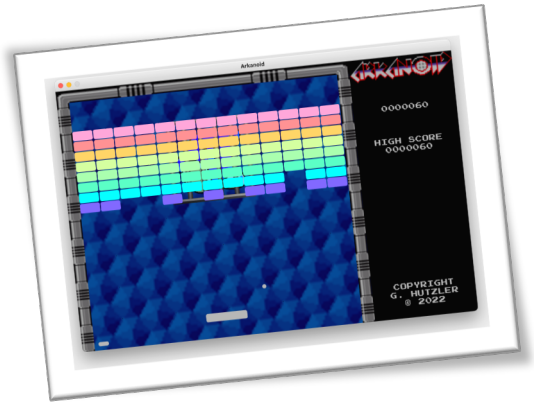
DM
Arkanoid



Objectif

L'objectif consiste à écrire un jeu ressemblant à Arkanoid, jeu d'arcade de type casse-briques développé par Taito en 1986, reprenant même principe que Breakout d'Atari Inc. (1976) (https://fr.wikipedia.org/wiki/Arkanoid) :

> *« Le joueur déplace de droite à gauche une barre horizontale censée représenter un vaisseau spatial. Ce plateau, positionné en bas de l'écran, permet de faire rebondir une balle qui va détruire des blocs situés en haut de l'écran. Le joueur passe au niveau suivant lorsque tous les blocs sont détruits (…) ; le joueur perd une vie à chaque fois qu'il laisse filer la balle au bas de l'écran. ».*

Les dix commandements de maître Yoda

⚠

1. **Ce document, avant de commencer, tu liras jusqu'au bout, et `DM_Arkanoid.pdf` pareillement**
2. Les instructions qui te guident pas à pas, tu suivras
3. Les différentes étapes de ton travail, tu conserveras
4. Ton programme, tu indenteras et commenteras
5. La signature des fonctions, tu respecteras
6. De la programmation orientée objet, tu ne feras pas
7. L'utilisation des variables globales, tu limiteras, et les variables locales préfèreras
8. Les tableaux, quand c'est possible, tu utiliseras
9. Seul, tu travailleras
10. Ce que tu fais, tu comprendras, et capable de l'expliquer, tu seras

Prise en main

- Téléchargez l'archive arkanoid.zip et décompressez-là dans un dossier sur votre ordinateur.
- Renommez `Arkanoid_v0.pde` en respectant la convention de nommage suivante : `GXX_NOM_Prenom.pde`, où `XX` doit être remplacé par :
  - `MIn` avec `n` votre n° de groupe si vous êtes dans le portail Maths-Info
  - `MPn` avec `n` votre n° de groupe si vous êtes dans le portail Maths-Physique
  - ME si vous êtes en double licence Maths-EcoB
  - BI si vous êtes en licence double diplôme Info-SdV
  - DI si vous êtes en licence double diplôme Droit-Info
- Chargez le programme dans Processing et complétez les informations vous concernant dans le bloc de commentaire en haut du programme
- Observez les différentes fonctions que vous allez devoir programmer
- Prenez connaissance du barème de notation :

$$Note_{finale} = \text{à définir…}$$

I TD1 = Qui c'est le boss ?

I.1. Dessiner le boss

Dans l'histoire originelle, une fois que vous avez passé les 32 niveaux du jeu, vous devez finalement détruire le grand méchant « Doh ». Pour notre jeu, une fois que toutes les briques auront été détruites, vous pourrez libérer le héros de votre jeu en détruisant la cage dans laquelle il est enfermé.
Il faut donc commencer par dessiner votre héros, de taille 400x400 pixels, avant de l'enfermer en affichant par-dessus l'image de grille (« cage.png ») fournie dans le dossier data.

Pour dessiner votre héros, vous devez :

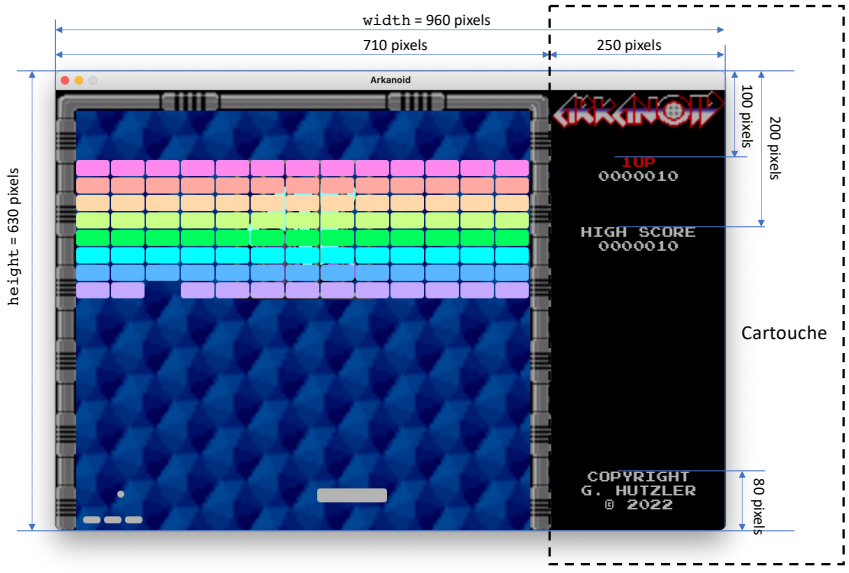
1. compléter la fonction `boss` en :
   a. utilisant au moins 20 instructions de tracé de base (point, line, rect, ellipse)
   b. utilisant au moins 10 instructions de couleurs (fill, stroke, noFill, noStroke)
   c. utilisant au moins 3 fonctions plus élaborées différentes (text, arc, quad, triangle, bezier, beginShape, etc.)
   d. commentant soigneusement votre programme
2. ajouter un appel à la fonction `boss` dans la fonction `afficheJeu`
3. ajouter un appel à la fonction `afficheJeu` dans la fonction `draw`

Pour afficher une image :

```
PImage img = loadImage( « cage.png » ) ;
image(img, 0, 0) ;
```

II TD2 = On complète l'interface

Dans la partie droite de l'image (voir schéma ci-dessous), apparaitra un « cartouche » avec le score du joueur, le meilleur score réalisé depuis le début, ainsi qu'une notice de copyright.



II.1. Afficher une ébauche du cartouche

1. Complétez la fonction `cartouche` en :
   a. dessinant un rectangle noir
   b. insérant l'image « arkanoid.png » en haut du cartouche
   c. affichant « `1UP` » et « `HIGH SCORE` »
   d. affichant les éléments de la notice de copyright (séparés verticalement de 20 pixels), en remplaçant le nom (`G. HUTZLER`) par le vôtre
2. Ajoutez un appel à la fonction `cartouche` dans la fonction `afficheJeu`

Pour afficher le texte « hello! » au centre de la fenêtre avec la police joystix.ttf :

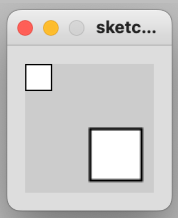
```
PFont fonte = createFont("joystix.ttf", 20);
textFont(fonte);
text("Hello!", width / 2, height / 2);
```

II.2. Afficher le cadre et repositionner le boss

1. choisissez un arrière-plan de couleur (0, 60, 130) pour l'espace de jeu
2. dessinez un cadre gris de couleur (128, 128, 128) tout autour de l'espace de jeu (voir schéma), de largeur 30 pixels
3. utilisez les fonctions `translate` et `scale` pour afficher votre boss de taille 200x200 avec le coin supérieur gauche en (255, 100). Notez que les fonctions `pushMatrix` et `popMatrix` permettent de changer temporairement l'origine du repère (avec `translate`) et l'échelle (avec `scale`), puis de revenir aux conditions de tracé initiales (voir ci-dessous).

Pour changer de repère :

```
pushMatrix() ;
translate(50, 50);
scale(2);
// affiche le grand carré gras en bas à droite
rect(0, 0, 20, 20);
popMatrix();
// affiche le petit carré fin en haut à gauche
rect(0, 0, 20, 20) ;
```



III TD3 = Le chat et la souris

III.1. Des constantes pour délimiter le terrain de jeu

1. déclarez des constantes pour définir les coordonnées des bords du terrain (`minX`, `maxX`, `minY`) et le centre de la zone de jeu (`centreX`)

III.2. Des variables pour la raquette

La raquette doit s'afficher sous forme de rectangle aux coins arrondis, de taille 100x20 pixels, centré horizontalement sur la souris et dont le dessus est à 50 pixels du bas de la fenêtre.

1. ajoutez des constantes `raquetteY`, `raquetteL`, `raquetteH` de type entier pour définir respectivement la position du milieu de la raquette (verticalement), la largeur et la hauteur de la raquette ;
2. ajoutez une variable `raquetteX` de type entier pour enregistrer le centre de la raquette (horizontalement) ;
3. complétez la fonction `initBalleEtRaquette` pour initialiser la raquette au centre du terrain de jeu ;
4. complétez la fonction **`mouseMoved`** pour ajuster le centre de la raquette en fonction de la position de la souris ;
5. complétez la fonction `afficheRaquette` pour afficher la raquette centrée sur le point (`raquetteX`, `raquetteY`), de couleur blanche ;
6. Ajoutez un appel à la fonction `afficheRaquette` dans la fonction `afficheJeu`.

III.3. Des variables pour la balle

La balle s'affiche sous forme d'un disque blanc de diamètre 10.

1. ajoutez une constante `balleD` pour définir le diamètre de la balle ;
2. ajoutez des variables `balleX` et `ballY` de type réel pour enregistrer les coordonnées du centre de la balle ;
3. complétez la fonction `initBalleEtRaquette` pour initialiser la balle posée sur la raquette ;
4. complétez la fonction `afficheBalle` pour afficher la balle centrée sur le point (`balleX`, `balleY`), de couleur blanche ;
5. Ajoutez un appel à la fonction `afficheBalle` dans la fonction `afficheJeu`

IV TD4 = On lance la balle

On souhaite maintenant pouvoir faire bouger la balle. On définira pour cela un vecteur vitesse (`balleVx`, `balleVy`).

IV.1. Encore des variables pour la balle

1. Déclarez une constante de type entier `balleVitesse` ;
2. déclarez deux variables de type réel `balleVx` et `balleVy` ;
3. complétez la fonction `initBalleEtRaquette` pour choisir aléatoirement un angle compris entre $5\pi/4$ et $7\pi/4$ et initialisez le vecteur vitesse de la balle de manière à ce qu'il soit orienté dans la direction de l'angle choisi et avec une norme `balleVitesse`. Pour rappel :
$$vx = norme.\cos(angle)$$
$$vy = norme.\sin(angle)$$
4. déclarez deux variables de type réel `newBalleX` et `newBalleY` (elles seront utiles par la suite pour calculer les rebonds de la balle sur les briques et sur le boss) ;
5. complétez la fonction `deplaceBalle` qui met à jour `newBalleX` et `newBalleY`, la nouvelle position de la balle (cf exo IV.2 du TD 4) ;
6. complétez la fonction `miseAJourBalle` qui recopie les coordonnées (`newBalleX`, `newBalleY`) dans (`balleX`, `balleY`)
7. appelez les fonctions `déplaceBalle` et `miseAJourBalle` au début de la fonction `draw`.