TBA

Rapport

Raphaël FEDRIGO - Alex FAUCHARD

---

1 - Guide Utilisateur

1.1 Contexte

Bonjour nouveau joueur, commençons avec un peu de contexte.
L'environnement de notre jeu est double, en effet vous commencerez dans un environnement plutôt classique, le Far West. Seulement vous n'êtes pas ici pour jouer les voyous, mais plutôt pour chercher à vous cacher et vous reposer.
Mais pourquoi donc chercher à se cacher, c'est une très bonne question. Dans ce jeu vous n'êtes pas n'importe qui, vous êtes Clyde Barrow, ce fameux bandit qui a terrorisé l'Amérique toute entière avec sa femme, Bonnie Parker. Sauf qu'aujourd'hui vous ne jouez pas un duo mais une seule personne.
La situation est celle-ci, vous êtes dans la peau de Clyde, celui-ci est en fuite depuis plusieurs jours car leur dernier braquage s'est mal passé. Malheureusement, vous avez dû abandonner Bonnie dans ce braquage. Ainsi, vous commencez le jeu en arrivant dans une ville du Far West, triste et fatigué.

La suite explique l'entièreté du scénario du jeu, si vous voulez conserver une expérience complète, il est déconseillé de lire les lignes ci-dessous.

Vous entrez alors dans l'hôtel de la ville pour avoir une chambre et vous reposer. Mais attention, le réceptionniste est un peu joueur et vous pose une devinette pour obtenir une chambre. Une fois que vous avez votre chambre, vous pouvez faire votre première sieste depuis des jours. Au réveil vous apercevez un objet brillant dans la table de chevet, c'est un morceau du collier que vous avez offert à Bonnie lors de votre première rencontre. Vous êtes alors pris de remords et allez passer la nuit au saloon. Au petit matin, vous vous réveillez au milieu de la rue complètement ivre. Mais votre humeur a changé, vous êtes maintenant déterminé à retrouver Bonnie. Ainsi, vous partez à la recherche du reste du collier en commençant par le bureau du shérif. Celui-ci vous pose aussi une petite devinette puis vous laisse aller voir le prisonnier qui a un deuxième morceau du collier.
Malheureusement, l'alcool est redescendu, et la tristesse est revenue. Alors vous décidez d'aller prier à la chapelle, et plus précisément dans l'isoloir. Seulement, vous vous endormez au lieu de prier.
À partir de là, toutes les choses se mélangent et vous êtes dès à présent dans un système de planètes inconnues. Actuellement, vous vous trouvez sur Kapry, et plein d'entrain vous partez à la recherche du reste du collier. Vous croisez tout d'abord un temple avec un Gardien qui vous pose une question en échange d'une nouvelle partie du collier. Ensuite, à force de vous balader, vous tombez nez à nez avec un gorille nommé Bakou, celui-ci est le chef d'une tribu de gorilles muets. Vous discutez bien et il vous propose alors d'aller sur une des lunes de sa planète pour trouver un autre morceau de collier. Enfin, en continuant d'explorer ce système planétaire, vous tombez sur la ville de Casino Land, où vous faites la rencontre de Jar Jar Binks qui, lui aussi, vous pose une devinette en échange d'un dernier morceau du collier. Vous récupérez alors le collier complet et commencez à apercevoir

Bonnie en face de vous. Malheureusement, au même moment, celle-ci disparaît et vous vous réveillez au milieu de l'isoloir. Cet espoir n'était qu'un rêve et votre situation ne s'est pas améliorée…

Précisions : Cette version est la plus directe pour terminer le jeu, mais il y a de multiples pnjs et objets avec lesquels interagir et de nombreuses pièces à visiter.

1.2 Conditions de victoire et défaite

Ce jeu ne propose pas réellement de scénario victorieux, comme vous avez pu le voir si vous avez lu le texte au-dessus. Cependant on peut tirer 2 fins possibles :

a - Sur cette première fin, vous réussissez à trouver toutes les devinettes et à trouver toutes les parties du collier (qui sont au nombre de 5). Ainsi, vous terminez complètement le scénario, ce qui peut être considéré comme une victoire.

b - La seconde fin peut être plus directe puisque vous pouvez perdre dès le début. En effet le jeu n'a pas de checkpoint ou de sauvegarde, ainsi chaque défaite signifie que vous avez totalement perdu. Par conséquent, si lorsqu'un personnage vous pose une devinette, vous ne trouvez pas la bonne réponse dans les 3 tentatives, la partie sera perdue.

1.3 Commandes et Carte

1.3.a Commandes

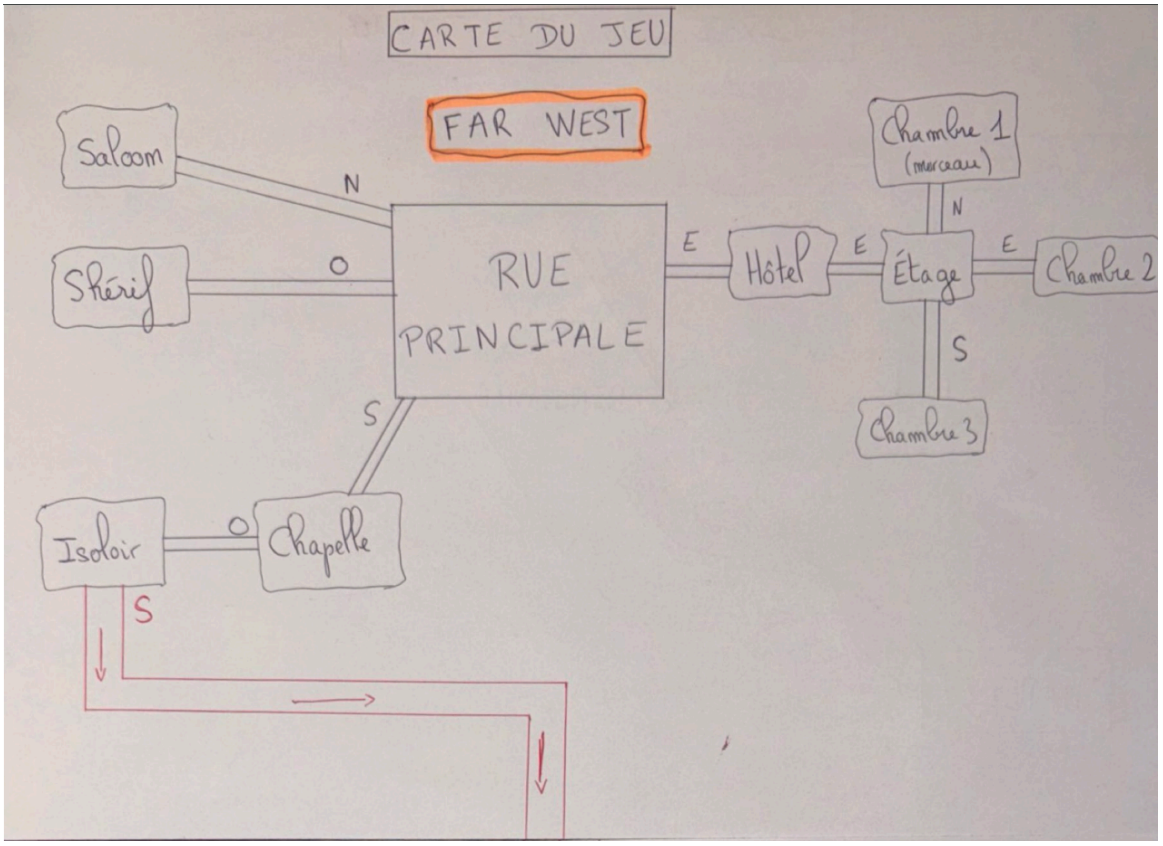
Voici les commandes du jeu avec leurs explications :

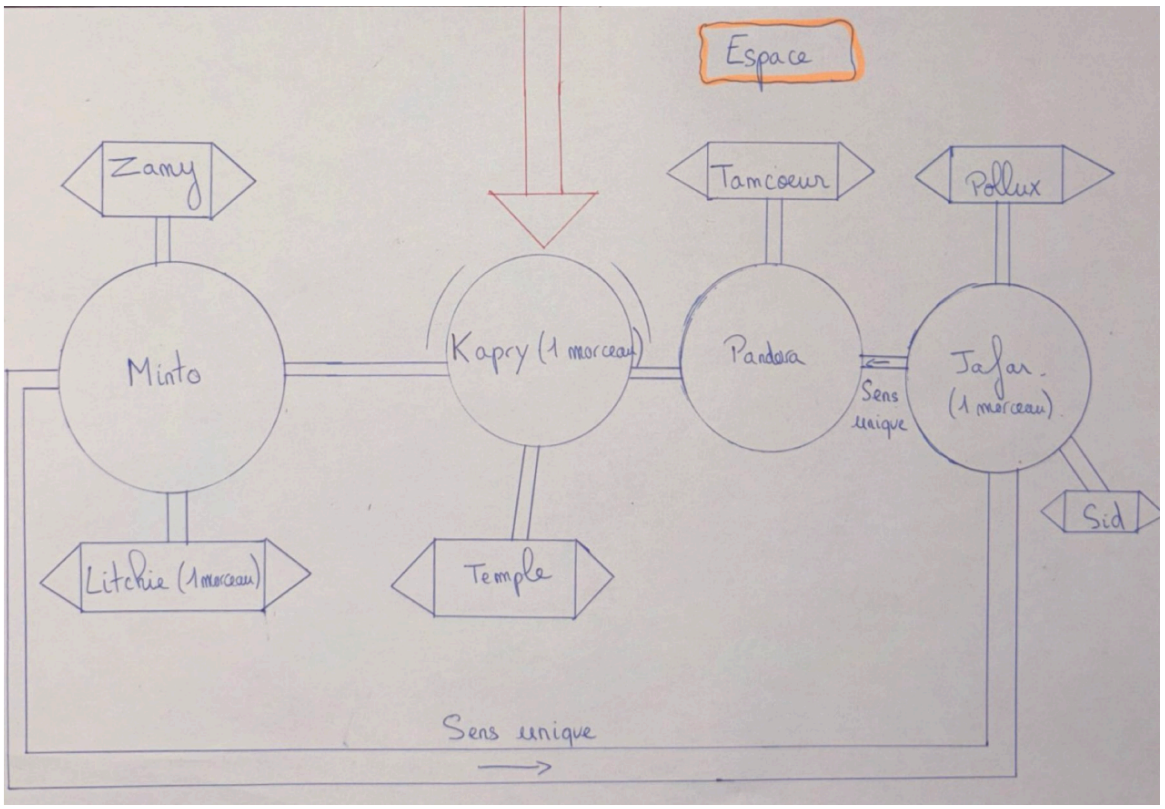
- help : affiche l'aide et toutes les commandes
- quit : quitte le jeu
- go <direction> : permet de se déplacer dans une direction (cardinale ou verticale)
- history : permet d'afficher la liste des pièces traversées
- back : permet de retourner dans la pièce précédente
- direction : donne votre position actuelle ainsi que les sorties possibles
- look : regarder dans la pièce pour voir les objets et les personnages présents
- take <objet> : permet de prendre un objet qui est dans la pièce
- drop <objet> : permet de jeter un objet de votre inventaire
- check : permet de regarder le contenu de votre inventaire
- talk <personnage> : permet d'interagir avec un personnage

Précision : les directions sont N, E, S, O, U et D.

1.3.b Carte

Voici la carte du jeu avec les 2 parties :

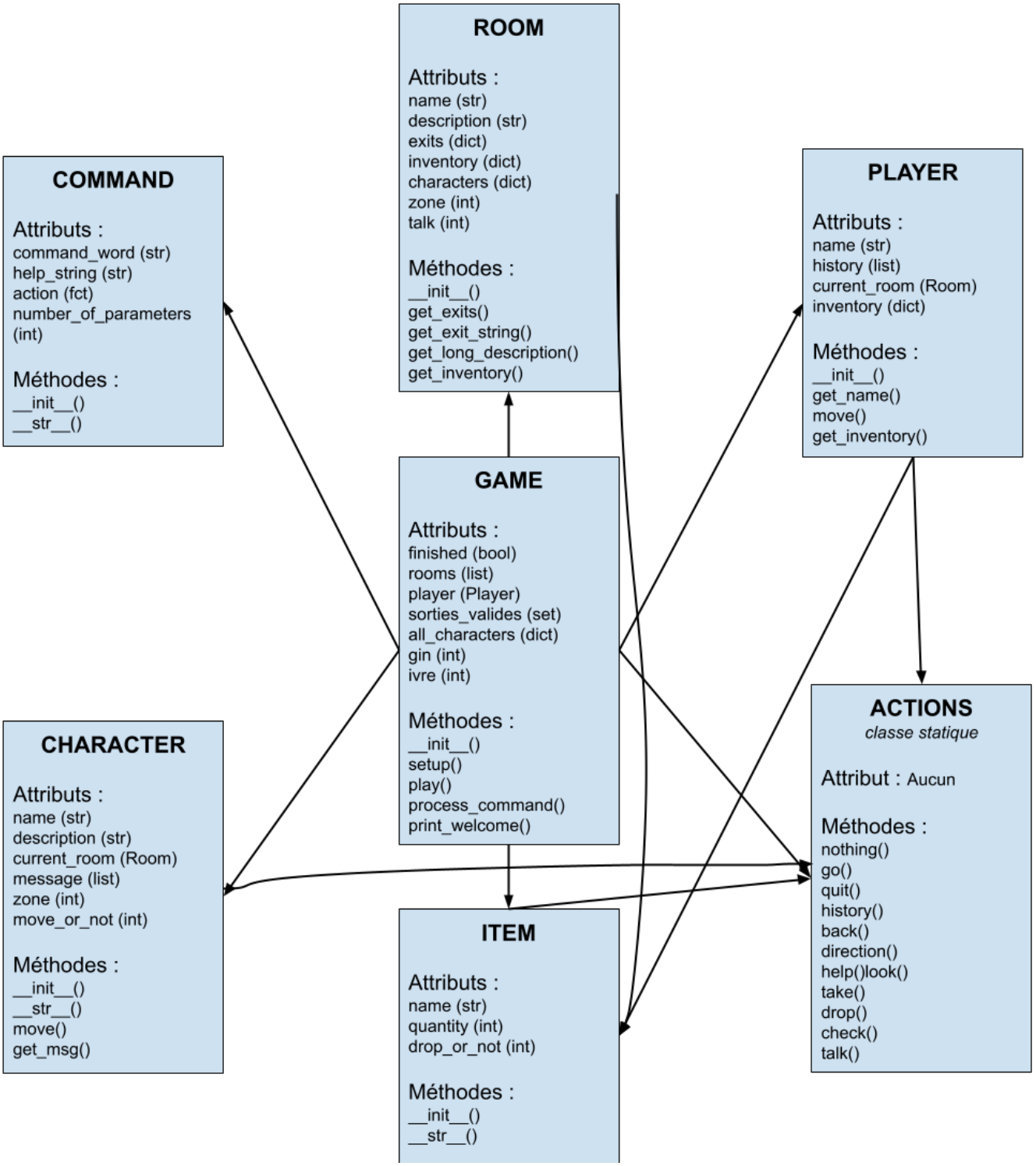




2 - Guide Développeur

2.1 Diagramme de classe

Voici le diagramme de classe du programme complet :



2.2 Points d'intérêts

Nous aimerions attirer votre attention sur quelques points qui nous semblent importants dans notre programme.

2.2.a Typographie

Tout d'abord, pour avoir un jeu simple à prendre en main, et donc favoriser l'expérience utilisateur, il nous a semblé important de simplifier la casse. C'est pourquoi lorsque que vous utilisez les commandes d'interactions (talk, take, drop, go), la typographie (majuscules, accents, etc) n'a aucune importance.
Pour cela, on a utilisé les tables de remplacement la fonction maketrans et on a décidé que tous les mots interprétés par l'ordinateur devaient être en majuscule sans accents ni espaces.
Ainsi, la commande go prend en charge les lettres uniques (ex : N), les mots complets (ex : nord) mais aussi les mots avec une typographie irrégulière (ex : Nord / nOrD/…). De même, les commandes avec interactions prennent en charge les noms complets quelque soit leur typographie.

Exemple :

```
    player = game.player
    table_remplacement = str.maketrans("ÉÈÀÙÇÊË","EEAUCEE")
    obj_recherche = list_of_words[1].upper().translate(table_remplacement).strip()
    if obj_recherche in player.inventory :
```

Extrait du code de drop()

2.2.b Zone et Move_or_not

Ensuite, comme notre jeu se déroule dans 2 univers différents mais liés, il a fallu trouver un moyen d'empêcher les personnages non joueurs de changer d'univers. Pour cela, nous avons ajouté 2 attributs à la classe Character : zone et move_or_not.
Le premier attribut permet de restreindre le pnj à une zone unique (Far West ou Espace) et est utilisé en tant que contrôleur dans la méthode move() de la classe character.
La seconde permet de bloquer les personnages non joueurs qui ne doivent pas bouger pour le bon déroulement du jeu. Celui-ci fonctionne de façon similaire mais dans la méthode move() de la classe Actions.
Ainsi, le joueur est à l'abri des incohérences et des ralentissement du scénario.

```
### La seconde partie du code permet aux pnjs de bouger de façon aléatoire
    for char in game.all_characters :
        ach = game.all_characters[char]
        # print(r.characters) # test
        if ach.move_or_not == 1 :
            ach.move()
```

Utilisation de l'attribut move_or_not

```
if self.zone == piece_adj.zone :
    self.current_room = piece_adj
    self.current_room.characters[self.name.upper()] = temp.characters[self.name.upper()]
    del temp.characters[self.name.upper()]
    return True
```

Utilisation de l'attribut zone

2.2.c Déclenchement d'événements et interactions

Enfin, le dernier point qui nous semble important est celui du déclenchement des évènements. En effet, le jeu réagit de façon "dynamique" aux interactions et agissements du joueurs. Pour cela, il a fallu ajouter des checkpoints dans le code permettant de déclencher des évènements et des contrôleurs.
Premièrement pour les devinettes posées au joueur et le contrôle des réponses. Ainsi nous avons créé des "dialogues spécifiques" qui se déclenchent quand le joueur parle à certains personnages (ex : Réceptionniste, Shérif, etc).
Ensuite, nous avons décidé d'ouvrir des sorties au fur et à mesure de l'avancement du joueur pour qu'il ne soit pas perdu. Pour cela, nous avons posé des conditions qui se déclenchent de façon unique grâce à la position du joueur, le contenu de son inventaire et/ou le des constantes uniques (ex : gin, tp, etc).
Pour finir, nous avons assemblé ces 2 méthodes pour pouvoir déclencher la fin du jeu, par mauvaise réponse ou fin du rêve, en combinant un contrôleur (la liste collier) et les dialogues avec les différents personnages importants de l'histoire.

```
            if spcr == self.rooms[6] and self.ivre == 1 and "PERLE" in spi :
                get_lines('Bar', 0)
                self.player.current_room = self.rooms[0]
                self.ivre = 0
                print(self.player.current_room.get_long_description())
                print("""
@ Vous n'avez plus accès au salon à cause de votre comportement @\n""")
```

Exemple de déclenchement par contrôleur, contenu de l'inventaire et position

3 - Perspectives de Développement

Dans cette troisième partie, on s'intéresse à ce qui peut être amélioré ajouté à notre code. pour améliorer le jeu sur le fond et la forme.

3.1 Optimisations

Pour commencer, plusieurs choses pourraient être optimisées.

3.1.a Répétition Character et Player

Les classes Character et Player restent assez similaires malgré les différents ajouts. Ainsi, il serait possible de créer une classe dont les 2 pourraient hériter. Cette optimisation permetterait d'alléger le code en réduisant un bon nombre de répétitions.

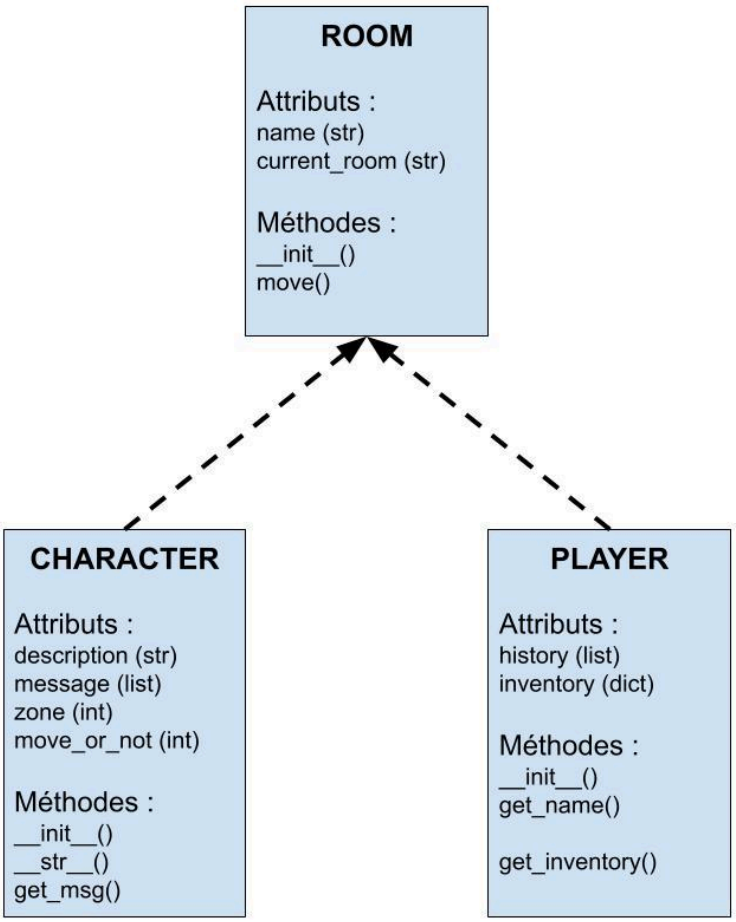


Diagramme d'héritage possible

3.1.a Dialogue

La deuxième optimisation qui pourrait être bénéfique au programme serait sûrement l'écriture des dialogues dans un fichier csv ou txt externe au programme. Ainsi, le code serait allégé et on pourrait mobiliser plus de connaissances techniques.

3.1.a Formalisation

Enfin, la dernière optimisation pourrait être la formalisation globale du code pour que celui-ci puisse être compris de façon plus simple. En effet, certains passages assez précis peuvent sembler obscurs ou très lourds en termes de programmation. Ainsi, il pourrait être intéressant de plus commenter les lignes et d'utiliser des variables clairement compréhensibles.

```
player = game.player
table_remplacement = str.maketrans("ÉÈÀÙÇÊË","EEAUCEE")
obj_recherche = list_of_words[1].upper().translate(table_remplacement).strip()
if obj_recherche in player.current_room.inventory :
    if obj_recherche not in game.player.inventory.keys() :
        player.inventory[obj_recherche] = player.current_room.inventory[obj_recherche]
    elif obj_recherche in game.player.inventory.keys() :
        player.inventory[obj_recherche].quantity += 1
    petit = player.inventory[obj_recherche].name
    print(f"\n{petit.capitalize()} est maintenant bien au chaud dans ton inventaire.\n")
    del player.current_room.inventory[obj_recherche]
    return True
```

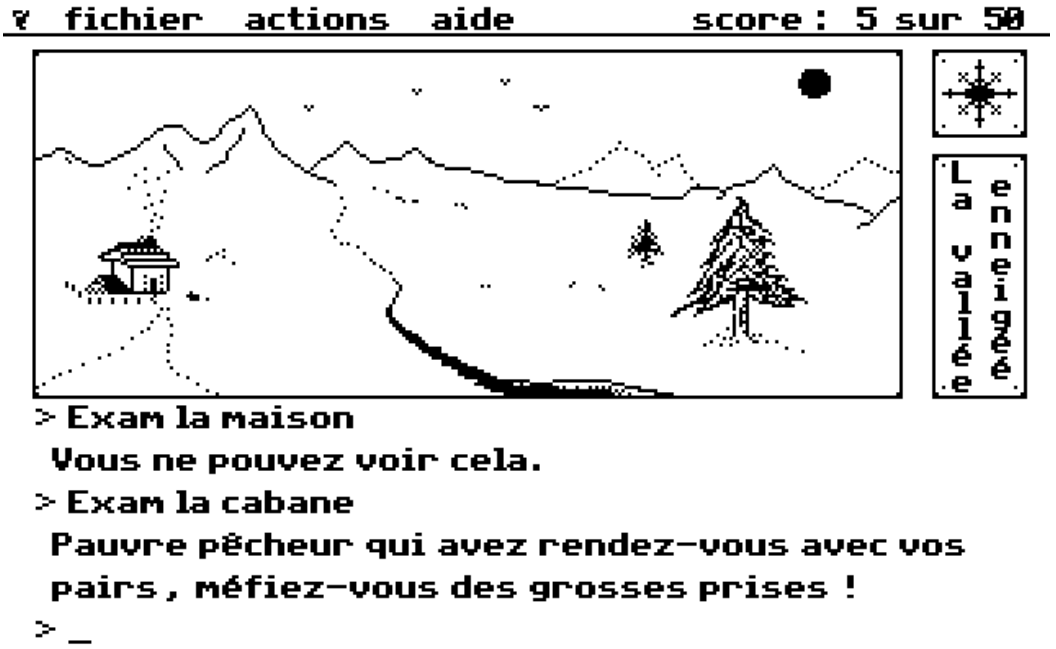
Partie du programme pour la méthode take

3.2 Améliorations

Dans un second temps, on pourrait penser à des améliorations pour rendre le jeu plus intéressant et captivant.

3.2.a Interface

L'interface est l'amélioration qui semble la plus évidente pour améliorer l'expérience utilisateur et la compréhension du scénario. En effet, l'ajout de boutons, d'une carte qui affiche la position, d'une image de la pièce ou encore d'un inventaire intéractif serait très intéressant. Initialement le jeu devait au moins avoir une interface graphique, seulement nous ne savions pas utiliser d'extension comme tkinter et le temps nous a manqué.



Type d'interface minimale que nous aurions pu implémenter

3.2.a Humanisation de Clyde

Une deuxième amélioration possible serait l'humanisation du personnage joué pour rendre le jeu plus captivant. Par exemple, on pourrait lui ajouter une barre de vie, de faim, de fatigue et des objets avec lesquels interagir. Ainsi, cela pourrait créer de nouvelles fins alternatives (mort de faim, plus de vie, etc). De plus, ces ajouts ne sont pas très techniques et pourraient facilement être implémentés sans grandement changer l'objectif et le scénario.

3.2.a Ajouts mineurs

Enfin, on pourrait implémenter plusieurs petits ajouts pour améliorer la jouabilité :

- le fait de pouvoir drop / take une quantité précise d'un objet (ex : take 2 cailloux)
- pouvoir interagir avec des personnages autrement que par le dialogue (ex : combats)
- pouvoir sauvegarder son avancement, pour ne pas tout recommencer en cas de défaite ou autre (plus compliqué à implémenter mais faisable)

4 - Vidéo

Voici le lien pour la vidéo de démonstration : Vidéo de présentation - Projet TBA

5 - Aide

Voici les réponses pour les devinettes posées par la personnage :

- Réceptionniste (Hôtel) : Son ombre
- Shérif (Bureau du Shérif) : Le feu
- Gardien (Temple) : Mon prénom
- Bakou (Minto) : Cupidon
- JarJarBinks (Jafar) : 13