TD Architecture

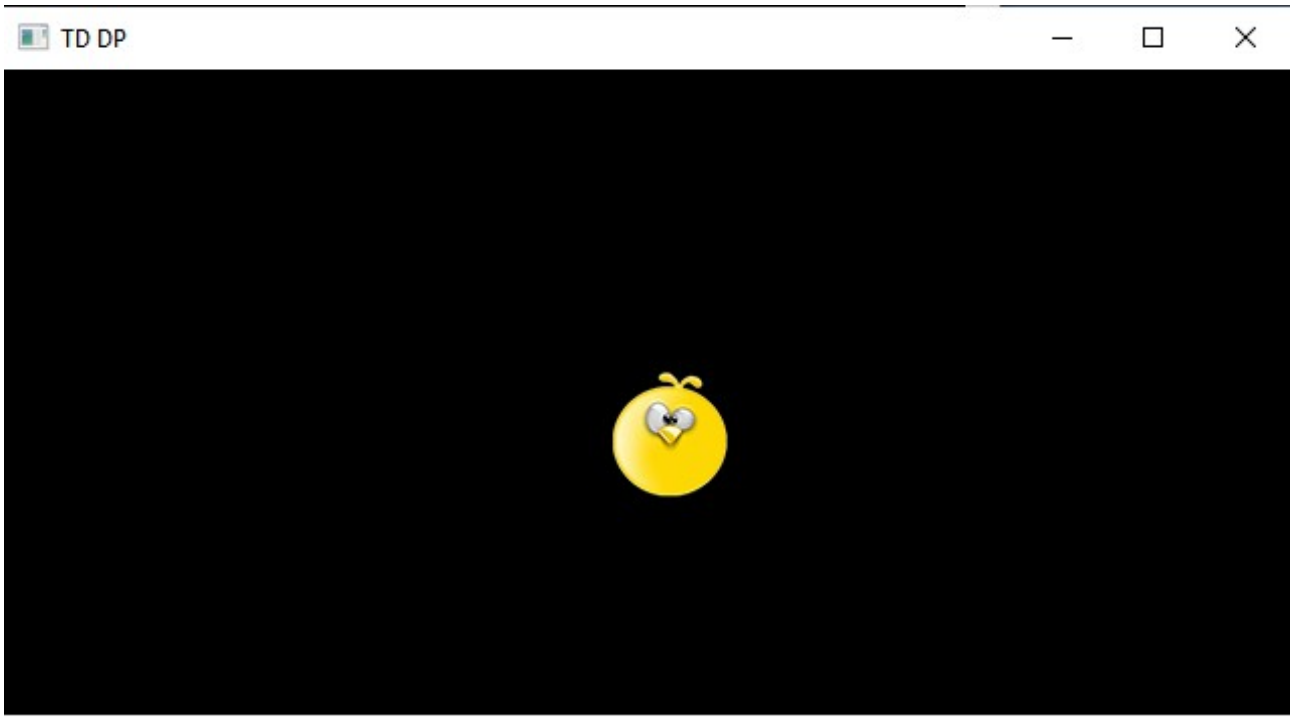
Le but de ce TD est de préparer le framework sur lequel nous allons développer les modules d'IA.

Etapes :

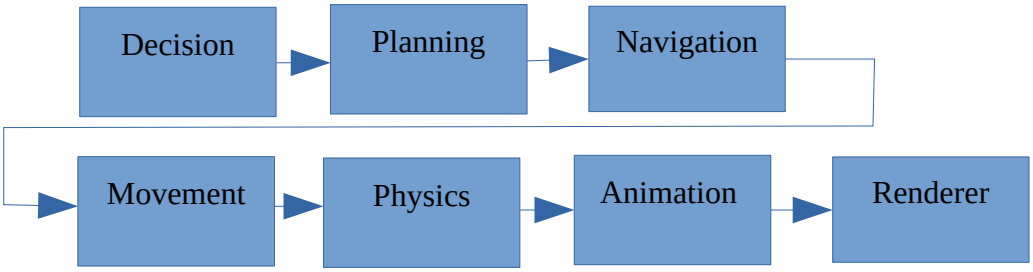
1) Extraire SFML (2.4.0, 32 bits) dans D : (il sera installé dans D:\SFML-2.4.2)

2) Extraire le fichier TD Architecture.zip

3) Compiler la solution TD Architecture\Cells\src\Cells.sln en Debug et Release pour obtenir ceci :

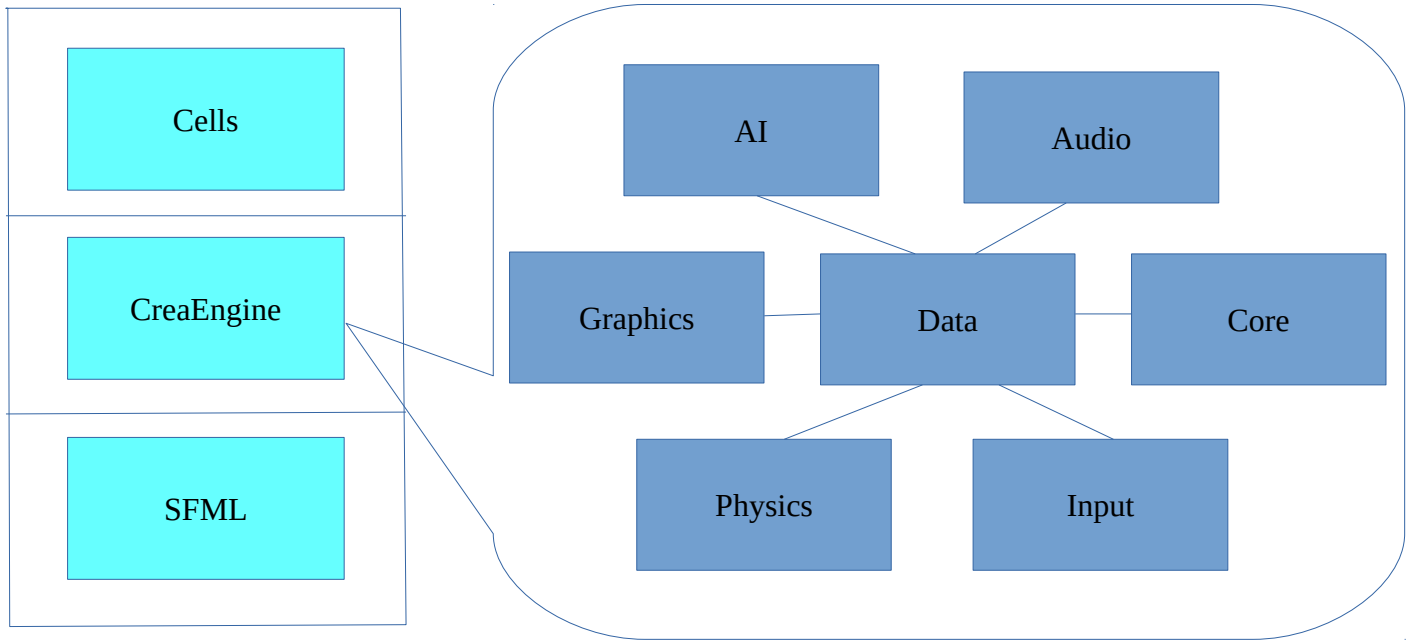


C'est la dernière étape de notre pipeline d'IA : l'affichage de l'entité à une certaine frame, position et animation.



Nous allons modifier la solution afin de se préparer à l'intégration des autres étapes du pipeline d'IA et selon une architecture centrée sur les données.

4) Architecture
Le projet possède une architecture en couches et centré sur les données de la forme :



Cette architecture se reflète au niveau :
- des **solutions** qui composent le projet
- des **répertoires** où sont rangés les fichiers (.h et .cpp)

A l'aide de Visio, réalisez un Diagramme de classe qui montre cette architecture ainsi que les classes qui la réalise.
n.b : Les formes sont dans « Logiciel → Classe UML »

5) Organisation des composantes
L'organisation des fichiers et répertoires du projet est la même que celle de la SFML.

Cells
bin : .exe, .dll (release et debug), .pdb et data
doc : toute la documentation sur l'application Cells
include : fichiers .h
src : .solution sln, projet .vcxproj et fichiers .cpp
tmp : fichiers temporaires ipch, Release et Debug

CreaEngine
bin : .exe, .dll (release et debug)
doc : toute la documentation sur le moteur CreaEngine
examples : les solutions de test du moteur
include : fichiers .h
lib : fichiers .lib et .pdb
src : projet .vcxproj et fichiers .cpp
tmp : fichiers temporaires ipch, Release et Debug

A l'aide de Visio, réalisez un Diagramme de composantes qui montre cette organisation
n.b : Les formes sont dans « Logiciel → Application d'entreprise »

Documentation

- Organisez l'arborescence de vos ressources (répertoires, .sln, .vcxproj, .cpp, .h) avec l'explorateur.

- Dans Visual Studio, vous pouvez organiser vos ressources en créant des filtres :
right-click → Ajouter → Nouveau filtre

- Pour ajouter des fichiers existants :
right-click → Ajouter→ Élément existant

- C'est dans les propriétés du projet que l'on peut organiser la compilation et la création de l'exe.

$(SolutionDir) : Répertoire du .sln
$(ProjectDir) : Répertoire du .vcxproj
$(OutDir) : Répertoire du .exe ou .dll

Général → Répertoire de sortie : répertoire ou sera créé le .exe ou le .dll
Général → Répertoire intermédiaire : répertoire ou seront créés les fichiers temporaires
Général → Nom de la cible : nom donné à votre .exe ou le .dll (pour différencier Debug et Release)
Général → Type de configuration : Application (.exe) ou Bibliothèque Dynamique (.dll)

Débogage → Répertoire de travail : Répertoire à partir duquel est lancé le .exe (comme un double-click sur un .exe)

C/C++ → Général → Autres répertoires Include : Répertoires où seront lus les fichiers .h
C/C++ → Définitions de préprocesseur : pour les #define (zex : CREAENGINE_EXPORTS)
C/C++ → En-tête précompilé : Utilisation (/Yu) ou Création (Yc) pour créer le stdafx.h

Éditeur de liens → Général → Fichier de sortie : Répertoire et non du fichier .lib
Éditeur de liens → Général → Répertoires de bibliothèques supplémentaires : Répertoires où seront lues les .lib
Éditeur de liens → entrée → Dépendances supplémentaires : Noms des .lib à ajouter/lier
Éditeur de liens → Débogage →Génération d'un fichier de base de données du programme : Répertoire et nom du fichier database .pdb
Éditeur de liens → Optimisation→Base de données guidée par profil: Répertoire et nom du fichier .pgd
Éditeur de liens → Avancé→Bibliothèque d'importation: Répertoire et nom du fichier .lib

Evènements de build → Evènement post-build → Ligne de commande : Ligne de commande qui peut être exécutée à la fin de la compilation, pour copier des fichiers par exemple :

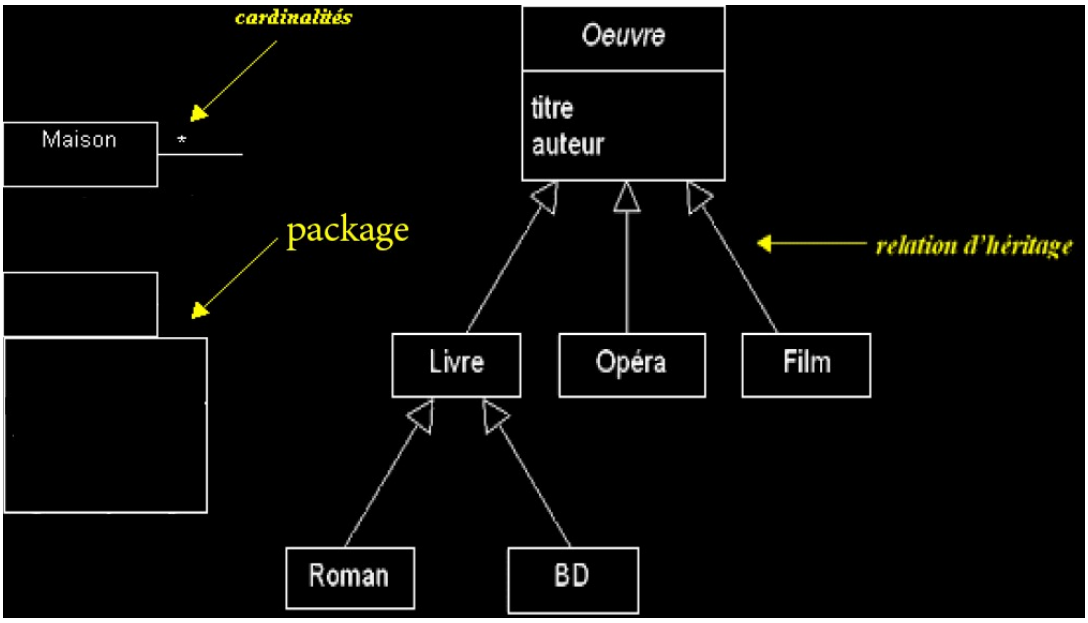
```
xcopy /y /d "$(ProjectDir)*.dll" "$(OutDir)"
```

- N'oubliez pas de configurer Debug et Release !

- Pour éviter les warnings, ajouter dans stdafx.h :

```
#pragma warning( disable: 4251 )
```

UML

Diagramme de classe type	
Diagramme de composants type