

Rapport sur le jeu d'aventure textuel

Création d'un document comportant :

- Une description détaillée du sujet en mettant l'accent sur les missions attendues par le projet (**liste des fonctionnalités**, cas d'utilisation cf. module d'analyse)
- Une **étude de l'existant** (recherche web, étude comparative, applications similaires...)
- Une étude technique sur l'**existence de solutions** (technologies, API) permettant de mener à bien le projet

Description du sujet :

On cherche à créer le moteur du jeu qui doit comprendre des lieux (situés sur une carte et à l'état variable en fonction du temps, des actions précédentes...), des objets avec lesquels interagir, et un inventaire contenant les objets ramassés.

Afin de déterminer les actions à déclencher on créera un analyseur syntaxique qui convertira les demandes de l'utilisateur en scénarios contextuels.

On développera un éditeur graphique pour créer facilement de nombreux jeux avec le moteur générique établi précédemment : ajouter les lieux, modifier les verbes et noms de l'analyseur syntaxique, préciser les objets et les états de jeu possibles et déclarer les résultats des différentes actions.

Étude de l'existant :

- **Applications similaires :**
 - **SRAM 1 & 2 (1986) :**

Jeu aventure textuel ou de fiction interactive par analyse syntaxique.
Via le clavier le joueur est invité à taper des chaînes de caractères sous forme de phrases contextuelles (demander qqch, aller vers, prendre...).

Chaque phrase déclenchera une action si celle-ci est compréhensible par l'analyseur syntaxique, à défaut une réponse humoristique sera donnée.

Le jeu se déroule sous forme de tableaux, une ou plusieurs actions est à réaliser dans un ordre qu'il faudra découvrir, les déplacements sont effectués en suivant les points cardinaux (nord, sud, est, ouest).

- Le passager du temps (1987) :



Fonctionnalités (use cases) :

Niveau Primaire	Niveau secondaire	Description de la fonctionnalité
Analyseur syntaxique		
Trouver expression		On crée un lexique de verbe listant les actions possibles, et un lexique de compléments suivant le verbe. On retirera les caractères inutiles (déterminants, espaces, ponctuation) dé ex : “Prendre/Ramasser/Attraper la clé” → “Tourner la clé dans la serrure” → “Ouvrir porte” En général : Verbe + complément → Action
		Utiliser un arbre n-aires qui déterminera selon la suite de mots, l'action à déclencher

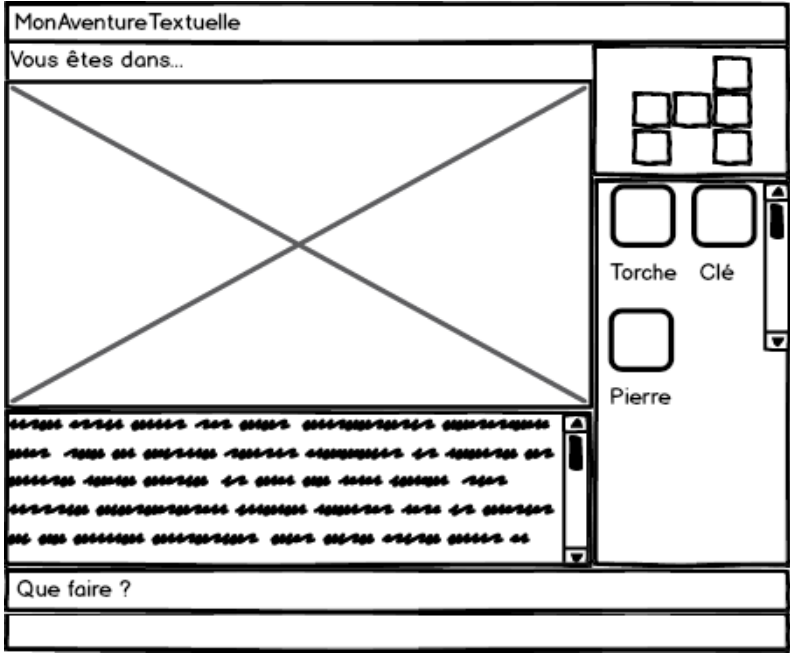
Moteur (à segmenter : déplacement, objets, inventaire...)

Se déplacer	Aller au nord	L'utilisateur se déplacera sur la carte du haut / bas / droite / gauche en fonction de la commande entrée. Il pourra se déplacer si une carte est existante et si le passage est possible. Le moteur placera le joueur dans le lieu demandé.
	Aller au sud	
	Aller à l'est	
	Aller à l'ouest	
Générer carte	Dessiner salles	Afficher lieu, afficher objets
	Dessiner liens	Marques les chemins possibles entre les salles
Agir sur environnement	Prendre objet	Ramasser un objet présent dans l'environnement
	Regarder	Objet (description des objets) ou lieu (description générale)
	Changement d'état	Le moteur met à jour les états selon la situation. Par exemple l'état de porte ouverte si le joueur a ouvert une porte.
Afficher inventaire	Lister objets	Affiche de manière graphique ou textuel le contenu de l'inventaire
	Chercher objet	
Gérer l'inventaire	Insérer objet	Le moteur place dans l'inventaire un objet il peut être utilisé par le joueur et remplir les conditions pour certaines de ses actions
	Retirer objet	Le moteur supprime de l'inventaire un objet, il n'est plus utilisable depuis l'inventaire par le joueur

Éditeur de niveaux

<div> <div>Menu</div> <div> <div>Carte</div> <div>Script</div> </div> <div> <div>Cartes existantes</div> <div> <div>Actions</div> <div>+</div> <div>-</div> </div> <div> <div>Objets</div> <div>+</div> <div>-</div> </div> </div> </div>		
Créer script	Créer liens entre salles	Définir les passages possibles pour le joueur
	Associer objets à une salle	Chaque salle comporte une liste d'objets récupérables ou non,
	Associer conséquence à objets	On donnera la liste des objets nécessaire pour déclencher une conséquence
	Définir conséquence	Plusieurs actions pourront être déclenchées par le système suite à une combinaison de conditions
Créer élément	Créer salle	Créer une salle consiste à créer une nouvelle carte et de définir l'image d'arrière plan de celle-ci
	Créer objet	Créer un objet permettra de stocker l'objet créer afin de pouvoir l'utiliser plus tard dans différentes cartes. On le définira par un nom, une description et une image
	Ajouter action	Ajouter une action servira à créer une possible interaction de l'utilisateur avec la carte ou un objet
	Ajouter salle	ajout la salle à la carte
Sauvegarder Carte	Sauvegarder	Sauvegarde la carte créer afin de pouvoir l'utiliser dans le jeu ou bien de la rééditer.
Charger Carte	Charger	Charge une carte déjà existante afin de la modifier

Implémentation (IG) :

Jeu aventure textuelle	Intermédiaire	<ul style="list-style-type: none"> • Modèle : liste des <u>objets du lieu courant</u>, liste des objets de l'<u>inventaire du joueur</u> • Vues : <ul style="list-style-type: none"> ○ <u>VueInventaire</u> : affiche les objets en inventaire ○ <u>VueCarte</u> : emplacement actuel et salles adjacentes et marquées comme déjà visités dans un même niveau ○ <u>VuePiece</u> : lieu courant où se trouve le joueur, affiche les objets de la pièce et le décor • Contrôleur : <u>ContrôleurSaisie</u> : zone de saisie des actions du joueur géré par un contrôleur (lexique) 
Éditeur de niveaux	Avancée	Proposition moteur graphique

Déroulement attendu :

A partir des use cases développés ci-dessus, on cherche à obtenir un comportement donné avec le moteur du jeu.

Ce dernier sera généraliste : il doit permettre de construire n'importe quel jeu textuel.

Ainsi, on ne devra pas écrire le nom des objets et leur effets dans le code, mais plutôt permettre l'ajout de conditions nécessaires pour que des conséquences (modification de l'inventaire, changement de lieu...) soient déclenchées.

Il s'agit en résumé d'une génération dynamique d'objets et de conditions amenant à des conséquences.

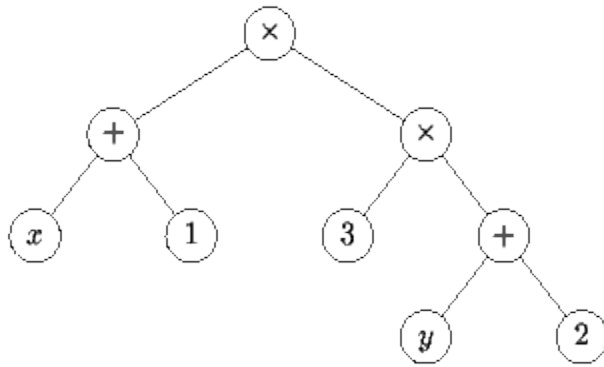
On aura par exemple l'énigme d'une porte verrouillée qu'on souhaite franchir, les objets qu'on aura à notre disposition seront : porte - poubelle - feuille - clé - serrure.

Si on trouve la suite correcte de conditions, on terminera le jeu. L'exemple suivant permet de visualiser une résolution type de ce genre d'énigme et comment le jeu devra gérer le cas.

- Regarder serrure (voir clé)
 - Interaction avec l'environnement par une syntaxe observer/regarder/...
 - Fouiller poubelle (--> feuille)
 - gestion de l'inventaire : ajout
 - environnement : la feuille n'y est plus
 - Placer feuille sous porte
 - Condition : avoir la feuille dans l'inventaire + faire action de placer la feuille
 - Secouer porte (clé tombe)
 - répond à la condition d'avoir la feuille placée et d'écrire l'action secouer
 - environnement : la clé change d'état
 - Tirer feuille (-->clé)
 - Récupération de la clé dans l'inventaire et changement d'état de la clé
 - Répond aux conditions : avoir fait tomber la clé sur la feuille et d'écrire l'action de tirer cette feuille
 - Ouvrir serrure avec clé (porte ouverte)
 - Changement d'état dans l'environnement : porte ouverte et donc nouveau lieu possible d'accès
 - Aller vers Nord
 - Le jeu change de lieu le personnage, vers le nouveau lieu souhaité par le joueur
 - Répond à la condition d'avoir ouvert la porte et de demander l'action
- **FIN DU JEU**

Solutions techniques :

- **Analyseur syntaxique :**
 - **Analyseur LR**, qui analyse une chaîne de caractères de gauche à droite (Left to right) et construit un arbre syntaxique en commençant par la racine
 - **Analyseur récursif descendant avec retours en arrière**
 - **Analyseur ascendant**



- **Java :**
 - **Structure de données : TreeMap**
 - **Méthode sur les String : (cf API)**
 - `parser`
 - `contains`
 - `compareTo / compareToIgnoreCase(String anotherString)`
 - `startsWith(String prefix)`
 - `endsWith(String suffix)`
 - `split(String stringSeparator) : String[]`
 - `match(String morceauChaine)`

Sources :

- Sujet du projet : <https://drive.google.com/file/d/0B-95anfbJhzEaFFOOGcxOGtrcjA/view>
- Analyseur syntaxique :
 - <http://stackoverflow.com/questions/8204914/syntax-analysis-and-syntax-tree>
 - <http://www.commentcamarche.net/forum/affich-13587602-analyse-syntaxiqueGén-en-java>
 - <http://blog.soat.fr/2013/12/08-jvm-hardcore-part-7-mon-premier-analyseur-syntaxique-12/>
 - <https://www.enseignement.polytechnique.fr/informatique/ARCHIVES/IF/poly/main004.html>
 - Wiki de Analyse_syntaxique
- Jeux textuels :
 - <http://www.grospixels.com/>
 - <http://www.planetemu.net/article/le-passager-du-temps>