Dossier Flappy Shark

https://github.com/clota974/projet-info-insa-2020-A2

I. Descriptif cahier des charges initial

A. Le but du Jeu

Le but de notre projet est de concevoir un Flappy Bird. Ce jeu est un clicker très connu dans le monde des jeux pour smartphone.Nous avons le contrôle sur un oiseau.

Des obstacles sont placés sur sa route de part et d'autre de l'écran (comme des stalactites et stalagmites pour imager), laissant un passage.

L'oiseau est en proie à la gravité : si le joueur appuie sur ESPACE, il saute et prend de la hauteur, ce qui lui permet de ne pas tomber et de remonter (s'il est trop "bas" par rapport au passage) ; s'il est trop "haut" par rapport à ce dernier, il suffit de ne pas cliquer pour qu'il chute puis de le faire sauter pour traverser l'obstacle.

Ainsi, plus le passage entre les deux obstacles est étroit, plus l'oiseau a des chances de se cogner.

Quand l'oiseau saute, sa trajectoire est une parabole concave.

Ce jeu est donc d'une part un jeu d'adresse et d'autre part un jeu d'endurance.

B. Cahier des charges initial

Pour le cahier des charges initial les fonctionnalités étaient les suivantes:

Le but est de gagner des points en allant le plus loin possible sur le parcours (1 obstacle passé = 1 point). Aussi, des graines seront disposées le long du parcours, le joueur a le choix de les récupérer ou non (1 graine = 10 points).

L'utilisateur peut avoir plusieurs choix de difficultés.

Il peut choisir une vitesse constante sur tout le parcours (soit une vitesse normale ou une vitesse plus rapide) ou une vitesse progressive qui évolue tout au long de la partie.

De plus, il pourra sélectionner la difficulté des obstacles voulus (de cette manière le passage de l'oiseau sera plus ou moins étroit).

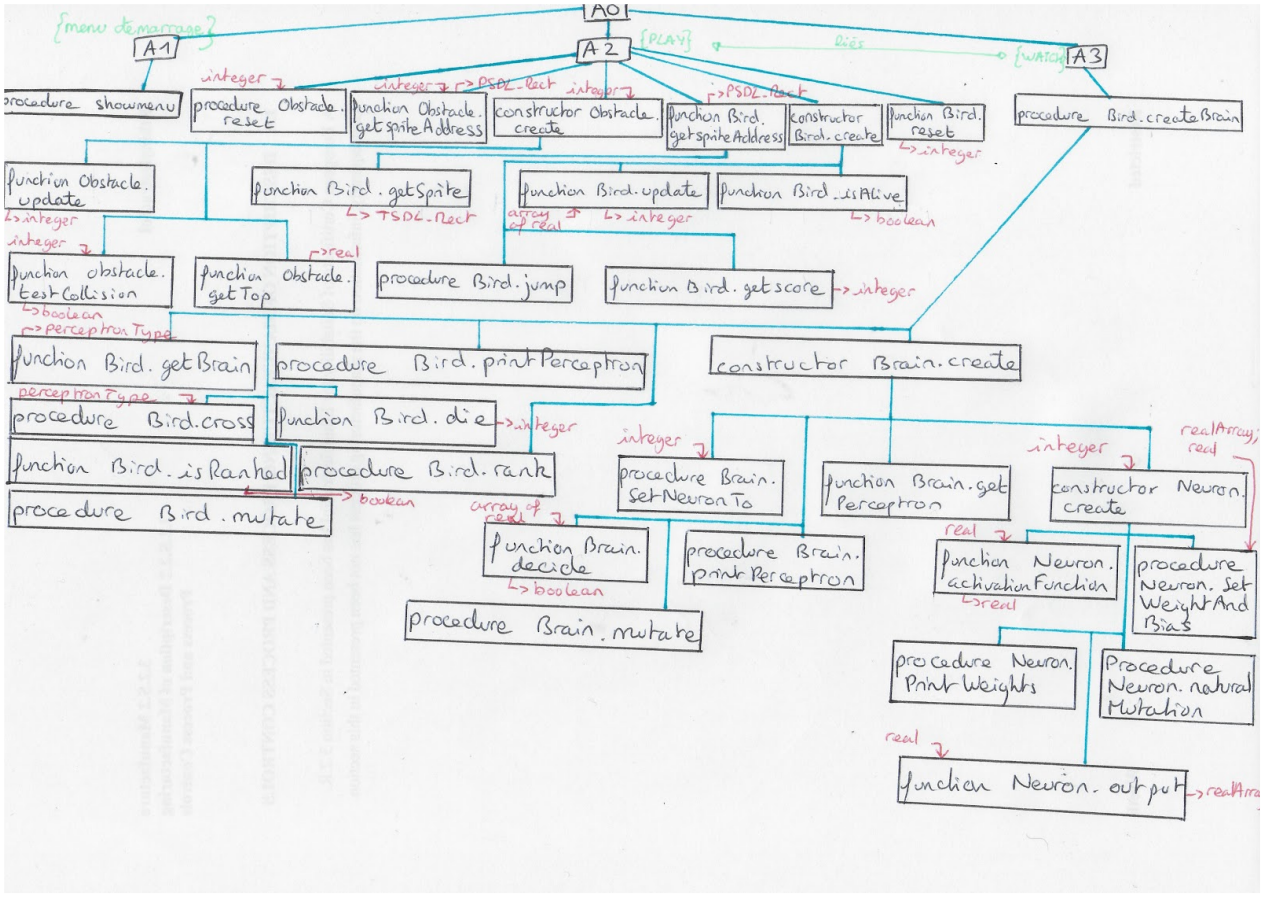
C. On avait donc les fonctionnalités suivantes:
   - ➔ Menu de démarrage (option jouer ou observer l'IA)
   - ➔ Intelligence artificielle génétique (Population, évaluation, reproduction)
   - ➔ SCORE / Menu de fin / Bonus
   - ➔ Images et SDL
   - ➔ Changement vitesse
   - ➔ Collisions et mouvements des objets du jeu

II. Conception globale

A. Modification sur le cahier des charges initial

Quelques modifications ont été apportées au cahier des charges initial dont les différentes vitesses en effet le programme ne permet que d'utiliser une vitesse constante et non de choisir un mode "expert", "avancé" ou "débutant". Nous n'avons pas ajouté de fonctionnalité vitesse car il aurait fallu la rajouter dans la partie IA et nous n'aurions pas eu le temps. Par ailleurs, il n'y a pas de bonus sur le parcours car nous avons privilégié le fait d'avoir un fond, des obstacles et des requins dessinés par nous ou via l'ordinateur.

B. Analyse descendante



C. <u>Signatures des fonctions et procédures</u>

Pour la structure du programme nous avons choisi de tout regrouper dans un seul fichier, bien qu'à terme, il s'agit d'une opportunité d'évolution de la structure du programme. Cependant, on remarque que les fonctions sont très élémentaires, ce qui améliore la lisibilité du code. En effet, nous avons un fil principal qui reste petit par rapport à celui du fichier.

Quant à l'intelligence artificielle, il s'agit d'un algorithme génétique avec crossover (simplifié) avec les paramètres suivants :

Nom de variable (insensible à la casse)	**Valeur**	**Explication**
`mutationProbability`	`0.25`	Probabilité de changer un poids/biais

`ellitism`	`0.1`	Part de la population qui reste inchangée lors la prochaine génération
`populationTotal`	`100`	Nombre d'individu au départ de chaque génération
`crossoverRate`	`0.15`	Probabilité de réaliser un "crossover", c'est-à-dire de copier les poids et biais du meilleur individu de la génération précédent
`randomBehaviour`	`0.2`	Part de la population dont le perceptron sera entièrement régénéré

Le perceptron est un réseau de neurones artificiels à 3 couches.
Toutes les valeurs sont standardisées en divisant par la taille de la fenêtre (500) pour obtenir des valeurs entre 0 et 1.

REMARQUE SUR LE MODÈLE : Avec plus de temps (et surtout plus de connaissances de notre part), nous aurions pu améliorer la qualité du modèle, notamment en ajustant les paramètres. En effet, le programme apprend bien à jouer mais cela peut prendre jusqu'à 60 générations avant de voir un individu passer la barre de 3s.

<table>
<tr><td rowspan="3">Couche 1
Entrée / Features</td><td>Position Y de l'oiseau</td><td rowspan="8">Fonction d'activation
SIGMOÏDE</td></tr>
<tr><td>Distance X au prochain obstacle</td></tr>
<tr><td>Distance Y au prochain obstacle (Y1)</td></tr>
<tr><td rowspan="5">Couche cachée</td><td>h_1</td></tr>
<tr><td>h_2</td></tr>
<tr><td>h_3</td></tr>
<tr><td>h_4</td></tr>
<tr><td>h_5</td></tr>
<tr><td>Sortie</td><td>OUTPUT</td></tr>
</table>

➔ Menu de démarrage:

```
procedure showmenu (); //Affiche le menu ainsi que
le texte qui l'accompagne
```

➔ Obstacle:

```
constructor Obstacle.create(ix: integer);
//Création obstacle: position du rectangle (sur les
axes) et taille (hauteur et largeur), espace entre
les espaces 0 et 1

function Obstacle.testCollision(birdY: integer):
boolean;  //Test de la collision avec l'oiseau
lorsqu'il passe l'obstacle en fonction de la
position de l'oiseau le long de l'axe Y et de sa
largeur

function Obstacle.update(): integer; // Evolution de
la position des obstacles 0 et 1 selon l'axe X

function Obstacle.getSpriteAdrress(ix: integer):
PSDL_Rect;
function Obstacle.getTop(): real;
procedure Obstacle.reset(ix: integer); //Reset
l'obstacle
```

➔ Neuron

NEURON

```
{ CYCLE }
constructor Neuron.create(outNeurons: integer);

{ FUNCTIONS }
function Neuron.activationFunction(v:real): real; // Sigmoide
function Neuron.output(v:real): realArray; // Retourne un tuple de forme de taille de la couche
suivante

{ PROCEDURES }
procedure Neuron.SetWeightAndBias(newWeights: realArray; newBias: real);
procedure Neuron.naturalMutation();
procedure Neuron.printWeights(); // Fonction de débogage
```

➔ Brain:

```
{CYCLE}
constructor Brain.create();

{FUNCTIONS}
function Brain.getPerceptron(): perceptronType;
function Brain.decide(features: array of real):
boolean;

{PROCEDURES}
procedure Brain.setNeuronTo(layerIx,NeuronIx:
integer; newNeuron: Neuron); // Copie les valeurs du
neurones afin de déréférencer le neurone à copier
procedure Brain.mutate(); // Effectue une mutation
procedure Brain.printPerceptron();// Fonction de
débogage
```

➔ Bird:

- ❑ intelligence artificielle:

```
procedure Bird.createBrain();
function Bird.die(): integer; // Mutateur
```

```
procedure Bird.mutate();
procedure Bird.rank(); // Mutateur de ranked
function Bird.isRanked(): boolean; // Assesseur
function Bird.getCoordinates(): coordsPtr; //
Renvoie un tuple de coordonnées
function Bird.getBrain(): perceptronType; //
Renvoie les poids et biais du perceptron
procedure Bird.printPerceptron(); // Fonction
de débogage
procedure Bird.cross(brainClone:
perceptronType); // Effectue un crossover
```

- programme basique:

```
constructor Bird.create(); //Création de
l'oiseau: position du point sur l'axe y à x=0,
évolue seulement sur l'axe y
procedure Bird.jump(); //Mise en place de la
réponse de l'oiseau par un saut sur l'axe y
lors de l'appuie sur la touche espace du
clavier
function Bird.update(inputs: array of real):
integer; //Evolution de la position de l'oiseau
le long de l'axe y
function Bird.reset(): integer; //Reset
l'oiseau
function Bird.isAlive(): boolean; //Vérifie que
l'oiseau n'a pas encore percuté un obstacle à
chaque reset, c'est-à-dire que les coordonnées
de l'oiseau ne sont pas les même que celles de
l'obstacle
function Bird.getSprite(): TSDL_Rect;
```

```
function Bird.getSpriteAddress(): PSDL_Rect;
function Bird.getScore(): integer; //Ajoute des
points au score total lorsque l'oiseau gravi un
obstacle
```

III. Guide d'utilisation si nécessaire

Pour jouer à ce jeu, il suffit de cliquer " jouer" sur le menu de démarrage. Le jeu se lancera lorsque l'utilisateur aura tapé pour la première fois sur la touche Entrée. Le requin commencera donc à avancer parmi les obstacles et le but, comme mentionné en première partie, sera de ne pas les toucher et de ne pas tomber sous l'effet de la gravité, en sautant avec la touche espace.
En mode Watch, il faut appuyer sur la touche ESCAPE du clavier afin de revenir au menu principal.

Un Easter Egg se trouve dans le jeu afin de pouvoir changer les sprites des joueurs.

IV. Le travail de groupe

Etant donné la situation actuelle et la répartition des épreuves, nous avons été bousculés que ce soit dans la répartition des tâches ou l'avancée générale du projet.
Julie était rentrée chez elle au début du confinement, ce qui n'a pas été pratique pour organiser des réunions à cause du décalage horaire.

Malgré tout, avec un peu de précipitations, nous avons réussi à boucler le projet dans les temps. Nous avons trouvé des alternatives pour travailler à distance, en envoyant le code sur GitHub, en faisant des réunions zoom lorsque c'était possible et en écrivant nos progressions, nos questions et des commentaires sur Google Drive et Messenger.

Le planning a été dur à tenir étant donné la situation et les changements d'emploi du temps qui ont suivi, malgré quelques modifications du cahier des charges faute de temps, le projet est achevé.