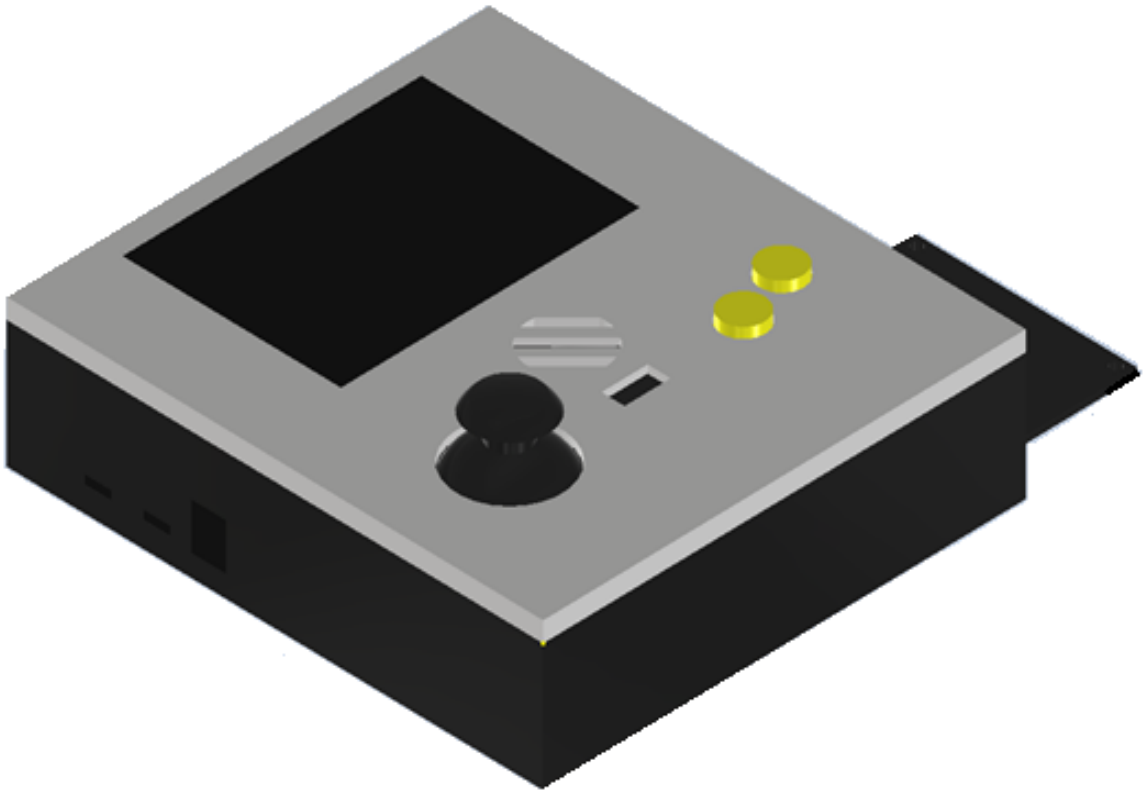
Université Nice Sophia Antipolis

Département d'Electronique

POLYTECH NICE SOPHIA

Gameduino



Projet Arduino – PEIP2

Année Scolaire 2019-2020

Etudiants :
Benharrats Amine
Manuel Enzo

Enseignants :
Masson Pascal
Abdelrahmen Nassim

Ecole Polytechnique Universitaire de Nice Sophia-Antipolis, PEIP2
1645 Route des Lucioles, Parc de Sophia Antipolis, 06410 Biot

SOMMAIRE

Table des matières

I. Présentation du projet

Etant tout deux passionnés de jeux vidéo et de codage informatique, nous avons décidé de réunir nos deux passions dans notre projet : la Gameduino. Cette dernière est une console portable de jeux-vidéos. Cette petite console vous permet de jouer à des jeux d'arcades tel que le Pong ou le Snake. Nous verrons, dans une première partie, la réalisation du boitier et le montage. Puis dans une seconde partie nous aborderons le code que nous avons réalisé pour l'électronique et celui concernant nos jeux.

II. Cahier des charges

Objectif : Créer une console de jeux vidéo compacte et portable avec deux jeux Pong et Snake

Découpage du projet :

Partie boitier :
- Design
- Modélisation
- Impression
- Montage

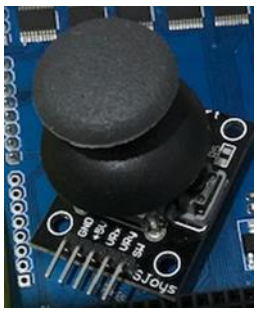
Partie code :
- Librairie utilisés
- Librairies codés
- Menu
- Pong
- Snake

Liste du matériel utilisé :

Les principaux éléments utilisés sont (figure 1)

Carte Arduino Due

Adaptateur carte/écran 3,5’’ tactile



Joystick

2 Boutons

Figure 1 : matériel utilisé

Autres :

- 1 Buzzer
- Acide polylactique (plastique pour réaliser notre boîtier à l’aide de l’imprimante 3D)

III. Partie Boîtier

1. Design

Pour la réalisation de notre projet nous avons dû dans un premier temps nous mettre d'accord à deux sur l'aspect final et l'esthétique que notre console aura (figure 2) : elle devait être ergonomique, adaptée pour des jeux vidéo, elle devait être mobile et devait contenir tous les composants électroniques.

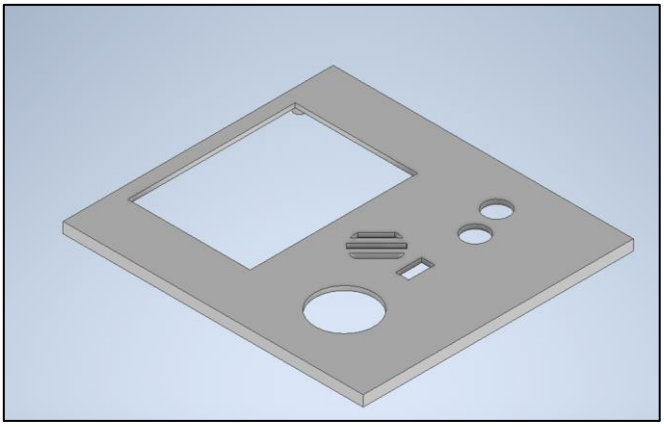


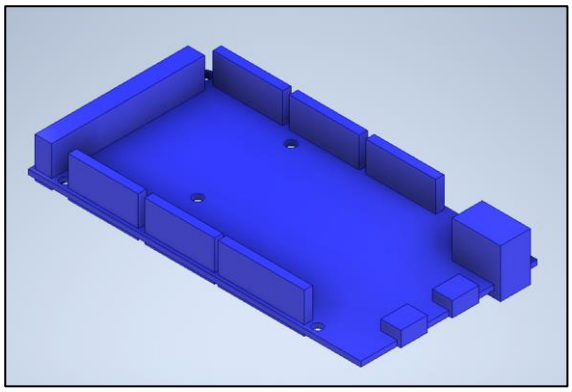
Figure 2 : Couvercle du boîtier supérieur

2. Modélisation

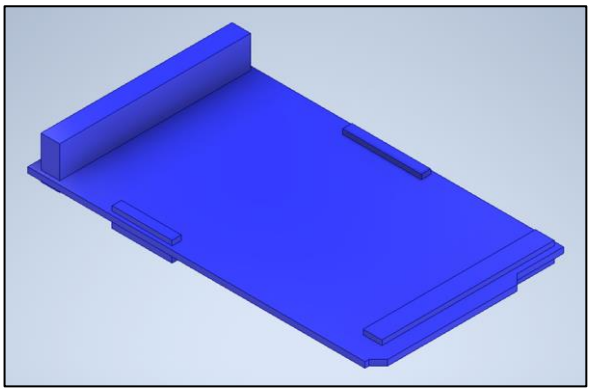
La contrainte de la taille de la console nous a posé un problème : Le boitier devait être le plus petit possible pour être tenu dans les mains et que l'on puisse y jouer et il devait également contenir tous les éléments qui composent notre console.

Pour éviter de se tromper dans la taille du boitier final de la console, nous avons modélisé la forme extérieure de chaque composant (figure 3), en tenant compte de sa taille exacte avec Autodesk Inventor. Une fois les pièces modélisées nous avons, toujours grâce à ce logiciel, simulé le montage final des pièces à l'intérieur du boitier.

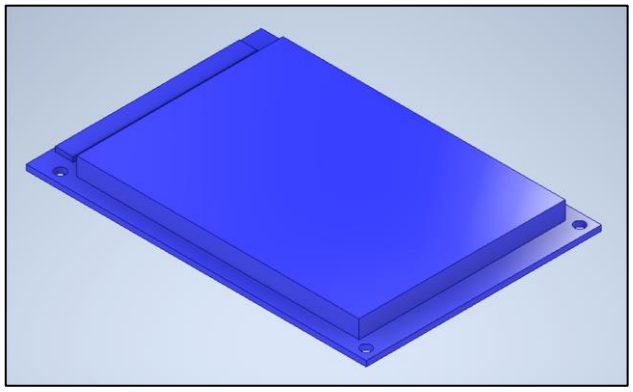
A partir de ce montage final, nous avions les dimensions exactes de notre boitier final ainsi que la positions exactes des ouvertures pour les différentes pièces (joystick, boutons..etc) Cette étape nous a permis d'avoir



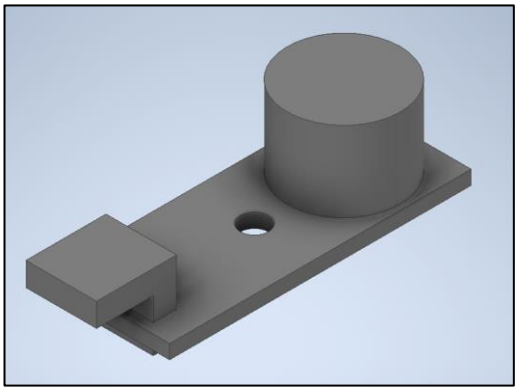
Carte Due



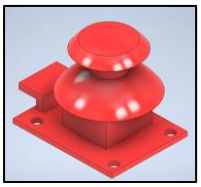
Adaptateur Carte/Ecran



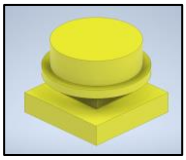
Ecran



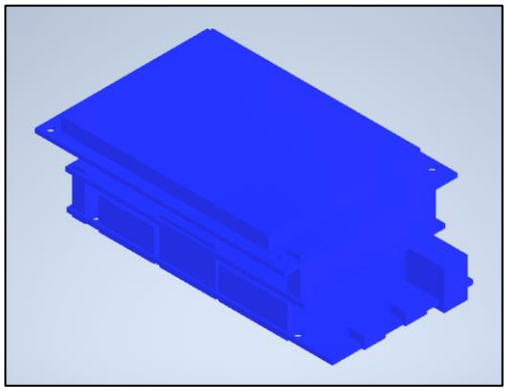
Buzer



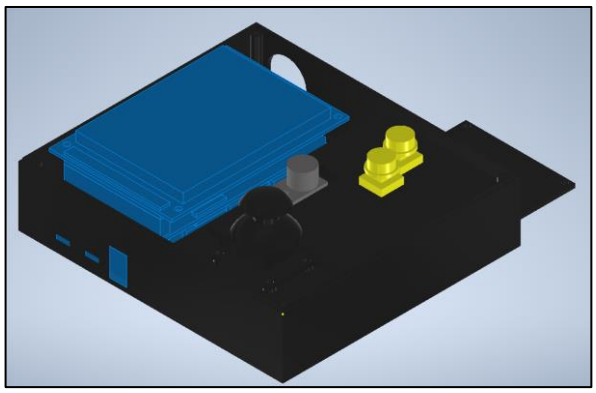
Joystick



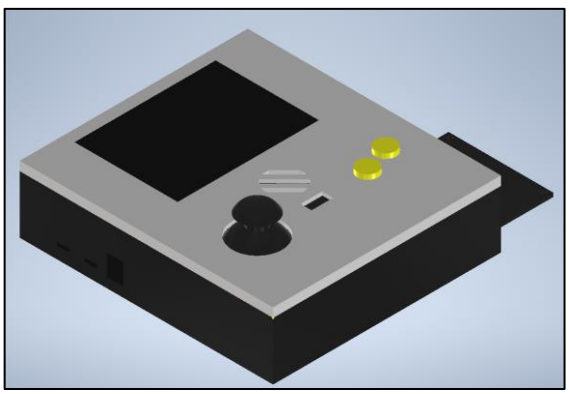
Bouton



Montage Carte/Écran



Console sans couvercle supérieur



Console avec couvercle supérieur

Figure 3 : Modélisations des pièces avec Autodesk Inventor

3. Impression

Pour imprimer, à l'aide d'une imprimante 3D, le boitier modélisé nous devions convertir nos modélisations au format « stl ». Nous nous sommes rendu compte que ce format divisait les dimensions par 10, le boitier final était de 1 cm par 2 cm. Nous avons donc adapté les bonnes dimensions.

Après l'impression nous nous sommes rendu compte que l'imprimante avait une grande imprécision au vu des détails que nous avions dans la modélisation : nous avions prévu des trous permettant de faciliter l'insertion des vis, or ces derniers n'étaient pas imprimé. Les deux parties du boitier ne s'emboîtaient pas parfaitement, nous avons dû les limer pour permettre leur emboitement. Ce problème d'imprécision concernait également l'ouverture à l'emplacement de l'écran et des boutons. Cette adaptation du modèle imprimé nous a pris énormément de temps.

4. Montage

Un fois le boitier adapté aux différents composants nous avons commencé le montage de ces éléments. Nous avions décidé d'utiliser des vis pour les différentes fixations, ceci permettait d'assurer une robustesse de l'ensemble et qui pouvait être facilement démontable si on le souhaitait pour changer les composants par exemple. Pour se faire, nous avions prévu des supports munis d'un petit trou en leur centre permettant de fixer nos composants avec des vis. Il s'est avéré que ces mêmes supports n'étaient pas assez résistants pour supporter le simple fait de visser dessus. Nous avons alors abandonné les vis comme mode d'assemblage.

Sur les conseils de notre professeur, Monsieur Masson, nous avons opté pour une fixation à l'aide de la colle à l'aide d'un pistolet à colle. Nous avons donc collé nos éléments, sauf la carte qui posait un problème quant à la fixation avec la colle, nous l'avons donc fixée avec du scratch.

Pour finir nous nous sommes penchés sur les câblages. Nous étions partis au préalable avec l'idée de ne faire que des soudures mais après quelques essaies nous nous sommes rendu compte que la solidité de nos soudures n'était pas suffisante, nous avons donc utilisé à la place des câbles mâle/femelle ce qui nous a permis de réaliser un câblage relativement propre et solide.

Nous avons malgré tout dû souder certains composants comme les résistances avec les boutons. Une fois avoir vérifié que tous les composants pouvaient bel et bien fonctionner à plusieurs reprises, nous avons celé le boîtier avec à nouveau le pistolet à colle.

IV. Partie code

1. Librairie utilisée

Pour notre projet nous avons dû utiliser une seule librairie à savoir la librairie UTFT qui nous donne accès aux fonctions de base nous permettant de contrôler notre écran, dessiner des fonctions de base tel que des ronds, des rectangles, redéfinir la couleur du fond, effacer l'écran, etc...

2. Librairies programmées

Nous avons dû programmer nos propres librairies afin d'utiliser nos boutons et notre jostick.
Ces fonctions, permettent de nous envoyer des données facilement exploitables par nos jeux. Pour notre joystick, la libraire nous permet d'obtenir son état pour :

- l'orientation horizontale
- l'orientation vertical
- le click du joystick, savoir si il est pressé ou non.

Pour nos boutons, la librairie permet d'obtenir son état, si il est pressé ou non.

Cependant nous avons rencontré un problème que nous n'avons pas réellement résolu : les entrées digitales avec lesquels nous lisions l'état des boutons nous renvoyaient des valeurs non cohérentes par rapport à notre circuit, nous nous sommes dit que c'était probablement dû au fait que nous avions branché beaucoup de composant sur la carte simultanément, nous avons donc vérifié l'état des boutons à partir des entrées analogiques qui nous donnaient des résultats qui étaient également absurdes, mais qui expliquaient les résultats des entrées digitales : sans câble branché sur ces entrées, le courant variait « naturellement » entre 2,5V et 3V. Les entrées analogiques nous permettaient donc une certaine flexibilité pour pouvoir avoir des données exploitables, cependant une variation de tension affecte le fonctionnement du programme, en effet lors de la démonstration lors de notre présentation, l'entrée analogique du bouton B (droit) dépassait les 4.88V, seuil au-delà duquel nous estimions que notre bouton était pressé, alors que nous n'appuyons pas dessus.

3. Pong

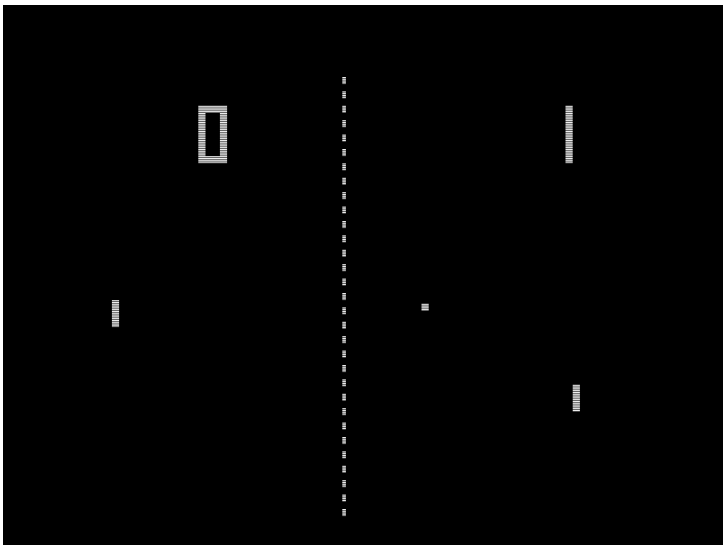


Figure 4

Le pong est un jeu d'arcade (Figure 4) , le but du jeu est simple, nous sommes le joueur à gauche de l'écran et notre but est de renvoyer la balle à l'adversaire jusqu'à ce qu'il échoue en se déplaçant verticalement. Ce jeu à été codé par Manuel Enzo.

Pour réaliser le jeu j'ai donc réalisé au préalable une liste des différentes variables et fonctions qu'allait devoir comporter le jeu. A l'aide cette liste, j'ai réalisé différentes classes :

- la classe balle qui permet de gérer son déplacement, sa vitesse, son orientation(angle) et son affichage.
- la classe paddle qui permet de gérer son déplacement, sa vitesse et son affichage.
- et enfin la classe pong qui regroupe le coeur du jeu, à savoir : le déroulement de la partie, l'affichage de la carte générale du jeu, le fonctionnement de l'ia (joueur ordinateur à droite), les collisions, le changement des scores,et la fin de partie.

La réalisation de ces classes m'a permis de facilité le codage du jeu par la suite, pour m'y retrouver plus facilement.

Voici le diagramme du fonctionnement du jeu (la fonction startGame de la classe pong) : (Figure 5)

4. Snake

Le but de ce jeu est assez simple. Nourrir votre serpent pour le faire grandir en évitant les collisions avec lui-même et l'environnement (dans mon jeu et dans la grande majorité des Snake, l'environnement se limite aux murs ou à lui même). Ce jeu a été codé par Benharrats Amine.

Pour ce jeu j'ai utilisé un type de variable personnalisé : les structures. C'est-à-dire que tous les objets de mon jeu (« gameItem ») auront les types définis dans la structure ici une position « X » et une position « Y ».
Il y'a donc deux types d'objets la nourriture (« snakeFood ») et le serpent en lui-même (« snake »). J'ai défini la taille des objets dans une variable et la longueur de départ du serpent. J'ai utilisé d'autres variables pour, notamment, la direction etc..
J'ai aussi utilisé différentes fonctions pour :

- dessiner le serpent,
- dessiner la nourriture,
- gérer les collisions
- les entrées et changer les valeurs du serpent (pour le faire avancer ou grandir).

Comme on peut le voir avec le diagramme de fonctionnement (Figure 6) du jeu il me manquait un élément majeur à ma partie programmation qui faisait que le jeu « buggait », c'était la gestion des états du jeu. J'avais au début essayé et avait créé des variables « enum » avec les différents états du jeu pour la fonctionnalité des « swtich/break » pour changer ses états. Cela provoquant plus de bug. Au final le jeu gérait les collisions (sauf avec lui-même), les déplacements, l'agrandissement du serpent mais à chaque boucle recréait toute la nourriture et réinitialiser les coordonnées des collisions (c'est pour cela qu'on pouvait indéfiniment tourner sur nous même).

Ainsi même si mon jeu n'a pas eu toutes les fonctionnalités, prévues initialement, j'ai pris un grand plaisir à le coder et découvrir un nouveau langage.

V. Conclusion

Pour conclure, malgré les difficultés rencontrées lors de la réalisation de notre projet, ce dernier rempli la plupart des objectifs fixés, la console est compacte et permet de jouer à des jeux d'arcades. Il reste cependant des améliorations possibles, comme changer le composant d'alimentation de pile pour rendre la Gameduino autonome sans la nécessité de la relier à un câble électrique. Nous pouvons également améliorer les différents jeux existants et en rajouter d'autres.

VI. Bibliographie

https://letmeknow.fr/blog/2013/11/07/tuto-utiliser-un-joystick-2-axes/

https://www.arduino.cc/en/Guide/ArduinoDue

https://www.arduino.cc/en/Hacking/PinMappingSAM3X

http://wiki.sainsmart.com/index.php/3.2%22_TFT_Touch_LCD

http://www.rinkydinkelectronics.com/library.php?id=87

Arduino - ArduinoDue
Open-source electronic prototyping platform enabling users to create interactive electronic objects.

Arduino - PinMappingSAM3X
Open-source electronic prototyping platform enabling users to create interactive electronic objects.

3.2%22_TFT_Touch_LCD

VII. Index

Figure 5

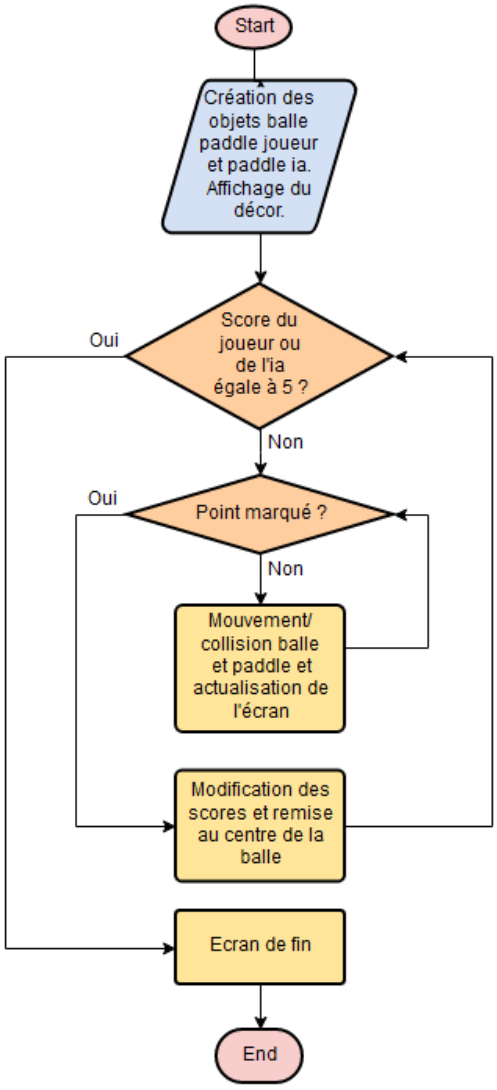
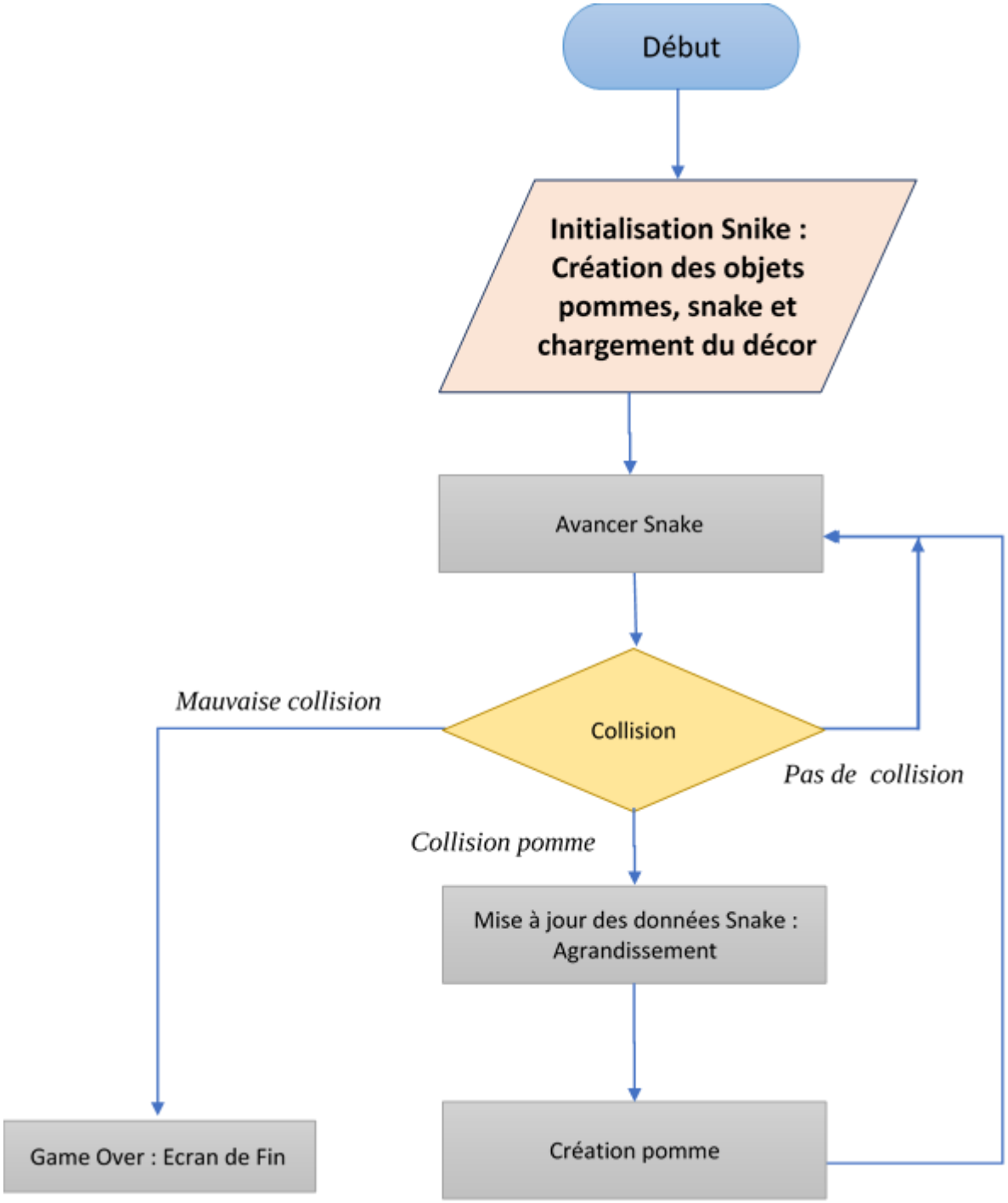


Figure 6


Début
Initialisation Snike : Création des objets pommes, snake et chargement du décor
Avancer Snake
Collision
Mauvaise collision
Pas de collision
Collision pomme
Mise à jour des données Snake : Agrandissement
Création pomme
Game Over : Ecran de Fin