«Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И.Ульянова (Ленина)»
(СПбГЭТУ «ЛЭТИ»)

Направление	09.03.04 – Программная инженерия
Профиль	Разработка программно-информационных систем
Факультет	КТИ
Кафедра	МО ЭВМ

К защите допустить

Зав. кафедрой А.А. Лисс

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

Тема: РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ ЛОКАЛЬНОЙ ОТЛАДКИ ЛАБОРАТОРНЫХ РАБОТ В СРЕДЕ MINIGRID

Студентка _____ Е.А. Виноградова
подпись

Руководитель _____ М.М. Заславский
(Уч. степень, уч. звание) *подпись*

Санкт-Петербург

2025

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

Утверждаю
Зав. кафедрой МО ЭВМ
_____ А.А. Лисс
«___»_____2025 г.

Студентка Е.А. Виноградова Группа 1303

Тема работы: Разработка приложения для локальной отладки лабораторных работ в среде minigrid

Место выполнения ВКР: СПбГЭТУ «ЛЭТИ» кафедра МОЭВМ

Исходные данные (технические требования):

Необходимо реализовать приложение для локальной отладки лабораторных работ в среде minigrid на языке программирования Python, используя фреймворк Flask.

Содержание ВКР:

Введение, Обзор предметной области, Формирование требований к реализации, Реализация решения, Исследование свойств разработанного решения, Заключение.

Перечень отчетных материалов: пояснительная записка, иллюстративный материал

Дополнительные разделы: безопасность жизнедеятельности

Дата выдачи задания
«___»_____2025 г.

Дата представления ВКР к защите
«___»_____20___ г.

Студентка _____ Е.А. Виноградова

Руководитель _____ М.М. Заславский
(Уч. степень, уч. звание)

КАЛЕНДАРНЫЙ ПЛАН ВЫПОЛНЕНИЯ
ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Утверждаю
Зав. кафедрой МО ЭВМ
_____ А.А. Лисс
«___»_____2025г.

Студентка Виноградова Е.А. Группа 1303

Тема работы: Разработка приложения для локальной отладки лабораторных работ в среде minigrid

№ п/п	Наименование работ	Срок выполнения
1	Обзор литературы по теме работы	02.04 – 04.04
2	Обзор предметной области	04.04 – 10.04
3	Формирование требований к решению	10.04 – 20.04
4	Реализация	20.04 – 25.04
5	Исследование реализованного инструмента	25.04 – 30.04
6	Безопасность жизнедеятельности	11.04 – 12.05
7	Оформление пояснительной записки	05.05 – 12.05
8	Оформление иллюстративного материала	
9	Предзащита	

Студентка _____ Е.А. Виноградова

Руководитель к.т.н., доцент _____ М.М. Заславский
(Уч. степень, уч. звание)

РЕФЕРАТ

Пояснительная записка 44 стр., 10 рис., 2 табл.

ПРИЛОЖЕНИЕ ДЛЯ ОТЛАДКИ, ВИЗУАЛИЗАЦИЯ, ЛОКАЛЬНЫЙ ОТЛАДЧИК.

Объектом исследования является отладочная система для решения задач по графическим алгоритмам.

Предметом исследования является процесс отладки студенческих решений задач с использованием графических библиотек.

Цель работы: разработка приложения для локальной отладки лабораторных работ в среде minigrid.

В ходе выполнения данной работы были проанализированы существующие решения для отладки кода в разных средах с использованием графических библиотек. Были выделены общие элементы отладки и сформулированы требования к новому решению по итогам анализа существующих.

Было реализовано приложение для локальной отладки в среде minigrid. Были сформулированы недостатки разработанного приложения и выделены направления для дальнейшей работы над проектом.

ABSTRACT

During the execution of this work, existing solutions for debugging code in different environments using graphical libraries were analyzed. The common elements of debugging were identified and requirements for a new solution were formulated based on an analysis of existing ones.

An application for local debugging in minigrid environment has been implemented. The shortcomings of the developed application were formulated and areas for further work on the project were identified.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

ЯП – язык программирования;

IDE – Integrated Development Environment;

Токен – существенная часть исходного кода, представленная в виде одного символа;

Токенизация – процесс преобразования исходного кода в последовательность токенов.

Минигрид –это упрощённая версия среды или платформы для обучения и тестирования алгоритмов, часто используемая в исследованиях в области искусственного интеллекта и обучения с подкреплением. Минигрид предоставляет пользователям набор простых задач, которые могут быть использованы для проверки алгоритмов в разных условиях

GPU – сокращение от Graphics Processing Unit, графический процессор, который используется для обработки и рисования изображений.

Фреймворк – это набор библиотек и инструментов, предоставляющий основу для разработки приложений. Он упрощает процесс создания программного обеспечения, предоставляя разработчикам готовую структуру и функциональность, которые можно использовать для построения более сложных решений. В контексте разработки веб-приложений, фреймворки, такие как Flask или Django, помогают организовать код и упростить работу с сервером, базами данных и клиентскими запросами.

ВВЕДЕНИЕ

Система образования, сформированная в предыдущем технологическом укладе, не соответствует потребностям современного общества.[1] На смену старым укладам пришло онлайн-образование. В современных условиях оно становится все доступнее, совершеннее и популярнее, однако еще довольно много направлений в котором оно может развиваться.[2]

При обучении студентов программированию, крайне важно, чтобы освоение инструментов было удобным, то есть не перегруженным количеством инструментов, но обладающими всеми необходимыми функциями, чтобы освоение новой технологии происходило, как можно легче, а отладочные системы были как можно нагляднее, то есть обладали большой степенью визуализации, чтобы можно было максимально подробно разобрать ошибки. Чем больше способов отладить свое решение и посмотреть на него с различных углов, тем эффективнее отладчик.

Поскольку сложность использования платформы напрямую влияет на скорость освоения материала, то обучающие платформы должны быть как можно удобнее и нагляднее.[3]

Целью данной работы является разработка приложения для локальной отладки лабораторных работ в среде minigrid.

Задачи данной работы:

- Проанализировать существующие платформы с отладочными системами для работы с графическими библиотеками;
- Сформулировать требования к реализации на основе анализа существующих решений;
- Реализовать приложение для локальной отладки лабораторных работ;
- Провести исследование производительности разработанного приложения.

Объектом исследования выступает отладочная система для решения задач по графическим алгоритмам.

Предметом исследования является процесс отладки для студенческих решений задач с использованием графических библиотек.

Практическая ценность работы: разработанное приложение позволит расширить возможности online обучения и поможет студентам как можно нагляднее разобраться в графических алгоритмах.

1 Обзор предметной области

1.1 Существующие решения

В приложении для локальной отладки лабораторных работ в среде MiniGrid, важно учесть, что аналогами будут такие решения, которые не только позволяют отлаживать код, но и визуализировать его выполнение, а также обеспечивают интерактивное взаимодействие с решением, где можно трассировать данные и поведение системы. Поиск аналогов производился с использованием ресурсов Google Chrome и GitHub. Аналоги искались по следующим запросам:

- GPU
- Отладочная система
- cross-platform game engine
- образовательная среда для обучения программированию
- моделирование роботов
- robot simulation

1.1.1 Godot Engine

Многофункциональный кроссплатформенный игровой движок для создания 2D- и 3D-игр с помощью единого интерфейса.[4]

1.1.2 Кулибин

Это приложение представляет собой виртуальную 3D-среду для изучения робототехники через программирование цифровых роботов. Оно позволяет пользователям программировать цифрового робота, используя язык на основе блоков, приобретая навыки в области робототехники и алгоритмического программирования.[5]

1.1.3 Blender (Python Debugging)

В Blender можно не только моделировать и анимировать 3D-объекты, но и использовать Python для написания кода. Встроенные инструменты отладки позволяют отслеживать выполнение кода и изменять параметры в

реальном времени. Особенность: Интеграция графики и программирования с возможностью манипулировать объектами сцены.[6]

1.1.4 Webots

Данная платформа для моделирования и симуляции робототехнических систем, которая поддерживает создание 3D-моделей роботов и позволяет отслеживать их выполнение в реальном времени с визуализацией. Веб-интерфейс позволяет вручную управлять параметрами и тестировать решения. Особенность: Визуальная трассировка выполнения алгоритмов с возможностью редактировать и управлять объектами в реальном времени.[7]

1.1.5 CoppeliaSim (V-REP)

Этот симулятор для разработки и тестирования робототехнических систем. Он поддерживает визуализацию и интерактивное управление роботами, а также позволяет отлаживать решения через графический интерфейс. Особенность: Поддержка визуализации и возможности вмешательства в выполнение симуляций, включая трассировку поведения роботов.[8]

1.2 Критерии сравнения

1.2.1 Качество отладки кода

В данном случае качество измеряется количественно, чем больше инструментов для отладки кода, тем выше качество отладки. Считается, что качество считается высоким, если 5 основных элементов реализованы в полном объеме:

1.2.1.1 Отладчик

Встроенный отладчик — это мощный инструмент для разработчиков, который облегчает процесс выявления и исправления ошибок. Установка точек остановок позволяет приостановить выполнение программы в заданный момент, что помогает анализировать состояние приложения. Благодаря этому разработчик может понять, на каком этапе возникает проблема, и своевременно заняться ее решением.

Отладчик также позволяет отслеживать значения переменных в реальном времени. Это особенно важно, когда необходимо понять, как изменяются данные в ходе выполнения программы. С помощью отладчика можно легко увидеть, какие значения принимают переменные на различных этапах и выявить возможные несоответствия.

Кроме того, отладчик обеспечивает пошаговое выполнение кода. Это означает, что разработчик может проходить через каждую строку кода, анализируя ее влияние на программу. Пошаговая отладка помогает разобраться в логике выполнения и устранять ошибки на ранних стадиях.

1.2.1.2 Консоль

Консоль — это важный инструмент для вывода сообщений и ошибок в реальном времени. Она служит интерфейсом между разработчиком и программой, позволяя получать информацию о процессе исполнения кода. С помощью консоли можно выявлять ошибки и отслеживать процесс выполнения, что значительно ускоряет отладку.

Кроме вывода данных, консоль также позволяет взаимодействовать с Кодом через введение команд. Это может быть очень полезно для тестирования отдельных функций или методов. Разработчики могут выполнять команды на лету, что делает процесс разработки более интерактивным и динамичным.

Консоль также предоставляет возможность видеть различные сообщения о состоянии приложения. Например, разработчики могут выводить предупреждения или информацию о выполнении определённых операций, что помогает корректно понимать логику приложения и упрощает анализ его работы.

1.2.1.3 Логирование

Система логирования — это ключевой элемент для мониторинга и анализа работы приложения. Она позволяет записывать события и ошибки, происходящие на протяжении работы программы. Эта информация может быть использована для последующего анализа и выявления узких мест в коде.

Логирование помогает не только в обнаружении ошибок, но и в понимании поведения приложения. Благодаря записям о событиях разработчики могут отслеживать, какие действия приводят к определённым результатам. Это важный аспект, который позволяет разработать код, который обладал бы более высоким качеством и улучшить производительность.

Кроме того, система логирования может быть настроена таким образом, чтобы фиксировать разные уровни важности сообщений. Например, разработчики могут записывать только критические ошибки или наоборот, отслеживать все события, включая отладочные сообщения. Это гибкость помогает адаптировать систему логирования под конкретные требования проекта.

1.2.1.4 Монитор переменных

Монитор переменных — это инструмент, который позволяет разработчикам видеть текущее состояние переменных во время выполнения

игры. Он обеспечивает наглядность данных, что делает анализ более эффективным. Особенно это важно в играх, где состояние переменных может меняться в короткие сроки.

Данный инструмент помогает выявить проблемы, связанные с неправильным обновлением переменных. Если разработчик замечает, что переменные не принимают ожидаемые значения, он может сразу внести изменения и протестировать их. Это экономит время и позволяет избежать долгих поисков ошибок в коде.

Также монитор переменных может быть полезен для улучшения качества игрового процесса. Зная текущее состояние различных переменных, разработчики могут принимать более обоснованные решения о том, как изменять логику игры. Это способствует созданию более качественного и интуитивно понятного игрового опыта.

1.2.1.5 Инспектор объектов

Инспектор объектов — это инструмент, который предоставляет возможность исследовать значения объектов и их свойства во время выполнения. Это особенно важно для работы с сложными структурами данных, где множество свойств и методов влияют на общее поведение приложения. Инспектор позволяет разработчикам глубже понять состояние объектов в любой момент времени.

Благодаря инспектору разработчики могут находить несоответствия и ошибки в данных объектов. Если какое-то свойство объекта ведет себя неправильно, инспектор позволяет своевременно определить, какое именно из свойств не соответствует ожиданиям. Это значительно облегчает процесс отладки и улучшает качество приложения.

Кроме того, инспектор объектов позволяет взаимодействовать с данными непосредственно на этапе выполнения. Разработчики могут изменять значения свойств объектов и сразу видеть результат этих

изменений в приложении. Это позволяет тестировать новые идеи и улучшать код в процессе разработки.

1.2.2 Уровень интерактивности

Уровень интерактивности приложения для отладки играет ключевую роль в процессе разработки и тестирования кода. Приложение должно не только предоставлять возможности для исправления ошибок, но и включать инструменты, которые позволяют разработчикам активнее взаимодействовать с программой. Высокий уровень интерактивности достигается через интеграцию функций, таких как взаимодействие с визуализацией данных, отслеживание работы кода и динамическое отображение результатов.

Чтобы создать действительно интерактивное приложение, необходимо включить три основных инструмента — взаимодействие, трассировку и динамическую визуализацию. Эти инструменты не только помогают разработчикам понимать, как работает их код, но и позволяют вовремя корректировать ошибки и повысить функциональность приложения в ходе выполнения. В результате, разработчики могут принимать более информированные решения и быстрее реагировать на возникающие проблемы.

Таким образом, уровень интерактивности приложения становится определяющим фактором его эффективности. Чем больше возможностей для взаимодействия у разработчиков, тем быстрее они могут достигнуть качества и надежности кода. Интерактивные инструменты делают процесс разработки более и понятным, что в итоге приводит к более качественному программному обеспечению.

1.2.2.1 Взаимодействие

Возможность взаимодействия — это один из важнейших аспектов современного инструментария для отладки. Эта функция предоставляет разработчикам возможность не только наблюдать за выполнением кода, но и

активно участвовать в нем. Например, разработчики могут менять параметры в реальном времени, видеть, как эти изменения влияют на ход исполнения программы и результат её работы. Это создает более динамичную атмосферу работы и позволяет выявлять ошибки быстрее.

Кроме того, взаимодействие с кодом позволяет проводить эксперименты, которые могут создать инновационные решения. Разработчики могут тестировать различные варианты алгоритмов и способа обработки данных без необходимости многократной компиляции и тестирования всей программы. Такой подход дает возможность своевременно проверять гипотезы и находить верные решения, делая процесс разработки более гибким и продуктивным.

Наличие функции взаимодействия также улучшает обучение новых разработчиков. Они могут экспериментировать с реальным кодом и наблюдать за его работой в режиме реального времени. Это облегчает понимание принципов работы программного обеспечения и способствует усвоению необходимых навыков. Интерактивная среда создает экосистему, в которой эксперименты и обучение идут рука об руку, повышая общую квалификацию команды.

1.2.2.2 Трассировка

Трассировка — это мощный инструмент, позволяющий отслеживать выполнение программного кода на каждом этапе. Эта функция дает возможность разработчикам наблюдать выполнение программы в реальном времени, что облегчает процесс выявления ошибок и недочетов. Каждый шаг исполнения может быть проанализирован, что позволяет глубже вникнуть в логику работы кода и понять, как каждая строка влияет на общее состояние приложения.

Трассировка особенно полезна при работе с большим объемом данных или сложными алгоритмами. Иногда ошибка может возникать из-за неверных данных или логики, и видеть каждый шаг выполнения программы помогает сузить поиск до конкретных участков кода. Это также позволяет

ловить моменты, когда состояние переменных и объектов изменяется неожиданным образом, что может привести к неправильному поведению приложения.

Кроме того, трассировка способствует глубокому анализу производительности программы. Разработчики могут выявлять узкие места в коде и ускорять выполнение тяжёлых операций, на которые уходит больше времени. Это существенно повышает качество итогового продукта и позволяет создавать более эффективные решения. Качественная трассировка — это залог стабильности и надежности приложения, ведь она помогает заранее выявлять проблемы и ограничивать их влияние на пользователей.

1.2.2.3 Динамическая визуализация.

Динамическая визуализация представляет собой инструмент, который позволяет отображать результаты выполнения программ в реальном времени. Эта функция предоставляет пользователям кода четкое и наглядное представление о том, как алгоритмы и данные взаимодействуют друг с другом. Динамическая визуализация может значительно повысить понимание сложного кода и структуры данных, что делает её незаменимой во время разработки.

Использование динамической визуализации помогает разработчикам и тестировщикам видеть, как изменения параметров или функций влияют на результаты симуляции. Это может быть особенно важно в сложных приложениях, где множество факторов влияет на конечный результат. Наличие визуальных представлений данных также облегчает коммуникацию между членами команды, так как сложные концепции можно объяснить с помощью графиков и диаграмм.

Визуализация в режиме онлайн освобождает разработчиков от необходимости вручную обрабатывать данные после каждой итерации кода и предоставляет мгновенную обратную связь. Такой подход позволяет проводить анализ и находить верные решения за короткое время. Динамическая визуализация делает программирование более доступным и

понятным для широкой аудитории, включая начинающих разработчиков, студентов и образовательные учреждения, желающие внедрить современные подходы к обучению программированию.

1.2.3 Удобство использования

Удобство использования приложения для отладки является важным аспектом, который влияет на эффективность и производительность разработчиков. Один из критериев удобства — это визуальное разделение инструментов. Если все инструменты взаимодействия с кодом и визуальной частью находятся в одном окне, это может привести к путанице и замешательству. Грамотное оформление интерфейса, где инструменты отладки кода сосредоточены рядом с окном самого кода, а визуализация представлена в отдельном окне, помогает пользователю четко понимать, где и как они взаимодействуют с программой.

Кроме того, такое разделение способствует сосредоточению внимания на конкретных аспектах разработки. Разработчики могут легче переключаться между задачами, не напрягаясь в поисках нужных инструментов. Это тем более важно в сложных проектах, где необходима быстрая реакция на ошибки, и разработчики должны иметь возможность работать одновременно с кодом и визуализацией. Структурный подход к организации инструментов улучшает опыт пользователя, создавая более комфортную среду для работы.

Таким образом, удобство использования приложения — это не просто вопрос эстетики, но и важный фактор повышения производительности. Правильное визуальное разделение инструментов позволяет сократить время на решение задач, что, в свою очередь, ускоряет процесс разработки. Удобный интерфейс способствует более эффективной работе команды и минимизирует количество ошибок, возникающих из-за несоответствия в организации пространства для взаимодействия с инструментами.

1.2.3.1 Визуальное разделение инструментов

Визуальное разделение инструментов является не мало важным аспектом удобства использования программного обеспечения для разработки. Когда инструменты отладки и все сопутствующие функции размещены в одном окне, пользователю трудно сконцентрироваться на конкретной задаче. Это может привести к путанице и потере времени, особенно в проектах, требующих высокой концентрации и своевременной реакции. Эффективное разделение инструментов помогает разработчикам четко понимать свои действия и поддерживать порядок во время работы.

Кроме того, структурированное визуальное разделение улучшает навигацию по интерфейсу. Разработчики могут своевременно находить необходимые инструменты и переключаться между ними, не тратя время на поиск нужных панелей или меню. Это обеспечивает более рациональное использование рабочего пространства. Разделение элементов интерфейса по категориям позволяет сосредоточиться на конкретных аспектах разработки, будь то отладка, визуализация или работа с базой данных, и тем самым повышает общую продуктивность.

Наконец, визуальное разделение инструментов способствует улучшению эргономики рабочего процесса. Когда разработчики имеют четкое представление о том, где находятся различные инструменты, они могут организовывать свою работу более эффективно. Это также позволяет снизить уровень стресса и повысить удовлетворенность от работы. Качественно структурированный интерфейс, где разные инструменты находятся в своих логичных местах, создает более приятные условия для работы и помогает разработчикам достигать наиболее прогрессивных результатов.

1.2.3.2 Настраиваемость

Настраиваемость интерфейса и функциональности приложения под индивидуальные потребности пользователя играет важную роль в удобстве работы. Возможность настройки фона, размера шрифта, цветовой схемы и

других параметров интерфейса позволяет разработчикам адаптировать рабочую среду под свои предпочтения. Это обеспечивает комфортные условия для работы, что особенно критично при длительных сессиях программирования, когда каждый аспект интерфейса может влиять на уровень усталости и продуктивности.

Кроме того, настраиваемость может быть полезной для различных категорий пользователей. Например, начинающие разработчики могут предпочитать более простую и понятную среду, тогда как опытные специалисты могут настраивать интерфейс под более сложные задачи и подходы. Такой индивидуальный подход обеспечивает повышенный уровень удбства для пользователя и комфортную производительность работы, поскольку каждому предоставляется возможность создать пространство, в котором ему удобнее и продуктивнее работать.

Важно отметить, что настойка интерфейса должна быть интуитивно понятной и доступной. Простые и понятные меню, а также возможность вернуться к стандартным настройкам облегчают этот процесс. Настраиваемость интерфейса — это масса возможностей, которые помогают разработчикам создать свою комфортную среду для работы, способствуя улучшению качества кода и всего процесса разработки в целом.

1.2.3.3 Подсветка кода

Подсветка кода является важной функциональностью, значительно улучшающей восприятие исходного текста программ. Она выделяет синтаксические конструкции, такие как ключевые слова, комментарии, строковые и числовые данные, что делает код более читаемым и удобным для анализа. Четкая визуализация разных элементов кода помогает разработчикам быстрее раскодировать структуру программы, что особенно актуально в больших проектах с множеством строк кода.

Эта функция также играет важную роль в предотвращении ошибок. Подсветка синтаксиса позволяет мгновенно выявлять несоответствия и синтаксические ошибки, что значительно упрощает процесс отладки.

Разработчики могут видеть, где конкретно произошла ошибка, сосредоточив внимание на проблемных участках. Это также позволяет новичкам быстрее освоить синтаксис языка, облегчая процесс обучения и повышения профессиональных навыков.

Кроме того, подсветка кода может быть настроена под индивидуальные предпочтения пользователя. Возможность изменения цветовой схемы и стиля подсветки позволяет каждому разработчику создать более комфортные условия для работы. Качественно реализованная подсветка кода помогает обеспечивает положительный общий опыт взаимодействия с приложением, делая процесс программирования более приятным и продуктивным.

1.2.4 Порог вхождения

Этот критерий прямо пропорционален остальным трём. Это обусловлено тем, что большое количество инструментов для отладки кода обеспечивает высокое качество отладки, но создаёт визуальную перегруженность среды. Чем меньше инструментов, тем меньше времени затрачивает человек на освоение, тем быстрее начнётся непосредственное обучение новым технологиям.

1.3 Выводы по итогам сравнения

Для удобства результаты анализа сведены в таблице 1.

Таблица 1 – Сравнение аналогов по 5 критериям.

Аналог	Качество отладки кода	Уровень интерактивности	Удобство использования	Порог вхождения
Godot Engine	5/5 Отладчик реализован в полном объеме, консоль для работы с кодом и отображения выводов, логирование обеспечивает анализ ошибок, есть возможность отслеживать переменные в реальном времени, для инспектора объектов имеются все необходимые инструменты.	3/3 Высокий уровень взаимодействия, трассировка реализована в полном объеме, возможность вмешиваться, реализованы инструменты для динамической визуализации.	3/3 Интерфейс обладает большим количеством настроек, чтобы каждый пользователь мог настроить среду под себя, подсветка поддерживает многие типы синтаксиса и данных, инструменты управления отладкой и визуализацией четко разделены.	Высокий порог вхождения обуславливается высокими показателями остальных критериев.
Кулибин	0/5 Минимально реализованна обработка ошибкок, остальные	1/3 Низкий уровень взаимодействия, возможность	1/3 Визуальное разделение инструментов реализовано четко, нет возможности	Низкий порог вхождения обуславливается минимальным

	параметры отсутствуют.	вмешиваться есть, но минимальная, реализованы инструменты для динамической визуализации.	настроить под себя интерфейс, подсветка минимальная.	количеством инструментов для взаимодействия и отладки кода.
Blender (Python Debugging)	5/5 Отладчик реализован в полном объеме, консоль для работы с кодом и отображения выводов, логирование обеспечивает анализ ошибок, есть возможность отслеживать переменные в реальном времени, для инспектора объектов имеются все необходимые инструменты.	3/3 Высокий уровень взаимодействия, возможность вмешиваться, реализованы инструменты для динамической визуализации.	3/3 Интерфейс обладает большим количеством настроек, чтобы каждый пользователь мог настроить среду под себя, подсветка поддерживает многие типы синтаксиса и данных, инструменты управления отладкой и визуализацией четко разделены.	Высокий порог вхождения обуславливается высокими показателями остальных критериев.
Webotse	4/5 Отладчик реализован в полном объеме, консоль для работы	2/3 Возможность вносить изменения в	2/3 Интерфейс обладает необходимым количеством	Средний порог вхождения обеспечивается меньшим

	с кодом и отображения выводов, логирование обеспечивает анализ ошибок, есть возможность отслеживать переменные в реальном времени, инспектор объектов реализован не в полном объеме по сравнению с Blender.	параметры во время выполнения, трассировка показывает только основные шаги без деталей, реализованы инструменты для динамической визуализации.	настроек, однако их количество значительно меньше, чем у Blender, подсветка поддерживает многие типы синтаксиса и данных, инструменты управления отладкой и визуализацией четко разделены.	количеством инструментов, чем у Blender.
Coppelia Sim (V-REP)	5/5 Отладчик реализован в полном объеме, консоль для работы с кодом и отображения выводов, логирование обеспечивает анализ ошибок, поддержка мониторинга в реальном времени, для инспектора	3/3 Высокий уровень взаимодействия, возможность вмешиваться, полная поддержка трассировки, реализованы инструменты для динамической визуализации.	3/3 Интерфейс обладает большим количеством настроек, чтобы каждый пользователь мог настроить среду под себя, подсветка поддерживает многие типы синтаксиса и данных, инструменты управления	Высокий порог вхождения обуславливаетс я высокими показателями остальных критериев.

	объектов имеются все необходимые инструменты.		отладкой и визуализацией четко разделены.	

После анализа различных платформ, предлагающих инструменты для отладки и визуализации в различных средах можно обнаружить одинаковые элементы, которые присутствуют во всех локальных отладчиках для своих конкретных библиотек и задач. А значит для решения поставленной задачи нужно реализовать отладчик, который обладал бы самыми распространёнными и основными инструментами, а именно:

- Отладчик и возможность поставить точки остановок
- Подсветка кода
- Трассировка
- Визуальное разделение кода

Помимо этих элементов приложение должно обладать низким порогом вхождения, так как пользоваться им будут студенты.

2 Формулировка требований к реализации инструмента

2.1 Постановка задачи

Необходимо реализовать приложение для локальной отладки кода в среде minigrid на ЯП Python, используя фреймворк Flask.

2.2 Требования к реализации

Данное приложение должно быть реализовано с использованием современного фреймворка для веб-приложения на языке Python — Flask. Поскольку этот фреймворк нужен для создания не перегруженных функционалом веб-приложений на Python, а значит подойдет для приложения для локальной отладки лабораторных работ. Flask предоставляет множество инструментов и библиотек для построения серверной части, а значит обеспечит возможность полноценной разработки.

Для предоставления качественного и доступного обучения от приложения требуется обеспечить работоспособность на большинстве устройств, в силу чего нужно реализовать отладчик в рамках архитектуры веб-приложения, потому как формат веб-прилоения позволяет использовать функционал на устройствах практически любого типа и мощности.

При реализации необходимо учесть угрозу потенциальной вредоносности кода, отправляемого пользователями в качестве решения задач, для чего неообходимо обеспечить исполнение кода студентов в изолированной среде.

Также нужно реализовать базовые инструменты для отладки и визуализации.

При реализации следует учитывать, что существуют два типа задач: со статической и динамической картами, что влечет необходимость по-разному обрабатывать код студента и выполнять проверку правильности решения для каждого из вышеуказанных типов задач.

2.3 Выбор метода решения

Исходя из поставленных требований и анализа аналогов, было принято решение писать отладчик с нуля, поскольку существующие отладчики реализованы в конкретных графических средах и для данной задачи нужен отладчик, который взаимодействовал бы со средой minigrid. Также из рассмотренных отладчиков с графическими средами были выделены основные элементы для отладки, которые так или иначе присутствующие во всех аналогах. Следовательно реализация этих элементов является необходимой, а именно:

1. Визуальное разделение на логические составляющие, то есть разделение экрана на часть с отладкой кода и часть с визуализацией;
2. Подсветка кода;
3. Вывод ошибок;
4. Трассировка и возможность отслеживать переменные;
5. Ставить брейк поинты;

3 Архитектура программной реализации

3.1 Описание приложения для отладки

Приложение для отладки – это набор инструментов, разработанный с целью локальной отладки студенческого кода в графической среде. Приложение содержит в себе элементы отладки, а так же инструменты для визуализации решения и взаимодействие с ним. Архитектура приложения для отладки кода показана на рис. 1.

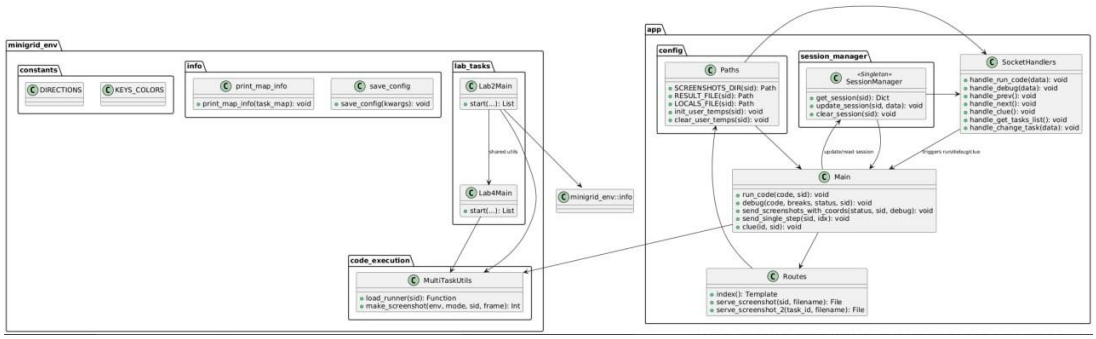

Рисунок 1 – Архитектура программного модуля

Согласно данной схеме архитектуре, данное приложение состоит из двух основных модулей minigrid_env и app. Minigrid_env состоит из задач, условий, генераций карт, как статических так и динамических и проверок студенческих решений на правильность. В этом модуле также создается папка temp для временных изолированных сессий, а так же создание скриншотов для трассировки. В модуле app содержится приложение для отладки лабораторных работ, который напрямую взаимодействует с задачами и их компонентами.

3.2 Входные и выходные данные

На вход принимается код пользователя, обрабатывается и на выходе получается результат исполнения пользовательского кода с набором промежуточных состояний во время исполнения. Набор промежуточных состояний во время исполнения в случае запуска без отладки представляет собой набор скриншотов соответствующих каждому шагу агента.

В случае отладки набор промежуточных состояний представлен в виде скриншотов соответствующих каждому шагу агента вместе с набором пользовательских переменных соответственно.

3.3 Сценарии использования

Основным пользователем данного приложения является студент. Основной сценарий использования выглядит следующим образом:

1. Пользователь открывает сайт и выбирает задачу для решения.
2. Внимательно читает условие и пишет код для решения задачи.
3. Запускает код на кнопку run и смотрит на результат.

Далее происходит один из двух сценариев:

1) Код студента верно решил задачу с первого раза и теперь он переходит решению следующей задачи;
2) Код студента имеет неточности, ошибки или его решение выходит за рамки выделенной памяти под эту задачу и в этом случае понадобятся инструменты для отладки.

4. Отладив свой код студент переходит к решению следующей задачи.

Отладка студенческого кода происходит следующим образом:

1. В первую очередь исправляются синтаксические ошибки, они выводятся в консоли при нажатии на кнопку run.
2. После того как код написан корректно, нажав на кнопку run в соседнем окне отображается визуализация этого кода, которая может выполнить текущую задачу и студент перейдёт к решению следующей или студент заметит, что его решение не соответствует необходимому для решения данной задачи и перейдёт к следующему шагу отладки.
3. Для отладки в полном объёме студент нажимает на конпку debug, далее у него появляется возможность трассировать свое решение, то есть прогнать визуализацию по шагам и понять в какой момент его решение работает некорректно.

4. Нажав на кнопку debug, можно увидеть как выглядит изначальная карта, а так же получить информацию о том правильный ли путь нашёл студент.

5. Для полного понимания происходящего в коде студенту необходимо установить точки остановки в тех строках, где, по его мнению, может быть ошибка. Затем следует нажать кнопку debug, что даст возможность не только трассировать шаги, но и наблюдать изменения переменных в точках остановки на каждом этапе. Также есть возможность установить несколько таких точек и отслеживать изменения переменных во всех нужных строках.

6. В случае, если студенту нужна помощь в решении какой либо задачи, ему следует нажать на кнопку clue и это даст ему возможность увидеть правильное решение и результат к которому следует стремиться.

7. После успешного решения задачи студент переходит к следующей.

3.4 Используемые технологии

3.4.1 Фреймворк Flask

Фреймворк Flask классифицируется как один из микрофреймворков — это минималистичный каркас, который предлагает только основные инструменты для выполнения различных задач. В Flask маршрутизация играет ключевую роль, связывая функции с определёнными URL-адресами. Это позволяет приложению реагировать на различные запросы, например, выводить разные страницы для разных адресов.

Представления в Flask выглядят как функции на языке Python, которые обрабатывают входящие запросы и формируют соответствующие ответы. Обычно результатом работы такой функции является HTML-страница, однако возможными вариантами ответа также могут быть текстовые данные, JSON или XML. Таким образом, Flask обеспечивает простоту в обработке запросов и формировании ответов.

Использование Flask в данном приложении для отладки играет ключевую роль, так как позволяет определять маршруты (routes.py) с помощью специальных декораторов, что делает приложение доступным через веб.

Обработка HTTP-запросов — Flask автоматически обрабатывает входящие HTTP-запросы и передает их в соответствующие функции обработчики. Например, функция index обрабатывает запрос на главную страницу, а функции serve_screenshot и serve_screenshot_2 отвечают за отображения шагов визуализации для задач.

Шаблонизация во Flask поддерживает рендеринг HTML-шаблонов через функцию render_template, что позволяет отделить логику от представления. Шаблон index.html используется для отображения данных, полученных из конфигурации задач.

Работа с файлами обеспечивает функция send_from_directory, данная функция позволяет безопасно и удобно отправлять файлы пользователю, что полезно при работе с изображениями, которые используются в качестве визуализации его кода.

Таким образом, использование Flask в данном приложении делает код структурированным, а веб- разработку на Python комфортной.

3.4.2 Docker

Docker в данном приложении выполняет несколько ключевых ролей, обеспечивая изоляцию, управление зависимостями и безопасность. Вот основные функции Docker, используемые в данном приложении:

- Изоляция исполнения кода пользователя - ответ студента на задачу в контексте лабораторной работы, представляющий собой код, описывающий решение задачи, запускается в изолированном Docker контейнере, что обеспечивает безопасное и контролируемое окружение для выполнения потенциально

небезопасного кода. Это защищает основную систему от возможных вредоносных действий.

- Так же использование контейнеризации служит не только для изолирования отдельных компонентов, но и всего приложения в целом, что позволяет упаковывать все зависимости, необходимые для выполнения приложения (в данном случае Python и его библиотеки), что исключает необходимость установки этих зависимостей на основной системе, где будет развернуто данное приложение
- Конфигурация окружения позволяет капускать код через Docker-контейнер с определенными параметрами, такими как ограничение памяти и ресурсов процессора, а также определение рабочего каталога. Это позволяет разработчику легко настраивать ресурсы, выделяемые для выполнения кода, что не мало важно, поскольку студентов может много и искусственные ограничения необходимы, чтобы избежать перегрузки мощностей сервера.
- Логирование вывода - Выполняя код внутри контейнера, приложение может перенаправлять вывод консоли обратно в сокет, позволяя пользователю видеть результаты выполнения кода в реальном времени.
- В коде используется привязка внешних каталогов к контейнеру (-v), что позволяет контейнеру получать доступ к файлам, необходимым для выполнения кода, сохраняя при этом контроль над их содержимым и безопасностью.

3.4.3 Socket-запросы

Взаимодействие серверной части(Backend) и браузерной(Frontend) реализовано с использованием сокетов, в частности посредством использования flask.socketio.

Взаимодействие через сокет, в отличие от более распространенного взаимодействия через HTTP(S)-протокол, который чаще всего в свою очередь базируется на UDP-протоколе, позволяет избежать затрат на так называемый TLS-handshake и полный обмен заголовками, инициируемый при очередном “контакте” сервера и клиента. За счет отсутствия вышеописанных затрат на установку соединения, значительно возрастает скорость взаимодействия между сторонами, фактически ограниченная физическими параметрами и мощностью сервера.

Использование socketio, помимо преимуществ сокетов, предоставляет возможность относительно простого масштабирования системы, а также built-in разделение взаимодействия на разные изолированные друг от друга подпространства, называемые в терминах фреймворка “комнатами”.

3.4.4 Графическая библиотека Minigrid

Для получения данных о карте, позиции агента и других компонентах окружения используется minigrid. Полученные с помощью библиотеки данные группируются в ассоциативный массив, на основании которого с использованием библиотеки PIL происходит генерация изображения кадра, отражающего актуальное состояние работы программы, соответствующее моменту исполнения кода, для которого сделано изображение.

3.5 Структура программной реализации

3.5.1 Кнопка запуска и отладки

После того как пользователь нажимает кнопку run или debug, Frontend приложения собирает данные, необходимые для запуска и/или отладки решения студента, а именно, написанный студентом код в текстовом поле для ввода кода, а также другие данные, необходимые для корректной обработки запроса, такие как идентификатор сессии или набор точек останова, установленных пользователем с целью трассировки. Из собранных данных формируется и отправляется запрос на Backend посредством socket’a.

Для обработки запросов на исполнение кода(run) и отладки кода(debug) на Backend в файле socket_handlers предусмотрены следующие обработчики событий, соответственно, handle_run_code и handle_debug, каждый из которых контролирует соответствование требованиям и обрабатывает полученные из запроса с фронтенда данные и запускает процесс исполнения или отладки, соответственно, по завершении которых отправляет результат операции обратно на Frontend.

3.5.2 Кнопка подсказки

При нажатии на кнопку “Clue” студенту сначала показывается окно с двумя кнопками выбора: использовать подсказку или отказаться от нее и попытаться найти решение самостоятельно. В случае нажатия на кнопку отказа студент возвращается к самостоятельному решению задачи. При выборе получения подсказки на Backend отправляется запрос о том, что была запрошена подсказка для решения. Запрос принимается обработчиком handle_clue, расположенном в файле socket_handlers.py, после чего определяется задача, для которой была запрошена подсказка. Определив текущую задачу, обработчик достает из соответствующего раздела хранилища данные подсказки, определенные разработчиками задачи, и посылает их обратно на Frontend. Frontend, получив ответ на запрос данных подсказки, отображает результат пользователю.

3.5.3 Точки остановки и трассировка

Для более глубокой отладки кода пользователю предоставляется возможность установить точки останова в своем коде.

Как было описано выше (см. раздел 3.5.1 “Кнопка запуска и отладки”), при нажатии debug для отправки запроса на сервер Frontend в числе прочих данных собирает информацию об установленных пользователем точках останова для строк его кода. Обработчик debug, получив список точек останова, если в нем присутствуют точки остановки

(точки остановки установленные в пустые строки игнорируются), дополнительно производит обработку кода студента таким образом, что в стоках его кода, для которых присутствуют точки останова, производится “скрин” состояния его локальных и глобальных переменных, доступных в конкретной строке. Информация, полученная в результате “скрина”, вместе с другими данными о результатах отладки и исполнения сохраняется в виде списка отображений “номер строки” -> “состояние” в изолированную директорию, соответствующую идентификатору сессии пользователя.

В процессе отладки решения при наличии точек останова пользователю, помимо местоположения агента на карте, выводится информация о переменных в соответствующих строках кода.

3.6 Модель данных

При реализации веб-приложениябыл произведен отказ от использования традиционных СУБД в пользу гибридной модели, представляющей собой in-memory/cold-storage архитектуру, в которой активные сессии пользователей хранятся в оперативной памяти, в то время как результаты исполнения решений пользователя и информация о задачах сериализуются на уровне файловой системы.

3.6.1 Технология хранения сессий пользователей

За хранение активных сессий пользователей отвечает session_manager, хранящий уникальные объекты, каждому из которых соответствует SID.

Для каждого SID сохраняются:

Path — массив координат пути, который возвращает студенческое решение.

Screenshots — список путей к сгенерированным PNG-файлам.

Current_step — индекс текущего шага для навигации.

Local — массив снимков локальных переменных для отладки.

Моделью данных в этом случае выступает одиночный JSON-файл, атомарно перезаписываемый при каждом обновлениипутем блокировки на уровне threding.lock и полного вызова json.dump. Это позволяет создавать ACID-подобную транзактность для небольших объемов данных.

Благодаря данному методу хранения достигается:

Быстрый доступ к сессионным данным пользователя без затрат на операции чтения и записи, связанные со взаимодействием с данными на диске;

За счет использования механизмов блокировки таких как treading.lock обеспечивается атомарность обновления данных в условия наличия нескольких конкурирующих потоков.

3.6.2 Технология хранения данных о задачах и решения пользователя

Временная директория temp хранит промежуточные данные. Каждая сессия изолирована в собственной директории, что значительно упрощает управление правами доступа

На уровне файловой системы хранятся данные о задачах и артефакты исполнения и отладки кода пользователей в пространствах minigrid_env/lab_tasks и minigrid_env/temp, соответственно.

В пространстве, где хранятся данные о задачах, для каждой задачи выделена отдельная директория с именем, соответствующим идентификатору данной задачи. Для задачи предусмотрено наличие кода, генерирующего карту, отвечающую условиям из описания данной задачи, а также кода, способного верифицировать правильность решения данной задачи, кроме того, предполагается наличие данных для получения подсказки к решению задачи.

На данных момент после добавления данных для задачи не предполагается их модификация в процессе работы приложения, что может быть изменено в процессе развития приложения.

В пространстве с пользовательскими данными данные каждой сессии изолированы в директории с названием, соответствующим идентификатору этой сессии. В отличие от директорий задач, предполагается, что в сессионных директориях данные будут меняться в процессе работы приложения.

Для каждой сессии в соответствующей ей директории выделяется дочерняя директория img, используемая как BLOB-хранилище для кадров с промежуточными состояниями передвижения агента по карте. Помимо этого присутствуют файлы temp_code.py с исполняемымы кодом, содержащим решение студента, locals.json, куда сериализуется информация о переменных, в моменты исполнения кода, соответствующие установленным точкам останова, result.json, куда сериализуется непосредственно результат работы кода пользователя.

Использование изолированных директорий для каждой сессии позволяет обеспечить простоту администрирования данных, атомарность операций в условиях конкурентности при работе с файловой системой.

3.7 Тестирование отладчика

Примеры работы приложения для отладки приведены на рис. 2, рис. 3, рис. 4, рис. 5, рис. 6.

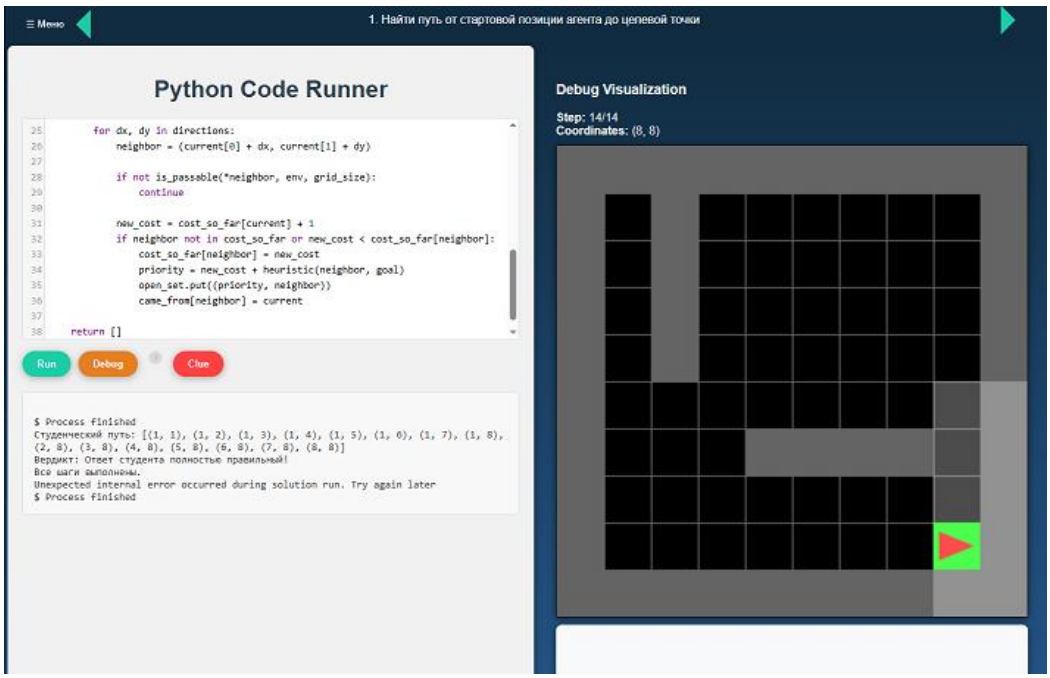

Рисунок 2 – Задача 1 пример запуска на статической карте.

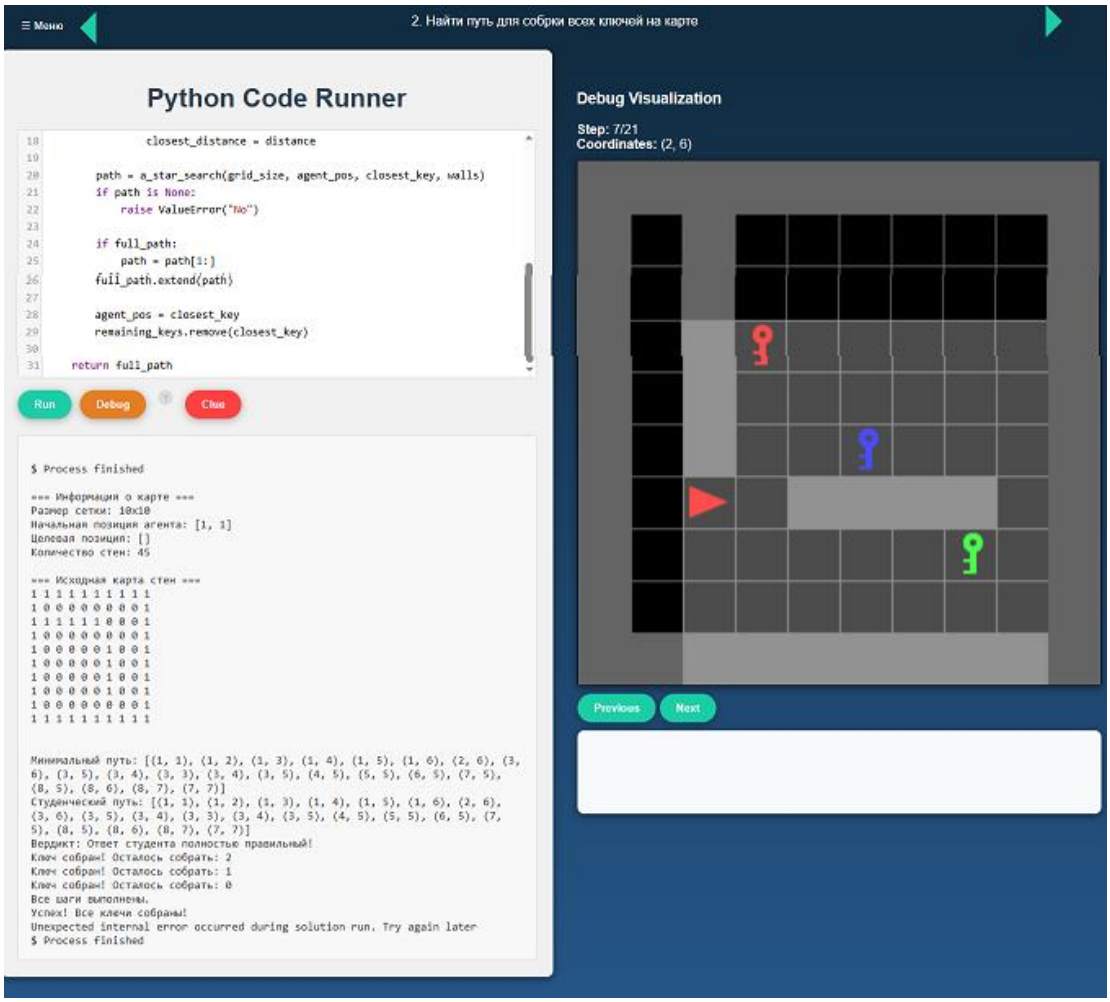

Рисунок 3 – Задача 2 пример запуска на кнопку debug и трассировки.

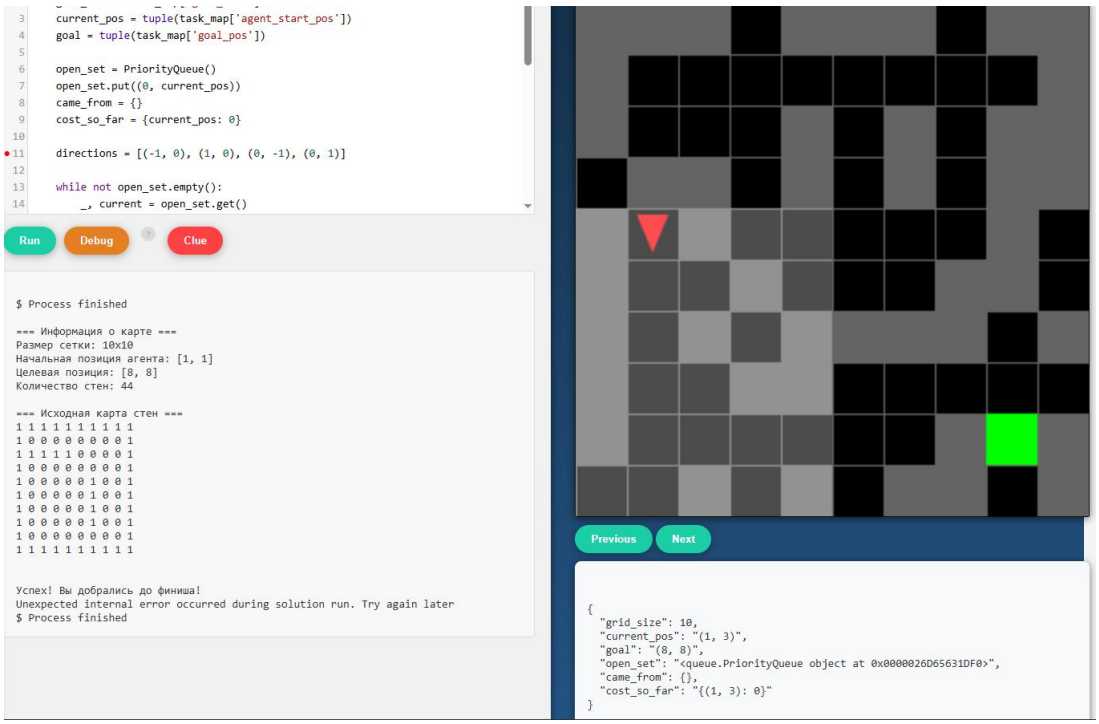

Рисунок 4 – Задача 4 пример запуска на кнопку debug и работы точки остановки на динамической карте.

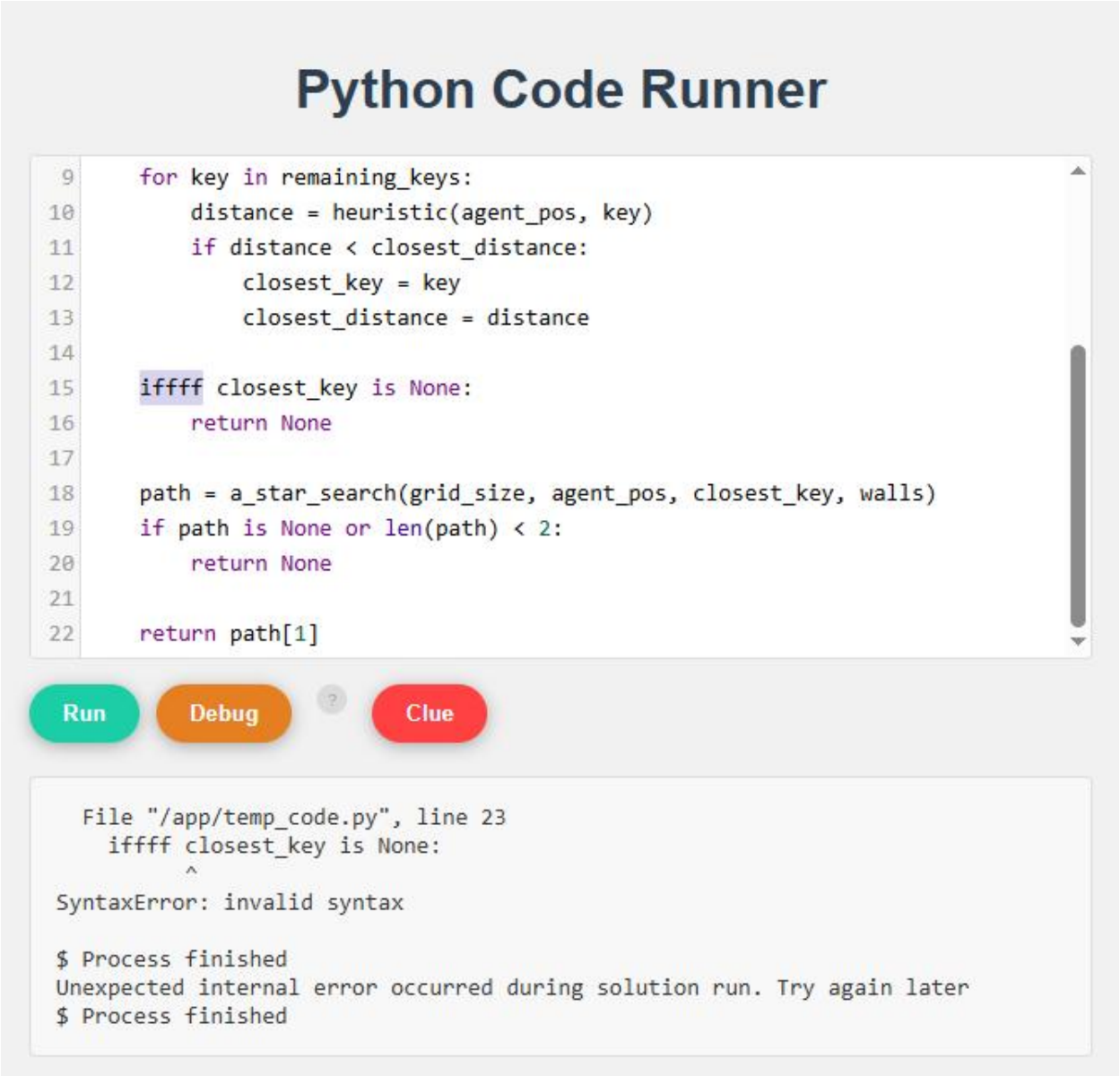

Рисунок 5 – Задача 3 пример ошибочного кода студента.

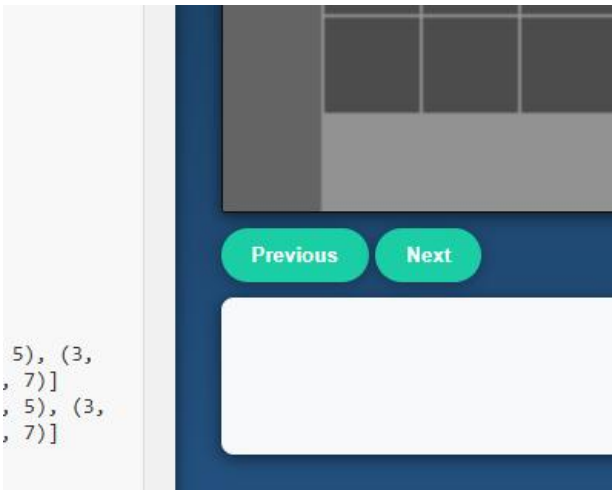

Рисунок 6 – Кнопки для трассировки и взаимодействия с визуализацией.

В случае когда студент нажимает на кнопку подсказки, всплывает окошко, с сообщением “точно ли подсказку показывать”. Данный механизм требуется для того, чтобы в будущем снижать отметку студенту за использование подсказки, так как подсказка сильно облегчает студенческое решение. Демонстрация работы подсказки на рис. 7, рис. 8.

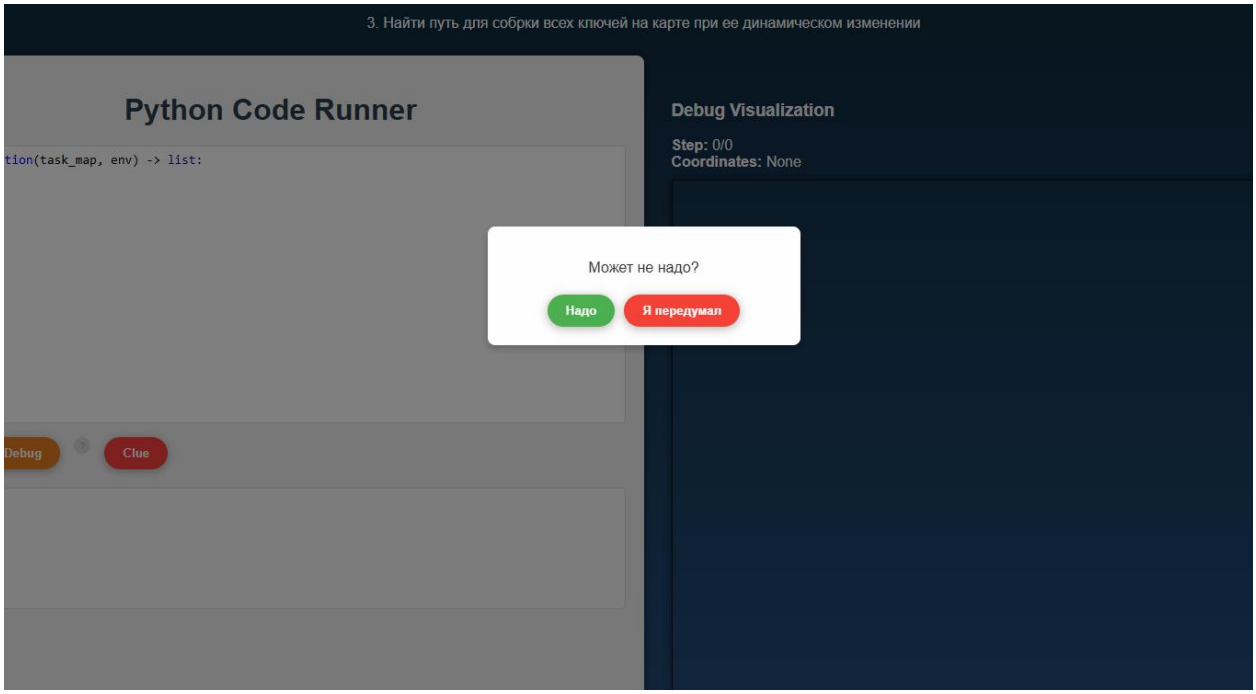

Рисунок 7 – Всплывающее окно.

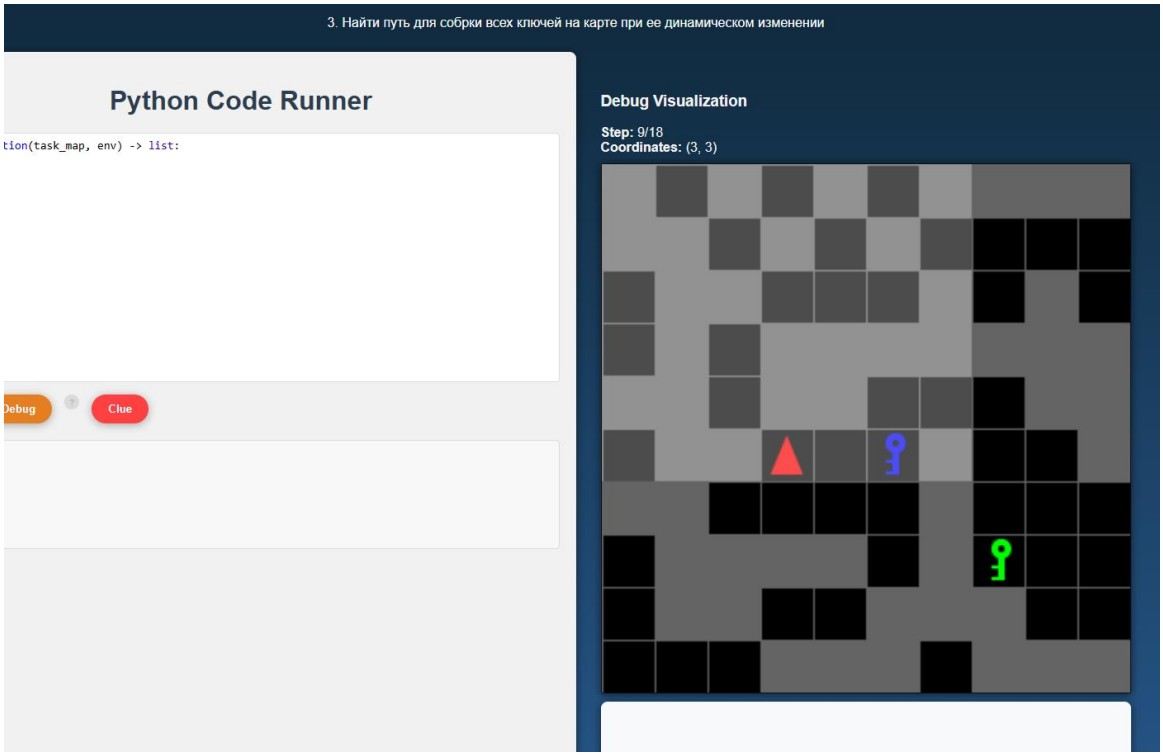

Рисунок 8 – Пример выполнения работы подсказки.

Для обеспечения понятной работы студенты не знающие о том, что можно поставить точки остановки могут навести на подсказку, данная деталь является элементом низкого порога вхождения за счет объяснений происходящего рис. 9.

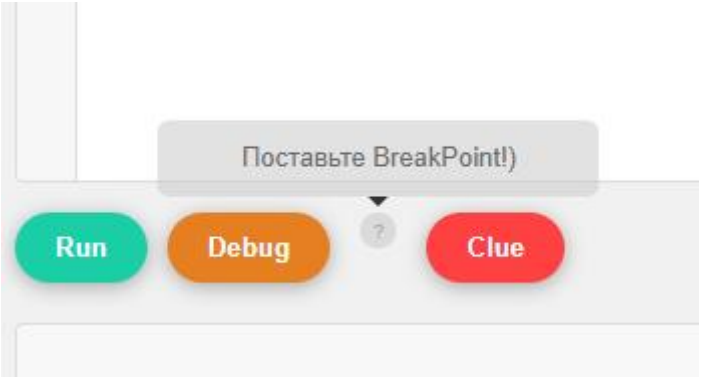

Рисунок 9 – Подсказка для точек остановки.

Обеспечение выбора для студентов может положительно сказываться на их продуктивности и энтузиазме, поэтому было реализовано меню в котором можно выбрать задачу для решения, нажав на кнопку в углу экрана, выпадает шторка со списком из существующих задач рис. 10.

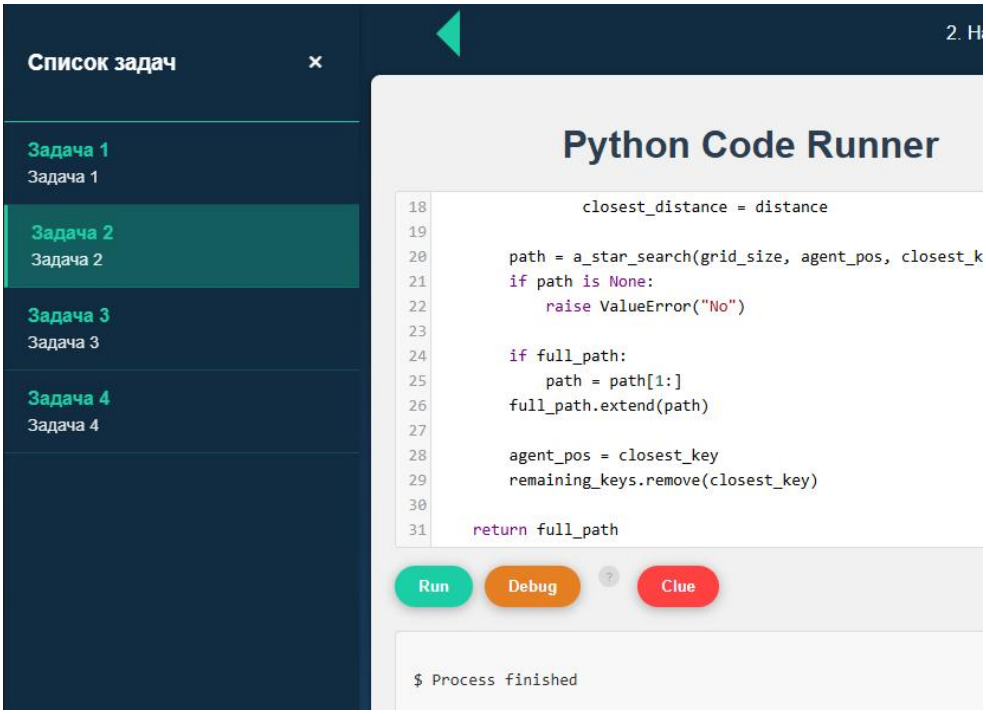

Рисунок 10 – Меню для выбора задач.

4 Исследование реализованного инструмента

4.1 Тестовые данные

В качестве данных для тестирования отладочной системы были использованы 4 задачи разработанные в среде minigrid, которые включают в себя генерацию карт, как статических так и динамических, нахождение верного пути, используя алгоритм A*, а также сопоставление верного пути со студенческим решением.

Были исследованы временные затраты и получены результаты экспериментов. Данное исследование производилось следующим образом: для двух задач, где первая со статической картой, а вторая с динамической, Задача1 и Задача2 соответственно. На каждой задаче 5 раз запускалось верное студенческое решение на кнопку run, 5 раз на кнопку debug и 5 раз на кнопку debug с точкой остановки, а в результат вносился средний временной промежуток. Результаты исследования приведены в таблице 2. В таблице приведено время, за которое отрабатывает каждая задача на кнопку run и debug.

Эксперименты проводились на персональном компьютере со следующими характеристиками:

- Процессор: AMD Ryzen 5 4600H with Radeon Graphics 3.00 GHz
- Оперативная память: 16.00 Гб

Данные, полученные во время исследования, находятся в таблице 2.

4.2 Результат исследования

Результат измерений представлен в таблице 2.

Таблица 2 – результат измерений.

	Задача1	Задача2
run, сек	5.3	5.35
debug, сек	5.5	5.7
debug+breakpoint, сек	6.0	15.0

4.3 Выводы и закономерности по данным исследования

Сравнение времени выполнения: Для обеих задач (статическая и динамическая карты) видно, что время выполнения студенческого решения при запуске через кнопку run значительно меньше, чем при использовании кнопки debug. Это подтверждает общее правило, что отладка добавляет дополнительные временные затраты из-за включения дополнительных функций и инструментов отслеживания переменных.

Влияние точки остановки: Время выполнения при использовании debug с точкой остановки (breakpoint) значительно увеличивается, особенно для задачи 2 (с динамической картой), где время составляет 15 секунд. Это говорит о том, что использование точек остановки сильно замедляет выполнение программы, что связано с дополнительными вычислительными затратами корректную обработку точек остановки.

Сравнение статической и динамической карты: Временные затраты для задания с динамической картой (Задача 2) несколько выше, чем для статической карты (Задача 1) во всех режимах работы (run, debug, debug с точкой остановки). Это свидетельствует о большем уровне сложности в навигации в динамических сценариях.

Оперативность обработки: В случае со статической картой, разница во времени между режимом run и debug составляет всего 0.2 секунды, в то время как для динамической карты эта разница достигает 0.35 секунды. Это подтверждает, что добавление отладчика в статических условиях вызывает меньшее воздействие, чем в динамических условиях.

Таким образом, данные указывают на необходимость осознанного выбора режима работы отладчика в зависимости от задач, что позволит ускорить время обработки и повысить эффективность работы с кодом.

5 Безопасность жизнедеятельности

5.1 Основные положения

ВКР посвящена отладке лабораторных работ и будет использоваться кафедрой в качестве обучающей платформы. Основными пользователями будут студенты обучающиеся на этой кафедре.

Интерфейс для пользователя играет ключевую роль в разработке программного обеспечения, так как именно он служит связующим звеном между пользователем и приложением. Правильное и понятное оформление такого интерфейса крайне важно для комфортного взаимодействия с системой и эффективного решения задач. Если интерфейс сложен для восприятия или вызывает негативные эмоции, пользователи могут отдать предпочтение более удобным вариантам. Важно отметить, что многие пользователи не отделяют интерфейс от функционала приложения, что делает качественный дизайн интерфейса особенно актуальным.

Основная задача при разработке интерфейса для таких приложений заключается в создании комфортного пространства для пользователей, которое не станет источником трудностей или стресса во время работы с системой. Недоработки в интерфейсе могут снизить продуктивность, повысить уровень стресса у пользователей и в крайних случаях создать угрозу для их здоровья и безопасности.

В ГОСТ Р ИСО 9241-110-2016 [9] перечислены следующие основные принципы организации диалога человек-система:

- информативность
- пригодность для обучения
- контролируемость
- устойчивость к ошибкам
- соответствие ожиданиям пользователей
- приемлемость организации диалога для выполнения производственного задания

Данный набор принципов представляют собой метод идентификации основных факторов, воздействующих на пригодность использования интерактивных систем.

При реализации приложения для локальной отладки лабораторных работ в среде minigrid Python была использована IDE PyCharm [10]. Данная IDE соответствует всем необходимым принципам организации человек-машина, представленным в ГОСТ Р ИСО 9241-110-2016.

5.2 Обзор среды разработки и соответствие принципам взаимодействия между человеком и системой

5.2.1 Информативность

Принцип информативности подразумевает, что пользователь должен получать четкую информацию о состоянии системы и программы в любой момент. Информация должна быть понятной, чтобы избежать недоразумений. Основные условия:

- Пользователь должен понимать, в каком диалоге он находится.
- Информация должна помогать завершению диалога.
- Обращение к справочной документации должно быть минимальным.

IDE PyCharm соответствует этим требованиям благодаря:

- Подсветке кода, помогающей различать синтаксис.
- Отображению информации о переменных, функциях и классах.
- Уведомлениям об ошибках, позволяющим своевременно определить проблемы.

5.2.2 Пригодность к обучению

Этот принцип предполагает, что диалог должен способствовать обучению пользователя. Основные условия:

- Правила использования программы должны быть легко доступны.

- Должна поддерживаться возможность повторного использования обучающего диалога.
- Предоставление обратной связи для понимания системы.

PyCharm поддерживает эти требования через:

- Ответы на действия пользователя.
- Типовые концепции программирования.
- Обширную справку и документацию.

5.2.3 Контролируемость

Принцип контролируемости обеспечивает возможность пользователю управлять процессом разработки. Основные условия:

- Пользователь должен выбирать варианты продолжения работы.
- Возможность определить точку для возобновления прерванной работы.
- Возможность отмены недавних действий.

PyCharm удовлетворяет эти условия через:

- Функции отмены и повтора действий.
- Настройку файловой структуры и горячих клавиш.
- Возможность вернуться к точке возобновления.

5.2.4 Устойчивость к ошибкам

Этот принцип нацелен на то, чтобы достичь результата с минимальными поправками со стороны пользователя. Условия:

- Система должна помогать в обнаружении ошибок.
- Предупреждать о действиях, ведущих к критическим ошибкам.
- Обеспечивать исправление типовых ошибок.

PyCharm выполняет эти требования благодаря:

- Функции автосохранения.

- Индикаторам и уведомлениям об ошибках.
- Подсветке ошибок в коде и предложению решений.

5.2.5 Адаптируемость к индивидуальным особенностям пользователя

Принцип заключается в том, что программа должна позволять пользователю изменять формат взаимодействия в соответствии с его потребностями. Основные условия:

- Система должна быть изменяемой для разных пользователей.
- Доступ к альтернативным формам представления информации.

PyCharm соответствует этим требованиям благодаря:

- Выбору тем (подсветка кода, яркость, размер шрифта).
- Ручной настройке параметров интерфейса.
- Установке плагинов для дополнительных настроек среды разработки.

5.2.6 Соответствие ожиданиям пользователя

Принцип гласит, что диалог должен отвечать ожиданиям пользователя и общепринятым соглашениям. Основные условия:

- - Использование знакомой терминологии.
- - Быстрая и удобная обратная связь.
- - Уведомление о задержках в реакции системы (например, индикатор загрузки).

IDE PyCharm соответствует этим требованиям благодаря:

- - Принятой среди разработчиков терминологии.
- - Точным уведомлениям об ошибках на уровне строки и символа.
- - Информация о текущем состоянии системы (обновление, загрузка библиотек, сборка проекта).оянии системы (обновление, загрузка библиотек, сборка проекта)

5.2.7 Приемлемость организации диалога для выполнения производственного задания

Принцип подразумевает, что интерактивная система должна помогать пользователю в выполнении производственного задания. Условия:

- Пользователь должен получать информацию об успешном завершении задания.
- Необходимо избегать ненужной информации, негативно влияющей на эффективность.
- Формат входных и выходных данных должен соответствовать заданию.

В PyCharm реализованы следующие функции соответствия:

- Подсветка кода в основной области.
- Автодополнение кода для повышения эффективности.
- Настраиваемый интерфейс.
- Поддержка различных дополнений.

5.3 Выводы

Используемая при разработке приложения для отладки лабораторных работ в среде Minigrid интегрированная среда разработки PyCharm обладает эргономичным и удобным интерфейсом, который полностью соответствует всем принципам, указанным в ГОСТ Р ИСО 9241-110-2016. Эта среда разработки обеспечивает высокую производительность программистов, что значительно упрощает написание кода для программного обеспечения.

Были представлены аргументы, подтверждающие, что выбор данной среды разработки обоснован с точки зрения соответствия принципам из ГОСТ Р ИСО 9241-110-2016. Примеры функционала PyCharm демонстрируют, как ее инструменты и возможности способствуют

повышению эффективности работы программистов, облегчая процесс разработки и отладки программ.

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы была успешно достигнута поставленная цель – разработано приложение для отладки лабораторных работ в среде minigrid. Исследование существующих решений позволило выделить ключевые элементы отладки и сформулировать требования к новому приложению, что в свою очередь обеспечило основу для его успешной реализации.

Полученные данные о временных затратах показывают эффективность и удобство приложения, однако выявленные недостатки указывают на необходимость дальнейшего совершенствования. Важно отметить, что каждая задача требует индивидуального подхода к отладке, особенно в условиях динамических сценариев.

На основании анализа текущего состояния разработки можно выделить несколько направлений для дальнейшей работы, такие как:

1. Расширение функционала для возможности одновременного использования данного приложения несколькими студентами,
2. Возможность регистрации каждого студента,
3. Сохранение оценок за прохождения задач, а так же механизм для понижения отметки в случае использования подсказки,
4. Добавление генерации карт, а так же возможность тестировать код студента на нескольких картах, чтобы убедиться, что решение студента работает универсально на всех картах, чтобы повысить сложность выполнения задач,
5. Добавление дополнительных инструментов для отладки и взаимодействия с визуализацией.

Эти шаги помогут сделать отладочную систему еще более наглядной и удобной для пользователей, что в конечном итоге повысит качество обучения и подготовки студентов в области графических алгоритмов.

Среди перспективных направлений для развития приложения также можно выделить интеграцию с другими образовательными инструментами и платформами. Это позволит создать единую экосистему, где студенты смогут не только выполнять лабораторные работы, но и получать обратную связь, сравнивать свои результаты с другими учащимися и более эффективно изучать графические алгоритмы.

Таким образом, дальнейшая работа над приложением не только усилит его функциональность, но и создаст дополнительные возможности для сотрудничества и обмена знаниями среди студентов. Реализация предложенных улучшений будет способствовать более глубокому пониманию материала и подготовит будущих специалистов к реальным условиям работы в области программирования и разработки.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Кузнецов Н. В. Онлайн-образование: ключевые тренды и препятствия // E-Management. 2019. №1. URL: https://cyberleninka.ru/article/n/onlayn-obrazovanie-klyuchevye-trendy-i-prepyatstviya (дата обращения: 19.12.2024).

2. Шурыгин Виктор Юрьевич ЭЛЕКТРОННЫЕ СИСТЕМЫ УПРАВЛЕНИЯ ОБУЧЕНИЕМ В АКАДЕМИЧЕСКОМ И КОРПОРАТИВНОМ ОБРАЗОВАНИИ // АНИ: педагогика и психология. 2021. №2 (35). URL: https://cyberleninka.ru/article/n/elektronnye-sistemy-upravleniya-obucheniem-v-akademicheskom-i-korporativnom-obrazovanii (дата обращения: 19.12.2024).

3. Жаворонкова О. Г. СРАВНЕНИЕ КРОССПЛАТФОРМЕННЫХ СРЕД РАЗРАБОТКИ ДЛЯ СОЗДАНИЯ 3D-ИГР //Состав редакционной коллегии и организационного комитета. – 2024.

4. Godot Engine // [URL](https://docs.godotengine.org/en/stable/about/introduction.html)

5. Кулибин // [URL](https://omega-ed.ru/kulibin)

6. Blender Debugger // [URL](https://github.com/AlansCodeLog/blender-debugger-for-vscode/tree/master)

7. Webots // [URL](https://cyberbotics.com/)

8. Godot Engine // [URL](https://docs.godotengine.org/en/stable/about/introduction.html)

9.ГОСТ Р ИСО 9241-110-2016 Эргономика взаимодействия человек-система. Часть 110. Принципы организации диалога. М.: Стандартинформ, 2016.

10. PyCharm [Электронный ресурс] URL: https://www.jetbrains.com/pycharm/

ПРИЛОЖЕНИЕ А