

Введение

На учебной практике была поставлена задача разработать игровое приложение на тему: «Энергоэффективность в доме».

Цель проекта – разработка игрового приложения, которое предоставит развлекательный метод обучения принципам энергосбережения, давая игрокам возможность в реальном времени понять, какие электроприборы потребляют больше энергии и каким образом их можно отключить для уменьшения нагрузки на энергосистему. Игра будет представлять собой набор уровней – виртуальных комнат дома, содержащих энерго-потребители. Каждая комната будет представлять собой новый уровень сложности, предоставляя новые вызовы и сценарии. Игроки будут получать обратную связь по результатам каждого уровня, а также будут иметь возможность отслеживать свой общий прогресс в освоении навыков энергосбережения. С целью вовлечения пользователя, игровое приложение будет иметь яркую и красочную стилизацию, вручную отрисованный интерфейс, уровни, элементы обстановки и декорации.

Далее приведём краткое описание разделов пояснительной записки.

Первый раздел носит название «Анализ задачи». Он посвящён изучению предметной области и организационно-экономической сущности поставленной задачи. Также в нем описано, как задача решается в настоящее время, перечислены входные и выходные данные. В подразделе «Инструменты разработки» будет рассмотрена среда, в которой создаётся данный проект. Здесь также будут установлены минимальные и оптимальные требования к аппаратным характеристикам, обеспечивающим правильное функционирование поставленной задачи.

В разделе «Проектирование задачи» будут рассмотрены основные аспекты разработки игрового приложения. Здесь можно будет узнать об организации данных в контексте среды разработки. В данном разделе будет чётко описан пользовательский интерфейс, составлены алгоритмы процесса обработки информации.

«Реализация задачи» – это третий раздел отчёта по практике, в котором описываются все элементы и объекты, которые будут использованы при реализации данного приложения. В этом разделе будут чётко описаны функции пользователя и их структура. Описано руководство программиста и будет предоставлена диаграмма компонентов.

Четвёртый раздел – «Тестирование». В нем будет описано полное и функциональное тестирование данной программы, т.е. будет оттестирован каждый пункт меню, каждая операция, которая выполняется игровым приложением. Будут

					УП ТРПО 2-40 01 01.35.38.17.24 ПЗ	Лист
Изм.	Лист	№докум.	Подпись	Дата		3

смоделированы все возможные действия пользователя при работе с игровым приложением, начиная от запуска приложения и заканчивая вариантами выхода.

В разделе «Применение» будет описано назначение, область применения, среда функционирования данного программного продукта.

«Заключение» будет содержать краткую формулировку задачи, результаты проделанной работы, описание использованных методов и средств, описание степени автоматизации процессов на различных этапах разработки.

В разделе «Список используемых источников» будет приведён список используемых при разработке источников.

В приложении А приведены диаграммы.

В приложении Б приведена демонстрация UI и UX дизайнов.

В приложении В приведен часть листинг программы на примере часто-используемого класса CustomSprite.

В приложениях к пояснительной записке будут приведены листинг программы с необходимыми комментариями и диаграммы.

					УП ТРПО 2-40 01 01.35.38.17.24 ПЗ	Лист
Изм.	Лист	№докум.	Подпись	Дата		4

1 Анализ задачи

1.1 Постановка задачи

1.1.1 Организационно-экономическая сущность задачи

Наименование задачи: игровое приложение «Эффективность в доме».

Цель разработки: предоставление пользователю возможности обучения правилам энергоэффективности в доме в игровой форме.

Назначение: данный программный продукт разрабатывается для обучения детей до 12 лет правилам энергоэффективности.

Периодичность использования: во время дополнительных занятий или по мере необходимости.

Источники и способы получения данных: документации Python, Pygame.

Информационная связь с другими задачами: так как программный продукт будет разрабатываться на языке программирования Python, игровое приложение связано с предметом «Инструментальное программное обеспечение». Тема «Энергоэффективность» относит ПП к предмету «Защита окружающей среды».

Обзор существующих аналогичных ПП: рассмотрим приложение «Мы и энергия». Эта игра в стиле квест позволяет в игровой форме познакомиться с методами и технологиями энергосбережения, суть игры в наборе определенного количества очков по показателям «экологичность» и «комфорт», которые игроки получают за каждый свой ход, за минимальное количество ходов. Все эти основные функции и механики так же будут присутствовать в новом ПП, но механика отклика на полезные действия будет сокращена до просмотра результата прохождения уровня только после его завершения, а основная механика изменена на переключение состояния потребителей на каждом уровне.

1.1.2 Функциональные требования

К поставленной задаче были заявлены следующие функциональные требования, которые может выполнять пользователь:

- Начало новой / продолжение существующей игры
- Просмотр карты
- Выбор и прохождение уровней
- Переключение состояния потребителей
- Просмотр результата прохождения уровня
- Просмотр информации о разработчике
- Просмотр справочной системы

					УП ТРПО 2-40 01 01.35.38.17.24 ПЗ	Лист
Изм.	Лист	№докум.	Подпись	Дата		5

1.1.3 Описание входной, выходной и условно-постоянной информации

Вся информация, которой оперирует пользователь в процессе использования игрового приложения подразделяется на:

- входную информацию;
- выходную информацию.

В начале новой игры весь существующий прогресс сбросится. Входная информация – текущий прогресс, выходная информация – доступен только первый уровень, условно-постоянная информация – данные прогресса прохождения.

При нажатии на кнопку «Продолжить» главного меню пользователь попадает на карту. Входная информация – отсутствует, выходная информация – отображение карты, условно-постоянная информация – список доступных уровней.

При нажатии на доступный маркер на карте пользователь попадает на уровень. Входная информация – отсутствует, выходная информация – отображение уровня, условно-постоянная информация – отсутствует.

При нажатии на кнопку «Меню» появляется внутриигровое меню. Входная информация – отсутствует, выходная информация – отображение внутриигрового меню, условно-постоянная информация – отсутствует.

При нажатии на интерактивный предмет его изображение изменяется. Входная информация – отсутствует, выходная информация – новое изображение предмета, условно-постоянная информация – текущее состояние предмета.

При подтверждении завершения уровня появляется результат прохождения уровня. Входная информация – отсутствует, выходная информация – результат прохождения уровня, условно-постоянная информация – состояние потребителей.

1.1.4 Нефункциональные требования

Требования к применению:

Интерфейс игрового приложения должен быть легким, понятным, функциональным и простым в использовании. Он не должен перегружать пользователя ненужной информацией.

Требования к производительности:

Время отклика интерфейса и элементов игрового приложения не должно превышать 1 секунды.

Требования к реализации:

Для реализации данного программного продукта будут использоваться язык программирования Python и библиотека Pygame.

					УП ТРПО 2-40 01 01.35.38.17.24 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		6

Требования к надёжности:

Игровое приложение должно иметь возможность самовосстановления после сбоя.

Требования к интерфейсу:

При разработке игрового приложения должны быть использованы жёлтые и белые тона. Все элементы интерфейса, уровни и элементы уровней должны быть отрисованы вручную. Должен быть разработан грамотный пользовательский интерфейс.

1.2 Диаграмма вариантов использования

Проектирование в разработке – это процесс создания плана или схемы для реализации программного обеспечения или системы. Цель проектирования - обеспечить соответствие системы требованиям заказчика и пользователей, а также упростить её разработку, тестирование и поддержку.

Диаграмма вариантов использования – это один из видов диаграмм UML, которая показывает, какие функции предоставляет система и как она взаимодействует с внешними сущностями, называемыми актерами.

Диаграмма вариантов использования состоит из следующих элементов:

- Вариант использования – это овал с названием, который описывает конкретную функцию или сервис, который система предоставляет актеру.

- Актер – это человек, организация или другая система, которая использует или влияет на систему.

- Связь – это отношения, которые определяют, как элементы диаграммы взаимодействуют друг с другом и с системой.

- Система – это то, что моделируется диаграммой вариантов использования.

На рисунке А. 1 приложения А представлена диаграмма вариантов использования для игрового приложения.

1.3 Разработка плана работы над проектом

Диаграмма Ганта — это популярный тип столбчатых диаграмм, который используется для иллюстрации плана, графика работ по какому-либо проекту. Является одним из методов планирования проектов. Используется в приложениях по управлению проектами.

Диаграмма Ганта по разработке данного программного обеспечения представлена на рисунке А. 2.

					УП ТРПО 2-40 01 01.35.38.17.24 ПЗ	Лист
Изм.	Лист	№докум.	Подпись	Дата		7

1.4 Выбор стратегии разработки и модели жизненного цикла

Для разработки игрового приложения следует выбрать стратегию разработки и модель жизненного цикла. Выбор модели жизненного цикла на основе характеристик требований находится в таблице 1.

Таблица 1 – Выбор модели жизненного цикла на основе характеристик требований

Критерии категории требований	Каскадная	V-образная	RAD	Инкрементная	Быстрого прототипирования	Эволюционная
1	2	3	4	5	6	7
1 Являются ли требования к проекту легко определяемыми и реализуемыми?	Нет	Нет	Да	Да	Нет	Нет
2 Могут ли требования быть сформулированы в начале ЖЦ?	Нет	Да	Да	Да	Да	Да
3 Часто ли будут изменяться требования на протяжении ЖЦ?	Нет	Нет	Да	Да	Нет	Да
4 Нужно ли демонстрировать требования с целью их определения?	Нет	Нет	Нет	Да	Нет	Да
5 Требуется ли проверка концепции программного средства или системы?	Нет	Нет	Нет	Да	Да	Да
6 Будут ли требования изменяться или уточняться с ростом сложности системы (программного средства) в ЖЦ?	Нет	Нет	Нет	Да	Да	Да
7 Нужно ли реализовать основные требования на ранних этапах разработки?	Да	Да	Да	Да	Да	Да
Итого	1	2	4	7	4	6

Вычисления: 1 за каскадную, 2 за V-образную, 5 за RAD, 7 за инкрементную, 4 за быстрого прототипирования и 6 за эволюционную.

На основе результатов заполнения таблицы 1, подходящей является инкрементная и эволюционная модель.

Выбор модели жизненного цикла на основе характеристик команды разработчиков находится в таблице 2.

Таблица 2 – Выбор модели жизненного цикла на основе характеристик команды разработчиков

Критерии категории команды разработчиков проекта	Каскадная	V-образная	RAD	Инкрементная	Быстрого прототипирования	Эволюционная
1	2	3	4	5	6	7
1 Являются ли проблемы предметной области проекта новыми для большинства разработчиков?	Да	Да	Нет	Нет	Да	Нет
2 Являются ли инструментальные средства, используемые в проекте, новыми для большинства разработчиков?	Нет	Нет	Нет	Да	Да	Нет
3 Изменяются ли роли участников проекта на протяжении ЖЦ?	Нет	Да	Нет	Да	Нет	Нет
4 Является ли структура процесса разработки более значимой для разработчиков, чем гибкость?	Нет	Нет	Да	Да	Нет	Да
5 Важна ли легкость распределения человеческих ресурсов проекта?	Нет	Нет	Да	Нет	Да	Нет
6 Приемлет ли команда разработчиков оценки, проверки, стадии разработки?	Нет	Да	Нет	Да	Нет	Нет
Итого	1	3	2	4	3	1

Вычисления: 1 за каскадную, 3 за V-образную, 2 за RAD, 4 за инкрементную, 3 за быстрого прототипирования и 1 за эволюционную.

На основе результатов заполнения таблицы 2, подходящими являются V-образная, инкрементная и модель быстрого прототипирования.

Выбор модели жизненного цикла на основе характеристик коллектива пользователей находится в таблице 3.

Таблица 3 – Выбор модели жизненного цикла на основе характеристик коллектива пользователей

Критерии категории коллектива пользователей	Каскадная	V-образная	RAD	Инкрементная	Быстрого прототипирования	Эволюционная
1	2	3	4	5	6	7
1 Будет ли присутствие пользователей ограничено в ЖЦ разработки?	Нет	Да	Нет	Нет	Нет	Нет
2 Будут ли пользователи оценивать текущее состояние программного продукта (системы) в процессе разработки?	Нет	Нет	Нет	Да	Да	Да
3 Будут ли пользователи вовлечены во все фазы ЖЦ разработки?	Нет	Нет	Нет	Нет	Да	Нет
4 Будет ли заказчик отслеживать ход выполнения проекта?	Нет	Нет	Да	Да	Нет	Нет
Итого	0	1	1	2	2	1

Вычисления: 0 за каскадную, 1 за V-образную, 1 за RAD, 2 за инкрементную, 2 за быстрого прототипирования и 1 за эволюционную.

На основе результатов заполнения таблицы 3, подходящими являются инкрементная и модель быстрого прототипирования.

Выбор модели жизненного цикла на основе характеристик типа проектов и рисков находится в таблице 4.

Таблица 4 – Выбор модели жизненного цикла на основе характеристик типа проектов и рисков

Критерии категории типов проекта и рисков	Каскадная	V-образная	RAD	Инкрементная	Быстрого прототипирования	Эволюционная
1	2	3	4	5	6	7
1 Разрабатывается ли в проекте продукт нового для организации направления?	Нет	Нет	Нет	Да	Нет	Да
2 Будет ли проект являться расширением существующей системы?	Нет	Да	Да	Да	Нет	Нет
3 Будет ли проект крупно- или среднемасштабным?	Нет	Нет	Нет	Да	Да	Нет

Продолжение таблицы 4

1	2	3	4	5	6	7
4 Ожидается ли длительная эксплуатация продукта?	Да	Да	Нет	Нет	Нет	Нет
5 Необходим ли высокий уровень надежности продукта проекта?	Нет	Да	Нет	Да	Нет	Да
6 Предполагается ли эволюция продукта проекта в течение ЖЦ?	Нет	Нет	Нет	Да	Да	Нет
7 Велика ли вероятность изменения системы (продукта) на этапе сопровождения?	Да	Нет	Да	Да	Да	Да
8 Является ли график сжатым?	Нет	Нет	Да	Да	Да	Да
9 Предполагается ли повторное использование компонентов?	Нет	Нет	Да	Да	Да	Да
10 Являются ли достаточными ресурсы (время, деньги, инструменты, персонал)?	Нет	Нет	Нет	Да	Да	Нет
Итого	2	3	4	9	6	5

Вычисления: 2 за каскадную, 3 за V-образную, 3 за RAD, 6 за инкрементную, 6 за быстрого прототипирования и 9 за эволюционную.

На основе результатов заполнения таблицы 4, подходящей является инкрементная модель.

Подведение итогов со всех предыдущих таблиц будет представлено в таблице 5.

Таблица 5 – Подведение итогов со всех предыдущих таблиц

№ таблицы	Каскадная	V-образная	RAD	Инкрементная	Быстрого прототипирования	Эволюционная
1	2	3	4	5	6	7
1	1	2	4	7	4	6
2	1	3	2	4	3	1
3	0	1	1	2	2	1
4	2	3	4	9	6	5
Итого	4	10	11	22	15	13

Общий итог: в итоге заполнения таблиц наиболее подходящей является инкрементная модель.

1.5 Инструменты разработки

Инструменты, используемые при разработке и написании сопутствующей документации:

Figma – будет использоваться для создания UX/UI макетов проекта, сцен и спрайтов.

DRAW.IO – будет использоваться для создания графической части и разработки UML-диаграмм;

Microsoft Office Word – для написания документации к программному продукту;

Wrike – это веб-ресурс для создания диаграммы Ганта;

Microsoft PowerPoint — программа подготовки презентаций и просмотра презентаций, являющаяся частью Microsoft Office.

Разработка проекта будет происходить на компьютере со следующими параметрами:

процессор: Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz 1.80 GHz

графический процессор: NVIDIA GeForce MX130

объем оперативной памяти: 8.00 GB;

объем места на жёстком диске: 256 GB;

ОС: Windows 11 Pro.

					УП ТРПО 2-40 01 01.35.38.17.24 ПЗ	Лист
Изм.	Лист	№докум.	Подпись	Дата		12

2. Проектирование задачи

2.1 Разработка структуры сайта, системы меню, навигации

Правильное проектирование игрового приложения является ключевым элементом его успеха, определяя удовлетворение пользователей, уникальность геймплея и эффективность технической реализации. Грамотный дизайн обеспечивает не только привлекательность визуального оформления, но и гармоничное сочетание игровых механик, что способствует долгосрочной приверженности пользователей и популярности приложения. (рисунки А. 3 и А. 4).

2.2 Разработка UML-диаграмм

2.2.1 Диаграмма классов

Диаграмма классов в UML - это инструмент визуализации структуры и отношений между классами в программной системе. Ее компоненты включают классы, атрибуты, методы, отношения (ассоциации, агрегации, композиции), модификаторы видимости и интерфейсы. Используется для проектирования архитектуры системы и облегчения понимания взаимосвязей между компонентами.

Диаграмма классов представлена на рисунке А. 5.

2.2.2 Диаграмма деятельности

Диаграмма деятельности в UML - это инструмент визуализации последовательности действий или процессов в системе. Ее компоненты включают активности (действия), решения (распределение потока управления), ресурсы, и переходы между действиями. Используется для моделирования бизнес-процессов, алгоритмов и взаимодействий в системе, облегчая понимание хода выполнения операций.

Диаграмма деятельности представлена на рисунке А. 6.

2.2.3 Диаграмма последовательности

Диаграмма последовательности — UML-диаграмма, на которой для некоторого набора объектов на единой временной оси показан жизненный цикл объекта и взаимодействие актеров информационной системы в рамках прецедента.

Диаграммы последовательностей используются для уточнения диаграмм прецедентов, более детального описания логики сценариев использования. Это

					УП ТРПО 2-40 01 01.35.38.17.24 ПЗ	Лист
Изм.	Лист	№докум.	Подпись	Дата		13

отличное средство документирования проекта с точки зрения сценариев использования.

Диаграммы последовательностей обычно содержат объекты, которые взаимодействуют в рамках сценария, сообщения, которыми они обмениваются, и возвращаемые результаты, связанные с сообщениями. Впрочем, часто возвращаемые результаты обозначают лишь в том случае, если это не очевидно из контекста.

Объекты обозначаются прямоугольниками с подчеркнутыми именами (чтобы отличить их от классов).

Сообщения (вызовы методов) – линиями со стрелками.

Возвращаемые результаты - пунктирными линиями со стрелками.

Прямоугольники на вертикальных линиях под каждым из объектов показывают «время жизни» (фокус) объектов. Впрочем, довольно часто их не изображают на диаграмме, все это зависит от индивидуального стиля проектирования.

Диаграмма деятельности представлена на рисунке А. 7.

2.3 Разработка пользовательского интерфейса

Важным элементом проектирования данного программного продукта является описание внешнего интерфейса разрабатываемого интернет-ресурса.

Для разработки визуального дизайна использовались сдержанные, мягкие цвета для удобства использования программного продукта.

В ходе разработки был спроектирован дизайн элементов интерфейса для игрового приложения «Эффективность в доме».

Для организации эффективной работы пользователя нужно создать целостный программный продукт данной предметной области, в котором все компоненты будут сгруппированы по функциональному назначению.

Прототип – это наглядная модель пользовательского интерфейса. В сущности, это «черновик», созданный на основе представления разработчика о потребностях пользователя. Итоговое отображение программы может отличаться от прототипа.

С прототипами UX/UI можно ознакомиться в приложении Б.

					УП ТРПО 2-40 01 01.35.38.17.24 ПЗ	Лист
Изм.	Лист	№докум.	Подпись	Дата		14

3. Реализация

3.1 Руководство программиста

Программный продукт разработан с использованием языка программирования Python и графической библиотеки Pygame. Библиотека Pygame предоставляет графическую основу для отрисовки спрайтов и собственную систему событий.

Добавление сцен.

Для добавления сцены необходимо создать класс, унаследовав один из трёх классов: Scene для создания обычных сцен с набором спрайтов, Level для создания уровня и реализации логики прохождения или MultiAngleLevel для реализации многоаккурсных уровней. Классы сцен находятся в соответствующих их названию файлах директории objects. Реализация желательна в модуле __init__.py отдельной папки директории scenes. Далее сцена должна быть импортирована в модуль main.py и добавлена в набор сцен.

Заполнение сцены спрайтами.

Для добавления спрайтов можно использовать следующий набор классов: CustomSprite и ConsumerSprite для добавления в Scene и Level, MultiAngleSprite или MultiAngleConsumer для MultiAngleLevel. Реализация схожа со сценами, только пакет спрайта уже будет находится в директории с самой сценой. Далее спрайт импортируется в модуль сцены и добавляется в набор спрайтов сцены.

В ходе создания программного продукта была создана необходимая для проекта структура классов, предоставляющая возможность создания сцен с интерактивной механикой и наборов спрайтов.

3.1.1 Организация данных

Директории, содержащиеся в корневой папке проекта:

.\ - содержит главный модуль, модуль и файл данных.

assets\ - содержит иконку приложения и пустое изображение для спрайтов.

music\ – содержит трек, воспроизводимый на фоне процесса игры.

scenes\ - директория сцен.

scenes\levels\ – директория уровней (сцен).

sounds\ - содержит набор звуков, используемых спрайтами.

objects\ – содержит модули классов.

objects\nested\ - содержит модули классов, реализующих объединение нескольких классов.

ui\ - содержит элементы интерфейса

					УП ТРПО 2-40 01 01.35.38.17.24 ПЗ	Лист
Изм.	Лист	№докум.	Подпись	Дата		15

Каждая директория сцены имеет следующую структуру:

.\objects (опционально) – содержит набор классов, используемых в пределах одной сцены.

.\sprites – содержит набор спрайтов и модуль сцены.

3.1.2 Структура программы

Состав основных модулей и файлов:

config.py – модуль, содержащий константы приложения.

data.py – модуль данных, отвечает за загрузку и сохранение данных в соответствующий файл.

data – файл данных, содержит прогресс прохождения уровней и некоторые флаги состояний, создаётся после первого запуска приложения.

main.py – главный модуль приложения, содержит класс Game и отвечает за основную логику выполнения программы.

3.1.3 Структура и описание процедур и функций пользователя

Процедуры и функции пользователя можно представить в соответствующих методах классов.

Scenes.scene.setter() – переход по сценам.

CustomSprite.on_click() – клик по графическим элементам.

CustomSprite.on_hover() – наведение курсора на графический элемент.

CustomSprite.on_unhover() – прекращение наведения курсора на графический элемент.

Level.confirm() – подтверждение завершения уровня.

Level.complete() – просмотр результата прохождения уровня.

Level.exit_main_menu() – выход из уровня в главное меню.

Consumer.switch_state() – включение/выключение потребителя по клику.

MultiAngleLevel.switch_angle() – интерактивное передвижение по комнате.

Game.quit() – выход пользователя из приложения

4 Тестирование

4.1 Тесты на использование

При разработке данной программы многие возникающие ошибки и недоработки были исправлены на этапе реализации проекта. После завершения испытания реализации программы было проведено тщательное функциональное тестирование. Функциональное тестирование должно гарантировать работу всех элементов программы в автономном режиме.

Отчёт о результатах тестирования предоставлен в таблице 6.

Таблица 6 – Отчёт результатах тестирования

№	Тест	Ожидаемый результат	Физический результат	Результат тестирования
1	2	3	4	5
1	Проверка кнопки «Продолжить»	Открытие карты	Открытие карты	Выполнено
2	Проверка кнопки «Новая игра»	Открытие окна «Подтверждение сброса прогресса»	Открытие окна «Подтверждение сброса прогресса»	Выполнено
3	Проверка кнопки «Выйти из игры»	Выход из игры	Выход из игры	Выполнено
4	Проверка маркера доступного уровня на карте	Открытие уровня	Открытие уровня	Выполнено
5	Проверка кнопки «Меню»	Открытие окна меню	Открытие окна меню	Выполнено
6	Проверка кнопки меню «Продолжить»	Закрытие окна меню	Закрытие окна меню	Выполнено
7	Проверка смены состояния спрайта шнура питания телевизора	Шнур и телевизор меняют своё состояния	Шнур и телевизор меняют своё состояния	Выполнено
8	Проверка звука клика по кнопкам	Воспроизведение звука клика по кнопке.	Воспроизведение звука клика по кнопке.	Выполнено
9	Подтверждение завершения уровня	Отображение результата прохождения	Отображение результата прохождения	Выполнено
10	Неполное прохождение предыдущего уровня	Следующий уровень недоступен	Следующий уровень недоступен	Выполнено

4.2 Отчёт о результатах тестирования

При разработке программного продукта было решено множество проблем, например, при выходе из уровня «Ванная» не отключался звук льющейся воды из душевой и крана или при входе на карту маркеры уровней были кликабельными, хотя сам уровень не должен был быть доступен.

Элементы программы были проверены, и было установлено, что все они работают правильно и выполняют задачи, указанные в процедурах.

					УП ТРПО 2-40 01 01.35.38.17.24 ПЗ	Лист
Изм.	Лист	№докум.	Подпись	Дата		18

5. Руководство пользователя

Целью проекта игрового приложения «Эффективность в доме» является предоставление эффективного и интересного метода обучения принципам энергосбережения, привлекающий внимание участников через разработку увлекательной компьютерной игры. Игра создана с учетом того, чтобы дать игрокам возможность в реальном времени понять, какие электроприборы потребляют энергию и каким образом их можно отключить для уменьшения нагрузки на энергосистему.

Игровое приложение поставляется вместе с установщиком, запустив который запускается стандартный процесс установки ПП.

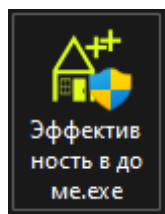


Рисунок 1 – Иконка установщика

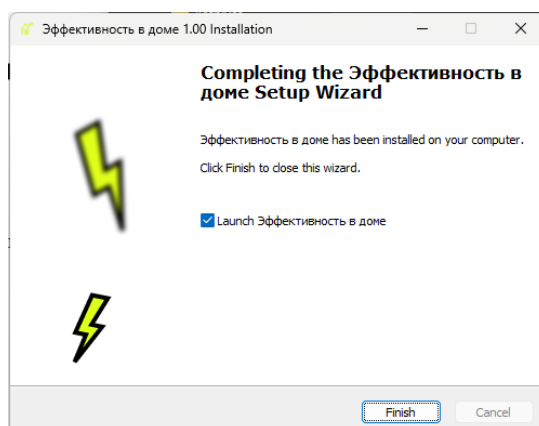


Рисунок 2 – Окно установщика

После установки нам предлагается запустить игру. После запуска приложения видно главное меню с предложением начать игру:

					УП ТРПО 2-40 01 01.35.38.17.24 ПЗ	Лист
Изм.	Лист	№докум.	Подпись	Дата		19



Рисунок 3 – Главное меню игры

После нажатия на кнопку «Новая игра» главного меню появляется окошко с правилами игры, которые помогут в прохождении:

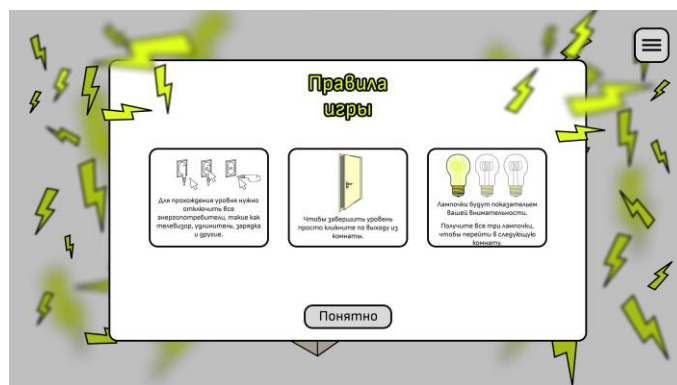


Рисунок 4 – Окно с правилами игры

Закрыв окошко по нажатию кнопки «Понятно», видно карту квартиры, где по стандарту доступен только 1-ый уровень «Спальня».



Рисунок 5 – Карта игры

Нажав на маркер «Спальня», появляется первая комната:



Рисунок 6 – Первый уровень

По нажатию электроприборы приборы отключаются:

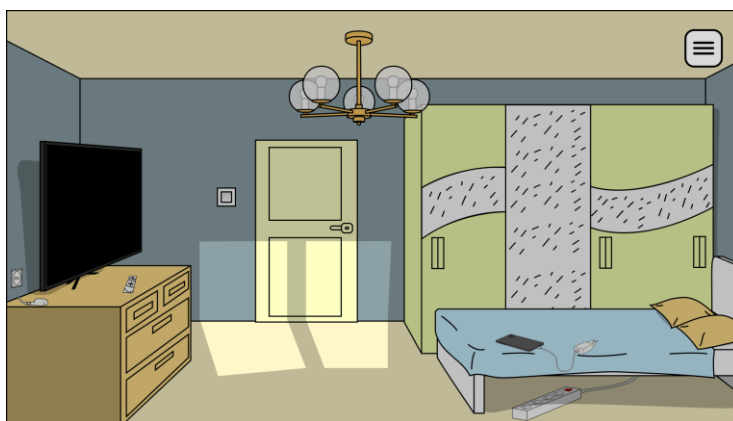


Рисунок 7 – Первый уровень, отключены все приборы

Отключив все приборы и кликнув по выходу, появляется результат прохождения комнаты:



Рисунок 8 – Результат прохождения уровня

После нажатия на «Продолжить», происходит переход на карту с уже доступным следующим уровнем «Ванная», кликнув на маркер которой, видно саму комнату:

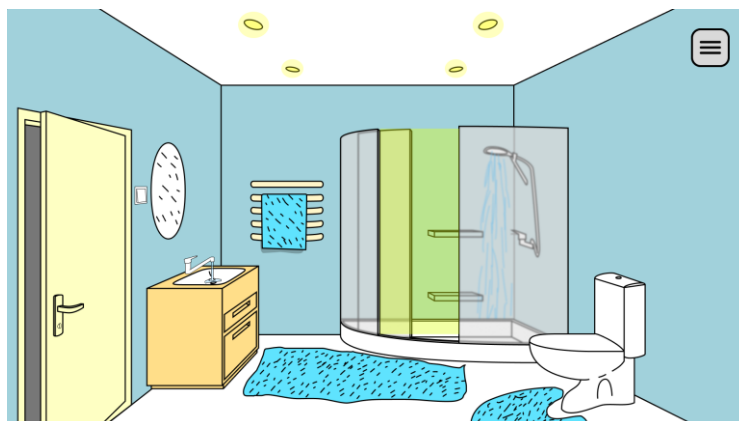


Рисунок 9 – Второй уровень

Внутриигровое меню, содержащее кнопки «Главное меню» для возможности выхода в главное меню и «Продолжить» для закрытия внутриигрового меню, открываемое по нажатию на кнопку в верхнем правом углу, доступно на карте и всех уровнях:

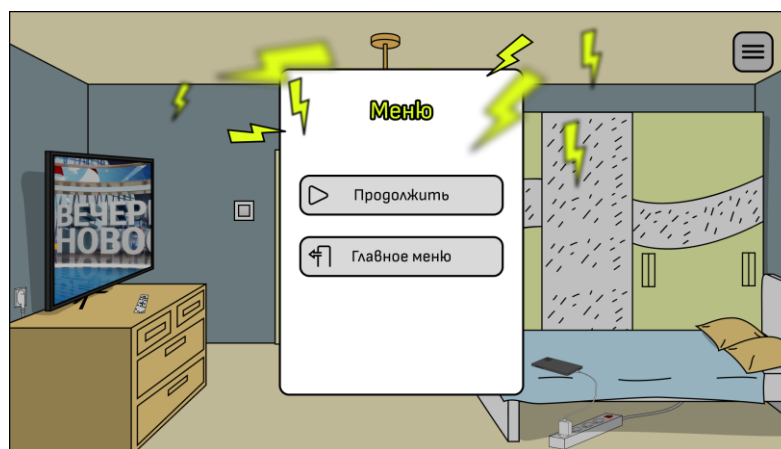


Рисунок 10 – Внутриигровое меню

Не отключив один и более потребителей и завершив уровень, результат прохождения уровня сообщит, что для прохождения на следующий уровень необходимо отключить все потребители:

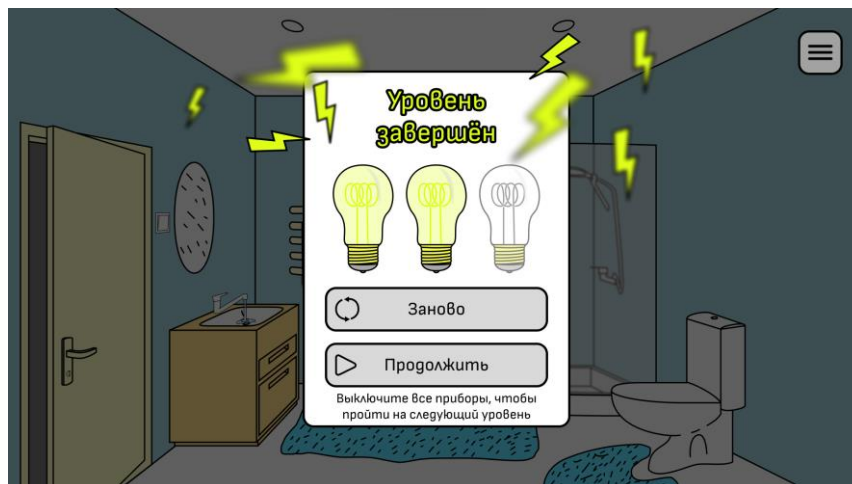


Рисунок 11 – Результат прохождения: неполное прохождение уровня

После отключения всех приборов появляется доступ к 3-ему уровню «Кухня»:



Рисунок 12 – Третий уровень «Кухня»

Ванная комната и кухня имеют механику смены ракурса – симуляцию передвижения по комнате. Области, по которым необходимо произвести нажатие для смены ракурса, подсвечены золотистым цветом:

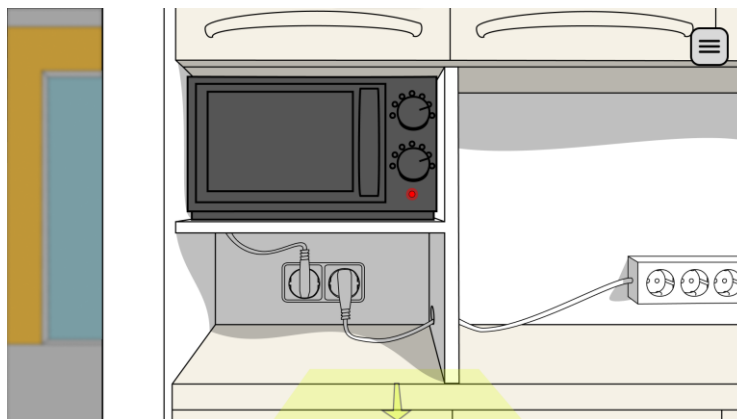


Рисунок 13 – Смена ракурса на кухне на полку с микроволновкой

После успешного завершения третьего и последнего уровня, при выходе на карту, появляется окошко благодарности за прохождение и короткая сводка важности энергоэффективности:



Рисунок 14 – Окно «Спасибо за прохождение игры»

Для выхода из игрового приложения, необходимо перейти в главное меню через внутриигровое меню как было указано ранее и нажать кнопку «Выйти из игры».

Заключение

В результате реализации проекта по учебной практике было разработано игровое приложение «Эффективность в доме», обучающее основам энергоэффективности, на основе библиотеки Python Pygame. Проект представляет собой многоуровневую систему, где каждый уровень представляет собой отдельную комнату с новыми электроприборами и задачами.

В игровом приложении присутствуют следующие механики:

Динамичные интеракции. Пользователи взаимодействуют с различными объектами в каждой комнате, предпринимая действия для отключения потребителей энергии. Динамичные анимации и звуковые эффекты добавляют реализму игровому процессу, делая обучение более захватывающим и запоминающимся.

Внедрена система достижений, стимулирующая игроков к активному участию и успешному завершению уровней.

Обратная связь и оценка прогресса. Игроки получают обратную связь по результатам каждого уровня, а также имеют возможность отслеживать свой общий прогресс в освоении навыков энергосбережения.

В результате использования этих механик проект становится не только образовательным инструментом, но и захватывающим опытом, который активно вовлекает пользователей в процесс обучения, повышая их осведомленность о вопросах энергосбережения и стимулируя практическое применение полученных знаний в повседневной жизни.

В ходе разработки возникали ошибки, которые к окончанию разработки были полностью устранены и больше не появляются.

Программный продукт полностью готов к эксплуатации и расширению через использование созданной структуры.

					УП ТРПО 2-40 01 01.35.38.17.24 ПЗ	Лист
Изм.	Лист	№докум.	Подпись	Дата		25

Список использованных источников

Документация Python [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://docs.python.org/>.

Документация Pygame [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://www.pygame.org/docs/>.

Леоненков, А. В. Самоучитель UML / А. В. Леоненков – СПб.: БХВ-Петербург, 2006.

Брауде, Э. Технология разработки программного обеспечения / Э. Брауде – СПб.: Питер, 2004.

Бахтизин, В. В. Технология разработки программного обеспечения / В. В. Бахтизин – Минск: БГУИР, 2004.

ГОСТ 19.102-77 «Стадии разработки».

ГОСТ 19.402-78 «Описание программы».

ГОСТ 19.502-78 «Описание применения требования к содержанию и оформлению».

ГОСТ 19.701-90 (ИСО 5807) «Условные обозначения и правила выполнения».

СТБ ИСО/МЭК 12207-2003 «Информационные технологии жизненного цикла».

СТБ ИСО/МЭК 9126-2003 «Информационные технологии оценка программной продукции».

					УП ТРПО 2-40 01 01.35.38.17.24 ПЗ	Лист
Изм.	Лист	№докум.	Подпись	Дата		26

Приложение А
Проектная документация

					УП ТРПО 2-40 01 01.35.38.17.24 ПЗ	Лист
Изм.	Лист	№докум.	Подпись	Дата		27

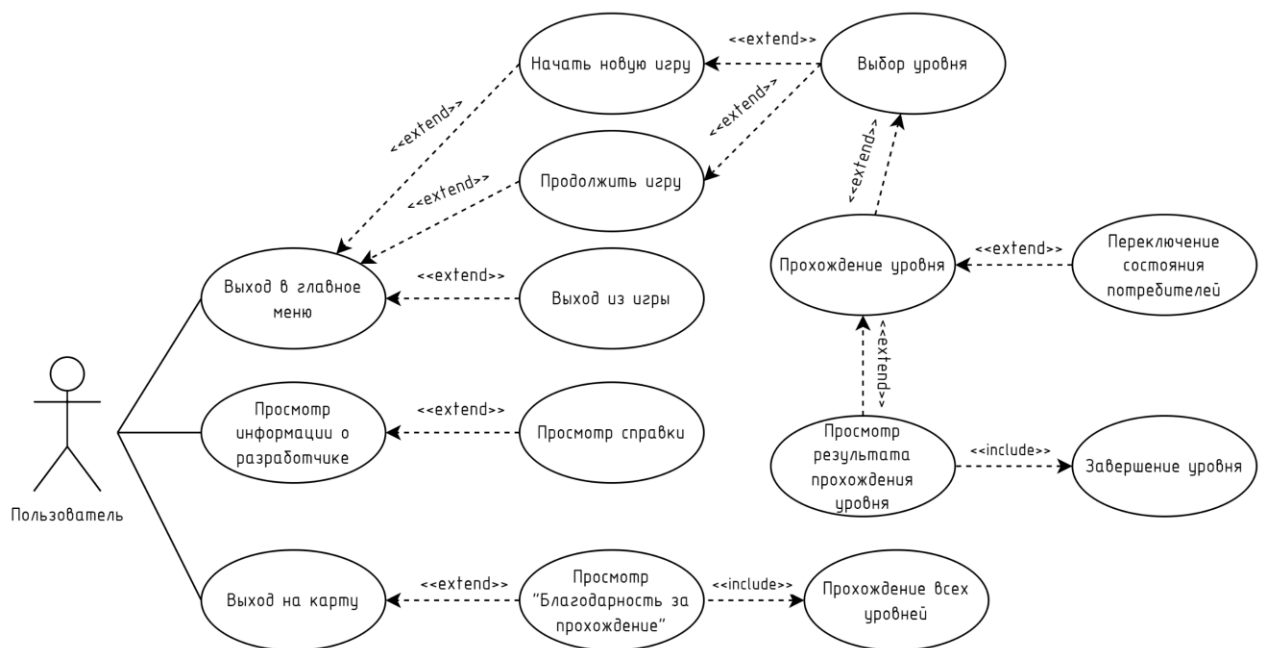


Рисунок А. 1 – Диаграмма вариантов использования

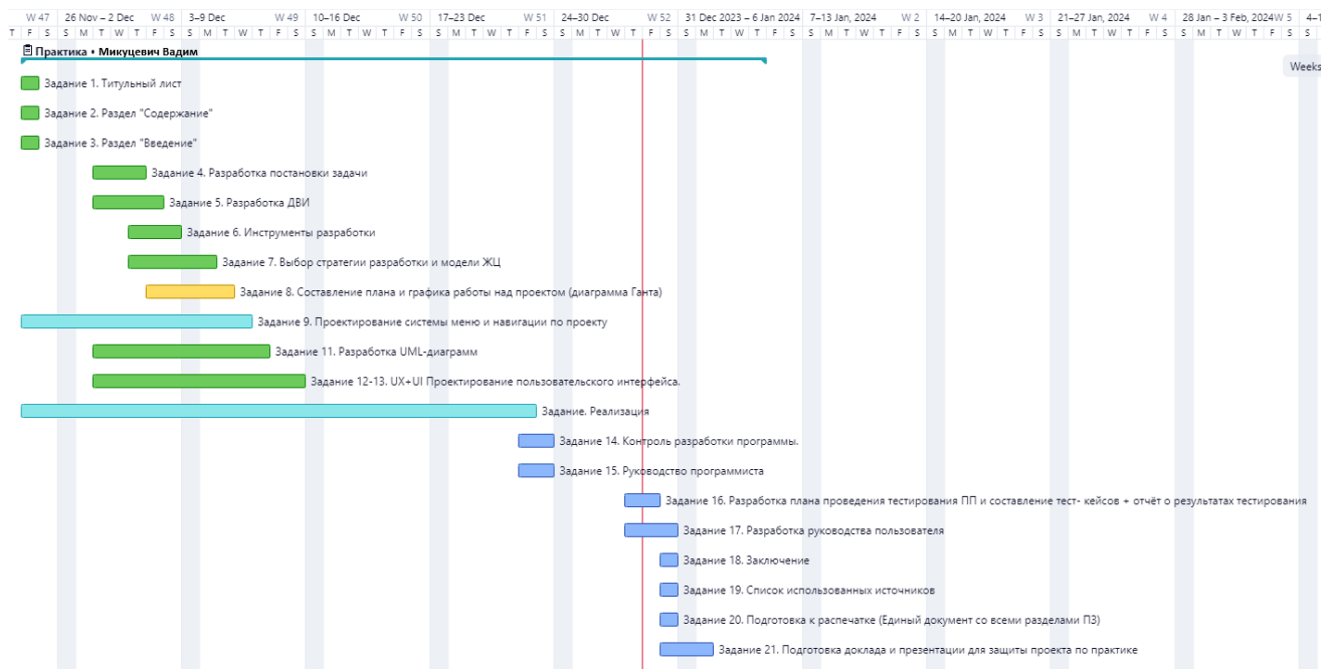


Рисунок А. 2 – Структурное проектирование ПО для сайта

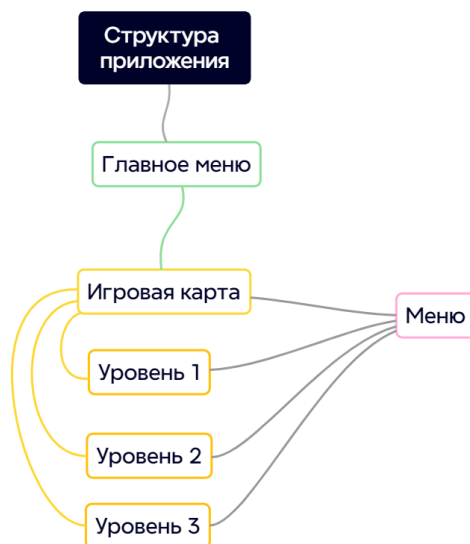


Рисунок А. 3 – Структура сайта



Рисунок А. 4 – Главное меню сайта

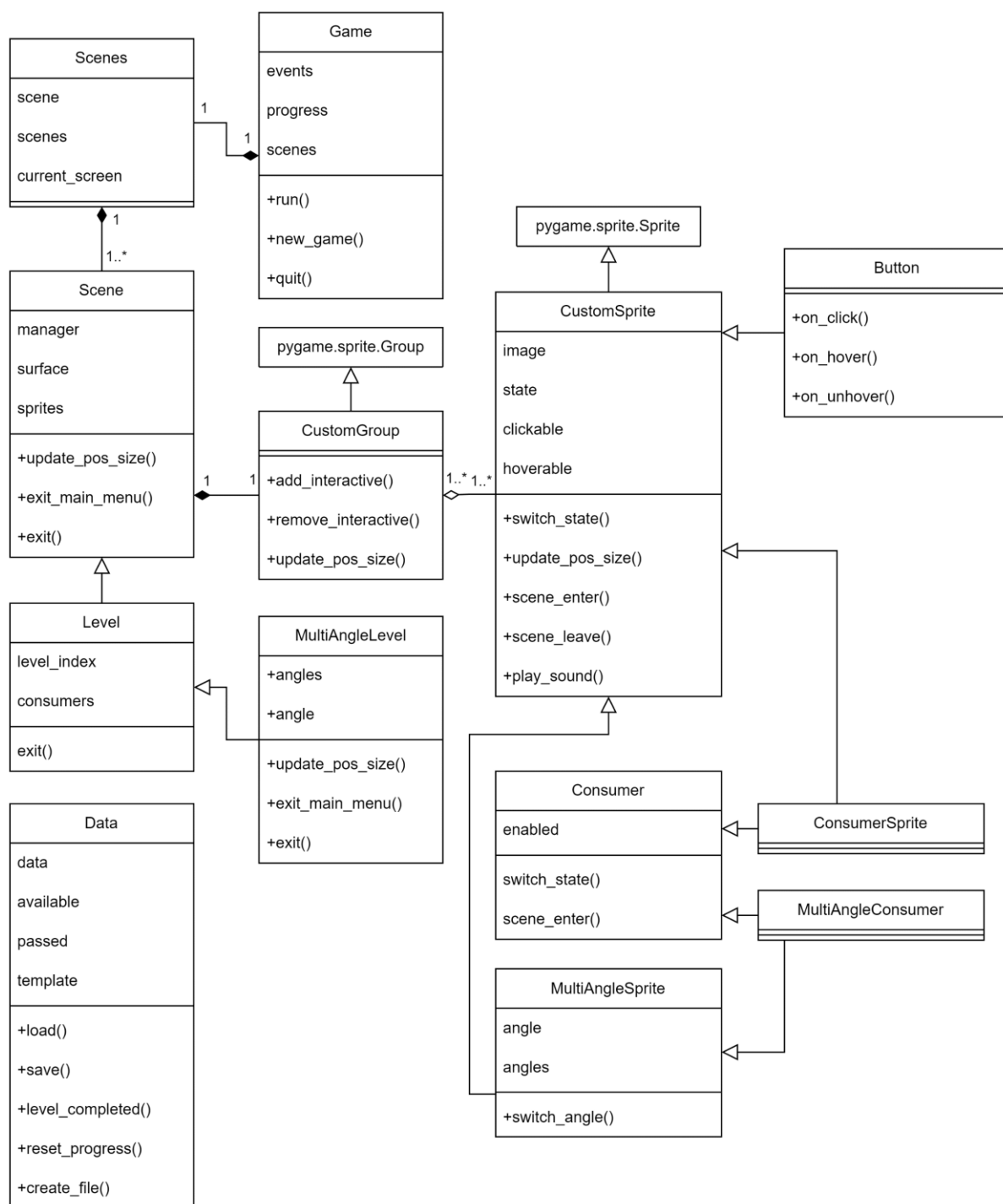


Рисунок А. 5 – Диаграмма классов

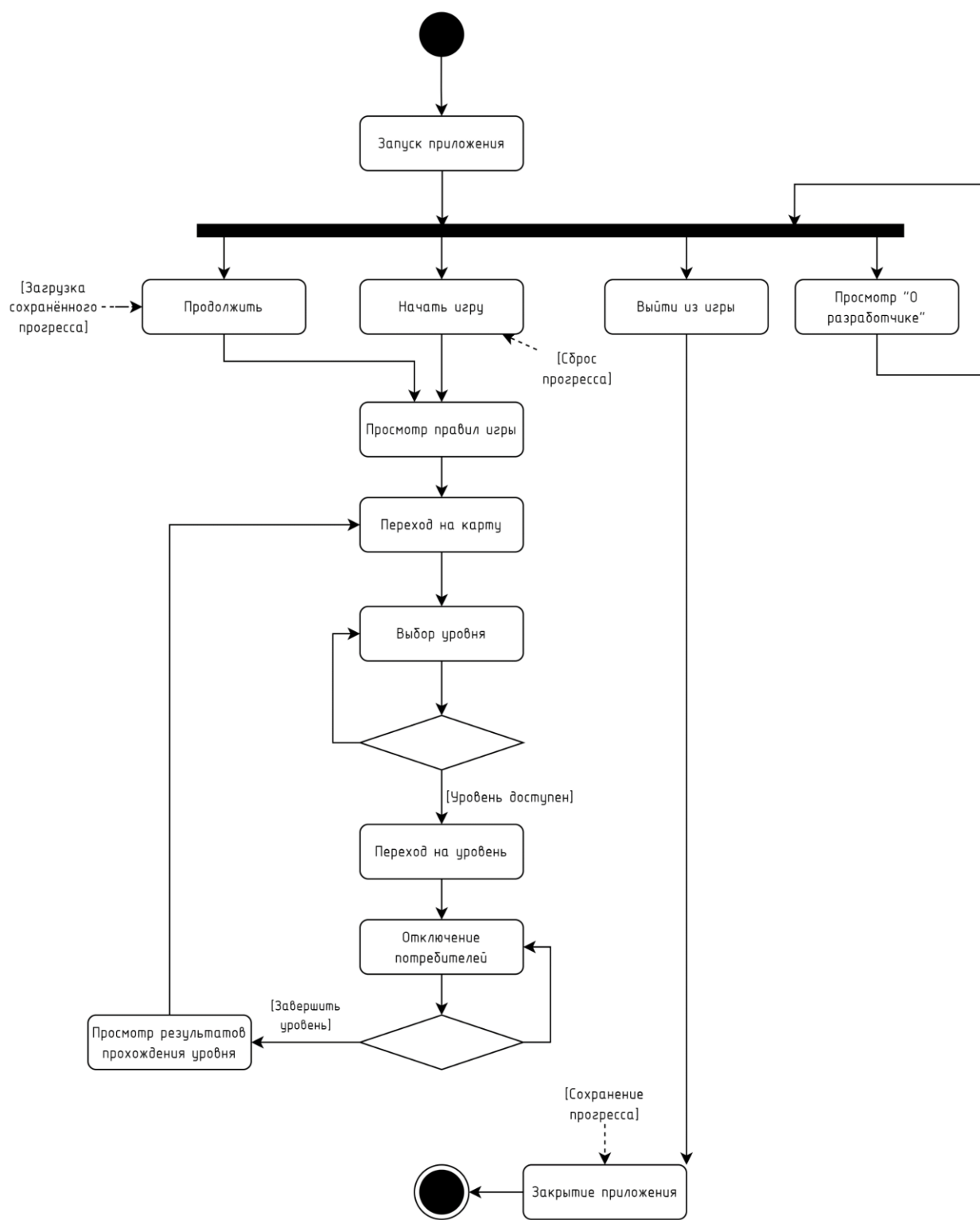


Рисунок А. 6 – Диаграмма деятельности

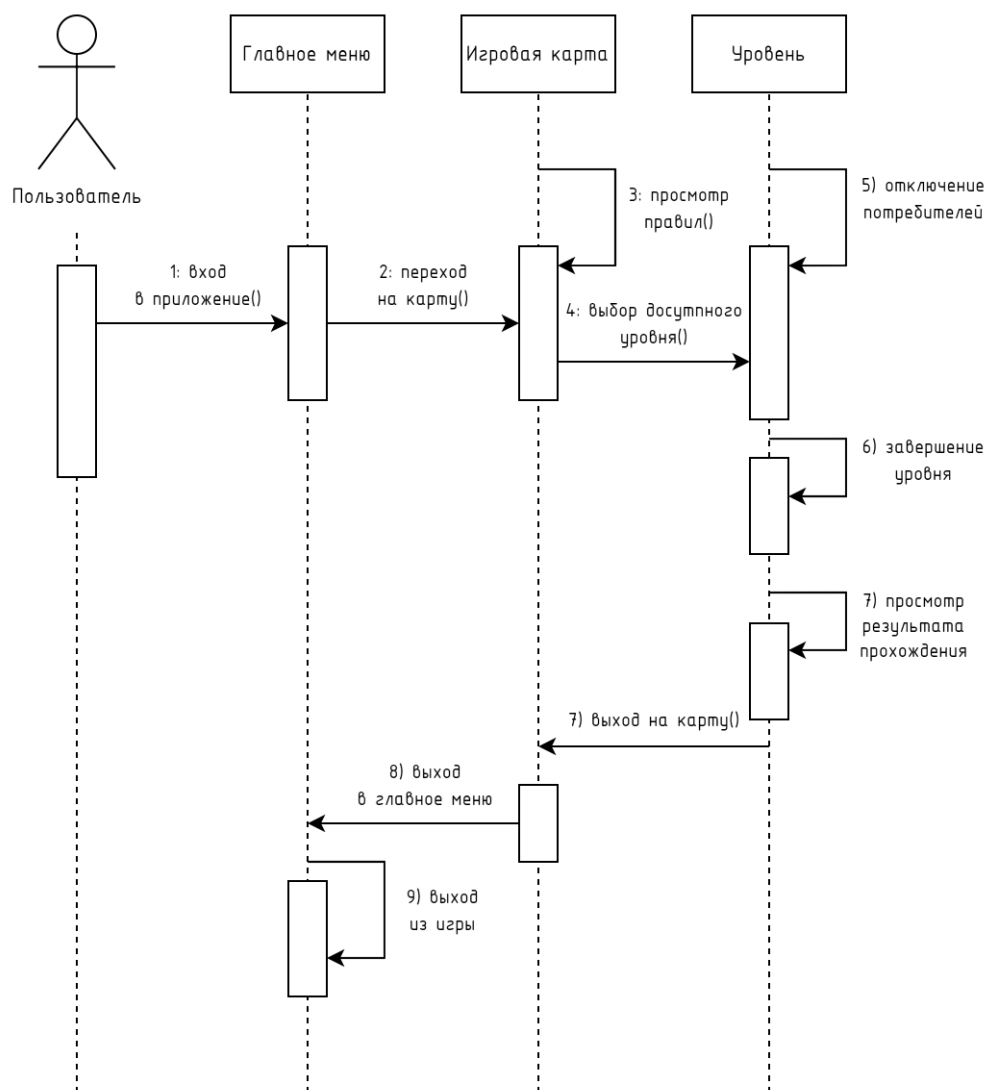


Рисунок А. 7 – Диаграмма последовательности

Приложение Б
UX и UI проектирование

					УП ТРПО 2-40 01 01.35.38.17.24 ПЗ	Лист
Изм.	Лист	№докум.	Подпись	Дата		33

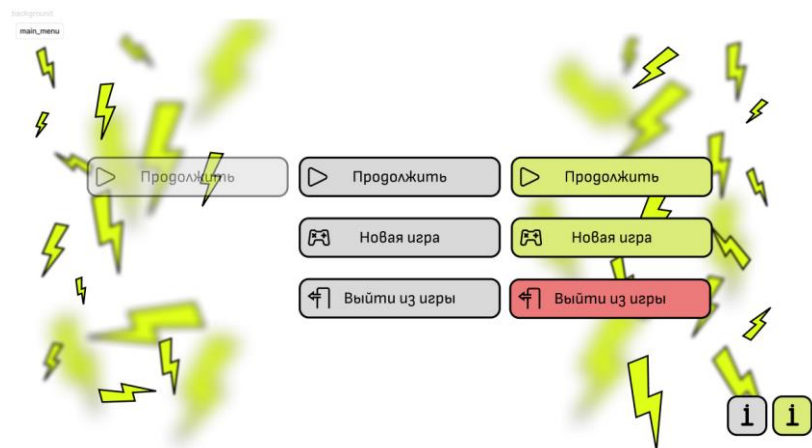


Рисунок Б. 1 – UX/UI проектирование главного меню

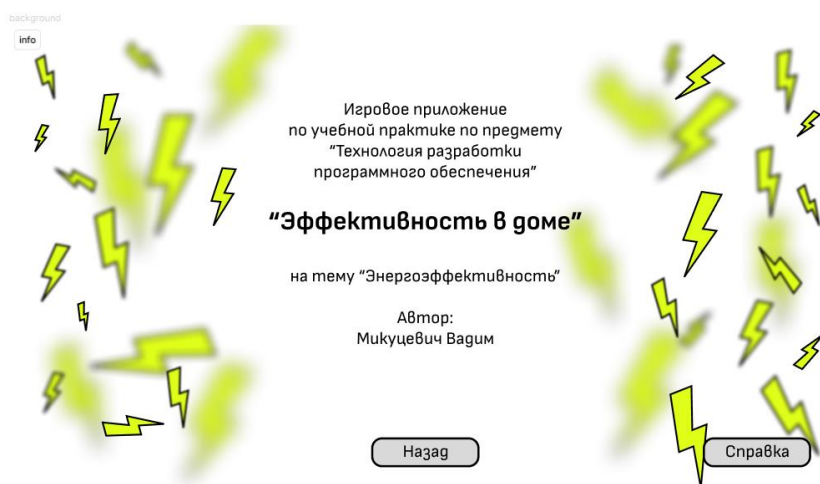


Рисунок Б. 2 – UX/UI проектирование информации о разработчике

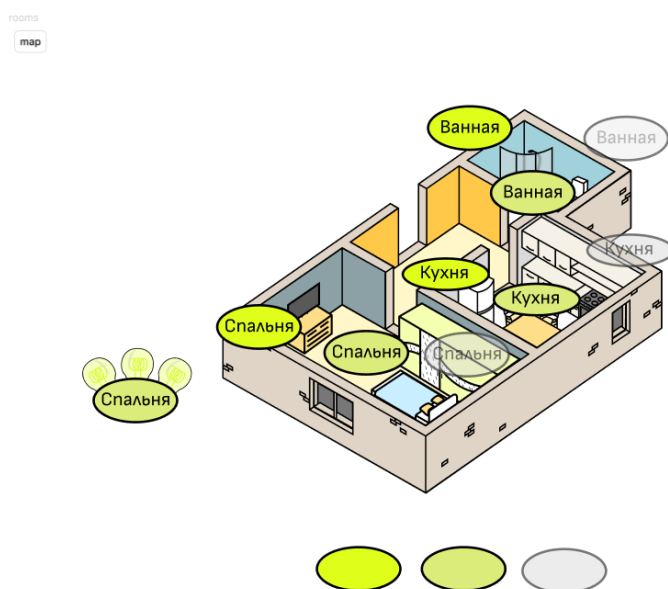


Рисунок Б. 3 – UX/UI проектирование карты

box
confirmLevelComplete

Завершить уровень?

ОтменаЗавершить

ОтменаЗавершить

Рисунок Б. 4 – UX/UI проектирование диалоговых окон

thanks
thanks

Спасибо за прохождение игры

Спасибо за ваше участие в нашем захватывающем квесте "Эффективность в доме!"

Помните, что каждое ваше решение, направленное на экономию энергии, является шагом к более устойчивому и заботливому миру. Благодаря вам, мы совместно строим будущее, где энергия используется разумно, и комнаты наполняются не только уютom, но и ответственностью.

Спасибо за ваш вклад в наше приключение! Мы надеемся, что уроки и впечатления из "Эффективность в доме" с вами навсегда, и вы будете продолжать применять свои знания в повседневной жизни.


Поздравляем вас с победой и благодарим за ваше стремление сделать мир лучше!

Понятно


Рисунок Б. 5 – UX/UI проектирование окна «Спасибо за прохождение игры»

rules
rules


Правила игры



Для прохождения уровня нужно отключить все энергопотребители, такие как телевизор, утюг, зарядка и другие.



Чтобы завершить уровень, просто кликните по выходу из комнаты.



Лампочки будут показателем вашей внимательности. Получите все три лампочки, чтобы перейти в следующую комнату.

Понятно

Рисунок Б. 6 – UX/UI проектирование окна «Правила игры»

Изм.	Лист	№докум.	Подпись	Дата

УП ТРПО 2-40 01 01.35.38.17.24 ПЗ

Лист

35

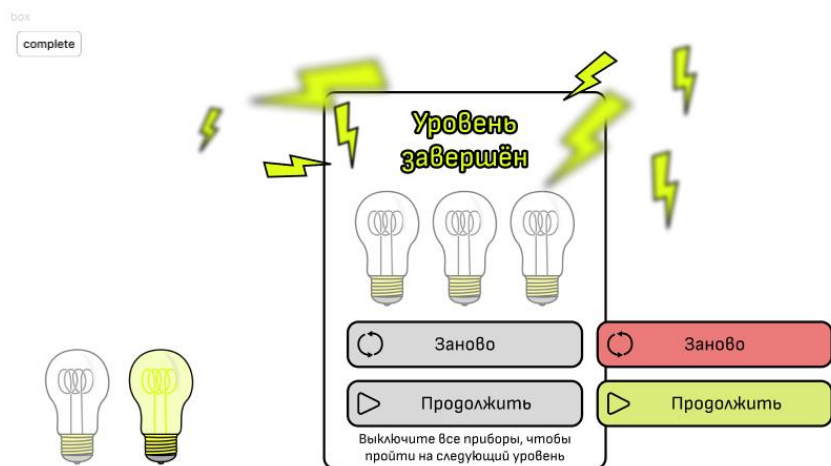


Рисунок Б. 7 – UX/UI проектирование результатов прохождения уровня

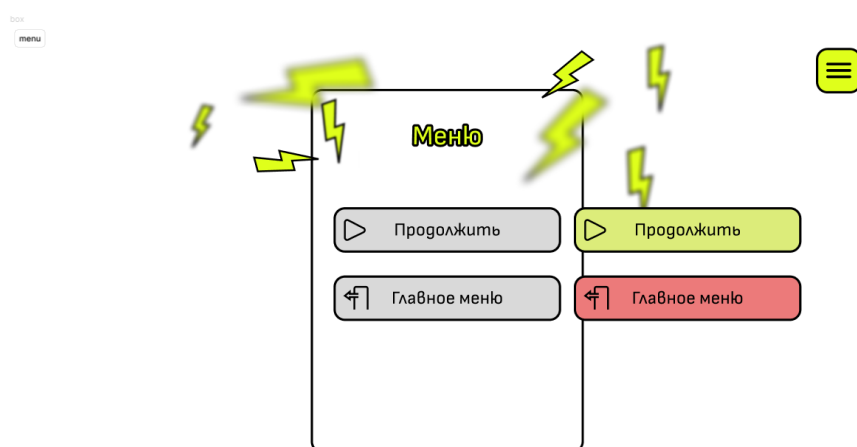


Рисунок Б. 8 – UX/UI проектирование внутриигрового меню

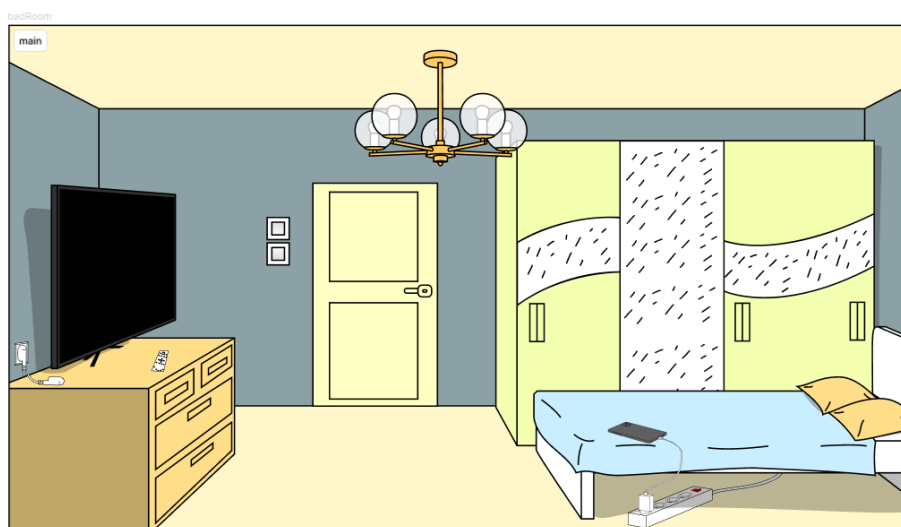


Рисунок Б. 9 – UX/UI проектирование первого уровня

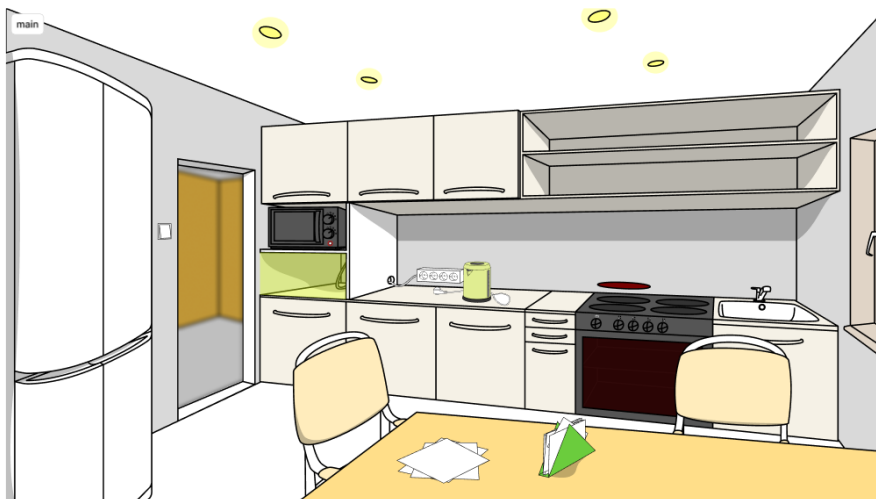


Рисунок Б. 10 – UX/UI проектирование третьего уровня

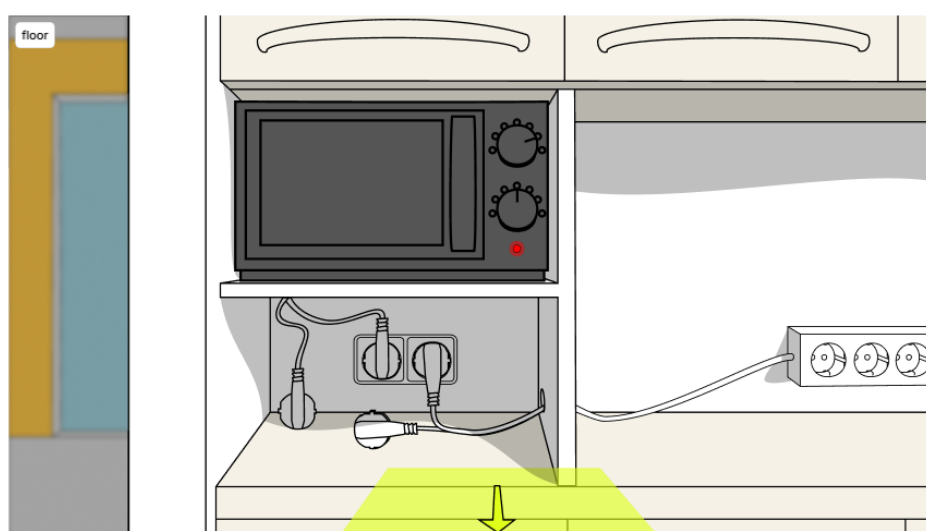


Рисунок Б. 11 – UX/UI проектирование ракурса третьего уровня на полку

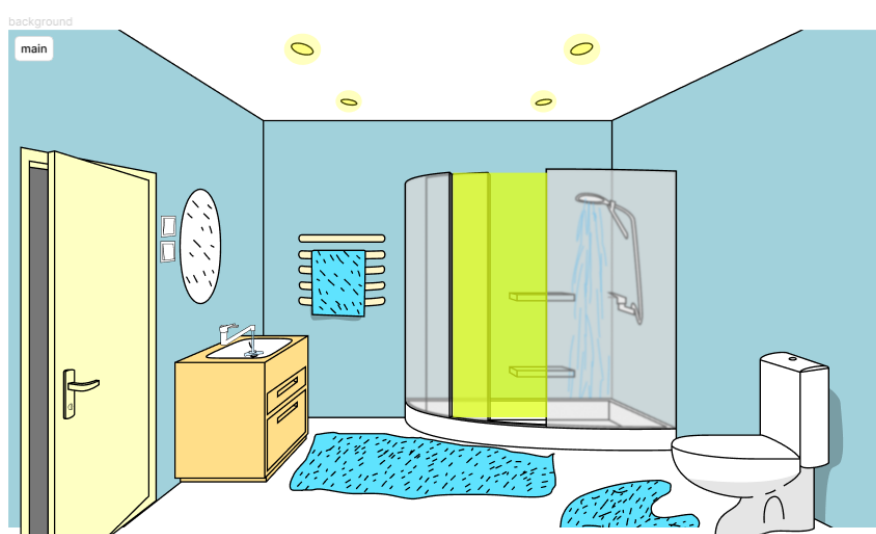


Рисунок Б. 12 – UX/UI проектирование второго уровня

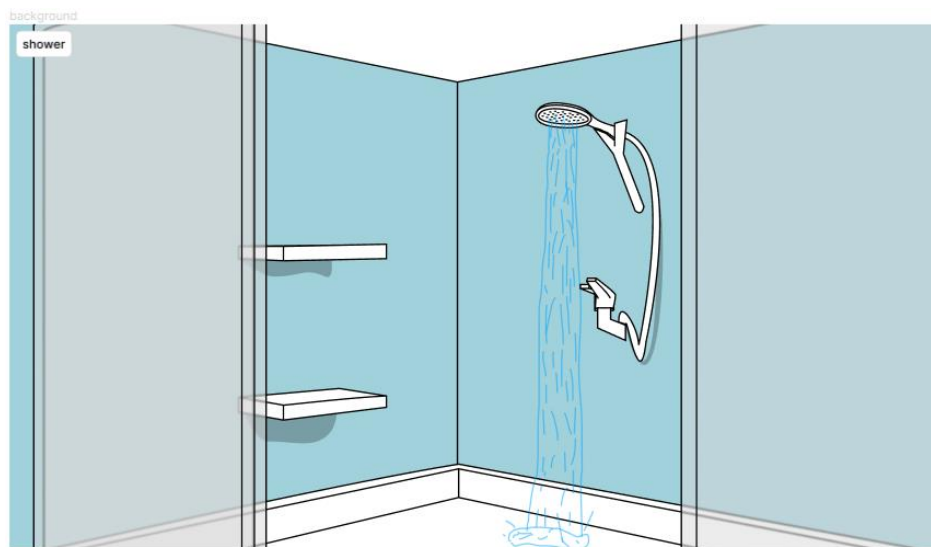


Рисунок Б. 13 – UX/UI проектирование ракурса второго уровня на душевую

Приложение В
Листинг программы

					УП ТРПО 2-40 01 01.35.38.17.24 ПЗ	Лист
Изм.	Лист	№докум.	Подпись	Дата		39

Файл objects/customSprite.py – класс CustomSprite

```
import pygame
from pygame import Surface
from pygame.event import Event
from pygame.mixer import Channel
from pygame.mixer import Sound
from pygame.sprite import Sprite

from .events import DRAW_EVENT


class CustomSprite(Sprite):
    """
    Субкласс класса Sprite.

    Класс спрайтов с состояниями и
    ссылочной связью с экраном.

    Состояние имеет название и определяет
    текущее изображение и позицию спрайта.

    Состояние спрайта изменяется путем передачи
    названия состояния атрибуту state.
    """

    ##### PROPERTIES #####

    ##### image #####

    @property
    def image(self) -> Surface:
        return self._image

    @image.setter
    def image(self, image:Surface) -> None:
        image_size = list(map(
            lambda size: size*self.scene.size_hint,
            image.get_size(),
        ))

        self._image = pygame.transform.smoothscale(image, image_size)
        if hasattr(self, 'rect'):
            x, y = self.rect.x, self.rect.y
            self.rect = self.image.get_rect()
            self.rect.x, self.rect.y = x, y
        else:
            self.rect = self.image.get_rect()

    ##### state #####

    @property
    def state(self) -> str:
        return self._state

    @state.setter
    def state(self, state:str) -> None:
```

					УП ТРПО 2-40 01 01.35.38.17.24 ПЗ	Лист
Изм.	Лист	№докум.	Подпись	Дата		40

```

self._state = state
self.image = self.states[self._state][0]
self.rect.x, self.rect.y = (
    round(self.scene.pos[0]+self.states[state][1][0]*self.scene.size_hint),
    round(self.scene.pos[1]+self.states[state][1][1]*self.scene.size_hint),
)
pygame.event.post(Event(DRAW_EVENT))

@property
def hoverable(self) -> bool:
    """
    Используется сценой при инициализации группы спрайтов сцены.

    При наличии методов on_hover и on_unhover,
    экземпляр автоматически заносится в список спрайтов,
    поддерживающих наведение курсором мыши.
    """

    if (
        hasattr(self, 'on_hover')
        and hasattr(self, 'on_unhover')
    ):
        return True

    return False

@property
def clickable(self) -> bool:
    """
    Используется сценой при инициализации группы спрайтов сцены.

    При наличии метода on_click,
    экземпляр автоматически заносится в список спрайтов,
    поддерживающих наведение курсором мыши.
    """

    if hasattr(self, 'on_click'):
        return True

    return False

#### METHODS ####

#### constructor ####
def __init__(
    self,
    scene: object,
    states: dict,
    state: str = None, # можно не указывать при инициализации
):
    """
    Параметр state можно не указывать при инициализации:
    подходит для спрайтов с одним состоянием
    или спрайтов, состояние которых определяется при вызове.
    """

    super().__init__()

    # сохранение ссылки на связанный экран

```

```

self.scene = scene

# замена путей изображений на сами изображения
# замена путей звуков, если они добавлены, на сами звуки
for key in states.keys():
    if type(states[key][0]) != Surface:
        states[key][0] = pygame.image.load(states[key][0])
    if len(states[key]) == 3:
        states[key][2] = Sound(states[key][2])
self.states = states

if not state:
    self.state = list(self.states.keys())[0]
else:
    self.state = state

# используется для сброса состояния при входе на сцену
self.dstate = self.state

# def on_click(self) -> None:
"""
Выполняется при нажатии на спрайт.
Вызывается в main.py для всех спрайтов текущего экрана,
с определённым методом.
"""

# def on_hover(self) -> None:
"""
Выполняется при наведении на спрайт,
с определённым методом.
"""

# def on_unhover(self) -> None:
"""
Выполняется при прекращении наведения на спрайт,
с определённым методом.
"""

def switch_state(
    self,
    state: str = None,
) -> None:
    """
    Вместо обычной установки состояния через свойство state,
    воспроизводит звук состояния, если установлен.

    Цикличная смена состояния,
    если не указан параметр state
    """

    # установка состояния по значению
    if state:
        self.state = state

    # цикличная смена состояния
    else:
        states_names = list(self.states.keys())
        cur_state_i = states_names.index(self.state)
        new_state_i = cur_state_i - 1

```



```

self.state = states_names[new_state_i]
self.current_state = states_names[new_state_i]

if len(self.states[self._state]) == 3:
    Channel(1).play(self.states[self._state][2])

def update_pos_size(self) -> None:
    self.state = self.state

def scene_enter(self):
    """
    Вызывается классом Scene при входе на сцену.

    Сбрасывает состояния спрайта в стандартное.
    """

    self.state = self.dstate

def scene_leave(self):
    """
    Вызывается классом Scene при выходе их сцены.
    """

```

					УП ТРПО 2-40 01 01.35.38.17.24 ПЗ	Лист
Изм.	Лист	№докум.	Подпись	Дата		43