Visualisation de données BIM avec Unity 3D

Thomas DURAND

7 avril 2015 - 30 septembre 2015

Maître de stage :
Daniel FORGUES

Accompagnateur ECN *:*
Jean-Yves MARTIN

Travaux de fin d'études 2015 – Montréal

Table des matières

Remerciements

Avant d'aborder ce rapport, je souhaiterais remercier toutes les personnes qui ont participé de près ou de loin à ma mobilité et à mon stage, tout particulièrement mon maître de stage M. Daniel Forgues ainsi que toute l'équipe du GRIDD de l'ETS, dont Mme. Souha Tahrani, Mme. Isabelle Justras et Mr Jean-Francois Fortin Tam.

Je souhaite également remercier chez Pomerleau : Mme Ivanka Iordanova et Mr. Fernando Valdivieso, ainsi que toute l'équipe BIM, dont Mr. Jean-François Dupuis, Mr. Michael Léonard, Mme. Lieu Dao, Mr. Isaac Charbonneau Beaulieu, Mr. Eric Lacelle et Mlle. Gulnaz Aksenova.

Je tiens également à remercier Mr. Vincent Tourre de l'école Centrale de Nantes sans qui je n'aurais pas eu accès à ce stage, Mme Myriam Servières et Mr. Jean-Yves Martin, mon accompagnateur ECN, et responsable de l'option Informatique.

Enfin, j'en profite également pour remercier les étudiants du club ApplETS, dans lequel j'ai également beaucoup appris dans le domaine de l'ingénierie mobile.

Introduction

Le laboratoire GRIDD

Le GRIDD, ou Groupe de Recherche en Intégration et Développement Durable en environnement bâti, est un laboratoire de recherche composé de chercheurs et d'étudiants de l'ETS [1]. [4]

L'objectif du GRIDD est modifier les façons de pratiquer et d'enseigner les principes de la construction. Il s'agit principalement de veiller autour des nouvelles techniques de développement dans les bâtiments, et également transmettre cette connaissance afin de permettre une réduction des coûts, et une amélioration de la qualité de l'environnement bâti lors du développement et la construction de nouvelles infrastructures.

Mon stage a été effectué au sein du GRIDD pour développer en collaboration avec Pomerleau une nouvelle solution de visualisation des données BIM [2] pouvant fonctionner indépendamment des logiciels propriétaires actuellement utilisés.

Pomerleau

Fondé en 1964, Pomerleau est un groupe important dans le domaine de la construction au Québec. Il offre des services allant de la pré-construction jusqu'à la mise en service des équipements. Avec un peu moins de 4000 employés, dont 2600 au Québec, Pomerleau se classe 56$^{\text{ème}}$ dans le classement des plus grandes sociétés du Québec. [7], et 5$^{\text{ème}}$ plus gros constructeur du Canada.

Depuis 2012, Pomerleau est en chaire avec le GRIDD afin de doter le Québec d'un centre de veille et de transfert technologique pour l'industrie de la construction. [3]

Les axes de recherche autour de cette chaire sont principalement autour de la modélisation des données du bâtiment (BIM), de la conception intégrée (CI) et du Lean qui permet de

1. École de Technologie Supérieure de Montréal
2. Modélisation des données du bâtiment, voir ANNEXE A

baser l'approche du bâtiment sur la gestion de la production, et donc de maximiser la valeur et réduire au maximum le gaspillage des ressources naturelles et financières.

Les retombées de cette chaire sont déjà visibles sur certains projets comme l'Amphithéâtre de Québec, renommé Centre Vidéotron, et présentent une meilleure valeur ajoutée, et une meilleure qualité pour des coûts réduits.

L'opportunité de développement autour de BIM

Le BIM est une méthode collaborative de travail, se basant sur une modélisation d'une puissance étonnante, permettant une hiérarchisation des données d'un bâtiment impossible à obtenir avec d'autres méthodes de conception dans le domaine de la construction.

Cependant, comme tout environnement informatique, le nombre important de fonctionnalités oblige une formation pour l'utilisation de ces logiciels complexes. C'est le cas par exemple du logiciel Revit, acheté et maintenu par Autodesk, qui permet la réalisation d'un bâtiment et l'association de ses données, afin de créer un modèle BIM depuis le début.

Or, certains usages courant du BIM deviennent difficile d'accès, par l'existence même de toutes les autres fonctions.

Le sujet

Face à la complexité de certains logiciels liés à BIM, Il y a une grande opportunité de disposer d'un programme permettant l'accès aux données de façon plus simple et ergonomique, en s'adaptant à seulement quelques cas d'utilisations.

Dans cette perspective, le but de mon stage était de réaliser un pont entre les données BIM réalisé par le logiciel professionnel Revit, avec un petit programme basé sur le moteur de jeux-vidéos Unity3D ; permettant quelques usages simples du BIM, comme par exemple pour permettre l'exploration du projet.

Il s'agissait également de prendre des choix sur l'ergonomie et les simplifications qu'il était possible d'apporter sur les actions préalablement choisies.

Enfin, il s'agissait de livrer ce programme, configurable en fonction des cas d'utilisation, et dont il sera possible par la suite de reprendre le développement pour ajouter des fonctionnalités ou des nouveaux cas d'utilisations reprenant ou pas les fonctions déjà présentes à l'heure actuelle.

Chapitre 1

Analyse des besoins et des logiciels existants

1.1 Étude de l'usage des logiciels utilisés chez Pomerleau

1.1.1 Autodesk Revit

Revit est une suite de logiciels de conception assistée par ordinateur ciblant la conception architecturale et la modélisation des données du bâtiment (BIM).

Maintenu par la société Autodesk, Revit permet la modélisation des bâtiments en 3 dimensions pour ensuite exporter des plans ou des vues. Il permet aussi d'accéder et de renseigner de nombreuses informations liées à BIM sur les différents objets.

Sur la FIGURE 1.1, on peut voir un projet exemple ouvert dans le logiciel Revit. La vue 3D est orthographique, les matériaux sont représenté par des couleurs simplifiées, et l'environnement est adapté à la modélisation architecturale du bâtiment.

Des informations apparaissent dans un panneau sur la gauche, et la totalité des outils sont disposés sous forme de ruban en haut du logiciel.

Cependant, le logiciel est difficilement utilisable sans formation préalable. Un simple mouvement de caméra peut se révéler difficile : il s'agit de maintenir le clic en déplaçant la souris pour se déplacer selon les axes x et y de l'écran, et utiliser la molette pour s'approcher ou s'éloigner. Enfin, il faut en plus maintenir la touche Shift gauche pour faire pivoter la caméra.

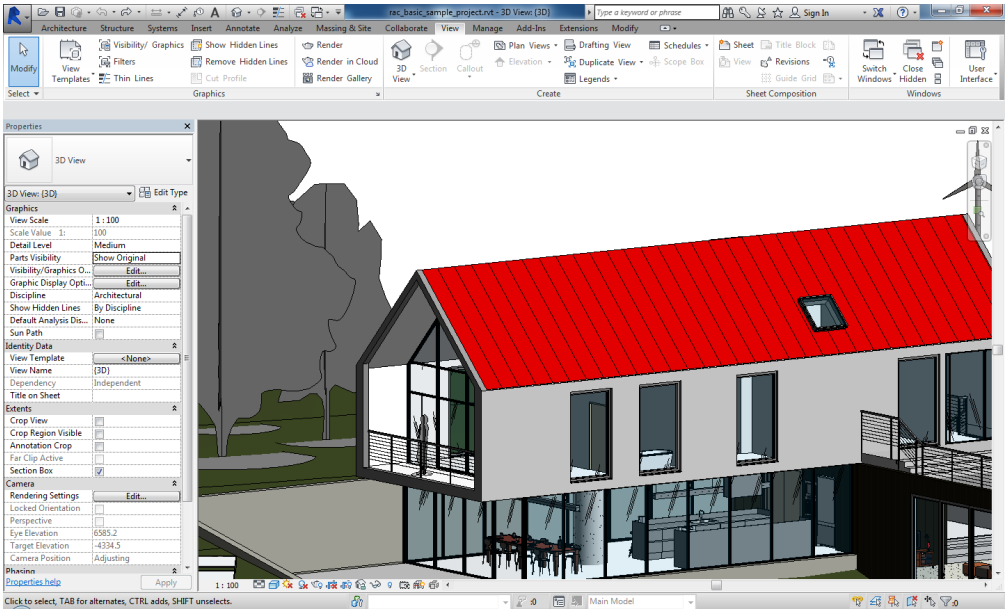


FIGURE 1.1 – Un projet exemple, ouvert dans Revit

1.1.2 Autodesk Navisworks Simulate

Navisworks est un logiciel de revue de fichiers 3D, capable d'ouvrir et de combiner de nombreux formats de fichiers, dont les fichiers Revit.

Comme Revit, il a été racheté et est désormais maintenu par Autodesk.

La FIGURE 1.2 montre le projet exemple ouvert précédemment dans Revit, tel que montré dans la FIGURE 1.1. La vue est en perspective, et les matériaux sont texturés pour un rendu plus réaliste.

Le rôle de Navisworks étant simplement la visualisation, les outils proposés dans le ruban supérieur sont bien moins nombreux. Pourtant, paradoxalement, se déplacer dans le modèle et réaliser les mouvements souhaités demandent un certain coup de main, d'autant plus que les contrôles ne sont pas les mêmes que dans Revit.

Navisworks dispose également d'outils permettant de relier un échéancier *Microsoft Project* à différents objets afin de pouvoir animer et simuler toutes les étapes du chantier d'un bâtiment, ainsi que d'autres outils très puissants permettant la réalisation de sélections d'objets, manuellement, ou automatiquement à partir du filtrage des données BIM sur de nombreux critères.

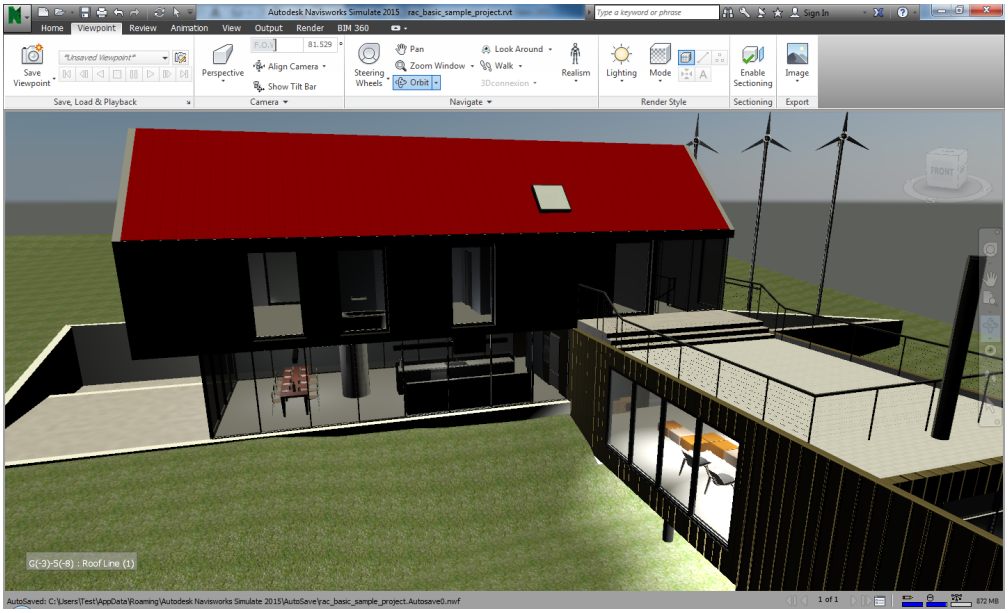


FIGURE 1.2 – Le même projet exemple, ouvert dans Navisworks Simulate

1.1.3 Limites de ces outils professionnels pour des cas d'utilisation simples

Malgré leur puissance évidente, et leur forte part de marché dans le domaine du BIM, Revit et Navisworks posent le problème souvent délicat d'être "trop complets".

Par exemple, comparons Microsoft Paint, et Adobe Photoshop. Le second sera bien plus complet et proposera des outils de retouche bien plus adaptés à la photographie professionnelle. Cependant, le premier, bien plus simple, est plus rapide à prendre en main, et est amplement suffisant pour certains cas d'utilisation, comme l'annotation rapide d'une image.

Dans le cas de Revit, nous sommes dans un logiciel complet et incontournable, mais de nombreuses actions simples en deviennent difficile d'accès et nécessitent une certaine formation, ou à défaut, un certain nombre d'explications sur le fonctionnement global du logiciel.

Or, il n'est pas possible pour toutes les personnes dans le besoin d'accéder à BIM et les information qu'il contient de suivre une formation pour connaitre le fonctionnement de ces logiciels ; de plus, la licence de ces logiciels impose un coût qui dépend du nombre de machines sur lesquelles il est installé. [6]

Une autre limite de Navisworks sera la durée de calcul nécessaire pour générer un aperçu réaliste du projet. Un outil permet par exemple la génération d'images pour un projet, mais comme montré en ANNEXE B, cette génération peut-être longue.

D'autres outils permettent la génération de vidéos, de présentation, mais la encore, outre le temps de calcul long, cela ne permet pas une exploration libre en temps réel.

L'utilisation d'un moteur de jeu vidéo permet cette exploration en temps réel d'un modèle 3D, avec certes des limitations dans le domaine du réalisme de certains effets (reflets, ombres), mais permettant en outre de se promener librement, de pouvoir profiter d'outils puissant et inédits, comme l'utilisation de la réalité virtuelle (avec, par exemple, un casque Oculus Rift).

1.2 Analyse des besoins pour un logiciel tiers

La première phase de conception d'un logiciel correspond à la définition des besoins et des fonctionnalités nécessaires. La totalité de ces besoins correspondent à des tâches qui sont déjà réalisées avec d'autres logiciels, cependant limiter le logiciel à quelques tâches simples permettra de rendre ces tâches plus simple à utiliser.

1.2.1 L'aspect modulaire au service de la simplification

Outre l'utilisation d'un moteur 3D de jeu vidéo pour supporter tout le domaine du calcul dans l'espace, la principale simplification qui sera apporté à ce logiciel sera son adaptabilité.

À l'opposé d'un logiciel comme Revit, proposant en une fois toutes ses fonctionnalités, l'application sera développée de façon modulaire, permettant alors d'ajouter ou de supprimer des modules en fonction des besoins, et assurer de ne pas proposer des fonctions inutiles à l'utilisation voulue du logiciel, pouvant alors perturber et complexifier l'utilisation de l'application.

Une autre force de cet aspect modulaire sera la possibilité d'ajouter de nouvelles tâches inédites au programme, en développant les modules correspondants. Le but de ce stage sera donc la spécification d'un outil servant de fondation à l'intégration d'un projet Revit dans un environnement dédié, avec la possibilité de développer et d'ajouter de nouveaux éléments en fonction des besoins.

1.2.2 Les modules retenus pour la première phase de développement

Après discussion avec l'équipe BIM, nous avons pu définir une liste de tâches qui seront dans un premier temps à implémenter pour cette application. Ces tâches sont listés exhaustivement dans le document de conception en ANNEXE D, § D.3.

La première phase de développement du logiciel correspondra donc au développement du cœur de l'application, permettant l'import d'un modèle associé à sa base de données, et de la possibilité de parcourir simplement ce modèle.

Par suite, il sera développé les 4 modules suivants :
- Consultation d'informations liés à un objet
- Création et chargement de sélections d'objets
- Chargement et visualisation d'un échéancier de construction
- Création et enregistrement d'un échéancier de construction

1.2.3 Diagramme de cas d'utilisation

L'intérêt principal de développement d'une solution alternative est d'étendre l'utilisation de celui-ci à un plus grand nombre d'utilisateurs. On peut alors citer que les acteurs du logiciels seront :
- Le client commanditaire du bâtiment
- Les prestataires, présents sur le chantier
- Les ingénieurs BIM de chez Pomerleau

Les tâches définies n'étant pas toutes dédiées à être utilisées par tous les utilisateurs, comme l'utilisation modulaire du logiciel le laissait supposer en § 1.2.1, les rôles des différents utilisateurs du logiciel est détaillé dans un diagramme de cas d'utilisation, en FIGURE D.1.

Chapitre 2

Spécifications d'une solution alternative

2.1 Le choix du moteur Unity, et du langage de programmation

De nombreux moteurs 3D existent dans le domaine de l'informatique, cependant les deux qui étaient retenus pour ce projet étaient Unity3D et Unreal Engine.

Suite à une étude préliminaire dont les résultats sont visibles en ANNEXE C, le choix s'est finalement, pour des raisons techniques, porté sur Unity3D ; en effet, Unity3D permet de conserver un accès aux identifiants des objets.

Une autre raison est que le langage de programmation utilisé avec Unreal Engine est le C++, langage qui n'est utilisé par aucun des développeurs de l'équipe BIM. Unity3D permet la programmation en Javascript et en C#Ċe dernier étant déjà utilisé pour le développement d'extensions pour Revit et Naviswork, il est plus adapté pour l'équipe BIM. De plus, le C# est plus flexible et plus proche du language objet que le Javascript, ce qui justifie le choix de celui-ci.

Bien qu'Unreal Engine serait totalement gratuit pour sa version professionnelle, les besoins de Pomerleau ne justifient pas pour l'instant l'utilisation de la version professionnelle d'Unity. La version gratuite suffit alors.

2.2 Compatibilité entre Revit et Unity

Cette étape d'analyse, effectuée en parallèle de l'étude des différents moteurs exploitables, permettait de trouver et vérifier les différentes façons d'importer les données 3D et les informations BIM dans le moteur. [8]

Parmi la totalité des formats d'exportations proposés par Revit, on trouve le format FBX, un format de fichier 3D propriétaire supporté nativement par Unity. C'est ce format qui a été choisi pour exporter le modèle en trois dimensions depuis Revit vers Unity.

En ce qui concerne les données associées aux objets, Revit ne propose pas d'exportation de sa base de données par défaut, mais il est possible de le faire à l'aide d'extensions comme *BIMLink* ou *RevitDbLink*. Cette exportation permet d'obtenir une base de données au format Access, ou au format ODBC, formats que l'on peut par la suite convertir au format MySQL, ou PostgreSQL.

Pour l'exploitation de ces formats de base de données dans un programme intégré à Unity, il a été possible de faire fonctionner des bases de données ODBC, MySQL et PostgreSQL ; en revanche, malgré un certain nombre d'efforts, il n'a pas été possible d'exploiter les données des bases de données au format Microsoft Access. (voir § 3.3.2)

Ce dernier cas a demandé 2 ou 3 implémentations différentes avant de pouvoir ouvrir la base, et sur la dernière implémentation, il n'est possible d'accéder qu'aux données numériques (entiers, ou décimaux), mais impossible de récupérer une chaîne de caractères. Pour cette raison, la compatibilité avec format Access a été abandonnée.

2.3 Structure du programme et classes

Le modèle étant dissocié de ses données lors de l'exportation, il est nécessaire de prévoir une classe dédiée à les réunir de nouveau. Cette classe est le « ModelHandler » elle assure le lien entre la base de données et les objets du modèle, comme présenté en FIGURE 2.1.

La base de données sera représentée par une classe, prévue pour faire le pont entre le ModelHandler et la base de données réelle. Cette classe dépendra du type de base de données configurée pour fonctionner avec le programme (MySQL, Access... etc)

Le ModelHandler va fonctionner en parallèle avec différents modules qui vont gérer les différentes fonctionnalités du programme. Cette séparation permet de désactiver un ou plusieurs comportements en fonction des besoins, mais permet aussi le développement futur de nouveaux modules, pour étendre les fonctionnalités du programme.

Outre la gestion du modèle et de ses données, il est aussi nécessaire de gérer les mouvements de l'utilisateur dans l'espace et de lui donner le contrôle sur différents paramètres comme la désactivation de la gravité ou des collisions. Cela sera géré par la classe « PlayerHandler » qui interagira directement avec les périphériques d'entrée (Clavier et souris dans un premier temps) et les objets représentant l'utilisateur virtuel, comme sa caméra, et sa lampe frontale.

Pour finir, le programme comportera des classes chargées de représenter des ensembles d'objets ou des étapes de constructions, afin de fonctionner étroitement avec les comportements implémentés.

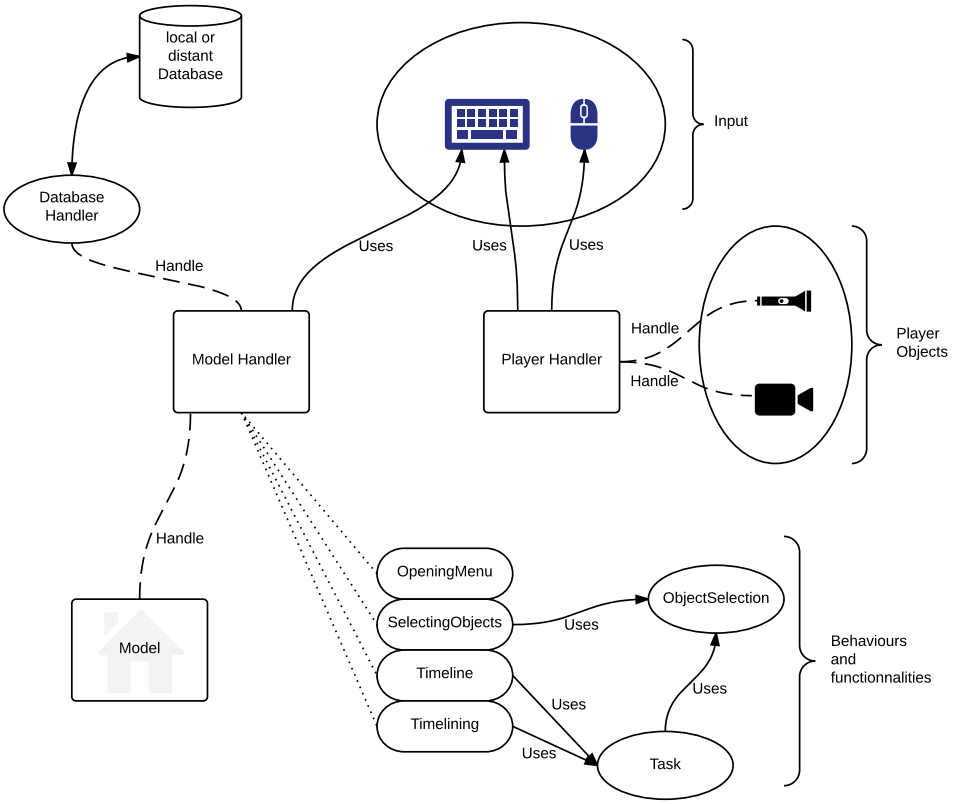


FIGURE 2.1 – Structure de fonctionnement du programme.

Enfin, de nombreuses classes utilitaires seront mises en places, dont le but sera de simplifier le code, et / ou d'empêcher les répétitions inutiles dans le code. Ces classes ne sont pas représentées dans la FIGURE 2.1.

2.4 Contraintes ergonomiques

L'intérêt de la programmation avec Unity est la portabilité de l'application sur différents supports, et différentes combinaisons de matériels.

On peut envisager l'utilisation sur un traditionnel ordinateur de bureau avec un clavier et une souris, jusqu'à l'utilisation avec un Oculus Rift et une manette de console de jeu, en passant par l'utilisation sur une tablette tactile.

Cette diversité est bien évidemment une contrainte pour l'interface utilisateur, dont l'utilisation tactile sera évidemment différente d'une utilisation avec le couple clavier et souris.

Bien que la solution tactile n'ai pas encore été intégralement implémentée, notamment pour ce qui est du déplacement dans le modèle, les éléments graphiques ont été pensés pour une utilisation tactile, afin de faciliter le développement futur de cette possibilité.

De plus, la nécessité d'obtenir un fonctionnement plus accessible que Revit ou Navis-work impose aussi des contraintes ergonomiques liés à la facilité d'utilisation et de compréhension du programme.

2.5 Opportunité de développement d'une structure de données alternatives

Lors de l'étape d'ingénierie inverse de Revit, nous avons constaté que la base de données utilisée par Revit pour décrire les données BIM est très complexe, et stocke des données similaires à des endroits différents. On retrouve également des problèmes de duplication de données, qui peuvent mener à des contradictions dans la base de données.

C'est pourquoi il pourrait être intéressant pour Pomerleau de disposer d'une structure de données plus adaptée à ces propres cas d'utilisation, avec un niveau de conception évitant la duplication des données ou l'impossibilité de savoir dans quelle table sont stockées les données relatives à un objet.

La FIGURE 2.2 représente un schéma physique qui pourrait être utilisé comme base pour le développement de cette nouvelle base de données. Bien qu'il sera nécessaire d'ajouter des attributs, et quelques tables supplémentaires, ce modèle a le mérite de ne stocker les

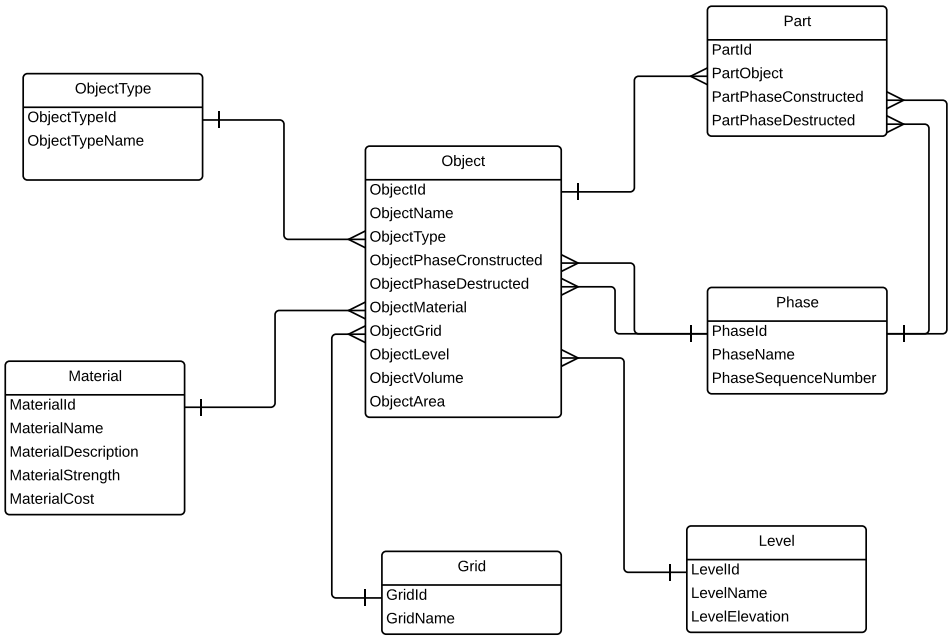


FIGURE 2.2 – Exemple d'une structure de données simplifiée pour BIM

objets que dans une seule table, et de référencer leur type ; contrairement à la base de données de Revit qui dispose d'une table différente par type d'objet, ce qui rend plus difficile d'accéder aux données de cet objet uniquement avec son identifiant, la recherche de cet identifiant dans toutes les tables d'objets étant alors nécessaire.

En attendant une base de données avec une structure cohérente, ce problème de performances a été réglé tel que décrit en § 3.3.4

Chapitre 3

Développement avec Unity3D

3.1 Familiarisation avec Unity

N'aillant aucune expérience avec Unity avant le début de mon stage, l'une des premières tâches nécessaires était de m'adapter à l'environnement de développement, et aux API [1] fournies par Unity et par .NET.[9]

Pendant les premières semaines du stage, j'ai pu comprendre comment fonctionnait le système de comportements de Unity, comment en créer, et ainsi créer quelques modèles simples de caméras interactives, et d'objets animés.

3.2 Premiers imports et applications : L'amphithéâtre de Québec

Peu de temps après les premiers résultats positifs avec Unity, on m'a fourni le fichier correspondant à l'Amphithéâtre de Québec. (renommé depuis en Centre Vidéotron) dans le but de réaliser des tentatives d'exportation de projets depuis Revit.

L'idée était de pouvoir utiliser Unity avec le modèle de ce stade de hockey sur glace, et lier ce modèle avec les données de coordonnées de sièges pour pouvoir y ajouter un modèle simple de personnes, et la possibilité d'avoir un aperçu de la vue depuis un siège donné.

La FIGURE 3.1 donne un aperçu de ce à quoi ressemblait le centre Vidéotron une fois importé dans Unity, la totalité des matériaux et textures utilisées ont été configurés et ajoutés manuellement à la scène.

1. Application Programming Interface : ensemble normalisé des classes et de méthodes propres à un language et à un environnement de développement.

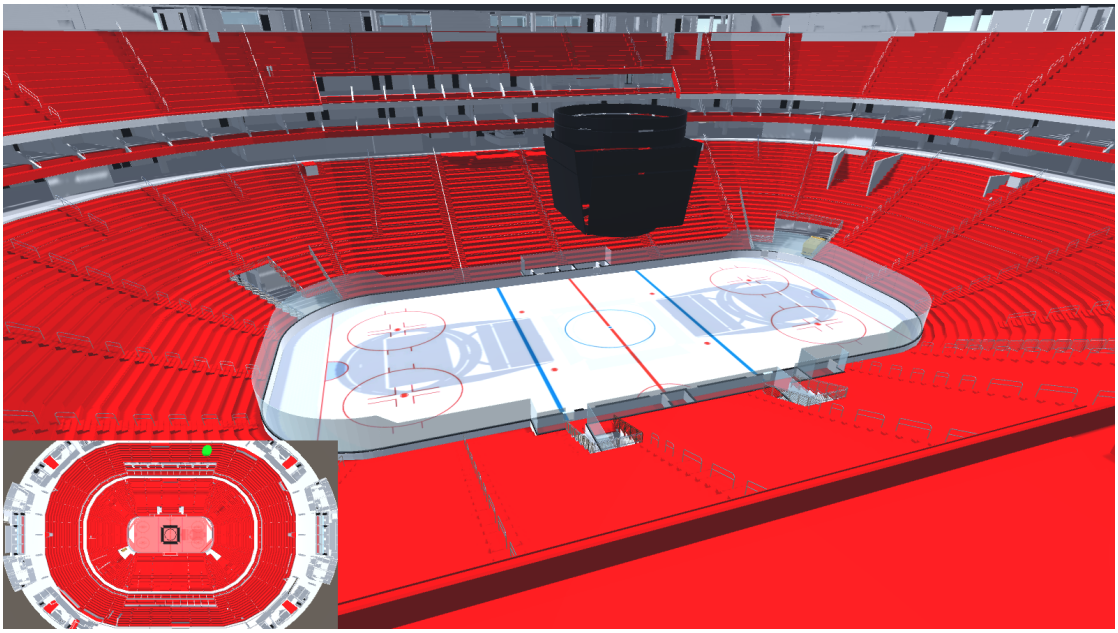


FIGURE 3.1 – Le Centre Vidéotron importé dans Unity

En ce qui concerne les coordonnées de sièges, celles-ci ont été exporté depuis Revit dans un format de type csv [2] et dans le système de coordonnées de Revit.

Il s'est avéré que le système de coordonnées de Revit était différent du système de coordonnées de Unity, ce qui a impliqué le développement d'une classe dédiée à la transcription de celles-ci dans un sens comme dans l'autre.

Finalement, nous ne disposions pas des coordonnées de la totalité des sièges du stade, et cette simulation n'a pas eu plus d'utilité qu'une preuve de concept du nombre infini de possibilités qu'Unity peut offrir dans le domaine du bâtiment.

3.3 Lien avec la base de données

L'un des points forts du stage a été la capacité de rétablir à nouveau le lien entre la base de données et les objets.

3.3.1 Accès aux identifiants des objets

Par chance, le fichier 3D exporté depuis Revit comporte toujours une référence à l'identifiant de chaque objet. En effet, chaque objet comporte un nom, suivi de son identifiant

2. Comma Separated Value, un format de fichier permettant l'exportation de tableaux

entre crochet (par exemple : « structural part [423584] »). Il est donc relativement aisé, dans Unity, de récupérer l'identifiant d'un objet, mais aussi un objet à partir de son identifiant.

Dans ce but, une classe a été développée comportant un certain nombre de fonctions statiques :

- Récupération de l'identifiant à partir d'un objet
- Récupération du ou des objets correspondant à un identifiant
- Récupération de la liste des identifiants depuis une liste d'objet
- Récupération d'une liste d'objets depuis une liste d'identifiants

Ces fonctions statiques sont accessibles depuis une classe nommée « ObjectIdentifier » et seront utilisées tout au long du programme pour permettre l'identification des objets.

Elles sont basées sur l'utilisation d'expressions régulières sur la liste des objets de Unity. De plus, pour améliorer les performances, les identifiants disponibles, associés à leurs objets respectifs sont stockés en mémoire, l'itération dans une liste en mémoire étant plus rapide que l'itération sur les objets disponibles dans la scène, associé à la recherche de l'expression régulière.

3.3.2 Premières tentatives avec Microsoft Access

Le logiciel utilisé pour récupérer la base de données de Revit proposait une exportation au format Microsoft Access. C'est donc naturellement que les premières tentatives d'accès à la base de données s'est portée sur ce format.

Cependant, les incompatibilités entre Mono.NET [3] et les API de Microsoft Office ont rendu difficile l'ajout du support de Microsoft Access.

Après avoir trouvé une solution alternative capable d'ouvrir la base de données Access, nous nous sommes rendu compte que seuls les nombres, entiers ou décimaux, étaient récupérables ; nous perdions toute l'information des chaînes de caractères, ce qui n'était pas compatible avec l'utilisation que nous souhaitions faire du programme.

La base de données au format Access a donc été mise de coté, celle-ci ne fonctionnant pas intégralement.

3.3.3 Solutions avec d'autres technologies

Pour résoudre le problème d'accès à la base de données, nous avons converti celle-ci dans divers formats (MySQL, PostgreSQL, Oracle), afin de vérifier si ces formats là étaient plus convaincants.

3. Mono.NET est une implémentation libre, mais moins complètes, des librairies de Microsoft .NET

Une fois les classes pour ces différents types de bases de données implémentés, toutes ces classes dérivant d'une même classe abstraite, nous avons pu tester quelques requêtes SQL depuis le programme pour s'assurer que celles-ci fonctionnaient correctement, ce qui fort heureusement était le cas.

3.3.4 Récupération des informations d'un objet

Comme décrit en § 2.5, la base de données fournie par Revit n'est pas optimale, et gagnerait à être modifiée en profondeur dans sa structure.

En attendant, il était nécessaire de connaitre le type de l'objet pour savoir dans quelles tables se trouvent les informations qui y sont liées car il y a une table distincte pour chaque type d'objets (il y a les tables "Walls", "StructuralElements"... etc).

Une première solution à ce problème, naïve, consiste à chercher l'objet depuis toutes les tables possibles ; soit plus d'une centaine ; à chaque fois qu'il est nécessaire de récupérer les informations d'un objet. Cette solution n'a pas été retenue, car elle aurait posé des problèmes évidents de performances à chaque requête d'accès aux données.

Une seconde solution, étant celle retenue, est de "scanner" toutes les tables existantes, dans le but de construire en mémoire une table associative des identifiants et de la table qui les contient.

Comme le présente la FIGURE 3.2 sous la forme d'un organigramme, à chaque fois que l'on souhaite récupérer la table correspondant à un identifiant, on identifie si le tableau associatif est bien en mémoire. Si oui, on l'utilise pour retourner la table auquel il appartient. Dans le cas contraire, on construit cette table d'association, en effectuant une requête par table. Cette opération n'est pas instantanée, mais elle se fait beaucoup moins souvent que la requête de données d'un objet.

3.3.5 Récupération des matériaux

L'une des incompatibilités les plus gênantes entre Unity et Revit réside dans le traitement des matériaux : le modèle de matériaux utilisé par Revit n'est pas compatible avec ceux utilisés par Unity.

Conversion

La solution de la conversion consiste à convertir les matériaux utilisés dans Revit afin qu'ils deviennent compatibles avec Unity. Cette solution est possible à l'aide d'un plug-in payant qui vient se greffer à 3DSMax, un logiciel de conception en 3 dimensions.

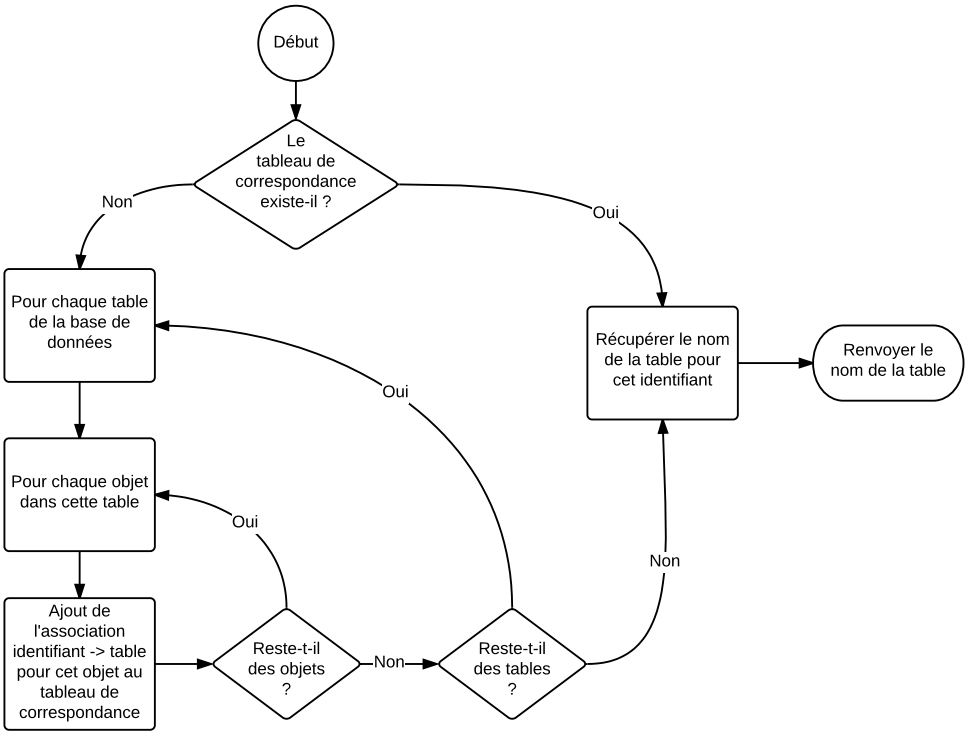


FIGURE 3.2 – Mécanisme de récupération de la table contenant les infos d'un objet

Cette solution n'a pas été retenue, ni testée, car en plus d'être payante, elle nécessite l'ajout d'une étape et d'un autre logiciel nécessaire lors du processus d'exportation du modèle depuis Revit vers Unity, complexifiant la tâche. [2]

Récupération automatique

La solution finalement retenue pour la récupération des objets est une récupération automatique de ceux-là depuis les informations stockées dans la base de données.

La référence du matériau utilisé pour un objet étant dans une table, il s'agit donc de récupérer le titre du matériau dans la base, et la comparer avec un répertoire de matériaux prédéfinis, créés spécialement pour Unity.

La principale limite de cette méthode est qu'il s'agit de créer une base de matériaux avec un certain nombre de mots clés.

L'avantage par contre est que cette base, déjà fournie avec quelques matériaux clés, pour le verre, le béton, ou le plastique, permet d'avoir rapidement un résultat correct.

Choix manuel des matériaux

Dans le dernier mois de stage, alors que nous testions le programme avec différents projets Revit, et que nous étudions les possibilités de réalité virtuelle, nous nous sommes rendu compte qu'une adaptation du programme était nécessaire pour pouvoir "imposer" un matériau donné.

La nécessité est causée par certaines limites du script de reconnaissance automatique des matériaux : certains objets sont assignés à plusieurs matériaux ; c'est le cas par exemple de certaines baies vitrées, qui auront du verre, mais aussi un cadre d'aluminium ; le script pourrait alors assigner un aluminium mat au lieu d'une vitre.

Quelques conditions ont donc été ajoutées au script d'assignation de matériaux, et dans son comportement par défaut, il ne tentera d'assigner un matériau automatiquement si et seulement si l'objet en question est vierge de matériaux dans la scène.

Cela offre la possibilité d'ajouter manuellement des matériaux dans la scène, et donc de pouvoir préparer la meilleure expérience possible pour une visite d'un bâtiment en réalité virtuelle par exemple.

3.4 Implémentation du déplacement du personnage

3.4.1 Modélisation du personnage

L'application aillant été prévue comme un jeu à la première personne, le personnage n'aura pas de modèle visible. Cependant, il doit comprendre une "boite de collision", une caméra, et une source de lumière (lampe frontale)

Le modèle choisi pour l'application est schématisé en Figure 3.3. Il comprend un boite de collision en forme de capsule, un caméra liée à cette capsule, et une lampe liée à la caméra. Seule la rotation autour de l'axe vertical est possible pour la capsule. La caméra, elle, peut également pivoter sur son axe de tangage. (mais pas de roulis)

Ainsi, la caméra suit le mouvement de la capsule, et la lampe suit les rotations de la caméra, donnant l'impression d'un corps physique avec des collisions, et d'une lampe frontale sur un casque.

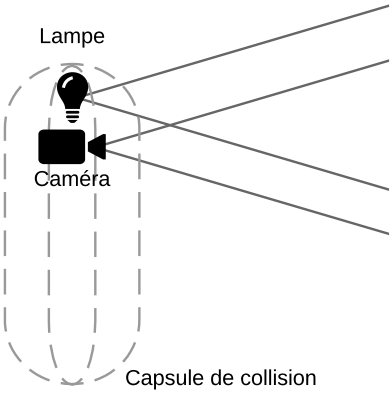


FIGURE 3.3 – Modèle du personnage retenu pour le programme

3.4.2 Récupération de la direction du déplacement

Lorsque la gravité n'est pas prise en compte, la direction de déplacement doit correspondre à la direction de la caméra, afin de donner une impression de déplacement libre dans le modèle. Il s'agit donc de récupérer les composantes du repère local de la caméra pour connaitre les vecteurs du déplacement.

Néanmoins, lorsque la gravité est désactivée, il faut supprimer la composante verticale de la caméra pour obtenir la direction de déplacement correspondant à l'angle de la caméra. Il s'agit d'une projection du vecteur directeur de la caméra sur le plan horizontal.

Si on considère un repère $\vec{\imath}$, $\vec{\jmath}$, $\vec{k}$ orthonormal de l'espace, on a donc :

$$\overrightarrow{V_{déplacement}} = \begin{pmatrix} \overrightarrow{V_{cam}} \cdot \vec{\imath} \\ \overrightarrow{V_{cam}} \cdot \vec{\jmath} \\ 0 \end{pmatrix} \tag{3.1}$$

Une fois ce vecteur rendu unitaire, on obtient la direction du déplacement qui sera utilisée par le programme pour toujours être cohérent avec la caméra.

FIGURE 3.4 – L'oculus Rift Development Kit 2

3.4.3 Compatibilité avec l'Oculus Rift

L'Oculus Rift est un casque de réalité virtuelle conçue par la société Oculus VR.

Nous avons pu travailler avec le *Development Kit 2* (FIGURE 3.4), un modèle dédié aux tests et au développement d'applications pour la réalité virtuelle.

Une fois les micro-logiciels d'Oculus installés sur la machine, et l'appareil branché, il suffit de remplacer la caméra classique de Unity par un modèle fourni par Oculus pour faire fonctionner correctement le casque dans Unity. [5]

Un modèle de personnage adapté à l'Oculus a donc été enregistré pour faciliter la mise en place d'une scène dédiée à la réalité virtuelle dans le programme.

À ce jour, la seule incompatibilité relevée avec l'Oculus réside dans l'impossibilité d'utiliser les interfaces graphiques, celles-ci n'apparaissant pas lors du port du casque.

3.5 Échéancier de construction

La création d'un échéancier de construction était un point majeur du développement de cette application ; être capable d'afficher le bâtiment non terminé, mais à un point donné du chantier peut être intéressant pour de nombreux cas de figures.

Un véritable échéancier de construction est sous la forme d'un diagramme de Gantt, comprenant des tâches complexes, chacune possédant une durée.

Pour simplifier dans un premier temps, il a été fait le choix d'une simple liste de tâches, où chaque tâche correspond à des objets qui vont être construits et d'autres qui vont être détruits.

Afin de calculer les objets visibles à un moment donné du programme, il est d'abord calculé les objets visibles en début de projet, puis chaque tâche est réalisée l'une après

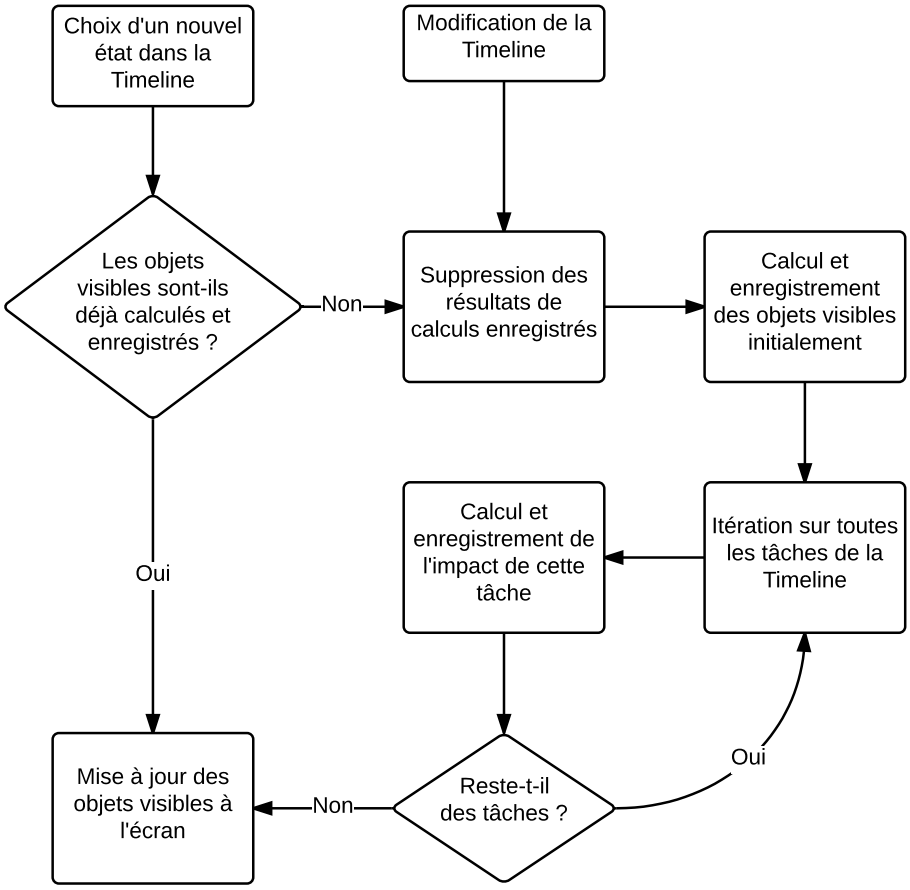


FIGURE 3.5 – Calcul des objets visibles à chaque état de la timeline

l'autre, en enregistrant à chaque fois en mémoire les objets présents et les objets absents pour chaque tâche.

Comme montré en FIGURE 3.5, ce calcul n'est effectué que quand cela est nécessaire, et non pas à chaque changement "d'instant", c'est à dire que le calcul ne se fait que lorsque l'échéancier est modifié, et non pas lorsqu'on souhaite changer quelle est la dernière tâche réalisée.

3.6 Création de sélections et d'échéanciers

3.6.1 Création d'un glisser-déposer

La création d'un mécanisme de glisser-déposer n'est pas toujours simple, cependant Unity offre dans ses API de GUI [4] un mécanisme permettant de le simplifier, celui des Tooltips.

Lorsque l'on créé un bouton, on lui donne un nom, et on peut préciser un tooltip. Le tooltip permet d'obtenir l'information que la souris est en train de survoler le bouton. Ainsi, lorsqu'un glisser déposer est en cours, en récupérant le tooltip actif au moment où l'on relâche le bouton de la souris, on sait sur quel bouton l'objet a été déposé.

Reste à savoir quel objet est en cours de déplacement, et pour cela, il suffit de le mettre en mémoire lorsque le bouton de la souris commence à être enfoncé.

3.6.2 Visualisation de sélections

Pour garder contrôle de ce que l'on fait, il était promordial de rappeler à l'utilisateur quel objet était en cours de sélection, et surtout de lui montrer sur quelle sélection il travaille.

Ainsi, lorsqu'un objet est sélectionné pour un glisser déposer, celui-ci va prendre une couleur verte, symbolisant l'ajout. Et lorsqu'il va survoler un bouton correspondant à une sélection, la sélection en question va prendre une couleur jaune, neutre.

Ainsi, lorsqu'on glisse un objet vers une sélection, on a un rappel de l'objet en cours de déplacement, et de la sélection dans laquelle il va être placé, et on peut donc vérifier la consistance de l'action avant de la valider en relâchant le bouton de la souris.

4. *Graphic User Interface* : interface utilisateur graphique

3.6.3 Format de fichiers pour l'enregistrement

La création de sélections et d'échéanciers implique la mise en place d'un mécanisme d'enregistrement et de chargement des données.

Pour faciliter le processus, le format JSON a été choisi pour assurer une consistance des données, et surtout pour permettre l'enregistrement hiérarchique et récursif de données consistantes.

Ainsi, un objet renverra son identifiant lorsqu'on demandera à l'enregistrer, mais une sélection renverra une liste d'identifiants, récupéré en appelant la méthode d'enregistrement de chaque objet.

Récursivement, pour l'enregistrement d'un ensemble de sélection, on obtiendra une liste de listes d'identifiants. Pour obtenir une tache, on obtiendra deux listes d'identifiants, une pour la construction et une pour la destruction. Et enfin, pour enregistrer un échéancier, on obtiendra une liste de tâches.

3.6.4 Rétro-compatibilité avec les fichiers XML déjà existants

Dans le but d'assurer une rétro-compatibilité avec le format XML largement utilisé dans le département BIM, l'enregistrement et le chargement des fichiers XML a été ajouté en se conformant au format déjà existant.

Ainsi, des sélections créées à l'aide d'autres logiciels, ou extensions de Revit seront compatible avec cette application.

Une autre compatibilité avait été imaginée avec Navisworks. Ce dernier logiciel dispose d'une interface puissante de création de sélections par critères de filtrage sur les données des objets. Ces critères de filtrages sont ensuite enregistrés au format XML pour une utilisation future.

L'importation de ces fichiers serait intéressant dans le programme, mais comme décrit en § 4.2.1, le processus d'ingénierie inverse n'a pas abouti par manque de temps.

Chapitre 4

Résultats obtenus

4.1 L'application

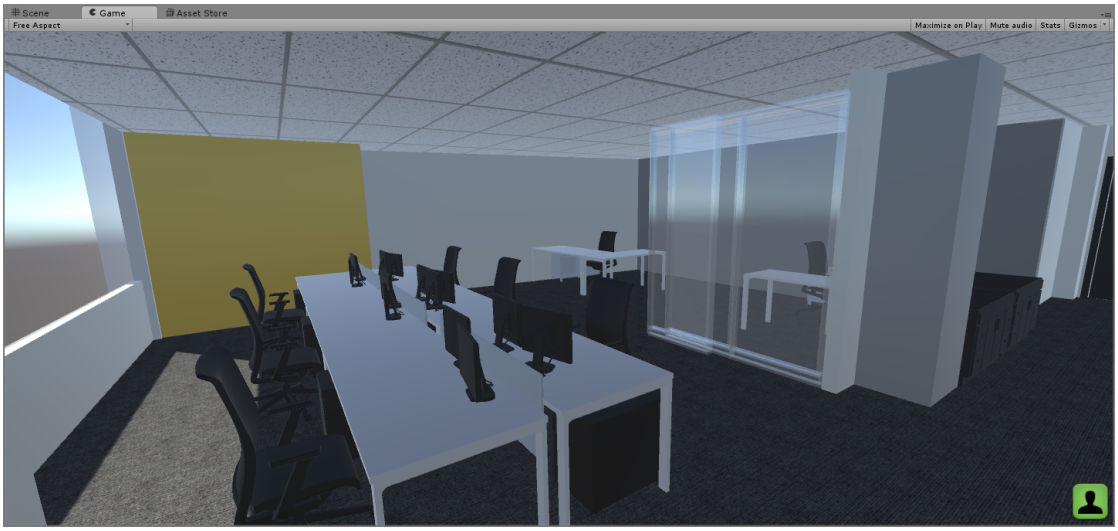


FIGURE 4.1 – Exemple de l'application ouverte sur une modélisation des bureaux de l'équipe BIM chez Pomerleau

L'application, une fois lancée, permet de visualiser le modèle en perspective avec des texture, et ce en temps réel.

Le programme en cours d'exécution dans l'éditeur Unity est visible en FIGURE 4.1, avec une modélisations des bureaux de l'équipe BIM chez Pomerleau.

En bas se situe un menu qui permet d'interagir avec certaines propriétés du joueur, comme l'activation ou non des collisions, et de la gravité.

Il est également possible d'interagir avec les objets, pour consulter les informations qui y sont relatives, mais également pour les placer dans des sélections, ou dans des tâches dans le but de réaliser un échéancier.

De plus, on peut également visualiser un échéancier, permettant alors d'avoir un aperçu du processus de construction, et des différentes étapes du chantier.

Enfin, cette application sous Unity est pleinement compatible avec des solutions de casques de réalité virtuelle, comme l'Oculus Rift, permettant alors une totale immersion dans un projet, et offrant alors la possibilité d'une expérience de visite immersive pour des investisseurs avant même que le projet ne soit validé.

4.2 Limites actuelles de l'application

4.2.1 Utilisation de filtres créés avec Navisworks

L'idée de récupérer une sélection d'objets depuis un ensemble de filtres créés par Naviswork était une des idées intéressantes dans le processus d'intégration des outils existants avec cette nouvelle application.

Cependant, le nombre d'options disponibles dans la création de filtres avec Navisworks rendent complexe la structure des fichiers qui enregistrent ces paramètres, et par conséquent la lecture et l'interprétation de ceux là par le programme.

Par manque de temps a la fin du stage, cette fonction a été délaissée, pour être ajoutée prochainement lors d'une autre phase de développement.

4.2.2 Enregistrement et chargement de fichiers

L'enregistrement et le chargement des fichiers implique la possibilité de pouvoir choisir l'emplacement du fichier à ouvrir, ou à enregistrer.

Cette fonction est disponible prête à l'emploi avec Unity, mais ne fonctionne que lorsque l'application est lancée depuis l'éditeur de Unity (et non pas lorsque l'application est compilée pour fonctionner seule.)

Pour contourner ce problème de façon provisoire, les fichiers s'enregistrent par défaut, lorsque le choix du chemin n'est pas possible, avec un nom horodaté. Il est donc toujours possible d'enregistrer son travail quelque soit la plateforme.

En revanche, à part pour le mécanisme de Timeline où l'on peut spécifier le chemin du fichier par défaut avant la compilation, il n'y a pas d'alternative pour charger des fichiers dans le programme pour le moment.

4.2.3 Dépendance au clavier et à la souris

Bien que certaines interfaces ne sont pas cantonnées à la souris et fonctionnent avec un écran tactile, le programme dans son état actuel n'est pas conçu pour se passer du clavier et de la souris.

La compatibilité de l'application avec d'autres périphériques d'entrées comme une manette de jeu (particulièrement utile pour les déplacements avec l'Oculus Rift) ou le "tout tactile" seraient intéressants

De plus, il y a probablement à gagner en donnant la possibilité d'attribuer plusieurs touches à une même action, contre une seule actuellement, afin d'assurer une certaine flexibilité du programme.

4.2.4 Limites de l'échéancier et améliorations possibles

Actuellement, l'échéancier est modélisé par une liste de tâches, ce qui est une modélisation simplifiée et naïve pour la construction d'une bâtiment complexe. Il n'est pas possible d'avoir des tâches en simultanées, ni même de leur donner une durée.

Dans un processus d'amélioration du programme, il s'agirait d'être capable de gérer les tâches non pas par liste obligeant une tâche après l'autre, mais avec une prise en compte du début d'une tâche et de sa fin (ou de sa durée).

Cela permettrait une gestion des évènements de type GANTT, plus réaliste pour une gestion de projet, mais qui demandera un certain travail de refonte du système d'échéancier, et de son mode de calcul.

4.3 Perspective de développements futurs

Dans une stratégie d'améliorer le travail commencé au cours de ce stage, il s'agit dans un premier temps de repousser les limites identifiées du programme, à savoir :

- Interprétation des fichiers de filtrage de Navisworks
- Résolution du problème de chargement/enregistrement des fichiers sur toutes les plate-formes
- Reprendre le modèle de l'échéancier pour le rendre compatible avec le modèle de GANTT
- Intégrer la possibilité de controller le programme via des gestes tactiles, et/ou à l'aide une manette de jeu.

On peut aussi suggérer le développement de nouveaux modules en fonction des besoins du BIM, et également étendre les recherches de nouvelles façons d'interagir avec les données grâce de nouvelles technologies proposant des interfaces d'entrées originales, tels que le *Leap Motion* ou *HP Sprout*, en conservant ce programme sous Unity comme une base à ces nouveaux champs d'étude.

Conclusion

En définitive, ce stage a été pour moi l'occasion de découvrir et de devenir pendant 6 mois un acteur du développement du BIM, une technologie aujourd'hui en essor dans le monde du génie civil.

En plus de m'apporter une expérience dans l'ingénierie du logiciel avec le moteur Unity, j'ai également pu m'intégrer l'équipe BIM, une équipe dynamique de l'entreprise Pomerleau, qui a su me guider dans la découverte des logiciels associés à BIM, et dans les besoins du développement.

Ma mobilité au Québec a été par ailleurs une expérience particulièrement formatrice, dans le domaine professionnel, comme dans le domaine personnel. Elle a été l'occasion de découverte d'une culture différente, dans des paysages d'une très grande diversité, mais m'a aussi permis de découvrir l'esprit au sein d'une équipe de travail québécoise.

C'était également une occasion de réunir à la fois les domaines de l'informatique, de la réalité virtuelle et du génie civil, ce qui en plus d'être instructif et intéressant, m'a permis de concilier des domaines que je n'aurais pas instinctivement rapproché, tout en me donnant une certaine liberté sur de nombreux choix techniques et visuels du produit final.

Il y a aussi une très grande satisfaction à avoir travaillé sur les fondations d'une application dont le développement sera continué, et qui sera prochainement utilisé à diverses occasions par l'une des plus grandes entreprises de constructions du Québec. Il s'agit une forme de reconnaissance du travail de recherche et de conception que j'ai effectué ces 6 derniers mois qui est grandement appréciée.

Je garderais un excellent souvenir du Québec, tant par mon stage, que par la vie de tout les jours à Montréal, et je n'exclus pas totalement un retour au Québec, si des opportunités professionnelles se présentent plus tard au cours de ma carrière.

Bibliographie

[1] About unity3d. http://unity3d.com/unity.

[2] Autodesk material converter. http://www.macrocad.nl/visualisatie-oplossingen/visualisatie-producten/autodesk-material-converter.

[3] Chaire entre le gridd et pomerleau. http://gridd.etsmtl.ca/fr/chaire-industrielle-pomerleau.

[4] Groupe de recherche gridd. http://gridd.etsmtl.ca/fr/.

[5] Oculus Rift documentation for unity. https://developer.oculus.com/documentation/game-engines/latest/concepts/unity-gsg/.

[6] Page officielle d'autodesk revit. http://www.autodesk.fr/products/revit-family/overview.

[7] Pomerleau. http://www.pomerleau.ca/construction-entrepreneur/index.aspx.

[8] Revit to unity workflow. https://unity3d.com/learn/resources/architectural-visualization-revit-unity-3d-rift.

[9] Unity3D documentation. http://docs.unity3d.com/Manual/index.html.

[10] What is unreal engine ? https://www.unrealengine.com/what-is-unreal-engine-4.

Annexe A

Modélisation des données du bâtiment (BIM)

BIM ou Building Information Modeling ; pour Modélisation des Données du Bâtiment ; est une méthode de conception et de suivi du bâtiment sur la totalité de son cycle de vie, reposant sur l'outil informatique.

Il se constitue comme un processus de production, d'intégration et de gestion des données d'un bâtiment, et se présente sous la forme d'un modèle du bâtiment associé à toute l'information technique nécessaire à son cycle de vie (construction, entretien, modifications, destruction).

De plus en plus utilisé dans le domaine de l'ingénierie du bâtiment, BIM porte sur cette industrie un enjeu majeur pour la réduction des coûts du bâtiment, pour sa construction comme pour son exploitation, tout en apportant de nouvelles dimensions à la conception de celle-ci, comme la dimension écologique.

Enfin, en associant au BIM des mécanismes de coordination et d'estimation, on peut éviter des problèmes liés à la coordination et à la coopération des prestataires cohabitant sur un chantier, assurer qu'il n'y ait pas d'incompatibilité entre les différentes œvres du chantier, et si il en a, les corriger avant que les problèmes ne se présentent tout en maîtrisant le coût et le rendement.

Dans l'industrie, BIM est de plus en plus présent sur les projets du bâtiment, et est devenu un enjeu de développement majeur pour toutes les grandes entreprises de la construction.

Annexe B

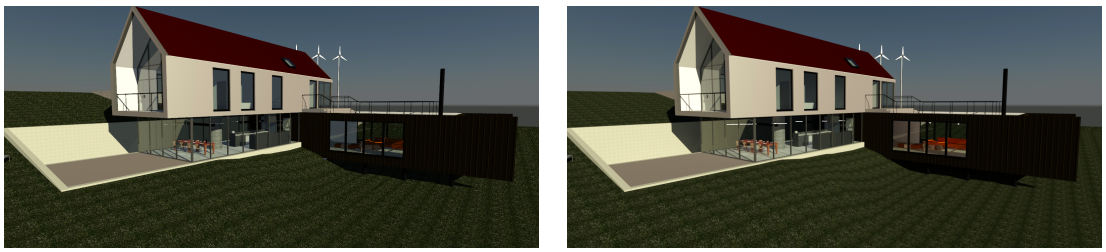
Limites de la génération d'images avec Navisworks

Le logiciel Naviswork permet, pour un temps de calcul variable, la génération d'images photo-réalistes du projet. Après avoir placé la caméra, on peut lancer le calcul à différents niveaux de qualité pour obtenir des images que l'on peut alors fournir au client.

Qualité	Temps de calcul
Faible	42s
Moyen	46min 47s

TABLE B.1 – Temps de calcul pour différents paramètres de qualité

Comme vu en TABLE B.1, on trouve un temps de calcul de trois quarts d'heure pour le paramètre de qualité "moyen", ce qui explique pourquoi je n'ai pas essayé des paramètres plus grands. La qualité de ce calcul en FIGURE B.1, est relativement mitigée, même si le projet n'est qu'un projet d'exemple.



(a) Qualité "faible" (b) Qualité "moyenne"

FIGURE B.1 – Résultats des images calculées par Navisworks

Annexe C

Unity3D ou Unreal Engine

Engine	**Unity3D**	**Unreal Engine**
Latest version	5.0.2	4.7.6
Revit compability		
Objects Identifiers	Accessible	Unaccessible (merging parts)
Revit Materials	Uncompatible (conversion needed) or DataBase use to assign materials	unknown
Scripting		
Language	C#	C++
Flexibility		
Web	Unity Web Player	Unreal Web Player
Computers	PC/Mac/Linux	idem
Mobile devices	iOS/Android/Windows Phone	idem
Game devices	Xbox/PS/Wii	idem
	No internet connexion to work	Internet connexion needed to work
Costs		
Price for pro version	$1500 or $75/month	Free (5% royalties on realeased app)
Price for no pro version	Free	n/a

TABLE C.1 – Comparatif entre Unity3D et Unreal Engine [1] [10]

Annexe D

Document de conception

D.1 Contexte

BIM, ou *Building Information Modeling* est une méthode de conception de bâtiments couvrant la totalité du cycle de vie du projet. Cependant, avec la couverture très vaste de cette technique de modélisation, les logiciels permettant l'accès et la modifications des données gagnent vite en complexité d'utilisation. C'est le cas de Revit, acheté et maintenu par la société Autodesk, qui permet de concevoir un projet avec BIM.

Dans cette perspective, la complexité d'accès au données requiert une certaine réflexion pour améliorer et proposer des solutions alternatives et plus simples d'accès aux informations proposées par BIM.

C'est pourquoi le développement d'une application indépendante, capable d'accéder aux données partagées d'un projet BIM est une opportunité pour Pomerleau.

D.2 Utilisateurs

On peut identifier un certain nombre d'utilisateurs différents pour cette application, amenant une certaine diversité dans les tâches effectuées par chacun.

Client Les clients sont les commanditaires de la conception, construction et maintenance du bâtiment. Ils ne sont généralement intéressés que par une visualisation du rendu final du projet, des étapes de construction, et d'arguments autour du coût et de l'impact écologique de celui-ci.

Prestataire Le prestataire est un utilisateur qui travaille directement sur le projet, que ce soit durant la construction ou la maintenance de celui-ci. Son principal rôle est un rôle de consultation, voire de modification des données de coordination.

Ingénieur L'ingénieur est un utilisateur qui fait partie de l'équipe de conception du bâtiment. Il a tant un rôle de consultation que de modification des données.

D.3 Tâches

Les différentes tâches sélectionnées pour les différents utilisateurs sont décrit en Figure D.1 sous la forme d'un diagramme de cas d'utilisation.

On y trouve :
- Visualisation du modèle en 3 dimensions
- Ouverture de menus contenant les informations des objets
- Modification des données qui sont en droit d'écriture
- Création de sélections manuelles
- Création de sélections depuis un critère
- Création d'un échéancier pour ordonner la création/destruction des objets
- Visualiser la construction créée ci-dessus.

D.4 Qualité ergonomique

La consultation de données dans un environnement en 3 dimensions n'est pas quelque chose d'acquis. C'est pourquoi il est primordial de conserver une certaine ergonomie dans l'affichage des données.

Dans ce sens, quelques choix ont étés faits pour préserver la qualité d'utilisation du logiciel :
- Les données associées à un objet s'affichent dans un menu qui semble lié à cet objet, pour garder l'idée de parenté de ces données.
- Pour simplifier la consultation de ces menus, il est possible de les étendre en plein écran, permettant alors de profiter de la taille de l'écran pour une consultation prolongée
- Certaines actions sont confirmées par une notification non-interruptive, tout comme certains avertissements qui ne nécessitent pas une attention particulière apparaissent de la même façon.
- Une option permettant de fermer tous les menus ouverts précédemment est proposée. De la même façon, il est possible de limiter le nombre de fenêtres pouvant être ouvertes simultanément lors de la configuration de l'application.
- Pour faciliter la selection des objets, un object sélectionné prendra une couleur distinctive, permettant de le dissocier la sélection des autres objets

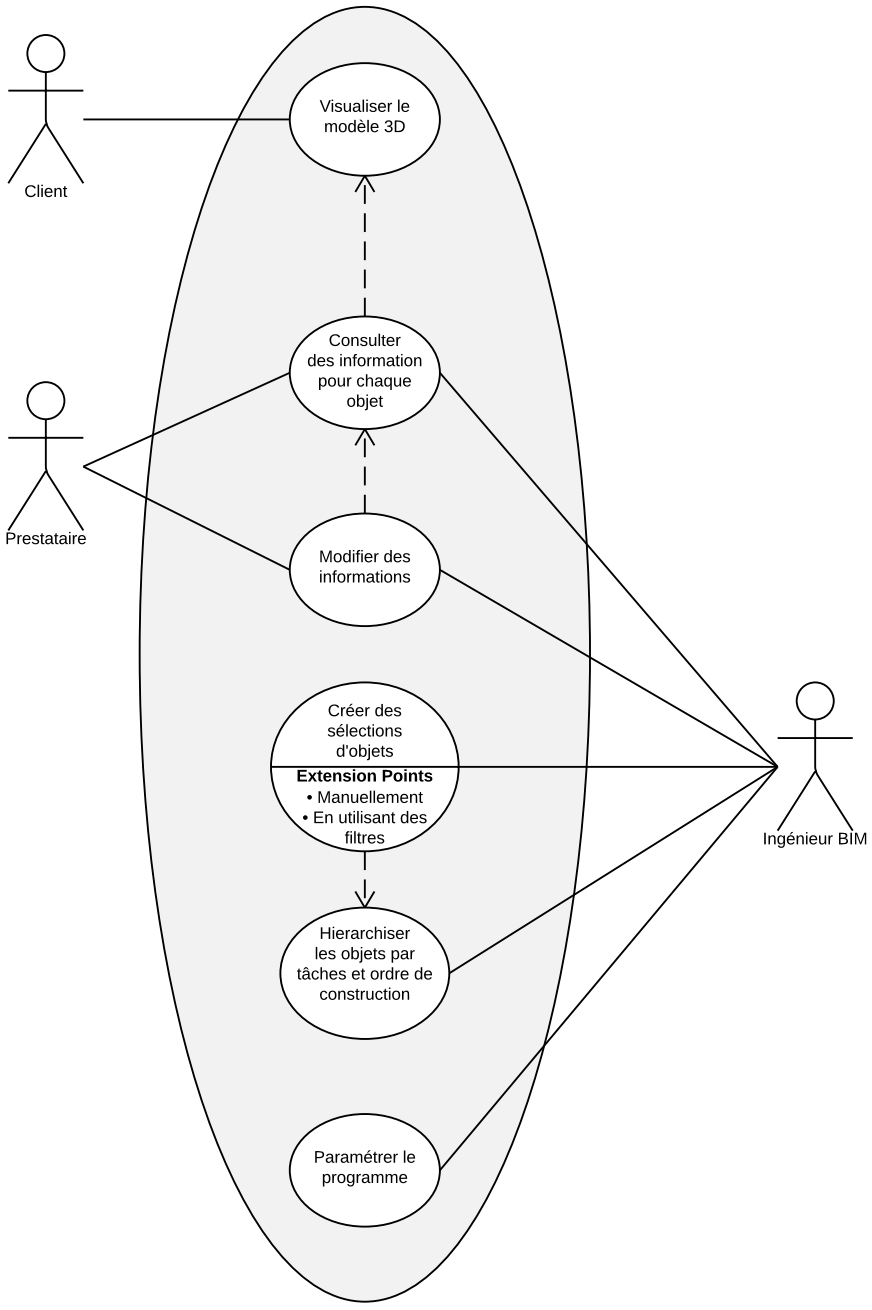


FIGURE D.1 – Diagramme de cas d'utilisation

D.5 Spécifications

D.5.1 Déplacements

Le déplacement choisi dans un premier temps sera un déplacement de type "First Person Shooter", c'est à dire une vue à la première personne, sujet à la gravité, et autorisant le déplacement dans les 4 directions de base, plus les 2 directions verticales.

La souris servira à déplacer la caméra, mais comme il est nécessaire d'effectuer des mouvements de souris pour sélectionner des objets, le choix de maintenir le bouton droit de la souris pour effectuer des mouvements de caméra a été fait.

D.5.2 Accès au données

L'accès au données se fera en cliquant sur un objet. Un menu s'ouvrira alors en superposition, permettant l'accès aux données liés à l'objet. Ce menu sera "attaché" à cet objet, et bougera donc de la même façon que celui-ci lors des mouvements de caméra.

D.5.3 Sélection d'objets

La sélection d'objet se fera en effectuant des "glisser-déposer" des objets vers un menu s'ouvrant pour cette occasion.

En déposant l'objet vers la zone dédiée, il sera ajouté à la sélection. Le cas échéant, une nouvelle sélection sera crée.

Il sera possible d'enregistrer et de charger des sélections séparément, et en groupe.

D.5.4 Exploitation de filtres pour créer des sélections

La création de filtres se cantonnera pour l'instant au logiciel Navisworks. Cependant, ces filtres peuvent être exportés au format XML, et devront pouvoir être importés dans le logiciel pour créer des sélections.

D.5.5 Visualisation de la construction

Visualiser la construction revient à pouvoir choisir un état de construction, dans une barre horizontale de sélection. Deux boutons permettrons d'atteindre simplement l'état suivant et l'état précédent de la construction, pendant que la barre horizontale permet de changer très rapidement l'état voulu.

D.5.6 Création d'un ordre de construction

La création d'un ordre de construction ressemblera a beaucoup de points à la création de sélections décrites en § D.5.3.

La principale différence est qu'il y aura un menu horizontal listant toutes les tâches, et chaque tâche disposera d'une sélection d'objets qui seront construits, et d'une autre sélection des objets qui seront détruits.

En glissant-déposant les objets vers les selections correspondantes, on peut construire un ordre de construction pour ce bâtiment.

D.6 Maquette

FIGURE D.2 – Menu permettant de gérer le comportement de l'utilisateur

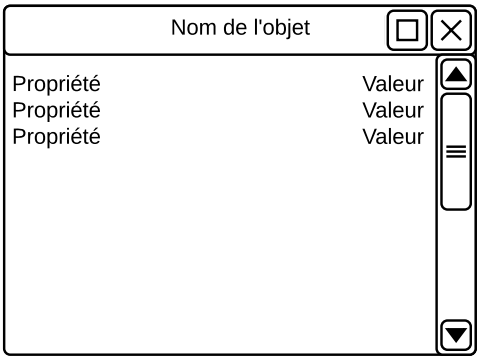


FIGURE D.3 – Menu d'informations

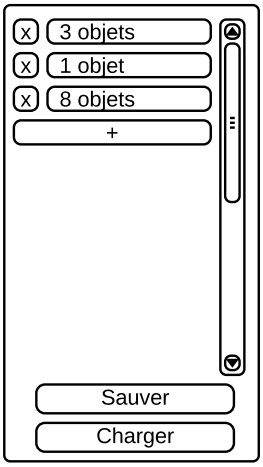


FIGURE D.4 – Création de sélections

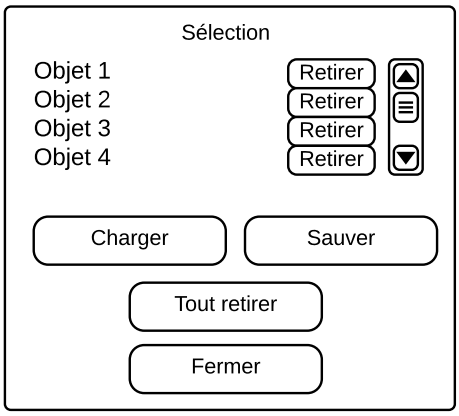


FIGURE D.5 – Gestion d'une sélection

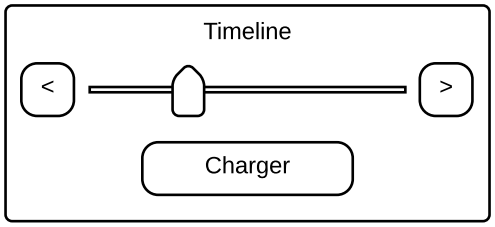


FIGURE D.6 – Choix de l'état de construction

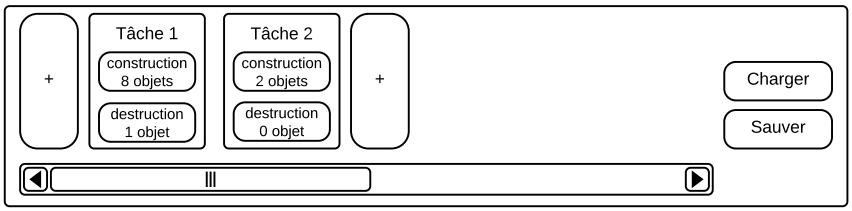


FIGURE D.7 – Création de l'ordre de construction