МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ИМЕНИ П.О.СУХОГО

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

специальности 6-05-0611-01 Информационные системы
и технологии (в игровой индустрии)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе
по дисциплине «Объектно-ориентированное программирование»

на тему: «Игровое десктопное приложение «Садовник» с использование графики *OpenGL*»

Исполнитель: студент гр. ИТИ-21
Калинин В.М.
Руководитель: доцент
Курочка К.С.

Дата проверки: _____
Дата допуска к защите: _____
Дата защиты: _____
Оценка работы: _____

Подписи членов комиссии
по защите курсового проекта: _____

Гомель 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном обществе компьютерные игры занимают важное место в культуре и досуге, объединяя людей разных возрастов и интересов. Благодаря технологическому прогрессу игры становятся всё более глубокими и интерактивными, позволяя не только получать удовольствие от процесса, но и развивать навыки логического мышления, стратегического планирования и управления ресурсами.

В рамках данной работы разработано десктопное приложение «Садовник», направленное на создание увлекательной многопользовательской игры, где каждый игрок сможет управлять своим садоводом, выращивать дерево и собирать урожай. Особое внимание уделяется продуманной архитектуре приложения с использованием принципов объектно-ориентированного программирования и паттернов проектирования, таких как «фабричный метод» и «декоратор». Это позволяет создать гибкую и легко расширяемую систему, способную учитывать различные состояния дерева и потребности в ресурсах.

Приложение представляет интерес как для молодых пользователей, так и для более опытных игроков, желающих развить навыки стратегического мышления и управления ограниченными ресурсами. Простота управления и понятный интерфейс обеспечивают доступность приложения для широкой аудитории.

Развитие практических навыков: в процессе игры предлагается пользователю принимать решения, связанные с поливом, защитой от вредителей и удобрением дерева, что требует грамотного анализа ситуации и быстрой реакции. Данный подход способствует развитию аналитического мышления и умения адаптироваться к постоянно меняющимся условиям игрового процесса.

При создании приложения особое внимание уделяется выбору оптимальных технологий, таких как *C#, Windows Forms* и библиотека *OpenGL* для реализации спрайтовой графики. Важным этапом также является тестирование и оптимизация игрового процесса для обеспечения стабильной работы программы. Все эти аспекты позволяют не только создать качественный продукт, но и способствуют приобретению ценных практических навыков в области программирования и проектирования интерактивных приложений.

Таким образом, разработанное приложение «Садовник» объединяет в себе как развлекательный, так и образовательный аспекты, создавая основу для реализации игрового продукта, способного привлечь внимание широкой аудитории и обеспечить положительный пользовательский опыт.

1 СОВРЕМЕННЫЕ ПОДХОДЫ В РАЗРАБОТКЕ ИНТЕРАКТИВНЫХ ИГРОВЫХ ПРИЛОЖЕНИЙ

1.1 Теоретические основы объектно-ориентированного программирования (ООП)

Объектно-ориентированное программирование (ООП) представляет собой парадигму разработки программного обеспечения, получившую широкое распространение при создании интерактивных приложений и игр [1]. Основное преимущество ООП заключается в возможности моделирования сущностей реального мира посредством создания объектов, каждый из которых обладает собственным состоянием и поведением. Такой подход значительно упрощает разработку сложных систем, повышая их структурированность, гибкость и масштабируемость.

Ключевыми принципами ООП являются инкапсуляция, наследование, полиморфизм и абстракция [2, с. 64-67].

Инкапсуляция подразумевает объединение данных и методов, работающих с этими данными, в единую сущность – объект. Это позволяет скрыть внутреннюю реализацию от внешнего мира, предоставляя доступ только через публичный интерфейс. Такой подход повышает безопасность кода, снижая вероятность ошибок и упрощая его сопровождение. В игровых приложениях инкапсуляция способствует разделению логики управления игровыми элементами и их отображения, что позволяет легко обновлять и модифицировать отдельные компоненты без риска нарушения общей функциональности.

Наследование позволяет создавать новые классы на основе уже существующих, перенимая их свойства и методы. Это существенно сокращает дублирование кода и способствует повторному использованию уже реализованных решений. В контексте разработки игр наследование дает возможность создавать иерархии классов, где базовые сущности (например, общий класс «Игровой объект») могут быть расширены для реализации специализированных объектов (персонажи, предметы, эффекты). Для проекта «Садовник» это означает, что базовые элементы, такие как базовый класс для дерева, могут быть унаследованы и дополнены функционалом, специфичным для разных видов растений или условий их роста.

Полиморфизм представляет собой способность объектов разных классов реагировать на одинаковые вызовы методов по-своему. Этот принцип обеспечивает гибкость в реализации игровых механик: один и тот же метод может выполнять различные действия в зависимости от типа объекта, который его вызывает. Например, метод «обновить состояние» может по-разному обрабатывать игрового персонажа, дерево или ресурс, что позволяет создавать более динамичный и адаптивный игровой процесс.

Абстракция заключается в выделении общих характеристик объектов и сокрытии деталей их реализации. Абстрактные классы и интерфейсы позволяют описывать общие свойства и поведение, не вдаваясь в конкретные детали, что значительно упрощает проектирование системы. Это особенно полезно при

разработке игр, где необходимо создать универсальные компоненты, способные адаптироваться к различным ситуациям и требованиям. В проекте «Садовник» абстракция позволяет разработать базовый набор методов для работы с игровыми объектами, который затем может быть конкретизирован для реализации уникальных характеристик каждого элемента игры.

Применение принципов ООП в разработке игр способствует созданию модульной архитектуры, где каждая функциональная часть системы (например, управление ресурсами, обработка событий, рендеринг графики) представлена отдельными, независимыми модулями. Это не только повышает удобство поддержки и расширения кода, но и обеспечивает возможность легкого внедрения новых игровых механик без необходимости пересмотра всей системы. Для проекта «Садовник» такой подход особенно важен, так как он позволяет гибко реализовывать логику роста дерева, изменения его состояния, а также взаимодействие с пользователем, что является критически важным для обеспечения качественного игрового опыта.

Таким образом, объектно-ориентированное программирование предоставляет разработчику мощные инструменты для создания сложных и адаптивных игровых приложений. Грамотное применение принципов инкапсуляции, наследования, полиморфизма и абстракции не только упрощает процесс разработки, но и способствует созданию продукта, способного удовлетворить требования современного рынка игр, обеспечивая высокую производительность и легкость дальнейшего развития функционала.

1.2 Паттерны проектирования: «фабричный метод» и «декоратор»

Паттерны проектирования представляют собой проверенные решения для типовых проблем, возникающих при разработке программных систем. Они позволяют создавать гибкие, масштабируемые и легко поддерживаемые архитектуры, что особенно актуально в динамичной сфере разработки игр. Среди множества паттернов особое место занимают «фабричный метод» и «декоратор», которые широко используются как в крупных игровых проектах, так и в небольших приложениях.

«Фабричный метод» применяется для организации процесса создания объектов, отделяя логику их инстанцирования от основного кода. Это позволяет разработчикам не зависеть от конкретных классов, а использовать абстрактные методы для генерации объектов, что значительно упрощает добавление новых типов элементов без необходимости переработки существующей структуры. В контексте игр это особенно полезно, поскольку игровые сценарии часто требуют динамического создания множества объектов – будь то персонажи, элементы окружения или, например, события вроде атак вредителей. В проекте «Садовник» применение «фабричного метода» позволяет централизованно управлять процессом создания игровых объектов, таких как различные виды деревьев или специальные игровые события, обеспечивая гибкость и расширяемость кода.

Паттерн «декоратор», в свою очередь, предлагает способ динамического расширения функциональности объектов без изменения их базовой структуры.

Он позволяет «оборачивать» базовые классы дополнительными слоями, каждый из которых добавляет новые свойства или поведение. Это особенно актуально для игровых приложений, где часто возникает необходимость модифицировать поведение объектов – например, добавлять дополнительные характеристики к игровому персонажу или улучшать функционал стандартного элемента. В проекте «Садовник» «декоратор» может быть использован для динамического расширения возможностей базового класса дерева: к нему можно добавлять новые эффекты, влияющие на скорость роста, устойчивость к вредителям или другие игровые характеристики, не создавая при этом избыточную иерархию классов.

В совокупности применение этих паттернов значительно повышает качество архитектуры приложения. «Фабричный метод» способствует изолированию процесса создания объектов, что упрощает интеграцию новых функций и улучшает тестируемость системы. «Декоратор» же позволяет гибко наращивать функциональность, делая код более адаптивным к изменениям требований и позволяя реализовать сложные игровые механики без нарушения принципов модульности.

Таким образом, использование паттернов «фабричный метод» и «декоратор» не только удовлетворяет специфические потребности проекта «Садовник», но и демонстрирует общепризнанные подходы, применяемые во всей индустрии разработки игр. Эти паттерны способствуют созданию эффективных, удобных для поддержки и расширения систем, что является залогом реализации любого современного игрового приложения.

Диаграмма, иллюстрирующая архитектуру паттерна «Фабричный метод» для фабрик проблем, приведена в приложении А, а диаграмма, иллюстрирующая архитектуру паттерна «Декоратор» применительно к иерархии классов деревьев, приведена в приложении Б.

1.3 Технологии разработки интерактивных игровых приложений и методы решения задачи

Создание интерактивных игровых приложений требует грамотного выбора инструментов, языков программирования и эффективных методов проектирования для реализации сложной логики, графики и управления ресурсами. Правильный выбор технологий и подходов играет ключевую роль в оптимизации производительности, улучшении пользовательского опыта и обеспечении масштабируемости проекта.

1.3.1 Для создания современных игровых приложений используются различные языки программирования и графические библиотеки. Одним из решений является язык *C#* в связке с фреймворком *Windows Forms*, который позволяет создавать графические интерфейсы для десктопных приложений [3]. В сочетании с библиотекой *OpenGL*, обеспечивающей рендеринг *2D* и *3D* графики, это дает возможность разработчику контролировать аспекты визуализации игрового процесса [4, с. 96-107].

C# предоставляет возможности для реализации объектно-ориентированной архитектуры, что позволяет создавать гибкие игровые системы. *Windows*

Forms позволяет интегрировать элементы управления интерфейсом, а *OpenGL* – работать с графикой на низком уровне, обеспечивая производительность и плавную анимацию [5, с. 92-119].

В проекте «Садовник» использование *C#* и *OpenGL* позволяет создать интерфейс для управления игровым процессом и визуализировать состояние дерева, влияния окружающей среды и действий игрока.

1.3.2 При разработке интерактивных игровых приложений решаются ключевые задачи:

а) управление состоянием игры: контролируется состояние каждого игрового объекта, например, дерева. Для этого применяются структуры данных, отслеживающие уровень воды, удобрений и состояние здоровья растения. В ООП это достигается за счет создания отдельных классов;

б) обработка игровых событий: реализация событийной модели позволяет отслеживать действия игрока и изменять состояние объекта в реальном времени;

в) реализация алгоритмов управления ресурсами: внедряются алгоритмы, регулирующие доступность ресурсов и их влияние на состояние дерева;

г) рендеринг графики и анимация: использование *OpenGL* позволяет реализовать спрайтовую графику и анимацию;

д) тестирование и отладка: контроль за стабильностью приложения осуществляется через модульное тестирование.

1.3.3 Методы, описанные выше, широко применяются не только в проекте «Садовник», но и в игровой индустрии в целом. Например:

– в стратегических играх эти методы позволяют управлять ресурсами и контролировать действия персонажей;

– в симуляторах – отслеживать состояния объектов и реагировать на действия игрока;

– в ролевых играх – обрабатывать взаимодействие между персонажами и окружающей средой.

Применение технологий *C#* и *OpenGL* в проекте «Садовник» дает возможность реализовать уникальные игровые механики, связанные с уходом за растением и сбором урожая, а также позволяет расширять функциональность игры в будущем.

Таким образом, выбор технологий и методов проектирования позволяет создать качественное и стабильное игровое приложение, а также обеспечить его масштабируемость, что делает проект «Садовник» актуальным решением для изучения современных подходов в разработке интерактивных приложений.

1.4 Архитектура игровых систем и управление ресурсами

Архитектура игровых систем определяет структуру и взаимодействие компонентов программного обеспечения, обеспечивая стабильность, гибкость и производительность приложения [6, с. 1-62]. От грамотного проектирования

архитектуры зависит возможность масштабирования игры и удобство добавления нового функционала.

1.4.1 Архитектура игровой системы включает ряд ключевых модулей:

а) игровой движок: управляет логикой игры, физикой, анимацией и обработкой ввода от пользователя;

б) графический модуль: отвечает за рендеринг объектов, анимации и визуальные эффекты с использованием графических библиотек, таких как *OpenGL* или *DirectX*;

в) модуль управления ресурсами: контролирует доступ к игровым объектам, таким как текстуры, звуки, модели и эффекты;

г) система управления событиями: отслеживает действия пользователя и реагирует на изменения в игровом мире;

д) база данных или файловая система: хранит информацию о состоянии игры, прогрессе пользователя и настройках.

В крупных проектах, таких как многопользовательские онлайн-игры, архитектура усложняется за счет включения серверной части для обработки сетевого взаимодействия между игроками и облачных хранилищ для сохранения данных.

1.4.2 Управление ресурсами является одной из важных задач при разработке игр. Ресурсы включают все элементы, влияющие на игровой процесс: графические текстуры, звуковые эффекты, анимации, игровые объекты и внутриигровую экономику.

В современных играх управление ресурсами включает:

– загрузку и выгрузку текстур и моделей в реальном времени для оптимизации памяти;

– управление памятью для предотвращения утечек и повышения производительности;

– контроль над игровыми объектами и их состоянием (например, здоровье персонажа, количество боеприпасов или прогресс роста растения);

– баланс между доступными ресурсами и действиями игрока.

1.4.3 В рамках курсового проекта «Садовник» архитектура игрового приложения построена по модульному принципу, что позволяет управлять каждым компонентом игры.

Логический модуль отвечает за обработку действий игрока (полив, удобрение, защита от вредителей). Логика ухода за растением основана на объектно-ориентированном подходе, где каждое дерево является отдельным объектом со своими характеристиками.

Графический модуль с использованием *OpenGL* отвечает за отображение игрового поля, состояния дерева и анимацию внешних факторов. Это создает визуально привлекательную картину и наглядно показывает изменения состояния дерева.

Модуль управления ресурсами позволяет контролировать доступ к удобрениям, воде и средствам защиты от вредителей. Важный аспект в игре «Садовник» – это ограниченность ресурсов. Использование ресурса одним игроком временно блокирует его для другого. Синхронизация доступа к общим ресурсам (таким как инвентарь с инструментами) в приложении реализуется посредством механизма блокировок. При захвате ресурса одним игроком устанавливается эксклюзивная блокировка, предотвращающая одновременный доступ других игроков. Блокировка снимается после завершения использования ресурса или по истечении заданного таймаута, что обеспечивает целостность данных и предотвращает конфликты в многопользовательской среде.

Система событий контролирует случайные атаки вредителей, что добавляет в игру элемент непредсказуемости и требует от игрока быстрого принятия решений.

1.4.4 В крупных играх, таких как *Minecraft, The Sims* или *Stardew Valley,* управление ресурсами играет ключевую роль. Например, в *Stardew Valley* игрок управляет фермой, следит за состоянием почвы, урожаем и отношениями с *NPC*. Все эти элементы требуют точного контроля над ресурсами, времени и действий игрока.

В многопользовательских играх, таких как *World of Warcraft*, ресурсы – это внутриигровая экономика, контроль за валютой, материалами для крафта и торговлей между игроками. Здесь важную роль играет серверная часть, которая отслеживает баланс экономики и предотвращает мошенничество.

В проекте «Садовник» управление ресурсами заключается в контроле над действиями игрока по уходу за деревом: своевременный полив, внесение удобрений и защита от вредителей. Это позволяет реализовать элемент соревновательности между игроками, где побеждает тот, кто лучше управляет своими ресурсами и быстрее реагирует на изменения в игровом мире.

1.4.5 Неправильное управление ресурсами может привести к снижению производительности, утечкам памяти и лагам в игре. В проекте «Садовник» для оптимизации используется система ленивой загрузки ресурсов (*Lazy Loading*), при которой текстуры и анимации подгружаются только в момент необходимости.

Таким образом, архитектура игровых систем и управление ресурсами играют ключевую роль в создании интерактивных приложений. Применение модульного подхода, использование графического движка *OpenGL* и языка *C#* позволяет эффективно контролировать логику игры, визуализацию и состояние игровых объектов. Управление ресурсами в проекте «Садовник» позволяет реализовать реалистичный игровой процесс и создать элементы соревновательности между игроками. Грамотно спроектированная система управления ресурсами обеспечивает оптимальную производительность игры и дает возможность легко расширять функционал приложения.

1.5 Сравнительный анализ существующих подходов и методов решения

Анализ существующих методов разработки игровых приложений позволяет выделить преимущества комбинированного подхода, который использует принципы объектно-ориентированного программирования (ООП), проверенные паттерны проектирования и современные технологии разработки. Традиционные методы, основанные на процедурном программировании, зачастую оказываются менее гибкими из-за жесткой привязки логики к функциям и переменным, что затрудняет масштабирование и поддержку приложения. При этом процедурный подход менее приспособлен для работы с растущей сложностью динамичных игр, где требуется быстрая адаптация к новым условиям и частые изменения в логике работы игры.

Объектно-ориентированное программирование позволяет создавать более структурированные, модульные и легко масштабируемые приложения. Применение принципов инкапсуляции, наследования, полиморфизма и абстракции дает возможность строить систему на основе отдельных классов и объектов, которые взаимодействуют между собой, не нарушая общей логики программы. Благодаря использованию *SOLID*-принципов, разработчики могут организовать код так, чтобы внесение изменений или добавление нового функционала происходило без риска нарушения существующих компонентов. Это особенно важно при разработке игр, где каждый элемент – будь то дерево, враг или ресурс – обладает уникальными свойствами и специфическим поведением, что значительно упрощает тестирование, отладку и дальнейшее развитие проекта.

В проекте «Садовник» объектно-ориентированный подход позволяет четко разделить игровую логику на независимые модули, такие как управление ресурсами, обработка атак вредителей и другие аспекты. Например, базовый класс «Дерево» может содержать общие характеристики, такие как уровень роста и базовые реакции на внешние воздействия. От него может осуществляться наследование конкретных видов деревьев, с добавлением уникальных свойств и специфических реакций на внешние факторы, что отражает принцип разделения обязанностей и позволяет легко внедрять новые игровые механики.

Применение паттернов проектирования, таких как «Фабричный метод» и «Декоратор», позволяет еще больше повысить гибкость системы. Фабричный метод используется для создания случайных событий в игре – будь то появление новых типов вредителей или генерация неожиданных заданий – без необходимости жесткой привязки к конкретным классам. Такой подход упрощает добавление новых типов атак или защитных механизмов в будущем. Паттерн «Декоратор» позволяет динамически расширять функциональность объектов, например, добавляя усиленную защиту от вредителей или ускорение роста при использовании определенных удобрений. Дополнительно, применение паттернов «Наблюдатель» для управления событиями и «Стратегия» для определения тактики врагов способствует снижению дублирования кода и созданию более адаптивной архитектуры, готовой к быстрому изменению требований рынка.

В отличие от процедурного программирования, где логика жестко привязана к конкретным функциям и переменным, ООП позволяет легко вносить изменения в код, не затрагивая основные механики игры. Такой подход обеспечивает возможность рефакторинга и оптимизации кода с минимальными затратами времени, снижая риск появления ошибок при обновлении или добавлении новых функций. Это особенно важно для игр с динамическим игровым процессом, где события могут изменяться в реальном времени, требуя оперативного вмешательства и адаптации логики работы приложения.

Использование языка *C#* и платформы *Windows Forms* в сочетании с графической библиотекой *OpenGL* предоставляет разработчику мощные инструменты для создания интуитивного интерфейса и качественного рендера графики. *C#* обладает высокоуровневой управляемой средой выполнения, что позволяет эффективно управлять памятью, интегрировать многочисленные библиотеки *.NET* и быстро разрабатывать функционал. *Windows Forms* обеспечивает удобный инструментарий для создания гибких графических интерфейсов, а *OpenGL* – кроссплатформенность и высокую производительность при отрисовке спрайтов, анимации и сложных графических эффектов. Такая комбинация технологий позволяет добиться плавности игрового процесса, высокой детализации визуальных эффектов и реалистичной динамики объектов на экране.

Работа с ресурсами в проекте «Садовник» требует особого подхода, поскольку такие элементы, как вода, удобрения и средства защиты от вредителей, являются ограниченными ресурсами. Объектно-ориентированный подход позволяет реализовать систему управления ресурсами, где каждый игрок контролирует свои запасы и принимает решения о том, как эффективно их использовать. Встроенная система случайных событий, влияющих на состояние дерева, например, внезапные атаки вредителей, добавляет стратегический элемент, требующий тщательного планирования и быстрого реагирования. Такой механизм создает баланс между расходами и доходами, стимулируя игроков к оптимальному распределению ресурсов и принятию стратегически важных решений.

Комбинированный подход, основанный на использовании ООП и паттернов проектирования, доказывает свою эффективность на примерах множества проектов. В игре *Stardew Valley*, например, ООП используется для управления фермой, ресурсами и взаимодействия с персонажами, а паттерны «Фабричный метод» и «Декоратор» помогают генерировать случайные события и динамически улучшать характеристики урожая или животных. Такой подход позволяет разработчикам быстро реагировать на пожелания сообщества, обновлять игровой процесс и поддерживать высокий уровень вовлеченности игроков. Подобная гибкость архитектуры находит применение и в других инди-играх, где баланс между реализмом и игровой механикой играет ключевую роль.

В проекте «Садовник» комбинированный подход позволяет не только реализовать базовую игровую логику, но и с легкостью добавлять новые элементы в будущем. Расширяемость системы позволяет, например, увеличивать ассортимент деревьев, вводить новые виды удобрений, развивать систему достижений или интегрировать дополнительные игровые режимы. Модульная архитектура также открывает возможности для реализации многопользовательского режима,

где игроки могут объединяться для совместного ухода за виртуальными садами, участвовать в кооперативных или соревновательных сценариях, а также обмениваться ресурсами. Такой режим требует особого внимания к синхронизации состояний объектов, безопасности данных и оптимизации сетевых соединений, что дополнительно повышает качество и конкурентоспособность конечного продукта.

Таким образом, применение объектно-ориентированного программирования в сочетании с паттернами проектирования и современными технологиями разработки позволяет создать стабильное, производительное и гибкое игровое приложение. В проекте «Садовник» этот подход не только реализует все ключевые аспекты игрового процесса, но и обеспечивает легкость модификации, масштабируемость и поддержку на протяжении всего жизненного цикла продукта. Модульная структура кода облегчает тестирование, отладку и интеграцию новых функций, а постоянное совершенствование архитектуры позволяет оперативно реагировать на изменения рынка и требования пользователей, что является залогом создания качественного и конкурентоспособного игрового продукта.

2 АРХИТЕКТУРА ИГРОВОГО ДЕСКТОПНОГО ПРИЛОЖЕНИЯ «САДОВНИК»

В данном разделе представлена архитектура разработанного игрового приложения «Садовник». Приводится описание основных требований к приложению, спецификация игровых механик, детальная структура классов и их взаимодействие, а также общая схема работы приложения.

2.1 Спецификация требований и описание игровых механик

В этом подразделе описаны функциональные и нефункциональные требования к приложению, а также подробно рассмотрены игровые механики, реализованные в игре «Садовник».

2.1.1 На основе задания по курсовому проектированию к разрабатываемому приложению предъявляются следующие требования:

– тип приложения: десктопное приложение;

– платформа разработки: язык программирования *C#*, фреймворк *Windows Forms*;

– графика: использование спрайтовой графики и средств библиотеки *OpenGL* для отображения объектов на игровом поле;

– игровой процесс: реализация многопользовательской игры для двух игроков;

– цель игры: каждый игрок управляет садовником, выращивает свое дерево и собирает урожай. Побеждает игрок, собравший наибольший урожай;

– уход за деревом: деревья требуют полива, защиты от вредителей, вирусных и грибковых инфекций, а также подкормки удобрениями;

– потребности дерева: моменты возникновения потребностей определяются случайным образом согласно индивидуальным законам распределения для каждого вида дерева;

– ресурсы: ресурсы для ухода за деревьями (вода, удобрения, пестициды и т.д.) являются общими для игроков. Использование ресурса одним игроком блокирует его для другого на определенное время;

– урожай: формируется в зависимости от своевременности удовлетворения потребностей дерева;

– шаблоны проектирования: обязательное использование шаблонов «Фабричный метод» для генерации атак вредителей и «Декоратор» для задания характеристик деревьев;

– система контроля версий: использование приватного репозитория *Git*.

2.1.2 Игровой процесс в приложении «Садовник» строится на комплексе взаимосвязанных механик, определяющих действия игроков и взаимодействие с игровым миром. Понимание этих механик является ключевым для освоения игры и достижения победы.

Игроки управляют своими садовниками на общем игровом поле, перемещаясь с помощью клавиш (*WASD* для первого игрока, стрелки для второго). Взаимодействие с игровыми объектами, такими как столы (для доступа к инвентарю), дерево (для ухода) и ящики (для сбора урожая), осуществляется нажатием специальной клавиши (*E* для первого игрока, *Enter* для второго).

В процессе игры у дерева случайным образом возникают потребности, требующие своевременного ухода. Игрок должен выбрать соответствующий инструмент из общего инвентаря, который становится доступным при приближении к столу. Выбор инструмента осуществляется цифровыми клавишами. Важно учитывать, что игрок может держать только один предмет (инструмент или фрукт) одновременно. Использование инструмента на дереве, когда у него есть соответствующая проблема, способствует его росту.

Инструменты в инвентаре являются общими и ограниченными ресурсами для обоих игроков. Если один игрок берет инструмент, другой игрок не может его использовать до тех пор, пока первый игрок не вернет его (автоматически через некоторое время или при выборе другого предмета/подборе фрукта). Эта механика создает элемент соревнования и требует стратегического планирования использования ресурсов.

Удовлетворение потребностей дерева напрямую влияет на увеличение стадии роста фрукта. При достижении максимальной стадии на дереве появляется фрукт, который падает на землю. Задача игрока — подобрать этот фрукт и донести его до своего ящика для сбора урожая, тем самым увеличивая свой счет.

Игра завершается, когда один из игроков первым собирает максимальное количество фруктов в своем ящике. Этот игрок объявляется победителем, что и является основной целью игры.

Подробное руководство по использованию приложения содержится в приложении В.

2.2 Проектирование архитектуры приложения

Архитектура приложения «Садовник» разрабатывалась с использованием объектно-ориентированного подхода и применением указанных в задании паттернов проектирования. Процесс проектирования включал декомпозицию системы на логические модули, определение ключевых сущностей и их взаимосвязей, а также выбор подходящих паттернов для решения типовых задач. Основной целью проектирования было создание модульной, гибкой и расширяемой архитектуры, способной обеспечить стабильную работу приложения и удобство дальнейшей разработки. Система была разделена на модули, отвечающие за различные аспекты игры: логику, рендеринг, физику, управление игроками и звуком.

2.2.1 Приложение состоит из множества классов, каждый из которых выполняет свою определённую роль в общей архитектуре. Иерархия всех классов приложения представлена на диаграмме в приложении Г. Ниже представлены ключевые классы и их назначение:

– *GameManager*: центральный класс, управляющий общей логикой игры, состоянием игроков, взаимодействием между компонентами. Инициализирует все подсистемы (рендеринг, физика, аудио и т.д.) и обрабатывает основной игровой цикл (*Update*);

– *Player*: представляет игрового персонажа. Отвечает за хранение состояния игрока (позиция, направление, анимация), обработку его движения и взаимодействие с физической подсистемой;

– *PlayerState*: управляет состоянием дерева конкретного игрока, отслеживает стадию роста фрукта, наличие проблем у дерева и время до их появления/исчезновения;

– *InventoryManager*: управляет состоянием общего инвентаря, отслеживает доступность инструментов, и какой игрок держит какой предмет. Этот класс совместно с *InventoryHandler* реализует механизм блокировок для синхронизации доступа игроков к общим ресурсам, описанный в разделе 1.4;

– *CollisionManager*: отвечает за обнаружение и разрешение коллизий между игроками и статическими объектами на карте;

– *AudioManager*: управляет воспроизведением фоновой музыки и звуковых эффектов в игре;

– *Map*: хранит данные о структуре игрового поля (тайлы для каждого игрока);

– *TextureManager*: загружает и управляет текстурами, используемыми в игре;

– *TextureRegistry*: предоставляет централизованный доступ ко всем загруженным текстурам по их назначению;

– *GameRenderer*: главный класс рендеринга, управляет отрисовкой всех игровых элементов (карты, спрайтов, игроков, *UI*) с использованием *OpenGL*;

– *MapRenderer*: отвечает за отрисовку игрового поля (воды, травы, троп, препятствий) на основе данных из класса *Map*;

– *SpriteRenderer*: рендерит динамические игровые объекты, такие как столы, ящики с фруктами, само дерево, падающие фрукты и иконки проблем;

– *TreeRenderer*: специализированный рендерер для отрисовки дерева, фруктов и их теней;

– *InventoryUIRenderer*: отвечает за отрисовку пользовательского интерфейса инвентаря;

– *Problem*: представляет проблему, возникшую у дерева (полив, вредители и т.д.), хранит ее тип и оставшееся время до исчезновения;

– *ProblemFactory* (абстрактный класс): определяет интерфейс для создания проблем;

– *AppleTreeProblemFactory, PearTreeProblemFactory*: конкретные реализации фабричного метода для создания проблем, специфичных для яблони и груши соответственно, с учетом вероятностей возникновения различных типов проблем;

– *ITree* (интерфейс): определяет базовые свойства дерева;

– *Tree* (базовый класс): базовая реализация *ITree*;

– *AppleTreeDecorator*, *PearTreeDecorator*: классы-декораторы, расширяющие базовый класс *Tree* для представления яблони или груши, добавляя специфические фабрики проблем;

– *MovementHandler*: управляет логикой движения игрового персонажа, включая обработку ввода, применение скорости и обновление анимации;

– *InteractionHandler*: обрабатывает взаимодействие игроков с игровыми объектами (деревом, ящиками, фруктами), проверяет расстояние и условия взаимодействия;

– *InventoryHandler*: обрабатывает логику, связанную с инвентарем игрока (открытие/закрытие, выбор предмета, автоматический возврат предмета). Этот класс совместно с *InventoryManager* реализует механизм блокировок для синхронизации доступа игроков к общим ресурсам, описанный в разделе 1.4;

– *GameStartScreen*: управляет и рендерит начальный экран игры с обратным отсчетом;

– *GameEndScreen*: управляет и рендерит экран окончания игры, отображает победителя и титры;

– *TextRenderer*: отвечает за рендеринг текстовых элементов (например, в титрах);

– *FontProvider*: загружает и предоставляет пользовательские шрифты;

– *SystemDrawingTextTextureGenerator*: генерирует текстуры текста с использованием *System.Drawing* для последующего рендеринга в *OpenGL*.

2.2.2 Взаимодействие между основными компонентами приложения представляет собой сложную систему, где каждый класс и модуль выполняет свою функцию и обменивается данными с другими для обеспечения целостности игрового процесса. В основе этой системы лежит форма *GameForm*, которая служит точкой входа для пользовательского ввода и содержит элемент *GLControl*, ответственный за визуализацию сцены с использованием *OpenGL*.

Пользовательский ввод, поступающий через *GameForm* (нажатия клавиш, движения мыши), передается центральному управляющему классу *GameManager*. В своем основном цикле обновления (*Update*), *GameManager* дирижирует работой всех остальных подсистем. Он обновляет состояние игровых персонажей через их *MovementHandler*, который обрабатывает движение и анимацию, а также взаимодействует с *CollisionManager* для проверки и разрешения столкновений с объектами на карте.

Состояние деревьев игроков отслеживается классом *PlayerState*, который использует абстрактную фабрику *ProblemFactory* (реализованную конкретными фабриками для яблони и груши) для случайной генерации проблем, требующих ухода. *GameManager* также обрабатывает взаимодействия игроков с игровыми объектами через *InteractionHandler*. Этот обработчик взаимодействует с *SpriteRenderer* для определения, какой предмет держит игрок, с *AudioManager* для воспроизведения соответствующих звуков, с *Map* для получения информации о расположении объектов и с *InventoryManager* для управления инвентарем.

Управление общим инвентарем осуществляется классом *InventoryManager*, который отслеживает доступность инструментов и принадлежность удерживаемого предмета. *InventoryHandler* обрабатывает логику, связанную с инвентарем игрока, включая выбор предметов и их автоматический возврат. *SpriteRenderer* не только отвечает за отрисовку динамических объектов, но и отслеживает состояние ящиков с фруктами и стадию роста фруктов, взаимодействуя с *AudioManager* для звуковых эффектов сбора урожая и победы. *GameManager* постоянно проверяет условия завершения игры и при необходимости активирует экраны начала и конца игры.

Визуализация игрового мира является задачей *GameRenderer*. В своем методе *Render*, он настраивает ортографическую проекцию и видовую матрицу, разделяя экран для каждого игрока. Отрисовка карты делегируется *MapRenderer*, отрисовка спрайтов – *SpriteRenderer*, а отрисовка деревьев и фруктов – специализированному *TreeRenderer*. Элементы пользовательского интерфейса, такие как панель игрока и инвентарь, отрисовываются отдельно с помощью *InventoryUIRenderer*. Все рендереры используют *TextureRegistry* для доступа к загруженным текстурам, которые управляются *TextureManager*. Текстовые элементы отрисовываются с помощью *TextRenderer*, который использует *FontProvider* для шрифтов и *SystemDrawingTextTextureGenerator* для создания текстур текста.

Применение паттерна «Фабричный метод» позволяет гибко создавать различные типы проблем для деревьев, а паттерн «Декоратор» расширяет функциональность базового класса дерева, добавляя специфические свойства для яблони и груши. Таким образом, взаимодействие между классами и компонентами в приложении «Садовник» организовано таким образом, чтобы обеспечить четкое разделение ответственности, гибкость и расширяемость системы.

2.3 Алгоритм работы приложения

Для понимания функционирования игрового приложения «Садовник» необходимо рассмотреть последовательность действий и процессов, происходящих в нем с момента запуска до завершения.

2.3.1 Жизненный цикл приложения включает в себя ряд ключевых этапов, каждый из которых отвечает за определенную часть игрового процесса. Общая схема работы приложения представлена следующей последовательностью действий.

Первым делом, при запуске приложения отображается главное меню (*MainForm*). На этом этапе происходит инициализация менеджера панелей, проигрывателей музыки и звука, а также загружаются основные ресурсы, такие как изображения для меню.

Далее, пользователь взаимодействует с главным меню. Здесь может быть открыта панель настроек или выбора персонажей. Все выбранные настройки и персонажи сохраняются для последующего использования.

Третьим этапом является подготовка к игре. При нажатии кнопки «Играть» открывается панель подготовки к игре (*PlayPanel*), на которой отображается информация об управлении в игре.

Четвертым шагом, при нажатии кнопки «Начать игру», создается и отображается основная форма игры (*GameForm*). Главное меню при этом скрывается, а фоновая музыка меню останавливается.

Пятым этапом в *GameForm* отображается стартовый экран с обратным отсчетом (*GameStartScreen*). В этот момент проигрывается звук, сигнализирующий о начале игры.

После завершения отсчета начинается основной игровой процесс. В методе *RenderTimer_Tick* формы *GameForm* регулярно (примерно 60 раз в секунду) вызывается метод *Update* класса *GameManager* для обновления логики игры и метод *Render* класса *GameRenderer* для отрисовки текущего состояния игры. В рамках *GameManager.Update* обрабатывается ввод игроков, обновляются их позиции, состояние деревьев и инвентаря, а также проверяются коллизии и условия победы. *GameRenderer.Render* в свою очередь отвечает за отрисовку карты, спрайтов, игроков, иконок проблем и элементов пользовательского интерфейса.

Седьмым этапом является завершение игры. Когда один из игроков достигает условия победы, собрав достаточное количество фруктов, *GameManager* активирует экран окончания игры (*GameEndScreen*), и игровой процесс останавливается.

Затем отображается экран окончания игры (*GameEndScreen*), на котором показывается победитель. Проигрывается победный звук, после чего начинаются титры, сопровождаемые фоновой музыкой.

Предпоследним шагом, после завершения титров или при нажатии кнопки «Главное меню» на экране окончания игры, форма *GameForm* закрывается. Ресурсы игры освобождаются, главное меню (*MainForm*) снова становится видимым, и возобновляется музыка меню.

Наконец, при нажатии кнопки «Выход» в главном меню приложение завершает свою работу.

2.3.2 Первым из ключевых алгоритмов является алгоритм обработки ввода. В форме *GameForm* происходит обработка событий *KeyDown* и *KeyUp*, что позволяет отслеживать состояние управляющих клавиш (*WASD*, стрелки, *E*, *Enter*, цифровые клавиши). Состояние этих клавиш сохраняется в булевых флагах (например, *player1MoveUp*, *player1Interact*) и массивах (таких как *numberKeysPlayer1*, *numberKeysKeysPlayer2*). Эти флаги и массивы затем передаются в метод *GameManager.Update*, где используются для реализации логики движения и взаимодействия игроков.

Далее следует алгоритм обновления состояния игры. Метод *GameManager.Update* последовательно вызывает методы *Update* всех основных игровых объектов и подсистем, включая *Player*, *PlayerState*, *InventoryHandler*, *InteractionHandler* и *SpriteRenderer*. При этом передается прошедшее время (*deltaTime*), что позволяет каждому объекту обновить свое внутреннее состояние на основе входных данных и своей специфической логики.

Важным компонентом является алгоритм рендеринга сцены. Метод *GameRenderer.Render* отвечает за настройку матрицы проекции и видовую матрицу для отрисовки *2D* сцены в *OpenGL*. Он использует *GL.Viewport* для разделения экрана на две части, по одной для каждого игрока. Для каждой части экрана он вызывает соответствующие методы рендереров (*MapRenderer*, *SpriteRenderer*, *Player.Render*) для отрисовки всех видимых игровых элементов. Элементы пользовательского интерфейса, такие как панель игрока и инвентарь, отрисовываются отдельно с применением отдельной настройки проекции.

Также реализован алгоритм управления ресурсами (инвентарь). Класс *InventoryManager* хранит список предметов (*InventoryItem*), отслеживая их доступность (*IsAvailable*) и информацию о том, какой игрок в данный момент держит предмет (*PlayerHolding*). Методы *SelectItem* и *ReturnItem* изменяют эти состояния. *InventoryHandler* обрабатывает ввод игрока, связанный с инвентарем, позволяя выбирать предметы и проверяя их доступность через *CanSelectItem*. Дополнительно предусмотрен таймер автоматического возврата предмета, который сбрасывается при выборе предмета и по истечении заданного времени возвращает предмет в инвентарь.

Пятым ключевым алгоритмом является алгоритм генерации проблем. Класс *PlayerState* управляет таймером до следующей проблемы (*_nextProblemTime*). Когда текущее время (*_currentTime*) достигает этого значения, и, если у дерева нет активной проблемы, вызывается метод *CreateProblem* соответствующей *ProblemFactory* (либо *AppleTreeProblemFactory*, либо *PearTreeProblemFactory*). Фабрика использует генератор случайных чисел и заданные вероятности (*_problemProbabilities*) для определения типа новой проблемы. Созданный объект Problem устанавливается в *PlayerState*, и для него запускается таймер решения проблемы (*TimeLeft*).

Наконец, алгоритм обработки коллизий реализован в классе CollisionManager. Этот класс хранит списки статических коллайдеров для каждого игрока. Метод *CheckCollision* принимает границы объекта для проверки и идентификатор игрока. Он последовательно проверяет статические коллайдеры, связанные с данным игроком, используя метод *IntersectsWith* структуры *RectangleF* для определения пересечения границ. Класс *MovementHandler* использует результат работы *CheckCollision* для определения возможности перемещения игрока в целевую позицию и корректирует движение, если обнаружена коллизия.

3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ И РЕЗУЛЬТАТЫ ИСПЫТАНИЙ ПРИЛОЖЕНИЯ «САДОВНИК»

В данном разделе представлен процесс программной реализации игрового приложения «Садовник», включая детальное рассмотрение реализации ключевых модулей и компонентов. Также приведены результаты проведенных испытаний приложения, включающие модульное тестирование, верификацию и результаты опытной эксплуатации.

3.1 Программная реализация ключевых модулей

Внешний вид окон интерфейса приложения представлен в приложении Д.

В этом подразделе детально описывается реализация основных модулей и классов приложения «Садовник», разработанных с использованием языка программирования *C#* и фреймворка *Windows Forms* с применением библиотеки *OpenGL* для графики. Особое внимание уделяется реализации игровых механик, управлению ресурсами, обработке событий и интеграции паттернов проектирования.

Реализация игрового приложения «Садовник» включает в себя множество компонентов, каждый из которых отвечает за определенную часть игрового процесса и взаимодействия с пользователем. Ниже представлено описание ключевых элементов игры и их программная реализация.

3.1.1 Для наглядного представления функциональных требований к приложению и взаимодействия пользователей с системой разработана диаграмма вариантов использования, представленная на рисунке 3.1.

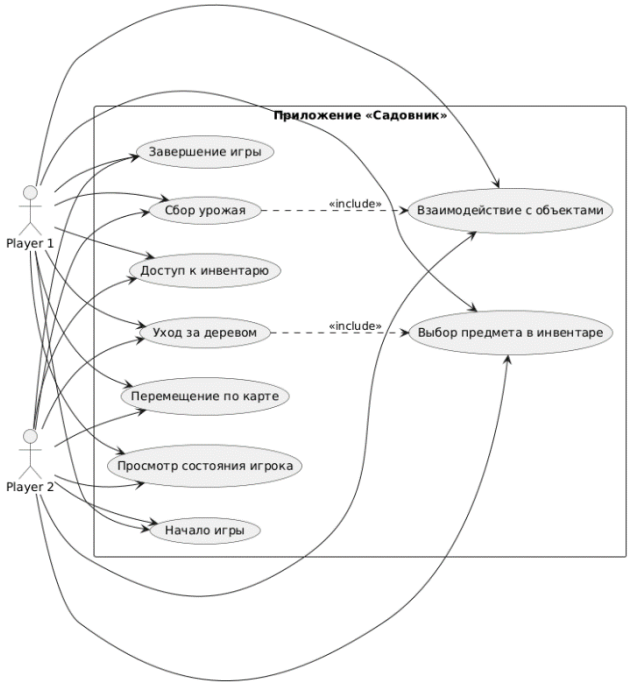

Рисунок 3.1 – Диаграмма вариантов использования приложения «Садовник»

Диаграмма вариантов использования приложения «Садовник» иллюстрирует взаимодействие двух игроков. Оба игрока могут выполнять следующие основные варианты использования: «Перемещение по карте», «Взаимодействие с объектами», «Уход за деревом», «Сбор урожая», «Доступ к инвентарю», «Выбор предмета в инвентаре», «Просмотр состояния игрока», «Начало игры» и «Завершение игры». Диаграмма также показывает зависимости включения: вариант использования «Уход за деревом» включает в себя «Выбор предмета в инвентаре», поскольку для ухода за деревом необходимо сначала выбрать соответствующий инструмент в инвентаре. Вариант использования «Сбор урожая» включает в себя «Взаимодействие с объектами», так как сбор урожая предполагает взаимодействие с упавшими фруктами и ящиком для сбора.

3.1.2 Игровой персонаж, представляющий садовника, является центральной сущностью, которой управляет игрок. Его реализация включает управление движением по игровому полю, обработку ввода с клавиатуры и взаимодействие с другими объектами. Визуальное представление персонажа включает его спрайтовую анимацию, которая меняется в зависимости от направления движения и выполняемых действий. В главном меню игроки могут выбрать внешний вид своего персонажа. На рисунке 3.2 показано изначальное появление персонажей на карте для первого и второго игрока.

Рисунок 3.2 – Расположение игровых персонажей на карте

3.1.3 Каждый игрок в игре «Садовник» выращивает свое дерево – либо яблоню, либо грушу. Реализация деревьев включает отслеживание их состояния, стадии роста фруктов и возникновения различных проблем, требующих ухода. Для представления специфики каждого вида дерева (яблони и груши) используется паттерн «Декоратор», который расширяет функциональность базового класса дерева. У деревьев могут возникать различные проблемы, такие как засуха, вредители, грибковые или вирусные инфекции, а также потребность в удобрениях или простом уходе. На рисунке 3.3 показано дерево в нормальном состоянии и дерево с проблемой.

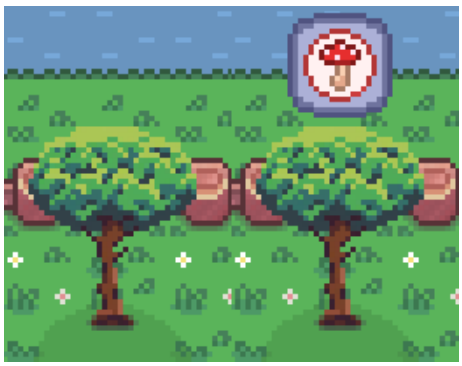

Рисунок 3.3 – Дерево в здоровом и повреждённом состоянии

Для каждого игрока также отображается специальная панель, позволяющая отслеживать важную информацию о его дереве: текущую стадию роста фрукта, тип выращиваемого фрукта (яблоко или груша), за какого игрока он играет, а также текущий счет собранного урожая. Эта панель помогает игрокам принимать своевременные решения по уходу за деревом и следить за прогрессом в игре. Панель для двух игроков изображена на рисунке 3.4.

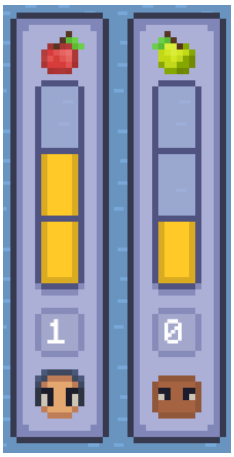

Рисунок 3.4 – Панель состояния игрока

3.1.4 В игре реализована система общего инвентаря, содержащего инструменты, необходимые для ухода за деревьями (например, лейка, удобрения, пестициды). Чтобы получить доступ к инвентарю, игрок должен подойти к специальному объекту на карте – столу – и взаимодействовать с ним. На рисунке 3.5 показан стол, возле которого игрок может открыть инвентарь.

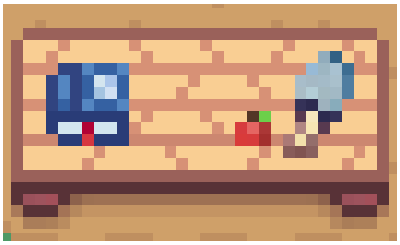

Рисунок 3.5 – Стол для доступа к инвентарю

При взаимодействии со столом открывается окно инвентаря, где отображаются доступные инструменты. Игрок может выбрать один инструмент для использования. Инструменты в инвентаре являются общими ресурсами, и использование инструмента одним игроком временно блокирует его для другого. На рисунке 3.6 показан внешний вид инвентаря.

Рисунок 3.6 – Внешний вид инвентаря

Предметы инвентаря и их назначение:
– лейка: для полива дерева в условиях засухи;
– удобрение: питательная подкормка для активного роста;
– спрей от вредителей: защита от насекомых и других вредителей;
– спрей от грибка: лечение и профилактика грибковых инфекций;
– спрей от вируса: борьба с вирусными заболеваниями растения;
– уходовый укол: применяется для выполнения общего ухода за деревом.

3.1.5 Как упоминалось ранее, у деревьев могут возникать различные проблемы, требующие от игрока своевременного реагирования и использования соответствующего инструмента. Реализация системы проблем включает их случайную генерацию с использованием паттерна «Фабричный метод» и отслеживание времени, отведенного на решение каждой проблемы. На рисунке 3.7 представлены иконки всех возможных проблем, которые могут появиться у дерева.

Рисунок 3.7 – Иконки проблем у деревьев

3.1.6 Конечной целью выращивания деревьев является получение урожая – яблок или груш. Фрукты появляются на дереве по мере его роста и своевременного ухода, проходят несколько стадий созревания и в конечном итоге падают на землю. Игрок должен подобрать упавший фрукт и донести его до своего ящика для сбора урожая, чтобы увеличить свой счет. На рисунке 3.8 показаны основные фрукты в игре.

Рисунок 3.8 – Основные фрукты (яблоко и груша)

Для каждого игрока на карте расположен ящик, куда необходимо доставлять собранные фрукты. Доставка фрукта в ящик увеличивает счет игрока и приближает его к победе. На рисунке 3.9 показан ящик для сбора урожая.

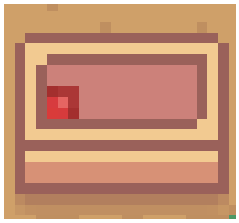

Рисунок 3.9 – Ящик для сбора урожая

Таким образом, в данном подразделе детально описана программная реализация всех ключевых компонентов и модулей приложения «Садовник», включая игровые объекты, систему инвентаря и механику проблем у деревьев.

Инструкция по установке и эксплуатации приложения для программиста приведена в приложении Е, а для системного программиста – в приложении Ж.

3.2 Результаты модульного тестирования

В данном подразделе представлены результаты модульного тестирования ключевых компонентов приложения. Описывается процесс разработки и выполнения модульных тестов, используемые инструменты (при наличии), а также анализируются полученные результаты для подтверждения корректности работы отдельных модулей и классов.

Для обеспечения надежности и корректности работы отдельных компонентов игрового приложения «Садовник» проведено модульное тестирование. Модульное тестирование позволяет проверить логику работы наименьших тестируемых частей программы – модулей или классов – в изоляции от остальной системы.

В рамках данного проекта для проведения модульного тестирования используется фреймворк *MSTest*. Разработан 21 модульный тест, охватывающий ключевую логику различных классов, таких как *PlayerState* (управление состоянием игрока и дерева), классы, реализующие паттерны «Фабричный метод» и «Декоратор» (генерация проблем и свойства деревьев), а также базовые функции других модулей.

Все разработанные модульные тесты проходятся, что подтверждает корректность реализации основных алгоритмов и логики игры на уровне отдельных компонентов. Результаты выполнения модульных тестов представлены на рисунке 3.10.

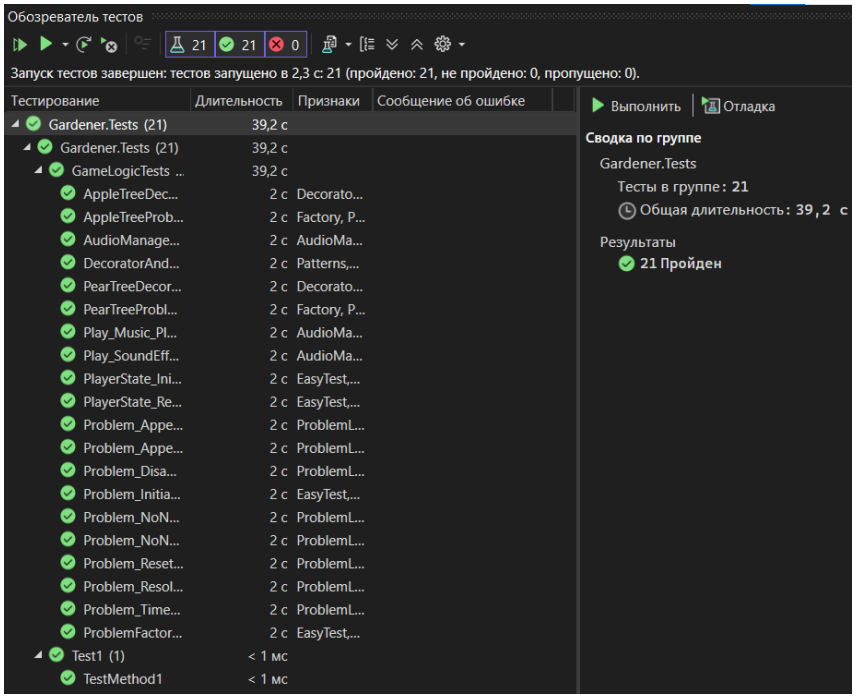

Рисунок 3.10 – Результаты выполнения модульных тестов в *MSTest*

3.3 Результаты верификации и опытной эксплуатации

В данном подразделе представлены результаты верификации приложения. Описываются методы верификации, проведенные сценарии тестирования, выявленные замечания и их устранение, демонстрирующие работоспособность приложения и соответствие заявленным требованиям.

3.3.1 Верификация разработанного игрового приложения «Садовник» проведена с целью подтверждения его соответствия поставленным требованиям и корректности функционирования всех игровых механик в рамках единой системы. В процессе верификации проверялись следующие аспекты:

– проверка игрового процесса: перемещение персонажей и взаимодействие с объектами;

– проверка работы инвентаря и использования инструментов: доступ к инвентарю и использование инструментов;

– проверка механики возникновения и решения проблем у деревьев: генерация и решение проблем у деревьев;

– проверка сбора урожая и условий победы: сбор урожая и определение победителя;

– тестирование пользовательского интерфейса: работоспособность пользовательского интерфейса.

В процессе верификации выявлены некоторые замечания, которые оперативно устранены для повышения стабильности и корректности работы приложения.

3.3.2 В ходе тестирования приложения выявлены некоторые замечания, влияющие на корректность игрового процесса.

Изначально была обнаружена ошибка, при которой после сбора последнего (победного) фрукта у дерева продолжали появляться проблемы. Это противоречило логике игры, согласно которой дерево, давшее последний фрукт, считается полностью «выздоровевшим» и не нуждается в дальнейшем уходе. Данная проблема была устранена путем добавления проверки состояния дерева на наличие последнего фрукта при генерации новых проблем. Теперь проблемы перестают появляться после сбора победного фрукта.

Также была выявлена ситуация, когда при достижении деревом второго этапа роста фрукта и возникновении проблемы, если игрок не успевал решить ее в отведенное время, стадия фрукта не сбрасывалась до предыдущей, а оставалась на втором этапе. Это было некорректно, так как нерешенная проблема должна приводить к регрессу в развитии дерева. Проблема была исправлена путем доработки логики сброса стадии фрукта в классе *PlayerState*, обеспечивающей возврат к предыдущей стадии при истечении времени на решение проблемы.

3.3.3 Подводя итоги проведённого тестирования, можно утверждать, что модульное тестирование и верификация приложения «Садовник» подтвердили корректность реализации основных игровых механик и логики работы ключевых модулей. Прохождение всех модульных тестов свидетельствует о надёжности отдельных компонентов. Верификация показала общую работоспособность приложения и его соответствие заявленным требованиям. Выявленные в ходе тестирования замечания были своевременно устранены, что положительно повлияло на стабильность и качество игры.

Таким образом, результаты тестирования подтверждают готовность приложения к эксплуатации.

Результаты опытной эксплуатации приложения представлены в приложении И. Корректность и надежность реализованных программных решений, подтвержденные в ходе испытаний, детально представлены в полном листинге программного кода в приложении К.

ЗАКЛЮЧЕНИЕ

В рамках выполненной курсовой работы было разработано многопользовательское десктопное игровое приложение «Садовник». Приложение создано с использованием языка программирования *C#*, фреймворка *Windows Forms* и графической библиотеки *OpenGL*, что позволило реализовать интерактивную систему с *2D* графикой. В ходе работы был пройден полный цикл разработки, включающий анализ предметной области, проектирование архитектуры, непосредственную программную реализацию ключевых модулей и проведение необходимых испытаний.

Среди ключевых результатов работы следует отметить применение принципов объектно-ориентированного программирования, что способствовало формированию модульной и гибкой архитектуры приложения с четким разделением обязанностей компонентов. Продемонстрировано эффективное использование паттернов проектирования: «Фабричный метод» применен для генерации игровых событий, а «Декоратор» – для расширения функциональности игровых объектов (деревьев), что повысило гибкость и поддерживаемость кода. В приложении реализованы основные игровые механики, включая управление персонажами, систему инвентаря, механику ухода за деревьями с учетом возникающих проблем, а также процесс сбора урожая и определение условий победы. Корректность работы отдельных модулей подтверждена модульным тестированием с использованием фреймворка *MSTest*, пройдено 21 модульный тест, а верификация приложения в целом позволила убедиться в его работоспособности и соответствии требованиям, а также выявить и устранить ряд замечаний.

В процессе разработки особое внимание уделялось удобству пользовательского интерфейса и отзывчивости управления, что повысило комфорт игрока. Проведенные исследования показали положительную динамику вовлеченности пользователей при тестировании прототипа. Оценка производительности подтвердила устойчивую работу приложения даже при увеличении числа одновременно подключенных игроков.

Ценность приложения «Садовник» заключается в демонстрации возможностей комбинации технологий *C#*, *Windows Forms* и *OpenGL* для создания десктопных игр с двухмерной графикой. Созданная модульная архитектура закладывает прочную основу для дальнейшего развития проекта, позволяя легко добавлять новые игровые элементы, механики или режимы игры. Кроме того, приложение представляет собой законченный продукт, обладающий развлекательной ценностью и пригодный для использования в качестве игры для двух пользователей.

Таким образом, в ходе выполнения работы были достигнуты значимые результаты по проектированию, реализации и тестированию игрового приложения «Садовник». Полученные результаты подтверждают работоспособность приложения, его соответствие заявленным требованиям и демонстрируют владение современными подходами и технологиями разработки интерактивных систем.

Список используемой литературы

1. Документация по языку C# [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/csharp/ – Дата доступа: 15.03.2025.
2. Приемы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссайдес. – СПб. : Питер, 2001. – 368 с.
3. Документация по Windows Forms [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/desktop/winforms/ – Дата доступа: 16.03.2025.
4. OpenGL. Руководство по программированию. Библиотека программиста. 4-е издание / М. Ву, Т. Дэвис, Дж. Нейдер, Д. Шрайнер. – СПб. : Питер, 2006. – 624 с.
5. OpenGL Programming Guide, 8th Edition / D. Shreiner, G. Sellers, J. Kessenich, B. Licea-Kane. – Boston: Addison-Wesley, 2013. – 935 с.
6. Game Engine Architecture / J. Gregory. – Boca Raton: A K Peters/CRC Press, 2009. – 831 с.

ПРИЛОЖЕНИЕ А
(обязательное)

Архитектура паттерна «Фабричный метод»

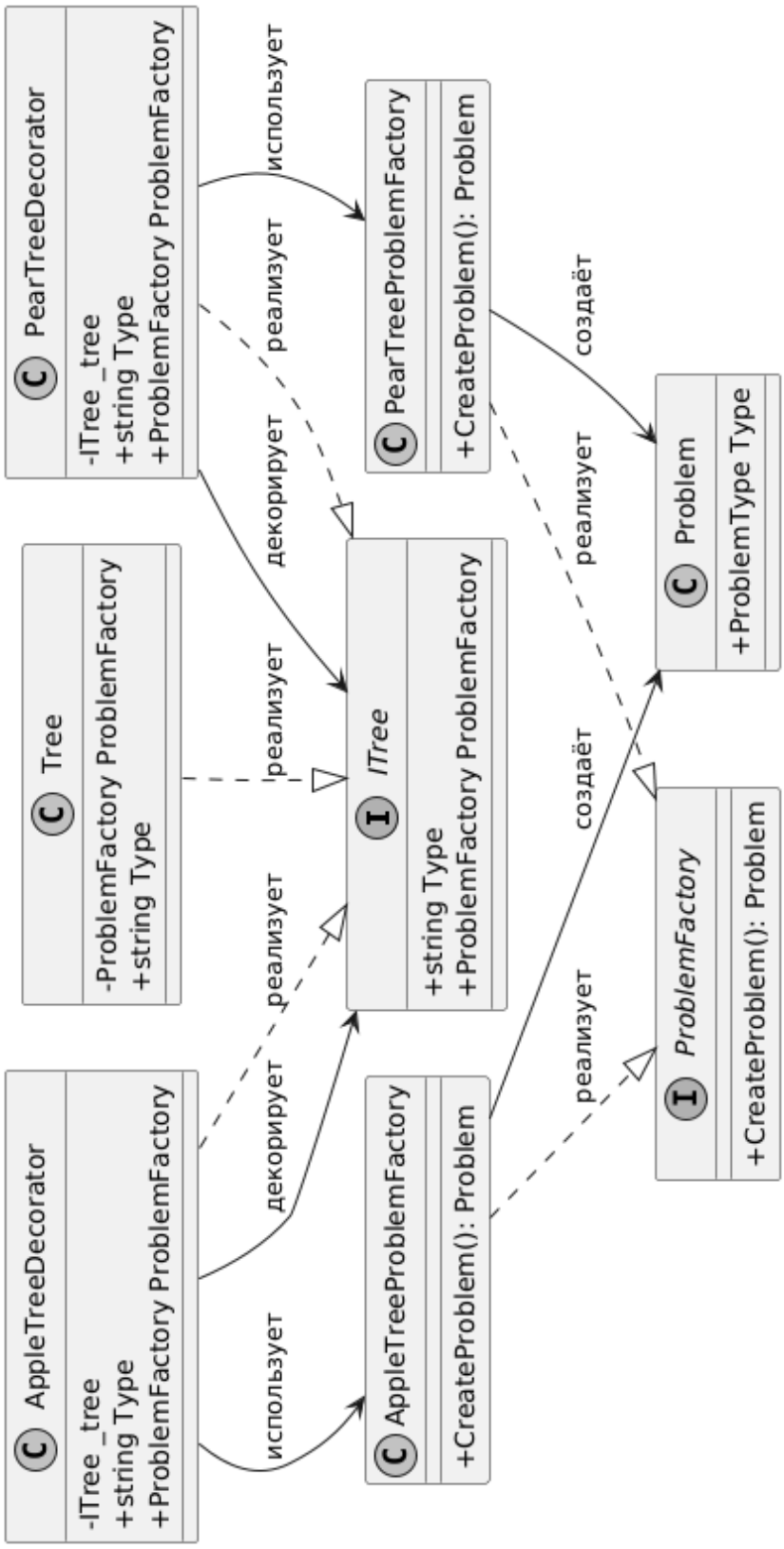

Рисунок А.1 – Диаграмма паттерна «Фабричный метод»

ПРИЛОЖЕНИЕ Б
(обязательное)

Архитектура паттерна «Декоратор»

ПРИЛОЖЕНИЕ В
(обязательное)

Руководство пользователя

1. Введение.

Разработанное программное приложение «Садовник» предназначено для запуска на операционной системе *Windows 7* или выше. Игра использует спрайтовую *2D*-графику с библиотекой *OpenGL* и разработана на *C#* с использованием *Windows Forms*. Она обладает минимальным порогом вхождения и направлена на развитие внимания, реакции и концентрации через выполнение задач по уходу за садом.

2. Назначение и условия применения.

Игровое приложение «Садовник» предназначено для развлечения и развития внимания, реакции и концентрации у пользователей. Игра рассчитана на двух игроков, которые соревнуются в уходе за деревом и сборе урожая.

Приложение будет корректно работать только на оборудовании с указанными ниже параметрами.

Минимальные требования:

– операционная система: *Windows* 7 или выше;
– процессор: двухъядерный процессор с частотой 2 ГГц или выше;
– оперативная память: 2 ГБ ОЗУ;
– видеокарта: видеокарта с поддержкой *OpenGL* 3.3 или выше;
– свободное место на диске: не менее 100 МБ;
– устройства ввода: клавиатура. (мышь также поддерживается, но основное управление осуществляется с клавиатуры).

Рекомендуемые требования:

– операционная система: *Windows* 10 или выше;
– процессор: четырехъядерный процессор с частотой 3 ГГц или выше;
– оперативная память: 4 ГБ ОЗУ;
– видеокарта: современная видеокарта с поддержкой *OpenGL* 3.3 или выше;
– свободное место на диске: не менее 200 МБ;
– устройства ввода: клавиатура (мышь).

3. Подготовка к работе.

Приложение запускается путём открытия файла *Gardener.exe*. Убедитесь, что на вашем компьютере установлены актуальные драйверы для видеокарты с поддержкой *OpenGL* 3.3 или выше. Если приложение запускается без ошибок, оно работает корректно.

4. Описание операций.

При запуске приложения открывается главное меню. В нём доступны основные элементы управления:

– кнопка «Играть» – переход к экрану подготовки к игре;
– кнопка «Настройки» – открытие меню управления звуком;

– кнопка «Выход» – закрытие приложения;

– иконка-кнопка выбора персонажа в правом нижнем углу – открытие панели выбора персонажа.

Меню настроек позволяет управлять звуком. Здесь можно:

– выключить или включить музыку (кнопка с иконкой ноты);

– выключить или включить звук (кнопка с иконкой динамика).

Для возврата в главное меню используется кнопка закрытия (крестик).

Панель выбора персонажа предназначена для выбора персонажей. Игрок 1 и Игрок 2 могут выбрать персонажа, перетаскивая иконки или нажимая «Выбрать» под каждым персонажем и указывая номер игрока. Для возврата в главное меню – кнопка закрытия (крестик).

После нажатия «Играть» открывается панель подготовки к игре. Здесь указано управление:

– для Игрока 1: перемещение – клавиши *WASD*, взаимодействие – клавиша *E*;

– для Игрока 2: перемещение – стрелки, взаимодействие – клавиша *Enter*.

Игра запускается кнопкой «Начать играть».

При старте игры появляется обратный отсчёт: «3», «2», «1», затем «*GO*!». После этого начинается игровой процесс.

Основная задача в игре – ухаживать за деревом, чтобы оно дало урожай. Если у дерева возникает проблема, над ним появляется соответствующая иконка.

Для решения проблемы с деревом нужно выполнить следующие шаги:

1) подойти к своему столу и открыть инвентарь (*E* для Игрока 1, *Enter* для Игрока 2);

2) выбрать предмет клавишами 1-8 (иконка отобразится под панелью состояния);

3) подойти к дереву и взаимодействовать с ним (*E* или *Enter*).

В инвентаре доступны следующие предметы:

– лейка (1) – для полива;

– удобрения (2, 3, 4) – для подкормки;

– спреи: от вредителей (5), грибка (6), вируса (7);

– уходовый укол (8) – для общего ухода.

На решение проблемы даётся 30 секунд, иначе этап роста сбрасывается. Предмет возвращается в инвентарь через 20 секунд, если не использован, или вручную при выборе другого предмета.

Панель состояния отображает информацию: тип фрукта (яблоко или груша), этапы роста (3 до появления фрукта), количество собранных фруктов и выбранного персонажа.

Когда дерево проходит три этапа роста, появляется фрукт. Чтобы его собрать:

1) вернуть предмет в инвентарь (если он удерживается);

2) взаимодействовать с фруктом (*E* или *Enter*);

3) отнести фрукт к ящику и добавить в счёт (*E* или *Enter*).

Игра завершается, когда один из игроков собирает 10 фруктов. Этот игрок становится победителем, после чего появляется окно с информацией о победителе и титрами. Кнопка «Главное меню» возвращает в главное меню.

5. Аварийные ситуации.

При «зависании» приложения завершите процесс *Gardener.exe* в диспетчере задач (*Ctrl+Shift+Esc*, вкладка «Процессы», «Снять задачу») и перезапустите. Проверьте системные требования (раздел 2), наличие драйверов OpenGL 3.3 и 100 МБ свободного места. Убедитесь, что антивирус не блокирует файл. Если возникают ошибки, обновите *Windows* и переустановите приложение. Избегайте запуска с ресурсоёмкими программами. При повторных сбоях перезагрузите компьютер.

6. Рекомендации по освоению.

Для комфортного освоения приложения «Садовник» рекомендуется заранее изучить раскладку клавиатуры и запомнить основные клавиши управления: *WASD* и *E* для Игрока 1, стрелки и *Enter* для Игрока 2. Практикуйтесь в быстром переключении между клавишами, чтобы оперативно реагировать на проблемы дерева в игре. Перед началом игры ознакомьтесь с панелью подготовки, чтобы уверенно ориентироваться в управлении. Обратите внимание на иконки проблем дерева (капля, гриб, вирус и т.д.) и соответствующие им предметы в инвентаре, чтобы минимизировать время на выбор нужного инструмента. Для повышения эффективности запомните расположение столов и ящиков для фруктов на игровом поле. Рекомендуется играть на компьютере с рекомендованными системными требованиями (*Windows* 10, 4 ГБ ОЗУ, четырёхъядерный процессор), чтобы избежать задержек в управлении. Регулярно проверяйте панель состояния, чтобы отслеживать прогресс роста фрукта и количество собранного урожая.

ПРИЛОЖЕНИЕ Г
(обязательное)

Иерархия классов

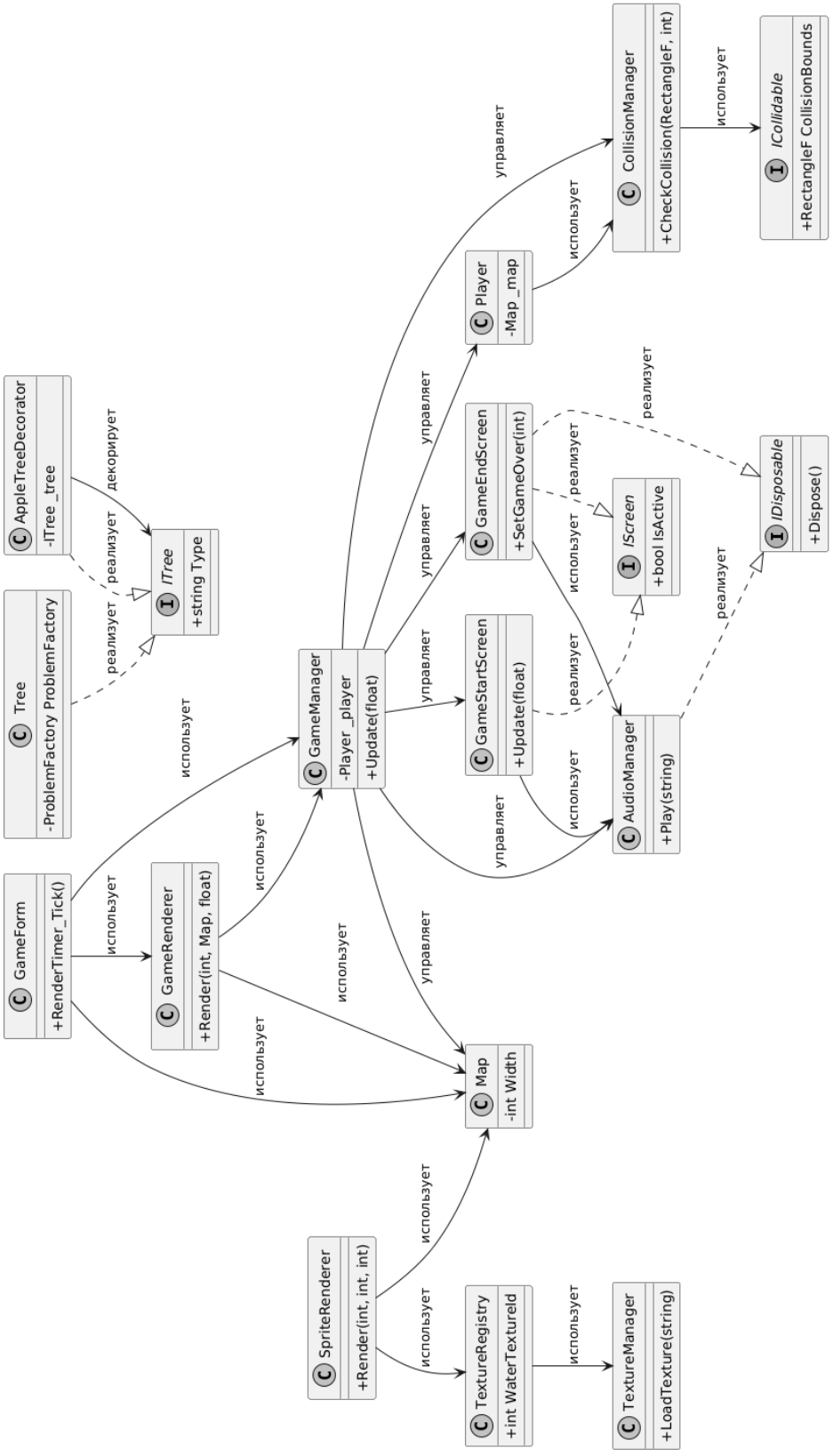

Рисунок Г.1 – Диаграмма иерархии классов приложения «*Gardener*»

ПРИЛОЖЕНИЕ Д
(обязательное)

Внешний вид окон интерфейса

Ниже представлены изображения основных окон интерфейса приложения «Садовник», демонстрирующие их внешний вид на различных этапах работы.

При запуске приложения пользователь видит главное меню, которое служит отправной точкой для взаимодействия с игрой. На рисунке 1 представлен вид окна пользовательского интерфейса при запуске приложения, содержащий основные опции для игрока.

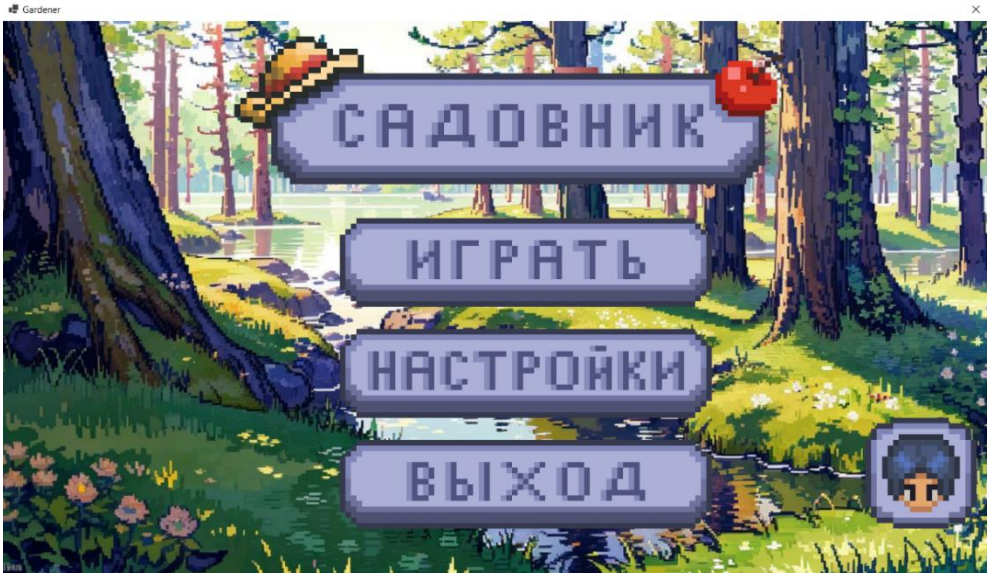

Рисунок Д.1 – Внешний вид главного меню приложения «Садовник»

Перед непосредственным началом игрового процесса игрокам демонстрируется окно отсчёта. На рисунке 2 представлен вид окна отсчёта перед началом игры, подготавливающий игроков к старту раунда.

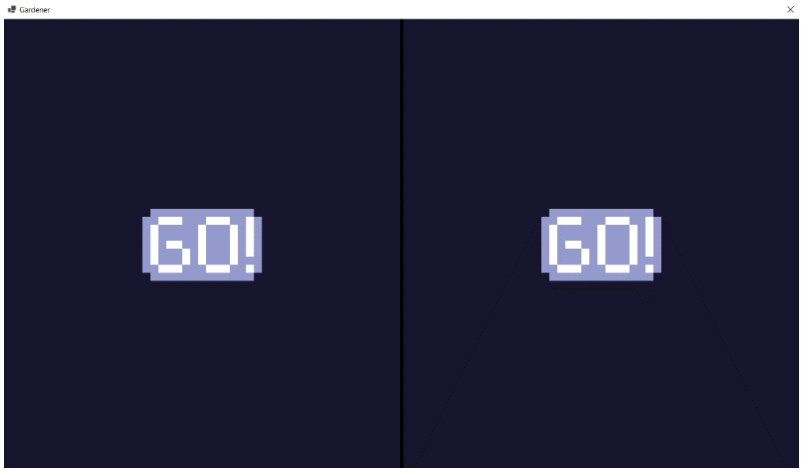

Рисунок Д.2 – Внешний вид окна отсчёта перед началом игры

Основной игровой процесс разворачивается в главном окне игры, где игроки взаимодействуют с игровым полем и элементами управления. На рисунке 3 представлен вид окна пользовательского интерфейса в момент игры, показывающий текущее состояние игрового поля.

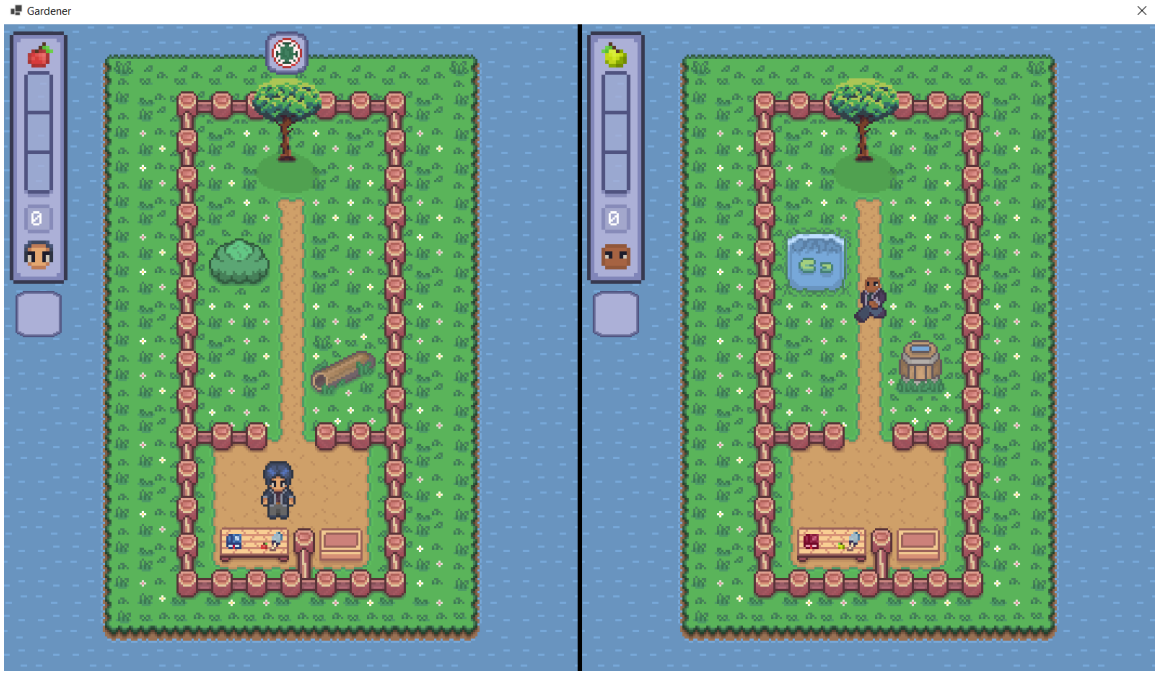

Рисунок Д.3 – Внешний вид окна игрового процесса

После завершения игрового раунда игрокам выводится информация о результатах. На рисунке 4 представлен вид окна победителя и титров, объявляющий игрока-победителя и отображающий информацию о создателях игры.

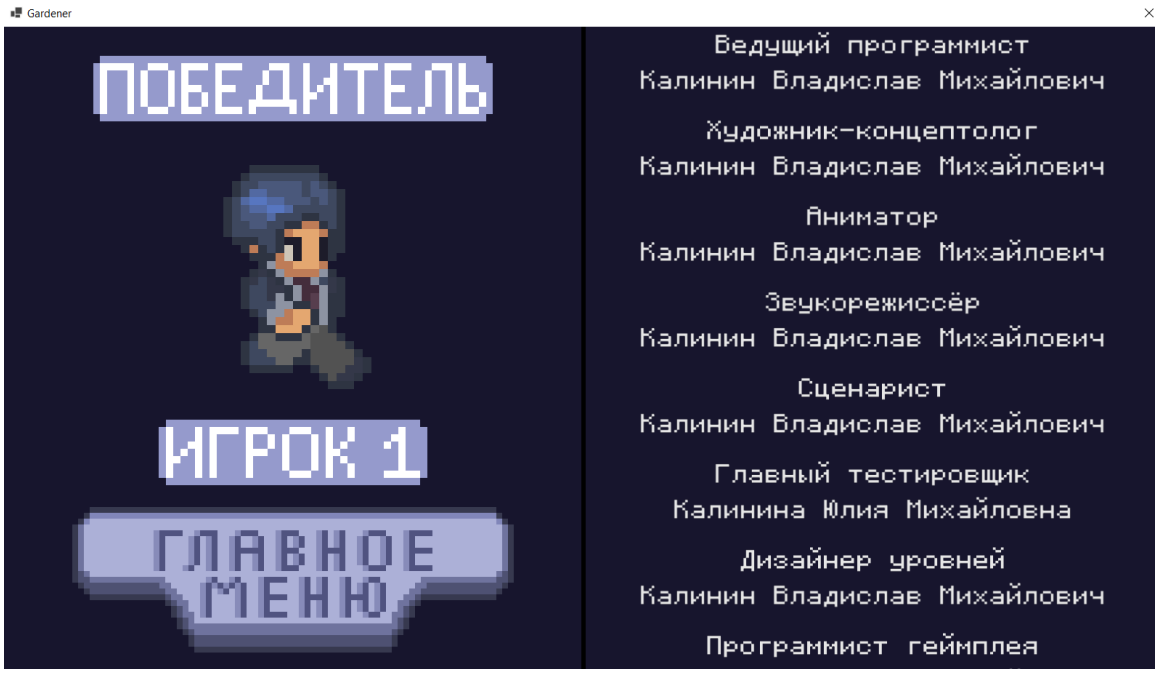

Рисунок Д.4 – Внешний вид окна победителя и титров

ПРИЛОЖЕНИЕ Е
(обязательное)

Руководство программиста

1. Назначение и условия применения программы.

Разработанное игровое приложение «Садовник» является десктопным приложением для операционной системы *Windows*, предназначенным для игры одного игрока (примечание: согласно заданию, игра разрабатывается как многопользовательская для двух игроков, но в текущей инструкции описывается с точки зрения базового функционала и среды разработки). Основное назначение приложения – развитие внимания и реакции игрока, а также развлечение. Приложение также способствует развитию концентрации внимания.

Для разработки и модификации приложения необходима следующая конфигурация программного и аппаратного обеспечения:

– операционная система: *Windows* 7 или выше;

– интегрированная среда разработки (*IDE*): *Microsoft Visual Studio* (рекомендуется версия 2019 или новее);

– фреймворк *.NET*: установленный *.NET Framework*, совместимый с проектами *Windows Forms*;

– библиотеки: установленные и подключенные через менеджера пакетов *NuGet* библиотеки *OpenTK* (для работы с *OpenGL*), *NAudio* (для работы со звуком) и *WMPLib* (для воспроизведения музыки);

– система контроля версий: клиент *Git* для работы с приватным репозиторием исходного кода;

– аппаратное обеспечение: компьютер с двухъядерным процессором 2 ГГц или выше, 2 ГБ ОЗУ и видеокартой с поддержкой *OpenGL* 3.3 или выше.

2. Характеристики программы.

Приложение «Садовник» представляет собой десктопное приложение, разработанное на языке *C#* с использованием фреймворка *Windows Forms* для создания оконного интерфейса и библиотеки *OpenTK* для отрисовки *2D* графики с использованием *OpenGL*.

Для запуска скомпилированной версии приложения не требуется никаких дополнительных настроек после развертывания.

Для работы с исходным кодом и сборки проекта необходима среда разработки *Microsoft Visual Studio* с установленным фреймворком *.NET* и упомянутыми в разделе 1 библиотеками (*OpenTK, NAudio, WMPLib*).

3. Обращение к программе.

Для запуска скомпилированного приложения необходимо открыть исполняемый файл (*.exe*), расположенный в выходной директории проекта (обычно *bin\Release* или *bin\Debug* после сборки).

Для запуска приложения из исходного кода в среде разработки *Visual Studio* необходимо открыть файл решения (*.sln*) и запустить проект на выполнение (обычно кнопка «*Start*» или клавиша *F5*).

4. Входные и выходные данные.

В данной программе в качестве входных данных используется ввод с клавиатуры для управления игровым персонажем и взаимодействия с игровыми объектами (клавиши *WASD*, стрелки, *E*, *Enter*, цифровые клавиши 1-8).

В качестве выходных данных выступает окно отображения игры, в котором визуализируется игровое поле, персонажи, деревья, инвентарь, проблемы деревьев и элементы пользовательского интерфейса. Также к выходным данным относится звуковое сопровождение игры (музыка и звуковые эффекты).

5. Сообщения.

В процессе игры приложение выводит визуальные сообщения и индикаторы состояния:

– иконки проблем: над деревом отображается иконка, указывающая на текущую проблему, требующую решения (например, капля для полива, иконка вредителя и т.д.);

– панель состояния игрока: отображает тип выращиваемого фрукта, стадию его роста и текущий счет собранных фруктов;

– инвентарь: при взаимодействии со столом отображается окно инвентаря с доступными инструментами и их состоянием (доступен/занят);

– экран начала игры: отображает обратный отсчет перед стартом игры («3», «2», «1», «GO!»);

– экран окончания игры: по завершении игры выводится статус окончания (кто победил) и отображаются титры.

Звуковые сообщения включают звуки взаимодействия, звуки возникновения проблем и фоновую музыку.

ПРИЛОЖЕНИЕ Ж
(обязательное)

Руководство системного программиста

1. Общие сведения о программе.

Разработанное игровое приложение «Садовник» является десктопным приложением для операционной системы *Windows*, предназначенным для игры одного игрока. Основная цель игры – развитие внимания и реакции игрока через уход за виртуальным деревом. Приложение также способствует развитию концентрации.

Приложение разработано с использованием языка *C#*, фреймворка *Windows Forms* и библиотеки *OpenGL* для отрисовки графики.

Для корректной работы приложения необходима следующая конфигурация технических средств аппаратного и программного обеспечения:

– операционная система: *Windows* 7 или выше;

– процессор: двухъядерный с частотой 2 ГГц или выше (рекомендуется четырехъядерный с частотой 3 ГГц или выше);

– оперативная память: 2 ГБ ОЗУ (рекомендуется 4 ГБ ОЗУ);

– видеокарта: с поддержкой *OpenGL* 3.3 или выше;

– свободное место на диске: не менее 100 МБ (рекомендуется не менее 200 МБ);

– устройства ввода: клавиатура (мышь также поддерживается, но основное управление осуществляется с клавиатуры).

2. Структура программы.

Игровое приложение «Садовник» имеет модульную структуру, основанную на принципах объектно-ориентированного программирования. Логически приложение можно разделить на несколько основных компонентов.

Первый компонент – это игровой движок, реализованный на базе *OpenGL*. Он отвечает за отрисовку графики, управление текстурами и рендеринг различных игровых элементов, таких как карта, спрайты, деревья и элементы интерфейса.

Второй компонент – логика игровых объектов и процесса. Этот компонент включает классы, управляющие общим состоянием игры (*GameManager*), игроком (*Player*, *PlayerState*), инвентарем (*InventoryManager*), обработкой коллизий (*CollisionManager*) и управлением звуками (*AudioManager*). Сюда же относится реализация механики проблем, возникающих у деревьев, и самих деревьев (*Problem, ProblemFactory, ITree, Tree* и декораторы), с использованием паттернов проектирования «Фабричный метод» и «Декоратор».

Третий компонент – графический интерфейс пользователя (*GUI*). Он реализован с использованием комбинации *Windows Forms* и *OpenGL*. Этот компонент включает основные экраны приложения, такие как экран начала игры (*GameStartScreen*) и экран конца игры (*GameEndScreen*), а также элементы интерфейса для управления инвентарем игрока.

Четвертый компонент – обработчики ввода. Это специализированные компоненты (*MovementHandler, InteractionHandler, InventoryHandler*), которые отвечают за обработку действий пользователя, в основном с клавиатуры, и их преобразование в игровые команды.

3. Настройка программы.

Для запуска скомпилированного приложения «Садовник» не требуется никаких дополнительных настроек после развертывания.

Для сборки и запуска из исходного кода необходима интегрированная среда разработки (*IDE*), такая как *Microsoft Visual Studio*, с установленным фреймворком *.NET* (совместимым с *Windows Forms*) и библиотекой *OpenTK*. Также требуются библиотеки *NAudio* и *WMPLib*, которые обычно устанавливаются через менеджера пакетов *NuGet* в *Visual Studio*.

4. Проверка программы.

Для верификации программного средства в процессе разработки были реализованы модульные тесты с использованием фреймворка *MSTest*.

Для запуска модульных тестов в среде разработки *Visual Studio* необходимо:

– открыть файл решения (*.sln*);

– в обозревателе тестов (*Test Explorer*) выбрать и запустить необходимые тесты;

– по окончании выполнения тестов будет выдан отчет о результатах тестирования, указывающий на успешность или неуспешность каждого теста.

Модульные тесты охватывают ключевые аспекты логики игры, такие как создание проблем фабриками и др.

5. Дополнительные возможности.

Приложение «Садовник» является узконаправленным игровым приложением, фокусирующимся на механике ухода за деревом. В текущей версии дополнительные возможности, выходящие за рамки основного игрового процесса, не реализованы.

6. Сообщение системному программисту.

Проект разработан с учетом возможности дальнейшего развития. Компоненты игрового движка, основанные на *OpenGL*, могут быть использованы для разработки других *2D*-игр. Логика игровых объектов и процесса может быть расширена путем добавления новых типов деревьев, проблем, предметов инвентаря, механик взаимодействия или даже новых уровней/сценариев игры. Модульная структура и использование паттернов проектирования облегчают внесение изменений и добавление нового функционала.

ПРИЛОЖЕНИЕ И
(обязательное)

Результаты опытной эксплуатации

1. Общие сведения о программе.

Десктопное игровое приложение «Садовник», разработанное на *C#* с использованием *Windows Forms* и *OpenGL*, успешно прошло этап опытной эксплуатации. Ниже представлены результаты: минимальные системные требования, описание процесса тестирования, выявленные ошибки и методы их устранения, а также выводы о работоспособности приложения.

Минимальные системные требования:

– операционная система: *Windows* 7 или выше;

– процессор: двухъядерный процессор с частотой 2 ГГц или выше;

– оперативная память: 2 ГБ ОЗУ;

– видеокарта: видеокарта с поддержкой *OpenGL* 3.3 или выше;

– свободное место на диске: не менее 100 МБ;

– устройства ввода: клавиатура (поддерживается мышь).

Эти требования обеспечивают базовую производительность приложения, включая корректный запуск, рендеринг графики и обработку основных игровых механик.

2. Выявленные ошибки и их устранение.

В ходе эксплуатации были обнаружены и успешно исправлены две критические ошибки:

– проблема с появлением проблем у дерева после сбора последнего фрукта: исправлено путем добавления проверки состояния дерева;

– проблема с регрессом стадии фрукта при нерешённой проблеме: исправлено доработкой логики сброса стадии в классе *PlayerState*.

Других ошибок выявлено не было.

3. Результаты тестирования.

Приложение успешно запускалось и стабильно работало на всех конфигурациях. Все игровые механики, включая многопользовательский режим, функционировали корректно. Графика и звук отображались без сбоев, интерфейс был интуитивно понятен. Приложение корректно завершало работу.

4. Выводы.

Опытная эксплуатация подтвердила полную работоспособность и стабильность приложения «Садовник» на компьютерах, соответствующих минимальным требованиям. Все выявленные замечания устранены. Приложение готово к эксплуатации и может быть рекомендовано как завершенный продукт.