КУРСОВ ПРОЕКТ ПО ОБЕКТНО-ОРИЕНТИРАНО ПРОГРАМИРАНЕ

ПРОЕКТ Тема 2 – Подходова игра

Изготвил: Огнян Александров Марински

Специалност: Информационни Системи

Факултетен номер: 1MI0700338

## Описание и идея на проект

Играта симулира походови битки между две фракции – живи и немъртви, всяка със свои единици и главнокомандващи. Целта е да се създаде система, която позволява:

* Управление на единици (живи и немъртви)
* Създаване и избор на главнокомандващи
* Провеждане на дуели по зададени правила
* Възможност за използване на специални умения
* Запазване/зареждане на състоянието на играта

## Подход при изпълнието

Решението беше разделено на следните етапи:

1. **Анализ на видовете единици и параметри** – групиране по общи характеристики.
2. **Дизайн на архитектурата с ООП** – използване на наследяване и полиморфизъм.
3. **Имплементация на единици и командири** – поетапно изграждане с конструктори и методи.
4. **Реализация на базата и системата за създаване на единици** – с лимити и злато.
5. **Изграждане на механиките на дуела** – с поетапна симулация на битката.
6. **Поддръжка на състояние (Save/Load/Restart)** – сериализация във файл.
7. **Потребителски интерфейс през конзолно меню** – избиране на команди от играча.

## Кратко описание на класовете

1. Unit (абстрактен клас)

* Член-данни: name, health, maxHealth, attack, armor, armorType, mana, goldCost, isCommander, armor
* Функции: takeDamage(int), attackEnemy(Unit\*), isAlive(), print(), clone() …
* Използване: Основен интерфейс за всички единици.

1. Производни на Unit

* Archer, Dibbuk, Ghost, Ghoul, Healer, Infantry, Knight, Mage, Necromancer, Revenant, Skeleton, Zombie
* Всеки клас задава конкретни стойности за своите характеристики.

1. Commander : public Unit

* Абстрактен базов клас за командири
* Производни: Lich, DreadLord, DeathKnight, UndeadHunter, BladeDancer, Paladin
* Всеки има специален метод castSpell(std::vector<Unit \*> &ownArmy, std::vector<Unit \*> &enemyArmy, std::vector<Unit \*> &graveyard, const std::string &extraInfo = "") = 0; с различна логика.

1. Base

* Класът Base представлява база на дадена фракция (жива или немъртва) и отговаря за:
* Управление на злато
* Управление и съхраняване на единици (вкл. главнокомандващи)
* Проверка и създаване на единици според лимит
* Зареждане и запазване на главнокомандващи
* Функции: createUnit(string unitType), addGold(int) ...

1. GameEngine

* Класът GameEngine изпълнява ролята на централен мениджър на цялата игра. Той управлява:
* Инициализацията от конфигурационен файл
* Избора на единици от страна на играча
* Провеждането на дуелите
* Проследяването на резултатите (победени дуели)
* Зареждането и запазването на състояние на играта
* Член-данни: Base base1, Base base2, wonDuels, configPath, selectedCommander, selectedAllies
* Методи: selectBoss(), saveGame(), loadGame(), restart(), startDuel(), run()..

1. UnitFactory

UnitFactory е utility клас, който не съдържа член-данни, а предлага само статични методи. Той отговаря за:

* Създаване на обикновени и главнокомандващи единици от конфигурация
* Възкресяване или преобразуване на мъртви единици
* Зареждане на всички командири от конфигурационен файл

1. ConfigLoader

Класът ConfigLoader е отговорен за зареждане на начални настройки и конфигурационни данни, което позволява лесна промяна на игровите параметри без нужда от промяна в кода.

## Идеи за бъдещи подобрения

Балансиране на силите – по-фино изчисляване на атака/броня/лечения.

Мултиплейър режим – възможност за двама играчи с редуващи се ходове.

Система с нива/опит – единиците да трупат опит и отключват подобрения.

Мод система – играчът да добавя свои единици/командири