‘

MASTER CULTURES NUMÉRIQUES

SAE IHM graphique et tangible

EL BABARTI Selma
NOURINE Jasmine
BAILLIE-DAWE Thomas

SOMMAIRE

Description du jeu

Lors de la première phase de conception, nous avons eu l'idée de réaliser un jeu inspiré du jeu T-Rex Chrome Dinosaure, qui consiste à faire sauter un petit dinosaure afin d'éviter des obstacles qui viennent à lui.

Cependant, nous avons décidé de changer quelques règles du jeu : notre personnage ne pourra pas que sauter, il pourra en fait se déplacer de haut en bas sur l'axe y mais restera toujours fixe en x. Nos obstacles quant à eux apparaitront donc aléatoirement sur l'axe des y et se déplaceront de la droite vers la gauche en x.

Nous avons aussi changé le contexte du jeu en imaginant une tortue à la place du dinosaure, qui devra éviter du plastique se trouvant dans l'océan. En revanche, elle pourra manger des méduses que nous avons décidé d'ajouter en guise de bonus (qui apporteront des points supplémentaires au joueur).

Pour mieux comprendre le contexte de notre jeu, voici son histoire. Notre jeu est intitulé **Crush survivor**. Le but est simple : le joueur est représenté par notre tortue, Crush (en référence à la tortue du dessin animé Le monde de Nemo). Crush au fil des années s'est rendu compte que son habitat est de plus en plus pollué par des déchets plastiques, que sa nourriture se fait de plus en plus rare, et tente tant bien que mal de survivre dans ces conditions.

Tous les jours Crush doit faire face aux sacs plastiques qui viennent à lui aléatoirement et qui le surprennent. A chaque sac évité, Crush gagne 1 point. Si Crush touche un sac plastique, il se trouve piégé et finit par mourir, si au contraire il réussit à manger des méduses (son repas préféré), cela le booste et il gagne plus de points. A nous de jouer pour l'aider !

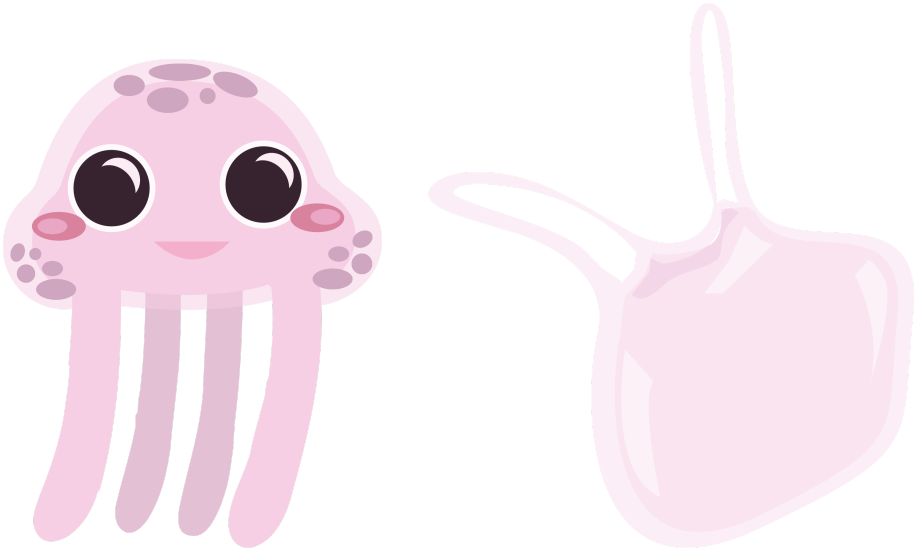
A partir de cette base-là, nous avons construit un diagramme de classe, un diagramme d'activité et un diagramme de séquence.

Maquettes (design)

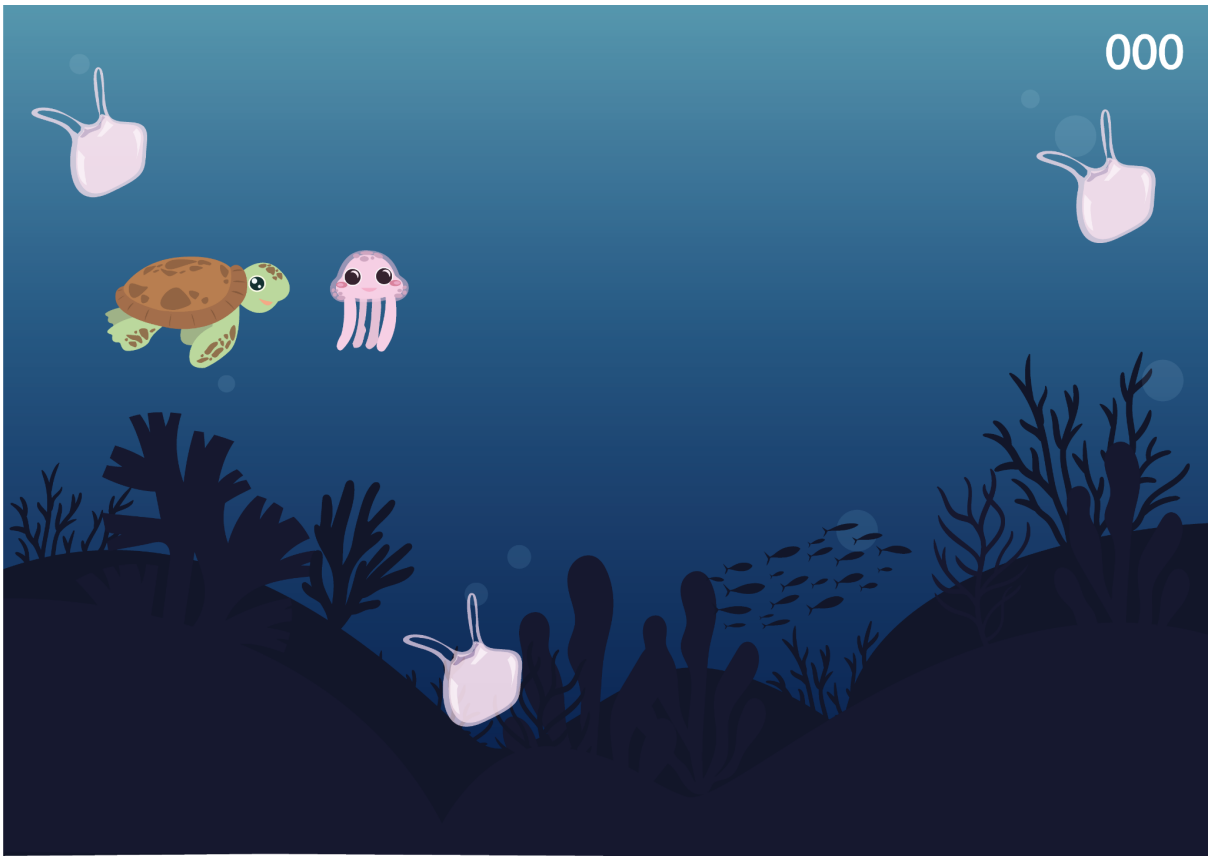
Pour le design du jeu nous avons décidé d'opter pour un graphisme simple et efficace. La tortue et la méduse sont mignonnes afin que le joueur ressente de l'empathie envers elles. Le fond est lui assez sobre afin de ne pas trop déconcentrer le joueur et de ne pas surcharger l'espace visuel avec un surplus d'informations.

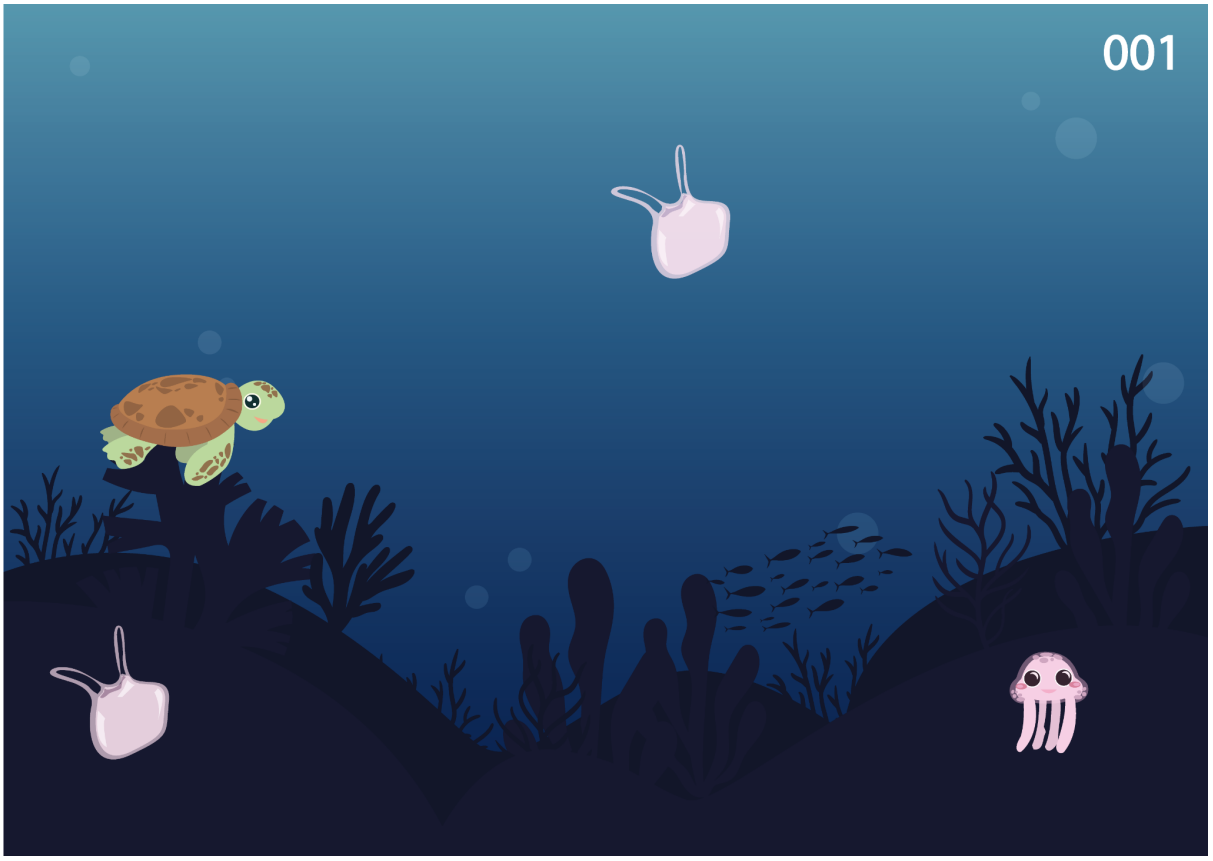
Voici la page de lancement du jeu avec le logo et le bouton "start" pour commencer. Nous avons choisi le nom "Crush" en référence au nom d'un personnage de Némo (qui est une tortue marine également).

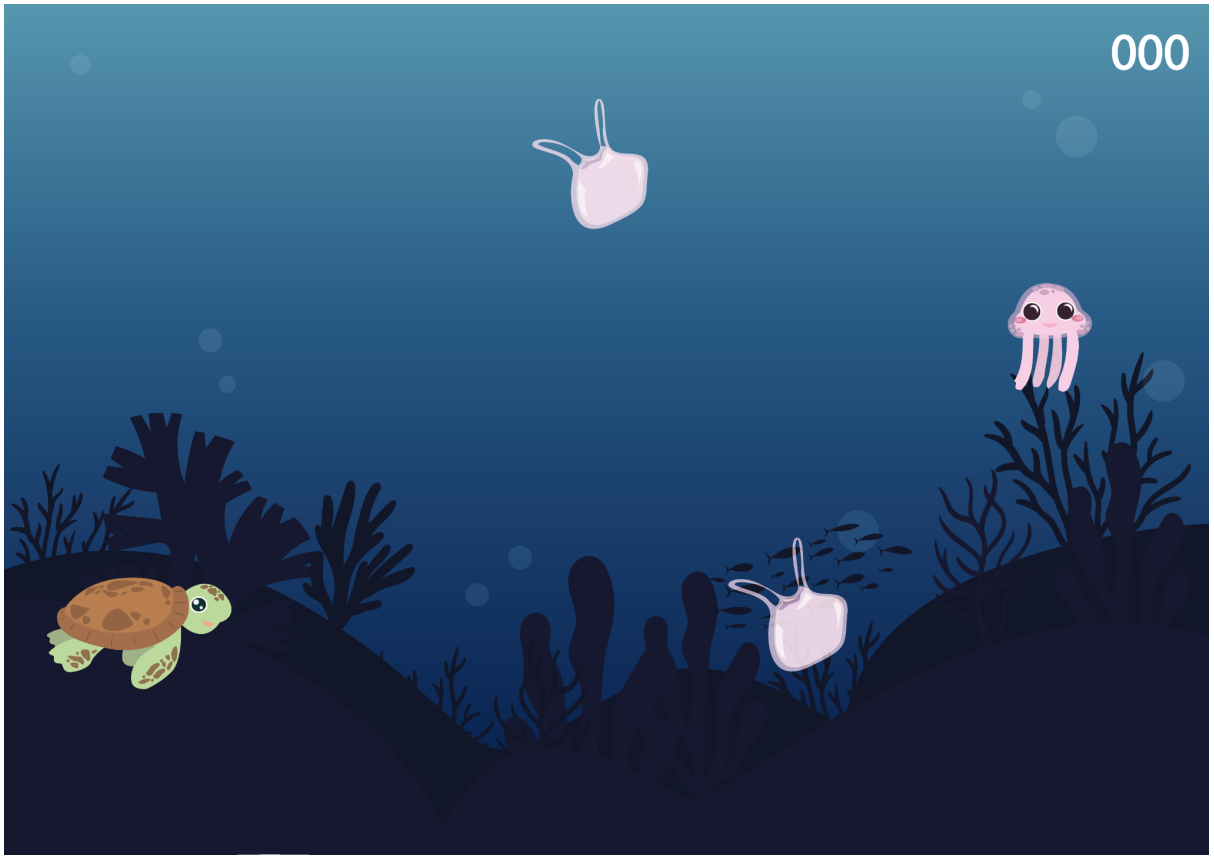
Le design de la méduse et du sac plastique sont proches car nous voulions que ces deux éléments soient assez semblables étant donné que dans la nature les tortues marines confondent les sacs plastiques avec des méduses du fait de leur forte ressemblance.



Voici l’interface final du jeu :





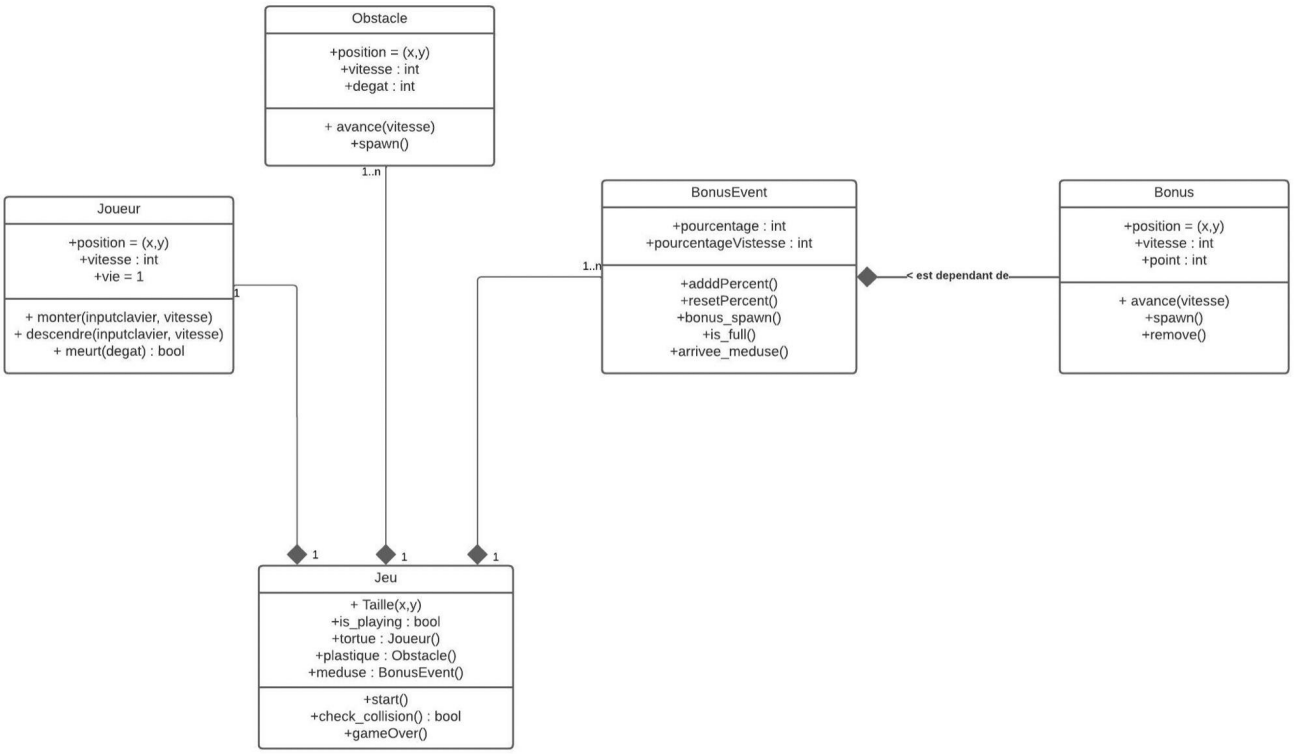

000

Documentation du projet informatique

Description UML

Nous sommes revenus à multiples reprises sur la conception de ce diagramme qui nous a à la fois aidé pour la construction du squelette de notre code et que nous avons aussi remodifié par la suite en codant pour l'adapter au mieux par rapport à ce que nous avons produit.

Voici notre version finale :



La classe Joueur : c'est la classe qui représente notre tortue. Notre joueur possède une position x qui sera fixe et une position y. Il a également une vitesse de déplacement et une vie.

Ce joueur peut soit monter lorsqu'une entrée clavier est activée, soit descendre. Il peut également mourir.

La classe Obstacle : c'est la classe qui représente nos plastiques. Ils possèdent une position x et une position y (qui sera fixe mais différente et aléatoire pour chaque plastique). Ils ont également une vitesse de déplacement et une valeur de dégâts causés en cas de collision. Ces obstacles apparaissent et peuvent avancer le long de l'axe x pour aller vers le

joueur. Ils ne disparaissent cependant pas car ils sont toujours présents de manière permanente.

La classe Bonus : c'est la classe qui représente nos méduses. Elles possèdent une position x et une position y (qui sera fixe mais différente et aléatoire pour chaque méduse). Elles ont également une vitesse de déplacement et une valeur de points qu'elles apportent en cas de collision.

Ces méduses apparaissent et peuvent avancer le long de l'axe x pour aller vers le joueur. Elles disparaissent également lorsqu'il y a une collision ou que le joueur n'a pas réussi à les attraper.

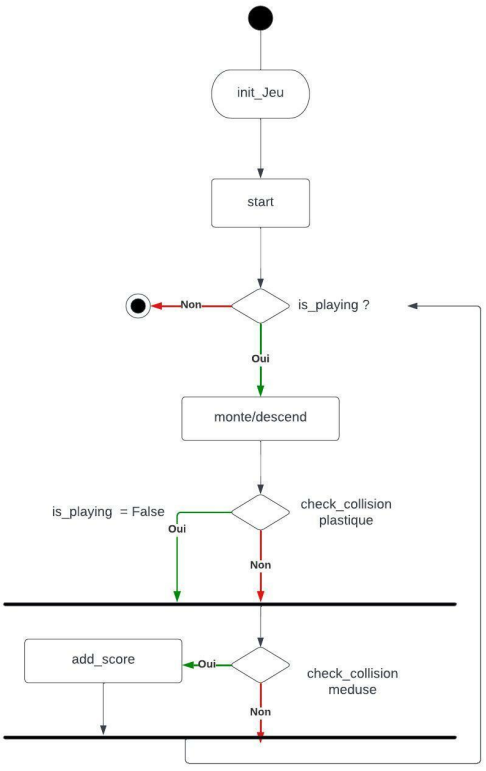
La classe BonusEvent : c'est la classe qui permet l'enclenchement de la classe méduse. Nous avons imaginé un système qui puisse faire apparaitre une méduse seulement ponctuellement : une jauge se remplit au fur et à mesure et lorsqu'elle est pleine, une méduse apparaît.

Cette jauge possède donc un pourcentage et une vitesse de montée de pourcentage. Elle peut faire apparaitre une méduse seulement lorsqu'elle est pleine.

La classe Jeu : c'est la classe qui permet d'agréger toutes les autres, c'est celle qui permet de faire commencer le jeu et de le terminer, et c'est aussi elle qui vérifie les collisions entre les différents objets.

Le diagramme d'activité

Nous avons également créé un diagramme d'activité qui nous a aussi servi d'appui pour notre code, et nous avons aussi réitéré nos versions.



La boucle ne tourne que si le jeu est toujours actif (sinon, le jeu est terminé et le joueur a perdu). **Tant que** le jeu est **actif**, les input définissent si le joueur (en l'occurrence la tortue) monte ou descend.

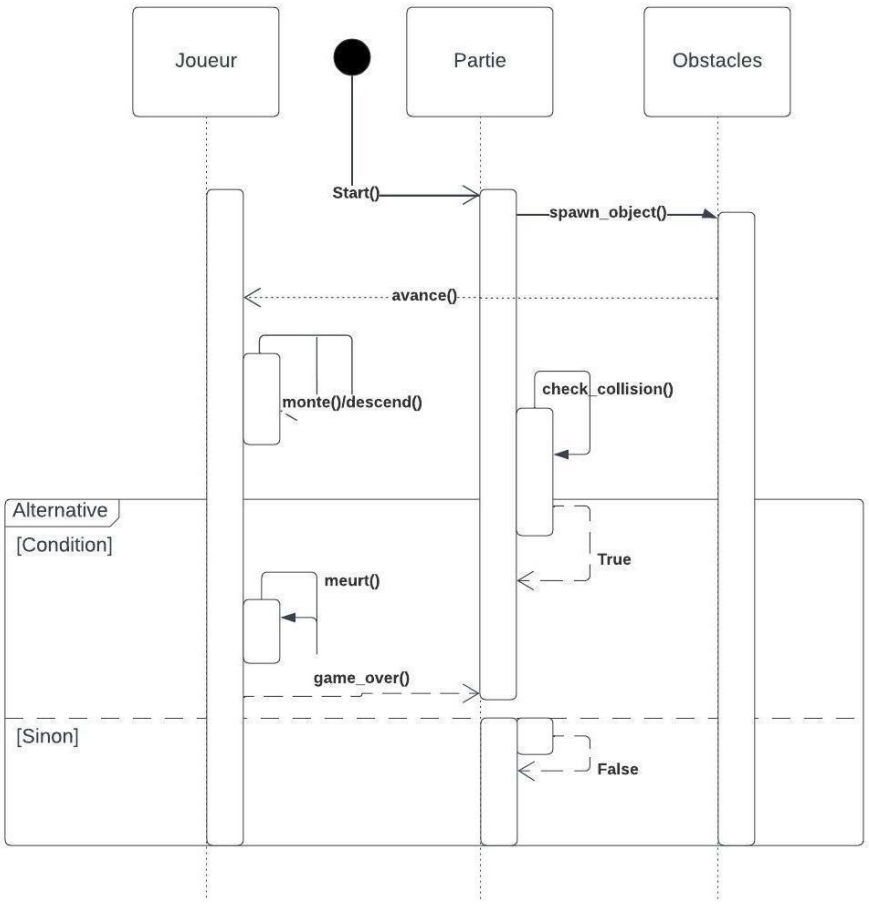
Le jeu vérifie ensuite s'il n'y a pas de **collision** avec les **plastiques** (auquel cas la boucle termine ce qu'elle doit faire mais le joueur aura perdu), et vérifie s'il y a une **collision** avec les **méduses** (dans ce cas, le joueur gagne des **points** en plus).

La boucle tourne tant qu'elle est active et qu'il n'y a pas de collision avec le plastique.

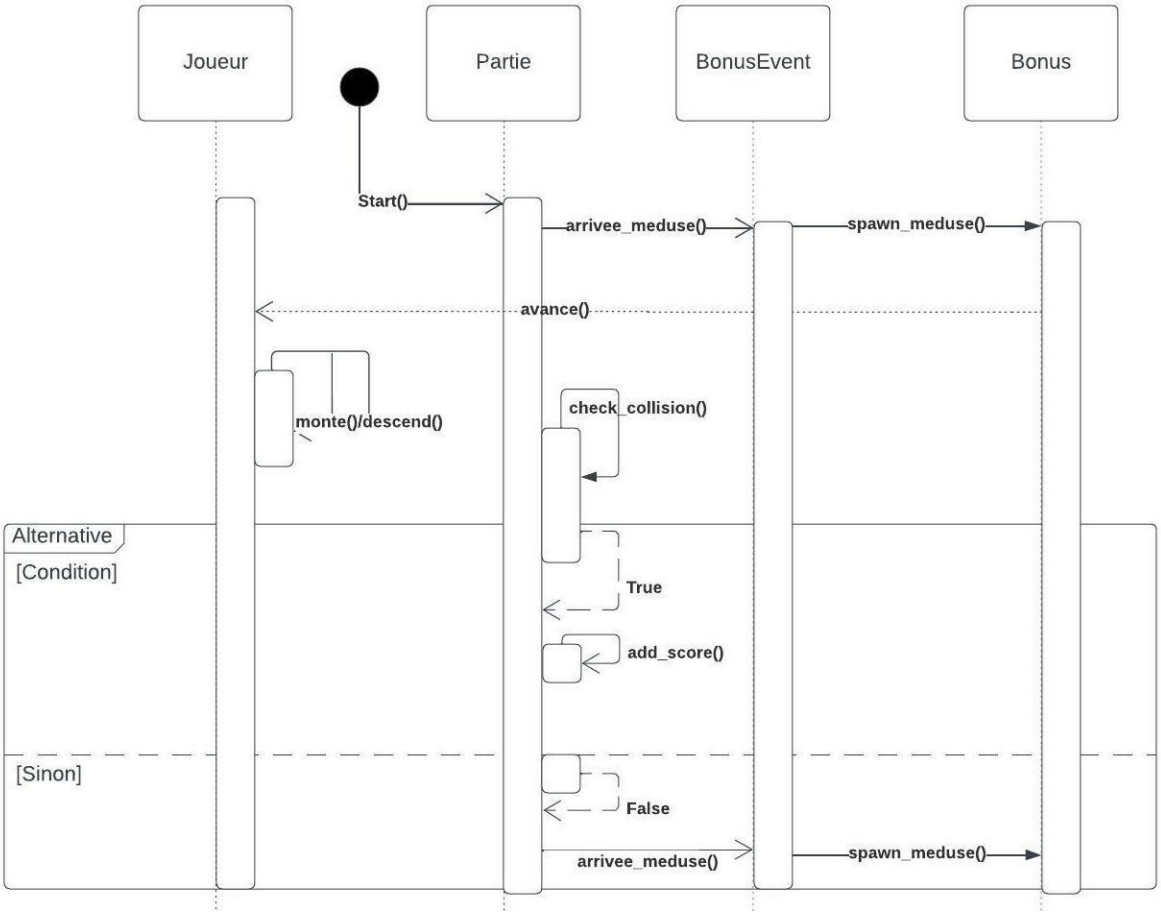
Le diagramme de séquence

Nous avons fait un diagramme de séquence pour le cas des obstacles, et un autre pour les bonus.

Pour les obstacles :



Pour les bonus :



Documentation du projet électronique

A) Principe

Nous devions réaliser une interface tangible pour l'intégrer à notre jeu. Pour cela, nous avons imaginé une situation dans laquelle les joueurs seraient face à un capteur capable de mesurer la distance grâce aux ultrasons. Ainsi, lorsque le joueur rapprocherait sa main du capteur, la tortue monterait, et lorsqu'il éloignerait sa main, celle-ci descendrait.

Nous avons également imaginé un système qui permettrait, lorsqu'il y a une collision avec une méduse, d'allumer une led en vert (pour indiquer que le joueur a bien gagné des points bonus)

Pour cela, nous allons donc utiliser deux capteurs :

- Le capteur à ultrasons KY-050 : il est capable de mesurer la distance avec un objet par retour d'ultrasons.
- Et la Module led RGB 5mm KY-016 : elle peut s'allumer.

Pour le capteur à ultrason :

Le capteur d'ultrasons KY-050 va de fait nous permettre de mesurer la distance à laquelle se trouve la main du joueur.

Après avoir effectué les branchements nécessaires (cf. Partie B), nous avons ensuite relié notre code Arduino avec notre jeu pour que celui-ci soit capable de lire les données que le capteur renvoie, afin de prendre une décision sur les déplacements de la tortue : lorsque la distance avec l'objet est inférieur à 70 cm, alors la tortue monte, sinon elle descend.

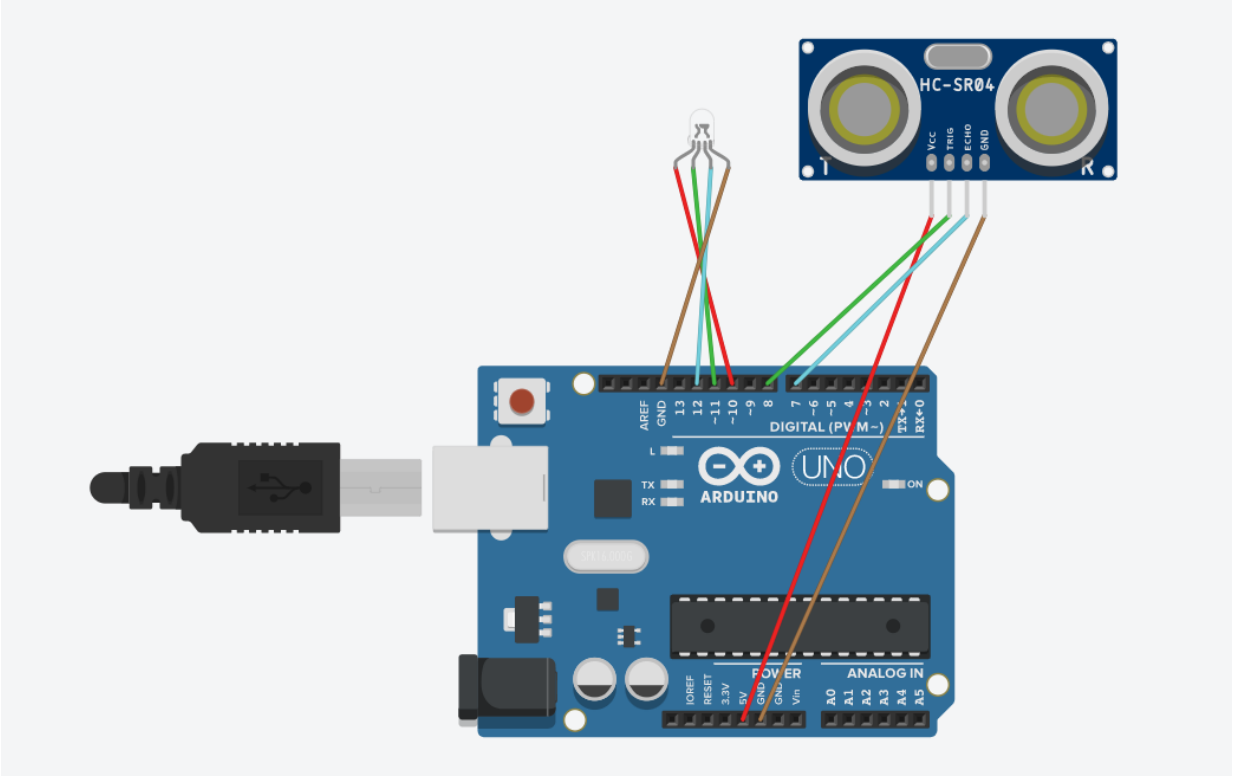
Pour la led :

La led doit s'allumer en cas de collision du joueur (tortue) avec un bonus (méduse).

Après avoir effectué les branchements nécessaires (cf. Partie B), nous avons ensuite relié notre jeu avec le code Arduino pour que celui-ci soit capable de lire les données que le jeu renvoie : lorsqu'une collision avec un bonus se produit, le jeu envois un signal à l'Arduino (un 1) qui se traduit par l'activation de la led pendant

3000ms. Le reste du temps, l'Arduino ne reçoit pas de signal (0) et la led ne s'active donc pas.

B) Schéma



Pour le capteur à ultrason :

Le capteur a ultrason est équipé de 4 broches :

- Vcc (+V)
- Trig (Déclencheur)
- Echo
- GND

De la même façon que le schéma ci-dessus, nous avons relié la broche Vcc au Pin 5V de l'Arduino, la broche Trig au Pin 8, la broche Echo au Pin 7 et enfin la broche GND au Pin GND (Masse) de la carte Arduino.

Pour la led :

La led est équipée de 4 broches :

- R (led Rouge)
- G (led Verte)
- B (led Bleue)
- GND

De la même façon que le schéma ci-dessus, nous avons relié la broche R au Pin 10 de l'Arduino, la broche G au Pin 11, la broche b au Pin 12 et enfin la broche GND au Pin GND (Masse) de la carte Arduino.

Nous avons réalisé ces branchement grâce à la documentation sur le site SensorKit :

- https://sensorkit.joy-it.net/fr/sensors/ky-016
- https://sensorkit.joy-it.net/fr/sensors/ky-050

Nous avons ensuite simplement relié la carte Arduino à notre ordinateur à l'aide du câble fourni.

Manuel d'utilisation en anglais

A) Selma

What is Crush Survivor?

Our game is called **Crush survivor**. The goal is simple: the player is represented by our turtle, Crush (in reference to the turtle from the film Finding Nemo). Over the years, Crush has realized that his ocean is increasingly polluted by plastic waste, that his food is becoming increasingly scarce, and tries somehow to survive in these conditions.

Every day Crush has to deal with plastic bags that come to him randomly and surprise him. For each plastic avoided, Crush earns 1 point. If Crush eats a plastic bag by mistake, he will be asphyxiated and dies. But if he manages to eat jellyfish (his favorite meal), it boosts him and he will earn more points. It's up to you to help him!

But how to install the game?

There are 2 possible versions of the game depending on the game mode you want to test: **the classic graphical version** (which can be played with a computer), and **the tangible version** (which requires specific hardware).

For the classic version, get a computer. Download the jeu-graphique.zip file. This folder must contain 7 files (sound.py, joueur.py, obstacle.py, bonus.py, bonus_event.py, jeu.py and main.py). It should also contain an assets folder which contains all the game images and a __pycache__ folder which simply allows faster execution of the game script.

Before launching the game, make sure you have already installed Python on your computer, if not, go to the official website and follow the instructions:

For Windows: https://www.python.org/downloads/windows/

For Mac: https://www.python.org/downloads/mac-osx/

Also make sure you have installed the Python library pygame beforehand so that the game can work properly. If not, just go to your terminal and run the following command line: pip install pygame

When this is done, you can open the main.py file via an IDE like Visual Studio Code, and launch it by using the play button at the top right of the window. Congrats! You can now play.

For the tangible version, get a computer. Download the jeu-tangible.zip folder. This folder must contain 8 files (arduino.py, sound.py, joueur.py, obstacle.py, bonus.py,

bonus_event.py, jeu.py and main.py). It should also contain an assets folder which contains all the game images and a __pycache__ folder which simply allows faster execution of the game script.

As for the classic version, make sure that you have already installed Python on your computer, as well as the Python library pygame so that the game can run correctly. For this version of the game, you will also need to have installed the pyserial library by executing the following command line in your terminal: pip install pyserial.

Then download the ultrason.ino file and open it via the Arduino IDE (which you can install using this link: https://www.arduino.cc/en/software).

When this is done, take an Arduino card and make the connections (in the same way as on the diagram in the previous part) with the KY-050 sensors (which is an ultrasonic sensor which can measuring the distance between itself and an object) and KY-016 (which is a led), then connect this with your computer using the supplied cable.

Now that your connections are made, open the ultrason.ino file, go to Tools -> Board and select Arduino UNO. Then go to Tools -> Port and select the port corresponding to your connection with the Arduino setup. You can then press the code verification button which is at the top left of the window and then press the arrow which is right next to it. Congrats! Your code is running! You can easily check that everything is working by pressing the serial monitor at the top right of the window. When you put your hand in front of the ultrasonic sensor, it sends you the distance at which you put your hand. Don't forget to close the serial monitor afterwards.

Don't be discouraged, you're almost at the end! You can now open the main.py file (which you downloaded earlier) via an IDE like Visual Studio Code, and launch it using the play button located at the top right of the window. Congrats! You can now play.

And how to play?

To play on the classic version, the rules are very simple: you will only need your mouse and the directional arrows on your keyboard.

Press Play with your mouse to start the game. Now all you have to do is press the up arrow if you want your turtle to go up, or press the down button if you want it to go down. If you don't want it to move then don't press any buttons. Very simple right? So it's up to you to avoid plastic bags!

To play on the tangible version, you will only need your mouse and your hand!

Press Play with your mouse to launch the game. Now all you have to do is approach your hand to make the turtle go up, or move it away to make it go down. Warning ! In this version of the game. The turtle is always moving, you cannot stop it. Bonus: when you manage to catch a jellyfish, the green LED will light up to reward you. So it's up to you!

B) Thomas

Crush Survivor

! Crush Survivor is now available for free !

Introduction : Where does this game come from ?

Founded in 2023, Crush Survivor aims to revolutionize how we think about turtles in the sea. Aiming to raise awareness on the effects of excessive plastic consumption, Crush Survivor promises to be the best game you have ever played. Selma El Babarti, CEO of Crush Survivor, imagined a totally new game that will change for a very long time the way we see and think about having fun.

Not at all plagiarizing Flappy Bird or T.Rex Chrome, Crush Survivor will offer the user a world to explore, to fight and to level-up your abilities at dodging plastic.

In this amazing universe, you, the hero, will control a turtle in the sea that has to swim to avoid the plastic Coca-Cola released into the ocean.

I : But how can we get this amazing free game ?

Easy, start by downloading Python, if you don't know how, no problem, Youtube is here to save you. Just type in : how to download Python 👋. After you downloaded Python, just download the files we provided for you on our website, and run the main file and you're good to go !

Now you can enjoy the pleasure of the best game you will ever play in your life.

II But you didn't tell me how to play !

Let's start at the beginning. Press play. After you press play, the game will start and it will be up to you to survive.

Plastic will start attacking you, and you need to dodge it in order to avoid death.

But if you want a high score… You must also try and eat what turtles love most, jellyfish.

Each jellyfish will give you an extra 20 points to your score. And your goal is to survive as long as possible in the dangerous oceans of 2023.

Of course, the game is realistic and you will be faced with more plastic than jelly.

III Help me to survive the Coca Cola attack :

You’re the hero, it’s up to you to save Crush. But I’ll help you get started. You’ll need to use your device to dodge the plastic. To do so, you can press your down key to tell Crush to go down, or press your up key if you want him to rise to the surface

That means you’ll have to time your flaps carefully if you want to avoid getting hit. So stay focused and react quickly.

IV How do I know how far I’ve gone ?

Easy, on the top right hand corner of your screen you’ll see your current score. It’s determined by how long you stay alive and how many jellyfish you were able to eat. But don’t forget, the game will end when you're unable to dodge the plastic.

V Can you summarize all of what you said ?

In summary, the most important thing in Crush Survivor is your timing. You have to react quickly in order to dodge the plastic and eat the jellyfish. To do so, you’ll be able to use your up key to go up and your down key to go down.

Don’t forget to keep your score in mind, it will be on the top right hand corner. And if you want to play, just click on our link on our website, but don’t forget to check if you have Python.

C) **Jasmine**

Crush game instructions:

Crush is a platform game where the player must control a small sea turtle named Crush, so that it can catch jellyfish for food. But you have to be careful with the plastic bags because if Crush has the misfortune to eat one, she will die and the player will lose the game.

The more jellyfish Crush eats, the more points the player gets. Each jellyfish is worth 30 points. The goal is to eat the most jellyfish without dying. So try to get as far as you can! There is no indigestion!

Press "start" to start the game. To control the jellyfish, just use the up and down arrows on your keyboard.

The turtle moves up and down while the jellyfish and the bags move from right to left.

Using the Arduino sensor, you have to control the turtle by moving your hand more or less away from the sensor. The closer your hand is to the sensor, the higher your turtle will go and vice versa. It's not that easy so good luck!

Bilan personnel en français

A) Selma

Commentaire d'un algorithme que j'ai codé :

J'ai choisi de commenter la **classe Bonus** que j'ai codé pour ce projet. C'était pour moi la première fois que je faisais un projet en python et cela m'a permis notamment de comprendre comment faire de la programmation orientée objet.

Pour la classe bonus du jeu j'ai d'abord codé mes constructeurs de la même façon que nous avions codé les constructeurs du joueur ou des obstacles (mon groupe et moi-même, nous sommes basés sur notre modélisation UML pour construire nos classes).

J'y ai d'abord mis des constructeurs de base, comme la **vitesse** à laquelle peuvent avancer les méduses, je leur ai également attribué une **position aléatoire en y** grâce à la fonction random.randint(), ainsi qu'un **nombre de points** (car en cas de collision avec le joueur, la méduse apporte des points supplémentaires au score). J'ai ensuite ajouté mes **arguments graphiques** en intégrant simplement l'image de notre méduse grâce à la bibliothèque **pygame** (qui est une librairie très connue permettant de fabriquer des jeux 2D).

J'ai ensuite attribué des **méthodes** à la classe Bonus(), notamment pour que les méduses puissent **avancer** (en cas de non-collision et à la vitesse initialisé plus haut) et être **retirées** (en cas de collision avec le joueur pour ensuite ajouter des points, ou en cas de collision avec le bord qui signifie que le joueur n'a pas réussi à les attraper).

Après avoir construit cela, j'étais tout de même face à un problème : il fallait faire en sorte que **mes méduses n'apparaissent pas à la même fréquence que mes obstacles** mais seulement une fois de temps en temps. J'ai alors eu l'idée de construire une jauge qui représenterait la **barre d'événement** méduse : quand la jauge se remplit, une méduse apparait. Pour cela, j'ai ajouté un argument évènement à ma classe Bonus() et j'ai créé une seconde classe **Bonus_event()** à l'intérieure de laquelle j'ai attribué les constructeurs **pourcentage** (qui correspond au pourcentage initial de la jauge, à savoir 0%) et une **vitesse** (qui correspond à la vitesse de remplissage de la jauge).

J’ai attribué à la classe Bonus_event() des **méthodes** qui permettent de faire en sorte de remplir cette jauge au fur et à mesure de la partie et de **faire apparaître une méduse** lorsque celle-ci est pleine. Pour faire apparaître une méduse, j’ai simplement ajouté un objet Bonus() au **groupe d’objets** que j’avais initialisé dans la classe Bonus_event().

J’ai enfin terminé par la méthode **update** qui correspond simplement à l’apparence de la jauge et qui permet de mettre à jour en continue la barre d'événement.

Retour d’expérience personnel :

Pour réaliser ce projet, j’ai beaucoup appris sur des méthodes de travail que je n’avais jamais expérimenté avant, je me suis notamment appuyé sur la documentation et j’ai également regardé ce qui se faisait : comment ont fait les autres personnes qui ont déjà codé un jeu en utilisant pygame, quelles méthodes ils utilisaient, pourquoi plutôt celles-ci que d’autres. Cela m’a aidé à mieux comprendre comment fonctionnaient certaines fonctions et pourquoi les choisir.

Ce projet m’a également beaucoup appris d’un point de vue technique : j’ai pu utiliser diverses ressources que je n’avais encore jamais utilisées avant ce projet, je pense notamment à Stack overflow, à la documentation pygame ou encore à GitHub.

En travaillant sur ce projet, j'ai découvert de nouvelles fonctionnalités de Pygame qui ont amélioré considérablement le rendu visuel et l'expérience utilisateur. J'ai notamment appris à implémenter des images pour donner plus de réalisme à l'environnement de jeu, ainsi que des effets sonores pour renforcer l'immersion.

J'ai également appris à implémenter un système de score pour rendre le jeu plus vivant et que les joueurs puissent se mesurer à leurs amis.

En somme, ce projet a été un véritable défi pour moi, mais j'ai appris énormément sur les différentes techniques de développement de jeux et sur l'utilisation des outils professionnels. Je suis très fier de ce que j'ai accompli et je suis impatient de continuer à développer mes compétences pour créer de nouveaux projets encore plus ambitieux à l'avenir.

Concernant l’organisation de notre groupe, nous avons certes échangé entre nous très souvent pour convenir de comment nous allions faire et pour nous entraider mais le code se faisait finalement souvent seul, chacun de notre côté et nous nous réunissions pour l’uniformiser, gérer certaines incohérences et cela n’était pas pratique. Je pense qu’à l’avenir apprendre à utiliser un outil comme GitHub pourrait être bénéfique et beaucoup plus efficace pour ce genre de projet, à la fois pour gérer les conflits entre nos versions mais aussi pour collaborer au mieux en ne perdant pas le fil et le suivi.

Quant à notre code, mon équipe et moi-même avons fait au mieux pour l’optimiser mais nous ne savons pas encore à quel point il pourrait l’être davantage (surtout pour la partie tangible qui perd beaucoup en fluidité par rapport à la version graphique). Nous sommes cependant globalement satisfaits par notre travail, car nous avons réussi à la fois à créer l’idée que nous avions en tête et à la rendre fonctionnelle.

Je pense qu’à l’avenir, lorsque j’aurai encore acquis en compétences en programmation, je me repencherai sur ce projet et j'essaierai d’ajouter de nouvelles fonctionnalités (comme la possibilité de choisir un personnage différent par exemple, ou encore de faire en sorte que mes obstacles arrivent de plus en plus vite au fur et à mesure que le joueur gagne en points). Ce fut pour moi un projet très intéressant, dans lequel j’ai beaucoup appris et qui reste encore une porte ouverte à beaucoup d’améliorations possibles à l’avenir.

En conclusion, je dirai que grâce à ce projet, j'ai pu développer quelques bases de compétences en développement de jeux et j'ai pu mettre en pratique les concepts appris au cours de mes recherches pour mettre en œuvre ce projet.

B) Thomas

Bilan personnel :

Pour ce bilan personnel sur une première expérience en programmation, nous nous centrerons sur quatre points. Dans un premier temps, nous allons développer comment s'est déroulée la phase de construction des diagrammes, c'est-à-dire, la partie qui c'est déployée avant que nous ne commencions à coder. Deuxièmement, nous porterons notre attention sur l'usage de Pygame. Pour ensuite dans un troisième temps commenter un code que j'ai rédigé. Enfin, dans un dernier temps, nous reviendrons sur cette première expérience en programmation en adoptant une posture plus réflexive.

Avant le code, les diagrammes :

Avant de commencer à coder, il nous a été conseillé de construire des diagrammes pour avoir une meilleure vision du travail à faire et comment il allait s'organiser.

Le premier diagramme que nous avons réalisé fut le diagramme d'activité, à la fois, parce que c'était le plus simple, mais aussi, car nous trouvions qu'il offrait une bonne base pour nos prochains diagrammes. Pour le faire, nous avons chacun de façon autonome réalisé notre diagramme d'activité pour ensuite les comparer. Après les avoir réunis, nous les avons fusionnés pour produire un diagramme collectif qui a ensuite évolué avec notre projet. Par exemple, dans la première itération du diagramme d'activité, il n'y avait pas de "score".

Ce diagramme ne fut pas le plus indispensable et le plus important de la suite du projet. En effet, j'ai eu le sentiment qu'il a eu pour rôle de surtout nous lancer dans une dynamique de production de diagrammes. A savoir, le diagramme de classe fut le plus important pour nous au cours de ce projet et c'est également celui qui a le plus changé entre le début et la fin de nos itérations. En effet, ce diagramme nous a beaucoup aidé pour poser les bases et les fondations de notre code notamment à travers son squelette. De plus, ce diagramme a aussi été modifié en fonction de comment notre code avançait pour ne pas que les deux parties soit dans une posture asymétrique.

Ainsi, notre diagramme de classe définitif est composé de 5 classes : joueur, jeu, bonus, bonus event et obstacle. La classe joueur qui est en somme la tortue avec une position en X et en Y. Ceci étant, cette classe pourrait très bien être utilisée pour représenter autre chose qu'une tortue. La classe obstacle qui va prendre la forme des plastiques avec une position en X et en Y (qui sera fixe), une vitesse et une valeur dégât. La classe Bonus, représentée par les méduses avec une valeur en points et une vitesse, comme pour le plastique leur valeur en Y est fixe. La classe Bonus Event, qui permet le démarrage des méduses (c'est pourquoi les classes

bonus et bonus event sont directement liées entre elles). Enfin, la classe jeu qui est le point de rencontre des autres classes et permet de les faire fonctionner ensemble.

Commentaire d'un code que j'ai rédigé : La classe joueur

Au moment de diviser le code, j'ai décidé de me centrer sur le classe joueur car elle me semblait comme étant une pièce importante du puzzle. De plus, comme c'est la partie du code que j'ai réalisé seul, elle me semblait comme la plus adaptée pour cette partie.

```
import pygame

#THOMAS

class Joueur(pygame.sprite.Sprite): # Nous commençons par
construire une classe joueur avec les paramètres
    #on définit notre classe joueur avec ces méthodes

    # Définition du constructeur (permet de construire la base
de la classe, c'est-à-dire, l'instancier)

    def __init__(self, jeu): #On demande l'instance de jeu. On
définit les variables de l'objet instancié
        #'__init__' est une méthode qui permet de créer un
nouvel objet. Le constructeur va prendre 'jeu' en argument.
        super().__init__() #appel au constructeur de la classe
supérieur
        # Définition des éléments graphiques de la classe.
        self.image = pygame.image.load("assets/tortue.png") #
on load l'image que nous avons sélectionné.
        self.rect = self.image.get_rect() #rect est une forme
à 4 côtés dont on souhaite récupérer les coordonnées et
```

```
déplacer. C'est-à-dire, que le joueur va prendre une forme
rectangulaire.
        self.all_players = pygame.sprite.Group() # all-player
est un attribut qui va permettre de ranger plusieurs instances
de la classe.
        #constructeurs de base
        self.jeu = jeu
        self.vie = 1 # définition du nombre de vie
        self.vie_max = 1
        self.vitesse = 7 # définition de la vitesse
        self.rect.y = 325 #valeur en partant du haut de
l'écran

    # Notre classe à donc 3 méthodes

    #le joueur peut mourir
    def meurt(self, degat):
        self.vie -= degat
        return True  # meurt va donc réduire la valeur de vie
par la valeur 'degat'

    #le joueur peut monter : méthode “monter” qui va faire
monter l'image du joueur
    def monter(self):
        #si le joueur n'est pas en collision on peut se
déplacer
        if not self.jeu.check_collision(self,
self.jeu.all_objets): #on compare que le joueur entre en
collision avec le groupe d'obstacles
            self.rect.y = self.rect.y - self.vitesse # s'il
n'y a pas de collision avec un obstacle, l'attribut rect
baisse de la valeur de la l'attribut vitesse


    #Le joueur peut aussi descendre : son fonctionnement est
proche de celui de la méthode "monter". Bien que, il va faire
descendre l'image en augmentant la valeur de "rect.y" par la
valeur de vitesse
```

```
    def descendre(self):
        #si le joueur n'est pas en collision on peut se
déplacer
        if not self.jeu.check_collision(self,
self.jeu.all_objets):
            self.rect.y = self.rect.y + self.vitesse
```

Retour sur cette expérience en programmation:

Nous pouvons dans un dernier temps avoir une posture plus réflexive sur notre première expérience de programmation. En effet, avant cette SAE, nous n'avions jamais touché à la programmation. Comme Selma, j'ai réalisé une licence de sociologie et Jasmine était en design, de fait, nous étions un peu inquiets à l'idée de réaliser ce projet. De surcroît, nous n'avons pas choisi le jeu le plus simple à réaliser, à savoir, l'intégration d'un aspect dynamique était peut-être le point qui me rendait le plus soucieux.
Désormais, notre jeu est terminé et je suis assez fière de notre travail. C'est un peu comique, mais quand le jeu a fonctionné, je croyais que j'allais devenir un as de la programmation. Je me suis rapidement rendu à l'évidence que nous n'avions qu'effleurer le pic de l'iceberg.
Bref, pour ne pas m'éterniser, je terminais en disant que je vois ce projet avant tout comme une expérience ou une découverte de ce qu'est une infime partie de la programmation avec python. J'ai conscience du fait que ce que nous avons réalisé n'est en aucun cas impressionnant pour les adeptes de l'information, mais je reste tout de même fière de ce que nous avons produit avec l'expérience que nous possédons.
En résumé, pour le cours, il faudrait peut-être allouer plus de temps pour intégrer un capteur au jeu (une séance supplémentaire), sinon j'ai trouvé que les cours étaient très enrichissants. Néanmoins, peut-être faudrait-il supprimer la semaine de remise à niveau sur python pour les étudiants déjà formés, pour plus l'orienter autour de ceux qui n'y connaissent rien.

C) Jasmine

La création du jeu s'est faite de façon plutôt fluide. J'ai proposé de s'inspirer du jeu rétro "T-Rex Game" qui est un jeu de plateforme jouable hors connexion sur le navigateur Google Chrome. Les autres étant d'accord et l'idée validée, nous avons chacun fait des recherches pour coder le jeu.

J'ai essayé de coder les bases du jeu en autonomie. Mais très vite après avoir fait les premiers diagrammes d'activité et de séquence, nous nous sommes répartis les tâches car il était très compliqué de coder l'ensemble, seul. Nous avons décidé d'avancer ensemble sur chaque fichier composant le jeu. J'étais chargée de certaines parties du jeu, notamment la création de la fenêtre du jeu, de l'interface graphique, de l'apparition des obstacles et de la classe joueur dans le fichier main. J'ai aussi créé les éléments graphiques sur Illustrator.

Avec mon équipe nous avons décidé de travailler avec pygame car il y avait beaucoup plus de ressources et d'aide sur internet pour réaliser notre jeu.

Voici une partie du jeu que j'ai programmé. Il s'agit essentiellement de la création de la fenêtre de jeu et de sa page de présentation qui se trouve dans le fichier "main" du jeu.

Tout d'abord, je crée une fonction "Clock" qui permet de compter le temps et de pouvoir créer des FPS (frame per seconde) et donc de contrôler le nombre d'images par seconde. Ainsi, le jeu sera fluide visuellement et la comptabilisation des points sera plus simple.

```
# clock utilisé pour les FPS
clock = pygame.time.Clock()
FPS = 60 # en majuscule car variable constante (variable qui change
pas)
```

J'utilise cette fonction pour initialiser tous les modules importés afin de ne pas avoir à le faire manuellement.

```
pygame.init()
```

Puis, j'ai créé la fenêtre du jeu avec la fonction *pygame.display.set_caption("titre")*. Avec la fonction *pygame.display.set_icon(pygame.image.load("icon.png"))* pour attribuer une icône au jeu. J'ai créé ensuite une variable "screen" pour créer un écran d'une certaine largeur.

Pour cela, j'utilise la fonction *pygame.display.set_mode((x,y))*. Puis, j'importe une image de fond en png pour l'intégrer à l'écran.
Avec la fonction *pygame.transform.scale(background, (x,y))* je modifie l'échelle de mon fond pour qu'il corresponde exactement à mon écran.

```
# création de la fenêtre de notre jeu.
pygame.display.set_caption("Crush Survivor") # titre de la fenêtre
pygame.display.set_icon(pygame.image.load("/Users/Mana-/Documents/MCN1/
Tech_1/SAE_Tech_1/jeu-graphique/jeu-graphique/assets/icon.png"))
screen = pygame.display.set_mode((600,400))  # Largeur et hauteur -->
envoie une surface qu'on récupère
background =
pygame.image.load("/Users/Mana-/Documents/MCN1/Tech_1/SAE_Tech_1/jeu-gr
aphique/jeu-graphique/assets/bg.jpg")
background = pygame.transform.scale(background, (600,400))
```

Pour importer le bouton start qui permet de lancer le jeu, je lui accorde un nom et l'importe avec la fonction *pygame.image.load.* Pour pouvoir modifier son échelle dans la fenêtre, j'utilise la fonction *pygame.transform.scale(objet, (x,y))*.

Ensuite je crée une zone rectangulaire cliquable nommée *button_rect* que je lie au *button* grâce à la fonction *button.get_rect()*. Enfin, je modifie la position du bouton dans la fenêtre, en attribuant une valeur à sa position x (*button_rect.x*) et à sa position en y (*button_rect.y*). La fonction *screen.get_width()* permet de connaître la largeur exacte de la fenêtre de jeu.

```
button =
pygame.image.load("/Users/Mana-/Documents/MCN1/Tech_1/SAE_Tech_1/jeu-gr
aphique/jeu-graphique/assets/button.png")
button = pygame.transform.scale(button, (100,50))
button_rect = button.get_rect()
button_rect.x = screen.get_width() / 2.4
button_rect.y = 300
```

Dans l'ensemble, j'ai apprécié ce projet. Notamment la partie conception où nous devions trouver l'idée d'un jeu, créer une charte graphique, une histoire. J'ai bien

aimé la partie Arduino même si intégrer le code de l’Arduino au code du jeu fût très complexe.