

Wild Code School

# Projet 1

Formation Développeur Web et Web mobile en Python



## Table des matières

Informations générales .....	1
L’histoire.....	2
Compétences mises en œuvre.....	3
Spécifications fonctionnelles .....	4
Spécifications détaillées.....	5
Lieu mystérieux 1 (défi : le nombre mystérieux) .....	6
Lieu mystérieux 2 (défi : le code César) .....	7
Lieu mystérieux 3 (défi : Le multi FizzBuzz) .....	8
Lieu mystérieux 4 (la porte vers le niveau suivant) .....	9
Spécifications techniques.....	10
Améliorations possibles .....	11
Conditions de réussite .....	11
Annexes.....	12

## Informations générales

- Nom : **Autonomie** (arrivée sur l’île et exploration)
- Durée : semaines 1 à 5, rendus étalés tout au long de la semaine 6.
- Résumé : Développer un jeu (console) mettant en scène l'avatar arrivant sur un nouveau territoire (l’île, dont il n'a accès qu'à une petite portion – voir la carte en annexe), il doit y découvrir de nouveaux lieux, une faune, une flore, des ressources et des énigmes à résoudre.
- Objectifs : l'avatar doit pouvoir gagner le jeu afin de passer au niveau (projet) suivant.

## L'histoire

*Cette histoire fait suite à la présentation du jeu de rôle en début de formation.*

Ce matin, tu n'as pas été réveillé par les mouvements du navire (ton dernier souvenir est de t'être endormi sur la confortable couchette de ta cabine à bord de l'Argo), mais par le bruit des vagues, la chaleur du soleil et le champ des oiseaux...

Il semble que tu ne sois plus sur le bateau.

Tu es allongé sur une plage !

Tu te redresses doucement, et, un peu aveuglé par le soleil, tu regardes autour de toi. D'un côté la mer... de l'autre un paysage sauvage, en partie steppe, en partie jungle. Assez loin, au-delà des arbres, tu penses apercevoir des montagnes.

Tu fais un rapide bilan de ta situation : outre les vêtements que tu as sur toi (curieusement tes chaussures ne semblent pas faire partie du lot 😊) tu trouves un sac à dos avec :

- une bouteille vide en verre,
- et un petit couteau multifonctions de bonne facture.

Un peu plus loin, posée sur une pierre, se trouve une mallette contenant :

- ton ordinateur portable (oui, il y a du réseau sur cette île),
- et un système de recharge photovoltaïque.

Enfin, glissée à l'intérieur du couvercle de la mallette se trouve une carte grossière plastifiée (voir en annexe).

Il est clair que tu n'es pas arrivé ici par hasard, ce n'est pas un naufrage, on t'a déposé sur cette plage. Mais entre le moment où tu t'es endormi et ton réveil il y a quelques minutes, c'est le trou noir...

## Compétences mises en œuvre

- Personnelles :
  - Exploration
  - Découverte
  - Adaptation
  - Autonomie
- Organisationnelles et fonctionnelles :
  - Compréhension d'un contexte
  - Organisation d'un projet
  - Communication (possibilité de communication et d'entraide illimitée avec les autres membres du groupe)
  - Travail en solo
- Techniques :
  - Algorithmique
  - Développement séquentiel et procédural
  - Python (tous les principes et syntaxes de base, incluant les variables, les types, les test, les boucles, les opérateurs- mathématiques, de comparaison et logiques-, les interactions de base- print, input-, les fonctions- avec et sans paramètres, return-, les listes (y compris à 2 dimensions)/tuples/dictionnaires, la sérialisation, les fonctions de string, la lecture et l'écriture de fichiers texte et JSON, le parsing de fichier, les modules- os, random, json, ...-, des mots clés- in, continue, break, None, ...-, la gestion des exceptions try/except, ...)
  - Environnement de développement, IDE (incluant linting et debugging)
  - Conceptualisation (diagramme de flux, product backlog de niveau 1)

## Spécifications fonctionnelles

Le jeu doit permettre de :

- Demander le nom du joueur.
- Représenter en mode texte (console) la carte (résolution minimum 50x30, la carte sera statique, c'est-à-dire qu'elle tient entièrement sur l'écran sans défilement) qui t'as a été remise avec tous les éléments indiqués (eau, plaine, arbres, rochers, endroits mystérieux- 4 en tout, etc.).
- Représenter ton avatar sur cette carte.
- À l'avatar de se déplacer sur la carte selon 4 directions en respectant les obstacles.
- Gérer le contenu de son sac à dos (avec une capacité d'objets maximum).
- Gérer une liste d'objets à trouver sur la carte (avec possibilité de les placer aléatoirement en début de partie), de pouvoir les ramasser (et les mettre dans le sac à dos) ou les redéposer au sol.
- Gérer l'utilité de ces objets, certains seront utiles pour certaines actions spécifiques, d'autres juste pour prendre de la place dans le sac à dos (ou peut-être à utiliser pour les étapes/projets suivants).
- Gérer 3 niveaux vitaux (soif, faim et fatigue) de l'avatar sur une échelle de 0 à 100. Si l'un des compteurs arrive à zéro, c'est cuit, la partie est perdue !

Les spécifications détaillées suivent.

A noter pour tout ce qui n'est pas explicitement détaillé (par exemple : comment représenter visuellement un arbre, comment nourrir mon personnage, à quelle fréquence/quelle quantité doit apparaître la nourriture, etc.) tu dois donner libre cours à ton imagination tout en restant crédible dans le scénario.

## Spécifications détaillées

- À chaque déplacement sur la carte, l'hydratation et la satiété diminuent de 2 et l'énergie (inverse de la fatigue) de 3.
- L'avatar peut être mis en sommeil pour un nombre d'heures déterminé, pour chaque heure passée à dormir, l'hydratation diminue toujours de 2 (il fait vraiment chaud sur l'île), la satiété de 1 (c'est bien connu, qui dort dine) et l'énergie remonte de 6.
- Tu dois gérer 1 compteur pour le nombre de déplacements et 1 pour le nombre d'actions (autres que déplacements) effectués depuis le démarrage de la partie.
- De la nourriture pourra être trouvée sur la carte, apportant un pourcentage de satiété lorsqu'elle est consommée (attention à la nourriture non comestible 😊).
- De l'eau pourra être trouvée sur la carte, apportant un pourcentage d'hydratation lorsqu'elle est bue (attention à l'eau salée), la bouteille en verre pourra également être remplie (ou vidée, ou bue).
- Lorsque l'avatar arrive à un emplacement mystérieux, cela correspond à un défi à relever (un petit jeu à programmer dans le jeu), la réussite du défi donne un objet particulier à l'avatar (les notions essentielles à la réalisation de ces défis sont traitées lors des dojo et live coding). Il convient de gérer correctement les défis déjà gagnés si le joueur revient au même endroit.
  - Défi 1 : Le nombre mystérieux. Donne la clé de bronze.
  - Défi 2 : Le code César. Donne la clé d'argent.
  - Défi 3 : Le multi Fizz-Buzz. Donne la clé d'or.
  - Sortie : le quatrième lieu contient la porte d'accès au projet 2 et nécessite les 3 clés pour être ouverte.
- Lorsque le jeu est arrêté, il doit sauvegarder l'état exact de la carte (objets présents, lieux explorés ou non, énigmes résolues ou non), l'emplacement de l'avatar, ses niveaux vitaux, les compteurs de déplacement et d'actions et le contenu de son sac à dos.
- Lorsque le jeu est (re)lancé, on doit pouvoir choisir de continuer la partie en cours (1 seule sauvegarde) ou de démarrer une nouvelle partie avec les paramètres initiaux (carte initiale, position et niveaux vitaux de l'avatar, contenu du sac à dos).
- Lorsque la partie est gagnée (ou perdue), on sauve le score (date et heure, nom du joueur, gagné/perdu, les compteurs, les niveaux vitaux).
- En début de partie, on propose d'afficher l'historique des parties, trié par date/heure décroissante
- Enfin, le jeu doit être "modable", donc aucune information ne doit être codée "en dur" dans l'application. En clair cela signifie que, sans toucher une seule ligne de code, on doit pouvoir lui fournir :
  - une nouvelle carte
  - de nouveaux objets à de nouveaux emplacements et avec leur éventuel impact sur les signes vitaux
  - une nouvelle échelle de signes vitaux et d'évolution de ces signes lors d'un déplacement, d'une autre action, d'une heure de sommeil
- Évidemment, il ne sera pas possible de gérer des comportements non prévus (par exemple "miner" une ressource, construire un bâtiment, combattre, ...)

## Lieu mystérieux 1 (défi : le nombre mystérieux)

### Scénario :

En haut de la falaise en bordure de forêt, pas très loin de la plage, tu découvres la statue d'un Sphinx avec une grosse clé en bronze posée sur les pattes.

Lorsque tu t'en approches, les yeux de la statue s'illuminent et une voix se fait entendre :

*« Bonjour explorateur ! Pour ouvrir la porte de la montagne, atteindre le cœur de l'île et rejoindre tes compagnons, tu devras tout d'abord prouver ta valeur individuelle en gagnant les 3 clés que tu obtiendras en relevant les défis appropriés. Ceci est le premier d'entre eux. »*



Il est bien entendu impossible de prendre la clé (qui semble collée aux pattes du Sphinx) tant que le défi n'est pas gagné.

La voix poursuit : *« 3 fois de suite, tu devras deviner le nombre que j'ai en tête, tu as 20 essais maximum pour tous les trouver, es-tu prêt ? »*

### Spécifications :

- Le sphinx choisi un nombre au hasard entre 1 et 100.
- Le joueur doit proposer un nombre dans cette fourchette.
- A chaque proposition, le Sphinx lui donne une indication du genre : « Le nombre que j'ai en tête est plus petit (ou plus grand) ».
- Une fois un nombre trouvé, le Sphinx félicite le joueur et choisi un nouveau nombre.
- Lorsque les 3 nombre sont trouvés, le Sphinx le félicite à nouveau en lui disant en combien de tours il a trouvé tous les nombre puis le joueur peut récupérer la clé.
- Si le joueur dépasse 20 essais, il a perdu sort du défi. Il peut le recommencer autant de fois qu'il le souhaite.

### Améliorations possibles :

Si tu trouves cela trop facile, tu peux améliorer le principe du jeu à ta guise (par exemple représenter visuellement - mais toujours en console, éventuellement avec une ligne de  et  - les nombres possibles et éliminés, etc.

Formation Python Développeur Web & Web Mobile	PROJET 1 (SUJET)	06/07/2020 v 1.1 AM
---	---------------------	---------------------------

## Lieu mystérieux 2 (défi : le code César)

### Scénario :

Au nord de la plage, tu trouves un petit temple taillé dans la paroi rocheuse. Au-dessus de l'arche d'entrée, est écrit un message dans une langue apparemment inconnue mais utilisant l'alphabet habituel.

Sur les colonnes de l'arche, sont d'ailleurs gravées en relief toutes les lettres de l'alphabet. Curieux, tu appuies au hasard sur l'une des lettres et tu t'aperçois que les lettres formant le message changent. Malgré cela, le message en lui-même reste incompréhensible. En effectuant plusieurs tests, tu te rends compte qu'à une lettre définie correspond une composition spécifique du message. Ne voyant pas trop quoi faire d'autre pour le moment, tu décides d'entrer dans le temple.

A l'intérieur, tout le sol est recouvert de sable. Au milieu de la petite pièce, se trouve un autel avec une niche fermée par une grille apparemment sans serrure. A l'intérieur de la niche tu vois une grosse clé en argent (inatteignable tant que la grille est en place). A côté de la niche, un écriteau en bois indique : « *Récite le crédo du Python, puis trace ton nom secret* ».

### Spécifications :

- La phrase incompréhensible au-dessus de l'arche constitue le crédo (en fait les 3 premières phrases du Zen de Python- <https://www.python.org/dev/peps/pep-0020/>)
- Première partie :
  - Le jeu doit proposer de saisir n'importe quelle lettre de l'alphabet et afficher à l'écran le crédo en utilisant le code César ([https://www.wikiwand.com/fr/Chiffrement par d%C3%A9calage](https://www.wikiwand.com/fr/Chiffrement_par_d%C3%A9calage)) correspondant à cette lettre.
  - Si la saisie est vide, le crédo est affiché en clair (non crypté).
- Deuxième partie :
  - Si le joueur saisi plusieurs lettres (en fait il trace les lettres dans le sable devant l'autel), le jeu lui décrypte sa saisie avec la dernière clé (lettre choisie précédemment) et affiche le résultat.
  - Si le résultat correspond au nom du joueur (demandé au début du jeu principal), alors la grille s'ouvre et le joueur obtient la clé.
  - Si au bout de 5 essais de phrase cryptée le nom du joueur n'est toujours pas apparu, le défi s'arrête et la clé reste derrière la grille. Le défi peut être recommencé autant de fois que souhaité.

### Améliorations possibles :

Si tu trouves cela trop facile, tu peux améliorer le principe du jeu à ta guise, par exemple :

- En utilisant une cryptographie plus complexe telle le code de Vigenère ([https://www.wikiwand.com/fr/Chiffrement par d%C3%A9calage#/Le chiffre de Vigen%C3%A8re](https://www.wikiwand.com/fr/Chiffrement_par_d%C3%A9calage#/Le_chiffre_de_Vigen%C3%A8re)).
- En améliorant la représentation visuelle du jeu.
- Etc.



### Lieu mystérieux 3 (défi : Le multi FizzBuzz)

#### Scénario :

En explorant la jungle, tu remarques une bande de singes braillards évoluant dans les arbres. En écoutant avec un peu plus d'attention, tu te rends comptes que les singes semblent articuler des nombres et que deux sons reviennent régulièrement : Fizz et Buzz...

A un moment, les cris s'arrêtent et tu te retrouves subitement entouré par les singes.

L'un d'eux, probablement le chef vu sa démarche assurée, descend de l'arbre et s'approche de toi. Il te toise un instant puis, du doigt, te désigne un endroit en hauteur. En levant la tête, tu aperçois une grosse clé en or, pendue à une solide liane, tu commences à chercher un moyen de grimper dans les arbres, mais cela déclenche une vive réaction de la part des singes. Il semble qu'ils ne soient pas disposés à te donner la clé comme ça !

Tu te ravises alors, et le chef se rapproche de toi et commence à te parler...

« *Toi jouer ! Si gagner alors clé à toi !* »

Jouer ? Mais à quoi ?

Semblant lire dans tes pensées, le chef dit alors « *Nous montrer.* » Puis il dit « *1* ». Un autre singe dans les arbres dit alors « *2* », puis un troisième « *Fizz* » et ainsi de suite : « *4* », « *Buzz* », « *Fizz* », « *7* », « *8* », « *Fizz* », « *Buzz* », « *11* », « *12* »... A ce moment, tous les autres singes se mettent à rire puis celui qui vient d'annoncer 12 pousse un cri de déception et se retourne sur sa branche en boudant, puis le jeu recommence avec les singes restants : « *1* », « *2* », « *Fizz* »...

Au bout de quelques tours tu comprends le principe du jeu et tu tentes ta chance pour obtenir la clé.

#### Spécifications :

- L'objectif est donc de jouer au jeu du FizzBuzz avec les singes (10 dont le chef) et d'être le dernier joueur encore dans la partie à la fin.
- La règle est simple :
  - On compte à tour de rôle en commençant à 1 (le premier joueur est désigné au hasard à chaque tour).
  - Si le nombre à annoncer est un multiple de 3, il faut dire Fizz au lieu du nombre.
  - Si c'est un multiple de 5, il faut dire Buzz.
  - Si c'est à la fois un multiple de 3 et 5 il faut dire FizzBuzz.
  - Si on se trompe, on est éliminé de la partie et les joueurs restants recommencent jusqu'à ce qu'il n'en reste qu'un.
- Tu dois simuler les 11 joueurs (toi inclus) et laisser ton programme jouer automatiquement.
- Chaque joueur doit avoir un nom ainsi qu'un pourcentage de chances (fixe) de donner la bonne réponse lorsque c'est son tour :
  - Chaque singe standard à entre 10% et 70% de chances (tiré aléatoirement en début de jeu) de donner la bonne réponse (un tirage par singe).
  - Le chef a entre 50% et 80% de donner la bonne réponse (après tout, il n'est pas chef pour rien).
  - Ton avatar a entre 80% et 90% de chances de donner la bonne réponse.

- Si l'avatar reste seul à la fin de la partie, les singes s'enfuient tous en criant de désespoir dans la jungle et tu peux prendre la clé. Si l'avatar donne une mauvaise réponse, les singes se moquent copieusement de lui et l'invitent à rejouer, s'il décline, il revient au jeu principal. Le joueur peut retenter le défi autant qu'il le souhaite.

#### Améliorations possibles :

Si tu trouves cela trop facile, tu peux améliorer le principe du jeu à ta guise, par exemple en affichant le nom des joueurs (et leurs réponses successives) en cercle sur l'écran (comme une ronde) puis en réajustant le cercle à chaque joueur éliminé.

#### Lieu mystérieux 4 (la porte vers le niveau suivant)

##### Scénario :

Tout au nord de la zone, à côté d'une magnifique cascade descendant de la montagne, tu remarques une grotte. Intrigué, tu y pénètres mais l'intérieur est très sombre.

Heureusement, une torche et un silex pour l'allumer se trouvent sur une pierre près de l'entrée. Tu allumes donc la torche et tu t'enfonces dans la grotte. Au bout de quelques minutes, tu trouves une grosse porte en bois qui bloque le passage. Au milieu de cette porte, trois grosses serrures qui semblent refléter les couleurs de 3 métaux précieux : bronze, argent et or.

##### Spécifications :

- Si le joueur a récupéré les 3 clés lors des défis (et les porte dans son sac à dos), la porte s'ouvre et le jeu annonce le passage au niveau suivant.
- Sinon la porte reste close.

#### Améliorations possibles :

Si tu trouves cela trop facile, tu peux améliorer le principe à ta guise, par exemple en demandant d'insérer successivement chaque clé et en ajoutant un petit effet sympa à chaque fois, idem lorsque la porte s'ouvre.

## Spécifications techniques

- L'application est documentée. C'est-à-dire que, outre les spécifications ci-présentes, doivent être inclus dans le dossier Documents :
  - Un diagramme de flux - avec sous-diagrammes - clair et bien légendé (par exemple réalisé avec Diagrams.net- <https://app.diagrams.net/>)
  - Une liste des tâches à effectuer (product backlog) avec catégorisation, sous-tâches, priorisation et dépendances (mais pas forcément rédigées en User Stories) (par exemple réalisé sous Trello – <https://trello.com>)
- Le code respecte autant que faire se peut la PEP8 et les bonnes pratiques (DRY, ...)
- Les fichiers de l'application sont correctement organisés (dossiers spécifiques - Documents, Resources, ProgramFiles, ...)
- Tous les nommages (dossiers, fichiers, variables, constantes, fonctions, etc.) doivent être en anglais
- Le code doit être commenté (docstrings et #) et aéré (lignes blanches)
- L'application doit utiliser des fonctions (def) et être développé en mode procédural
- Le programme principal doit contenir un point d'entrée (if \_\_name\_\_ == "\_\_main\_\_":) et seulement des appels à des fonctions (pas de code direct)
- Tous les fichiers Python doivent être préfixé du codage UTF-8
- Il faut autant que possible éviter l'utilisation du mot clé "global"
- L'application doit offrir la meilleur expérience (UX/UI) possible à l'utilisateur (chaque action effectuée doit afficher un message explicite, elle doit être visuellement agréable - autant que faire se peut en console, intuitive, addictive - disons un minimum pour vouloir arriver à la fin au moins une fois :-D)
- Le package de l'application (incluant la documentation) doit être considéré par les autres développeurs au minimum comme acceptable pour être repris (pour maintenance et évolutions)
- L'application est développée en environnement virtuel
- Le package est publié sur GitHub sous licence libre

## Améliorations possibles

Si tu es en avance sur le projet, tu peux ajouter les spécifications suivantes :

- Fonctionnelles :
  - La représentation de la carte peut tirer parti des caractères semi-graphiques de la table ASCII étendue en place des seules lettres, chiffres et signes standards (attention, le fichier contenant le plan de la carte doit lui n'être constitué que de signes standards)
  - Toutes les infos représentées peuvent utiliser du texte riche (couleurs, positionnement absolu) grâce à l'implémentation de codes "escape"
  - La carte peut être représentée dans une résolution dépassant la taille de l'écran (elle devra donc pouvoir défiler lorsque le personnage se déplace, ce dernier sera centré tant que la carte peut défiler, ou s'approcher des bords de l'écran lorsque la bordure de la carte coïncide avec la bordure de l'écran)
  - Les lieux mystérieux peuvent faire l'objet de sous-cartes dans lesquelles il est également possible de se déplacer (par exemple une grotte), donc il convient de gérer les changements de carte (entrée/sortie)
  - Gérer plusieurs sauvegardes, associées à des joueurs distincts
  - Inventer une formule mathématique permettant de calculer un véritable score en fin de partie (et pas seulement l'affichage des différents paramètres)
  - Ajouter de nouveaux lieux mystérieux et de nouveaux défis
- Techniques :
  - Le code peut être développé en respectant les principes de Programmation Orientée Objet (normalement vu et appliqué dans le cadre du projet 2)
  - Les tâches du Product Backlog peuvent être rédigées comme des User stories (normalement vu et appliqué dans le cadre du projet 2)

## Conditions de réussite

Le jeu est considéré terminé :

- s'il répond à toutes les spécifications fonctionnelles et techniques
- et s'il est gagnable en démarrant une nouvelle partie (avec les paramètres initiaux)

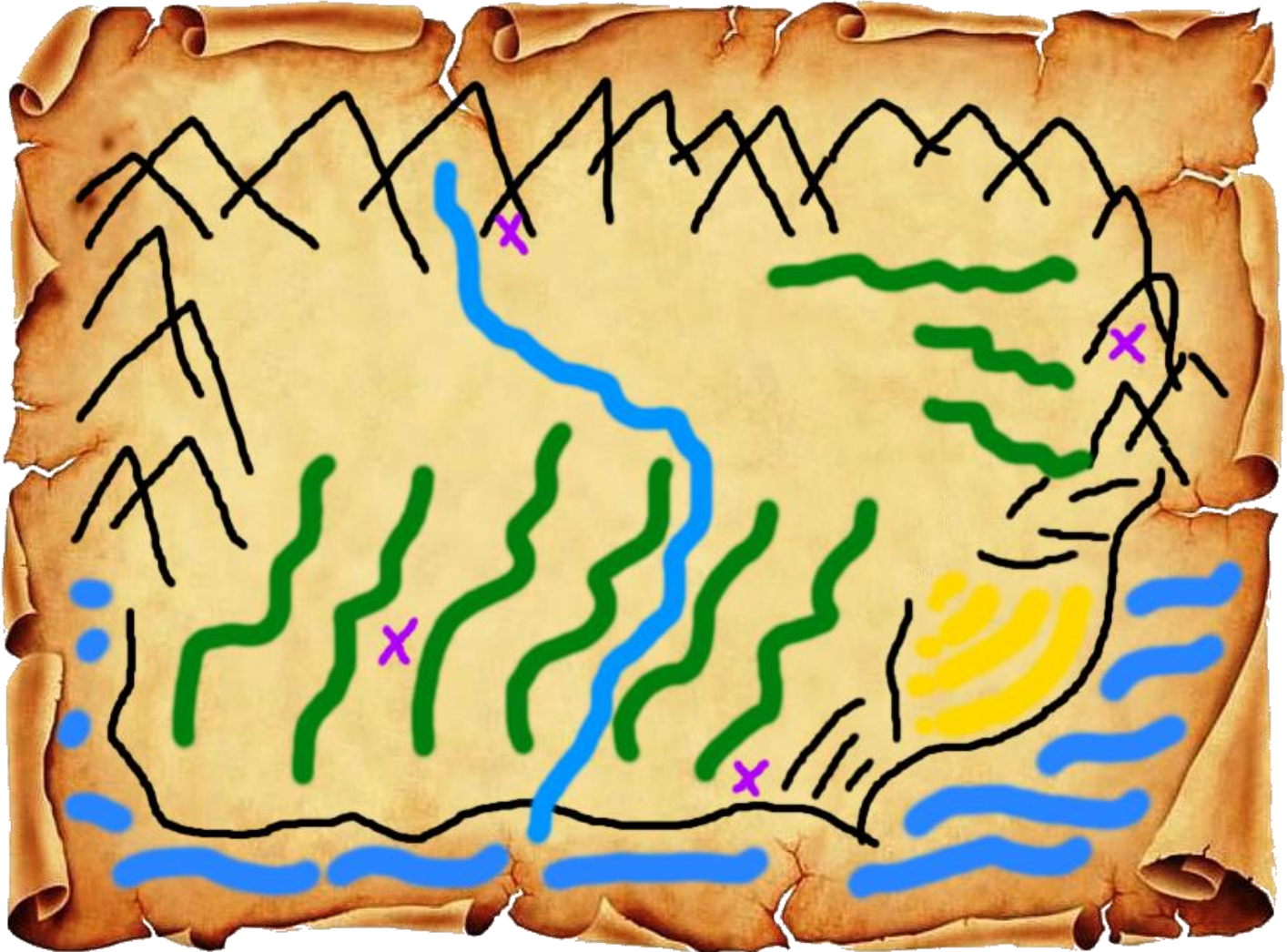
Le projet est validé si, en plus du point précédent, tu es en mesure d'expliquer :

- ta démarche
- tes documents
- ton code
- tes choix d'architecture...

Bref, si tu maîtrises parfaitement ta solution.

## Annexes

Carte de la portion d'île (ta position initiale : la plage).



Légende :

- En jaune : la plage
- En bleu : l'eau (la mer et une rivière)
- En vert : la jungle
- En noir : les rochers, falaises et montagnes
- Croix violettes : les emplacements mystérieux



Les objets de départ

