Департамент образования и науки Курганской области

Государственное бюджетное профессиональное

образовательное учреждение

«Курганский государственный колледж»

Специальность 09.02.07

Информационные системы и программирование

КУРСОВОЙ ПРОЕКТ

по МДК 02.01. Разработка, внедрение и адаптация программного обеспечения отраслевой направленности

Тема: Разработка десктопного приложения для общения

Разработчик: студент гр. ИС-417

Еланцев К.В.

Руководитель: Билан О.О.

Результат защиты\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата защиты\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Курган, 2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ4

ГЛАВА 1 ОБЩАЯ ЧАСТЬ6

1.1 Основы WPF: архитектура и компоненты 6

1.2 C# как язык программирования для WPF 8

1.3 Принципы проектирования пользовательского интерфейса 9

1.4 Сетевое взаимодействие в приложениях 14

ГЛАВА 2 СПЕЦИАЛЬНАЯ ЧАСТЬ20

2.1 Разработка функционала чата 20

2.2  Реализация обмена сообщениями  23

2.3 Обработка ошибок и исключений 24

2.4 Тестирование приложения24

ЗАКЛЮЧЕНИЕ26

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ27

ВВЕДЕНИЕ

В современном мире, где коммуникация играет ключевую роль в различных аспектах жизни, приложения для обмена сообщениями становятся все более значимыми. Они позволяют пользователям общаться в реальном времени, обмениваться информацией и поддерживать социальные связи, независимо от расстояний. Разработка таких приложений требует внимательного подхода к проектированию, функциональности и удобству использования.

Процесс создания приложения для чата включает несколько этапов, начиная с определения требований и проектирования пользовательского интерфейса, и заканчивая реализацией сетевых взаимодействий и тестированием. В рамках данного проекта мы исследуем ключевые аспекты разработки WPF приложения для чата, акцентируя внимание на элементах, которые обеспечивают удобство и надежность общения.

Сначала мы рассмотрим основы WPF и его возможности, что позволит понять, почему эта технология была выбрана для разработки. Это также поможет определить, как эффективно использовать компоненты WPF для создания интуитивно понятного интерфейса.

Затем мы изучим процесс реализации основных функций чата, таких как ввод имени пользователя, создание чата и обмен сообщениями. Мы также рассмотрим обработку ошибок и исключений, что является важным аспектом для обеспечения стабильности и надежности приложения.

Наконец, будет проведено тестирование приложения, что позволит оценить его производительность и выявить возможные улучшения. Этот проект стремится предоставить всесторонний обзор процесса разработки WPF приложения для чата, что позволит создать функциональное и привлекательное решение для пользователей.

Цель курсового проекта заключается в разработке игрового приложения для мобильных устройств. Для достижения этой цели необходимо решить следующие задачи:

1. Изучение современных тенденций в разработке игровых приложений для мобильных устройств;
2. Разработка прототипа игрового приложения;
3. Создание установщика для мобильной платформы Android

1 ОБЩАЯ ЧАСТЬ

1.1 Основы WPF: архитектура и компоненты

Windows Presentation Foundation (WPF) — это графическая подсистема для создания пользовательских интерфейсов в приложениях Windows. WPF предоставляет богатый набор инструментов и возможностей для разработки современных и интерактивных приложений. В этом разделе мы рассмотрим основные аспекты архитектуры WPF и её ключевые компоненты.

Архитектура WPF построена на основе нескольких ключевых концепций и технологий, которые обеспечивают гибкость и мощность при разработке приложений. XAML (Extensible Application Markup Language) — это язык разметки, используемый для определения пользовательского интерфейса в WPF. Он позволяет разделить логику приложения и его представление, что упрощает разработку и поддержку кода. Data Binding позволяет связывать элементы пользовательского интерфейса с данными, что обеспечивает автоматическое обновление интерфейса при изменении данных и наоборот. Templates и Styles позволяют определять внешний вид элементов управления и задавать их стили, что способствует созданию единообразного и привлекательного интерфейса. Commands предоставляют механизм для выполнения действий в ответ на пользовательские взаимодействия, такие как нажатие кнопки или выбор меню.

Ключевые компоненты WPF включают элементы управления, панели компоновки, ресурсы и анимации. WPF предоставляет широкий набор встроенных элементов управления, таких как кнопки, текстовые поля, списки и таблицы, которые можно легко настраивать и расширять. Панели компоновки используются для организации элементов управления в интерфейсе и включают Grid, StackPanel, DockPanel и Canvas. Ресурсы позволяют определять и повторно использовать стили, шаблоны и другие объекты в приложении, что способствует уменьшению дублирования кода и упрощению его поддержки. WPF также поддерживает создание анимаций для элементов управления, что позволяет создавать динамичные и интерактивные интерфейсы.

WPF предоставляет мощные и гибкие инструменты для создания современных пользовательских интерфейсов. Понимание архитектуры и ключевых компонентов WPF является основой для успешной разработки приложений на этой платформе.

1.2 C# как язык программирования для WPF

C# (произносится как "си шарп") — это мощный и гибкий язык программирования, разработанный компанией Microsoft. Он является частью платформы .NET и широко используется для разработки приложений на Windows, включая приложения с графическим интерфейсом, созданные с использованием Windows Presentation Foundation (WPF). В этом разделе мы рассмотрим, почему C# является предпочтительным языком для разработки WPF-приложений и какие возможности он предоставляет.

C# предоставляет богатый набор возможностей, которые делают его идеальным выбором для разработки WPF-приложений. Во-первых, C# является объектно-ориентированным языком, что позволяет создавать модульные и легко поддерживаемые приложения. Объектно-ориентированное программирование (ООП) позволяет разработчикам создавать классы и объекты, которые могут быть повторно использованы и легко расширены. Это особенно важно для создания сложных пользовательских интерфейсов, где требуется высокая степень модульности и повторного использования кода.

Во-вторых, C# поддерживает современные парадигмы программирования, такие как асинхронное программирование и функциональное программирование. Асинхронное программирование позволяет выполнять задачи параллельно, что улучшает производительность и отзывчивость приложений. Это особенно важно для WPF-приложений, где пользовательский интерфейс должен оставаться отзывчивым даже при выполнении ресурсоемких операций. Функциональное программирование, в свою очередь, позволяет писать более компактный и выразительный код, что упрощает разработку и поддержку.

C# также предоставляет мощные средства для работы с данными и интеграции с различными системами. Language Integrated Query (LINQ) позволяет выполнять запросы к данным прямо в коде C#, что упрощает работу с базами данных и коллекциями. Это особенно полезно для WPF-приложений, где часто требуется работа с большими объемами данных и их отображение в пользовательском интерфейсе.

Кроме того, C# тесно интегрирован с платформой .NET, что предоставляет доступ к обширной библиотеке классов и инструментов для разработки. .NET Framework включает в себя множество библиотек для работы с графикой, сетевыми операциями, файловой системой и другими аспектами разработки приложений. Это позволяет разработчикам сосредоточиться на создании пользовательского интерфейса и логики приложения, не беспокоясь о низкоуровневых деталях.

Наконец, C# поддерживает разработку кроссплатформенных приложений через .NET Core и .NET 5/6. Это позволяет создавать WPF-приложения, которые могут работать на различных операционных системах, включая Windows, Linux и macOS. Это особенно важно для разработчиков, которые хотят создавать приложения, доступные для широкой аудитории.

В заключение, C# является мощным и гибким языком программирования, который идеально подходит для разработки WPF-приложений. Его поддержка объектно-ориентированного программирования, асинхронного программирования, функционального программирования, а также тесная интеграция с платформой .NET делают его выбором номер один для создания современных и интерактивных пользовательских интерфейсов.

1.3 Принципы проектирования пользовательского интерфейса

Проектирование пользовательского интерфейса (UI) играет ключевую роль в разработке приложений, особенно в контексте Windows Presentation Foundation (WPF). Эффективный и удобный интерфейс не только улучшает пользовательский опыт, но и способствует более высокой производительности и удовлетворенности пользователей. В этом разделе мы рассмотрим основные принципы проектирования пользовательского интерфейса, которые помогут создать интуитивно понятные и функциональные приложения.

Одним из основных принципов является учет пользовательского опыта (UX). Пользовательский опыт включает в себя все аспекты взаимодействия пользователя с приложением, начиная с первого впечатления и заканчивая выполнением задач. Важно, чтобы интерфейс был интуитивно понятным и легко осваиваемым, что позволит пользователям быстро и эффективно выполнять свои задачи. Для этого необходимо проводить исследования пользователей, создавать прототипы и тестировать их, чтобы выявить и устранить возможные проблемы на ранних стадиях разработки.

Еще одним важным принципом является консистентность. Консистентность означает, что все элементы интерфейса должны быть единообразными и последовательными. Это включает в себя использование одних и тех же стилей, цветовых схем, шрифтов и элементов управления по всему приложению. Консистентность помогает пользователям быстро ориентироваться в интерфейсе и понимать, как взаимодействовать с различными элементами. В WPF это достигается с помощью стилей и шаблонов, которые позволяют легко применять единообразный дизайн ко всем элементам интерфейса.

Минимализм и простота также играют важную роль в проектировании пользовательского интерфейса. Излишняя сложность и перегруженность интерфейса могут отвлекать пользователей и затруднять выполнение задач. Важно сосредоточиться на основных функциях и убрать все лишнее, чтобы пользователи могли легко находить нужные им элементы и выполнять действия. Минималистичный дизайн помогает создать чистый и понятный интерфейс, который не перегружает пользователя.

Обратная связь является еще одним важным принципом. Пользователи должны получать мгновенную обратную связь на свои действия, будь то нажатие кнопки, ввод данных или выполнение операции. Это помогает пользователям понимать, что их действия были зарегистрированы и обработаны системой. В WPF это можно реализовать с помощью анимаций, уведомлений и других визуальных эффектов, которые информируют пользователя о состоянии приложения.

Доступность также является важным аспектом проектирования пользовательского интерфейса. Приложение должно быть доступным для всех пользователей, включая тех, кто имеет ограничения по зрению, слуху или двигательным функциям. В WPF существуют инструменты и библиотеки, которые помогают создавать доступные интерфейсы, такие как поддержка экранных читателей, возможность изменения размера шрифта и контрастности, а также навигация с помощью клавиатуры.

Наконец, тестирование и итеративное улучшение являются неотъемлемой частью процесса проектирования пользовательского интерфейса. Важно регулярно тестировать интерфейс с реальными пользователями, собирать их отзывы и вносить необходимые улучшения. Это позволяет постоянно совершенствовать интерфейс и адаптировать его под потребности пользователей.

1.3 Принципы проектирования пользовательского интерфейса

Сетевое взаимодействие является неотъемлемой частью современных приложений, особенно тех, которые разрабатываются с использованием Windows Presentation Foundation (WPF). Взаимодействие с сетью позволяет приложениям обмениваться данными с удаленными серверами, обеспечивать многопользовательские функции и интегрироваться с другими системами. В этом разделе мы рассмотрим основные аспекты сетевого взаимодействия в WPF-приложениях, не используя веб-сервисы.

Одним из ключевых аспектов сетевого взаимодействия является работа с сокетами. Сокеты предоставляют низкоуровневый доступ к сетевым операциям, таким как отправка и получение данных через сеть. В .NET Framework, который используется в WPF, существует класс Socket, который позволяет создавать и управлять сетевыми соединениями. Сокеты могут работать как с TCP, так и с UDP протоколами, что делает их универсальным инструментом для сетевого взаимодействия.

Асинхронное программирование играет важную роль в сетевом взаимодействии. Асинхронные операции позволяют выполнять сетевые запросы и другие ресурсоемкие задачи без блокировки основного потока выполнения, что обеспечивает отзывчивость пользовательского интерфейса. В C# асинхронное программирование реализуется с помощью ключевых слов async и await. Эти ключевые слова позволяют писать асинхронный код, который выглядит как синхронный, что упрощает его чтение и поддержку.

Безопасность является критически важным аспектом сетевого взаимодействия. При передаче данных через сеть необходимо обеспечить их конфиденциальность и целостность. Для этого используются различные методы шифрования и аутентификации. В WPF-приложениях можно использовать SSL/TLS для защиты данных при передаче, а также различные механизмы аутентификации, такие как сертификаты и токены.

Обработка ошибок также является важным аспектом сетевого взаимодействия. Сетевые запросы могут завершаться неудачей по различным причинам, таким как отсутствие сетевого подключения, ошибки сервера или неправильные запросы. Важно корректно обрабатывать эти ошибки и предоставлять пользователю информативные сообщения. В WPF можно использовать механизмы обработки исключений для перехвата и обработки ошибок сетевых запросов.

2 СПЕЦИАЛЬНАЯ ЧАСТЬ

2.1 Разработка игрового процесса

Это является одним из ключевых этапов в создании успешной игры. Игровой процесс - это то, что делает вашу игру увлекательной и удерживает игроков на протяжении длительного времени. Для создания увлекательного игрового процесса необходимо тщательно продумать механику игры, управление и логику.

В процессе разработки игрового процесса используются современные подходы и технологии, чтобы создать игру, которая не только будет соответствовать современным стандартам, но и превзойдет ожидания игроков. В основе будут использоваться: платформа Unity и язык программированияC#.

В первом этапе разработки, были созданы сцена и объекты (Рисунок 6):

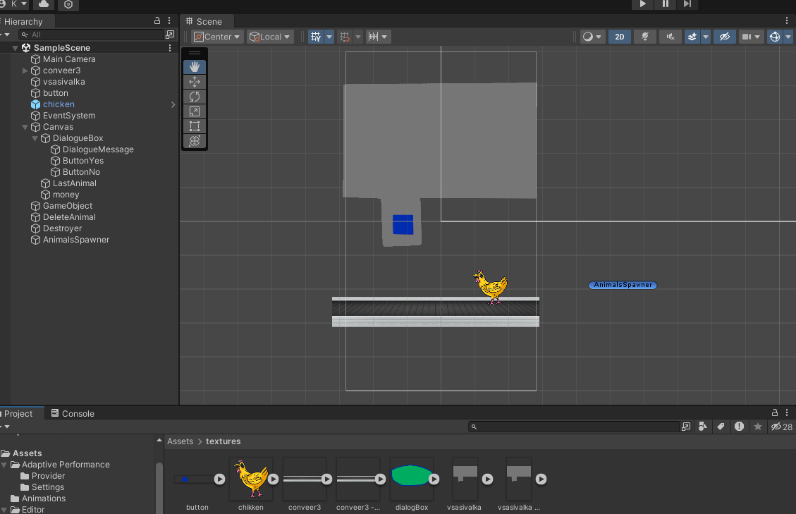


Рисунок 6 – Сцена и объекты

Во втором этапе разработки, после создания сцены и объектов, так же был создан базовый скрипт для добавления характеристик животному:

1. rares = new List<string>() { "Обычная", "Необычная", "Редкая", "Мифическая", "Легендарная" };
2. rare = rares[Random.Range(0, rares.Count)];
3. if (rare == "Обычная")
4. {
5. level = Random.Range(1, 20);
6. coast = level \* 1.2f;
7. }
8. else if (rare == "Необычная")
9. {
10. level = Random.Range(21, 40);
11. coast = level \* 1.4f;
12. }
13. else if (rare == "Редкая")
14. {
15. level = Random.Range(41, 60);
16. coast = level \* 1.6f;
17. }
18. else if (rare == "Мифическая")
19. {
20. level = Random.Range(61, 80);
21. coast = level \* 1.8f;
22. }
23. else if (rare == "Легендарная")
24. {
25. level = Random.Range(81, 100);
26. coast = level \* 2f;
27. }

Затем, были написаны скрипты для взаимодействия нашего животного с объектами на сцене и меню выбора:

* Общий скрипт взаимодействия животного с объектом и выводом данных на меню с выбором:

1. private void OnTriggerEnter2D(Collider2D collision)
2. {
3. rb = collision.GetComponent<Rigidbody2D>();
4. ac = collision.GetComponent<AnimalsController>();
5. effector.speed = 0;
6. anim.SetBool("BoxOpen", true);
7. text.text = $"Животное: Курица \n Редкость: {ac.rare} \n Уровень: {ac.level}";
8. }
9. private void Update()
10. {
11. if (choose == 1)
12. {
13. rb.AddForce(new Vector2(0, 1) \* 1000);
14. anim.SetBool("BoxOpen", false);
15. effector.speed = -2;
16. choose = 2;
18. }
19. if (choose == 0)
20. {
21. effector.speed = -2;
22. anim.SetBool("BoxOpen", false);
23. choose = 2;
24. }
25. }
26. private void Awake()
27. {
28. choose = 2;
29. }

* Скрипт, который взаимодействует с животным после того, как была нажата кнопка «Да» в меню выбора:

1. private void OnTriggerEnter2D(Collider2D collision)
2. {
3. ac = collision.GetComponent<AnimalsController>();
4. lastAnimal.text = $"Животное: Курица \n Редкость: {ac.rare} \n Уровень: {ac.level}";
5. Destroy(collision.gameObject);
6. }

* Скрипт для удаления животного:

1. private void OnTriggerEnter2D(Collider2D collision)
2. {
3. Destroy(collision.gameObject);
4. }

* Скрипт для вспомогательного функционала меню выбора:

1. public void Ok()
2. {
3. stopper.choose = 1;
4. }
5. public void No()
6. {
7. stopper.choose = 0;
8. }

* Скрипт для логики денег в игре:

1. void Update()
2. {
3. if (moneyTime > 36000f)
4. {
5. moneyTime = 0;
6. MoneyGiver();
7. moneyTable.text = $"{money}";
8. }
9. }
10. void FixedUpdate()
11. {
12. moneyTime += 1;
13. }
14. private void OnTriggerEnter2D(Collider2D collision)
15. {
16. ac = collision.GetComponent<AnimalsController>();
17. moneyPerTM = ac.coast;
18. }
19. void MoneyGiver()
20. {
22. if (lastAnimal.text != "Пусто")
23. {
25. money += moneyPerTM;
26. }
27. else
28. {
29. moneyPerTM = 0;
30. }
31. }

2.2 Создание установщика для мобильной платформы Android

Для того, чтобы игра могла быть установлена на мобильное устройство платформы Android, был сделан установщик.

1. Настройка экрана перед компилированием APK-файла (Рисунок 7)

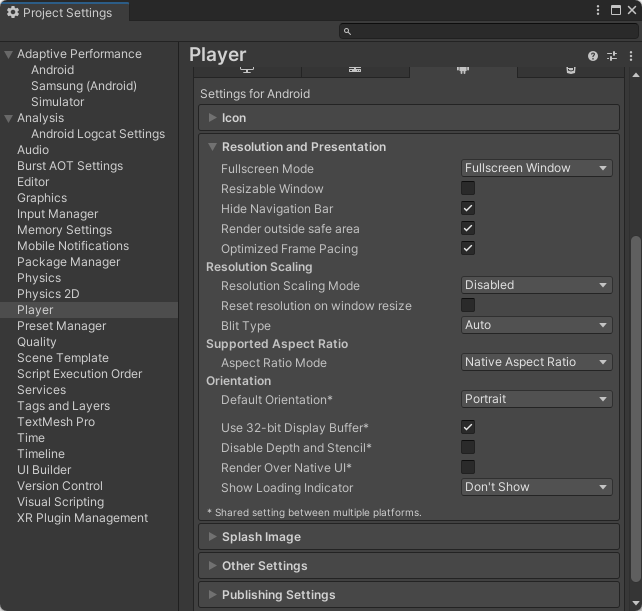


Рисунок 7 – Настройка проекта

1. Процесс компилирования APK-файла:

2.1. После настройки экрана можно приступать к компиляции (Рисунок 8)

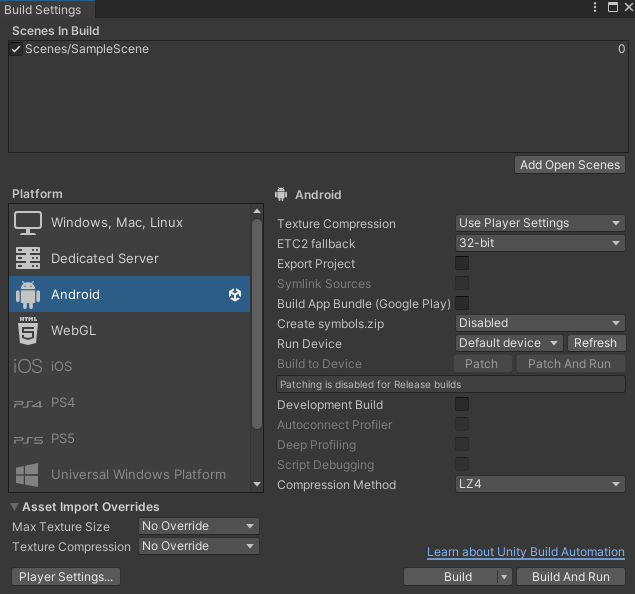


Рисунок 8 – Основное меню настроек проекта

1. Затем, как сборка завершена, можно отправлять файл на мобильное устройство, для дальнейшей установки.

2.3 Руководство пользователя

1. Установка и запуск:

Так как данная игра предоставляется в виде APK-файла, для ее установки не требуется специального установщика. Процесс установки и запуска следующий:

1.1. Загрузка APK-файла:

1.1.1. Скачайте APK-файл с игрой с официального источника или любого другого надежного ресурса.

1.2 После завершения загрузки откройте APK-файл и установите игру.

1.3. Запуск игры:

1.4. В экране приложений появилась игра, которую можно открыть.

2. Основные элементы управления:

2.1. Меню выбора, в котором есть две кнопки «Да» или «Нет», на которые игроку необходимо нажимать.

ЗАКЛЮЧЕНИЕ

В результате выполнения курсового проекта была успешно разработана игра для мобильных устройств. Было проведено исследование современных тенденций в разработке игровых приложений для мобильных устройств, что позволило спроектировать игровой процесс, соответствующий потребностям и ожиданиям современных пользователей.

Была изучена среда разработки Unity, ее основные принципы и функции. Это позволило создать прототип игрового приложения, который был далее разработан и доработан в соответствии с требованиями задания

Также было создано руководство пользователя, которое подробно описывает игровой процесс и особенности управления. Для удобства установки игры на мобильные устройства был создан установщик для платформы Android.

В ходе работы над проектом были приобретены навыки разработки игровых приложений, использования среды разработки Unity, а также навыки создания установщиков для мобильных платформ.

В целом, цель курсового проекта была успешно достигнута. Была разработана игра для мобильных устройств, которая соответствует современным тенденциям и требованиям пользователей. Разработанное приложение может быть использовано для дальнейшего развития и улучшения, а также для создания новых игровых проектов.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Компонентно-ориентированная архитектура [Электронный ресурс]. // Режим доступа: https://coderlessons.com/tutorials/akademicheskii/izuchite-dizain-arkhitektury-programmnogo-obespecheniia/komponentno-orientirovannaia-arkhitektura (Дата обращения: 13.03.2024)
2. 8 лучших методологий разработки ПО в 2023 году [Электронный ресурс]. // Режим доступа: https://www.purrweb.com/ru/blog/metodologii-dlya-razrabotki-po/ (Дата обращения: 13.03.2024)
3. Agile: что это такое и где используется, принципы методологии [Электронный ресурс]. // Режим доступа: https://practicum.yandex.ru/blog/metodology-agile/ (Дата обращения: 13.03.2024)
4. Что такое Scrum-фреймворк и как его использовать? [Электронный ресурс]. // Режим доступа: https://kaiten.ru/blog/chto-takoie-scrum-i-kak-ispolzovat/#:~:text=Scrum%20—%20это%20набор%20правил%2C%20благодаря,методологии%20управления%20проектами%20по%20Agile. (Дата обращения: 13.03.2024)
5. Lean-методология — принципы бережливого производства [Электронный ресурс]. // Режим доступа: https://teachline.ru/blog/lean-metodologiya-printsipy-berezhlivogo-proizvodstva/ (Дата обращения: 13.03.2024)
6. Что такое Прототипирование? [Электронный ресурс]. // Режим доступа: https://www.uprock.ru/articles/chto-takoe-prototipirovanie#:~:text=По%20сути%2C%20прототип%20—%20это%20предварительный,мышления%2C%20который%20сопровождается%20тестированием%20юзабилити./ (Дата обращения: 13.03.2024)
7. Костер Р. Разработка игр и теория развлечений / Рэф Костер – 2018. – 270 с. : 131 ил.
8. Зубек Р. Элементы гейм-дизайна. Как создавать игры, от которых невозможно оторваться / Роберт Зубек. . – 2020. – 272 с. : ил.
9. Создание дизайн-системы для игры: детальный разбор подхода [Электронный ресурс]. // Режим доступа: https://habr.com/ru/companies/plarium/articles/495864/ (Дата обращения: 13.03.2024)