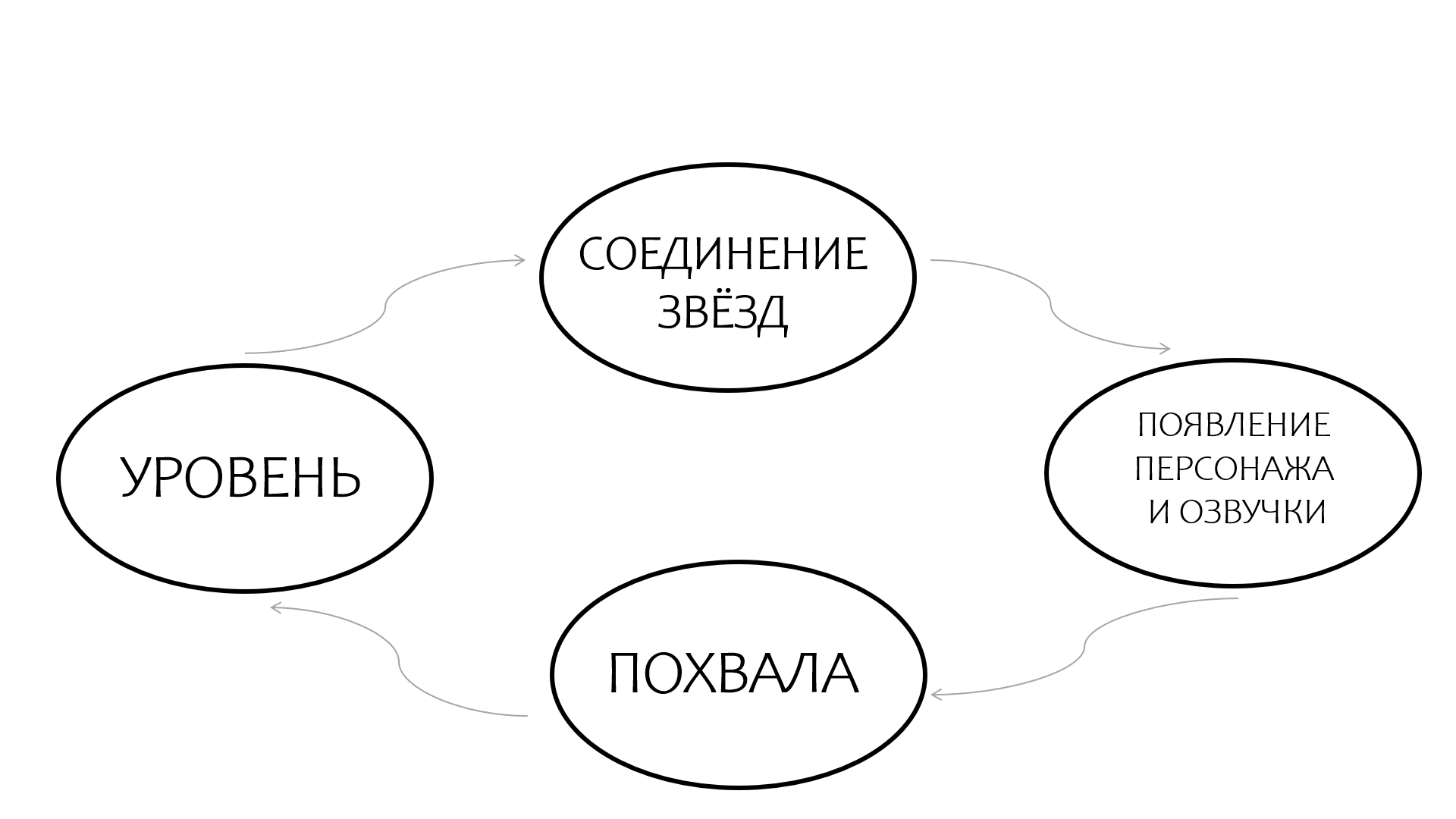


Рисунок 10 – Игровой цикл

**2.6 Макет игры**

Дизайн меню состоит из частей:

1. Кнопки начала уровня;
2. Фон игры;
3. Кнопка настроек.

При нажатии кнопки уровня начинается прохождение уровня.

При нажатии кнопки настроек игрок может отключить музыку, а также звуки игры (нажатия кнопок, соединения линий). Схема расположения элементов интерфейса меню изображена на рис. 11, 12.

Изображение выглядит как текст, черный

Автоматически созданное описание

Рисунок 11 - Интерфейс меню

****

Рисунок 12 - Интерфейс меню

Дизайн игрового процесса состоит из частей:

1. Поле с точками, которые можно соединять
2. Кнопка паузы
3. Кнопка подсказки

При нажатии кнопки паузы игроку предоставляется возможность: продолжить игру, начать игру заново и выйти в главное меню.

Экраны прототипа игры из Figma представлены в Приложении А.

**3. РАЗРАБОТКА ИГРЫ**

**3.1 Требования к продукту MVP**

**Функциональные:**

- Переход из главного меню в сцену игры;

- Переход из главного меню в сцену настроек;

- Возможность соединять точки на игровом поле;

- Возможность посмотреть подсказку;

- Возможность видеть, когда отрисована неправильная линия;

- Возможность удалять линии на игровом поле;

- Возможность убрать звук и включить его обратно;

- Рабочая кнопка для прохождения первого уровня;

- Появление экрана похвалы с рабочими кнопками;

- Нажатие кнопки перезапуска уровня;

- Анимация животных;

- Должно быть реализовано два уровня;

**Нефункциональные:**

- Игра должна быть реализована под платформу Android;

- Игра должна работать только в режиме альбомной ориентации экрана;

- Игра должна быть разработана на движке Unity, язык программирования C#

**3.2 Техническая реализация**

Игровое приложение состоит из 6 сцен: Меню (Menu), Уровень 1 с механикой (Unicorn), Уровень 1 пройденный (CompletedUnicorn), Настройки (Settings), Уровень 2 с механикой (Dragon), уровень 2 пройденный (CompletedDragon).

**Реализация Core-механики**

**Изображение выглядит как текст

Автоматически созданное описание**

Рисунок 13 - Иерархия сцены первого уровня

Игровые объекты на данной сцене:

Main Camera - камера.

EventSystem - объект, позволяющий работать с рейкастами и коллайдерами объектов.

*Canvas и вложенные объекты* - отвечают за фон и отображение кнопок на сцене.

Star(n) - объекты, являющиеся звездами с прикрепленным скриптом CoreMech.

*ScriptHolder* - отвечает за проверку корректности соединения звезд, содержит флаг окончания уровня (соединения).

Horse - содержит контролер анимации лошади и скрипт запуска анимации при окончании уровня.

PostLevelMenu - отвечает за показ сообщения, оповещающего об успешном окончании уровня.

InstructionMenu - отвечает за отображение обучения при первом открытии уровня.

SceneLoader - отвечает за переход между сценами.

Pause - отвечает за возможность поставить паузу во время прохождения уровня.

ShowHint - отвечает за возможность посмотреть подсказку (корректно соединенные звезды) во время прохождения уровня.

**Создание звезд**

**Изображение выглядит как текст, снимок экрана, черный, несколько

Автоматически созданное описание**

Рисунок 14 – компоненты одной звезды

Каждая звезда создана по префабу, имеет такие компоненты как: Box Collider 2D (для обработки нажатий по ней), Sprite Renderer (для отображения спрайта), Core Mech (скрипт с механикой соединения) (Рисунок 14).

Рисунок расположения звезд на сцене можно увидеть в Приложения А (Рисунок А.1).

**Core-механика Скрипт (CoreMech)**

Если у звезды отсутствует компонент LineRender (линия), то он создается со стартовыми настройками. В листинге 1 приведен фрагмент кода, с создание компонента LineRenderer со стартовыми настройками.

Листинг 1. CoreMech

void Update()

{

if (!gameObject.TryGetComponent(out LineRenderer x))

{

LineRenderer line = gameObject.AddComponent<LineRenderer>();

line.material = mat;

line.startColor = new Color(1f, 0.945f, 0.463f, 1);

line.endColor = new Color(1f, 0.945f, 0.463f, 1);

line.startWidth = 0.15f;

line.endWidth = 0.15f;

line.positionCount = 2;

}

Если пользователь нажал на звезду и начал вести пальцем по экрану (и линии от этой звезды еще не существует), то включается интерфейс OnDrag, который отрисовывает линию с началом в центре звезды и концом в текущем положении курсора/пальца пользователя (Фрагмент скрипта в листинге 2).

Листинг 2. CoreMech

public void OnDrag(PointerEventData data)

{

if (!LineDrawn)

{

LineRenderer line = GetComponent<LineRenderer>();

line.SetPosition(0, this.gameObject.transform.position);

line.SetPosition(1, Camera.main.ScreenToWorldPoint(new Vector3(data.position.x, data.position.y, 1)));

if (data.hovered.Count == 1)

{

line.SetPosition(0, this.gameObject.transform.position);

line.SetPosition(1, data.hovered[0].transform.position);

}

}

}

Когда пользователь отрывает палец от экрана, осуществляется проверка на корректность (Приложение Б) данной линии (из одной звезды в другую и соответствуя заданному созвездию). Если линия проведена между двумя звездами, то спрайт начальной звезды меняется с белого на желтый. Если конечная точка линии не лежит на другой звезде, то линия удаляется (проверка с помощью data.hovered.Count). Если линия проведена между двумя звездами, но не является корректной, то она окрашивается в красный цвет. Если линия проведена между двумя звездами и корректна, то она остается желтой. В листинге 3 фрагмент кода, осуществляющий вышеперечисленные функции.

Листинг 3. CoreMech

public void OnEndDrag(PointerEventData data)

{

if (data.hovered.Count == 1 & !LineDrawn)

{

LineRenderer line = GetComponent<LineRenderer>();

var pos1 = this.gameObject.transform.position;

var pos2 = data.hovered[0].transform.position;

line.SetPosition(0, pos1);

line.SetPosition(1, pos2);

LineDrawn = true;

spriteRenderer.sprite = yellowStar;

if (!IsLineCorrect(pos1, pos2))

{

line.startColor = Color.red;

line.endColor = Color.red;

}

else

CorrectlyConnected = true;

}

else if (!LineDrawn)

{

LineRenderer line = GetComponent<LineRenderer>();

Destroy(line);

LineDrawn = false;

spriteRenderer.sprite = whiteStar;

}

}

Проверка соответствия линии данному созвездию проверяется посредством сравнения координат конечной точки с установленными при фиксированном значении начальной точки.

Удаление линии между двумя звездами осуществляется посредством нажатия (недолгого удержания) в окружности радиусом 0.3 и центром в середине данной линии. При удалении линии звезда, из которой была проведена эта линия, меняет спрайт обратно на белую звезду (Листинг 4).

Листинг 4.

LineRenderer line = GetComponent<LineRenderer>();

var pos1 = line.GetPosition(0);

var pos2 = line.GetPosition(1);

var pointerPos = Camera.main.ScreenToWorldPoint(Input.mousePosition);

var delta = 0.3;

if ((Math.Abs(pointerPos.x - (pos1.x + pos2.x) / 2) < delta) &

(Math.Abs(pointerPos.y - (pos1.y + pos2.y) / 2) < delta) &

Input.GetMouseButton(0)) //Input.touchCount > 0 for mobiles

{

Time++;

if (Time > 200)

{

Destroy(line);

LineDrawn = false;

CorrectlyConnected = false;

spriteRenderer.sprite = whiteStar;

}

}

if (Input.GetMouseButtonUp(0))

{

Time = 0;

}

Если все звезды соединены (корректно) (проверка с помощью скрипта CompleteScene см. далее), то все звезды с линиями удаляются, уровень пройден (Листинг 5). Рисунки соединения звезд можно увидеть в Приложении А (Рисунок А.2, А.7).

Листинг 5.

if (complete.Completed)

{

Destroy(this.gameObject);

LevelCompleted = true;

}

**Проверка окончания уровня (cкрипт CompleteScene)**

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 15 – Скрипт, у объекта ScriptHolder

Содержится на объекте ScriptHolder и имеет доступ к звездам на данной сцене (Рисунок 15).

С помощью флагов на каждой звезде проверяет корректность соединения каждой, если все в порядке, то дальше идет проверка: проходил ли игрок первый уровень до этого? Если нет, то уровень пройден, сохраняется информация о первом прохождении. Если игрок уже проходил первый уровень, то его переносит на сцену CompletedUnicorn. Данное разветвление создано для того, чтобы поменять иконку первого уровня после прохождения уровня.

Код скрипта CompleteScene приведен в Приложение В.

**Появление анимации (скрипт PlayAnimation)**

Данный скрипт обращается к скрипту CompleteScene, проверяя завершенность уровня.

При окончании уровня появляется анимация лошади, проигрывается звук завершения уровня, появляются кнопки далее и озвучки теории, картинка с надписью “Созвездие Единорога”. Код скрипта PlayAnimation приведен в Приложении Г. Рисунки экрана можно увидеть в Приложении А (Рисунок А.18, А.14).

**Экран похвалы после прохождения уровня (скрипт PostLevelMenu)**

После появления анимации кнопка “далее” приводит к появлению EndLevelMessage (Листинг 7), с похвалой за прохождение уровня и тремя кнопками: Retry - начинает данный уровень сначала, Menu - переход в главное меню, Next - переход на следующий уровень. Код скрипта PostLevelMenu приведен в листинге 6. Рисунок экрана можно увидеть в Приложении А (Рисунок А.10).

Листинг 6.

public class PostLevelMenu : MonoBehaviour

{

Button tomenuButton;

private void Start()

{

tomenuButton = GameObject.Find("Next").GetComponent<Button>();

tomenuButton.onClick.AddListener(() =>

{

EndLevelMessage.ShowMessage();

});

}

}

Листинг 7.

public class EndLevelMessage : MonoBehaviour

{

private static EndLevelMessage instance;

public GameObject Template;

void Awake()

{

instance = this;

}

public static void ShowMessage()

{

GameObject messageBox = Instantiate(instance.Template);

Transform image = messageBox.transform.Find("Image");

Button retry = image.Find("Retry").GetComponent<Button>();

Button menu = image.Find("Menu").GetComponent<Button>();

Button next = image.Find("Next").GetComponent<Button>();

retry.onClick.AddListener(() =>

{

SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);

});

menu.onClick.AddListener(() =>

{

SceneManager.LoadScene(0);

});

if ((SceneManager.GetActiveScene().buildIndex == 4))

next.interactable = false;

next.onClick.AddListener(() =>

{

SceneManager.LoadScene(3);

});

}

}

**Обучение (скрипт InstructionMessages)**

На первом уровне проводится проверка: первый ли раз игрок зашел на первый уровень с помощью:

Листинг 8.

If (Player Prefs.HasKey(“IsFirstLaunch”))

IsFirstLaunch = false;

Значение IsFirstLaunch сохраняется после того, как игрок зашел на первый уровень в скрипте CoreMech:

Листинг 9.

PlayerPrefs.SetInt(“IsFirstLaunch”, 1)’

PlayerPrefs.Save();

Таким образом обучение показывается только один единственный раз в момент первого попадания игрока на первый уровень.

Само обучение представляет собой 3 возникающих друг за другом префаба.

Код скрипта InstructionMessages приведен в Приложении Д.

Изображения экранов можно увидеть в Приложении А (Рисунок А.3, А.4, А.5).

**Пауза (скрипт Pause)**

Скрипт связан с кнопкой паузы на сцене, при завершении уровня кнопка удаляется.

При нажатии на кнопку показывается префаб окна паузы с тремя кнопками: Resume - удаляет окно паузы, Retry - перезагружает данный уровень, Menu - переносит игрока в главное меню. Код скрипта приведен в Приложении Е. Изображение экрана можно увидеть в Приложении А (Рисунок А.11).

**Подсказка (скрипт ShowHint)**

Показывает префаб подсказки в зависимости от текущего уровня.

Если номер сцены 2 или 3 (Unicorn или CompletedUnicorn) - то подсказка с рисунком созвездия Единорог.

Если номер сцены 4 или 5 (Dragon или CompletedDragon) - то подсказка с рисунком созвездия Дракон.

Подсказка представляет собой картинку с корректным соединением звезд для данного уровня и кнопкой, удаляющей данную картинку с экрана пользователя, после чего пользователь продолжает игру. Код скрипта ShowHint приведен в Приложении Ж. Рисунки можно увидеть в Приложении А (Рисунок А.9, А.12)

**Изменение спрайта кнопки в главном меню после прохождения уровня (скрипт MainButtons)**

До того, как пройден уровень с единорогом, кнопка дракон не активна и имеет изображение вопросительного знака.

Далее проверка на прохождения первого уровня с помощью FirstLvlCompleted, если он пройден, то спрайт кнопки единорога меняется с не закрашенного на закрашенный, теперь эта кнопка ведет в сцену CompletedUnicorn (сразу показывается анимация без соединения звезд), кнопка уровня дракон становится активной и ее спрайт меняет на спрайт не закрашенного дракона. Изменение можно увидеть на Рисунке 16.

После прохождения второго уровня появляется значение SecondLvlCompleted, спрайт кнопки уровня дракона меняется с не закрашенного дракона на закрашенного дракона, кнопка ведет в сцену CompletedDragon (сразу показывается анимация без соединения звезд).

Код скрипта MainButtons приведен в Приложении К.

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 16 – Изменение спрайта кнопки после прохождения уровня

**Сцены завершенных уровней**

**Изображение выглядит как текст, монитор, электроника

Автоматически созданное описание**

Рисунок 17 – Сцена завершенного уровня

Сцены завершенных уровней не имеют звезд (Рисунок 17), попадая на них игрок сразу видит анимацию, появляется кнопка Retry, которая ведет на сцену с соединением звезд для данного созвездия (на сцену Unicorn из CompletedUnicorn и на сцену Dragon из CompletedDragon), чтобы игрок мог повторно пройти первый уровень.

**Все скрипты для второго уровня выполнены аналогично.**

**3.3 Анимирование изображений**

Существует несколько вариантов анимирования в 2D играх: покадровая, шейповая и костная анимация. Для нашей игры мы выбрали костную анимацию. Так как это является оптимальным и самым производительным вариантом в индустрии 2D игр.

**Анимация Единорога**

После того, как был нарисован арт. Картинку, выбранную для анимации, разделили на спрайты. При работе в программе размер кусочков подгонялся до нужных размеров с помощью полигонов.

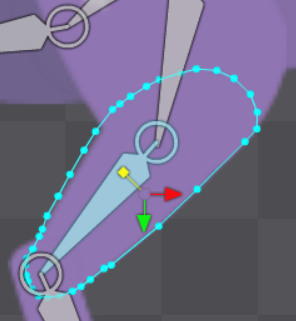


Рисунок 18 – выделенный полигон

Далее, изображение из полигонов собирается обратно в одно. Мы выбрали костную анимацию, поэтому к полигонам цепляются кости (Рисунок 19). Кости – это главное средство передвижения картинок в анимации. При передвижении кости передвигается изображение, прикрепленное к ней. Прикрепив изображения к костям, начинаем их передвигать в соответствии с задуманным движением, в нашем случае бег. При каждом передвижении изображения фиксируем изменение. Т. е. создаем дополнительный кадр. Анимация состоит из большого количества отдельных кадров.



Рисунок 19 – изображение, полностью покрытое костями

После создания всех кадров. Анимация была сохранена в 48 FPS и перенесена в проект в Unity, где на экран выводятся все изображения, тем самым создается движение картинки. Так как в программе не оказалось ползунка, необходимого для регулирования скорости анимации, была создана переменная float, с помощью которой удалось изменять скорость анимации.

Чтобы изображение при проигрывании в Unity не стояло неподвижно, в течении одного кадра, был удален последний кадр анимации.

**Анимация Дракона**

Из-за того, что конечности и сгибы суставов широкие, между конечностями при повороте может образоваться зазор, который приходилось устранять смещением одной из конечностей.

**В остальном этапы анимации аналогичны.**

**Заключение**

В ходе проектной деятельности первого курса, решены следующие задачи:

1. Проведен анализ аудитории.
2. Проведен анализ игр-конкурентов.
3. Продуман USP игры и фан составляющая.
4. Подобраны референсы геймплея, механик и стиля.
5. Создан кликабельный прототип в Figma [12].
6. Разработана анимация персонажей первого и второго уровня в программе DragonBones.
7. Реализована Core-механика игры.
8. Реализованы всплывающие окна (подсказки, похвала).
9. Внедрена визуальная составляющая (спрайты, соответствующие прототипу в Figma).
10. Реализована сцена настроек с возможностью отключения звуков и музыки.

С результатом нашей работы можно ознакомиться на соответствующем электронном ресурсе [13].

**СПИСОК ЛИТЕРАТУРЫ**

1. Терехин, В. А. Особенности детей 9 -12 лет (физиологические, психологические, социальные)[Электронный ресурс] / В. А. Терехин – Режим доступа: <https://starogorodkovskay.odinedu.ru/school_life/Psiholog/%D0%A0%D0%BE%D0%B4%D0%B8%D1%82%D0%B5%D0%BB%D1%8F%D0%BC/%D0%9E%D1%81%D0%BE%D0%B1%D0%B5%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D0%B8%20%20%D0%B4%D0%B5%D1%82%D0%B5%D0%B9%209-12%20%D0%BB%D0%B5%D1%82.pdf>. , свободный. Дата обращения: 7.04.2021

# Верещагин А. Особенности создания интерфейса для мобильной игры

[Электронный ресурс] / А. Верещагин – Режим доступа: <https://dtf.ru/gamedev/44813-osobennosti-sozdaniya-interfeysa-dlya-mobilnoy-igry> , свободный. Дата обращения: 9.04.2021

# Лазебный Е. Советы по разработке пользовательского интерфейса для игр [Электронный ресурс] / Е. Лазебный – Режим доступа: <https://app2top.ru/interfaces/sovety-po-razrabotke-pol-zovatel-skogo-interfejsa-dlya-igr-182657.html> , свободный. Дата обращения: 11.04.2021.

# Хозлова Д. Введение в геймдизайн: Разработка и дизайн головоломок в видеоиграх [Электронный ресурс] / Д. Хохлова – Режим доступа: <https://vc.ru/flood/17028-gamedesign-challenges-3> , свободный. Дата обращения: 21.03.2021.

1. Официальный сайт Unity [Электронный ресурс] - Режим доступа: <https://unity.com/ru> . Дата обращения: 28.04.2021.
2. Официальный сайт GooglePlay. [Электронный ресурс] - Режим доступа: <https://play.google.com/store/apps/details?id=com.pixelsunsetstudio.ConstellationPuzzle> , свободный. Дата обращения: 29.04.2021.
3. Официальный сайт GooglePlay. [Электронный ресурс] - Режим доступа: <https://play.google.com/store/apps/details?id=com.bigbadbros.starpaths> , свободный. Дата обращения: 29.04.2021.
4. Официальный сайт GooglePlay. [Электронный ресурс] - Режим доступа: <https://play.google.com/store/apps/details?id=com.whitewind.star> , свободный. Дата обращения: 29.04.2021.
5. Официальный сайт GooglePlay. [Электронный ресурс] - Режим доступа: <https://play.google.com/store/apps/details?id=com.vitotechnology.StarWalkKids> , свободный. Дата обращения: 29.04.2021.
6. Единорог (созвездие). [Электронный ресурс] – Режим доступа: <https://www.galactic.name/constellations/monoceros_constellation.php> , свободный. Дата обращения: 30.04.2021.
7. Unity Documentation. [Электронный ресурс] – Режим доступа: https://docs.unity3d.com/Manual/index.html, свободный. Дата обращения: 01.05.2021.
8. Прототип в Figma [Электронный ресурс] – Режим доступа: https://www.figma.com/file/sXaZt6uUAbOfAeD4HDW89n/CosmoTour?node-id=0%3A1, свободный.
9. Проект на GitHub [Электронный ресурс] – Режим доступа: https://github.com/levonte4/astronomy, свободный.
10. Морозов О. Spine – первые шаги [Электронный ресурс] / О. Морозов – Режим доступа: https://habr.com/ru/post/207904/, свободный.

**Приложения**

# **Приложение А**

Прототипы экранов, разработанных в Figma.



Рисунок A.1 – начало уровня

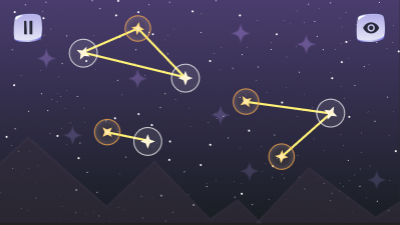
 

Рисунок A.2 – соединение звезд

Изображение выглядит как текст, часы

Автоматически созданное описание

Рисунок А.3 – первое окно обучения

Изображение выглядит как текст

Автоматически созданное описание

Рисунок А.4 – второе окно обучения

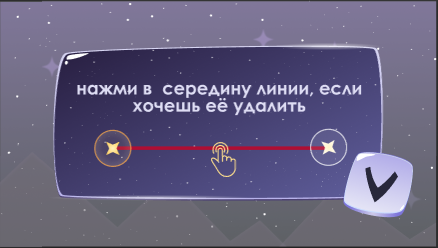


Рисунок А.5 – третье окно обучения

Изображение выглядит как текст, природа, наружный объект, ночное небо

Автоматически созданное описание

Рисунок А.6 – удаление линии

Изображение выглядит как текст, природа, ночное небо

Автоматически созданное описание

Рисунок А.7 – полностью нарисованное созвездие

Изображение выглядит как текст, ночное небо

Автоматически созданное описание

Рисунок А.8 – экран с анимацией Единорога

Изображение выглядит как текст, знак, закрыть

Автоматически созданное описание

Рисунок А.9 – экран подсказки первого уровня

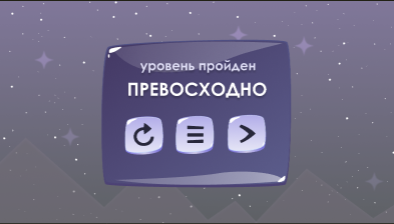


Рисунок А.10 – экран похвалы

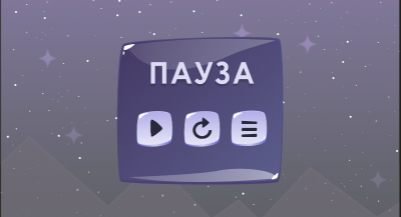


Рисунок А.11 – экран паузы

Изображение выглядит как текст, знак, ночное небо

Автоматически созданное описание

Изображение выглядит как текст, природа, ночное небо

Автоматически созданное описаниеРисунок А.12 – экран подсказки второго уровня

Рисунок А.13 – соединение точек второго уровня

Изображение выглядит как текст

Автоматически созданное описание

Рисунок А.14 – экран с анимацией Дракона

# **Приложение Б**

Фрагмент скрипта CoreMech. Фрагмент отвечает за проверку линии на корректность.

public bool IsLineCorrect(Vector3 pos1, Vector3 pos2)

{

if ((pos1 == new Vector3(7f, 0f, 1f)) & (pos2 == new Vector3(4.5f, -2.5f, 1f) | pos2 == new Vector3(2.5f, 0.5f, 1f)))

return true;

else if ((pos1 == new Vector3(4.5f, -2.5f, 1f)) & (pos2 == new Vector3(7f, 0f, 1f)))

return true;

else if ((pos1 == new Vector3(2.5f, 0.5f, 1f)) & (pos2 == new Vector3(-1.5f, 2f, 1f) | pos2 == new Vector3(-3.5f, -2f, 1f) | pos2 == new Vector3(7f, 0f, 1f)))

return true;

else if ((pos1 == new Vector3(-1.5f, 2f, 1f)) & (pos2 == new Vector3(2.5f, 0.5f, 1f) | pos2 == new Vector3(-4f, 4f, 1f) | pos2 == new Vector3(-6.5f, 3f, 1f)))

return true;

else if ((pos1 == new Vector3(-4f, 4f, 1f)) & (pos2 == new Vector3(-1.5f, 2f, 1f) | pos2 == new Vector3(-6.5f, 3f, 1f)))

return true;

else if ((pos1 == new Vector3(-6.5f, 3f, 1f)) & (pos2 == new Vector3(-4f, 4f, 1f) | pos2 == new Vector3(-1.5f, 2f, 1f)))

return true;

else if ((pos1 == new Vector3(-5.5f, -1.5f, 1f)) & (pos2 == new Vector3(-3.5f, -2f, 1f)))

return true;

else if ((pos1 == new Vector3(-3.5f, -2f, 1f)) & (pos2 == new Vector3(-5.5f, -1.5f, 1f) | pos2 == new Vector3(2.5f, 0.5f, 1f)))

return true;

return false;

}

# **Приложение В**

Скрипт CompleteScene. Отвечает за проверку корректного соединения каждой звезды.

public class CompleteScene : MonoBehaviour

{

public bool Flag = false;

public bool Completed = false;

public CoreMech star;

public CoreMech star1;

public CoreMech star2;

public CoreMech star3;

public CoreMech star4;

public CoreMech star5;

public CoreMech star6;

public CoreMech star7;

void Update()

{

if (star.CorrectlyConnected & star1.CorrectlyConnected & star2.CorrectlyConnected & star3.CorrectlyConnected &

star4.CorrectlyConnected & star5.CorrectlyConnected & star6.CorrectlyConnected & star7.CorrectlyConnected & !Flag)

{

Flag = true;

if (!PlayerPrefs.HasKey("FirstLvlCompleted"))

{

Completed = true;

PlayerPrefs.SetInt("FirstLvlCompleted", 1);

PlayerPrefs.Save();

}

else

SceneManager.LoadScene(3);

}

}

}

# **Приложение Г**

Скрипт PlayAnimation. Отвечает за появление анимации.

public class PlayAnimation : MonoBehaviour

{

public Animator animator;

public CompleteScene completeScene;

public Canvas canvas;

public bool Flag = false;

public Button Volume;

public Button Open;

public Button Next;

public Image Text;

public AudioSource sound;

void Start()

{

sound = GetComponent<AudioSource>();

}

void Update()

{

if (completeScene.Completed & !Flag)

{

Flag = true;

animator.SetBool("LevelCompleted", true);

Volume.transform.localPosition = new Vector3(-950, -400, 1);

Next.transform.localPosition = new Vector3(950, -400, 1);

Text.transform.localPosition = new Vector3(589.83f, 386.71f, 1);

sound.Play();

}

}

}

# **Приложение Д**

Скрипт InstructionMessages. Отвечает за появление обучения.

public class InstructionMessages : MonoBehaviour

{

public bool Flag = false;

public SceneLoad sceneLoad;

public GameObject Instruction1;

public GameObject Instruction2;

public GameObject Instruction3;

void Update()

{

if (!Flag & sceneLoad.IsFirstLaunch)

{

Flag = true;

ShowInstructions();

}

}

public void ShowInstructions()

{

GameObject instruction1 = Instantiate(Instruction1);

Transform image1 = instruction1.transform.Find("Image");

Button next1 = image1.Find("Next").GetComponent<Button>();

next1.onClick.AddListener(() =>

{

GameObject instruction2 = Instantiate(Instruction2);

Destroy(instruction1);

Transform image2 = instruction2.transform.Find("Image");

Button next2 = image2.Find("Next").GetComponent<Button>();

next2.onClick.AddListener(() =>

{

GameObject instruction3 = Instantiate(Instruction3);

Destroy(instruction2);

Transform image3 = instruction3.transform.Find("Image");

Button next3 = image3.Find("Next").GetComponent<Button>();

next3.onClick.AddListener(() =>

{

Destroy(instruction3);

});

});

});

}

}

# **Приложение Е**

Скрипт Pause. Отвечает за появление экрана паузы.

public class Pause : MonoBehaviour

{

public GameObject Template;

public Button PauseButton;

public bool Flag = false;

public CompleteScene Script;

void Start()

{

PauseButton.onClick.AddListener(() =>

{

ShowPauseMenu();

});

}

void Update()

{

if (Script.Completed & !Flag)

{

Flag = true;

Destroy(PauseButton.transform.gameObject);

}

}

public void ShowPauseMenu()

{

GameObject pauseMenu = Instantiate(Template);

Transform image = pauseMenu.transform.Find("Image");

Button resume = image.Find("Resume").GetComponent<Button>();

Button retry = image.Find("Retry").GetComponent<Button>();

Button menu = image.Find("Menu").GetComponent<Button>();

resume.onClick.AddListener(() =>

{

Destroy(pauseMenu);

});

retry.onClick.AddListener(() =>

{

SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);

});

menu.onClick.AddListener(() =>

{

SceneManager.LoadScene(0);

});

}

}

# **Приложение Ж**

Скрипт ShowHint. Отвечает за появление подсказки.

public class ShowHint : MonoBehaviour

{

public GameObject UnicornHint;

public GameObject DragonHint;

public Button Open;

void Start()

{

Open.onClick.AddListener(() =>

{

Show();

});

}

public void Show()

{

if (SceneManager.GetActiveScene().buildIndex == 2 | SceneManager.GetActiveScene().buildIndex == 3)

{

GameObject unicornHint = Instantiate(UnicornHint);

Transform image = unicornHint.transform.Find("Image");

Button next = image.Find("Next").GetComponent<Button>();

next.onClick.AddListener(() =>

{

Destroy(unicornHint);

});

}

else if (SceneManager.GetActiveScene().buildIndex == 4 | SceneManager.GetActiveScene().buildIndex == 5)

{

GameObject dragonHint = Instantiate(DragonHint);

Transform image = dragonHint.transform.Find("Image");

Button next = image.Find("Next").GetComponent<Button>();

next.onClick.AddListener(() =>

{

Destroy(dragonHint);

});

}

}

}

# **Приложение К**

Скрипт MainButtons. Отвечает за смену спрайта кнопки уровня в главном меню, после первого прохождения уровня.

public class MainButtons : MonoBehaviour

{

public GameObject ScriptHolder;

public CompleteScene Script;

public bool Completed;

public int Flag = 0;

public Sprite FinalUnicorn;

public Sprite Dragon;

public Sprite FinalDragon;

void Start()

{

var canvas = GameObject.Find("Canvas");

Button dragonButton = GameObject.Find("DragonButton").GetComponent<Button>();

dragonButton.interactable = false;

Button lastButton = GameObject.Find("LastButton").GetComponent<Button>();

lastButton.interactable = false;

}

void Update()

{

if (Flag == 0 & PlayerPrefs.HasKey("FirstLvlCompleted"))

{

Flag = 1;

var canvas = GameObject.Find("Canvas");

Button unicornButton = GameObject.Find("UnicornButton").GetComponent<Button>();

unicornButton.image.sprite = FinalUnicorn;

unicornButton.onClick.RemoveAllListeners();

unicornButton.onClick.AddListener(() =>

{

SceneManager.LoadScene(3);

});

Button dragonButton = GameObject.Find("DragonButton").GetComponent<Button>();

dragonButton.interactable = true;

dragonButton.image.sprite = Dragon;

Destroy(ScriptHolder);

}

if (Flag == 1 & PlayerPrefs.HasKey("SecondLvlCompleted"))

{

Flag = 2;

var canvas = GameObject.Find("Canvas");

Button dragonButton = GameObject.Find("DragonButton").GetComponent<Button>();

dragonButton.image.sprite = FinalDragon;

dragonButton.onClick.RemoveAllListeners();

dragonButton.onClick.AddListener(() =>

{

SceneManager.LoadScene(5);

});

Destroy(ScriptHolder);

}

}

}