

CHRONO PARK

Rapport de soutenance finale

Charlotte BUAT
Anthony VERKINDERE
Nathan LABERNARDIERE



EPITA, Juin 2022.

Table des matières

1	Introduction	4
2	État de l’art	5
2.1	Technologies existantes	5
2.2	Notre solution	5
3	Découpage des tâches	7
3.1	Algorithme du plus court chemin (Anthony)	7
3.1.1	Ce qui était prévu	7
3.1.2	Ce qui a été fait	7
3.2	Options (Charlotte)	10
3.2.1	Options principales	10
3.2.2	Options secondaires	11
3.3	Graphismes (Nathan)	13
3.3.1	Ce qui était prévu	13
3.3.2	Ce qui a été fait	13
3.4	Plan du parc (Nathan)	19
3.4.1	Ce qui était prévu	19
3.4.2	Ce qui a été fait	20
3.5	Déplacement des humains (Anthony / Nathan)	23
3.5.1	Ce qui était prévu	23
3.5.2	Ce qui a été fait	23
3.6	Début de l’intelligence artificielle (Charlotte)	27
3.7	Ajout de traits humains (Charlotte)	29
3.7.1	Ce qui était prévu	29
3.7.2	Ce qui a été fait	29
3.8	UI avec options (Nathan)	30
3.8.1	Ce qui était prévu	30
3.8.2	Ce qui a été fait	30
3.9	Site Web (Charlotte)	33
3.9.1	Ce qui était prévu	33
3.9.2	Ce qui a été fait	33
3.10	Répartition des tâches	36

3.11	Avancement	36
4	Structure du répertoire	37
5	Ressenti des membres	40
5.1	Charlotte Buat	40
5.2	Nathan Labernardiere	41
5.3	Anthony Verkindere	42
6	Conclusion	43
7	Bibliographie	44

1 Introduction

Depuis toujours, un des plus gros défaut des parcs d'attractions sont les files d'attente. En effet, les files d'attente sont la hantise des visiteurs.

Prenons par exemple le célèbre parc Disneyland : un visiteur moyen passe 3 à 4 heures dans des files pour une journée de 8 à 9 heures soit plus d'un tiers de son temps. Il y a plusieurs facteurs à prendre en compte pour comprendre pourquoi le temps d'attente des files peut sembler interminable.

Ainsi, nous avons décidé de travailler sur un simulateur de trafic dans un parc d'attraction le plus complet et réaliste possible pour comprendre quels sont les facteurs qui influent sur les files d'attente et, par conséquent, voir comment les optimiser.

2 État de l'art

2.1 Technologies existantes

En ce qui concerne l'état de l'art en lien avec notre projet, il existe déjà plusieurs technologies aidant à l'optimisation de file d'attente.

Parmi celles-ci, nous pouvons citer toutes les applications directement créées par et pour certains parcs d'attractions tels que Disneyland Paris ou le parc Astérix en France mais aussi plein d'autres à travers le monde. Celles-ci permettent aux personnes dans le parc de savoir en temps réel quel serait le temps d'attente pour chaque attraction ou encore de localiser des attractions à travers le parcs. Mais aussi, depuis peu sur l'application Disneyland Paris, de réserver des « coupes files » pour certaines attraction ce qui permet de ne pas faire la queue du tout. Tout ceci dans le but de diminuer au maximum les temps d'attente de chaque visiteur.

Mais il existe aussi des sites Internet et applications indépendantes tels que **ParkTrips.fr**. Ceux-ci donnent des astuces afin d'éviter l'afflux touristique ou encore d'organiser son parcours au préalable. Tout ceci permettant aux visiteurs d'éviter les longues files d'attente.

2.2 Notre solution

Concernant notre projet, nous avons donc prévu de faire en sorte que les utilisateurs puissent optimiser leur visite de parc d'attraction en réduisant le temps passé dans les files d'attente.

Pour cela, nous allons recréer un parc avec un nombre défini d'attractions pour l'instant, tout en essayant plus tard de rendre le nombre d'attractions personnalisable. Puis, un algorithme permettra de définir un chemin optimal entre ces attractions tout en prenant en compte les goûts renseignés au préalable par l'utilisateur. Nous prendrons aussi en compte toute panne éventuelle d'une attraction ou encore la météo car la pluie diminuerait l'affluence générale du parc.

Tout ceci pour au final renvoyer le ou les chemins optimaux pour un utilisateur afin qu'il ait au final beaucoup moins d'heures d'attente.

Nous avons aussi décider d'implémenter des graphismes afin de laisser l'utilisateur voir ce qu'il se passe exactement avec les IA, les humains présents dans le parcs, se déplaçant entre les attractions.

3 Découpage des tâches

3.1 Algorithme du plus court chemin (Anthony)

3.1.1 Ce qui était prévu

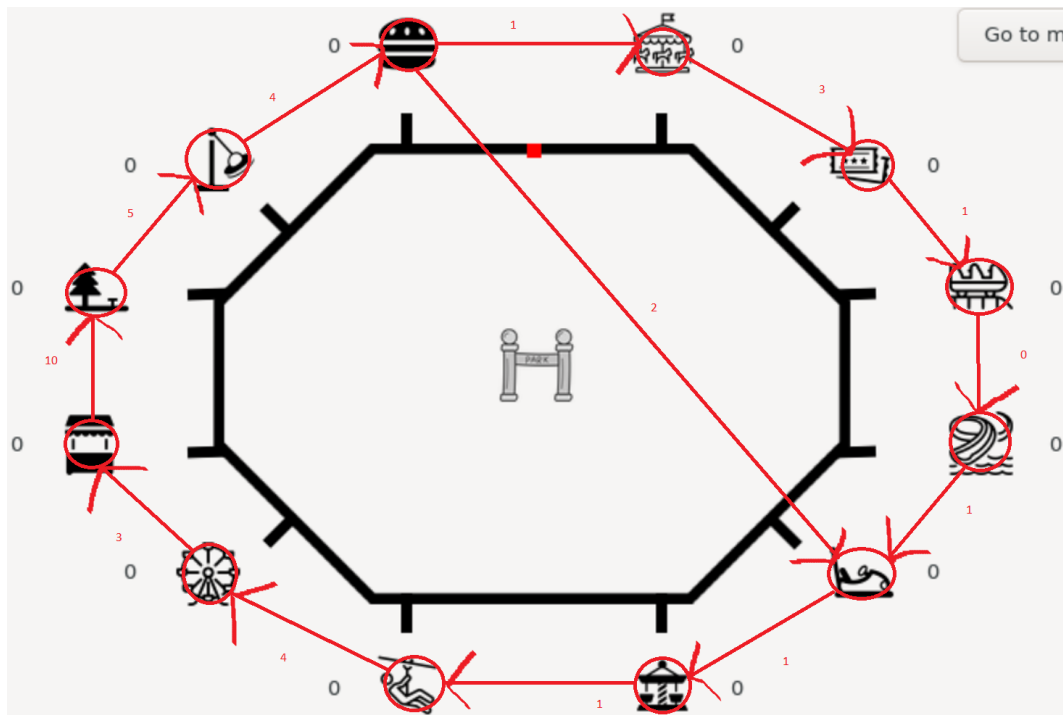
L'objectif du logiciel est de proposer un enchaînement d'attractions en fonction de l'affluence qu'il y a sur chaque attraction. Dans l'objectif que le parc soit terminé avec un temps d'attente dans les files le plus faible possible.

C'est pourquoi nous avons besoin de faire un algorithme du plus court chemin, qui peut prendre en compte le fait qu'une file d'attente pour une attraction soit plus ou moins pleine.

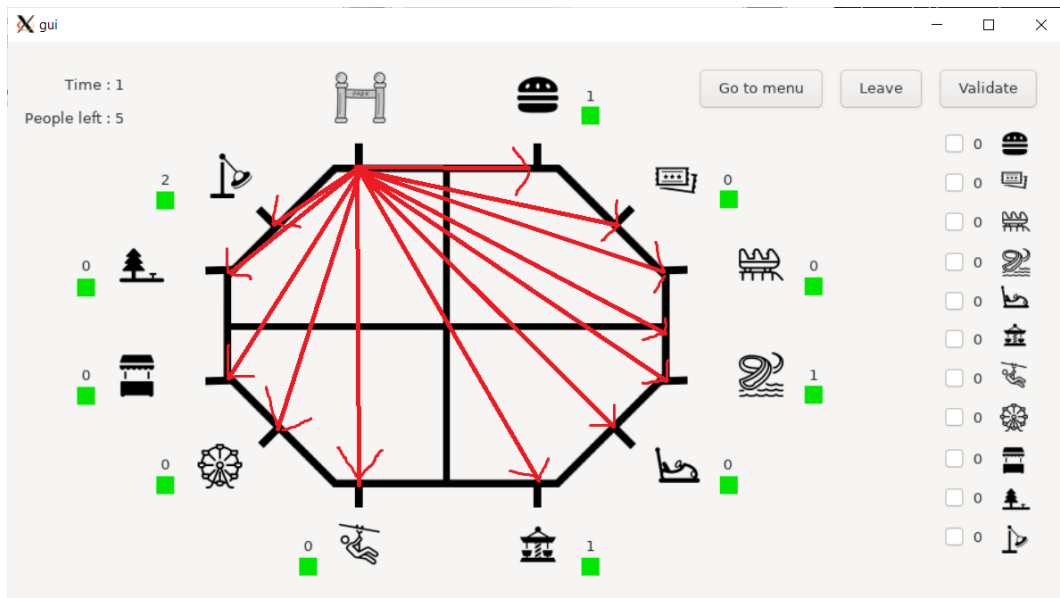
Nous voulons permettre à nos utilisateurs de parcourir un parc réaliste, il y aura donc plusieurs chemins possibles pour atteindre les différentes attractions, et des circuits possibles. Ainsi, nous ne pouvons pas utiliser l'algorithme de Dijkstra qui n'accepte pas les circuits. Étant donné que nous voulons proposer des parcours permettant à l'utilisateur de parcourir les attractions de son choix, nous allons plutôt nous tourner vers des algorithmes tels celui de Floyd ou de Johnson. Mais par la suite nous souhaitons pouvoir utiliser les différents algorithmes qui vont nous permettre de voir une différence de chemin. Bien évidemment, nous allons adapter nos graphes pour que les deux algorithmes soient compatibles avec celui-ci.

3.1.2 Ce qui a été fait

Pour commencer nous avons dû développer une classe "Graph". Comme nous avons pu voir en cours d'algorithme mais ici en C dans le but de représenter un parc d'attraction sous forme de graphe. Dans notre graphe, les sommets correspondent à une attraction et chaque chemin entre deux attractions a un poids qui correspond à plusieurs critères comme la distance et le nombre de personne dans l'attraction. De plus, nous avons implémenté une classe liste pour faciliter l'implémentation de notre algorithme. Voici comment l'on peut visualiser notre graphe (le plan ci joint n'est pas celui présent dans notre simulateur, c'est tout simplement pour comprendre l'idée), ici c'est un exemple les liaisons entre chaque sommet et leur coût peuvent changer et nous pouvons avoir plus ou moins de sommets.



Nous avons développé deux algorithmes de plus court chemin, Floyd et Dijkstra. Floyd était celui que nous avions prévu de développer et c'est celui que nous utilisons pour le moment dans notre logiciel. Finalement, nous avons préféré nous concentrer sur l'algorithme de Floyd qui était tout simplement le plus optimal pour notre projet. Nous pouvons donc dire que notre algorithme est terminé. Finalement, pour cette soutenance finale il a surtout été question de complexifier les poids et notre graphe que l'on entre dans notre algorithme grâce aux différentes options que Charlotte vous présentera par la suite. Voici comment peut on réellement représenter notre graphe dans le parc d'attraction.



Ici nous pouvons donc en partant de chaque attractions aller à n'importe laquelle autre attraction, tout en passant par les différents chemins possible, qui nous sont possible grace aux de routes passant par le centre de la carte. Les flèches vont bien évidemment dans les deux sens et à imaginer pour chaque autres attractions(Ici ce n'est qu'une représentation simplifié). Nous avons donc mis en place un système qui tourne en boucle et qui a pour but de mettre à jour la liste adjacente de notre graphe.

3.2 Options (Charlotte)

3.2.1 Options principales

Ce qui était prévu :

Nous allons implémenter de nombreux paramètres afin d'améliorer le réalisme du simulateur. La première option essentielle est le temps d'attente dans les files d'attente en fonction du nombre de visiteurs présents dans le parc. Idéalement, nous allons essayer de donner la possibilité à l'utilisateur de faire varier le nombre de personnes présentes.

Ce qui a été fait :

Pour la première soutenance, nous avons implémenté la variation du nombre de personnes présentes dans le parc. De plus, nous avons la possibilité de positionner ces personnes aléatoirement sur chaque attraction du parc tout en tenant compte certains critères précis expliqués plus tard. Cependant, nous n'avons pas encore implémenté un code nous permettant de récupérer un temps d'attente en file. Cependant, il y en a bien un "caché". C'est à dire quand nous arrivons à une attraction nous ne pouvons pas directement entrer dans celle ci sans attendre que toutes les personnes (l'IA) soient sorties de l'attraction.

Pour cette deuxième soutenance, nous avons aussi rajouter plusieurs options concernant la météo. L'utilisateur peut ainsi choisir si il veut de la pluie tout le temps, parfois ou pas du tout sur l'interface graphique. Plus il y a de la pluie, plus les gens sont enclins à ne pas être dans les files d'attentes des attractions. Ils préfèrent prendre une pause et s'abriter de la pluie.

Ce qui est prévu :

Pour la soutenance finale. Nous allons essayer de mettre sur l'interface graphique plus d'informations à propos du temps d'attente quand nous arrivons à une attraction tout en essayant de le complexifier en le faisant varier en fonction de critères spéciaux à chaque attraction.

3.2.2 Options secondaires

Ce qui était prévu :

Nous allons donc donner la possibilité à l'utilisateur de changer le nombre de visiteurs manuellement et automatiquement. Implémenter le fait qu'une attraction puisse être en maintenance ou tout simplement hors service (le but est que cela se fasse aléatoirement). De plus nous allons ajouter un système de « météo » qui va nous permettre d'influencer le nombre de personnes présentes dans le parc. Pour optimiser encore notre algorithme et la fiabilité de notre logiciel, nous allons prendre en compte la densité de personnes présentes sur un trajet entre deux attractions pour avoir le parcours le plus optimal dans le but de terminer le parc tout en ayant un temps d'attente dans les files le plus faible possible. Pour finir, nous essaierons de proposer plusieurs parcours différents en fonction des préférences de l'utilisateur.

Ce qui a été fait :

Comme dit précédemment, le nombre de visiteurs présents dans le parc au démarrage peut être saisi manuellement. Pour ce qui est de la météo, l'interface est déjà prête à l'avoir cependant cela n'a pas encore été implémenté. Notre algorithme a la possibilité de donner toutes les distances qu'il peut y avoir en partant de chaque attraction.

Ce qui est prévu :

Nous allons donc implémenter le système de météo pour la soutenance finale. De plus le système d'attraction hors service / maintenance sera mis en place pour ajouter du réalisme à notre logiciel. Nous allons essayer de prendre en compte le nombre de personnes présentes sur le chemin pour améliorer la précision de notre algorithme de plus court chemin mais c'est une partie assez longue et difficile donc nous allons privilégier les autres fonctionnalités avant celle-ci.

3.3 Graphismes (Nathan)

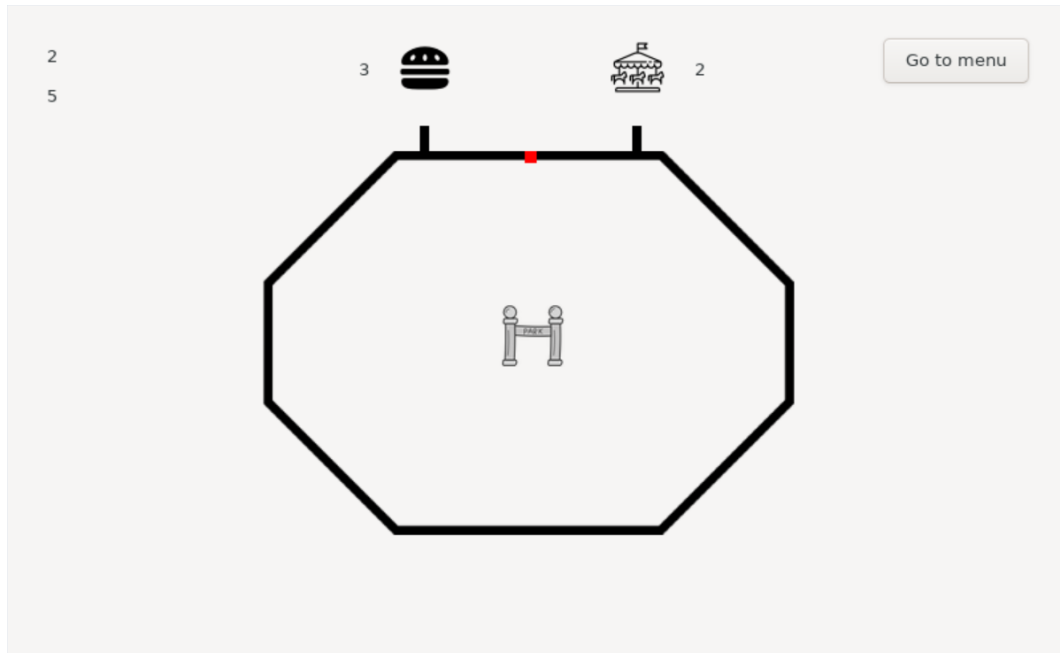
3.3.1 Ce qui était prévu

Notre projet ayant un sujet portant sur l’algorithmique, il était évident pour nous que notre objectif premier n’était pas de faire une interface avec des graphismes réalistes. Néanmoins, nous comptions faire une interface la plus soignée possible avec des dessins simples. Cela étant mieux que de simples pavés de chiffres et résultats illisibles. Une interface avec quelques graphismes permet de mieux voir et comprendre ce qui se passe derrière notre code et permet également de pouvoir expliquer plus simplement comment notre projet fonctionne.

3.3.2 Ce qui a été fait

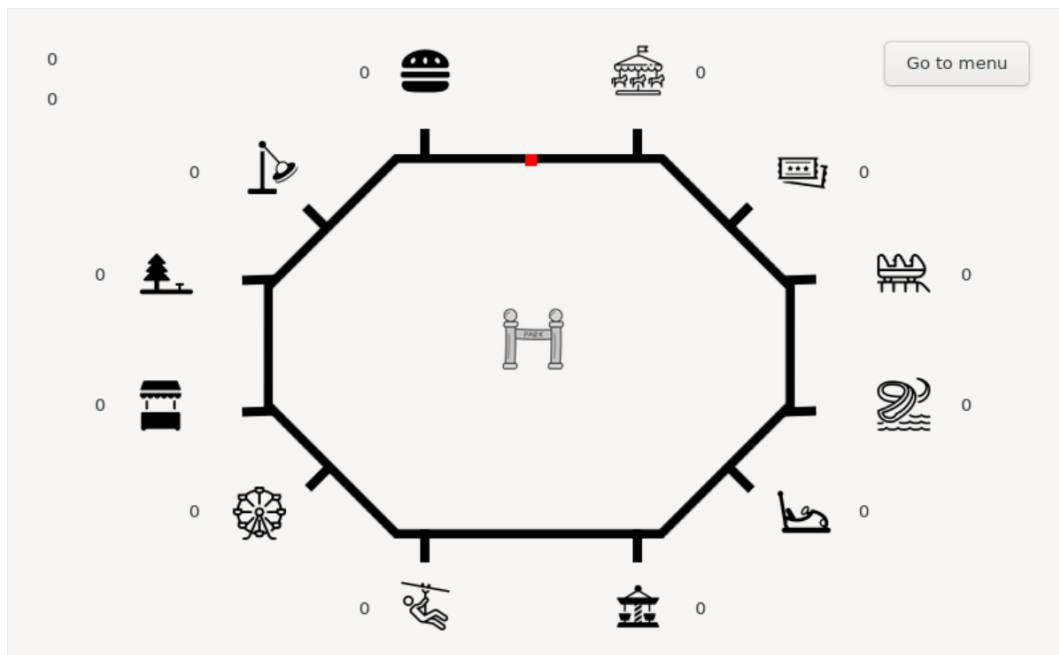
Ainsi, pour cette soutenance finale il y a beaucoup de changements par rapport à ce que nous avons dit dans notre cahier des charges. Pour cela, nous avons utilisé l’application "Glade" pour concevoir les interfaces de notre projet. Nous allons dans cette partie nous intéresser à l’interface qui s’affiche une fois que l’on commence la simulation.

Voici une possibilité d’affichage ci-dessous que nous avons lors de la soutenance intermédiaire :



Dans cet exemple, il y a le nombre minimal d’attractions qui est de 2. Pour réaliser cela, nous avons pris des illustrations sur Internet et dessinés une route faisant le tour du parc.

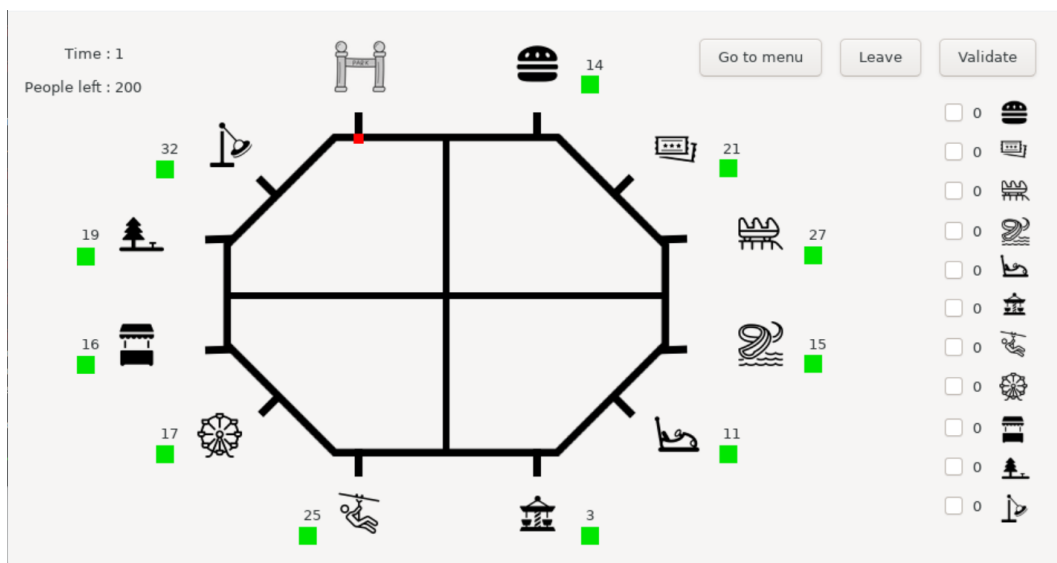
Maintenant, voici un exemple d’affichage avec cette fois-ci 12 attractions toujours lors de soutenance intermédiaire :



Sur l'exemple ci-dessus, il y a cette fois le nombre maximal d'attractions qui est 12. Nous étions au moment de la soutenance intermédiaire plutôt satisfaits du rendu qui est obtenu grâce aux petites illustrations cependant nous avons encore pas mal de changements à faire.

Cette interface était vraiment basique avec peu d'options et relativement vide nous avons rajouté plusieurs éléments pour conclure ce projet.

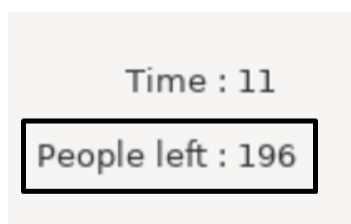
Voici donc un exemple de l'interface finale :



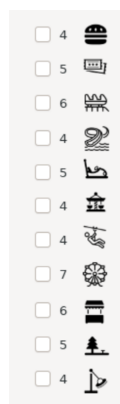
Nous allons donc expliquer tous les éléments présents dans cette interface. Commençons tout d'abord par en haut à gauche. On y retrouve le temps écoulé depuis que la simulation a démarré. Cela permet de regarder le temps moyen des simulations :



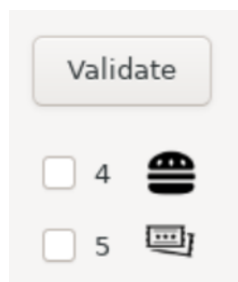
Juste en dessous on peut voir un compteur qui représente tous les humains qui sont encore présents dans le parc :



Il y a ensuite, au centre, le parc ainsi que les routes qui le compose. Maintenant sur la droite, on peut observer plusieurs nouveautés. Commençons par les petites icones avec les cases à cocher.



Cela permet à l'utilisateur de choisir la liste des attractions qu'il veut parcourir de manière optimisée et rapide. Une fois la liste choisie, il faut alors cliquer sur le bouton "Validate" qui va faire déplacer le carré rouge automatiquement :



Une fois que l'utilisateur a fini de parcourir toutes ses attractions souhaitées, il lui suffit alors de cliquer sur le bouton "Leave" qui mènera le petit carré vers la sortie.



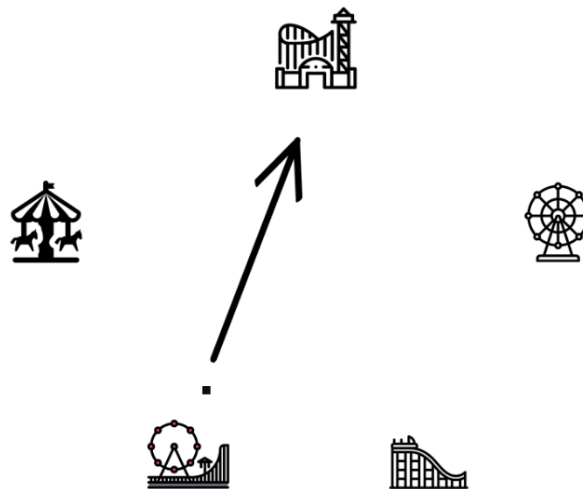
Et enfin, on retrouve comme lors de la dernière soutenance, le bouton "Go to menu" ramène tout simplement sur la fenêtre de démarrage et permet de relancer une simulation plus rapidement.

3.4 Plan du parc (Nathan)

3.4.1 Ce qui était prévu

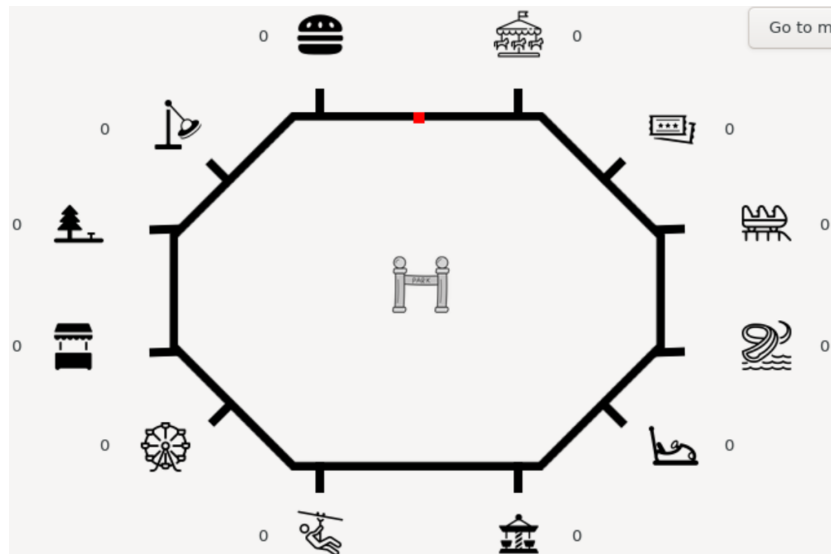
Nous avons prévu que pour le plan du parc, le nombre d'attractions soit défini pour que cela soit plus simple à réaliser. Nous avons pensé également qu'il serait pratique de faire en sorte que les attractions soient arrangées de manière circulaire afin de faciliter la représentation des déplacements humains.

Voici un schéma du parc que nous avons imaginé :



Sur le schéma ci-dessus, il y a 5 attractions et on a un humain, représenté par le point noir, qui s'apprête à se déplacer en ligne droite vers une nouvelle attraction.

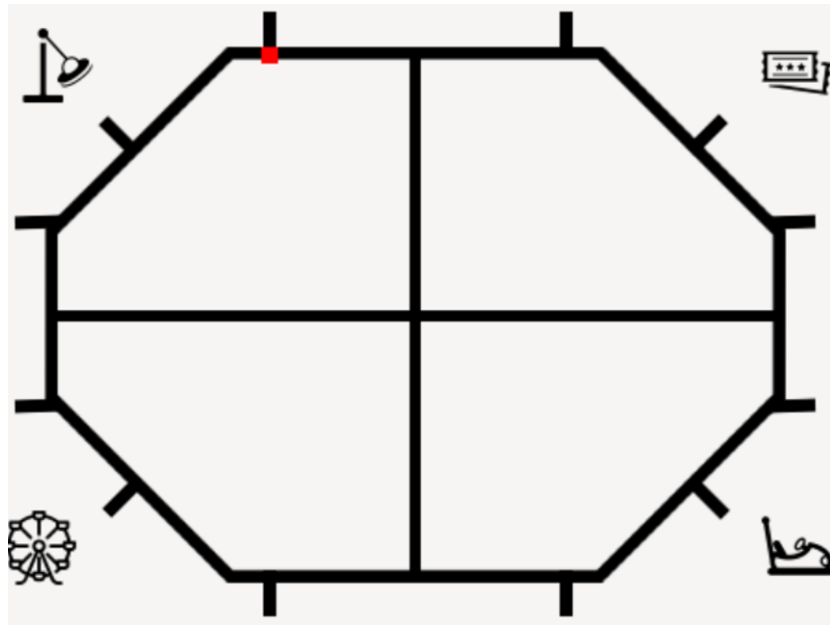
Voici le plan que nous avons lors de la soutenance intermédiaire :



3.4.2 Ce qui a été fait

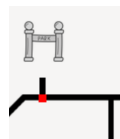
Nous nous sommes rendus compte lors de la soutenance intermédiaire que notre route était vraiment simple et peu réaliste étant donné que cela ne formait qu'une unique route en octogone. Nous avons ainsi amélioré notre parc d'attraction en lui ajoutant 2 nouvelles routes qui coupent le centre de celui-ci.

Voici le nouveau plan du parc ci-dessous :



Ces 2 nouvelles routes rendent le parc plus réaliste et permettent de passer de certaines attractions à d'autres plus rapidement qu'auparavant.

Il y a également un changement dans le plan du parc qui est l'ajout d'une entrée pour le point rouge car pour l'instant il n'y avait aucun endroit où l'on pouvait démarrer. Nous avons par conséquent supprimé 1 attraction pour la remplacer par une porte d'entrée et de sortie. Désormais le nombre d'attractions varie entre 2 et 11.



Enfin pour représenter le parc, nous avons créé une structure comprenant une liste des attractions et le nombre de gens présents dans le parc. Chaque attraction est aussi représentée par une structure qui comprend elle aussi le nombre de gens mais cette fois propre à l'attraction, mais aussi un nombre représentant à quel point chaque utilisateur aime l'attraction que l'on appelle : *likeness* et enfin le nombre de gens qui peuvent rentrer dans l'attraction par tour.

3.5 Déplacement des humains (Anthony / Nathan)

3.5.1 Ce qui était prévu

Pour ce qui est des déplacements, nous n'avions pas prévu de modéliser des déplacements complexes car cela relève plus du détail pour l'affichage que de l'algorithmique.

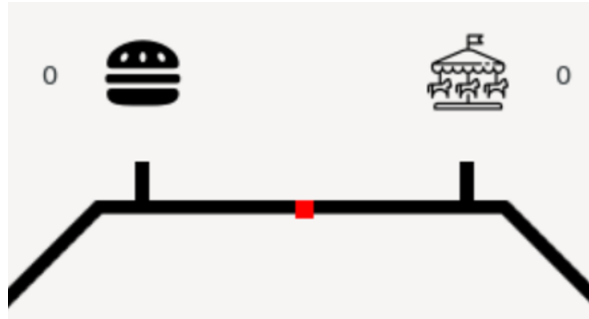
Cependant, nous voulions quand même représenter les humains par des points et les faire se déplacer en ligne droite, d'attraction en attraction. Cependant, nous avons abandonné cette idée car cela ferait beaucoup trop de points affichés à l'écran et rendrait le tout illisible et incompréhensible. De plus, comme nous avions prévu de ne pas non plus représenter graphiquement les files d'attente, nous voulions ajouter des jauges allant du vert (file vide ou presque) au rouge (beaucoup d'attente) permettant d'avoir une idée du nombre de personnes qu'il y a.

3.5.2 Ce qui a été fait

Cela n'a toujours pas changé depuis la dernière soutenance, nous n'avons donc pas modélisé chaque personne une par une mais nous les affichons uniquement dans les attractions (le nombre qui est à côté de l'icône de l'attraction).



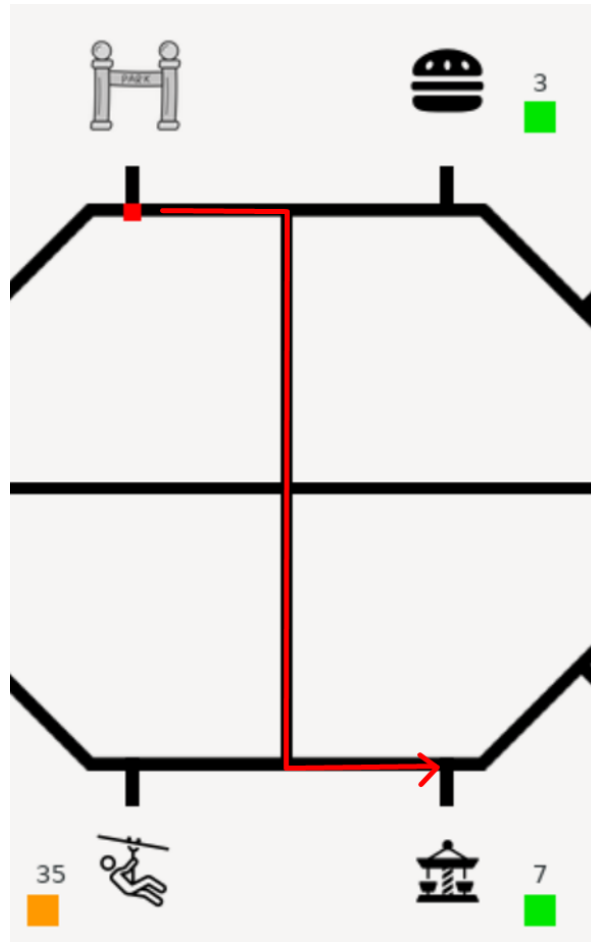
Nous avons cependant toujours notre petit point rouge dont nous avons la possibilité de contrôler pour se déplacer dans le parc.



Le déplacement de ce point n'était pas fonctionnel lors de la soutenance intermédiaire mais désormais il l'est. De plus, étant donné que la route n'est plus la même il a fallu adapter les déplacements.

Désormais à chaque fois que notre algorithme choisit la meilleure attraction à faire en premier parmi celles présentes dans la liste choisie. Le carré rouge se déplace automatiquement vers l'attraction puis s'arrête une fois à destination. Le déplacement est bien plus réaliste maintenant grâce aux nouvelles routes car il ne s'agit plus maintenant de juste aller vers la gauche ou la droite.

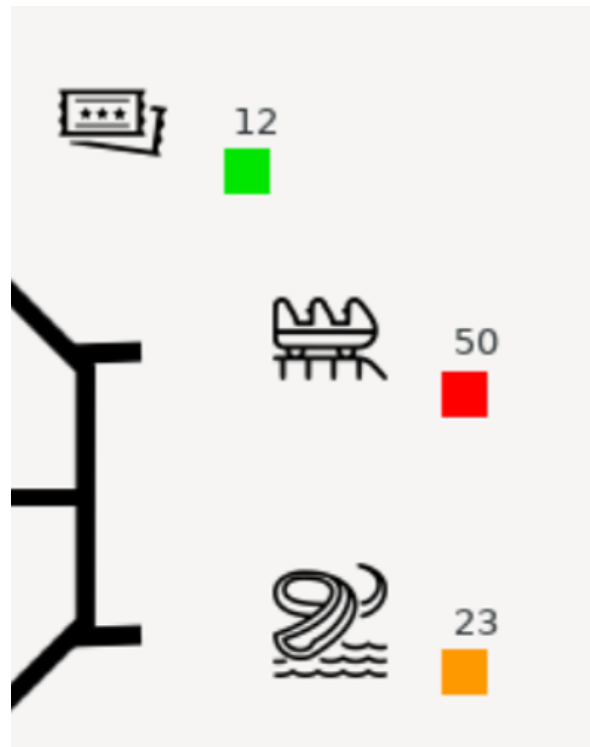
Voici un exemple de déplacement passant par une nouvelle route :



Le tracé rouge est le déplacement qu'aurait suivi le point rouge s'il voulait aller dans l'attraction en bas à droite.

Nous avons également implémenté un système de couleurs qui était prévu. Celui-ci nous permet de voir plus facilement quels attractions sont remplies ou non. Si la couleur est verte c'est qu'il n'y a personne ou peu de monde, si la couleur est orange c'est qu'il commence à y avoir du monde sinon si c'est rouge c'est que beaucoup de monde est dans l'attraction. Le calcul de cette couleur se fait en fonction du nombre total d'attractions choisi ainsi que du nombre total de personnes dans le parc et des personnes présentes actuellement dans l'attraction.

Voici un exemple des couleurs dans le parc :



Dans cet exemple l'attraction avec 12 personnes est verte, avec 23 personnes orange et enfin avec 50 personnes l'attraction est rouge.

3.6 Début de l'intelligence artificielle (Charlotte)

L'intelligence artificielle contrôle la population constituée du nombre de gens entré par le joueur initialement. Chaque personne de cette population va, en entrant dans le parc, choisir une attraction vers laquelle aller. Ce choix se fait aléatoirement. En effet, chaque attraction a un paramètre *likeness* qui représente à quel point les gens aiment l'attraction permettant grâce à des probabilités de répartir la population dans le parc.

Une fois la population répartie, les humains doivent se déplacer, comme dans un vrai parc. Pour la première soutenance et afin de rendre le tout plus facile, nous étions partis sur le principe que toutes les attractions prennent la même durée à être effectuée, 2 secondes. Afin de rendre le tout plus réaliste, nous avons finalement décidé, pour cette soutenance finale, de faire en sorte que chaque attraction ait un temps aléatoire attribués pendant leurs initialisations. Pour que chaque attraction marche séparément au lieu de dans une unique boucle, nous avons décidé d'utiliser des threads. Chaque attraction a donc son propre thread qui va prendre un nombre défini de personnes de sa file d'attente et les envoyer dans une autre attraction, ayant fini l'attraction dans laquelle ils étaient. Encore une fois l'attraction dans laquelle ils sont envoyés étant choisi grâce au paramètre *likeness* des attractions.

Les humains peuvent maintenant aussi prendre des pauses comme expliquer dans le paragraphe 3.7. Ce qui veut dire que "la pause" a aussi son propre thread et fonctionne en quelque sorte comme une attraction.

Durant le choix des attractions, il y a aussi une probabilité que la personne sorte du parc, en allant dans "l'attraction" sortie. En effet, celle-ci a aussi un paramètre *likeness* qui, lui, démarre à 0 et augmente au fur et à mesure de la journée. Ce faisant, de plus en plus de personnes sortent du parc. Nous avons aussi fait en sorte que le parc soit fermé à la fin de la journée. C'est-à-dire qu'au bout d'un certain nombre de tours de boucle, tous les visiteurs sortent du parc.

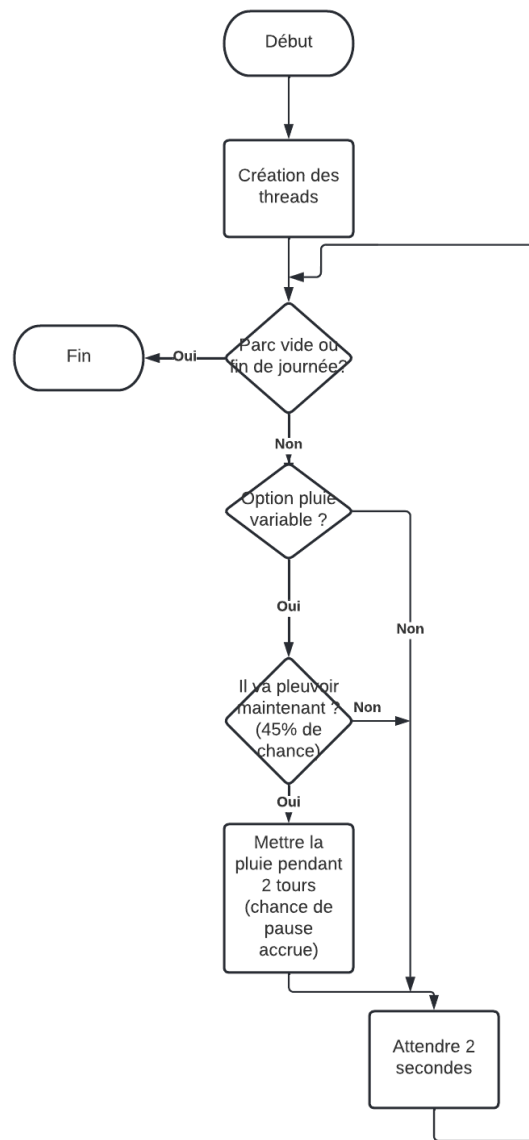


FIGURE 1 – Boucle simplifié du fonctionnement du parc

3.7 Ajout de traits humains (Charlotte)

3.7.1 Ce qui était prévu

Un facteur que nous trouvions important à prendre en compte était les traits humains. En effet, pour rendre le tout vraiment réaliste il est intéressant d'implémenter des traits humains comme la fatigue après une attraction ou encore la faim. Ces implémentations sont importantes pour pouvoir étudier une simulation la plus réelle possible même si ces implémentations passent après les autres car le plus important pour nous reste en priorité de réussir au bon fonctionnement du programme.

3.7.2 Ce qui a été fait

Concernant les traits humains, nous avons décidé de faire en sorte que les gens dans le parcs ne soit pas tout le temps dans une attraction ou en train de faire la queue. Les humains peuvent donc tous s'arrêter un certain temps à n'importe quel moment de la journée. Plusieurs facteurs peuvent jouer avec le nombre de gens qui font des pauses comme la pluie mentionner dans la partie options par exemple, ou encore l'heure de la journée. En effet, aux alentours de midi, les chances qu'ils prennent une pause sont accrues afin de mimiquer le mieux possibles une situation qui pourrait se produire dans la vrai vie dans les parcs d'attraction, qui serait un déclin du nombre de personnes dans les files d'attentes.

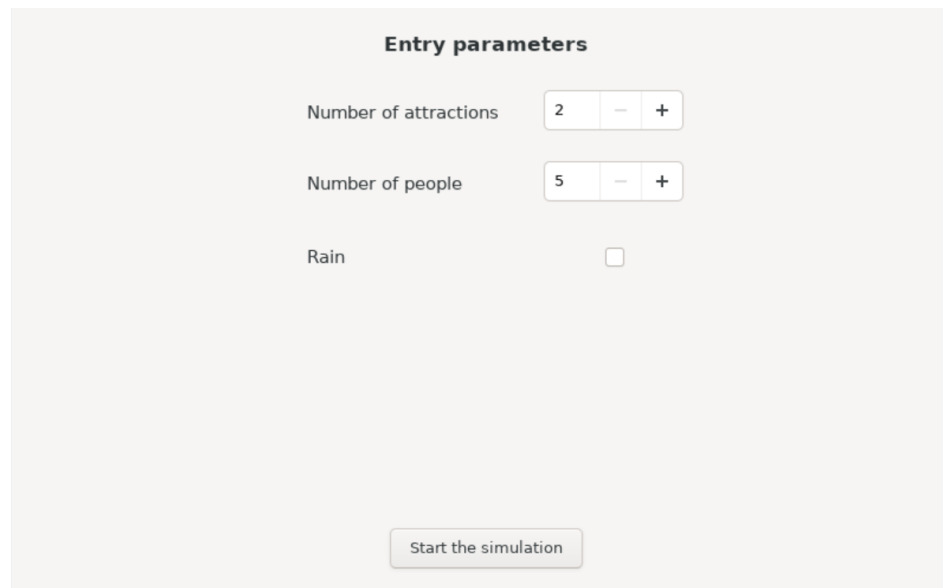
3.8 UI avec options (Nathan)

3.8.1 Ce qui était prévu

Pour rendre le logiciel plus simple d'utilisation, nous voulions faire une interface graphique, dans une esthétique simple et sobre qui donnera accès aux paramètres. Cela est beaucoup plus simple que de rentrer les paramètres via le terminal quand on lance le programme. L'interface permet de rajouter des préférences sur certaines attractions et la vitesse des humains contrôlée par l'intelligence artificielle (pour ajouter la possibilité de « courir »). L'ensemble rendra le logiciel plus modulable et permettra d'observer une grande diversité de situations.

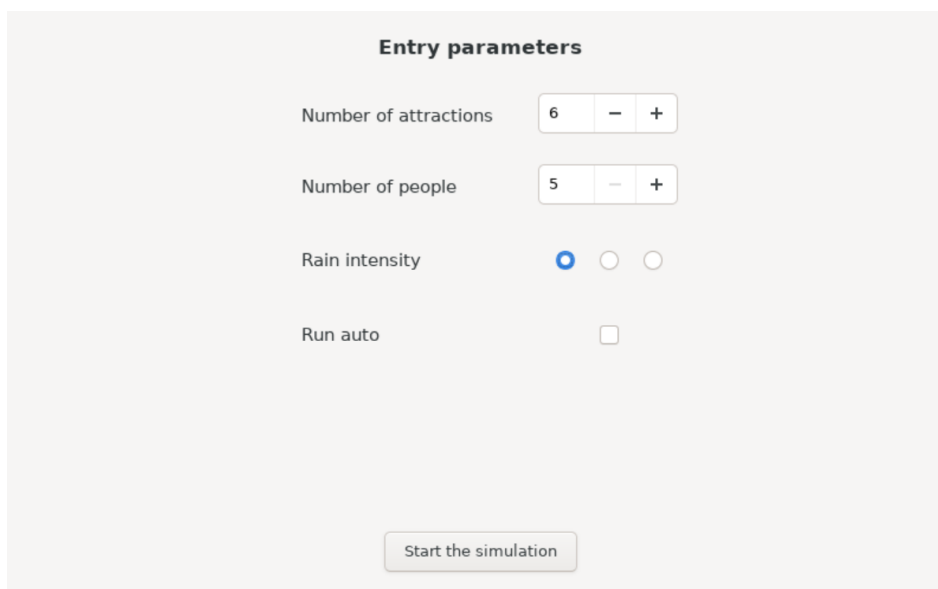
3.8.2 Ce qui a été fait

Voici l'interface que nous avons lors de la précédente soutenance :



The image shows a web-based interface titled "Entry parameters". It contains three input fields: "Number of attractions" with a value of 2, "Number of people" with a value of 5, and "Rain" with an unchecked checkbox. Each numerical field has minus and plus buttons for adjustment. At the bottom, there is a button labeled "Start the simulation".

À ce moment, seules 3 options étaient possibles : le nombre d'attractions, le nombre de personnes ainsi que le choix de la pluie ou non. Mais lors de cette soutenance l'option d'ajout de pluie n'était pas encore fonctionnel. Désormais cette fenêtre est très importante pour l'utilisateur. En effet, une fois que la personne a fini de choisir ses réglages, il suffit de cliquer sur le bouton "Start the simulation" et une fenêtre comme celles montrées dans la partie "3.3 Graphismes" apparaîtra. Rentrions maintenant dans les détails de la fenêtre actuelle, voici à quoi elle ressemble désormais :



The image shows a window titled "Entry parameters" with a light gray background. It contains four settings:

- Number of attractions:** A numeric input field with the value "6", and minus/plus buttons.
- Number of people:** A numeric input field with the value "5", and minus/plus buttons.
- Rain intensity:** Three radio buttons. The first (left) is selected (blue dot), the second (middle) is unselected, and the third (right) is unselected.
- Run auto:** An unchecked checkbox.

At the bottom center, there is a button labeled "Start the simulation".

On retrouve comme auparavant le nombre d'attractions et de personnes. Le choix de la pluie a été modifié et est maintenant fonctionnel. Maintenant on a le choix entre 3 météo. Soit, par défaut, pas de pluie (le bouton tout à gauche) ou bien avec le bouton du milieu cela devient aléatoire (un pourcentage décide de la pluie) et enfin le bouton tout à droite applique une pluie constante dans le parc. Cela va affecter le comportement des personnes dans l'attraction, elles auront tendance à sortir plus tôt ou bien à ne pas faire une attraction par temps de pluie et, en quelque sorte, s'abriter.

La grande nouveauté dans cette interface est surtout le bouton "Run auto" qui va faire que le carré rouge se déplacera de lui-même dans le parc en choisissant le meilleur chemin en fonction du nombre de personnes puis sortira.

3.9 Site Web (Charlotte)

3.9.1 Ce qui était prévu

Nous souhaitions créer un site Web pour représenter, à la fois, notre avancement tout au long de ce projet, mais aussi faire une page de présentation pour notre simulateur de trafic.

Cependant, le projet restant centré sur le côté algorithmique du simulateur, en C, nous ne souhaitions pas nous attarder de trop sur cet aspect de notre projet.

3.9.2 Ce qui a été fait

Concernant le site, nous avons donc décidé de le faire à l'aide du créateur de site Internet *Wix*.

Sur ce site, nous pouvons retrouver trois pages différentes :

- La page d'accueil, où nous présentons rapidement le projet et les membres du groupe.
- La page des téléchargements, où l'on peut télécharger le projet ainsi que les rapports de soutenance et le cahier des charges.
- La page à propos, permet quant à elle, de donner plus d'informations sur le projet.

Voici le lien de notre site :
<https://nathanlrbn.wixsite.com/projet-s4>
ainsi que des captures d'écran prise à partir de celui-ci :



Présentation

Nous sommes un groupe de 3 étudiants d'EPITA composé de **BUAT Charlotte**, **LABERNARDIERE Nathan** et **VERKINDERE Anthony**. Ce projet a été conçu pour être réalisé en équipe. L'objectif est de réaliser un programme de simulation de trafic dans un parc d'attraction en optimisant le plus possible le temps d'attente aux attractions.

Ce projet dans lequel l'algorithmique a une part très importante nous a grandement permis de nous améliorer dans ce domaine. Pour plus de détails, vous pouvez retrouver notre travail dans la section [Téléchargements](#) où vous pourrez télécharger le projet complet avec le cahier des charges et les rapports de soutenance et tester le logiciel qui a été fait.



Téléchargements

Cahier des charges



Soutenance n°1



Projet n°1

Projet n°1 (lite)

Soutenance finale



Projet final

Projet final (lite)

3.10 Répartition des tâches

Voici la répartition des tâches ci-dessous. Celle-ci n'a pas changée depuis le cahier des charges.

	Anthony	Charlotte	Nathan
Algorithmes de déplacement			
Création du parc			
Options			
Interface et graphismes			
Déplacement des Humains			
IA			
Ajout de traits humains			
Site web			



Responsable



Suppléant

3.11 Avancement

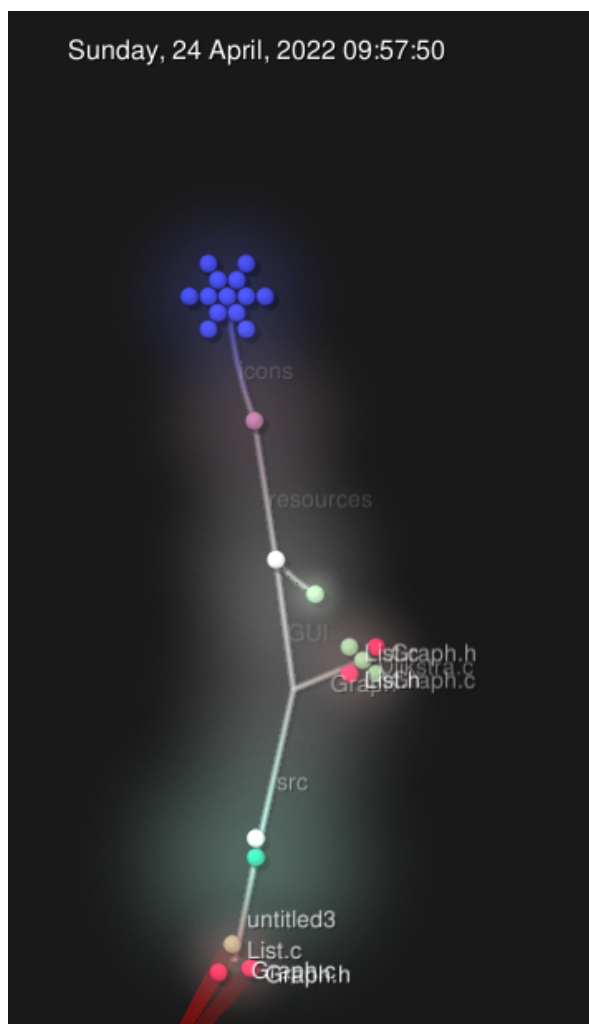
Voici ci-dessous le tableau d'avancement qui avait été fait lors du cahier des charges. Nous trouvons que dans l'ensemble nous avons plutôt bien respectés chaque pourcentage.

	Soutenance n°1	Soutenance n°2
Algorithmes de déplacement	100%	100%
Création du parc	60%	100%
Options	33%	100%
Interface et graphismes	50%	100%
Déplacement des Humains	50%	100%
IA	50%	100%
Ajout de traits humains	25%	100%

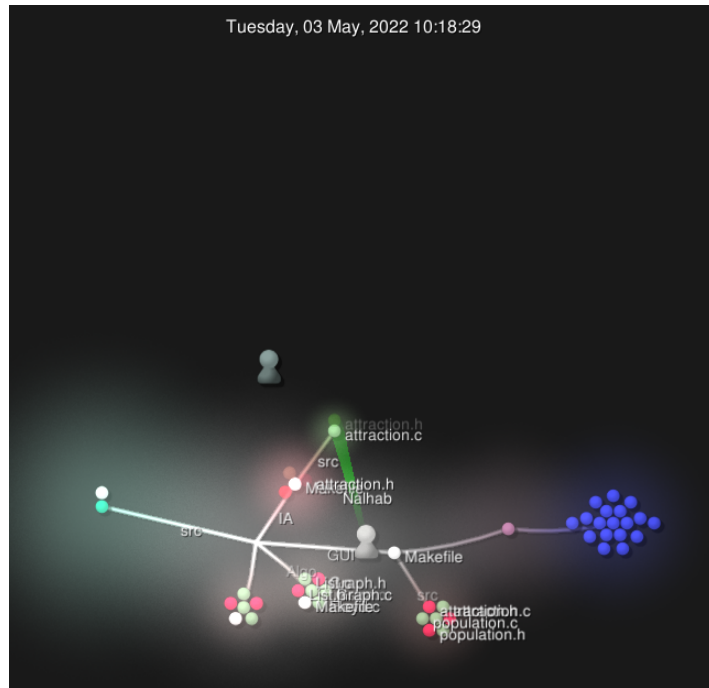
4 Structure du répertoire

La structure du répertoire Git, c'est-à dire du projet, est représentée par cet arbre ci-dessous. Cette visualisation a été générée à l'aide de l'outil Gource à partir de l'historique de modification Git. Chaque branche représente un dossier, chaque élément représente un fichier et chaque couleur représente un type de fichier.

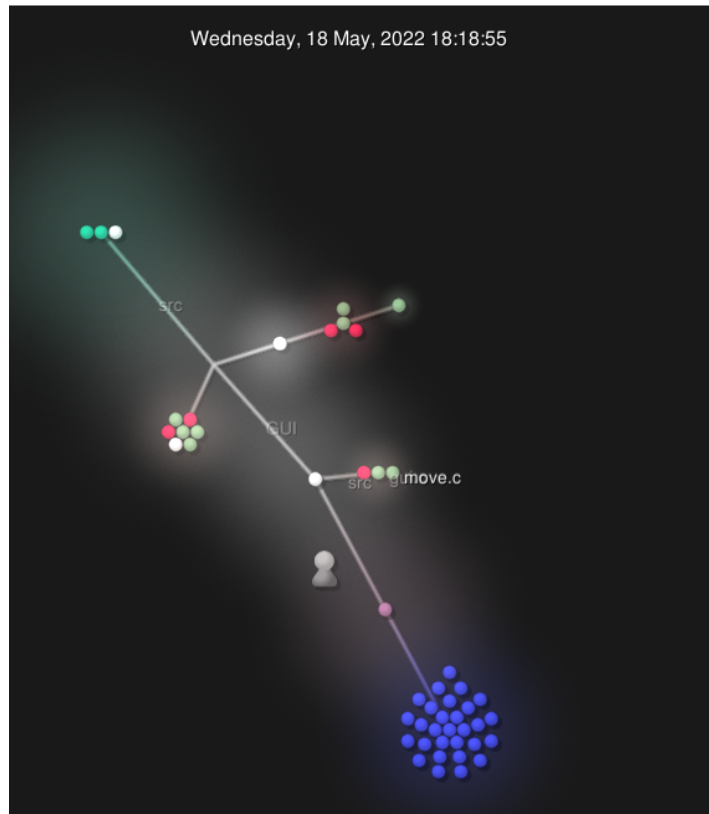
Voici l'évolution de notre répertoire au fil du temps (les dates sont indiquées sur chaque image) :

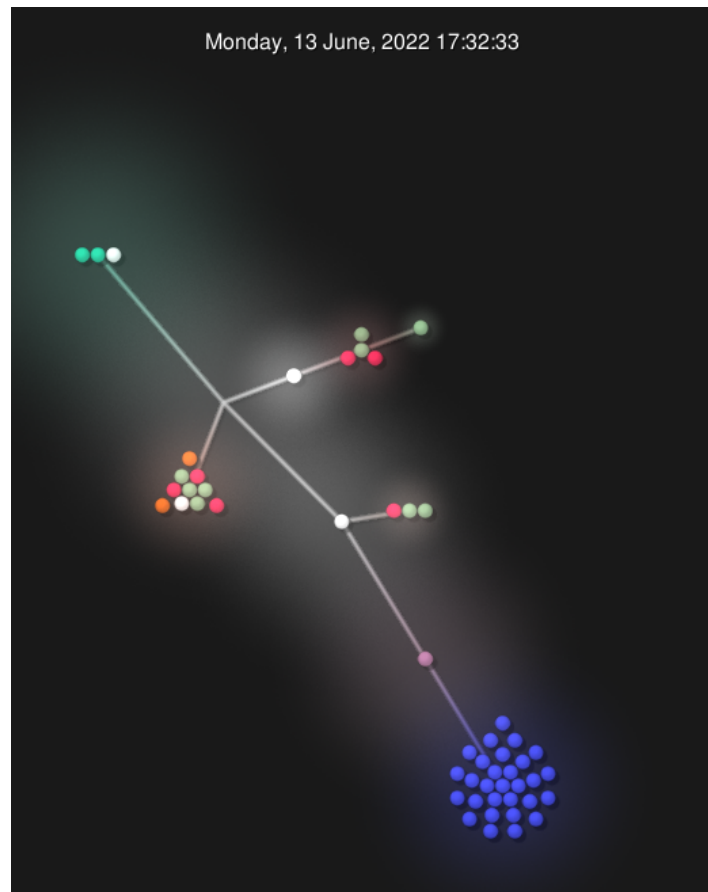


Tuesday, 03 May, 2022 10:18:29



Wednesday, 18 May, 2022 18:18:55





On peut voir qu'assez rapidement, la forme de la structure s'est stabilisée. On n'ajoutait plus beaucoup de nouveaux dossiers / fichiers on les complétait uniquement.

5 Ressenti des membres

5.1 Charlotte Buat

Ressenti lors de la précédente soutenance :

Personnellement, ce projet me faisais au début un peu peur à cause des consignes très larges par rapport au sujet et ne connaissant pas les membres de mon groupe. Au final, tous mes problèmes ont été résolus rapidement. En effet, nous avons réussi à trouver un sujet et en plus mon groupe est sympathique. De plus, nous avons aussi réussi à nous organiser et à respecter nos dates limites.

Par ailleurs, étant déjà habituée au langage de programmation C grâce au S3, la mise en place du projet s'est faite assez facilement et le travail a pu commencer plus vite que lors du S3 pour le projet OCR.

Donc le projet de S4 est pour l'instant une bonne expérience pour moi et m'apprends à travailler avec des personnes que je ne connaissais pas du tout et cela pourrait m'arriver plus tard dans le milieu du travail.

Ressenti final :

Au final, terminer ce projet m'a permis d'apprendre beaucoup de choses pour ma vie future. Par exemple, ce projet m'a appris à travailler avec des gens à distance que je ne connaissais pas. Ce qui s'est finalement très bien passé puisqu'on a réussi à bien se répartir les différentes parties du projet et à rester organisé en finissant les différentes parties en temps et en heure.

Personnellement j'ai aussi, je pense, réussi à terminer mes objectifs et à en apprendre plus sur le langage de programmation C.

Ce fut donc une bonne expérience pour moi.

5.2 Nathan Labernardiere

Ressenti lors de la précédente soutenance :

Les consignes étant plutôt vagues, j'appréhendais un peu ce projet au départ. Cependant, je me suis rapidement documenté sur Internet et j'ai réussi à avancer plus facilement que je ne le pensais. Je pense que la bonne cohésion qu'il y a dans notre groupe a permis à chacun d'entre nous de travailler efficacement et réussir ce que nous entreprenions.

De plus, je trouve cela vraiment intéressant de faire pour la première fois un projet où l'on maîtrise déjà le langage contrairement en SUP où l'on découvrait le C en même temps et en S3 où l'on découvrait le C. Ce projet me permet donc d'approfondir mes connaissances du langage C.

Pour finir, je suis plutôt confiant pour la soutenance finale car je pense que nous allons réussir à développer ce qui était prévu à l'origine.

Ressenti final :

Je trouve que ce projet ainsi que tous ceux que nous avons pu effectuer durant nos 2 années préparatoires sont les meilleurs moyens de progresser dans un langage et développer de nouvelles compétences. Au cours de ce projet, avec Anthony VERKINDERE et Charlote BUAT, nous avons travaillé ensemble régulièrement et ce dès le début.

En nous confrontant directement face à une tâche conséquente, cela nous a ainsi mis face à plusieurs problèmes à résoudre.

Ainsi, au final je suis satisfait de notre travail fourni, de ce que nous avons réussi à produire et aussi des liens que nous avons réussi à souder pour former une bonne équipe. Je n'hésiterai pas à recommencer une expérience similaire !

5.3 Anthony Verkindere

Ressenti lors de la précédente soutenance :

À première vue, le projet faisait assez peur car nous n'avions pas vraiment d'idée de ce que nous pouvions faire. Finalement, nous avons réussi à trouver une bonne idée qui validait les consignes du projet de S4. Au début du projet, j'ai eu un peu peur que le code et les algorithmes attendus soient difficiles à implémenter. Mais avec les bases du semestre 3 et les compléments du semestre 4 à notre grand bonheur cela a été plus facile que prévu. De plus, nous avons été plutôt bien organisés à trois et avons quasiment réussi à travailler sur le projet chaque semaine tous ensemble. Ce qui nous a fait avancer petit à petit sans avoir de surcharge de travail avec la soutenance. Au final, même en distanciel il y a une très bonne entente dans le groupe même si réussir à se mettre au travail dans les conditions actuelles reste assez difficile.

Ressenti final :

La finalisation de ce projet a tout simplement été un plaisir. Premièrement, grâce au groupe où nous avons réussi très tôt à s'organiser pour travailler de manière régulière tout en s'aidant les uns les autres. Deuxièmement, grâce au langage de programmation que nous commençons à maîtriser plus que au s2 avec unity par exemple. Et pour finir, grâce à notre projet, qui est je trouve très intéressant. Ce projet m'a encore énormément appris que ce soit la capacité à me mettre au travail ou à organiser mon code pour qu'il soit plus lisible par une autre personne ou encore moi même. Ce serait avec plaisir de recommencer un projet comme celui ci !

6 Conclusion

Nous avons enfin terminé la dernière étape de la conception de ce projet d'algorithmique. Cela fut éprouvant mais cela nous a permis d'en apprendre beaucoup et que cela est utile. Pour ce qui est du projet en lui-même, ce dernier peut s'apparenter à celui que nous avons pu réaliser au cours du S3. Cependant, le fait de cette fois pouvoir travailler sur un sujet qui nous est libre rappelle la liberté du projet de S1 / S2 qui est vraiment pratique pour développer quelque chose qui nous tient à cœur.

Il est important de souligner que nous avons réussi à pallier plusieurs problèmes rencontrés à la précédente soutenance. Nous avons notamment réussi grâce aux outils Glade et GTK, à créer améliorer notre interface et nous avons également du point de vue algorithmique rendu notre projet plus réaliste. Malgré les quelques détails et options que nous aurions bien aimés pouvoir ajouter.

Pour conclure, nous sommes heureux d'enfin pouvoir voir notre projet finalisé et fiers de tout le travail qui a été accompli pour cela.

7 Bibliographie

- * "Classe éco été: comment réduire les files d'attente au parc d'attraction?"
Franceinfo 2013, Alexandre DELAIGUE
- * "Parcs d'attractions et files d'attente" Tejix, Henry CORRADO
- * Floyd Algorithm
- * Dijkstra Algorithm
- * Floyd Algorithm