AXEL GUIONNET - JULIEN RODRIGUES

SOUTENANCE
PROJET IA

AXEL GUIONNET - JULIEN RODRIGUES

SOMMAIRE

2- FSM

POUR LE PREMIER ENNEMI, NOUS AVONS UTILISÉ LE FSM. C'EST L'ENNEMI ORANGE.

```
void Enemy::patrol(Grid& grid) {

    static int currentWaypoint = 0;
    static sf::Vector2f waypoints[4] = /*{ sf::Vector2f(100, 300), sf::Vector2f(500, 100), sf::Vector2f(100, 3
        sf::Vector2f(initialPos.x - 150, initialPos.y + 150), sf::Vector2f(initialPos.x + 150, initialPos.y -
    sf::Vector2f target = waypoints[currentWaypoint];
    sf::Vector2f direction = target - pos;
    float distance = std::sqrt(direction.x * direction.x + direction.y * direction.y);
    if (grid.getCell(pos.x / CELL_SIZE, pos.y / CELL_SIZE).walkable == true) {
        if (distance < 5.0f) {
            currentWaypoint = (currentWaypoint + 1) % 4;
        }
        else {
            direction /= distance;
            pos += direction * 1.7f;
            e_direction = direction;
        }
        //shape.move(direction * SPEED);

    }


}
```

```
void Enemy::chase(Vector2f playerPos) {
    Vector2f direction = playerPos - pos;
    float distance = sqrt(direction.x * direction.x + direction.y * direction.y);

    if (distance > 0) {
        direction /= distance;
        pos += direction * 2.0f;
        e_direction = direction;
    }

    shape.move(direction*SPEED);
}
```

```
void Enemy::update(float deltaTime, Grid& grid, std::vector<Entity*> players, sf::Vector2f playerPos) {
    shape.setPosition(pos);
    switch (currentState) {
    case PATROL: {


        patrol(grid);
        if (detectPlayer(players[0]->pos)) {
            lastPlayerPosition = players[0]->pos;
            currentState = CHASE;
        }
        break;
    }


    case CHASE: {

        chase(players[0]->pos);
        if (!detectPlayer(players[0]->pos)) {

            currentState = PATROL;
        }
        break;
    }


    case SEARCH:
        search(lastPlayerPosition, deltaTime);
        break;
    }
```

3- A*

POUR LE DEUXIÈME ENNEMI, NOUS AVONS UTILISÉ LE FSM. C'EST L'ENNEMI BLEU.

```
void A_Ennemy::Path(Vector2i start, Vector2i end, Pathfinding& path, Grid& grid) {
    if (start != previousStart || end != previousEnd) {
        currentPath.clear();
        currentPathIndex = 0;
        previousStart = start;
        previousEnd = end;
    }

    if (currentPath.empty()) {
        currentPath = path.findPath(grid, start, end);
        currentPathIndex = 0;
        reversePath = false;
    }

    if (!currentPath.empty() && currentPathIndex < currentPath.size()) {
        Vector2i target = currentPath[currentPathIndex];
        Vector2f targetPos(target.x * CELL_SIZE, target.y * CELL_SIZE);
        Vector2f direction = targetPos - pos;

        float distance = sqrt(direction.x * direction.x + direction.y * direction.y);

        if (distance < 5.0f) {
                currentPathIndex++;
        }
        else {
            direction /= distance;
            pos += direction * 2.0f;
        }
    }
    shape.setPosition(pos);
```

4- BT

POUR LE TROISIÈME ENNEMI, NOUS AVONS UTILISÉ LE BT. C'EST L'ENNEMI VERT.

```
NodeState ActionNode::execute(Grid& grid, Blackboard& blackboard, sf::RectangleShape& shape, std::vector<Entity*> players, std::vector<std::shared_ptr<Projectile>>& projectiles) {
    if (actionName == "shoot") {
        projectiles.emplace_back(std::make_shared<Projectile>(sf::Vector2f(players[0]->pos.x + players[0]->shape.getSize().x/2,
            players[0]->pos.y + players[0]->shape.getSize().y / 2), sf::Vector2f(blackboard.pos.x + shape.getSize().x/2, blackboard.pos.y + shape.getSize().y/2)));
    }
    if (actionName == "movement") {
        static int currentWaypoint = 0;
        static sf::Vector2f waypoints[4] = { sf::Vector2f(blackboard.initialPos.x - 150, blackboard.initialPos.y - 150), sf::Vector2f(blackboard.initialPos.x + 150, blackboard.initialPos.y + 150),
        sf::Vector2f(blackboard.initialPos.x - 150, blackboard.initialPos.y + 150), sf::Vector2f(blackboard.initialPos.x + 150, blackboard.initialPos.y - 150) };
        sf::Vector2f target = waypoints[currentWaypoint];
        sf::Vector2f direction = target - shape.getPosition();
        float distance = std::sqrt(direction.x * direction.x + direction.y * direction.y);

        if (distance < 5.0f) {
            currentWaypoint = (currentWaypoint + 1) % 4;
        }
        else {
        direction /= distance;
        //blackboard.pos += direction * 0.7f;
        //shape.setPosition(shape.getPosition() + direction * 1.7f);
        }

        shape.move(direction * 2.5f);

        if (shape.getGlobalBounds().intersects(grid.getCell(shape.getPosition().x / CELL_SIZE, shape.getPosition().y / CELL_SIZE).shape.getGlobalBounds())) {
            if (!grid.getCell(shape.getPosition().x / CELL_SIZE, shape.getPosition().y / CELL_SIZE).walkable) {
                blackboard.wallCollision = true;
                //shape.move(-direction * 1.7f);
            }
            else {
                blackboard.previousPos = shape.getPosition();
                blackboard.wallCollision = false;
            }
        }
    }
    if (actionName == "getAway") {
        sf::Vector2f direction = -(blackboard.target - shape.getPosition());
        float distance = std::sqrt(direction.x * direction.x + direction.y * direction.y);

        if (distance < 5.0f) {
```

MERCI DE NOUS AVOIR ÉCOUTÉ