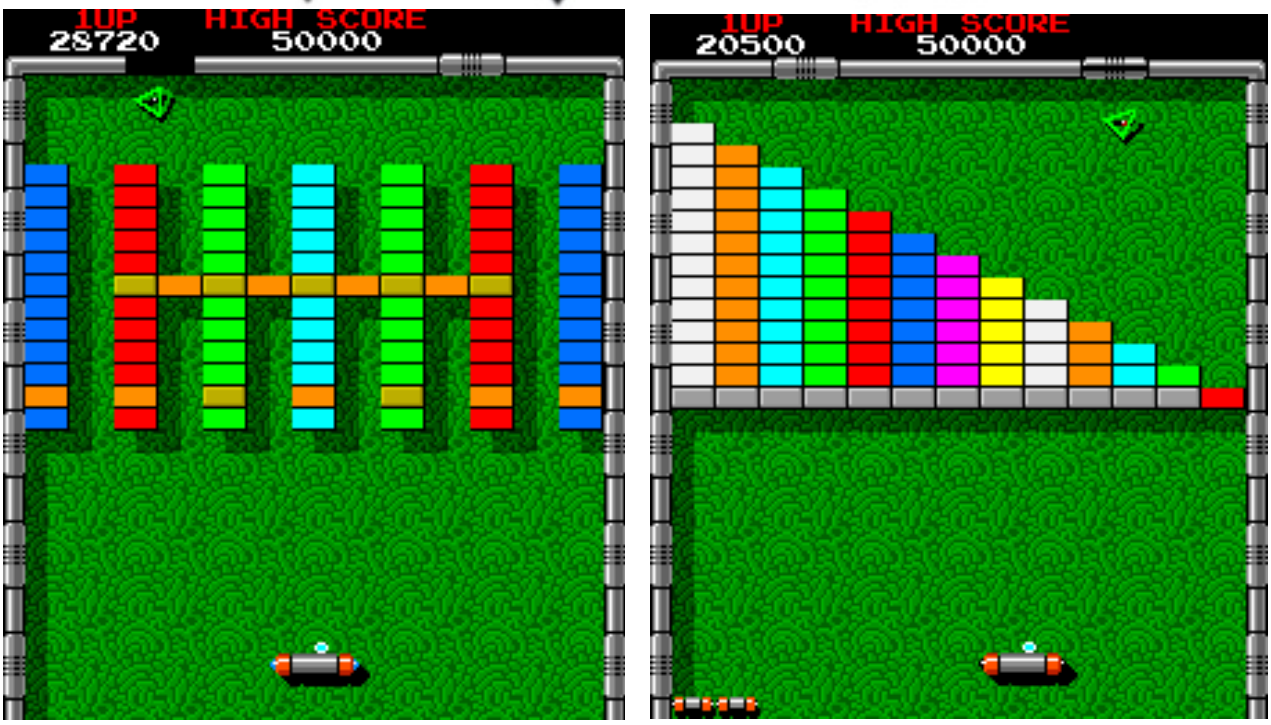
**Projet de Programmation Avancée (50h)**

**A faire en binôme** **A rendre le 17/12/2018 à 12:00**




Faire un remake de ‘Arkanoid’ en C++ (11/14) avec SDL2 .

Le but est de proposer une version qui respectera les règles de la version originale (cf play instruction)
L’implémentation devra être modulaire et permettre de les modifier et les étendre facilement:

- taille du plateau de jeu variable (original 13x25 briques) (pendant le jeu ?)
- plateau “cylindrique” (pas de bord gauche/droite)
- ajout de nouveau type de bonus/malus
- mode 2 joueurs
  - collaboratif / course / combat

Vous utiliserez le maximum de paradigmes de programmation vus en cours.

**L’évaluation prendra en compte le design, l’élégance du code, la modularité, l’optimisation, …**

**Le code sera sur le gitlab unistra et vous rendrez un court (4p max) rapport individuel sur votre partie détaillant un choix de design et son implémentation.**

**Une semaine avant la dead-line, vous aurez le détail d’un mode deux joueurs à implémenter. Si votre code es modulaire et bien designé cela devrait être facile.**

# ARKANOID™

## PLAY INSTRUCTIONS

Shatter the wall sections with your energy ball by moving your VAUS craft left & right.

There are 3 types of walls:

(1) NORMAL WALL SECTIONS:
You can break a normal wall section by hitting it with the energy ball once. 50 to 120 points are awarded depending on the color.

(2) HARD WALL SECTIONS:
You need to hit these with the energy ball several times in order to break them.
The number of hits required are:

| | |
|---|---|
| 2 times — | 1st to 8th rounds |
| 3 times — | 9th to 16th rounds |
| 4 times — | 17th to 24th rounds |
| 5 times — | 25th to 32nd rounds |

Bonus points awarded for breaking the barrier wall section —— 100 points times the number of the round.

(3) INDESTRUCTIBLE WALL SECTIONS:
You cannot break these wall sections.

Some wall sections contain power-up capsules. Catch the capsules to:

| | |
|---|---|
| (S) SLOW DOWN | Slows down the energy ball. |
| (C) CATCH & FIRE | Catch the energy ball and shoot it back. |
| (E) EXPAND | Expands the length of the VAUS craft. |
| (D) DIVIDE | Splits the energy ball into three particles. |
| (L) LASER BEAM | Enables the VAUS to fire laser beams. |
| (B) BREAK | Allows the player to warp into the next play-field. |
| (P) PLAYER ADDITION | An additional VAUS awarded. |

Power-up capsules are effective until the player is shot down, the round cleared, or until another capsule is picked up.

1000 points awarded for each capsule picked up.

HARMFULS appear from the top of the screen and creep through the broken walls. Hit them with the energy ball (100 points).

The round is cleared when all wall sections are broken.

There are 33 rounds in this game. In the final round, a huge enemy fortress appears. While avoiding bullets, hit him many times with your energy ball.

Game ends when all VAUS are lost or you clear all 33 rounds.

Additional VAUS awarded for higher scores.


**TAITO CORPORATION**
C.P.O.Box 1164, TOKYO 100-91, JAPAN
CABLE : EPTRA TOKYO
TELEX : J22931 EPTRA
FACSIMILE : (03)234-2690
TELEPHONE : (03)264-8614

**TAITO CORPORATION**
660 S.Wheeling Road
Wheeling, IL 60090
TEL: (312)520-9280
FAX: (312)520-1309
TLX: 25-3290 TAMCO

**Détails d'implémentations:**

Pour l'affichage et le graphisme on utilisera le code d'exemple et les resources disponibles sur http://igg.unistra.fr/people/thery/Prog_Avancee/

Voud utiliserez uniquement la copie de sprite à l'aide de *SDL_BlitSurface.*

Remarque: la SDL a une interface "pur C" vous en ferez une version OO qui s'intégrera mieux au reste de votre code.

Les niveaux seront stockés dans des fichiers textes (grille de codes de brique)

Les donnéee de la balle (position, direction, vitesse) seront calculé et stockés en réel pour plus de précision, même si les positions à l'écran sont entières (pixel)

Les objets à l'écran (hormis la balle) étant tous rectangulaires, l'algorithme de détection de collision se fait simplement en comparant x_min, x_max, y_min, y_max avec la position de la balle. Les briques étant alignées régulièrement,

Si on touche quelquechose de vertical, la coordonnée x de la vitesse s'inverse.
Si on touche quelquechose d'horizontal, la coordonnée y de la vitesse s'inverse.

Vous prendrez en compte la vitesse (horizontale) du vaisseau pour changer la direction du rebond.

Options orientées graphique: ombres, traces derrière la balle, disparition progressive des briques.

Options orientées réseau: multi-joueurs sur de machines différentes, tournoi.

Option sorientées IA : distribution bonus/malus non aléatoire, faire jouer l'ordinateur,

Remarque:
Le code devra compiler et s'exécuter sur les machines du Département Informatique, même si vous le présentez sur votre machine personnelle.