# Mini-projet de développement C

FIP 1A – SIT151

## 1 Développer un jeu vidéo : *Comet busters!*

Dans les années 90, l'un des intervenants du module « SIT151 » jouait de temps à autre au jeu *Comet busters!* sur ces bêtes de courses qu'étaient les machines dotées de processeurs Intel 286 [1] et 386 [2]. Les équipements ainsi que les systèmes d'exploitation ayant légèrement évolué depuis que l'intervenant a mis sa machine [3] au rebus, nous vous proposons de réimplémenter un tel jeu. Pour vous donner une idée du jeu original, vous pouvez jeter un œil sur les sites suivants :

— `http://www.old-games.com/download/1941/comet-busters-`
— `https://en.wikipedia.org/wiki/Comet_Busters!`
— `https://archive.org/details/CometBusters_1020`

Une preuve de concept a été implémentée par les intervenants du module pour vous éviter de lire un long descriptif formel. Une démonstration vous sera faite en séance.

## 2 Principes généraux du jeu *Comet busters!*

**But du jeu :** Vous incarnez un pilote de vaisseau spatial et devez survivre au milieu de l'espace en détruisant et évitant les astéroïdes, ainsi que les éventuels extra-terrestres vous disputant l'espace intersidéral.

**Mécanique du jeu :** Le vaisseau spatial a un canon avant qui lui permet de détruire un astéroïde ou tout autre ennemi. Un astéroïde détruit se casse en plusieurs morceaux, jusqu'à ce que l'astéroïde soit suffisamment petit pour être détruit sans se diviser. Par défaut, il y a trois tailles de comètes (et donc deux divisions), et chaque division d'astéroïde en produit deux autres. Généralement, tous les objets célestes ont la même résistance : un seul projectile suffit à les briser. Les plus petits objets se déplacent plus rapidement et sont donc plus difficiles à toucher. Un extra-terrestre agressif traverse régulièrement l'écran.

**Déplacements :** Le vaisseau peut pivoter et se déplacer dans la direction vers laquelle il pointe. S'il atteint un bord de l'écran, il réapparaît de l'autre côté (maîtrise de l'hyperespace ou espace torique). Ainsi, voler vers la droite de l'écran fait réapparaître le vaisseau à gauche (et vice-versa). De même, faire voler le vaisseau en haut de l'écran le fait réapparaître en bas (et vice-versa).

**Score :** Chaque ennemi détruit rapporte des points. Plus l'astéroïde est petit, plus il rapporte lorsqu'on le détruit, car plus difficile à détruire (taille et vitesse).

## 3 Proposition initiale et contraintes

Nous vous proposons quelques réglages initiaux ainsi que des contraintes. Vous adapterez ensuite votre jeu au fur et à mesure du projet :

— sprites : nous vous proposons une série de sprites pour que vous ne perdiez pas de temps dessus. Cependant, si vous avez des compétences graphiques et du temps, rien ne vous empêche d'utiliser vos propres productions. Quelques propositions de conventions sur les tailles des objets :

1. `https://en.wikipedia.org/wiki/Intel_80286`
2. `https://en.wikipedia.org/wiki/Intel_80386`
3. `https://en.wikipedia.org/wiki/Amstrad_PC2286`

  - plateau : 640x480 ;
  - vaisseau : 32x32 ;
  - projectiles : 8x8 ;
  - astéroïdes : 64 / 32 / 16 ;
- réglages par défaut :
  - 5 vies ;
  - pas de résistance ou de barre de vie ;
  - pas de limitation du nombre de projectiles ;
  - scores : 20, 50 et 100 points pour respectivement les astéroïdes de grande, moyenne et petite taille ;
- vous devrez respecter des conventions de programmation cohérentes ;
- vos projets **doivent** pouvoir être compilés sur les systèmes GNU/Linux (au moins ceux de l'école), si vous travaillez sur Windows ou MacOS chez vous, prévoyez l'adaptation de votre code.

# 4 Travail à effectuer

Votre objectif général est de développer le logiciel afin qu'il soit « jouable ». Vous ne partez pas de rien : un prototype opérationnel écrit par les intervenants vous est fourni. Il compile et se lance, mais une structure de données est vide, ce qui ne donne pas le comportement attendu. Il n'est peut-être pas non plus exempt de *bugs*. Vous devez le modifier et l'améliorer.

**À la fin du mini-projet, vous livrerez votre travail dans l'espace Moodle dédié.** Vous livrerez une archive au format `.tar.gz` ou `.zip`, dont le nom contient votre (ou vos) nom(s). Vous modifierez le fichier `README` afin d'intégrer le(s) nom(s) du (des) contributeurs(s) ainsi que toute information pertinente (si vous avez modifié la manière d'exécuter, vos contributions, etc.). L'archive doit contenir le *Makefile* (éventuellement modifié), votre fichier de scores, le `README` ainsi que tout ce qui est nécessaire pour compiler et exécuter le logiciel.

De manière plus détaillée, voici différentes tâches à mener pour atteindre cet objectif :

1. Récupérez l'archive contenant les fichiers de départ sur Moodle (squelette de code, code à trous, *Makefile*, polices d'écriture, *sprites*, etc.).
2. Explorez le contenu de l'archive :
   (a) lisez le fichier `README` ;
   (b) identifiez le point d'entrée du programme ;
   (c) identifiez les structures de données (spécifications `.h` et implémentations proposées) ;
   (d) parcourez la physique du jeu (collisions, friction) ;
   (e) imprégnez-vous de la mécanique technique générale liée à la SDL(événements clavier, boucle principale, affichage, *sprites*, couleurs, etc.).
3. Réfléchissez aux structures de données :
   (a) la structure *sprites* est la structure essentielle pour décrire les objets du jeu auxquels sont associés des images. Vous devrez les manipuler pour les déplacer, les supprimer, les ajouter, etc. ;
   (b) la structure qui stockera tous les *sprites* du jeu est vide. Seule sa spécification vous est fournie. Implémentez-la. Elle devra permettre un parcours aisé ainsi que de l'ajout et suppression.
4. Écrivez un mécanisme de gestion des scores, c'est-à-dire qu'il faudra :
   (a) pouvoir compter les points lors d'une partie ;
   (b) pouvoir sauvegarder le score dans un fichier `scores.txt` une fois la partie terminée (le fichier sera une succession de lignes de scores au format `nickname:score`) ;
   (c) pouvoir charger un fichier de scores existant afin de l'afficher et de le compléter.
5. Personnalisez votre programme en modifiant certains paramètres et en ajoutant des fonctionnalités, notamment en vous inspirant des variantes proposées dans la section 5.
6. Testez votre programme régulièrement.

7. Si vous identifiez un *bug* dans le prototype, corrigez-le ;
8. À la fin du mini-projet, **vous livrerez votre travail sur Moodle**.

**Note :** la section suivante est là pour vous aider à trouver des extensions au logiciel. Un autre axe de travail peut éventuellement consister à chasser les `TODO` et les `FIXME`. Ce type de tâche est utile, mais pas toujours visible sur le résultat affiché.

# 5 Inspirations de fonctionnalités et de variantes de jeu possibles

Vous aurez une base de jeu similaire, cependant vous pourrez développer des fonctionnalités et variantes pour personnaliser et améliorer votre jeu. Ces dernières n'ont de limite que votre imagination, votre temps et votre capacité à traduire votre imagination en code correct et en images. Pour vous aider, voici une liste non exhaustive de variantes et fonctionnalités :

- on peut ajouter l'accélération ;
- les vaisseaux ont aussi un pouvoir (bouclier, bouclier anti-grav, vitesse accrue, téléportation, invincibilité temporaire, tir spécial, etc.) ;
- on peut ajouter de l'inertie ;
- on peut ajouter des extra-terrestres qui nous tirent dessus ;
- des astéroïdes peuvent se casser en plus que deux morceaux ;
- il peut y avoir une probabilité qu'un astéroïde libère un extra-terrestre lors de sa destruction ;
- des objets aléatoires se promènent dans l'espace ;
- les scores ont un traitement spécifique : tri, affichage particulier pour le podium, . . . ;
- des statistiques le jeu sont produites en fin de partie (précision, nombre d'ennemis détruits, temps de jeu, temps total de jeu, . . . )
- Triangle des Bermudes de l'espace : un trou noir peut être placé quelque part sur l'écran (ou au contraire un repoussoir) ;
- il peut y avoir des murs au bord de l'espace ou dans l'espace ;
- des vaisseaux peuvent avoir une barre de vie pour résister aux chocs ;
- des vaisseaux peuvent avoir une barre d'énergie pour tirer ;
- des éléments du jeu sont très rapides ou très lents ;
- des bonus peuvent apparaître temporairement à l'écran ;
- certains astéroïdes lâchent des ennemis à têtes chercheuses ;
- une ennemi est plus ou moins solide ;
- on peut jouer à plusieurs, en mode coopératif ou chacun pour soi :
  - les tirs amis n'ont pas d'effet sur vous ;
  - les tirs amis n'abîment pas votre vaisseau mais le poussent ;
  - les tirs amis. . . ne sont pas amis ;
- et si le son se propageait tout de même dans le vide spatial ?
- contre la montre : détruire toutes les comètes ou tous les ennemis avant la fin du temps imparti
- les projectiles peuvent parfois rebondir ;
- fusion de comètes : certains objets fusionnent lorsqu'ils se rencontrent ;
- conquête : extra-terrestres particulièrement agressifs (vaisseaux puissants, ennemis avec des vies, boucliers, barres de vie, tir multiple, EGV — Ennemi à Grand Vitesse —, ennemis à tête chercheuse, etc.) ;
- *spawn* en folie : apparition plus ou moins aléatoire et plus ou moins rapide d'astéroïdes et/ou d'ennemis, la vitesse d'apparition peut croître avec le temps, un ennemi particulier peut apparaître à chaque destruction de comète, etc.
- il peut y avoir un mode chaotique : physique en folie, flux dans l'espace donnant une direction, trous de ver, contrôles inversés ou déraillés, etc.
- . . .

« **May the force be with you.** »