Министерство науки и высшего образования РФ
Федеральное государственное бюджетное учреждение высшего образования
“Тверской государственный технический университет”
(ТвГТУ)
Факультет информационных технологий
Программное обеспечение вычислительной техники

Теория алгоритмов

Отчёт по лабораторной работе №2
Разработка консольной игры в парадигме ООП на Python

Выполнил:
А. В. Малов
Группа:
Б.ПИН.РИС-20.05
Проверила:
преподаватель
кафедры ПО
Е. И. Корнеева

Тверь
2022

Содержание

Постановка задачи

Разработать консольную игру в парадигме объектно ориентированного программирования на языке программирования `Python`.

Ссылка на репозиторий с программами и отчёт написанный с помощью LaTeX: `https://github.com/andreymlv/tstu-computation-theory`

1 Дизайн игры

Центральным объектом в игре является игровое поле. На игровом поле есть следующие объекты:

1. База.
2. Юниты.
3. Ландшафт.

Игровое поле можно разделить на ячейки, которые будут хранить в себе игровые объекты. На текущий момент ячейка хранит в себе свои координаты в игровом поле, объекты базы, юнита, курсора и ландшафта.

Игровой курсор представляет собой интерфейсом взаимодействия игры и игрока.

Управление курсора возможно с помощью клавиатуры.

Чтобы играть необходим терминал, который поддерживает `ASCII` коды. Они необходимы для того, чтобы были видны цвета.

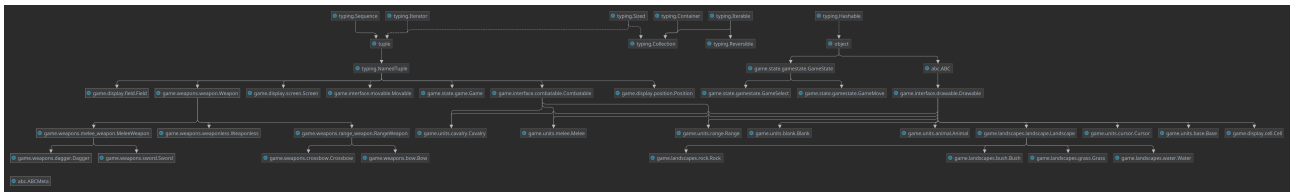

Рисунок 1: Диаграмма классов

2 Детали реализации

Как можно видеть из диаграммы классов, большинство объектов унаследованы либо от `NamedTuple` или декорируются с помощью `dataclass`ов.

По ходу написания реализации проект был полностью переписан, из-за изначально неверных архитектурных решений.

В итоге использовались следующие шаблоны проектирования и подходы реализации:

- Шаблон состояние. Выделили состояние “игра”, где управление клавиатурой переводит игру в новое состояние. Каждое новое состояние игры является новым объектом.
- Неизменяемые объекты. Поля объектов являются неизменяемыми во время исполнения.
- Декларативное программирование. В итоге наша реализация является композицией функций (считаем, что создание нового объекта тоже является функцией).

- Предпочитаем композицию вместо наследования.
- Не используем `None` объекты. Таким образом избегаем `NullPointerException`.
- Используем типизацию.
- Не используем статические методы.
- Не используем глобальные переменные.
- Не используем `mixin`ы. Так как в `Python` можно использовать абстрактные классы или интерфейсы (класс `ABC`).

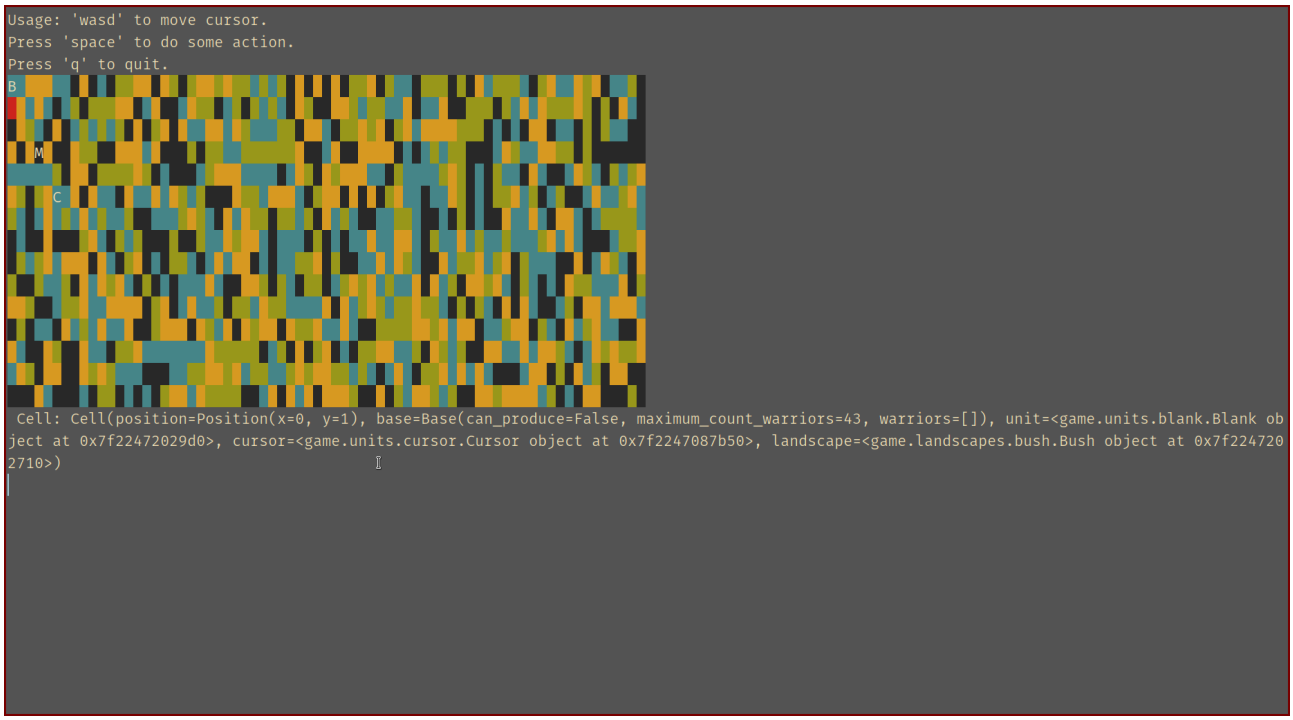

Рисунок 2: Запущенная игра

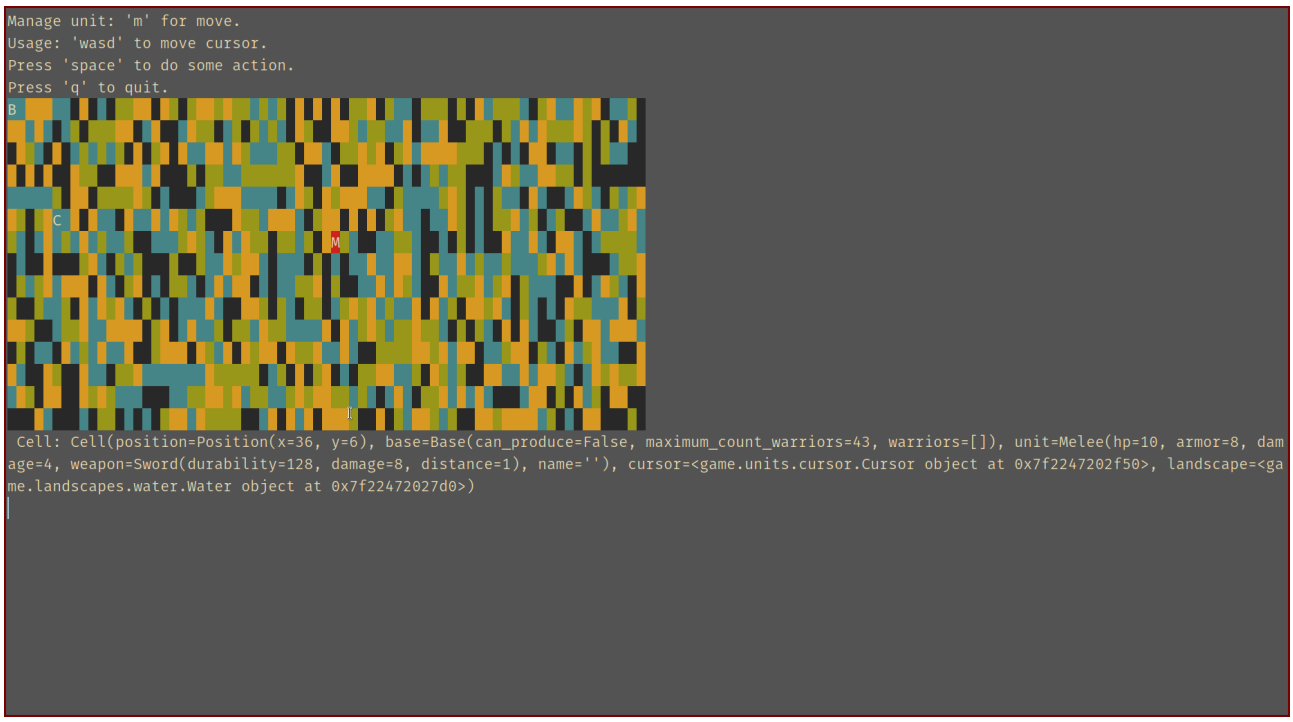

Рисунок 3: Запущенная игра

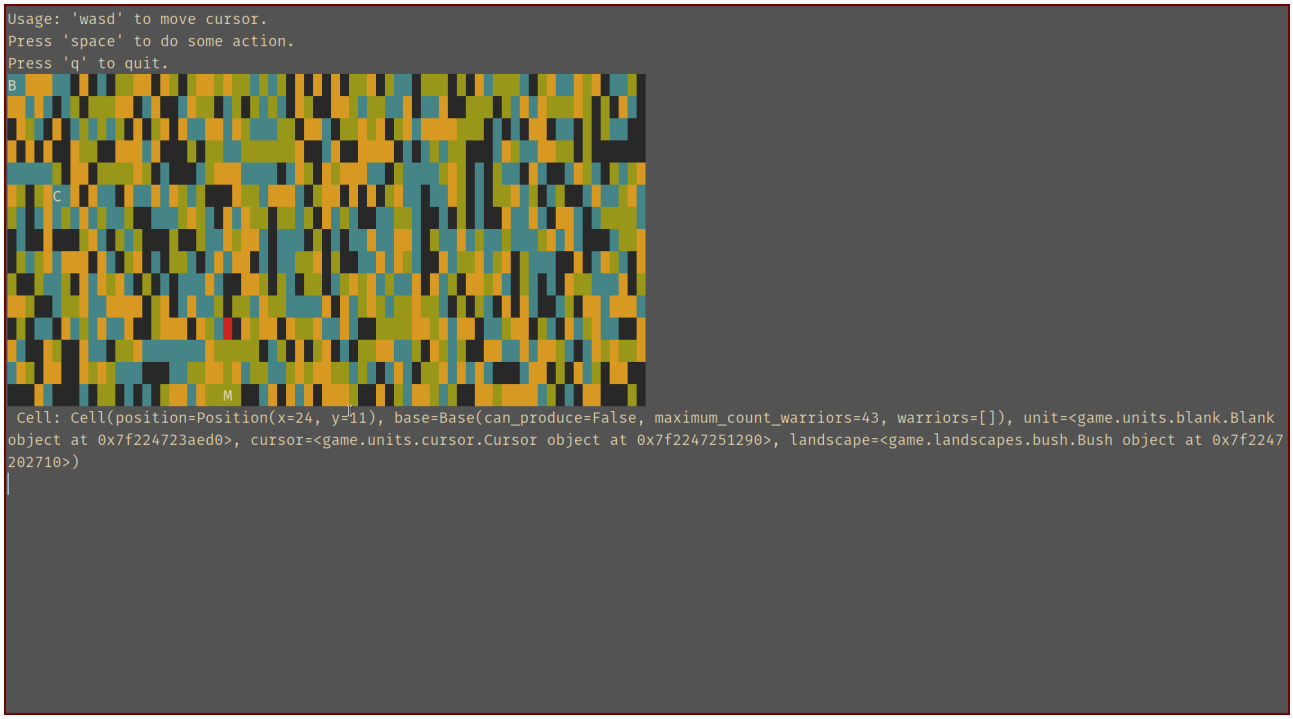

Рисунок 4: Запущенная игра

Главный скрипт, который инициализирует и запускает цикл игры выглядит примитивно. Сначала инициализируем игровое поле. Заполняем клетки игры объектами. Запускаем цикл игры: очищаем экран, печатаем текущее состояние игры, переводим игру в новое состояние (действия, которые независимы от ввода игрока), и затем снова переводим игру в новое состояние, где мы читаем ввод игрока.

```python
#!/usr/bin/sudo python3

import random

import colorama

from game.display.cell import Cell
from game.display.field import Field
from game.display.position import Position
from game.display.screen import Screen
from game.landscapes.bush import Bush
from game.landscapes.grass import Grass
from game.landscapes.landscape import Landscape
from game.landscapes.rock import Rock
from game.landscapes.water import Water
from game.state.game import Game
from game.state.gamestate import GameState
from game.units.base import Base
from game.units.blank import Blank
from game.units.cavalry import Cavalry
from game.units.cursor import Cursor
from game.units.melee import Melee
from game.units.range import Range
from game.utils import clear_screen, dimensions, flush_input
from game.weapons.bow import Bow
from game.weapons.sword import Sword


def main() -> None:
    colorama.init()
    clear_screen()
    print(colorama.Fore.RED + "Welcome to the game!" + colorama.Style.RESET_ALL)
    input("To start the game press any key")
    clear_screen()
    width, height = map(lambda d: d // 2, dimensions())
    # width: int = 5
    # height: int = 2
    cells: list[list[Cell]] = []
    landscapes: list[Landscape] = [
        Grass(),
        Bush(),
        Rock(),
        Water(),
    ]
    max_warriors = (width + height) // 2
    for x in range(width):
        for y in range(height):
            if y == 0:
                cells.append([])
            cells[x].append(
```

```
                Cell(
                    Position(x, y),
                    Base(False, max_warriors, []),
                    Blank(),
                    Blank(),
                    landscapes[random.randint(0, len(landscapes) - 1)],
                )
            )
    cells[0][0] = Cell(
        Position(0, 0),
        Base(
            True,
            max_warriors,
            [
                Melee(10, 8, 4, Sword(128, 8, 1)),
                Range(10, 8, 4, Bow(128, 8, 1)),
                Cavalry(10, 8, 4, Sword(128, 8, 1)),
            ],
        ),
        Blank(),
        Cursor(),
        cells[0][0].landscape,
    )

    # debug
    cells[3][3] = Cell(
        Position(3, 3),
        Base(
            True,
            max_warriors,
            [],
        ),
        Melee(10, 8, 4, Sword(128, 8, 1)),
        Blank(),
        cells[3][3].landscape,
    )
    cells[5][5] = Cell(
        Position(5, 5),
        Base(
            True,
            max_warriors,
            [],
        ),
        Cavalry(10, 8, 4, Sword(128, 8, 1)),
        Blank(),
        cells[5][5].landscape,
    )

    game: Game = Game(
        GameState(
```

```
            Field(Screen(width, height), cells),
            Position(0, 0),
            Position(0, 0),
            False,
        )
    )

    while not game.is_over():
        clear_screen()
        game.print()
        game = game.next()
        # Blocks game loop.
        game = game.poll()

    flush_input()
    colorama.deinit()


if __name__ == "__main__":
    main()
```

Листинг 1: main.py — главный скрипт игры

Заключение

Реализовали консольную игру, используя принципы объектно ориентированного программирования.