Extension utilisée : IZIone + Snake

Partie 1 - Comprendre

1 Le Snake, c'est quoi ?

Avant-propos : Durant cette activité répartie sur 3 séances, vous allez apprendre un programmer un Snake sur votre IZIone et sur mBlock. La difficulté sera croissante de séance en séance. Cette première séance sera très « simple » afin que tu puisses déjà t'amuser à programmer et à jouer !

Le Snake (=serpent en anglais) est un jeu où le but est de faire grandir son serpent en en attrapant de la nourriture. Le serpent n'a pas le droit de se toucher, sinon le joueur perd. Il n'a pas le droit de toucher les murs, sinon le joueur perd également. Le serpent se dirige grâce aux actions du joueur.

À TOI ! - À partir du descriptif du jeu, détermine les directions possibles que peut prendre le serpent

……

CONTINUE - Détermine le nombre d'actions nécessaires pour faire bouger le serpent dans tous les sens. Que devras-tu utiliser sur ton IZIone ? Décris leurs actions.

……

Impression écran Snake qui va manger

Regardez ce serpent dans son environnement naturel qui s'apprête à manger sa proie afin qu'il puisse grandir !

Détermine le type de boucle dans mBlock pour programmer cette action. Comment ça marche ?

……

Partie 2 – Si… alors...

1 Liens entre l'homme et la machine

KEEP GOING - Rédige sous la forme Si… Alors… tout ce qu'il faut pour faire bouger ton serpent dans toutes les directions. (Ex : Si j'appuie sur le bouton gauche, Alors…)

……

GO ON - Rédige une condition supplémentaire qui concerne la taille du serpent.

……

2 Une difficulté supplémentaire

À TOI ! - Il faudra rajouter dans ton code une fonction précise qui s'appelle « renouveler la nourriture ». Dans quel bloc conditionnel vas-tu la mettre et dans quel ordre ?

……

Est-ce que l'ordre des blocs à l'intérieur d'une boucle est important ? Dans notre cas ?

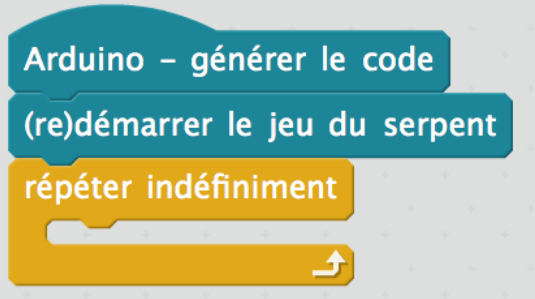
……

Partie 3 - Réalisation

1 C'est l'heure de programmer sur mBlock

VA SUR MBLOCK - Lance mBlock et installe l'extension Snake et IZIone si ce n'est pas fait (retourne sur le Manuel d'Utilisateur si tu as un doute)

LANCE-TOI - Tu as tous les éléments pour réussir, alors réalise le programme qui te permettra de jouer.



Coup de pouce : ton programme devra commencer comme ceci :

Intègre le reste de ton programme dans la boucle « répéter indéfiniment ».

Cette dernière te permet de faire bouger ton serpent plus d'une fois.

2 Limites

N'OUBLIE PAS - Pour l'instant tu as programmé un Snake très simple. Il manque plusieurs conditions. C'est entre autres la raison pour laquelle tu ne pourras pas perdre (pour l'instant) avec votre Snake. Tu n'as pas encore programmé les conditions d'échecs ou de victoire.

Nous verrons tout ceci lors de la séance 2. Sauvegarde ton travail sur ta session !

Fiche Travail : PROGRAMMER « SNAKE » - Séance 1 - CORRECTION

Partie 1 - Comprendre

1 Le Snake, c'est quoi ?

À partir du descriptif du jeu, détermine les directions possibles que peut prendre le serpent.
Vers le haut, bas, gauche, droite.
Il faut bien poser les bases de la réflexion autour de cet exercice de serpent et surtout bien comprendre en quoi consiste le jeu. Faire déplacer le serpent pour qu'il attrape de la nourriture afin qu'il puisse grandir au maximum.

Notez que dans le jeu du Snake, le serpent n'est pas censé revenir sur lui-même. Il s'agit ici d'un problème de physique. En revanche le programme ne se préoccupe pas de la physique rien n'empêche le serpent de revenir sur lui-même. Si ce n'est le programme lui-même. Pour nous vous facilitez la tâche, nous avons fait en sorte que le serpent ne puisse pas revenir en arrière. Donc si le serpent va vers la droite, il ne pourra pas aller vers la gauche SAUF s'il effectue un virage au préalable.

Détermine le nombre d'actions nécessaires pour faire bouger le serpent dans tous les sens. Que devras-tu utiliser sur ton IZIone ? Décris leurs actions.
Quatre. Lorsque j'appuie sur le bouton du haut, le serpent se dirige vers le haut. Lorsque j'appuie sur le bouton du bas, le serpent se dirige vers le bas. Etc…
Vous allez devoir utiliser les actionneurs de votre IZIone, c'est-à-dire les boutons. Ils sont au nombre de 5, mais vous en utiliserez que 4 aujourd'hui.
Leurs actions sont décrites comme telles : si j'appuie sur un des actionneurs, j'envoie un message à mon IZIone. Ici faire changer le serpent de direction.

Détermine le type de boucle dans mBlock pour programmer cette action
Boucle conditionnelle. Si… Alors… SI le serpent mange nourriture ALORS il grandit.

De la même manière vous pourrez comprendre qu'il s'agit du même type de boucle pour faire déplacer le serpent : SI bouton du haut pressé ALORS le serpent va vers le haut.
Si les conditions sont vraies, alors les instructions sous le « Si » seront exécutées.

Remarque : À ce stade, vous êtes déjà capable d'écrire et de faire un programme pour avoir un Snake très simplifié. Le serpent bouge et s'agrandit.
À ce niveau, le serpent peut « traverser » les murs et se toucher. Ces contraintes physiques seront vues plus tard lors de la séance 2 pour affiner le programme.

Partie 2 – Si… alors...

L'intérêt de cette partie est de faire travailler l'élève sur la boucle conditionnelle Si… Alors…
L'élève pourra déduire à l'aune de la première partie (Comprendre) de l'activité l'effet de causalité. Si j'appuie sur un bouton, alors le serpent bouge.

1 Liens entre l'homme et la machine

Rédige sous la forme Si… Alors… tout ce qu'il faut pour faire bouger ton serpent dans toutes les directions. (Ex : Si j'appuie sur le bouton gauche, Alors…)
Si j'appuie sur le bouton de gauche, Alors le serpent tourne vers la gauche. Si j'appuie sur le bouton de droite, Alors le serpent tourne vers la droite.
Si j'appuie sur le bouton du haut, Alors le serpent tourne vers le haut. Si j'appuie sur le bouton du bas, Alors le serpent tourne vers le bas.

On voit bien ici l'intérêt des actionneurs et les interactions en l'homme et votre IZIone. Elle exécute ce que vous lui dites.

Rédige une condition supplémentaire qui concerne la taille du serpent.
Si le serpent mange la nourriture, alors il grandit.

Dans ce cas, la cause de l'exécution des instructions n'est plus l'homme, mais une fonction qui est dans votre programme. Lorsque cette fonction est « vraie » alors le programme exécute les instructions de la boucle conditionnelle.

2 Une difficulté supplémentaire

Il faudra rajouter dans ton code une fonction précise qui s'appelle « renouveler la nourriture ». Dans quel bloc conditionnel vas-tu la mettre et dans quel ordre ?

Il faut rajouter la fonction « renouveler nourriture » dans le bloc conditionnel lié à la taille du serpent.

Si le serpent mange la nourriture, alors :
- le serpent grandit
- renouveler la nourriture

Cette fonction permet de changer les coordonnées de la nourriture. Sinon elle réapparaît au même endroit. C'est un peu dommage.

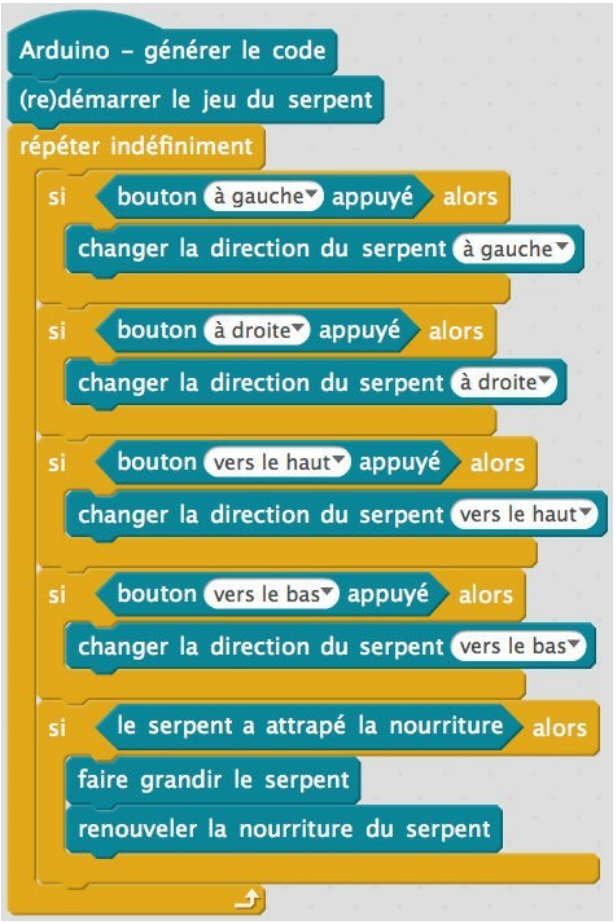
Est-ce que l'ordre des blocs à l'intérieur d'une boucle est important ? Dans notre cas ?
En général l'ordre dans lequel les instructions sont écrites est très important, nous le verrons dans la séance 2.
Dans notre cas précis, faire renouveler la nourriture avant de faire grandir le serpent ne serait pas dérangeant. Ça contrevient simplement à la bonne compréhension et à l'ordre des choses. Mais dans les faits, rien ne va changer dans l'utilisation de votre programme.

Partie 3 - Réalisation

1 C'est l'heure de programmer sur mBlock

Comme vous pouvez le voir, ce programme pour le Snake est très simple. Mais il permet d'avoir une première expérience avec ce jeu et en programmation. Du moins pendant cette activité.

Dans ce programme nous avons :
- « Arduino - générer le code ». Il faut toujours avoir ce bloc en tête de votre programme principal pour ensuite téléverser le code dans votre IZIone. **Si vous ne savez pas comment faire, consulter notre manuel d'utilisation complet.**
- la fonction « (re) démarrer le jeu du serpent est assez explicite. Elle gère le démarrage du jeu. Nous travaillerons dessus un peu plus lors de la séance 2.
- la boucle « répéter indéfiniment » est là pour permettre à votre programme de pouvoir exécuter plusieurs fois les différentes boucles conditionnelles à l'intérieur. Et vous aurez besoin de faire bouger votre serpent vers le haut plusieurs fois par exemple.



Maintenant c'est à vous de jouer, littéralement.

En revanche après quelques essais vous allez peut-être ressentir de la frustration. En effet avec ce programme, vous ne pouvez pas perdre et le jeu est très, voire trop facile.
Ne vous inquiétez pas, tout va se corser dans la séance 2 où l'on va complexifier ce programme.

Extensions utilisées : IZIone + Snake

Durant cette séance 2, nous allons rajouter des composantes à notre programme initial réalisé lors de la séance 1. Pour commencer, récupérer le programme mBlock que tu as déjà fait. Nous allons partir de cette base. Nous partons du principe que tu as un programme identique à celui fourni dans le corrigé de la séance 1. Si vous n'êtes pas sûr de vous, nous vous invitons à récupérer le corrigé.
Sinon c'est parti !

Nous allons complexifier le programme du Snake étape par étape. À la fin, tu auras un programme intéressant.

Partie 1 - Initialisation

1 Variables

POURSUIS - Rappelle ce qu'est une variable en programmation.

……

RETOURNE SUR MBLOCK

Crée 2 variables dans mBlock. Une appelée « TailleMax » et un autre « Score ». Sous « Arduino - générer le code » mets la première variable à 10 et la seconde à 0
Explique pourquoi on associe ces valeurs à nos variables.

……

2 Démarrage du programme

CONTINUE - Fais en sorte qu'au démarrage ton serpent commence en allant vers le bas (du haut vers le bas).

DON'T STOP - Fais en sorte que ton programme attende jusqu'à ce que « commencer » soit relâché pour démarrer le Snake.
N'hésite pas à tester ton programme pour vérifier que l'initialisation est correcte.

Partie 2 – Victoire ou défaite

1 Les contraintes physiques

KEEP GOING - Liste l'ensemble des contraintes physiques qui s'exercent sur le serpent (Coup de pouce : il y en a 3. Dont une que tu connais déjà, il ne peut pas revenir sur lui-même).

……

GO ON - Explique ce qu'il se passe si le serpent est confronté à ces limites.

……

À TOI - Combien y a-t-il de cas où le joueur perd ? Est-ce que le joueur peut gagner ?

……

Ci-contre, tu as tous les éléments pour gérer les cas où le joueur perd. Retourne sur mBlock et ordonne les instructions pour que le programme fonctionne correctement.

Le jeu Snake a pour réputation d'être un jeu très difficile à finir. Alors on a décidé de créer un programme pour le jeu où l'on peut gagner (quasiment) à tous les coups.

Voici une partie du programme du Snake. C'est elle qui permet au joueur de gagner.

Refais le programme et complète-le (ligne 2,5, 7) et explique ce qu'il se passe ligne par ligne.

……

……
…

Partie 3 - Réalisation

1 Assemblage de tous les blocs

SUR MBLOCK - Assemble tous les morceaux de programme réalisés jusque-là pour faire fonctionner ton code. Tu devrais avoir un serpent d'une taille de 10 pixels maximum avec un score de 7.
Tu peux augmenter la difficulté en augmentant la valeur de la variable « TailleMax ».
Tu peux encore accélérer ton serpent dès qu'il mange la nourriture pour pimenter davantage ton jeu. Autrement c'est la fin de la séance 2.

Partie 1 – Initialisation

1 Variables

Rappelle ce qu'est une variable en programmation.
Une variable en programmation permet d'associer une valeur à un nom. On va pouvoir y stocker une valeur.

Crée 2 variables dans mBlock. Une appelée « TailleMax » et un autre « Score ».

Allez dans l'onglet Script > Blocs & variables > Créer une variable > renseignez les noms > faites OK.

Notez bien que vous allez créer en même temps les fonctions
« mettre x à y », « ajouter à x y », « montrer la variable x » et
« cacher la variable x ».
Ces fonctions apparaissent à la création d'un variable. Elles vous seront utiles.

Sous « Arduino - générer le code » mets la première variable à 10 et la seconde à 0

Rien de compliqué ici. Faites glisser les fonctions « mettre x à y » et choisissez la variable adéquate.

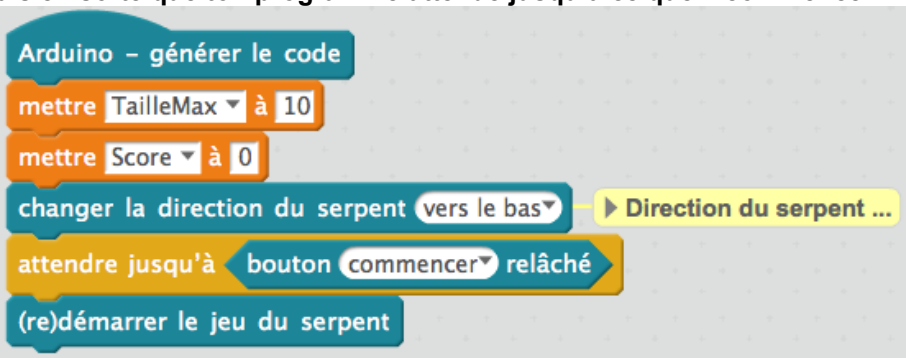
Explique pourquoi on associe ces valeurs à nos variables.
La variable « Score » va vous servir de compteur de point. Dès que vous allez manger un morceau de nourriture, on fera +1 au score. Comme vous commencez avec 0 point et qu'il faut les accumuler, on stocke la valeur 0 dans « Score ».
Pour la variable « TailleMax », nous la mettons à 10 arbitrairement. Cette variable stocke le nombre 10, et dès que le serpent aura une taille de 10 pixels, nous ferons gagner le joueur. Mais tout ça reste à programmer. Il faudra rajouter quelques instructions plus tard.

2 Démarrage du programme

Fais en sorte qu'au démarrage ton serpent commence en allant vers le bas (du haut vers le bas).

Très facile aussi.

Fais en sorte que ton programme attende jusqu'à ce que « commencer » soit relâché pour démarrer le Snake.



Un peu plus complexe. Il faut que vous utilisiez la fonction « attendre jusqu'a » et y insérer la fonction IZIone « bouton commencer relâcher ». Dès lors votre IZIone attendra d'exécuter le reste des instructions tant que vous n'avez pas appuyé sur le bouton Start. Et c'est exactement l'effet que l'on recherche.

Partie 2 – Victoire ou défaite ?

1 Les contraintes physiques

Liste l'ensemble <u>des contraintes physiques</u> qui s'exercent sur le serpent. (Coup de pouce : il y en a 3)
Le serpent ne peut pas toucher les murs, le serpent ne pas se toucher lui-même, le serpent ne peut pas revenir sur lui-même.
Lors de la séance 1, nous ne nous étions pas préoccupés de ces contraintes physiques. C'est pour ça que votre serpent pouvait se toucher et traverser les murs. Maintenant on essaie de rectifier ceci pour que notre Snake ressemble un peu plus au « vrai » Snake.

Explique ce qu'il se passe si le serpent est confronté à ces limites.

Il faut que vous imaginiez quel serait l'ensemble des possibilités si les contraintes physiques sont atteintes. Et bien le joueur perd. On introduit la condition d'échec. Notez que le joueur perd uniquement lorsque le serpent se touche ou touche les murs. Le serpent ne peut pas revenir sur lui-même et c'est tout. Pas de défaite dans ce cas.

Cependant
À ce moment de l'activité, si vous veniez à coder le programme, les contraintes physiques ne seraient pas si contraignantes. En effet si le serpent touche un mur, il réapparaît de l'autre côté. Si le serpent se touche lui- même, il se traverse. En effet votre code n'a pas suffisamment changé depuis la fin de la séance 1. Vous en êtes presque au même point. Il vous sera possible de modifier l'environnement du programme en laissant ou non certaines contraintes. Nous allons voir ça très rapidement dans cette séance.
Évidemment, ici nous allons définir une défaite.

Combien y a-t-il de cas où le joueur perd ? Est-ce que le joueur peut gagner ?
Le joueur perd dans 2 cas. (et non 3)
-Si le serpent touche les murs
-Si le serpent se touche lui même
(- le fait que le serpent ne puisse pas revenir sur lui-même n'est pas une condition de défaite, simplement une condition limitative de mouvement, le serpent ne revient pas en arrière et c'est tout).

Dans le cas d'une victoire :
Souvenez-vous que nous avons créé une variable « TailleMax » et qu'elle est à 10. On peut imaginer que si la taille du serpent = 10 alors le joueur gagne.
Vous n'avez pas encore programmé de conditions de victoire, même si vous parveniez à remplir toutes les cases de la matrice à LED avec votre serpent, vous serez confronté à une contrainte physique : le serpent ne peut pas se toucher. Et cette contrainte, nous la mettons en place tout de suite.

Retourne sur mBlock et ordonne les instructions pour que le programme fonctionne correctement.

Voilà, vous pouvez enfin perdre !!!
Remarquez la fonction verte « ou » ici présente, elle vous permet d'éviter de rédiger 2 fois la fonction Si… Alors….
Vous regroupez en une seule les deux contraintes qui impliquent une défaite du joueur. Donc si l'une des conditions est vraie, votre IZIone exécutera les instructions suivantes dans l'ordre :

- La fonction « le joueur a perdu » s'exécute, elle arrête le programme.
- Vous demandez à afficher le nombre stocké dans la variable « Score ». Votre score apparaît sur la matrice.
  Note : Nous verrons dans la partie du programme concernant la victoire comment programmer l'augmentation de votre score qui est censé être à 0 en début de programme.
- Votre IZIone attend que vous appuyiez sur le bouton Start. Nous avons déjà vu ces instructions à l'initialisation.
- Votre IZIone remet votre score à 0. Vous n'allez pas repartir du score précédent qui est toujours stocké.
- Le jeu redémarre.

Avec ces instructions vous avez enfin un Snake digne de ce nom où vous pouvez perdre. Sympa.

Voici une partie du programme du Snake. C'est elle qui permet au joueur de gagner. Refais le programme et complète-le (ligne 2,5, 7) et explique ce qu'il se passe ligne par ligne.

Allons-y pas à pas, vous connaissez le début :

- rien de nouveau dans les 4 premières lignes hormis « ajouter à Score 1 ». C'est assez clair, dès que vous attrapez de la nourriture, votre programme rajoute 1 à la variable « Score ». C'est ainsi que vous augmentez votre score et c'est pourquoi vous deviez le remettre à 0 dans l'exercice précédent.
  - On imbrique une nouvelle condition Si… Alors.

Dans celle-ci on définit clairement quand le joueur gagne. La fonction « taille du serpent » joue le rôle d'une variable. Elle stocke toute seule la taille en pixel du serpent. Quand ce dernier atteint la « TailleMax », le programme exécute le reste des instructions. Souvenez-vous que nous avons défini « TailleMax » à 10, mais vous pouvez tout à fait changer.

- Le reste du programme est assez clair, il est similaire à ce que nous avons vu dans l'exercice précédent. Notez simplement que l'on fait appel à la fonction « le joueur a gagné » dans ce cas-là.

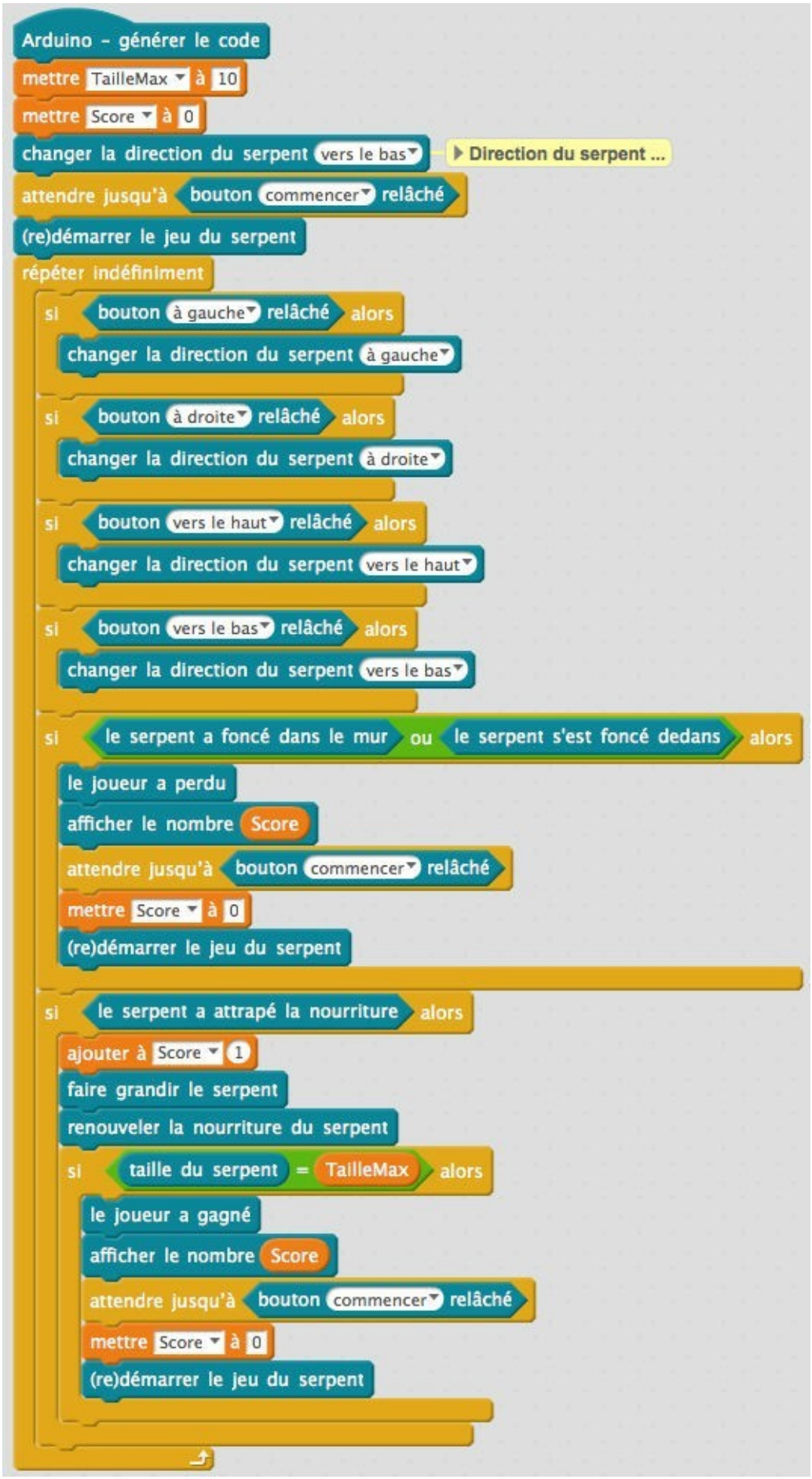
Partie 3 – Réalisation

1 Assemblage de tous les blocs

Vous avez fait le plus dur. Le reste c'est du plaisir.

À présent vous avez un Snake qui correspond en tout point à un « vrai ».

Voir le code final page suivante.


Arduino - générer le code
mettre TailleMax à 10
mettre Score à 0
changer la direction du serpent vers le bas
Direction du serpent ...
attendre jusqu'à bouton commencer relâché
(re)démarrer le jeu du serpent
répéter indéfiniment
si bouton à gauche relâché alors
changer la direction du serpent à gauche
si bouton à droite relâché alors
changer la direction du serpent à droite
si bouton vers le haut relâché alors
changer la direction du serpent vers le haut
si bouton vers le bas relâché alors
changer la direction du serpent vers le bas
si le serpent a foncé dans le mur ou le serpent s'est foncé dedans alors
le joueur a perdu
afficher le nombre Score
attendre jusqu'à bouton commencer relâché
mettre Score à 0
(re)démarrer le jeu du serpent
si le serpent a attrapé la nourriture alors
ajouter à Score 1
faire grandir le serpent
renouveler la nourriture du serpent
si taille du serpent = TailleMax alors
le joueur a gagné
afficher le nombre Score
attendre jusqu'à bouton commencer relâché
mettre Score à 0
(re)démarrer le jeu du serpent