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

ОТЧЕТ

по лабораторной работе №1
по дисциплине «Объектно-ориентированное программирование»
Тема: Создание игры с помощью классов

Студентка гр. 4381 _____ Гайнутдинова З. Р.

Преподаватель _____ Жангиров.Т.Р.

Санкт-Петербург
2025

Цель работы.

Создать игру с управлением игроком с помощью классов.

Задание.

1. Создать класс игрока, который должен хранить информацию об игроке (его жизни, урон, очки, и т.д. - студент сам определяет необходимые для работы характеристики). Объект класса игрока должен перемещаться по карте. Если у игрока кончаются жизни, то происходит конец игры.
2. Создать класс врага, который хранит параметры жизней и урона. Объектами класса врага управляет компьютер. При перемещении, если враг пытается перейти на клетку с игроком, то перемещение не происходит, и игроку наносится урон.
3. Создать класс квадратного/прямоугольного игрового поля, по которому перемещаются игрок и враги. Игровое поле не должно быть меньше 10 на 10 клеток, и не больше 25 на 25 клеток. Размеры поля задаются через конструктор. Рекомендуется для хранения информации об отдельных клетках поля создать отдельный класс. Реализовать конструкторы перемещения и копирования для поля, а также соответствующие операторы присваивания с копированием и перемещением (должна происходить глубокая копия).

4. Реализовать непроходимые клетки на поле. При попытке врагов или игрока перейти на такую клетку, перемещение не происходит. Заполнение поля непроходимыми клетками происходит в момент создания поля.

5. Добавить возможность для игрока переключаться на ближний или дальний бой с изменением значения наносимого урона. Такое переключение требует один ход.

6. Добавить класс вражеского здания. Такое здание размещается на карте, и раз в несколько ходов создает нового врага возле себя. Количество ходов до создания нового врага задается в конструкторе.

7. Реализовать замедляющие клетки на поле. Если игрок переходит на такую клетку, то он он не может двигаться на следующий ход.

Примечания:

Не забывайте для полей и методов определять модификаторы доступа

Для обозначения переменной, которая принимает небольшое ограниченное количество значений, используйте enum

Не используйте глобальные переменные

При реализации перемещения, не должно быть лишнего копирования

У поля не должно быть методов возвращающих указатель на поле в явном виде, так как это небезопасно

Информация о координатах игрока не должна дублироваться у игрока и у поля

Выполнение работы.

ОСНОВНАЯ СТРУКТУРА КЛАССОВ

Класс GameField (Игровое поле)

Управляет всей картой, объектами и их размещением. Содержит игровое поле с клетками, отслеживает позиции всех объектов и обеспечивает взаимодействие между ними.

Основные методы: initializeGrid() создает случайную карту с клетками разных типов, где 10% клеток становятся заблокированными, 10% замедляющими, а остальные 80% остаются обычными. Затем размещает игрока в левом верхнем углу и добавляет случайных врагов и здания. addRandomEnemies(count) добавляет указанное количество врагов в случайные позиции на карте, гарантируя что они не появятся в стенах, на игроке или других врагах. addRandomBuildings(count) размещает здания в случайных позициях, каждое с уникальным интервалом спавна врагов от 10 до 15 ходов, и делает эти клетки непроходимыми. setPlayerPosition(x, y, player) перемещает игрока на новые координаты, предварительно проверяя возможность движения и атакуя врага если он находится на целевой клетке. spawnEnemiesFromBuildings() проверяет все здания и создает новых врагов

вокруг них когда приходит время, находя свободные клетки в радиусе 8 направлений. getPlayerPosition() возвращает текущие координаты игрока. moveEnemies(player) и removeDeadEnemies() делегируют работу CombatManager. Методы проверки isCellPassable(), hasPlayer(), hasEnemy(), hasBuilding() проверяют состояние конкретных клеток.

Класс CombatManager

Отвечает за всю боевую механику игры - атаки, перемещение врагов и управление их состоянием.

Основные методы: attackEnemy(x, y, player) находит врага по координатам и наносит ему урон от игрока, добавляет очки если враг убит и выводит информацию о бое в консоль. attackPlayer(enemyIndex, player) позволяет врагу атаковать игрока, нанося урон основанный на характеристиках врага. moveEnemies(player) перемещает всех живых врагов в случайных направлениях, проверяя возможность движения и атакуя игрока при встрече. removeDeadEnemies() проходит по всем врагам и удаляет мертвых с карты и из всех списков, корректно обрабатывая сдвиг индексов при удалении. Вспомогательные методы isValidPosition(), isCellPassable(), hasPlayer(), hasEnemy(), hasBuilding() обеспечивают безопасную проверку состояний клеток.

Класс Game

Управляет всем игровым процессом, обрабатывает ввод пользователя, отображает состояние игры и координирует взаимодействие между игроком и игровым миром.

Основные методы: display() показывает текущее состояние игры включая номер хода, здоровье игрока, счет, тип оружия и возможность движения, затем отрисовывает карту символами где P - игрок, E - враг, B - здание, # - стена, ~ - замедляющая клетка. processPlayerMove(direction) обрабатывает ход игрока, проверяет возможность движения учитывая эффекты замедления, обрабатывает команды движения WASD и смены оружия на F. processEnemyTurns() запускает

ходы всех врагов и спавн новых врагов из зданий. play() главный игровой цикл который продолжается пока игрок жив, обрабатывает ввод команд, обновляет состояние игры и проверяет условия победы или поражения.

Класс Player (Игрок)

Представляет игрового персонажа с его характеристиками и возможностями.

Основные методы: getDamage() возвращает урон игрока в зависимости от выбранного типа оружия. takeDamage(amount) уменьшает здоровье игрока на указанное значение. addScore(points) увеличивает счет игрока. switchCombatType() переключает между ближним и дальним боем. canMoveNextTurn() и setMovement() управляют эффектом замедления от специальных клеток. resetMovement() сбрасывает ограничения движения в начале нового хода. isAlive() проверяет осталось ли у игрока здоровье.

Класс Enemy

Назначение: Представляет вражеских персонажей с их боевыми характеристиками.

Основные методы: takeDamage(amount) получает урон и уменьшает здоровье. isAlive() проверяет жив ли враг. getDamage() возвращает урон который наносит враг. getHealth() и getMaxHealth() предоставляют информацию о здоровье.

Класс Building

Представляет здания которые периодически спавнят врагов.

Основные методы: shouldSpawnEnemy() отслеживает интервалы спавна и возвращает true когда пришло время создать нового врага.

Класс Cell (Клетка)

Представляет одну клетку игрового поля с ее свойствами.

Основные методы: setType() и getType() управляют типом клетки (пустая, заблокированная, замедляющая). setPlayerPresence(), setEnemyPresence(), setBuildingPresence() и соответствующие методы проверки отслеживают наличие объектов на клетке. isPassable() определяет можно ли пройти через клетку.

ВЗАИМОДЕЙСТВИЕ КЛАССОВ

Игровая система состоит из нескольких классов, которые работают вместе. Главный класс Game управляет всей игрой. Он создает игровое поле GameField и игрока Player, а также обрабатывает команды от пользователя.

Все классы взаимодействуют через четкое разделение обязанностей: Game управляет процессом, GameField - картой, CombatManager - боем, а остальные классы представляют конкретные игровые объекты.

ИГРОВОЙ ПРОЦЕСС

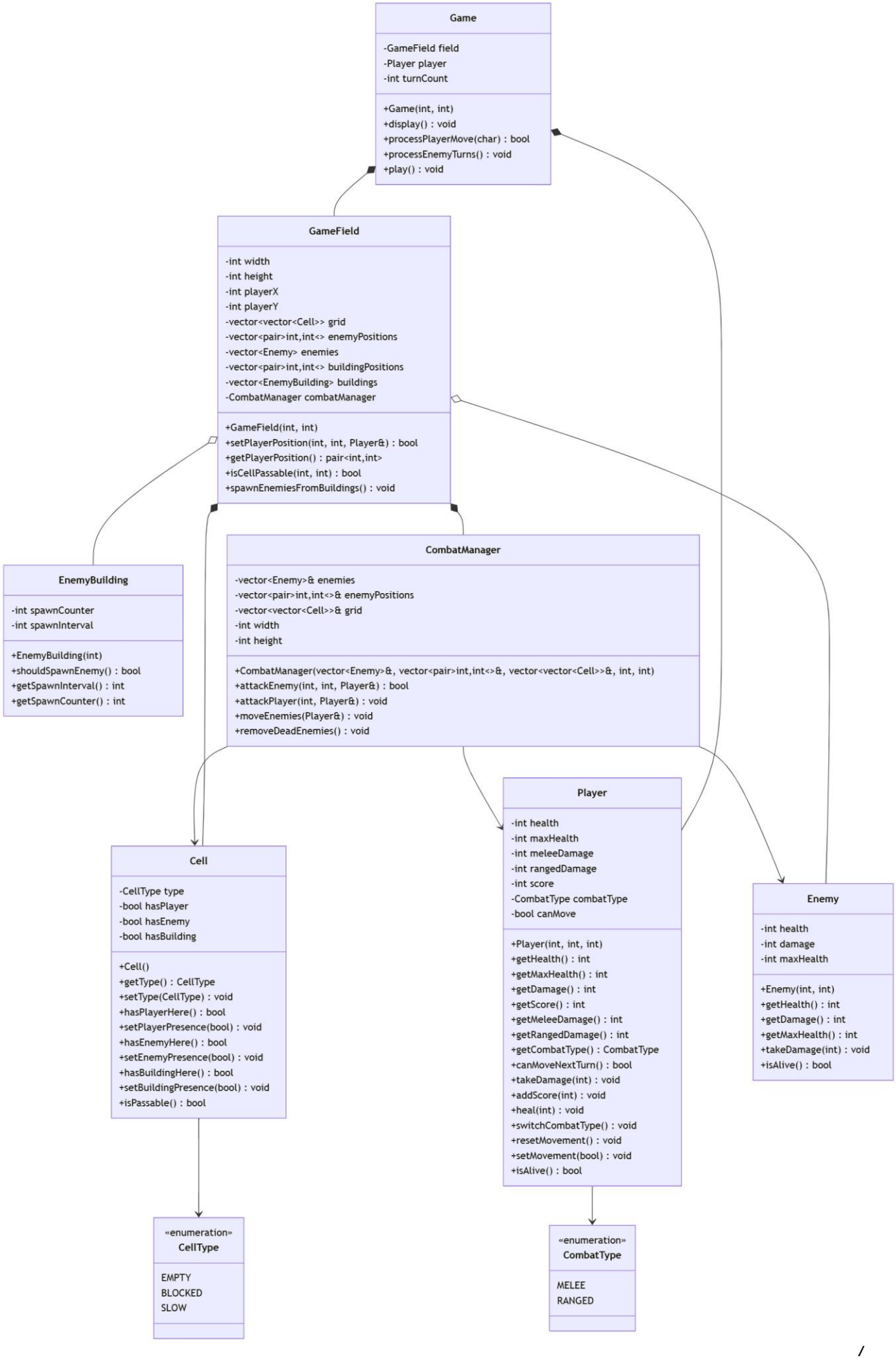
Игровой процесс представляет собой пошаговую стратегию с элементами выживания. Игрок управляет персонажем на случайно сгенерированной карте, состоящей из различных типов клеток - обычных, заблокированных и замедляющих.

В начале игры на карте размещаются враги и здания. Игрок может перемещаться по карте с помощью клавиш WASD. При встрече с врагом происходит автоматическая атака. Особые замедляющие клетки лишают игрока возможности двигаться в следующем ходу.

Игрок имеет два типа атаки - ближний и дальний бой, которые можно переключать клавишей F. Система подсчитывает набранные очки за побежденных врагов.

Игра продолжается до тех пор, пока игрок не погибнет от атак врагов или не решит выйти самостоятельно. Цель - продержаться как можно дольше и набрать максимальное количество очков.

UML диаграмма:


Game
-GameField field
-Player player
-int turnCount
+Game(int, int)
+display() : void
+processPlayerMove(char) : bool
+processEnemyTurns() : void
+play() : void
GameField
-int width
-int height
-int playerX
-int playerY
-vector<vector<Cell>> grid
-vector<pair>int,int<> enemyPositions
-vector<Enemy> enemies
-vector<pair>int,int<> buildingPositions
-vector<EnemyBuilding> buildings
-CombatManager combatManager
+GameField(int, int)
+setPlayerPosition(int, int, Player&) : bool
+getPlayerPosition() : pair<int,int>
+isCellPassable(int, int) : bool
+spawnEnemiesFromBuildings() : void
EnemyBuilding
-int spawnCounter
-int spawnInterval
+EnemyBuilding(int)
+shouldSpawnEnemy() : bool
+getSpawnInterval() : int
+getSpawnCounter() : int
CombatManager
-vector<Enemy>& enemies
-vector<pair>int,int<>& enemyPositions
-vector<vector<Cell>>& grid
-int width
-int height
+CombatManager(vector<Enemy>&, vector<pair>int,int<>&, vector<vector<Cell>>&, int, int)
+attackEnemy(int, int, Player&) : bool
+attackPlayer(int, Player&) : void
+moveEnemies(Player&) : void
+removeDeadEnemies() : void
Player
-int health
-int maxHealth
-int meleeDamage
-int rangedDamage
-int score
-CombatType combatType
-bool canMove
+Player(int, int, int)
+getHealth() : int
+getMaxHealth() : int
+getDamage() : int
+getScore() : int
+getMeleeDamage() : int
+getRangedDamage() : int
+getCombatType() : CombatType
+canMoveNextTurn() : bool
+takeDamage(int) : void
+addScore(int) : void
+heal(int) : void
+switchCombatType() : void
+resetMovement() : void
+setMovement(bool) : void
+isAlive() : bool
Enemy
-int health
-int damage
-int maxHealth
+Enemy(int, int)
+getHealth() : int
+getDamage() : int
+getMaxHealth() : int
+takeDamage(int) : void
+isAlive() : bool
Cell
-CellType type
-bool hasPlayer
-bool hasEnemy
-bool hasBuilding
+Cell()
+getType() : CellType
+setType(CellType) : void
+hasPlayerHere() : bool
+setPlayerPresence(bool) : void
+hasEnemyHere() : bool
+setEnemyPresence(bool) : void
+hasBuildingHere() : bool
+setBuildingPresence(bool) : void
+isPassable() : bool
«enumeration»
CellType
EMPTY
BLOCKED
SLOW
«enumeration»
CombatType
MELEE
RANGED