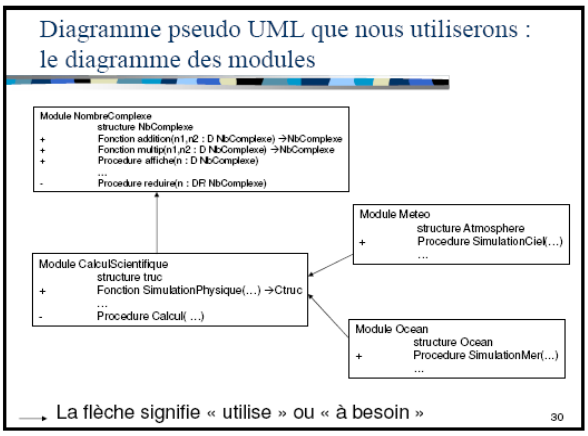
# TD Conception




Rappel du cours : un exemple de graphe des modules

## Exercice 1

On désire stocker un ensemble de voitures dans un conteneur de type tableau dynamique. Une voiture est caractérisées par une marque, un modèle, une date de mise en circulation, un kilométrage, une couleur, une liste de propriétaires présents et passés, etc. Les fonctionnalités désirées sont : ajouter une voiture, rechercher la plus veille, celle avec le plus de km, calculer la moyenne d'age, etc. Ecrivez le diagramme des modules correspondant.

## Exercice 2

Une médiathèque vous demande de développer son logiciel de gestion des emprunts. Les employés de la médiathèque vous expliquent qu'ils aimeraient pouvoir gérer :

- Une liste d'emprunteurs avec leur nom, prénom, adresse (pour les retrouver en cas de problème) et leur age (à des fins de statistique).
- Une liste de média avec le titre, l'auteur et le type : CD, livres, cassettes vidéo ou DVD.
- Un historique des emprunts : la personne qui emprunte, le média emprunter, la date de sortie et la date de retour (ou la date de retour prévu si le média est encore sorti).

Vous êtes plusieurs développeurs sur ce projet, en tant que chef de projet vous devez spécifier au mieux les différents modules nécessaires et leurs interactions afin que chaque développeurs puissent commencer à travailler. Ils ont besoin du graphe de dépendance et de la spécification de tous les modules (fonctions/procédures/types ; et leur utilisation interne ou externe).

## Exercice 3

La scolarité de la Faculté des Sciences et Technologies de Lyon 1 souhaite informatiser la gestion des étudiants, des enseignants et des UEs. Vous êtes chargés de développer une application en C. Il faut notamment pouvoir représenter :

- l'ensemble des UEs proposées (code, intitulé, contenu, n° séquence, nombre de crédits ECTS, nombre de CM/TD/TP, …)

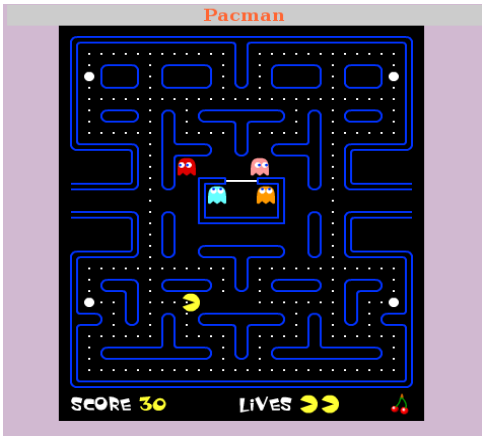
- l'ensemble des enseignants intervenants (coordonnées, n° employé, statut, …)
- la liste des étudiants inscrits dans l'UFR (coordonnées, n° étudiant, …)
- quel étudiant est (ou a été inscrit) à telle UE pendant tel semestre. On souhaite également faire un suivi des étudiants (notes, absences, remarques éventuelles, …)
- quel enseignant intervient (ou est intervenu) dans telle UE pendant tel semestre

Ecrivez le diagramme des modules, en détaillant les champs des structures. Faîtes apparaître les dépendances entre modules.

## Exercice 4

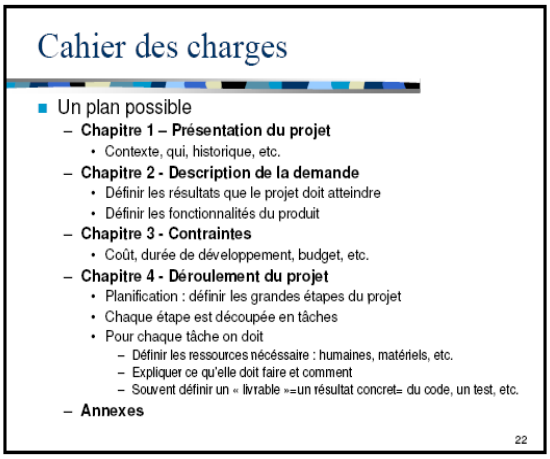
Vous êtes chef du projet **Pacman**. Pour le cahier des charges :

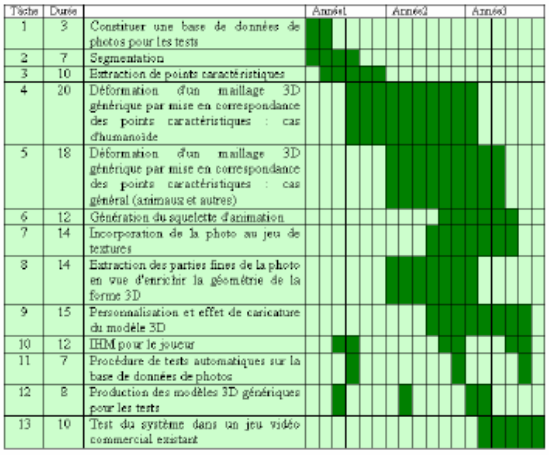
- Faites le diagramme des modules
- Découpez le projet en sous-tâches et faites le diagramme de Gantt




Rappel du cours :

- un plan possible de cahier des charges (à gauche)
- le chapitre 4 contient le diagramme de Gantt (exemple à droite) et le diagramme des modules



# Corrigé – Médiathèque

## Liste des modules

Champs ou fonctions noté(e)s entre [] : facultatif

**Module Date**

Structure Date {
jour : entier dans [1…31]
mois : entier dans [1…12]
annee : entier
}
+ getJourDate(IN Date) -> entire
+ setJourDate(IN-OUT Date, IN entier)
+ getMoisDate(IN Date) -> entire
+ setMoisDate(IN-OUT Date, IN entier)
+ getAnneeDate(IN Date) -> entire
+ setAnneeDate(IN-OUT Date, IN entier)
[ +getChaineDate(IN Date) -> chaine ]

**Module Media**

Structure Media {
num_media : entier
titre : chaine
auteur : chaine
type : entier dans {LIVRE, CD, DVD, VHS}
}
+ initMedia(IN-OUT Media)
+ libereMedia(IN-OUT Media)
+ getNumMedia(IN Media) -> entier
+ setNumMedia (IN-OUT Media, IN entier)
+ getTitreMedia(IN Media) -> chaine
+ setTitreMedia (IN-OUT Media, IN chaine)
+ getAuteurMedia(IN Media) -> chaine
+ setAuteurMedia (IN-OUT Media, IN chaine)
+ getTypeMedia(IN Media) -> entier
+ setTypeMedia (IN-OUT Media, IN entier)

**Module Personne**

Structure Media {
num_personne : entier
nom : chaine
prenom : chaine
adresse : chaine
age : entier
}
+ initPersonne(IN-OUT Personne)
+ liberePersonne(IN-OUT Personne)
+ getNumPersonne(IN Personne) -> entier
+ setNumPersonne (IN-OUT Personne, IN entier)
+ getNomPersonne(IN Personne) -> chaine
+ setNomPersonne (IN-OUT Personne, IN chaine)
+ getPrenomPersonne(IN Personne) -> chaine
+ setPrenomPersonne (IN-OUT Personne, IN chaine)
+ getAdressePersonne(IN Personne) -> chaine
+ setAdressePersonne (IN-OUT Personne, IN chaine)
+ getAgePersonne(IN Personne) -> entier
+ setAgePersonne(IN-OUT Personne, IN entier)

**Module Emprunt**

Structure Emprunt {
[num_emprunt : entier]
num_media : entier
num_personne : entier
date_debut : Date
date_fin : Date
}
+ initEmprunt(IN-OUT Emprunt)
+ libereEmprunt(IN-OUT Emprunt)
+ getNumMediaEmprunt(IN Emprunt) -> entier
+ setNumMediaEmprunt (IN-OUT Emprunt, IN entier)
+ getNumPersonneEmprunt(IN Emprunt) -> entier
+ setNumPersonneEmprunt (IN-OUT Emprunt, IN entier)
+ getDateDebutEmprunt(IN Emprunt) -> Date
+ setDateDebutEmprunt (IN-OUT Emprunt, IN Date)
+ getDateFinEmprunt(IN Emprunt) -> Date
+ setDateFinEmprunt (IN-OUT Emprunt, IN Date)

**Module ListeMedias**

Structure ListeMedias {
tab_Medias : tableau de Media
nb_Medias : entier
}
+ initListeMedias(IN-OUT ListeMedias)
+ viderListeMedias(IN-OUT ListeMedias)
+ getNbMedias(IN ListeMedias) -> entier
+getMediaListe(IN ListeMedias, IN entier) -> Media
+ ajouterMediaListe(IN-OUT ListeMedias, IN Media)
+supprimerMediaListe(IN-OUT ListeMedias, IN entier)

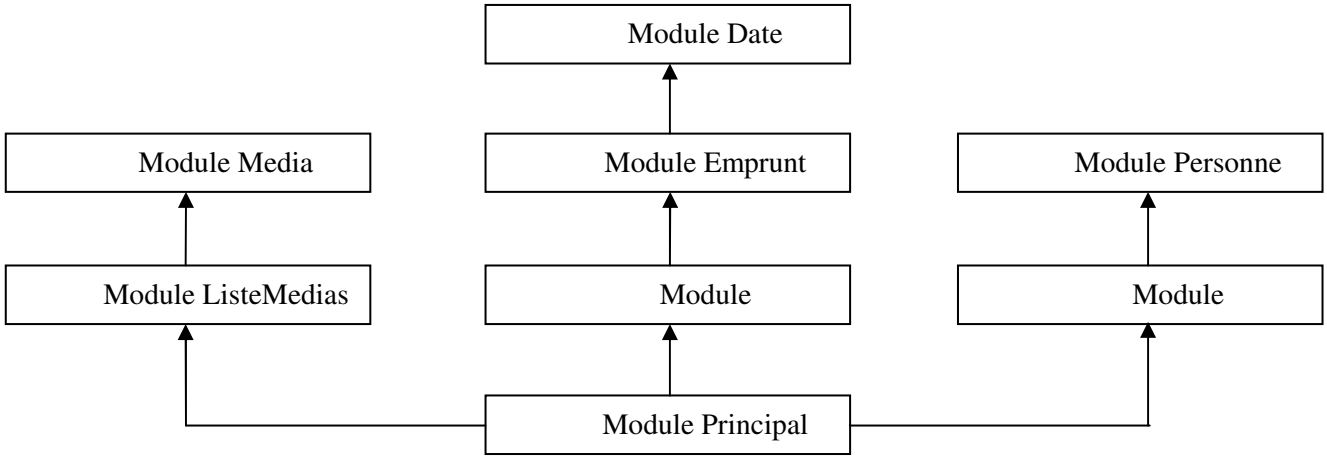
**Module ListePersonnes**

Structure ListePersonnes {
tab_Personnes : tableau d'Personne
nb_Personnes : entier
}
+ initListePersonnes(IN-OUT ListePersonnes)
+ viderListePersonnes(IN-OUT ListePersonnes)
+ getNbPersonnes(IN ListePersonnes) -> entier
+getPersonneListe(IN ListePersonnes, IN entier) -> Personne
+ ajouterPersonneListe(IN-OUT ListePersonnes, IN Personne)
+supprimerPersonneListe(IN-OUT ListePersonnes, IN entier)

**Module ListeEmprunts**

Structure ListeEmprunts {
tab_emprunts : tableau d'Emprunt
nb_emprunts : entier
}
+ initListeEmprunts(IN-OUT ListeEmprunts)
+ viderListeEmprunts(IN-OUT ListeEmprunts)
+ getNbEmprunts(IN ListeEmprunts) -> entier
+getEmpruntListe(IN ListeEmprunts, IN entier) -> Emprunt
+ ajouterEmpruntListe(IN-OUT ListeEmprunts, IN Emprunt)
+supprimerEmpruntListe(IN-OUT ListeEmprunts, IN entier)

**Diagramme**


Module Date
Module Media
Module Emprunt
Module Personne
Module ListeMedias
Module
Module
Module Principal

# Pacman en folie

Une version légère de cahier des charges

## 1. Présentation des acteurs

La société InfogamesProf est une Société Anonyme au capital de 1000000 d'euros. Son activité principale est l'édition de jeu vidéo sur PC. blahblah...

La société UnivGamesEtu est une start-up issue de l'université Lyon 1. Elle est composée de 4 développeurs à pleins temps. Son activité principale est le développement de produits informatiques pour des grands groupes. Son expertise est essentiellement centrée sur le C/C++ et le développement de listes chaînées et de tableau dynamique. blahblah ...

Dans ce contexte, la société InfogamesProf commande le produit « Pacman en folie » à la société UnivGamesEtu. Le présent cahier des charges est composé

- d'une description détaillée du produit,
- une prévision détaillée du déroulement du développement
- un diagramme de Gantt
- un diagramme des modules

L'ensemble des ces documents sera maintenu pendant toute la durée de développement du produit.

## 2. Description du produit

### Principe et règles du jeu

Le jeu commence en cliquant sur l'écran de jeu. Le but du jeu est de collecter tous les points dans chaque niveau. Pendant ce temps on est suivi par plusieurs fantômes qui ne doivent pas nous toucher.

Pacman - son pion - est déplacé grâce aux touches flèche. Partout où il passe, il mange les points jaunes. Lorsque tous les points ont été mangés, le niveau est terminé et on passe au suivant. Pacman se transforme en super-pacman en mangeant les pastilles vertes. Super-pacman peut manger les fantômes. Super-pacman redevient pacman après N secondes. Une fois toutes les pastilles jaunes mangées, le jeu passe au niveau suivant. Les fantômes ont un déplacement automatique en direction de pacman pour le manger ou alors en direction opposé à super-pacman pour ne pas se faire manger.

Le jeu se déroule avec une musique et des bruitages en fonction de qui mange qui.

Fin de la partie : Le jeu se termine lorsque l'on a perdu ses trois vies.

### Détail de l'interface et des niveaux du jeu

Voici un exemple du niveau 1


Croquis des niveaux (au stylo par exemple) ...
Description de la gestion des scores

## 3. Contraintes

- Le jeu sera développé en C/C++ sous Linux
- Les librairies utilisées seront SDL et éventuellement ncurses pour une version alpha en mode texte
- Le code respectera le coding standard suivant : code indenté, variable ayant du sens, etc.
- Le code sera géré et archivé par SVN
- La documentation du code sera produite par Doxygen.
- Un diagramme des modules permettra d'avoir une vision de haut niveau de l'implantation.
- L'équipe de développement utilisera les outils de debug et de profiling : gdb, valgrind, gprof, etc.
- Le code sera fourni à la société InfogamesProf.

## 4. Déroulement du projet

### Tâche 0 : rédiger le cahier des charges

Membres impliqués : tous
Durée : 2 semaines

### Tâche 1 : définir le diagramme des modules

Membres impliqués : tous
Durée : 1 semaines

### Tâche 2 : développement d'un 1er prototype en mode texte

Durée : 3 semaines

**Tâche 2.1** : écriture et test du module 'Pacman.h/.cpp'
Membres impliqués : Bob Lecodeur
Un pacman est défini par une position, un état, un score, blahblah

**Tâche 2.2** : écriture et test du module 'Terrain.h/.cpp'
Membres impliqués : Pierre Lerouleur et Joe Lefort
Ce module pourra charger un terrain depuis un fichier. Le format d'un fichier de données respectera la forme suivante : blahblah

**Tâche 2.3** : écriture et test du module 'Fantôme.h/.cpp'
Membres impliqués : XXX
Définition des structures pour un Fantome et DesFantomes.
blahblah

**Tâche 2.4** : écriture et test du module 'Jeu.h/.cpp'
Membres impliqués : XXX
Ce module propose toutes les fonctions pour gérer une étape du jeu :
- reçoit en entrée ce qu'un utilisateur a tapé au clavier
  => définir une structure pour ça
  Déplace le pacman en utilisant le module Pacman et Terrain
- déplacement d'un fantôme en utilisant le module Fantôme et Terrain
- Mise à jour du score
- Test de fin de niveau

Attention : ce module ne fait aucun affichage. Il doit pouvoir servir à la version texte comme à la version graphique.

**Tâche 2.5** : écriture du module 'BoucleJEU-TXT.h/.cpp'
Membres impliqués : XXX
Récupère les données de Terrain, Pacman et Fantome et les affiches au format texte.
blahblah

**Tâche 2.6** : mise en commun des modules et tests
blahblah

## Tâche 3 : développement d'un 2e prototype en mode graphique (SDL)

Durée : 4 semaines

**Tâche 3.1** : exploration et compréhension de SDL
blahblah

**Tâche 3.2** : Affichage de sprites, tests de performance
blahblah

**Tâche 3.3** : Développement d'une boucle de jeu en SDL et intégration au reste du jeu.
blahblah

**Tâche 3.4** : Test et debug
blahblah

**Tâche 4 : développement d'un 3e prototype en mode graphique avec la musique et gestion des score**

blahblah

**Tâche 5 : développement d'un 4e prototype avec ajout d'un mode multi-joueurs**

blahblah

**Tâche 6 : développement d'un 5e prototype multi-joueurs en réseau**

blahblah

## 5. Diagramme de Gantt

| | Semaine1 | Semaine2 | Semaine3 | Semaine4 | Semaine5 | Semaine6 | Semaine7 |
|---|---|---|---|---|---|---|---|
| Tâche 0 | X | | | | | | |
| Tâche 1 | X | X | | | | | |
| Tâche 2.1 | | X | X | | | | |
| Tâche 2.2 | | X | X | | | | |
| Tâche 2.3 | | X | X | | | | |
| Tâche 2.4 | | | X | X | | | |
| Tâche 2.5 | | | X | X | | | |
| Tâche 3.1 | | X | X | X | | | |

ETC.

## 6. Diagramme des modules