

2D MULTIPLAYER RPG GAME

Antonio Penev

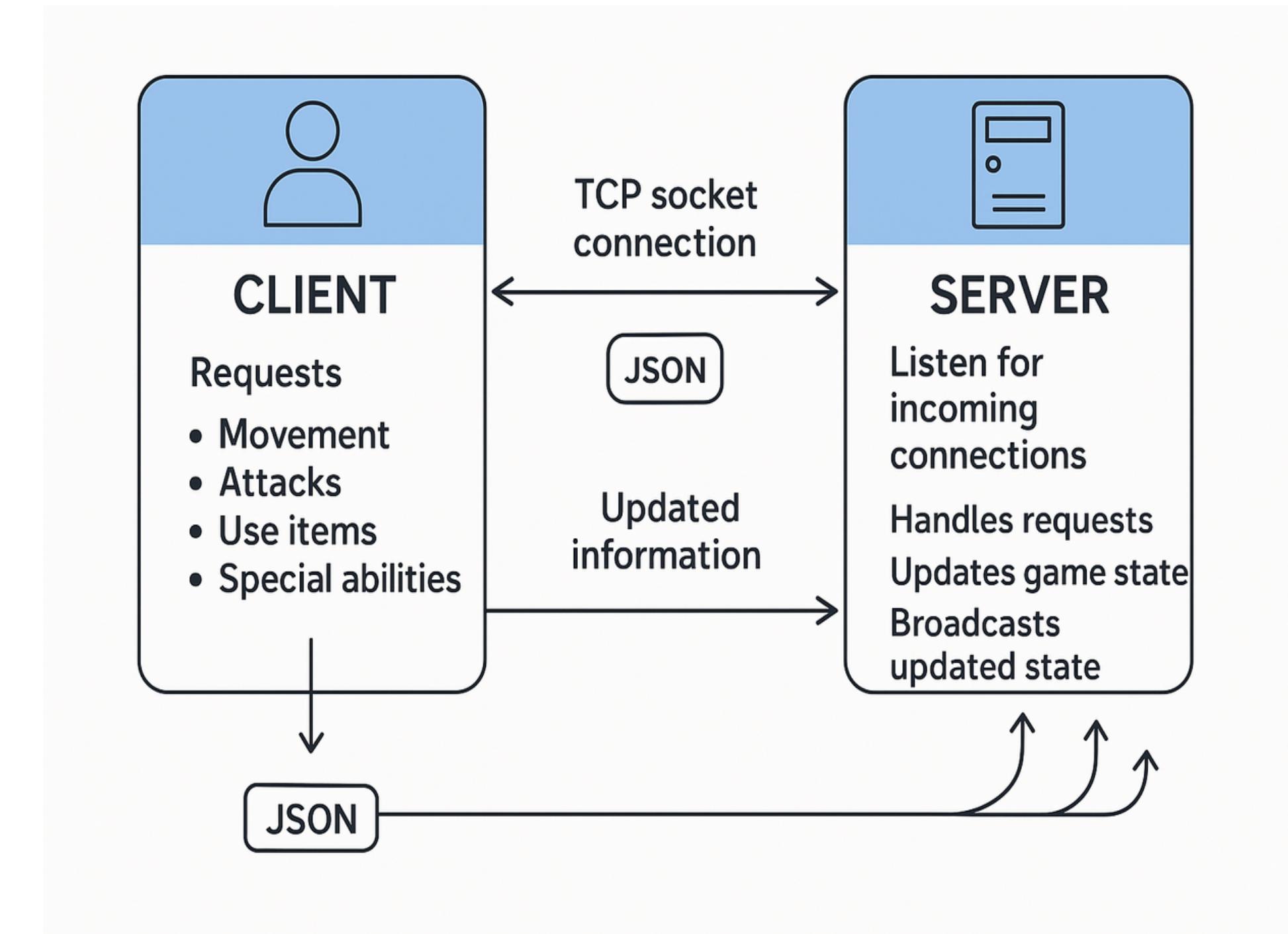
ОСНОВНА ИДЕЯ

Проектът представлява 2D multiplayer пиксел-арт ролева игра, реализирана чрез Python и библиотеките Pygame и Socket.



КОМПОНЕНТИ

- Сървър: Управлява връзките, състоянието на играта и синхронизация.
- Клиент: Отговаря за графика, управление на потребителския интерфейс, анимации и взаимодействия.



ИМПЛЕМЕНТАЦИЯ

- Използване на multithreading за паралелна обработка на връзки и обновяване на състоянието на играта.
 - 1 глобална сървърна нишка
 - N клиентски нишки
 - N сървърни нишки
- JSON комуникация за простота и ефективност.
- Design patterns

```
json.dumps({  
    "type": "...",  
    "data": {...}  
})
```

ТЕХНОЛОГИИ

- Работат за графика и ресурси за анимациите.
- Socket programming за мултиплейър комуникация.
- JSON за сериализация на данни.
- Използват се design patterns:
 - Factory (за създаване на heroes, enemies и items)
- Алгоритми:
 - Bresenham's line за проверка на видимостта и логика за преследване и атака.

КЛАСОВЕ

Server:

- Network
- GameState

Client:

- Game
- Hero
- Enemy
- Map
- Item
- Weapon
- Animation
- HomeScreen

SERVER

- **Network**
 - Приема нови клиенти.
 - Създава нишка за всеки клиент, която комуникира само с него.
 - Следи кога играчи се присъединяват, напускат или умират, и координира комуникацията между клиента и сървърната логика.
 - Използва broadcast() за изпращане на глобалното състояние до всички клиенти.
- **GameState**
 - Управлява логиката на самата игра – позициите и състоянието на играчите.
 - Съдържа всички данни за играчи, предмети, врагове, карта, ефекти.
 - Управлява движения, атаки, умения, предмети и др.
 - Осигурява централизирано и синхронизирано състояние на света, достъпно за сървъра, което се изпраща към клиентите.

Start the server

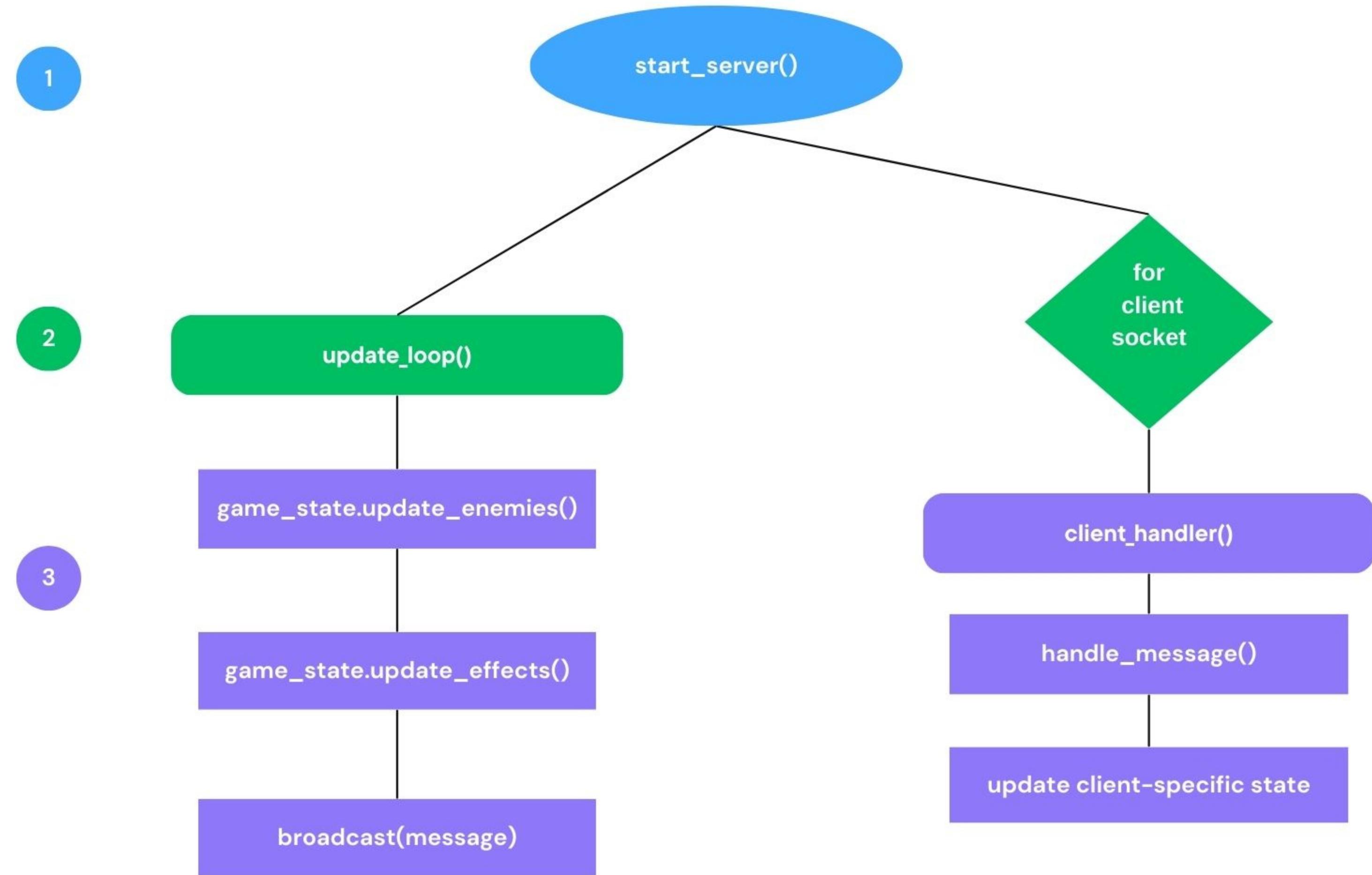
Antonio

Create the threads

Antonio

handle the main state and the client state

Antonio



CLIENT

- **Game**

- Обработва вход от потребителя
- Обновява състоянието на играта
- Визуализира всичко на екрана
- Комуницира със сървъра чрез NetworkClient.

- **NetworkClient**

- Отговаря за мрежовата комуникация
- Създава TCP връзка със сървъра
- Изпраща съобщения
- Приема входящи данни в отделна нишка





MAP

Мар класът отговаря за визуализиране и структуриране на игровата карта.

initialize_tiles()

tile_size: 64x64 пикселя

load_textures()

0: {"type": "grass", "passable": True},

1

ENEMY

Класът Enemy е базовият клас, който представя енемутиата в играта, всеки от които има следните функции:

- Движат се и взаимодействват с играча.
- Нанасят щети и получават щети.
- Зареждат се визуално с анимации.

Класове наследници: Goblin, Skeleton, Orc

`create_enemy()` - Използва Factory Pattern

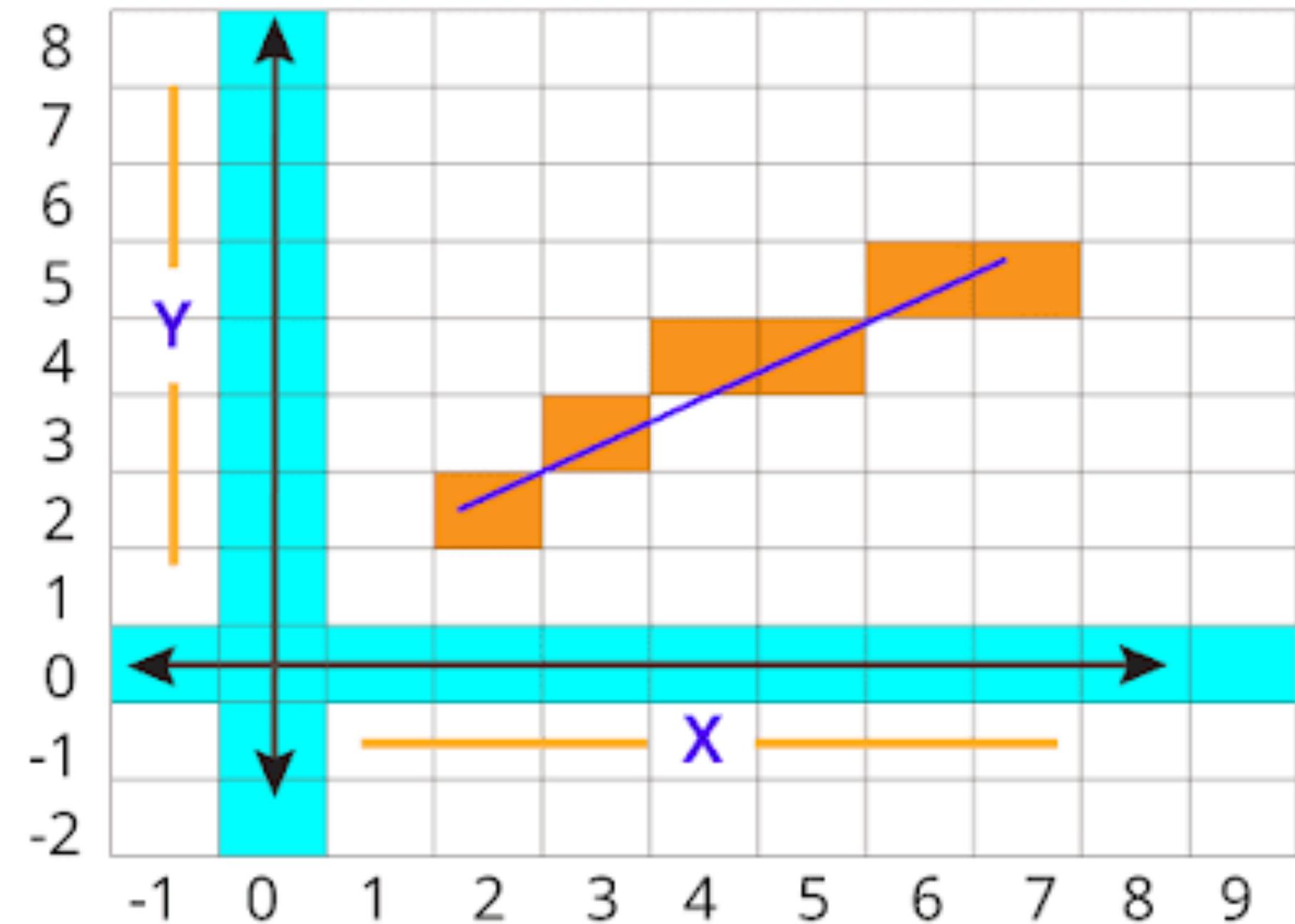


АЛГОРИТМИ

Алгоритъм на Брезенхам:

Използва, за да се определи дали между две точки (x_0, y_0) и (x_1, y_1) има непрекъсната видимост, т.е. няма стени между тях.

Функция:
`has_line_of_sight()`



HERO

Класът Hero е базовият клас, който представя героя ни. Той капсулира общото поведение, характеристики и визуализация на героя.

Всеки герой има специфични характеристики и умения.

Клasse наследници: Warrior, Archer, Mage

create_hero() - Използва Factory Pattern



ITEM

Класовете Item дефинират предметите в играта, които играчът може да използва, събира или да бъде засегнат от тях.

Всеки клас наследник override-ва функцията use().

Класове наследници: HealPotion, Shield, Coin и т.н.

create_Item() - Използва Factory Pattern



THANK YOU