Centrale Nantes

LAMIC

Géraud Le Falher

Stage ingénieur

Visite virtuelle de la chapelle des Ursulines de Québec

Fig. 0.1. : L'intérieur de la chapelle dans la version finale.

- Dates du stage : du 7 mai au 17 août 2012
- Établissement d'accueil : Laboratoire de muséologie et d'ingénierie de la culture de l'université Laval, Québec.
- Adresse : 1055 avenue du Séminaire, G1V 0A6, Québec, Canada
- Maître de stage : M. Phillipe Dubé
- Accompagnateur ECN : M. Erwan Verron

Table des matières

Table des matières 4
Introduction 11
Remerciements 14

Contexte du projet 14

1. Intervenants 15
1.1. Le LAMIC 15
1.2. La SAT 16

2. Chapelle des Ursulines 18
2.1. Historique 18
2.2. Chronologie 19
2.3. Plans architecturaux 19

3. Historique du projet 21
3.1. Avant Elsa 21
3.1.1. La numérisation, juin 2010 21
3.1.2. Maillage 21
3.1.3. Transfert sur le panoscope 22
3.1.4. Exploitation sur le panoscope 22
3.2. Après Elsa 22

Modélisation 24

4. La chapelle intérieure 25
4.1. Retopologie 25
4.1.1. Modèle existant 25
4.2. Texturation 27
4.2.1. Nouvelle texturation : UV mapping 29
4.2.2. Prise de photos avec Luc-Antoine Couturier 30
4.2.3. Problèmes et avancées 31

5. Mobilier 32
5.1. Méthode employée pour le maillage 32
5.2. La texturation 32

6. Les autels 33
6.1. Nouveau scan 33
6.2. Photogrammétrie 35
6.2.1. Introduction 35

6.2.2. Principe . . . 35
6.2.3. Solutions testées . . . 36
6.2.4. Récapitulatif . . . 38

7. Façade, sanctuaire et chapelle publique 39
7.1. Nuage de points . . . 39
7.2. Affichage tel quel . . . 39
7.3. Triangulation . . . 40

8. Modèle filaire 42
8.1. Les plans . . . 42
8.2. Choix du logiciel . . . 42
8.3. Choix de l'aspect : arrêtes et détails . . . 43
8.4. Choix pour le mobilier . . . 44

Exploitation du modèle 46

9. Unity 47
9.1. Présentation . . . 47
9.2. Fonctionnement visuel . . . 47
9.3. Aspect programmable . . . 48

10. Animation 49

11. Interactions 52
11.1. Modalités . . . 52
11.2. Gestion de contenu . . . 52
11.2.1. Type des contenus . . . 52
11.2.2. Stockage . . . 54
11.2.3. Insertion de contenus . . . 55
11.2.4. Affichage . . . 58

12. Interface 62
12.1. Clavier et souris . . . 62
12.2. iPad . . . 62
12.2.1. Approche simpliste . . . 62
12.2.2. Orientation . . . 62
12.2.3. Déplacement . . . 63

13. Immersion 65
13.1. Dôme . . . 65
13.2. Panoscope . . . 65
13.3. Ambiance sonore . . . 67

Conclusion 68

14. Perspectives 69
14.1. Nouveaux dispositifs . . . 69
14.2. Utilisation des cinq sens . . . 69

14.3. Personnalisation de la visite 69

15. Apports personnels **71**

Références **72**

A. Manipulation de la base de données **73**
A.1. Exporter la base de données 73
A.2. Importer la base de données 73
A.3. Télécharger le contenu « lourd » dans unity 73

B. Visite du 20 juillet à la SAT **74**
B.1. Situation 74
B.2. Déroulement de la journée 74
B.3. Conclusion 75

Index **77**

Glossaire **77**

C. Rapport de Camille sur les techniques de modélisation 3D **78**

D. Codes source **92**

Table des figures

0.1. L'intérieur de la chapelle dans la version finale. . . . 1

1.1. Vision d'artiste du LAMIC . . . 15
1.2. Le panoscope du LAMIC . . . 16
1.3. La disposition de la SAT . . . 17

2.1. Le plan au sol de la chapelle . . . 20
2.2. La coupe longitudinal de la chapelle . . . 20

4.1. État initial du modèle . . . 25
4.2. La retopologie du chœur en cours sous 3D Coat . . . 26
4.3. la retopologie de la nef, une fois recollée, sous Blender . . . 27
4.4. Une nouvelle tranche doit être insérée . . . 28
4.5. Création de la coupole par révolution . . . 28
4.6. Le chœur texturé. . . . 29
4.7. Luc-Antoine Couturier prenant des photos de la façade de la chapelle . . . 30
4.8. Deux exemple de textures pour l'UV mapping . . . 30

6.1. Modèle d'autel obtenu avec Polyworks . . . 34
6.2. Modèle d'autel obtenu avec my3Dscanner . . . 37
6.3. Modèle d'autel obtenu avec arc3D . . . 37

7.1. Le sanctuaire affiché en nuage de points . . . 40
7.2. Modèle polygonal de l'extérieur de la chapelle. . . . 41

8.1. Apparence partielle du modèle filaire en mode texture . . . 43
8.2. Première éventualité pour l'aspect des arrêtes . . . 44
8.3. Aspect visuel du modèle filaire à notre départ . . . 45

10.1. Squelette prêt à être animé. . . . 50
10.2. Silhouettes dans le modèle filaire. . . . 51

11.1. Signalement d'une possibilité d'interaction. . . . 53
11.2. Visualisation du contenu d'un objet. . . . 53
11.3. Schéma physique de la base de données. . . . 54
11.4. Interface de saisie et de lecture des commentaires. . . . 55
11.5. Gestion de contenus : page d'accueil. . . . 56
11.6. Interface de modification des contenus. . . . 56
11.7. L'architecture globale du système de gestion de contenus. . . . 57
11.8. Disposition des contenus au premier niveau. . . . 58
11.9. Disposition des contenus au deuxième niveau. . . . 59
11.10. Disposition des contenus au troisième niveau. . . . 59
11.11. Interface de visualisation de contenu en 3D. . . . 61

12.1. Affichage en réalité augmentée à l'extérieur de la chapelle. . . . 64

13.1. Dessin d'un panoscope. 66

C.1. Raccourcis clavier utiles dans Blender 2.63 90
C.2. Liste des fonctionnalités souhaitées lors de notre arrivée au LAMIC 91

Liste des tableaux

Chronologie de la chapelle des Ursulines 19

6.1. Récapitulatif des solutions testées en photogrammétrie. 38

Liste des codes source

D.1. unity/Bouton.cs . . . 92
D.2. unity/CameraFly.cs . . . 92
D.3. unity/CameraGUI.cs . . . 93
D.4. unity/Commentaires.cs . . . 93
D.5. unity/GetObject.cs . . . 94
D.6. unity/HandleImageEffect.cs . . . 96
D.7. unity/HGui.cs . . . 97
D.8. unity/IPServeur.cs . . . 101
D.9. unity/Layout.cs . . . 101
D.10. unity/Menu.cs . . . 106
D.11. unity/MouseLook.cs . . . 108
D.12. unity/NomsObjets.cs . . . 109
D.13. unity/OscReceive.cs . . . 109
D.14. unity/OscSend.cs . . . 110
D.15. unity/OSCServer.cs . . . 111
D.16. unity/OutputSetup.cs . . . 112
D.17. unity/OuvrirPorte.cs . . . 114
D.18. unity/Peuple.cs . . . 115
D.19. unity/PlacePlane.cs . . . 117
D.20. unity/Pointer.cs . . . 117
D.21. unity/PointerEvents.cs . . . 119
D.22. unity/PointerHandler.cs . . . 119
D.23. unity/Statut.cs . . . 120
D.24. unity/TextureState.cs . . . 120
D.25. unity/VoitTableau.cs . . . 121
D.26. unity/GyroCam.js . . . 130
D.27. unity/mycon.py . . . 130
D.28. CMS/form.php . . . 131
D.29. CMS/gestion.php . . . 132
D.30. CMS/get.php . . . 133
D.31. CMS/getxml.php . . . 133
D.32. CMS/modif.php . . . 134
D.33. CMS/send.php . . . 135
D.34. CMS/update.php . . . 135
D.35. CMS/upload.php . . . 136
D.36. CMS/utils.php . . . 137
D.37. CMS/file_upload.js . . . 143
D.38. CMS/validation.js . . . 147
D.39. CMS/bdd−structure.sql . . . 147

À la mémoire de mon père

Introduction

Comme tous les domaines, la muséographie est affectée depuis des années par l'essor des Nouvelles Technologies de l'Information et de la Communication (NTIC). Elles peuvent être utilisées pour connecter les musées et les fonds entre eux, en assurer la promotion auprès du public, élargir l'expérience pré et post visite, etc. C'est aussi l'occasion de repenser le musée, en tant que lieu d'exposition mais aussi comme vecteur de transmission de l'histoire et du patrimoine. Ainsi ces outils numériques semblent parfaitement adaptés pour toucher les jeunes générations mais aussi pour mettre en valeur de façon inédit tout ce que la culture comporte d'immatériel. Une telle refondation nécessite toutefois d'indéniables compétences techniques, ce qui justifie l'appellation du Laboratoire de Muséologie et d'Ingénierie de la Culture (LAMIC) (ainsi que celle de la Société des arts technologiques (SAT) , qui est l'autre acteur du stage) et ma venue dans celui-ci. Parmi les différents projets destinés à explorer le potentiel de nouvelles technologies dans la conception muséographique, j'ai travaillé sur la modélisation et l'exploration de la chapelle des Ursulines de Québec. Il s'agissait de présenter lors de la dernière semaine un modèle convaincant et exploitable qui permette de réaliser une visite virtuelle et enrichie en contenus de ce lieu de la ville de Québec, et notamment de sa partie qui est d'ordinaire inaccessible au public. Une vue d'ensemble des tâches qui nous ont été proposées est disponible en annexe, p. 91.

C'est un projet qui a débuté en 2010 avec le scan du bâtiment, et qu'a poursuivi une autre stagiaire de Centrale l'année dernière. Après avoir pris connaissance de ses travaux, nous nous sommes penchés sur les méthodes à mettre en œuvre pour obtenir un modèle à la fois précis et performant du lieu. Celles-ci se sont révélées aussi variées qu'il y avait d'éléments différents. Pour la chapelle intérieure il a fallu combiner retopologie et prises de vue réelles, ce qui n'a pas été nécessaire pour le mobilier. Nous nous en sommes en revanche intéressés à la photogrammétrie pour les autels intérieurs et avons cherchés une méthode plus simple pour la façade extérieur et la chapelle publique. Enfin, comme pour confirmer la nature expérimentale et artistique du LAMIC , nous avons aussi réalisé un modèle filaire destiné à faire ressortir l'architecture de la chapelle.

En parallèle de ces travaux qui concernent essentiellement l'aspect visuel, nous nous sommes demandés quels moyens nous permettraient de rendre ce modèle attractif du point de vue muséal, c'est-à-dire qui engage le public dans une expérience novatrice, captivante et pédagogique. Le principal outil qui a servi à support de ces réflexions a été Unity, même si des animations ont été réalisées avec Blender . L'interactivité avec les objets remarquables présents dans la chapelle a été traitée par la création d'une architecture client/serveur permettant de gérer un contenu muséographique et de le présenter au visiteur à sa demande. Il s'est en revanche révélé plus délicat de définir des modalités d'interface fournissant une navigation naturelle satisfaisante au sein du modèle. Enfin, nous avons pu bénéficier des dispositifs d'affichage du LAMIC pour travailler l'immersion dans le modèle[1].

Au risque de susciter l'ire des lecteurs friands de développements haletants et d'invraisemblables retournements de situation, je me résous à dresser dès à présent un bilan des résultats obtenus par rapport à ces objectifs. Partant d'un scan très lourd, nous sommes parvenus à obtenir un

1. Si j'ai écrit ces deux paragraphes à la première personne du pluriel, c'est pour souligner que ces résultats sont le fruit d'un travail d'équipe qu'il serait assez puéril d'essayer de fractionner, bien que cela soit techniquement possible. L'honnêteté intellectuelle me pousse toutefois à dire que c'est Camille qui a majoritairement réalisé la première partie à laquelle je me suis contenté de participer par l'intermédiaire de quelques opinions. Il est en de même pour la rédaction de ce document, dont certains passages ne sont pas de ma main. Ces précisions étant faites, je n'alourdirai pas la suite du rapport en revenant sur le sujet, d'autant qu'il est censé donné une vue d'ensemble du travail accompli.

modèle relativement manipulable tout lui donnant une facture visuelle qui remplit parfaitement sa fonction illusoire[2]. En ce qui concerne la partie exploitation, le constat est plus mitigé. Unity s'est certes révélé bien adapté à notre usage. Et le système de gestion de contenu est fonctionnel et répond à l'essentiel de son cahier des charges. Mais l'immersion et les interfaces, si elles sont présentes, auraient pu bénéficier d'un travail additionnel, notamment en utilisant des outils plus novateurs. Globalement, le résultat de ce stage — en ce qui me concerne — est plutôt positif.

2. en ce sens que nous ne visions pas le photoréalisme, qui lui aurait fait perdre sa fonction de médiation entre le contenu présenté et le lieu réel

Remerciements

Ce stage m'a permis de rencontrer de nombreux individus passionnants qui m'ont tous offert leur aide sans la moindre arrière pensé et que je tiens ici à remercier.

Tout d'abord le directeur du LAMIC , **M. Philippe Dubé**, qui nous a accueillis chaleureusement et dont l'enthousiasme indéfectible et la confiance quasi-aveugle furent d'un grand soutien tout au long de ces quinze semaines. Au quotidien, nous avons aussi pu compter sur deux doctorants du LAMIC, **Mme Dominique Gélinas** — dont les discussions stimulantes sur *Star Wars* et *Diablo* ne doivent pas faire oublier sa générosité à mon égard, puisqu'elle m'a offert un vélo et une place pour le festival d'été — ainsi que **M. Jean-Ambroise Vesac**, dont la bonne humeur et l'heureux événement familial qu'il a vécu n'ont jamais nuit à son implication dans le projet. Enfin, si nous avons souvent été bloqué par des problèmes matériels, nous avons toujours pu compter sur la diligence de **M. Louis-Robert Bouchard**, technicien de l'université.

Au sein de l'université, nous avons aussi eu le plaisir de collaborer avec **M. Thierry Moszkowicz**, du laboratoire de vision numérique, **Mme Sylvie Danielle**, dont les éclairages nous furent fort utiles et qui nous a mis en relation avec l'un de ses étudiants pour la partie géolocalisation, **M. Benoit Duinat**. Pour ne pas trop souffrir de la nostalgie de la mère patrie, nous avons contacté **MM. Robert Vergnieux et Pascal Mora** du laboratoire SHS3D de Bordeaux qui nous ont fournis de précieux conseils sur la photogrammétrie. Et même si nous n'avons pas eu l'occasion de nous revoir, notre rencontre avec **M. Sebastien Roy**, qui travaille à l'université de Montréal sur la projection de contenu, fut très enrichissante.

Notre modèle n'aurait sans doute pas atteint son aspect final sans la participation active de **M. Luc Antoine Couturier**, qui par les contorsions auxquels il s'est prêté dans un lieu saint nous a fournit de nombreux points de vue indispensables, ni sans l'énergie déployée par **Guy Côté et Claire Delisle** de la société MCG3D. Toute la communauté des Ursulines s'est aussi mobilisée pour le succès de notre entreprise, des techniciens jusqu'aux sœurs. Je tiens en particulier à souligner le concours de **sœur Gabrielle Noël** qui nous a guidés dans la chapelle, ainsi que celui de **Mme Christine Cheyrou** la directrice du musée des Ursulines et de **Mme Mélanie Girard**, qui nous l'a fait visiter. Nous avons aussi pu bénéficier des explications enrichissantes des employés de chez Créaform dans leur locaux de Levis, notamment celles de **M. Louis-Philippe Gendron**.

Lors de notre passage à la SAT , la bienveillance de son directeur, **M. Luc Courchesne**, ainsi que la mobilisation de toute son équipe et notamment de **M. Sébastien Gravel** nous a permis de surmonter de nombreux problèmes techniques afin de profiter pleinement de leur dispositif d'affichage immersif.

Si j'ai pu trouver ce stage presque sans aucun effort, c'est grâce à ceux qu'a déployés **Elsa Goarrin**, qui m'a précédé au LAMIC . Elle a répondu à toutes mes questions pratiques avec une grande disponibilité et son rapport [4] s'est révélé précieux.

Ces longues heures devant l'ordinateur auraient été bien ternes et peu productives sans la présence à mes côtés de **Mme Camille Autran** dont je ne pourrais chanter les louanges à leur juste valeur même en y consacrant le reste de mon rapport. Je me contenterai donc de citer son entrain et sa détermination perpétuels, sa patience angélique envers mes marmonnements et ses précieux conseils culinaires — quand bien même leur exécution n'a pas toujours produit les résultats escomptés. Et au-delà du stage, je n'aurais sans doute pas vécu le quart de mes expériences québecoises sans son sens développé de l'initiative[3].

3. et le tout à l'abri des moustiques…

Enfin, il serait discourtois de ne pas saluer la chaleur et l'hospitalité de tous les québecois que j'ai rencontré et qui ont contribué à faire de ce séjour une expérience inoubliable – à l'exception notable d'un dénommé Jean Charest, dont le mauvais esprit l'a probablement conduit à une déroute bien mérité[4].

4. à l'heure où j'écris ces lignes, les urnes n'ont pas encore livré leur verdict mais on ne peut douter de la précision légendaire des sondages.

Première partie .

Contexte du projet

1. Intervenants

1.1. Le LAMIC

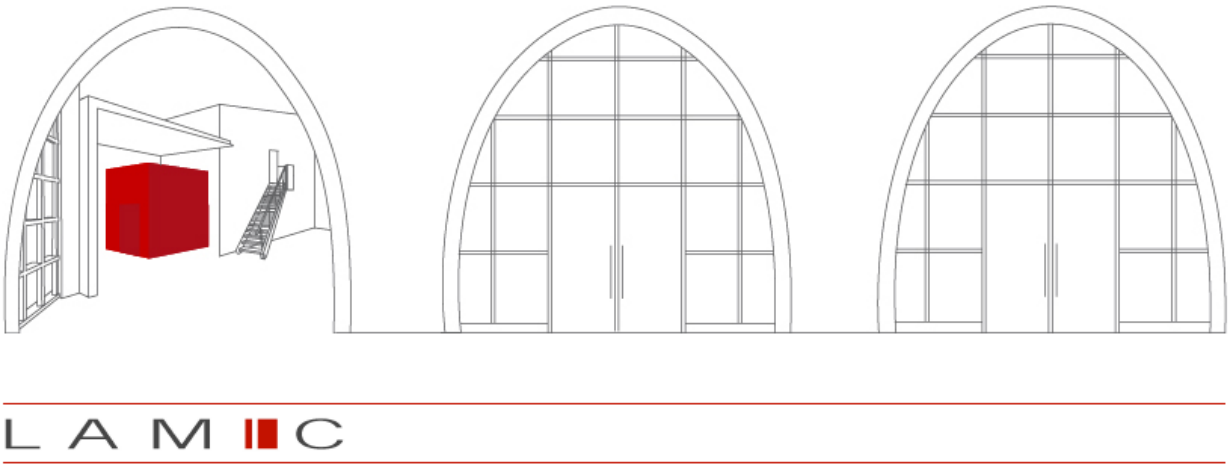


Fig. 1.1. : Vision d'artiste du LAMIC

Le LAMIC, laboratoire de muséologie et d'ingénierie de la culture, se situe au sein de l'Université de Laval à Québec (Canada). Fondé en 2004 par son directeur actuel, M. Philippe Dubé et grâce à une subvention de trois millions de dollars de la Fondation Canadienne de l'Innovation, le LAMIC a pu s'installer dans l'ancien centre muséographique de l'université. Il occupe des locaux s'étendant sur une superficie de 500m^2.

Ce laboratoire est une petite structure constituée d'un nombre variable de personnes. On y retrouve actuellement deux doctorants — Dominique Gélinas, doctorante en muséologie et Jean-Ambroise Vesac, doctorant en arts visuels —, son directeur Philippe Dubé et occasionnellement un polytechnicien spécialisé en informatique, Louis-Robert Bouchard.

Le laboratoire se veut une plate-forme d'expérimentation en muséologie : musées, artistes et organismes proches sont encouragés à venir y découvrir les possibilités que recèlent pour eux les nouvelles technologies. Elles sont représentées par un panoscope 360° (que l'on peut voir Fig. 1.2 p. 16), un scanner 3D et d'autres encore. Celles-ci se situent dans différentes salles : laboratoire de captation équipée d'un poste de numérisation 3D (scanner Arius 3D) et d'un poste de tracéographie (microscope électronique), laboratoire de spatialisation Expolab (le panoscope s'y trouve actuellement) communiquant directement avec une salle de montage vidéo et une régie. Cette dernière sert en particulier à créer et réaliser des expositions, et permet en outre l'observation du comportement des visiteurs face à celles-ci. On peut également noter que le LAMIC possède sa propre salle de séminaire (située à un niveau supérieur).

Le LAMIC participe à de nombreux projets en collaboration avec différents organismes tels que la SAT , MCG3D (compagnie de numérisation 3D dont un des locaux se trouve être le laboratoire

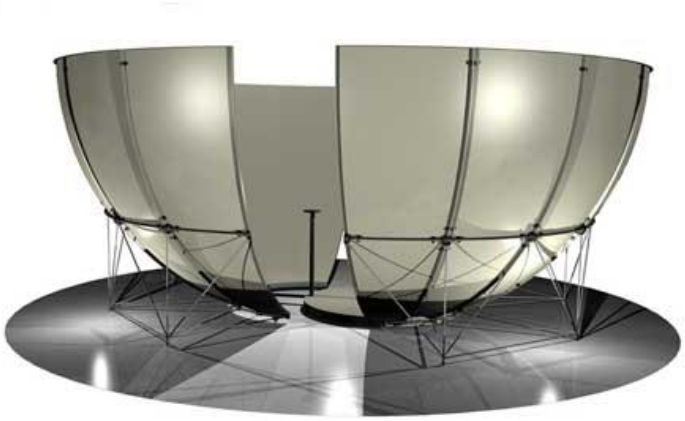


FIG. 1.2. : Le panoscope du LAMIC, fourni par la SAT

de captation du LAMIC) et bien sûr différents départements de l'Université Laval. Il est rattaché au centre d'étude interdisciplinaire sur les lettres, les arts et les traditions, qui est situé à Montréal.

1.2. La SAT

Le projet de numérisation et d'exploitation du modèle 3D de la chapelle intérieure du monastère des Ursulines est mené en collaboration avec la SAT. Elle se décrit ainsi sur son site internet :

> « La SAT est un centre transdisciplinaire de recherche et création, de production, de formation et de diffusion voué au développement et à la conservation de la culture numérique. »

Il s'agit d'un regroupement à but non lucratif d'artistes et de chercheurs ayant la même envie d'expérimenter ou de travailler avec les nouvelles technologies numériques, chacun dans leur domaine respectif. Elle fut fondée en 1996 par Monique SAVOIE, actuellement présidente et directrice artistique du centre, qui est basé à Montréal.

Le panoscope 360° du LAMIC est une interface immersive développée par la SAT, pouvant se présenter sous deux formes selon les besoins : un dôme ou un dôme inversé. Une image est projetée sur le dôme qui constitue un écran circulaire enveloppant. L'immersion est autant sonore que visuelle grâce à des haut-parleurs situés tout autour du dôme. Contrairement au LAMIC, la SAT posède un très grand dôme « à l'endroit » sur lequel nous avons pu tester notre modèle pour une complète immersion.

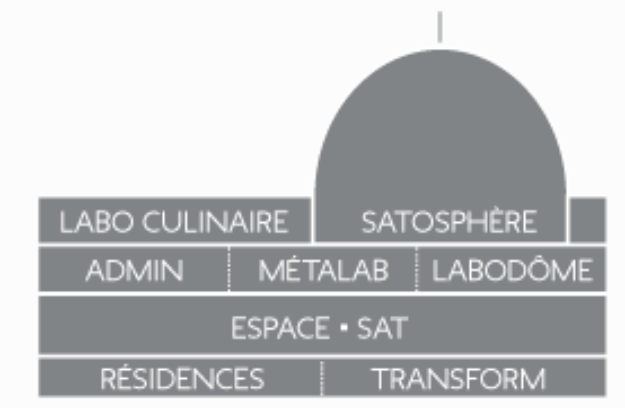


Fig. 1.3. : La disposition de la SAT

2. Chapelle des Ursulines

2.1. Historique

L'ordre des Ursulines fut fondé en 1535 par Angèle de Merici en Italie. La vocation de cet ordre est le soin des malades et l'éducation de jeunes filles. À Québec, il s'intéressa d'abord à l'éducation des jeunes filles amérindiennes, mission complémentaire à celle des jésuites qui avaient ouvert une école pour garçons en 1635[1]

Le monastère des Ursulines de Québec a été fondé le 1er août 1639 par Marie de l'Incarnation et Marie-Madeleine de Chauvigny de la Peltrie (la bienfaitrice du monastère) qui avaient entrepris dans ce but un voyage depuis la France. Initialement dans la Basse-Ville de Québec et dans un immeuble étroit, il fut décidé de construire un nouveau bâtiment en 1637 sur un terrain cédé par la Compagnie de Cent-Associés.

Ce monastère fut détruit (tout ou en partie) et reconstruit à plusieurs reprises et pour diverses raisons. Un incendie le détruisit complètement en 1651, et en partie en 1686 (chapelle extérieure). En 1870 le monastère devint trop étroit pour les religieuses et leurs élèves, et on y ajouta donc la chapelle Sainte Angèle, qui plus tard sera dédiée à Marie de l'Incarnation. Enfin en 1900 on décida d'agrandir le Choeur des religieuses mais pour des raisons techniques (risque d'effondrement) il fallu également reconstruire l'ancienne chapelle.

Ce monastère se situe actuellement au 12 rue Donnacona à Québec et est reconnu comme patrimoine de l'UNESCO. À côté de la chapelle extérieure on trouve un musée dédié à l'histoire de ces religieuses.

1. Il faudra d'ailleurs attendre 2010 pour voir les premiers garçons accueillis dans cette école

2.2. Chronologie

1639	Les Ursulines s'installent dans la Basse-Ville de Québec
1637	Concession d'un terrain par la compagnie des Cent-Associés sur lequel les Ursulines vont construire leur monastère
1641	Début de la construction du monastère
2 novembre 1642	Les Ursulines entrent dans leur premier monastère avant la fin de sa construction
1648	Fin de la construction
30 décembre 1650	Un incendie rase entièrement le monastère qui doit alors être reconstruit.
Mai 1652	Les Ursulines entrent dans le nouveau monastère
1656—1667	Construction de la première chapelle intérieure du monastère
20 octobre 1686	Un incendie détruit complètement le monastère et la chapelle extérieure
Été 1687	Début de la reconstruction du monastère et de la chapelle intérieure
1715	Construction de la nouvelle chapelle extérieure
19 mars 1723	Première messe dans la chapelle extérieure
1726—1739	Bombardement de Québec par les armées anglaises. La sacristie, la chapelle des Saints, une partie du chœur et de l'église sont faiblement touchées.
21 novembre 1902	La chapelle devenue trop petite est détruite puis reconstruite. La majorité du décor de bois sculpté est conservée.

2.3. Plans architecturaux

Pendant le stage, il nous a fallu récupérer les plans de la chapelle dessinés par David Ouellet, pour que Camille puisse esquisser l'ensemble du bâtiment, afin de former un modèle que nous appellerons « filaire » : une sorte de plan 3D, à but esthétique et architectural. Voici ces plans p. 20, pour donner une idée de l'architecture des lieux.

Le monastère possède de nombreuses ailes, une école, des logements, les bâtiments administratifs, les archives, etc. Ainsi évidemment qu'une chapelle, divisée en deux partie. D'un côté, la chapelle publique (à gauche sur le plan), et la chapelle privée réservée aux sœurs. Toutes deux sont reliées par le sanctuaire de marie de l'Incarnation, avec un tombeau, une statue de Marie de l'Incarnation, et des vitraux remarquables. Deux maitres autels se trouvent dans la chapelle privée, l'un faisant face aux sœurs dans leur chapelle, l'autre au public. Dans la chapelle privée, deux petits retables se trouvent de part et d'autre des arches menant à la chapelle publique. Ils sont dorés et très détaillés, et ont posés d'insolubles problèmes de modélisation. Enfin, il faut noter que les déambulatoires de la nef de la chapelle publique ont deux étages.

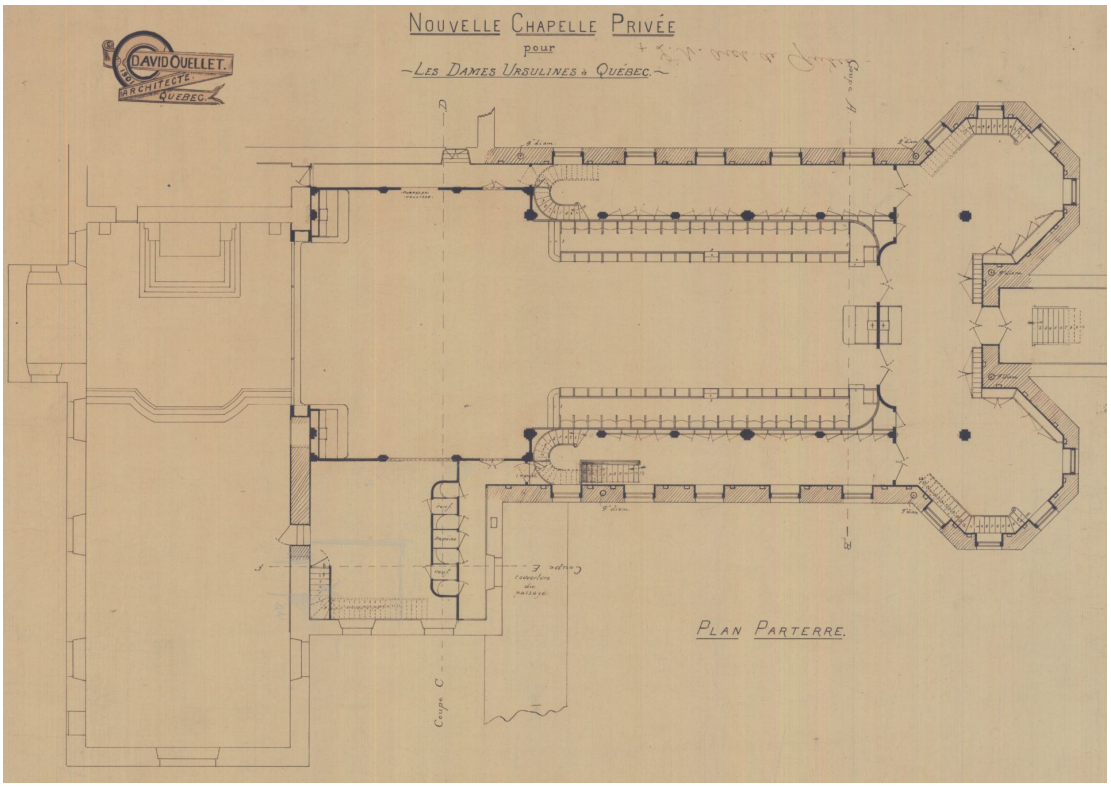


Fig. 2.1. : Le plan au sol de la chapelle

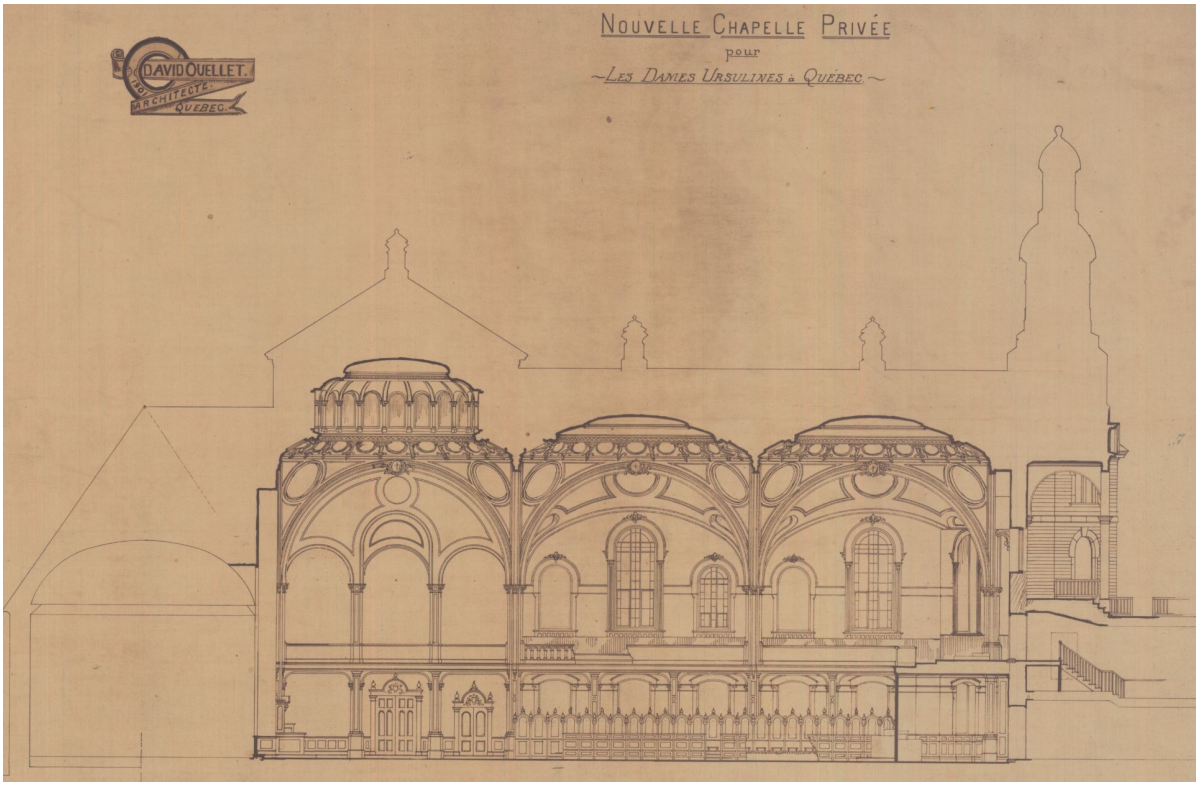


Fig. 2.2. : La coupe longitudinal de la chapelle

3. Historique du projet

> Nous sommes des nains assis sur des épaules de géants. Si nous voyons plus de choses et plus lointaines qu'eux, ce n'est pas à cause de la perspicacité de notre vue, ni de notre grandeur, c'est parce que nous sommes élevés par eux.
>
> *Metalogicon*

Jean de Salisbury

3.1. Avant Elsa

Puisqu'Elsa a travaillé quatre mois sur le modèle, il sera aussi simple de lui laisser la parole pour qu'elle nous explique qu'est-ce qui avait été réalisé avant elle :

> Certaines des étapes que je vais décrire à présent ont dû être reproduites lors de ce stage car je suis intervenue sur différentes phases du processus pour modifier le modèle.

3.1.1. La numérisation, juin 2010

Le scan 3D de la chapelle des Ursulines de Québec a été réalisé par Guy Côté et Claire Delisle, fondateurs de la société MCG3D basée à Québec.

Ils ont créé trois nuages de points de densité différentes afin d'assurer, au moins dans un des trois nuages, un bon compromis entre l'exploitabilité du modèle et son réalisme après maillage.

Les trois densités étaient de 10, 20 et 30 mm. Au final, seul le modèle le plus léger aura pu être exploité. Les ordinateurs disponibles n'étaient pas assez performants pour un fichier plus lourd.

3.1.2. Maillage

Le maillage a été effectué par une équipe du Laboratoire de Vision et Systèmes Numériques (LVSN), je citerai en particulier M. Thierry Moszkowicz car c'est lui qui m'a aidé à refaire le travail qu'ils avaient déjà accompli.

Le fichier contenant le nuage de points n'était pas exploitable sous sa forme initiale car beaucoup trop lourd : Polyworks , le logiciel utilisé pour le maillage, ne parvenait

pas à le charger. M. Moszkowicz a donc écrit un programme permettant de couper le modèle en tranches et ainsi de pouvoir ouvrir les fichiers et mailler les modèles indépendamment les uns des autres. Au final seize tranches se recouvrant légèrement (pour être certain de la continuité) ont été créées. Elles contiennent chacune en moyenne 250 000 polygones.

3.1.3. Transfert sur le panoscope

Le panoscope est contrôlé par le logiciel SPIN qui ne peut prendre en entrée pour des modèles 3D que des fichiers OSG ou IVE. L'exportation vers ces formats était impossible sur Polyworks . La solution utilisée par le LVSN était d'utiliser 3DStudioMax (disponible au LVSN et au LAMIC) comme convertisseur moyennant l'installation d'un plugin se téléchargeant librement sur internet. Le passage de Polyworks à 3DStudioMax se faisait via le format VRML. Le format d'exportation de 3DStudioMax vers le panoscope qui a été choisi est l'IVE. En effet le format OSG est plus lourd : le logiciel ne le supporte pas.

- Scan 3D : Données en ASCII
- Polyworks : Sauvegarde en .pol, export et import en .wrl
- 3DStudioMax® : Sauvegarde en .max, export en .obj, .osg ou .ive (conseillé)
- Panoscope : Logiciel spin

3.1.4. Exploitation sur le panoscope

Pendant l'été 2010 Louis-Robert Bouchard, polytechnicien au LAMIC a travaillé sur l'installation de SPIN, à sa prise en main et à l'augmentation du modèle. Il a importé toutes les tranches créées, inséré des images provenant d'archives et mis en place une spatialisation sonore (prières en provenance du chœur et musique de l'orgue). Il a également animé un livre de chant grégorien sur MaxMSP en utilisant des messages Open Sound Control (OSC) (la texture changeait selon un tempo donné).

3.2. Après Elsa

Voyons à présent la situation à notre arrivée, qui correspondait au travail d'Elsa sur les deux aspects de notre mission, le modèle à proprement parler et son exploitation en vue de visites virtuelles :

Partant du volumineux nuage de points capté par MCG3D, elle l'a découpé en différentes tranches pour qu'il soit plus facilement manipulable. Elle les a ensuite convertis en modèles polygonaux à l'aide du logiciel Polyworks puis a interpolé la couleur de chaque point sur les faces auxquelles ils appartenaient. Enfin a eu lieu une étape de simplification. Néanmoins, ce processus a rendu les couleurs floues et introduit du bruit. Ce modèle, jusque là utilisé, ne donnait pas entière satisfaction. De plus, une fois une des tranches importée dans le nouveau logiciel de visualisation Unity , d'autres problèmes sont apparus. Les surfaces sensées être planes étaient bruitées, ce qui dégradait l'éclairage. De plus, par défaut, Unity ne permet pas d'exploiter l'information de

couleur associée à chaque sommet et affichait donc une teinte uniforme, ce qui n'était pas acceptable. Enfin, cet affichage médiocre n'était pas très performant car chaque seizième de chapelle comportait environ 250000 faces (plus que l'ensemble du modèle complet que nous avons obtenu ultérieurement).

Après avoir obtenu le modèle décrit précédemment, Elsa l'avait exploité avec un système logiciel. Celui-ci souffrait néanmoins de deux problèmes. D'abord, alors qu'il fonctionnait à son départ, ce n'était plus le cas à notre arrivée. En outre, ce dysfonctionnement était difficile à diagnostiquer car malgré l'important effort de documentation réalisé, le système était un assemblage hétérogène de plusieurs logiciels et langages. Il s'appuyait sur une plateforme linux qui exécutait, outre un serveur audio et un serveur DHCP, un ensemble de composants développés par la SAT : SPIN. Cette bibliothèque libre (écrite en C++) encapsule l'utilisation d'OpensceneGraph pour la visualisation 3D, OpenSoundControl pour la communication réseau et la téléprésence, ainsi que python pour le scripting et les interactions. Le tout était piloté par une interface en pure data à l'aide de pdsheefa. Autant dire que les messages d'erreur étaient pour le moins perdus dans ces diverses couches et donc assez ardus à exploiter. Pour assombrir encore le tableau, tous ces logiciels étaient restés dans leur version de l'été 2011 et mes essais de mis à jour n'ont fait que rajouter des incompatibilités.

Deuxième partie .

Modélisation

4. La chapelle intérieure

4.1. Retopologie

4.1.1. Modèle existant

Comme évoqué dans l'historique du projet, le modèle existant présentait des défauts visuels rédhibitoires, comme on peut le voir sur cette capture d'écran :

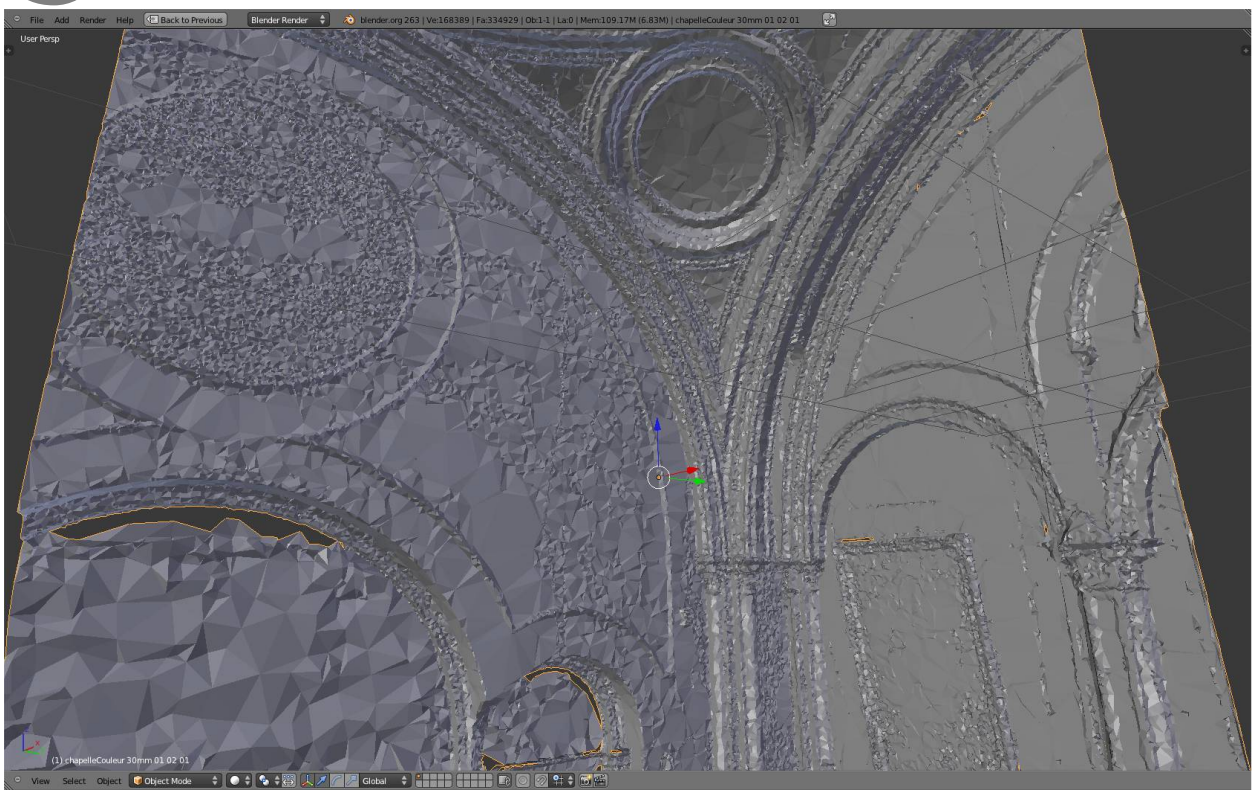


Fig. 4.1. : Une des tranches originelles, sur laquelle on l'on peut apprécier toute l'étendu du bruit.

Suivant les recommandations de MM. Dubé et Vesac, nous avons donc décidé de repartir sur de nouvelles bases (en espérant que d'éventuels futurs stagiaires n'aient pas à emprunter une troisième voie l'année prochaine…). En l'occurrence, il s'agissait de reconstruire le modèle à partir du scan initial.

3D-Coat

3D-Coat permet de charger ce modèle polygonal trop détaillé afin de redessiner une surface qui s'appuie dessus. C'est un processus manuel, donc long et fastidieux, mais il permet de choisir le niveau de détail de chaque zone ce qui donne un résultat plus léger et mieux adapté. Malgré

une machine performante, 3D-Coat n'a pas supporté de charger l'ensemble de la chapelle et il a donc fallu effectuer la retopologie tranche par tranche . C'est d'autant plus frustrant que nous avons spontanément commencé par les tranches du cœur avant de constater, arrivés à la fin, qu'elles étaient les moins réussies, ce qui a forcé Camille à les refaire. Cette manière de procéder a aussi nécessité de recoller les différents morceaux entre eux, ce qui n'était pas trivial car ils n'avaient pas forcément la même géométrie aux extrémités (par exemple un mur pouvait être constitué de cinq bandes d'un côté et seulement trois de l'autre). J'ai pour cela utilisé le logiciel Blender (décrit par la suite, qui a aussi servi par la suite pour le plaquage de textures et les animations.

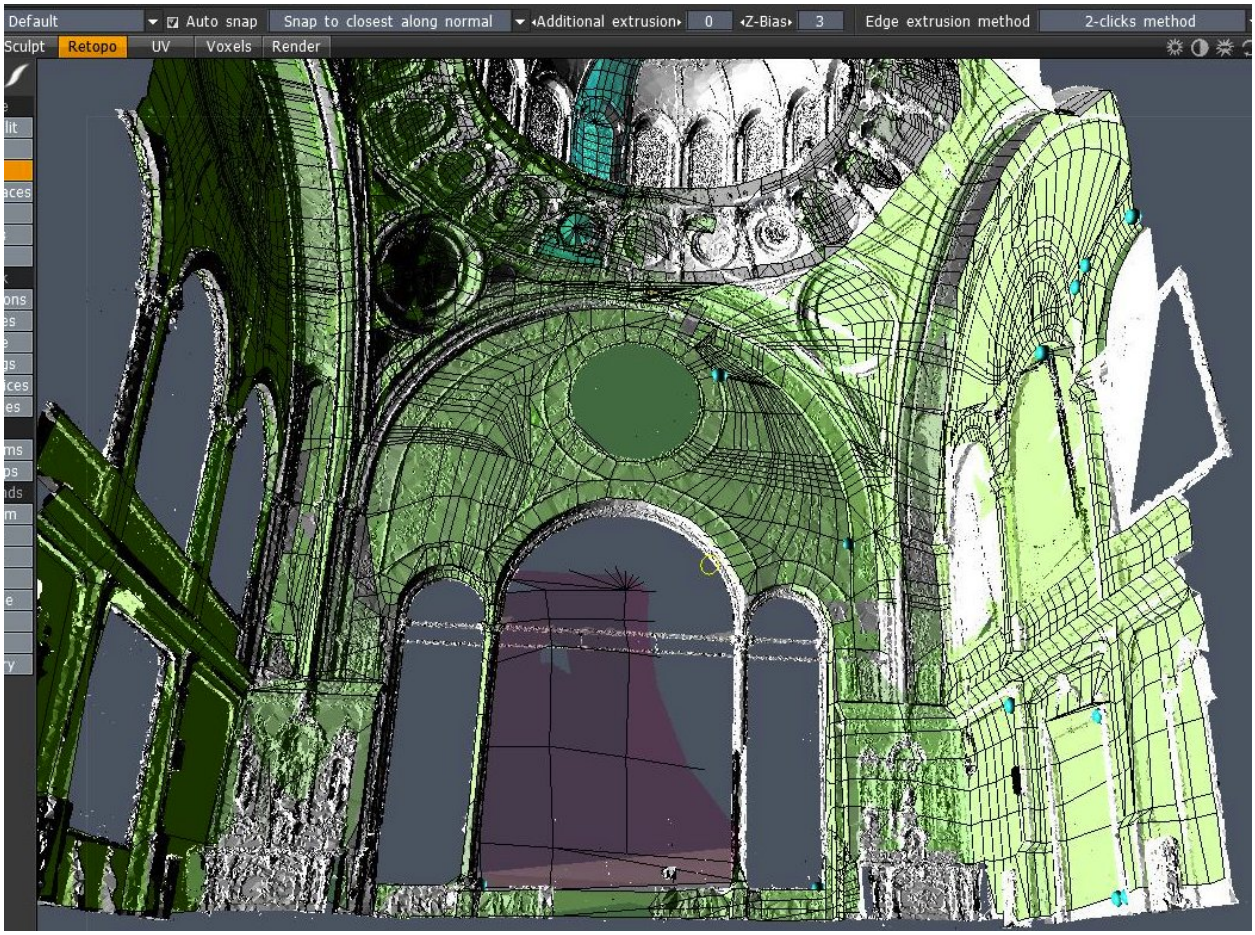


FIG. 4.2. : La retopologie du chœur en cours sous 3D Coat

Blender

D'après Wikipedia :

> « Blender est un logiciel libre d'animation, de modélisation et de rendu 3D. Il dispose de fonctions avancées de modélisation, de sculpture 3D, de dépliage UV, de texturage, de (gréage), d'animation 3D, et de rendu. Il gère aussi l'édition non linéaire, les compositions, la création nodale de matériaux, la création d'applications 3D interactives, ainsi que diverses simulations physiques telles que les particules, les corps rigides, les corps souples et les fluides. Traduit dans de nombreuses langues, Blender est également disponible sur plusieurs plates-formes telles que Microsoft Windows, Mac OS X, GNU/Linux, IRIX, Solaris, FreeBSD, SkyOS, MorphOS et Pocket PC. De plus, c'est un programme extensible (ajout de scripts) à l'aide du langage Python. »

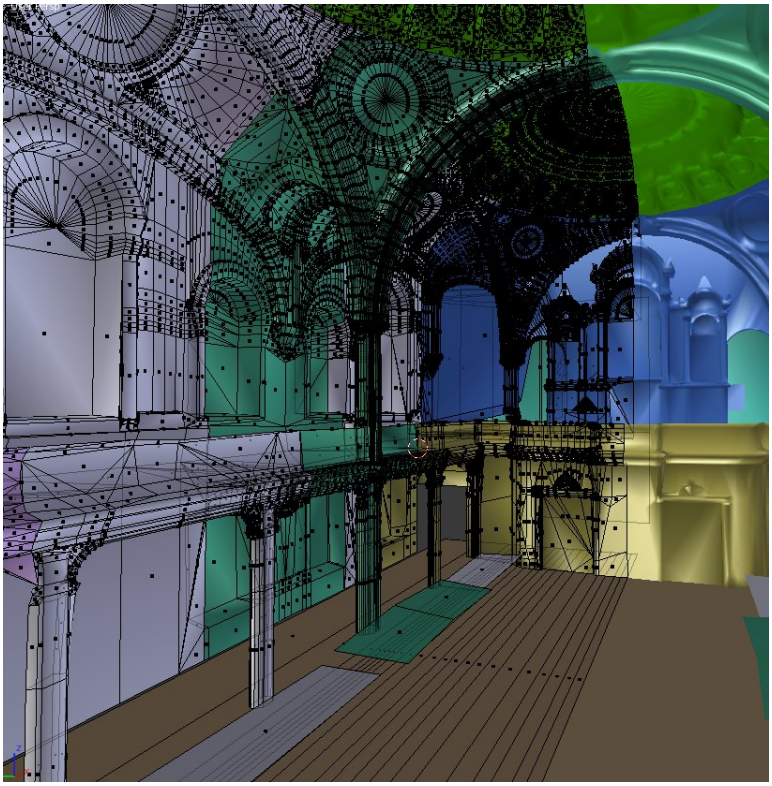


FIG. 4.3. : la retopologie de la nef, une fois recollée, sous Blender

Blender présente l'avantage d'un grand nombre de fonctionnalités, qui évoluent très vite (grâce à son passage en Open Source). « Les amateurs de Blender peuvent ainsi suivre son évolution à partir de nombreuses compilations personnalisées. » Malgré une amélioration de l'ergonomie après la version 2.5, Blender demeure assez peu intuitif et son apprentissage reste long. En conséquence, et comme il n'en existait pas encore pour cette version 2.63, Camille a créé un fichier de recensement des fonctionnalités clavier les plus employées (voir Fig. C.1 p. 90).

Ce logiciel nous a servi tout au long de la création du modèle de la chapelle afin de :

- Recoller les tranches (voir Fig. 4.4 p. 28)
- Générer les coupoles par « spin » (voir Fig. 4.5 p. 28)
- Générer par miroir l'ensemble de la chapelle
- Rectifier et aplanir certaines zones spécifiques

4.2. Texturation

Comme expliqué précédemment, le modèle présentait des couleurs floues, dues à l'interpolation de la couleur de chaque point sur les triangles auxquels ils appartenaient. Il a donc été jugé plus juste que le rendu des couleurs soit basé sur des photographies, travail qu'Elsa avait entamé l'an dernier, sur le précédent modèle, à partir d'autres logiciels, mais qui était hélas peu exploitable.

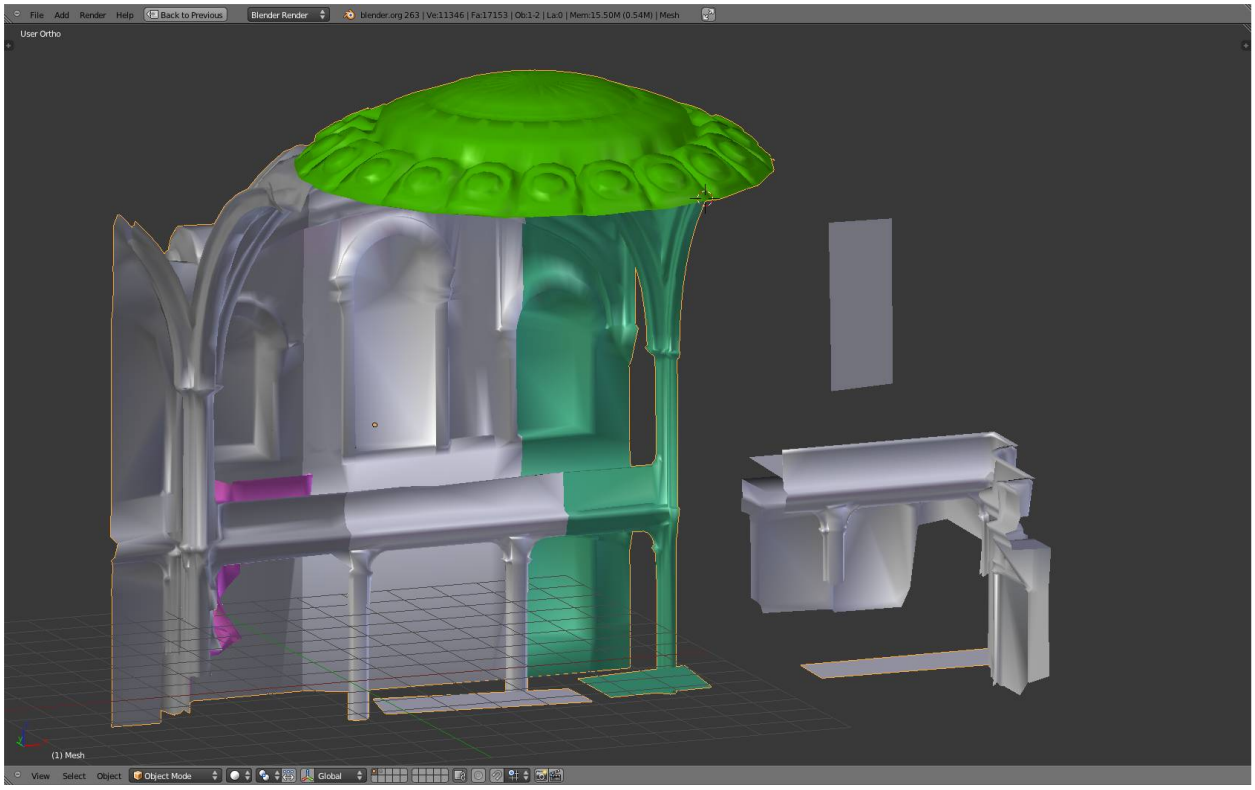


FIG. 4.4. : Une nouvelle tranche doit être insérée

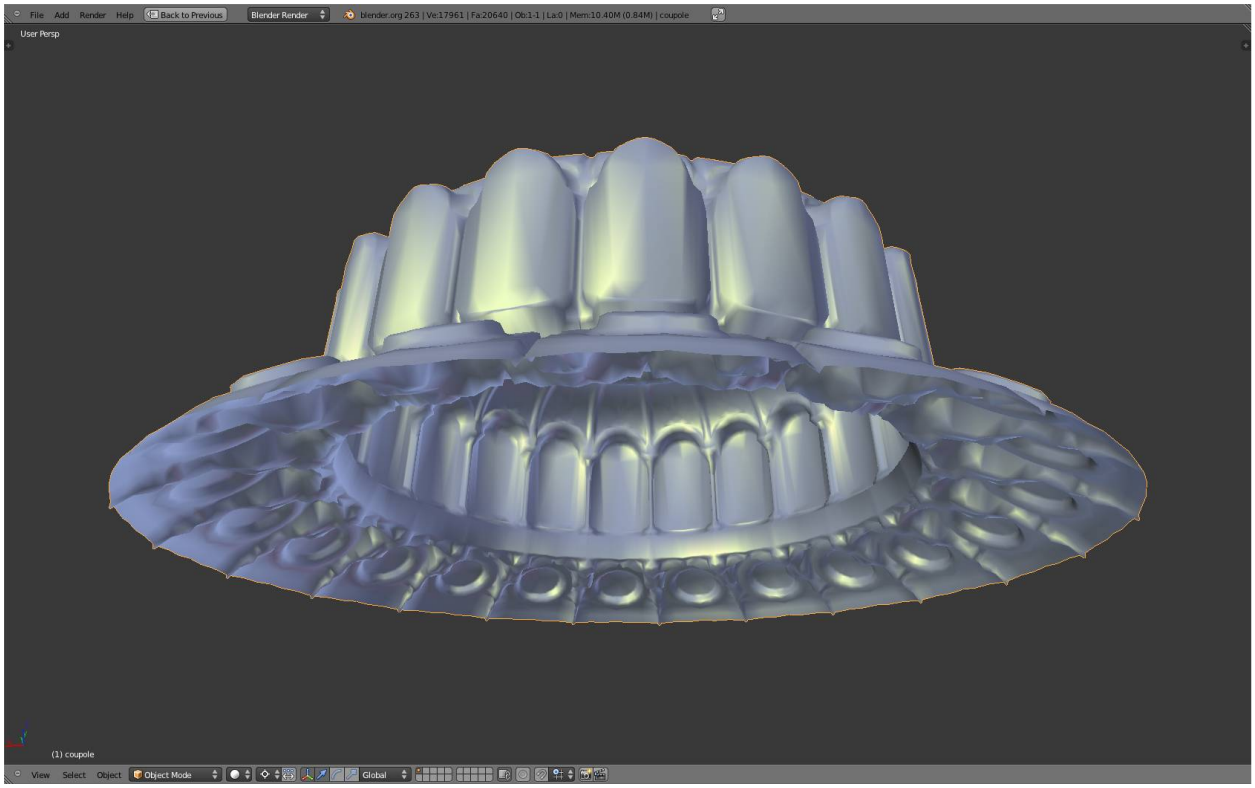


FIG. 4.5. : Seule une des vingt tranches de la coupole a été modélisée sous 3D-coat, avant de réaliser une révolution en s'assurant que les morceaux se recollaient harmonieusement.

4.2.1. Nouvelle texturation : UV mapping

Nous avons choisi de texturer notre chapelle sur Blender par UV-mapping. C'est en effet la méthode la plus simple pour obtenir un bon rendu dans Unity et s'il était aussi possible de le faire dans 3D-Coat (de manière plus intuitive semble-t-il), le résultat était plus flou. Cela consiste à déplier le modèle 3D afin de faire correspondre les coordonnées *(x,y,z)* de chaque sommet avec un point *(u,v)*[1] d'une image qui fournit l'information de couleur (dans notre cas, il s'agissait d'images de l'intérieur de la chapelle prises par Luc-Antoine Couturier). Camille a choisi de projeter les vues qui l'intéressaient de chaque morceau à colorer sur des plans de vue les plus équivalents possibles aux photos, pour faire correspondre ces projections aux prises de vue. Néanmoins, certaines zones ne pouvaient pas se faire aussi simplement et il a fallu rectifier un certain nombre de parties à la main (c'est-à-dire polygones par polygones, en les déformant sur l'image pour qu'ils correspondent au mieux à la réalité).

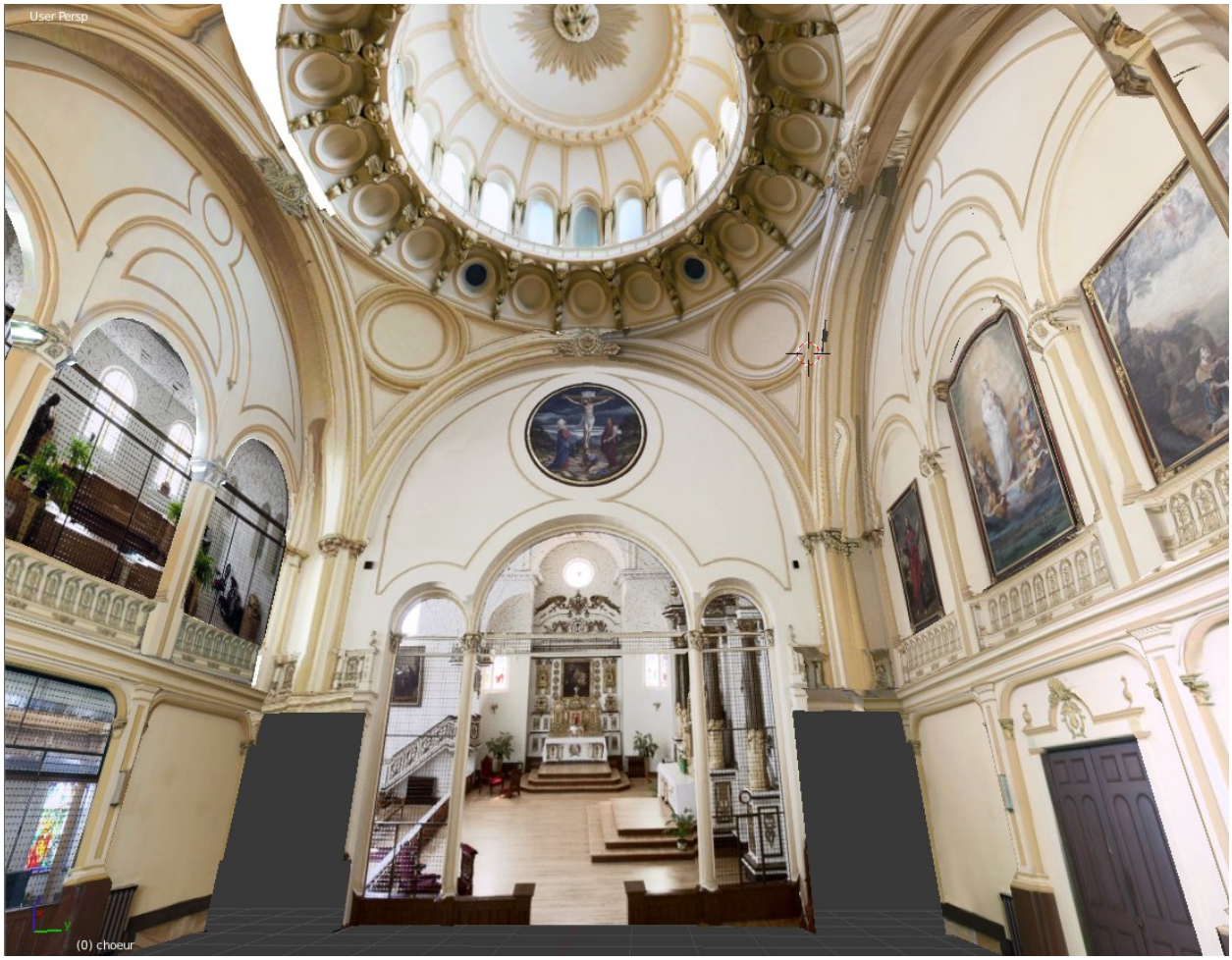


Fig. 4.6. : Le chœur une fois texturé. Les zones grises correspondant aux autels encore manquants.

C'est d'ailleurs à ce stade que nous avons constaté qu'il nous manquait des photos de la nef, ce qui nous a amené à réaliser une autre campagne dans la chapelle. Cela nous a aussi permis de scanner la façade extérieure, la chapelle publique et le sanctuaire afin d'intégrer notre travail à un modèle 3D de la ville et de contextualiser l'arrivée du visiteur dans la chapelle privée. Cette campagne a eu lieu le 8 juin 2012 avec l'équipe de MCG3D et Luc-Antoine Couturier.

1. d'où le nom d'UV mapping de cette technique.

4.2.2. Prise de photos avec Luc-Antoine Couturier

FIG. 4.7. : Luc-Antoine Couturier prenant des photos de la façade de la chapelle

Il s'agissait, comme Elsa l'avait requis pour le chœur l'an dernier, de prendre des photos avec le moins de déformation possible, le plus « en face » possible des zones à imprimer, pour que le plaquage se fasse le plus naturellement possible. Nous avons néanmoins rencontré quelques problèmes. D'abord, le Génie n'a pas tout de suite fonctionné et nous n'avons pas pu l'utiliser pour la nef, il a donc fallu travailler avec un escabeau, ce qui n'a pas facilité le plaquage. Ensuite, pour le fond des murs des déambulatoires, nous manquions de recul pour les prendre. Il a donc fallu recréer artificiellement un mur sur Photoshop en y appliquant des fenêtres et les médaillons du chemin de croix (voir Fig. 4.8b p. 30).

(a) partie supérieure de la nef (b) mur des déambulatoires

FIG. 4.8. : Deux exemple de textures, l'une composée de différentes photos regroupées dans une image carrée de 2^{12} pixels de côté, l'autre dessinée artificiellement.

4.2.3. Problèmes et avancées

Il existe encore quelques problèmes au niveau des raccords des photos, malgré le temps passé à estomper les différences de couleurs sur Photoshop. Camille a tout de même tenté de faire ces raccords aux endroits où ils paraissaient le plus discrets, de sorte que cela est loin de sauter aux yeux.

5. Mobilier

C'est au moment de remailler avec 3D Coat la chapelle que s'est posé le problème du mobilier. En effet, le scan de 2010 ayant eu pour but de capter le bâtiment, sa résolution n'était pas adaptée à la taille de ces objets, et les traitements ultérieurs n'ont fait qu'empirer les choses.

5.1. Méthode employée pour le maillage

En ce qui concerne les bancs, sièges, et lampes, nous avons décidé de les modéliser directement dans Blender , en nous appuyant sur le nuage de points pour les dimensions et sur des photos pour les détails. Sachant que ces objets devant par la suite être dupliqués en dizaine d'exemplaires, à l'aide du script `Peuple.cs`, ils ne devaient donc pas comporter trop de sommets, ce qui explique l'importance de la texturation, qui permet de remplacer en partie la géométrie par la couleur.

5.2. La texturation

Nous avons également texturé par UV-mapping. Certains bancs possédaient une structure plutôt complexe, due à des barreaux sur le dossier. Plutôt que de complexifier la structure, Camille a détouré les bancs et les a mis sur un fond transparent pour le mapping. Blender et Unity prennent en compte cette transparence et nous avons bien un effet de transparence. Cette méthode a également été utilisée pour les ornements des sièges le long de la nef.

Un problème subsistait pourtant, bien qu'il ne soit pas flagrant : les bancs en question apparaissent légèrement translucides dans Unity. Il s'agissait en fait d'un problème de paramétrage sous Unity . Si le matériau était en mode *Transparent/diffuse*, la transparence s'affichait plus que nécessaire. Le matériau *Transparent/cut out/diffuse* impose un « seuil de transparence » qui correspond ici mieux à nos besoins.

6. Les autels

Nous avons éprouvé nettement plus de difficulté à choisir un mode opératoire pour les autels[1] que pour le reste du mobilier car deux écoles s'affrontaient. Nous avions le choix entre scanner avec une meilleure précision ou utiliser la photogrammétrie.

6.1. Nouveau scan

Un nouveau scan, focalisé sur les autels, a été effectué par MCG3D. Les scans sont tellement détaillés que l'œil humain n'a aucun mal à en extraire une représentation surfacique[2]. Mais ce n'est manifestement pas le cas de l'ordinateur, d'abord à cause du bruit de mesure, qui perturbe les surfaces planes[3]. Ensuite parce qu'à cause des occlusions, il faut scanner les objets sous différents angles et que la réunion de ces données multiples introduit de nouvelles ambiguïtés. Ceci est particulièrement flagrant sur l'autel, qui possède des formes organiques assez fines.

Nous avons utilisé deux logiciels pour essayer d'obtenir des polygones à partir d'un nuage de points, même s'il en existe bien d'autres, comme Geomagic, présenté de façon impressionnante lors d'une réunion chez Creaform.

- Polyworks : utilisé par MCG3D et dont le LAMIC possède la version 2010. C'est un logiciel complexe destiné à l'industrie. À force d'essais, nous avons abouti à un modèle probant de 1,7 millions de faces, qui possédait malheureusement l'agaçant défaut de ne pas avoir toutes ses faces orientées du même sens, ce qui était difficile à corriger, tant manuellement qu'automatiquement.

- Meshlab : initialement développé au sein de l'université de Pise par des étudiants en informatique, il implémente de nombreuses techniques concernant les maillages parues dans des articles universitaires. On peut ainsi calculer les normales des points[4] à partir de ses voisins avant d'effectuer une reconstruction de Poisson et de transférer les couleurs des points à la face dont ils sont les plus proches. Là encore, les résultats ne sont pas ridicules mais jurent avec la modélisation soignée du chœur de la chapelle.

1. Qui présentaient le même problème que les meubles, d'autant que leur structure est particulièrement complexe (statues, plis des robes, etc)

2. À titre d'exemple, nous avons mené nos premiers essais sur l'autel droit issu du scan de 2010 qui compte 130 000 points, avant de récupérer une version fusionnée du scan de 2012 de près de 900 000 points.

3. problème que l'on peut atténuer en utilisant un filtre médian, au prix de la perte de détails.

4. qui correspond grosso modo à la moyenne des normales des faces contenant ce point dans le polygone convexe enveloppant le nuage.

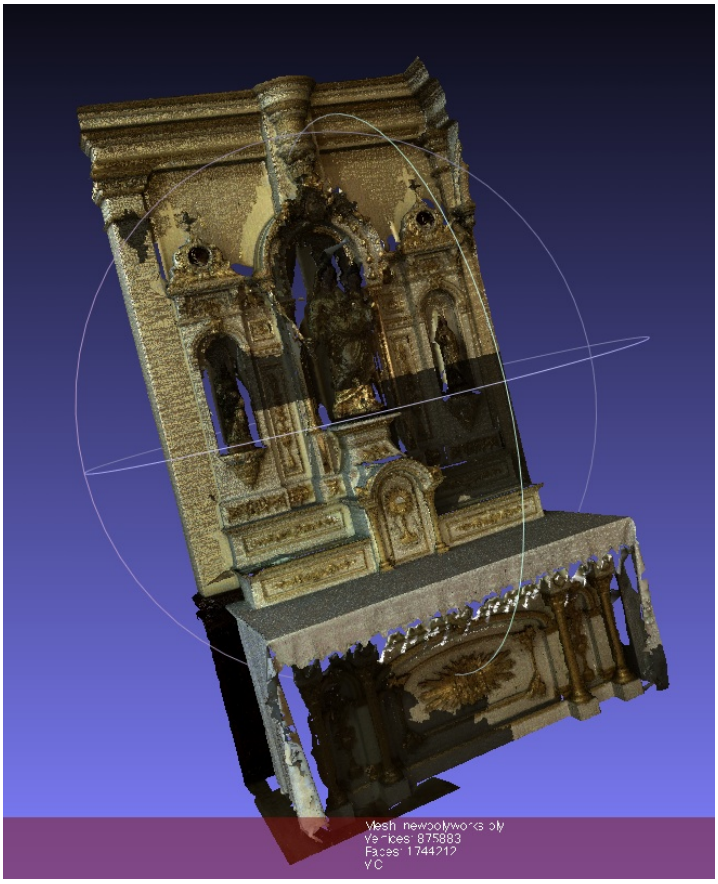


FIG. 6.1. : Outre les faces du mauvais sens, on remarque aussi que les couleurs sont comme bruitées, sans doute à cause de problème d'éclairage et d'ombre.

6.2. Photogrammétrie

> La photographie a ouvert des horizons illimités à la pathologie du progrès, puisqu'elle nous a incités à déléguer à la multitude de nos machines de vision le pouvoir exorbitant de regarder le monde, de le représenter, de le contrôler.
>
> Paul Virilio, urbaniste nantais

6.2.1. Introduction

De nos jours, une grande partie de la surface de la Terre est accessible sous forme de photographies numériques, que ce soit grâce au déploiement de satellites ou la généralisation des appareils photos numériques et des smartphones, voire aux efforts de certaines sociétés commerciales. En revanche, malgré leur indéniable perfectionnement technologique, les scanneurs 3D restent des appareils onéreux et généralement peu maniables, ce qui les confine pour l'instant au domaine professionnel, où ils sont notamment utiles en métrologie. Partant de ce constat, et tenant compte de l'attractivité de la troisième dimension pour le grand public, il parait logique d'essayer de reconstruire un objet en 3D à partir de photos pris sous différents angles, ce qui est le but de la photogrammétrie. Afin d'appliquer cette méthode, Géraud a testé plusieurs solutions qui se basent toute sur un principe similaire.

6.2.2. Principe

Dans un premier temps, il faut relier les photos entre elles. Cela ne peut se faire que si elles respectent plusieurs contraintes[5]. : le fond de l'image doit rester constant au cours des prises de vue – qui doivent préférablement s'effectuer avec le même appareil et les mêmes réglages. Il doit – ainsi que l'objet – présenter des textures variées et pour les photos se recoupent le mieux possible, il faut éviter qu'elles soient séparées de plus de dix degrés[6] . Dans ces conditions, on peut en extraire des points de similarité grâce à la méthode SIFT. Cela peut être réalisé avec un logiciel développé par David Lowe. Cette méthode est aussi intégrée au logiciel Bundler, qui l'effectue après un pré traitement des images[7], et on peut supposer qu'elle est aussi utilisé par Photosynth[8] dont il sera question plus loin. À noter qu'il existe des alternatives à SIFT, notamment la méthode SURF, qui se présente comme plus robuste et dont il existe de nombreuses implémentations open source[3].

Une fois que l'on dispose de ces points de correspondance, on peut déterminer l'orientation relative des photos grâce aux principes de stéréovision. On peut ensuite trianguler la position des points de correspondance à partir de la position des différentes caméras, dont on connaît idéalement la distance focale. Voir le chapitre 7 du livre de Richard Szeliski[5] pour plus de détails.

5. dont un des sites donne une liste imagée

6. ou formulé autrement, deux photos consécutives doivent présenter au moins 80% de contenu en commun.

7. détermination de la focale en fonction des données EXIF, redimensionnement, …

8. puisque Bundler a été développé dans le cadre du projet Photo tourism mené par Microsoft qui vise à exploiter les nombreuses des monuments célèbres disponibles sur internet pour en donner une représentation 3D.

6.2.3. Solutions testées

- MICMAC : développé par l'IGN pour traiter des photos satellites, cela semble une solution robuste mais plus adaptée à des dimensions larges et surtout très difficile d'accès car comme le dit l'article de présentation[2] : « *it is probably more complex but also more complete, its targeted user is rather professionals (architects, archaeologist, geomophologist) than people* ». Effectivement, sorti des exemples, Géraud n'a pas eu le courage de modifier les cinq cents lignes du fichier de configuration en XML.
- 3DSOM Pro : logiciel de la société Creative Dimension Software Ltd qui donnait aussi de très bons résultats sur les photos fournis à titre de démonstration[9]. Malheureusement, la version d'essai était limitée à 14 jours, ne permettait pas d'exporter les modèles obtenus et coûtait 1349$ pour être débloquée.
- CVMS et PVMS2 : ensemble de logiciels[10] qui exploitent la sortie de Bundler. CVMS sert à préparer les données pour un traitement par lot sur un cluster tandis que PVMS2 reconstitue effectivement un nuage de points dense. La configuration n'est pas trop compliquée mais les résultats n'étaient pas très complets et sollicitaient pourtant lourdement la machine pour être obtenus.
- VisualSFM : il s'agit d'une interface graphique aux deux logiciels précédents qui présente donc les mêmes défauts et les mêmes qualités.
- PhotoSynth Toolkit : Solution développée par un ingénieur français sur son temps libre et qui consiste à utiliser le service PhotoSynth de Microsoft pour extraire les relations entre les images et les fournir en entrée à PVMS. Là encore, les résultats semblaient prometteurs mais le logiciel ne s'exécutait que sur Windows® 64 bit que seul mon ordinateur portable personnel possédait. Cette machine n'est pas malheureusement pas assez puissante pour traiter plus d'une vingtaine de photos en un temps raisonnable[11]

Restent les solutions en ligne, qui ne présentent pas cet inconvénient :

- My 3D scanner : service gratuit qui récupère au plus 200 Mo de photos d'un objet[12] et en propose — après quelques heures d'un traitement qui se base notamment sur les incontournables Bundler et PVMS — une version en nuage de points et un modèle polygonal sur lequel on peut transférer la couleur des points individuels.
- ARC 3D : Un service de même nature est proposé par l'université de Louvain. Il se distingue par son intégration poussée avec Meshlab[6] qui permet de régler différents paramètres de la reconstruction et par le fait que le modèle polygonal est *UV-mappé* sur les photos d'entrée, même si cela reste impossible à manipuler à la main.

Notons enfin qu'il existe de nombreuses autres solutions référencées dans ce forum ou plus simplement sur wikipédia.

9. qui avaient manifestement été obtenues dans de très bonnes conditions
10. des binaires pour windows sont disponibles sur ce site
11. c'est-à-dire qui ne se compte pas en dizaines d'heures d'écran noir et de processeur chauffant comme les forges de l'enfer.
12. voire une vidéo, ce qui permet d'exploiter les techniques de « structures from motion »

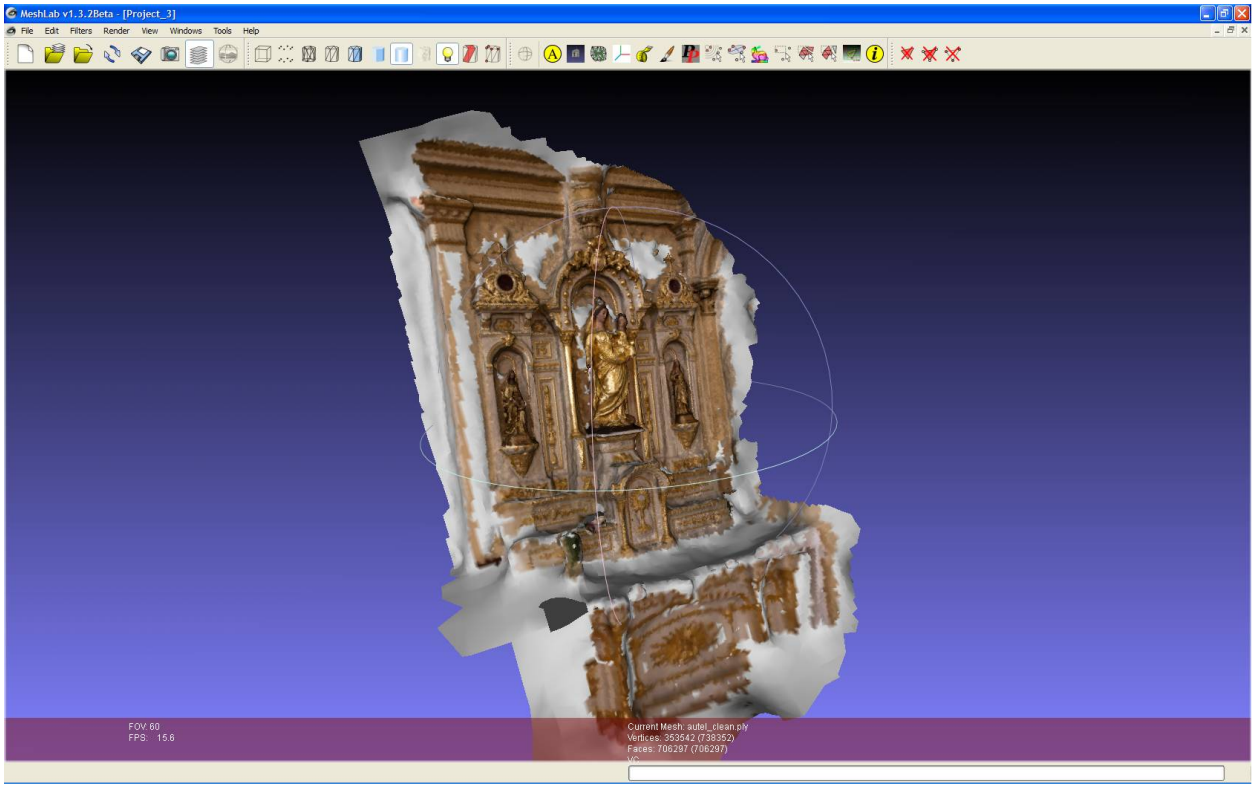


FIG. 6.2. : Après quelques heure d'attente, voilà le résultat fournit par my3Dscanner. La forme est plutôt bien respectée mais les couleurs ne sont pas complètes à cause des occlusions sur les photos, qui ne sont pas en nombre suffisant.

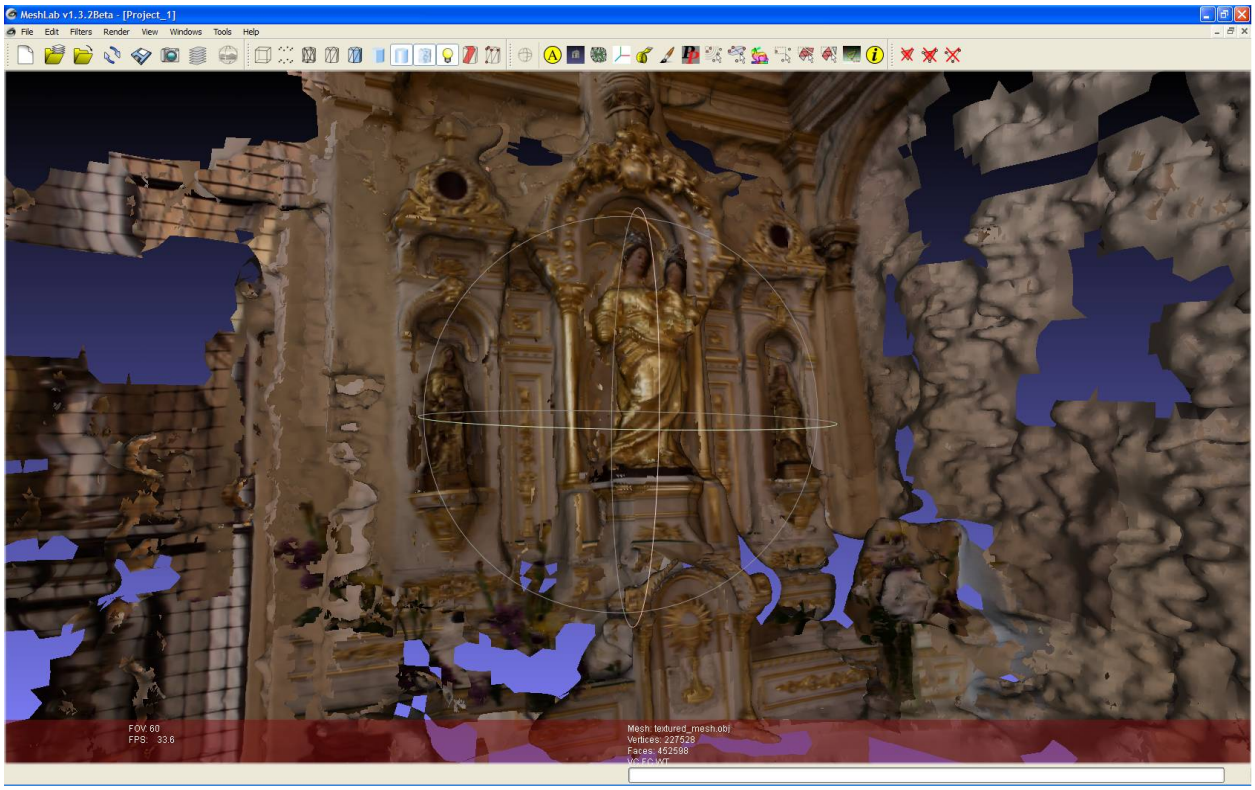


FIG. 6.3. : Cette fois, les couleurs sont satisfaisantes mais il reste des trous et le mur de côté est étrangement déformé.

6.2.4. Récapitulatif

Système	Type	Licence et coût	Documentation	Résultat
MICMAC	logiciel	GPL, gratuit	succincte	N/A
3DSOM	logiciel	propriétaire, 1349$	N/A	N/A
PVMS2	logiciel	GPL, gratuit	suffisante	très lourd
VisualSFM	logiciel	GPL, gratuit	suffisante	lourd
PhotoSynth Toolkit	logiciel	mixte, gratuit	suffisante	lourd
My 3D scanner	web	propriétaire, gratuit	imagée	bon
ARC 3D	web	propriétaire, gratuit	fournie	bon

Tab. 6.1. : Récapitulatif des solutions testées en photogrammétrie.

7. Façade, sanctuaire et chapelle publique

Une nouvelle campagne de scan fut effectuée le 8 juin 2012 par MCG3D pour capturer, en plus des deux autels latéraux de la chapelle intérieure, la chapelle publique, sa façade, et le sanctuaire de Marie-de-l'Incarnation.

Il a donc fallu choisir ce que nous ferions de ces données, sachant qu'elles ne nous serviraient qu'à contextualiser la chapelle, à situer le visiteur.

7.1. Nuage de points

La matière première de notre modélisation a consisté en des nuage de points. Il existe de nombreuses méthodes pour scanner un objet en 3D[1]. Ainsi, MCG3D utilise un produit conçu par Arius3D, qui est un scanner sans contact actif par triangulation laser, fournissant en sortie trois coordonnées spatiales par points associées à une couleur. Ces techniques ont toutefois en commun de ne fournir qu'un échantillon des sommets qui constitue l'objet, sans pouvoir capturer d'information sur la surface continue qui les relie. Or la solution la plus courante (et donc devenue la plus riche et la plus performante au fil du temps) pour afficher des objets en synthèse d'image se base sur des modèles polygonaux[2], ce qui nécessite un post-traitement qui s'est avéré particulièrement ardu et chronophage.

7.2. Affichage tel quel

Les fichiers fournis par MCG3D étaient dans un format particulièrement simple : chaque ligne contient les trois coordonnées d'un point et son intensité suivant les trois composantes rouge, verte et bleue. Une idée tout aussi simple est alors de lire ce fichier ligne à ligne et d'ajouter à chaque fois la particule correspondante à un système de particules disponible dans Unity. Malheureusement, si le résultat est globalement celui auquel on s'attend, les performances sont déplorables (8 images par seconde pour 350 000 points alors que la chapelle intérieure à la résolution de 2mm en contient plus de quatre millions) et l'affichage intermittent. Une des solutions est de ne conserver qu'un point sur n, car même si cette approche paraît un peu brutale, un exemple sur Internet semblait prometteur. Peut-être serait-il plus tolérable d'afficher un nombre de points basés sur la distance de l'observateur à ces derniers, approche *Level Of Detail* mise en application avec succès à l'université de Vienne. On pourrait aussi essayer de découper le nuage entier en sous cubes pour n'afficher que ceux qui sont dans le champ de vision de l'utilisateur[3]. Dans tous les cas, cela aurait nécessité un long travail assez spécifique, d'autant que

1. Camille a d'ailleurs rédigé un état de l'art au cours du stage.

2. même si, dans une perspective artistique, on peut essayer de travailler l'esthétique du nuage de points.

3. Ou alors une fonctionnalité de la version professionnelle d'Unity qui permet de réaliser directement des appels à OpenGL, en espérant que la géométrie sera bien stockée sur la carte graphique.

plusieurs interlocuteurs nous fait part de leurs doutes à propos de l'adéquation d'Unity à ces tâches.

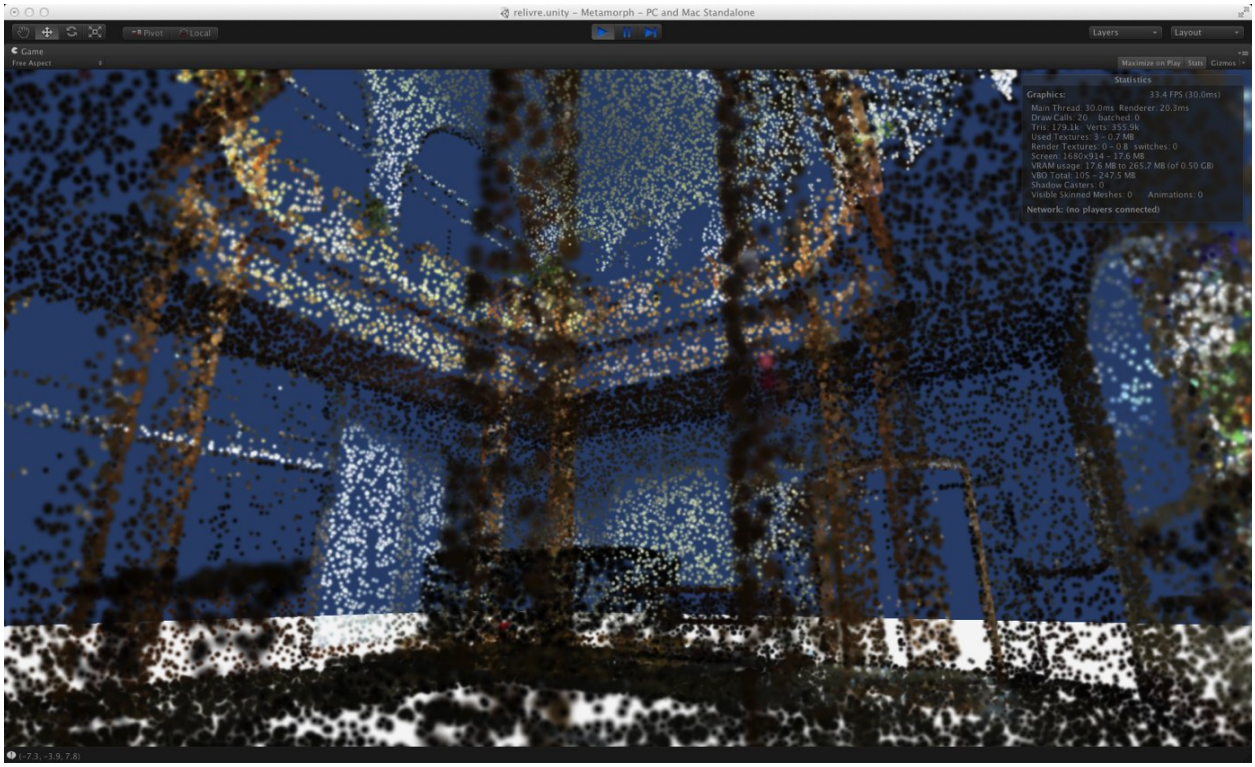


Fig. 7.1. : Il s'agit là d'une version avec encore moins de points (88 000) de sorte que les performances sont acceptables (environ 24 images par seconde) mais on commence à avoir du mal à distinguer les volumes.

7.3. Triangulation

Des méthodes semblables à celles envisagées pour les autels (avec Meshlab ou Polyworks) sont assez gourmandes en ressource. Il nous a paru plus prudent de procéder autrement pour les nuages correspondants à l'extérieur (2,3 millions de points), à l'intérieur de la chapelle publique (4,5M) et au sanctuaire (2,8M), d'autant que ce sont des zones qui ne sont pas prioritaires[4]. En l'occurrence, en utilisant Blender , on crée une esquisse plane qui reprend les grands traits de la structure considérée[5], on la subdivise en utilisant l'algorithme de Catmull-Clark pour affiner sa précision, on projette chacun de ses sommets sur le sommet le plus proche dans le nuage et on effectue un filtre médian pour adoucir le résultat et enlever les points aberrants. Le résultat peut évidemment s'avérer assez éloigné de la réalité, mais suffit à en fournir une illusion satisfaisante et la surface obtenue est d'une grande régularité.

4. même s'il serait aussi tentant de les afficher directement en points.

5. puisque selon Thierry Moszkowicz, la détection d'arêtes en 3D est un problème délicat, alors même qu'il est bien maitrisé en 2D et qu'on possède des informations de couleur.

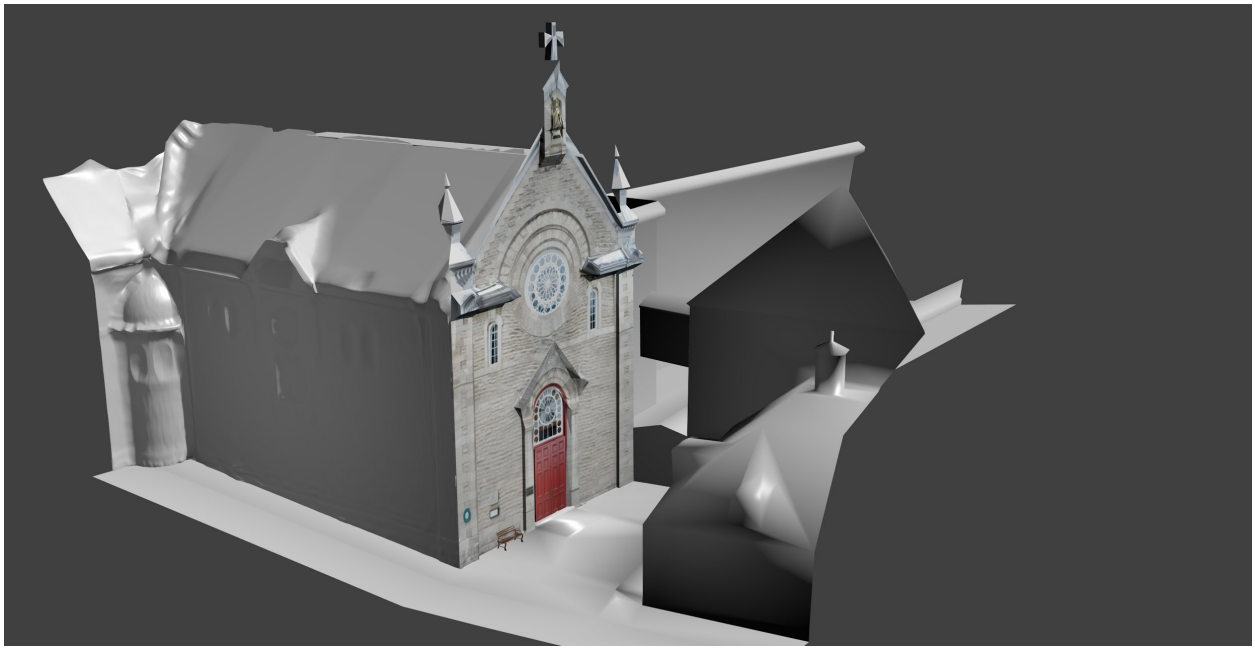


FIG. 7.2. : Les murs de la chapelle sont plus détaillés que les autres bâtiments ce qui permet de hiérarchiser l'information. Le réalisme est augmenté par le plaquage d'une texture sur la façade.

8. Modèle filaire

La chapelle a été reconstruite à plusieurs reprises au cours de son histoire. En résulte une architecture riche en passé qu'il semble pertinent de commenter. Pour cela, il a été décidé, d'après une suggestion de M. Vesac, de créer un second modèle, filaire, formant une sorte de plan en trois dimensions, en n'indiquant que les arrêtes, poteaux, et contours d'éléments architecturaux marquants. Il fallait pour cela s'appuyer sur des plans de la chapelle, qui ont été demandés aux archives du monastère.

8.1. Les plans

Au sol, Camille s'est appuyée sur un plan récent (voir Fig. 2.1 p. 20), pour être sûre que les récentes modifications du sanctuaire soient prises en compte et que le modèle filaire corresponde au modèle réaliste. Pour les élévations, seuls étaient disponibles les plans de 1902 (voir Fig. 2.2 p. 20) de David Ouellet, grand architecte de la ville de Québec qui a principalement consacré sa carrière à des édifices religieux. Malgré leur âge et mis à part le sanctuaire , ces plans sont toujours à jour. Le sanctuaire possédant une structure assez simple, il fut aisé de le reconstruire de mémoire en considérant que la hauteur de l'étage, elle, n'avait pas changé.

D'autre part, pour mieux illustrer les fils, qui pourraient sembler quelque peu abstrait à quelqu'un de complètement étranger à la chapelle, Camille a choisi de plaquer les dessins de David Ouellet sur des plans extérieurs à la chapelle, en arrière du modèle. En plus d'orienter le visiteur, ils possèdent des qualités esthétiques non négligeables. Au sol, c'est également un plan de 1902[1] actuel qui guidera nos pas. Il a ensuite été décidé d'étendre ce modèle à la chapelle extérieure dont il a fallu acheter les élévations[2].

8.2. Choix du logiciel

Au bout d'une semaine de courriels et de détails bureaucratiques, Camille a réussi à télécharger la version étudiante du logiciel AutoCAD2013. Ce logiciel d'architecture semblait parfaitement adapté aux besoins. Pourtant, le logiciel s'est révélé particulièrement difficile à utiliser. L'architecture de la chapelle, composée de voutes et de courbes non instinctives, ne semblait pas être la plus simple à dessiner via un logiciel d'architecture moderne. L'aspect filaire que nous désirions pouvoir moduler à notre goût ne semblait pas si simple à obtenir et surtout, à exporter. En effet, en traçant ses traits et ses arches (non sans difficultés car cela ne semblait pas être un mouvement naturel via un tel logiciel), Camille a réalisé qu'il était impossible d'exporter ces arrêtes. Il fallait les transformer en volume pour les obtenir en un format lisible par Unity (.obj, idéalement). Or, transformer des arrêtes en tubes n'est pas une opération classique pour de la

1. car il diverge suffisamment peu de la répartition spatiale.

2. Celles-ci ayant été transmises en blanc sur bleu alors que les plans précédents étaient en noirs sur beige, il a également fallu modifier ces teintes sur Adobe Photoshop.

modélisation architecturale : il convient plutôt d'élever des murs pleins. Ce procédé s'annonçait long et laborieux.

Là-dessus, Camille a choisi d'essayer d'effectuer ce modèle sur Blender. Après tout, ce logiciel semblait ouvert à une myriade d'utilisations différentes, et avoir nos deux modèles (filaire et réaliste) dans le même format semblait cohérent, et plus simple à adapter ultérieurement. Effectivement, créer le modèle filaire sous Blender s'avéra plus simple et plus rapide. Pour ce faire, il fallu tout d'abord créer un plan et plaquer dessus, en texture UV mapping, le plan au sol (récent, pour commencer), puis tracer des bandes sur l'épaisseur des murs. Ensuite, il s'agissait d'élever les arrêtes selon les élévations de Ouellet, plaquées sur des plans autour de la chapelle (voir Fig. 8.1). Il fallait dessiner en plaçant la vue parfaitement parallèle au plan en question.

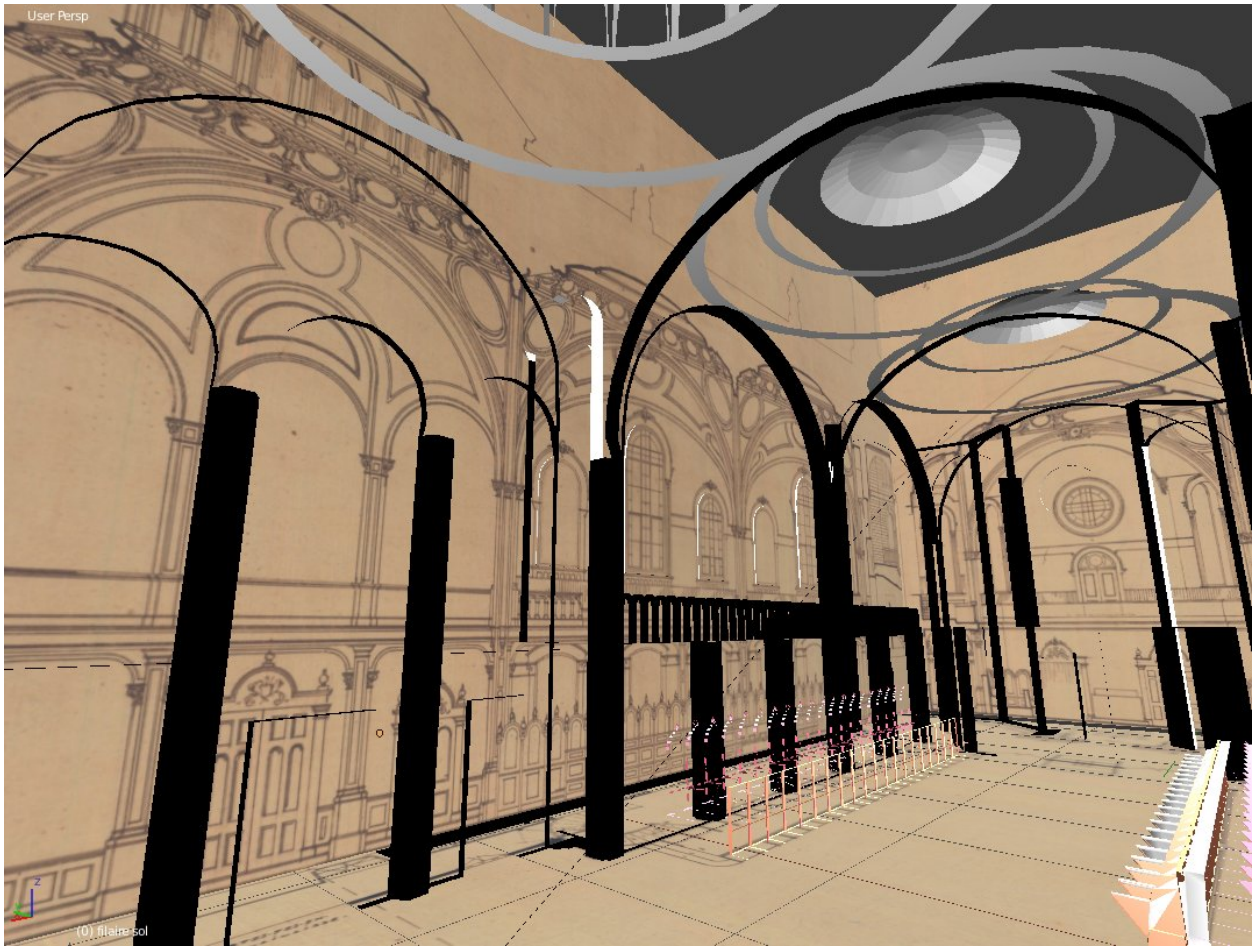


Fig. 8.1. : Apparence partielle du modèle filaire en mode texture

8.3. Choix de l'aspect : arrêtes et détails

L'architecture de la chapelle est très riche. Il fut nécessaire de la simplifier pour en garder l'essentiel. Plusieurs « couches »[3] ont été différenciées pour les arrêtes uniquement, les fenêtres, les éléments de décoration supplémentaires. Dominique Gélinas (doctorante) se chargea de ces choix.

3. modélisées par plusieurs objets sous Blender

En ce qui concerne l'aspect, les arrêtes sont actuellement des bandes de l'épaisseur des murs et les poteaux correspondent aux poteaux réels. La couleur reste à adapter en fonction du contenu que trouvera Mme Gélinas. L'aspect devra a priori être fait le plus mat possible, bien que des modifications ont déjà été faites sous Blender.

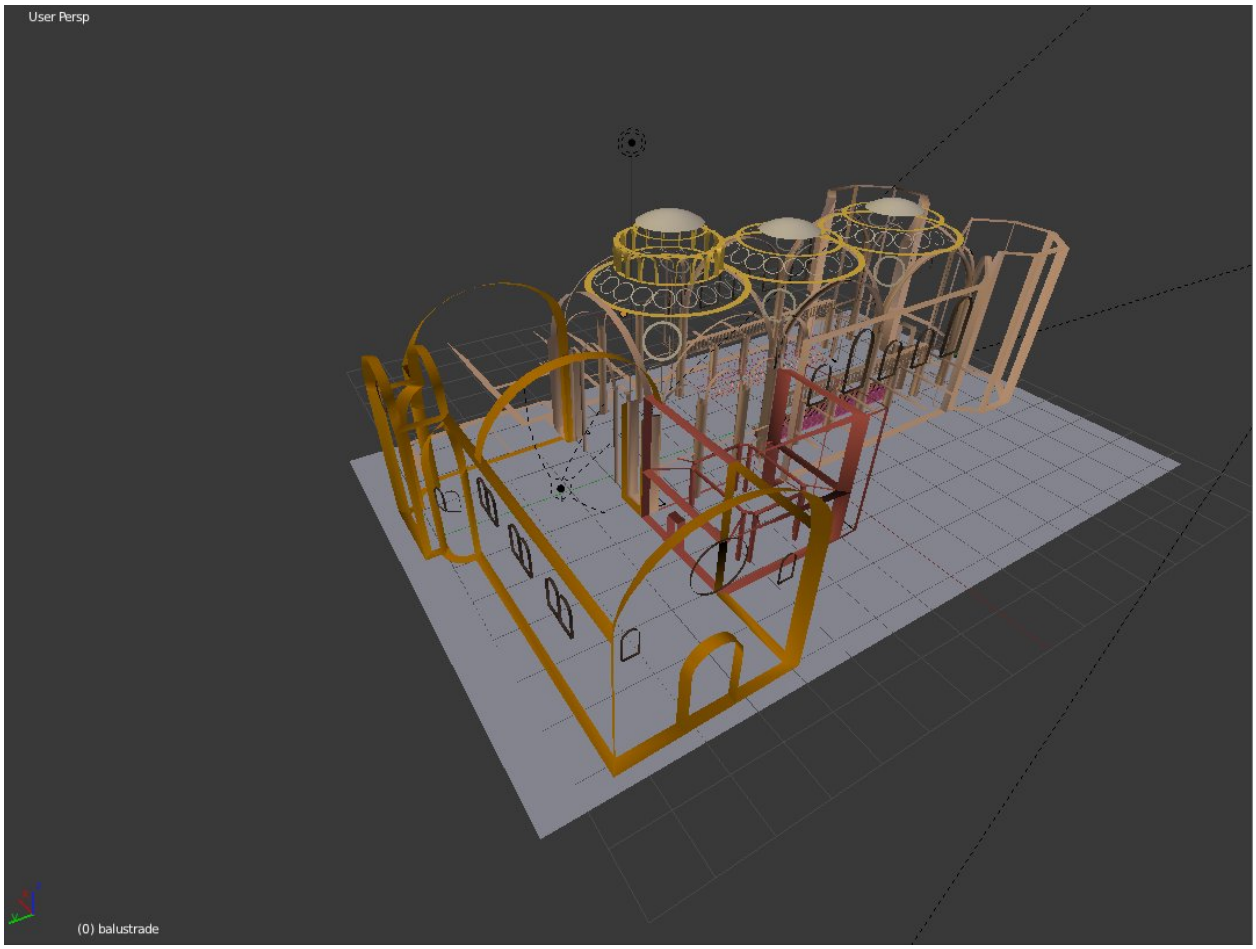


Fig. 8.2. : Première éventualité pour l'aspect des arrêtes

Lors de la réunion du 3 juilllet, M. Dubé a finalement exprimé des réserves sur l'aspect général qui semblait trop éloigné du réel et transmettre une fausse idée de la chapelle. L'hypothèse de l'affichage des arrêtes des polygones uniquement a été évoquée, pour bien signifier que le visiteur se situe dans un modèle irréel, dans un plan en trois dimensions. Néanmoins, une telle visualisation semble difficilement compréhensible.

Toutefois, et à notre relative surprise, lors de la dernière réunion, sœur Gabrielle a été touchée par ce modèle qui lui semble donner un nouveau jour au bâtiment qu'elle connaît parfaitement et dont l'aspect aérien vient rappeler à propos son objectif initial d'élévation de l'esprit.

8.4. Choix pour le mobilier

Sur les plans de Ouellet, les chaises de la nef étaient indiquées. Il a donc été décidé d'également les indiquer dans le modèle filaire. Néanmoins, elles ne devaient pas écraser le reste, tout en restant compréhensibles. Trois options sont apparues, après quelques tests et suggestions des différentes parties :

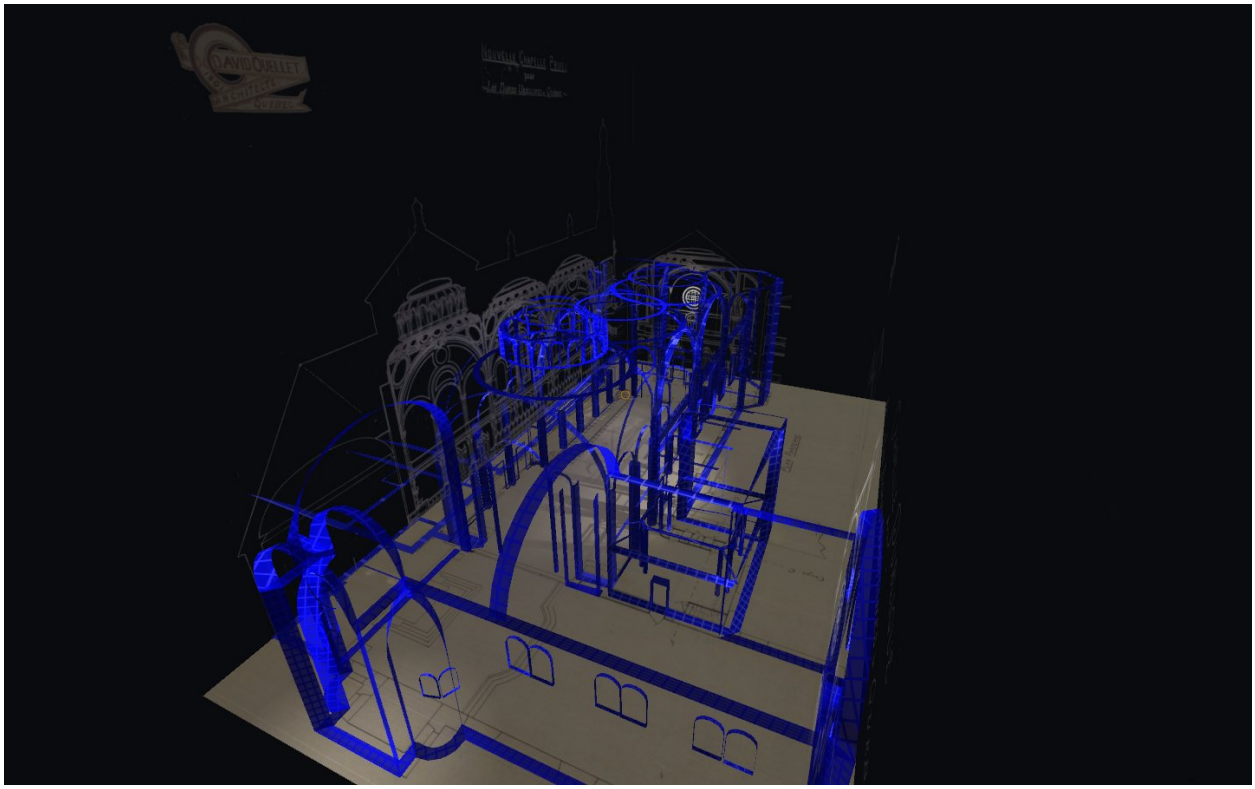


FIG. 8.3. : Aspect visuel du modèle filaire à notre départ

- Des sièges modélisés simplement, en quelques traits seulement, mais entièrement pleins.
- Ces mêmes sièges, indiqués uniquement par leurs arrêtes (aux assises et dossiers creux, donc)
- Ces mêmes sièges en arrêtes, surmontés d'une boule représentants leurs ornements, mais dont les arrêtes ne sont plus qu'en "pointillés" (un petit polygone sur deux)

Après plusieurs semaine de discussions, il a finalement été décidé de ne pas placer ces sièges.

Troisième partie .

Exploitation du modèle

9. Unity

9.1. Présentation

Unity est à la fois un moteur de rendu temps réel, un intégrateur de contenus multimédias et un éditeur/compilateur de code. D'après Wikipédia, on peut le comparer à Virtools, à l'UDK ou au CryEngine. S'il est majoritairement utilisé pour réaliser des jeux vidéos, il est aussi utilisé dans le domaine de l'architecture ou de l'éducation. Basé sur de nombreuses technologies libres (OpenGL, Mono[1], OggVorbis, Boo, theora, zlib, png, curl, scite, freetype, etc) il est néanmoins propriétaire et peut donc aussi intégrer le moteur PhysX de Nvidia pour la simulation physique. Il existe une version gratuite qui ne permet pas d'accéder à l'ensemble des fonctionnalités de la version professionnelle, facturée 1140€ — et qui s'est avérée nécessaire pour réaliser la projection dans le panoscope ou pour la lecture de vidéos. Il est possible d'exporter ses résultats sous forme de binaires pour Windows, MacOS (et Linux dans la future version 4) ainsi qu'en flash et dans un format destiné à un plugin web spécifique. Moyennant une nouvelle contribution de 305€ à la cause, on peut aussi exporter vers iOS et Android.

9.2. Fonctionnement visuel

Concrètement, après avoir lancé l'éditeur, on peut importer des *assets*[2] qui peuvent être des textures, des sons, des vidéos, des scripts ou encore des modèles 3D dans divers formats, chaque importation étant spécifiée par différents paramètres, notamment le degré de compression du son ou des images. Tous ces *assets* sont regroupés dans une banque de données[3] et on peut ensuite créer une scène puis la peupler avec des *assets*. Ils accèdent alors au statut de *GameObject*[4], à qui on peut affecter des Composants[5] pour préciser leur comportement. Ainsi, tous les objets possèdent un composant *transform* qui stocke leur position, la rotation et leur échelle et que l'on peut modifier en temps réel. La plupart ont aussi un composant *renderer* (qui définit les paramètres de représentation comme les matériaux utilisés ou les ombres projetées) et un *collider* (qui spécifie la forme et les dimensions de la bounding box). Certains objets spécifiques disposent de composants particuliers. Par exemple, le composant *Camera* d'une caméra, qui permet de fixer les options de projection, de culling, etc ; le *Character Controller* qui gère la physique irréaliste d'un personnage de jeu vidéo (qui peut faire des demi-tours instantanés quel que soit sa masse et sa vitesse) ou le *rigidbody*, plus conforme à la science (ou du moins à sa version simulée). La

1. c'est bizarre à dire, mais c'est libre aussi…
2. qu'on pourrait traduire par *ressource* mais ce serait plus vague (et plus long à taper…).
3. c'est là une des particularités d'Unity puisque les assets sont stockés dans un dossier spécial sur le disque dur et sont automatiquement versionnés. En cas de crash de l'interface et même sans sauvegarde, tous les assets sont encore présents au redémarrage, sauf le dernier, qui est soupçonné d'être le responsable de l'incident.
4. on ne peut désormais plus douter de l'orientation initiale de Unity…
5. *Components* dans la version originale, mais on est tout de même au Québec, un peu de respect pour la langue de Molière !

liste complète est volumineuse et comprend aussi ce qui se rapporte à l'audio, aux particules, à l'interface utilisateur, etc.

9.3. Aspect programmable

Au-delà de tous ces composants réutilisables qui couvrent déjà un large panel de besoins, Unity permet de spécifier des comportements personnalisés au moyen de scripts. Plus précisément, chaque script implémente une sous classe de la classe `MonoBehaviour` et devient un composant qu'on peut alors ajouter à n'importe quel GameObject. Il est paramétrable par ses variables membres publiques qui sont visibles dans l'interface principale. Ces scripts peuvent être écrits en Javascript[6], C#[7] ou Boo[8] et sont recompilés à chaque édition[9]. Ces sous-classes peuvent en particulier redéfinir les méthodes `Start()`, qui sert de constructeur, et `Update()`, qui est appelée à chaque affichage d'une image et qui permet par exemple d'animer un objet en interpolant sa position en fonction du temps écoulé depuis le dernier appel[10]. On peut aussi réagir à différents évènements en implémentant les méthodes dont le nom commence par `On`[11]. Ainsi `OnGUI()` est appelée à chaque événement d'entrée de l'utilisateur et `OnTriggerEnter()` est déclenché lorsque l'utilisateur entre dans une zone prédéfinie.

À ce stade, le lecteur est normalement impatient d'avoir un exemple de ces fameux scripts. Il y a en beaucoup en annexe mais on peut commencer par s'intéresser à `OuvrirPorte.cs` dont le nom résume pour une fois bien la fonction. Schématiquement, lorsque l'application démarre, il traite ses variables d'entrées fournies par l'utilisateur et récupère des références vers les objets sur lesquels il va agir. Lorsque le visiteur entre dans la zone à laquelle ce script est attachée, ce dernier lance l'animation d'ouverture, éventuellement accompagné d'un son puis il la bloque à la fin, pour laisser le temps au visiteur de franchir le seuil de la porte. Et quand il s'éloigne, le script rejoue l'animation à l'envers pour refermer la porte.

Une des particularités de la programmation sous Unity et qu'il y a souvent un choix à faire entre implémenter une fonctionnalité totalement par le code ou en laissant reposer une partie du travail sur ce que l'utilisateur peut faire dans l'éditeur. Ainsi pour accéder à un objet, on peut soit faire une recherche exhaustive à l'aide de la fonction `GameObject.Find()`, soit créer une variable publique de type GameObject et espérer qu'elle soit définie quand on y accède durant l'exécution[12]. Un autre point à garder à l'esprit est que puisque la fonction `Update()` est idéalement appelée plus de soixante fois par seconde, il faut être attentif à son coût temporel. La documentation fournit à ce sujet plusieurs pistes d'optimisation.

6. il s'agit en fait d'une version plus stricte que la norme ECMA ou toutes variables doivent être typées pour des raisons de performance.

7. ce qui leur permet d'accéder à une grande part de l'API .NET ; celle implémentée par le projet Mono qui présente l'avantage d'être disponible sur d'autre plateforme que Windows.

8. décrit comme un mélange entre python et C# par son créateur[1], et que j'aurais bien essayé par curiosité mais qui est manifestement très peu utilisé par la communauté ce qui réduit l'aide disponible.

9. une première fois sous forme de bibliothèques partagées .NET et recompilés juste-à-temps à l'exécution, ce qui permet d'atteindre la moitié des performances d'un code natif en C++

10. pour une simulation physique, on peut aussi utiliser `OnFixedUpdate()` qui est appelé à intervalle constant.

11. et dont la liste est disponible dans la documentation en ligne

12. ce qui est plus performant mais présente le risque de se retrouver avec de nombreux messages d'erreur à propos de `NullReference`.

10. Animation

En sus de l'interactivité, on peut rendre le modèle plus crédible et donc plus immersif en y ajoutant du mouvement. Il faut toutefois garder à l'esprit que l'atmosphère de recueillement qui imprègne le lieu empêche de se lancer dans une débauche d'effets cinématique. Nous avions donc considérer initialement trois type de mouvements : l'ouverture et la fermeture des portes lors des entrées et sorties afin d'augmenter le réalisme, le déplacement d'une barre pour les genoux sous les sièges, ce qui permet de mieux appréhender l'objet[1] et enfin le mouvement des pages d'un livre qu'on tourne[2].

C'est aussi l'occasion de présenter trois méthodes d'animation disponibles sous Blender qui s'exportent avec des fortunes diverses sous Unity

- images clés : la plus simple des trois, qui est d'ailleurs disponible directement dans Unity . Elle consiste à choisir la propriété numérique que l'on veut voir varier au cours du temps (la position en x du centre de gravité de l'objet ou la composante rouge de son matériau), lui donner des valeurs à certains temps clé[3] et à interpoler entre ces différents instants[4]. Cela convient bien à des mouvements de solides rigides et on peut s'aider d'un objet qui sert de chemin pour suivre des trajectoires complexes. En revanche, tous les sommets de l'objet conservent leur position relative. Si l'on veut lui faire subir des déformations, il faut donc se tourner vers une des deux autres méthodes.

- squelette : on attache à l'objet un squelette composé de différents os organisés de façon hiérarchique[5]. Chacun de ces os affecte chaque sommet avec un poids compris entre 0 et 1 (la situation la plus simple étant celle ou tous les poids valent soit 0 soit 1, ainsi à chaque os est associé un ensemble unique de sommets). On fait alors bouger chaque os avec la technique des images clés et l'objet se déforme pour suivre son squelette. C'est évidemment une technique particulièrement employée pour les personnages de jeux vidéo[6], donc bien supportée par Unity .

- *vertex morphing* : Cette dernière méthode consiste à stocker différente version d'un modèle, qui diffère par la position relative de certains sommets. Ces versions sont appelées *morph target* et à l'affichage de l'objet, on réalise une combinaison linéaire de certaines *morph target* pour obtenir la position effective de tous les sommets de l'objet. Pour reprendre l'exemple d'un jeu vidéo, un artiste peut réaliser des versions mâle, femelle, grosse et maigre du personnage et via des curseurs, le joueur peut ainsi personnaliser la morphologie de son avatar. C'est toutefois une méthode moins anthropomorphique que l'animation squelettique qui peut donc s'adapter à une classe plus large d'objets. En revanche, elle n'est

1. mais que nous n'avons finalement pas réalisé
2. qui a été fait mais pas intégré pour des problèmes détaillés à la fin du chapitre.
3. l'unité de temps étant la frame (d'où encore une fois le nom de la technique en anglais, *keyframe*), qu'on retrouve 24 ou 60 fois par seconde, et qui permet de faire varier la vitesse de l'animation suivant les besoins.
4. on peut choisir entre une interpolation constante, linéaire ou de bézier.
5. les os fils « subissent » les rotations des os desquelles ils descendent (ceux à qui ils sont attachés par la tête).
6. puisqu'outre son aspect intuitif, on peut la combiner avec la motion capture ou la cinématique inverse, qui consiste à spécifier les positions finales des os terminaux et laisser le logiciel calculer les mouvements des os intermédiaires

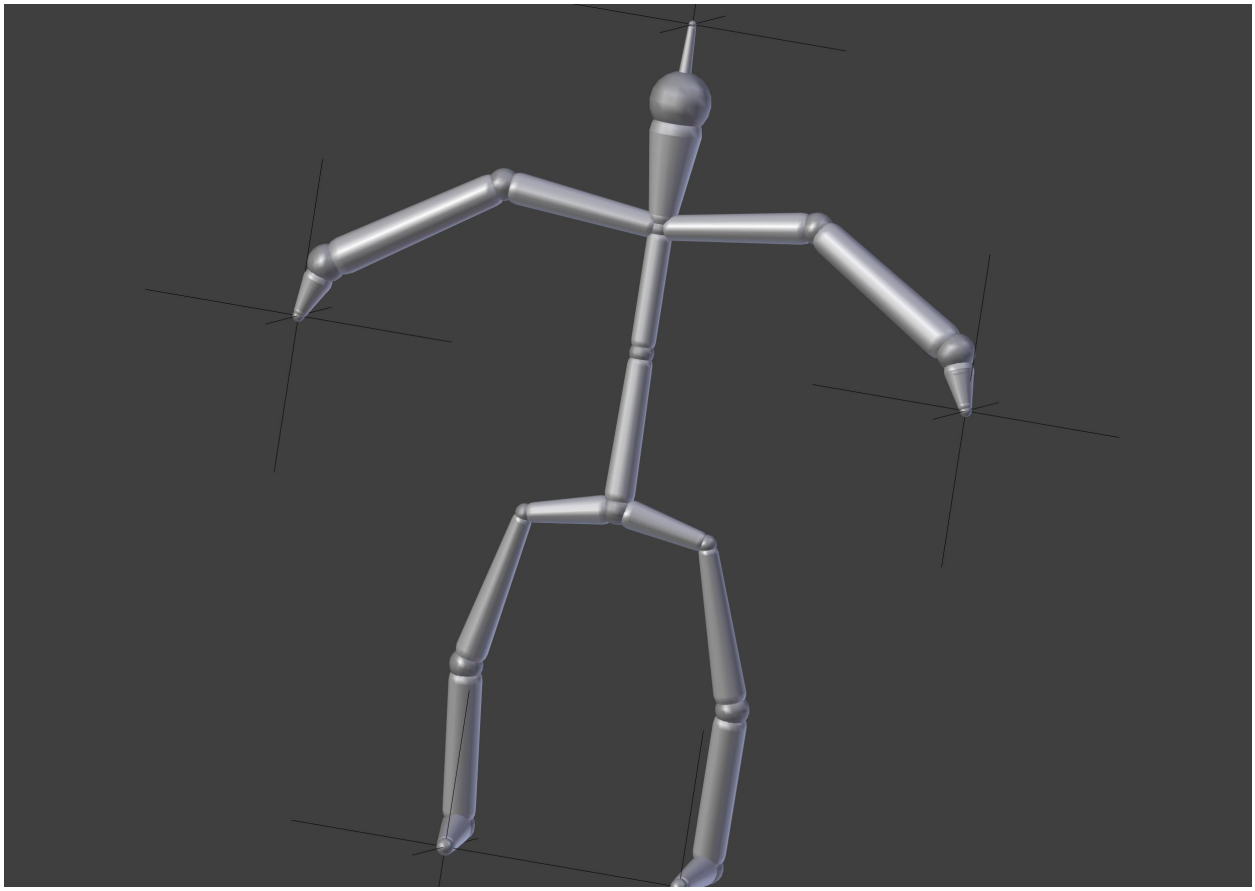


Fig. 10.1. : Un squelette basique dans Blender , qui n'attend plus que de recevoir ses membres.

pas supportée nativement par Unity . Il existe un plugin pour Blender, qui ne fonctionne pas avec la dernière version et qui donc été assez laborieux à mettre en place. J'ai essayé de l'utiliser pour corner les pages du livre en train de se tourner mais il stocke les déplacements des sommets dans une image (dont les composantes rouge, vert, bleu représentent un vecteur) dont la résolution n'était compatible avec la finesse des pages, ce qui donnait un résultat décevant.

Puisque le *vertex morphing* s'est révélé trop difficile à faire fonctionner, et qu'en revanche la méthode des images clés étaient convaincantes pour les mouvements de porte, Camille a décidé de s'en servir pour animer les pages du livres[7]. Nous nous sommes alors aperçus que même si les pages ne se déformaient pas, cela restait assez explicite pour donner une idée du mouvement. Mon projet était alors de récupérer ce modèle 3D avec une animation de changement de page et de faire un script qui remplace la texture de la page et lance l'animation à chaque fois que l'utilisateur clique[8]. Par manque de temps, cela n'a abouti et je me suis résolu à intégrer directement la vidéo à la scène. Là encore, après m'être entêté à essayer d'employé le canal alpha de la vidéo pour que le livre s'insère plus harmonieusement aux autres objets, j'ai fini par abandonner devant la syntaxe peu familière des shaders d'Unity .

Ensuite, après avoir constaté qu'en l'absence de mobilier, les échelles dans le modèle filaire pouvaient se montrer difficile à évaluer au premier abord, nous avons décidé d'y ajouter un personnage animé qui aurait marché en même temps que le visiteur se déplace. Mais par manque de temps et d'expérience, le résultat n'est pas apparu très probant et il a été décidé d'insérer plutôt des silhouettes planes dessinées à la main par Camille pour remplir cet office.

7. reprenant ainsi le travail réalisé par Elsa.
8. ou effectue éventuellement une action plus naturelle comme un geste de la main.

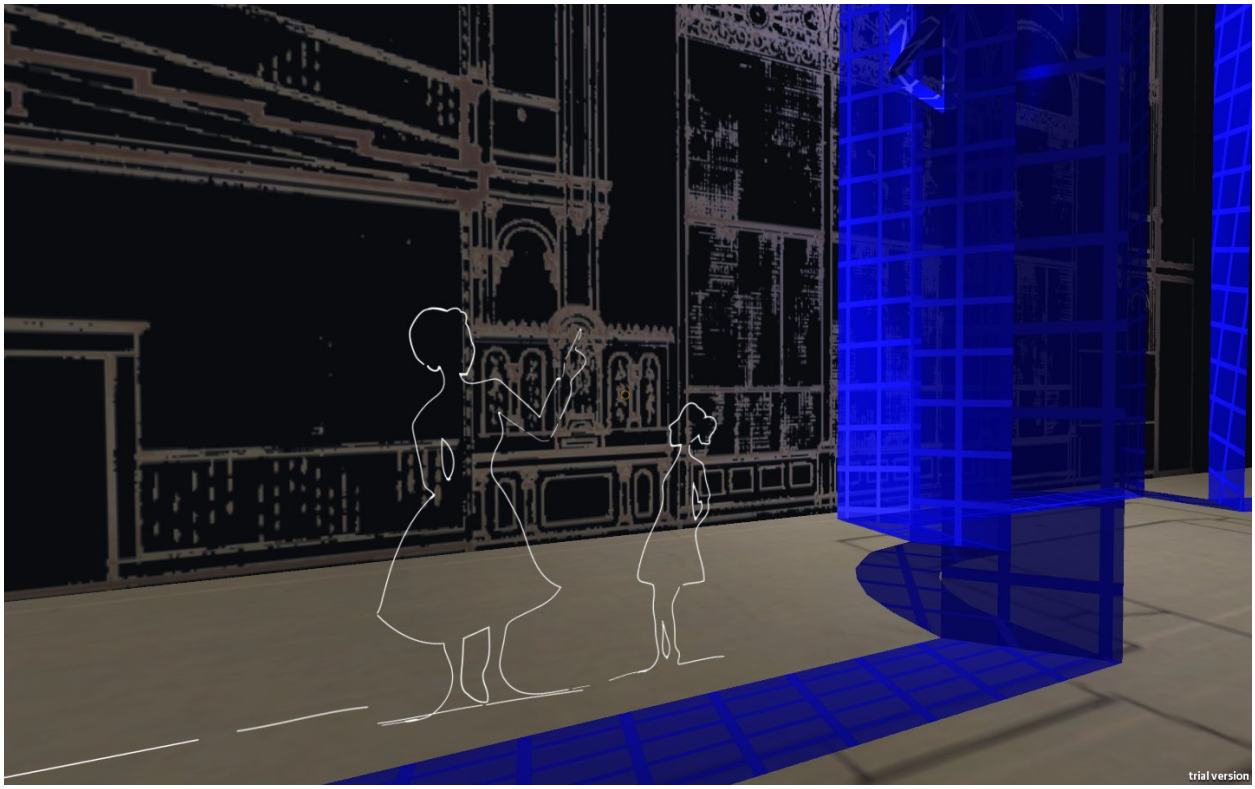


FIG. 10.2. : Avec une méthode plus « manuelle », on remplit l'objectif de donner une échelle du lieu tout en gagnant en esthétisme.

11. Interactions

Dans un environnement virtuel, l'interactivité consiste à permettre à l'utilisateur de reproduire dans le modèle numérique les actions qu'il pourrait entreprendre dans le lieu réel. Viser l'exhaustivité dans ce domaine est évidemment aussi long que compliqué. Heureusement, le caractère spirituel et essentiellement contemplatif de la chapelle, ainsi que la vocation muséale de notre application, a nettement réduit le champ des possibles. Au final, les seules interactions implémentées sont l'ouverture des portes et l'affichage de contenu en relation avec certains objets particuliers. Nous allons donc voir les modalités de déclenchement de ces interactions, le système de gestion de contenu et celui de son affichage à l'écran.

11.1. Modalités

Dans un souci de simplicité, les portes s'ouvrent automatiquement dès lors que l'utilisateur s'en approche, et se referme toute seule lorsqu'il s'éloigne suffisamment. Afin d'ajouter un peu de réalisme, ce mouvement s'accompagne de la lecture d'un léger bruit de grincement. En ce qui concerne les objets interactifs, l'approche est différente car il s'agit de la part du visiteur d'un processus volontaire. Néanmoins, dans la version actuelle de l'application, le visiteur étant en situation de totale autonomie, il lui faut un moyen de découvrir ces objets particuliers. Il a pour ce faire à sa disposition un curseur placé au centre de l'écran[1] qui déclenche l'allumage d'une lampe et l'apparition d'un court texte au survol d'un tel artefact, tel qu'illustré sur la Fig. 11.1 p. 53.

Afin que toutes les lumières ne s'allument pas un coup et ne transforment la chapelle en boîte de nuit mais aussi pour que l'utilisateur ne soit pas désorienté par la densité importante d'objets cliquables[2], l'objet ne s'illumine que s'il se trouve à une distance minimum du visiteur.

Lorsque celui-ci clique, on lance une transition qui l'amène de sa position initiale dans la chapelle entière à un point de vue défini à l'avance en fonction de l'objet tandis que la chapelle se colore progressivement en gris foncé pour continuer à donner une idée discrète de l'espace tout en mettant en valeur le contenu choisi, donnant l'apparence que l'on observe Fig. 11.2 p. 53.

11.2. Gestion de contenu

11.2.1. Type des contenus

Actuellement, quatre types de contenus sont supportés : des textes, des images fixes (photos, dessins, plans, schémas, ...), des vidéos et des sons. Cela permet de couvrir l'éventail des besoins

1. voir les codes `Pointer.cs`, `PointerHandler.cs` et `PointerEvents.cs`
2. 17 dans le cœur et la nef.

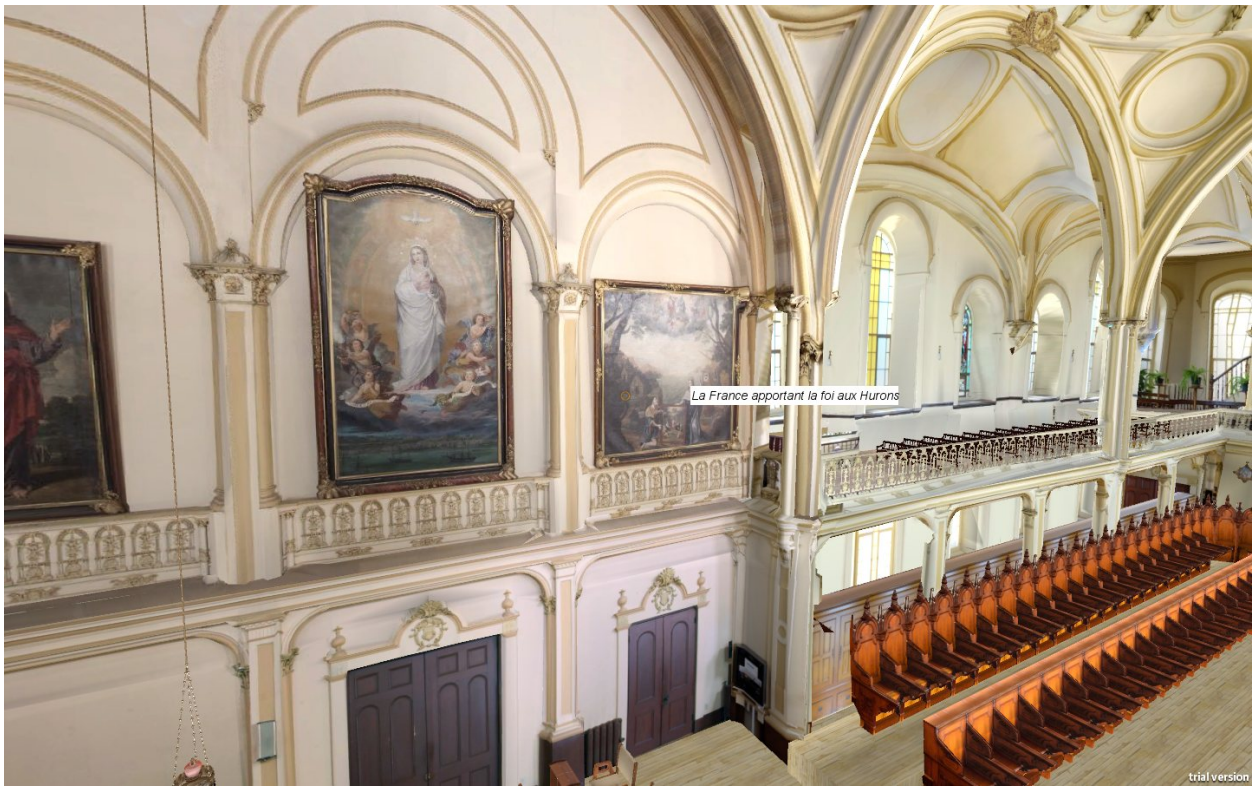


FIG. 11.1. : Ici, l'utilisateur fait passer son curseur au-dessus d'un tableau et la lumière et le texte sont censés capter son intention et l'inciter à cliquer dessus.

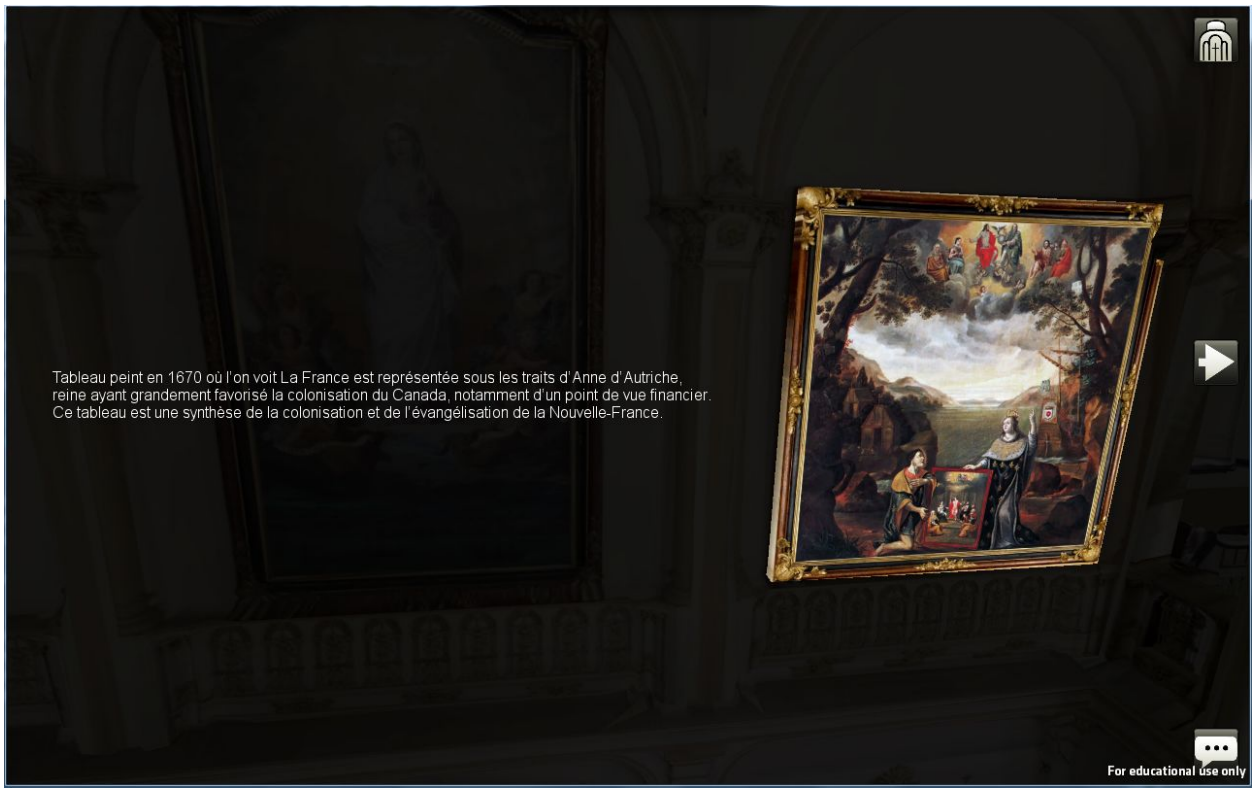


FIG. 11.2. : Un texte descriptif du tableau, qui reste lumineux alors que le reste du lieu est plongé dans les ténèbres

les plus courants. Néanmoins, un dernier type de contenu n'a paradoxalement pas été exploité, il s'agit des modèles 3D. En effet, lorsque le visiteur clique sur un objet, on lui substitue une version de meilleure résolution modélisée en 3D[3]. On pourrait donc concevoir que le visiteur puisse manipuler cet objet dans l'espace. Que ce soit pour zoomer dessus afin d'afficher tous les détails d'un tableau ou pour le percevoir sous ses différents angles dans le cas de l'orgue par exemple.

11.2.2. Stockage

La liste des objets interactifs ainsi que leur contenu afférent est stockée dans une base de données hébergée sur un des serveurs du LAMIC . C'était une demande forte qui permet un dynamisme certain. En effet, on peut ajouter une photo en corriger un texte sans avoir à recompiler l'application. Dans les faits, cela a engendré des complications mais il c'est révélé satisfaisant de pouvoir corriger une faute d'orthographe entre deux clique sur le même objet.

Il a en revanche été délicat de déterminer quelles informations étaient susceptibles d'être ainsi stockées. En particulier, alors que je projetais initialement d'inclure la disposition des éléments directement dans la base de données cela n'a finalement pas pu être réalisé. J'en dirai néanmoins quelques mots dans la suite.

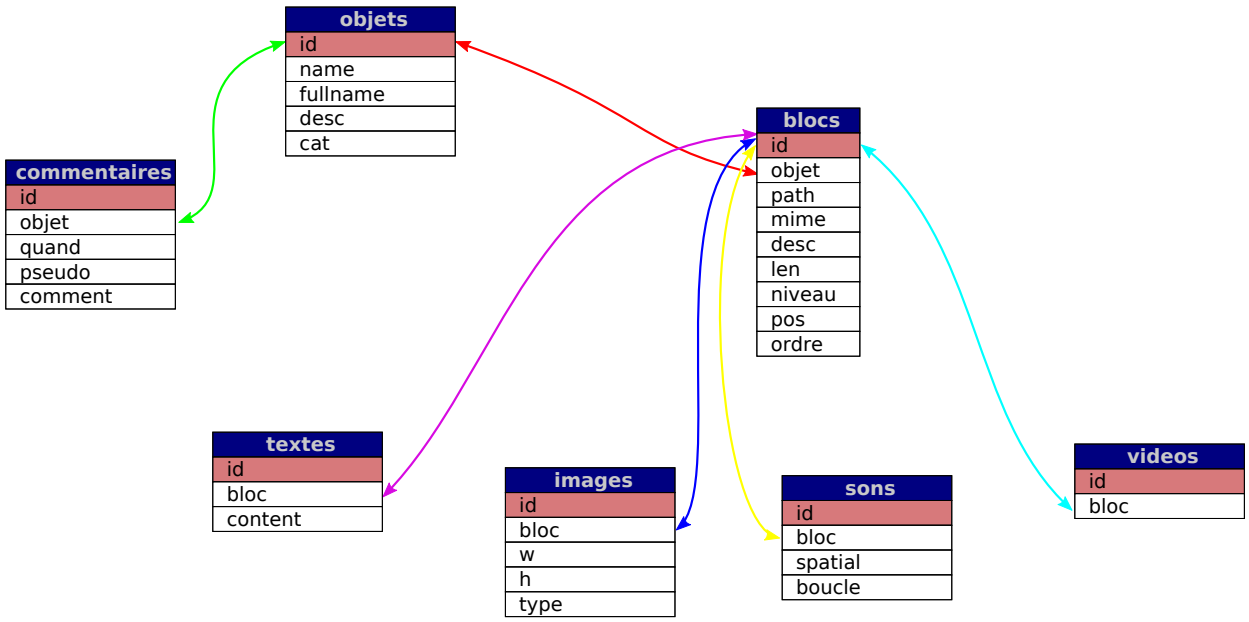


FIG. 11.3. : Schéma physique de la base de données.

À la lecture du schéma 11.3 p. 54[4], on pourrait considérer que chaque table correspond à une classe. La classe principale est celle qui représente un objet, identifié par son nom interne `name` et accompagné de diverses informations : le nom qui est affiché aux visiteurs, une brève description et la catégorie de l'objet. Ces deux dernières ne sont pas actuellement exploitées, et on pourrait même imaginer ajouter d'autres métadonnées. Si l'on disposait de l'année de création de l'objet ainsi que de son auteur, on pourrait par exemple n'afficher que les tableaux peints par des artistes français entre 1750 et 1850. Il est bien évident que cette possibilité ne prend son sens qu'avec un

3. à noter toutefois que le script `VoitTableau.cs` prévoit la possibilité d'interagir avec un objet immatériel — par exemple, l'autel principal qui n'apparaît que sur une photo ou des objets trop volumineux comme la coupole. Dans ce cas l'objet n'est pas affiché et le contenu remplit tout écran.

4. dont les informations sont plus détaillées dans le fichier `bdd-structure.sql`.

nombre d'objets plus conséquent que ce trouve pour le moment dans la chapelle. À chaque objet est aussi associé une liste de commentaires et une liste de blocs. Les commentaires sont laissés par le visiteur lorsqu'il a terminé de lire le contenu de l'objet (ou il peut se contenter de lire les messages de ceux qui l'ont précédés, voir Fig. 11.4 p. 55). C'est un premier pas modeste vers une version multi-utilisateurs de l'application.

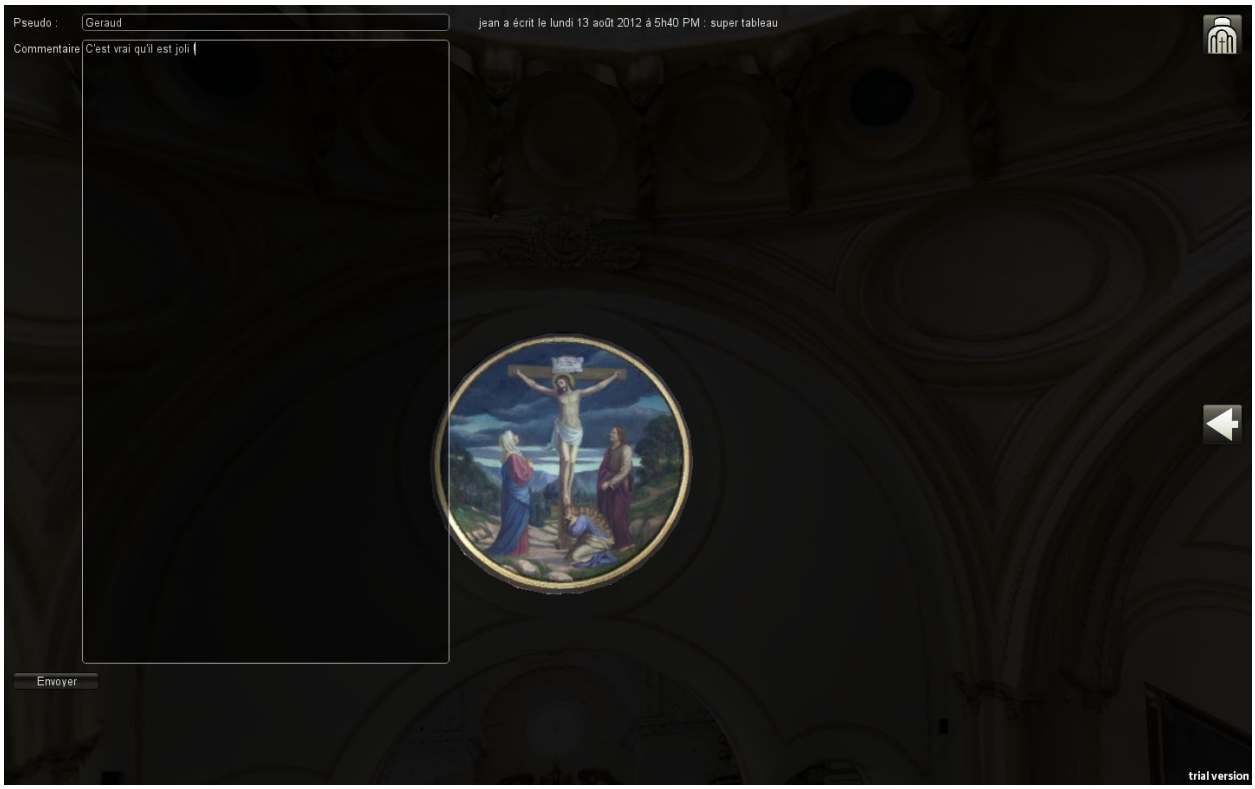


FIG. 11.4. : Interface de saisie et de lecture des commentaires.

Un bloc constitue en quelque sorte une classe mère des différents types de contenu et regroupe leur métadonnées commune comme leur nom, leur type, leur taille ainsi que le chemin où ils sont stockés sur le serveur. D'un point de vue organisation, c'est aussi là qu'est indiqué le niveau auquel se situe le contenu ainsi que son ordre d'affichage. En effet, les contenus sont classés suivant trois niveaux d'approfondissement qui sont affichés les uns à la suite des autres. Les quatre sous-classes contiennent elles des informations spécifiques au type de contenu envisagé. Les dimensions et la nature d'une image, le contenu d'un texte ou encore le caractère spatial ou non d'un son.

Mon projet initial était qu'une fois les contenus ajoutés à la base de données, un programme Unity les récupère et les affiche tous à l'écran dans des fenêtres déplacables et redimensionnables. Le créateur du contenu aurait ainsi pu les disposer comme il l'aurait souhaité, avant de cliquer sur un bouton de confirmation qui aurait renvoyé au serveur la position de chaque rectangle dans le bloc correspondant. Ce n'est finalement pas cette solution qui a été retenue comme nous allons bientôt le voir dans la partie affichage. Mais intéressons-nous d'abord au système qui permet d'ajouter initialement les contenus.

11.2.3. Insertion de contenus

L'insertion des contenus s'effectue grâce un formulaire html relativement classique, dont on peut voir des illustrations aux Fig. 11.5 p. 56 et 11.6 p. 56. Dans la mesure du possible, j'ai

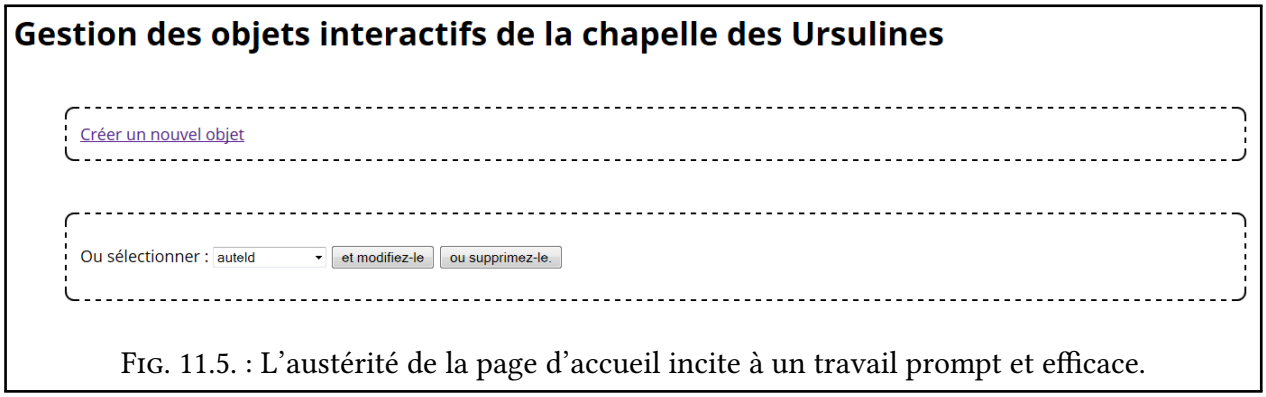


FIG. 11.5. : L'austérité de la page d'accueil incite à un travail prompt et efficace.

tenté d'utiliser les nouvelles fonctionnalités proposées par le standard html5 en cours d'écriture.

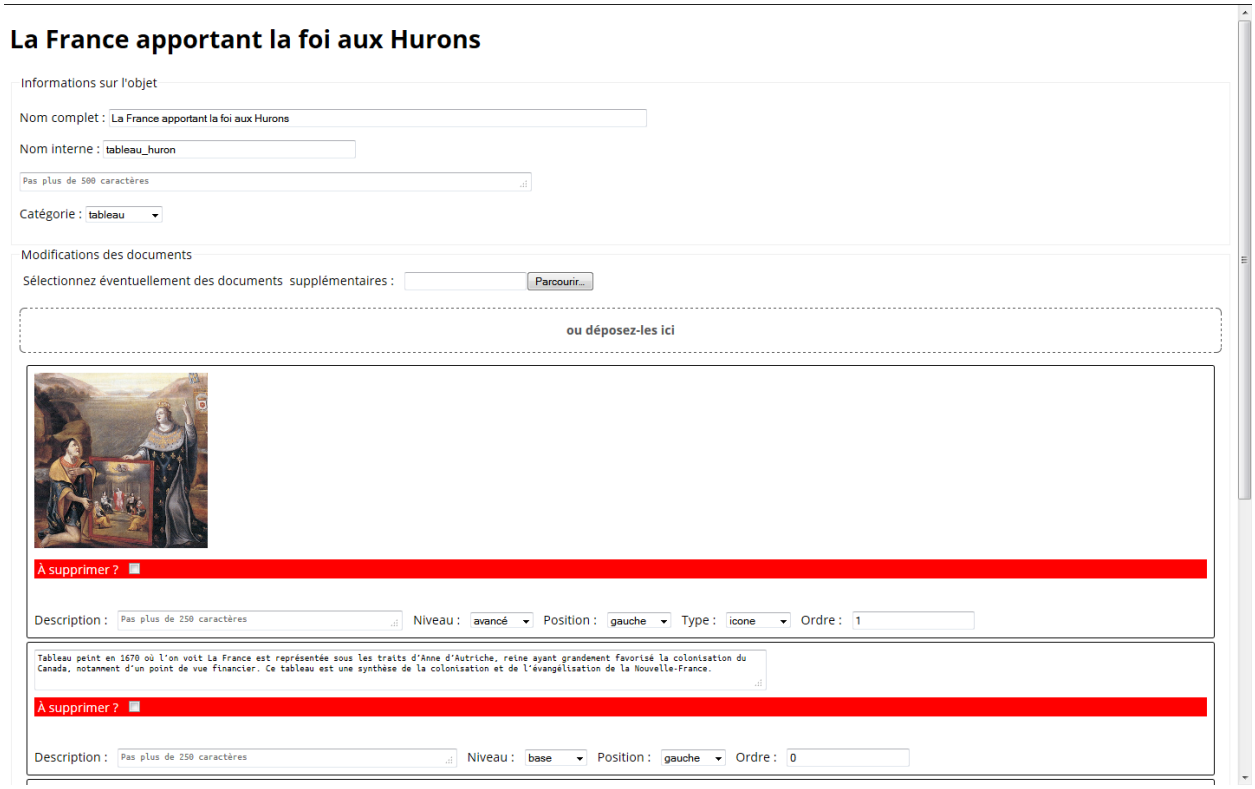


FIG. 11.6. : L'interface basique mais dynamique qui permet d'ajouter, modifier et supprimer des contenus

- champs typés
- lecture audio et vidéo
- barres de progression
- sélection de fichiers multiples
- glisser-déposer

Cela a été l'occasion de constater que toutes ces fonctionnalités n'étaient pas encore implémentées même dans les navigateurs récents et notamment en ce qui concerne la validation automatique des informations entrées dans un champ.

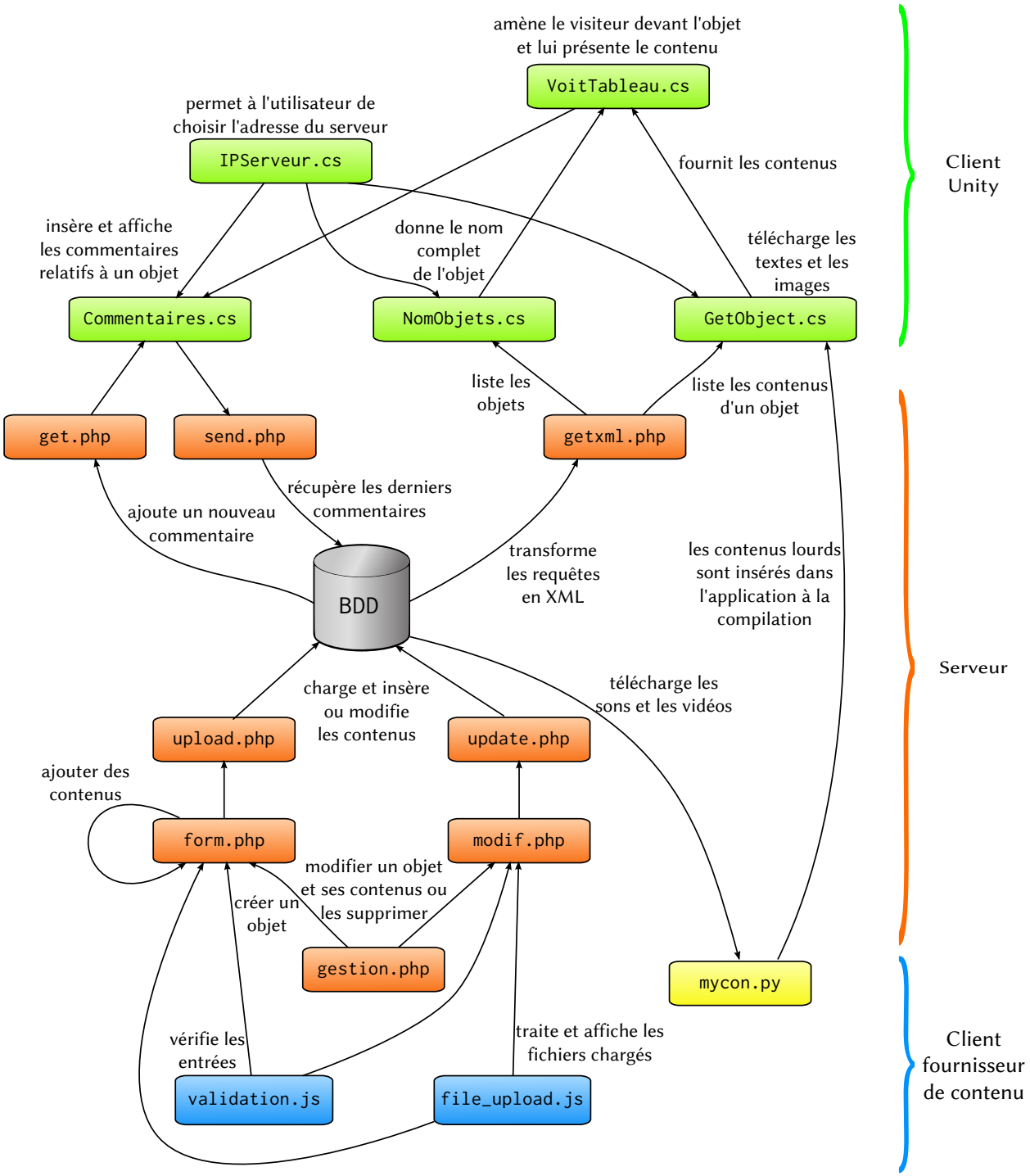


FIG. 11.7. : L'architecture globale du système de gestion de contenus.

Ce formulaire communique avec une partie serveur écrite en PHP qui insère et met à jour la base de données et permet de récupérer les informations relatives à un objet par l'intermédiaire de fichier XML. Une autre partie cliente écrite en C# s'exécute dans Unity et s'enquiert du contenu et des commentaires à la demande, en téléchargeant le texte et les images et en supposant que les sons et les vidéos sont déjà disponibles dans un dossier `Ressources` d'où Unity peut les chargés à l'exécution (pour plus de détails, voir le schéma 11.7 p. 57 et lire le document *Manipulation de la base de données* en annexe ainsi évidemment que le code source[5]). À noter que le serveur par défaut est `localhost` et qu'il est stocké dans les préférences utilisateur. Cela peut être changé soit à la compilation, soit durant l'exécution en appuyant sur la touche `k` et en entrant une nouvelle adresse.

11.2.4. Affichage

En effet, plutôt que de laisser la personne qui ajoute le contenu le disposer elle-même, j'ai préféré écrire un code qui choisi automatiquement et toujours de la même manière où placer les éléments en fonction de leur taille et du niveau d'intérêt (voir Fig. 11.8, 11.9, 11.10 pp.58—59). Cela présente l'avantage d'être très rapide[6] et assez général, ce que l'on peut aussi considérer comme un défaut lorsque certains contenus exigeraient une mise en forme particulière[7]. Concrètement, on passe à la fonction `ecran()` une zone de l'écran, un texte et une liste de textures (animées ou non) et celle-ci renvoie une liste de rectangles[8] qui indique où placer ces éléments à l'écran. Cet affichage peut se réaliser de deux façons différentes.

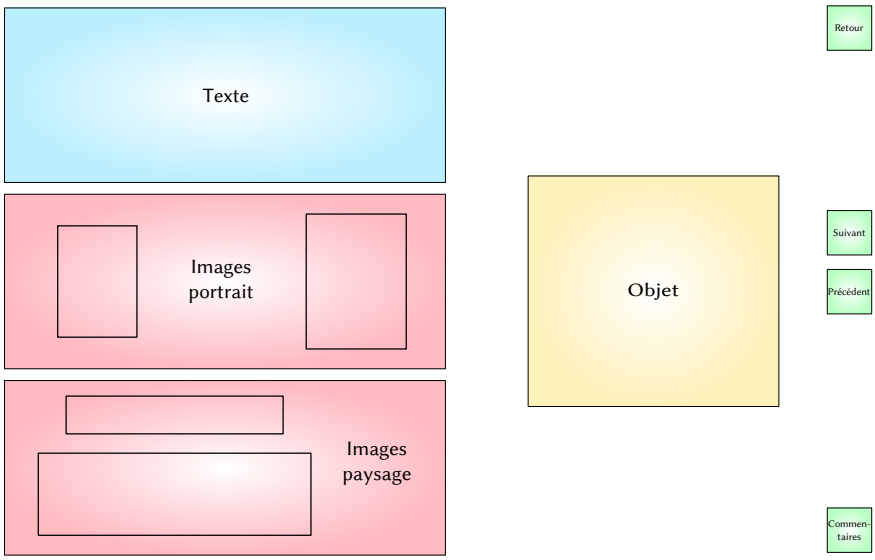


Fig. 11.8. : Au premier niveau, le texte est généralement court pour laisser de la place à l'objet et aux images

5. Outre les fichiers `file_upload.js`, `validation.js`, `mycon.py`, `gestion.php`, `modif.php`, `form.php`, `upload.php`, `update.php`, `get.php`, `send.php`, `getxml.php`, `Commentaires.cs`, `NomsObjets.cs`, `GetObject.cs`, `IPServeur.cs` et `VoitTableau.cs`, le fichier `utils.php` comprend plusieurs fonctions… utiles.

6. du moins quand on ne tient plus compte du temps passé à écrire le code, notamment celui de `Layout.cs`.

7. on peut notamment penser au 14 médaillons qui composent le chemin de croix faisant le tour de la nef.

8. définis par leur coin supérieur gauche ainsi que leur largeur et leur hauteur.

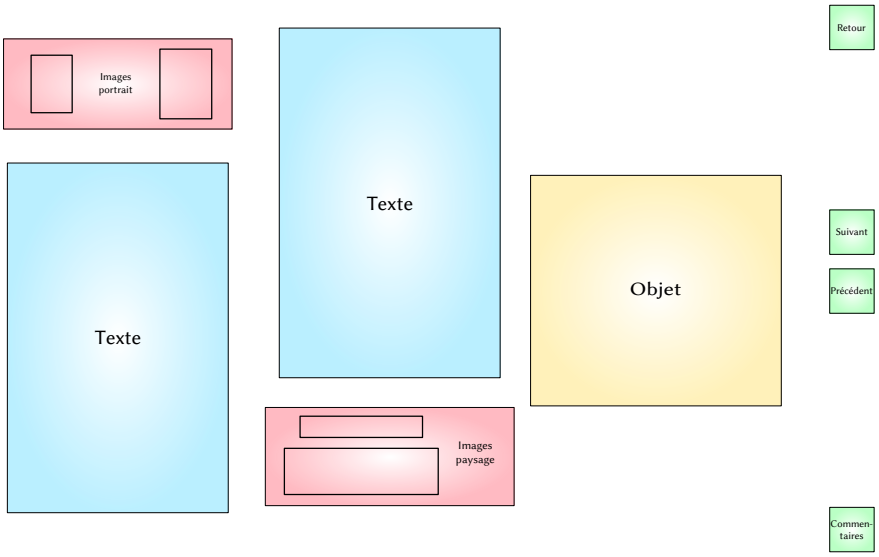


FIG. 11.9. : Le deuxième niveau a une mise page plus délicate mais qui peut se simplifier dans les cas où l'on a qu'un type d'image, voir aucun.

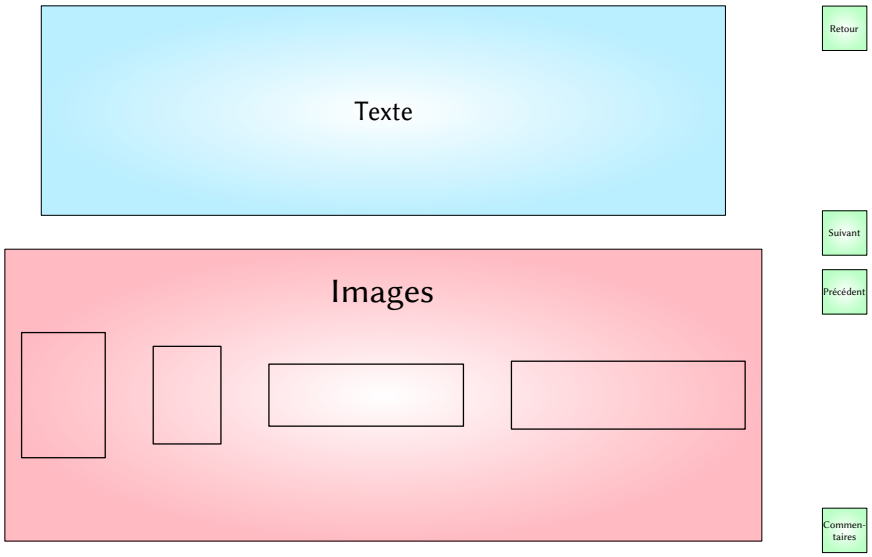


FIG. 11.10. : Au troisième niveau, l'objet disparait complètement au profit d'une analyse plus poussée

2D

La plus simple et la plus évidente est d'utiliser les fonctions qu'Unity met à notre disposition à cet effet. Elles permettent de dessiner sur une sorte de calque 2D attaché à une caméra de nombreux contrôles différents, tels que des labels, des boutons, des cases à cocher, des champs de texte ou encore des images. On peut voir que son utilisation est assez évidente au début de la fonction `OnGUI()` de `VoitTableau.cs`. Néanmoins, elle présente l'inconvénient d'être liée à une caméra, ce qui pose problème lorsque l'on filme avec un anneau de caméras pour projeter l'images à 360°. Le contenu doit alors directement être affiché dans l'espace de la scène et non pas superposé à un écran qui est désormais circulaire.

3D

Malheureusement cette approche n'est pas prévue de base et il a donc fallu l'a développer par moi-même, dans le fichier `HGui.cs`. Je me suis concentré sur quatre éléments : mesurer la taille qu'un objet occupe l'écran, afficher une image, afficher un texte et afficher un bouton.

Chaque objet 3D peut être muni d'un `Collider` qui peut prendre la forme d'une boîte, d'une sphère voir même être directement calculé au plus près de sa géométrie. Connaissant un rectangle censé contenir l'objet, `ApproxSize()` lance des rayons issus de la caméra avec un certain pas, en partant des quatres coins du rectangle jusqu'à ce qu'ils rencontrent l'objet. On détermine ainsi quatre droites dont l'intersection nous donne un rectangle plus petit qui contient l'objet. `ObjectSize()` appelle itérativement cette fonction en commençant par balayer l'écran entier et en divisant à chaque fois le pas par trois.

Les images sont affichées comme texture d'un plan comprenant quatre sommets. Pour les placer correctement, le script `PlacePlane.cs` lance quatre rayons issus de la caméra et ayant une direction telle qu'il passe par un point donné de l'écran[9]. Ces quatre rayons intersecte un plan placé un peu après la caméra et on déplace les quatre sommets du plan pour qu'ils coïncident avec ces points d'intersection.

Unity dispose déjà d'un composant permettant d'afficher du texte en 3D, la difficulté étant que dans ce cas, le texte n'est pas mis en forme et il faut insérer manuellement des sauts de ligne. Pour ce faire, on calcule la taille la taille moyenne de chaque caractère en pixels dans la fonction `HGui.getCharPixelSize()` et on en déduit où aller à la ligne dans `Layout.formatter()`[10]

Enfin, un bouton n'est qu'un plan qui affiche successivement une image ou l'autre suivant que la souris se situe au dessus de lui et exécute une fonction lorsque que l'on clique dessus (voir `Bouton.cs`)

Le résultat visuel est globalement comparable même s'il est un peu moins net en 3D (voir Fig. 11.11 p. 61) mais cela offre des possibilités de visualisation amusantes puisque des visiteurs se déplaçant dans la chapelle pourraient éventuellement voir apparaître le contenu visualisé par d'autres.

Le mode d'affichage est 2D par défaut sur les plates-formes mobiles car il n'y a qu'un seul utilisateur sur un écran plat. Il est défini à la compilation dans les autres cas[11] et on peut passer d'un mode à l'autre pendant l'exécution en appuyant sur la touche `i`, même si ce basculement n'est pas totalement au point et conduit à des bogues d'affichage.

9. un tel rayon que étant obtenu par la fonction ScreenPointToRay de la caméra

10. malheureusement, `getCharPixelSize` ne fonctionne pas dans tous les cas à cause de la rotation du texte autour de l'axe Y et il a fallu se résoudre à pré-calculer la taille des caractères, aussi peu satisfaisant soit cette approche.

11. par la variable booléenne `interface2D` dans `Statut.cs`

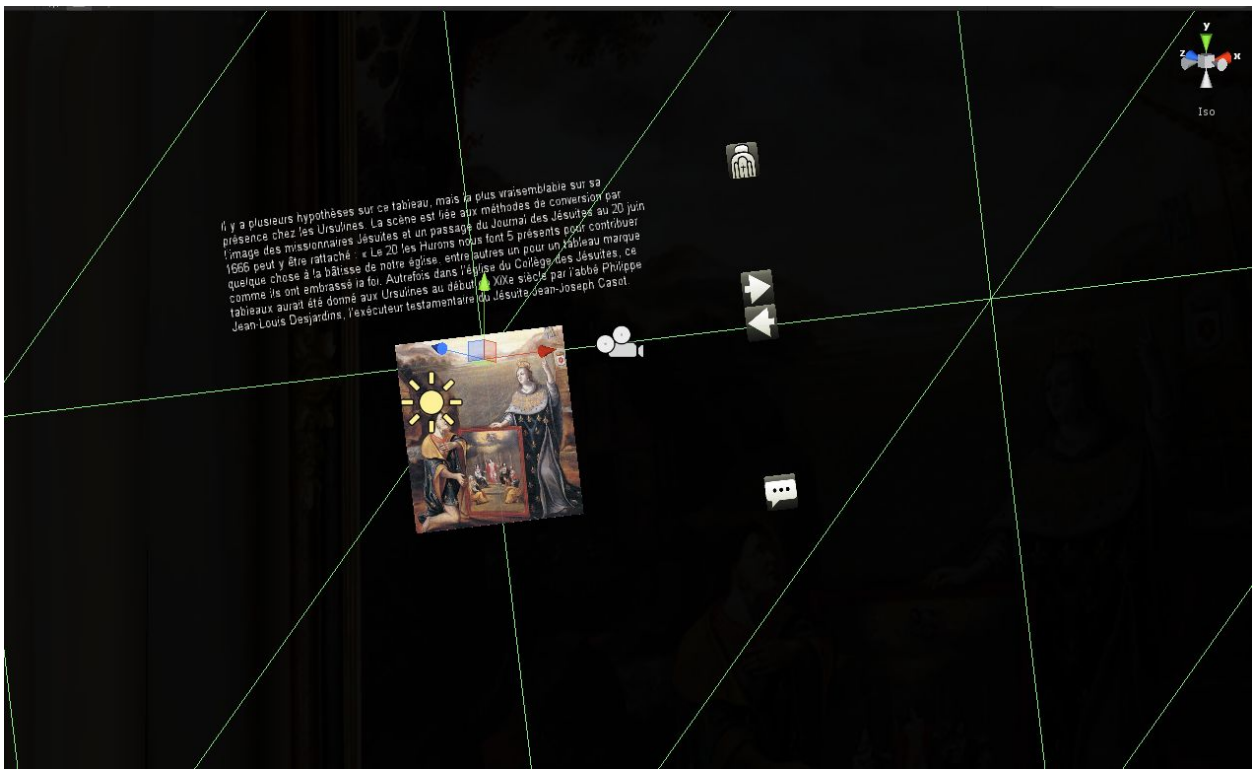


Fig. 11.11. : Les éléments d'interfaces sont projetés sur un plan normal à la caméra, ici dessiné en vert.

12. Interface

Pour que la visite ait un sens, on doit établir une interface entre le visiteur et le modèle qui leur permette tout deux de communiquer. Le modèle communique surtout visuellement, que ce soit par son aspect général ou par les lumières et le texte qui signalent une possible interaction au visiteur. Il est aussi présent sur le plan sonore par la musique d'orgue qui est jouée durant la visite. On pourrait enfin envisager d'agir sur les autres sens du visiteur, comme nous le verrons dans le chapitre consacré aux perspectives. Dans l'autre sens, l'utilisateur a essentiellement besoin de naviguer et donc de communiquer sa position et son orientation par rapport au modèle. Cela peut se faire suivant les différentes méthodes que nous allons maintenant expliciter :

12.1. Clavier et souris

La plus simple à mettre en œuvre, et que j'estimais aussi être la plus simple à manipuler[1], est d'utiliser les touches directionnelles du clavier pour se déplacer et les mouvements de la souris pour orienter son regard. Pour des raisons d'immersion, la scène est filmée dans Unity par un anneau de cinq caméras et on peut utiliser celle à l'avant, à gauche et à droite pour projeter sur trois murs contigus d'une pièce. Dans ce cas, le visiteur n'est plus physiquement devant le clavier et on doit lui substituer une manette de jeu. Cette approche classique, qui est implémentée dans les scripts `CameraFly.cs` et `MouseLook.cs`, présente néanmoins l'inconvénient d'être peu naturelle. Nous nous sommes donc tournés vers l'iPad .

12.2. iPad

12.2.1. Approche simpliste

La première idée qui vient à l'esprit est d'utiliser l'iPad comme un clavier portatif et de reproduire les touches utiles dessus. C'est d'ailleurs l'approche utisée par la SAT dans le dôme et en faisant communiquer l'iPad avec l'application Unity par le protocole OSC, cela permet effectivement de se déplacer tout en étant à distance de l'ordinateur. On ne peut toutefois pas prétendre que cette manière de faire soit vraiment plus naturelle.

12.2.2. Orientation

Il faut donc tirer parti des caractéristiques propres de ce dispositif. En l'occurrence, il est muni d'un gyroscope, et Unity permet d'accéder un quaternion représentant la rotation menant l'iPad de sa position posé sur une table à sa position actuelle. Le script `GyroCam.js` — trouvé sur Internet

1. même si j'ai pu constater que cela suppose de ne pas avoir trop de lacunes en matière de culture vidéoludique.

— récupère cette information et la transmet à la caméra d'Unity. Cette fois, le résultat semble assez naturel même s'il demeure un problème d'angle de vue de la caméra si l'on veut superposer l'image affichée par l'iPad à la réalité. Un autre souci plus gênant provient de l'incertitude qui règne sur l'orientation initiale de l'iPad , ce qui empêche de recaler le modèle virtuel avec son homologue réel. Deux pistes ont été avancées :

- utiliser la boussole intégrée à l'iPad , qui n'est pas un dispositif analogique et qui ne donne donc qu'une information de qualité assez mauvaise.
- s'assurer que tous les utilisateurs ont la même orientation initiale, par exemple en leur demandant de pointer un objet précis lorsqu'ils lancent l'application.

On peut aussi transmettre par OSC cette information de rotation issue de l'application qui s'exécute sur la tablette à une autre application tournant sur un ordinateur afin que le point de vue d'un autre utilisateur suive les mouvements du premier[2].

12.2.3. Déplacement

Concernant la question du déplacement, nous avons tout d'abord utilisé deux joysticks sur les côtés de l'écran, l'un permettant d'avancer, de reculer et de réaliser des mouvements de côté, tandis que l'autre contrôle l'élévation verticale. Mais la physique d'Unity ne permettant pas de monter sur place, cette action s'accompagne d'un mouvement horizontal qu'on ne peut que minimiser à défaut d'éliminer. En pratique, on constate que ce n'est toutefois pas gênant. À ce stade, on peut envisager des applications de réalité augmentée.

En intérieur, on pourrait afficher les informations relatives aux objets sur l' iPad lorsque celui-ci pointe dans la direction du dit objets. À l'extérieur, nous avons tenté de placer un marqueur de réalité virtuelle sur la façade de la chapelle, afin d'y projeter l'intérieur de la chapelle, pour donner une impression de transparence (voir Fig. 12.1 64).

Enfin, lors du dernier jour, nous avons effleuré une autre possibilité de l'iPad , qui pourrait donner une nouvelle dimension à la navigation. Il s'agit de récupérer les coordonnées Global Positioning System (GPS) de la tablette, de supposer qu'à l'échelle considérée, la Terre est plate donc que ces deux angles constituent en réalité un point dans le plan du sol. Les différents points obtenus au cours du temps permettent alors de faire suivre une trajectoire au visiteur qui correspond à ses déplacements réels. L'inconvénient est que la chapelle est un lieu fermé dans lequel on ne peut recevoir le signal GPS. Une des solutions qui nous a été proposée consiste à tapisser les murs de marqueur de réalité virtuelle, mais elle est manifestement trop invasive pour le lieu. Il a alors été envisagé de disposer des bornes wifi dans la chapelle afin de trianguler le signal de l'iPad . Néanmoins, l'approche du GPS s'est révélée utilisable dans la pelouse du LAMIC , où l'on peut ainsi faire surgir la chapelle et déambuler dedans.

2. C'est d'ailleurs ce qui est fait par l'intermédiaire des scripts `OSCServer.cs`, `OscSend.cd` et `OscReceive.cs`.

FIG. 12.1. : L'intérieur de la chapelle à sa position réelle affichée alors qu'on est encore à l'extérieur. En pratique, je subodore que l'effet manque d'impact à cause d'un problème d'échelle entre l'écran — tout *rétina* qu'il soit — de l'iPad et le bâtiment réel.

13. Immersion

Dernier axe de mon travail cet été, l'immersion regroupe l'ensemble des moyens mis en œuvre dans le but de faire oublier au visiteur qu'il n'est qu'en présence d'un modèle numérique, qui s'accompagne inévitable de déperdition d'informations par rapport à la réalité. Elle dépend en grande partie du dispositif d'affichage utilisé. Car même s'il n'est pas rare de croiser des gens totalement absorbés par l'écran de 15 cm² de leur téléphone[1], plus l'affichage se rapproche de la vision naturelle, plus l'impression de réalisme est renforcée. Nous allons donc étudier la mise en œuvre de ce principe à travers l'exemple du dôme et du panoscope avant de nous intéresser un autre sens, l'ouïe.

13.1. Dôme

Le dôme désigne la satosphère située à Montréal, demi sphère de 18 mètres de diamètre allant jusqu'au sol, éclairée par quatre vidéo-projecteurs et munie de 157 haut-parleurs qui permet une expérience immersive englobante. Le 20 juillet, nous avons pu bénéficier de plusieurs heures pour adapter et projeter les deux modèles plein et filaire dans cet environnement, en omettant la partie interactive. La nef de la chapelle mesurant 30 mètres de long, on est donc là à une échelle proche de la réalité, d'autant plus que la forme des coupoles épousent globalement celle de l'écran. Ce fut donc une expérience particulièrement concluante, malgré plusieurs soucis techniques détaillés en annexe ici. Elle aurait d'ailleurs pu être enrichie par une visualisation en relief, et ce sans modifier l'installation existante mais en utilisant des lunettes bicolores et en projetant un modèle anaglyphe[2]. Cela demeure néanmoins une installation particulièrement lourde et donc coûteuse qui ne permet pas de toucher un large public. Or, comme mentionné dans l'introduction, le LAMIC dispose d'un panoscope et nous avons donc cherché à nous en servir.

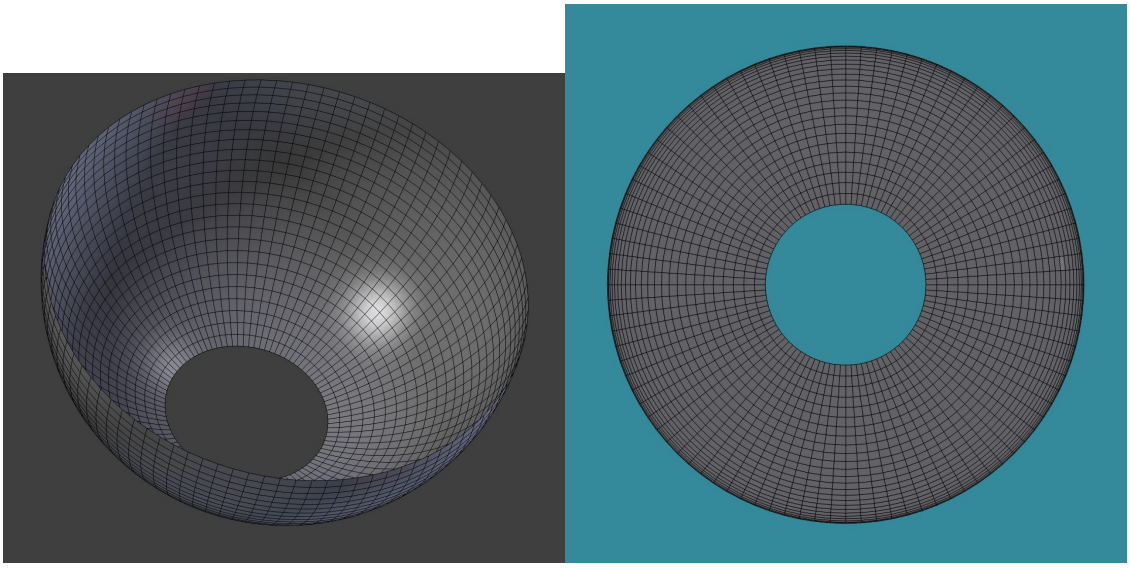
13.2. Panoscope

Le panoscope est donc une version plus petite et plus maniable du dôme mais renversée. C'est d'ailleurs son premier problème car il affiche beaucoup de plancher et prend moins bien en compte la géométrie des coupoles. Comme nous l'a expliqué Sébastien Roy, il « suffit » de projeter la géométrie de l'écran à plat et d'y plaquer une image qui va compenser les distorsions induites par la lentille. On peut donc réutiliser la méthode de l'UV mapping et c'est d'ailleurs ce qu'a fait la SAT . J'ai récupéré un projet Unity qui mettait cette technique à l'œuvre : six caméras attachées au personnage filmaient la scène vers la gauche, la droite, le haut, le bas, l'avant et l'arrière et plaçaient le résultat dans six *RenderTexture*. Ces textures étaient alors appliquées à la bonne position sur un cercle aplati représentant le dôme, qui était à son tour filmé par la caméra principale.

1. ce qui pourrait bien leur valoir une amende, notamment dans la riante bourgade de Fort Lee, New Jersey
2. même si ce faisant, nous aurions perdu toute information de couleurs.

FIG. 13.1. : Une vision assez représentative du panoscope du LAMIC , même si ce dernier est un peu plus resserré.

Dans un premier temps, j'ai simplement renversé cet anneau de caméras pour tenir compte du changement entre le dôme et le panoscope[3]. Cette solution de facilité n'a pas produit les résultats escomptés, sans doute parce que la projection de la géométrie aplatie du dôme ne correspond pas à celle du panoscope, qui présente par exemple un trou en son centre. J'ai donc sommairement modélisé le panoscope sous Blender (voir Fig. 13.2a p. 67) avant de refaire une nouvelle projection (Fig. 13.2b p. 67)mais s'est alors posé la question de savoir s'il fallait utiliser toute la hauteur des images filmées par les différentes caméras ou les couper à la moitié comme la SAT le fait. Malgré plusieurs tests, je n'ai jamais réussi à trancher et la projection n'a jamais atteint le niveau qu'elle avait lorsque Elsa a quitté le lamic, ce qui constitue une régression[4]. Un autre problème a été que l'obtention d'un résultat optimal nécessite d'utiliser la résolution native du projecteur, ce que Mac OS s'est refusé à faire avec un acharnement confinant à l'insolence, à moins d'acheter un programme dédié[5]. Cela nous a tout de même permis de réaliser des tests de navigation avec l'iPad .



(a) Le panoscope modélisé dans Blender.

(b) Le panoscope projeté à plat.

13.3. Ambiance sonore

Au-delà du rendu visuel, on peut aussi essayer de renforcer l'immersion par l'aspect sonore. Dans l'état actuel des choses, la visite s'accompagne d'un son d'orgue de huit minutes enregistré dans la chapelle. Deux idées ont été évoquées pour enrichir cette ambiance, même si nous n'avons pas eu le temps d'y donner suite. M. Vesac a proposé de capturer un flux sonore produit dans la chapelle en temps réel et de le retransmettre au visiteur du modèle numérique. Toutefois la chapelle n'est pas encore équipée pour assurer cette connexion. Une autre idée a été proposée par Mme Marjolaine Boutin, étudiante en maîtrise d'ethnologie et qui a travaillé à l'inventaire du patrimoine immatériel religieux du Québec. Il s'agissait de rendre compte des sons et des silences émis lors des diverses cérémonies par les sœurs fréquentant la chapelle, ce qui n'est pas sans demander un certain sens de la mesure.

3. ces changements de configuration étant géré par le script `OutputSetup.cs`

4. qu'on pourrait justifier par le fait qu'elle à réutilisé le code de la SAT , ce qui ne serait guère satisfaisant dans la mesure où j'ai fait de même. Donc pas de justification.

5. pour changer une résolution ! C'est décevant de la part d'une société qui a élevé la simplicité d'utilisation au rang de vertu cardinale.

Quatrième partie .

Conclusion

14. Perspectives

Lors de l'avant-dernier jour de notre stage, Camille et moi avons effectué une présentation de notre travail devant plusieurs professeurs de l'université ainsi que des représentants des Ursulines et de leur musée. C'était une occasion de faire le bilan sur le travail accompli et de dégager de nouvelles directions pour les années à venir. Il en est ressorti que le modèle avait atteint une qualité visuelle satisfaisante pour une utilisation muséographique qui n'exige pas un rendu photoréaliste[1]. En revanche, les interactions pourraient exploiter de nouveaux concepts pour recentrer l'expérience de visite sur l'utilisateur.

14.1. Nouveaux dispositifs

Une Kinect de Microsoft pourrait être employée afin de reconnaître les gestes de l'utilisateur indiquant son souhait de se déplacer ou d'interagir avec certains éléments du modèle. Plus coûteux, on pourrait envisager l'utilisation d'un *Head-Mounted Display* (HMD) pour l'affichage immersif et le suivi précis du regard. Ou se contenter d'un écran 3D assez grand, appareils qui sont en ce moment même en voie de généralisation et à même de toucher une grande partie du public dans les années à venir. Cela suppose néanmoins d'adapter le contenu à ces nouvelles formes de visualisation et de développer des modalités d'interaction muséographique qui ont pour l'instant été peu étudiées.

14.2. Utilisation des cinq sens

Deux des cinq sens sont déjà pris en compte dans notre modèle. Mais s'il paraît a priori assez improbable d'avoir à stimuler le goût du visiteur pour rendre que son expérience plus réaliste, ce n'est pas le cas de l'odorat et du toucher. On peut imaginer qu'une odeur d'encens ou de meubles anciens contribue à l'impact de la visite. Il en serait de même si l'on pouvait éprouver physiquement le poids et la forme des objets considérés. Cela reste néanmoins une perspective assez lointaine, car les dispositifs haptiques et olfactifs sont encore assez confidentiels pour ne pas dire ésotériques.

14.3. Personnalisation de la visite

Actuellement, l'ensemble des objets est uniformément accessible et il n'y a qu'un seul fond sonore, ce qui reflète de manière peut-être trop rigide les choix des concepteurs[2]. Comme l'a avance l'un

1. Un moyen coûteux en temps et en énergie de s'en approcher serait de retravailler l'éclairage en l'éliminant des photos qui servent de textures, en mesurant la radiosité de chaque surface, et en calculant le nouvel éclairage en temps réel.

2. Eh oui, en cette fin de rapport, j'estime avoir bien mérité le droit de parler de moi à la troisième personne.

des participant à la réunion, M. Milad Doueihi[3], l'expérience serait sans doute plus riche si le visiteur pouvait influencer l'ambiance, que ce soit par le choix de la musique jouée lors de sa visite, le moment de la journée ou de la nuit auquel elle a lieu, la catégorie des objets à afficher ou encore son mode de déplacement. Certains préféraient une visite guidée de façon plus ou moins stricte tandis que d'autres goûteraient mieux le fait de pouvoir voler dans les coupoles. Ce souhait de placer la liberté du visiteur au cœur de l'expérience apparaît certes louable mais ne doit pas conduire à tous les débordements. C'est d'ailleurs l'un des points qu'ont soulevés les sœurs, car elles demeurent très attachées à la transmission de valeurs spirituelles, même à travers un modèle numérique.

3. historien des religions et titulaire de la Chaire de recherche sur les cultures numériques à l'Université Laval

15. Apports personnels

J'ai déjà eu l'occasion de le dire dans mes remerciements, mais ce stage fut pour moi une expérience des plus enrichissantes, et j'aimerais à présent préciser ma pensée en explicitant ce que j'en ai retiré.

Tout d'abord, cela a été l'occasion de mettre en œuvre des connaissances et des compétences acquises lors de cours à Centrale au sein d'un cadre applicatif, notamment en termes de base de données, de technologies web, de traitement de l'image ou d'ingénierie virtuel. J'ai ainsi eu le plaisir d'écrire du code en Python, en PHP, en Javascript et en SQL, ainsi que la joie de découvrir un nouveau langage, le C#[1]. Au-delà de l'aspect purement programmation, j'ai pu renoué avec la pratique de Blender , un des premiers logiciels non ludiques auquel je me suis consacré étant jeune et qui a joué un rôle majeur dans mon attrait pour l'informatique. Surtout, l'utilisation intensive d'Unity m'a permis de me familiariser avec ce logiciel qui est de plus en plus utilisé, que ce soit dans le domaine professionnel ou académique.

C'est aussi un des projets les plus longs que j'ai jamais mené à terme, ce qui constitue une satisfaction personnelle. D'autant que les conditions de travail du laboratoire nous ont imposé une grande autonomie qui, si elle fut parfois un tantinet décourageante, s'est aussi révélée très formatrice et je ne doute pas de pouvoir mettre cette expérience à profit dans l'avenir.

De plus, effectuer ce stage dans une université étrangère m'a permis de découvrir une autre atmosphère et une autre organisation du travail. D'abord parce qu'il s'agit d'un laboratoire de recherche avec ce que cela implique en matière de clients et d'horizon temporel, différent de ce qu'on peut voir dans l'industrie. Mais aussi parce que même s'il est basé à Québec, l'environnement est imprégné d'un état d'esprit nord américain qui met l'accent sur la capacité de pouvoir démontrer la viabilité du projet à tous moments afin de convaincre des partenaires potentiels.

Enfin j'ai trouvé le contexte muséographique et patrimonial de ce stage plutôt rafraîchissant. Il n'était en effet pas question d'optimiser un processus industriel pour améliorer ses performances économiques ou écologiques mais bien d'imaginer des solutions permettant de préserver et de diffuser l'essence d'un lieu marquant de l'histoire du Québec. Il m'a semblé que c'était là un défi certes moins rémunérateur mais tout aussi excitant[2].

1. qui s'est révélé agréable à utiliser et dont la bibliothèque standard est très complète, malgré les réserves que peuvent inspirer son entreprise fondatrice : Microsoft

2. je ne peux toutefois écarter que cette vision romantique ne soit en partie due au fait que ma famille a financé mon séjour, ce qui me fait penser que j'ai oublié de la remercier au début du rapport.

Bibliographie

[1] Rodrigo Barreto de Oliveira, *The boo Programming Language*, 29 juin 2006, url : http://boo.codehaus.org/BooManifesto.pdf.

[2] Pierrot Deseilligny et Isabelle Clery, « APERO, an Open Source Bundle Adjusment Software for Automatic Calibration and Orientation of a Set of Images », dans : *Proceedings of the ISPRS Commission V Symposium*, éd. de Remote Sensing and Spatial Information Sciences International Archives of the Photogrammetry, 2011, url : http://recherche.ign.fr/labos/matis/pdf/articles_conf/2011/pierrot_clery.pdf.

[3] Christopher Evans, *Notes on the OpenSURF Library*, rap. tech. CSTR-09-001, University of Bristol, jan. 2009, url : http://www.chrisevansdev.com.

[4] Elsa Goarrin, *De la numérisation à l'interaction, modèle 3D de la chapelle intérieure du monastère des Ursulines de Québec*, rapport de stage ingénieur, École Centrale de Nantes, 7 sept. 2011.

[5] Richard Szeliski, *Computer Vision : Algorithms and Applications*, éd. de Springer, 2010, url : http://szeliski.org/Book/drafts/SzeliskiBook_20100903_draft.pdf (visité le 03/09/2010).

[6] Maarten Vergauwen et Luc Van Gool, « Web-based 3D Reconstruction Service », dans : *Mach. Vision Appl.* 17.6 (2006), p. 411–426, issn : 0932-8092, doi : http://dx.doi.org/10.1007/s00138-006-0027-1.

Ce stage ayant eu une vocation essentiellement pratique, nous avons pas énormement eu l'occassion de lire de publications techniques.

A. Manipulation de la base de données

A.1. Exporter la base de données

- ouvrir la table `unity` dans phpmyadmin et utiliser la méthode d'exportation rapide avec un minimum d'option qui convient parfaitement dans ce cas.
- copier le répertoire `uploads` qui doit être situé là où sont tous les scripts php.

A.2. Importer la base de données

- ouvrir phpmyadmin et supprimer la table `unity` si elle existe déjà. Créer ensuite une table `unity` avec le moteur innoDB et l'interclassement `utf8_unicode_ci`. La sélectionner puis aller dans importer et choisir le fichier créé précédemment.
- copier le répertoire `uploads` au même endroit.

A.3. Télécharger le contenu « lourd » dans unity

La première fois, il faut commencer par installer certains logiciels :

- Python 2
- setuptools : télécharger et installer ce fichier pour windows. Pour mac, il faut télécharger celui-là avant de lancer dans un terminal `sh setuptools 0.6c11 py2.7.egg`
- pip : `easy_install pip`
- MySQL Connector/Python : `pip install mysql connector python`

Il faut ensuite récupérer `mycon.py` et l'ouvrir avec un éditeur de texte et modifier éventuellement les lignes `SERVER_URL` et `UNITY_DIR` avant de lancer `python mycon.py NOMDELOBJET` pour télécharger les sons et les vidéos relatifs à cet objet.

B. Visite du 20 juillet à la SAT

B.1. Situation

Jeudi 19 juillet a été archivée sur une clé USB une version représentant les derniers avancements du projet Unity contenant le modèle de la chapelle ainsi que les premières interactions associées. Cette archive a ensuite été transférée sur l'ordinateur de M. Vesac où elle s'est exécutée sans qu'aucun nouveau problème ne se fasse jour.

Cette version a été réalisée pour être testée sur les ordinateurs de la SAT à Montréal, qui différent par leur système de projection (satosphère) et de contrôle (par iPhone, car les machines sont trop éloignées de l'affichage), qui a lui révélé plusieurs soucis spécifiques.

B.2. Déroulement de la journée

- 10h00 : première déconvenue, après avoir importé le projet, aucun modèle ne s'affiche à l'écran. Blender étant installé sur la machine, il paraît raisonnable de penser qu'il s'agit d'un problème lors de l'export vers le format fbx que Unity réalise automatiquement. Après avoir mis à jour Blender , un export manuel confirme cette hypothèse. Cette fois, le modèle est visible mais les animations ne sont plus fonctionnelles. Surtout il faudrait réimporter tous les modèles à la main[1].
- 10h15 : Pour tenir compte des particularités de la SAT , il faut importer deux packages Unity supplémentaires `OscUnity` et `CameraRig2012`. Ce dernier provoque malheureusement une erreur de compilation car certains scripts dépendent d'autres qui ne sont pas présents. Nous essayons alors d'installer le paquet manquant `M2H Culling`, ce qui nécessite la création d'un compte et une connexion internet dont nous ne disposons pas. Il faut alors descendre d'un étage pour récupérer un câble réseau, ce qui nous permet de rencontrer Sébastien Gravel, qui a créé ces packages.
- 10h30 : Unity ne délivre les erreurs qu'au compte-gouttes, donc l'installation de `M2H Culling` ne sert qu'à faire apparaitre de nouvelles erreurs en relation avec le paquet `OscUnity`. Même Sébastien ne parvient pas à les régler et je décide dans un premier temps de supprimer tout le paquet pour me concentrer sur la caméra.
- 10h40 : La caméra affiche l'image attendue mais elle ne se déplace pas avec le personnage. Or le script responsable de ce comportement ne fait qu'une ligne donc peut difficilement être erroné. Il est donc probable que l'erreur se terre dans un endroit plus reculé. Il parait alors plus simple d'importer la scène de départ de la SAT et d'y placer nos objets. Néanmoins, même cette solution échoue, encore pour des problèmes de dépendances.

1. ce qui est réalisable, mais un peu long étant donné le retard déjà accumulé.

- 10h55 : Il apparait plus raisonnable d'importer notre modèle dans un projet vide (donc fonctionnel). Cela s'avère assez long car il faut exporter près de 200 Mo de données avant de les réimporter juste après. Surtout, on y perd toutes les interactions car elles étaient basées sur une ancienne version du script `Pointer.cs` dont les noms de fonctions ont changé.
- 11h30 : Les modèles filaire et complet sont disponibles. Sans les portes pour le second, car leur ouverture dépend de la présence d'un personnage particulier qui n'est pas présent dans la scène de la SAT . J'essaye de faire fonctionner à nouveau les scripts qui ajoutent le mobilier mais je m'emmêle et chaque banc ajouté essaye à son tour d'en créer dix autres derrière lui, ce qui finit par faire planter Unity , à court de mémoire.
- 11h45 : Finalement, une version compilée est disponible, et Luc Courschene nous informe que nous disposons encore de la satosphère.
- 12h00 : Indépendamment de nous, les projecteurs de la SAT semblent récalcitrants, notamment à cause de la résolution choisie, et ils se figent ou projettent trois images décalées. Mais les techniciens parviennent à résoudre ces soucis.
- 12h20 : Enfin, la chapelle s'affiche dans toute sa splendeur mais malheureusement, nous sommes coincés dans un mur. Il faut donc descendre compiler une version où la position de départ du personnage est plus pertinente.
- 13h05 : M. Vesac estime qu'il faut compiler une nouvelle version pour observer le rendu du modèle filaire.
- 13h10 : le modèle filaire fonctionne du premier coup.
- 13h40 : Nous quittons la satosphère.

B.3. Conclusion

Il aurait sans doute été plus simple de travailler en plus étroite collaboration avec la SAT afin de s'assurer d'avoir la dernière version de leur paquet Unity (ainsi que de leur application iPhone).

Index

B
Blender, 10, 26, 27, 29, 32, 40, 43, 49, 50, 67, 71, 74

C
Camille, 10, 12, 19, 26, 27, 29, 31, 32, 39, 42, 43, 50, 69, 78
chapelle, 10, 18, 19, 21, 23, 2527, 29, 32, 3943, 52, 60, 6365, 67, 74, 75

I
iPad, 6264, 67

L
LAMIC, 10, 12, 15, 22, 33, 54, 63, 65, 66, 91

N
nuage de points, 21, 22, 32, 33, 36, 39, 40

O
OSC, 22, 63

P
panoscope , 15, 16, 22, 47, 6567
Polyworks, 21, 22, 33, 40

S
sanctuaire, 19, 29, 39, 40, 42
SAT, 10, 12, 15, 23, 62, 65, 67, 74, 75

T
tranche, 22, 2528

U
Unity, 11, 22, 29, 32, 39, 40, 42, 4750, 55, 58, 60, 62, 63, 65, 71, 74, 75

V
visiteur, 10, 29, 44, 48, 50, 52, 54, 55, 62, 63, 65, 67, 69, 70

Glossaire

GPS Global Positioning System. 63

HMD *Head-Mounted Display*. 69

LAMIC Laboratoire de Muséologie et d'Ingénierie de la Culture. 10, 12, 15, 16, 22, 33, 54, 63, 65

LVSN Laboratoire de Vision et Systèmes Numériques. 21, 22

marqueur de réalité virtuelle Image de référence placée dans l'environnement réel et qu'une application de réalité augmentée peut exploiter pour afficher du contenu si elle est dans son champ de vision. 63

muséographie Étude théorique et pratique de l'exposition d'objets culturels. 10

NTIC Nouvelles Technologies de l'Information et de la Communication. 10

OSC Open Sound Control. 22, 63

photogrammétrie Technique consistant à obtenir un modèle d'un objet à partir de photos de celui-ci. 12

SAT Société des arts technologiques. 10, 12, 15, 16, 23, 62, 65, 74, 75

tracéographie Discipline scientifique liée à l'archéologie qui a pour but de déterminer la fonction des outils par l'étude des traces produites lors de leur utilisation. 15

ursuline Ordre de religieuses qui se consacre essentiellement à l'éducation des jeunes filles et qui est présent en France, Belgique, Allemagne et Canada. 18

C. Rapport de Camille sur les techniques de modélisation 3D

Modélisation 3D

La modélisation 3D d'objets existants plus ou moins grands se fait la plupart du temps par captation 3D créant un nuage de points exploitable, soit directement, soit par triangulation automatique, soit par retopologie. Voici un court résumé de différents modes de captation. Nous verrons également que des reconstructions se font directement à partir de plans plutôt que d'objets réels.

Table des matières

1 SCANNERS LASER 2

1.1 SCANNER PAR TRIANGULATION 2
1.1.1 PRINCIPE 2
1.1.2 MARCHE ET ACTEURS PRINCIPAUX 3
1.1.3 AVANTAGES ET INCONVENIENTS 3
1.2 SCANNER PAR TEMPS DE VOL 3
1.2.1 PRINCIPE 3
1.2.2 MARCHE ET ACTEURS PRINCIPAUX 3
1.2.3 AVANTAGES ET INCONVENIENTS 3
1.3 HANDYSCAN 5
1.3.1 PRINCIPE 5
1.3.2 MARCHE ET ACTEURS PRINCIPAUX 5
1.4 HOLOGRAPHIE CONOSCOPIQUE 5
1.4.1 MARCHE ET ACTEURS PRINCIPAUX **ERREUR ! SIGNET NON DEFINI.**
1.4.2 AVANTAGES ET INCONVENIENTS 5

2 PHOTOGRAMMETRIE ET AUTRES SYSTEMES PASSIFS 6

2.1 PRINCIPE 6
2.2 MARCHE ET ACTEURS PRINCIPAUX 6
2.3 AVANTAGES ET INCONVENIENTS 7

3 LUMIERE STRUCTUREE 8

3.1 PRINCIPE 8
3.2 AVANTAGES ET INCONVENIENTS 9

4 RECONSTRUCTION SUR PLAN 10

5 SOURCES 11

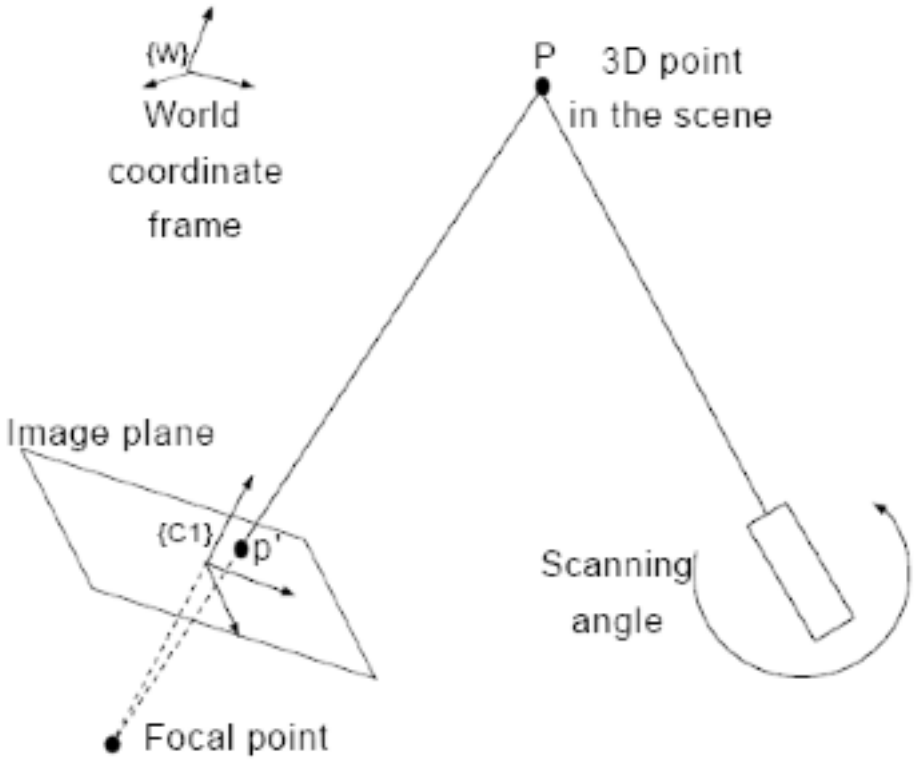
Le but est ici de mesurer le positionnement d'une partie des points de la surface d'un objet dans un système de coordonnées fixé. La forme de l'objet sera ensuite extrapolée à partir de la répartition de ces points. Il existe plusieurs méthodes pour obtenir ce nuage de points, les plus importantes étant les scans laser et les scans par lumière structurée ou modulée en ce qui concerne les scanners dits actifs, et la photogrammétrie en ce qui concerne les scanners dits passifs.

1 Scanners laser

Bien qu'ils aient tous pour point commun d'envoyer un rayon laser pour sonder leur environnement, il existe beaucoup de modes de captation différents dans cette catégorie.

1.1 Scanner par triangulation

1.1.1 Principe



Ce scanner pointe son faisceau laser vers l'objet et utilise un appareil photo pour situer le point, qui apparaît à un endroit différent du champ de vision selon la distance du scanner à la surface. Le triangle appareil-laser-point P (voir figure ci-dessus) est parfaitement déterminable : la longueur appareil-laser et l'angle sur le laser sont connus, et l'angle sur l'appareil est mesurable. On peut donc situer le point P.

Souvent, une bande laser plutôt qu'un point balaie l'objet dans un souci de vitesse.

1.1.2 Marché et acteurs principaux

Le Conseil National de recherches Canada a été parmi les premiers instituts à développer une telle technologie, en 1978.

Ce type de scanner est adapté pour scanner de grands volumes peu éloignés ou des objets de petites tailles.

D'une manière générale les scanners laser s'appliquent à la modélisation et à la mesure 3D.

1.1.3 Avantages et inconvénients

La portée des scanners par triangulation est assez faible (quelques mètres), mais leur précision est excellente : de l'ordre du dixième de micromètre.

D'une manière générale les scanners laser font preuve, comparés aux scanners passifs :

- D'une reconstruction dense, haute résolution, grande précision
- D'un temps d'acquisition plus long qu'en stéréovision (balayage laser mécanique)
- Ils ont le défaut d'être intrusifs.

1.2 Scanner par temps de vol

1.2.1 Principe

Ce type de scanner, apparu au début des années 2000, possède un télémètre laser, qui permet de calculer la distance d'un point au laser en fonction du temps que prend l'objet pour réfléchir l'impulsion d'un faisceau laser émis par le scanner.

1.2.2 Marché et acteurs principaux

Le scanner par temps de vol est surtout adapté pour scanner de grands volumes éloignés (architecture, archeo, etc). Il est donc principalement utilisé par les acteurs de :

- La topographie (levé, conception, contrôle, calcul des volumes...),
- L'industrie (saisie d'environnements complexes...),
- L'architecture (plan de façade, relevé d'intérieur, coupe, modélisation...)

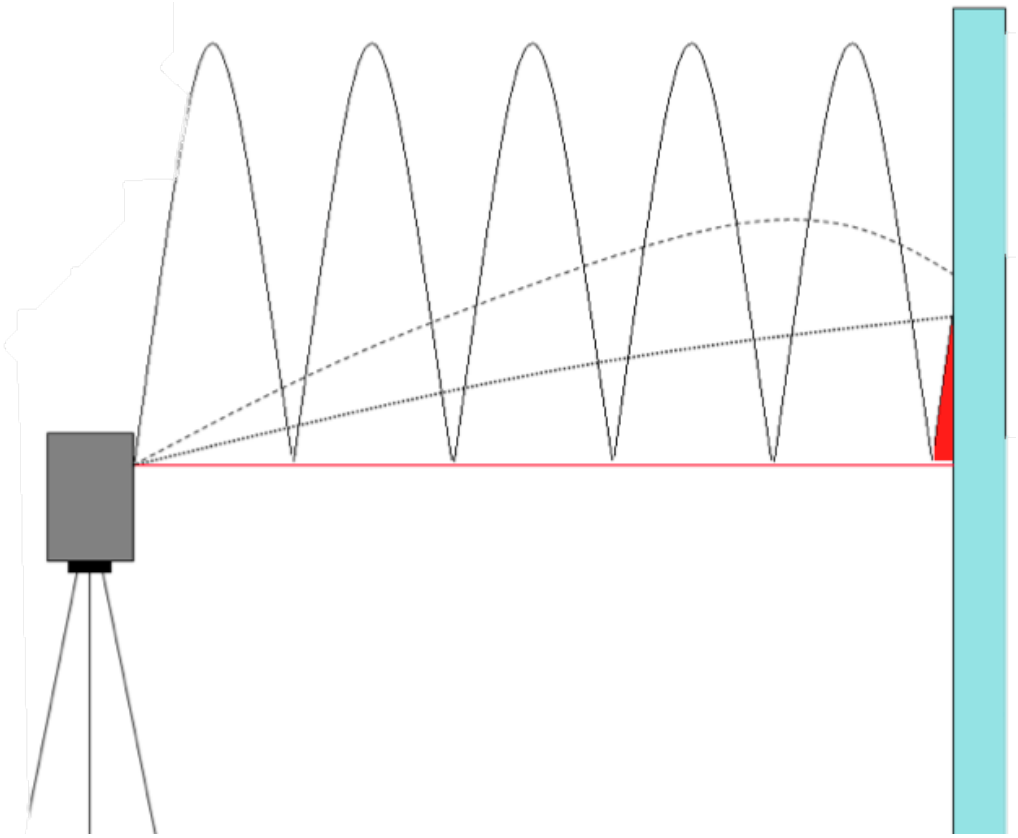
1.2.3 Avantages et inconvénients

Les scanners lasers 3D « temps de vol » permettent de couvrir un champ angulaire plus important que les scanners par triangulation, tout en gagnant en portée (jusqu'à plusieurs centaines de mètres), et en vitesse (il calcule la distance de 10 000 à 100 000 points par seconde).

La majeure faiblesse des scanners par temps de vol est leur imprécision. En effet, vu les vitesse concernées, chronométrer l'aller-retour du signal n'est pas chose facile et la précision se joue à quelques millimètres.

1.3 Scanners à mesure de différence de phase

1.3.1 Principe



En anglais « phase-shifted method », cette technologie utilise un faisceau laser continu, de forte intensité et modulé de façon sinusoïdale (modulation généralement basée sur l'amplitude ou sur la fréquence). L'onde envoyée du scanner sur la surface de l'objet est renvoyée par la surface vers le scanner qui calcule la distance à la surface en mesurant, cette fois, la différence de la phase de l'onde. Le scanner qui a servi pour capter la chapelle des Ursulines de Québec est un scanner Zoller & Frolich (ZF) modèle 5006 fonctionnant selon ce principe.

1.3.2 Marchés et acteurs

Sur le marché d'aujourd'hui, dans le domaine du bâtiment surtout, le scanner à mesure de différence de phase est le type de scanner le plus utilisé.

1.3.3 Avantages et inconvénients

Les scanners à mesure de différence de phase sont beaucoup plus rapides que les scanners à temps de vol, environ 500 000 à 1 200 000 points par seconde peuvent être captés. Ils sont aussi beaucoup plus précis, parfois de l'ordre du millimètre. En contrepartie, leur portée est plus courte en général que les scanners à temps de vol, de l'ordre de 50 à 150 mètres en moyenne contre 300 à 2000 mètres pour les scanner à temps de vol.

1.4 Handyscan

1.4.1 Principe

Le principe est souvent le même que pour les scanner par triangulation : un laser et un capteur (souvent « Position sensitive device »). Le référentiel est interne à l'appareil, on doit calculer en plus sa position propre dans un référentiel fixe. Cela est fait grâce à des points de repère (typiquement des pastilles blanches). Ou par d'autres méthodes grâce à des machines à mesurer tridimensionnelles.

1.4.2 Marché et acteurs principaux

Le scanner à main convient aux objets de petites tailles : de la pièce mécanique à l'objet archéologique. Il est très emploé en rétro-ingénierie et en métrologie. Creaform, à Québec, commercialise des handyscan divers très performants.

1.5 Holographie conoscopique

1.5.1 Principe

Dans un système conoscopique un rayon laser est projeté sur une surface, puis la réflexion au travers du même faisceau passe à travers un cristal biréfringent et est envoyé sur un capteur CDD. La fréquence des motifs de diffraction peut être analysée et permet de déterminer la distance par rapport à cette surface.

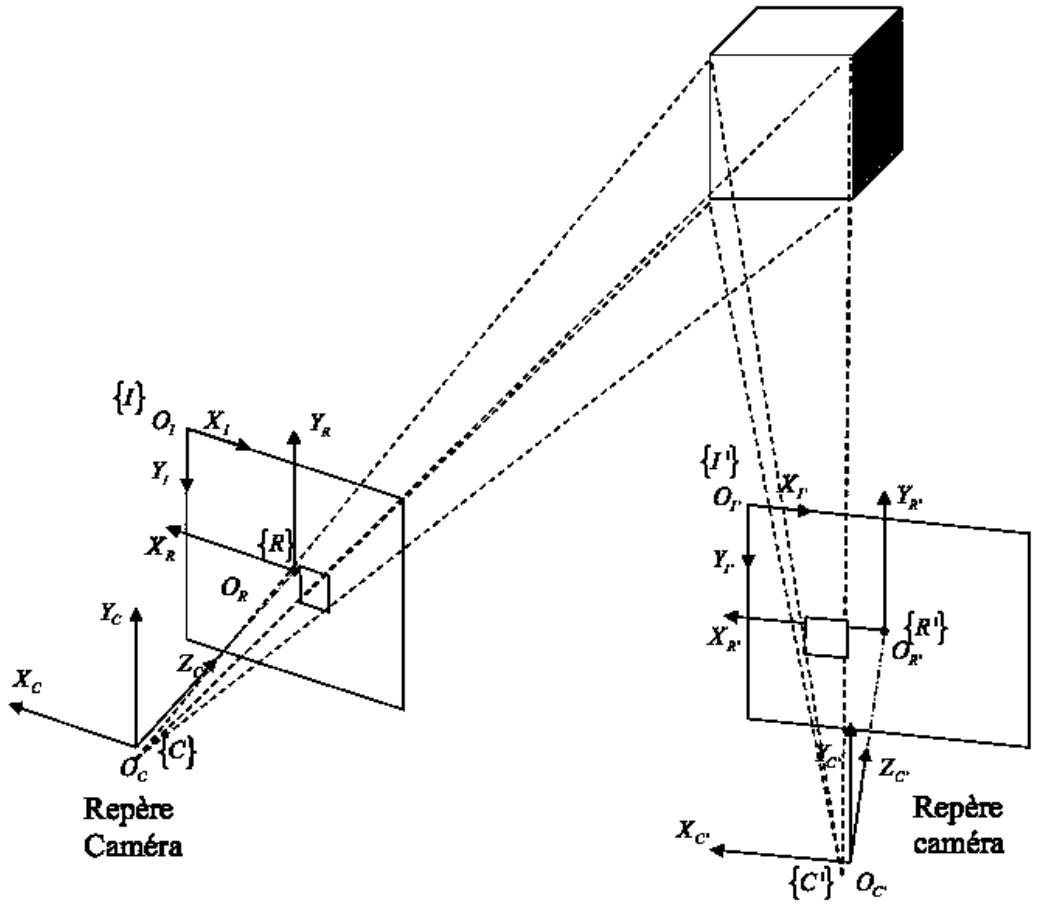
1.5.2 Avantages et inconvénients

L'atout principal de l'holographie conoscopique est qu'un unique faisceau (aller-retour) est nécessaire pour effectuer la mesure, permettant de mesurer par exemple la profondeur d'un trou finement percé.

2 Photogrammétrie et autres systèmes passifs

2.1 Principe

La photogrammétrie s'appuie sur la vision stéréoscopique humaine. Avec plus clichés de différents points de vue d'un même objet, par reconnaissance de couleurs, d'entourage,…, on peut déterminer la position des points. Dans la figure ci-dessous, on peut voir qu'un sommet du cube est repéré en comparant les deux projections sur les champs de vision des deux caméras ou des deux clichés. S'il s'agit de clichés, il faut également déterminer la position de la caméra d'un cliché à l'autre.



Dans la même veine on peut également citer les scanners stéréoscopiques, qui utilisent deux caméras vidéos et se basent sur la vision stéréoscopique de l'œil humain. Il s'agit donc, en somme, d'une captation 2D par la suite convertie en image 3D, par traitement d'images.

2.2 Marché et acteurs principaux

La photogrammétrie s'applique :

- A la robotique mobile
- A la métrologie industrielle, parfois. Mais pour avoir la précision nécessaire, le procédé est particulièrement long et laborieux. Pourtant, grâce à de récents progrès

dans le domaine de la stéréoscopie, l'emploi de la *prise de vue convergente* se développe rapidement.

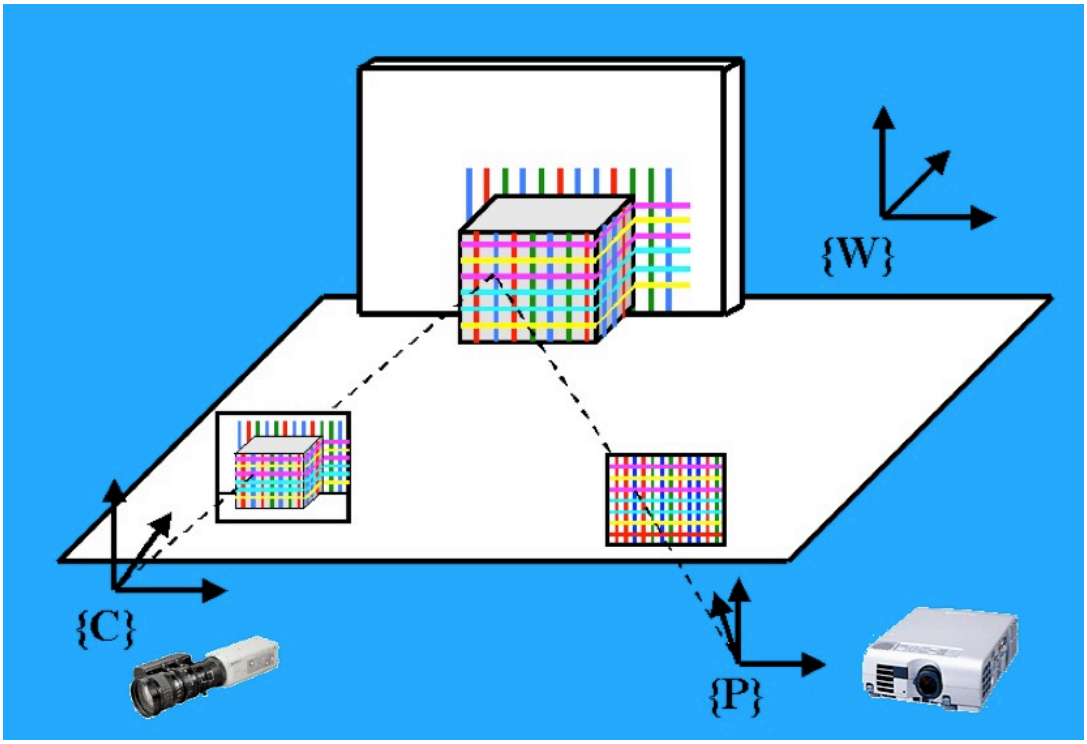
- A la cartographie : les photogrammétries aérienne et terrestre se développe de plus en plus. Ce genre de méthodes est aussi cohérent pour la topographie, l'architecture, l'archéologie, l'investigation policière, la géologie,…

2.3 Avantages et inconvénients

La méthode est non intrusive, relativement simple et plutôt efficace. Malgré tout, la densité de points reste assez faible (une reconstruction dense est possible, mais complexe et particulièrement gourmande en temps de calcul) et donc la résolution également. Il reste d'autre part des problèmes sur la mise en correspondance et les traitements associés.

3 Lumière structurée

3.1 Principe



Les scanners 3D à lumière structurée projettent un motif lumineux sur le sujet et en observent la déformation. Le motif peut être à une ou deux dimensions (typiquement des lignes ou des carreaux). Il est projeté sur le sujet à l'aide d'un vidéoprojecteur LCD ou laser. Une caméra légèrement décalée du projecteur est utilisée pour enregistrer les déformations et un programme informatique complexe sert à calculer les distances des points composant ce motif. La complexité est due à l'ambigüité. Prenons un groupe de bandes verticales balayant horizontalement un sujet. Dans le cas le plus simple, l'analyse repose sur la présomption que la séquence des bandes visibles de gauche à droite correspond à celle de l'image laser projetée, de tel sorte que l'image de la bande la plus à gauche est bien la première de la projection du laser, la suivante est le seconde et ainsi de suite. Cependant, dans le cas de cibles non-triviales comportant des trous, des occlusions, ou des changements de profondeur rapide, l'ordre n'est plus forcément vérifié du fait que des bandes sont souvent masquées et peuvent même apparaitre dans un ordre différent, donnant lieu à une ambigüité des bandes lasers. Ce problème spécifique a récemment été résolu par une avancée technologique appelé "Multistripe laser Triangulation (MLT)" (en: Triangulation laser multibande). Le scan 3D à lumière structurée est encore un domaine de recherche actif donnant lieu à de nombreuses publications chaque année.

Il existe également des scanners à lumière modulée : la source a un cycle, et une caméra détecte la lumière réfléchie, mesure la variation de l'amplitude, et calcule ainsi la distance que la lumière a parcourue. Cette méthode évite les interférences avec d'autres sources de lumière.

3.2 Avantages et inconvénients

Cette technologie peut être considérée comme un bon compromis entre les deux précédentes. La densité de points est en général plus haute qu'en stéréovision, mais la précision un peu plus faible que les scanners lasers.

Le point fort des scanners 3D à lumière structurée est leur rapidité. Au lieu de scanner un point à la fois, ils scannent tout le champ de vision à la fois. Ceci limite ou élimine les problèmes de distorsion liés au mouvement. Des systèmes existant sont capables de scanner des objets en mouvement en temps réel. Récemment, Song Zhang et Peisen Huang de l'Université de Stony Brook ont mis au point un scanner à la volée utilisant une projection d'une frange numérique et une technique à phase modulée (une autre méthode à lumière structurée). Ce système est capable de saisir, de reconstruire et de restituer les détails d'objets se déformant dans le temps (comme une expression faciale) à une fréquence de 40 images par secondes.

Conclusion

En plus de ces types d'acquisitions, il existe d'autres méthodes de scan 3D, mais peu utiles dans le cadre de la modélisation. Par exemple la palpation est bien plus adaptée à la métrologie.

Certains types d'objets restent toutefois encore difficile à numériser : par exemple les appareils utilisant des technologies optiques rencontrent beaucoup de problèmes avec des objets brillants, réfléchissant ou transparents.

Il existe cependant des méthodes permettant de scanner les objets brillants, par exemple en les recouvrant d'une fine couche de poudre blanche qui permettra à plus de photon de se réfléchir et de parvenir à l'optique du scanner. Les scanners laser peuvent envoyer des trillions de photons vers un objet et en retour recevoir un petit pourcentage de ces photons par les optiques qu'ils utilisent. La réflectivité d'un objet dans le visible est fonction de sa couleur, c'est l'albédo. Une surface blanche reflètera beaucoup de lumière en retour et une surface noire en réfléchira seulement une petite quantité. Les objets transparents comme le verre vont seulement réfracter de la lumière et donner de fausses informations sur les trois dimensions.

Une autre méthode serait de recouvrir d'un filet l'objet.

Si l'on ne peut pas effectuer ce genre d'opérations sur l'objet, il reste la modélisation à partir de plans uniquement.

4 Reconstruction sur plan

Malgré toutes ces technologies, on continue, particulièrement en architecture ou en mécanique, à modéliser sur plan par CAO.

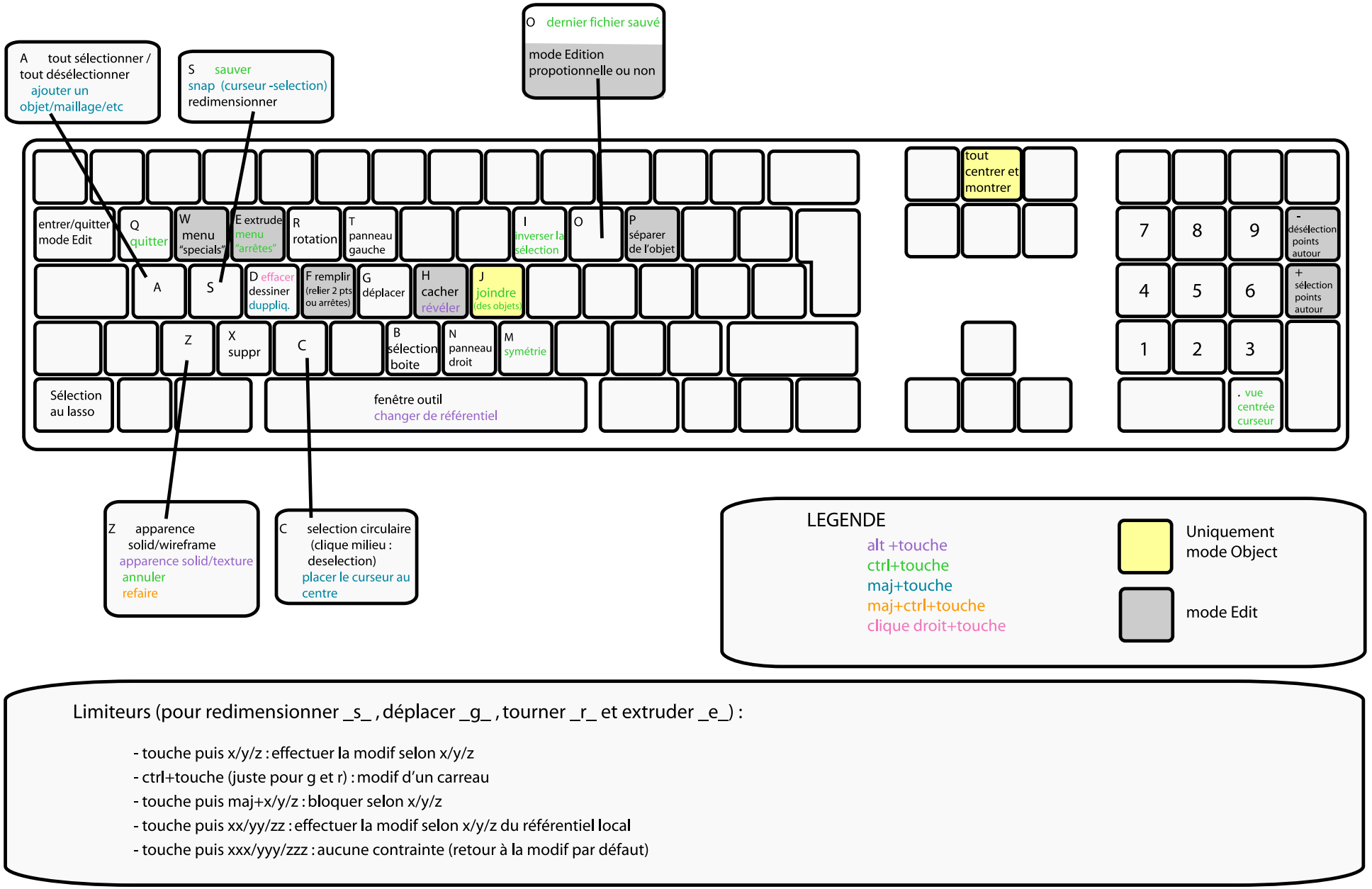
En architecture et en design, ce procédé permet de prévisualiser de façon particulièrement réaliste à la fois l'intérieur et l'extérieur d'un bâtiment parfois non-existant. Des logiciels aujourd'hui particulièrement performants comme Kinetix 3D Studio Viz, ou Softimage 3D, et des outils CAD (comme Autocad) permettent de construire des modèles 3D à partir de plans au sol 2D. Malgré tout la méthode n'est pas entièrement automatique et une étape manuelle demeure indispensable.

Les tâches des logiciels semi-automatiques se décomposent en 3 étapes :

- L'extrusion des murs : convertir les dessins 2D des structures murales en murs 3D.
- Le mapping des objets particuliers : placer des objets 3D et des éléments spécifiés dans les plans
- La reconstruction des plafonds et sols, qui requiert une méthode pour détourer le périmètre du sol des plans 2D pour prendre en compte aussi bien les régions convexes que les trous dans les plans originaux.

5 Sources

- MGC3D
- Creaform
- SHS3D
- Numérisation-3D.info
- Site de l'ENSG (Ecole Nationale des Sciences Géographiques)
- Map.archi.fr
- Site du Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier : http://www2.lirmm.fr/~strauss/PageImage3//Fofi_Vision3D.pdf
- Planete-tp.com : Association pour la connaissances des Travaux Publiques (France)
- « Reconstruction of 3D virtual buildings from 2D architectural floor plans », Clifford So, George Baciu, Hangiu Sun de l'Université de Hong Kong
- Wikipedia



Limiteurs (pour redimensionner _s_ , déplacer _g_ , tourner _r_ et extruder _e_) :

- touche puis x/y/z : effectuer la modif selon x/y/z
- ctrl+touche (juste pour g et r) : modif d'un carreau
- touche puis maj+x/y/z : bloquer selon x/y/z
- touche puis xx/yy/zz : effectuer la modif selon x/y/z du référentiel local
- touche puis xxx/yyy/zzz : aucune contrainte (retour à la modif par défaut)

FIG. C.1. : Raccourcis clavier utiles dans Blender 2.63

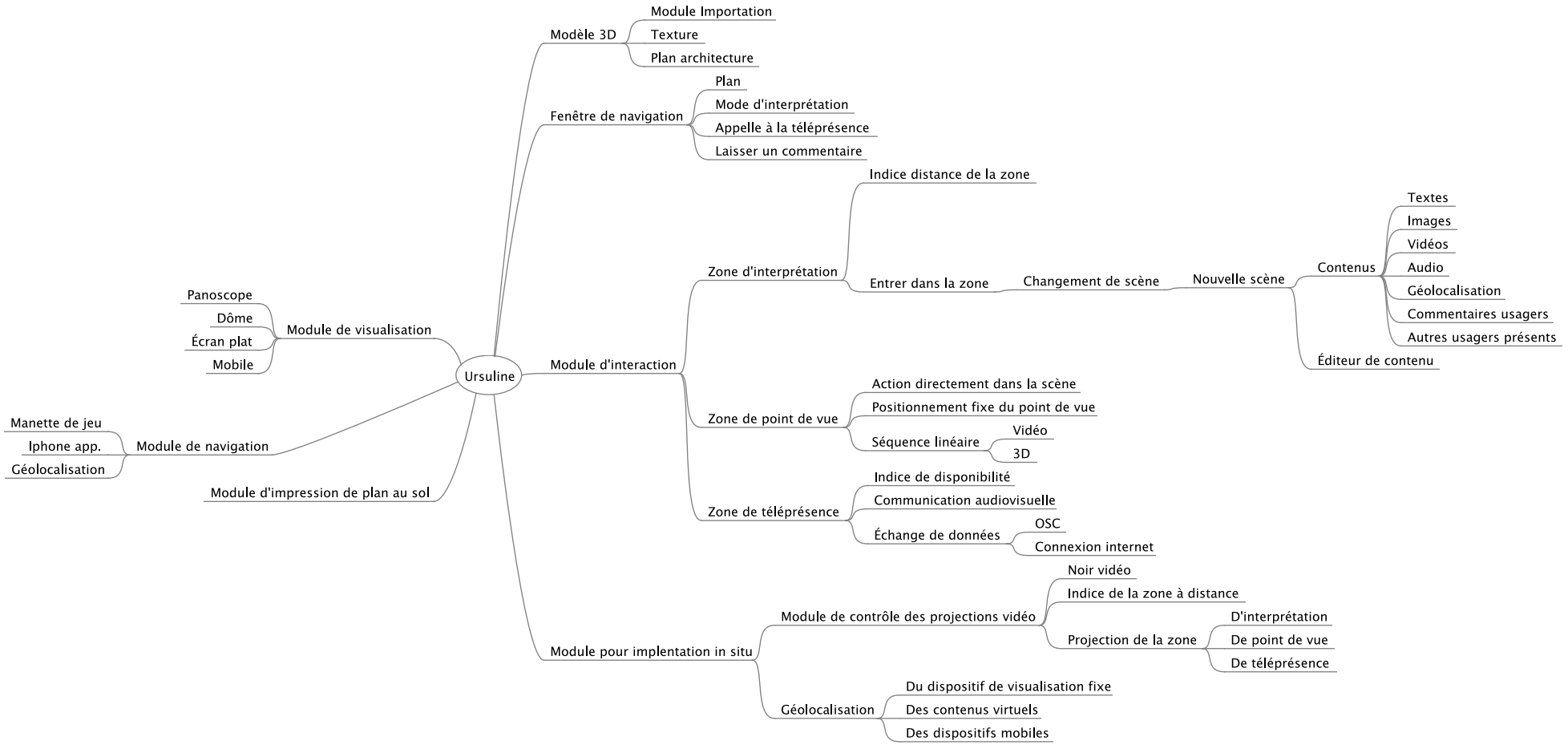


FIG. C.2. : Liste des fonctionnalités souhaitées lors de notre arrivée au LAMIC

D. Codes source

fichier D.1 : unity/Bouton.cs

```
using UnityEngine;

/* Dessine une 'icon' à la position 'pos' et appele le fonction 'action' si on
 * clique dessus (à condition d'appeler Click dans OnGUI). Peut aussi afficher
 * une image alternative 'icon_h' quand la souris passe dessus (sous réserve
 * d'appeler Highlight dans OnGUI pour détecter l'évènement).
 */
public class Bouton {
  /*TODO : une amélioration possible serait de garder en mémoire le billboard qui
   * contient l'image pour ne pas avoir à le rechercher à chaque appel à
   * Highlight (donc à chaque mouvement de souris, ce qui est assez fréquent).
   */
  public Rect pos { get; set; }
  public Texture2D icon { get; set; }
  public Texture2D icon_h { get; set; }
  public System.Action action { get; set; }
  public HGui gui { get; set; }
  private bool m_draw;
  public bool draw {
    get { return m_draw; }
    set {
      m_draw = value;
      GameObject g = GameObject.Find(gui.GetHash(icon.ToString()));
      if (g != null) g.renderer.enabled = m_draw;
    }
  }
  public void Draw(float dst) {
    draw = true;
    gui.affiche_image(icon.ToString(), icon, pos, dst);
  }
  public void Click() {
    if (draw &&
        pos.Contains(new Vector2(Input.mousePosition.x, gui.h - Input.mousePosition.y)))
      action();
  }
  public void Highlight() {
    if (draw &&
        pos.Contains(new Vector2(Input.mousePosition.x, gui.h - Input.mousePosition.y)))
      GameObject.Find(gui.GetHash(icon.ToString())).renderer.material.mainTexture = icon_h;
    else
      GameObject.Find(gui.GetHash(icon.ToString())).renderer.material.mainTexture = icon;
  }
}
```

fichier D.2 : unity/CameraFly.cs

```
/**
 * Géraud Le Falher <daureg@gmail.com>
 * 2012-06-05
 * Conversion de
 * http ://www.andrewnoske.com/wiki/index.php?title=Unity_-_Scripts#CameraFly en c#
 */
using UnityEngine;
using System.Collections;

[RequireComponent (typeof (CharacterController))]
/* Permet de déplacer un objet avec les flèches directionnelles où l'avant
 * correspond à son vecteur avant local. */
public class CameraFly  : MonoBehaviour {
  public float moveSpeed = 3.0f;
  public float rotateSpeed = 20.0f;
  private CharacterController perso;
```

```
  void Start() {
    Screen.showCursor = true;
    perso = GetComponent<CharacterController>();
  }

  void Update () {
    float moveZ = Input.GetAxis("Vertical");
    float moveX = Input.GetAxis("Horizontal");
    Vector3 moveDirection = new Vector3(moveX, 0, moveZ) * Time.deltaTime * moveSpeed;
    moveDirection = transform.TransformDirection(moveDirection);
    /* Déplace le visiteur en tenant compte des collisions, par exemple en le
     * faisant glisser le long des murs. */
    perso.Move(moveDirection);

  }
}
```

fichier D.3 : unity/CameraGUI.cs

```
using UnityEngine;
using System.Collections;

/* Affichage 2D supplémentaire par rapport à VoitTableau :
 * - un bouton pour alterner entre les modèles pleins et filaires
 * - le nom complet de l'objet visé s'il n'est pas vide.
 */
public class CameraGUI : MonoBehaviour {

  public Texture2D changement;
  public GUISkin skin;
  private float fraction = .55f;
  private Statut visiteur;
  [HideInInspector]
  public string msg;

  void Start () {
    msg = "";
    visiteur = GameObject.FindWithTag("Player").GetComponent<Statut>();
  }

  void OnGUI () {
    GUI.skin = skin;
    float w = Screen.width;
    float h = Screen.height;
    if (msg != "") {
      GUIContent g = new GUIContent(msg);
      Vector2 size = GUI.skin.GetStyle("label").CalcSize(g);
      GUI.Label(new Rect(fraction*w, 0.5f*(h - size.y), size.x, size.y), g);
    }
    if (visiteur.objet == "none") {
      if (GUI.Button(new Rect((w-HGui.bouton_w)/2f, 5, HGui.bouton_w,
              HGui.bouton_h), changement)) {
        if (Application.loadedLevelName == "default") Application.LoadLevel("filaire");
        else Application.LoadLevel("default");
      }
    }
  }
}
```

fichier D.4 : unity/Commentaires.cs

```
/**
 * Géraud Le Falher <daureg@gmail.com>
 * 2012-05-24
 */
using UnityEngine;
using System.Collections;

/* Affiche des zones de texte pour insérer et lire des commentaires relatif à un
 * objet. */
public class Commentaires : MonoBehaviour {

  private GameObject player;
  private Statut visiteur;
```

```
  private string retour_php = "";
  private string pseudo = "";
  private string texte = "";

  void Start () {
    player = GameObject.FindWithTag("Player");
    visiteur = player.GetComponent<Statut>();
    enabled = false;
  }

  void OnGUI() {
    if (visiteur.admire) {
      GUI.Label(new Rect(10, 10, 80, 20), "Pseudo :");
      pseudo = GUI.TextField(new Rect(90, 10, .35f*Screen.width-80, 20), pseudo, 25);
      GUI.Label(new Rect(10, 40, 80, 20), "Commentaire :");
      texte = GUI.TextArea (new Rect (90, 40, .35f*Screen.width-80, 0.8f*Screen.height), texte);
      if (GUI.Button(new Rect(10,0.8f*Screen.height+50,100,20), "Envoyer")) {
        envoyer_commentaire();
      }
      GUI.Label(new Rect(.38f*Screen.width, 10,
            (1-.38f)*Screen.width-50, Screen.height - 10), retour_php);
    }
  }
  /* Envoie un formulaire au serveur avec le pseudo et le texte du commentaires. */
  void envoyer_commentaire() {
    var form = new WWWForm();
    form.AddField("pseudo", pseudo);
    form.AddField("texte", texte);
    form.AddField("objet", visiteur.objet);
    var w = new WWW(PlayerPrefs.GetString("url_contenu")+"send.php", form);
    pseudo = "";
    texte = "";
    StartCoroutine(WaitForRequest(w));
    /* On récupère les commentaires avec le nouveau en prime */
    recuperer_commentaires();
  }

  /* Envoie effectif en attendant la réponse */
  IEnumerator WaitForRequest(WWW www) {
    yield return www;
    if (www.error == null) retour_php = www.text;
    else Debug.LogWarning("WWW Error : "+ www.error);
  }
  public void recuperer_commentaires() {
    var form = new WWWForm();
    form.AddField("objet", visiteur.objet);
    var w = new WWW(PlayerPrefs.GetString("url_contenu")+"get.php", form);
    StartCoroutine(WaitForRequest(w));
  }
}
```

fichier D.5 : unity/GetObject.cs

```
using UnityEngine;
using System.Collections;
using System;
using System.Xml;

/* TODO : utiliser des énumérations plutôt que des string. Et en bonus,
 * s'arranger pour qu'elles soient issues du schéma SQL de la base de données.
 */
public class Bloc {
  public string path {get; set;}
  public uint len {get; set;}
  public string desc {get; set;}
  public int niveau {get; set;}
  public string pos {get; set;}
  public string ordre {get; set;}
}
public class Image : Bloc {
  public uint w {get; set;}
  public uint h {get; set;}
  public string type {get; set;}
  public Texture2D tex {get; set;}
}
public class Text : Bloc {
```

```
  public string content {get; set;}
}
public class Audio  : Bloc {
  public bool spatial {get; set;}
  public bool boucle {get; set;}
}
public class Video  : Bloc {}

/* À la demande (fonction recuperer_elements), le script se connecte au serveur
 * et reçoit un document xml décrivant les contenus de l'objet. Il télécharge
 * les images et les textes et expose le tout dans des listes publiques (donc
 * accessibles par VoitTableau). */
public class GetObject  : MonoBehaviour {

  [HideInInspector]
  /* D'après la documentation, utiliser des tableaux plutôt que des listes est
   * plus performant, mais au détriment évidemment de la simplicité. */
  public ArrayList audios;
  public ArrayList images;
  public ArrayList texts;
  public ArrayList videos;
  public Hashtable objet;
  public Hashtable textures;
  public string[] textes = { "", "", "" };

  private Statut visiteur;
  private XmlDocument xml;

  /* Renvoie la valeur numérique correspondant au niveau désigné par 'n'.
   * TODO : * Là aussi, une énumération serait la bienvenue... (ou au moins
   * récupérer niveau_val sur le serveur.
   */
  private int getNiveau(string n) {
    switch (n.Substring(0,4)) {
      case "base" :
        return 0;
      case "moye" :
        return 1;
      case "avan" :
        return 2;
      default :
        return -1;
    }
  }
  public IEnumerator recuperer_elements() {
    string url = PlayerPrefs.GetString("url_contenu");
    WWWForm form = new WWWForm();
    form.AddField("name", gameObject.name);
    WWW info = new WWW(url+"getxml.php", form);
    WWW www;
    yield return info;
    if (info.error == null) {
      /* Si la récupération s'est déroulé sans problème, on crée les structures
       * contenantes. */
      objet = new Hashtable(4);
      textures = new Hashtable();
      images = new ArrayList();
      texts = new ArrayList();
      audios = new ArrayList();
      videos = new ArrayList();
      /* Puis on commence à lire le document en s'intéressant d'abord à l'objet
       */
      xml = new XmlDocument();
      xml.LoadXml(info.text);
      XmlNode root = xml.DocumentElement;
      objet = new Hashtable(root.Attributes.Count);
      foreach (XmlAttribute att in root.Attributes) {
        objet.Add(att.Name, att.Value);
      }
      /* Et on fini par parcourir les contenus. */
      string kind = "";
      string path="";
      int niveau;
      foreach (XmlNode liste in root.ChildNodes) {//pour chaque type de contenu
        kind = liste.Name;
        foreach (XmlNode doc in liste.ChildNodes) {//pour chaque contenu de ce type
```

```
          XmlNode bloc = doc["bloc"]; //les informations communes sont dans <bloc>
          path = bloc.Attributes["path"].Value;
          niveau = getNiveau(bloc.Attributes["niveau"].Value);
          switch (kind) { // on crée l'instance de la classe correspondant au type du contenu
            case "images" :
              /* TODO charger l'image de façon ASYNCHRONE dans une coroutine au
               * lieu de suspendre la fonction et donc la lecture du xml. */
              www = new WWW(url+path);
              yield return www;
              textures.Add(path, www.texture);
              images.Add(new Image {w=UInt32.Parse(doc.Attributes["w"].Value),
                  h=UInt32.Parse(doc.Attributes["h"].Value),
                  tex=www.texture,
                  type=doc.Attributes["type"].Value,
                  path=path,
                  desc=bloc.InnerText,
                  len=UInt32.Parse(bloc.Attributes["len"].Value),
                  niveau=niveau,
                  ordre=bloc.Attributes["ordre"].Value,
                  pos=bloc.Attributes["pos"].Value });
              break;
          case "texts" :
            texts.Add(new Text {content=doc.InnerText,
                path=bloc.Attributes["path"].Value,
                desc=bloc.InnerText,
                len=UInt32.Parse(bloc.Attributes["len"].Value),
                niveau=niveau,
                ordre=bloc.Attributes["ordre"].Value,
                pos=bloc.Attributes["pos"].Value });
            textes[niveau] += doc.InnerText + "\n";
            break;
          case "audios" :
            audios.Add(new Audio {
                spatial=Convert.ToBoolean(UInt32.Parse(doc.Attributes["spatial"].Value)),
                boucle=Convert.ToBoolean(UInt32.Parse(doc.Attributes["boucle"].Value)),
                path=bloc.Attributes["path"].Value,
                desc=bloc.InnerText,
                len=UInt32.Parse(bloc.Attributes["len"].Value),
                niveau=niveau,
                ordre=bloc.Attributes["ordre"].Value,
                pos=bloc.Attributes["pos"].Value });
            break;
          case "videos" :
            videos.Add(new Video {path=bloc.Attributes["path"].Value,
                desc=bloc.InnerText,
                len=UInt32.Parse(bloc.Attributes["len"].Value),
                niveau=niveau,
                ordre=bloc.Attributes["ordre"].Value,
                pos=bloc.Attributes["pos"].Value });
            break;
          }
        }
      }
      /* On signal à VoitTableau qu'il peut lancer l'animation (parce si on
       * anime pendant qu'on charge, le mouvement est TRÈS saccadé. */
      gameObject.GetComponent<VoitTableau>().attend = false;
    }
    else {
      Debug.LogWarning(System.String.Format("Impossible de recuperer le xml."+
                          " Est-ce que cette adresse ({0}) est accessible dans un navigateur ?",
                          url+"getxml.php"));
    }
  }
}
```

fichier D.6 : unity/HandleImageEffect.cs

```
/**
 * Géraud Le Falher <daureg@gmail.com>
 * 2012-06-13
 * Active et désactive les effets d'images pendant la pause parce que menu.js
 * ne peut accéder directement à GlowEffect et EdgeDetectEffect.
 * TODO : maintenant que Menu est c#, on doit pouvoir le faire directement, mais
 * c'est sûrement plus propre comme ça.
 */
using UnityEngine;
```

```
using System.Collections;

public class HandleImageEffect  : MonoBehaviour {
  public bool utiliserEffetsImage = false;
  public Transform cam;

  void Start() {
    if (cam == null) cam = transform.Find("-CameraRig/RealRig/FrontCam");
    if (utiliserEffetsImage) enableImageEffects();
  }

  void disableImageEffects() {
    if (utiliserEffetsImage) {
      cam.GetComponent<GlowEffect>().enabled = false;
      cam.GetComponent<EdgeDetectEffect>().enabled = false;
    }
  }

  void enableImageEffects() {
    if (utiliserEffetsImage) {
      cam.GetComponent<GlowEffect>().enabled = true;
      cam.GetComponent<EdgeDetectEffect>().enabled = true;
    }
  }
}
```

fichier D.7 : unity/HGui.cs

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;
using System.Security.Cryptography;
using System.Text;

/* Affiche des éléments d'interface en vrai 3D, dans la scène. */
public class HGui {
  public static int bouton_w = 48;
  public static int bouton_h = 48;
  private Camera m_cam;
  private static bool UGLY_CAMILLE_S_HACK = true;
  public Camera cam {
    get { return m_cam; }
    set {
      m_cam = value;
      h = value.pixelHeight;
      w = value.pixelWidth;
      refPlane = value.transform.Find("ScreenPlane").gameObject;
      tooltip_back = value.transform.Find("tooltip_back").gameObject;
    }
  }
  public float h { get; private set; }
  public float w { get; private set; }
  private GameObject refPlane;
  private GameObject tooltip_back;
  private HashAlgorithm Hash;

  public GameObject probe = (GameObject)Resources.Load("probe2", typeof(GameObject));

  public HGui(Camera c) {
    cam = c;
    Hash = new SHA1Managed();
  }

  /* Renvoie le nom de fichier contenu dans le chemin 's' */
  public static string filenameFromPath(string s) {
    string[] path = s.Split('/');
    string[] name = path[path.Length-1].Split('.');
    string filename = "";
    for (int j = 0; j < name.Length-1; j++) {
      filename += name[j];
    }
    return filename;
  }

  /* Afficher la texture 'tex' (normalement nommée 'path') à la position écran
   * 'pos', sur un plan situé à la distance 'dst' de la caméra.
```

```
 */
public void affiche_image(string path, Texture tex, Rect pos, float dst) {
  /* On crée un billboard a une position approximative */
  GameObject back =
    (GameObject) Object.Instantiate(Resources.Load("billboard", typeof(GameObject)),
    cam.ScreenToWorldPoint(new Vector3(pos.x+pos.width/2, pos.y-pos.height/2, dst)),
    cam.transform.rotation);
  back.name = GetHash(path);
  /* On place temporairement le plan à distance voulue pour projeter le
   * billboard dessus et lui affecte la texture sur fond transparent. */
  float tmpz = refPlane.transform.localPosition.z;
  refPlane.transform.localPosition = new Vector3(0,0,dst);
  back.GetComponent<PlacePlane>().Change(cam, pos, refPlane);
  refPlane.transform.localPosition = new Vector3(0,0,tmpz);
  back.renderer.material = (Material)
    Resources.Load("alpha_background", typeof(Material));
  back.renderer.material.mainTexture = tex;
  /* Si c'est un film, on lance sa lecture avec le son. */
  if (tex.GetType() == typeof(MovieTexture)) {
    if (((MovieTexture)tex).isReadyToPlay) {
      if (((MovieTexture)tex).isPlaying) {
        ((MovieTexture)tex).Stop();
      }
      ((MovieTexture)tex).Play();
    }
    back.AddComponent<AudioSource>();
    back.audio.clip = (AudioClip)((MovieTexture)tex).audioClip;
    back.audio.Play();
  }
}

/* Utilise l'objet 'texte' pour afficher 'txt' face à 'cam', en démarrant à la
 * position 'positionEcran' (où positionEcran.z est la distance souhaitée entre
 * texte et la caméra), en utilisant au plus 'largeur' pixels (avec une
 * certaine 'marge' pour ne pas l'atteindre complètement) et en placant
 * éventuellement un 'fond' derrière. Renvoie environ la taille du texte
 * résultant (modulant l'espace inter ligne)
 */
public Vector2 affiche_texte(TextMesh texte, string txt, Vector3
    positionEcran, float largeur, bool fond=false, float marge=1.00f) {
  /* On place le texte à sa position finale et on regarde quelle taille il
   * occupe en fonction de sa longueur */
  texte.renderer.enabled = true;
  Vector3 pos = positionEcran;
  texte.transform.position = cam.ScreenToWorldPoint(pos);
  texte.transform.rotation = cam.transform.rotation;
  texte.text = txt; /* il y a de moins en moins de 'e' dans cette ligne;
  bientôt «  La disparition  » en C# */
  Vector2 texte_size = getCharPixelSize(texte, pos);
  /* On insère les sauts de ligne nécessaire */
  string cut = "";
  texte_size = Layout.formatter(texte_size, txt,
      System.Convert.ToInt32(marge*largeur), out cut);
  texte.text = cut;
  if (texte_size == Vector2.zero) return texte_size;

  if (fond) {
    /* On projette le billboard sur le fond de l'écran avec un matériau transparent.*/
    tooltip_back.renderer.enabled = true;
    float tmpz = refPlane.transform.localPosition.z;
    refPlane.transform.localPosition = new Vector3(0,0,1.05f*pos.z);
    tooltip_back.GetComponent<PlacePlane>().Change(cam,
           /* comme on sait qu'on utilise fond que pour le tooltip, on
            * n'utilise pas la vrai taille du texte mais une approximation
            * précalculée. */
           new Rect(pos.x, h - pos.y, txt.Length*Screen.width/95f, Screen.height/32f),
          refPlane);
    refPlane.transform.localPosition = new Vector3(0,0,tmpz);
  }
  return texte_size;
}

/* Retourne les 2*'size' premiers caractères du hash de 'input' */
public string GetHash(string input, int size=4) {
  byte[] data = Hash.ComputeHash(Encoding.UTF8.GetBytes(input));
  size = Mathf.Clamp(size, 4, data.Length);
```

```
  StringBuilder sBuilder = new StringBuilder();
  for (int i = 0; i < size; i++) sBuilder.Append(data[i].ToString("x2"));
  return sBuilder.ToString();
}

/* Renvoie la largeur et la hauteur moyenne d'un caractère de 'texte' quand il
 * est placé en 'pos' où  : pos.x et pos.y sont les coordonnées à l'écran du
 * sommet haut-gauche du texte et pos.z la distance à la caméra
 */
public Vector2 getCharPixelSize(TextMesh texte, Vector3 pos, bool debug=false) {
  if (UGLY_CAMILLE_S_HACK) return new Vector2(9.0f, 23f);
  TextMesh sample = GameObject.Find("sample_text").GetComponent<TextMesh>();
  sample.renderer.enabled = true;
  sample.characterSize = texte.characterSize;
  sample.text = texte.text;
  float sx = sample.renderer.bounds.size.x;
  float sy = sample.renderer.bounds.size.y;

  /* POURQUOI IL Y A BESOIN DE FAIRE ÇA !!!! */
  if (sample.collider == null) sample.gameObject.AddComponent<BoxCollider>();
  if (debug) Debug.Log("box "+((BoxCollider)sample.collider).size);
  sx = ((BoxCollider)sample.collider).size.x;
  Object.Destroy(sample.collider);

  /* En gros, sx et sy sont la taille du texte affiché sur une seule ligne
   * sans rotation. On lance un rayon partant de la caméra à la position de
   * l'écran où est censé débuter le texte et on mémorise son intersection
   * avec le plan derrière la caméra. Puis on se déplace vers la droite de sx
   * dans le plan et on regarde à quel point cela correspond à l'écran. On
   * fait de même vers le bas avec sy. Et y'a des fois où ça marche ! */
  Transform tPlane = refPlane.transform;
  float ppos = tPlane.localPosition.z;
  tPlane.localPosition = new Vector3(0,0, pos.z);
  Ray ray = cam.ScreenPointToRay(pos);
  RaycastHit hit;
  tPlane.collider.Raycast(ray, out hit, 30*pos.z);
  Vector3 droite = cam.WorldToScreenPoint(hit.point + sx*tPlane.right);
  ray = cam.ScreenPointToRay(new Vector3(cam.pixelWidth*.5f, pos.y, pos.z));
  tPlane.collider.Raycast(ray, out hit, 30*pos.z);
  Vector3 bas = cam.WorldToScreenPoint(hit.point - sy*tPlane.forward);
  tPlane.localPosition = new Vector3(0,0, ppos);
  sample.renderer.enabled = false;
  return new Vector2(Mathf.Abs((droite-pos).x)/sample.text.Length, Mathf.Abs((pos-bas).y));
}

/* Renvoie la taille d'un objet vu à l'écran en pixel en se basant sur des
 * approximation successives de lancer de rayon avec son collider (donc est
 * très précis pour les objets qui disposent d'un mesh collider, moins avec
 * juste une box).
 */
public Rect ObjectSize(GameObject g, bool invisible) {
  if (invisible) {
    return new Rect(w-HGui.bouton_w, h/2f, HGui.bouton_w, 0f);
  }
  /* En gros, 90px pour 1024x768 (donc l'objet ne doit pas être plus petit que
   * cette taille) */
  int precision = (int) Mathf.Ceil((h+w) / 20f);
  int raffinement = 0;
  // On commence par parcourir tout l'écran
  Rect estim = new Rect(0,0, cam.pixelWidth, cam.pixelHeight);
  while (raffinement < 3) {
    /* A chaque tour, estim se resserre autour de l'objet donc on peut se
     * permettre de lancer plus de rayons la fois d'après en diminuant
     * precision.
     */
    estim = ApproxSize(g, estim, precision);
    precision /= 3;
    raffinement++;
  }
  return estim;
}

/* Calcule la taille (en coordonnées écrans) de 'g' à l'écran vu par 'cam', à
 * partir d'une estimation rectangulaire supérieure. De manière peu subtile,
 * on lance des rayons à partir des coins de 'estim' tous les 'precision'
 * pixels et on remplit min et max dès qu'on touche 'g' Renvoie l'estimation
```

```
  * obtenue sous forme de Rect (en coordonnée GUI, ie le (0,0) est en haut à
  * gauche).
  */
 private Rect ApproxSize(GameObject g, Rect estim, int precision, bool debug=false) {
   Vector2 min = Vector2.zero;
   Vector2 max = Vector2.zero;
   float i, j;

   if (debug) Debug.Log("estim au debut (Rect coord)  : " + estim + precision);
   Ray ray = new Ray(Vector3.zero, Vector3.one);
   Vector3 dest = Vector3.zero;
   bool quit = false;
   RaycastHit hit;
   float len = 15f*(cam.transform.position - g.transform.position).magnitude;

   for (j = estim.yMax; j > estim.yMin; j-=precision) {
     for (i = estim.xMin; i < estim.xMax; i+=precision) {
       dest.Set(i, h-j, 0);
       ray = cam.ScreenPointToRay(dest);
       if (g.collider.Raycast(ray, out hit, len)) {
         min.y = h-j - precision;
         quit = true;
         break;
       }
     }
     if (quit) {
       quit = false;
       break;
     }
   }
   for (i = estim.xMin; i < estim.xMax; i+=precision) {
     for (j = estim.yMax; j > estim.yMin; j-=precision) {
       dest.Set(i, h-j, 0);
       ray = cam.ScreenPointToRay(dest);
       if (g.collider.Raycast(ray, out hit, len)) {
         min.x = i - precision;
         quit = true;
         break;
       }
     }
     if (quit) {
       quit = false;
       break;
     }
   }
   for (j = estim.yMin; j < estim.yMax; j+=precision) {
     for (i = estim.xMin; i < estim.xMax; i+=precision) {
       dest.Set(i, h-j, 0);
       ray = cam.ScreenPointToRay(dest);
       if (g.collider.Raycast(ray, out hit, len)) {
         max.y = h-j + precision;
         quit = true;
         break;
       }
     }
     if (quit) {
       quit = false;
       break;
     }
   }
   for (i = estim.xMax; i > estim.xMin; i-=precision) {
     for (j = estim.yMax; j > estim.yMin; j-=precision) {
       dest.Set(i, h-j, 0);
       ray = cam.ScreenPointToRay(dest);
       if (g.collider.Raycast(ray, out hit, len)) {
         max.x = i + precision;
         quit = true;
         break;
       }
     }
     if (quit) {
       quit = false;
       break;
     }
   }
   return new Rect(min.x, h-max.y, (max-min).x, (max-min).y);
```

```
  }
}
```

fichier D.8 : unity/IPServeur.cs

```
using UnityEngine;
using System.Collections;

/* Au démarrage, vérifie si l'adresse du serveur a déjà été définie. Le cas
 * échéant, on la récupère, sinon on en fixe une par défaut. Ensuite, on peut
 * proposer à l'utilisateur d'en changer.
 */
public class IPServeur  : MonoBehaviour {

  private string url_contenu;
  private float nextFire = 0.0f;
  private bool affiche = false;
  void Start () {
    if (PlayerPrefs.HasKey("url_contenu"))
      url_contenu = PlayerPrefs.GetString("url_contenu");
    else {
      url_contenu = "http ://127.0.0.1/URS/";
      PlayerPrefs.SetString("url_contenu", url_contenu);
    }
    GameObject.Find("gestion_objets").SendMessage("getData");
  }
  /* Si l'utilisateur appuie sur K, on affiche un champ de texte qui permet
   * d'entrer une nouvelle adresse, qui est à son tour inscrite dans les
   * préférences. */
  void OnGUI() {
    if (Input.GetKeyDown(KeyCode.K) && Time.time > nextFire) {
      nextFire = Time.time + 1f;
      affiche = true;
    }
    if (affiche) {
      float w = Screen.width;
      float h = Screen.height;
      GUI.Label(new Rect(w/2, h/2, 100, 40), "URL du contenu");
      url_contenu = GUI.TextField(new Rect(w/2+100, h/2, 200, 40), url_contenu, 40);
      if (GUI.Button(new Rect(w/2+300,h/2,50,40), "OK")) {
        PlayerPrefs.SetString("url_contenu", url_contenu);
        GameObject.Find("gestion_objets").SendMessage("getData");
        affiche = false;
      }
    }
  }
}
```

fichier D.9 : unity/Layout.cs

```
using UnityEngine;
using System.Collections.Generic;
using System.Text.RegularExpressions;
/* Contient diverses fonctions statiques qui disposent du contenu à l'écran en
 * fonction de divers contraintes et en essayant de respecter les règles
 * établies pour chaque niveau d'écran.
 */
public class Layout {
  public static Vector2 supp_text_size;
  public static List<Texture> textures;
  private static float marge = .025f;

  /* Indique à quel position couper un texte pour avoir (environ)
   * 100*coupure % des mots d'un côté et les autres de l'autre. Coupe
   * seulement à la fin d'une phrase, sauf si 'au_mot' est vrai.
   */
  public static int couperChaine(string s, float coupure, bool au_mot=false) {
    int l = s.Length;
    if (l == 0) return 0;

    int positionIdeale = (int)(coupure * l);
    Regex rx = (au_mot) ? new Regex(@"[.\s?!;]")  : new Regex(@"[.?!;]");
    int avant = 0, apres = l - 1;
    /* On parcourt la chaine vers la fin en cherchant la prochaine
     * ponctuation ou éventuellement un espace. */
```

```
  for (int i = positionIdeale; i < l; i++) {
    if (rx.IsMatch(s.Substring(i,1))) {
      apres = i;
      break;
    }
  }
  /* Puis vers le début */
  for (int i = positionIdeale; i >= l; i--) {
    if (rx.IsMatch(s.Substring(i,1))) {
      avant = i;
      break;
    }
  }
  /* On renvoie le plus près de la position idéale, ou le plus loin si on
   * a rien trouvé avant. */
  if (avant == 0 || (apres - positionIdeale) < (positionIdeale - avant))
    return apres;
  return avant;
}

/* Renvoie la taille en pixel d'une version avec des saut de ligne de 'txt'
 * (stockée dans 'res') afin qu'il ne dépasse pas 'largeur', sachant que
 * chaque caractères mesure environ 'char_size' en pixels.
 */
public static Vector2 formatter(Vector2 char_size, string txt, int largeur,
    out string res) {
  res = txt.Trim();
  if (res.Length == 0) return Vector2.zero;
  int last_space = 0;
  string cut_text = "";
  int nb_lignes=1;
  int longest_line = 0;

  /* On parcourt la chaine caractère par caractère en notant la position
   * des espaces et des saufs lignes existants. Quand on a accumulé assez
   * de caractère pour dépasser la largeur, on les ajoute à 'cut_text' en
   * regardant si ce n'est pas la plus longue. */
  for (int c = 0; c < res.Length; c++) {
    if (res[c] == ' ') last_space = c;
    if (res[c] == '\n' || (c - cut_text.Length+1)*char_size.x > largeur) {
      if (res[c] == '\n') last_space = c;
      if ((last_space - cut_text.Length) > longest_line) {
        longest_line = last_space - cut_text.Length;
      }
      cut_text += res.Substring(cut_text.Length,
          last_space - cut_text.Length).Trim() + "\n";
      nb_lignes++;
    }
  }
  if (longest_line == 0) longest_line = res.Length; //une seule ligne

  cut_text += res.Substring(cut_text.Length, res.Length - cut_text.Length);
  res = cut_text;
  return new Vector2(char_size.x * longest_line,  char_size.y * nb_lignes);
}

/* Essaye de faire rentrer 'txt' dans un colonne qui prend entre un tiers et
 * deux tiers de la largeur de 'utile' et renvoie sa nouvelle taille.
 */
public static Vector2 colonneMax(Vector2 char_size, string txt, Rect utile) {
  float tw = utile.width/3f;
  string useless = "";
  Vector2 ts = Layout.formatter(char_size, txt, (int)tw, out useless);
  while (ts.y > utile.height && tw < 2*utile.width/3f) {
    tw *= 1.1f;
    ts = Layout.formatter(char_size, txt, (int)tw, out useless);
  }
  return ts;
}

/* Réduit une image de dimension texw × texh pour qu'elle rentre dans
 * l'espace (plus petit) allowedw × allowedh
 */
public static float resizeWithRatio(float texw, float texh, float allowedw,
    float allowedh) {
  /* On modifie d'abord la plus grande dimension pour coller au bord.
```

```
    * Si c'est possible, fini, sinon on modifie l'autre dimension.
    */
   float coeff;
   if (texw > texh) {
     coeff = allowedw/texw;
     if (coeff * texh <= allowedh) return coeff;
     else return allowedh/texh;
   }
   else {
     coeff = allowedh/texh;
     if (coeff * texw <= allowedw) return coeff;
     else return allowedw/texw;
   }
 }

 /* Renvoie la position d'une image de dimension 'imw' × 'imh', insérée à la
  * 'idx'eme position (sur nb_images au total) dans le sens horizontal ou non
  * au sein d'une bande rectangulaire. 'bande' est spécifiée en coordonnées
  * écran, tout comme le rectangle retourné.
  */
 public static Rect placeIemeImage(Rect bande, float imw, float imh, int idx,
     int nb_images, bool horizontal) {
   float x, y;
   if (horizontal) {
     x = bande.x + 1f*(1+2*idx)/(2*nb_images)*bande.width - imw/2f;
     y = bande.y - (bande.height - imh)/2f;
   }
   else {
     x = bande.x + (bande.width - imw)/2f;
     y = bande.y - 1f*(1+2*idx)/(2*nb_images)*bande.height + imh/2f;
   }
   return new Rect(x, y, imw, imh);
 }

 /* 'pos' doit contenir la position d'un texte et de plusieurs images en
  * dessous de lui qui tiennent tous dans 'utile'. La fonction renvoie cette
  * même liste de façon à ce que le texte se situe en-dessous.
  */
 public static List<Rect> inverserTexteEtImages(Rect utile, List<Rect> pos) {
   List<Rect> npos = new List<Rect>(pos.Count);
   Rect tmp = new Rect(pos[0].x, utile.y + pos[0].height,
       pos[0].width, pos[0].height);
   npos.Add(tmp);
   float decal = pos[0].y - pos[1].y;
   for (int i = 1; i < pos.Count; i++) {
     tmp.Set(pos[i].x, pos[i].y + decal, pos[i].width, pos[i].height);
     npos.Add(tmp);
   }
   return npos;
 }

 /* Renvoie la fonction correspondant à l'écran 'i'. */
 public static System.Func<Rect, Vector2, List<Rect>> ecran(int i) {
   switch (i) {
     case 1 :
       return ecran1;
     case 2 :
       return ecran2;
     case 3 :
       return ecran3;
     default :
       return oups;
   }
 }

 /* classe les images de 'texs' en deux catégories (portraits et paysages) en
  * changeant l'ordre original et en indiquant combien il y en a dans chaque
  * catégorie.
  */
 public static void trieImages(ref List<Texture> texs, out List<Texture>
     portrait, out List<Texture> landscape, out float nbp, out float nbl,
     bool inverse=false) {
   portrait = new List<Texture>(texs.Count);
   landscape = new List<Texture>(texs.Count);
   foreach (Texture t in texs) {
     if (t.width > t.height) landscape.Add(t);
```

```
      else portrait.Add(t);
    }
    nbp = portrait.Count;
    nbl = landscape.Count;
    /* Remettre les textures originales dans ce nouvel ordre. */
    texs.Clear();
    if (inverse) {
      foreach (Texture t in landscape) texs.Add(t);
      foreach (Texture t in portrait) texs.Add(t);
    }
    else {
      foreach (Texture t in portrait) texs.Add(t);
      foreach (Texture t in landscape) texs.Add(t);
    }
  }

  /* Sachant que le texte a une taille 'txt_size' (et qu'éventuellement le
   * deuxième bout à une taille 'Layout.supp_text_size'), les trois fonctions
   * suivantes renvoient la liste des positions que devraient occuper le texte
   * et les textures contenues dans 'Layout.textures' pour remplir la zone
   * 'utile' (indiquée en coordonnées écran).
   */
  public static List<Rect> ecran1(Rect utile, Vector2 txt_size) {
    /* On trie d'abord les images en deux types */
    List<Texture> portrait, landscape;
    float nbp, nbl;
    Layout.trieImages(ref textures, out portrait, out landscape, out nbp, out nbl);

    /* Ensuite on les redimensionne. Pour cela, on calcule d'abord la taille
     * maximum qu'elles peuvent occuper, sans tenir compte des marges.
     * Malheureusement, les noms de variables sont abrégés :
     * a -> allowed
     * w -> width
     * h -> height
     * p -> portrait
     * l -> landscape
     */
    List<Vector2> pDims = new List<Vector2>((int)nbp);
    List<Vector2> lDims = new List<Vector2>((int)nbl);
    float awp = (nbp > 0) ? utile.width/nbp  : 0f;
    float ahp = (nbp > 0) ? ((utile.height - txt_size.y)*((nbl>0)?.5f :1f))  : 0f;
    float awl = (nbl > 0) ? (utile.width)  : 0f;
    float ahl = (nbl > 0) ? ((utile.height - txt_size.y)/nbl*((nbp>0)?.5f :1f))  : 0f;
    float r;

    /* On conserve en mémoire la plus grande hauteur pour les portraits et
     * leur somme pour les paysages. */
    float maxhp = .0f, maxhl = .0f;
    foreach (Texture t in portrait) {
      /* Ici en revanche, les marges importent */
      r = resizeWithRatio(t.width, t.height, (1-2*marge)*awp, (1-2*marge)*ahp);
      pDims.Add(new Vector2(t.width*r,t.height*r));
      if (t.height*r > maxhp) maxhp = t.height*r;
    }
    foreach (Texture t in landscape) {
      r = resizeWithRatio(t.width, t.height, (1-2*marge)*awl, (1-2*marge)*ahl);
      lDims.Add(new Vector2(t.width*r,t.height*r));
      maxhl += t.height*r;
    }

    /* Finalement, on obtient la hauteur de chacune des deux bandes d'images.
     * Avec en bonus, une nouvelle abréviation, r -> réel */
    float rhp = maxhp + 2*marge*ahp;
    float rhl = maxhl + 2*marge*ahl*nbl;
    /* La hauteur totale du contenu */
    float rhtot = rhp + rhl + txt_size.y;
    /* On va ensuite créer tous les rectangles (en coordonnées écran)
     * correspondant au texte puis à chaque image.  */
    List<Rect> pos = new List<Rect>(1 + textures.Count);
    /* On commence en haut de la zone utile, mais en centrant verticalement */
    float debut = utile.y + utile.height - (utile.height - rhtot)/2f;
    /* On place le texte puis on descend d'un cran */
    pos.Add(new Rect(utile.x, debut, txt_size.x, txt_size.y));
    debut -= txt_size.y;
    /* Les portraits seront contenus dans cette bande */
    Rect bande = new Rect(utile.x, debut, utile.width, rhp);
```

```
  for (int i = 0; i < (int)nbp; i++) {
    pos.Add(Layout.placeIemeImage(bande, pDims[i].x, pDims[i].y, i, (int)nbp, true));
  }
  /* On descend ensuite la bande pour placer les paysages */
  bande.y -= bande.height;
  bande.height = rhl;
  for (int i = 0; i < (int)nbl; i++) {
    pos.Add(Layout.placeIemeImage(bande, lDims[i].x, lDims[i].y, i, (int)nbl, false));
  }
  return pos;
}

public static List<Rect> ecran2(Rect utile, Vector2 txt_size) {
  /* s'il n'y a pas d'image, on renvoie juste le texte centré, comme en 1 */
  if (textures.Count == 0) return ecran1(utile, txt_size);
  /* sinon on trie et on compte les images */
  List<Texture> portrait, landscape;
  float nbp, nbl;
  Layout.trieImages(ref textures, out portrait, out landscape, out nbp, out nbl, true);
  List<Rect> pos = new List<Rect>(1 + textures.Count);
  List<Texture> tmp = new List<Texture>(textures);

  /* s'il n'y a que des paysages, on renvoie l'écran 1 avec le texte en bas */
  if (nbp == 0) {
    textures = landscape;
    pos = Layout.inverserTexteEtImages(utile, ecran1(utile, txt_size));
    textures = tmp;
    return pos;
  }
  /* s'il n'a a que des portraits, on veut les mettre en colonnes avec le texte */
  if (nbl == 0) {
    List<Vector2> pDims = new List<Vector2>((int)nbp);
    float awp = utile.width - txt_size.x;
    float ahp = utile.height/nbp;
    float r, himg = .0f;

    foreach (Texture t in portrait) {
      r = resizeWithRatio(t.width, t.height, (1-2*marge)*awp, (1-2*marge)*ahp);
      pDims.Add(new Vector2(t.width*r,t.height*r));
      himg += t.height*r;
    }
    float rhp = himg + 2*marge*ahp*nbp;
    /* On ajoute d'abord le texte à droite de utile */
    pos.Add(new Rect(utile.x + awp, utile.y +
          utile.height - (utile.height - txt_size.y)/2f, txt_size.x, txt_size.y));
    /* On crée la colonne de portraits */
    Rect bande = new Rect(utile.x, utile.y +
        utile.height - (utile.height - rhp)/2f, awp, rhp);
    for (int i = 0; i < (int)nbp; i++) {
      pos.Add(Layout.placeIemeImage(bande, pDims[i].x, pDims[i].y, i, (int)nbp, false));
    }
    return pos;
  }

  /* On a deux type d'images donc on va faire deux colonnes, l'une avec
   * les paysages verticaux au dessus du début du texte, l'autre avec les
   * portraits horizontaux en dessous du la fin du texte.
   * (pourquoi faire simple...)
   */
  Rect utile1 = new Rect(utile.x, utile.y, 2*utile.width/3f - 30, utile.height);
  Rect utile2 = new Rect(utile.x + 2*utile.width/3f + 60, utile.y,
      utile.width/3f, utile.height);
  textures = landscape;
  pos = Layout.inverserTexteEtImages(utile1, ecran1(utile1, txt_size));
  int idxtxt2 = pos.Count;
  textures = portrait;
  pos.AddRange(ecran1(utile2, Layout.supp_text_size));
  /* On place le second texte à la fin pour pouvoir afficher toutes les
   * textures d'un coup */
  Rect tmpr = new Rect(pos[idxtxt2].x, pos[idxtxt2].y, pos[idxtxt2].width,
      pos[idxtxt2].height);
  pos.RemoveAt(idxtxt2);
  pos.Add(tmpr);
  textures = tmp;
  return pos;
}
```

```
  public static List<Rect> ecran3(Rect utile, Vector2 txt_size) {
    /* C'est le même principe que ecran1, sauf qu'on mets dans la même bande
     * les portraits et les paysages. */
    List<Texture> portrait, landscape;
    float nbp, nbl;
    Layout.trieImages(ref textures, out portrait, out landscape, out nbp, out nbl);
    List<Vector2> pDims = new List<Vector2>((int)nbp);
    List<Vector2> lDims = new List<Vector2>((int)nbl);
    float awp = (nbp > 0) ? utile.width/nbp*((nbl>0)?.5f :1f)  : 0f;
    float ahp = (nbp > 0) ? utile.height - txt_size.y  : 0f;
    float awl = (nbl > 0) ? (utile.width)*((nbp>0)?.5f :1f)  : 0f;
    float ahl = (nbl > 0) ? ((utile.height - txt_size.y)/nbl)  : 0f;
    float r, maxhp = .0f, maxhl = .0f;

    foreach (Texture t in portrait) {
      r = resizeWithRatio(t.width, t.height, (1-2*marge)*awp, (1-2*marge)*ahp);
      pDims.Add(new Vector2(t.width*r,t.height*r));
      if (t.height*r > maxhp) maxhp = t.height*r;
    }
    foreach (Texture t in landscape) {
      r = resizeWithRatio(t.width, t.height, (1-2*marge)*awl, (1-2*marge)*ahl);
      lDims.Add(new Vector2(t.width*r,t.height*r));
      maxhl += t.height*r;
    }

    float rhp = maxhp + 2*marge*ahp;
    float rhl = maxhl + 2*marge*ahl*nbl;
    float rhtot = System.Math.Max(rhp, rhl) + txt_size.y;
    List<Rect> pos = new List<Rect>(1 + textures.Count);
    float debut = utile.y + utile.height - (utile.height - rhtot)/2f;
    pos.Add(new Rect(utile.x, debut, txt_size.x, txt_size.y));
    debut -= txt_size.y;
    /* Les portraits seront contenus dans cette bande */
    Rect bande = new Rect(utile.x, debut, nbp*awp, rhp);
    for (int i = 0; i < (int)nbp; i++) {
      pos.Add(Layout.placeIemeImage(bande, pDims[i].x, pDims[i].y, i, (int)nbp, true));
    }
    /* On décale ensuite la bande pour placer les paysages */
    bande.x = (nbp > 0) ? utile.x + utile.width/2f  : utile.x;
    bande.height = rhl;
    bande.width = awl;
    for (int i = 0; i < (int)nbl; i++) {
      pos.Add(Layout.placeIemeImage(bande, lDims[i].x, lDims[i].y, i, (int)nbl, false));
    }
    return pos;
  }
  /* Indique qu'ecran() a été appelé avec un numéro non valide */
  public static List<Rect> oups(Rect utile, Vector2 txt_size) {
    Debug.LogWarning("cet ecran n'existe pas (encore...)");
    return null;
  }
}
```

fichier D.10 : unity/Menu.cs

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;

/* d'après http ://wiki.unity3d.com/index.php?title=PauseMenu#C.23_version
 * Affiche un menu qui met l'application en pause et permet de régler des
 * options (volume, informations systèmes et catégorie d'explications [pas
 * utilisés pour l'instant]) ou de la quitter.
 */
public class Menu  : MonoBehaviour {

  public GUISkin skin;
  public Color statColor = Color.white;
  public string[] credits= { "Chapelle des Ursulines", "Copyright (c) 2010-2012 LAMIC, université Laval"};

  private enum Page {None,Main,Options,Credits}
  private float savedTimeScale;
  private Page currentPage;
  private int toolbarInt=0;
  private string[] toolbarStrings = {"Audio","Systeme","Explications"};
```

```
void Start() { Time.timeScale = 1f; }

void LateUpdate () {
  if (Input.GetKeyDown("escape")) {
    switch (currentPage) {
      case Page.None : PauseGame(); break;
      case Page.Main : UnPauseGame(); break;
      default : currentPage = Page.Main; break;
    }
  }
}

void OnGUI() {
  if (skin != null) GUI.skin = skin;
  if (IsGamePaused()) {
    GUI.color = statColor;
    switch (currentPage) {
      case Page.Main : PauseMenu(); break;
      case Page.Options : ShowToolbar(); break;
      case Page.Credits : ShowCredits(); break;
    }
  }
}

void ShowToolbar() {
  BeginPage(300, 300);
  toolbarInt = GUILayout.Toolbar(toolbarInt, toolbarStrings);
  switch (toolbarInt) {
    case 0 : VolumeControl(); break;
    case 1 : ShowDevice(); break;
    case 2 : NiveauExplications(); break;
  }
  EndPage();
}

void ShowCredits() {
  BeginPage(300,300);
  foreach (string credit in credits) GUILayout.Label(credit);
  EndPage();
}

void ShowBackButton() {
  if (GUI.Button(new Rect(20,Screen.height-50,50,20),"Back"))
    currentPage = Page.Main;
}

void ShowDevice() {
  GUILayout.Label ("Unity version "+Application.unityVersion);
  GUILayout.Label("Graphique  : "+SystemInfo.graphicsDeviceName+" "+
          SystemInfo.graphicsMemorySize+"MB\n"+
          SystemInfo.graphicsDeviceVersion+"\n"+
          SystemInfo.graphicsDeviceVendor);
  GUILayout.Label("Ombres  : "+SystemInfo.supportsShadows);
  GUILayout.Label("Image Effects  : "+SystemInfo.supportsImageEffects);
  GUILayout.Label("Render Textures  : "+SystemInfo.supportsRenderTextures);
}

void VolumeControl() {
  GUILayout.Label("Volume");
  AudioListener.volume = GUILayout.HorizontalSlider(AudioListener.volume,0f,1f);
}

void BeginPage(int width, int height) {
  GUILayout.BeginArea(new Rect((Screen.width -
                  width)/2,(Screen.height-height)/2,width,height));
}

void EndPage() {
  GUILayout.EndArea();
  if (currentPage != Page.Main) ShowBackButton();
}

void PauseMenu() {
  BeginPage(200,200);
  if (GUILayout.Button("Reprendre")) UnPauseGame();
```

```
    if (GUILayout.Button("Options")) currentPage = Page.Options;
    if (GUILayout.Button("Credits")) currentPage = Page.Credits;
    if (GUILayout.Button("Quitter")) Application.Quit();
    EndPage();
  }

  void PauseGame() {
    savedTimeScale = Time.timeScale;
    Time.timeScale = 0;
    AudioListener.pause = true;
    currentPage = Page.Main;
    GameObject player = GameObject.FindWithTag("Player");
    player.SendMessage("enableImageEffects");
    Screen.showCursor = true;
  }

  void UnPauseGame() {
    Time.timeScale = savedTimeScale;
    AudioListener.pause = false;
    currentPage = Page.None;
    GameObject player = GameObject.FindWithTag("Player");
    player.SendMessage("disableImageEffects");
    Screen.showCursor = false;
  }

  bool IsGamePaused() { return Time.timeScale==0; }

  void OnApplicationPause(bool pause) {
    if (IsGamePaused()) AudioListener.pause = true;
  }
  void NiveauExplications() {
    Statut visiteur = GameObject.FindWithTag("Player").GetComponent<Statut>();
    visiteur.explicationArchi = GUI.Toggle (new Rect (20, 20, 100, 30), visiteur.explicationArchi, "Architecture");
    visiteur.explicationPeinture = GUI.Toggle (new Rect (20, 60, 100, 30), visiteur.explicationPeinture, "Peinture");
    visiteur.explicationReligion = GUI.Toggle (new Rect (20, 100, 100, 30), visiteur.explicationReligion, "Religion");
  }
}
```

fichier D.11 : unity/MouseLook.cs

```
using UnityEngine;
using System.Collections;

/* MouseLook rotates the transform based on the mouse delta. Minimum and Maximum
 * values can be used to constrain the possible rotation.
 * Pour un personnage de jeu vidéo, on ne le fait tourner qu'autour de Y (ce qui
 * correspond à LookX) et on lui ajoute comme enfant une caméra qui peut elle
 * tourner autour de X (ce qui correspond à LookY) donc « pencher la tête » sans
 * que tout le corps du personnage pivote avec.
 */
[AddComponentMenu("Camera-Control/Mouse Look")]
public class MouseLook  : MonoBehaviour {
  public enum RotationAxes {MouseXAndY, MouseX, MouseY}
  public RotationAxes axes = RotationAxes.MouseXAndY;
  public float sensitivityX = 15F;
  public float sensitivityY = 15F;
  public float minimumX = -360F;
  public float maximumX = 360F;
  public float minimumY = -60F;
  public float maximumY = 60F;

  private float rotationY = 0F;

  void Start() {
    // Make the rigid body not change rotation
    if (rigidbody) rigidbody.freezeRotation = true;
  }

  void Update() {
    if (axes == RotationAxes.MouseXAndY) {
      float rotationX = transform.localEulerAngles.y +
        Input.GetAxis("Mouse X") * sensitivityX;
      rotationY += Input.GetAxis("Mouse Y") * sensitivityY;
      rotationY = Mathf.Clamp (rotationY, minimumY, maximumY);
      transform.localEulerAngles = new Vector3(-rotationY, rotationX, 0);
    }
```

```
    else if (axes == RotationAxes.MouseX)
      transform.Rotate(0, Input.GetAxis("Mouse X") * sensitivityX, 0);
    else {
      rotationY += Input.GetAxis("Mouse Y") * sensitivityY;
      rotationY = Mathf.Clamp (rotationY, minimumY, maximumY);
      transform.eulerAngles = new Vector3(rotationY, transform.eulerAngles.y, 0);
    }
  }
}
```

fichier D.12 : unity/NomsObjets.cs

```
using UnityEngine;
using System.Xml;
using System.Collections;

/* Lorsqu'on appelle sa fonction getData, cette classe se connecte au serveur
 * courant et remplit une table associative nom court -> nom long avec la liste
 * des objets, information à laquelle on peut accéder avec fullname(). */
public class NomsObjets  : MonoBehaviour {
  private Hashtable objets;

  IEnumerator getData() {
    string url = PlayerPrefs.GetString("url_contenu")+"getxml.php";
    WWW info = new WWW(url);
    yield return info;
    if (info.error == null) {
      XmlDocument xml = new XmlDocument();
      Debug.Log(info.text);
      xml.LoadXml(info.text);
      XmlNode root = xml.DocumentElement;
      objets = new Hashtable(root.ChildNodes.Count);
      foreach (XmlNode obj in root.ChildNodes) {
        objets.Add(obj.Attributes["name"].Value, obj.Attributes["fullname"].Value);
      }
    }
    else Debug.LogWarning("Echec en accedant a "+url +"  : "+info.error);
  }

  public string fullname(string name) { return objets[name].ToString(); }
}
```

fichier D.13 : unity/OscReceive.cs

```
using UnityEngine;
using System.Collections;
using Bespoke.Common.Osc;

/* Reçoit des informations de position et de rotation depuis une source mobile
 * et les applique à l'objet auquel il est attaché. */
public class OscReceive  : MonoBehaviour {
  public string sendAddress1 = "/1/toggle1";
  public string sendAddress2 = "/test2";
  public float oscData;
  private bool ip_once = true;

  //need osc server to receive osc messages
  private OSCServer oscServer;
  private Quaternion rot;
  private Vector3 pos;

  void Start() {
    //Get the singleton OSCServer instance
    oscServer = OSCServer.Instance;
    rot = Quaternion.identity;
    pos = Vector3.zero;
  }

  /* Demande une première fois d'où viennent les messages et lance ensuite la
   * réception. */
  void OnGUI() {
    if (ip_once) {
      GUI.Label(new Rect(40, 40, 100, 40), "IP de provenance  :");
      oscServer.serverIP = GUI.TextField(new Rect(150, 40, 120, 40),
          oscServer.serverIP, 16);
```

```
      if (GUI.Button(new Rect (300, 40, 60, 40), "OK")) {
        ip_once = false;
        //Register methods we will handle
        oscServer.RegisterMethod(sendAddress1);
        oscServer.RegisterMethod(sendAddress2);
        //Sets our message received handler
        oscServer.AddMessageReceivedCallback(new OscMessageReceivedHandler(myCallback));
        oscServer.StartServer();
      }
    }
  }

  void Update() {
    float smooth = 2f;
    transform.rotation = Quaternion.Slerp(transform.rotation, rot,
        Time.deltaTime * smooth);
    transform.position = pos;
  }
  //Callback method to handle osc messages
  void myCallback(object sender, OscMessageReceivedEventArgs e) {
    Debug.Log("message recu  : " + e.Message.Address);
    /* puisqu'on reçoit des chaines comme message, on supprime les parenthèses
     * autour et on récupère chaque composante séparée des autres par une
     * virgule avant de la convertir en flottant. */
    string s;
    string[] vals;
    switch (e.Message.Address) {
      case "rotation" :
        s = e.Message.At<string>(0);
        vals = s.Substring(1, s.Length - 2).Split(',');
        rot.Set(System.Convert.ToSingle(vals[0]),
            System.Convert.ToSingle(vals[1]),
            System.Convert.ToSingle(vals[2]),
            System.Convert.ToSingle(vals[3]));
        break;
      case "position" :
        s = e.Message.At<string>(0);
        vals = s.Substring(1, s.Length - 2).Split(',');
        pos.Set(System.Convert.ToSingle(vals[0]),
            System.Convert.ToSingle(vals[1]),
            System.Convert.ToSingle(vals[2]));
        break;
      default :
        break;
    }
  }
}
```

fichier D.14 : unity/OscSend.cs

```
using UnityEngine;
using System.Collections;
using System.Net;
using Bespoke.Common.Osc;

/* Envoie des informations de position et de rotation à un serveur OSC. */
public class OscSend  : MonoBehaviour {
  //Send address and port
  public string sendAddress = "192.168.0.101";
  public int sendPort = 8000;
  public string sendPath = "/rotation";
  public string sendPath2 = "/position";

  //OSC client to pack and send osc messages
  private OscClient oscClient;
  private IPEndPoint sendDestination;
  private bool ip_once = true;
  private float fov = 70f;
  private string sfov;

  void Start() {
    //IP and port to send data on
    changerIp(sendAddress);
    oscClient = new OscClient();
  }
```

```
  void changerIp(string ip) {
    sendDestination = new IPEndPoint(IPAddress.Parse (ip), sendPort);
    Debug.Log("J'envoie maintenant mes messages a  : " + sendDestination);
  }

  void OnGUI() {
    /* Demande une fois où envoyer les messages */
    if (ip_once) {
      GUI.Label(new Rect(40, 40, 100, 40), "Destination IP");
      sendAddress = GUI.TextField(new Rect(150, 40, 120, 40), sendAddress, 16);
      if (GUI.Button(new Rect(300, 40, 60, 40), "OK")) {
        ip_once = false;
        changerIp(sendAddress);
      }
    }
    /* Permet de régler l'angle de vue de la caméra, textuellement ou en faisant
     * glisser un curseur. */
    GUILayout.BeginArea(new Rect(Screen.width-50,25,50,Screen.height-50));
    GUILayout.BeginVertical();
    fov = GUILayout.VerticalSlider(fov, 165f, 35f);
    sfov = fov.ToString();
    sfov = GUILayout.TextField(sfov, (fov > 100f)?5 :4);
    GUILayout.EndVertical();
    GUILayout.EndArea();
    if (!Mathf.Approximately(fov, System.Convert.ToSingle(sfov)))
      fov = System.Convert.ToSingle(sfov);
    Camera.mainCamera.fov = fov;
  }

  /* Envoie les messages contenant la position et la rotation */
  void Update() {
    //Create a new osc message
    OscMessage sMessage = new OscMessage(sendDestination, sendPath, oscClient);
    sMessage.Append<string>(Camera.mainCamera.transform.rotation.ToString());
    sMessage.Send(sendDestination);
    //Create a another osc message
    OscMessage sMessage2 = new OscMessage(sendDestination, sendPath2, oscClient);
    sMessage2.Append<string>(Camera.mainCamera.transform.position.ToString());
    sMessage2.Send(sendDestination);
  }
}
```

fichier D.15 : unity/OSCServer.cs

```
using UnityEngine;
using System.Net;
using System.Collections;
using Bespoke.Common.Osc;

/* Cette classe est un wrapper de Bespoke.Common.Osc.OscServer qui fournit en
 * plus un singleton. */
public class OSCServer  : MonoBehaviour {

  public int ServerPort = 8000;
  public string serverIP = "192.168.0.101";

  private static OSCServer instance = null;

  public static OSCServer Instance {
    get {
      if (instance == null) {
        Debug.Log("instantiate OscServer");
        GameObject go = new GameObject();
        instance = go.AddComponent<OSCServer>();
        go.name = "singleton";
      }
      return instance;
    }
  }

  private IPAddress ServerAddress;
  private OscServer sOscServer;

  void Start () {
    instance = this;
    ServerAddress = IPAddress.Parse(serverIP);
```

```
    sOscServer = new OscServer(TransportType.Udp, ServerAddress, ServerPort);
  }

  public void StartServer() {
    sOscServer.Start();
    Debug.Log(System.String.Format("OSC server listening on {0} ://{1} :{2} for  :",
         sOscServer.TransmissionType, sOscServer.IPAddress, sOscServer.Port));
    foreach(string method in sOscServer.RegisteredMethods) Debug.Log(method);
  }

  public void RegisterMethod(string method) {
    sOscServer.RegisterMethod(method);
  }

  public void AddMessageReceivedCallback(OscMessageReceivedHandler callbackMethod) {
    sOscServer.MessageReceived += callbackMethod;
  }

  public void Stop() {
    sOscServer.Stop();
  }

  public void OnDestroy() {
    sOscServer.Stop();
  }

  void OnApplicationQuit() {
    sOscServer.Stop();
  }
}
```

fichier D.16 : unity/OutputSetup.cs

```
using UnityEngine;
using System.Collections;

/* Les différents mode d'affichage possible et les résolutions possibles. */
public enum OutputConfig {dome_180, dome_210, dome_230, flat, troisEcrans, panoscope};
public enum OutputResolution {_3840x1080, _1920x2160, _2240x2240};

[ExecuteInEditMode]
/* Alterne dynamiquement entre les différents modes d'affichage.*/
public class OutputSetup  : MonoBehaviour {
  public OutputConfig config;
  public OutputResolution resolution;
  public int stitcherLayer;

  /* les stitchers, les caméras et les textures qu'on aura besoin de modifier.*/
  public Camera cameraRigFront;
  public Camera cameraRigLeft;
  public Camera cameraRigRight;
  public Camera stitcherCam_Back;
  public Camera stitcherCam_Front;
  public GameObject cameraRig;
  public GameObject stitcher180_back;
  public GameObject stitcher180_front;
  public GameObject stitcher210_back;
  public GameObject stitcher210_front;
  public GameObject stitcher230_back;
  public GameObject stitcher230_front;
  public GameObject troisPlans;
  public RenderTexture frontTexture;
  public RenderTexture leftTexture;
  public RenderTexture rightTexture;

  private OutputConfig currentConfig;

  void Start() {
    float stitcherScaling = 1.0f;
    Vector3 stitcherBackPos = Vector3.zero;
    Vector3 stitcherFrontPos = Vector3.zero;

    float w = 1.0f*Screen.width;
    float h = 1.0f*Screen.height;
    switch (resolution) {
      case OutputResolution._1920x2160 :
```

```
      stitcherCam_Front.rect = new Rect(0f, 0f, 1f, 1f);
      stitcherCam_Back.enabled = false;
      stitcherScaling = stitcherCam_Front.orthographicSize  * w / h;
      stitcherBackPos = new Vector3(0f, -1f, 0.12f);
      stitcherFrontPos =  new Vector3(0f, 0f, 0.12f);
      break;
    case OutputResolution._3840x1080 :
      stitcherCam_Front.rect = new Rect(0f, 0f, 0.5f, 1f);
      stitcherCam_Back.enabled = true;
      stitcherScaling = stitcherCam_Front.orthographicSize  * w / 2 / h;
      stitcherBackPos = new Vector3(0f, -0.5f, 0.12f);
      stitcherFrontPos = new Vector3(0f, 0.5f, 0.12f);
      break;
    case OutputResolution._2240x2240 :
      stitcherCam_Front.rect = new Rect(0f, -0.5f, 1f, 1f);
      stitcherCam_Back.rect = new Rect(0f, 0.5f, 1f, 1f);
      stitcherCam_Back.enabled=true;
      stitcherScaling = stitcherCam_Front.orthographicSize  *(w / h)*2f;
      stitcherBackPos = new Vector3(0f, -0.5010f, 0.12f);
      stitcherFrontPos = new Vector3(0f, 0.5010f, 0.12f);
      break;
    /* TODO : ajouter une résolution genre 2400x600 et changer la taille de la
     * caméra ou celle de troisPlans pour filmer les 3 textures dans leur
     * entièreté. */
    default :
      print(resolution + "n'est pas pris en charge");
      break;
  }
  // Scale and position stitchers to fit screen
  Vector3 scaleVector = stitcherScaling * Vector3.one;
  stitcher180_back.transform.localScale = scaleVector;
  stitcher180_front.transform.localScale = scaleVector;
  stitcher180_back.transform.localPosition = stitcherBackPos;
  stitcher180_front.transform.localPosition = stitcherFrontPos;
  stitcher210_back.transform.localScale = scaleVector;
  stitcher210_front.transform.localScale = scaleVector;
  stitcher210_back.transform.localPosition = stitcherBackPos;
  stitcher210_front.transform.localPosition = stitcherFrontPos;
  stitcher230_back.transform.localScale = scaleVector;
  stitcher230_front.transform.localScale = scaleVector;
  stitcher230_back.transform.localPosition = stitcherBackPos;
  stitcher230_front.transform.localPosition = stitcherFrontPos;
  // Set stitchers culling
  stitcher180_back.layer = stitcherLayer;
  stitcher180_front.layer = stitcherLayer;
  stitcher210_back.layer = stitcherLayer;
  stitcher210_front.layer = stitcherLayer;
  stitcher230_back.layer = stitcherLayer;
  stitcher230_front.layer = stitcherLayer;
  stitcherCam_Front.cullingMask = 1 << stitcherLayer;
  stitcherCam_Back.cullingMask = 1 << stitcherLayer;

  SwitchConfig(config);
}

/* les touches numériques permettent de changer de configuration lors de
 * l'exécution. */
void Update() {
  if (Input.GetKeyUp (KeyCode.Keypad1) || Input.GetKeyUp (KeyCode.Alpha1))
    config = (OutputConfig)0;
  if (Input.GetKeyUp (KeyCode.Keypad2) || Input.GetKeyUp (KeyCode.Alpha2))
    config = (OutputConfig)1;
  if (Input.GetKeyUp (KeyCode.Keypad3) || Input.GetKeyUp (KeyCode.Alpha3))
    config = (OutputConfig)2;
  if (Input.GetKeyUp (KeyCode.Keypad4) || Input.GetKeyUp (KeyCode.Alpha4))
    config = (OutputConfig)3;
  if (Input.GetKeyUp (KeyCode.Keypad5) || Input.GetKeyUp (KeyCode.Alpha5))
    config = (OutputConfig)4;
  if (Input.GetKeyUp (KeyCode.Keypad6) || Input.GetKeyUp (KeyCode.Alpha6))
    config = (OutputConfig)5;
  if(config != currentConfig)
    SwitchConfig(config);
}

/* Effectue les réglages propres à chaque configuration */
void SwitchConfig(OutputConfig c) {
```

```
    Debug.Log("Switch Config...");
    /* On met d'abord tous les objets dans leur « position initiale » */
    stitcherCam_Back.gameObject.SetActiveRecursively(true);
    stitcherCam_Front.gameObject.SetActiveRecursively(true);
    stitcher180_back.SetActiveRecursively(false);
    stitcher180_front.SetActiveRecursively(false);
    stitcher210_back.SetActiveRecursively(false);
    stitcher210_front.SetActiveRecursively(false);
    stitcher230_back.SetActiveRecursively(false);
    stitcher230_front.SetActiveRecursively(false);
    cameraRig.SetActiveRecursively(true);
    cameraRig.transform.localEulerAngles = new Vector3(90,0,0);
    cameraRigFront.tag = "";
    cameraRigFront.targetTexture = frontTexture;
    cameraRigFront.fieldOfView = 90;
    cameraRigRight.tag = "";
    cameraRigRight.targetTexture = rightTexture;
    cameraRigRight.fieldOfView = 90;
    cameraRigLeft.tag = "";
    cameraRigLeft.targetTexture = leftTexture;
    cameraRigLeft.fieldOfView = 90;
    stitcherCam_Back.tag = "MainCamera";
    stitcherCam_Front.tag = "MainCamera";
    Vector3 rr = new Vector3(-0.09134388f, 0.005270023f, 0.03524756f);
    cameraRig.transform.localPosition = rr;
    /* Puis on active/déplace ceux qui vont jouer un rôle */
    switch(c) {
      case OutputConfig.dome_180 :
        stitcher180_back.SetActiveRecursively(true);
        stitcher180_front.SetActiveRecursively(true);
        break;
      case OutputConfig.dome_210 :
        stitcher210_back.SetActiveRecursively(true);
        stitcher210_front.SetActiveRecursively(true);
        break;
      case OutputConfig.dome_230 :
        stitcher230_back.SetActiveRecursively(true);
        stitcher230_front.SetActiveRecursively(true);
        break;
      /* Dans ce cas, on arrête de rendre dans des textures et on ne récupère
       * que l'image de la caméra de devant (qui devient principal) pour
       * l'afficher directement à l'écran. */
      case OutputConfig.flat :
        cameraRig.SetActiveRecursively(false);
        stitcherCam_Back.tag = "";
        stitcherCam_Front.tag = "";
        stitcherCam_Back.gameObject.SetActiveRecursively(false);
        stitcherCam_Front.gameObject.SetActiveRecursively(false);
        cameraRigFront.tag = "MainCamera";
        cameraRigFront.fieldOfView = 70;
        cameraRigFront.targetTexture = null;
        cameraRigFront.gameObject.SetActiveRecursively(true);
        break;
      /* On envoie les textures de gauche, devant et droite sur un rectangle qui
       * est filmé de dessus. */
      case OutputConfig.troisEcrans :
        stitcherCam_Back.gameObject.SetActiveRecursively(false);
        stitcherCam_Front.gameObject.SetActiveRecursively(false);
        troisPlans.SetActiveRecursively(true);
        break;
      /* En première approximation, le panoscope correspond au dome 180, sauf
       * qu'on renverse l'anneau et qu'on le remonte pour ne pas voir trop de
       * plancher. */
      case OutputConfig.panoscope :
        stitcher180_back.SetActiveRecursively(true);
        stitcher180_front.SetActiveRecursively(true);
        cameraRig.transform.Translate(new Vector3(0f, 8f, 0f));
        cameraRig.transform.localEulerAngles = new Vector3(-90,0,0);
        break;
    }
    currentConfig = c;
  }
}
```

fichier D.17 : unity/OuvrirPorte.cs

```
using UnityEngine;
using System.Collections;

/* Affecté à une zone Trigger, ce script permet de jouer l'unique animation de
 * son objet parent dans un sens puis l'autre, en la bloquant à la fin tant que
 * Player reste dans la zone, et en jouant éventuellement un son à chaque fois. */
public class OuvrirPorte  : MonoBehaviour {
  public float dureeOuverture = 1.5f;
  public float dureeFermeture = 2.3f;
  public AudioClip sonOuverture;
  public AudioClip sonFermeture;

  private Animation anims;
  private string clip_name;
  private float open_speed;
  private float close_speed;
  private AudioSource paudio;

  /* On récupère les composants nécessaire à la suite et on convertit les durées
   * en vitesse. */
  void Start () {
    anims = transform.parent.animation;
    if (transform.parent.audio == null)
      transform.parent.gameObject.AddComponent<AudioSource>();
    paudio = transform.parent.audio;
    clip_name = anims.clip.name;
    open_speed = anims[clip_name].length/dureeOuverture;
    close_speed = -anims[clip_name].length/dureeFermeture;
  }

  void OnTriggerEnter(Collider other) {
    if (other.gameObject.CompareTag ("Player")) {
      anims[clip_name].speed = open_speed;
      anims.CrossFade(clip_name);
      paudio.Stop();
      if (sonOuverture != null) {
        paudio.clip = sonOuverture;
        paudio.Play();
      }
    }
  }

  void OnTriggerExit(Collider other) {
    if (other.gameObject.CompareTag ("Player")) {
      anims[clip_name].speed = close_speed;
      /* Si l'animation est déjà revenue au début après s'être terminée, on la
       * replace à la fin, sinon on laisse le temps courant pour être continu
       */
      if (anims[clip_name].time < .01f)
        anims[clip_name].time = anims[clip_name].length;
      anims.CrossFade(clip_name);
      paudio.Stop();
      if (sonFermeture != null) {
        paudio.clip = sonFermeture;
        paudio.Play();
      }
    }
  }
}
```

fichier D.18 : unity/Peuple.cs

```
/* Géraud Le Falher <daureg@gmail.com>
 * 2012-05-28
 */
using UnityEngine;
using System.Collections;

/* On construit cet objet en lui fournissant trois réels qui représentent des
 * bornes d'intervalles centrés en zéro. Ensuite, il peut renvoyer une rotation
 * aléatoire autour de l'axe Y par rapport à une rotation initiale dans les
 * limites fournies à la construction. De même pour une position aléatoire dans
 * le plan XZ autour d'un point de départ. */
public class Variation {
  private float rotV;
  private float pxV;
```

```
  private float pzV;
  private System.Random random;
  public Variation(float r, float x, float z) {
    random = new System.Random();
    rotV = r;
    pxV = x;
    pzV = z;
  }
  private float plus_ou_moins_un() {
    return (float)random.NextDouble()*2.0f -1.0f;
  }
  public Vector3 random_pos(Vector3 init) {
    return new Vector3(pxV*plus_ou_moins_un()+init.x, init.y,
        pzV*plus_ou_moins_un()+init.z);
  }
  public Quaternion random_rot(Vector3 init) {
    return Quaternion.Euler(init.x, init.y + rotV*plus_ou_moins_un(), init.z);
  }
}
/* Les modes de placement sont prédéfinis par le type d'objet considéré. */
public enum Quoi {Chaises, Bancs, ChaisesFond, BancsHauteur};

/* Place des copies d'un objet en ajoutant un peu d'aléatoire. */
public class Peuple  : MonoBehaviour {
  public GameObject objet;
  public Quoi quelType = Quoi.Chaises;
  /* les variabilités, en degré pour la rotation et en % entre 0 et 1 de la
   * taille de l'objet pour les positions. */
  public float rotation;
  public float posX;
  public float posZ;

  private Variation aleas;
  private Vector3 box;
  private Transform tr;

  void Start () {
    tr = objet.transform;
    box = objet.GetComponent<BoxCollider>().size;
    box.Scale(tr.localScale);
    aleas = new Variation(rotation, posX*box.x, posZ*box.z);
    switch (quelType) {
    case Quoi.Chaises :
      place(3, 3, -1.14f, .13f, true, false);
      place(3, 3, -1.14f, .13f, true, true);
      break;
    case Quoi.Bancs :
      place(10, 1, -.13f, .58f, false, false);
      break;
    case Quoi.ChaisesFond :
      place(2, 4, -1.14f, 10.0f, false, false);
      break;
    case Quoi.BancsHauteur :
      place(13, 1, -.58f, 0.0f, false, false);
      place(13, 1, -.58f, 0.0f, false, true, 12.05f);
      break;
    default :
      break;
    }
  }
  void place(int lignes, int colonnes, float dz, float dx, bool ecart, bool
      droite, float decal_symetrie=11.2f) {
    Vector3 veci = (dz + Mathf.Sign(dz)*box.z)*tr.forward;
    Vector3 vecj = (dx + Mathf.Sign(dx)*box.x)*tr.right;
    Vector3 vecart = (ecart) ? (1.05f*Mathf.Sign(dx)*box.x)*tr.right  : Vector3.zero;
    if (droite) {
      vecj *= -1.0f;
      vecart *= -1.0f;
    }
    Vector3 debut = tr.position;
    if (droite) {
      debut += decal_symetrie*tr.right;
      Instantiate(objet, debut, aleas.random_rot(tr.rotation.eulerAngles));
    }
    debut += veci;
```

```
    for (int i = 0; i < lignes; i++) {
      for (int j = 0; j < colonnes; j++) {
        Instantiate(objet, debut + aleas.random_pos(i*veci + j*(vecj+vecart)),
            aleas.random_rot(tr.rotation.eulerAngles));
        if (ecart)
          Instantiate(objet, debut + aleas.random_pos(i*veci + vecart +
                j*(vecj+vecart)), aleas.random_rot(tr.rotation.eulerAngles));
      }
    }
  }
}
```

fichier D.19 : unity/PlacePlane.cs

```
using UnityEngine;
using System.Collections;

public class PlacePlane  : MonoBehaviour {
  private Mesh mesh;
  private Vector3[] vertices;
  private Matrix4x4 meshMatrix;

  /* Ajuste les coordonnées des sommets du plan pour qu'ils coïncident avec le
   * rectangle GUI projeté sur 'refPlane'
   */
  public void Change(Camera cam, Rect pos, GameObject refPlane) {
    meshMatrix = transform.worldToLocalMatrix;
    mesh = GetComponent<MeshFilter>().mesh;
    vertices = mesh.vertices;
    float h = cam.pixelHeight;

    RaycastHit hit;
    // haut gauche
    Ray ray = cam.ScreenPointToRay(new Vector3(pos.xMin, h - pos.yMin, 0));
    refPlane.collider.Raycast(ray, out hit, 100);
    vertices[3] = meshMatrix.MultiplyPoint3x4(hit.point);
    // bas gauche
    ray = cam.ScreenPointToRay(new Vector3(pos.xMin, h - pos.yMax, 0));
    refPlane.collider.Raycast(ray, out hit, 100);
    vertices[1] = meshMatrix.MultiplyPoint3x4(hit.point);
    // bas droite
    ray = cam.ScreenPointToRay(new Vector3(pos.xMax, h - pos.yMax, 0));
    refPlane.collider.Raycast(ray, out hit, 100);
    vertices[0] = meshMatrix.MultiplyPoint3x4(hit.point);
    // haut droite
    ray = cam.ScreenPointToRay(new Vector3(pos.xMax, h - pos.yMin, 0));
    refPlane.collider.Raycast(ray, out hit, 100);
    vertices[2] = meshMatrix.MultiplyPoint3x4(hit.point);

    mesh.vertices = vertices;
    mesh.RecalculateBounds();
  }
}
```

fichier D.20 : unity/Pointer.cs

```
/* Commentaire de la SAT  :
 * Ce script doit être assigné à l'objet Pointer ayant comme enfant l'objet
 * cursor et comme parent le cameraRig. Il est important d'intégrer le module
 * OSC comprenant l'oscControler avant d'intégrer le script Pointer. Le script
 * fait un raycast sur tous les objets. Il change la valeur des variables
 * pressing et pointerOver du script selection.
 * TODO : faire un script seulement pour l'ipod et un seulement pour joystick à
 * partir du présent script.
 */
/* Modification au LAMIC par Géraud Le Falher <daureg@gmail.com>  :
 * - le OscController est en commentaire parce qu'il ne servait pas notre cas
 *   mais rajoutait plutôt des problèmes.
 * - ajout de la variable booléenne 'actif', qui permet de désactiver le lancer
 *   de rayon et les clics, sans désactiver tout le script.
 * - la texture n'alterne plus entre deux états, pour permettre de la remplacer
 *   par un film lors de l'attente du chargement d'un objet.
 */
using UnityEngine;
using System.Collections;
```

```
public class Pointer  : MonoBehaviour {
  public ControllerType controllerType = ControllerType.OscController;
  public GameObject cursor;
  public GameObject objectToFollow;
  public float pointerSpeed = 2f;
  public Texture2D pointerUpImage;
  public Texture2D pointerDownImage;
  public bool actif = true;

  //private OscController oscController;
  private PointerHandler selectionScript;

  void Start () {
    //oscController = OscController.Instance;
    if (cursor == null) cursor  = transform.Find("Cursor").gameObject;
  }

  void Update () {
    // position of the pointer
    transform.position=objectToFollow.transform.position;
    ////orientation of the cursor
    /*
    if(oscController.orientationChanged)
    {
      transform.rotation = oscController.orientation;
    }

    else
    {
    */

    float x = transform.rotation.eulerAngles.x + Input.GetAxis("RightVertical") * pointerSpeed;
    float y = transform.rotation.eulerAngles.y + Input.GetAxis("RightHorizontal") * pointerSpeed;
    float z = transform.rotation.eulerAngles.z;
    Quaternion rot = Quaternion.Euler(x, y, z);
    transform.rotation = rot;
    //}


    // Store all the click buttons in one variable
    bool userIsClicking = false;
    if (actif) cursor.renderer.enabled = true;
    else {
      cursor.renderer.enabled = false;
      return;
    }
    if(Input.GetButton("Fire1") )//|| oscController.button1 == 1)
      userIsClicking=true;
    else
      userIsClicking=false;


    ///image of the cursor

    /*
    if(userIsClicking)
      cursor.renderer.material.mainTexture = pointerDownImage;
    else
      cursor.renderer.material.mainTexture = pointerUpImage;
    */

    Vector3 org = transform.position;
    Vector3 end = cursor.transform.position - org;
    Ray ray = new Ray(org, end);
    RaycastHit hit;
    ////If pointer is OVER something
    if(Physics.Raycast(ray, out hit)) {
      selectionScript = hit.collider.GetComponent<PointerHandler>(); //get selection script
      if (selectionScript != null) {//check if selection script exists
        //set the over variable of selection script to true
        selectionScript.pointerOver=true;
        if(userIsClicking) //check if visitor is holding a click
          //the user is clicking now, set the click variable of selection script to true
          selectionScript.pressing=true;
        if (!userIsClicking) //if user stopped pressing the click button
```

```
          selectionScript.pressing=false;
      }
    }
  }
}
```

fichier D.21 : unity/PointerEvents.cs

```
/* Base class to handle pointer events. Override these methods in a derived
 * class and use with a PointerHandler to handle pointer events. */
using UnityEngine;
using System.Collections;
public class PointerEvents  : MonoBehaviour {
  public virtual void PointerDown(){}
  public virtual void PointerUp(){}
  public virtual void PointerIn(){}
  public virtual void PointerOut(){}
}
```

fichier D.22 : unity/PointerHandler.cs

```
using UnityEngine;
using System.Collections;

/* Adaptation du script selection.cs de la SAT en supprimant les changement de
 * couleur mais en utilisant l'interface PointerEvents (ie en faisant appel aux
 * fonctions de handler au bon moment).
 */
public class PointerHandler  : MonoBehaviour {
  public PointerEvents handler;
  [HideInInspector]
  public bool over=false; //pointer over ?
  public bool click=false; //pointer over et click ??
  public bool holdClick=false; //pointer over et click maintenu ??
  public bool pressing=false; //pointer script will change this value if over and pressing
  public bool pointerOver=false; //pointer script will change this value if over
  public bool actif=true; //pointer script will change this value if over
  //value used to reset the click variable so click is true only one frame
  private bool clicked=false;

  void Start() {
    if (gameObject.GetComponent<PointerEvents>() != null && handler == null) {
      handler = gameObject.GetComponent<PointerEvents>();
    }
  }

  void Update () {
    if (!actif) return;
    /* reset click variable so it stays true for a single frame only (will be
     * set to true later in the script if click) */
    click=false;
    if (pointerOver == true) { //if the pointer is over
      handler.PointerIn();
      over=true;
      if (pressing == true) {//pointer is over and pressing
        holdClick=true; //check if user was pressing last frame
        if (clicked == false) {
          click=true; //click is true only if user was not pressing last frame
          clicked=true;
          handler.PointerDown();
        }
      }
      else { //if pointer is over but not pressing reset these values
        holdClick=false;
        clicked=false;
      }
    }
    else { //if pointer is not over :
      //if pointer was over last frame, it means it's leaving now
      if (over == true) handler.PointerOut();
      over=false; // if pointer is not over reset those values to false
      click=false;
      clicked=false;
    }
    /* put pressing and pointerOver to false so pointer script can put them back
```

```
     * on or not next frame. */
    pressing=false;
    pointerOver=false;
  }
}
```

fichier D.23 : unity/Statut.cs

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;

/* Cette classe stocke divers variables décrivant la situation du visiteur et
 * accessible de l'extérieur, ainsi que la liste de tous les matériaux des
 * objets possédant le tag "Chapelle" afin de pouvoir les noircir et leur rendre
 * leur couleur originel lorsque le visiteur observe un objet.
 * TODO : ce couplage est assez désorientant et il faudrait y remédier.
 */
public class Statut  : MonoBehaviour {
  public bool explicationArchi;
  public bool explicationReligion;
  public bool explicationPeinture;
  public bool admire;
  public bool interface2D;
  public string objet;
  public List<Material> mat_vers_noir;
  public Hashtable couleurs_debut;
  public GameObject[] chapelles;
  public Color noirceur = new Color(.09f, .09f, .09f, 1f);

  private bool already_did_it = false;
  private float nextFire = 0.0F;

  void Awake() {
    interface2D = true;
    objet = "none";
  }

  /* Un appuie sur la touche I bascule entre interface en 2D et en 3D. */
  void OnGUI() {
    if (Input.GetKeyDown(KeyCode.I) && Time.time > nextFire) {
      nextFire = Time.time + 1f;
      interface2D = !interface2D;
    }
  }
  /* La première fois que cette méthode est appelée, elle récupère tous les
   * matériaux de la chapelle. */
  public void refChapelle() {
    if (already_did_it) return;
    already_did_it = true;
    chapelles = GameObject.FindGameObjectsWithTag("Chapelle");
    mat_vers_noir = new List<Material>();
    couleurs_debut = new Hashtable();
    foreach (GameObject g in chapelles) {
      foreach (Renderer r in g.GetComponentsInChildren<Renderer>()) {
        foreach (Material m in r.materials) {
          try {
            couleurs_debut.Add(m.name,  new Color(m.color.r, m.color.g,
                  m.color.b, m.color.a));
          }
          catch (System.ArgumentException) {}
          mat_vers_noir.Add(m);
        }
      }
    }
  }
}
```

fichier D.24 : unity/TextureState.cs

```
using UnityEngine;
using System.Collections;

/* Permet d'afficher le sanctuaire avec un texture quand on est dans la chapelle
 * privée et couleur pleine le reste du temps. */
```

```
public class TextureState  : MonoBehaviour {
  public GameObject sanctuaireBas;
  public GameObject sanctuaireHaut;
  private Material mat;
  private Color col;
  private Texture tex;

  void Start() {
    mat = sanctuaireHaut.renderer.material;
    col = sanctuaireBas.renderer.material.color;
    mat.color = col;
    tex = mat.mainTexture;
    mat.mainTexture = null;
  }
  void OnTriggerEnter(Collider other) {
    if (other.tag == "Player") {
      mat.mainTexture = tex;
      mat.color = Color.white;
    }
  }
  void OnTriggerExit(Collider other) {
    if (other.tag == "Player") {
      mat.mainTexture = null;
      mat.color = col;
    }
  }
}
```

fichier D.25 : unity/VoitTableau.cs

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;

public enum TextLayout { DEUX_COLONNES, UNE_COLONNE, PLEINE_LARGEUR };

[RequireComponent (typeof (PointerHandler))]
[RequireComponent (typeof (GetObject))]
[RequireComponent (typeof (Collider))]

/* Cette classe, qui a pris beaucoup d'embonpoint au cours du temps, avait
 * pourtant un objectif simple, amener l'utilisateur devant l'objet sur lequel
 * il a cliqué et lui présenter le contenu correspondant. Comment ces deux
 * lignes d'explication se sont changées en plus de 550 lignes de codes, cela
 * reste un profond mystère...
 * TODO : Puisqu'elle ne tient aucun compte de la nature de l'objet dans son
 * fonctionnement, sinon par quelque variables publiques, on devrait pouvoir
 * s'épargner d'avoir à en faire tourner une instance par objet en la
 * factorisant, ce qui devrait améliorer les performances.
 * TODO : Nettement plus simple, mais que je me refuse à faire par
 * sentimentalisme, changer son nom en VoirObjet.
 */
public class VoitTableau  : PointerEvents {
  /* TODO : Peut-être serait-il plus simple de récupérer par le code tous les
   * enfants qui ont le composant lumière (avec GetComponentsInChildren) et le
   * transform appelé pdv (transform.Find("pdv")) pour remplir les variables
   * lumieres et destination plutôt que de le faire dans l'interface d'Unity ?
   * cf http ://en.wikipedia.org/wiki/Convention_over_configuration
   */
  public Light[] lumieres;
  public float intensiteMax = 8.0f;
  public Transform destination;
  public float dureeAnimation = 2.3f;
  public AnimationCurve tempoAnimationAller;
  public AnimationCurve tempoAnimationRetour;
  public float distanceMax = 10.0f;
  public Material videoMat;
  public GameObject objetAvant;
  public bool invisible = false;
  public GUISkin skin;
  public bool interface2D;
  public bool attend = true;

  private string fullname;
  private bool send_request = false;
  private float temps;
```

```
private bool anim;
private bool admire;
private bool idle;
private bool versObjet;
private Vector3 position_debut;
private Vector3 decalageTete;
private GameObject player;
private Statut visiteur;
private GetObject infosObjet;
private Camera cam;
private Rect obj_size;
private List<Bloc> imgEcran;
private List<Texture> texs;
private List<GameObject> sonEcran;
private List<Rect> epos;
private string txt, txt2;
private Vector2 txt_size;
private Rect utile;
private int ecranCourant = 0;
private float prochainEcran = .0f;
private float dureeEcran = 0.5f;
private HGui gui;
//TODO utiler cam.nearClipPlane*1.3f plutôt (sans UGLY_CAMILLE_HACK du coup)
private float IDC = .5f;
private float distanceObjet;
private List<Bouton> controles;
private int bouton_w = HGui.bouton_w;
private int bouton_h = HGui.bouton_h;
private Vector3 camMvt;
private Material cursorMat;
private Texture2D chapelle_img;
private Texture2D suivant_img;
private Texture2D precedent_img;
private Texture2D commentaires_img;
private TextLayout dispoTexte;
private bool seulMvtParVisite = false;
private TextMesh tooltip;
private Color objetCouleurAvant;
private bool goingBack = false;

/* Comme on a beaucoup (trop) de membres privés, cela prend beaucoup de lignes
 * de tous les initialiser. */
void Start () {
  tempoAnimationAller = new AnimationCurve(new Keyframe(0, 0), new Keyframe(1, 1));
  tempoAnimationRetour = new AnimationCurve(new Keyframe(0, 1), new Keyframe(1, 0));
  fullname = "";
  if (renderer.bounds.size.magnitude < .01f && !invisible)
          Debug.LogWarning(gameObject.name + " est très petit. Peut-être "+
              "faudrait-il le déclarer invisible dans VoitTableau");
  anim = false;
  admire = false;
  versObjet = true;
  player = GameObject.FindWithTag("Player");
  visiteur = player.GetComponent<Statut>();
  cam = GameObject.Find("FrontCam").camera;
  cam.backgroundColor = visiteur.noirceur;
  float h = cam.pixelHeight;
  float w = cam.pixelWidth;
  gui = new HGui(cam);
  decalageTete = new Vector3(0, cam.transform.localPosition.y, 0);
  Raz();
  objetCouleurAvant = new Color(renderer.material.color.r,
      renderer.material.color.g, renderer.material.color.b,
      renderer.material.color.a);

  /* Prépare les variables qui vont stocker les contenus */
  infosObjet = GetComponent<GetObject>();
  imgEcran = new List<Bloc>();
  texs = new List<Texture>(5);
  sonEcran = new List<GameObject>();
  obj_size = new Rect(w,h/2f,0,0);
  tooltip =cam.transform.Find("tooltip").GetComponent<TextMesh>();
  /* Charge les icônes et crée les boutons pour naviguer entre les écrans. */
  controles = new List<Bouton>(4);
  chapelle_img = (Texture2D)Resources.Load("chapelle", typeof(Texture2D));
  suivant_img = (Texture2D)Resources.Load("suivant", typeof(Texture2D));
```

```
  precedent_img = (Texture2D)Resources.Load("precedent", typeof(Texture2D));
  commentaires_img = (Texture2D)Resources.Load("commentaires", typeof(Texture2D));
  controles.Add(new Bouton {icon=chapelle_img, action=do_retour, gui=gui,
      pos = new Rect(w-bouton_w-5,20,bouton_w,bouton_h),
      icon_h=(Texture2D)Resources.Load("chapelle_h", typeof(Texture2D))});
  controles.Add(new Bouton {icon=suivant_img, action=do_suivant, gui=gui,
      pos = new Rect(w-bouton_w-5, h/2-1.6f*bouton_h, bouton_w, bouton_h),
      icon_h=(Texture2D)Resources.Load("suivant_h", typeof(Texture2D))});
  controles.Add(new Bouton {icon=precedent_img, action=do_precedent, gui=gui,
      pos = new Rect(w-bouton_w-5,h/2-0.6f*bouton_h,bouton_w,bouton_h),
      icon_h=(Texture2D)Resources.Load("precedent_h", typeof(Texture2D))});
  controles.Add(new Bouton {icon=commentaires_img, action=do_commentaires,
      gui=gui, pos = new Rect(w-bouton_w-5,h-20-bouton_h,bouton_w,bouton_h),
      icon_h=(Texture2D)Resources.Load("commentaires_h", typeof(Texture2D))});
  /* Choix de l'interface et de la taille des caractères en 2D. */
  if (SystemInfo.deviceType == DeviceType.Handheld) {
    skin.GetStyle("descButton").fontSize = 32;
    interface2D = true;
  }
  else {
    skin.GetStyle("descButton").fontSize = 16;
    interface2D = visiteur.interface2D;
  }
}

/* Mise à zéro du visiteur quand il revient d'un objet. */
private void Raz() {
  ecranCourant = -1;
  temps = 0.0f;
  foreach (Light lumiere in lumieres) lumiere.intensity = 0.0f;
  renderer.enabled = false;
  player.GetComponent<MouseLook>().enabled = true;
  player.GetComponent<CameraFly>().enabled = true;
  player.transform.Find("-CameraRig/-Pointer").GetComponent<Pointer>().actif = true;
  attend = true;
}

/* Appelée à la fin de l'animation */
private void TransitionAdmire() {
  anim = false;
  cam = GameObject.Find("FrontCam").camera;
  gui.cam = cam;
  obj_size = gui.ObjectSize(gameObject, invisible);
  admire = versObjet;
  if (admire) {
    // bounds.center est plus précis que transform.position, d'après la doc
    player.transform.LookAt(renderer.bounds.center);
    do_suivant();
  }
  visiteur.admire = admire; //TODO variable dupliquée
  Screen.showCursor = versObjet;
  versObjet = !versObjet;
}

/* Appelée au début de l'animation, que ce soit vers l'objet ou en le quittant */
private void TransitionAnim() {
  player.GetComponent<CameraGUI>().msg = "";
  if (!invisible) renderer.enabled = true;
  player.GetComponent<MouseLook>().enabled = false;
  player.GetComponent<CameraFly>().enabled = false;
  visiteur.objet = (versObjet) ? name  : "none";
  GetComponent<PointerHandler>().actif = !versObjet;
  if (versObjet) {
    send_request = false;
    if (!interface2D) {
      fullname = "";
      gui.cam.transform.Find("tooltip_back").renderer.enabled = false;
    }
    position_debut = player.transform.position;
    if (objetAvant != null) objetAvant.renderer.enabled = false;
  }
  else { // avant de partir, on remet la camera sur la tête du visiteur
    cam.transform.Translate(-camMvt);
    seulMvtParVisite = false;
    if (objetAvant != null) objetAvant.renderer.enabled = true;
  }
```

```
  Screen.showCursor = !versObjet;
  temps = 0.0f;
  anim = true;
  admire = false;
}

/* Appelée après avoir relâché le clic sur l'objet */
public override void PointerDown() {
  /* de n'importe quel endroit qu'admire l'objet, c'est toujours assez proche
   * pour retourner dans la chapelle, sans qu'il y ait besoin de réfléchir ! */
  bool assez_proche = admire || distanceObjet < distanceMax;
  if (!assez_proche) return;
  /* Si on est assez proche mais pas déjà en train de bouger, on lance
   * l'animation, soit pour aller vers l'objet (si idle est vrai), soit pour
   * retourner à son point de départ (si admire est vrai). Mais avant de
   * bouger, on demander à récupérer le contenu de l'objet et on place le
   * visiteur en attente, avec un petit curseur animé pour l'occuper (et oui,
   * je suis sûr que le visiteur en est friand, sinon ça ferait longtemps
   * qu'on utiliserait un autre symbole ; ça ne peut pas être que de la
   * nostalgie du moyen-âge  : http ://en.wikipedia.org/wiki/Hourglass#History)
   */
  if (attend && !send_request) {
    if (interface2D) player.GetComponent<CameraGUI>().msg = "";
    infosObjet.SendMessage("recuperer_elements");
    send_request = true;
    player.transform.Find("-CameraRig/-Pointer").GetComponent<Pointer>().actif = false;
    Material cursor = GameObject.Find("Cursor").renderer.material;
    cursorMat = cursor;
    GameObject.Find("Cursor").renderer.material = videoMat;
    ((MovieTexture)GameObject.Find("Cursor").renderer.material.mainTexture).Play();
    player.GetComponent<MouseLook>().enabled = false;
    player.GetComponent<CameraFly>().enabled = false;
  }
  if (!anim && !attend) {
    TransitionAnim();
  }
}

/* Affiche l'interface de lecture et de saisi des commentaires. */
void do_commentaires() {
  ecranCourant = 3;
  clean_content();
  player.GetComponent<Commentaires>().enabled = true;
  player.GetComponent<Commentaires>().recuperer_commentaires();
}

/* Efface les contenus actuellement présent dans la scène (donc à l'écran) */
void clean_content() {
  foreach (Bloc img in imgEcran)
    Object.Destroy(GameObject.Find(gui.GetHash(img.path)));
  foreach (GameObject g in sonEcran) Object.Destroy(g);
  imgEcran.Clear();
  texs.Clear();
  sonEcran.Clear();
  GameObject.Find("texte").renderer.enabled = false;
  GameObject.Find("texte2").renderer.enabled = false;
}

/* Efface tous les éléments d'interface */
void clean_ui() {
  if (!interface2D) foreach (Bouton b in controles)
    Object.Destroy(GameObject.Find(gui.GetHash(b.icon.ToString())));
  tooltip.renderer.enabled = false;
  player.GetComponent<Commentaires>().enabled = false;
}

/* Ramène le visiteur à la chapelle */
void do_retour() {
  clean_content();
  clean_ui();
  if (!invisible) renderer.material.color = objetCouleurAvant;
  PointerDown();
}

/* Renvoie les coordonnées monde de l'intersection du rayon partant de 'pos' à
 * l'écran avec le plan parallèle à la caméra situé au niveau de l'objet
```

```
 * observé.  */
private Vector3 hitScreen(Vector2 pos) {
  Transform refPlane = cam.transform.Find("ScreenPlane");
  float ppos = refPlane.localPosition.z;
  distanceObjet = (cam.transform.position - renderer.bounds.center).magnitude;
  refPlane.localPosition = new Vector3(0,0, distanceObjet);
  Ray ray = cam.ScreenPointToRay(new Vector3(pos.x, pos.y, distanceObjet));
  RaycastHit hit;
  refPlane.collider.Raycast(ray, out hit, 20f*distanceObjet);
  refPlane.localPosition = new Vector3(0,0, ppos);
  return hit.point;
}

/* Approche la caméra pour essayer que l'objet occupe la taille 'size' en
 * pixel à l'écran. Cela ne marche que si l'objet est initialement au centre
 * de l'écran.  */
private void changeObjectSize(Vector2 size, float pas=.3f) {
  uint iter = 0;
  const uint max_iter = 7;
  Vector3 previous_pos = cam.transform.position;
  Vector3 mvt = new Vector3(0f, 0f, pas);
  obj_size = gui.ObjectSize(gameObject, invisible);
  while (iter < max_iter && obj_size.width < size.x && obj_size.height < size.y) {
    previous_pos = cam.transform.position;
    cam.transform.Translate(mvt);
    pas *= .85f;
    mvt.Set(0f, 0f, pas);
    obj_size = gui.ObjectSize(gameObject, invisible);
    iter++;
  }
  cam.transform.position = previous_pos ;
  distanceObjet = (cam.transform.position - renderer.bounds.center).magnitude;
  obj_size = gui.ObjectSize(gameObject, invisible);
}

/* Déplace la caméra pour essayer de placer l'objet observé en 'pos' à l'écran
 * (donc pos est en coordonnées écran).  */
private void moveObject(Vector2 pos) {
  if (invisible) return;
  Vector3 dest = hitScreen(pos);
  Vector3 tr = dest - renderer.bounds.center;
  camMvt = cam.transform.InverseTransformDirection(-tr);
  cam.transform.Translate(camMvt);
}

/* Revient à l'écran précédent (en revenant deux écrans en arrière et en
 * faisant suivant, ce qui n'est pas très très élégant... */
void do_precedent() {
  ecranCourant -= 2;
  clean_content();
  clean_ui();
  goingBack = true;
  do_suivant();
}

/* Ajoute à l'objet visualisé un enfant capable de jouer 'son' */
void ajouterSon(string name, AudioClip son) {
  GameObject ap = new GameObject(gui.GetHash(name));
  ap.transform.parent = gameObject.transform;
  ap.AddComponent<AudioSource>();
  ap.audio.clip = son;
  ap.audio.Play();
  sonEcran.Add(ap);
}

/* Passe à l'écran suivant */
void do_suivant() {
  ecranCourant++;
  TextMesh texte = GameObject.Find("texte").GetComponent<TextMesh>();
  TextMesh texte2 = GameObject.Find("texte2").GetComponent<TextMesh>();
  txt = "";
  txt2 = "";
  epos = new List<Rect>();
  if (!invisible) {
    renderer.enabled = true;
    renderer.material.color = objetCouleurAvant;
```

```
    }
    clean_content();
    clean_ui();

    /* Récupération des contenus du niveau courant s'il y a lieu */
    if (ecranCourant < 3) {
      Screen.showCursor = true;
      txt = infosObjet.textes[ecranCourant];
      foreach (Image img in infosObjet.images) {
        if (img.niveau == ecranCourant) {
          imgEcran.Add(img);
          texs.Add(img.tex);
        }
      }
      foreach (Video vid in infosObjet.videos) {
        if (vid.niveau == ecranCourant) {
          imgEcran.Add(vid);
          string filename = HGui.filenameFromPath(vid.path);
          texs.Add((MovieTexture)
              Resources.Load(gameObject.name+"/"+filename, typeof(MovieTexture)));
          ((MovieTexture)texs[texs.Count-1]).Stop();
          if (interface2D) {
            ajouterSon(HGui.filenameFromPath(vid.path),
                ((MovieTexture)texs[texs.Count-1]).audioClip);
          }
        }
      }
      foreach (Audio son in infosObjet.audios) {
        if (son.niveau == ecranCourant) {
          string filename = HGui.filenameFromPath(son.path);
          ajouterSon(filename, (AudioClip)
              Resources.Load(gameObject.name+"/"+filename, typeof(AudioClip)));
        }
      }
      dispoTexte = TextLayout.PLEINE_LARGEUR;

//    /* Si l'écran est dépourvu de contenus, on peut aller au suivant/précédent directement */
      /* TODO : marche en 2D mais l'écran précédent n'est pas correctement
       * nettoyé en 3D, donc on désactive cette fonctionnalité. */
//    if (txt.Length + txt2.Length == 0 && texs.Count == 0 && sonEcran.Count == 0) {
//      /* TODO ne fonctionne sans doute pas si le premier écran est vide, mais
//       * quel intérêt, il a toujours une description sommaire.
//       */
//      if (goingBack) do_precedent();
//      else do_suivant();
//    }
    }

    /* Pour chacun des trois écrans, on va chercher la taille que va occuper le
     * texte en fonction de celle de l'objet et du nombre d'images.
     * TODO : cette étape est un bon candidat à avoir sa fonction à part. */
    if (ecranCourant == 0) {
      if (!seulMvtParVisite && distanceObjet < 6f) {
        moveObject(new Vector2(gui.w - (gui.w/2f - bouton_w)/2f, gui.h/2f));
        seulMvtParVisite = true;
      }
      obj_size = gui.ObjectSize(gameObject, invisible);
      utile = new Rect(40, 40, obj_size.x - 80, gui.h - 80);
      if (!interface2D) {
        texte.characterSize = .017f;
        txt_size = gui.affiche_texte(texte, txt, new Vector3(40, gui.h*.8f, IDC), utile.width);
        txt_size.y *= 1.15f;
      }
    }
    else { /* dans le cas où l'on voudrait que les écrans suivants, censés
            * contenir plus de texte, soit écrit plus petit (ce qui suppose
            * là encore ne pas utiliser UGLY_CAMILLE_HACK. */
      texte.characterSize = .017f;
      texte2.characterSize = .017f;
    }
    if (ecranCourant == 1) {
      utile = new Rect(40, 40, obj_size.x - 80, gui.h - 80);
      texte.text = txt;
      Vector2 char_size = gui.getCharPixelSize(texte, new Vector3(40, gui.h*.7f, IDC));
      List<Texture> lp,ll;
      float np, nl;
```

```
    Layout.trieImages(ref texs, out lp, out ll, out np, out nl);
    uint p = (uint) np;
    uint l = (uint) nl;

    if (p == 0) { // aucun portrait
      if (!interface2D) txt_size = gui.affiche_texte(texte, txt,
          new Vector3(40, gui.h*.7f, IDC), utile.width);
    }
    else {
      if (l == 0) { // que des portraits
        if (!interface2D) txt_size = Layout.colonneMax(char_size, txt, utile);
        dispoTexte = TextLayout.UNE_COLONNE;
      }
      else { // les deux
        int c = Layout.couperChaine(txt, 2/3f, true);
        string t1 = txt.Substring(0,c);
        txt2 = txt.Substring(c);
        txt = t1;
        dispoTexte = TextLayout.DEUX_COLONNES;
        if (!interface2D) {
          txt_size = gui.affiche_texte(texte, txt,
              new Vector3(40, gui.h*.4f, IDC), 2*utile.width/3f);
          Layout.supp_text_size = gui.affiche_texte(texte2, txt2,
                      new Vector3(40+utile.width/3f, gui.h*.7f, IDC), utile.width/3f);
        }
      }
    }
  }

  if (ecranCourant == 2) {
    renderer.material.color = visiteur.noirceur;
    utile = new Rect(40, 40, gui.w - 80 - bouton_w, gui.h -80);
    if(!interface2D) {
      txt_size = gui.affiche_texte(texte, txt,
          new Vector3(40, gui.h*.6f, IDC), utile.width);
      txt_size.y *= 1.05f;
    }
  }

  /* En 3D, on dessine les boutons une fois pour toute et on calcule la
   * position de chaque contenu. */
  if(!interface2D) {
    foreach (Bouton b in controles) b.Draw(IDC);
    if (ecranCourant < 3) {
      Layout.textures = texs;
      epos = Layout.ecran(ecranCourant+1)(utile, txt_size);
      texs = Layout.textures;
    }
  }
  if (invisible) renderer.enabled = false;
  if (ecranCourant < 3 && !interface2D) {
    /* Affichage */
    Rect tmp = new Rect(0,0,0,0);
    int i = 0;
    tmp.Set(epos[i].x, gui.h - epos[i].y, epos[i].width, epos[i].height);
    gui.affiche_texte(texte, txt, new Vector3(epos[i].x, epos[i].y, IDC), tmp.width);
    int size = texs.Count;
    for (i = 1; i <= size; i++) {
      tmp.Set(epos[i].x, gui.h - epos[i].y, epos[i].width, epos[i].height);
      gui.affiche_image(imgEcran[i-1].path, texs[i-1], tmp, IDC);
    }
    if (i<epos.Count) {
      tmp.Set(epos[i].x, epos[i].y, epos[i].width, epos[i].height);
      gui.affiche_texte(texte2, txt2, new Vector3(tmp.x, tmp.y, IDC), tmp.width);
    }
  }
  goingBack = false;
  if (ecranCourant == 3) {
    do_commentaires();
  }
  /* suivant n'est pas affiché dans les commentaires donc ceci n'est pas
   * nécessaire mais je le laisse au cas où je me tromperais, ce serait juste
   * la 150e fois après tout...
  if (ecranCourant > 3) {
    do_retour();
  }
```

```
   */
}

/* Met à jour deux ou trois variables */
void Update() {
  distanceObjet = (cam.transform.position - transform.renderer.bounds.center).magnitude;
  idle = (!anim && !admire);
  interface2D = visiteur.interface2D;
  if (send_request && !attend && idle) {
    GameObject.Find("Cursor").renderer.material = cursorMat;
    TransitionAnim();
  }
}

/* Si on est en animation, on interpole diverses variables vers leur valeurs
 * finales. */
void LateUpdate() {
  if (anim) {
    temps += Time.deltaTime;
    float coeff = (versObjet) ?
      tempoAnimationAller.Evaluate(temps/dureeAnimation)  :
      tempoAnimationRetour.Evaluate(temps/dureeAnimation);
    player.transform.rotation = Quaternion.Slerp(player.transform.rotation,
        Quaternion.LookRotation(transform.position - player.transform.position), coeff);
    player.transform.position = Vector3.Lerp(position_debut,
        destination.position-decalageTete, coeff);
    visiteur.refChapelle();
    foreach (Material m in visiteur.mat_vers_noir) {
      m.color = Color.Lerp((Color)visiteur.couleurs_debut[m.name],
          (Color)visiteur.noirceur, coeff);
    }
    foreach (Light lumiere in lumieres)
      lumiere.intensity = (1-0.5f*coeff)*intensiteMax;
    if (temps > dureeAnimation) {
      TransitionAdmire();
      /* Bon là c'est un peu confus. En effet, quand on quitte l'objet, versObjet
       * est à faux mais toute à la fin de TransitionAdmire(), on change sa
       * valeur donc finalement, on appelle bien Raz() en quittant l'objet. */
      if (versObjet) Raz();
    }
  }
  if (!interface2D && fullname != "") {
    gui.affiche_texte(tooltip, fullname,
        new Vector3(.55f*gui.w, .45f*gui.h, .2f), .35f*gui.w, true);
  }
}

public void OnGUI() {
  GUI.skin = skin;
  if (admire) { // il n'y a des choses à afficher que devant l'objet
    if (interface2D) { /* En 2D, on effectue globalement les mêmes opérations
      de placement et de dessin qu'en 3D dans do_suivant, mais nettement plus
      souvent. */
      if (ecranCourant >= 0 && ecranCourant < 3) {
        Screen.showCursor = true;
        GUIContent g = new GUIContent(txt.Replace("\n", " ").Trim());
        if (g.text.Length == 0) txt_size = Vector2.zero;
        else {
          switch (dispoTexte) {
          case TextLayout.PLEINE_LARGEUR :
            txt_size = new Vector2(utile.width,
                skin.GetStyle("descButton").CalcHeight(g, utile.width));
            break;
          case TextLayout.UNE_COLONNE :
            float ww = 100f;
            while (ww < 2*utile.width/3f &&
                skin.GetStyle("descButton").CalcHeight(g, ww) > utile.height) {
              ww*=1.12f;
            }
            txt_size = new Vector2(ww*1.2f,
                skin.GetStyle("descButton").CalcHeight(g, ww/1.12f));
            break;
          case TextLayout.DEUX_COLONNES :
            float w1 = 2*utile.width/3f - 30;
            float w2 = utile.width/3f;
            txt_size = new Vector2(w1, skin.GetStyle("descButton").CalcHeight(g, w1));
```

```
            Layout.supp_text_size = new Vector2(w2,
                skin.GetStyle("descButton").CalcHeight(
                  new GUIContent(txt2.Replace("\n", " ").Trim()), w2));
            break;
          default :
            break;
          }
        }
        Layout.textures = texs;
        epos = Layout.ecran(ecranCourant+1)(utile, txt_size);
        texs = Layout.textures;
        GUI.skin = skin;
        Rect tmp = new Rect(0,0,0,0);
        int i = 0;
        tmp.Set(epos[i].x, gui.h - epos[i].y, epos[i].width, epos[i].height);
        if (txt.Trim().Length > 0) GUI.Button(tmp, txt, skin.GetStyle("descButton"));
        int size = texs.Count;
        for (i = 1; i <= size; i++) {
          if (texs[i-1].GetType() == typeof(MovieTexture) &&
              ((MovieTexture)texs[i-1]).isReadyToPlay &&
              !((MovieTexture)texs[i-1]).isPlaying) {
            ((MovieTexture)texs[i-1]).Play();
            ((MovieTexture)texs[i-1]).Play();
          }
          tmp.Set(epos[i].x, gui.h - epos[i].y, epos[i].width, epos[i].height);
          GUI.DrawTexture(tmp, texs[i-1]);
        }
        if (i<epos.Count && txt.Trim().Length > 0) {
          tmp.Set(epos[i].x, gui.h - epos[i].y, epos[i].width, epos[i].height);
          GUI.Button(tmp, txt2, skin.GetStyle("descButton"));
        }
      }
      /* Activer les bons boutons au bon moment et traiter d'éventuels clics */
      if (ecranCourant > 0 &&
          GUI.Button(new Rect(gui.w-bouton_w-10,gui.h/2f+10, bouton_w, bouton_h), precedent_img)) {
        do_precedent();
      }
      if (ecranCourant < 3) {
        if(GUI.Button(new Rect(gui.w-bouton_w-10,gui.h/2f-bouton_h-10, bouton_w, bouton_h),
              suivant_img)) do_suivant();
        if(GUI.Button(new Rect(gui.w-bouton_w-10,gui.h-bouton_h-10, bouton_w, bouton_h),
              commentaires_img)) do_commentaires();
      }
      if(GUI.Button(new Rect(gui.w-bouton_w-10,10, bouton_w, bouton_h), chapelle_img)) do_retour();

    }
    else { // en 3D, on a juste à s'occuper des boutons.
      /* Activer les bons boutons au bon moment */
      if (ecranCourant == 0) controles.Find(b => b.action == do_precedent).draw=false;
      else controles.Find(b => b.action == do_precedent).draw=true;
      if (ecranCourant == 3) {
        controles.Find(b => b.action == do_suivant).draw=false;
        controles.Find(b => b.action == do_commentaires).draw=false;
      } else {
        controles.Find(b => b.action == do_suivant).draw=true;
        controles.Find(b => b.action == do_commentaires).draw=true;
      }
      /* et tenir compte des actions de la souris  : survol et clic */
      foreach (Bouton b in controles) b.Highlight();
      if (Input.GetButton("Fire1") && Time.time > prochainEcran) {
        prochainEcran = Time.time + dureeEcran;
        foreach (Bouton b in controles) {
          b.Click();
        }
      }
    }
  }
}

/* Quand le pointeur passe au dessus de l'objet, on affiche des indices
 * d'interactivité, en fonction du mode d'interface choisi. */
public override void PointerIn() {
  bool assez_proche = distanceObjet < distanceMax;
  if (idle && assez_proche) {
    fullname = GameObject.Find("gestion_objets").GetComponent<NomsObjets>().fullname(name);
    foreach (Light lumiere in lumieres) lumiere.intensity = intensiteMax;
```

```
      if (interface2D) player.GetComponent<CameraGUI>().msg = fullname;
      else {
        gui.cam.transform.Find("tooltip_back").renderer.enabled = true;
        tooltip.characterSize = .2f* (.15f/5f);
        gui.affiche_texte(tooltip, fullname, new Vector3(.55f*gui.w, .45f*gui.h,
              .2f), .35f*gui.w, true);
      }
    }
  }

  /* Quand le pointeur quitte l'objet, on efface tous les indices d'interactivité */
  public override void PointerOut() {
    if (idle) {
      if (interface2D) player.GetComponent<CameraGUI>().msg = "";
      tooltip.renderer.enabled = false;
      gui.cam.transform.Find("tooltip_back").renderer.enabled = false;
      fullname = "";
      foreach (Light lumiere in lumieres) lumiere.intensity = 0f;
    }
  }
}
```

fichier D.26 : unity/GyroCam.js

```
/* iPhone gyroscope-controlled camera demo v0.3 8/8/11
 * Perry Hoberman <hoberman@bway.net>
 * Directions : Attach this script to main camera.
 * Note : Unity Remote does not currently support gyroscope. */

private var gyroBool : boolean;
private var gyro : Gyroscope;
private var rotFix : Quaternion;

function Start() {
  // check if this transform has a parent
  var originalParent = transform.parent;
  var camParent = new GameObject("camParent"); // make a new parent
  // move the new parent to this transform position
  camParent.transform.position = transform.position;
  // make this transform a child of the new parent
  transform.parent = camParent.transform;
  // make the new parent a child of the original parent
  camParent.transform.parent = originalParent;
  gyroBool = SystemInfo.supportsGyroscope;
  if (gyroBool) {
    gyro = Input.gyro;
    gyro.enabled = true;
    if (Screen.orientation == ScreenOrientation.LandscapeLeft) {
      camParent.transform.eulerAngles = Vector3(90, 90, 0);
    }
    else if (Screen.orientation == ScreenOrientation.Portrait) {
      camParent.transform.eulerAngles = Vector3(90, 180, 0);
    }
    if (Screen.orientation == ScreenOrientation.LandscapeLeft) {
      rotFix = Quaternion(0, 0, 0.7071, 0.7071);
    }
    else if (Screen.orientation == ScreenOrientation.Portrait) {
      rotFix = Quaternion(0, 0, 1, 0);
    }
  }
  else {
    print("NO GYRO");
  }
}

function Update() {
  if (gyroBool) {
    var camRot : Quaternion = gyro.attitude * rotFix;
    transform.localRotation = camRot;
  }
}
```

fichier D.27 : unity/mycon.py

```
#! /usr/bin/python
```

```
# vim : set fileencoding=utf-8
import mysql.connector as ms, sys, re, urllib, os, os.path

# Les valeurs à configurer débutent ici
SERVER_URL = "http ://localhost/URS/"
UNITY_DIR = "/Users/admin/URSULINE_2012/Documents/LamicDome-SAT"
cs = ms.connect(host='localhost', user='root', password='root', database='unity')
categories = ["videos", "sons"] #éventuellement images, textes

# Fin de la configuration, on peut laisser parler la poudre ;)

if (SERVER_URL[-1] != '/') :
    SERVER_URL += '/'

UNITY_DIR = os.path.normcase(os.path.normpath(UNITY_DIR))
SAVE_DIR = os.path.join(UNITY_DIR, "Assets", "Resources")

if len(sys.argv) < 1 :
    print "appeler le script avec le nom interne d'un objet"
    sys.exit()

name = sys.argv[1].strip()
# avec un nom de 4 à 20 lettres minuscules, on se prémunit des injections SQL
forbiden = re.compile(r'[^a-z_]')
if not (4 <= len(name) <= 20) or forbiden.search(name) != None :
    print "{0} n'est pas un nom valide".format(name)
    sys.exit()

# donc on peut faire des requêtes sans échappement
cur = cs.cursor()
# en l'occurrence, on récupère l'id de l'objet
cur.execute("SELECT id FROM objets WHERE name='{0}';".format(name))
oid = int(cur.fetchone()[0])
# on crée éventuellement le répertoire dans lequel on va télécharger
SAVE_DIR = os.path.join(SAVE_DIR, name)
if not os.path.isdir(SAVE_DIR) :
    os.mkdir(SAVE_DIR)

# puis pour chaque type de contenus choisi, on récupère la liste des fichiers
# correspondant et on les télécharge au bon endroit.
for i in categories :
    cur.execute("SELECT b.path FROM blocs b, {1} d WHERE b.objet={0} AND"+
                " d.bloc=b.id;".format(oid, i))
    for row in cur.fetchmany(10) :
        filename = row[0].split('/')[-1]
        print "{0} -> {1}".format(SERVER_URL+row[0], os.path.join(SAVE_DIR, filename))
        urllib.urlretrieve(SERVER_URL+row[0], os.path.join(SAVE_DIR, filename))

cur.close()
cs.close()
```

fichier D.28 : CMS/form.php

```
<?php
/* Un formulaire en deux parties utilisé à la création d'un objet. On demande
* d'abord les informations relatives à l'objet, on appele upload.php qui crée
* l'objet (et donc son id) puis on revient ici et on affiche la partie relative
* au chargement des contenus.
 */
session_start();
if (!isset($_SESSION["quoi"])) { $_SESSION["quoi"] = "objet"; }
if (!isset($_SESSION["fichiers"])) { $_SESSION["fichiers"] = array(); }
try {
  $bdd = new PDO($_SESSION["connect_str"], $_SESSION["user"], $_SESSION["pwd"],
    $_SESSION["pdo_options"]);
}
catch (Exception $e) {
  die('Erreur : ' . $e->getMessage());
}
?>
<!DOCTYPE HTML>
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <meta name="author" content="Géraud Le Falher" />
    <link href="style.css" rel="stylesheet" media="screen" type="text/css" />
```

```
    <script src="sha1-min.js" type="text/javascript" charset="utf-8"></script>
    <script src="validation.js" type="text/javascript" charset="utf-8"></script>
    <title>Form</title>
  </head>
  <body>
    <h1 id="titre_objet"></h1>
<?php include "utils.php";
if ($_SESSION["quoi"] == "objet") {
  echo make_object_form($bdd, $_SESSION["objet_courant_id"], "Créer");
}
if ($_SESSION["quoi"] == "doc") { include("charger.html"); } ?>
    <script src="file_upload.js" type="text/javascript" charset="utf-8"></script>
  </body>
</html>
```

fichier D.29 : CMS/gestion.php

```
<?php
/* Page d'acceuil, qui propose de créer un nouvel objet ou de modifier les
 * existants. */
/* On démarre la session et on récupère les informations de connection */
session_start();
include "utils.php";
include "sql_connect.php";
if (!isset($_SESSION["fichiers"])) { $_SESSION["fichiers"] = array(); }
if (!isset($_SESSION["server"])) { $_SESSION["server"] = $server; }
if (!isset($_SESSION["pdo_options"])) { $_SESSION["pdo_options"] =
  $pdo_options;}
if (!isset($_SESSION["user"])) { $_SESSION["user"] = $user; }
if (!isset($_SESSION["pwd"])) { $_SESSION["pwd"] = $pwd; }
if (!isset($_SESSION["database"])) { $_SESSION["database"] = $database; }
if (!isset($_SESSION["connect_str"])) { $_SESSION["connect_str"] = $connect_str;
}
/* On va placer les fichiers chargés dans basepath, qui doit être créé s'il
 * n'existe pas. */
if (!isset($_SESSION["basepath"])) { $_SESSION["basepath"] = "uploads/";
  if (!is_dir($_SESSION["basepath"])) {
    mkdir($_SESSION["basepath"]);
  }
}
/* Quand on arrive sur cette page, on revient au début, donc plus d'objet
 * courant */
$_SESSION["objet_courant_id"] = 0;
$_SESSION["objet_courant_name"] = "__none__";
/* On essaye de se connecter à la base de données */
try {
  $bdd = new PDO($_SESSION["connect_str"], $_SESSION["user"],
    $_SESSION["pwd"], $_SESSION["pdo_options"]);
}
catch (Exception $e) {
  die('Erreur : ' . $e->getMessage());
}

/* On récupère différentes énumération dans la BDD, qu'on écrit dans un fichier
 * pour que le code en javascript puisse y accéder.
 */

if (!isset($_SESSION["niveau_vals"])) {
  $_SESSION["niveau_vals"] = get_enum_values($bdd, "blocs", "niveau");
  file_put_contents('niveau_vals', $_SESSION["niveau_vals"]);
}
if (!isset($_SESSION["cat_vals"])) {
  $_SESSION["cat_vals"] = get_enum_values($bdd, "objets", "cat");
  file_put_contents('cat_vals', $_SESSION["cat_vals"]);
}
if (!isset($_SESSION["pos_vals"])) {
  $_SESSION["pos_vals"] = get_enum_values($bdd, "blocs", "pos");
  file_put_contents('pos_vals', $_SESSION["pos_vals"]);
}
if (!isset($_SESSION["type_vals"])) {
  $_SESSION["type_vals"] = get_enum_values($bdd, "images", "type");
  file_put_contents('type_vals', $_SESSION["type_vals"]);
}
?>
<!DOCTYPE HTML>
<head>
```

```
  <meta http-equiv="content-type" content="text/html;charset=utf-8" />
  <meta name="author" content="Géraud Le Falher" />
  <meta name="description" content="Gestion d'objets interactifs" />
  <link href="style.css" rel="stylesheet" media="screen" type="text/css" />
  <title>Objets interactifs   chapelle des Ursulines</title>
  <script type="text/javascript" charset="utf-8">
    function $id(id) { return document.getElementById(id); };
  </script>
</head>
<body>
  <h1>Gestion des objets interactifs de la chapelle des Ursulines</h1>
  <div class="demande">
    <a href="form.php" title="Créer un nouvel objet">Créer un nouvel objet</a>
  </div>
  <div class="demande">
<form action="modif.php" method="POST" enctype="multipart/form-data">
    <p>
    <label for="cat">Ou sélectionner  :</label>
    <select name="objet" id="objet">
<?php
/* Afficher la liste des objets existant */
  $reponse = $bdd->query('SELECT `id`, `name` FROM `objets` ORDER BY `name`;');
while ($donnees = $reponse->fetch()) { ?>
      <option value="<?php echo $donnees['id']; ?>"><?php echo $donnees['name']; ?></option>
<?php }
$reponse->closeCursor();
 ?>
    </select>
    <input type="hidden" name="action" value="" id="action" />
    <button onclick="$id('action').value='modifier'" type="submit">et modifiez-le</button>
    <button onclick="$id('action').value='supprimer'" type="submit">ou supprimez-le.</button>
  </div>
</body>
</html>
```

fichier D.30 : CMS/get.php

```
<?php
/* D'après http ://www.unifycommunity.com/wiki/index.php?title=WWWForm_PHP_Example
 * un script qui se connecte à la BDD pour renvoyer les commentaires formatés associés à
 * l'objet fournit par la méthode POST.
 */

include "sql_connect.php";
$table = "commentaires";
$objet = $_POST['objet'];
$nb_com = 10;

try {
  $bdd = new PDO($connect_str, $user, $pwd, $pdo_options);
  $bdd->query("SET lc_time_names = 'fr_CA';");
  /* On récupère l'id associé à 'name' (qui est un champ unique) */
  $req = $bdd->prepare("SELECT id FROM `objets` WHERE `name`=?;");
  $data = $req->execute(array($objet));
  $data = $req->fetch();
  /* On récupère les nb_com derniers commentaires avec l'heure bien affiché */
  $select = $bdd->prepare("SELECT DATE_FORMAT(`quand`, ' a écrit le %W %e %M %Y à
    %lh%i %p') AS `date`, pseudo, comment FROM $table WHERE objet=? ORDER BY
    id DESC LIMIT ".$nb_com);
  $select->execute(array($data["id"]));
  /* On les affiche les uns à la suite des autres */
  while ($donnees = $select->fetch()) {
    echo $donnees['pseudo'].$donnees['date']."  : ".$donnees['comment']."\n\n";
  }
  $select->closeCursor();
  $bdd = null;
}
catch (Exception $e) {
  die('Erreur  : ' . $e->getMessage());
}
?>
```

fichier D.31 : CMS/getxml.php

```
<?php
```

```
/* Appelé sans paramètre, renvoie la liste de tous les objets en associant leur nom court
* et leur nom long. Appelé avec le nom d'un objet, renvoie un fichier xml décrivant tous
* les contenus associés à l'objet.
 */
include "sql_connect.php";
try {
  $bdd = new PDO($connect_str, $user, $pwd, $pdo_options);
  $name = (isset($_POST["name"]) ? trim($_POST["name"]) : (isset($_GET["name"]) ? trim($_GET["name"]) : false ));
  include "utils.php";
  if ($name) {
    $req = $bdd->prepare("SELECT `id` from `objets` WHERE `name`=?;");
    $req->execute(array($name));
    $data = $req->fetch();
    contents_to_xml(get_object_contents($bdd, $data["id"]));
    $req->closeCursor();
  }
  else {
    list_all_objects($bdd);
  }
  $bdd = null;
}
catch (Exception $e) {
  die('Erreur : ' . $e->getMessage());
}
?>
```

fichier D.32 : CMS/modif.php

```
<?php
/* Permet de modifier le contenu d'un objet mais aussi de le supprimer entièrement. */
session_start();
include "utils.php";
try {
  $bdd = new PDO($_SESSION["connect_str"], $_SESSION["user"],
    $_SESSION["pwd"], $_SESSION["pdo_options"]);
}
catch (Exception $e) {
  die('Erreur : ' . $e->getMessage());
}
$_SESSION["objet_courant_id"] = trim($_POST["objet"]);
$_SESSION["objet_courant_name"] = get_object_info($bdd,
      $_SESSION["objet_courant_id"], "name");
?>
<!DOCTYPE HTML>
<head>
  <meta http-equiv="content-type" content="text/html; charset=utf-8" />
  <meta name="author" content="Géraud Le Falher" />
  <meta name="description" content="Gestion des objets" />
  <link href="style.css" rel="stylesheet" media="screen" type="text/css" />
  <script src="sha1-min.js" type="text/javascript" charset="utf-8"></script>
  <script src="validation.js" type="text/javascript" charset="utf-8"></script>
  <title>Modification d'un objet</title>
  </head>
  <body>
<?php
$id = $_SESSION["objet_courant_id"];
switch ($_POST["action"]) {
case 'modifier' :
  /* On affiche le formulaire permettant de modifier les contenus existant
  * ainsi que d'en ajouter de nouveau, grâce notamment au script
  * file_upload.js
   */
  echo '<h1 id="titre_objet"></h1>';
  echo make_object_form($bdd, $id, "Modifier");
  contents_to_form(get_object_contents($bdd, $id));
  echo '<script src="file_upload.js" type="text/javascript" charset="utf-8"></script>';
  break;
case 'supprimer' :
  /* On supprime l'objet (ainsi que le contenu par l'intermédaire des directives ON
   * CASCADE de SQL) et son répertoire.
   */
  $name = get_object_info($bdd, $id, "fullname");
  $req = $bdd->prepare("DELETE FROM `objets` WHERE `id`=?;");
  $req->execute(array($id));
  $updir = $_SESSION["basepath"].$_SESSION["objet_courant_name"].'/';
  rmdir($updir);
```

```
  header("refresh :5;url=gestion.php" );
  echo '<p>« '. $name .  ' » a été supprimé, ainsi que tous les contenus associés.</p>
     <p><a href="gestion.php">OK</a></p>';
  break;
}
?>
</body>
</html>
```

fichier D.33 : CMS/send.php

```
<?php
/* D'après http ://www.unifycommunity.com/wiki/index.php?title=WWWForm_PHP_Example
 * Se connecte à la BDD pour insérer le commentaire 'comment' de l'utilisateur
 * 'pseudo' à propos de 'objet'.
 */
include "sql_connect.php";
$table = "commentaires";

$pseudo = $_POST['pseudo'];
$comment = $_POST['texte'];
$objet = $_POST['objet'];
try {
  $bdd = new PDO($connect_str, $user, $pwd, $pdo_options);
  $req = $bdd->prepare("SELECT id FROM `objets` WHERE `name`=?;");
  $data = $req->execute(array($objet));
  $data = $req->fetch();
  $insert = $bdd->prepare("INSERT INTO $table (`quand`, `objet`, `pseudo`,
    `comment`) VALUES (NOW(), ?, ?, ?)");
  $insert->execute(array($data["id"], $pseudo, $comment));
  echo "TODO : afficher un message de remerciement/confirmation ou rien du tout";
  $bdd = null;
}
catch (Exception $e) {
  die('Erreur  : ' . $e->getMessage());
}
?>
```

fichier D.34 : CMS/update.php

```
<?php
/* Géraud Le Falher <daureg@gmail.com>
 * 2012-07-16
 * Charge les fichiers sur le serveur et remplit la base de données en mettant à
 * jour les contenus déjà existant et en insérant les nouveaux.
 */
session_start();
try {
  $bdd = new PDO($_SESSION["connect_str"], $_SESSION["user"],
    $_SESSION["pwd"], $_SESSION["pdo_options"]);
}
catch (Exception $e) {
  die('Erreur  : ' . $e->getMessage());
}
include "utils.php";
?>
<!DOCTYPE HTML>
<head>
  <meta http-equiv="content-type" content="text/html; charset=utf-8" />
  <meta name="author" content="Géraud Le Falher" />
  <meta name="description" content="Gestion des objets" />
  <link href="style.css" rel="stylesheet" media="screen" type="text/css" />
  <title>Modification d'un objet</title>
  </head>
  <body>
<?php
$new_file = (isset($_SERVER['HTTP_X_FILENAME']) ? $_SERVER['HTTP_X_FILENAME'] : false);
/* Si c'est le premier contenu de l'objet, il faut créer le répertoire
 * correspondant. */
$updir = $_SESSION["basepath"].$_SESSION["objet_courant_name"].'/';
if (!is_dir($updir)) {
  mkdir($updir);
}
if ($new_file) { //appel par AJAX
  file_put_contents($updir . $new_file, file_get_contents('php ://input'));
```

```
  $req = $bdd->prepare("SELECT id FROM `blocs` WHERE path=?;");
  $req->execute(array($updir.$new_file));
  $data = $req->fetch();
  if ($data){ //on dépose en AJAX une nouvelle version d'un fichier existant
    $_SESSION['fichiers'][$new_file] = "already_inserted";
  }
  else { //on dépose en AJAX un nouveau fichier
    $_SESSION['fichiers'][$new_file] = "just_uploaded";
  }
}
else { //submit normal
  // insérer les fichiers ajoutés mais pas dropés
  $files = $_FILES['fileselect'];
  foreach ($files['error'] as $id => $err) {
    if ($err == UPLOAD_ERR_OK ) {
      $fn = $files['name'][$id];
      if (!isset($_SESSION['fichiers'][$fn])) {
        move_uploaded_file($files['tmp_name'][$id], $updir . $fn);
        $_SESSION['fichiers'][$fn] = "just_uploaded";
        inserer_fichier(get_file_infos($fn), $bdd);
        $_SESSION['fichiers'][$fn] = "newly_inserted";
      }
    }
  }
  /* Insérer ensuite les nouveaux fichiers dropés (qui n'ont pas pu l'être avant
   * puisqu'en AJAX, on a pas encore POST)
   */
  foreach ($_SESSION['fichiers'] as $fichier => $statut) {
    if ($statut == "just_uploaded") {
      inserer_fichier(get_file_infos($fichier), $bdd);
      $_SESSION['fichiers'][$fichier] = "newly_inserted";
    }
  }
  /* Dans le doute, mettre tous les autres à jour (ceux qui n'ont pas subis
   * d'upload, ou ceux qui ont été uploadés alors qu'il existait déjà.
   */
  update_objet($bdd, $_SESSION["objet_courant_id"]);
  foreach ($_POST as $key => $value) {
    if (substr($key, 0, 6) == "sqlid_") {
      $hash = substr(trim($key), 6);
      $id = $value;
      $blocid = $_POST["blocid_".$hash];
      if (isset($_POST["del_".$hash])) { //supprimer le bloc
        $req = $bdd->prepare("DELETE FROM `blocs` WHERE `id`=?;");
        $req->execute(array($blocid));
        unlink($_POST["path_".$hash]);
      }
      else { //le mettre à jour
        $tmp = explode('/', trim($_POST["path_".$hash]));
        $filename = $tmp[count($tmp)-1];
        update_fichier(get_file_infos($filename), $bdd, $id, $blocid);
        $_SESSION['fichiers'][$filename] = "freshly_updated";
      }
    }
  }
  header("refresh :5;url=gestion.php" );
  echo '<p>Vos modifications ont bien été prises en compte.<p>
    <p><a href="gestion.php">Retour à l\'accueil</a></p>';
}
?>
</body>
</html>
```

fichier D.35 : CMS/upload.php

```
<?php
/* Géraud Le Falher <daureg@gmail.com>
 * 2012-07-11
 * Charge les fichiers sur le serveur et remplit la base de données.
 * C'était initialement :
 * Server-side PHP file upload code for HTML5 File Drag & Drop demonstration
 * Featured on SitePoint.com
 * Developed by Craig Buckler (@craigbuckler) of OptimalWorks.net
 */
session_start();
```

```
try {
  $bdd = new PDO($_SESSION["connect_str"], $_SESSION["user"], $_SESSION["pwd"],
    $_SESSION["pdo_options"]);
}
catch (Exception $e) {
  die('Erreur : ' . $e->getMessage());
}

include "utils.php";
$quoi = $_SESSION['quoi'];
$fichiers = $_SESSION['fichiers'];
switch ($quoi) {
case 'objet' :
  inserer_objet($bdd);
  break;
case 'doc' :
  $fn = (isset($_SERVER['HTTP_X_FILENAME']) ? $_SERVER['HTTP_X_FILENAME'] : false);
  $updir = $_SESSION["basepath"].$_SESSION["objet_courant_name"].'/';
  if ($fn) { // AJAX call
    file_put_contents($updir . $fn, file_get_contents('php://input'));
    $_SESSION['fichiers'][$fn] = "just_uploaded";
    echo "$fn uploaded donc f[] = " . $_SESSION['fichiers'][$fn];
    exit();
  }
  else { // form submit
    // commencer par uploader ceux qui n'ont pas été droppés
    $files = $_FILES['fileselect'];
    foreach ($files['error'] as $id => $err) {
      if ($err == UPLOAD_ERR_OK ) {
        $fn = $files['name'][$id];
        if (!isset($_SESSION['fichiers'][$fn])) {
          move_uploaded_file($files['tmp_name'][$id], $updir . $fn);
          $_SESSION['fichiers'][$fn] = "just_uploaded";
        }
      }
    }
    // Insérer ensuite tous les fichiers uploadés
    foreach ($_SESSION['fichiers'] as $fichier => $statut) {
      if ($statut == "just_uploaded") {
        inserer_fichier(get_file_infos($fichier), $bdd);
        $_SESSION['fichiers'][$fichier] = "newly_inserted";
      }
    }
    $_SESSION['quoi'] = "objet";
    redirect("gestion.php");
  }
  break;
default :
  break;
}
?>
```

fichier D.36 : CMS/utils.php

```
<?php
/* Contient la plupart des fonctions dont on besoin les pages php */
/* Renvoie en xml la liste de tous les objets contenus dans 'bdd' accompagnés de
 * leur nom complet. */
function list_all_objects($bdd) {
  $list = $bdd->query("SELECT `name`, `fullname` FROM `objets` ORDER BY `name`;");
  require_once "XmlEasyWriter.class.php";
  $xml = new XmlEasyWriter("objets");
  while ($data = $list->fetch()) {
    $xml->setFullElement("objet", "",
      array("name" => $data["name"], "fullname" => $data["fullname"]));
  }
  $xml->endElement();
  $xml->output();
}
/* Renvoie la valeur de la colonne 'key' pour l'objet identifié par 'id' */
function get_object_info($bdd, $id, $key) {
  $req = $bdd->prepare("SELECT `".$key."` FROM `objets` WHERE `id`=?;");
  $req->execute(array($id));
  $info = $req->fetch();
  return $info[0];
}
```

```
/* Renvoie dans un tableau les informations associés à l'objet 'id' */
function get_object_contents($bdd, $id) {
  $contents = array();
  $type = array('images','sons','textes','videos');
  $req = $bdd->prepare("SELECT * FROM `objets` WHERE id=?;");
  $req->execute(array($id));
  /* On récupère d'abord les infos sur l'objet */
  $i = 0;
  $contents[$i] = $req->fetch();
  $i++;
  foreach ($type as $c) { /* pour chaque type de contenus,
    on récupère tous ceux associés à l'objet en respectannt leur ordre. */
    $req = $bdd->prepare("SELECT * FROM `".$c."` info, `blocs` bloc WHERE
    bloc.objet=? AND info.bloc = bloc.id ORDER BY bloc.ordre;");
  $req->execute(array($id));
  while ($info = $req->fetch()) {
    /* Comme la colonne id porte le même nom dans la table info et dans bloc,
     * on utilise leur index numérique pour replacer ces deux informations
     * différentes au bon endroit.
     */
    $info["bloc_id"] = $info["id"]; // l'id du bloc
    $info["id"] = $info[0]; // l'id du contenu dans sa table
    $filename = explode('/', $info["path"]);
    $info["hash"] = substr(sha1(end($filename)), 0, 8);
    $split = explode('/', $info["mime"]);
    $info["kind"] = $split[0]; // le type de contenu
    $contents[$i] = $info;
    $i++;
  }
  }
  return $contents;
}
/* Crée un champ de sélection multiple HTML identifié par l'utilisateur par
  * 'label' et en interne par 'name' dont les différents sont dans 'value', avec
  * éventuellement 'current' sélectionné. */
function create_select($xml, $name, $label, $values, $current="") {
  $xml->setFullElement("label", $label." :", array("for" => $name));
  $xml->setFullElement("select", "", array("id" => $name,"name" => $name), false);
  $values = explode(';', $values);
  $selected = array('selected' => "selected");
  foreach ($values as $v) {
    $attribs = array("value"=>$v);
    if ($v == $current) { $attribs = array_merge($attribs, $selected); }
    $xml->setFullElement('option', $v, $attribs);
  }
  $xml->fullEndElement();
}
/* Crée les éléments de formulaire qui stockeront les informations communes tous
 * les contenus, dans ce cas celui décrit par 'bloc'. */
function add_bloc_form($xml, $bloc, $progress=false) {
  $hash = $bloc["hash"];
  if ($progress) {
    $xml->setFullElement("progress", "", array("id" => "progress_".$hash,
      "max" => 100));
  }
  $xml->writeElement("br");
  $xml->setFullElement("label", "Description :", array("for" => "desc_".$hash));
  $xml->setFullElement("textarea", $bloc["desc"], array("name" => "desc_".$hash,
    "id" => "desc_".$hash, "placeholder" => "Pas plus de 250 caractères", "rows"
    => 5, "cols" => 48, "maxlength" => 250));
  create_select($xml, "niveau_".$hash, "Niveau", $_SESSION["niveau_vals"],
    $bloc["niveau"]);
  create_select($xml, "pos_".$hash, "Position", $_SESSION["pos_vals"],
    $bloc["pos"]);
  if ($bloc["kind"] == "image") {
    create_select($xml, "type_".$hash, "Type", $_SESSION["type_vals"],
      $bloc["type"]);
  }
  $xml->setFullElement("label", "Ordre :", array("for" => "ordre_".$hash));
  $xml->setFullElement("input", "", array("id" => "ordre_".$hash, "name" =>
    "ordre_".$hash, "type" => "number", "min" => 0, "max" => 100, "value" =>
    $bloc["ordre"]));
}
/* À partir de la description d'un objet 'c', renvoie un fragment de HTML qui
  * consiste en un formulaire affichant tous les contenus de l'objet et
  * proposant de les modifier comme lors de sa création. */
```

```
function contents_to_form($c) {
  require_once "XmlEasyWriter.class.php";
  $checked = array("checked" => "checked");
  $xml = new XmlEasyWriter("fieldset");
  $xml->setElement("legend", "Modifications des documents");
  $xml->startElement("div");
  $xml->setFullElement("label", "Sélectionnez éventuellement des documents
    supplémentaires :", array("for" => "fileselect"));
  $xml->setFullElement("input", "", array("type" => "file", "id" => "fileselect",
      "name" => "fileselect[]", "multiple" => "multiple"));
  $xml->setFullElement("div", "ou déposez-les ici", array("id" => "filedrag"));
  $size = count($c);
  $ordre_max = 0;
  $mime = "";
  $first_time = true;
  $mime_used = array("image" => false, "text" => false, "audio" => false, "video" => false);
  for ($i = 1; $i < $size; $i++) { // pour chaque contenu
    $hash = $c[$i]["hash"];
    if ($c[$i]["ordre"] > $ordre_max) {
      $ordre_max = $c[$i]["ordre"];
    }
    $tmp =$c[$i]["kind"];
    if ($mime != $tmp) { /* on rencontre ce type de fichiers pour la première
    fois, on crée un div qui va tous les contenir. */
      if (!$first_time) {
        $xml->fullEndElement();
      }
      $first_time = false;
      $mime = $tmp;
      $xml->setFullElement("div", "", array("id" => $mime."s"), false);
      $mime_used[$mime] = true;
    }
    $xml->setFullElement("div", "", array("class" => "content", "id" => $hash),
    false);
    /* On ajoute les champs cachés qui contienne les métadonnées.
     * D'abord ceux qui sont communs. */
    $xml->setFullElement("input", "", array("type" => "hidden", "name" =>
      "mime_".$hash, "id" => "mime_".$hash, "value" => $c[$i]["mime"]));
    $xml->setFullElement("input", "", array("type" => "hidden", "name" =>
      "len_".$hash, "id" => "len_".$hash, "value" => $c[$i]["len"]));
    $xml->setFullElement("input", "", array("type" => "hidden", "name" =>
      "path_".$hash, "id" => "path_".$hash, "value" => $c[$i]["path"]));
    $xml->setFullElement("input", "", array("type" => "hidden", "name" =>
      "blocid_".$hash, "id" => "blocid_".$hash, "value" => $c[$i]["bloc_id"]));
    $xml->setFullElement("input", "", array("type" => "hidden", "name" =>
      "sqlid_".$hash, "id" => "sqlid_".$hash, "value" => $c[$i]["id"]));
    switch ($mime) { // puis ceux qui dépendent du type.
    case 'image' :
      $xml->setFullElement("input", "", array("type" => "hidden", "name" =>
        "h_".$hash, "id" => "h_".$hash, "value" => $c[$i]["h"]));
      $xml->setFullElement("input", "", array("type" => "hidden", "name" =>
        "w_".$hash, "id" => "w_".$hash, "value" => $c[$i]["w"]));
      $xml->setFullElement("img", "", array("class" => "resize", "src" =>
        $c[$i]["path"], "id" => "img_".$hash, "alt"=>$c[$i]["desc"]));
      break;
    case 'text' :
      $xml->setFullElement("textarea", $c[$i]["content"], array("class" =>
        "resize", "name" => "content_".$hash,"id" => "content_".$hash));
      break;
    case 'audio' :
      $xml->setFullElement("audio", "", array("controls" => "controls",
        "preload" => "auto", "src" => $c[$i]["path"], "id" => "son_".$hash));
      $xml->setFullElement("label", "En boucle  :", array("for" => "boucle_".$hash));
      $tmp = array("type" => "checkbox", "name" => "boucle_".$hash, "id" =>
        "boucle_".$hash);
      if ($c[$i]["boucle"] == 1) { $tmp =array_merge($tmp, $checked); }
      $xml->setFullElement("input", "", $tmp);
      $xml->setFullElement("label", "Spatial  :", array("for" => "spatial_".$hash));
      $tmp = array("type" => "checkbox", "name" => "spatial_".$hash, "id" =>
        "spatial_".$hash);
      if ($c[$i]["spatial"] == 1) { $tmp =array_merge($tmp, $checked); }
      $xml->setFullElement("input", "", $tmp);
      break;
    case 'video' :
      $xml->setFullElement("video", "", array("controls" => "controls",
        "preload" => "auto", "src" => $c[$i]["path"], "id" => "vid_".$hash));
```

```
      break;
    default :
      break;
    }
    $xml->setFullElement("div", "", array("class" => "suppr"), false);
    $xml->setFullElement("label", "À supprimer ?", array("for" => "del_".$hash));
    $xml->setFullElement("input", "", array("type" => "checkbox", "name" =>
      "del_".$hash, "id" => "del_".$hash));
    $xml->fullEndElement();
    add_bloc_form($xml, $c[$i]);
    $xml->fullEndElement();
  }
  $xml->fullEndElement(); //le dernier mime+s
  // écrire les div des mimes non vus pour des ajouts ultérieurs
  foreach ($mime_used as $mime => $used) {
    if (!$used) {
      $xml->setFullElement("div", "", array("id" => $mime."s"));
    }
  }
  $xml->setFullElement("input", "", array("type" => "hidden", "name" => "ordre_max",
    "id" => "ordre_max", "value" => $ordre_max+1));
  $xml->setFullElement("p", "", array("id" => "submitbutton"), false);
  $xml->setFullElement("button", "Effectuer les modifications", array("type" => "submit"));
  $xml->endElement(); //fieldset
  $xml->endElement(); //form
  echo substr($xml->getDocument(), 38);
}
// Renvoie un fichier xml qui explicite l'objet et ses contenus, décrit par 'c'
function contents_to_xml($c) {
  /* La méthode fonctionne de manière semblable à contents_to_form(),
   * ça aide bien n'est pas ? */
  require_once "XmlEasyWriter.class.php";
  $xml = new XmlEasyWriter("objet", array("name" => $c[0]["name"],
    "fullname" => $c[0]["fullname"],
    "desc" => $c[0]["desc"],
    "cat" => $c[0]["cat"]));
  $size = count($c);
  $mime = "";
  $first_time = true;
  for ($i = 1; $i < $size; $i++) {
    $tmp = explode('/', $c[$i]["mime"]);
    if ($mime != $tmp[0]) {
      if (!$first_time) {
        $xml->endElement();
      }
      $first_time = false;
      $mime = $tmp[0];
      $xml->startElement($mime."s");
    }
    switch ($mime) {
    case 'image' :
      $xml->setFullElement($mime, "", array("w" => $c[$i]["w"],
        "h" => $c[$i]["h"],
        "type" => $c[$i]["type"]), false);
      break;
    case 'text' :
      $xml->setElement($mime, $c[$i]["content"], false);
      break;
    case 'audio' :
      $xml->setFullElement($mime, "", array( "boucle" => $c[$i]["boucle"],
        "spatial" => $c[$i]["spatial"]), false);
      break;
    case 'video' :
      $xml->setElement($mime, "", false);
      break;
    }
    $xml->setFullElement("bloc", $c[$i]["desc"], array(
      "path" => $c[$i]["path"],
      "len" => $c[$i]["len"],
      "niveau" => $c[$i]["niveau"],
      "pos" => $c[$i]["pos"],
      "ordre" => $c[$i]["ordre"]));
    $xml->endElement();
  }
  $xml->endElement();
  $xml->output();
```

```
}
/* Renvoie vers la page 'url' */
function redirect($url, $permanent = false) {
  if ($permanent) {
    header('HTTP/1.1 301 Moved Permanently');
  }
  header('Location : '.$url);
  exit();
}
/* Renvoie toutes les valeurs que peut prendre l'énumération 'col' dans 'table'
 * séparées par ; */
function get_enum_values($bdd, $table, $col) {
  $data = $bdd->query("SHOW COLUMNS FROM ".$table ." LIKE '".$col."';");
  $data = $data->fetch();
  $res = substr($data["Type"], 6, -1);
  return str_replace(',',';',str_replace("'","",$res));
}
/* Renvoie le formulaire permettant de modifier l'objet désigne par 'id' si
 * 'action' vaut "Modifier", ou juste de le créer sinon. */
function make_object_form($bdd, $id, $action) {
  $name = "";
  $fullname = "";
  $desc = "";
  $cat = "";
  if ($id != 0) {
    $req = $bdd->prepare('SELECT `name`, `fullname`, `desc`, `cat` FROM `objets` WHERE id=?;');
    $req->execute(array($_POST["objet"]));
    $data = $req->fetch();
    $name = $data["name"];
    $fullname = $data["fullname"];
    $desc = $data["desc"];
    $cat = $data["cat"];
  }
  $categories = explode(';', $_SESSION["cat_vals"]);
  $selected = 'selected="selected" ';
  $selection = "";
  foreach ($categories as $c) {
    $selection .= '<option '.(($c == $cat)?$selected :'').'value="'.$c.'">'.$c.'</option>';
  }
  return '<form id="upload" action="'
    .(($action != "Modifier")?'upload.php' :'update.php').'"
    method="POST" enctype="multipart/form-data"
    onsubmit="verif_objet(this);return false;">
    <fieldset>
    <legend>Informations sur l\'objet</legend>
    <p><label for="fullname">Nom complet  :</label>
    <input type="text" onblur="$id(\'titre_objet\').textContent=this.value;"
    name="fullname" id="fullname" size="100" maxlength="100" value="'
    .$fullname.'" required autofocus /></p>
  <p><label for="name">Nom interne  :</label>
    <input type="text" onblur="verif(this);" name="name" id="name" size="45"
    maxlength="20" placeholder="20 lettres minuscules ou underscore maximum"
    required ' .  (($name != "")?'value="'.$name .'"' :'') . '/></p>
  <p><textarea name="desc" id="desc" placeholder="Pas plus de 500 caractères"
  rows="5" cols="48" maxlength="500" />' .
  (($desc != "")?$desc :'') . '</textarea></p>
  <p><label for="cat">Catégorie  :</label>
  <select name="cat" id="cat">'.
  $selection . ' </select></p>'
  .(($action != "Modifier")?
  '<p><button id="objet_submit" type="submit">'.$action.' l\'objet...</button></p>
  </fieldset></form>'
   :'</fieldset>');
}
/* renvoie un tableau associatif contenant toute les informations incluses dans
 * le formulaire à propos du fichier 'name'.
 */
function get_file_infos($name) {
  /* le hash du nom permet d'accéder aux champs du formulaire relatif au
   * fichier. */
  $hash = substr(sha1($name), 0, 8);
  $updir = $_SESSION["basepath"].$_SESSION["objet_courant_name"].'/';
  /* D'abord les infos communes */
  $res = array(
    "objet" => $_SESSION["objet_courant_id"],
    "desc" => trim($_POST["desc_".$hash]),
```

```
    "mime" => trim($_POST["mime_".$hash]),
    "path" => $updir.$name,
    "len" => trim($_POST["len_".$hash]),
    "niveau" => trim($_POST["niveau_".$hash]),
    "pos" => trim($_POST["pos_".$hash]),
    "ordre" => trim($_POST["ordre_".$hash]),
  );
  // Puis en fonction du type de fichiers, on récupère les champs particuliers.
  $split = explode('/', $res["mime"]);
  $mime = $split[0];
  $res["kind"] = $mime;
  if ($mime == "image") {
    $res["w"] = trim($_POST["w_".$hash]);
    $res["h"] = trim($_POST["h_".$hash]);
    $res["type"] = trim($_POST["type_".$hash]);
  }
  if ($mime == "audio") { //ces deux variables sont booléennes.
    $res["spatial"] = isset($_POST["spatial_".$hash]);
    $res["boucle"] = isset($_POST["boucle_".$hash]);
  }
  if ($mime == "video") {}
  if ($mime == "text") {
    $res["kind"] = "text";
    $res["content"] = trim($_POST["content_".$hash]);
  }
  return $res;
}
/* Insère dans 'bdd' le fichier décrit par 'info' */
function inserer_fichier($info, $bdd) {
  /* Crée d'abord le bloc correspondant */
  $req = $bdd->prepare("INSERT INTO blocs (`objet`, `path`, `mime`, `desc`,
  `len`, `niveau`, `pos`, `ordre`) VALUES (?,?,?,?,?,?,?,?);");
  $req->execute(array($info["objet"], $info["path"], $info["mime"], $info["desc"],
    $info["len"], $info["niveau"], $info["pos"], $info["ordre"]));
  /* Récupère son id et en fonction du type, insère le reste des informations
   * dans la table approprié */
  $data = $bdd->query("SELECT LAST_INSERT_ID();")->fetch();
  switch ($info["kind"]) {
  case 'image' :
    $req = $bdd->prepare("INSERT INTO images (`bloc`,`w`,`h`,`type`) VALUES (?,?,?,?)");
    $req->execute(array($data[0], $info["w"], $info["h"], $info["type"]));
    break;
  case 'text' :
    $req = $bdd->prepare("INSERT INTO textes (`bloc`,`content`) VALUES (?,?)");
    $req->execute(array($data[0], $info["content"]));
    break;
  case 'audio' :
    $req = $bdd->prepare("INSERT INTO sons (`bloc`,`spatial`,`boucle`) VALUES (?,?,?)");
    $req->execute(array($data[0], $info["spatial"], $info["boucle"]));
    break;
  case 'video' :
    $req = $bdd->prepare("INSERT INTO videos (`bloc`) VALUES (?)");
    $req->execute(array($data[0]));
    break;
  }
}
/* Met à jour le fichier 'id' donc le bloc est 'blocid'. */
function update_fichier($info, $bdd, $id, $blocid) {
  $req = $bdd->prepare("UPDATE blocs SET `desc`=?, `niveau`=?, `pos`=?,
    `ordre`=? WHERE `id`=?;");
  $req->execute(array($info["desc"], $info["niveau"], $info["pos"],
    $info["ordre"], $blocid));
  switch ($info["kind"]) {
  case 'image' :
    $req = $bdd->prepare("UPDATE `images` SET `type`=?, `w`=?, `h`=? WHERE `id`=?;");
    $req->execute(array($info["type"], $info["w"], $info["h"], $id));
    break;
  case 'audio' :
    $req = $bdd->prepare("UPDATE `sons` SET `spatial`=?, `boucle`=? WHERE `id`=?;");
    $req->execute(array($info["spatial"], $info["boucle"], $id));
    break;
  case 'text' :
    $req = $bdd->prepare("UPDATE `textes` SET `content`=? WHERE `id`=?;");
    $req->execute(array($info["content"], $id));
    break;
    /* Dans l'état actuel des choses, il n'y a rien à modifier spécifiquement
```

```
     * pour celui là.
  case 'video' :
    $req = $bdd->prepare("UPDATE videos SET bloc=? WHERE id=?;");
    $req->execute(array($blocid, $id));
    break;
     */
  }
}
/* Insère l'objet défini par POST dans 'bdd' et met à jour les variables de
 * session avant de rediriger vers form.php pour insérer les contenus.
 */
function inserer_objet($bdd) {
  $name = trim($_POST["name"]);
  $fullname = trim($_POST["fullname"]);
  $desc = trim($_POST["desc"]);
  $cat = trim($_POST["cat"]);
  // Créer le répertoire où seront placés les fichiers relatifs à l'objet
  mkdir($_SESSION["basepath"].$name);
  try {
    $req = $bdd->prepare("INSERT INTO `objets` (`name`, `fullname`, `desc`,
      `cat`) VALUES (?,?,?,?);");
    $req->execute(array($name, $fullname, $desc, $cat));
    $data = $bdd->query("SELECT LAST_INSERT_ID();")->fetch();
    $_SESSION["objet_courant_id"] = $data[0];
    $_SESSION["objet_courant_name"] = $name;
    $_SESSION["quoi"] = "doc";
    redirect("form.php");
  }
  catch (Exception $e) {
    die('Erreur : ' . $e->getMessage());
  }
}
/* Comme inserer_objet(), mais en mettant à jour l'objet désigné par 'id' */
function update_objet($bdd, $id) {
  try {
    $req = $bdd->prepare("UPDATE `objets` SET
    `name`=?,`fullname`=?,`desc`=?,`cat`=? WHERE `id`=?;");
    $name = trim($_POST["name"]);
    $fullname = trim($_POST["fullname"]);
    $desc = trim($_POST["desc"]);
    $cat = trim($_POST["cat"]);
    $req->execute(array($name, $fullname, $desc, $cat, $id));
    $_SESSION["objet_courant_name"] = $name;
  }
  catch (Exception $e) {
    die('Erreur : ' . $e->getMessage());
  }
}
?>
```

fichier D.37 : CMS/file_upload.js

```
/* Initialement :
 * filedrag.js - HTML5 File Drag & Drop demonstration
 * Featured on SitePoint.com
 * Developed by Craig Buckler (@craigbuckler) of OptimalWorks.net
 *
 * Récupère les fichiers déposés sur la page pour les afficher à l'utilisateur,
 * en récupérer certaines métadonnées et les envoyer sur le serveur par
 * l'intermédiaire d'un script php.
 */

niveau = "FAUX;base;moyen;avancé";
img_type = "FAUX;icone;tableau;photo;plan;dessin;autre";
pos = "FAUX;gauche;droite";

function $id(id) { return document.getElementById(id); };
/* Crée un champ de sélection multiple HTML identifié par l'utilisateur par
 * 'label' et en interne par 'name' dont les différents sont dans 'value'. */
function create_select(root, name, label, values) {
  add_element(root, "label", label+' :', {"for":name});
  var select = add_element(root, "select", "", {"name":name, "id":name});
  values = values.split(";");
  for (var i = 0; i < values.length; i++) {
    add_element(select, "option", values[i], {"value":values[i]});
```

```
  };
}
/* Crée les éléments de formulaire qui stockeront les informations communes tous
 * les contenus, dans ce cas celui identifié par 'hash'. */
function add_bloc_form(div, hash, image) {
  if (image === undefined) {
    image = false;
  }
  add_element(div, "progress", "", {"id":"progress_"+hash, "max":100});
  add_element(div, "br", "", {});
  add_element(div, "label", "Description :", {"for":"desc_"+hash});
  add_element(div, "textarea", "", {"name":"desc_"+hash, "id":"desc_"+hash,
    "placeholder":"Pas plus de 250 caractères", "rows":5, "cols":48,
    "maxlength":250});
  create_select(div, "niveau_"+hash, "Niveau", niveau);
  create_select(div, "pos_"+hash, "Position", pos);
  if (image) {
    create_select(div, "type_"+hash, "Type", img_type);
  }
  add_element(div, "label", "Ordre :", {"for":"ordre_"+hash});
  add_element(div, "input", "", {"id":"ordre_"+hash, "name":"ordre_"+hash,
    "type":"number", "size":3, "min":0, "max":100,
    "value":parseInt(_compteur)});
}
/* ajoute à 'root' un enfant de type 'name' contenant 'text' et ayant pour
 * attributs le tableau 'attrib'. Renvoie l'enfant ainsi créé. */
function add_element(root, name, text, attrib) {
  var el = root.appendChild(document.createElement(name));
  for (var key in attrib) {
    el.setAttribute(key, attrib[key]);
  }
  el.textContent = text;
  return el;
}

all_file={};
_compteur=($id("ordre_max") !== null)?parseInt($id("ordre_max").value):0;
(function() {

  /* Si 's' est un nom de fichier, renvoie son nom et vice-versa */
  function convert_file(s) {
    if (s.indexOf('.') !== -1) {
      return hex_sha1(s).substr(0,8);
    }
    else {
      return all_file[s];
    }
  }

  // file drag hover
  function FileDragHover(e) {
    e.stopPropagation();
    e.preventDefault();
    e.target.className = (e.type == "dragover" ? "hover" : "");
  }


  // file selection
  function FileSelectHandler(e) {

    // cancel event and hover styling
    FileDragHover(e);

    // fetch FileList object
    var files = e.target.files || e.dataTransfer.files;

    // process all File objects
    for (var i = 0, f; f = files[i]; i++) {
      ParseFile(f);
      UploadFile(f);
    }

  }

  // upload files
  function UploadFile(file) {
```

```
  var xhr = new XMLHttpRequest();
  if (xhr.upload) {
    var hash = convert_file(file.name);
    var progress = $id("progress_"+hash);
    xhr.upload.addEventListener("progress", function(e) {
      progress.value = (e.lengthComputable)?parseInt(100*e.loaded/e.total) :50;}, false);
    // file received/failed
    xhr.onreadystatechange = function(e) {
      if (xhr.readyState == 4 && xhr.status !== 200) {
        progress.className ="failure";
      }
    };
    xhr.open("POST", $id("upload").action);
    xhr.setRequestHeader("X_FILENAME", file.name);
    console.log("send xhr request");
    xhr.send(file);
  }
}

// read file informations
function ParseFile(file) {
  console.log(file.type);
  var hash = convert_file(file.name);
  var ajoute=true;
  _compteur++;

  /* TODO si le fichier existe déjà et qu'on upload une nouvelle version, on
   * ne crée pas  un nouveau div pour lui mais surtout, on ne met pas à jour
   * les méta données donc la  base de données est dans un état incohérent.
   * Pour l'instant, ces informations ne servent à rien donc ce n'est pas
   * dramatique.
   * En revanche, ce serait plus sympa pour l'utilisateur de * mettre src à jour
   */

  if ($id(hash) !== null) {
    ajoute = false;
  }
  all_file[hash] = file.name;
  if (file.type.indexOf("text") == 0) {
    var reader = new FileReader();
    var plain = file.type.indexOf("plain") == 5;
    if (ajoute) {
    var div = add_element($id("texts"), "div", "", {"class" :"content",
      "id" :hash});
    add_element(div, "input", "", {"type" :"hidden", "name" :"mime_"+hash,
      "id" :"mime_"+hash, "value" :file.type});
    add_element(div, "input", "", {"type" :"hidden", "name" :"len_"+hash,
      "id" :"len_"+hash, "value" :0});
    var txt_tag = (plain?"span" :"span");
    add_element(div, "textarea", "", {"class" :"resize",
      "name" :"content_"+hash, "id" :"content_"+hash});
    add_bloc_form(div, hash);
    }
    reader.onload = function(e) {
      $id("content_"+hash).textContent = (e.target.result);
      $id("len_"+hash).value = e.target.result.split(" ").length;
    }
    reader.readAsText(file);
  }

  if (file.type.indexOf("image") == 0) {
    var reader = new FileReader();
    if (ajoute) {
    var div = add_element($id("images"), "div", "", {"class" :"content",
      "id" :hash});
    add_element(div, "input", "", {"type" :"hidden", "name" :"mime_"+hash,
      "id" :"mime_"+hash, "value" :file.type});
    add_element(div, "input", "", {"type" :"hidden", "name" :"len_"+hash,
      "id" :"len_"+hash, "value" :file.size});
    add_element(div, "input", "", {"type" :"hidden", "name" :"h_"+hash,
      "id" :"h_"+hash, "value" :0});
    add_element(div, "input", "", {"type" :"hidden", "name" :"w_"+hash,
      "id" :"w_"+hash, "value" :0});
    add_element(div, "img", "", {"class" :"resize", "src" :"", "id" :"img_"+hash});
    add_bloc_form(div, hash, true);
    }
```

```
    reader.onload = function(e) {
      var img = new Image();
      /* TODO ne marche pas avec les jpeg ? */
      img.onload = function() {
        $id('w_'+hash).value = this.width;
        $id('h_'+hash).value = this.height;
        console.log(this.width);
      };
      img.src = file.name;
      $id("img_"+hash).src = e.target.result;
    }
    reader.readAsDataURL(file);
  }

  if (file.type.indexOf("audio") == 0) {
    var reader = new FileReader();
    if (ajoute) {
    var div = add_element($id("audios"), "div", "", {"class":"content", "id":hash});
    add_element(div, "input", "", {"type":"hidden", "name":"mime_"+hash,
      "id":"mime_"+hash, "value":file.type});
    add_element(div, "input", "", {"type":"hidden", "name":"len_"+hash,
      "id":"len_"+hash, "value":file.size});
    add_element(div, "audio", "", {"controls":"controls", "preload":"auto",
      "src":"", "id":"son_"+hash});
    add_bloc_form(div, hash);
    add_element(div, "label", "En boucle :", {"for":"boucle_"+hash});
    add_element(div, "input", "", {"type":"checkbox", "name":"boucle_"+hash,
      "id":"boucle_"+hash});
    add_element(div, "label", "Spatial :", {"for":"spatial_"+hash});
    add_element(div, "input", "", {"type":"checkbox", "name":"spatial_"+hash,
      "id":"spatial_"+hash});
    }
    reader.onload = function(e) {
      $id("son_"+hash).src = e.target.result;
      //TODO marche pas sous firefox
      $id("son_"+hash).addEventListener("loadedmetadata",
          function(){$id("len_"+hash).value=this.duration;}, false);
    }
    reader.readAsDataURL(file);
  }
  if (file.type.indexOf("video") == 0) {
    var reader = new FileReader();
    if (ajoute) {
    var div = add_element($id("videos"), "div", "", {"class":"content", "id":hash});
    add_element(div, "input", "", {"type":"hidden", "name":"mime_"+hash,
      "id":"mime_"+hash, "value":file.type});
    add_element(div, "input", "", {"type":"hidden", "name":"len_"+hash,
      "id":"len_"+hash, "value":file.size});
    add_element(div, "video", "", {"controls":"controls", "preload":"auto",
      "class":"resize", "src":"", "id":"vid_"+hash});
    add_bloc_form(div, hash);
    }
    reader.onload = function(e) {
      $id("vid_"+hash).src = e.target.result;
    }
    reader.readAsDataURL(file);
  }
}

function Init() {

  var fileselect = $id("fileselect"),
      filedrag = $id("filedrag"),
      submitbutton = $id("submitbutton");

  fileselect.addEventListener("change", FileSelectHandler, false);

  // is XHR2 available?
  var xhr = new XMLHttpRequest();
  if (xhr.upload) {
    // file drop
    filedrag.addEventListener("dragover", FileDragHover, false);
    filedrag.addEventListener("dragleave", FileDragHover, false);
    filedrag.addEventListener("drop", FileSelectHandler, false);
    filedrag.style.display = "block";
  }
```

```
    /* Récupérer les différentes énumérations de la base SQL, écrite par
     * gestion.php au début de la session. Pas très élégant néanmoins...
     */
    var request = new XMLHttpRequest();
    request.onreadystatechange = function(e) {
      if (request.readyState == 4 && request.status === 200)
      { niveau = request.responseText; } };
    request.open('GET', "niveau_vals", false);
    request.send();

    request.onreadystatechange = function(e) {
      if (request.readyState == 4 && request.status === 200)
      { img_type = request.responseText; } };
    request.open('GET', "type_vals", false);
    request.send();

    request.onreadystatechange = function(e) {
      if (request.readyState == 4 && request.status === 200)
      { pos = request.responseText; } };
    request.open('GET', "pos_vals", false);
    request.send();

  }

  // call initialization file
  if (window.File && window.FileList && window.FileReader) {
    Init();
  }
})();
```

fichier D.38 : CMS/validation.js

```
/* Vérifie les éléments entrés dans un formulaire
 * TODO : il y aurait normalement beaucoup d'autre éléments à valider, comme par
 * exemple les champs censés contenir des valeurs numériques comprise entre des
 * borne mais par manque de temps, il faudra ce contenter de cette preuve de
 * concept...
 */
/* les variables globales qui contiennent le résultat de chaque validation */
var _val_nom_objet = false;
/* Vérifie l'élément 'e' suivant une règle interne */
function verif(e) {
  kind = e.id.split('_')[0];
  switch(kind) {
    case 'name':
      // La regexp est inversée
      var re = /[^a-z_]/;
      if (e.value.length >= 3 && e.value.length <= 20 && !re.test(e.value)) {
        e.className="";
        e.title="";
        _val_nom_objet = true;
      }
      else {
        e.className="invalide";
        e.title="entre 4 et 20 lettres minuscules ou underscore maximum (et seulement ça)";
        _val_nom_objet = false;
      }
      return _val_nom_objet;
    default:
      return true;
  }
}
/* Vérifie chacun des champs vérifiable de 'form' et soumet le formulaire si ils
 * sont tous corrects.
 */
function verif_objet(form) {
  verif(document.getElementById("name"));
  if (_val_nom_objet) {
    form.submit();
  }
}
```

fichier D.39 : CMS/bdd−structure.sql

```
SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO";
```

```
SET time_zone = "+00 :00";

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;


DROP TABLE IF EXISTS `blocs`;
CREATE TABLE IF NOT EXISTS `blocs` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `objet` int(10) unsigned NOT NULL,
  `path` varchar(100) NOT NULL,
  `mime` varchar(25) NOT NULL,
  `desc` varchar(250) NOT NULL,
  `len` mediumint(8) unsigned NOT NULL,
  `niveau` enum('base','moyen','avancé') NOT NULL,
  `pos` enum('gauche','droite') NOT NULL,
  `ordre` smallint(5) unsigned NOT NULL,
  PRIMARY KEY (`id`),
  KEY `objet` (`objet`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=77 ;

DROP TABLE IF EXISTS `commentaires`;
CREATE TABLE IF NOT EXISTS `commentaires` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `objet` int(10) unsigned NOT NULL,
  `quand` datetime NOT NULL,
  `pseudo` varchar(127) NOT NULL,
  `comment` text NOT NULL,
  PRIMARY KEY (`id`),
  KEY `objet` (`objet`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=3 ;

DROP TABLE IF EXISTS `images`;
CREATE TABLE IF NOT EXISTS `images` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `bloc` int(10) unsigned NOT NULL,
  `w` mediumint(8) unsigned NOT NULL,
  `h` mediumint(8) unsigned NOT NULL,
  `type` enum('icone','tableau','photo','plan','dessin','autre') NOT NULL,
  PRIMARY KEY (`id`),
  KEY `bloc` (`bloc`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=30 ;

DROP TABLE IF EXISTS `objets`;
CREATE TABLE IF NOT EXISTS `objets` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(20) NOT NULL,
  `fullname` varchar(100) NOT NULL,
  `desc` text NOT NULL,
  `cat` enum('tableau','mobilier','architecture','autre') NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `name` (`name`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=26 ;

DROP TABLE IF EXISTS `sons`;
CREATE TABLE IF NOT EXISTS `sons` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `bloc` int(10) unsigned NOT NULL,
  `spatial` tinyint(1) NOT NULL,
  `boucle` tinyint(1) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `bloc` (`bloc`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=2 ;

DROP TABLE IF EXISTS `textes`;
CREATE TABLE IF NOT EXISTS `textes` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `bloc` int(10) unsigned NOT NULL,
  `content` text NOT NULL,
  PRIMARY KEY (`id`),
  KEY `bloc` (`bloc`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=45 ;

DROP TABLE IF EXISTS `videos`;
```

```
CREATE TABLE IF NOT EXISTS `videos` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `bloc` int(10) unsigned NOT NULL,
  PRIMARY KEY (`id`),
  KEY `bloc` (`bloc`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=3 ;

ALTER TABLE `blocs`
  ADD CONSTRAINT `blocs_ibfk_1` FOREIGN KEY (`objet`) REFERENCES `objets` (`id`)
  ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE `commentaires`
  ADD CONSTRAINT `commentaires_ibfk_1` FOREIGN KEY (`objet`) REFERENCES `objets`
  (`id`) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE `images`
  ADD CONSTRAINT `images_ibfk_1` FOREIGN KEY (`bloc`) REFERENCES `blocs` (`id`)
  ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE `sons`
  ADD CONSTRAINT `sons_ibfk_1` FOREIGN KEY (`bloc`) REFERENCES `blocs` (`id`) ON
  DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE `textes`
  ADD CONSTRAINT `textes_ibfk_1` FOREIGN KEY (`bloc`) REFERENCES `blocs` (`id`)
  ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE `videos`
  ADD CONSTRAINT `videos_ibfk_1` FOREIGN KEY (`bloc`) REFERENCES `blocs` (`id`)
  ON DELETE CASCADE ON UPDATE CASCADE;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```