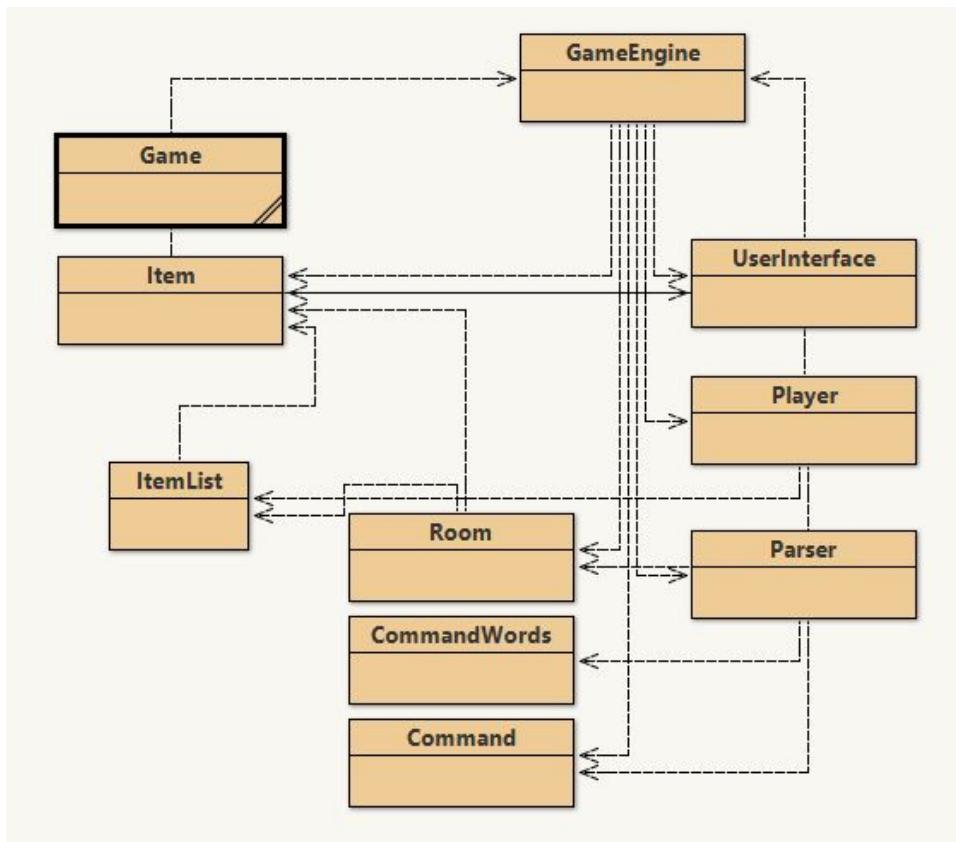


Rapport: Mission CAT!

disponible sur perso.esiee.fr/~molaeia

(A3P(AL) 2018/2019 G12c ESIEE Paris)

Auteur: Avesta MOLAEI



Professeur: M. Denis BUREAU (5356)

I/

A)

>Auteur: Avesta MOLAEI 12c

B)

>Phrase-Thème: à Moscou en 1964; l'agent 42 doit empêcher le lancement de missiles russes vers les Etats Unis!

C)

>Résumé du scénario: [NB: Je sais que Staline est mort en 1953 (merci a ma copine de m'avoir notifié ce détail :-)) qui m'aurait fait passer pour un inculte. L'histoire se passe dans une réalité alternative)]

>1964. Les tensions de la guerre froide sont à leur apogée. Les Etats Unis, et l'Union Soviétique sont tous les deux sur le qui-vive pour engager un conflit direct. Staline décide d'envoyer les derniers missiles nucléaires soviétiques sur les Etats Unis.

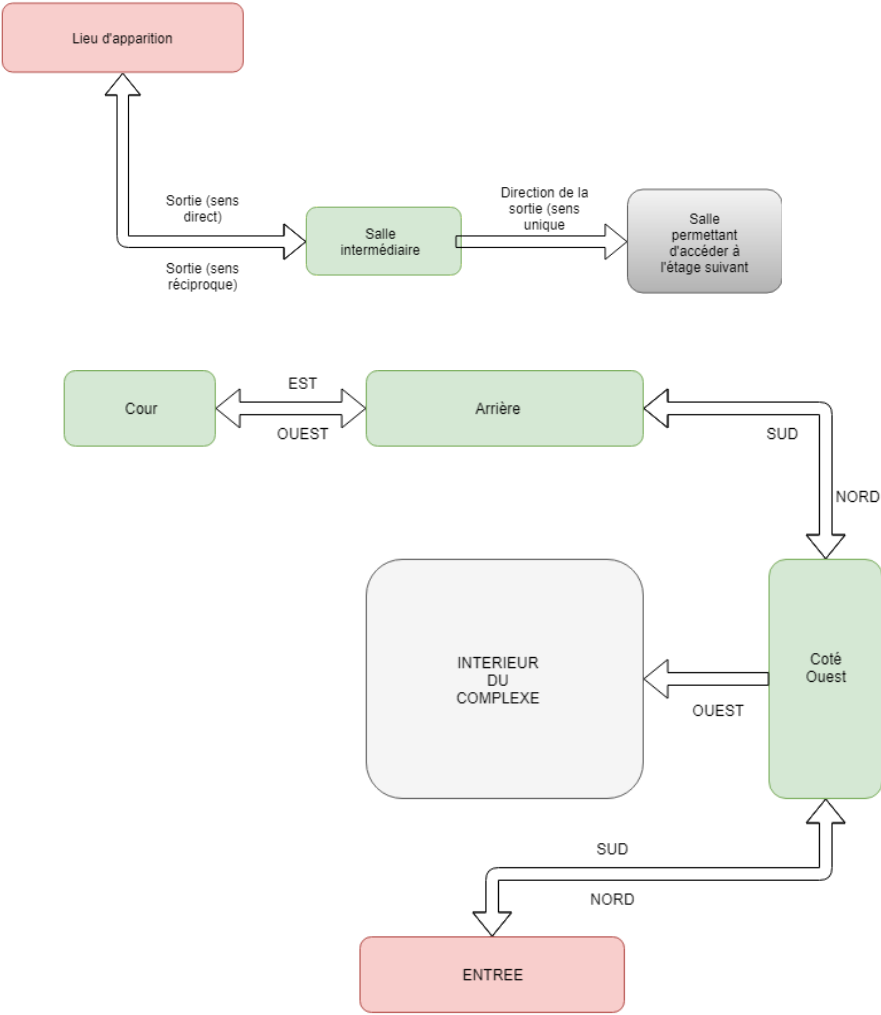
>Le Joueur incarne l'agent 42. Personne ne sait qui il est. Même lui ne sait pas. La seule chose connue de lui est qu'il est le meilleur agent qu'il y ait.

>Sa mission? En tant que meilleur élément de la section CAT! (Counter of annihilating threats, ou brigade anti-menaces d'annihilation en Français), il doit trouver un moyen de s'infiltrer dans le complexe de lancement, et d'annuler le lancement!

D)

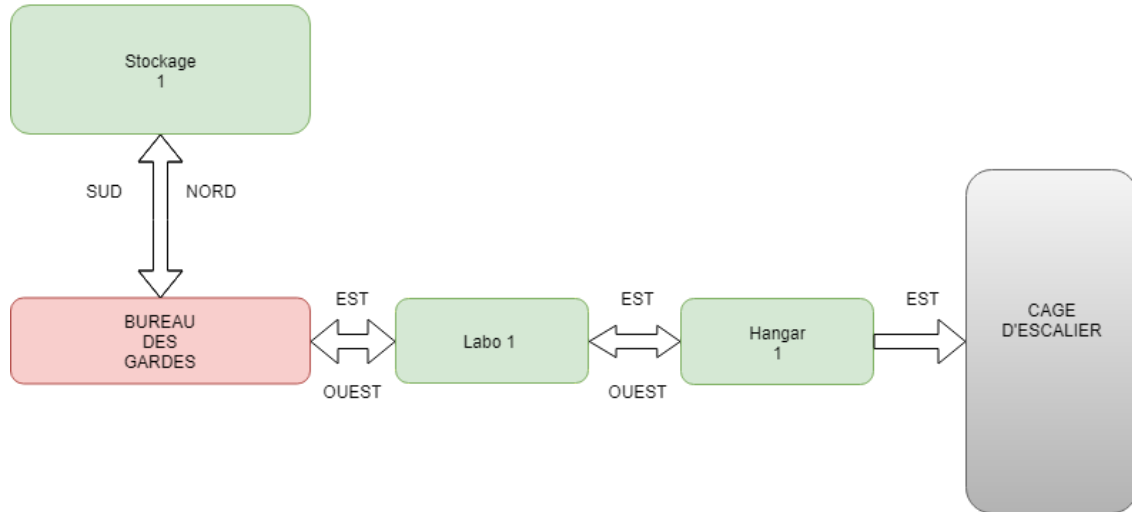


LEGENDE

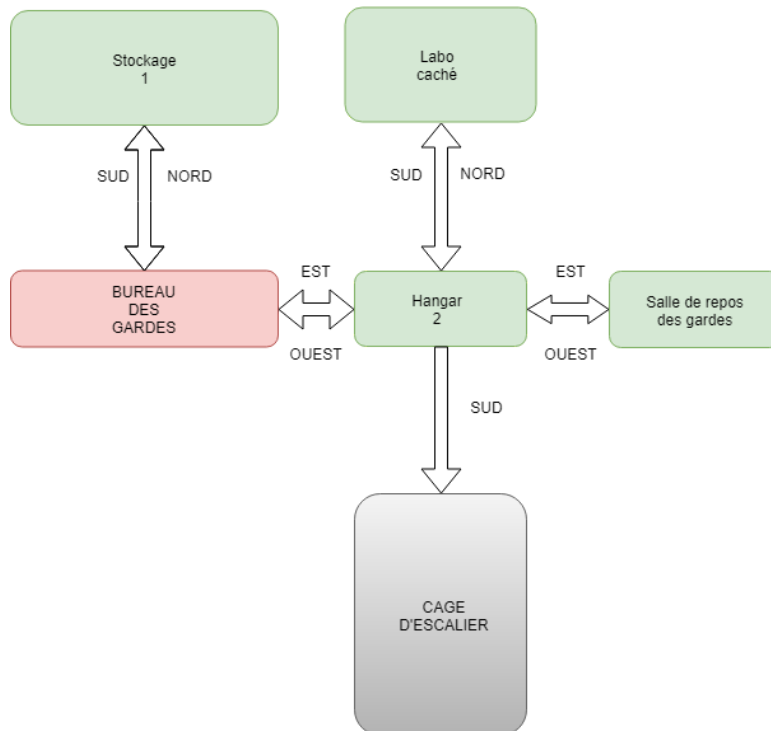


EXTERIEUR

ETAGE 1



ETAGE 2



ETAGE 3



E) Le scénario complet détaillé est mentionné en I.C)

F) Les lieux sont représentés dans la question I.D) Le complexe en lui même est situé dans une zone inconnue en Russie. Il est constitué de 3 étages. Tout d'abord, pour rentrer, il est évident qu'on ne pourra pas s'infiltrer par l'entrée principale... Pour cela, il faut passer par la bouche d'aération présente sur le côté est du complexe. Pour pouvoir atteindre cette bouche d'aération, il faut utiliser les outils trouvés à l'arrière du complexe. Dans la cour, on peut y trouver des munitions pour l'arme de base, un pistolet. Une fois que l'agent est rentré, il doit se frayer un chemin jusqu'à la salle de contrôle, au troisième étage. Lorsqu'il entre par la bouche d'aération, l'agent entre sans le savoir dans le bureau des gardes, avec un garde ennemi présent! il doit se frayer un chemin jusqu'à la cage d'escalier (certaines sorties sont bloquées par des cartes d'accès à trouver soit par des puzzles, soit en battant des ennemis). A l'étage suivant, il peut:

soit directement aller à la cage d'escalier si il obtient la carte qui bloque la sortie du hangar vers la cage d'escalier, soit aller dans le laboratoire caché au nord ou un scientifique fou qui n'a plus toute sa tête raconte son histoire. Sinon... Il peut aller dans la salle de repos des gardes... Ou 4 d'entre eux jouent aux cartes. Soit il les bat, soit il emploie un code de triche ^_^ . Les battre permettra au joueur d'accéder à une arme surpuissante.

Au dernier étage, le joueur affronte Staline, afin d'obtenir sa carte d'accès à la salle des contrôle des missiles. Dans cette dernière, il dispose d'un temps limité pour annuler le lancement. Si oui, "GG WP" comme il est souvent mentionné dans le monde du jeu vidéo, sinon, il a perdu!

G) Les Situations gagnantes: il n'y en a qu'une seule.... Le joueur tue Staline et annule le lancement. Pour les perdantes, il peut: soit mourir, soit prendre trop de temps pour annuler le lancement

H) Il y aura en effet des énigmes: des puzzles de logique permettant de débloquent des items ou des salles. Les combats seront aussi présents, contre les gardes et Staline.

I) Beaucoup d'éléments sont encore manquants. La liste des exercices n'est pas finie, et les images ne sont que provisoires.

II Réponse aux exercices

>Au 11.02.2019: TP 4.2

>Exercices 7.5, 7.6, 7.7, 7.8 et 7.8.1 réalisés sans difficultés & avec succès

>7.5: pour réaliser `printLocationInfo()`, il a fallu modifier une grande partie du Game. En effet, il y avait 3 occurrences d'un même segment de code indiquant au joueur où il est et quelles sont les sorties disponibles (dans `goRoom()`, `printHelp()` et `printWelcome()`). Afin d'ETDDC, on a créé la procédure `printLocationInfo()`, ainsi, on remplace chaque occurrence mentionnée précédemment par un appel de la procédure. Pas de duplication de code ainsi, et un couplage diminué. Si on veut modifier l'affichage, on modifie ainsi qu'une seule procédure, et non plus 45 segments de code différents.

>Site web du projet commencé (dispo sur perso.esiee.fr/~molaeia (templates et CSS utilisés tirés de W3C schools. Je suis comme modèles mon site personnel (avestamolaei.ml) et mes autres réalisations HTML de MSH et de Terminale))


>Au 12.02.2019: RESA 4.3

>Exercices 7.9, 7.10, 7.10.1, 7.10.2, 7.11, 7.14, 7.15, 7.16, 7.18, 7.18.1, 7.18.2, 7.18.3, 7.18.4, 7.18.5, 7.18.6, 7.18.7 réalisés sans difficultés. et avec succès

>7.10: Avec la boucle `for each`, on parcourt la `HashMap` comme si c'était un tableau. On crée une `String`, à laquelle on concatène un retour chariot et les sorties disponibles (grâce à la `hashmap` du coup). Ainsi, on obtient sous forme d'une jolie "liste" (en `String`) les Sorties, ainsi facile à manipuler

>7.18.7: `.addActionListener()` inscrit que son paramètre va réagir aux modifications sur l'objet sur laquelle la méthode est appelée

`.actionPerformed(ActionEvent e)` intervient pour traiter les événements en appelant la bonne méthode en fonction du bouton pressé.



>Au 14.02.2019: TEMPS PERSO

>Exercices 7.18.8, 7.19, 7.19.2, 7.20, 7.21, 7.22, 7.22.1, 7.22.2 réalisés sans difficultés et avec succès

>7.19: Le but d'un modèle MVC est de réaliser un nombre minimal de modifications lorsqu'un nouveau besoin est discerné. En effet, les différentes composantes d'un programme sont ainsi plus structurées, et indépendantes. Aussi, le code devient ainsi plus accessible par les autres développeurs. Le but principal reste à minimiser le couplage. (et ETDDC!)

>7.21: Les infos à propos d'un `Item` doivent venir d'une classe `Item` en elle même.

La `String` décrivant le dit `Item` est produite par lui même, mais l'affichage se fait dans `GameEngine` (là ou l'affichage via `aGui.print()` se fait déjà)

>7.22.1: On préfère le choix de la `HashMap` en collection puisqu'ainsi, un `Item` n'est plus attribut d'une `Room` mais est stocké dans une collection, la `HashMap`, dont chaque `Room` dispose.

>Site Web mis en ligne sur perso.esiee.fr/~molaeia

>Au 20.02.2019: TP 5.2:

>Exercices 7.23, 7.24 et 7.25 réalisés avec succès et sans difficultés.

>Exercice 7.26 réalisé avec succès, mais avec un peu de difficultés

>Exercice 7.26.1, 7.27, 7.28, 7.28.1, 7.28.2, 7.29 réalisé avec succès et sans difficultés.



>Au 17.04.2019: Rendu intermédiaire:

>Exercices 7.29 à 7.34 réalisés avec succès mais avec beaucoup de difficultés. J'ai du regarder plusieurs tutoriels, me faire aider par mon père + des connaissances, et passer beaucoup de temps à essayer. Au final, les notions sont acquises.

EDIT au 07/05: il semble que le take ne soit pas fonctionnel?? dzkfjdoiusfdkjsqdlkjqlkjqs

>Exercices 7.34 à 7.34 réalisés avec succès et sans difficultés. Projet près pour le rendu intermédiaire le 18.04.2019 - 03:39:23

>Suivi d'un tutoriel et apprentissage de l'utilisation des StringBuilder, simplifiant ainsi la création de descriptions

>Composition d'une vraie musique propre au jeu en cours. Actualisation de la page web

EDIT au 07/05: Cookie magique appelé White_Powder (référence évidente à de la cocaïne, un cookie n'ayant pas sa place dans le scénario)

>Au 07.05.2019: Rendu FINAL

>Exercice 7.43 réalisé depuis bien longtemps: les "staircase" (cage d'escaliers) sont à sens unique.

>Exercice 7.42 implémenté, ainsi que le `switch()` au lieu des nombreux `else if()`

>Exercice 7.45., 7.45.2 implémentés sans soucis

EDIT: J'ai remarqué que le mécanisme d'inventaire n'était pas fonctionnel/très buggé (fonctionne en générant un nombre assimilable à mon QI de `nullPointerException` (environ 3) >_ <), il est désormais fonctionnel. Cela m'apprendra à tester de manière légère tard la nuit/tôt le matin avant le publipostage... J'ai été aidé par Inel Ghazli & Hugo Deniau, mes camarades de 12c pour la résolution de ce soucis.

III/ Pour lancer Mission CAT!

Il faut avoir le `JDK8`, mais aussi une classe `j11.0` que j'ai utilisée pour jouer la musique (qui est la musique de Metal Gear Solid 1, le jeu pionnier dans le genre que j'ai voulu "imiter". J'en composerais une moi-même par la suite). Le jeu se lance en créant une instance de `Game` via `BlueJ` ou le `cmd`.

Aussi, il y a un script que j'ai fait dans un langage que j'affectionne particulièrement: le `AutoHotKey`. Ce n'est qu'un script qui réalise des macros claviers. Dans notre cas, il envoie bêtement "go north" "go south" etc lorsque l'on appuie sur les touches fléchées, et "go up" lorsque l'on appuie sur `Page Up`. Si l'utilisateur souhaite utiliser le script, il faut qu'il télécharge le compilateur `AHK` et lance le script.

IV/ Ce que je n'ai pas fait:

la classe `jl1.0` évidemment. Aussi, dans la class `Game`, il y a la méthode `void playMusic()`, qui permet de lancer le son. Même si *en soi*, je l'ai écrite moi même, je me suis fortement basé sur des tutoriels présents sur Internet.

Aussi, les images sont tirées d'une bête recherche google pour le nom de la room suivie de "russian" ou "soviet". Elles ont ensuite été modifiées, rognées & stylisées, et mises en forme par ma copine afin d'avoir un très agréable & cohérent thème graphique pour le jeu.

V/ NOTES:

J'ai omis la musique du `.jar`, car beaucoup trop lourde (format WAV...).

J'ai reçu de l'aide lors de la réalisation de Mission Cat!, de par mon père, informaticien de longue date (même si j'essayais de limiter ses consultations), je travaille de manière générale avec un groupe d'amis, qui me furent de grande aide pour pas mal de sessions de déboguage. Aussi, j'ai pu leur apporter mon aide de manière réciproque.

