|  |  |
| --- | --- |
| Image illustrant une route sinueuse et des arbres  Rapport sur le projet Iltras  Un projet qui a commencé comme travail noté lors de la première année de NSI, finit en deuxième année de BUT Informatique. | Résumé  "Iltras" est un projet qui a été initié en 2021 avec l'objectif initial de créer un jeu narratif sur console en utilisant le langage de programmation Python. Ce développement a eu lieu au cours de ma première année d'études en NSI (Numériques et sciences informatiques), pendant laquelle je ne possédais ni les connaissances nécessaires sur les classes ni une compréhension approfondie des bases de données. Au fil de mes essais, j'ai identifié les erreurs commises et compris qu'une reconstruction complète s'imposait. Il devenait impératif d'adopter une approche plus robuste, de reconsidérer ma philosophie en matière de programmation, et de débuter cette fois par une phase de conception approfondie. Le présent document a pour dessein d'archiver l'évolution du projet "Iltras" depuis sa création en 2021 jusqu'à sa revitalisation en 2024.  Léo Christophe  Projet Iltras |

Table des matières

[Premiers développements 2](#_Toc157883461)

[Le sujet 2](#_Toc157883462)

[Le concept 2](#_Toc157883463)

[La réalisation 2](#_Toc157883464)

[La conclusion 2](#_Toc157883465)

[Résolution des problèmes et améliorations 2](#_Toc157883466)

[Le problème de l’histoire à choix multiples 2](#_Toc157883467)

[Une base de données 2](#_Toc157883468)

[Des sauvegardes 2](#_Toc157883469)

[Des classes 2](#_Toc157883470)

[Github 2](#_Toc157883471)

[Les nouveaux développements 2](#_Toc157883472)

[Contexte 2](#_Toc157883473)

[Diagrammes de conception 2](#_Toc157883474)

[Architecture du projet 2](#_Toc157883475)

[Base de données 2](#_Toc157883476)

[Réalisation 2](#_Toc157883477)

[Produit final 2](#_Toc157883478)

# Premiers développements

Conception et développement du jeu pensé au tout début, en 2021.

## Le contexte

Dans la phase préliminaire du développement du jeu Iltras, j'étais inscrit en classe de NSI lors de ma première année de lycée, une période où les élèves avaient la possibilité de choisir des spécialités, parmi lesquelles se trouvait NSI. Par conséquent, ce projet constituait l'un des travaux de fin d'année de première en NSI.

## Le sujet

Le but de ce travail, en fin d’année était de mettre en pratique les premières notions que nous avions vu en programmation sur le langage Python. Durant quelques semaines, nous nous sommes donc mis à programmer un projet pour la première fois en équipe de deux. Accompagné de mon collègue il fallait donc penser à un jeu, le programmer et le présenter.

## Le concept

### La phase de conception

Le but était de trouver quoi faire. Puisqu’on était en première année de la spécialité NSI, la plupart d’entre nous avions aucune expérience en programmation. Ainsi, nous avons passé la phase de conception habituelle où on devrait faire un modèle conceptuel de donnée (MCD), un diagramme de classe, un diagramme de cas d’utilisation et plus encore.

### L’idée du thème

Choisir le thème et l'envergure du projet s'est avéré initialement délicat, surtout compte tenu de notre manque d'expérience en matière de programmation. La décision a été prise de créer un jeu sur console, étant donné que nous maîtrisions principalement les scripts consoles en Python à ce moment-là. En ce qui concerne le thème, l'idée d'un jeu narratif à choix multiples a émergé pour accroître l'implication du joueur. Nous avons opté pour un contexte fantastique et médiéval, où le joueur commencerait en tant que jeune aventurier devant évoluer pour affronter un roi démoniaque et tyrannique. Les multiples fins possibles dépendraient des choix du joueur, avec diverses options à chaque étape du jeu, comme la découverte d'un coffre, la confrontation avec un ennemi, ou d'autres événements.

La gestion des données relatives aux objets et aux ennemis était effectuée à partir d'un fichier .csv, facilement réalisable sur Excel. Des requêtes au fichier .csv permettaient d'obtenir les diverses données nécessaires au jeu.

## La réalisation

Une fois le développement commencé, la progression était rapide et existante. On s’entraidant entre collègue quand on le pouvait et on avançait chez soi. Nous n’avions pas de dépôt Github donc c’était très difficile de réunir le travail. Ainsi, on s’est partagé le travail et on a programmer de façon séparée.

#### CSV

Pour récupérer les données, nous avons utiliser des documents .csv. C’était une solution très simple lors de notre premier projet. La récupération de données dans un programme Python, d’un fichier .csv se fait de la façon suivante :

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

D’un premier temps, on importe la bibliothèque csv, on parcours les lignes du document en les ajoutant au tableau donnees\_csv puis on lit les données du tableau en faisant une boucle simple.

#### Gestion des choix du joueur

Une image contenant texte, capture d’écran, Police

Description générée automatiquementPour gérer le choix des joueurs, nous étions contraint d’utiliser un système très simple. Avec moins d’un an d’expérience en programmation, on a ainsi décidé de faire un script linéaire d’une façon simple mais peu optimisée et peu lisible : AFFICHAGE DU CONTEXTE, PROPOSITION DE CHOIX, RECUPERATION DU CHOIX, ENVOI VERS UN AUTRE EVENEMENT SELON LE CHOIX.

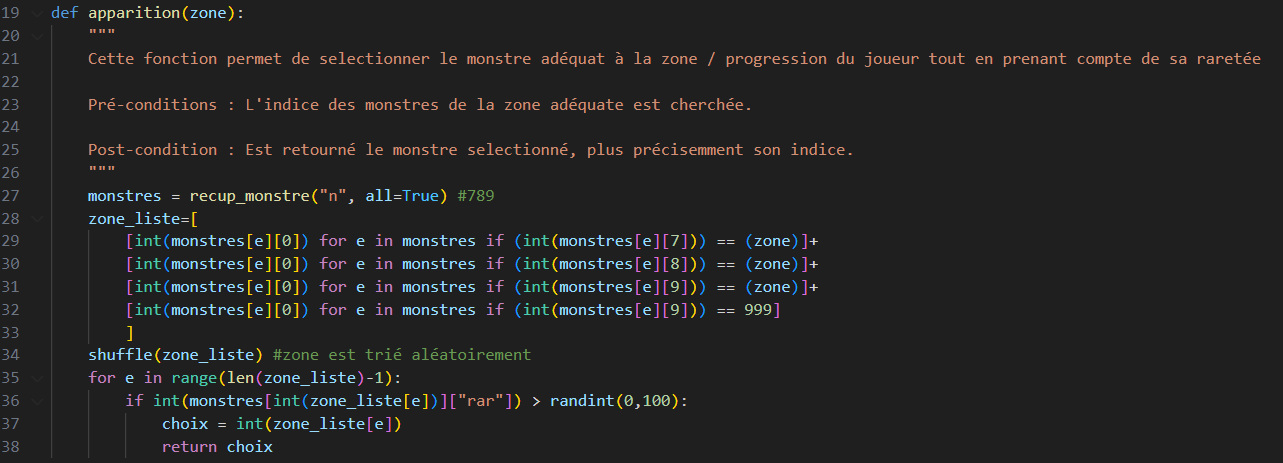
Le code pouvait devenir très conséquent et illisible. De plus, nous avions très mal géré les erreurs possibles.

#### Structure du projet

Le projet était un minimum structuré bien qu’il ne suivait aucune norme.

Instinctivement, on a séparé plusieurs fichiers d’une fonction différente.

La gestion des combats avec les ennemis,



le début de l’histoire, Une image contenant texte, capture d’écran

Description générée automatiquementla suite de l’histoire,

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquementla gestion d’une boutique.

Par ailleurs, on avait les différents fichiers CSV comme liste\_objets.csv ou ennemies.csv. Enfin on avait un README.md qui permettait d’informer les personnes qui utilisaient / jouer le jeu.

#### Bonnes pratiques

Même si c’était notre premier projet de programmation, on a réalisé de la documentation pour toutes les fonctions, des commentaires pour mieux comprendre le code. Evidemment, Python oblige une bonne indentation donc bien ou mal indenter n’était pas un problème.

Sinon, comme dit précédemment, on a essayé de séparer les fichiers avec des fonctions différentes.

#### Ce qui s’est mal passé

Travailler avec un fichier csv était difficile car on n’avait pas trop de liberté quant à la structure des données. Par ailleurs, on a plutôt mal nommé les colonnes, rendant plus ou moins difficile la relecture du code par soi-même à une date antérieure ou par quelqu’un d’autre.

Par ailleurs, au bout d’un moment, les bugs rencontrés nous a beaucoup posé problème nous ralentissant beaucoup. Débutants, il fallait donc que le professeur nous aide. Petit à petit, on a remarqué que nous ne finirions pas le jeu et qu’il était simple de tomber sur un bug.

## La conclusion

Après la date de rendu passé, le projet n’étant toujours pas finit, j’étais cependant fier de ce que j’avais fait. Sinon, pour le retour du professeur, la principale critique était la mauvaise gestion des arguments et des variables de retour.

En général, le projet était cependant approuvé en grande partie par le professeur, bien que je fusse résolus à le continuer et à l’améliorer.

Plusieurs problèmes ont été perçu du côté du développement en général :

1. Mauvaise gestion des bugs probablement dû à une mauvaise connaissance du débuggeur de Visual Studio Code et du manque d’expérience.
2. Mauvaise communication en équipe : Dû au contexte de l’exercice, on n’a pas utilisé GitHub, ce qui a beaucoup compliqué la collaboration en équipe. De plus, la façon de programmer est bien visible entre les deux partis. Enfin, le manque de communication s’est surtout fait remarquer quand il a fallu rassembler les divers fichiers.
3. Mauvaise gestion des erreurs possibles. Puisqu’on demande à l’utilisateur son choix, il aurait fallu mieux vérifier la justesse du choix. Par exemple, lorsqu’on doit rentrer un nombre, on doit vérifier le type (int) mais aussi le nombre (de 1 à 3 par exemple).

# Résolution des problèmes et améliorations

## Le problème de l’histoire à choix multiples

L'un des problèmes les plus évidents rencontrés concernait la gestion de l'histoire à choix multiples. L'approche consistait à afficher un événement A, puis à présenter les choix 1, 2, 3 de la manière suivante :Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Le principal inconvénient de cette méthode pour la réalisation d'une histoire à choix multiples réside dans l'optimisation, la lisibilité du code, et surtout, le stockage des données de l'histoire.

Diverses solutions ont été envisagées pour résoudre ce problème.

### Rentrer la donnée dans un fichier CSV

Cette approche implique l'utilisation d'un fichier CSV pour stocker les données de l'histoire à choix multiples. Cependant, elle présente un inconvénient majeur : la difficulté à reconnaître les différentes branches de dialogue dans le fichier CSV. La nature tabulaire du format CSV peut rendre complexe la représentation et la navigation des choix narratifs.

Prenons l'exemple du chemin A-1-B-2-C-1. Représenter cette séquence dans un fichier CSV peut se révéler laborieux, car il pourrait être nécessaire d'ajouter des colonnes supplémentaires pour chaque embranchement potentiel, rendant la structure du fichier peu intuitive. La recherche de chemins spécifiques devient ainsi peu pratique, impactant la maintenance et l'évolutivité du système.

### Rentrer la donnée dans une base de données

L'idée d'utiliser une base de données pour stocker les données de l'histoire a également été envisagée, mais elle a été écartée. La principale justification repose sur le fait que l'utilisation d'une base de données pour du texte uniquement peut être perçue comme une solution excessive. Les bases de données sont généralement optimisées pour gérer des données structurées et volumineuses, ce qui peut entraîner une complexité inutile pour un contexte principalement textuel.

En outre, l'utilisation d'une base de données pourrait entraîner des coûts de performance supplémentaires, en particulier lorsqu'il s'agit de charger et de requêter des données textuelles. Par conséquent, cette option a été écartée au profit de solutions plus simples et adaptées à la nature spécifique du contenu narratif du jeu.

En résumé, bien que le stockage dans un fichier CSV et l'utilisation d'une base de données aient été envisagés, ils présentent des défis importants en termes de représentation des choix narratifs et de complexité inutile pour des données principalement textuelles. La solution retenue, l'utilisation de JSON, offre une alternative plus adaptée en offrant une représentation externe et une lisibilité accrue tout en minimisant les inconvénients des autres approches.

### Un simple dictionnaire Python

Un des choix que j’ai considéré le plus judicieux était un dictionnaire sur Python. D’un premier abord, l’idée me semblait être très efficace et optimisée mais elle posait toujours un problème majeur : la lisibilité de mon code étant donné qu’un dictionnaire s’écrira directement sur le code. En effet, un des problèmes dans le projet réalisé auparavant était la lisibilité du programme, elle peut beaucoup jouer quand on essaye de résoudre les bugs.

### Solution choisie : JSON

JSON est peut-être la meilleure solution parmi celles étudiées. La sélection de JSON comme solution de stockage pour l'histoire à choix multiples se justifie par plusieurs avantages. JSON, acronyme de JavaScript Object Notation, est un format de données textuelles léger, structuré, et facilement lisible. Voici les points clés de cette solution :

#### Représentation Externe :

JSON offre une représentation externe des données, permettant de stocker l'histoire dans des fichiers distincts du code source. Cette caractéristique facilite la modification de l'histoire sans altérer le code, offrant une flexibilité appréciable lors des phases d'édition et de développement.

#### Lisibilité Structurée :

En raison de sa structure basée sur des paires clé-valeur, JSON est intrinsèquement lisible, facilitant la compréhension du contenu de l'histoire. Cette lisibilité accrue est cruciale pour le débogage, la maintenance du code, et la collaboration au sein de l'équipe de développement.

#### Facilité d'Édition :

L'édition d'une histoire au format JSON peut se faire de manière intuitive, même pour des personnes non impliquées dans la programmation. Des éditeurs de texte ou des outils spécialisés facilitent la manipulation et la création de branches narratives, améliorant ainsi la productivité de l'équipe.

#### Adaptabilité Dynamique :

JSON permet une adaptation dynamique de l'histoire pendant l'exécution du programme. Les modifications peuvent être apportées sans nécessiter une redéfinition complète de la structure du code, offrant une souplesse appréciable pour des scénarios où des ajustements en temps réel sont nécessaires.

Cependant, il est important de noter quelques limitations, notamment en termes de performance par rapport aux dictionnaires Python. Le processus de chargement et de transformation des données JSON peut être légèrement moins rapide, mais ces considérations doivent être mises en balance avec les avantages offerts en termes de lisibilité et de facilité d'édition.

En somme, l'utilisation de JSON comme format de stockage pour l'histoire à choix multiples représente un compromis équilibré entre facilité d'édition, lisibilité du code, et flexibilité durant le développement d'un jeu narratif.

## Une base de données

## Des sauvegardes

## Des classes

## Meilleurs suivis des erreurs

## Github

# Les nouveaux développements

## Contexte

## Diagrammes de conception

## Architecture du projet

## Base de données

Réalisation du script de création de la table sous PostgreSQL 15

## Réalisation

## Produit final