

Site web : <https://epita-the-raiders.github.io/architecture/>

Rapport de projet



Nom du projet : Raider's Architecture

Durée du projet : Février 2024 - Juin 2024

Nom du groupe : The Raiders

Chef du projet : Maël ROUSTIT

Nom des membres du groupe :

- Axel OURY
- Damien MARASSÉ
- Maël ROUSTIT
- Romain D'ANGE-BOURGUIGNON

Sommaire

I – Rappel du projet.....	4
II - Rappel : Cahier des charges.....	5
Projet S4 : Membres du groupe.....	5
Chef du projet Raider's Architecture.....	5
Les autres membres du groupe.....	5
L'objectif de Raider's Architecture.....	5
Prérequis et outils.....	6
Les détails.....	6
Rapide présentation du projet.....	6
Pourquoi ce projet ?.....	6
Pourquoi Raider's Architecture ?.....	7
Le déploiement.....	7
Les fonctionnalités et les qualités du projet.....	7
Que doit faire le projet ?.....	7
● Générer des plans.....	7
● Une interface “User-friendly”.....	7
● Un algorithme.....	8
Que devrait faire le projet ?.....	8
● Les autres algorithmes.....	8
Que pourrait faire le projet ?.....	8
● Algorithmes supplémentaires.....	8
● Page de gestion de projet.....	8
● La possibilité d'une version Windows.....	9
Liste des contraintes.....	9
Les contraintes obligatoires.....	9
Les contraintes facultatives.....	10
Répartition des tâches et estimation pour l'achèvement du projet.....	11
Note.....	11
Important.....	11
Tableau des tâches.....	11
III – Rappel des tâches à accomplir pour la première soutenance.....	12
Répartition des tâches.....	12
Prévisions faites pour le cahier des charges.....	13

Détails des tâches.....	14
Algorithme principal.....	14
Algorithmes secondaires.....	15
L'interface graphique.....	16
Gestionnaire de projet.....	16
Site Web.....	16
IV – Rappel des tâches à accomplir pour la deuxième soutenance.....	17
Répartition des tâches.....	17
Prévisions faites pour le cahier des charges.....	18
Détails des tâches.....	19
Algorithme principal.....	19
Algorithmes Display.....	20
L'interface graphique.....	21
L'interface graphique.....	21
La connexions des algorithmes à l'interface.....	23
Gestionnaire de projet.....	25
Site Web.....	25
Introduction.....	25
Le Header.....	26
Les intégrations.....	26
Téléchargements.....	26
Les designs.....	27
V – Rappel des tâches à accomplir pour la dernière soutenance.....	28
Répartition des tâches.....	28
Prévisions faites pour le cahier des charges.....	29
VI – Tâches réalisées pour cette dernière soutenance.....	30
Détails des tâches.....	30
Algorithme principal.....	30
create_rooms.....	30
create_connections.....	30
Generate_walls.....	31
Mise à jour des fonctions de sauvegarde et de chargement.....	32
Algorithme display.....	32
Recherche d'une librairie.....	32
Création du fichier svg.....	33
Création de l'image en png.....	34
L'interface graphique.....	36

Organisation.....	36
Gestionnaire de projet.....	36
Sélecteur de contraintes.....	38
L'affichage du résultat.....	41
La connexions des algorithmes à l'interface.....	41
Les problèmes de Slint.....	42
La création et l'export d'un programme main.rs.....	43
Site Web.....	43
VII - Les ressenties individuelles sur la conception du projet.....	44
Maël ROUSTIT (Chef du groupe The Raiders).....	44
Peine.....	44
Joie.....	44
Axel OURY (Membre du groupe The Raiders).....	45
Peine.....	45
Joie.....	46
Damien MARRASSÉ (Membre du groupe The Raiders).....	46
Peine.....	46
Joie.....	46
Romain D'ANGE-BOURGUIGNON (Membre du groupe The Raiders).....	47
Peine.....	47
Joie.....	47
VIII - L'intérêt du travail en groupe.....	48
IX – Conclusion.....	49
X - Annexe.....	50
Dessins d'origine.....	50
Exemple d'écran du projet.....	51

I – Rappel du projet

L'objectif de ce projet de S4 est de réaliser un logiciel où l'algorithme a une part extrêmement importante, cependant le projet reste libre.

Avant toute chose pour réaliser ce projet, nous utiliserons différents supports (Linux, Visual Code,...) qui nous permettront d'écrire l'entièreté du projet en Rust. De plus, pour avoir un suivi constant des différents avancements de chacun des membres du groupe, nous avons créé un serveur sur l'application Discord. Nous avons aussi créé un drive Google que nous complétons au fur et à mesure pour avoir un aperçu des différentes tâches que nous avons accomplies et de celles qui sont en cours de réalisation. Nous utilisons l'application Notion pour nous fixer des deadlines pour certaines parties du projet à effectuer en priorité, mais aussi pour avoir un planning simple et compréhensible pour tout le monde, pour ainsi mieux s'organiser.

Au début, nous étions partis sur un explorateur de fichiers. Celui-ci devait être composé d'une interface graphique qui aurait permis de parcourir les dossiers et les fichiers de l'ordinateur. Il devait nous permettre d'effectuer des actions, comme la création d'un nouveau fichier/dossier, la suppression, le déplacement, la recherche et plein d'autres que l'on peut retrouver sur d'autres explorateurs de fichiers existants. Nous voulions aussi implémenter une recherche par métadonnées, une prévisualisation de fichiers ainsi qu'un compresseur de fichiers. Malheureusement, ce projet a été plusieurs fois refusé, car il a été jugé comme étant un peu trop facile et il manquait également une IA ou de l'OCR à y implémenter.

Ensuite, nous sommes partis sur une IA permettant d'écrire une partition musicale à partir de l'écoute d'une musique. Elle devait être capable de reconnaître les instruments utilisés séparément, deviner les notes jouées et trouver le rythme de cette musique pour créer la parfaite partition. Le projet devait s'accompagner d'une interface graphique user-friendly. Cependant, ce projet nous a aussi été refusé, car il était considéré trop complexe à réaliser pour le temps que nous possédions pour faire ce projet.

Ainsi, nous sommes donc partis sur une nouvelle idée que nous nous efforcerons de vous présenter tout au long de ce rapport. "The Raiders" est heureux de vous montrer les débuts de leur nouveau projet dénommé "Raider's Architecture". Cette application permet à ses utilisateurs de créer des plans de maison en suivant un certain nombre de contraintes.

II - Rappel : Cahier des charges

Projet S4 : Membres du groupe

Chef du projet Raider's Architecture

Maël ROUSTIT : mael.roustit@epita.fr
GitHub : <https://github.com/Neogduh>

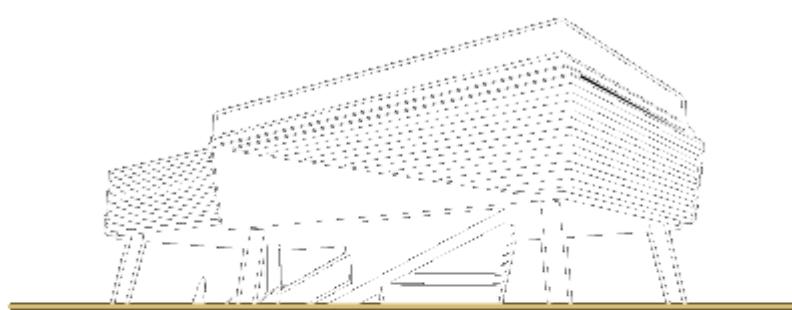
Les autres membres du groupe

Axel OURY : axel.oury@epita.fr
GitHub : <https://github.com/AxelOury>

Damien MARASSÉ : damien.marrasse@epita.fr
GitHub : <https://github.com/Dindam331>

Romain D'ANGE-BOURGUIGNON : romain.d-ange-bourguignon@epita.fr
GitHub : <https://github.com/Rom1DB>

L'objectif de Raider's Architecture



RAIDER'S

ARCHITECTURE

Pour notre projet libre du quatrième semestre, nous avons eu l'idée de programmer un créateur de plan pour des maisons ou d'appartements en utilisant le langage de programmation nommé Rust. Ce programme sera conçu pour être utilisé sur Linux à-travers l'installation de Cargo.

Prérequis et outils

Pour réaliser ce projet, nous allons utiliser différentes librairies comme slint (<https://crates.io/crates/slint>) pour construire notre interface graphique. Nous utiliserons également les bibliothèques svg (<https://crates.io/crates/svg>) et resvg (<https://crates.io/crates/resvg>) pour la conception et l'affichage des plans.

Les détails

Rapide présentation du projet

Notre projet est un logiciel qui permet de générer des plans de maison ou d'appartement après lui avoir indiqué différentes contraintes. Ces contraintes peuvent aller du nombre de chambres, à la superficie minimale du salon en passant par la plomberie et l'électricité. Il est donc possible que pour une même liste de critères nous ayons différents plans qui puissent être proposés pour aménager votre futur espace.

Pourquoi ce projet ?

Nous avons eu un certain nombre d'idées pour notre projet du quatrième semestre comme un traducteur texte en langage des signes, un logiciel qui permet la conception à l'instar d'Unity et encore d'autres idées que nous n'avons pas énormément approfondies. Ainsi nous nous sommes légitimement demandés quelle idée pourrait posséder l'intérêt algorithmique qu'il nous manquait tout en restant utile à chacun des membres de notre groupe. C'est alors qu'une idée nous est apparue presque sortie de nulle part, un explorateur de fichier pour Linux malheureusement ce projet a été jugé trop simple pour que nous puissions le réaliser. Nous avons par la suite proposé de concevoir un programme qui serait capable de créer une partition de musique à partir de notes de musique jouées et ainsi avoir la possibilité d'écouter la musique une fois la partition finie. Nous voulions faire cela car nous aimons écouter de la musique, jouer de la musique et surtout car nous ne possédons absolument pas l'oreille absolue. Cependant, cette idée était trop complexe pour être réalisée et fut donc logiquement refusée à son tour. Suite aux différents refus auxquels nous avons dû faire face, une nouvelle idée nous est

venue, celle de créer un programme qui pourrait créer des plans de maison et d'appartement d'après une liste de critère préalablement donné par le client. La suite vous la connaissez, ce projet fut accepté et porte le nom de Raider's Architecture.

Pourquoi Raider's Architecture ?

Notre groupe se prénomme “The Raiders” d'après la série de jeux Tomb Raider car nous sommes des mordus de jeux vidéos notamment de cette série-là en particulier et cela était une véritable aventure de trouver notre sujet pour ce projet. Et par la suite, comme notre projet tourne autour de l'architecture, le nom du projet “Raider's Architecture” nous est apparu assez naturellement et rapidement.

Le déploiement

Notre logiciel sera effectué en Rust comme dit précédemment et il tournera sur Linux. Cependant si nous en avons le temps, nous voudrions également le faire tourner sur Windows pour ainsi proposer ce logiciel à un maximum de personnes.

Les fonctionnalités et les qualités du projet

Que doit faire le projet ?

- **Générer des plans**

Le projet repose dans un premier temps sur le fait que l'on puisse créer un plan. On doit donc faire générer au programme une ou plusieurs images qui seront des plans plus ou moins complexes du futur bâtiment en question.

- **Une interface “User-friendly”**

Ici, dans la mesure où nous voulons que quiconque qui souhaite utiliser notre logiciel puisse le faire, qu'il soit un architecte ou non. Nous avons donc choisi de mettre en évidence les contraintes les plus importantes, tout du moins celles qui sont fondamentales. Il sera toujours possible de complexifier cela à n'importe quelle moment.

- **Un algorithme**

Évidemment, nous devons concevoir un ou plusieurs algorithmes qui nous permettront de mener à terme notre projet. Tout d'abord, il nous faut un algorithme de génération de plan qui variera selon les contraintes que donnera le client par exemple dans le cadre d'une maison avec deux chambres ou pour un appartement avec une cuisine et sept espaces libres.

Que devrait faire le projet ?

- **Les autres algorithmes**

Étant donné que nous avons pour objectif d'optimiser un maximum ainsi que de proposer un maximum de contraintes possible pour l'utilisateur nous souhaitons ajouter à notre programme un système de génération de plan prenant en compte d'autres paramètres telle que la simulation du coût total du bien ou encore la planification des travaux pour réaliser le bâtiment. Il est évident que nous devrons optimiser et complexifier l'algorithme principal.

Que pourrait faire le projet ?

- **Algorithmes supplémentaires**

Toujours dans l'optique de proposer une expérience plus immersive pour les individus qui utiliserons notre logiciel dans le futur, nous voudrions essayer de mettre en oeuvre une possibilité d'optimisation énergétique au sein du logement et d'autres détails plus ou moins essentiels dans l'optique de pourvoir avoir un bien qui répond au maximum aux attentes des personnes.

- **Page de gestion de projet**

Si nous avons le temps, nous souhaiterions proposer un menu avec lequel il serait possible de pouvoir gérer différents projets d'habitats. On voudrait également y ajouter un système de sauvegarde et de téléchargement pour récupérer les plans effectués par exemple.

- **La possibilité d'une version Windows**

Comme nous l'avons dit précédemment, nous voudrions si nous avons vraiment le temps déployer le logiciel sur Windows à l'aide d'un fichier exécutable dit aussi fichier.exe.

Liste des contraintes

Nous avons plusieurs contraintes, nous en avons beaucoup, c'est pour cela que nous avons décidé de les classer pour que l'on sache dans un premier temps celles que nous devons faire en priorité et celles qui pourront être réalisées une fois les contraintes prioritaires effectuées.

Les contraintes obligatoires

Parlons d'abord de toutes les contraintes obligatoires que nous avons à traiter. En voici la liste :

- L'utilisateur pourra choisir le nombre de pièces soit pour la maison dans son ensemble, soit par type de pièce.
- L'utilisateur pourra choisir la taille de la maison et de chaque pièce individuellement en fixant un minimum et/ou un maximum. Il est évident que l'algorithme traitera les erreurs par exemple un salon de 500m² avec une maison de 400m² cela sera impossible donc ce ne sera pas traité.
- Les différents utilisateurs pourront demander des maisons avec un ou plusieurs étages ou de plein pieds, l'algorithme imposera un emplacement pour l'escalier dans le cas d'une maison à étage
- Les utilisateurs auront la possibilité de choisir l'orientation de leur maison (orientée Sud pour la lumière par exemple).

- L'utilisateur sera en capacité de choisir s'il souhaite des portes battantes ou des portes coulissantes, un choix qui aura un impact dans la conception finale du plan.
- L'utilisateur pourra également choisir le nombre de fenêtre soit par pièce, soit pour l'ensemble de la maison directement avec un minimum et/ou un maximum.
- L'utilisateur pourra décider de l'endroit où placer sa maison ou son appartement. Ici aussi, un algorithme traitera les éventuels erreurs.

Les contraintes facultatives

Parlons maintenant de toutes les contraintes facultatives que nous avons à traiter si on a fini celles citées ci-dessus. En voici la liste :

- Il pourrait être possible de gérer les murs porteurs pour concevoir une nouvelle maison en réaménageant une maison existante.
- La nouvelle maison pourrait être optimisée sur le plan énergétique.
- Les circuits électriques ainsi que la plomberie pourraient être conçus pour être le mieux placé possible. Par exemple, on pourrait toutes les conduites d'eau sur un même mur pour éviter d'avoir trop de canalisations.
- On pourrait également prévoir l'ajout de diverses autres pièces comme le garage ou le bureau.

Répartition des tâches et estimation pour l'achèvement du projet

Note

Malheureusement en raison du manque de temps nécessaire pour fournir un projet qui puisse être accepté, nous n'étions pas en mesure de démarrer le projet pour la première soutenance, d'où le 0% ou les faibles pourcentage pour la majorité des colonnes de la première soutenance.

Important

Ce tableau est un aperçu du travail que nous avons à réaliser mais aussi une vue de ce que chaque personne sera amenée à diriger. En fait, ce que l'on veut dire par "Gestionnaire(s)" ou "Directeur(s)", il s'agit simplement de ou des personnes qui seront chargées du développement de la tâche, pas nécessairement celles qui devront la réaliser ou la développer. Nous avons à cœur de travailler en équipe, ainsi nous ne travaillerons jamais seuls, mais ensemble sous la direction du directeur de la tâche qui rapportera les avancées au chef de projet (Maël ROUSTIT). Ce même chef qui vérifiera avec les directeurs de tâche que tout avance normalement en suivant le cours des choses (le planning fixé) et que tout sera fonctionnel au final.

Tableau des tâches

	Directeur(s)	Travailleurs	Soutenance n°1	Soutenance n°2	Soutenance n°3
Algorithmes principaux	Damien/ Romain	Axel/Maël	0%	70%	100%
Algorithme secondaires	Damien/ Romain	Axel/Maël	0%	40%	100%
Interface	Axel	Damien/ Maël/ Romain	5%	50%	100%
Project Manager	Maël	Damien/ Axel/ Romain	0%	10%	100%
Site Internet	Maël	Damien/ Axel/ Romain	5%	90%	100%

III – Rappel des tâches à accomplir pour la première soutenance

Répartition des tâches

Nous avons dû nous adapter à la validation de notre cahier des charges qui a pris beaucoup de temps. C'est pour cela que notre répartition des tâches est la suivante :

Axel :

- L'interface graphique
- L'implémentation des algorithmes

Damien :

- L'implémentation de l'algorithme principal
- L'implémentation des algorithmes secondaires

Maël :

- Le site internet
- Gestionnaire de projets dans l'application

Romain :

- L'implémentation de l'algorithme principal
- L'implémentation des algorithmes secondaires

On ajoutera également qu'il n'y a pas uniquement une personne qui fera une tâche. Il s'agit juste du responsable de celle-ci sinon en pratique tout le monde travaillera sur un peu tout, le but étant de pouvoir s'aider mutuellement un maximum en cas de nécessité et à tout moment.

Prévisions faites pour le cahier des charges

Ci-dessous le tableau récapitulant les prévisions faites lors de la rédaction du cahier des charges (colonne « Période 1 ») et ce qui a réellement été fait (« Validation »).

Tâches	Période 1 (en %)	Validation
L'implémentation de l'algorithme principal	0%	✓
L'implémentation des algorithmes secondaires	0%	✓
Le site internet	5%	✓
Gestionnaire de projet	0%	✓
L'interface graphique	5%	✓

Malgré ces pourcentages très faibles, nous souhaitons préciser que ceux-ci ne représentent que l'implémentation et la conception de ces tâches. Il va de soi que malgré les 0% sur les algorithmes principaux et secondaires, nous savons parfaitement dans quelle direction nous nous orientons.

Détails des tâches

Algorithme principal

Il s'agit là de développer un voire plusieurs algorithmes de création de plan plus ou moins complexe. Un pour les maisons, l'autre pour les appartements par exemple. Pour ainsi permettre la génération de plans simples qui donne ainsi un ou plusieurs aperçus sur ce que l'on veut si c'est possible (ça doit malgré tout respecter les normes d'habitats françaises). On voudrait au final que cet algorithme nous renvoie quelque chose qui ressemble au plan de conception des maisons et/ou des appartements.

Pour une maison nous pourrions avoir ceci :



Et pour un appartement nous pourrions obtenir cela :



Algorithmes secondaires

Nous avons également pensé à voir un peu plus loin à-travers d'autres algorithmes pour rendre notre projet le plus complet possible. Nous sommes dit que nous pouvons rajouter :

- Un algorithme qui simule le coût de conception du bien potentiellement même par rapport à sa situation géographique (le mètre carré est plus cher à Paris qu'à Bizeneuille par exemple)
- Un algorithme qui calculerait le temps de travail nécessaire pour construire entièrement le bien en question avec aussi une certaine marge d'erreur qui sera calculée à partir de la première estimation.
- Un algorithme d'optimisation énergétique pour mieux placer les différentes pièces (par exemple privilégier la cuisine à côté de la buanderie pour limiter la taille des canalisations d'eau)
- Et peut-être bien plus tard, un algorithme qui pourrait convertir le plan en un modèle 3D (mais ça c'est si et seulement si nous avons tout fait correctement avant)

L'interface graphique

Il nous faut, pour que notre logiciel soit simple d'utilisation, une interface graphique qui permet à n'importe qui de commencer un projet plus ou moins complexe. Qui pourrait potentiellement ressembler à quelque chose comme cela :

Taille maison (en mètre carré)	<input type="text"/>					
Nombre de toilettes	<input type="text"/>	Jardin	<input type="radio"/>	Oui	<input type="radio"/>	Non
Nombre de chambre	<input type="text"/>	Terrasse	<input type="radio"/>	Oui	<input type="radio"/>	Non
Nombre de salle de bains	<input type="text"/>	Piscine	<input type="radio"/>	Oui	<input type="radio"/>	Non
Taille (minimum) de la cuisine	<input type="text"/>	Cuisine ouverte	<input type="radio"/>	Oui	<input type="radio"/>	Non
Taille (minimum) du salon	<input type="text"/>	Suite parentale	<input type="radio"/>	Oui	<input type="radio"/>	Non
Nombre d'espaces libres	<input type="text"/>	Garage	<input type="radio"/>	Oui	<input type="radio"/>	Non
Taille (minimum) d'une chambre	<input type="text"/>	Ascenseur	<input type="radio"/>	Oui	<input type="radio"/>	Non
Nombre d'étages (0 = plein pied)	<input type="text"/>	Escalier	<input type="radio"/>	Oui	<input type="radio"/>	Non


RAIDER'S
ARCHITECTURE

CRÉATION

Gestionnaire de projet

Dans une version avancée de notre projet, nous aimerais intégrer un gestionnaire de projet couplé à un système de sauvegarde ce qui permettrait à l'utilisateur de pouvoir reprendre et modifier un des ces anciens projets à tout moment.

Site Web

Dans le cadre du projet, on nous demande un site web. Pour cette première période il reste basique mais le voici : <https://epita-the-raiders.github.io/architecture/>.

IV – Rappel des tâches à accomplir pour la deuxième soutenance

Répartition des tâches

Nous avons dû nous adapter à la validation de notre cahier des charges qui a pris beaucoup de temps. C'est pour cela que notre répartition des tâches est la suivante :

Axel :

- L'interface graphique
- La connexions des algorithmes à l'interface

Damien :

- L'implémentation et optimisation de l'algorithme principal (création du plan)

Maël :

- Le site internet
- Gestionnaire de projets dans l'application
- Gestion du projet

Romain :

- L'implémentation des algorithmes secondaires (Display dans un premier temps, et optimisation dans un second)

On ajoutera également qu'il n'y a pas uniquement une personne qui fera une tâche. Il s'agit juste du responsable de celle-ci sinon en pratique tout le monde travaillera sur un peu tout, le but étant de pouvoir s'aider mutuellement un maximum en cas de nécessité et à tout moment.

Prévisions faites pour le cahier des charges

Ci-dessous le tableau récapitulant les prévisions faites lors de la rédaction du cahier des charges (colonne « Période 2 ») et ce qui a réellement été fait (« Validation »).

Tâches	Période 2 (en %)	Validation
L'implémentation de l'algorithme principal	70%	✓
L'implémentation des algorithmes secondaires	40%	✓
Le site internet	90%	✓
Gestionnaire de projet	10%	✓
L'interface graphique	50%	✓

Malgré des pourcentages très faibles à la première soutenance, dû à un retard sur le cahier des charges, nous sommes dans les temps par rapport à ce qui a été annoncé. Nous pensons également pouvoir finir le projet dans les temps.

Détails des tâches

Algorithme principal

Pour structurer notre projet, nous avons adopté une approche basée sur les structures (**structs**), qui nous permettent de représenter la maison et ses pièces de manière organisée. Cette méthodologie offre la flexibilité nécessaire pour intégrer des paramètres supplémentaires, tels que des contraintes spécifiques, tout en développant des algorithmes efficaces pour la gestion et la manipulation de ces données.

La structure principale de notre système est la **House**, qui contient actuellement une liste de pièces appelées **rooms**. Nous envisageons d'ajouter ultérieurement des paramètres tels que le nombre d'étages de la maison (**floors**) et même l'orientation de la maison pour optimiser l'emplacement des fenêtres et éviter une exposition excessive au soleil.

Chaque pièce est représentée par la structure **Room**, qui inclut le type de la pièce (salon, chambre, cuisine, etc.) ainsi que ses dimensions. Nous prévoyons également d'intégrer des fonctionnalités telles que la gestion des connexions entre les pièces, la présence de portes et de fenêtres, ainsi que leur positionnement dans l'espace.

Du point de vue des algorithmes, nous avons déjà mis en place un système de sauvegarde fonctionnel qui peut être adapté pour répondre à différentes contraintes spécifiques. Le programme de sauvegarde convertit une instance de **House** en un fichier texte au format spécifique, où chaque ligne représente une pièce avec ses dimensions.

Lorsque l'utilisateur utilise l'interface graphique pour créer une maison, celle-ci génère également un fichier texte conforme au format attendu, qui est ensuite chargé par la fonction de chargement. Cette fonction lit le fichier ligne par ligne, créant ainsi les instances de **Room** correspondantes, puis assemble une **House** contenant la liste de ces pièces pour être utilisée dans le programme.

Actuellement, l'utilisateur peut créer une maison en entrant manuellement les dimensions de chaque pièce. Cependant, notre objectif final est de développer un algorithme capable de générer automatiquement une disposition de pièces optimale, en tenant compte de diverses contraintes. Ceci nécessitera le développement de plusieurs programmes et une recherche approfondie pour déterminer la meilleure répartition des espaces, entre autres considérations.

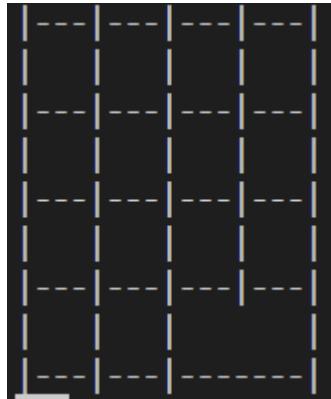
Algorithmes Display

Pour cette première soutenance, nous avons décidé de nous focaliser sur le fait d'avoir un élément complètement fini, plutôt que de s'éparpiller sur pleins de contraintes sans pour autant les réaliser à 100%. Donc nous avons entrepris de travailler complètement sur la contrainte qui est le nombre de pièces, afin d'avoir une ébauche de plan "viable".

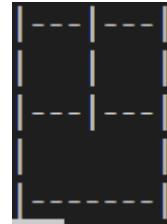
Afin de représenter de la manière la plus claire possible les pièces de notre plan, nous avons opté pour le fait de dessiner les pièces tels des carrés (à base de 'l' et de '-' afin de faire les contours de ces carrés), et de les coller afin d'avoir un rendu ressemblant à un véritable plan.

La solution de facilité (qui a été implémentée au tout début de cette période) est de réaliser une maison toute en longueur, c'est-à-dire en collant toutes les pièces côté à côté dans le même sens. Le rendu donnait donc un espèce de rectangle très long, très loin d'un vrai plan de maison.

Il est donc paru évident que nous devions implémenter une méthode différente afin d'avoir un plan cohérent. Pour cela, nous avons opté pour un algorithme permettant de réaliser un plan bien plus cohérent, c'est-à-dire avec des pièces pouvant être collées de la gauche vers la droite et également du haut vers le bas. Maintenant, l'algorithme réalise des plans cohérents, avec des pièces organisées de manière réaliste. Voici quelques exemples pour imager ces propos qui sont probablement assez flou sans support visuel :



Plan de 14 pièces



Plan de 3 pièces



Plan de 9 pièces

L'algorithme n'est pour le moment pas modulable, c'est-à-dire qu'il ne prend pas en compte les contraintes de taille, de forme ou encore de placement. Cela viendra plus tard, dans les versions futures, et cet algorithme sera perpétuellement amélioré afin d'avoir un rendu propre et répondant à toutes les contraintes auxquelles nous avons pensé.

L'interface graphique

L'interface graphique

Pour que notre idée puisse prendre vie, nous nous devons de posséder une interface simple et compréhensible par tout le monde. Pour réaliser cette interface, nous avons décidé d'utiliser la bibliothèque `slint` car nous avons appris que celle-ci nous permettait de faire une multitude de choses essentielles à la réalisation de ce projet. Cependant sa prise en main ne fut pas aussi aisée qu'espéré. En effet, son installation fut extrêmement complexe (surtout pour coder l'interface sur MacOS), on a dû s'y reprendre plusieurs fois pour réussir son installation pour pouvoir être utilisé. Nous avons également trouvé que `Slint` était peu documenté ce qu'on veut dire par là c'est que n'avons trouvé que trop peu d'exemples pour aiguiller dans cette tâche.

On a réussi selon nous à atteindre les 50% qu'on s'était fixé. On a une interface qui possède tous les critères que nous voulions proposer aux futures utilisateurs de notre logiciel.

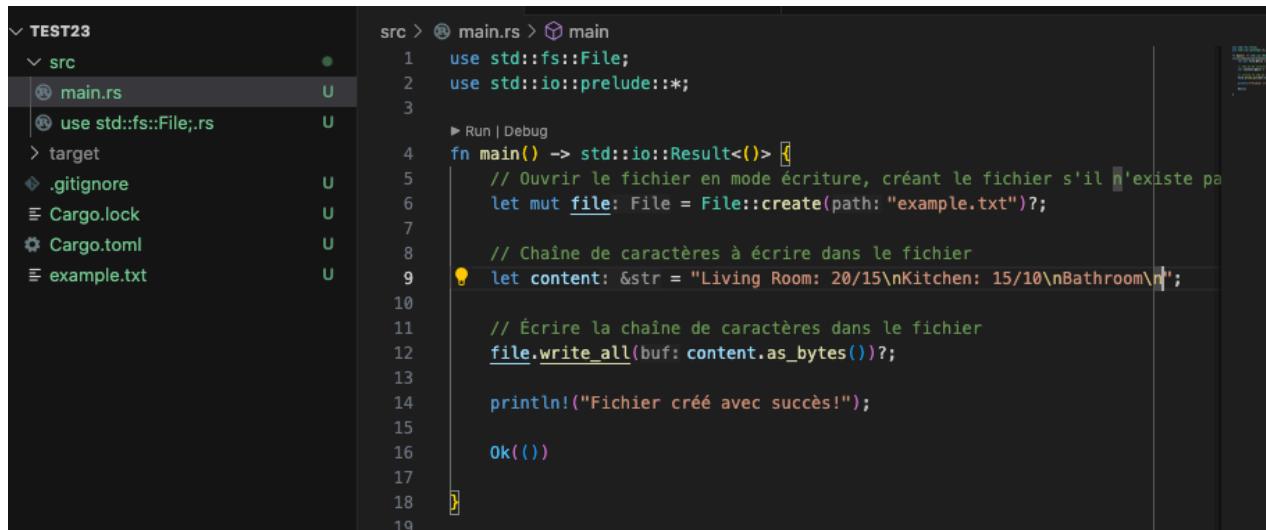


Comme vous pouvez le constater, nous avons décidé de tout écrire en anglais pour que ce soit compréhensible pour un maximum de personnes dans le monde et que tout le monde puisse l'utiliser assez facilement. On alterne entre les "LineEdit" pour pouvoir récolter les nombres qui sont des strings et ainsi les récupérer pour les utiliser dans nos algorithmes. Les "CheckBox" quant à elles seront utilisées de la manière suivante, ces boutons renvoie "True" or "False" et de ce fait, on peut se servir de ces issues booléennes pour ajouter ou non l'un des critères que nous proposons. Cependant nous avons rencontré différents problèmes qu'à l'heure actuelle nous n'avons pas pu résoudre et qui empêchent l'interface de tourner sur Linux. Voici l'erreur qui s'affiche : "Error: Other("Could not initialize backend.\nError from Winit backend: Error initializing winit event loop: the requested operation is not supported by Winit\nNo backends configured.")"

On espère que le jour de la soutenance ce problème sera résolu pour que vous puissiez regarder l'interface s'exécuter sur Linux. Cela étant dit, nous nous sommes plus concentrés sur l'esthétique de l'interface que sur sa fonctionnalité et nous faisons tout pour régler ce problème majeur en priorité.

La connexions des algorithmes à l'interface

Concernant la connexion des algorithmes avec l'interface, nous n'avons pas pu les avancer autant qu'on voulait à cause du problème qui est apparu. Nous avons tout de même commencé les fondations, on a commencé à écrire les fonctions pour pouvoir faire le pont entre l'interface et les algorithmes. En voici quelques images des algos que nous avons commencé à faire pour créer une connexion entre les algorithmes et l'interface.



```
src > main.rs > main
1 use std::fs::File;
2 use std::io::prelude::*;
3
4 fn main() -> std::io::Result<()> {
5     // Ouvrir le fichier en mode écriture, créant le fichier s'il n'existe pas
6     let mut file: File = File::create(path: "example.txt")?;
7
8     // Chaine de caractères à écrire dans le fichier
9     let content: &str = "Living Room: 20/15\nKitchen: 15/10\nBathroom\n";
10
11    // Écrire la chaîne de caractères dans le fichier
12    file.write_all(buf: content.as_bytes())?;
13
14    println!("Fichier créé avec succès!");
15
16    Ok(())
17
18 }
```

Cet algorithme va permettre la génération d'un document en **.txt** qui pourra être exploité pour faire fonctionner nos autres algorithmes.

The screenshot shows a code editor interface with a toolbar at the top featuring 'RUN ▶' and other buttons. The main area displays a Rust program. The code uses the 'rand' crate to generate a random number and prints it to the console. It includes comments in French explaining the steps: creating a generator, reading user input, parsing it to a float, checking if it's negative, generating a random number between 1 and 100, and printing the result.

```
1 use rand::Rng;
2
3 fn main() {
4     // Créez un générateur de nombres aléatoires
5     println!("Veuillez saisir un nombre :");
6
7     // Créez une nouvelle instance de `std::io::Stdin` pour lire l'entrée de l'utilisateur
8     let input = String::from(" 27 ");
9
10    // Convertir la chaîne entrée en nombre entier
11    let number: f32 = match input.trim().parse() {
12        Ok(n) => n,
13        Err(_) => {
14            println!("Ce n'est pas un nombre valide !");
15            return;
16        }
17    };
18    if number as f32 - 2.0 < 0.0
19    {
20        println!("Impossible");
21        return;
22    }
23    let mut rng = rand::thread_rng();
24
25    // Générez un nombre aléatoire entre 1 et 100
26    let random_number = rng.gen_range(number-number/2.0..=number-2.0);
27
28    // Affichez le nombre aléatoire
29    println!("Nombre aléatoire: {} ", (random_number*100.0).round()/100.0);
30
31 }
```

Cet algorithme qu'en a lui est une sorte de convertisseur en fait pour augmenter de manière significative l'aléatoire dans la génération de plan ainsi qu'être sur d'avoir les paramètres qu'il faut pour faire fonctionner la génération de plan. Ces deux algorithmes n'ont pas été implémentés car ils n'ont pas étaient totalement terminés dû notamment à un certain manque de temps et le fameux problème que nous avons dû nous préoccuper en urgence.

On a malgré tout les obstacles rencontrés sur ce chemin, nous sommes heureux de vous dire qu'on a réussi à réaliser à plus de 50% l'interface graphique.

Gestionnaire de projet

Dans les 10% prévus pour cette soutenance, nous n'avions pas pour objectif d'implémenter en lui-même le gestionnaire de projet, mais des outils nous permettant de le faire plus naturellement lors de la prochaine période. Nous avons donc conçu un système de sauvegarde de fichiers, c'est-à-dire qu'il est possible de sauvegarder 2 choses, les données rentrées dans l'interface, comme le nombre de pièces, leur taille et les différentes autres contraintes, afin que dans le futur, on puisse recharger ces données et créer de nouveaux plans depuis celles-ci où les modifier.

Mais il est également possible de sauvegarder un plan de maison, afin de pouvoir le recharger plus tard, et tout ça, depuis le gestionnaire de projet. Le but de ce dernier sera donc de pouvoir avoir un accès direct à toutes les créations d'un individu sur notre application. Il devrait être possible de supprimer des projets pour libérer de l'espace mémoire même si le type de sauvegarde choisi, n'est pas très gourmand, nous utilisons des fichiers **.txt**. A terme quand vous ouvrirez l'interface, vous pourrez soit gérer, soit charger un projet de plan existant, ou en créer un nouveau.

Site Web

Introduction

Pour avoir l'entièreté de la 3ème et dernière période de travail consacrée au projet en lui-même, Nous avons décidé de faire l'entièreté du site web sur la 2nd période. Le cahier des charges indique 90% car il n'est pas 100% fini, Nous ajouterons des choses pour le rendu final.

Le site est disponible à ce lien : <https://epita-the-raiders.github.io/architecture/>

Le Header

La première chose qui a été faite sur le site est la création d'un "header". Déjà présent lors de la soutenance précédente avec 1 seul bouton qui permettait simplement de passer de index.html à index_en.html, nous avons ajouté 5 boutons pour que la navigation entre les pages soit plus fluide. Nous avons donc ajouté 4 pages et leurs versions anglaises pour ne pas tout avoir sur la même page. Pour que le switch de langue soit plus simple nous avons parser le lien pour en déduire est-ce qu'il contient "_en" ou non. S'il contient "_en", nous appelons la page du même nom sans "_en", sinon nous passons la page du même nom en ajoutant "_en". De même pour le passage d'une page à une autre, si on est sur une page anglais, l'appuie sur un bouton de redirection appellera la page anglaise et de même pour les pages françaises.

Les intégrations

Pour plus de réalisme et de fonctionnalités, nous avons ajouté un module pour parser et afficher du markdown directement sur le site. J'ai ensuite créé un "objet" pop-up pour afficher le markdown, qui a l'appuis sur le bouton d'ouverture, ouvre une pop-up avec le contenu demandé. Vous pouvez la fermer via un clique à côté de cette dernière, l'appuie de la touche "echap" ou encore un croix placée en haut à droite de la pop-up. J'ai ensuite ré-utilisé la pop-up pour les rapport, produit en pdf j'ai utilisé un **<iframe>** pour les afficher dans ma pop-up.

Téléchargements

Dans la mesure où nous voulions proposer 2 versions du projet, il nous fallait au moins 2 boutons, un pour Linux et l'autre pour Windows. Pour faire ça, nous avons créé une sorte de "mise en avant", quand un des deux boutons est mis en avant, il est grossi comparé à l'autre qui rapetit, le logo correspondant apparaît et une animation quand vous gardez votre souris dessus a été appliquée. Nous avons également ajouté la mise en avant automatique en fonction de votre système d'exploitation, en lisant les informations du navigateur, cela deduit si vous êtes sur Linux ou Windows et met en avant la version correspondante.

Les designs

Nous avons fait un gros travail de design sur le site pour qu'il soit le plus lisible possible et agréable à regarder. J'ai tenté d'ajouter des animations sur les boutons, notamment ceux du header, les boutons GitHub, etc. Pour cela je me suis rendu sur un site web qui proposait des boutons en HTML/CSS et nous les avons modifiés pour qu'ils collent à notre site. Nous avons choisi une couleur bleue pas trop flash qui permet de faire un mix entre un thème clair et un thème sombre.

V – Rappel des tâches à accomplir pour la dernière soutenance

Répartition des tâches

Nous avons dû nous adapter à la validation de notre cahier des charges qui a pris beaucoup de temps. C'est pour cela que notre répartition des tâches est la suivante :

Axel :

- L'interface graphique
- La connexions des algorithmes à l'interface

Damien :

- L'implémentation et optimisation de l'algorithme principal (création du plan)

Maël :

- Le site internet
- Gestionnaires de projets dans l'application
- Gestion du projet

Romain :

- L'implémentation des algorithmes secondaires (Display dans un premier temps, et optimisation dans un second)

On ajoutera également qu'il n'y a pas uniquement une personne qui fera une tâche. Il s'agit juste du responsable de celle-ci sinon en pratique tout le monde travaillera sur un peu tout, le but étant de pouvoir s'aider mutuellement un maximum en cas de nécessité et à tout moment.

Prévisions faites pour le cahier des charges

Ci-dessous le tableau récapitulant les prévisions faites lors de la rédaction du cahier des charges (colonne « Période 3 ») et ce qui a réellement été fait (« Validation »).

Tâches	Période 3 (en %)	Validation
L'implémentation de l'algorithme principal	100%	✓
L'implémentation des algorithmes secondaires	100%	✓
Le site internet	100%	✓
Gestionnaire de projet	100%	✓
L'interface graphique	100%	✓

Malgré des petits problèmes de librairie, dû à un retard global sur le projet, commençant par le cahier des charges et notre souhait de ne pas prendre des trucs classiques, nous avons réussi à être dans les temps par rapport à ce qui a été annoncé puis que nous avons livré un produit fonctionnel.

VI – Tâches réalisées pour cette dernière soutenance

Détails des tâches

Algorithme principal

Pour la dernière soutenance et le rendu de fin de projet, nous avons terminé d'implémenter l'algorithme principal en ajoutant 3 nouvelles fonctions et en améliorant celles qui s'occupaient de la sauvegarde et du chargement :

create_rooms

La fonction `create_rooms` est responsable de l'initialisation des pièces dans la maison. Elle remplit la maison avec un ensemble prédéfini de pièces, chacune attribuée à un type spécifique (comme Salon, Cuisine ou Chambre) avec des dimensions optionnelles (largeur, longueur) et des coordonnées de position. Cette fonction garantit que la maison possède une structure de base avec les pièces essentielles avant que des connexions ou des murs ne soient ajoutés. La fonction parcourt une liste prédéfinie de types de pièces et crée de nouvelles instances de pièces avec les propriétés données, les ajoutant à la collection de pièces de la maison.

create_connections

La fonction `create_connections` établit des connexions logiques entre les pièces de la maison. Elle vérifie si une pièce peut avoir plusieurs connexions et connecte les pièces en conséquence pour s'assurer que la maison est entièrement accessible. La fonction parcourt chaque pièce et détermine les connexions potentielles en fonction de règles prédéfinies. Elle ajoute des portes entre les pièces connectées, garantissant que chaque pièce est accessible depuis d'autres, soit directement, soit par une série de connexions. Cette fonction est cruciale pour définir la disposition et le flux de la maison.

Generate_walls

La fonction `generate_walls` génère les murs et les fenêtres pour chaque pièce de la maison. En fonction des dimensions et des positions des pièces, elle calcule les coordonnées des murs et des fenêtres, garantissant qu'ils s'intègrent logiquement dans l'espace donné. La fonction crée quatre murs pour chaque pièce (haut, bas, gauche et droite) et attribue des coordonnées pour les fenêtres en fonction de la direction de la façade de la pièce. Des cas particuliers sont gérés où aucun mur n'est généré entre des pièces spécifiques connectées (par exemple, aucun mur entre la Cuisine et le Salon). Cette fonction garantit que la maison a une structure réaliste et fonctionnelle avec des limites et des ouvertures correctement définies.

```
Room: LivingRoom (LivingRoom) (4.2221737 x 6.7258024)
Connected to Kitchen with a sliding door
Connected to Bathroom with a sliding door
Connected to Bedroom with a sliding door
Connected to Empty with a sliding door

Wall from (5.466833, 22.570202) to (9.689007, 22.570202)
Wall from (9.689007, 22.570202) to (9.689007, 29.296005)
Wall from (9.689007, 29.296005) to (5.466833, 29.296005)
Wall from (5.466833, 29.296005) to (5.466833, 22.570202)
Window from (5.466833, 22.570202) to (6.466833, 22.570202)
Window from (8.689007, 22.570202) to (9.689007, 22.570202)
Room: Kitchen (Kitchen) (7.2552176 x 10.597157)
Connected to LivingRoom with a sliding door

Wall from (13.904672, 28.987034) to (21.15989, 28.987034)
Wall from (21.15989, 28.987034) to (21.15989, 39.58419)
Wall from (21.15989, 39.58419) to (13.904672, 39.58419)
Wall from (13.904672, 39.58419) to (13.904672, 28.987034)
Window from (13.904672, 28.987034) to (14.904672, 28.987034)
Window from (20.15989, 28.987034) to (21.15989, 28.987034)
```

Exemple de maison créée

Mise à jour des fonctions de sauvegarde et de chargement

Les fonctions `house_to_file` et `file_to_house` ont été mises à jour pour gérer la structure actuelle de l'objet `House`. La fonction `house_to_file` écrit les dimensions de la maison, la direction de la façade, et les détails de chaque pièce, mur, porte et fenêtre dans un fichier. Elle garantit que toutes les informations pertinentes sont stockées dans un format lisible. La fonction `file_to_house` lit ces informations à partir d'un fichier, reconstruisant la maison avec tous ses composants. Elle gère correctement les valeurs optionnelles telles que les dimensions et les positions des pièces, en s'assurant que les valeurs `None` sont traitées correctement. Cette mise à jour garantit que l'état de la maison peut être sauvegardé et restauré avec précision, en maintenant l'intégrité de la structure et de la disposition de la maison.

```
House: Some(20.0)/Some(30.0)/North
Room: LivingRoom:LivingRoom/Some(28.377169)/Some(12.480785)/None
Wall: (17.30257,1.2823713)-(29.783356,1.2823713)
Wall: (29.783356,1.2823713)-(29.783356,29.65954)
Wall: (29.783356,29.65954)-(17.30257,29.65954)
Wall: (17.30257,29.65954)-(17.30257,1.2823713)
Window: (17.30257,1.2823713)-(18.30257,1.2823713)
Window: (28.783356,1.2823713)-(29.783356,1.2823713)
Room: Kitchen:Kitchen/Some(13.311487)/Some(14.763635)/None
Window: (13.882301,28.471481)-(14.882301,28.471481)
Window: (27.645935,28.471481)-(28.645935,28.471481)
```

Exemple de fichier .txt créé

Algorithme display

Recherche d'une librairie

La recherche d'une librairie graphique complètement adaptée à notre utilisation a vraiment ressemblé à la recherche d'une aiguille dans une botte de foin pour reprendre l'expression. Nous avons trouvé de nombreuses librairies qui pouvaient se rapprocher de ce que nous voulions, sans pour autant être parfaites.

Notre premier choix a été la librairie Turtle, l'avantage étant que nous la connaissions déjà étant donné que nous l'avions déjà tous utilisé en Python durant les cours de SNT/NSI au lycée. Nous ne nous sommes pas penchés sur d'autres alternatives car Turtle est bien documenté et qu'elle nous aurait permis de gagner un temps d'apprentissage conséquent dû à notre expérience passée sur cette librairie. Notre professeur nous ayant refusé cette bibliothèque (en nous faisant par la suite des propositions), nous avons alors recherché plus en détail les autres possibilités. Après avoir comparé plusieurs librairies, notre choix s'est porté vers la librairie Plotters.

Cette librairie était très bien documentée, elle semblait également assez abordable en termes de difficulté de compréhension et était selon nous plutôt bien adaptée à ce que nous souhaitons faire malgré le fait qu'elle soit légèrement orientée vers la création de graphiques.

Les autres options que nous avions envisagées étaient les librairies Turtle, Imageproc, Keyframe et Piston (et quelques autres comme Nannou, qui n'étaient pas vraiment adaptées). Après un autre refus, notre professeur nous a proposé les librairies svg et resvg. Les librairies semblent plutôt adaptées et aussi par peur de la deadline, nous avons donc choisi de nous orienter vers ces 2 librairies.

Création du fichier svg

La première étape de la création de notre plan est donc de faire un fichier en .svg qui transforment les murs, fenêtres, pièces et portes en un dessin de plan. Pour cela, nous avons dû transformer nos diverses struct House, Room, Door, Point, Walls ou encore Windows (fenêtre) afin de récupérer toutes les informations utiles à la création du plan. Pour cela, nous avons codé la fonction `create_svg(mut house)` qui prend en paramètre une house de type House et qui crée un fichier `house_plan.svg`.

La fonction crée donc un document dans lequel est ajouté dans l'ordre les murs, les fenêtres, les portes et enfin le nom de chacune des pièces. Ensuite, ce document est sauvegardé dans le fichier `house_plan.svg`. Ce fichier doit être supprimé après chaque utilisation, sinon ce fichier ne sera jamais modifié.

Voilà à quoi ressemble le fichier house_plan.svg sur un appel avec des dimensions et des emplacements aléatoires :

```
<svg height="900" viewBox="0 0 1600 900" width="1600" xmlns="http://www.w3.org/2000/svg">
<path d="M0,0 L388.82928,0" fill="none" stroke="black" stroke-width="2"/>
<path d="M388.82928,0 L388.82928,504.0266" fill="none" stroke="black" stroke-width="2"/>
<path d="M388.82928,504.0266 L0,504.0266" fill="none" stroke="black" stroke-width="2"/>
<path d="M0,504.0266 L0,0" fill="none" stroke="black" stroke-width="2"/>
<path d="M0,0 L100,0" fill="none" stroke="blue" stroke-width="2"/>
<path d="M288.82928,0 L388.82928,0" fill="none" stroke="blue" stroke-width="2"/>
<path d="M388.82928,0 L488.82928,0" fill="none" stroke="blue" stroke-width="2"/>
<path d="M575.0989,0 L675.0989,0" fill="none" stroke="blue" stroke-width="2"/>
<path d="M0,504.0266 L303.43207,504.0266" fill="none" stroke="black" stroke-width="2"/>
<path d="M303.43207,504.0266 L303.43207,1043.7891" fill="none" stroke="black" stroke-width="2"/>
<path d="M303.43207,1043.7891 L0,1043.7891" fill="none" stroke="black" stroke-width="2"/>
<path d="M0,1043.7891 L0,504.0266" fill="none" stroke="black" stroke-width="2"/>
<path d="M0,504.0266 L100,504.0266" fill="none" stroke="blue" stroke-width="2"/>
<path d="M303.43207,504.0266 L303.43207,504.0266" fill="none" stroke="blue" stroke-width="2"/>

<path d="M355.0431,1484.3967 L455.0431,1484.3967" fill="none" stroke="blue" stroke-width="2"/>
<path d="M668.8467,1484.3967 L768.8467,1484.3967" fill="none" stroke="blue" stroke-width="2"/>
<text font-size="20" text-anchor="middle" x="0" y="0">
LivingRoom
</text>
<text font-size="20" text-anchor="middle" x="3.8882928" y="0">
Kitchen
</text>
<text font-size="20" text-anchor="middle" x="0" y="5.040266">
Bathroom
</text>
<text font-size="20" text-anchor="middle" x="3.0343206" y="5.040266">
Bedroom
</text>
<text font-size="20" text-anchor="middle" x="0" y="10.43789">
Hall
</text>
<text font-size="20" text-anchor="middle" x="3.4229875" y="10.43789">
Toilet
</text>
```

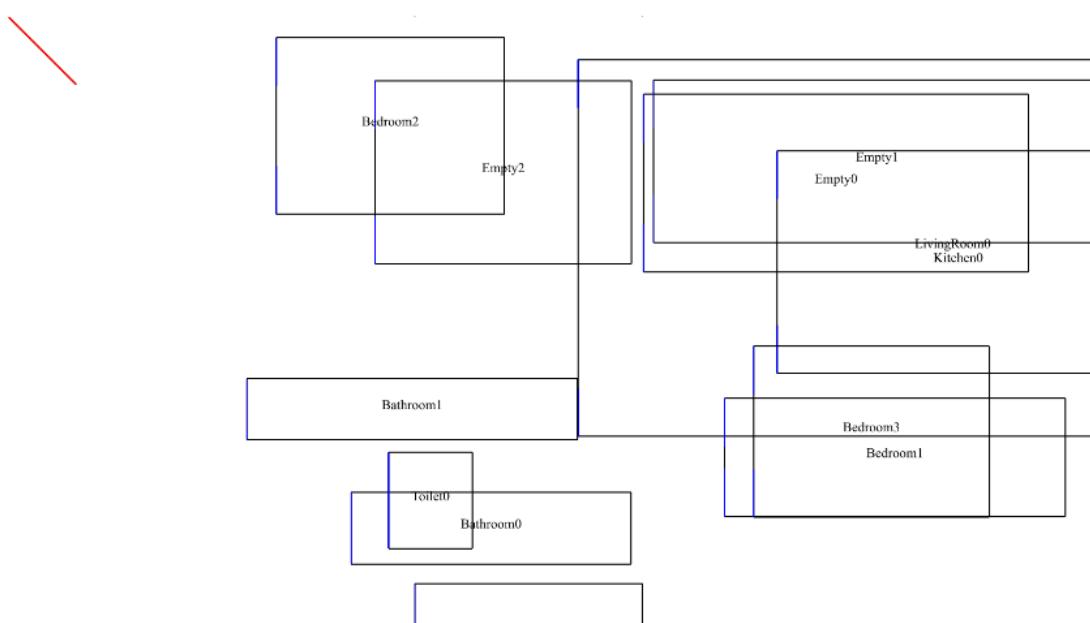
Création de l'image en png

Après avoir généré notre fichier en .svg à l'aide de la librairie svg, la dernière étape était de transformer ce fichier en une image en .png. Pour cela, nous avons dû utiliser la librairie resvg afin de créer notre fonction. Nous avons tout d'abord créé une première version de cette fonction, ne prenant aucun paramètre (*render_svg_to_png()*). Cette fonction était appelé dans le *main()* et était utilisé en écrivant ‘cargo run <fichier.svg>’, ou alors ‘cargo run <fichier.svg> -z <Zoom>’ (cette deuxième version permettant de zoomer sur l'image directement lors de sa création).

Cependant, cette version n'était pas hyper optimale car il fallait d'abord appeler *create_svg()*, puis écrire cargo run house_plan.svg. Cela causait des problèmes pour exécuter notre projet dans sa globalité. Pour pallier ce problème, nous avons implémenté la même fonction, mais prenant en paramètre le chemin du fichier house_plan.svg (*render_svg_to_png(svg_file)*). Désormais, il est possible d'exécuter le projet complet d'une seule traite, ce qui est bien plus optimal.



Résultat d'une maison avant
la soutenance

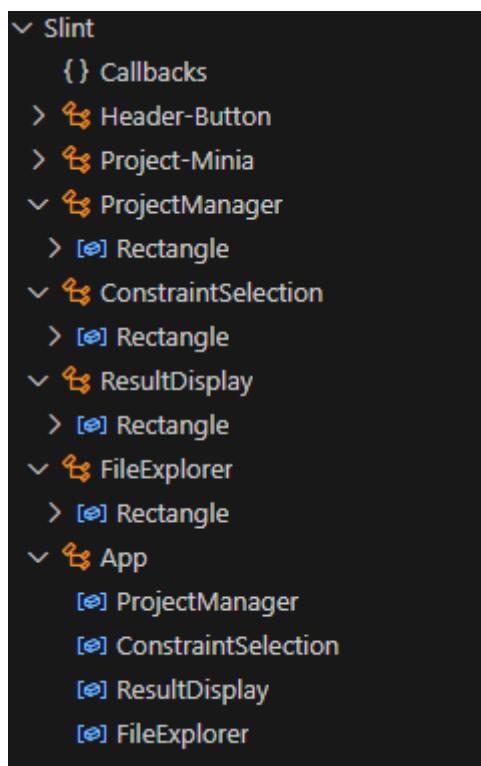


Résultat d'une maison après
la soutenance

L'interface graphique

Organisation

Au début de ce projet, nous ne savions pas comment faire pour avoir 3 "fenêtres" différentes dans Slint, donc notre principal objectif a été de comprendre comment faire. Nous avons donc pensé à un système de switch entre les différents composants. On crée 3 **component** Slint, un par truc afficher, et un **component** que l'on export et grâce à des boutons on fait des switch entre l'état visible ou non des ces derniers. Ce sont des **Rectangle** qui font tous la taille de la fenêtre et donc on a un switch fluide entre tous les différents affichages.

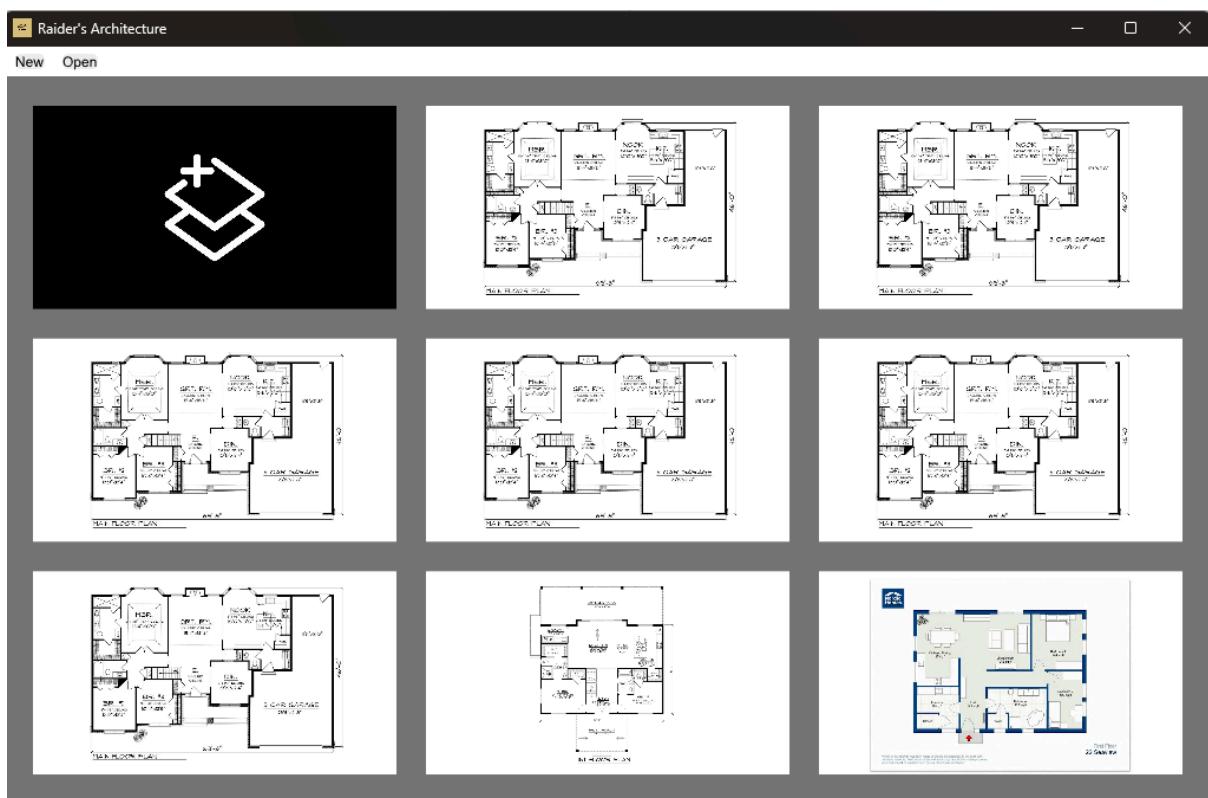


L'outline du projet

Gestionnaire de projet

Un gestionnaire de projet, un défi qui représente bien notre premier projet puisque c'est limite un petit explorateur de fichier (un petit clin d'œil à première suggestion de projet). Première étape créer l'interface, premier problème, si les images ne font pas la même taille cela ne fonctionne pas du tout, donc on va faire avec des images de même taille. Ensuite il a fallu faire en sorte que ce soit

dynamique à chaque ouverture, on va donc chercher les images dans le dossier de Saves et on va les charger. Ensuite il faut bien faire attention que quand on clique sur une image vide, cela ne fasse pas planter le programme. Après que tout cela soit fait, il reste à pouvoir charger n'importe quelle save, pour ça quoi de mieux qu'un explorateur de fichier (encore un petit clin d'œil à première suggestion de projet). Bon en soit vous demandez un chiffre de sauvegarde et vous cela vous dit si la sauvegarde existe mais l'idée est là et elle fonctionne parfaitement. Tout ça donne au final une interface qui charge les dernières sauvegardes.



Fichiers Récents (Images d'illustration):



Explorateur de fichier, Par exemple, la save 1 (cf. 38) n'étant pas "récente", on la charge depuis ici. Si la save n'existe pas, cela l'indique.

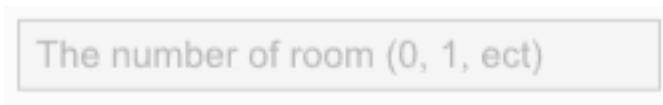
Sélecteur de contraintes

Pour cette interface, nous avons eu beaucoup de difficulté à la mettre en place entre les réunions qu'on a mis en place pour avancer dessus un maximum malgré le peu de temps qu'il nous restait. Comme nous l'avons énoncé lors du dernier rapport de soutenance nous avons utilisé Slint et encore une fois, le manque de documentation sur la librairie Slint ne nous a pas franchement aidé. Même si maintenant le programme tourne sans problème et nous a donné ceci avant les différentes fusions de programmes à la fin du projet :

Choose the number of room	Choose your house orientation	Initialization
The number of room (0, 1, ect)	West	
Choose the number of living room	Choose your types of doors	
The number of living room (0, 1, ect)	Swing Doors	
Choose the number of kitchen	Choose the superficy of habitation (m ²)	
The number of kitchen (0, 1, ect)	The superficy of habitation (m ²)	
Choose the number of bathroom	Choose the superficy of ground (m ²)	
The number of bathroom (0, 1, ect)	The superficy of ground (m ²)	
Choose the number of toilets	Choose the superficy of living room (m ²)	
The number of toilets (0, 1, ect)	The superficy of living room (m ²)	
Choose the number of bedroom	Choose the superficy of kitchen (m ²)	
The number of bedroom (0, 1, ect)	The superficy of kitchen (m ²)	
Choose the number of free room	Choose the superficy of bathroom (m ²)	
The number of free room (0, 1, ect)	The superficy of bathroom (m ²)	
	Choose the superficy of bedroom (m ²)	
	The superficy of bedroom (m ²)	

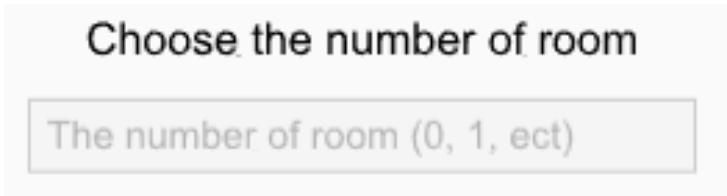
Comme vous pouvez le voir, nous avons écrit l'interface graphique que vous voyez ci-dessus en anglais pour que cela puisse toucher un maximum de personnes et on voulait également qu'elle reste la plus simple et compréhensible possible. Ce que vous voyez ci-dessus est une représentation de l'interface sur le système

d'exploitation MACOSX et donc sa représentation sera différente au final. Comme les résultats sont similaires que ce soit sur Windows, Linux ou MACOSX et que c'est le même programme de toute façon (On ne s'est pas amusé à en faire plusieurs pour le plaisir), on va vous exposer ce qui compose notre interface. Tout d'abord, les **LineEdit** une composante de slint qui permet de créer des lignes de texte pour ainsi pouvoir récupérer les données (ici des nombres) que nous transmet l'utilisateur et c'est à nous d'interpréter les résultats. Voici donc un exemple de **LineEdit** :



The number of room (0, 1, ect)

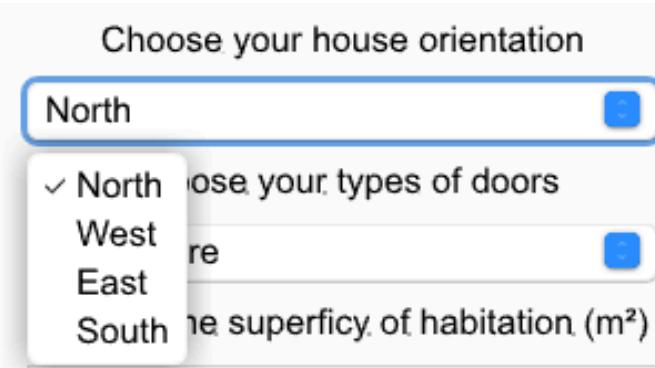
Et pour les titres au-dessus des **LineEdit** on a utilisé des blocs de **Text** avant les **LineEdit** pour leur donner un titre dans l'optique une nouvelle fois que ce soit compréhensible un maximum par le plus de monde possible. Pour en revenir au bloc **Text** il permet tout simple d'ajouter du texte un peu où l'on souhaite selon comment on le place dans le programme ce qui nous donne dans notre cas ceci :



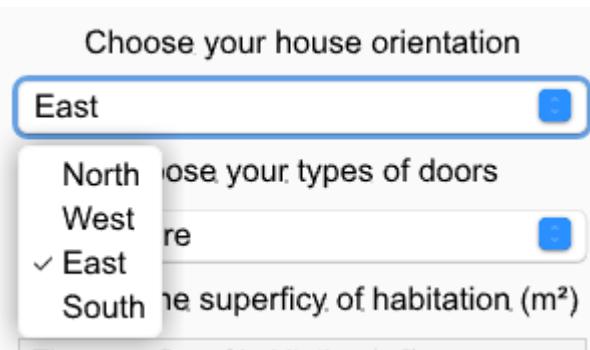
Choose the number of room

The number of room (0, 1, ect)

On peut constater que dans notre cas la composante **Text** est bien au-dessus de la composante **LineEdit** ce qui nous permet bien d'avoir une sorte de titre pour celle-ci. Parlons des **ComboBox** qui sont elles aussi des composantes de Slint permettent de faire beaucoup, elles permettent notamment de créer des menus déroulant comme ceci :



Mais on peut également changer la valeur en cours en sélectionnant une autre des valeurs qui sont proposées comme dans ce cas-là où choisi “East” plutôt que “North”:



De plus, la **ComboBox** à l'instar de la **VerticalBox** et de l' **HorizontalBox** sert à mettre en page d'une certaine manière l'interface. Exposons, la dernière composante visible de l'interface, nous parlons de la composante **Button** alors celle-ci comme son nom l'indique c'est un bouton sur lequel on appuie dessus pour générer un plan une fois tout rempli entièrement bien évidemment et donc il ressemble à ceci :



Après malgré le manque de documentation, on trouve que l'interface reste agréable et facile à comprendre mais c'est vrai qu'elle nous a pris quand même énormément de temps. À cause de ça, nous avons eu beaucoup à rattraper en

termes de retard mais nous nous sommes débrouillés malgré le peu de temps qu'il nous restait. On ajoutera aussi que entre l'écriture du rapport et l'interface, il s'est écoulé quelques jours donc il est possible que certaines modifications sur le choix des fonctionnalités que l'on voulait proposer mais cela devrait être exactement le même format à un ou deux détails près.

L'affichage du résultat

Après avoir créé 2 interfaces, cela a été plutôt simple de créer cette dernière page qui a pour unique but que d'afficher le résultat. En fait en apparence oui, mais en fait non... Il fallait que ce qui est affiché et l'image sur lequel on a cliqué ou qu'on a chargé, donc il fallait un minimum de dynamique qui n'a pas été simple à mettre en place, beaucoup de fonctions de parse de liens ont dû être mises en place.



La connexion des algorithmes à l'interface

Pour cette partie, il a fallu utiliser des **callback** de Slint, ayant déjà fait une interface avec GTK pour un tp de programmation, c'est nous avons rapidement compris le concept pour certains mais cela a été plus difficile pour d'autres. Étant

donné que nous avons plusieurs **component** dans notre interface, pour que tout puisse être appelé à droite et à gauche, nous avons créé des callbacks “globaux”. Il a également fallu définir des alias sur les propriété pour pouvoir récupérer les info que l'utilisateur rentre dans les zone de texte. Notre interface contenant beaucoup de **LineEdit**, nous avons en conséquence beaucoup de callback à faire sans se mélanger les pinceaux. Et en plus, on doit pouvoir mettre et charger les données dans/d'un fichier.txt pour pouvoir les utiliser c'est pour cela qu'on va utiliser des programmes de “conversions” pour mettre les valeurs obtenues dans ce fameux fichier. Ces programmes permettent la connexion entre tous les autres programmes. Cela va d'un programme qui effectue des générations de nombres aléatoires, à de la sauvegarde de fichier.

Ainsi on peut facilement dire que tout cela n'a pas été une partie de rigolade et que au moment où on écrit ce rapport on n'est pas encore sûr que cela fonctionne.

Les problèmes de Slint

Comme dit et redit, le rust étant un langage assez nouveau (2006 si on deit pas de bêtises), nous avons dû faire face au manque cruel de documentation mais en soit on a pu se débrouiller pour en comprendre un maximum. Le véritable problème est que nous avons fait fasse à un manque d'information concernant tout ce qui est le debug, ou les différentes erreurs que nous avons rencontré. Pour vous donner un exemple, nous avons trouvé une erreur qui n'a pas trouvé de solution dans le peu de temps que nous avions. En fait, nous avions créé 2 fenêtres via 2 macro **slint::slint!**, une pour l'application en elle-même, et une autre pour une boîte de dialogue. Tout fonctionnait très bien, jusqu'à ce que nous fassions le merge final, où, pour une raison qui nous est encore malheureusement obscure, cela ne marchait tout simplement plus. Après des heures de recherche sur l'internet mondial en commençant par l'anglais en passant par l'espagnol, l'allemand et également le français, nous avons fait le choix de passer par une autre alternative puisque rien n'a été trouvé. Ceci n'est qu'un exemple parmi tout ce que nous avons rencontré durant la folle épopée qu'a été la conception de ce projet.

La création et l'export d'un programme main.rs

Pour pouvoir faire tourner cette application, nous avons eu besoin d'un **main.rs**. Notre choix d'avoir notre macro `slint` dans ce même fichier nous a posé un problème de taille, puisque nous avons facilement atteint les 1000 lignes dans ce fichier, pour s'y retrouver, nous avons donc refacto le code, tout ce qui n'est pas du `slint` va dans des fichier avec un **lib.rs** et un **mod.rs** et nous avons également segmenter le code grâce à des boite de commentaire pour chaque parti du code. Pour exporter ce programme nous voulions réaliser un build pour Windows et linux, mais la diversité des systèmes d'exploitation linux étant divergent, nous avons donc décidé d'en faire un que pour Windows. Il n'en restait qu'à le faire...

Nous devions donc créer un fichier `build.rs` et encore une fois une galère sans nom pour trouver des infos. Mais finalement cela a marché. Nous avons ensuite utilisé `inno setup compiler` pour créer un setup d'installation.

Site Web

À l'exception des descriptions, des liens de téléchargement et des annexes, le Site web n'a pas changé fondamentalement, comme prévu dans l'avancement du projet, nous nous sommes concentrées sur l'interface et seulement ajouter les éléments concernant la dernière période de travail.

VII - Les ressenties individuelles sur la conception du projet

Nous nous sommes dit que plutôt que de vous faire un point de vue global on pourrait vous donner le point de vue de chacun car chacun à sa propre perception des choses et à le droit de la partager.

Maël ROUSTIT (Chef du groupe The Raiders)

Peine

Slint... Que dire, c'est bien, mais ça a quand même été un défi pour moi d'apprendre une nouvelle lib après m'être déjà fait a GTK, j'ai du re apprendre de 0 une nouvelle lib avec pour seul information une documentation Slint, et un rust. Cela est très maigre et bien chiant pour résoudre les problèmes. La mise en commun finale du code n'a pas non plus été une partie de plaisir, on a tous codés des feature et quand on devait les mettre ensemble on s'est très souvent confronté à des "merge conflict" ce qui ne nous a pas facilité la tâche pour rendre les release..

Joie

La création du site m'a bien amusé, c'était sympa, pas mal de tuto, d'explication ce qui a mené à pas trop de galère même si ça n'a clairement pas été facile. J'ai bien aimé la gestion de projet aussi, c'était ma première fois et à part les quelques difficultés de librairie vers la fin, le projet a été selon moi plutôt bien organisé dans son ensemble.

Axel OURY (Membre du groupe The Raiders)

Peine

Tout d'abord, la chose qui me peine c'est qu'on a dû faire 3 propositions de projet avant que celui-ci soit validé. En outre, celui-ci fut approuvé durant la première soutenance donc on n'était même pas sûr d'avoir un projet à l'issue de cet oral et on commençait avec plus d'un mois voire deux de retard par rapport aux délais initiaux. Je trouve aussi qu'on n'a pas reçu assez de détails sur ce que l'on devait pour ce projet, le sujet n'a pas été assez aiguillé, il était trop évasif je dirai pour être plus précis. Bon, il a eu aussi des problèmes qu'on a rencontré avec différents librairies, le fait qu'on n'avait pas tous le même système d'exploitation (j'étais sur MACOSX, les autres sur Windows et ils utilisaient Ubuntu pour bosser moi le terminal de mon mac). En parlant de librairie, je trouve aussi qu'au lieu de nous proposer de plus en plus de librairies (ce qui n'était pas une mauvaise chose en soi) à une semaine de notre dernière soutenance alors que l'on demandait juste une validation pour avancer. D'accord, on n'était déjà pas en avance, mais là nous donner l'accord pour une librairie seulement le vendredi 31 Mai aux alentours de 15h40 à cinq jours de la soutenance et sachant qu'on avait tous des examens cette semaine de soutenance, alors que la conversation a débuté le samedi 25 Mai. J'ajouterais qu'en plus d'une charge de travail importante cela a été une stress et une charge mentale très importante pour moi, en plus d'avoir fait des semi-insomnies à cause de ce projet notamment durant la dernière semaine avant l'oral final. Pour finir, je trouve que le barème est incohérent et qu'il ne laisse pas assez de libertés à l'examinateur de noter comme il le souhaite (Soit on a 0%, 40%, 75% ou 100% mais jamais 50% ou 20%). Ce serait aussi une bonne de mettre moins de points sur le code pour la première soutenance car on se retrouve avec une note qui est de manière à ce que l'on est 70% sur code et 30% sur l'oral pour une première soutenance on pourrait plutôt faire du 50%-50% ce serait mieux pour débuter espérer avoir la moyenne. Voilà je pense avoir dit tout ce qui me dérangeait, en espérant que ce soit pris en compte pour les années à venir.

Joie

Malgré un certain nombre de problèmes, de désillusions et de dépressions mentales, il a quand même de bonnes choses à retenir de cette expérience. Premièrement, je trouve qu'on a quelque chose qui fonctionne et qui est cohérent avec ce qu'on attendait comme résultat final. Je trouve aussi qu'on c'est tous bien entendu entre nous. Pour ma part, quand j'arrivais ou quand on arrivait à faire fonctionner une nouvelle chose après plusieurs heures ou jours de tentatives, j'étais très heureux. Pour revenir sur les librairies, je voulais remercier le professeur pour toute l'aide qui nous a donnée. Je suis ravi aussi de voir que l'on s'est tous battu jusqu'au bout du bout pour que notre projet puisse voir le jour. Et je conclurai mes propos en citant les mots de Friedrich Nietzsche un philosophe prussien qui a dit dans **le Crépuscule Des Idoles**(1888) "Ce qui ne fait pas mourir me rend plus fort" et malgré les épreuves que nous avons affronté, je trouve que cela nous a rendu bien plus fort qu'au début de ce projet.

Damien MARRASSÉ (Membre du groupe The Raiders)

Peine

Ça a été un projet très compliqué qui a demandé beaucoup de travail. Le premier mois c'est mal passé du fait que notre projet a été refusé plusieurs fois nous forçant à aller sur un projet qui ne m'intéresse pas tant que ça. De fait, cela a été dur de me motiver à travailler dessus et j'avais hâte que ça se finisse d'autant plus que j'avais des maths à réviser. C'est officiel, je déteste l'architecture.

Joie

Un point positif à tout ça : je suis quand même content et fier de nous pour avoir fini le projet.

Romain D'ANGE-BOURGUIGNON (Membre du groupe The Raiders)

Peine

De mon côté, le gros point négatif a été la librairie graphique. Il a fallu beaucoup de temps avant qu'une de nos librairies ne soit validée, et au final la librairie qui a été validée était une librairie que j'ai dû apprendre à comprendre et à utiliser en moins d'une semaine. Forcément, en un laps de temps aussi court, apprendre entièrement une toute nouvelle librairie n'a pas été de tout repos, d'autant plus qu'il n'y avait aucun tuto trouvable sur youtube ou sur des forums. J'ai donc dû m'y atteler quasiment non-stop durant cette semaine, et malgré cela je suis certain de ne comprendre qu'une infime partie de la librairie.

Joie

Il est difficile de trouver à l'heure où nous écrivons ce rapport : nous avons tout donné pour terminer ce projet, sacrifiant des heures de sommeil et même des heures de révisions. Mais je pense que dans quelques jours, lorsque tout le monde aura repris un vrai rythme de sommeil, nous pourrons tout de même être fiers de ce que nous avons réussi à accomplir sur ce projet.

VIII - L'intérêt du travail en groupe

Notre projet “Raider’s Architecture” va nous apporter de grandes choses en termes de valeurs.

Nous allons devoir communiquer, nous allons devoir discuter, nous allons devoir argumenter comme une véritable bande d'amis cherchant à choisir la meilleure série à regarder. Chacun donne son avis et, ensemble, on décide ce qui fonctionne le mieux, ce qui est le plus intéressant à rajouter, quelle méthode semble être la meilleure.... Cela fait ressortir l'esprit d'équipe, nous permettant d'être soudés et de travailler ensemble dans la même direction, afin d'atteindre le but commun qui est de réaliser toutes nos tâches. Les moments difficiles sont comme des défis de jeux vidéo : on n'abandonne pas, on s'entraide afin de pouvoir les surpasser.

Bien sûr, tout n'est pas tout rose, il nous arrive de nous énerver à propos de certains détails, il nous arrive également de se tromper sur la direction à prendre par rapport à certains programmes algorithmiques. Quand une idée ne marche pas, on cherche à s'adapter, par exemple en expérimentant une autre approche. On apprend tous ensemble, et chacune de nos erreurs sont des petits coups de pouce vers le succès.

En fin de compte, notre projet “Raider’s Architecture” n'est pas juste une simple machine à réaliser des plans de construction. C'est bien plus que cela : c'est une création d'équipe, qui montre le fruit d'une lourde réflexion sur un problème plus que complexe où chacun a apporté et apporte encore ses idées et ses compétences. C'est cela le véritable travail d'équipe : non pas une simple collaboration, mais une expérience qui nous fera à coup sûr grandir.

Et puis pour rappel, notre groupe se nomme "The Raiders", en référence au jeu Tomb Raider, parce que nous aimons tous les jeux vidéo et que la recherche d'un sujet à la fois intéressant et challengeant algorithmiquement a été une véritable aventure. Tout comme ce projet qui sera notre incroyable épopée vers quelque chose qui nous rendra tous fiers.

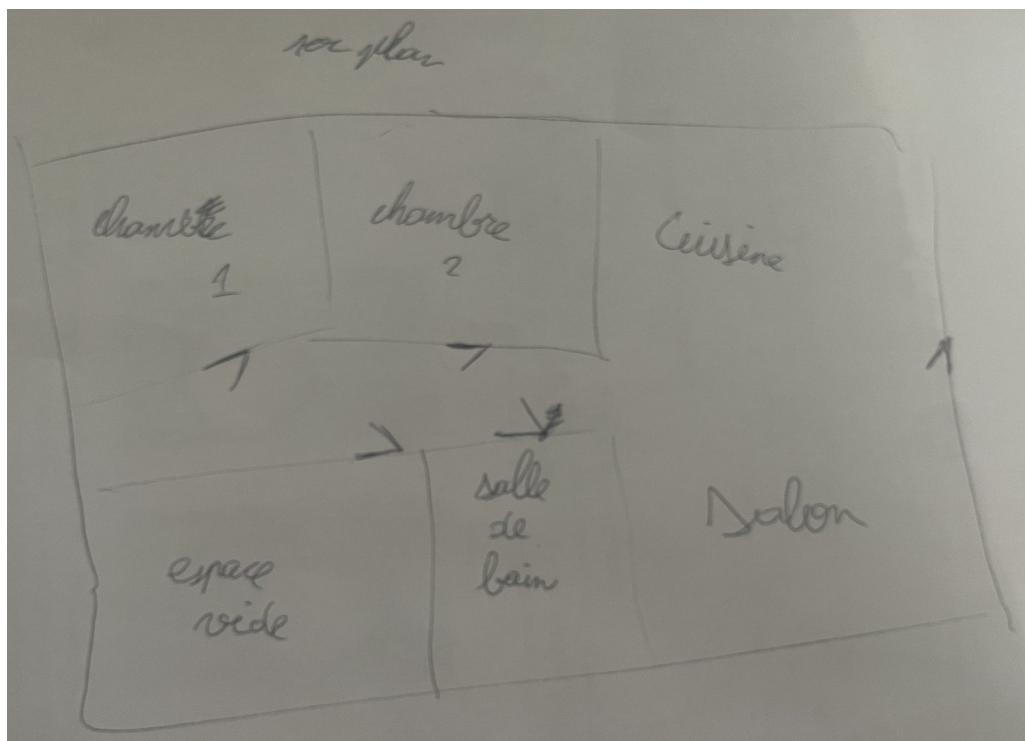
IX – Conclusion

Depuis notre soutenance précédente, nous avons réalisé pas mal de tâches et notre projet s'oriente désormais dans un chemin qui devrait nous permettre d'offrir un rendu propre et répondant à vos attentes, mais également aux nôtres. Nous avons travaillé d'arrache-pied afin de rattraper le retard que nous avions accumulé lors de notre précédente soutenance, et sommes désormais en bonne voie pour atteindre notre objectif final. Nous espérons que le fruit du travail que nous avons fourni jusqu'à maintenant vous a permis de partager notre engouement pour ce projet, et que cela vous aura convaincu. Merci beaucoup d'avoir pris le temps de lire ce rapport, et nous espérons de tout coeur que celui-ci vous aura plu. Le groupe "The Raiders" vous souhaite une excellente journée et vous dit à très bientôt pour notre dernière soutenance, où nous aurons l'occasion de vous présenter notre projet final.

X - Annexe

Dessins d'origine

Taille maison (en mètre carré)	<input type="text"/>	Jardin	<input type="radio"/> Oui	<input type="radio"/> Non	 CRÉATION
Nombre de toilettes	<input type="text"/>	Terrasse	<input type="radio"/> Oui	<input type="radio"/> Non	
Nombre de chambre	<input type="text"/>	Piscine	<input type="radio"/> Oui	<input type="radio"/> Non	
Nombre de salle de bains	<input type="text"/>	Cuisine ouverte	<input type="radio"/> Oui	<input type="radio"/> Non	
Taille (minimum) de la cuisine	<input type="text"/>	Suite parentale	<input type="radio"/> Oui	<input type="radio"/> Non	
Taille (minimum) du salon	<input type="text"/>	Garage	<input type="radio"/> Oui	<input type="radio"/> Non	
Nombre d'espaces libres	<input type="text"/>	Ascenseur	<input type="radio"/> Oui	<input type="radio"/> Non	
Taille (minimum) d'une chambre	<input type="text"/>	Escalier	<input type="radio"/> Oui	<input type="radio"/> Non	
Nombre d'étages (0 = plein pied)	<input type="text"/>				



Exemple d'écran du projet

The screenshot shows a software window titled "Raider's Architecture". The top bar includes standard window controls (minimize, maximize, close) and menu items "Home" and "Save". The main area contains several input fields and dropdown menus for configuring a house project:

- Choose the number of living room**: "The number of living room (0, 1, ect)"
- Choose your house orientation**: "North" (selected)
- Process** button
- Choose the number of kitchen**: "The number of kitchen (0, 1, ect)"
- Choose your types of doors**: "I don't care" (selected)
- Choose the superficy of habitation (m²)**: "The superficy of habitation (m²)"
- Choose the superficy of living room (m²)**: "The superficy of living room (m²)"
- Choose the number of bathroom**: "The number of bathroom (0, 1, ect)"
- Choose the superficy of kitchen (m²)**: "The superficy of kitchen (m²)"
- Choose the number of toilets**: "The number of toilets (0, 1, ect)"
- Choose the number of bedroom**: "The number of bedroom (0, 1, ect)"
- Choose the superficy of bathroom (m²)**: "The superficy of bathroom (m²)"
- Choose the number of free room**: "The number of free room (0, 1, ect)"
- Choose the superficy of bedroom (m²)**: "The superficy of bedroom (m²)"