Modélisation Paramétrique en Réalité Virtuelle

Parametric modelling in Virtual Reality

Adrien Coppens[1,*], *Tom* Mens[1] et *Mohamed-Anis* Gallas[2]

[1]Service de Génie Logiciel, Université de Mons, Belgique
[2]Service Conception Architecturale, Université de Mons, Belgique

Résumé. Les technologies immersives ont fait leur apparition dans bon nombre d'outils de modélisation architecturale. Néanmoins, leur usage se limite bien souvent à des fins de visualisation, par exemple pour valider un design auprès d'un client muni d'un casque de réalité virtuelle. Notre travail vise à permettre une utilisation de ce medium immersif durant l'activité de conception architecturale elle-même. Nous présentons dès lors un outil de modélisation paramétrique en réalité virtuelle permettant de combiner, en immersion, l'édition de modèles Grasshopper et la visualisation des géométries générées. Nous validerons notre approche auprès d'architectes et d'étudiants formés à ce paradigme de conception.

Mots-clés. Modélisation paramétrique, Réalité Virtuelle, Immersion, Interaction Homme-Machine, Graphe, Visualisation.

Abstract. Immersive technologies have made their way into numerous architectural design tools. Nevertheless, their use is often limited to visualisation purposes, e.g. in order to validate a design with a customer wearing a Virtual Reality headset. Our work aims at enabling the use of that immersive medium as part of the architectural modelling process itself. We therefore describe a Virtual Reality-based parametric modelling tool, combining immersive editing and visualisation capabilities for Grasshopper models. We will validate our approach with architects and students familiar with that architectural design paradigm.

Keywords. Parametric modelling, Virtual Reality, Immersion, Human-Computer Interaction, Graph, Visualisation

1. Introduction

L'arsenal technologique à disposition des architectes et plus généralement du secteur AEC (Architecture, Engineering, et Construction) n'a cessé de grandir au cours de l'histoire de la discipline. Les architectes d'aujourd'hui continuent bien évidemment à réaliser des

* Corresponding author: adrien.coppens@umons.ac.be

esquisses et des maquettes, mais ils font également appel à de nombreux outils numériques, notamment aux logiciels de Conception Assistée par Ordinateur (CAO), couvrant les différentes phases de développement du projet d'architecture.

Ces logiciels, qui ont désormais pris une place prépondérante, ont eux-mêmes évolué, notamment via l'apparition de modules immersifs, en Réalité Augmentée (abrégé « AR » en anglais) et/ou en Réalité Virtuelle (VR). Ces technologies permettent à l'utilisateur de travailler avec une représentation virtuelle d'un modèle 3D dans un contexte réel (cas de l'AR) ou virtuel (VR). Cependant, cette intégration est bien souvent limitée à des fins de visualisation via un outil d'export de modèle en une expérience AR/VR.

2. Contexte de la recherche

2.1 Immersion et architecture

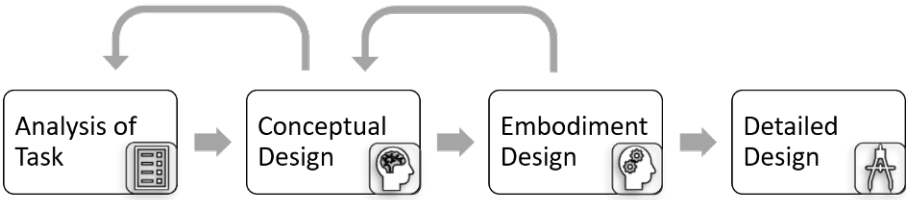


Figure 1. Processus de design créatif, selon (Howard et al., 2008)

Si on s'en réfère à (Howard et al., 2008), le processus de design est généralement composé de quatre étapes : (1) analyse de la tâche, (2) design conceptuel, (3) design concret, et (4) design détaillé. Comme le montre la Figure 1, le concepteur analyse d'abord la tâche à réaliser, avant de travailler sur une ou souvent plusieurs ébauches de design. Sur base de l'idée retenue, le concepteur concrétise le design en le structurant, avant de finaliser les détails pour la réalisation physique du modèle.

Comme précisé dans l'introduction, la plupart des intégrations de composantes AR/VR dans les outils de conception architecturale se limitent à des fins de visualisation. Par exemple, des outils comme IrisVR Prospect (irisvr.com/prospect) et Twinmotion (unrealengine.com/twinmotion) permettent en un simple clic d'intégrer une géométrie au sein d'une expérience VR. Néanmoins, ces outils ne permettent que des interactions très limitées avec le modèle, comme un changement de texture ou de position du soleil virtuel.

Plusieurs travaux ont tout de même permis d'amener des capacités d'annotation et d'esquisse dans un contexte VR ; certains de ces travaux visant clairement les pratiques architecturales, comme Hyve-3D (Dorta et al., 2016) qui permet de tracer des traits via un curseur 3D contrôlé par une tablette. Ces travaux amènent donc de l'interaction en immersion, typiquement durant la phase conceptuelle d'un projet.

Depuis peu, le plugin MARUI (marui-plugin.com) pour le logiciel Autodesk Maya (autodesk.com/maya) permet même aux utilisateurs de modéliser des objets en trois dimensions depuis une application VR. Mindesk (mindeskvr.com) fonctionne de la même manière pour Rhinoceros (rhino3d.com) et une intégration plus poussée avec le plugin Grasshopper (grasshopper3d.com) semble même prévue.

2.2 Immersion et modélisation paramétrique

Les outils présentés dans la section précédente représentent des avancées importantes dans l'intégration des technologies immersives pour les activités de conception

architecturale. Bien que nos travaux visent un objectif similaire, nous avons choisi de nous focaliser sur la modélisation paramétrique : un paradigme de conception qui accuse encore davantage de retard dans l'usage interactif des technologies immersives.

La modélisation paramétrique, au sens où nous l'entendons ici, permet à un concepteur de créer une géométrie à partir d'un algorithme, composé d'opérations géométriques. Typiquement, cet algorithme est représenté dans un langage de programmation visuelle, basé sur un graphe dirigé représentant le flux de contrôle ou le flux de données. Dans un tel graphe, les arcs (les liens) permettent de véhiculer des paramètres (comme des nombres ou des booléens) ou des géométries. Les nœuds (les composants) utilisent ces données pour produire d'autres géométries qui deviennent ainsi disponibles pour d'autres nœuds. Des modèles complets (sculptures, bâtiments, etc.) peuvent ainsi être générés ; les modèles finaux étant généralement en « sortie » du graphe, dans ce que l'on appelle un « puit ». La Figure 2 montre un graphe permettant de générer un cube, défini par deux sommets opposés : P_1 à l'origine et P_2 en (4,4,4), puisque le même paramètre de longueur de côté (« Side length ») est utilisé pour les nœuds « Addition ».

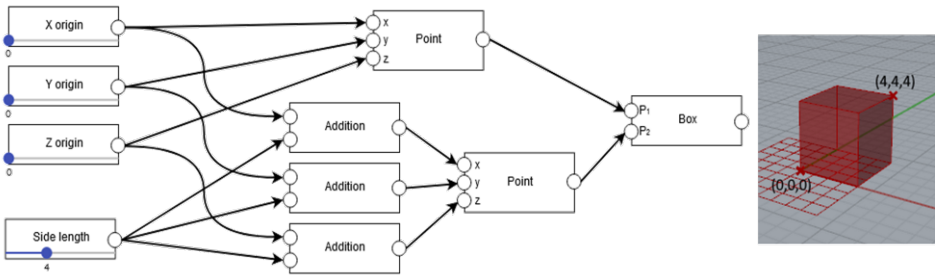


Figure 2. Graphe (à gauche de l'image) pour un modèle paramétrique simple, produisant un cube (à droite de l'image) en sortie du composant « Box ».

Les logiciels les plus populaires en architecture qui supportent ce paradigme sont Dynamo Studio (autodesk.com/products/dynamo-studio), GenerativeComponents (bentley.com/products/product-line/modeling-and-visualization-software/generativecomponents) et Grasshopper, le plugin pour Rhinoceros mentionné précédemment. D'autres outils de CAO dont l'utilisation première s'éloigne de l'architecture peuvent être utilisés dans ce contexte, comme Maya via PyFlow* (github.com/wonderworks-software/PyFlow), ou encore Blender (blender.org) via Sverchok* (github.com/nortikin/sverchok) ou Sorcar* (github.com/aachman98/Sorcar), mais aussi Houdini (sidefx.com).

Puisque ces outils sont intégrés dans des logiciels de CAO « standards », que ce soit sous forme de composante paramétrique (plugin) ou au sein d'une suite logicielle, il est tout à fait possible de faire appel aux outils de visualisation présentés en section 2.1.

Néanmoins, à notre connaissance, seul Grasshopper a fait l'objet de développements permettant une interaction avec un modèle paramétrique depuis une application VR (Coppens et al., 2018). Cette interaction étant limitée à l'ajustement des valeurs de certains paramètres, nous souhaitons aller plus loin en permettant l'édition des algorithmes visuels eux-mêmes, depuis l'environnement immersif. Le Tableau 1 résume les possibilités de visualisation et d'interaction pour les logiciels précités. On précisera que celui-ci indique entre parenthèses une manière de réaliser la fonctionnalité concernée, éventuellement via l'interconnexion de plusieurs logiciels. Concernant l'export en VR, la mention « .fbx » signifie qu'il n'existe pas d'intégration « en 1 clic », mais que le logiciel concerné gère

* Plugin non-officiel

l'export au format .fbx,. Il est donc ensuite possible d'importer le modèle dans un logiciel capable de créer une expérience VR, comme Twinmotion, que nous avons déjà mentionné en section 2.1.

Tableau 1. Fonctionnalités visuelles et d'interaction dans les outils de modélisation paramétrique supportant la programmation visuelle basée sur les graphes

Fonctionnalité / Logiciel	Visuel 3D (perspective)	Export en VR possible	Export VR en 1 clic	Lien VR (export auto + ajustement de paramètres)	Interaction avec le graphe en VR
Grasshopper	✓ (Rhino)	✓ (Rhino†)	✓ (Rhino†)	✓ (‡)	
Dynamo Studio	✓	✓ (Revit†)	✓ (Revit†)		
GenerativeComponents	✓	✓ (.fbx†)			
Houdini	✓	✓ (.fbx†)			
Sverchok/Sorcar	✓ (Blender)	✓ (.fbx†)			
PyFlowMaya	✓ (Maya)	✓ (.fbx†)			

† via Twinmotion et/ou IrisVR ; ‡ via le plugin proposé par (Coppens et al., 2018)

2.3 Programmation en 3D

De nombreux langages de programmation en 3D ont été créés ; la plupart étant basés sur des diagrammes de flux de données. CUBE/CUBE-II (Najork, 1996; Najork & Kaplan, 1991) et Lingua Graphica (Stiles & Pontecorvo, 1992) avaient par exemple déjà fait leur apparition dans les années 1990. Bien que les deux exemples précités aient été conçus avec les environnements virtuels en vue, ils n'ont jamais été adaptés dans un contexte immersif (par exemple en VR).

Par contre, avec une approche similaire mais en intégrant le tout dans une application VR, (Steed & Slater, 1996) ont créé un système qui permet de définir le comportement d'objets à partir de diagrammes de flux de données. Ce système peut par exemple être utilisé pour créer des animations ou développer des applications interactives, telles que des jeux.

La réalité augmentée n'est pas en reste, avec notamment les travaux de (Lee et al., 2004) qui rendent possible la définition du comportement des objets d'une scène 3D via une interface tangible en AR.

Dans les cas mentionnés ci-dessus, la capacité à créer ou éditer du contenu 3D tout en visualisant celui-ci de manière immersive a été mise en avant comme étant clairement bénéfique. Puisque les modèles architecturaux sont également en trois dimensions, nous pensons que permettre l'édition de ces modèles depuis un environnement immersif serait tout aussi bénéfique. En plus du contexte 3D qui est complètement différent de l'interface 2D habituelle (un écran d'ordinateur), on bénéficie typiquement d'interactions plus « libres » avec ce type de technologies qu'avec la traditionnelle combinaison clavier-souris. Il est par exemple courant de bénéficier d'une manette dite *6-DoF* pour six degrés de liberté (« Degrees of Freedom »), c'est-à-dire une manette complètement traquée en 3D, à la fois en position et en rotation. Il reste donc à déterminer comment interagir avec le contenu souhaité dans ce contexte.

2.4 Techniques d'interaction pour les environnements 3D

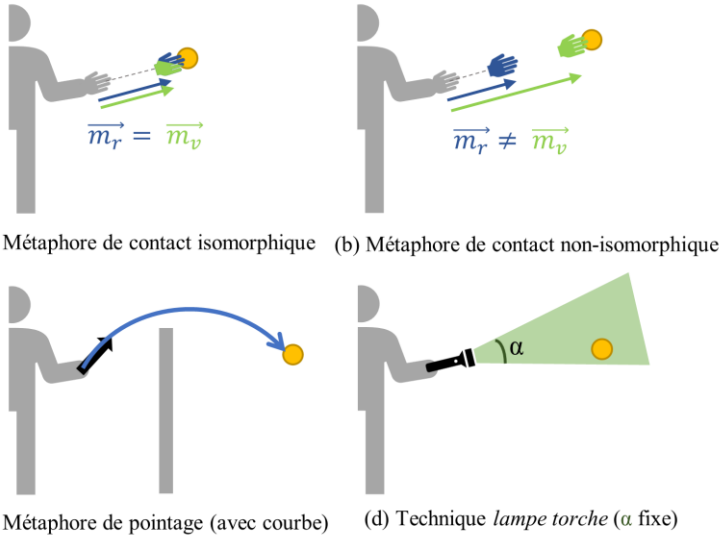
Interagir efficacement, depuis un environnement immersif en trois dimensions, nécessite des techniques différentes de celles utilisées pour de classiques applications PC.

Dans la littérature (LaViola, 2017), on considère généralement que la manipulation est constituée de 4 tâches : sélection, positionnement, rotation, et mise à l'échelle. Dans le cas présent, nous nous concentrerons sur la sélection et le positionnement, qui sont les besoins primaires dans notre contexte de manipulation de graphes.

2.4.1 Manipulation directe

Une manière de catégoriser les techniques d'interactions immersives est basée sur l'isomorphisme. Les approches isomorphiques préservent une correspondance naturelle entre les actions de l'utilisateur et les effets résultants, alors que les techniques non-isomorphiques permettent des interactions moins réalistes, potentiellement basées sur des outils virtuels ayant des effets « surnaturels ». Généralement, ces techniques font référence à une métaphore de *contact* ou de *pointage*.

Les techniques basées sur le contact nécessitent que l'utilisateur atteigne l'objet avec lequel il souhaite interagir. Par exemple, on peut faire correspondre la position et l'orientation d'une manette 6-DoF avec un curseur virtuel, de sorte qu'un objet avec lequel le curseur entre en contact puisse être contrôlé, via l'appui sur un bouton. Dans ce cas, on a une correspondance « 1 pour 1 » entre les positions et rotations, l'approche est donc dite isomorphique (Figure 3a, où l'on voit que mouvement virtuel $\overrightarrow{m_v}$ correspond au mouvement réel $\overrightarrow{m_r}$). Dans le cas où la zone de tracking de la manette est limitée par rapport à l'espace virtuel disponible pour l'interaction, on fait typiquement appel à des correspondances non-isomorphiques. Les techniques *Go-Go* (Poupyrev et al., 1996) et *PRISM* (Frees & Kessler, 2005) sont deux exemples typiques qui font usage d'une correspondance non-linéaire entre la manette traquée et le curseur virtuel. La Figure 3b montre un exemple d'une technique de ce type, où $\overrightarrow{m_v}$ et $\overrightarrow{m_r}$ sont bien différents.



(a) Métaphore de contact isomorphique (b) Métaphore de contact non-isomorphique

(c) Métaphore de pointage (avec courbe) (d) Technique *lampe torche* (α fixe)

Figure 3. Exemples de techniques d'interaction directe

De son côté, la métaphore de pointage permet naturellement de limiter le problème de l'espace de tracking restreint, puisqu'il suffit de pointer vers un objet pour interagir avec celui-ci.

Bien souvent, l'orientation de la manette définit une sorte de « rayon laser », dont l'intersection avec l'environnement est utilisée pour déterminer la position ou l'objet cible. De nombreuses techniques basées sur cette métaphore divergent simplement par la manière dont l'orientation de la manette définit le rayon laser. La version la plus basique se contente de définir une demi-droite partant de la manette, dans la direction vers laquelle celle-ci pointe. Certaines techniques plus complexes permettent d'incurver le rayon, par exemple via des courbes de Bézier (Feiner, 2003) pour limiter les problèmes d'occlusion. La Figure 3c illustre l'utilité d'un pointage courbé pour atteindre un objet qui serait resté inaccessible en ayant recours à une simple demi-droite. D'autres définissent des volumes de sélection plutôt que de simples rayons, ce qui peut faciliter la sélection ou permettre les sélections multiples. Ces volumes de sélection peuvent être statiques (par exemple la technique de la *lampe torche* (Liang & Green, 1994) illustrée en Figure 3d) ou dynamiques (par exemple la technique de l'*aperture* (Forsberg et al., 1996)) qui étend la technique précédente en permettant le contrôle de l'angle d'ouverture du cône de sélection (l'angle α de cette Figure 3d n'est donc plus fixe).

2.4.2 Reconnaissance vocale

Un autre type d'interaction peut grandement simplifier la réalisation d'actions spécifiques : la reconnaissance vocale, qui utilise de fait une autre modalité, la voix. Généralement, les applications qui font appel à ce type de technologies utilisent une approche multimodale : elles combinent la reconnaissance vocale avec d'autres modes d'interaction. Par exemple, la métaphore *Put-That-There* (Bolt, 1980) tire avantage d'une technique de pointage pour le positionnement d'un objet créé via une commande vocale.

De nombreux outils de reconnaissance vocale sont disponibles aujourd'hui. Ils se distinguent notamment par la nécessité (ou non) d'une connexion internet active ou par le déclencheur d'écoute (en continu ou à l'appui sur un bouton par exemple). Une autre distinction importante est le type de contenu attendu, ainsi que les restrictions dès lors imposées : les systèmes dits à *dialogue dirigé* (Pieraccini & Huerta, 2005) n'acceptent qu'un ensemble de commandes (ou plus généralement de suites de mots) prédéfinies, alors que des approches de type *texte libre* permettent de reconnaitre un contenu plus large. Les approches à dialogue dirigé sont principalement subdivisées en 2 catégories : celles chargées de reconnaitre des mots-clefs, qui extraient donc des mots spécifiques ; et celles basées sur des grammaires, qui produisent des phrases en fonction de règles définies.

Dans le contexte de la manipulation de modèles paramétriques, des commandes vocales comme « Créer PointXYZ » ou « Ajouter slider avec valeur 7 » permettraient de faciliter la création des entités respectives. Les approches de type « mots-clefs » et « texte libre » pourraient permettre de réaliser cet objectif, grâce à un traitement subséquent de l'information récupérée par le système de reconnaissance. Néanmoins, une approche à base de grammaire semble plus appropriée, puisque l'on s'assurerait ainsi de limiter les commandes reconnues par le système à des combinaisons prévues dans la grammaire. Si celle-ci est bien définie, ces combinaisons correspondront toujours à des actions réalisables.

3. Actions à réaliser et possibilités d'interactions

Puisqu'il est question de modifier un graphe, les actions primaires qu'il faut pouvoir réaliser sont l'ajout et la suppression de nœuds (composants) ou d'arcs (liens). On veut également être capable de déplacer un nœud. Cette section décrit les prototypes (P_1 et P_2)

développés lors du 15e workshop eNTERFACE (Coppens et al., 2019). Le Tableau 2 permet de visualiser les possibilités d'interaction que nous avons pu explorer, en fonction du matériel que nous avions à notre disposition.

On notera que, pour faciliter la tâche du concepteur mais surtout pour permettre le travail sur des graphes de plus grande taille, une deuxième catégorie d'actions « secondaires » est nécessaire. Nous incluons dans cette catégorie la manipulation de la visibilité (niveau de zoom et translation de la partie visible du graphe) ainsi que la gestion des groupements (groupes et clusters). Ces actions n'ont cependant pas encore fait l'objet de développements au moment de la rédaction.

Tableau 2. Actions à réaliser et techniques d'interaction.

Techniques / Actions		Modalité			Type d'interaction			
		Manette 6-DoF	Mains	Parole	Direct	Indirect	Isomorphique	Non-isomorphique
Nœud	Ajout	P_1	P_2	P_1	P_1, P_2	P_1, P_2	P_1, P_2	P_2
	Suppression	P_1	P_2		P_1, P_2		P_1, P_2	P_2
	Déplacement	P_1	P_2		P_1, P_2		P_1, P_2	P_2
Arc	Ajout	P_1	P_2		P_1, P_2		P_1, P_2	P_2
	Suppression	P_1	P_2		P_1, P_2		P_1, P_2	P_2

3.1 Premier prototype : manettes Vive

Notre premier prototype (P_1 dans le Tableau 2) utilise les manettes fournies avec le HTC Vive. Celles-ci sont traquées en trois dimensions, en position et en rotation (6-DoF). Ce prototype fait usage d'une technique d'interaction directe et isomorphique, basée sur une métaphore de saisie : l'utilisateur touche l'élément virtuel avec lequel il souhaite interagir et appuie sur un bouton pour effectuer une action donnée. Par exemple, pour attacher un nœud à la manette, il suffit d'appuyer sur un bouton lorsque la manette est en contact avec ce nœud. Une fois attaché, le nœud se déplace avec la manette et peut être relâché en réappuyant sur le même bouton. L'utilisateur peut également supprimer un nœud attaché en le jetant. Une vidéo de démonstration de ce prototype peut être visionnée (informatique.umons.ac.be/staff/Coppens.Adrien/?video=eNTERFACE2019) et le code source de l'application est disponible en ligne (github.com/qdrien/Grasshopper-VR-graph). La partie gauche de la Figure 4 montre un utilisateur commençant à interagir avec un nœud.

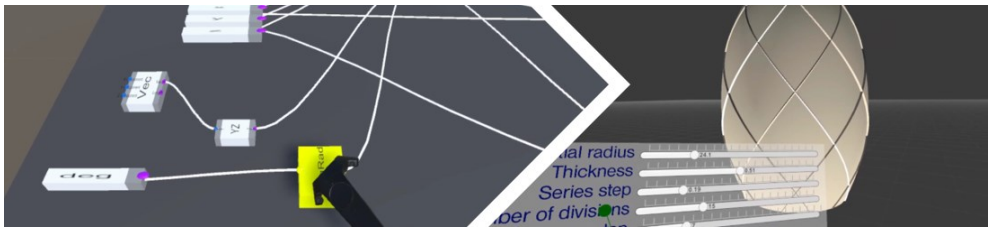


Figure 4. Modélisation paramétrique en VR : modification du graphe, de paramètres, et visualisation 3D

3.2 Second prototype : Leap Motion

Le second prototype (P_2) vise deux objectifs : répliquer le type d'interaction de P_1 (de sorte à comparer le type de modalité) et explorer des techniques non-isomorphiques, basées

sur la métaphore de pointage. Malheureusement, les développements réalisés dans cet objectif n'étaient pas compatibles avec le framework développé pour le workshop. Il n'a donc pas été possible de les intégrer directement pour obtenir un prototype fonctionnel.

3.3 Reconnaissance vocale

Puisque nos prototypes sont réalisés avec le moteur de jeu Unity (unity.com), nous bénéficions de son support intégré pour les outils de reconnaissance vocale de Windows, qui inclut le traitement basé sur les grammaires. Nous avons donc développé une grammaire qui gère la création de nœuds, via une commande vocale « Ajouter composant `<type>` », où `<type>` peut prendre un ensemble de valeurs définies au lancement de l'application, en fonction des composants disponibles dans le système.

Il est parfois nécessaire d'affecter une valeur numérique à un composant (typiquement pour un slider) et nous avons donc également besoin de pouvoir reconnaitre des nombres. Gérer du contenu alphanumérique est non-trivial (Wang & Ju, 2004) mais nous avons pu nous servir d'un ensemble de règles fournies dans le SDK « Microsoft Speech Platform » (microsoft.com/download/details.aspx?id=27226).

Enfin, un utilisateur pouvant souhaiter assigner une valeur textuelle à un nœud (typiquement un panel), nous avons également fait appel à une approche de type « texte libre », qui n'est activée que lorsque certaines règles de la grammaire sont déclenchées (par exemple, lorsque l'utilisateur dit « avec valeur »).

4. Modèle paramétrique générique

Afin de lier nos expérimentations à des exemples concrets, nous avons mis en place un framework qui stocke un modèle paramétrique sous forme de graphe générique, dans le sens où il n'est pas lié à un logiciel de modélisation paramétrique particulier. Puisque nous avons besoin de « ports d'entrée/sortie » et qu'un graphe classique ne comprend pas de telle notion, nous avons choisis de décomposer un « nœud paramétrique » en un nœud principal (le composant lui-même ; par exemple, un « Point XYZ ») et des nœuds secondaires qui lui sont liés : un nœud pour chaque port d'entrée et de sortie (donc dans l'exemple, un nœud par composante et un nœud de sortie). Les liens entre les ports et le nœud principal ne peuvent pas être modifiés et l'utilisateur ne peut créer de liens qu'avec des nœuds de type « port ». La Figure 2 est dès lors quelque peu simplifiée par rapport à notre représentation interne du graphe. Nous travaillons en effet avec une structure plus proche du *Generalized Parametric Model (GPM)* tel que proposé par (Janssen & Stouffs, 2015).

Notre framework est, au moment de la rédaction de cet article, capable de lire des fichiers Grasshopper (extensions .gh et.ghx) pour les convertir dans notre format de graphe générique. Une fois modifié, le graphe peut être reconverti au format Grasshopper et il est donc tout à fait possible de reprendre le travail sur celui-ci depuis le logiciel d'origine.

Dans (Coppens et al., 2018), l'option choisie pour la gestion des changements de valeurs des paramètres est d'utiliser un plugin pour Grasshopper comme relai entre l'environnement immersif et Grasshopper lui-même. Nous avons ici opté pour cette approche à base de conversion en un graphe générique parce que nous souhaitions permettre des modifications sur l'entièreté du modèle et pas seulement des changements de valeur de paramètres, qui sont eux plus aisément réplicables via le SDK Grasshopper (developer.rhino3d.com/api/grasshopper). Un autre avantage de cette approche est qu'elle permet de rendre plus modulaire la solution proposée, puisqu'il devient dès lors possible, via l'implémentation d'un convertisseur ad hoc, d'ajouter un support pour tout autre logiciel de modélisation paramétrique.

5. Visualisation immersive de géométries Grasshopper

Puisque notre solution permet d'obtenir un fichier Grasshopper à partir du graphe générique, potentiellement modifié depuis l'application VR, nous pouvons simplement ouvrir ce fichier avec Rhino/Grasshopper pour en récupérer une visualisation. Rhino Inside (rhino3d.com/inside) nous permet de démarrer et d'interagir avec une instance de Rhino (et donc de Grasshopper) depuis l'application VR. Nous pouvons dès lors synchroniser le fichier exporté via notre framework, sans que l'utilisateur ne doive retirer son casque.

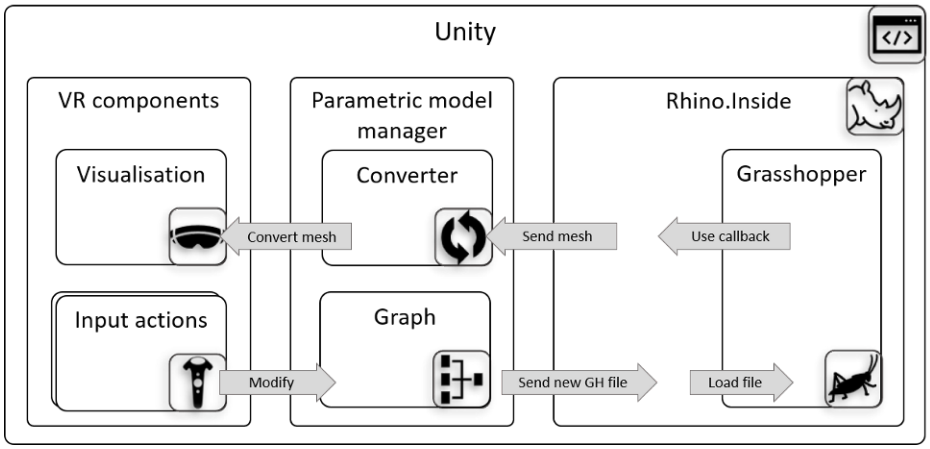


Figure 5. Solution proposée pour la modélisation paramétrique en Réalité Virtuelle (ici via Grasshopper et Rhino.Inside)

Néanmoins, pour que cette visualisation soit disponible depuis l'environnement immersif, il reste nécessaire d'avoir recours à un script d'envoi de géométrie similaire à (Coppens et al., 2018). Ce script utilise les fonctionnalités de « callback » de Rhino Inside pour communiquer le mesh (le rendu de la géométrie) à l'application immersive, qui peut dès lors recréer la géométrie en VR. Nous assimilons également cette étape à une « conversion » puisqu'il s'agit de générer procéduralement un mesh dans Unity en traduisant les données reçues au format de mesh de Rhino/Grasshopper.

La Figure 5 schématise le processus décrit dans les sections 4 et 5 : le fonctionnement de notre solution pour la manipulation d'un modèle paramétrique (ici Grasshopper) en VR.

6. Perspectives et conclusion

Après avoir identifié un manque dans l'intégration des technologies immersives dans les outils de modélisation paramétrique, nous avons ici proposé une solution pour permettre l'édition de tels modèles (via une représentation intermédiaire sou forme de graphe) depuis une application VR. Notre solution permet à un utilisateur de visualiser les changements réalisés sur le graphe, sans avoir à quitter l'environnement immersif. Nous avons exploré différentes modalités et techniques d'interaction pour réaliser les actions identifiées comme primaires et nous avons ouvert la possibilité à davantage d'options. Nous allons notamment adapter notre solution en AR, avec une interaction basée sur des interfaces tangibles.

Notre solution sera évaluée auprès d'architectes et d'étudiants en architecture dans différents établissements en France et en Belgique, afin de valider nos choix d'interactions et la pertinence de la mise à disposition d'un tel outil. Nous allons également récolter des informations sur les ressentis, attentes et appréhensions des architectes concernant la VR et son utilisation en architecture, via un sondage en ligne.

Enfin, nous allons étendre les fonctionnalités présentes dans l'outil, en travaillant sur les actions dîtes secondaires, mais aussi en permettant à un utilisateur de contrôler directement un point (par exemple un ancrage) utilisé dans son graphe, depuis l'application VR.

Nous ne prévoyons pour l'instant pas d'étendre notre solution à d'autres logiciels de modélisation paramétrique mais, comme précisé en section 4, notre framework, open source et disponible en ligne, pourrait tout à fait permettre de tels développements.

Bibliographie

Bolt, R. A. (1980). *"Put-that-there": Voice and gesture at the graphics interface* (Vol. 14).

Coppens, A., Mens, T., & Gallas, M.-A. (2018). Parametric Modelling Within Immersive Environments: Building a Bridge Between Existing Tools and Virtual Reality Headsets. *36th eCAADe Conference*.

Coppens, A., Bicer, B., Yilmaz, N., & Aras, S. (2019). Exploration of Interaction Techniques for Graph-based Modelling in Virtual Reality. *In Proceedings of the 15th Summer Workshop on Multimodal Interfaces (eNTERFACE'19)*

Dorta, T., Kinayoglu, G., & Hoffmann, M. (2016). Hyve-3D and the 3D Cursor: Architectural co-design with freedom in Virtual Reality. *IJAC*, *14*(2), 87–102.

Feiner, A. O. S. (2003). The flexible pointer: An interaction technique for selection in augmented and virtual reality. *ACM UIST*, 81–82.

Forsberg, A., Herndon, K., & Zeleznik, R. (1996). Aperture based selection for immersive virtual environments. *ACM UIST*, 95–96.

Frees, S., & Kessler, G. D. (2005). Precise and rapid interaction through scaled manipulation in immersive virtual environments. *IEEE VR.*, 99–106.

Howard, T. J., Culley, S. J., & Dekoninck, E. (2008). Describing the creative design process by the integration of engineering design and cognitive psychology literature. *Design Studies*, *29*(2), 160–180.

Janssen, P., & Stouffs, R. (2015). Types of parametric modelling. *Proc. Computer-Aided Architectural Design Research in Asia (CAADRIA 2015)*, 157–166.

LaViola, J. J. (2017). *3d user interfaces: Theory and practice* (2nd edition).

Lee, G. A., Nelles, C., Billinghurst, M., & Kim, G. J. (2004). Immersive authoring of tangible augmented reality applications. *Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality*, 172–181.

Liang, J., & Green, M. (1994). JDCAD: A highly interactive 3D modeling system. *Computers & Graphics*, *18*(4), 499–506.

Najork, M. A. (1996). Programming in three dimensions. *Journal of Visual Languages & Computing*, *7*(2), 219–242.

Najork, M. A., & Kaplan, S. M. (1991). The CUBE languages. *Proceedings 1991 IEEE Workshop on Visual Languages*, 218–224.

Pieraccini, R., & Huerta, J. (2005). Where do we go from here? Research and commercial spoken dialog systems. *6th SIGdial Workshop on Discourse and Dialogue*.

Poupyrev, I., Billinghurst, M., Weghorst, S., & Ichikawa, T. (1996). The go-go interaction technique: Non-linear mapping for direct manipulation in VR. *ACM UIST,* 79–80.

Steed, A., & Slater, M. (1996). A dataflow representation for defining behaviours within virtual environments. *IEEE VR*, 163–167.

Stiles, R., & Pontecorvo, M. (1992). Lingua graphica: A visual language for virtual environments. *Proceedings IEEE Workshop on Visual Languages*, 225–227.

Wang, Y.-Y., & Ju, Y.-C. (2004). Creating speech recognition grammars from regular expressions for alphanumeric concepts. *Eighth International Conference on Spoken Language Processing*.