Urbex Mortel

ARNAUD Ivan, THOMAS Virginie et TONNIS Dorian

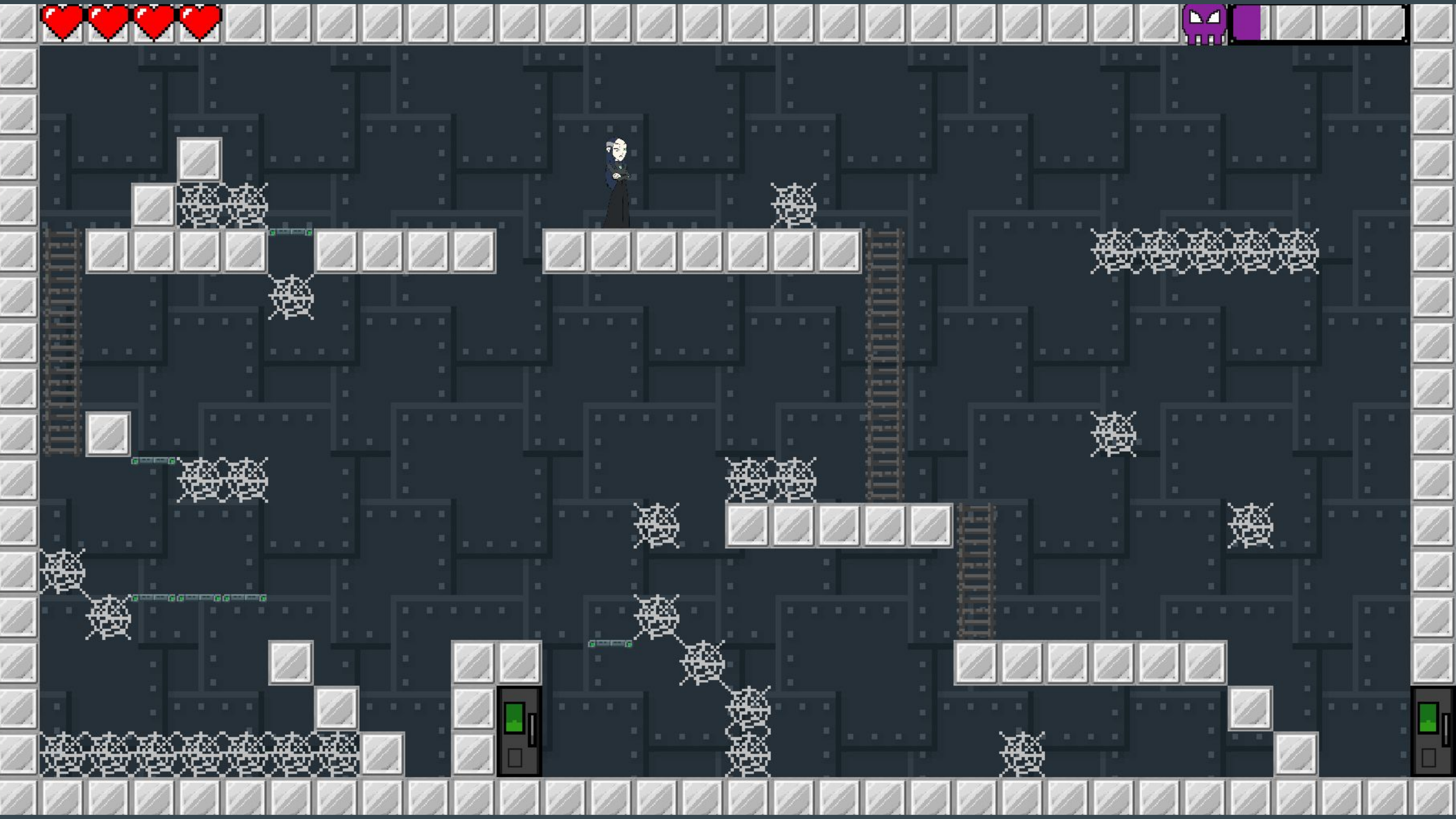
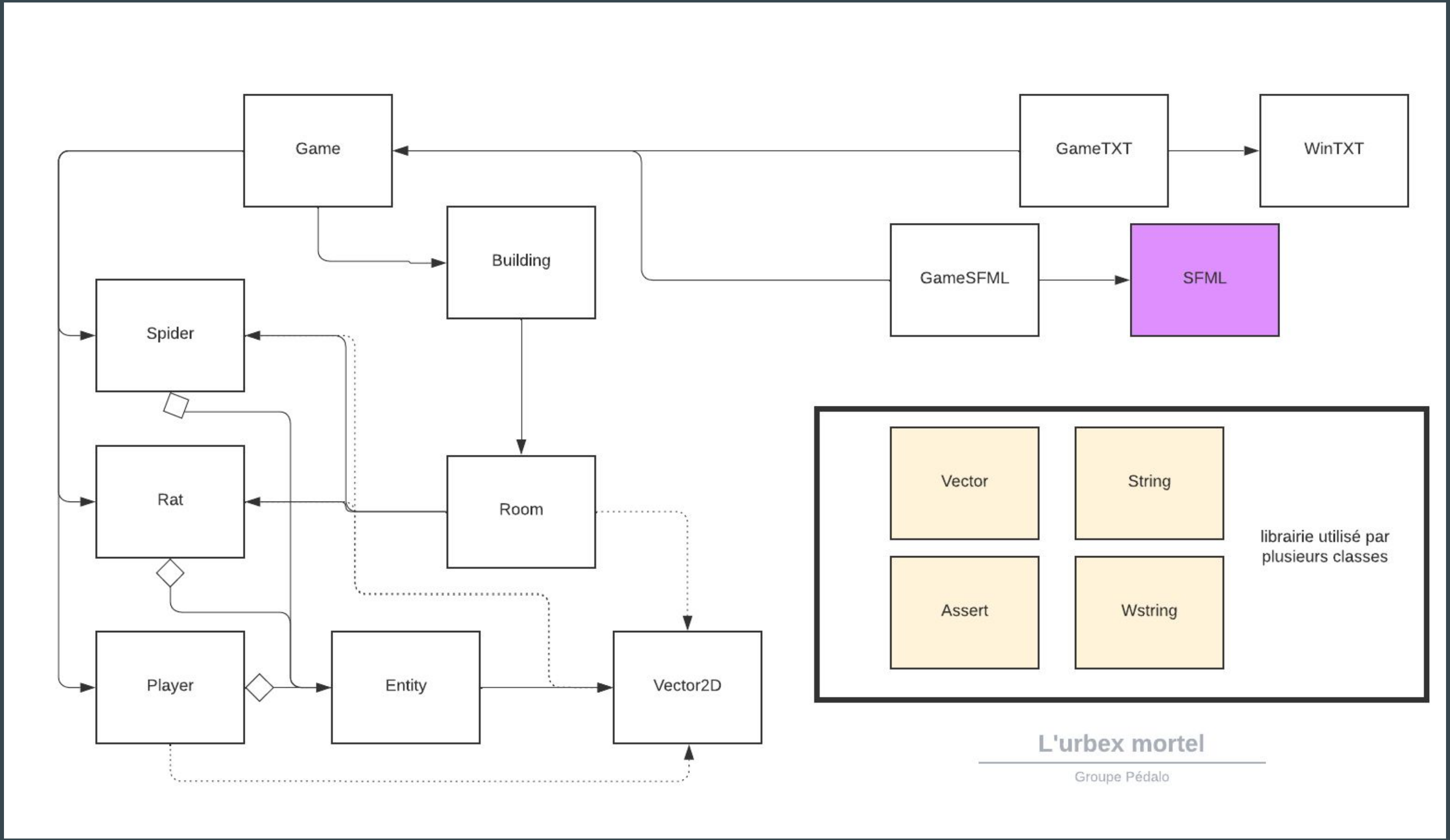
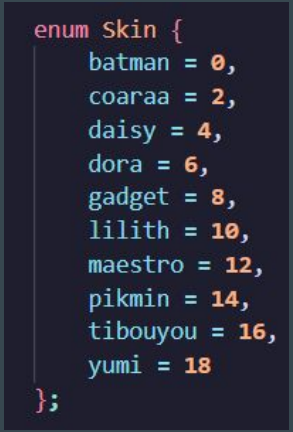


Diagramme des classes

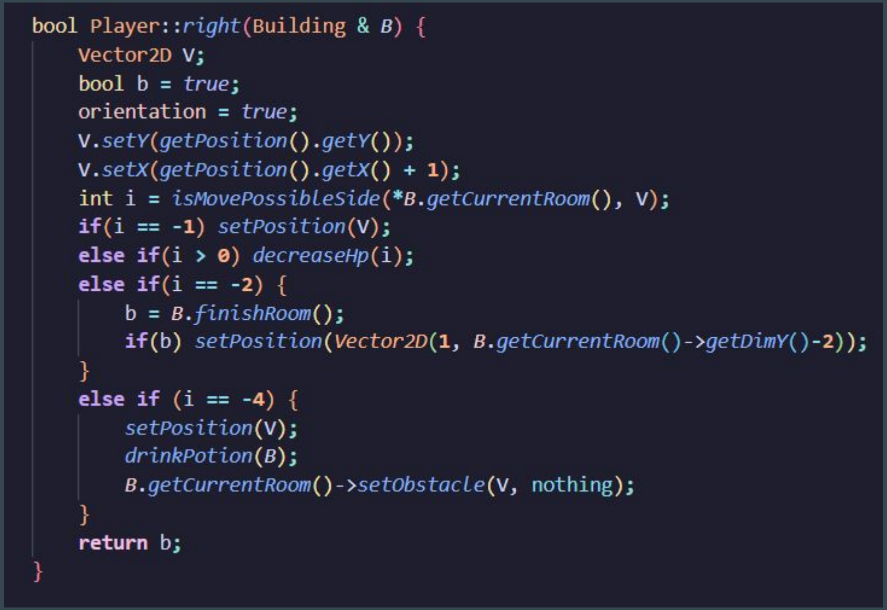


Classe Player

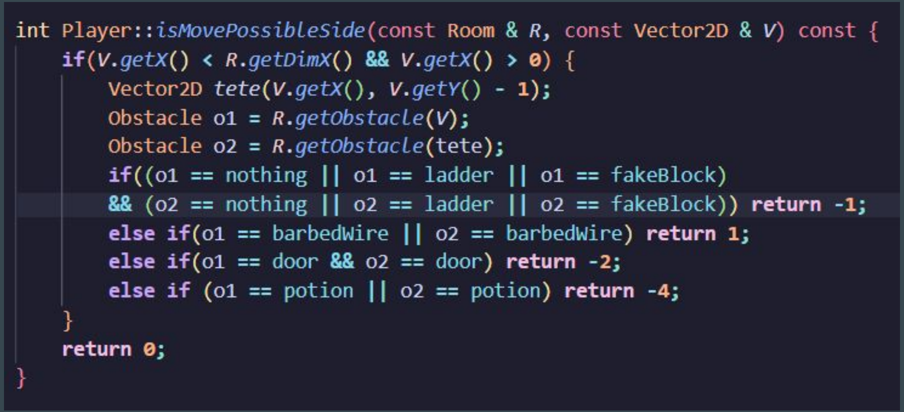


```cpp
enum Skin {
    batman = 0,
    coaraa = 2,
    daisy = 4,
    dora = 6,
    gadget = 8,
    lilith = 10,
    maestro = 12,
    pikmin = 14,
    tibouyou = 16,
    yumi = 18
};
```

exemple de skin

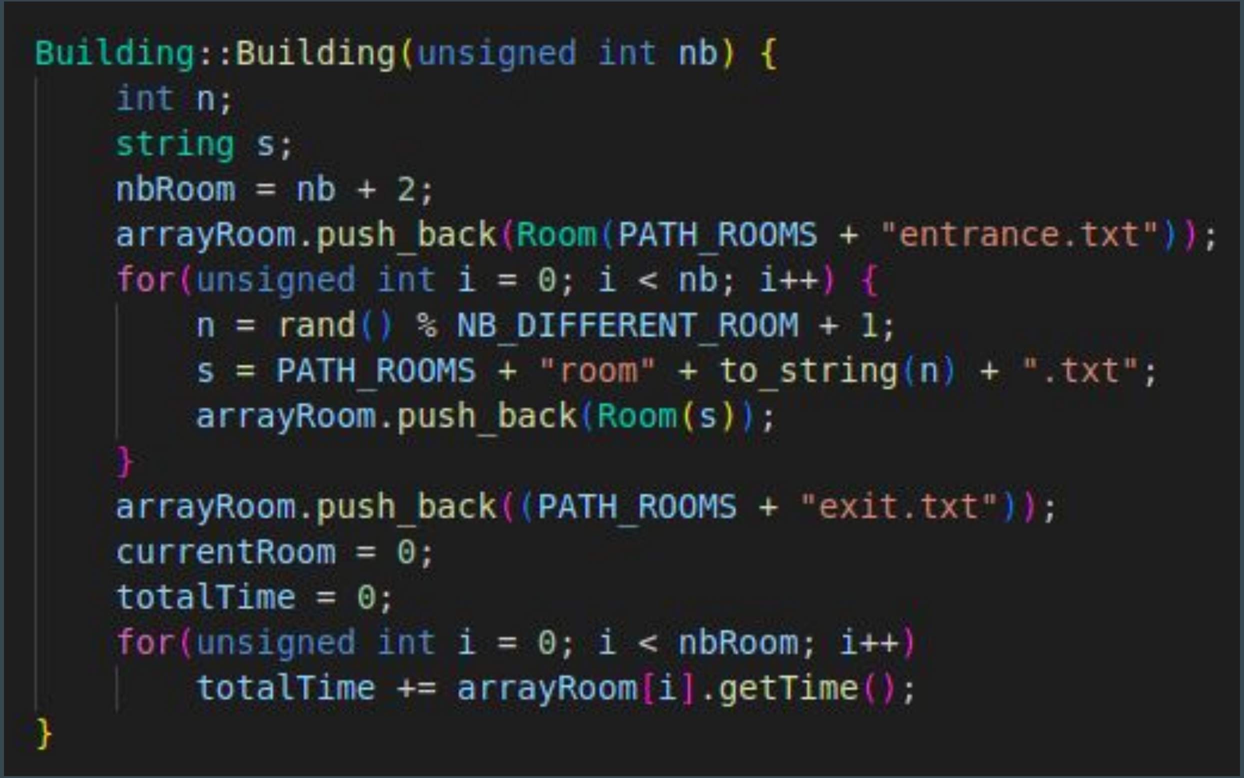


```cpp
bool Player::right(Building & B) {
    Vector2D V;
    bool b = true;
    orientation = true;
    V.setY(getPosition().getY());
    V.setX(getPosition().getX() + 1);
    int i = isMovePossibleSide(*B.getCurrentRoom(), V);
    if(i == -1) setPosition(V);
    else if(i > 0) decreaseHp(i);
    else if(i == -2) {
        b = B.finishRoom();
        if(b) setPosition(Vector2D(1, B.getCurrentRoom()->getDimY()-2));
    }
    else if (i == -4) {
        setPosition(V);
        drinkPotion(B);
        B.getCurrentRoom()->setObstacle(V, nothing);
    }
    return b;
}
```



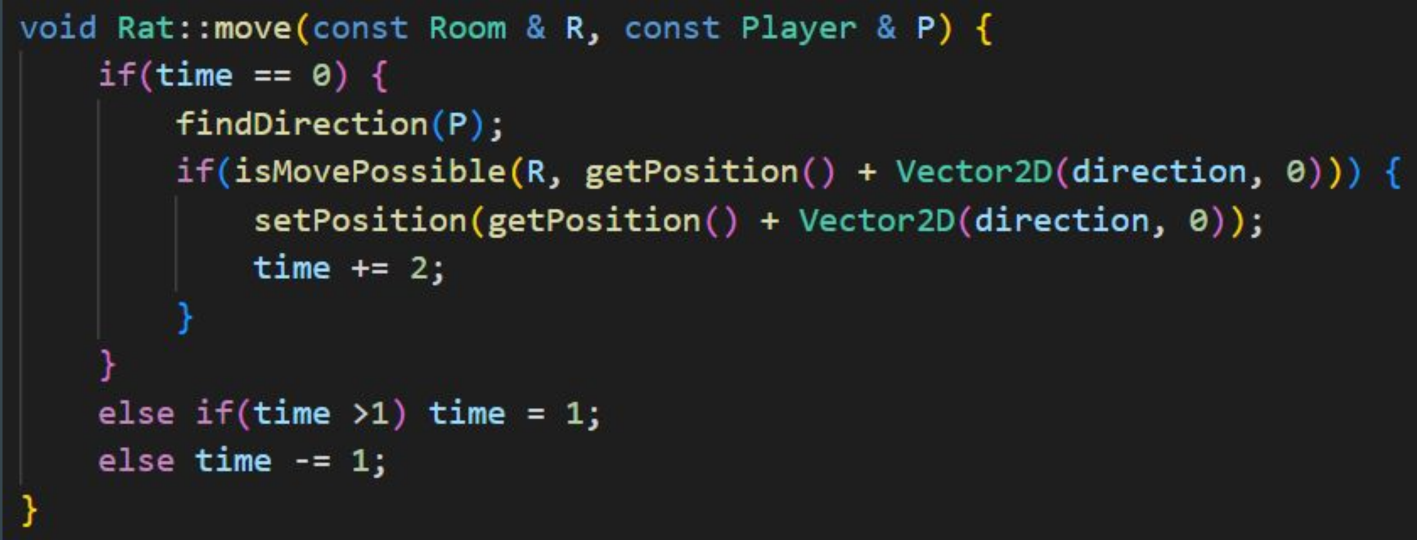
```cpp
int Player::isMovePossibleSide(const Room & R, const Vector2D & V) const {
    if(V.getX() < R.getDimX() && V.getX() > 0) {
        Vector2D tete(V.getX(), V.getY() - 1);
        Obstacle o1 = R.getObstacle(V);
        Obstacle o2 = R.getObstacle(tete);
        if((o1 == nothing || o1 == ladder || o1 == fakeBlock)
        && (o2 == nothing || o2 == ladder || o2 == fakeBlock)) return -1;
        else if(o1 == barbedWire || o2 == barbedWire) return 1;
        else if(o1 == door && o2 == door) return -2;
        else if (o1 == potion || o2 == potion) return -4;
    }
    return 0;
}
```

Classe Building

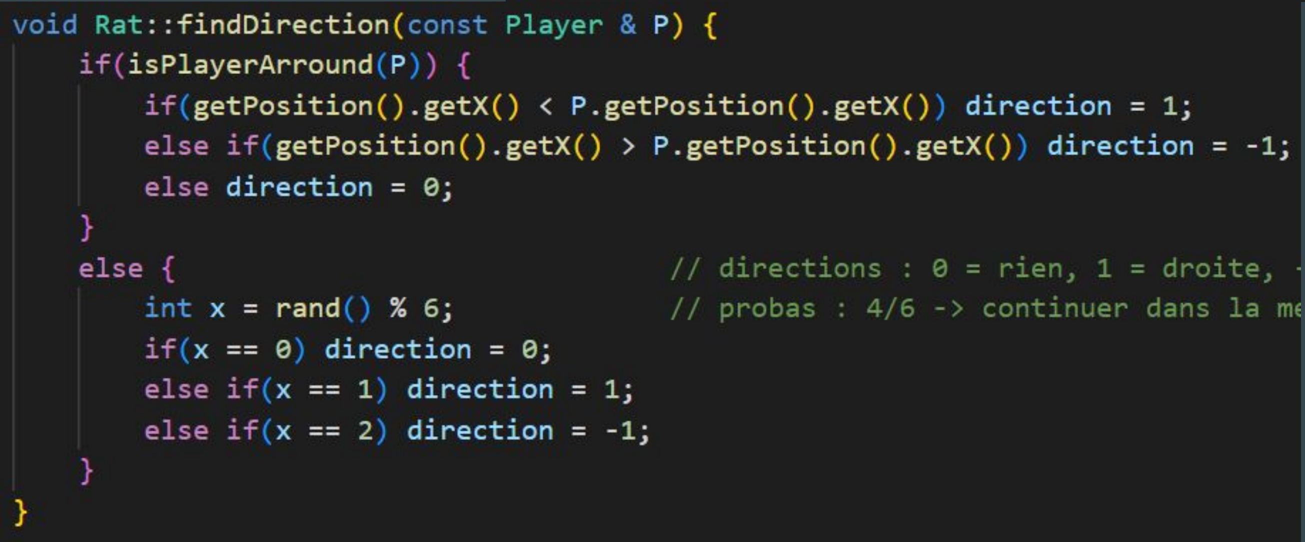


```cpp
Building::Building(unsigned int nb) {
    int n;
    string s;
    nbRoom = nb + 2;
    arrayRoom.push_back(Room(PATH_ROOMS + "entrance.txt"));
    for(unsigned int i = 0; i < nb; i++) {
        n = rand() % NB_DIFFERENT_ROOM + 1;
        s = PATH_ROOMS + "room" + to_string(n) + ".txt";
        arrayRoom.push_back(Room(s));
    }
    arrayRoom.push_back((PATH_ROOMS + "exit.txt"));
    currentRoom = 0;
    totalTime = 0;
    for(unsigned int i = 0; i < nbRoom; i++)
        totalTime += arrayRoom[i].getTime();
}
```

Classe Rat

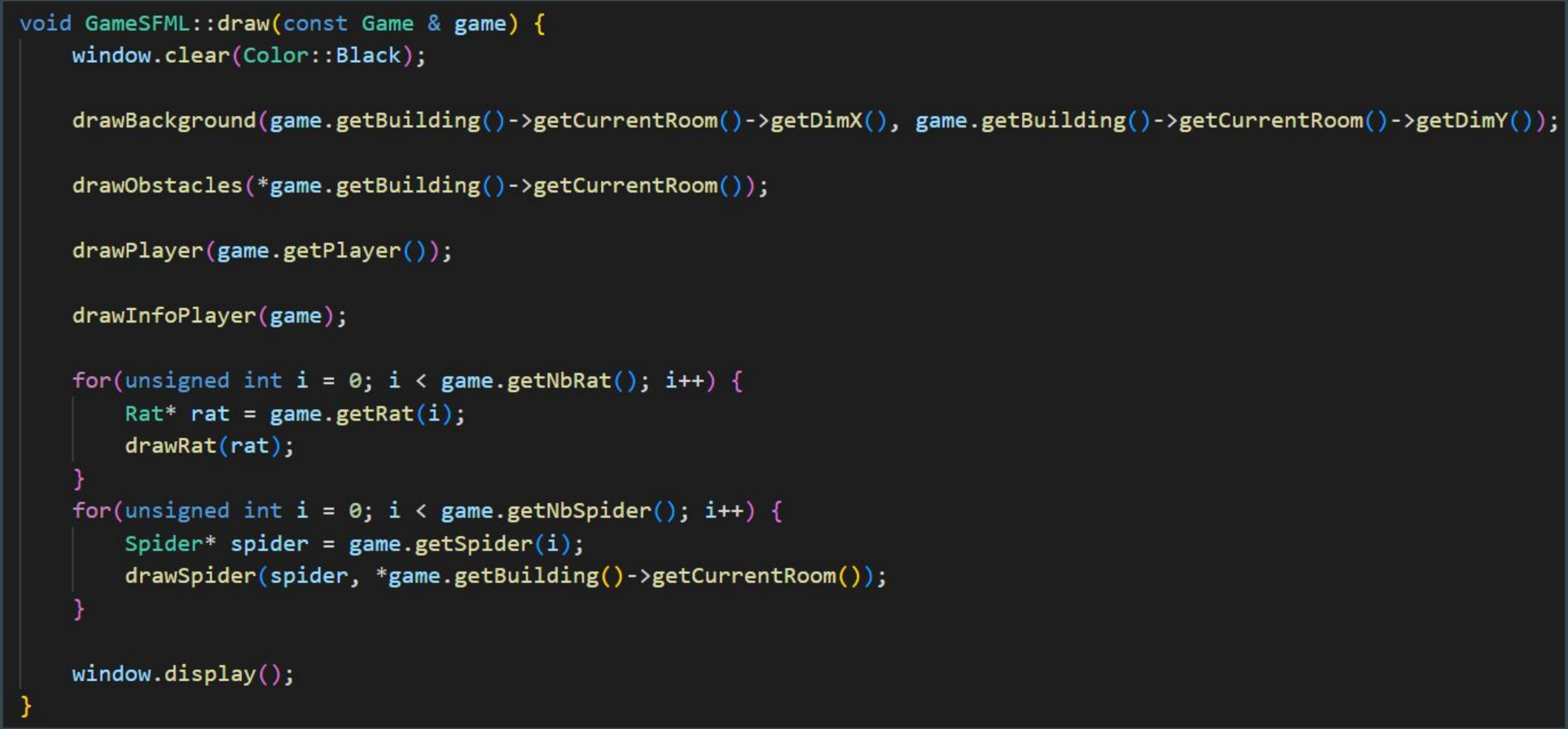


```cpp
void Rat::move(const Room & R, const Player & P) {
    if(time == 0) {
        findDirection(P);
        if(isMovePossible(R, getPosition() + Vector2D(direction, 0))) {
            setPosition(getPosition() + Vector2D(direction, 0));
            time += 2;
        }
    }
    else if(time >1) time = 1;
    else time -= 1;
}
```



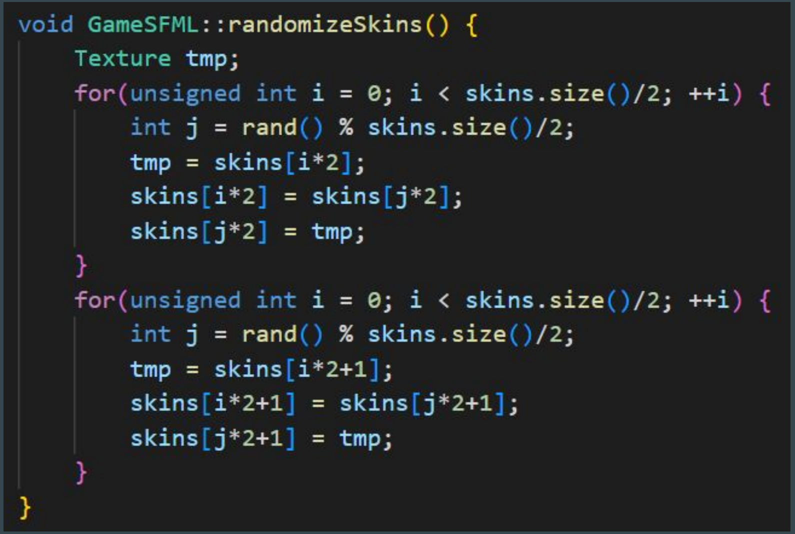
```cpp
void Rat::findDirection(const Player & P) {
    if(isPlayerArround(P)) {
        if(getPosition().getX() < P.getPosition().getX()) direction = 1;
        else if(getPosition().getX() > P.getPosition().getX()) direction = -1;
        else direction = 0;
    }
    else {                              // directions : 0 = rien, 1 = droite, -
        int x = rand() % 6;             // probas : 4/6 -> continuer dans la me
        if(x == 0) direction = 0;
        else if(x == 1) direction = 1;
        else if(x == 2) direction = -1;
    }
}
```

Classe GameSFML

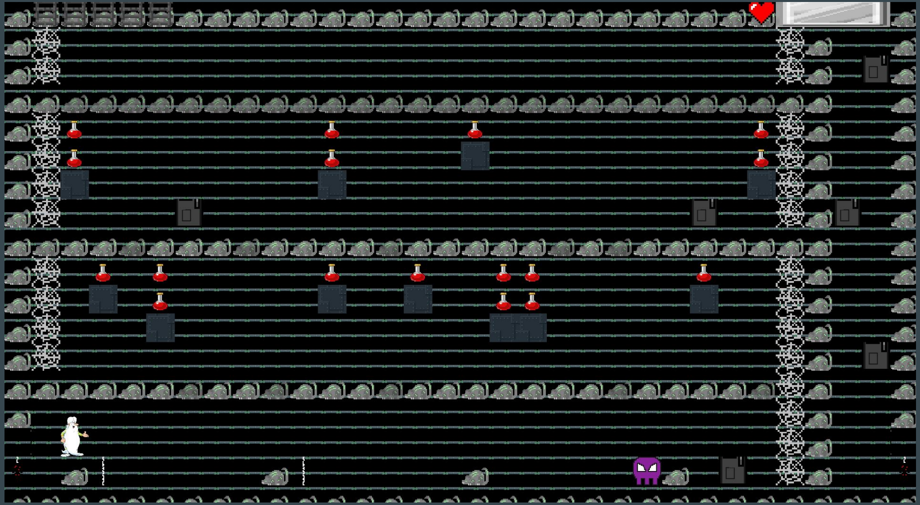


```
void GameSFML::draw(const Game & game) {
    window.clear(Color::Black);

    drawBackground(game.getBuilding()->getCurrentRoom()->getDimX(), game.getBuilding()->getCurrentRoom()->getDimY());

    drawObstacles(*game.getBuilding()->getCurrentRoom());

    drawPlayer(game.getPlayer());

    drawInfoPlayer(game);

    for(unsigned int i = 0; i < game.getNbRat(); i++) {
        Rat* rat = game.getRat(i);
        drawRat(rat);
    }
    for(unsigned int i = 0; i < game.getNbSpider(); i++) {
        Spider* spider = game.getSpider(i);
        drawSpider(spider, *game.getBuilding()->getCurrentRoom());
    }

    window.display();
}
```

Classe GameSFML



```
void GameSFML::randomizeSkins() {
    Texture tmp;
    for(unsigned int i = 0; i < skins.size()/2; ++i) {
        int j = rand() % skins.size()/2;
        tmp = skins[i*2];
        skins[i*2] = skins[j*2];
        skins[j*2] = tmp;
    }
    for(unsigned int i = 0; i < skins.size()/2; ++i) {
        int j = rand() % skins.size()/2;
        tmp = skins[i*2+1];
        skins[i*2+1] = skins[j*2+1];
        skins[j*2+1] = tmp;
    }
}
```



Conclusion

Ce qui n’a pas été fait :

- le mode multijoueur
- les mouvements fluides du joueur

Ce qui n’était pas prévu :

- la musique
- les fakeBlocks
- l’easter egg

Les difficultés :

- les inclusions circulaires
- les affectations de classes contenant des pointeurs
- les unsigned int
- les const et les pointeurs
- WSL