

Somebody stole my hat !

Boissot Aurélien
Laqueuvre Damien
Moulon Florent



Personnage : 2 classes

Pathfinding : 2 classes

Terrain : 4 classes

Item : 3 classes

Structure élémentaire : 3 classes

Génération procédurale :

```

Terrain
+ m_matrice : undef
+ m_nbMeuble : unsigned int
+ m_meubles : undef
+ m_nbPiece : unsigned int
+ m_pieces : undef
+ m_nbPorte : unsigned int
+ mportes : undef
+ m_hauteur : unsigned int
+ m_largeur : unsigned int
+ m_hauteurPieceMax : unsigned int
+ m_hauteurPieceMin : unsigned int
+ m_largeurPieceMax : unsigned int
+ m_largeurPieceMin : unsigned int
+ m_probaDiv : unsigned int
+ Terrain() «constructor»
+ Terrain(hauteur : unsigned int, largeur : unsigned int) «constructor»
+ ~Terrain() «destructor»
+ zoneEstVide(chg : Point2D const&, cbd : Point2D const&) : bool
+ zoneMur(chg : Point2D const&, cbd : Point2D const&) : bool
+ caseEstVide(y : unsigned int, x : unsigned int) : bool
+ retrouvePorte(NbPiece : int) : Point2D
+ generer()
+ vider()
+ caseLibrePourDebuter() : Point2D
+ hitboxEstDansMur(hitbox : const HitboxRectangle&) : bool
+ collisionTerrain(hitbox : const HitboxRectangle&, hitboxVirtuelle : HitboxRectangle, direction : cons
+ droiteEstDansMur(pt1 : const Point2D&, pt2 : const Point2D&) : bool
+ getHauteur() : unsigned int
+ getLargeur() : unsigned int
+ getYX(y : const unsigned int, x : const unsigned int) : char
+ getNbMeuble() : unsigned int
+ getMeuble(i : int) : Meuble
+ getNbPiece() : unsigned int
+ getMeublePiece(i : int) : Piece
+ getNbPorte() : unsigned int
+ getMeublePorte(i : int) : Porte
+ porteAutour(chg : Point2D const&, cbd : Point2D const&) : bool
+ corrigerPorteVerticale(porte : undef)
+ corrigerPorteHorizontale(porte : undef)
+ corrigerPortes()
+ remplacerCase(ancien : char, nouveau : char)
+ diviser(pieceChg : Point2D const&, pieceCbd : Point2D const&, probaDiv : unsigned int)
+ meubleEstDans(meuble : undef, pieceChg : Point2D const&, pieceCbd : Point2D const&) : bool
+ ajouterMeubleAuMur(meuble : undef, pieceChg : Point2D const&, pieceCbd : Point2D const&)
+ ajouterMeubleAuCentre(meuble : undef, pieceChg : Point2D const&, pieceCbd : Point2D const&)
+ ajouterMeubleAuCentreGrand(meuble : undef, pieceChg : Point2D const&, pieceCbd : Point2D const&)
+ dessinerMeuble(meuble : undef)
+ remplirPieces()
+ remplirPieceChambre(pieceChg : Point2D const&, pieceCbd : Point2D const&)
+ remplirPieceDortoir(pieceChg : Point2D const&, pieceCbd : Point2D const&)
+ remplirPieceStockage(pieceChg : Point2D const&, pieceCbd : Point2D const&)
+ remplirPieceBureau(pieceChg : Point2D const&, pieceCbd : Point2D const&)
+ remplirPiecePrincipale(pieceChg : Point2D const&, pieceCbd : Point2D const&)
+ remplirPieceDeau(pieceChg : Point2D const&, pieceCbd : Point2D const&)
+ ajouterPiece(pieceChg : Point2D const&, pieceCbd : Point2D const&)
+ corrigerPorteVerticale(porte : Porte const&)
+ corrigerPorteHorizontale(porte : Porte const&)
+ meubleEstDans(meuble : Meuble&, pieceChg : Point2D const&, pieceCbd : Point2D const&) : bo
+ ajouterMeubleAuMur(meuble : Meuble&, pieceChg : Point2D const&, pieceCbd : Point2D const&)
+ ajouterMeubleAuCentre(meuble : Meuble&, pieceChg : Point2D const&, pieceCbd : Point2D const&)
+ ajouterMeubleAuCentreGrand(meuble : Meuble&, pieceChg : Point2D const&, pieceCbd : Point2D const&)
+ dessinerMeuble(meuble : Meuble const&)
    
```

On ne fait pas de hasard, on génère
à l'aide d'un algorithme déterministe

```

XXXXXXXXXXXXXXXXXXXXXXXXX
X  LL  XPPP  X
X  LL  X  TTT  X
X      X  TTT  X
X      TTT  X
X      AA X  X
X      X  PP  X
XXXXXXXXXXXX  XXXXXXXXXX
X      X  T  X
X BB  X
XXXXXXXXXXXXXXXXXXXXXXXXX
    
```

Pathfinding

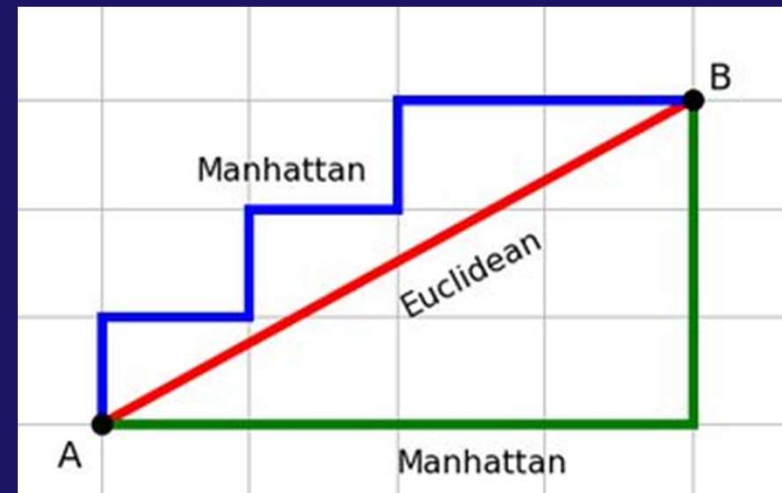
Utilisation de l'algorithme A*

```

Noeud
+ parent : Point2D
+ position : Point2D
+ heuristique : int
+ distanceDepart : int
+ score : int
+ operator =(n : const Noeud&) : N
    
```

```

Chemin
- m_graph : Terrain
- m_depart : Point2D
- m_arrivee : Point2D
- m_listeOuverte : std::vector< Noeud >
- m_listeFermee : std::vector< Noeud >
+ Chemin() «constructor»
+ Chemin(t : const Terrain&) «constructor»
+ Chemin(t : const Terrain&, ptArrivee : Point2D&, ptDepart : Point2D&) «constructor»
+ ~Chemin() «destructor»
+ setArriveeDepart(ptArrivee : Point2D&, ptDepart : Point2D&)
+ setGraph(t : const Terrain&)
+ trouveChemin(l : std::vector< Point2D >)
+ cheminExiste() : bool
+ definitNoeud(parent : const Noeud&, coordY : int, coordX : int) : Noeud
- cheminReset()
- InitialisationNoeud() : Noeud
- traiteNoeudsAdjacents(n : const Noeud&)
- traiteNoeud(n : const Noeud&, x : int, y : int)
- retrouveChemin(l : std::vector< Noeud >) : std::vector< Point2D >
- estPlacable(c : const Case&) : bool
- estDansListe(l : const std::vector< Noeud >&, n : const Noeud&) : bool
- maiScore(l : std::vector< Noeud >&, n : const Noeud&)
- trouveNoeud(l : std::vector< Noeud >&, pt : const Point2D&) : Noeud
    
```



Gestion des menus (Polymorphisme)

```
38 //tableau servant à stocker nos écrans
39 std::vector<EcranBase*> Ecrans;
40 int idEcran = 0;
41
42 //création de nos écrans et ajout dans le tableau d'écrans
43 Menu * menu = new Menu;
44 Ecrans.push_back((EcranBase *)(menu));
45
46 JeuSFML * jeuSFML = new JeuSFML;
47 jeuSFML->init();
48 jeuSFML->genererNiveau(5, 12, 12);
49 Ecrans.push_back((EcranBase *)(jeuSFML));
50
51 Pause * pause = new Pause;
52 Ecrans.push_back((EcranBase *)(pause));
53
54 GameOver * gameOver = new GameOver;
55 Ecrans.push_back((EcranBase *)(gameOver));
56
57 //boucle permettant de naviguer entre les différents écrans en fonction de l'idEcran
58 while (idEcran >= 0) {
59     idEcran = Ecrans[idEcran]->boucle(window, view);
60 }
61
62 //destruction de nos écrans
63 delete menu;
64 delete jeuSFML;
65 delete pause;
66 delete gameOver;
```

Conclusion

Objectifs atteints :

- pathfinding
- génération procédurale
- jeu fonctionnel

Si on avait eu plus de temps:

- mode campagne
- autre type d'arme

Difficultés :

- séparer l'affichage SFML du core
- gérer les grosse classes et leurs différents algorithmes