МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Объектно-ориентированное программирование»

Тема: Создние цикла игры и сохранение игры

Студент гр. 4381 _____ Ишамчурин Д.И.

Преподаватель _____ Жангиров.Т.Р.

Санкт-Петербург

2026

Цель работы.

Доработать игру, добавив полноценный игровой цикл с разделением на этапы, систему сохранения и загрузки игры с возможностью продолжить с любого момента, а также обработку исключительных ситуаций при работе с файлами, чтобы программа не падала и сообщала понятную информацию об ошибках.

Задание.

На 6/3/1 баллов:

1. Создать класс(ы) игры, который реализует основной цикл игры, и которому передаются команды от пользователя. Игровой цикл состоит из следующих шагов:
    a. Начало игры
    b. Запуск уровня
    c. Ход игрока. Ход, атака или применение заклинания.
    d. Ход союзников - если имеются
    e. Ход врагов
    f. Ход вражеской базы и башни - если имеются

    Условие прохождения уровня студент определяет самостоятельно. Если игрок проигрывает, то игроку должно предлагаться начать заново игру, либо выйти из программы.

    Все взаимодействие должно происходить через классы игры.

2. Реализовать систему сохранения и загрузки игры. Пользователь должен иметь возможность сохранить игру в любой момент. Пользователь должен иметь возможность загружаться при запуске программы (или выбрать новую), либо во время игры. Сохранения должны оставаться в консистентном состоянии между запусками игры.
3. Добавить обработку исключительных ситуаций для загрузки/сохранения, например, невозможность записать в файл, нельзя загрузиться так как файл не существует или в нем некорректные данные.

Примечания:

- Класс игры может знать о игровых сущностях, но не наоборот
- При работе с файлом используйте идиому RAII.
- Исключения должны обязательно обрабатываться, и программа не должна завершаться
- Исключения должны быть информативными (содержать информацию о том, что и где произошло), на разные виды исключительных ситуаций должны быть свои исключения

Выполнение работы.

ОСНОВНАЯ СТРУКТУРА КЛАССОВ

После третьей лабораторной работы структура классов стала более организованной и получила новые возможности. Появился новый класс Game, который теперь управляет всем игровым процессом вместо прежнего GameController. В нем хранятся игровое поле, игрок, рука с заклинаниями, а также переменные состояния игры, текущий уровень и флаг работы. Класс Game содержит основной игровой цикл, меню для начала новой игры или загрузки, обрабатывает перемещение и применение заклинаний, проверяет условия победы и поражения, а также управляет загрузкой уровней.

Класс GameController из второй лабораторной был удален, его функциональность полностью перешла в Game. Теперь игра начинается с меню, где можно выбрать новую игру, загрузить сохранение или выйти. При новой игре загружается первый уровень с определенным набором врагов. После победы на уровне автоматически загружается следующий, а после прохождения всех выводится сообщение о победе.

Самое важное нововведение это система сохранения и загрузки. Появился отдельный класс SaveLoadManager, который отвечает за запись и чтение состояния игры в файл. В нем определены вложенные классы исключений

SaveException и LoadException для обработки ошибок при работе с файлами. Метод saveGame записывает в бинарный файл текущий уровень, состояние игрока, список врагов и количество заклинаний. Метод loadGame читает эти данные и восстанавливает игру. Также есть метод для проверки существования файла сохранения.

Классы Player и Enemy получили конструкторы по умолчанию, что необходимо для корректного чтения из файла. Теперь их можно создавать без параметров, а потом заполнять данными из сохранения.

В классе GameField ничего принципиально не изменилось, он по-прежнему хранит клетки, врагов и игрока, предоставляет методы для перемещения и отрисовки.

Классы заклинаний Spell, DirectDamageSpell и AreaDamageSpell остались без изменений, как и PlayerHand для управления рукой с заклинаниями. SpellFactory так же создает заклинания.

В классе Game появились новые методы handleSaveLoad для обработки сохранения и загрузки во время игры, loadLevel для загрузки конкретного уровня с нужным набором врагов, showMenu для отображения главного меню. Состояние игры теперь отслеживается через перечисление GameState, которое может принимать значения MENU, PLAYING, GAME_OVER и EXIT.

Файл main.cpp стал совсем простым он просто создает объект Game и запускает его метод run.

ВЗАИМОДЕЙСТВИЕ КЛАССОВ

Взаимодействие классов в третьей лабораторной работе строится вокруг центрального класса Game, который координирует работу всех остальных компонентов. Когда игрок запускает программу, создается объект Game, и вызывается его метод run, внутри которого запускается основной игровой цикл.

В этом цикле в зависимости от состояния игры либо показывается меню, либо происходит сам игровой процесс.

Если игрок выбирает новую игру в меню, класс Game вызывает свой метод loadLevel, который обращается к полю, чтобы разместить на нем игрока и врагов, а также наполняет руку игрока начальными заклинаниями через методы PlayerHand. При этом для создания заклинаний Game использует статические методы класса SpellFactory, который возвращает готовые объекты заклинаний.

Когда игрок перемещается по полю, нажатие клавиши обрабатывается в Game, который определяет направление и передает его в метод movePlayer объекта GameField. GameField в свою очередь проверяет клетки через объекты Cell, выясняя, есть ли там враг, и если есть, обращается к соответствующему объекту Enemy в своем векторе врагов, вызывая у него метод takeDamage для нанесения урона. Если враг погибает, GameField удаляет его из вектора и очищает клетку.

При применении заклинания игрок вводит цифру, Game обращается к объекту PlayerHand с просьбой использовать заклинание под указанным индексом. PlayerHand берет нужное заклинание из своего вектора и вызывает его виртуальный метод cast. В зависимости от реального типа заклинания DirectDamageSpell или AreaDamageSpell выполняется соответствующая логика. Заклинание через переданные ссылки обращается к GameField, получает доступ к вектору врагов и применяет к ним урон. После успешного применения PlayerHand удаляет использованное заклинание из своего вектора.

Если игрок решает сохранить игру, нажимая клавишу K, Game вызывает метод handleSaveLoad, который запрашивает имя файла и выбор действия. При сохранении Game собирает текущие данные уровень, состояние игрока, список врагов из GameField и количество заклинаний из PlayerHand и передает их в

статический метод saveGame класса SaveLoadManager. SaveLoadManager открывает файл и записывает все данные в бинарном формате, при возникновении ошибок выбрасывая исключение, которое Game перехватывает и выводит понятное сообщение.

При загрузке игры через клавишу L или из главного меню Game вызывает метод loadGame у SaveLoadManager, передавая имя файла. SaveLoadManager читает данные из файла и возвращает структуру GameData. Если при чтении возникает ошибка например, файл не существует или данные повреждены выбрасывается исключение, которое Game обрабатывает, сообщая игроку о проблеме и предлагая начать новую игру. Если загрузка прошла успешно, Game восстанавливает состояние: создает новое поле, размещает игрока в сохраненных координатах через placePlayer, добавляет всех врагов из сохранения через placeEnemy, и наполняет руку заклинаниями через PlayerHand.

После каждого хода игрока Game вызывает метод enemyTurn у GameField, который проходит по всем врагам в своем векторе, заставляя их двигаться в случайном направлении. При этом GameField проверяет клетки через Cell, и если враг натыкается на игрока, наносит ему урон через метод takeDamage объекта Player. Если игрок погибает, Game при следующей проверке состояния через checkGameState переводит игру в режим GAME_OVER.

Когда все враги убиты, GameField сигнализирует об этом через метод allEnemiesDead, и Game в методе checkGameState проверяет, есть ли еще уровни. Если есть, вызывается loadLevel со следующим номером, создавая новое поле и новый набор врагов. Если уровни закончились, игра завершается победой.

Таким образом, все взаимодействия завязаны на классе Game, который выступает посредником между пользовательским вводом, игровой логикой и системой сохранения. GameField управляет пространственным расположением

объектов, PlayerHand отвечает за коллекцию заклинаний, SaveLoadManager занимается файловыми операциями, а конкретные заклинания реализуют свою логику урона, не зная ничего об общем ходе игры.

ИГРОВОЙ ПРОЦЕСС

Игровой процесс после нововведений начинается с главного меню, где можно выбрать новую игру, загрузить сохранение или выйти. При старте новой игры загружается первый уровень, игрок получает начальные заклинания, а на поле появляются враги.

На каждом ходу игрок видит поле, свое здоровье и список доступных заклинаний. Он может перемещаться клавишами WASD, при этом если наступает на врага, автоматически атакует его. Также можно применить заклинание цифрой от 1 до 5, указав координаты цели. Заклинания бывают двух типов: прямого урона по одной клетке и урона по области два на два.

После хода игрока враги ходят случайным образом. Если враг наступает на игрока, он атакует и наносит урон. В любой момент можно сохранить игру клавишей K или загрузить клавишей L. При сохранении записываются текущий уровень, состояние игрока, все живые враги и количество заклинаний. При загрузке данные восстанавливаются из файла, а если файл поврежден или не существует, программа показывает ошибку и предлагает начать новую игру.

Если убиты все враги на уровне, автоматически загружается следующий уровень с более сильными противниками. Если уровень был последним, игра завершается победой. Когда здоровье игрока падает до нуля, игра заканчивается поражением и предлагает начать заново или выйти. Все ошибки при сохранении и загрузке обрабатываются через исключения, программа не падает, а выводит понятные сообщения и продолжает работу.

UML диаграмма: