Table des matières

1 Introduction **1**

2 Contexte **3**
2.1 La collaboration 4
2.1.1 Les systèmes collaboratifs 7
2.1.2 Les systèmes d'édition collaboratifs 7
2.2 La collaboration 3D en accord avec l'évolution du web 7
2.2.1 Introduction 7
2.2.2 Le web et le P2P : WebRTC 7
2.2.3 Le web et la 3D : WebGL 7
2.3 Les architectures évènementielles pour la collaboration 7
2.3.1 Sensibilisation lors de la collaboration 7
2.3.2 Intégration des contraintes métiers 7

3 État de l'art **9**
3.1 La visualisation et manipulation 3D collaborative sur web 10
3.1.1 Introduction 10
3.1.2 Les approches centralisées 10
3.1.3 Les approches décentralisées 10
3.1.4 Conclusion 10
3.2 Systèmes d'édition collaborative 10
3.2.1 Le modèle de cohérence CCI 10
3.2.2 Les approches pour les données 3D 10
3.2.3 Conclusion 10
3.3 Les systèmes évènementiels distribués pour la collaboration 10
3.3.1 Introduction 10
3.3.2 Domain Driven Design : lier le fonctionnel et le code 10
3.3.3 Command Query Responsability Segregation : garantir de l'intégrité des données 12
3.3.4 Event Sourcing : certifier la fiabilité et traçabilité des données 13
3.3.5 Conclusion 17

4 Contributions scientifiques **19**
4.1 Introduction 20
4.2 Modèle évènementiel pour l'intégration du domaine 3D lors de la manipulation d'objets 3D 20
4.2.1 Modèle général 22

4.2.2 Mécanisme de gestion de version . . . 25
4.2.3 Cohérence Eventuelle en CQRS . . . 26
4.2.4 Potentielles applications et autres utilisations . . . 26
4.2.5 Bilan . . . 27
4.3 Architecture de communication hybride . . . 27
4.3.1 Présentation générale . . . 30
4.3.2 Event Store . . . 32
4.3.3 Persistance à long terme . . . 33
4.3.4 Synchronisation client-serveur . . . 33
4.3.5 Gestion de la cohérence . . . 34
4.3.6 Bilan . . . 34
4.4 Conclusion du chapitre . . . 34

5 Implantation **35**
5.1 3DEvent : Plateforme web de manipulation et visualisation collaborative d'objets 3D . . . 36
5.1.1 Introduction . . . 36
5.1.2 Interface utilisateur . . . 37
5.2 Intergiciel P2P pour l'échange de données 3D . . . 38
5.2.1 Données d'échange . . . 40
5.3 Synchronisation des données . . . 40
5.3.1 Persistance à court terme . . . 40
5.3.2 Protocole de streaming pour la synchronisation . . . 40
5.3.3 Introduction . . . 41
5.3.4 Interface utilisateur . . . 41

6 Expérimentations et résultats **43**
6.1 Assemblage d'objets 3D sur le web par état . . . 44
6.1.1 Présentation de l"expérimentation . . . 44
6.1.2 Résultats . . . 44
6.1.3 Discussion et Conclusion . . . 44
6.2 Assemblage d'objets 3D sur le web avec architecture évènementielle . . . 44
6.2.1 Présentation de l"expérimentation . . . 44
6.2.2 Résultats . . . 44
6.2.3 Discussion et Conclusion . . . 44
6.3 Comparaison entre l'expérimentation 1 et l'expériementation 2 . . . 44
6.3.1 Résultats . . . 44
6.3.2 Discussion et Conclusion . . . 44

7 Conclusion **45**

Chapitre 1

Introduction

Chapitre 2

Contexte

Contents

2.1 La collaboration 4
2.1.1 Les systèmes collaboratifs 7
2.1.2 Les systèmes d'édition collaboratifs 7
2.2 La collaboration 3D en accord avec l'évolution du web . 7
2.2.1 Introduction 7
2.2.2 Le web et le P2P : WebRTC 7
2.2.3 Le web et la 3D : WebGL 7
2.3 Les architectures évènementielles pour la collaboration 7
2.3.1 Sensibilisation lors de la collaboration 7
2.3.2 Intégration des contraintes métiers 7

2.1 La collaboration

La collaboration est souvent définie comme un processus récursif[1] où deux (ou plus) personnes (ou organisations) travaillent ensemble à la croisée de buts communs en partageant leurs connaissances pour apprendre et bâtir un consensus. La collaboration permet l'émergence de conceptions partagées dans la réalisation de visions partagées dans des environnements et des systèmes complexes. Les imbrications de chaque domaine et la transdiciplinarité sont acceptés comme dans le concept de pensée complexe. D'ailleurs, dans sa définition de la complexité, Edgar Morin fait référence au sens étymologique latin « complexus » qui signifie « ce qui est tissé ensemble » [**?**]. La plupart des collaborations requièrent un élément dirigeant qui peut prendre une forme sociale (personne) au sein d'un groupe décentralisé et égalitaire (tout le monde au même niveau). L'élément dirigeant va souvent aider également trouver des consensus. La disponibilité des ressources peut également devenir un élément dirigeant dans la collaboration. Une équipe travaillant de manière collaborative peut concentrer plus de ressources, de reconnaissances et de récompenses lors d'une compétition comportant des ressources finies. La collaboration est aussi présente dans la recherche de buts opposés mettant en avant la notion de collaboration contradictoire (en opposition avec la collaboration constructive) ; la négociation et la compétition peuvent également faire partie du terrain de la collaboration.

Une application collaborative peut aussi intégrer les notions de coordination et de coopération :

— La **coordination** se base sur le principe d'harmonisation des tâches, des rôles et du calendrier dans des systèmes et environnements simples.

— La **coopération** permet de résoudre des problèmes dans des systèmes et environnements complexes dans le les participants aurait été incapables (temps, espace, connaissance, matériel) d'accomplir le travail seul.

Dans les années 2000, deux classifications ont été retenue concernant la collaboration. La première classification, décrite en 2007 par Gotta [**?**], propose un modèle segmentant la collaboration de manière structurée en quatre catégories : de la plus dirigée à la plus volontaire, en passant par l'hybride.

— *Collaboration centrée processus.* Les conditions requises du processus nécessitent l'engagement de l'utilisateur, qui doit, de part son rôle ou sa responsabilité,

1. Le « principe de boucle récursive » se retrouve dans le concept de la pensée complexe. Edgar Morin explique qu'« un processus récursif est un processus où les produits et les effets sont en même temps causes et producteurs de ce qui les produit » [**?**, p. 100].

diriger ses efforts dans la collaboration avec les autres. Cette stratégie se concentre sur les activités de manipulation collaborative 3D plutôt que sur leur contexte organisationnel afin de favoriser la synergie autour de la réussite d'un processus. Par exemple, pour la création et l'utilisation d'un modèle 3D dans un Business Information Modeling (BIM), il s'agit de favoriser la prise de décision sur un projet et communiquer à propos.

— *Collaboration centrée activité* Les activités partagées créent un sentiment de co-dépendance qui motive la collaboration entre les membres. La co-dépendance prend l'avantage sur le propre intérêt de chacun comme motivation pour collaborer. Le groupe a besoin de chacun pour que l'objectif soit considéré comme réalisé. L'intérêt personnel ou l'allégeance à l'esprit d'équipe peut aussi promouvoir la collaboration. Par exemple, la visualisation collaborative des activités des différents contributeurs dans l'Environnement Virtuel Collaboratif 3D (EVC 3D) permet à chacun de rendre compte de ses réalisations. Ce type de collaboration doit beaucoup à l'ergonomie de l'activité qui insiste sur la différence entre le travail prescrit et le travail réel : la tâche et l'activité.

— *Collaboration centrée communauté.* La participation de la communauté à la collaboration induit la contribution. En effet, les interactions professionnelles ou sociales peuvent encourager ou persuader les utilisateurs de partager leurs informations ou connaissances (exemple : les logiciels *open-source*)

— *Collaboration centrée réseau.* Les connexions réseau favorisent la coopération réciproque. Dans le but de récupérer des avis ou du savoir faire externe, un utilisateur peut faire appel à son réseau social pour supplémenter une autre interaction collaborative. Souvent utilisé dans le cadre d'urgences écologiques ou sanitaires (exemple : contributions OpenStreetMap lors d'ouragan ou de tsunami), la collaboration centrée réseau est très présente dans des situations l'expertise est fortement valorisée comme dans les BIM ou la visualisation scientifique, les contributeurs peuvent être intégrés en fonction des besoins des utilisateurs déjà présents sur le projet.

Dans le cadre de cette thèse, en se référant à cette première classification, les aspects centrés sur les activités sont mis en avant. En effet, la collaboration portant sur la modélisation 3D attend un résultat porté sur l'activité de conception. Celle-ci nécessite l'implication de personnes avec différentes compétences / connaissances qui doivent s'entraider pour parvenir à la mise en commun des objets 3D et réaliser leur objectif.

Une seconde classification proposée par Callahan et al. [**?**] s'intéresse au triplé collaboration par équipe, collaboration communautaire, collaboration en réseau. En contraste avec la précédente classification qui se concentre sur les équipes et une collaboration formelle et structurée, celle-ci offre plus d'ouverture :

— *Collaboration par équipe.* Dans une équipe tous les membres se connaissent. Il y a une interdépendance claire des tâches à effectuer où la réciprocité est attendue, avec un échéancier et des objectifs explicites. Pour réaliser son but, l'équipe doit réaliser les tâches dans un temps imparti. La collaboration par équipe suggère que les membres coopèrent sur un pied d'égalité (bien qu'il y ait souvent un chef) recevant une reconnaissance égale.

— *Collaboration communautaire.* L'objectif de ce type de collaboration est plus orienté sur la possibilité d'apprendre que sur la tâche elle-même, même si les centres d'intérêt sont partagés par la communauté. Les utilisateurs sont là pour partager et construire la connaissance plus que compléter un projet. Les membres vont aller voir leur communauté pour demander de l'aide sur un problème ou un avis et ramener la solution à implémenter dans leur équipe. L'adhésion peut être limitée et explicite, mais les périodes de temps sont souvent ouvertes. Les membres sont considérés comme égaux bien que les plus expérimentés peuvent avoir des statuts privilégiés. La réciprocité est un facteur important dans la communauté pour que cela fonctionne.

— *Collaboration en réseau.* La collaboration en réseau est une sur-couche de la collaboration traditionnellement centrée sur la relation d'une équipe ou d'une communauté. Elle s'appuie sur une action individuelle et un intérêt personnel qui resurgissent ensuite sur le réseau sous la forme de personnes qui contribuent ou cherchent quelque chose à partir du réseau. L'adhésion et les périodes sont ouvertes et non limitées. Il n'y a pas de rôle explicite. Les membres ne se connaissent pas forcément. Le pouvoir est distribué. Cette forme de collaboration est dirigée par l'avènement des réseaux sociaux, des accès à internet omniprésents et la capacité de se connecter avec divers individus malgré la distance.

Cette thèse, en se référant à cette seconde classification, s'intéresse plutôt sur la collaboration par équipe. La conception d'un objet 3D et ses différentes phases de modélisation constitue une problématique nécessitant l'apport de plusieurs intervenants avec leurs capacités propres et travaillant de concert à la réalisation d'un objectif commun dans un temps imparti (exemple : revue de projet). Là où la coopération et

l'effort conjoint pour réaliser un objectif sont nécessaires, le facteur temps reste un élément important à prendre en compte pour évaluer la productivité d'une session collaborative.

Le travail dans un EVC 3D facilite la compréhension de certaines problématiques liées à l'espace 3D ; c'est également un point de rencontre et d'échanges entre contributeurs sur le court terme et le long terme. Le croisement de ces deux dimensions, spatiale et temporelle, implique une multiplication des points de vue et donc des données à traiter sur le problème lors de la collaboration.

2.1.1 Les systèmes collaboratifs

2.1.2 Les systèmes d'édition collaboratifs

Modèle d'édition collaborative

2.2 La collaboration 3D en accord avec l'évolution du web

2.2.1 Introduction

2.2.2 Le web et le P2P : WebRTC

2.2.3 Le web et la 3D : WebGL

2.3 Les architectures évènementielles pour la collaboration

2.3.1 Sensibilisation lors de la collaboration

2.3.2 Intégration des contraintes métiers

Chapitre 3

État de l'art

Contents

3.1 La visualisation et manipulation 3D collaborative sur web 10
3.1.1 Introduction 10
3.1.2 Les approches centralisées 10
3.1.3 Les approches décentralisées 10
3.1.4 Conclusion 10
3.2 Systèmes d'édition collaborative 10
3.2.1 Le modèle de cohérence CCI 10
3.2.2 Les approches pour les données 3D 10
3.2.3 Conclusion 10
3.3 Les systèmes évènementiels distribués pour la collaboration 10
3.3.1 Introduction 10
3.3.2 Domain Driven Design : lier le fonctionnel et le code . . . 10
3.3.3 Command Query Responsability Segregation : garantir de l'intégrité des données 12
3.3.4 Event Sourcing : certifier la fiabilité et traçabilité des données 13
3.3.5 Conclusion 17

3.1 La visualisation et manipulation 3D collaborative sur web

3.1.1 Introduction

3.1.2 Les approches centralisées

3.1.3 Les approches décentralisées

3.1.4 Conclusion

3.2 Systèmes d'édition collaborative

3.2.1 Le modèle de cohérence CCI

3.2.2 Les approches pour les données 3D

3.2.3 Conclusion

3.3 Les systèmes évènementiels distribués pour la collaboration

3.3.1 Introduction

Les évènements comme base du comportement réactif

Systèmes Publish / Subscribe

Outils de monitoring et de benchmark

3.3.2 Domain Driven Design : lier le fonctionnel et le code

Domain Driven Design (DDD) (ou Conception Pilotée par le Domaine) est une approche de développement logicielle qui a pour objectif de définir une vision et un **langage partagé** (*ubiquitous language*) pour les personnes impliquées dans la construction d'une application [**?**].

Le but est de mettre l'accent principal d'un projet sur le domaine et la logique du domaine en basant des conceptions complexes sur un modèle du domaine afin d'initier une collaboration créative entre les experts techniques et les experts du

domaine pour raffiner itérativement un modèle conceptuel qui relève de problèmes spécifiques au domaine. Les différents concepts du DDD sont listés en suivant :

Le contexte est le cadre dans lequel un mot apparaît qui détermine sa signification

Le domaine est une ontologie, une influence, ou une activité. L'étendue du sujet auquel l'utilisateur applique un programme est le domaine du logiciel

Le modèle est un système d'abstraction qui décrit les aspects sélectionnés du domaine et qui sont utilisés pour résoudre les problèmes liés à ce domaine

Le langage partagé est un langage structuré autour du modèle du domaine utilisé par tous les membres de l'équipe pour faire référence aux activités de l'équipe permettant d'éviter la redondance et les ambigüités dans un contexte donné.

Le DDD permet de connecter le modèle et son implémentation en offrant plusieurs avantages :

— La plasticité du système est mise en avant par l'expression de règles et de comportements qui facilitent les changements fréquents.

— L'accent mis sur l'identification des interactions dans le système encourage la mise en œuvre d'une interface orientée tâches (*task-based UI*).

— La testabilité fonctionnelle est intégrée via les règles métier explicitées et concentrées dans une couche spécifique de l'application. Cela les rend plus facilement identifiable et testable automatiquement.

— La robustesse est améliorée face aux changements dans le SI.

Les notions plus détaillées qui se rapportent au DDD sont présentées dans [**?**] et [**?**].

Le DDD a également l'avantage de fonctionner en harmonie avec les principes de l'Event Sourcing (ES). L'approche logicielle DDD étant conçue pour refléter les évènements se déroulant dans la réalité métier qui ne sont pas interchangeables – la plupart du temps. L'utilisation d'architectures basées évènements est naturelle dans ce genre d'environnement.

Les disciplines liées à la 3D dans un contexte industrielles ont besoin de pouvoir communiquer sur un langage commun pour permettre à chacun des intervenants de s'exprimer dans la création du produit. Par exemple, la Conception Assistée par Ordinateur (CAO) est très utile dans un contexte d'ingénierie par l'utilisation de quatre propriétés fondamentales telles que l'historique, les fonctionnalités, la paramétrisation et le haut niveau de contrainte.

Programmation réactive Les différents livres publiés par Vaughn s'attachent également à montrer la nécessitée pour les différentes branche de l'informatique à proposer des systèmes pour robustes et flexibles face aux nouvelles demandent. La publication du Manifeste Réactif encourage le développement de systèmes « plus flexibles, à couplage faible et extensibles ».

3.3.3 Command Query Responsability Segregation : garantir de l'intégrité des données

Introduit par Greg Young en 2009 [**?**], Command Query Responsability Segregation (CQRS) est un patron de conception qui repose sur le principe de séparation des composants de traitement métier de l'information (écriture) et de la restitution de l'information (lecture). Le cadre offert par ce principe permet de lever certaines contraintes d'architecture comme la (*scalability*) en faisant apparaître de nouvelles forces : la gestion de la concurrence dans la collaboration sur des règles métier propre à la modélisation 3D dans des cadres d'application spécifiques. En effet, selon le type d'application des règles vont régir qui peut effectuer quelle type de modification ou quelles sont les modifications qui peuvent être acceptables dans un cas et non dans un autre. Le caractère vicié (*staleness*) d'une donnée dans un environnement collaboratif est récurrent. Une fois que la donnée a été montrée à un utilisateur, la même donnée peut être changée par un autre utilisateur, elle est altérée. CQRS pallie cela en répondant aux besoins suivants :

— En traitement/ écriture : besoins transactionnels, garantie de cohérence (*consistency*) des données, de normalisation.

— En consultation/lecture : dénormalisation, scalabilité.

La plupart des architectures en couches ne font pas explicitement référence à ces problèmes. Le fait de tout sauvegarder dans une base de données centralisée peut être une étape dans la gestion de la collaboration mais l'altération des données est souvent exacerbée par l'utilisation de caches comme accélérateur de performance.

L'**immuabilité** en CQRS est un concept clé qui prévient la modification de l'état interne d'une commande ou d'un évènement. Les commandes sont immuables car leur usage nécessite un envoi direct au domaine pour être traitées. Quant aux évènements, ils sont immuables car ils représentent ce qui s'est produit dans le passé (qu'on ne peut donc pas changer).

3.3.4 Event Sourcing : certifier la fiabilité et traçabilité des données

L'ES est une approche complémentaire au CQRS pour gérer la concurrence des données et capturer l'intention de l'utilisateur. Ce patron de conception stocke les évènements en mode ajout seulement (*append-only*) pour sauvegarder le résultat des commandes sur le domaine. Cela permet de conserver tous les changements qui ont mené à un état plutôt qu'uniquement l'état. Ce paradigme permet de recréer n'importe quel état d'un agrégat à partir de la liste d'évènements qu'on lui a appliqué. Cette liste représente une base la vérité du système. ES permet de simplifier les tâches complexes dans des domaines complexes comme la 3D en ne requérant pas la synchronisation de modèle de données et du métier.

L'ES est souvent présent dans des architecture asynchrones qui ont l'avantage de pouvoir utiliser des queues de message, plusieurs bases de données et où la partie lecture est éventuellement consistante. La plupart de la littérature concernant ce patron se trouve en ligne, dans des billets de blog, des présentations, ou de la documentation logicielle. La littérature académique est relativement réduite, souvent rapproché des travaux sur l'évolution de graphes. Cette section fournit un aperçu des différentes définitions données de l'ES et du vocabulaire lié à ce patron de conception.

Martin Fowler Martin Fowler a été le premier à utiliser le terme d'Event Sourcing en 2005. Il définit l'ES comme « une série de changements de l'état d'une application »'. « série d'évènements capture tout ce qui est nécessaire à la reconstruction de l'état courant' ». Il voit les évènements comme immuables et le journal d'évènement (*event log*) comme un stockage linéaire (*append only store*) des évènements. Les évènements ne sont jamais supprimés, le seul moyen de l"'annuler" consiste à effectuer générer un évènement rétroactif. Une fois qu'un évènement rétroactif est ajouté, l'évènement rétroactif agit à l'inverse de l'évènement précédent pour compenser ses effets. Dans ce billet, Fowler n'établit pas clairement la distinction entre les évènements et les commandes qui déclenchent ces évènements. Ce problème est considéré dans plusieurs travaux.

Greg Young Auteur renommé dans le domaine de l'ES, Greg Young décrit l'ES comme « le stockage de l'état courant sous la forme d'une série d'évènements et la reconstruction de l'état du système en rejouant cette série d'évènements' ». D'après lui, le journal d'évènements a également un comportement linéaire : les évènements qui sont déjà arrivés ne peuvent être défaits. Ce que Fowler appelle évènements rétroactifs, Young le décrit comme des actions inverses.

Udi Dahan Udi Dahan est également un auteur de billets de blog prolifique sur les systèmes ES. Dans sa définition de l'ES, Dahan insiste sur le fait que "l'état du modèle du domain est persisté comme un *flux* d'évènements plutôt qu'un simple instantané".

Les avantages et les inconvénients de l'approche ES, incluant et discutant notamment ceux relevés par [**?**].

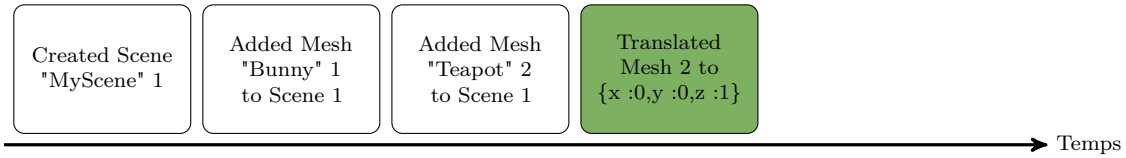


FIGURE 3.1 – Transaction en Event-Sourcing

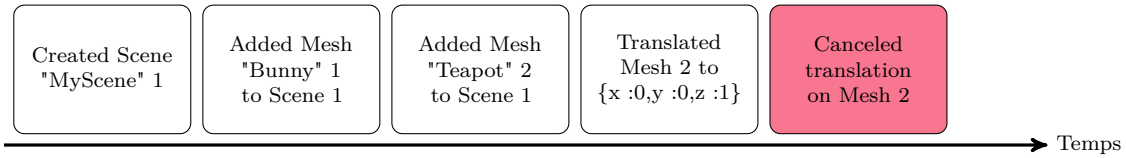


FIGURE 3.2 – Transaction avec compensation en Event-Sourcing

Avantages

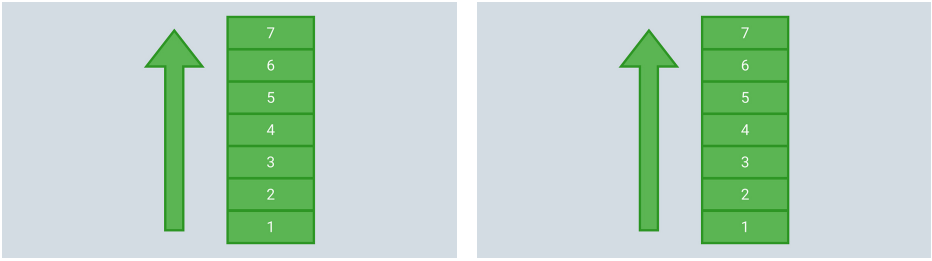
L'ES peut apporter beaucoup d'avantages à une application, notamment lorsque les besoins en traçabilité de l'information sont importants comme dans la modélisation collaborative de données 3D. L'historique du système est accessible tout au long de la vie de l'application ce qui implique qu'il est non seulement possible d'accéder à l'état courant du système, mais également toutes les actions ayant mené jusqu'à cet état. C'est un avantage certain pour les système critiques, les applications d'Informatique Décisionnelle (Business Intelligence (BI)) ou les applications collaboratives. La pratique la plus courante est de proposer un système principal et d'ajouter une multitude de sous-systèmes qui enregistrent les différentes métriques (analyse d'un ou plusieurs axes, *reporting* sur une propriété) du système principal. Avec l'ES, il est toujours possible de regarder «dans le passé» et récupérer les données à partir de ce moment. Par comparaison, les avantages de l'ES sur l'Active Record sont :

— un journal complet de tous les changements d'état,

— une traçabilité et un débogage efficace,

— de très bonnes performances,

— pas de mapping objet-relationnel (ORM).



(a) Sans la notion de snapshot (b) Avec la notion de snapshot

FIGURE 3.3 – Snapshot en Event-Sourcing

Exemple Une revue de projet d'une scène est effectuée dans le cadre d'une application collaborative de modélisation 3D en ligne. Le chef de projet remarque qu'un objet a subi énormément de modifications par rapport aux autres. Il examine en particulier cet objet pour en connaître les causes. Voici deux exemples d'observations possibles : 1/ les spécifications ne sont pas assez claires 2/ deux collaborateurs sont opposés sur la façon de modifier l'objet. L'origine de ces changements identifiée, le chef de projet pourra alors intervenir et clarifier le sujet.

Résolution Avec un système classique, la fonctionnalité doit être créée pour enregistrer ce traitement puis être testée et implémentée (dans cet exemple, il en faudrait deux). Seulement après, un rapport pourra être délivré. Dans un environnement utilisant l'ES, tous les évènements existent déjà donc les données sont déjà disponibles et ont seulement à être analysées (dans notre exemples, les informations sont inhérentes aux évènements). De plus, les évènements étant stockés depuis le début de la vie de l'application, beaucoup plus de données peuvent être analysées. On pourra par exemple demander toutes les modifications d'un objet depuis la dernière connexion d'un utilisateur pour lui communiquer visuellement les différences par rapport à sa dernière visite.

Inconvénients

Les performances de l'application peuvent être affectées par l'utilisation de l'ES. En effet, pour récupérer l'état courant à partir de l'Event Store, il est nécessaire de le calculer à partir des évènements reçus. A chaque évènement créé (et stocké), ce processus sera alourdi car ils ne peuvent être supprimés. Plus la pile d'évènements est longue, plus cela prend du temps (linéaire). On peut considérer d'une part que l'on a besoin des évènements que lorsque qu'une commande est effectuée. Un Snapshot est une capture de l'état de l'agrégat à un moment qui correspond à

l'empilement d'évènements ayant mené à cet état. En créant un Snapshot, on évite de reconstruire un état à partir du début de la vie de l'agrégat ; on empile les nouveaux évènements à partir de ce snapshot. L'utilisation du Snapshot est intéressante à partir du moment où la pile d'évènements devient plus lourde que le Snapshot lui-même. Il est donc important de bien déterminer la périodicité du déclenchement du Snapshot (temporelle ou selon le nombre d'évènements).

Le parcours des évènements se fait classiquement du premier au dernier (*bottom-up*). Si on utilise les Snapshot, on peut se permettre de les parcourir dans l'autre sens (*top-bottom*) jusqu'à trouver un Snapshot puis appliquer tous les évènements qui sont arrivés entre ce Snapshot et l'état courant. Dans le cas où l'on souhaite accéder à des données historiques plus anciennes que le dernier Snapshot, le second parcours fonctionne aussi. Cependant, elle doit être évitée pour ne pas créer de dépendance entre les Snapshot. Sans les snapshots le modèle peut encore varier « librement » tant que l'on sait comment lui appliquer un évènement passé. Le fait de travailler avec des Snapshot crée une dépendance des snapshots qui doivent intégrer les modifications du domaine. Une solution est de recalculer les snapshots quand le domaine est modifié mais cela reste coûteux (et à éviter).

Event sourcing vs command sourcing

Les patrons de conception Command Sourcing (CS) et ES sont strictement déterministes pour avoir une exactitude rigoureuse de ce qui se passe dans le système.

Un évènement représente quelque chose qui est arrivé dans le domaine. Un évènement appliqué sur un état, donne un nouvel état. La condition pour appliquer un pur déterminisme en ES est la fonction suivante : `State` → `Event` → `State`. Cette fonction met en correspondance les (mêmes) entrées avec les (mêmes) sorties, on peut se reposer dessus pour reconstituer un état à n'important quel moment dans le temps. Le déterminisme est appliqué en assurant à l'évènement tous les informations lui permettant de faire la transition (i.e. sans effet de bord). L'évènement est alors considéré comme une encapsulation de toutes les informations pertinentes concernant la transition du système d'un état à l'autre.

Une commande va être déclenchée par l'utilisateur qui veut modifier le domaine. Une commande appliquée sur un état va produire un ou plusieurs évènements (qui ne sont pas forcément appliqués à l'état courant de l'application par la suite) : `State` → `Command` → `Event list`

La différence principale entre un évènement et une commande réside donc dans l'**intention**. D'un point de vue fonctionnel, le CS est un patron de conception lié à une décision qui produit plusieurs évènements, tandis que l'ES se contente d'appliquer un changement à l'état courant. De plus, en ES, l'évènement stocké a été produit par l'agrégat. La différenciation majeure des deux patrons de conception s'accentue lors l'interaction avec des systèmes externes. L'aspect fonctionnel de l'application de changement d'état de l'ES al'avantage de permettre de reconstruire l'état sans effets de bord car elle ne travaille que sur l'état interne de l'application.

Historiquement, l'ES de Fowler dans ses premières versions ressemblait plus à du CS. L'évolution de l'ES a conduit à revoir ce patron théoriquement sous une forme fonctionnelle pure avec l'apparition de base de données comme EventStore ou le langage de programmation plus adaptés comme F#.

Les deux patrons peuvent cohabiter si le CS a un cadre d'action bien délimité et de les composants de l'architecture ont un couplage lâche afin que seuls les évènements produits par les agrégats ne modifient l'état interne de l'application.

3.3.5 Conclusion

Chapitre 4

Contributions scientifiques

Contents

4.1 Introduction 20
4.2 Modèle évènementiel pour l'intégration du domaine 3D lors de la manipulation d'objets 3D 20
4.2.1 Modèle général 22
4.2.2 Mécanisme de gestion de version 25
4.2.3 Cohérence Eventuelle en CQRS 26
4.2.4 Potentielles applications et autres utilisations 26
4.2.5 Bilan 27
4.3 Architecture de communication hybride 27
4.3.1 Présentation générale 30
4.3.2 Event Store 32
4.3.3 Persistance à long terme 33
4.3.4 Synchronisation client-serveur 33
4.3.5 Gestion de la cohérence 34
4.3.6 Bilan 34
4.4 Conclusion du chapitre 34

4.1 Introduction

Nous avons vu que les EVC 3D doivent considérer beaucoup de critères pour proposer un système réactif, robuste et permettant le passage à l'échelle en intégrant les contraintes liées à la gestion de données 3D. La dimension collaborative ajoute les problématiques de gestion de données concurrentes.

4.2 Modèle évènementiel pour l'intégration du domaine 3D lors de la manipulation d'objets 3D

L'event processing (EP) occupe, dans le champ de l'informatique, une place centrale dans beaucoup de systèmes comme l'énergie, la santé, l'environnement, les transports, la finance, les services et l'industrie. L'EP réunit des méthodes et des outils pour filtrer, transformer, et détecter des motifs dans des évènements, dans le but de réagir à des conditions qui changent, généralement liées à des contraintes de temps [**?**]. L'EP intègre plusieurs fonctionnalités :

— Obtenir des données à partir de plusieurs sources en temps (quasi) réel

— Agréger et analyser ces données pour détecter des motifs qui indiquent la présence de situation critiques qui nécessitent une réponse

— Déterminer la réponse la plus adaptée à ces situations

— Surveiller (*monitor*) l'exécution de cette réponse

Le panorama d'applications et de technologies proposé par [**?**] permet de définir les termes et les interprétations communes à différents domaines se basant sur le paradigme de l'EP.

La méthodologie orientée évènements intègre plusieurs aspects souvent laissés de côté dans la littérature concernant le développement les environnements virtuels collaboratifs pour la visualisation et la manipulation d'objets 3D. Par exemple, elle a l'avantage de proposer un couplage lâche. Dès lors, la réutilisation des données produite lors de la collaboration peut facilement être réutiliser pour un traitement secondaire comme la sensibilisation aux éléments de l'environnement.

Le découpage de la modélisation logicielle et l'abstraction qu'elle apporte est aussi l'occasion de s'intéresser à la façon dont sont générées les données produites par les utilisateurs et la façon dont elles sont affichées.

Constat

Par nature, une architecture orientée évènements est extrêmement peu couplée et hautement distribuée. Le créateur de l'évènement sait seulement que l'évènement se produit. Le créateur de l'évènement n'a aucune idée du traitement l'évènement va subir par la suite ou qui cela va concerner. La traçabilité peut donc devenir difficile lorsqu'un évènement empreinte de multiples chemin. C'est pourquoi les architectures orientée évènements sont plus utilisées dans un contexte de flux d'information asynchrone.

Il existe différent types de traitement d'évènement : simple, en continu, complexe. Ils peuvent être utilisés ensemble dans le cadre.

— Traitement d'évènement simple. (*simple event processing*). Dans le traitement simple d'évènement, un évènement notable arrive, initiant une cascade d'actions. Le traitement simple d'évènement est généralement utilisé pour orienter un flux temps-réel qui peut prendre un peu de temps, n'entraînant pas de coût sur la partie métier.

— Traitement de flux continu d'évènement. (*stream event processing*).

— Traitement d'évènement complexe. (*complex event processing*).

Le choix de baser la gestion des données sur le patron de conception CQRS combiné à de l'ES repose sur le constat suivant : dans un cadre industriel, le besoin de traçabilité de l'information est très important.

CQRS est principalement utilisé sur des applications CS qui sont toujours connectées (pour récupérer les mises à jour). Ce fonctionnement ne permet pas le travail hors ligne.

Côté serveur, le stockage des données est de moins en moins cher, on peut donc se permettre de stocker beaucoup de données de manière distante notamment grâce au cloud. Le serveur a une puissance de calcul plus importante (et surtout ajustable).

Côté client, la puissance de calcul des machines sur lesquelles sont installés les navigateurs web évolue rapidement (notamment les appareils mobiles comme les *smartphones* et les tablettes). Les navigateurs suivent cette tendance en puisant dans ces ressources pour effectuer des traitements similaires à ceux que l'on trouve côté serveur traditionnellement et pour proposer des fonctionnalités avancées telles que le stockage important de données sur le client (IndexedDB, storageAPI), pour l'affichage 3D (WebGL) et pour la communication en pair à pair (P2P) (Web Real-Time Communication (WebRTC)). En déportant ainsi la charge que pourrait subir une architecture traditionnelle client-serveur côté client, les échangent réseaux sont

limités car qui sont très coûteux d'un point de vue énergétique pour les appareils mobiles [**?**]. De plus, l'utilisation de la bande passante est onéreuse et parfois limitée voire inexistante, il est donc nécessaire de tirer parti de tous les appareils participants à la collaboration au lieu de tout faire reposer sur le serveur. Chaque appareil participant à la collaboration doit être autonome et le plus indépendant possible en termes de ressources (données, réseaux, validation experte).

Contribution

Pour répondre à cette problématique, nous proposons une solution pour la transmission des données 3D et collaboratives qui permet de limiter le nombre de requêtes et la taille des données transmises sans perdre la traçabilité de celles-ci. L'idée est de profiter de la puissance du client pour créer une architecture assurant l'autonomie de l'utilisateur en cas de déconnexion volontaire (travaille hors ligne) ou involontaire (coupure). On garantit ainsi à l'utilisateur l'utilisation du système avec un historique performant où chaque connexion est l'occasion de mettre le système à jour.

4.2.1 Modèle général

La composition de l'architecture s'est effectuée avec en arrière pensée les lignes directrices données par énoncées plus haut [**?**]. Utiliser une architecture orientée évènements pour faire de la modélisation 3D peut sembler non nécessaire. Cependant lorsqu'on s'intéresse aux apports que cela peut générer pour tous les métiers engagés (utilisateurs, développeurs, analystes métier) on peut se demander pourquoi cela n'a pas été plus mis en avant auparavant. La sensibilisation à l'historique des données et aux interactions inter-utilisateurs est partagée par les utilisateurs et les analystes métier. Quand à la sensibilisation à la distribution des données elles est une composante importante pour les utilisateurs et les développeurs. Concernant les premiers, cela garantie une certaine autonomie dans la création. Pour les seconds, la répartition de la charge permet de profiter du potentiel computationnel de toutes les parties prenantes du réseau.

Dans 3DEvent, le langage partagé se réfère au domaine de la manipulation d'objets 3D mais aussi au domaine de la collaboration. Par exemple le terme de maillage peut se référer à la fois au maillage géométrique ou bien au maillage de l'architecture réseau, d'où l'importance de définir les différents contextes en amont. Notons que le contexte de l'application peut faire varier les frontières d'un domaine.

Le modèle issu du domaine défini permet de mettre en valeur les aspects métiers liés à l'application.

Le patron Event Sourcing (ES) permet de capturer tous les changements d'état d'une application sous la forme d'une séquence d'évènements. Ces évènements sont conservés dans un journal d'évènements et peuvent être rejoués pour retrouver l'état de l'application. Les évènements représentent des faits immuables qui sont seulement ajoutés au journal les un après les autres, ce qui permet des taux de transaction élevés et une réplication efficace (cf Section 3.3.4). Dans 3DEvent, plusieurs composants d'ES sont étendues selon les applications :

Acteur Un acteur consomme des évènements à partir d'un journal d'évènements et produit des évènements pour le même journal d'évènements. L'état interne dérivé à partir des évènements consommés est un modèle d'écriture en mémoire (*in-memory*) et contribue à la partie commande (C) du CQRS.

Vue Une vue est un acteur qui ne fait que consommer des évènements à partir du journal d'évènement. L'état interne dérivé à partir des évènements consommés est un modèle de lecture en mémoire et contribue à la partie requête (Q) du CQRS.

Producteur Un producteur est un acteur qui produit des évènements à partir du journal d'évènements pour mettre à jour la base de données. L'état interne dérivé à partir des évènements consommés est un modèle de lecture en mémoire et contribue à la partie requête (Q) du CQRS.

Processeur Un processeur est un acteur qui consomme des évènements à partir d'un journal d'évènements et produit les évènements traités pour un autre journal d'évènements. Les processeurs peuvent être utilisés pour connecter les journaux d'évènement au traitement des évènements..

Les évènements produits par un des composants présentée ci-dessus peuvent être consommés par d'autres de ces abstractions s'ils partagent un journal d'évènements local ou distribué.

Un journal d'évènements peut fonctionner sur un seul site ou alors être répliqué sur plusieurs sites. Le site est considéré comme une zone disponible qui accepte l'écriture d'un journal d'évènements local même s'il est partitionné sur plusieurs sites. Les journaux d'évènements locaux situés sur plusieurs sites peut être connectés par le biais d'un journal d'évènements dit « repliqué » qui a pour responsabilité de préserver l'ordre causal des évènements.

Les sites peuvent être situés à des endroits géographiquement distincts ou sur des nœuds à l'intérieur d'une même grappe (*cluster*) ou encore être sur le même nœud mais traités séparément selon les zones disponibles nécessaires au fonctionnement de l'application. Les Acteurs et les Processeurs écrivent cependant toujours sur leur journal d'évènements local. Les composants peuvent soit collaborer sur un journal d'évènements local sur le même site, ou bien au travers d'un journal répliqué sur différents sites.

Il est important de différencier le journal d'évènements de la base de données (côté serveur). La base de données peut ne contenir qu'une partie du journal. Une base de données peut également être considérée comme un élément complémentaire au journal d'évènements, cependant et bien que parfois confondus, ils restent bien distincts conceptuellement.

Le journal d'évènements commun est la base des échanges pour communiquer par le biais d'évènements de collaboration. Ce type d'architecture se retrouve dans différents cas d'utilisation :

— *Processus métier distribués.* Les acteurs de différents types utilisent des évènements pour communiquer et parvenir à résoudre un problème commun. Bien qu'ils jouent des rôles différent dans le processus métier, ils réagissent à la réception d'évènements (*reactive programming*) en mettant à jour l'état de l'application et en produisant de nouveaux évènements. Cette forme de collaboration est appelée collaboration dirigée par les évènements.

— *Réplication d'état d'Acteur.* Les acteurs de même type consomme les évènements de chacun pour répliquer l'état interne avec une cohérence causale. Dans 3DEvent, les opérations concurrentes sont autorisées dans pour mettre à jour l'état des acteurs répliqués et permettre la résolution interactive de conflit en cas de mises à jour concurrentes et conflictuelles.

— *Agrégation d'évènement.* Les vues et les producteurs agrègent des évènements a partir d'autres composants pour générer des vues spécifiques à l'application. La collaboration évènementielle apporte de la fiabilité dans la gestion des données dans un système distribué. Par exemple, si un processus distribué échoue à cause d'un problème sur une partie du réseau, le système reprend automatiquement dès les répliques sont à jour.

Les composants souscrivent à leur journal d'évènements en s'accrochant au bus d'évènements. Les évènements nouveaux sont poussés vers les souscripteurs, ce qui leur permet de mettre à jour l'état de l'application avec une latence minimale. Un

évènement écrit à un endroit est publié de manière fiable aux souscripteurs sur ce site et aux souscripteurs des sites distants. Par conséquent, les composants qui échanges par le biais d'un journal d'évènements répliqué communiquent via un bus qui préserve l'ordre causal des évènements de manière durable et tolérant au partitionnement. De ce fait, les services sur les partitions du réseau inter-sites (lien entre les sites) peuvent continuer d'écrire des évènements localement. La livraison des évènements sur les sites distants reprend automatique lorsque les partitions sont à jour.

Le journal d'évènements répliqué localement et fournit un ordre total des évènements stockés et appartient à un site. Le site est une zone de disponibilité qui héberge un ou plusieurs journal d'évènements. Les évènements d'un journal d'évènements sont répliqués de manière asynchrone sur les autres des journaux d'évènements des sites distants. Afin de lier des journaux d'évènements (localisés sur différents sites) à un journal d'évènements répliqué, les journaux d'évènements locaux doivent être accessibles à partir des points d'entrées de réplication. De plus, ces points d'entrée doivent être connectés entre eux afin de créer un réseau de réplications. Un journal d'évènements répliqué est représenté par un journal d'évènements local sur chacun des sites participants.

Les points d'entrée permettent de gérer un ou plusieurs journaux d'évènements. Ces journaux sont identifiés pour permettre à la réplication de ne s'intéresser qu'aux journaux de même identifiant. Les journaux avec différents identifiants sont ainsi isolés les uns des autres et leur distribution peut donc varier selon les sites.

Les journaux répliqués fournissent l'ordre causal des évènements stockés : l'ordre de stockage est le même sur tous les sites, ce qui veut dire que les consommateurs qui lisent les évènements du journal local vont toujours voir les effets avant leurs causes.

4.2.2 Mécanisme de gestion de version

3DEvent intègre une procédure de gestion de version dans l'event store afin de gérer au mieux la cohérence des données. Chaque gestion de commande (Figure **??**) entraîne la génération d'un ou plusieurs évènements. Ces évènements sont considérés comme « soumis » (*uncomitted*) mais pas encore « publiés » (*comitted*). Pour qu'ils le deviennent, l'agrégat concerné par ces évènements doit produire une nouvelle version sans être en conflit avec la précédente. En passant la version attendue v_a au gestionnaire de conflit, on est à même de la comparer avec la version courante v_c. Il existe deux cas où un conflit est levé :

a) La version v_a correspond à la valeur d'initialisation

b) La version v_a est différente de la version v_c

Dans a), on s'assure qu'après une action la version initiale de l'agrégat ne peut être la même. Cet item peut sembler évident mais il est important de le noter car il dépend entièrement la valeur initiale choisie pour les agrégats du framework (on peut commencer à n'importe quelle valeur – -1, 0. . .).

4.2.3 Cohérence Eventuelle en CQRS

Linéaire (strong consistency) one version replace another – one parent and one children in the sequence each version is immutable each version has an identity each new version is a replacement of the previous (earlier) directed acyclic graph (eventual consistency) each version may have one or more parent each parent may have one or more parent each parent may have children with different parents each version is immutable each version has an identity

each version may be viewed as one of many replacement version for its parents version are immutable and (should) have immutable names

4.2.4 Potentielles applications et autres utilisations

La conception et l'implémentation d'une plateforme comme 3DEvent qui est asynchrone, distribuée et orientée évènements (notamment la persistance) peut être appliquée à différents champs.

GIT-like app Les solutions pour faire de la gestion de version, comme MeshGit [**?**] par exemple qui fait du *diff and merge* de maillages polygonaux pour des données 3D, sont rarement implémentées (a fortiori en temps-réel) sur des plateformes web à cause du coût et de la complexité que cela peut apporter dans des architectures traditionnelles. 3DEvent peut reconstruire n'importe quel état antérieur grâce à son architecture orientée évènements et indiquer les différences entre deux états.

Scenarii et *path recording* Pour des jeux sérieux, les graphistes 3D ou des études d'ergonomie, cette fonctionnalité est particulièrement pertinente. Le framework peut proposer une comparaison entre deux traces laissées par un ou plusieurs utilisateurs. Dans le cas où les utilisateurs ont la même tâche à réaliser, il est facile de faire la différence entre deux réalisations pour comparer, analyser et montrer les résultats pour des raisons pédagogiques ou pour relever des

habitudes (de travail) par exemple. Dans l'exemple du jeu sérieux, on peut comparer la trace utilisateur à la trace experte et permettre de rejouer le même scénario plusieurs fois facilement pour observer l'évolution. Ce type de fonctionnalité est intrinsèque à 3DEvent.

Traçage utilisateur et *crowdsouring* 3DEvent peut se révéler être un bon outil pour enregistrer la trace d'un utilisateur lorsqu'il navigue dans la scène. Si on choisit d'enregistrer le chemin de la caméra et les actions utilisateur comme des évènements que l'on considère comme des informations "issues de la foule" (*crowdsouring*). En utilisant un processus d'apprentissage, il est possible de proposer de meilleurs chemins, repérer des points d'intérêt, ou même proposer des résumés de scène générés à partir des traces des collaborateurs (que s'est-il passé depuis la dernière connexion du collaborateur X ?).

Audit et monitoring de données 3D L'ES fournit un mécanisme d'audit intégré qui assure la cohérence des données transactionnelles. Utiliser ce mécanisme pour faire un audit ou surveiller en temps réel l'activité de l'application peut fournir une meilleure compréhension du travail d'équipe ainsi que l'évolution de la conception. Cela peut permettre de repérer (avec du Complex Event Processing (CEP)) et corriger certaines fonctionnalités afin d'améliorer l'ergonomie de l'application. Par exemple, si un évènement est anormalement représenté dans le journal des évènements, on va pouvoir lever une alerte facilement.

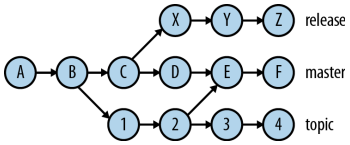


FIGURE 4.1 – Exemple d'arbre de commits Git

4.2.5 Bilan

4.3 Architecture de communication hybride

Dans la section précédente, l'introduction du modèle orienté évènement s'intéresse principalement à ce qui se passe au sein d'un seul client. Or, la collaboration passe par la mise en relation des différents collaborateurs et l'échange des données nécessaires à la collaboration : données de connexion, mises à jour de la scène, sensibilisation ...

Afin de pouvoir proposer un Système d'Edition Collaborative (SEC) adapté aux besoins de l'édition de scènes 3D, plusieurs critères doivent êtres respectés :

— granularité fine pour l'édition massive ;

— spécifique à la modélisation 3D ;

— reposant sur des technologies web (réseau, interface, interaction 3D).

Les SEC ont connu un fort développement avec l'intérêt croissant pour le P2P dans les années 2000. La base théorique des SEC s'appuie sur les propriétés de convergence, préservation de la causalité et préservation de l'intention du modèle CCI [**?**] (cf Section **??**). La plupart des travaux liés aux SEC P2P s'intéressent à l'édition collaborative massive de documents textuels dont les propriétés de commutativité sont plus faciles à gérer (insertion/suppression) en comparaison à celles concernant la 3D (multiplication de matrices de rotation). La génération de conflit en 3D peut vite devenir écrasante. Il est donc nécessaire de mettre en place des mécanismes de détection de conflit et de maintient de cohérence au cours des sessions de collaboration. Cela passe par la mise en place une architecture de communication hybride pour la collaboration. Le terme « hybride » invoque un compromis entre la centralisation de l'information nécessaire à la prise de décision globale et la décentralisation des échanges utile à l'amélioration du partage de contenu 3D à l'échelle locale. En agissant sur ces deux échelles la granularité de la collaboration est plus fine. Par exemple, le passage à l'échelle est facilité par la présence de ressources locales et la coordination des utilisateurs se fait à une échelle globale ce qui permet également l'accès à une source de vérité centralisée (base de données) et commune.

Cette section décrit le modèle d'architecture mis en place pour gérer la transmission de contenu entre les différents clients participant à l'édition collaborative d'un espace de travail 3D. Ces travaux s'inscrivent dans un contexte où les besoins d'interopérabilité et de standardisation sont élevés pour permettre à des utilisateurs de prendre le système rapidement en main sans installer autre chose qu'un navigateur.

Constat

WebRTC est une technologie qui fournit aux navigateurs *destkop* et mobiles la possibilité de communiquer en temps-réel (*Real-Time Communications*) via une collection de standards, protocoles et Application Programming Interfaces (APIs) JavaScript. Le standard WebRTC est développé au sein du World Wide Web Consortium (W3C) et de l'Internet Engineering Task Force (IETF) depuis 2011 dans sa version 1.0 (Working Draft). L'un des atouts de cette technologie est de permettre de façon simple et sans module d'extension la capture d'un flux audio et / ou vidéo (ex : applications de VoIP), ainsi que l'échange de données arbitraires entre navigateurs

sans nécessiter d'intermédiaires (ex : partage de fichier en P2P). Techniquement, WebRTC supporte un canal temps-réel bidirectionnel pour l'échange de données. Contrairement à WebSocket, qui est basé sur Transmission Control Protocol (TCP), WebRTC se base sur UDP en intégrant une pile de plusieurs protocoles (Figure 4.2) qui lui offre des fonctionnalités similaires (fiabilité, ordonnancement, sécurité).

La jeunesse du protocole (2011) fait que peu de travaux académiques l'utilisent pour créer des EVC 3D [**?**, **?**]. Côté commercial, un grand nombre d'applications et de services basés sur ce protocole ont fait leur apparition (par ordre d'importance) : des systèmes d'échanges audio/vidéo (VoIP) [1], des systèmes de partage de fichiers [2] et autres applications (capture d'écran, réalité augmentée, jeux) [3].

La simplicité de la mise en place (très semblable à WebSocket) et la richesse apportée par le P2P n'y sont pas étrangers. Les réseaux P2P ne sont souvent pas bien connus des développeurs web qui sont habitués aux architectures client-serveur, ce qui peut représenter une barrière d'entrée pour de nouvelles applications réseau. Dans les systèmes P2P, il n'y a pas de fermes de serveurs comme il peut en exister dans les applications client-serveur. Les machines clientes et les systèmes d'extrémité (*end system*) sont considérée comme « les ressources » dans un réseau P2P. En principe, ces ressources sont difficilement administrables à l'échelle dans un réseau non structuré ; la qualité de service peut en pâtir. Cependant, lors d'un déploiement P2P réel, des serveurs sont en général utilisés pour télécharger l'application cliente qui contient la couche intergicielle P2P et fournit à l'utilisateur le service de connexion nécessaire à l'accès au service. Ces serveurs sont moins coûteux que peut l'être une ferme de serveurs. De plus, pour beaucoup de services décentralisés, il n'existe pas de solution pérenne pour tester ces applications à l'échelle ; chaque service doit créer son propre système de test.

Ce contexte technologique est un motivation supplémentaire dans les travaux de cette thèse.

Contribution

Une architecture de communication hybride en trois parties serveur, persistance, pairs est utilisée dans 3DEvent [**?**]. Cela permet de concilier à la fois les avantages d'une architecture client-serveur et ceux du P2P en évitant certains désavantages

1. WebRTCWorld en liste un peu plus de 140 http://www.webrtcworld.com/webrtc-list.aspx. Consulté le 07/07/2017.
2. WebTorrent https://github.com/feross/webtorrent. Consulté le 04/09/2017.
3. Curation de ressources et modules WebRTC recensant plus de 100 projets https://github.com/openrtc-io/awesome-webrtc. Consulté le 07/07/2017.

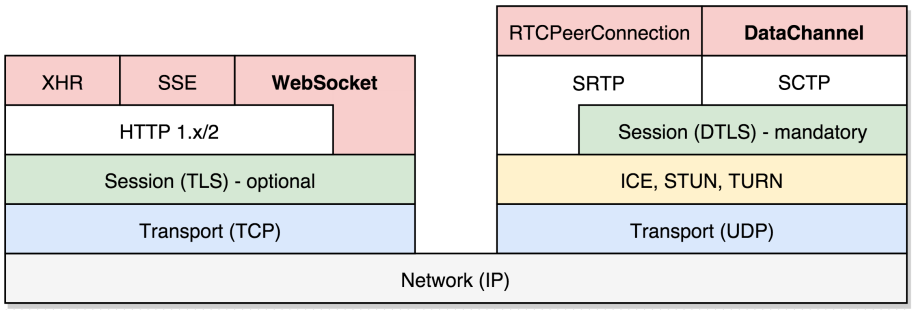


FIGURE 4.2 – Pile des protocoles IP : TCP vs UDP

occasionnés dans ceux-ci (engorgement du serveur, pair seul sur le réseau...). Dans le contexte des Environnement Virtuel Collaboratif (EVC) tridimension (3D), le but est d'obtenir une architecture de communication :

— entièrement basée web,

— robuste à l'évolution du nombre de collaborateurs,

— efficace en terme de d'accès et distribution de la donnée,

— et qui s'adapte à l'échelle selon les besoins de la collaboration.

Dans cette section, les différents composants de l'architecture sont détaillés. Ensuite sont expliqués les mécanismes de mise en relation de ces composants pour que les différents participants d'une session collaborative puissent se communiquer de manière transparente, cohérente et résistante aux pannes.

4.3.1 Présentation générale

Un intergiciel (*middleware*) se compose en général d'une API de recherche 4.3. Dans notre modèle cette brique est principalement en lien avec le gestionnaire d'instance qui se charge de la recherche. L'API de messagerie concerne plus directement le système d'envoie et de réception de message. Les différents message vont concerner la partie routage de l'information, le maintien à jour du répertoire de voisin et leur dynamicité, ainsi que la description des connexions à maintenir. Le bloc de gestion de session permet au pair de savoir avec qui il est connecté et contient également les méta informations liées à ces liens (temps de connexion). Chaque pair doit également posséder un endroit où stocker les information de contenu à traiter qui correspond à stockage de contenu. Cette dernière peut prendre plusieurs formes : in-memory, stockage local, disque dur... selon le type de stockage adapté et disponible. Un pair est également au courant de son rôle. Le rôle d'un pair permet de distinguer s'il est un super pair, un pair normal, un pair actif, ou un pair passif... Le rôle

induit les capacités et la configuration de chaque pair, i.e. un pair passif est configuré uniquement pour relayer les données tandis qu'un pair actif sera configurer pour traiter et relayer les données tandis qu'un pair passif ne fait que stocker et relayer les données.

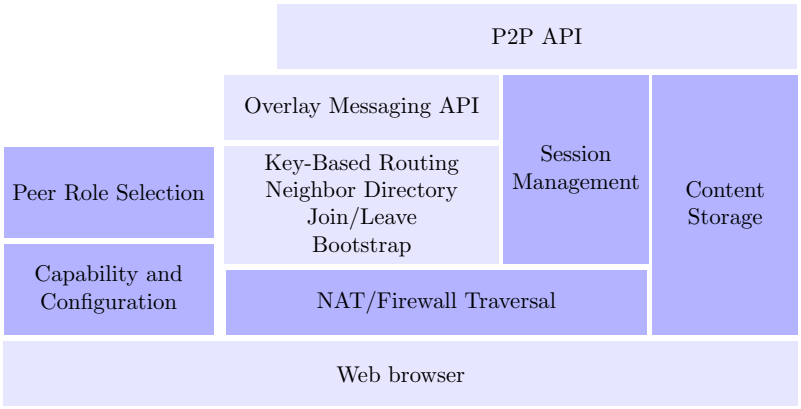


FIGURE 4.3 – Composants de chaque pair dans 3DEvent (point de vue réseau)

Le serveur assure d'une part la centralisation du stockage à long-terme et d'autre part la mise en relation des différents clients.

La couche P2P fournit une dissémination rapide de l'information et décentralise la réplication des données à court terme. Cela évite donc au serveur d'être le point central des échanges en déchargeant les canaux passant par le serveur.

Connexion des pairs en début de session La Figure 4.4 représente la séquence d'actions nécessaire à une instance 3DEvent (idA) pour rejoindre le réseau contenant déjà d'autres instances 3DEvent. L'action *join* est exécutée lorsqu'un utilisateur envoie ses informations de connexion sur portail de connexion (à partir d'une instance web) ou lorsqu'une instance serveur est lancée. Cette action ajoute le nouveau pair à la liste des pairs présents sur le réseau, gérée par le gestionnaire d'instance, et retourne la liste des pairs avec lesquels le pair doit se connecter. Pour chaque pair idB de la liste retournée ids, idA utilisé le mécanisme de signalisation (offre/demande). Le mécanisme est déclenché par l'instanciation d'un *network bridge* dans l'*event store* de idA puis celui de idB. Afin de resynchroniser les deux pairs (après cette série d'échanges asynchrones), idA et idB s'échanges des métadonnées sur la situation respective de leurs ESM afin de se synchroniser .

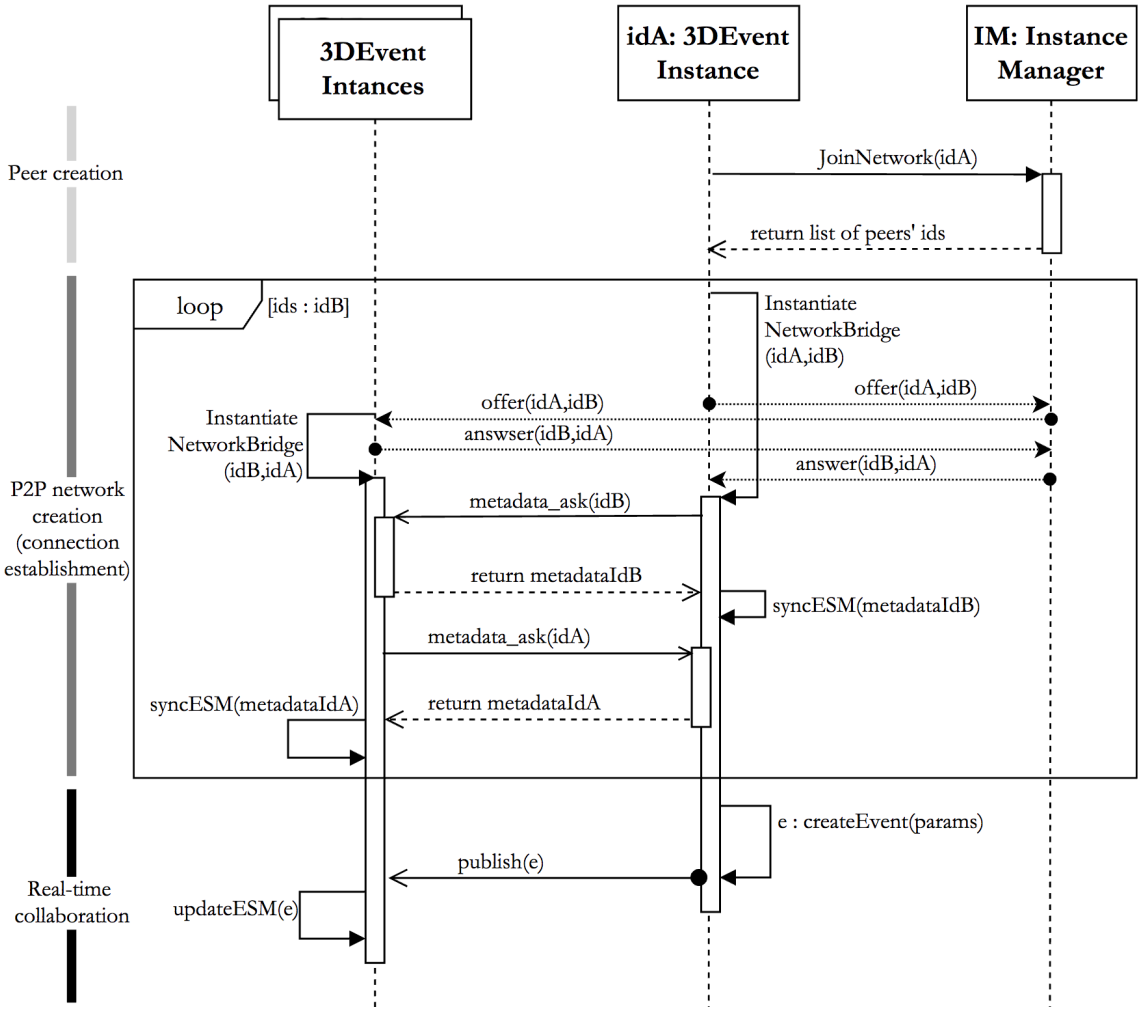


FIGURE 4.4 – Protocole de connexion au réseau d'instance 3DEvent

4.3.2 Event Store

L'Event Store est un composant clé dans le traitement des évènements. Il prend en entrée des évènements générés ou reçus extérieurement et produit en sortie des évènements cohérents qui peuvent être par la suite publiés. Les évènements sont considérés comme cohérents lorsqu'il n'y a pas d'erreur de cohérence, i.e. lorsque les numéros de versions sont bien ordonnés. Chaque Event Store contient deux éléments : l'Event Stream Manager (ESM) et des Network Bridges (NB). Un ESM est une structure de données représentée par une collection de flux d'évènements. Un flux d'évènement est représenté par un tableau d'évènements indexés en séquence permettant de stocker les évènements d'un aggrégat dans l'ordre temporel. Si un flux ne rencontre pas de problème de cohérence alors le dernier index correspond au numéro de version de l'agrégat. Lorsque l'Event Store reçoit un évènement de l'instance courante, l'ESM récupère (ou crée) le flux d'évènements associé à l'agrégat

référencé par l'évènement. Alors, la cohérence de la version est vérifiée en comparant la version attendue (exposée dans les métadonnées de l'évènement) et la version actuelle de l'agrégat. Si les deux numéros de versions sont égaux, l'évènement est ajouté à la fin tableau du flux pour être stocké, sinon une exception est levée. La gestion des exceptions est expliquée dans . Une fois le stockage effectué dans l'ESM, l'évènement est publié.

4.3.3 Persistance à long terme

Dans un contexte de collaboration sur le long terme pour des entreprises où le besoin d'information accessible sur le long terme est nécessaire. La persistance long-terme stocke le journal d'évènement qui est la source de vérité de l'application. Elle peut également stocker des projections prédéfinies, calculées à la volée ou encore des *snapshots* de l'application.

4.3.4 Synchronisation client-serveur

Lors de la connexion d'un nouveau client a lieu la synchronisation des deux systèmes de persistance pour obtenir les mises à jour :

1. depuis le client, où l'on distingue trois cas :
   (a) travail "*offline*" (hors ligne) : l'utilisateur a travaillé hors ligne et doit maintenant publier "en ligne" son travail. Les mises à jour publiées sur la base de données ; le serveur vérifie si aucun conflit ne survient puis fusionne (*merge*) les nouvelles entrées avec l'existant ;
   (b) travail "*serverless*" (en collaboration avec d'autres pairs sans le serveur) : dans le cas où le serveur est absent, les clients peuvent continuer de créer en collaborant. Ces données n'étant stockées que sur le client, il est de les transmettre à la base de données dès qu'une connexion est possible. Cela peut être fait en une fois ou de manière partagée ;
   (c) travail "*online*" (en collaboration avec d'autres pairs avec le serveur) : le client envoie régulièrement ses nouvelles modifications pour qu'elles soient intégrées à la base de données.
2. depuis la base de données : Le client reçoit toutes les nouvelles mises à jour de la scène depuis la dernière fois qu'il s'est connecté. Cela peut également inclure des mises à jour qui sont en conflit avec ce qu'il y a dans son propre espace de stockage qu'il lui faut donc modifier. Dans le cas où un utilisateur est seul

connecté, la base de données est la seule source disponible pour la mise à jour du client.

4.3.5 Gestion de la cohérence

Respect de la causalité

Convergence des répliques

Préservation de l'intention

4.3.6 Bilan

4.4 Conclusion du chapitre

Dans ce chapitre nous avons vu les différents composants de l'architecture de communication pour réalisation d'un EVC 3D. En mettant en avant la technologie WebRTC, nous avons montré qu'il était possible de réaliser une architecture qui respecte les standards du web et l'interopérabilité nécessaire à ce type d'environnement. La mise en place d'une architecture décentralisée dans un environnement distribué permet de mettre à contribution tous les acteurs de la collaboration. De ce fait, l'accessibilité des ressources est renforcée par la présence de nombreuses unités présentes sur le réseaux. Cela permet à la fois une récupération du contenu rapidement et octroie une autonomie certaine aux contributeurs.

Chapitre 5

Implantation

Contents

5.1 3DEvent : Plateforme web de manipulation et visualisation collaborative d'objets 3D 36
5.1.1 Introduction 36
5.1.2 Interface utilisateur 37
5.2 Intergiciel P2P pour l'échange de données 3D 38
5.2.1 Données d'échange 40
5.3 Synchronisation des données 40
5.3.1 Persistance à court terme 40
5.3.2 Protocole de streaming pour la synchronisation 40
5.3.3 Introduction 41
5.3.4 Interface utilisateur 41

3DEvent est à la fois un framework et un éditeur 3D pour la visualisation et la manipulation d'objets 3D. La partie framework est basée évènement pour répondre à des contraintes liés à la temporalité de l'information traitée via l'éditeur ainsi qu'à la distribution de l'information en terme d'intégrité et de poids. L'éditeur va réagir et interpréter les informations (évènements) fournies par le framework pour proposer visualisation et interactions adaptées.

5.1 3DEvent : Plateforme web de manipulation et visualisation collaborative d'objets 3D

5.1.1 Introduction

3DEvent propose un éditeur permettant de visualiser et manipuler des objets 3D avec des collaborateurs en temps-réel. L'application 3DEvent est un éditeur 3D reposant sur les principes et les technologies du web qui permet de manipuler des objets 3D de manière collaborative en temps-réel. Les interactions possibles sont :

Visualiser, naviguer, utiliser les outils de transformation L'utilisateur peut comme dans un environnement 3D classique interagir avec la vue en utilisant la souris (survol, clic) et en bougeant la caméra (déplacements). Il peut utiliser les commandes clavier et souris pour effectuer des opérations de translation, rotation et homothétie directement dans le *viewport* ou via le menu ou via la console du navigateur.

Charger des modèles 3D L'éditeur gère la plupart des formats de fichier 3D (OBJ, PLY, DAE, glTF...)

Changement de référentiel La modification des coordonnées de références (local/global) pour les différentes transformations possibles

Grid snapping Cette fonctionnalité permet d'aligner les modèles avec la grille avec un effet de magnétisme sur les intersection de la grille.

Changement de point de vue L'utilisateur peut à tout moment passer de son point de vue à celui d'un autre utilisateur. Le choix d'implanter ce type de fonctionnalité s'inscrit dans la perspective de sensibilisation de l'utilisateur au travail de ses collaborateurs. Ainsi, lors de la session, le fait de prendre de point de vue d'un collaborateur, se mettre à sa place, est une manière de comprendre son point de vue, son fonctionnement et d'imaginer ses perspectives de conception à travers le point de vue qu'il a choisit.

5.1.2 Interface utilisateur

Dans le but de proposer une Interface Utilisateur (IU) proche des fonctionnalités métier liées à la modélisation 3D avec une interface orientée tâche. De cette manière, le modèle orienté évènements présenté dans le précédent chapitre est directement orientée métier.

Présentation de l'interface

Lorsqu'un utilisateur se connecte à une scène, il a accès à une interface web (dans un navigateur) qui représente l'espace de travail collaboratif lui permettant d'utiliser différentes fonctionnalités. Les deux modalités d'interaction sont le clavier et la souris. Le premier niveau de cette interface est scindée en deux panneaux :

1. L'espace 3D consacré à la visualisation des objets et à leur manipulation dans l'environnement 3D ;
2. La barre d'outils qui contient trois onglets :
   (a) "Scene" contient tous les détails de la scène et des maillages qu'elle inclue ;
   (b) "Collaboration" fournit les informations liées à la collaboration ;
   (c) "History" liste tous les évènements qui ont eut lieu dans la scène et leurs détails.

L'onglet "Scène" possède un bloc contenant les détails d'un maillage en cours de sélection. Cela permet d'avoir la description des propriétés de l'objet sélectionné et une manipulation de ses paramètres (position, rotation et mise à l'échelle) plus précise que via l'espace 3D avec le cliqué/déplacé .

L'onglet "Collaboration" présente la liste des collaborateurs qui participent à la scène. Chacun d'eux est décrit par son nom, son état (connecté ou déconnecté) et son rôle (administrateur, éditeur, lecteur ou autre[1]). En cliquant sur un élément de la liste, l'utilisateur accède au dernier point de vue dans l'espace 3D connu du collaborateur représenté.

L'onglet "History" liste tous les évènements passés dans la scène en fournissant l'accès à leur détail. Pour chaque évènement, le système est capable de montrer dans l'espace 3D la différence entre l'état après l'évènement cliqué $state_x$ et l'état courant $state_n$. L'utilisateur peut à partir de cette visualisation choisir de « revenir en arrière » sans perdre les données entre $state_n$ et $state_x$ car dans notre système cela s'effectue par compensation (cf Event-Sourcing Section X).

1. Un rôle peut être défini par le biais du framework 3DEvent

Dans chaque onglet on trouve donc différent blocs HyperText Markup Langage (HTML), avec des comportements spécifiques à un agrégat et injectés dynamiquement. Ces blocs correspondent aux Views de notre modèle.

Flexibilité de la visualisation

Dans l'approche CQRS, une projection est une dérivation de l'état courant à partir du flux d'évènements. Pour Abdullin, «la projection est le processus de conversion (ou d'agrégation) d'un flux d'évènement en une représentation structurelle. Cette dernière (qui est mise à au moment où le flux est parcourue) peut être avoir différentes appellations : modèle de lecture persistent, vue ou état.» La partie lecture du modèle (l'affichage sur interface utilisateur) bénéficie des projections en lui permettant de réduire l'afflux des évènements, ne laissant filtrer que ceux qui sont pertinents pour la vue. La projection fournit une vue adaptée (filtrée, enrichie...) du flux d'évènements au client. Elle peut également être utilisée pour mettre en avant des aspects experts (notifications, déclenchement d'action) ou des raisons de confidentialité. Une projection peut être créée de manière synchrone (à la volée) au fur et à mesure de la publication des évènements ou de manière asynchrone et donc découplée du flux des évènements.

Du fait de la nature d'un réseau P2P, les pairs ne reçoivent pas forcément les paquets réseau de manière ordonnée. Par conséquent, les messages pouvant arriver dans n'importe quel ordre, qu'arriverait-il si un évènement A (eA) nécessitant un autre évènement B (eB) pour être appliqué arrivait avant ? Dans cette situation, le système va générer une erreur en essayant d'appliquer eA sur un état inadéquat car il n'a pas d'information sur la hiérarchie d'application des évènements (eB puis eA).

Pour pallier ce problème, l'introduction du système de projection permet d'avoir un mécanisme (comme un automate fini) qui défini les transitions nécessaires pour passer d'un état à l'autre et qui réalisent les actions déterminées en fonction des évènements qui arrivent. Par exemple, si on essaie d'ajouter un objet dans une scène (eA) sans avoir créer la scène (eB) la projection met en attente eA jusqu'à recevoir eB. Dans le cas où eB n'arrive jamais, la projection ne pourra jamais utiliser eA.

5.2 Intergiciel P2P pour l'échange de données 3D

L'assomption est faite que tous les clients utilisent des navigateurs qui implémentent et supportent le protocole WebSocket et l'API RTCDataChannel.

La topologie de l'architecture de communication repose sur la mise en relation automatique des clients par le biais du serveur pour établir une connexion WebRTC. Pour ce faire, chaque client envoie son identifiant (ID) lors de sa première requête au serveur qui le stocke. Selon le paramétrage de la connectivité directe minimum établie préalablement, le serveur recherche aléatoirement l'ID d'autres clients qui satisfont la règle de connectivité. Cette règle de connectivité minimum permet d'ajuster la densité du maillage (connectivité élevée : maillage partiel dense voire complet ; connectivité faible : maillage partiel éparse) et d'obtenir une topologie maillée adaptée aux besoins de l'application en termes de synchronisation (temps-réel ou pas) ou aux capacités des appareils. Il faut noter cependant que plus la connectivité est faible, plus l'information a besoin de « rebondir » pour atteindre tous les pairs et par conséquent le temps de transmission est augmenté (exemple d'une distribution en ligne).

De navigateur à serveur

La connexion entre un pair (client) et le serveur est établie sur la base du protocole WebSocket. Cette connexion bi-directionnelle est initialisée lors de la première requête du client pour récupérer le contenu de l'application. Cette connexion sert à la fois pour la phase de *signaling* lors de l'établissement d'une connexion WebRTC mais également pour que le client puisse envoyer des mises à jours originales à la base de données via le serveur.

De navigateur à navigateur

Lors de la connexion d'un nouveau client à la scène, le serveur effectue la phase de signaling permettant de le mettre en relation avec un autre client. Le mécanisme est répété tant que la règle de connectivité peut s'appliquer. Le client reçoit une notification de l'établissement de la connexion avec un autre client ce qui lui permet de démarrer l'échange de données.

L'API RTCDataChannel permet à chaque pair d'échanger des données arbitraires avec d'autres à partir du navigateur avec des propriétés de livraison personnalisables (fiable ou non fiable, ordonné ou non ordonné) selon le transport sous-jacent. Dans 3DEvent, le choix d'avoir un transport fiable et non ordonné a été fait pour respectivement garantir l'arrivée d'une donnée émise par l'utilisateur au sein de l'application et permettre des échanges asynchrones. En cas d'arrêt soudain du serveur, si une connexion a été établie préalablement entre les clients et est toujours en cours, elle n'est pas impactée par la défaillance du serveur.

5.2.1 Données d'échange

En sachant que le modèle est conçu pour des applications web, 3DEvent a besoin d'un format de données permettant de faire communiquer des acteurs hétérogènes du système. Le format JavaScript Object Notation (JSON), dérivé de la notation des objets du langage JavaScript, il est lisible et interopérable. Le format de fichier GL Transmission Format (glTF) se base sur la représentation JSON afin de décrire une scène 3D. Ce type de données est assez abstrait et suffisamment générique pour représenter n'importe quelle donnée. Par exemple le format de fichier glTFse base sur la représentation JSON afin de décrire une scène 3D. Le format JSON est aussi utilisé pour la sérialisation et la désérialisation des objets transmis par RTCDataChannel qui prend n'importe quel format de données.

L'API RTCDataChannel supporte beaucoup de types de données différents (chaînes de caractères, types binaires : Blob, ArrayBuffer...). Dans un environnement multi-utilisateur avec des données hétérogènes (3D, images, informations) tel que 3DEvent cela facilite l'interopérabilité.

5.3 Synchronisation des données

5.3.1 Persistance à court terme

Le navigateur (client) offre un espace de stockage avec l'interface *Storage* de l'API Web Storage qui donne accès au `session storage` ou au `local storage`. Grâce à un système clé/valeur, il est possible d'avoir une persistance des données à travers les sessions du navigateur. Le contenu stocké correspond aux données générées par un utilisateur et par ses collaborateurs. Les répliques stockées sur chaque navigateur permettent à un utilisateur une plus grande autonomie en cas de déconnexion. C'est également un moyen de distribuer les mises à jour générées par les clients entre les clients grâce au protocole de streaming 3D (cf. 5.3.2) sans passer par le serveur.

Ce stockage fonctionne sur un système de clé/valeur qui rend facile l'accès aux évènements enregistrés sur le client. La configuration du client est également stockée localement.

5.3.2 Protocole de streaming pour la synchronisation

Il existe plusieurs méthodes de transmission de contenu au sein d'un réseau P2P que l'on peut catégoriser selon deux modes : le téléchargement (*download*) et le

flux continu (*streaming*). Le téléchargement requière que le contenu soit entièrement téléchargé pour pouvoir être lu. Tandis que le flux continu peut être lu au fur et à mesure de sa récupération. Ce mode est principalement utilisé pour la lecture de vidéo en ligne. En comparaison le mode téléchargement est moins restrictif et relativement simple à mettre en place. Tout comme les systèmes utilisant une architecture client-serveur, la transmission de contenu en P2P peut également être catégorisée selon ces deux modes. Une catégorisation plus précise du flux continu peut être donnée selon quand le contenu est généré : en direct (live) et à la demande (*on-demand*).

Le mécanisme de routage que nous avons utilisé dans [**?**] est proche du routage de GNutella.

Base de données NoSQL

L'évolution de l'utilisation du web en tant que plateforme applicative a encouragé le changement dans le stockage des données pour de nouveaux besoin supportant de larges volumes de données (comme les données 3D). Une base de données Not Only SQL (NoSQL) fournie un schéma libre et dynamique ainsi qu'une API de requête riche pour la manipulation de données. De plus, la possibilité d'enrichir un document à la volée facilite l'évolution des objets (3D) et la maintenance. de l'application. 3DEvent intègre un système de persistance sur le long terme caractérisé par une base de données NoSQL.

La base de données (NoSQL) conserve tous les évènements qui sont produits dans une scène. La mise en place d'une base de données centralisée apporte de la robustesse au système en lui fournissant un référent sans toutefois le surcharger ainsi qu'une expérience utilisateur transparente limitant les interruptions de service.

5.3.3 Introduction

5.3.4 Interface utilisateur

Présentation de l'interface

Flexibilité de la visualisation

Chapitre 6

Expérimentations et résultats

Contents

6.1 Assemblage d'objets 3D sur le web par état 44
6.1.1 Présentation de l"expérimentation 44
6.1.2 Résultats 44
6.1.3 Discussion et Conclusion 44
6.2 Assemblage d'objets 3D sur le web avec architecture évènementielle 44
6.2.1 Présentation de l"expérimentation 44
6.2.2 Résultats 44
6.2.3 Discussion et Conclusion 44
6.3 Comparaison entre l'expérimentation 1 et l'expériementation 2 44
6.3.1 Résultats 44
6.3.2 Discussion et Conclusion 44

6.1 Assemblage d'objets 3D sur le web par état

6.1.1 Présentation de l"expérimentation

6.1.2 Résultats

6.1.3 Discussion et Conclusion

6.2 Assemblage d'objets 3D sur le web avec architecture évènementielle

6.2.1 Présentation de l"expérimentation

6.2.2 Résultats

6.2.3 Discussion et Conclusion

6.3 Comparaison entre l'expérimentation 1 et l'expériementation 2

6.3.1 Résultats

6.3.2 Discussion et Conclusion

Chapitre 7

Conclusion