Jeu d'aventure 2D

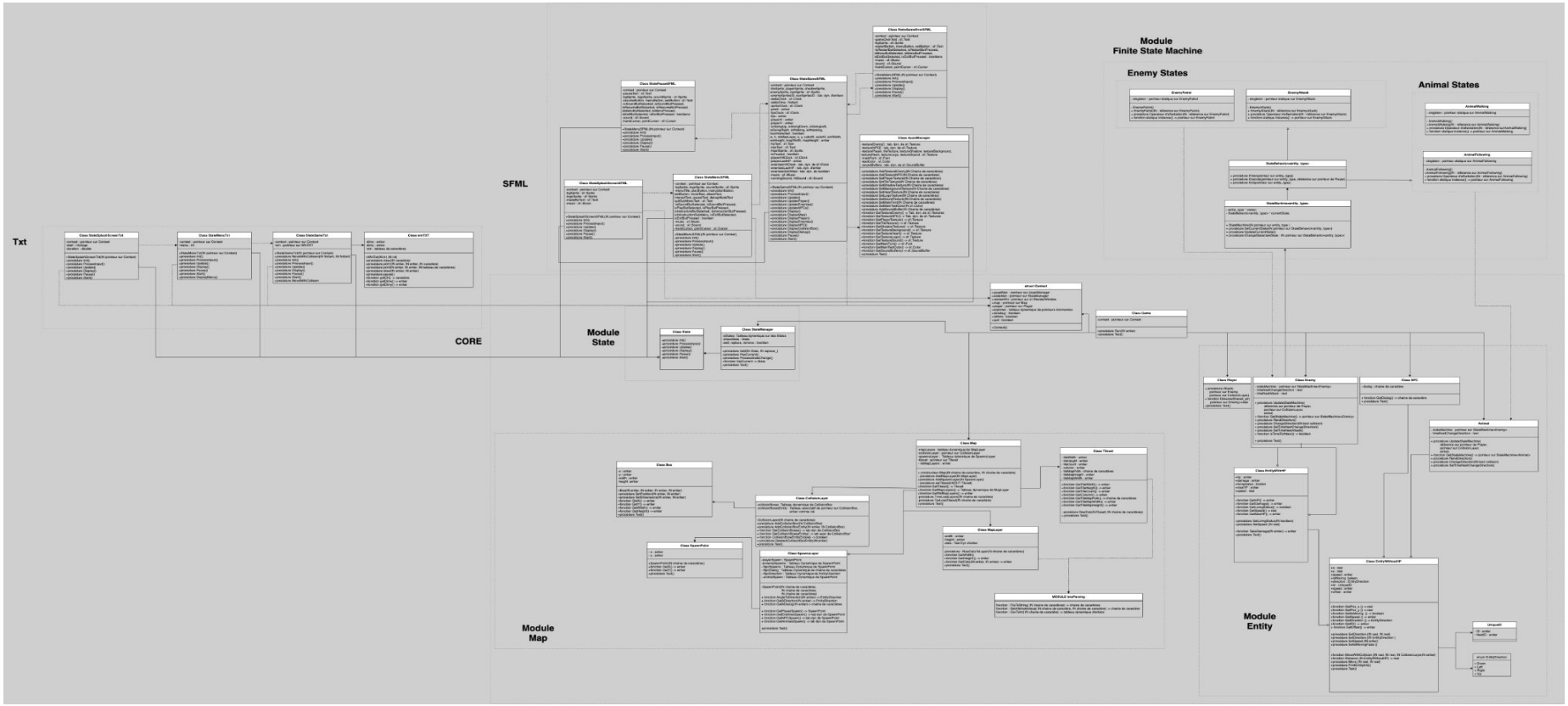
LEGEND OF NAUTIBUS

LIFAP4
PROJET SFL

- SIONI Farès
- BAGNOL Stanislas
- CHOUGAR Lyes

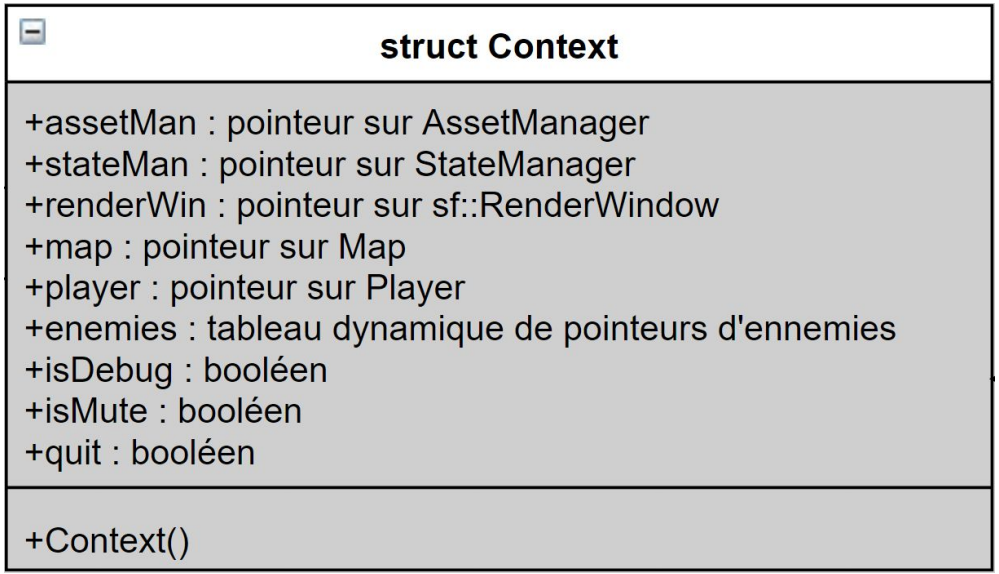
S4 - 2021
Université Claude Bernard Lyon 1

Diagramme des classes



Structure Context Alias : l'état courant

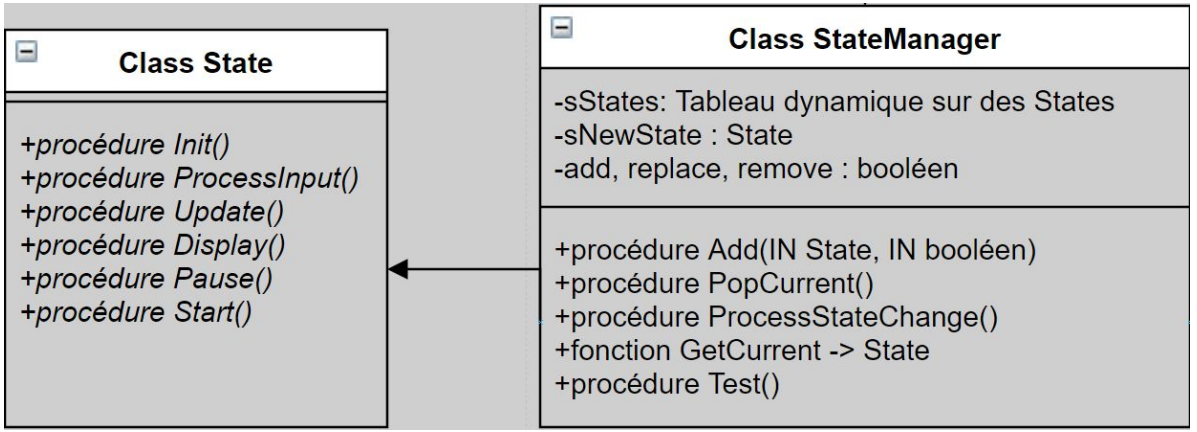
- Elle est le "coeur" des données du jeu.
- Elle est initialisée au lancement et vit aussi longtemps que le programme.
- Elle contient toutes les variables essentielles.
  Ex. : L'état de la fenêtre, les textures (AssetManager), le joueur, etc.



Le State Manager

- **Classe StateManager :**

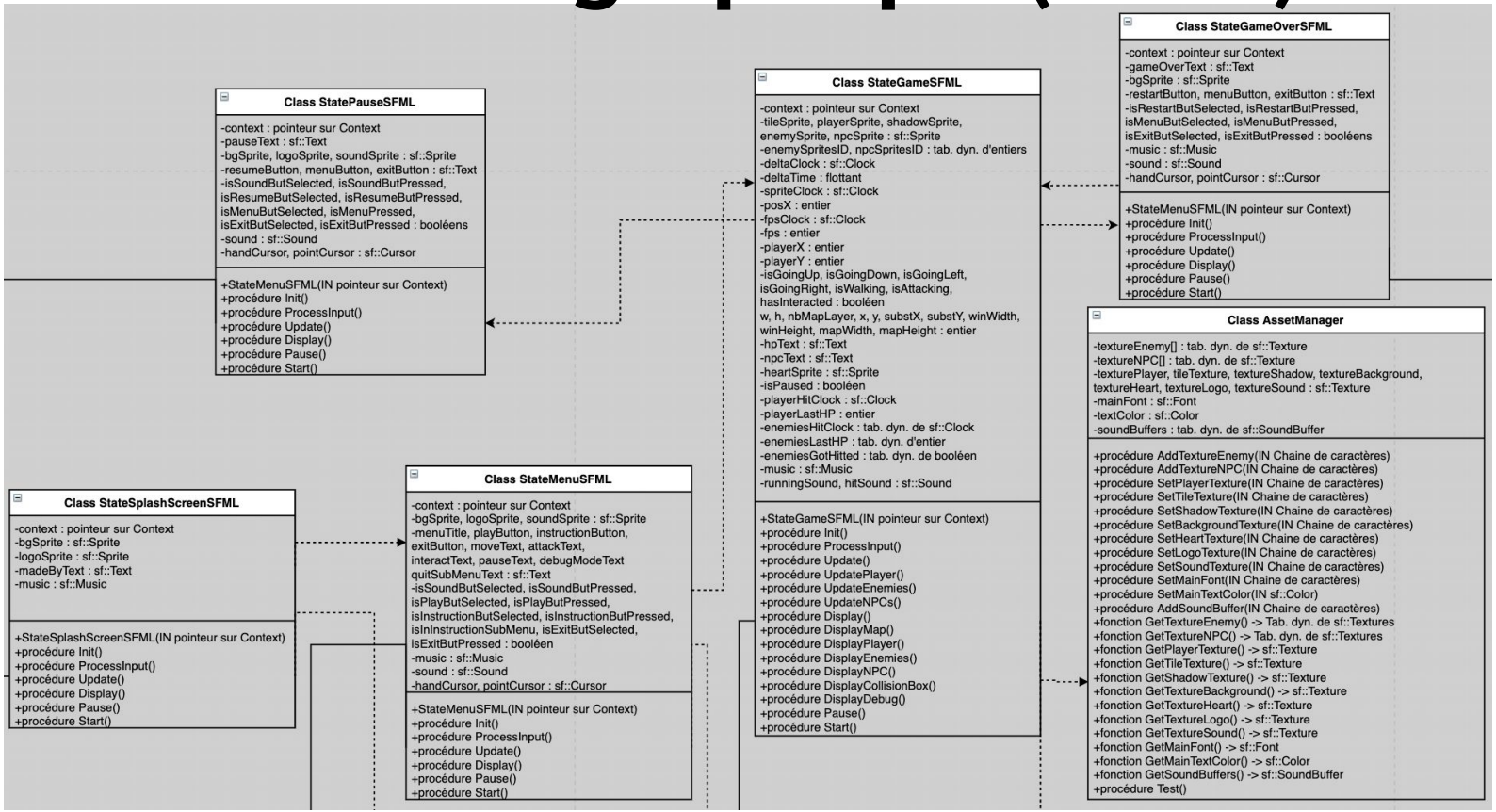
Permet de gérer les passage d'un état à un autre.
Exemple : Passage de SplashScreen au Menu.



- **Classe State :**

Classe représentant un état du programme..
State est une classe virtuelle pure : elle doit être héritée et toutes les fonctions qui en héritent sont implémentées.
Exemple de State : SplashScreen, Menu, Game.

Le mode graphique (SFML)



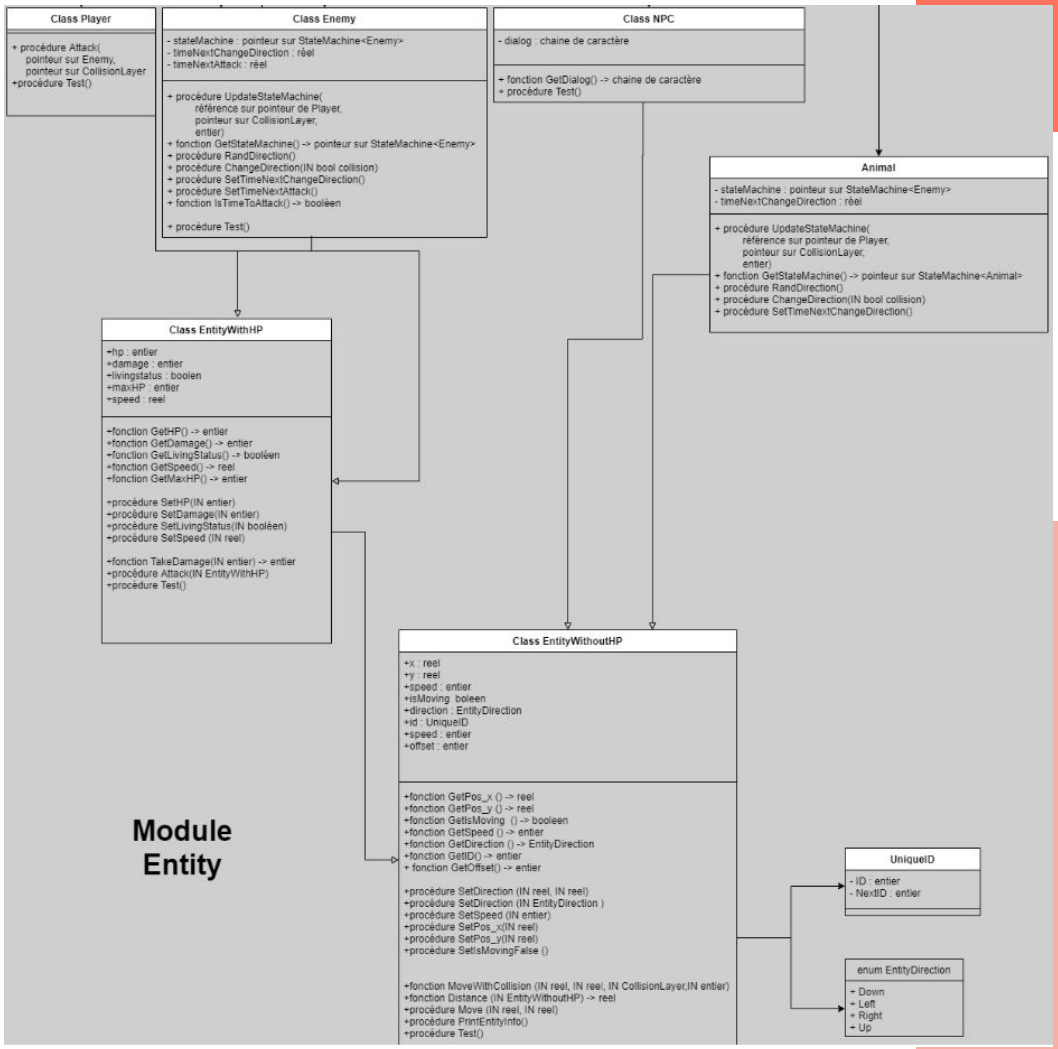
Module Entity

Toutes les classes du module Entity héritent de la classe mère EntityWithoutHP.

Une distinction est faite entre les entités :

- Les classes Player et Enemy (entités mortelles) sont des classes filles de EntityWithHP.
- Les classes NPC et Animal (entités immortelles) sont des classes héritées de EntityWithoutHP.

La classe EntityWithHP hérite d'EntityWithoutHP.

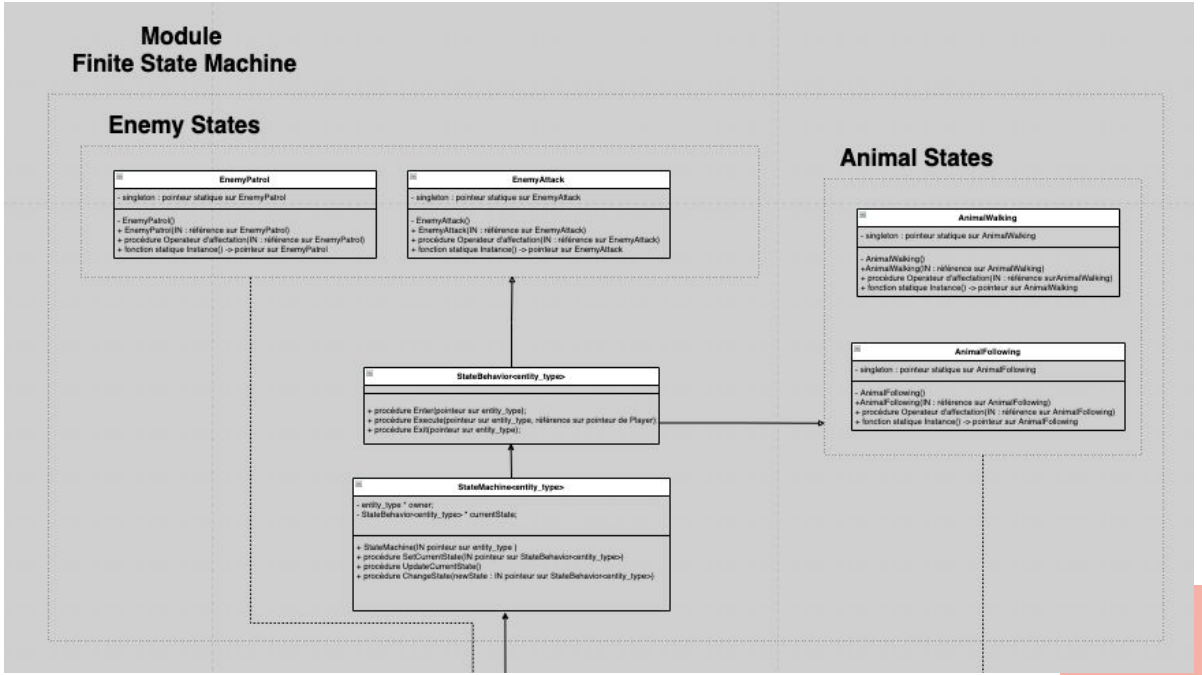


Module Finite State Machine

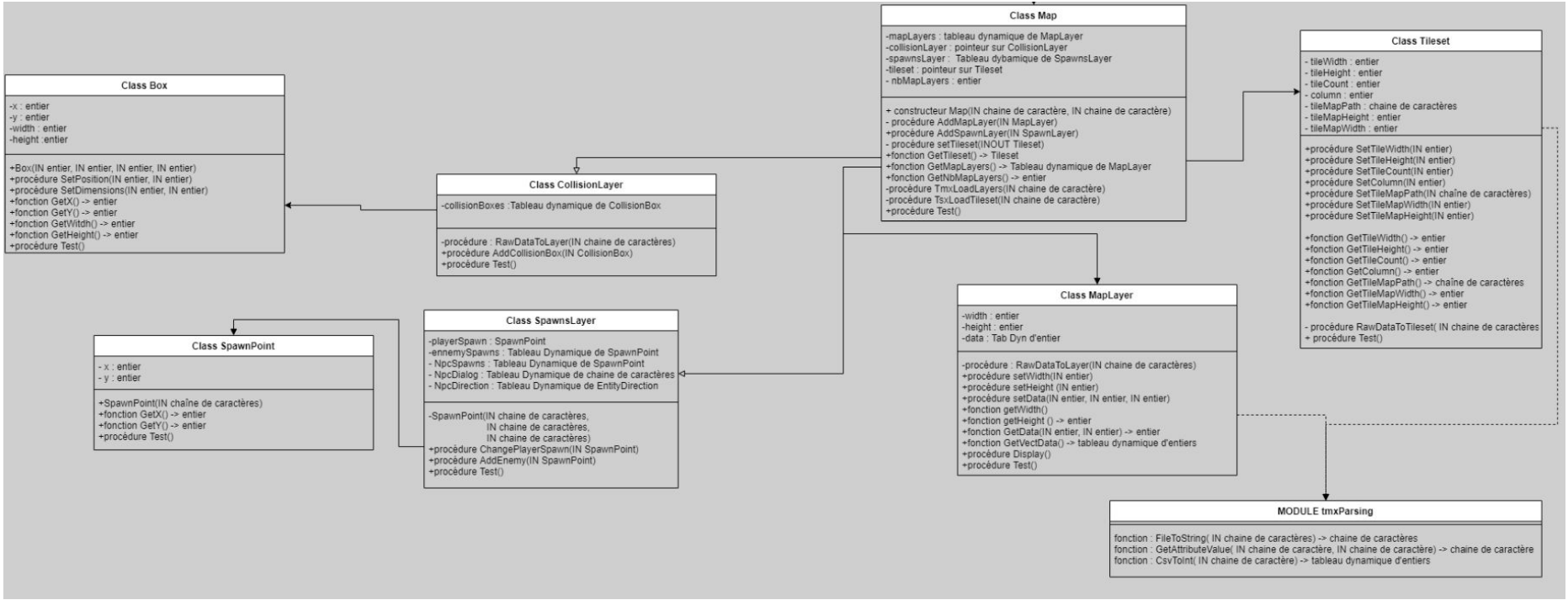
Un Finite State Machine permet de définir des états prédéfini pour les entités. C'est une méthode plutôt simple pour créer une intelligence artificielle.

Class StateBehavior<entity_type>

Class StateMachine<entity_type>

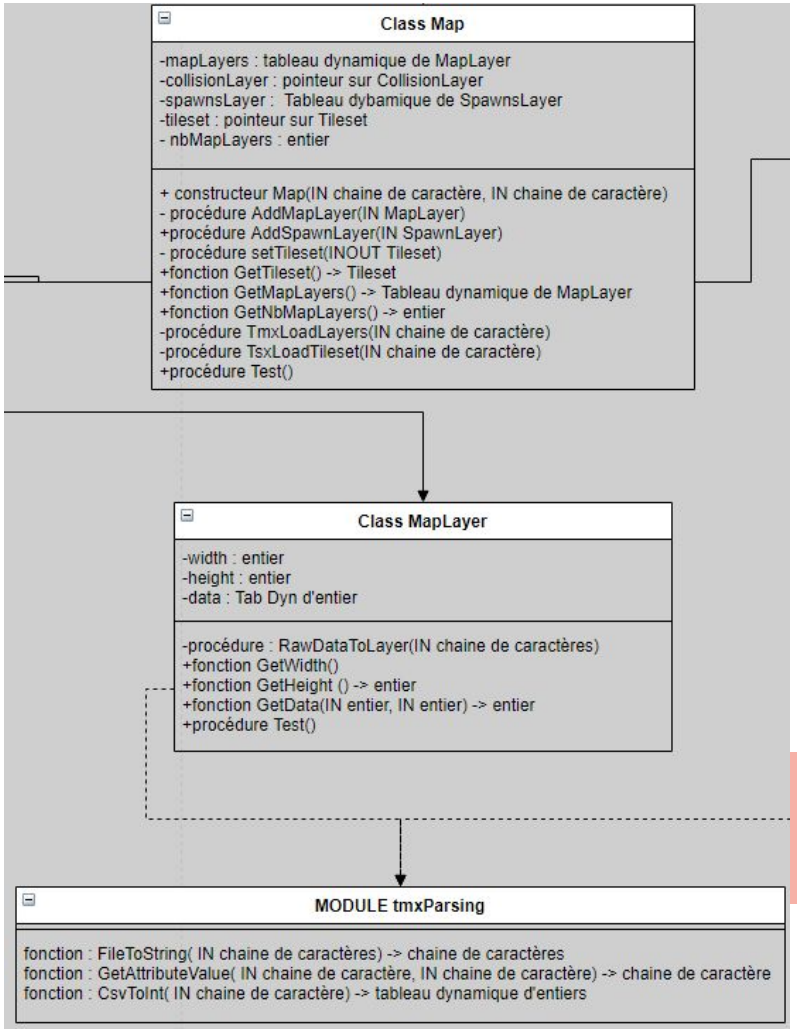


tmxParsing et Module Map



- **Classe MapLayer :**

Classe représentant une couche de tuiles, elle contient les informations de la map à proprement parler.



tmxParsing

<layer id="2" name="Trees" width="100" height="50">

<data encoding="csv">

0,15,16,0,0,15,16,0,15,16,0,0,0,0,0,0,0,11,12,0,0,0,0,0,11,12,0,0,0,0,11,12,0,0,0,0,0,0,15,16,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,15,16,0,

0,15,16,0,15,16,0,0,0,15,16,0,0,0,0,0,15,16,0,0,0,0,15,16,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,23,24,15,16,23,24,0,23,24,0,0,0,11,12,0,0,19,20,0,11,12,0,0,19,20,0,11,12,0,19,20,0,0,11,12,0,0,23,24,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,23,24,15,16,23,24,0,23,24,0,15,16,23,24,0,15,16,0,0,23,24,0,0,0,0,23,24,0,0,0,0,44,0,0,0,0,0,0,0,44,0,

0,0,0,23,24,0,0,0,0,15,16,0,0,19,20,0,0,0,11,12,19,20,0,0,0,0,0,0,19,20,0,0,0,15,16,19,20,0,15,16,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

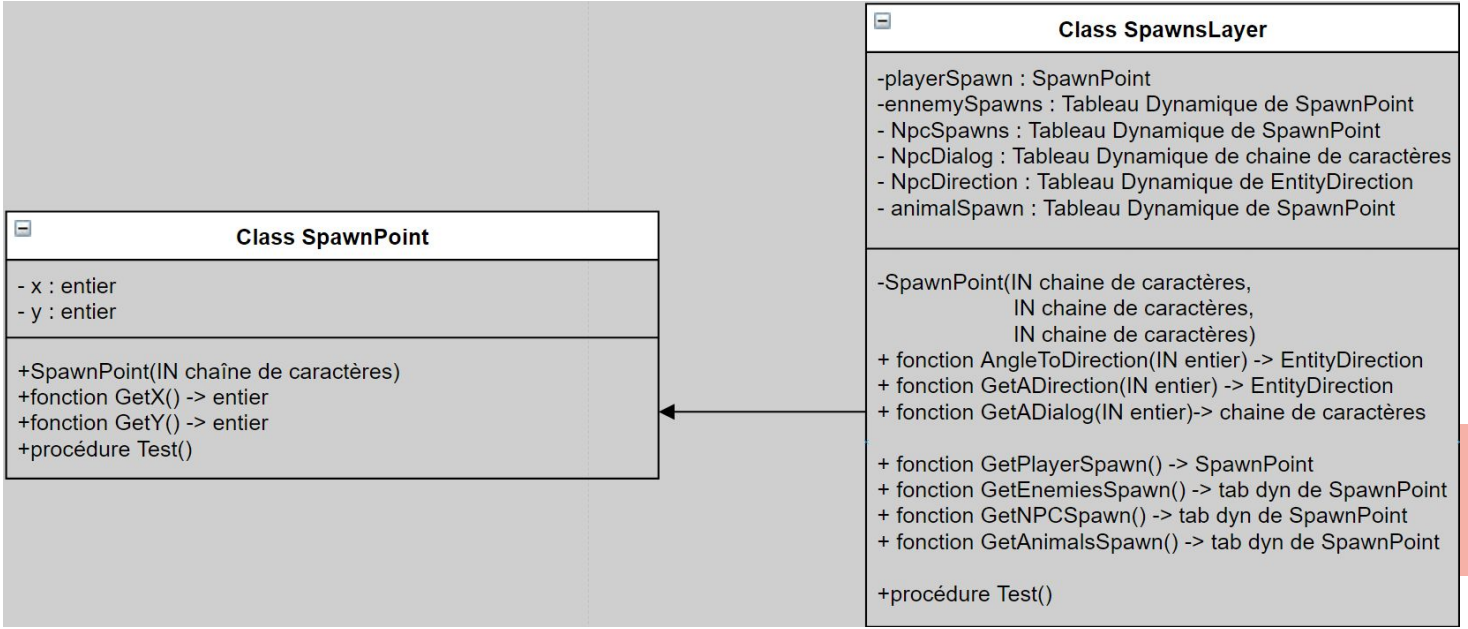
0,23,24,0,0,15,16,0,0,23,24,0,0,0,23,24,0,0,0,15,16,0,15,16,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,0,15,16,0,0,15,16,0,23,24,15,16,0,0,0,11,12,19,20,0,0,0,11,12,0,0,0,0,15,16,0,23,24,15,16,0,23,24,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

15,16,0,0,0,0,23,24,0,0,0,0,0,15,16,0,0,15,16,0,23,24,0,23,24,0,15,16,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

- **Classe SpawnLayer :**

Représente tous les SpawnPoint du jeu :
des ennemis, du joueur et des NPC.



tmxParsing

```
<objectgroup id="6" name="PlayerSpawn">
      <object id="8" x="478.333" y="1311"/>
</objectgroup>
<objectgroup id="10" name="NPCSpawn">
     <object id="405" x="2678" y="1397.33" rotation="270">
          <properties>
               <property name="Dialog" value="There was a time when we controlled the whole village and it may still be the case ..."/>
          </properties>
      </object>
 <object id="406" x="2291.33" y="1142">
          <properties>
               <property name="Dialog" value="I love animals ! But they     don't want to play with me..."/>
           </properties>
 </object>
```

```
case 0: //mode SFML
{
    int winWidth = 700;
    int winHeight = 700;
    context->renderWin->create(sf::VideoMode(winWidth, winHeight, 32),
                               "Legend Of Nautibus");
    context->stateMan->Add(std::make_unique<StateSplashScreenSFML>
                        (context));
    while (!context->quit)
    {
        context->stateMan->ProcessStateChange();
        context->stateMan->GetCurrent()->ProcessInput();
        context->stateMan->GetCurrent()->Update();
        context->stateMan->GetCurrent()->Display();
    }
    break;
}

case 1: //mode txt
    context->stateMan->Add(std::make_unique<StateSplashScreenTxt>(context));
    while (!context->quit)
    {
        context->stateMan->ProcessStateChange();
        context->stateMan->GetCurrent()->ProcessInput();
        context->stateMan->GetCurrent()->Update();
        context->stateMan->GetCurrent()->Display();
    }
    break;
```

La fonction Run() de Game, lancée depuis le main.

Aller plus loin

Système de sauvegarde

Possibilité d'enregistrer/charger la structure Context

Gestion des armes et armures

Le joueur peut porter des armes et armures différentes, visuellement ou dans les caractéristiques

Histoire

Le joueur a un/des objectifs précis, des quêtes à accomplir, etc.

Réglages plus poussés

Possibilité de modifier l'assignation des touches, le volume du son, etc.

Système de compétences

Après avoir gagné de l'expérience, le joueur peut augmenter ses dégâts, sa vitesse, etc.

Bâtiments et PNJ interactifs

Possibilité de rentrer dans des bâtiments, interagir avec les PNJ : nous donner des objets/soins/quêtes

Ce que cela nous a appris

Le travail en équipe	La gestion de projet	Les outils de développement
Communication, affectation/division des tâches, développement en parallèle	Respect des délais, rigueur, Trello, Gantt, cahier des charges	Gitlab, Doxygen, Valgrind, SFML

L’autonomie

Le problème ne peut être réglé que par nous-même

MERCI
A VOUS!

Avez-vous des questions ?

LEGEND OF
NAUTIBUS