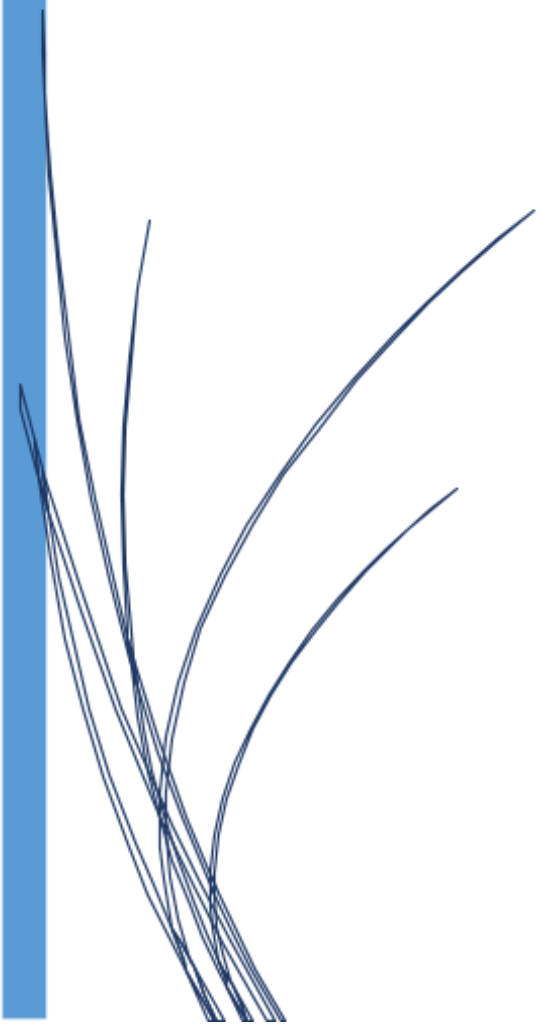
Remis le 07/04/2023

# ICreate :
# Rapport de fin de projet



**Auteur :**
Jamil Hajjar,
Marc Barbier,
Adrien Germain, élèves en
INFO 4 à Polytech Nantes

# Sommaire

# 1. Introduction

A l'occasion de l'exposition temporaire “Océan" du musée d'histoire naturelle de Nantes, nous avons eu pour mission de créer un projet intéractif autour du thème "Un monde à explorer". Nous étions 3 élèves d'ingénieur en informatique (Adrien Germain, Marc Barbier et Jamil Hajjar) et 3 élèves d'école de design (Nathanael Cerise, Maël Gajcanin et Matteo Merand) . Le but était de créer un projet interactif, voué à prendre la forme d'un pupitre exposé dans un musée. En ce qui nous cerne, nous avons eu en premier lieu une présentation de l'exposition "Océan", puis une rencontre avec nos futurs collègues designer pour enfin avoir 4 jours étalés sur 2 semaines pour réaliser notre partie du projet. L'équipe de designers eux, ont travaillé avant nous puis après nous sur ce même projet.

# 2. Présentation du projet

Notre projet est inspiré du Polar Pod qui est lui-même un projet de navire océanographique conçu pour l'étude de l'océan Austral. Nous avons alors imaginé un dispositif autonome et modulable permettant l'extraction de données sous-marines. Nommé Semeo, il a la forme d'une méduse et dispose de plusieurs emplacements destinés à des capteurs. Le but est de rendre accessible les données marines difficiles d'accès tout en pouvant adapter le dispositif aux types de données à extraire.

Cependant, dans le contexte de l'exposition temporaire et avec la contrainte de l'interactivité, nous avons travaillé avec des élèves de l'école de design de Nantes pour imaginer et créer un jeux-vidéos en 3D autour de notre projet et tournant sur Unreal Engine 5. Eux se sont occupés d'imaginer le projet, les scénarios d'usages de Semeo, ainsi que le design général du dispositif. Nous nous sommes occupés du côté technique de la création du jeu vidéo tout en ayant, au préalable, discuté de leurs idées et envies et de leur faisabilités compte tenu de nos bagages techniques et du temps imparti.

Le joueur contrôle Semeo. Différentes missions lui sont confiées afin qu’il puisse extraire les données voulues. Le joueur peut se déplacer avec un Joystick et de haut en bas grâce à un slider. Il possède aussi un bouton d’action lui permettant entre autres de lancer une analyse lorsqu'il se trouve dans la zone adaptée et qu’il est équipé du capteur correspondant, ce qui se fait en tournant des cylindres (chaque moitié de cylindre correspond à 1 capteur).

# 3. Compétences acquises / mises en oeuvre

## 3.1. Capacité à analyser et comprendre un problème

Quand nous sommes intervenus sur le projet, l'équipe design avait déjà imaginée et réfléchie à ce dernier. C'est eux qui ont promu le jeu vidéo comme média d'interaction avec les visiteurs du musée. Alors, notre rôle a été dans un premier temps de comprendre ce qu'ils nous proposaient et de discuter des différentes idées émises afin de sortir de cette discussion un projet compris et agréé de tous. La réelle difficulté fut de nous comprendre précisément. En effet bien qu'il y eut une très bonne entente au sein du groupe, nous n'avions pas les mêmes bagages techniques et donc quand nos conversations se précisait, il était important d'avoir conscience de cette différence afin de mieux se comprendre. En outre, lorsque nous, futurs ingénieurs, parlions techniques, il fallait le faire clairement sans pour autant s'abstenir des détails. A l'inverse, quand c'est eux qui nous parlaient, nous avons posé les questions nécessaires à l'éclaircissement de certains points qui nous semblaient plus ou moins flous. Ainsi comprendre et faire comprendre étaient les premières compétences utilisées et renforcées lors de ce projet.

## 3.2. Capacité à concevoir et formaliser une solution

Là, pour nous, il s'agissait de savoir ce qui était possible de réaliser à partir des idées émises. Lors de nos discussions, quand nous parlions de ce qu'allait être notre solution en détail, nous jouions 2 rôles. Le premier, comme dit précédemment, consistait à participer à l'échange d'idées. Le second était plus technique car quand une idée plaisait à tous, nous nous devions d'estimer sa faisabilité. En effet, il fallait prendre en compte le temps impartis et nos connaissances techniques. Ainsi, quand nous parlions du jeu en lui-même et sachant que personne n'avait de réelle expériences concernant un projet de la sorte, nous voyions les ambitions un peu à la baisse car il fallait que la solution soit fonctionnelle à la fin de notre participation. Ainsi, dès le début, nous avons jalonné ce travail afin qu'un premier jalon fonctionnelle soit atteignable et renforçable avec d'autres jalons correspondants à des fonctionnalités à ajouter. A l'inverse, quand nous parlions de solution pour effectuer la liaison entre notre joy-stick, slider et bouton à un ordinateur, nous savions que l'un de nos membres, Marc Barbier s'y connaissait alors nous avons pu voir à la hausse les ambitions concernant cela.

## 3.3. Capacité à trouver l'information pertinente et à l'exploiter

L'une des choses à faire avant de se lancer dans un projet est de savoir quelles ressources sont à notre disposition. Évidemment, internet et ses nombreux tutoriels, mais aussi l'atelier numérique de l'école de design ainsi que les différents intervenants/encadrant du projet. Ainsi, nous avons essayé d'exploiter au mieux ses ressources pour répondre à nos questions le plus rapidement et précisément possible. Par exemple, quand il fallait choisir le moteur du jeu, nous savions que 2 des principaux existant étaient Unreal Engine ou Unity. Alors, nous en avons discuté entre nous, puis avec les intervenants ainsi qu'à nos connaissances qui étaient mieux informées sur le sujet que nous.

## 3.4. Capacité à mettre en oeuvre et évaluer une solution

La solution choisie fut de faire le jeu vidéo à l'aide d'Unreal Engine 5. Pour mettre en œuvre ceci, quelques tutoriels de prises en main nous ont été utiles. Il faut savoir que coder sur UR 5 se fait via BluePrint, un système de programmation appelé “système de script visuel” ou en C++ pour les plus aguerris du moteur. Nous nous sommes concentré sur le BluePrint qui a fait largement l'affaire car nous étions débutant avec UR 5. Un exemple de mise en œuvre de fonctionnalité est la flèche qui tourne dynamiquement, indiquant la direction que le personnage doit prendre pour arriver au point voulu. Cette fonctionnalité a d'abord été implémentée par Adrien Germain à l'aide d'un tutoriel YouTube qui réalisait pas à pas précisément cela. Ensuite, nous l'avons greffé au code principal avec Jamil Hajjar. En effet, le plus gros bémol de notre solution fut dans la collaboration de code. Nous n'avons pas trouvé de moyen viable pour coder tous ensemble sur le même projet alors, Jamil avait la version la plus aboutie et Marc et Adrien réalisait des fonctionnalités indépendantes de leur côté.

## 3.5. Capacité à collaborer avec des personnes hors de mon champs disciplinaire

En ce qui concerne la collaboration de pré-développement, nous en avons déjà parlé, il s'agissait de discussions entre designers et ingénieurs. Pendant le développement de la solution, les designers avaient aussi du travail de leur côté mais nous étions dans les mêmes locaux donc la communication entre les deux groupes était omniprésente. De plus, nous avions mis en place un serveur discord dédié au projet ce qui favorise cette communication tout en y ajoutant la possibilité d'échange de fichiers, liens, … Pendant le développement, régulièrement et à chaque fonctionnalité, nous demandions l'avis de l'équipe de designers pour ajuster des points précis (mise en formule de l'HUD, images à utiliser dans le jeu, …). Enfin, sur la fin du projet, lors de la dernière journée où il ne fallait plus que calibrer les dernier paramètres pour rendre notre jeu le mieux possible, Adrien s'est retrouvé à aider les designers dans leur tâches car 2 personnes étaient amplement suffisant pour finir le travail sur le jeu.))

## 3.6. Capacité à organiser et planifier le travail

Très vite, Jamil Hajjar s'est imposé en tant que leader du groupe d'ingénieur. Il avait la version la plus aboutie du code, et Marc et Adrien travaillaient alors à côté sur les fonctionnalités à ajouter. Au début, Marc s'est occupé de la partie électronique du projet pour faire fonctionner les controllers. Dans le même temps, Jamil mettait en place la base du projet tandis qu'Adrien prenait en main le moteur. Enfin, une fois l'électronique fini, le projet mis en place et la prise en main effectuée, Jamil corrigeait les bugs et détails globaux tandis que Marc et Adrien travaillaient indépendamment sur différentes fonctionnalités avant de les fusionner au projet de base.

# 4. Bilan et retour d'expérience

Ce fut une expérience riche et originale. Il s'agissait de l'une des rares fois où nous faisions un projet avec des personnes hors de notre champ disciplinaire. Il faut dire qu'en tant qu'étudiant informaticien, on côtoie quotidiennement des personnes de cette spécialité technique, que ce soit des professeurs, des doctorants ou des élèves. Dans ce contexte, on se retrouve vite enfermé dans l'informatique et réaliser un tel projet en collaboration direct avec des designers nous à permis de nous décloisonner du monde informatique en prenant conscience de ce qui semblait évident pour nous et qui ne l'est pas forcément pour autrui, en outre, nous avons pris du recul sur nos connaissances et compétences.
De plus, l'entente au sein du groupe était des plus agréable, tout le monde était investi dans le projet et l'expérience d'un projet qui nous prend toute la journée sur plusieurs jours est agréable comparé aux études ou nous pouvons enchaîner jusqu'à 6 domaines ou projets dans la même journée.

# 5. Guide technique

## 5.1. Prérequis

Logiciels

- Unreal Engine 5.1
- Arduino et bibliothèques nécessaires (voir Annexe 1.5)

Matériel

- 3 VL6180 (capteur de temps de vol)
- 1 multiplexeur I2C
- 1 arduino 32 bits
- 1 joystick
- 1 slider
- 1 bouton
- 1 écran
- 1 ordinateur de jeu (sous linux de préférence, voir 1.5.1)
- Hauts-parleurs

## 5.2. Installation

Télécharger le projet depuis https://github.com/mperreir/ocean.s-23 à l'aide d'un git clone récursif :
git clone --recursive

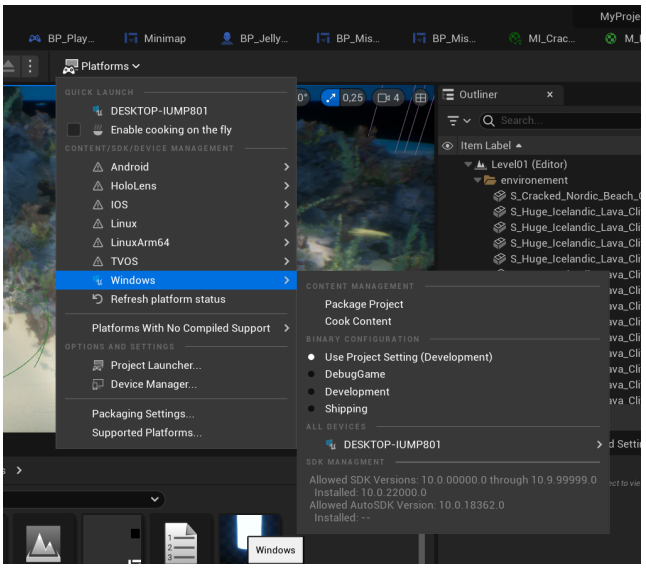
Brancher l'ordinateur en USB à l'arduino. Brancher les hauts-parleurs et l'écran au PC.

Flasher l'arduino avec le code du dossier "Un monde à explorer/Arduino" avec l'IDE Arduino.
La partie câblage électronique est détaillée dans les annexes 1.1 et 1.3.
Nous recommandons de re-calibrer l'arduino en suivant les instructions de l'annexe 1.2.

Ouvrir le fichier uproject du dossier "Un monde à explorer" avec unreal engine.

## 5.3. Lancer le projet

Pour obtenir un exécutable du projet, se rendre dans Platforms > Windows> sélectionner Shipping.
Puis dans la même section cliquer sur “Package Project”.



# Annexes

## 1. Électronique

### 1.1. Schéma de câblage

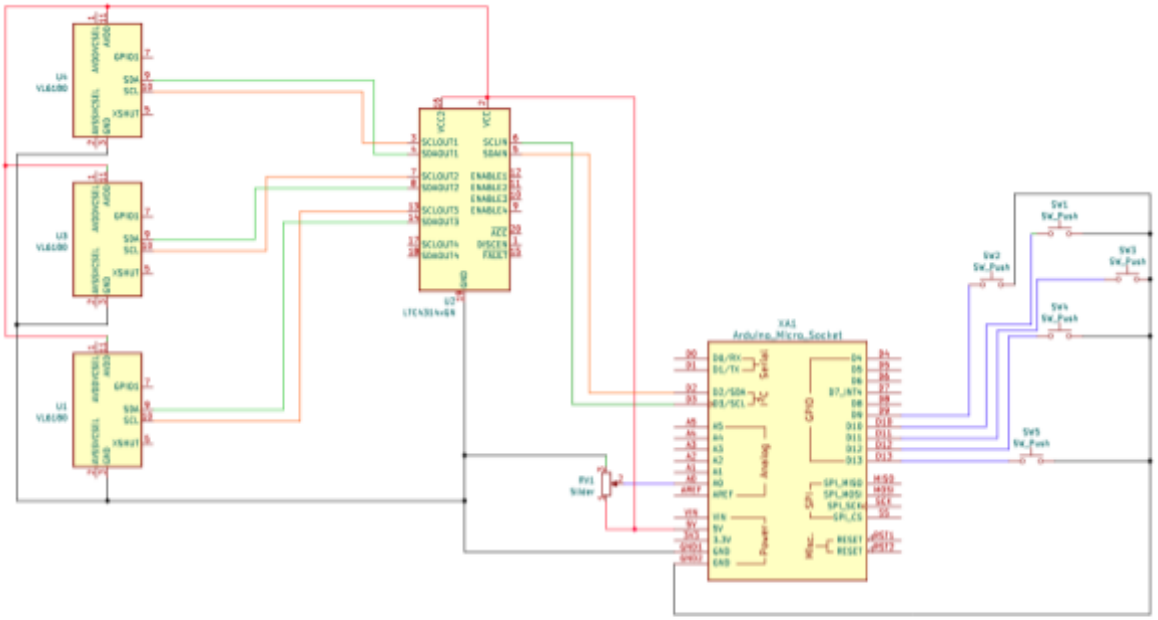


Schéma de câblage en blocs

### 1.2. Calibration

L'usage des capteurs de temps de vol nous impose une phase de calibrage. En effet, ils ne sont pas tous installés à la même distance et ne fonctionnent pas tous de la même façon, par exemple le capteur 0 à un décalage de 15 mm dans sa mesure comparé aux capteurs 1 & 2.

Dans les premières lignes du programme Arduino, il y a cette ligne:

```
#define HID
```

Si elle est mise en commentaire, le programme ne va plus répondre comme une manette, mais comme un simple périphérique série qui affiche dans la console les valeurs mesurées. De cette façon, l'utilisateur voulant calibrer va pouvoir observer les mesures des différents capteurs.

Dans un premier temps l'utilisateur mesure les 3 valeurs avec système en position courte et les 3 valeurs système en position longue puis ajuste dans le programme les constantes correspondantes

```
#define OFFSET_TOP 17 //Mesure ouverte Du haut
#define OFFSET_MID 15 //Mesure ouverte Du milieu
#define OFFSET_BOTTOM 0 //Mesure ouverte Du bas
#define LASER_FLOOR 5 //increase to increate the minimum distance
```

Ces variables vont permettre de mettre les valeurs mesure dans un intervalle qui commence à 0. Maintenant, il faut choisir le seuil à partir duquel on considère que nous sommes dans une position longue, il faut une valeur suffisamment grande pour prendre en compte les mauvaises mesures et suffisamment petite pour que le seuil soit atteint.

## 1.3. Câblage du joystick et du bouton

Le joystick est en réalité quatre boutons qui se partagent une masse. Sur la nappe, il y a cinq fils. Sur l'une des extrémités, il y a le fil de masse qui doit être relié à une broche GND sur l'Arduino. Les autres fils doivent être câblés sur des broches d'entrée numérique (D8 a D11 dans l'idéal).

Si le joystick n'a pas la bonne orientation ou que des actions ne correspondent pas, il faut vérifier le câblage et potentiellement modifier les broches utilisées dans le programme.

```
#define UP_PIN 8
#define DOWN_PIN 9
#define LEFT_PIN 11
#define RIGHT_PIN 10
```

pour ce qui est du bouton, il doit être connecté entre la broche 12 et le GND au besoins la broche peut être changé

```
#define BTN_PIN 12
```

## 1.4. Les différents firmwares Arduino

Nous avons 2 versions du programme Arduino car nous avons eu des problèmes avec la première, mais la seconde n'a pas encore pu être testé. Les différents firmwares sont disponibles dans le dossier Arduino du projet.

### 1.4.1 Dinput

La première est la version DInput qui émule des anciennne manette très simple, l'installation de cette version est très simple, il suffit d'installer la librairie Adafruit_VL6180X ainsi que la librairie https://github.com/MHeironimus/ArduinoJoystickLibrary.

Cette version a le défaut que sur Windows Unreal Engine 5 ne peut pas interagir avec la manette et donc il faut mettre le jeu sous Mac OS (non testé) ou Linux.

Il existe des plugins pour ajouter le support du Dinput sur Unreal Engine 5 mais ils sont payants.

### 1.4.2. Xinput

La seconde version du firmware Arduino est conçue pour simuler une manette d'Xbox 360, la rendant ainsi compatible avec n'importe quel Moteur de jeu sur n'importe quel système d'exploitation.

Son installation en revanche n'est pas aussi simple, car il faut ajouter en plus de la bibliothèque Arduino: https://github.com/dmadison/ArduinoXInput Des cœurs AVR alternatif (pour faire simple, c'est la portion du code qui sert à démarrer l'Arduino) https://github.com/dmadison/ArduinoXInput_AVR et au moment de flasher la carte Arduino, il faut sélectionner le cœur correspondant, dans notre cas, c'est le cœur "Arduino Micro w/ Xinput"

Cette version n'a pas pu être testée, mais devrait être fonctionnelle grâce à la faible quantité de code qui a dû être adaptée.

## 1.5. Les différents choix de design

### 1.5.1 Le HID

Dans un premier temps, nous avons pris la décision d'utiliser un Arduino 32 bit, car ils supportent tous le HID, qui est un protocole nous permettant d'assurer une communication fiable et une mise en place facile du système. En effet, contrairement à la communication série, le HID ne change pas d'identifiant et ne nécessite pas de code supplémentaire pour le faire communiquer avec Unreal Engine 5.

### 1.5.2 Le multiplexer I2C

Pour relier ensemble tous nos capteurs, nous aurions pu changer leurs adresses, mais vu qu'à la fin du projet ICreate tout ce matériel sera utilisé pour d'autres projets, il fallait qu'ils puissent être réutilisés sans avoir besoin d'une longue reconfiguration, nous avons pris la décision d'utiliser un multiplexeur.

### 1.5.3 Temps de vol

Les capteurs temps de vol permettent avec un seul capteur simple et très compact de déterminer sans nécessité définir la position initiale du système.

On aurait pu envisager des capteurs à effet hall, mais il nous aurait fallu beaucoup d'aimants ou plusieurs capteurs pour déterminer avec certitude la position de la méduse. Nous avons aussi envisagé d'utiliser des fourches laser, mais cela revient à faire des temps de vol plus gros et donc plus difficile à installer.