3ETI, Développement logiciel en C

Présentation du projet:

Le but du projet est de réaliser le fonctionnement d'un jeu de Siam valide.
Plus spécifiquement, il consiste à implémenter l'organisation générale du jeu, et le suivi des règles du mouvement des pièces.

Le projet mettra en avant un ensemble de bonnes pratiques de codage permettant de simplifier la lecture du code, son évolutivité et sa robustesse.
Des aspects généraux de l'informatique seront également mises en pratiques (compilation, lignes de commande, scripts, etc).

Organisation du code source:

Un certain nombre de fichiers sources vous sont fournis. Le projet est lui-même formé de deux parties comportant chacun un exécutable.

- Le répertoire **visualiseur/**:
  Permet de visualiser l'état d'un jeu. Il contient un programme indépendant de votre projet qui permet de visualiser le placement des pièces sur le plateau en venant lire un fichier sur le disque dur mis à jour lors d'une partie.

  Notez que ce programme ne contient aucune *intelligence* au niveau de la connaissance des règles du jeu, il s'agit uniquement d'un visualiseur d'un état donné.

  Ce code est déjà complet, il ne fait pas partie du cadre du projet à compléter. Il peut être utilisé en tant qu'exécutable simple.
  Pour les étudiants intéressés, le code source vous est entièrement fourni. L'ensemble du code est écrit en C, et il utilise la librairie graphique OpenGL ainsi que le gestionnaire de fenêtre Glut (pour la partie 3D).

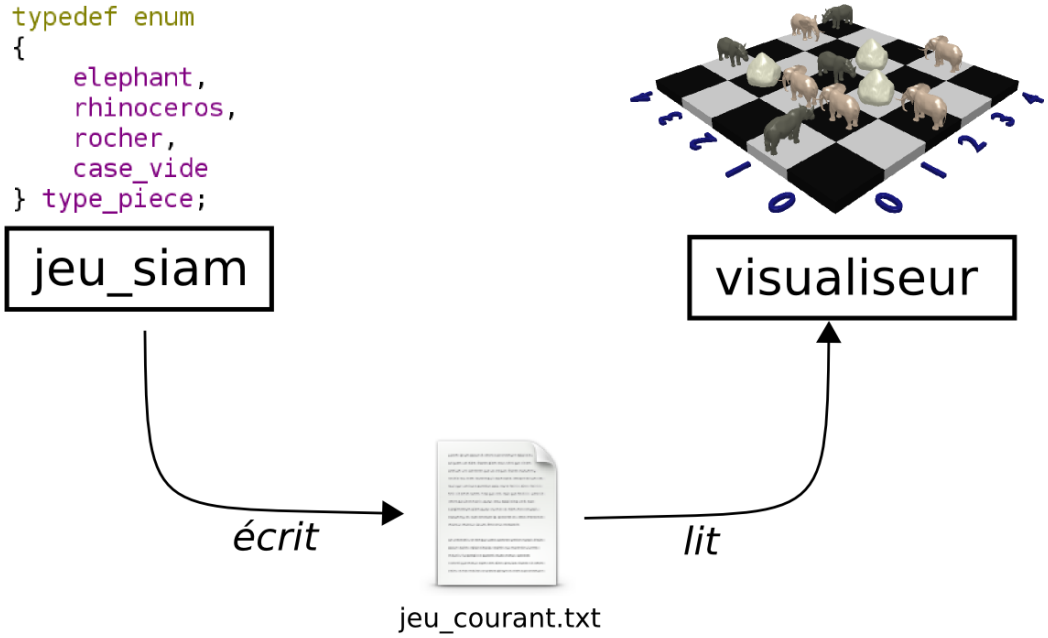
- Le répertoire **src/**:
  Contient l'ensemble des fichiers permettant de manipuler effectivement le jeu.

  Il contient le code permettant de
  - manipuler le jeu,
  - analyser le suivi des règles du jeu lors de la demande d'action sur une pièce,
  - communiquer avec l'utilisateur par le biais de la ligne de commande.
  - stocker l'état d'un jeu sur le disque dur.

  C'est cette partie du code que vous allez compléter lors de ce projet.

La communication entre ces deux programmes (visualiseur et jeu) est réalisée à l'aide d'un fichier texte écrit sur le disque dur.

- L'exécutable du jeu écrit à chaque coup dans ce fichier.
- Le visualiseur vient quant à lui lire ce même fichier lorsqu'il est modifié.



Organisation du projet:

Le code source du projet est réparti en différent fichiers.
Chaque fichier représente un niveau **d'abstraction** spécifique.

Au plus haut niveau d'abstraction, l'utilisateur peut définir une action sur une pièce (introduction, déplacement) étant donné l'information de coordonnées (x,y) et d'orientations.

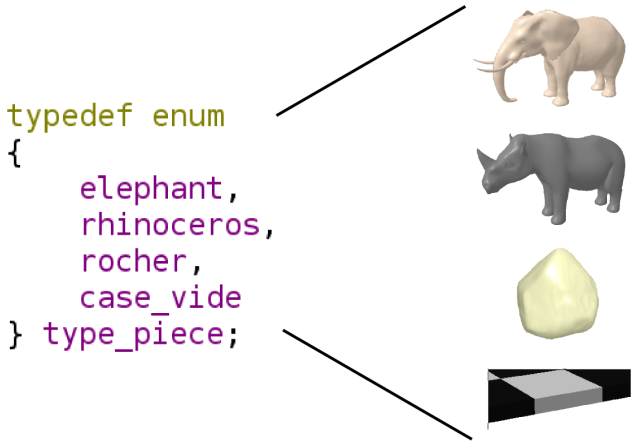
Ces fonctions d'appels de haut niveau qui permettent de manipuler l'échiquier sont situées dans un un fichier spécifique dénommé **API** (Application Programming Interface).

Au niveau d'abstraction plus bas, nous viendrons spécifiquement stocker les pièces en mémoire sous forme de struct, et un jeu sera formé d'un tableau de pièces.

Le détail de ces niveaux d'abstractions est fourni-ci après.

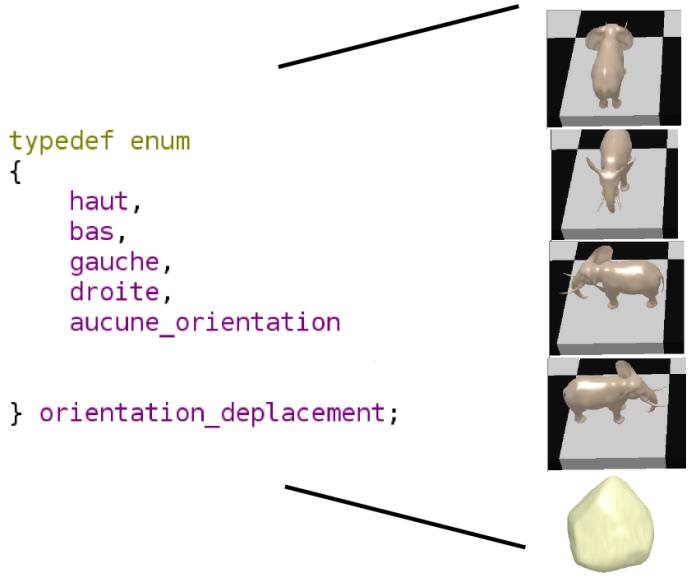
Type_piece

Le premier niveau d'abstraction est celui représentant le type d'une pièce.
Une pièce est une enum C définissant si une pièce est de type rhinocéros, éléphant, rocher, ou case vide.



Orientation_deplacement

L'orientation_deplacement définit la direction d'une pièce. Une direction est une enum C utilisé pour indiquer soit la direction d'un déplacement, soit l'orientation d'une pièce.



Piece_siam

```
typedef struct
{
    type_piece type;
    orientation_deplacement orientation;

}piece_siam;
```

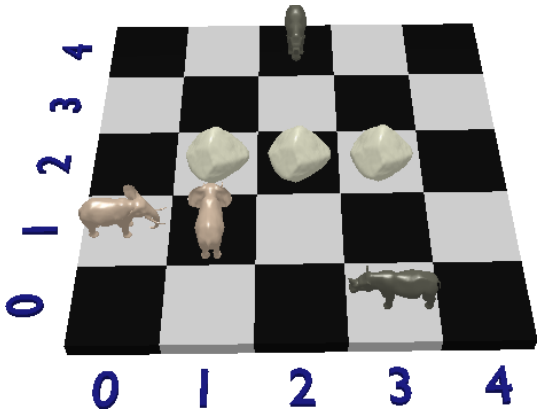
Une pièce du jeu correspond à une case du plateau. Chaque case est caractérisé par son type et son orientation potentielle.

Plateau_siam

Un plateau de siam est une grille de 5x5 cases, où chaque case contient une pièce. Ici, la grille est implémentée en mémoire comme un tableau statique à deux entrées (x et y).

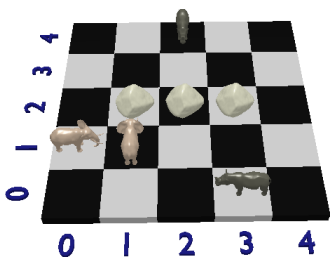
```
typedef struct
{
    piece_siam piece[NBR_CASES][NBR_CASES];

} plateau_siam;
```



Jeu_siam

Un jeu de siam contient un plateau plus l'information du joueur courant.



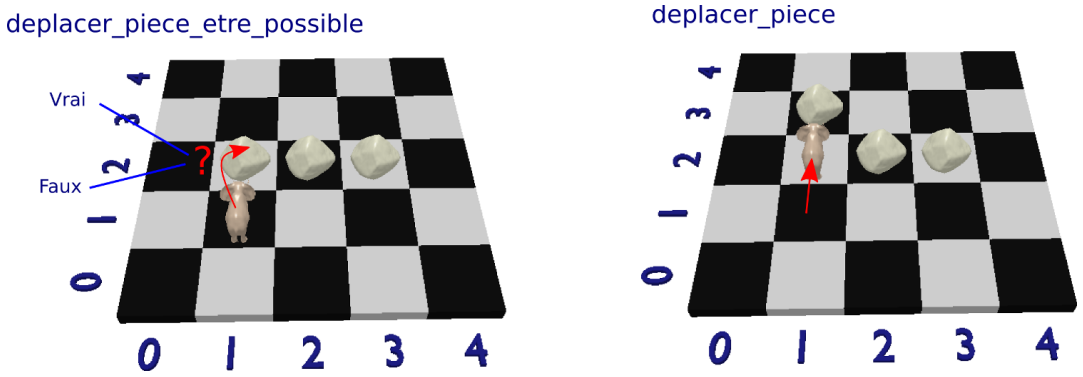
+ Joueur courant

Plateau_modification:

Plateau modification est une abstraction qui permet ou non de valider une action sur une pièce du jeu en fonction des règles.
C'est en quelque sorte la partie *intelligence* du jeu.
Notez qu'il ne s'agit pas de la modélisation d'une entité physique telle qu'une pièce du jeu.

Contrairement aux structures de stockages précédentes, cette abstraction est supposée être de haut-niveau. Elle n'a pas à interagir directement avec le tableau statique contenant les pièces, et doit pouvoir être indépendante du type de stockage.

Elle doit offrir par contre une interface simple telle que: déplacement (ou introduction) de la pièce située aux coordonnées (x0,y0) dans une direction donnée.
L'abstraction fournie un niveau permettant de vérifier si une action est possible, et un autre niveau permettant d'appliquer celle-ci sur le plateau.



API:

L'API est l'interface de plus haut niveau fournie par une librairie. C'est cette interface qui est utilisée lorsqu'un développeur veut utiliser une librairie donnée afin de l'interfacer avec son propre code.
Dans notre cas, l'API représente le niveau d'abstraction de l'intention (ou la main) de l'utilisateur vis à vis du jeu.

L'API viens proposer une interface simple de haut niveau permettant de manipuler un jeu. Elle fait appel aux méthodes de plus bas niveau tel que plateau_modification.

Toutes les variables donnée en tant qu'argument des fonctions de l'API doivent être génériques (ex. Coordonnées x et y).
Elle ne doivent pas dépendre du type de la structure de stockage par exemple.

Entree sortie :

Entree sortie est un niveau d'abstraction permettant de faire le lien entre l'échiquier existant en RAM et un état d'échiquier stocké sur le disque dur.
Il propose une interface de lecture et d'écriture dans des fichiers du disque.

Il s'agit principalement de concentrer la partie spécifique à l'écriture sur le disque dans une abstraction spécifique afin d'offrir une interface de sauvegarde et de chargement simple pour les autres partie du programme.

De même, il s'agit d'une structure qui ne doit pas dépendre du type de stockage du jeu d'échec en interne. On considère que les appels de cette abstraction sont de haut-niveau.

Parseur mode interactif:

Tout comme entree_sortie, ce niveau d'abstraction vient principalement décharger l'API de la problématique de la validation et de l'interprétation d'une chaîne de caractère donnée par l'utilisateur lors du dialogue en ligne de commande.

Le parseur vient tout d'abord analyser si la chaîne de caractère est valide au niveau de sa syntaxe.
Ensuite, il vient interpréter ce message.
Enfin, il renvoie une structure contenant les informations de ce message.

Note: Parseur est un terme commun d'informatique désignant l'analyse syntaxique d'un texte.
(ex. Parseur xml, parseur de ligne de commande, etc).

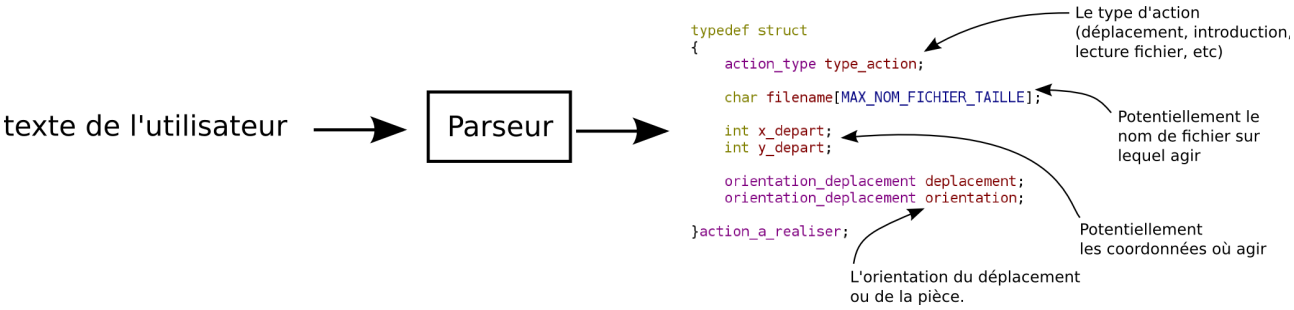


Diagramme des niveaux d'abstractions:

Le diagramme suivant résume l'organisation (simplifiée) des différents niveaux d'abstractions utiles au début du projet ainsi que leurs relations.