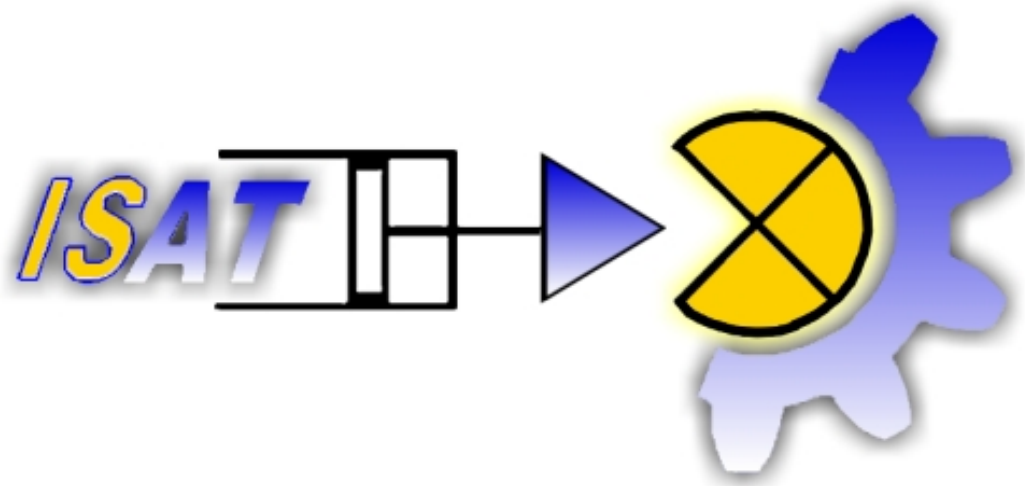


PROJET D'INFORMATIQUE APPLIQUE

Rapport de fin de parcours

HEMELEERS Emile [HE303682]
IDRISSI Sami [HE303584]
KORKUT Caner [HE303710]
ZEUKENG Ronald [HE303643]

16 Mai 2019

Table des matières

1 Introduction

Dans le cadre de l'Unité d'Enseignement *[A108] Informatique Appliqué*, il nous a été demandé de développer une application avec une interface graphique permettant d'interagir avec un utilisateur et qui possède des animations. Suite à une mise en commun de nos idées, nous avons décidé de faire une modeste version d'un système de Gestion Technique de Bâtiment.

Définition 1. *La Gestion Technique de Bâtiment (GTB) est un système informatique généralement installé dans des grands bâtiments ou dans des installations industrielles afin de superviser l'ensemble des équipements qui y sont installés.* [1]

Dans cette prémière modélisation, nous avons considéré un immeuble de 8 étages ayant chacun 2 chambres de 6 objets dits *intéractifs*. Le lancement de notre application donne accès à une page d'accueil permettant de faire le choix de l'étage à prendre le contrôle. Une fois l'étage choisi, il est possible de mouvoir un *agent* qui représente le déplacement d'une personne dans l'étage en utilisant les touches du clavier($\uparrow$, $\downarrow$, $\leftarrow$, $\rightarrow$). Certains objets (marqués d'une couleur jaune) offrent une possibilité d'intéraction avec l'agent. En effet, dans notre modélisation, on considère que lorsqu'un agent est au-dessus de ces éléments, ces derniers changent de couleur et un fichier correspondant à l'objet en question est généré de manière dynamique (ex : `etage_5_chambre2_lit.txt`). Ainsi, le document créé contient le *temps* (en secondes) que l'agent a passé sur l'objet et son nom permet de le distinguer des autres fichiers. Lorsque l'utilisateur souhaite interrompre le programme, les touches `Q`, `ESC` et le bouton *QUIT* permettent de quitter l'interface et revenir au choix d'étage. Afin d'aborder le projet de manière efficace, nous avons décidé de structurer étape par étape les tâches que nous devions accomplir :

- Croquis de l'interface utilisateur
- Choix des fonctionnalités
- Choix de la structure du code
- Implémentation du code
- Rédaction du rapport
- Présentation du Programme

1. *https ://fr.wikipedia.org*, visité le 28 février

La première étape consistant à décider du visuel de notre interface graphique, il nous a semblé indispensable de faire un croquis de ce dernier et d'y annoter les différentes dimensions [2].

Après la prise en main du module Tkinter, nous avons décidé des fonctionnalités dont disposa notre application et de la sctructure de ce dernier. Voyant une claire interaction entre l'agent et les différents objets, nous avons décidé de créer différentes classes les représentant. Ce choix de structure a été fait dans un but d'automatiser la génération d'objets et de créer des méthodes communes qui y sont associés. Bien que ce rapport ne consiste pas en une explication du script, il nous a semblé indispensable de décrire deux méthodes clefs dans ce rapport afin d'expliquer le fonctionnement du programme :

bloc_interac() : Il s'agit d'une méthode associée à l'agent. Elle sert à enclencher le mécanisme de vérification des objets intéractifs à chaque deplacement de l'agent.

agent_interact() : Il s'agit d'une méthode partagée par tous les objets interactifs. Elle sert à lancer une procédure dans le cas où l'agent chevauche un objet. Concrètement, elle se charge d'enclancher un compteur et de créer le fichier associé à l'objet.

```
def block_interact(self):

    """
    Method used to check below the agent the presence of a block.

    """

    overlapped = self.can.find_overlapping(*self.can.bbox(self.id))
    if 41 in overlapped:


        for elem in self.interface.room1.objects + self.interface.room2.objects:
            elem.agent_interact(consumption = 25)
```

FIGURE 1 – Exemple de méthode utilisée par une instance *agent*

2. Voir en annexe

2 Problèmes rencontrés lors de l'implémentation du code

Dans cette partie, nous présenterons deux problèmes que nous avons rencontrés lors de l'élaboration du projet et décrirons les pistes de solutions que nous y avons apportés.

Une première difficulté à laquelle nous avons eue affaire fut les collisions entre l'agent et les murs. En effet, il fallait intégrer une fonctionnalité qui empêche l'agent de traverser certains objets. La tâche était d'autant plus difficile puisqu'il fallait rendre ces murs infranchissables peu importe le côté du mur où l'agent se trouve. La solution apportée fut de mettre une condition sur les positions des objets. La librairie `Tkinter` permettant de comparer leurs coordonnées relatives, nous avons pu mettre une condition sur ces derniers et avons ainsi pu contourner le problème. Une autre difficulté rencontrée fut de générer le nom du fichier dynamiquement pour pouvoir distinguer les différents éléments mis en service. Pour ce faire, nous nous sommes servis du système de tags mis à notre disposition par la librairie. Les forums et certains sites spécialisés [3] ainsi que les séances prévues au projet nous ont permis d'avancer dans notre travail

3 Fonctionnement du groupe

En dehors des séances de cours prévues au projet, nous avons fait le choix de se rencontrer une semaine sur deux afin de faire un bilan de l'état d'avancement du projet et de nous fixer de nouveaux objectifs. Nous avons également pu échanger de nouvelles idées et poser des questions entre nous. Un groupe de discussion a également été créé dans le but de favoriser l'interaction entre les différents membres du groupe.

Ayant fait le choix d'une structure modulaire, nous avons réparti le codage des différents éléments au sein du groupe. Une convention a dû être adoptée pour créer les différentes classes et les intéractions entre les différents objets. Par exemple, deux personnes du groupe se chargeaient de rédiger le script de l'interface tandis que les deux autres s'occupaient des méthodes précitées plus haut.

3. https ://stackoverflow.com, http ://effbot.org, etc

4 Améliorations envisageables

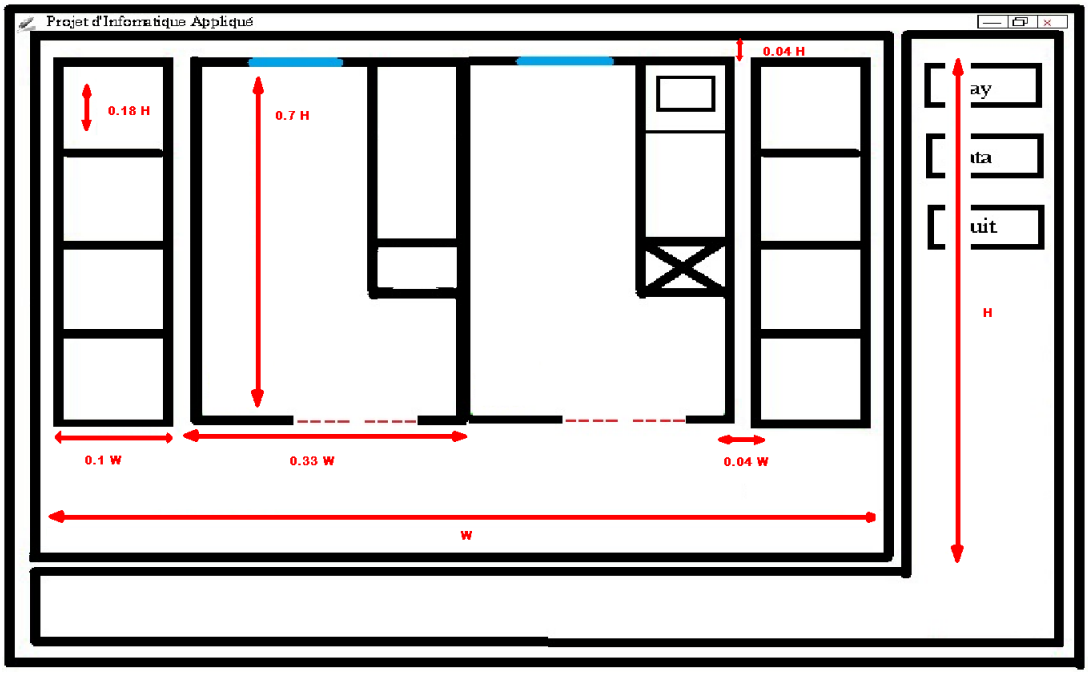
Dans une prochaine modélisation, il pourrait être envisageable d'ajouter des fonctionnalités à notre interface. Par exemple, les valeurs enregistrées dans le fichier .txt ne se limiteraient pas au temps passé dessus et pourrait contenir la valeur d'une consommation, archiver les heures d'activation, etc. Cependant, bien qu'il soit possible d'ajouter de multiples fonctionnalités à notre application ceci ne sera toujours qu'une simulation. Afin de se raprocher de la réalité, un système du même type pourrait être implémenté dans un microcontrôleur connecté à des capteurs. Une autre amélioration qu'on pourrait ajouter à notre travail est l'édition et le partage du script en temps réel. En effet, l'utilisation d'une plateforme d'édition et de versionnage collaborative aurait facilité le partage des informations et nous aurait permis d'être plus efficace.

5 Conclusion

La première chose que nous retenons de ce projet d'informatique, c'est que la maîtrise de la librairie utilisée est très importante. Une brêve prise en main du module à utiliser est nécessaire afin d'être efficace dans l'avancement du travail. Ensuite, nous avons constaté qu'il était indispensable de découper les problèmes en sous-problèmes afin de simplifier les tâches. A plusieurs reprises lors de notre travail, nous avons dû mettre nos problèmes sur papier et pener à les diviser.

Malgré les difficultés rencontrées, ce projet fut une expérience enrichissante et nous a permis de développer nos compétences en programmation.Nous pensons tous avoir appris énormément lors de l'élaboration de notre application. Cela a également stimulé notre créativité et renforcé notre esprit d'équipe. Nous retenons que les logiciels de travaux collaboratifs sont un outil indispensables dans ce genre de travail.

A Annexe 1 : Croquis de l'interface utilisateur



B Annexe 2 : Rendu final de l'application et de l'interface utilisateur