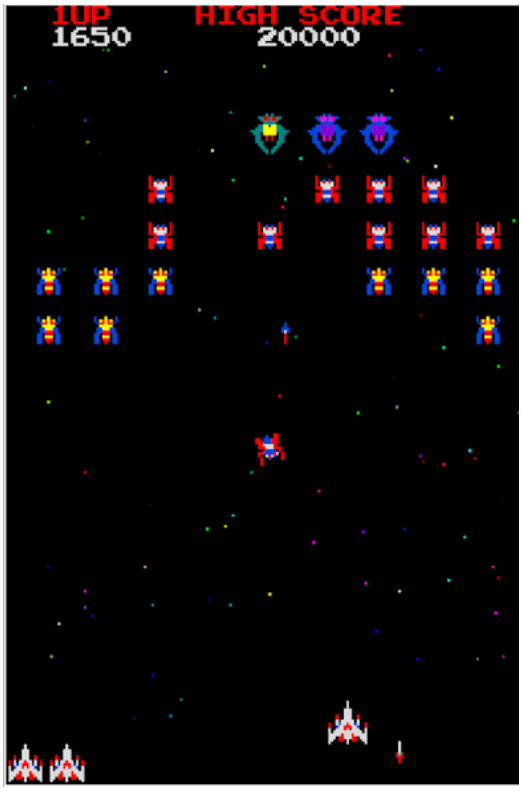
Projet Galaga 2025

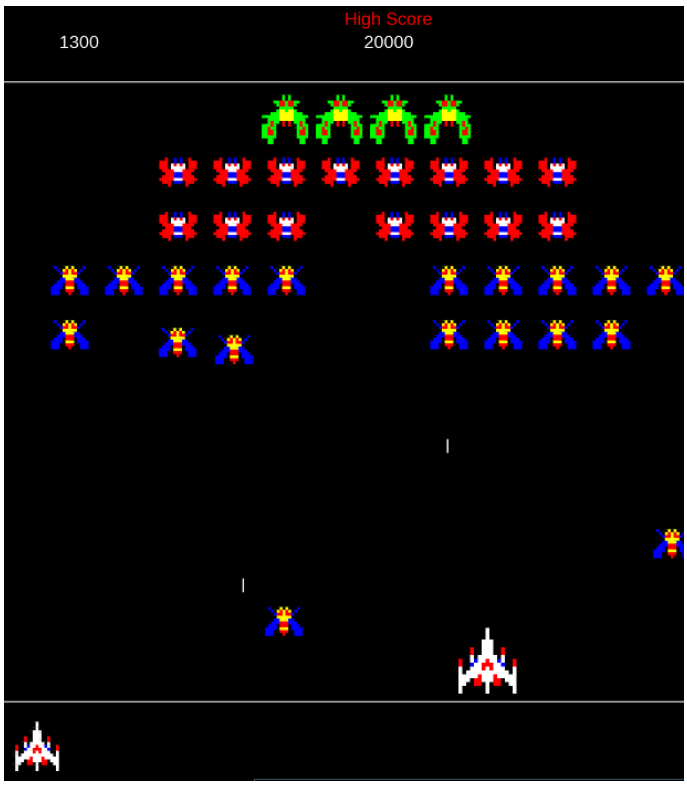
---

1. Présentation générale

Pour ce projet de Programmation Objet, vous allez réaliser un jeu d'arcade inspiré du jeu d'arcade Galaga.



Version original



Notre version

Le principe : le joueur contrôle un vaisseau spatial en bas de l'écran et doit détruire les vagues d'ennemis qui l'attaquent en retour.

Le jeu se déroule **en temps réel**, avec des ennemis qui se déplacent, attaquent, et tirent des projectiles que le joueur doit éviter.

Le joueur gagne s'il élimine toutes les vagues d'ennemis. Il perd si son vaisseau est détruit.

Une vidéo du jeu original en action : https://www.youtube.com/watch?v=dvjapcHsqXY

---

2. Déroulement du projet

Vous trouverez sur Moodle un squelette de projet contenant un code très basique qui gère un jeu en temps réel avec un rond rouge qui bouge lors d'interaction avec les touches du clavier.
Le but du projet est, partant du squelette que l'on vous fournit, de réaliser un jeu de Galaga respectant le cahier des charges qui suit.
Ce projet sera à réaliser en binôme à renseigner sur Moodle.
Chaque fonctionnalité sera découpée en trois niveaux de difficulté :

Niveau 1 fonctionnalités basiques

Niveau 2 fonctionnalités plus complexes qui rendent le jeu plus actif

Niveau 3 fonctionnalités bonus

Nous vous conseillons fortement de réaliser le projet dans l'ordre des niveaux.

---

3. Cahier des charges

Vue générale du projet

Le projet doit comporter :

- La **gestion du joueur** (déplacement, tir, points de vie).
- La **gestion des ennemis** (apparition, déplacement, tir).
- La **gestion des projectiles** (déplacement, collision).
- La **progression des niveaux** (mise en place des formations ennemies, passage au niveau suivant).
- La **gestion du score** (augmentation du score, nouveau meilleur score).
- Une **interface graphique** simple, réalisée avec la librairie `StdDraw`.

Niveau 1 Faire un diagramme UML du projet avant même de commencer à développer le projet (à faire valider par votre enseignant.e).

Niveau 2 Mettre à jour le diagramme UML une fois le projet terminé.

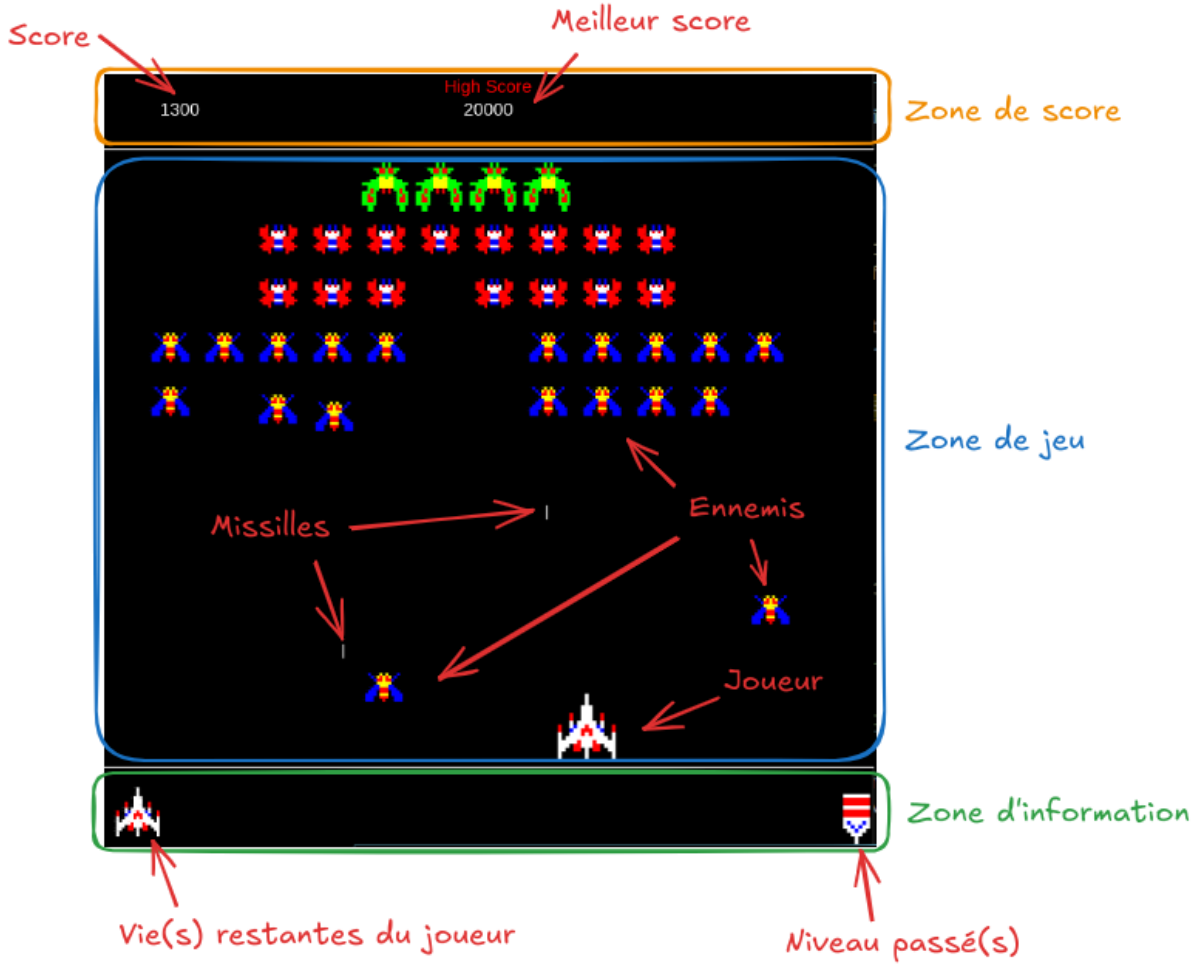
3.1. Interaction Homme Machine

Interface Graphique

L'écran est découpé en **trois zones principales** :

- **Zone de jeu (bleu)** : le champ de bataille où évoluent le vaisseau du joueur, les ennemis et leurs tirs respectifs.
- **Zone de score (orange)** : le score et le meilleur score.

- **Zone d'information (verte)** : vie du joueur, indication sur le nombre de niveaux passés.



Niveau 1 Les entitées sont représentées par des formes géométriques simples (`circle`, `square`, `triangle`) et des couleurs distinctes.

Niveau 2 Les sprites des différents acteurs du jeu sont lus depuis les fichiers `ressources/sprites/*.spr` (cf détails en partie 4).

Interaction clavier

Le joueur pourra :

- se déplacer vers la gauche avec la flèche gauche (`keycode: 37`).
- se déplacer vers la droite avec la flèche droite (`keycode: 39`).
- tirer un missile avec la barre espace (`keycode: 32`).

À la fin d'une partie le joueur pourra relancer une partie avec la barre espace (`keycode: 32`).

Niveau 1 Déplacement du joueur et tir de missiles.

Niveau 2 Relancer une partie.

3.2. Les entités du jeu

Joueur

Le joueur contrôle un vaisseau :

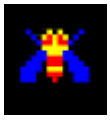
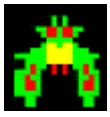
- Il commence la partie avec 3 vies.
    - Il perd une vie lorsqu'il rentre en colision avec un ennemi ou se fait toucher par un missile ennemi.
- Il peut se déplacer horizontalement mais pas verticalement.
- Il peut tirer des missiles vers le haut.
    - Seuls trois missiles peuvent être présent à l'écran à la fois.
    - les missiles ne sont pas guidés : ils se déplacement en ligne droite verticalement depuis leur point de tir.
- Quand le joueur détruit un ennemi avec un missile, la valeur de cet ennemi est ajouté au score.

Ennemis

Un ennemi :

- N'a qu'une seule vie.
- Peut tirer des missiles vers le bas (dans les niveaux avancés).
    - là non plus les missiles ne sont pas guidés.
- Peut se déplacer vers le joueur pour lui faire perdre une vie.
    - Lors de la collision avec le joueur l'ennemi est détruit et le vaisseau du joueur aussi.
- Il n'y a pas de dégâts entre ennemis (pas de gestion de collisions entre les ennemis).

Il y a trois types d'ennemis qui ont tous un comportement différent :

Nom	Sprite	Trajectoire d'attaque
Bee		Zig-zag
Butterfly		Ligne
Moth		Ligne puis capture d'un vaisseau

Formation Ennemie

Les ennemis se déplacent en formation :

- Ils se déplacent tous ensemble vers la gauche puis vers la droite mais ne descendent pas en formation.

- Seul un ennemi en bas d'une formation peut tenter un déplacement (c'est-à-dire une attaque vers le joueur).
- Seul un ennemi en bas d'une formation ou en mouvement peut tirer des missiles.

Niveau 1

- Le joueur se déplace, tire et peut être détruit au contact d'un missile ou d'un ennemi.
- Les ennemis se déplacent de gauche à droite en formation.
- Un ennemi en bas d'une formation peut tirer des missiles.

Niveau 2

- Les `Bee` et les `Butterfly` au bas d'une formation peuvent attaquer le joueur suivant leur trajectoire respective.

Niveau 3

- Les `Moth` peuvent descendre au dessus d'un joueur et capturer son vaisseau.
  - Si le joueur détruit le Moth qui a capturé son vaisseau il récupère sa vie.

3.3. Déroulement d'une partie

Au lancement

Le jeu charge le premier niveau :

- Il place les ennemis à leur emplacement respectif.
- Il place le joueur en bas au centre de l'espace de jeu.

Lors de la partie :

- Le déplacement du joueur se fait avec les flèches gauche/droite.
- Le tir de missile se fait avec la barre espace.
- Un ennemi descend en suivant son motif à un intervalle de temps défini pour chaque niveau.
- Si le joueur est touché :
  - Tous les missiles disparaîssent de l'écran.
  - Les ennemis encore en vie retournent tous à leur emplacement initial.
  - Après 3 secondes de cool-down le joueur réapparaît à son emplacement initial et la partie reprend.

Fin de partie

- Le jeu se termine quand :
  - Le joueur a perdu ses 3 vies.
  - Ou que tous les niveaux sont finis.
- Quand la partie est finie, on propose de rejouer avec un appui sur la touche espace.

Déroulement des niveaux

- Au lancement du jeu le niveau 1 décrit dans le fichier `ressources/levels/level1.lvl` se met en place.
    1. Son nom est affiché à l'écran pendant 2 secondes.
    2. Les ennemis sont mis en place.
    3. Le jeu se lance.
- Quand tous les ennemis d'une formation sont éliminés, on charge le niveau suivant.
    1. On replace le joueur au centre.
    2. On efface tous les missiles restant.
    3. On affiche le nom du nouveau niveau.
    4. On relance le jeu.

Gestion du score

- Quand la partie est finie, on affiche le score en grand sur l'écran et on propose de rejouer avec la touche espace.

Niveau 1

- Une formation ennemie quelconque est à l'écran, une fois cette formation éliminée le jeu est fini.
- Lorsque le joueur se fait toucher, la partie est finie.

Niveau 2

- Une formation ennemie est chargée depuis un fichier `ressources/level/*.lvl`
- Lorsque le joueur se fait toucher, il perd une vie et continue le niveau.
- Lorsque le joueur n'a plus de vies, la partie est finie.
- Lorsqu'un niveau est fini, un nouveau niveau est chargé.
- Gestion du score.

Niveau 3

- Afficher le nom des niveaux à l'écran pendant 2 secondes avant de lancer le niveau.
- Créer un niveau boss avec un nouvel ennemi de votre choix.

---

4. Fonctionnement du projet

4.1. Structure du projet

Vous devez récupérer le squelette de projet sur Moodle.
Ce squelette contient deux dossiers :

- `src` contient :
    - une classe `App.java` avec la fonction `main` du projet,
    - une classe `Game.java` qui sera appelée par le `main` qui contient la gestion de la boucle de temps réel du jeu.
- `ressources` contient toutes les ressources dont vous aurez besoin pour la gestion du score, des niveaux et des sprites.
    - `level/*.lvl` Décrit la formation ennemie d'un niveau avec le nombre d'ennemis, leur emplacement, leur vitesse et leur valeur en points.
- `# Nom du niveau, vitesse de la formation, cooldown entre deux attaques (-1 quand il n'y a pas d'attaques), cooldown entre deux tirs de missiles`
- `Level1 0.001 -1 4000`
- `# Type d'ennemi, position sur l'axe x, position sur l'axe y, taille, valeur, vitesse`
- `Moth 0.38 0.85 0.06 300 0.0005`
- `...`
- `Butterfly 0.24 0.78 0.06 200 0.001`
- `...`
- `Bee 0.1 0.64 0.06 100 0.002`
- `...`
    - `sprites/*.spr` Décrit un sprite pour chaque acteur du jeu, chaque lettre désigne une couleur et correspond à un pixel du sprite
- `...`
- `NNNBNNNYNYNNNBNN`
- `NNNNBNYRYRYNBNNN`
- `NNNNNBRRYRRBNNNN`
- `...`

  Le code des couleurs est :

Lettre	Couleur	Dans la librairie StdDraw
R	Rouge	`StdDraw.RED`
B	Bleu	`StdDraw.BLUE`
G	Vert	`StdDraw.GREEN`
Y	Jaune	`StdDraw.YELLOW`
W	Blanc	`StdDraw.WHITE`
N	Noir	`StdDraw.BLACK`

    - `highscore/highscore.sc` Sauvegarde le meilleur score jamais enregistré, par défaut c'est 20 000

  `20000`

4.2. Boucle de jeu

La boucle de jeu vous permettra de gérer les éléments de votre jeu.
Afin de bien démarrer le projet, nous vous fournissons un squelette pour l'application principale (`App`), qui vous permettra de lancer le jeu (`Game`).
Le code de ces classes **devra changer** quand vous réaliserez les différents niveaux du cahier des charges.

4.3. Interface graphique avec `StdDraw`

La library StdDraw vous permettra de gérer un affichage et l'interface graphique associée. Elle est présente dans le package `engine` de votre projet.
Vous y trouverez notamment :

- Gérer le canvas (zone de dessin) :
  - `setCanvasSize(int width, int height)`
  - `clear()`
- Afficher du texte :
  - `text(double x, double y, String text)`
- Afficher différentes formes géométriques :
  - `circle(double x, double y, double radius)`
  - `filledCircle(double x, double y, double radius)`
  - `square(double x, double y, double radius)`
  - `filledSquare(double x, double y, double radius)`
  - `rectangle(double x, double y, double halfWidth, double halfHeight)`

4.4. Utilisation des IAs génératives

Vous êtes autorisés à utiliser des IA génératives comme ChatGPT, dans certaines limites détaillées ci-dessous. Les utilisations que nous autorisons sont :

- de l'aide ponctuelle pour comprendre comment réaliser certaines parties du projet (l'IA vous explique comment il faut faire en français, et vous écrivez le code vous même) ;
- de l'aide au débogage : une de vos fonctions a une erreur que vous n'arrivez pas à trouver (que ce soit à la compilation ou à l'exécution), vous copiez son code dans l'IA générative pour lui demander de vous aider à trouver l'erreur.

Nos conditions à l'utilisation d'IA génératives sont les suivantes :

- Le projet doit principalement être réalisé par vous. Si la majorité du projet est écrite par une IA générative (c'est à dire > 50%), vous aurez la note de 0 ;
- Soyez transparents : indiquez en commentaire les endroits du code où l'IA générative vous a aidé, avec une phrase brève pour expliquer la nature de cette aide (idée, quelques lignes de code, débogage…). Ne détaillez pas trop dans ces commentaires, mais ajoutez dans votre projet un fichier texte `IAGenerative.txt` où vous expliquez plus précisément les différents endroits où l'IA vous a aidé et comment. Vous pouvez copier dans ce fichier certaines de vos conversations avec l'IA pour nous aider à comprendre comment vous vous en êtes servi. Indiquez aussi le modèle d'IA que vous avez utilisé.

Attention, les IA génératives peuvent mal comprendre vos questions et donner des réponses inadaptées, et peuvent tout simplement se tromper. Ayez donc un regard critique sur leurs réponses : si vous ne faites pas attention, ces outils peuvent vous faire perdre beaucoup de temps et annuler le temps que vous aurez gagné par ailleurs !
Enfin, si vous ne souhaitez pas ou ne pouvez pas utiliser d'IA générative en ligne, vous pouvez installer des versions locales sur votre machine. L'interface LM Studio est une manière conviviale d'installer une IA générative locale sur votre machine, et est disponible

sur toutes les plateformes. Il vous faudra ensuite télécharger un modèle, nous vous recommandons des modèles comme Mistral, Llama, Qwen, Gemma ou olmo. Les modèles étiquetés 7B sont un bon compromis et tournent sur la plupart des machines, les modèles 3B voire 1B sont adaptés aux configurations modestes mais peuvent faire plus d'erreurs.
Faites vos recherches sur Internet pour l'installation et l'utilisation de ces approches, de très nombreux tutoriels sont disponibles (ces recherches d'une IA générative ne font pas partie du temps du projet !)

---

5. Livrables

Référez vous à moodle pour la date de rendu. Attention de bien enregistrer votre binôme dès que possible, il ne sera pas possible de rendre votre projet si vous n'êtes pas dans un binôme enregistré.
Le projet sera à déposer sur moodle.
Il faudra rendre une archive compressée (zip), contenant :

- Un **diagramme de classe** de votre projet à jour sous forme d'image (format `.png` par exemple).
- Un **zip de projet Visual Studio Code** contenant l'essentiel pour exécuter votre projet :
    - Vos sources (dossier `src`)
    - Vos ressources (dossier `ressources`).
    - Attention :
        - Le projet rendu doit obligatoirement être un projet Visual Studio Code, nous n'acceptons pas de projets d'autres éditeurs (note de 0 au projet si cette consigne n'est pas respectée).
        - Le projet ne doit s'appuyer sur aucune librairie externe autre que `StdDraw`.
        - Vérifiez que le projet que vous rendez compile et s'exécute avant de le rendre !
- Un fichier **`README.txt`** de présentation de votre projet :
    - Noms des membres du binôme.
    - Explications des fonctionnalités réalisées.
    - Éventuelles fonctionnalités supplémentaires.
    - Guide pour exécuter votre projet.
    - Description de l'interface pour jouer avec votre projet.
- La **Javadoc générée**.

    Pour générer la documentation dans un dossier "docs" à côté de votre dossier `src` :

    - Ouvrez un terminal et placez-vous dans votre dossier `src`
        - Utilisez la commande : `javadoc -d ../docs ./*`
        - `javadoc` lira automatiquement tous vos fichiers `.java` dans tous les sous-dossiers. La commande vous affichera également de nombreuses erreurs/warning qu'il faudra lire et corriger pour

générer correctement
la documentation.
    - La commande `javadoc` est inclue dans le Java Development Kit (jdk)
- Le fichier **`IAGenerative.txt`** si vous avez utilisé une IA pour le projet.

---

6. Notation

À titre indicatif.

Critère	Notation
Modèle UML	3
Rapport (README.txt + IAGenerative.txt)	2
Javadoc	2
Niveau 1	6
Niveau 2	5
Niveau 3	2
Total	20