TD #1 — Birds in the storm

Le contexte

L'objectif de ce TP est de développer 3 classes simples afin de mettre en œuvre un jeu similaire à *Birds in the storm* de la première semaine en C. Le diagramme de classes est donné et doit être respecté afin d'implémenter la spécification imposée.

Le jeu

Le terrain. Le terrain de jeu est représenté par un rectangle de largeur L cases et de hauteur H cases, où chaque case correspond à un carré de $1m \times 1m$.

Les ennemis. Sur ce terrain sont disposés x ennemis en haut d'une tour, qui sera représentée par le symbole X dans la grille. Ces ennemis sont de deux sortes :

1. `E1` est un ennemi qui est détruit lorsqu'il est touché. Il est représenté par le symbole @ en haut de sa tour. Cet ennemi rapporte 10 points lorsqu'il est détruit.
2. `E2` est un ennemi qui lorsqu'il est touché voit sa tour juste détruite au-dessus du tir. Il tombe donc de quelques cases et n'est détruit que si on touche le bas de sa tour. Il est représenté par le symbole ˜ en haut de sa tour. Cet ennemi rapporte 10 points lorsqu'il tombe et 20 points supplémentaires quand il est finalement détruit.

Lors de la mise en place du jeu, le choix du type des ennemis est aléatoire.

Les oiseaux. Pour détruire des ennemis des oiseaux sont à disposition. Ils vont suivre une trajectoire qui dépend du choix de l'oiseau. Lors du jeu c'est le joueur qui décide de l'oiseau qu'il souhaite lancer. Ils sont de trois types.

1. *Red* est un oiseau qui suit une trajectoire parabolique selon la formule
$$y(x) = \frac{g}{2v2_{0x}} \times x^2 + \frac{v_{0y}}{v_{0x}} \times x \quad (1)$$
Le joueur donne donc l'angle et la vitesse associés à l'oiseau et lorsqu'il choisit de jouer avec *Red* il n'a aucun point de malus.
2. *Jay* est un oiseau qui suit une trajectoire oblique à partir d'un point $(0, i)$ dans la grille où i est choisi aléatoirement par le jeu. Arrivé en haut de la grille, l'oiseau rebondit et redescend également selon une oblique. Les équations sont les suivantes
$$y_1(x) = ax + b_1 \quad (2)$$
$$y_2(x) = -ax + b_2 \quad (3)$$
et le joueur doit donner l'angle de la droite à partir duquel b_1 est calculé de façon à ce que le point $(0, i)$ appartienne à la droite. Le point b_2 est calculé afin que le premier point de cette droite soit le dernier point de la droite précédente. Si le joueur utilise cet oiseau, il a un malus de 10 points.
3. *Chuck* le dernier oiseau suit une trajectoire horizontale dont la hauteur est choisi aléatoirement par le jeu. Il rapporte un malus de 20 points.

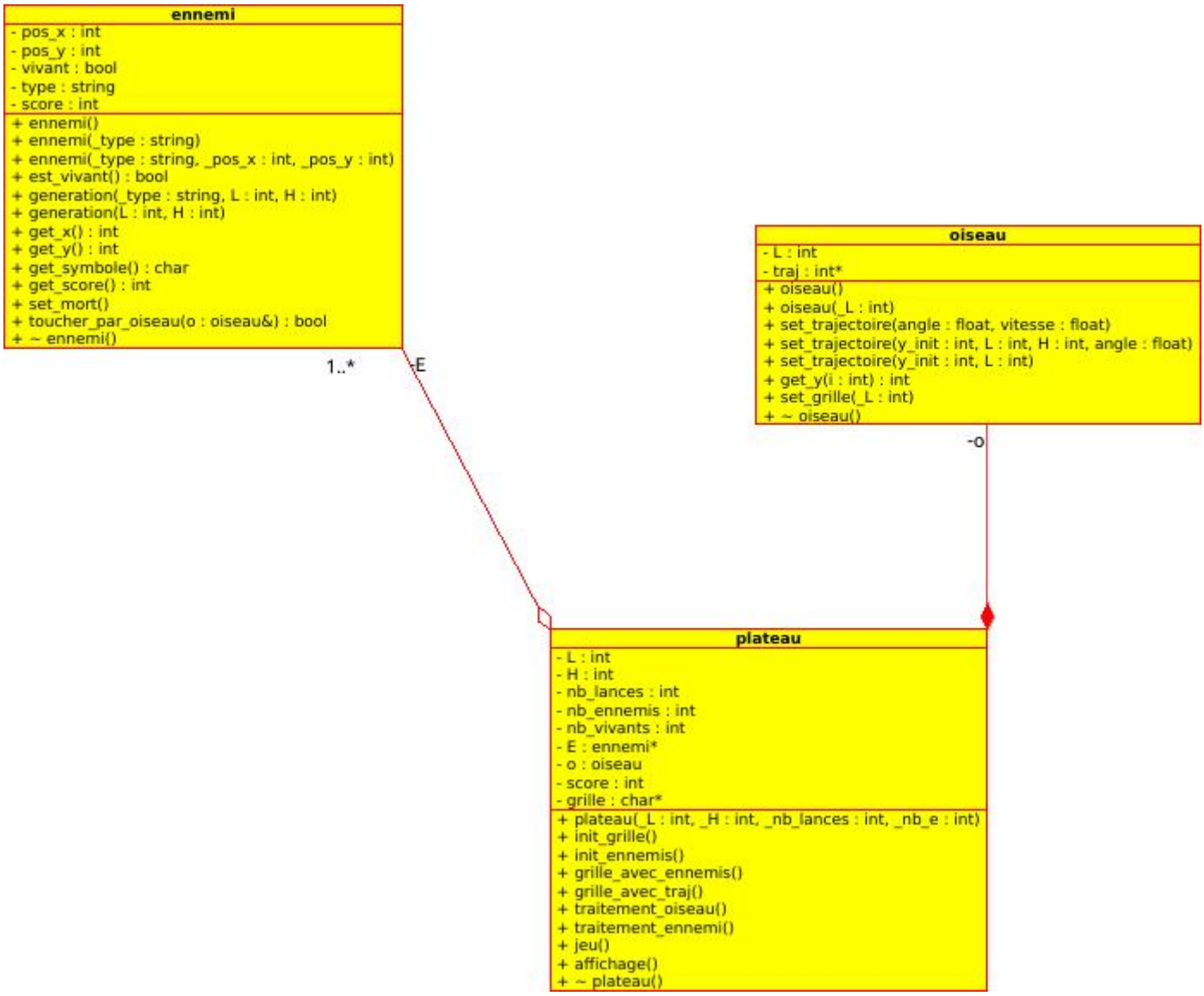
Le déroulé du jeu consiste à :

1. afficher la grille avec les ennemis,
2. choisir l'oiseau à lancer,
3. afficher la nouvelle grille faisant apparaître la trajectoire de l'oiseau
4. afficher la nouvelle grille ou le score final si tous les ennemis ont été détruits. Si le nombre de lancers est dépassé, un message indiquant que le joueur a perdu est affiché.

L'implémentation .

Vous devez implémenter les 3 classes ainsi qu'un programme principal permettant de lancer et de vérifier le bon fonctionnement du jeu. Il est également nécessaire de valider la gestion de la mémoire avec valgrind qui doit renvoyer comme unique erreur *still reachable : 72,704 bytes in 1 blocks* dûe la librairie STL.

Le diagramme de classes.



La classe `Ennemi`. Un ennemi est défini par une position en x et en y, un booléen indiquant s'il est vivant ou non, un type pour différencier les différents ennemis possibles et un score initialisé à 0. Les méthodes -en plus des accesseurs (*getters*) et mutateurs (*setters*) classiques- sont la génération aléatoire d'une position et la méthode *toucher_par_oiseau* qui renvoie un booléen indiquant si la trajectoire d'un oiseau rencontre la tour de l'ennemi. Cette méthode met également à jour le score et le statut vivant ou non de l'ennemi. Avant de commencer l'implémentation, voici quelques questions :

1. Pourquoi un constructeur sans argument ?
2. Pourquoi deux méthodes *generation* (à voir par rapport à l'initialisation du jeu) ?
3. Pourquoi un passage par référence dans la méthode *toucher_par_oiseau* ?

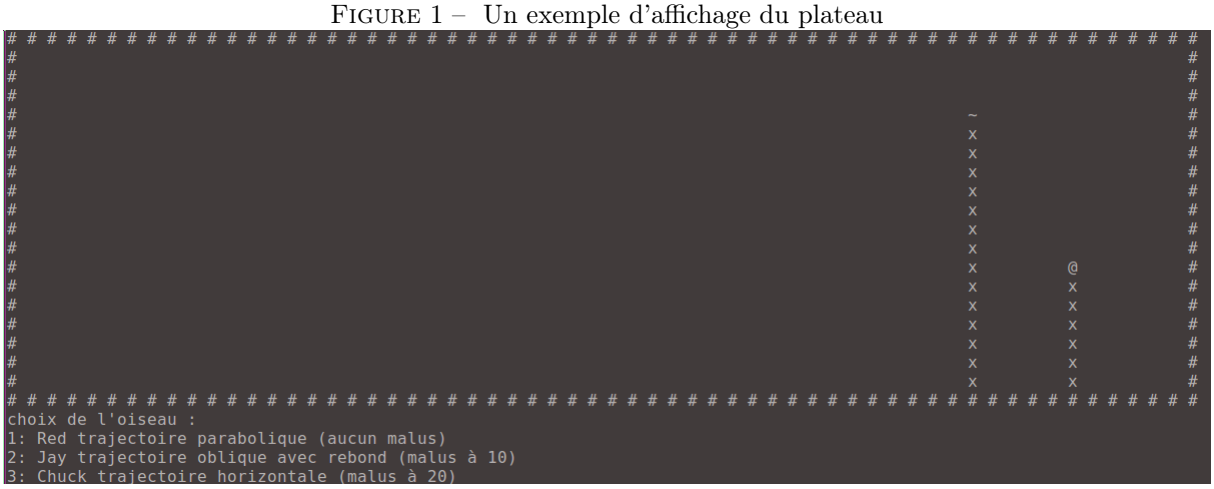
La classe *oiseau* Un oiseau est défini par un entier L et un tableau d'entiers permettant de définir sa trajectoire. Cette classe ne gère pas les différents types des oiseaux et ce type sera géré directement dans la classe suivante *plateau*. Les méthodes demandées consistent essentiellement à définir différentes trajectoires. A noter que *set_ grille* permet juste d'allouer la taille de l'attribut *traj* de l'oiseau. Avant de commencer l'implémentation, voici trois questions :

1. Pourquoi trois fonctions *set_ trajectoire* ?
2. Pourquoi un attribut L ?
3. En quoi consiste le destructeur ?

La classe *plateau* Cette classe permet de construire et d'exécuter le jeu. Les différents éléments du jeu se retrouvent dans les attributs :

— L, H pour la taille de la grille
— Un seul attribut *oiseau* est utilisé. Son type et -surtout- sa trajectoire seront différenciés lors du déroulement du jeu.
— *nb_ lancers* permet de définir le nombre d'oiseaux à disposition pour détruire les ennemis.
— E est un tableau d'ennemis en fonction de l'attribut *nb_ ennemis*.
— *o* est l'objet oiseau utilisé pour les différents lancers.
— *score* permet de calculer au fur et à mesure des lancers le gain du joueur.
— *grille* est le plateau de jeu. Pour définir les différents symboles pour représenter les différents éléments, n'hésitez pas à utiliser des #DEFINE.

FIGURE 1 – Un exemple d'affichage du plateau



Pour les méthodes, le constructeur permet d'initialiser les valeurs du plateau (nombre de lancers, nombre d'ennemis, etc). La méthode *init_ ennemis* permet de créer les *nb_ ennemis* et d'initialiser le tableau E sachant que le choix du type d'ennemis est un choix aléatoire. Attention à ne pas générer deux ennemis au même endroit. La méthode *init_ grille* permet d'initialiser l'attribut *grille* avec des bords et les ennemis vivants en appelant la méthode *grille_ avec_ ennemis*. La méthode *grille_ avec_ traj* met à jour l'attribut *grille* avec la trajectoire de l'oiseau que le joueur est en train de lancer. Enfin *traitement_ oiseau* et *traitement_ ennemi* permettent respectivement de lancer l'oiseau après avoir interagi avec le joueur pour qu'il définisse celui qu'il souhaite jouer, et de calculer si cet oiseau touche un ou des ennemis. La méthode *affichage* est juste une fonction d'affichage de l'attribut *grille*.

Enfin, la méthode *jeu* implémente le déroulement du jeu sachant qu'à chaque itération,

1. La grille initiale est affichée avec les ennemis restants,
2. Le joueur choisit un oiseau et les paramètres associés à son lancer afin de pouvoir calculer la trajectoire,
3. La grille complète est affichée, ainsi que le résultat des ennemis touchés et du score courant.

La figure 1 donner une illustration de l'affichage et de l'étape 2 du déroulement du jeu.