Sébastien Schneider

Julien Hage

Olivier Nobili

Johanna Nobili

Projet de Maîtrise :

Modeleur de Métaformes

18 juin 1997

Résumé

Notre projet a consisté à créer un logiciel de modelage de formes molles (*metaballs*, métaformes). Ceci implique la réalisation d'une interface, que nous avons appelée Xblob, permettant la modélisation et la visualisation des métaformes.
Nous nous proposons tout d'abord de situer le problème : nous allons définir ce qu'est une métaforme, voir des exemples d'applications que l'on peut faire de ce type d'objet ce qui permettra de bien comprendre le sujet de notre projet.
Nous décrirons ensuite ce que nous avons réalisé : l'état d'avancement du logiciel, les fonctionnalités existantes de celui-ci, mais aussi les nouveautés et améliorations que nous lui apporterons par la suite.
Nous aborderons également des aspects techniques en décrivant les principaux algorithmes et méthodes que nous avons utilisés.
Nous expliquerons enfin comment utiliser notre application : nous donnerons un mode d'emploi.

Table des matières

Liste des tableaux

Liste des figures

Partie I

Introduction

Avant d'expliquer le sujet de notre projet, nous nous proposons définir ce qu'est une métaforme et de voir également quelques exemples d'application de ces objets.

1 Métaformes : définition

1.1 Des *metaballs* ...

Nous avons parlé des formes molles : *metaballs*, métaformes.
Expliquons premièrement le cas de base, la *metaball* :
Une *metaball* est composée de plusieurs atomes (points) qui émettent ou absorbent de l'énergie. Un point P de l'espace reçoit de l'énergie de tous les émetteurs , on fait la somme algébrique de toutes ces énergies (certaines peuvent en effet être négatives lorsque la puissance [1] de l'émetteur est négative).
Le but est de trouver la surface[2] qui rassemble tous les points où l'énergie est égale à une certaine valeur seuil [3] que l'on fixe.
On obtient alors une "peau" autour des émetteurs comme le montre la figure 1 page 1.

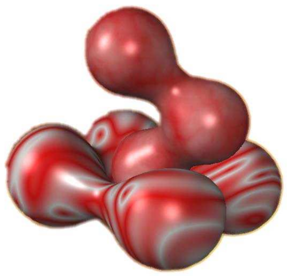

Figure 1: Exemple de métaforme

1.2 ... aux métaformes

Les métaformes constituent une extension des *metaballs*, puisqu'elles sont constituées de plusieurs types d'émetteurs en plus du point. Nous avons choisi de traiter 5 types d'émetteurs :

- Le point : voir la figure 2 page 2
- Le segment : voir la figure 3 page 2

[1] Power.
[2] Ces surfaces sont souvent appelées iso-surfaces ou surfaces implicites : comme le nom l'indique, la surface de l'objet n'est pas modélisée explicitement, on utilise des équations.
[3] Treshold.

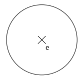

Figure 2: L'émetteur point ⇒ *Metaball*

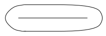

Figure 3: L'émetteur segment

- Le triangle (facette) : voir la figure 4 page 2

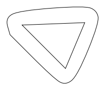

Figure 4: L'émetteur triangle

- La ligne brisée : voir la figure 5 page 3
- Le réseau de triangles : voir la figure 6 page 3

La distance d'un point P de l'espace à un émetteur quelque soit son type, est la distance la plus courte de P à n'importe quel point de l'émetteur.
Une métaforme possède un squelette qui peut donc comporter 5 types d'émetteurs différents, pour le calcul de la surface tout se passe comme nous l'avons vu précédemment pour une *metaball*.

1.3 Un exemple

Les métaformes permettent de créer plus facilement (qu'avec une simple *metaball*) des objets complexes.
Nous nous proposons, avec la figure 7 page 4, de voir un exemple d'utilisation d'une métaforme.
Nous désirons représenter un personnage : nous allons utiliser l'émetteur point pour modéliser la tête, comme le montre la figure, l'émetteur triangle pour le tronc, et l'émetteur ligne brisée pour les bras et les jambes. Cet ensemble d'émetteur constitue une métaforme. La figure 8 page 5 montre le résultat : on peut remarquer la souplesse des formes du bonhomme. On voit ainsi que l'on arrive rapidement et facilement à modéliser divers objets aux contours souples et arrondis avec un bon résultat .

1.4 Au niveau d'un émetteur

Une métaforme possède un seuil d'énergie qui lui est propre et qui est commun à tous les émetteurs qui la constituent.
En revanche, chaque émetteur possède sa propre puissance[4] qui représente

[4] Power.

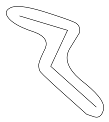

Figure 5: L'émetteur ligne brisee

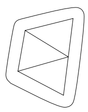

Figure 6: L'émetteur réseau de facettes

l'intensité de l'émission énergétique de l'émetteur. Si la puissance est négative, l'émetteur absorbe de l'énergie (émise par d'autres émetteurs) au lieu d'en émettre.
Un émetteur possède aussi sa propre fonction de distribution de l'énergie[5]. Cette fonction calcule en chaque point de l'espace une valeur qui est fonction de la distance entre le point et l'émetteur. Cette valeur représente l'énergie que le point reçoit de l'émetteur. Plus le point est loin de l'émetteur, moins il reçoit d'énergie. A chaque fonction est associé un rayon qui représente la limite d'influence de l'émetteur : l'énergie reçue en un point P situé à une distance d égale au rayon (ou supérieure) de l'émetteur est nulle. La section 2 page 10 parle plus en détail de ces fonctions.

2 Les métaformes : leurs applications

Elles sont particulièrement adaptées pour modéliser des objets au rendu lisse qui se déforment en se déplaçant et qui peuvent subir des modifications topologiques. De tels objets changent de forme à cause de phénomènes externes, se déforment pendant une collision avec d'autres objets ou se fondent dans leur environnement.
La déformation d'un objet se fait simplement en changeant la relation spaciale entre primitives. Une surface fermée et compacte est toujours garantie bien qu'elle puisse être constituée de plusieurs parties déconnectées.
Les surfaces implicites peuvent être utilisées pour diverses applications.
Blinn qui a le premier introduit l'utilisation des surfaces implicites pour le graphisme et l'animation sur ordinateur, utilise les *blobby* molécules (voir [Bli82]) pour visualiser des champs énergétiques d'électrons. Tatsumi présente l'utilisation des métaballs pour la modélisation de certaines cellules dans le foie de morue. Fujita représente des éclaboussements d'eau grâce aux métaballs. Max et Wyvill ont rendu du corail en utilisant des objets mous (*Soft Objects*, voir [Gra93]). Payne et Toga ont modélisé un cerveau de rat (voir [PT]). Les objets mous ont aussi été utilisés pour visualiser des éclairs (voir [PW94]). Au Japon, les métaformes sont beaucoup utilisées pour la modélisation et l'animation de formes complexes comme des parties de l'anatomie humaine (voir [RW90]).

[5] ou fonction de densité

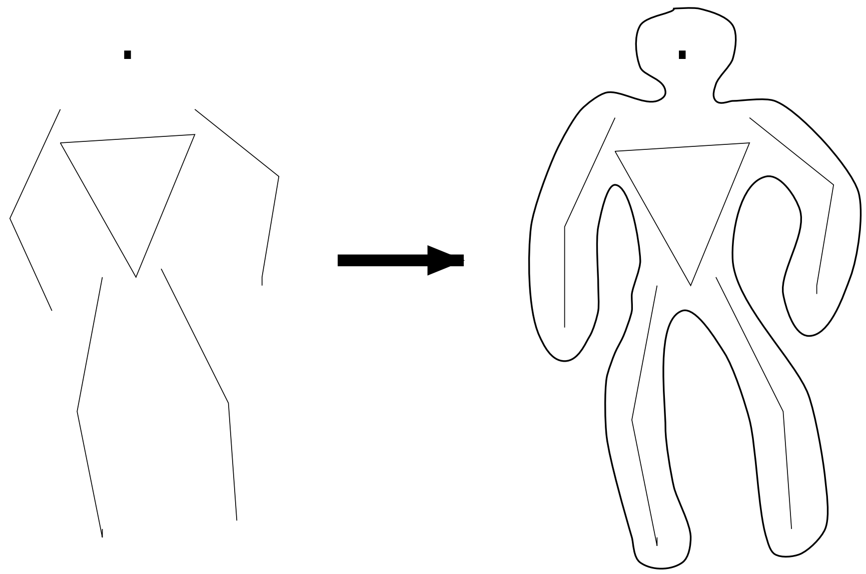

Figure 7: Squelette composé de différents types d'émtteurs

Les métaformes servent dans le domaine de la simulation sur ordinateur. On les utilise en physique pour modéliser des détections de collisions, des déformations ou des fusions d'objets.
Dans le domaine de l'animation, les avantages des métaformes sont exploités pour les déformations d'objets, les *morphings* .... Des articles présentent des exemples d'animations qui ont beaucoup de succès, en voici quelques uns : des lettres descendant un escalier (voir [Wyvb]), un train traversant une banque (voir [Wyva]), un guitariste émergeant de bulles de dentifrice (voir [Bei]).

3 Sujet de notre projet

Maintenant que nous avons définit et décrit les métaformes, que nous avons mentionné quelques une de leurs applications, nous pouvons expliquer en quoi a consisté notre projet.
Notre travail a été de réaliser un logiciel de modelage et de visualisation de métaformes. Pour réaliser notre programme Xblob, nous avons utilisé la librairie 3D 'Photon 4D' créée par Franck Diard. Cette librairie a pour but de calculer des images en *raytracing*[6] sur de nombreux types d'objets (cube, sphère,...), cependant elle ne gère pas les métaformes. Nous avons dû, dans un premier temps, ajouter la possibilité de créer et de traiter des métaformes dans Photon 4D, de manière à ce que ces dernières soient gérées comme n'importe quel autre objet dans la librairie.
Ensuite nous avons créé notre outil Xblob qui est composé d'une interface qui permet de modéliser et de les visualiser dans différents modes de rendus (voir la partie 1 page 29 sur l'interface et son fonctionnement). Notre logiciel permet

[6] Lancé de rayons.

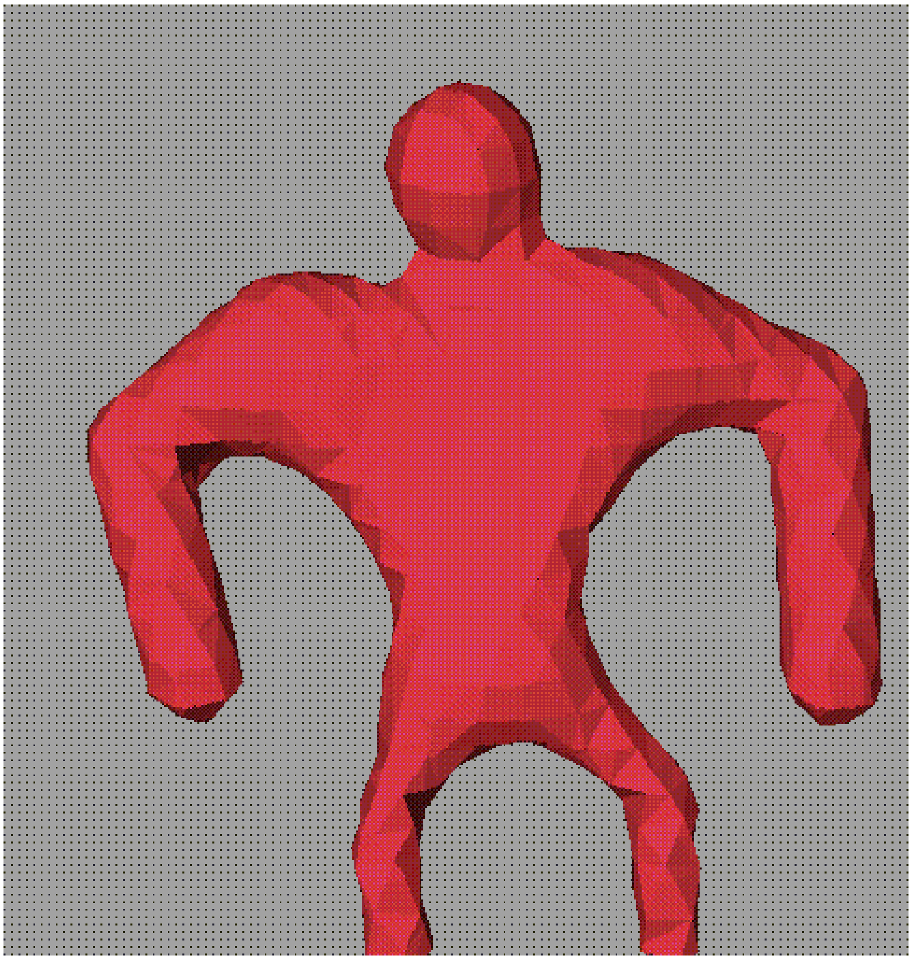

Figure 8: Le bonhomme en *flat*

aussi d'animer les métaformes créées.
Une autre fonctionnalité importante est de pouvoir importer des objets 3D provenant d'autres modeleurs (différents formats), et de les visualiser sous forme de métaformes. Ceci est également valable dans l'autre sens : nos métaformes peuvent être exportées dans divers formats pour être visualisées sous forme de réseau de facettes (voir la section 1.2 page 1) dans d'autres logiciels.
La technologie des métaformes est récente et encore en développement, les créateurs de logiciels de modelage s'y intéressent et certains ont déjà intégré les *metaballs* dans leur modeleur. Nous avons donc pu explorer un domaine nouveau

3.1 Les outils utilisés

- Logiciels
  Notre application est programmée en langage C++ (le compilateur utilisé

est g++), ce qui nous a permis de profiter des nombreuses possibilités (généricité, polymorphisme, héritage) d'un langage à objet tout en ayant un code généré efficace et rapide.
Nous avons utilisé la librairie Photon4D comme nous l'avons expliqué précedemment pour réaliser notre projet.
Nous nous nous sommes servis de la librairie MOTIF pour la création de l'interface.
En plus des outils de programmation énoncés précedemment, nous avons utilisé un script zsh que nous avons écrit qui permet une gestion du travail à plusieurs sur les mêmes fichiers, voir la section 3.1.1 page 6

- Plate-formes
  Nous avons développé notre logiciel sous UNIX (Solaris, Linux), mais nous avons aussi une version moins aboutie pour Windows NT/95 (utilisation de Visual C++).

3.1.1 Notre script zsh : mk

Nous avons programmé, pour pouvoir travailler tous les quatre ensemble, un utilitaire que nous avons appelé mk.
Son fonctionnement est décrit dans la figure 9 page 6.

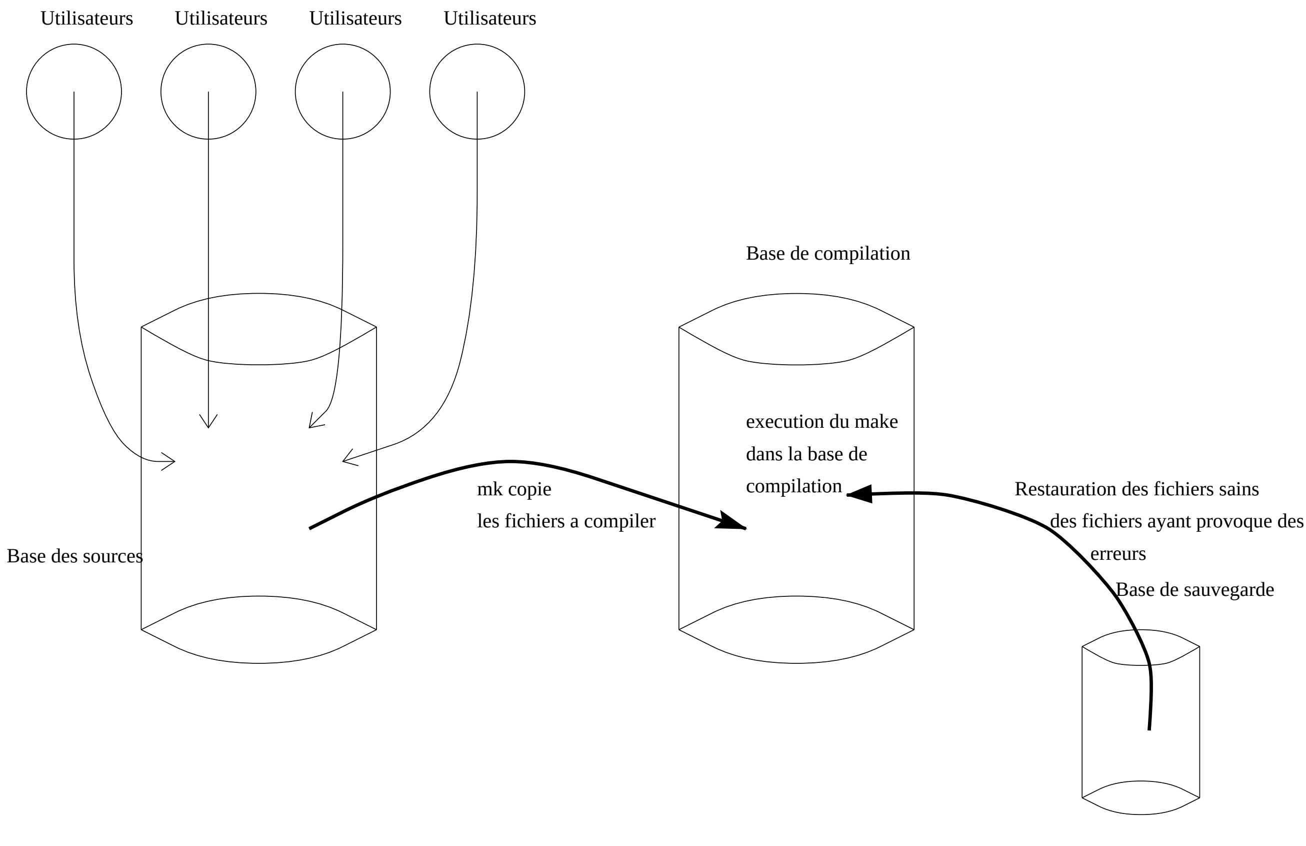

Figure 9: Fonctionnement de mk

Le problème que gomme mk est les erreurs dûes à la recompilation de tous les fichiers qui sont édités.
Par exemple imaginons qu'un utilisateur A travaille sur un fichier toto.cpp et qu'un utilisateur B travaille sur le fichier titi.cpp. Maintenant A veut recompiler pour voir ses modifications de toto.cpp prendre effet mais le make va aussi lancer la compilation de titi.cpp puisque B est en train de le modifier, et cette compilation va générer des erreurs car le fichier est en cours d'édition.
Ce genre de problème arrive constamment surtout lorsque on travaille à 4. Le problème peut etre detourné de plusieurs façons mais toutes font perdre du temps. mk résoud ces problèmes de manière élégante.
Le principe de mk est basé sur le fait que les sources qu'on édite sont toutes ensemble dans un répertoire, il n'y aura donc jamais de compilation dans ce répertoire. Quand un utilisateur veut faire une compilation il passe par l'intermediaire de mk. mk fait une copie des fichiers que l'utilisateur a modifié dans le repertoire de compilation, apres avoir sauvegardé les fichiers qu'il y avait avant. Les fichiers du repertoire de compilation sont toujours sans erreur (de compilation) car dès qu'il s'en produit une, la précédente version du fichier (celle qui a été sauvegardée) sera restaurée dans le fichier de compilation ainsi le prochain utilisateur à essayer de compiler disposera d'un "espace" de compilation sain. D'autre part mk n'autorise qu'une seule compilation à la fois en gérant un système de verrous.

3.2 Ce qui est fait et ce qu'il reste à faire

Nous avons donc créé un modeleur de métaformes.

3.2.1 Ses fonctionnalités actuelles

- Modeliser avec 5 types d'émetteurs possibles.
  Comme l'émetteur réseau de facettes est très general, il nous permet de représenter n'importe quel objet.
- Positionner et déplacer des émetteurs dans l'espace.
- Modifications des attributs des émetteurs, métaformes, et des objets issus d'une sélection multiple.
  Cela concerne la position dans l'espace, la puissance, la fonction d'énergie ainsi que l'iso valeur (seuil) de la métaforme. Le fait de pouvoir modifier la fonction de répartition d'énergie dans l'espace permet de donner aux métaformes, des formes très variées et particulières (gondolement de la surface etc).
- Nous pouvons visualiser la scène avec différents rendus :
  - Points
  - Fil de fer
  - Faces cachées
  - Gouraud
  - Phong
  - Raytracing

- L'utilisateur peut donner la précision voulue pour le maillage. On peut donc modéliser avec une précision moyenne et avoir une grande précision pour la visualisation finale.
- Nous pouvons aussi sauvegarder/charger une scène grâce à des scripts. Xblob nous permet aussi d'importer des objets variés provenant d'autres logiciels de synthèse d'image. De plus, nous pouvons sauvegarder une scène au format de ces logiciels.
- Xblob offre aussi la possibilité de faire des animations. En effet, il suffit de spécifier le nombre d'images que l'on désire avoir pour l'animation, et d'enregistrer la scène aux différents moments que l'on souhaite.
- Xblob permet aussi de modéliser à plusieurs en même temps via le reseau. Les personnes desirant travailler en reseau se partagent la même scene. Elles peuvent modéliser simultanément. Les interactions d'une personne sont perçues par les autres. Cela permet de créer des scènes complexes rapidement.

3.2.2 Evolutions à venir

- Possibilité d'avoir une fonction 'undo' (annule la dernière action)
- Permettre à l'utilisateur de placer lui-même des lampes et des caméras.
- Sauvegarder une animation dans un fichier graphique.
- Ameliorer l'ergonomie générale du logiciel.
- Etendre les fonctions de modelage.

Partie II

Techniques et algorithmes

Nous verrons dans cette partie les principaux algorithmes et méthodes que nous avons utilisés pour résoudre divers problèmes.

1 Calcul de distance d'un point par rapport à un émetteur

1.1 Distance d'un point par rapport à chaque type d'émetteur

Comme nous l'avons vu plus haut, pour voir une métaforme il faut regarder en tout point de l'espace (dans la théorie) pour trouver la surface où l'énergie est égale au seuil.
L'énergie de ces points est la somme de l'énergie reçue de chaque émetteur. Cette énergie reçue dépend de la distance à l'émetteur, il faut donc calculer la distance dans l'espace d'un point par rapport à chaque type d'émetteur :

Type de l'émetteur	**Distance d'un point P de l'espace par rapport à l'émetteur**
Point	Trivial
Segment	Si le projeté P' du point P appartient au segment, on prend la distance PP', sinon on calcule la distance de P à chaque extrémité du segment et on prend la plus petite des 2.
Triangle (facette)	Si le projeté P' de P appartient à la facette, on prend la distance PP', sinon on calcule la distance de P à chaque segment formant le triangle et on prend la plus petite des 3.
Ligne brisée	Pour chaque segment formant la ligne brisée, on calcule la distance du segment par rapport à P, et on prend le minimum de ces distances.
Le réseau de facettes	Pour chaque facette du réseau, on calcule la distance de la facette par rapport à P, et on prend le minimum de ces distances.

Tableau 1: Calcul de distances

1.2 Problème de la cassure

Pour les lignes brisées et le réseau de facettes, du "côté interne" de la pliure se forme un effet de cassure inesthétique. Cette cassure se matérialise par un angle vif, comme le montre la figure 10 page 10.

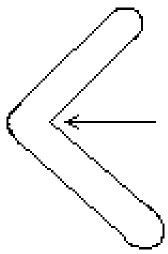

Figure 10: Illustration du problème de la cassure : on voit bien l'angle vif.

Pour gommer cette cassure nous devons raffiner notre méthode de calcul de distance par rapport à une ligne brisée.
En effet la méthode que nous venons de voir plus haut produit un passage brutal entre 2 segments consécutifs.
Pour éviter cela nous avons une méthode expérimentale :

- on fixe une valeur ϵ,
- On conserve les 2 distances minimales d1 et d2 avec $d_1 < d_2$,
- si $(d_2 - d_1) < \epsilon$ cela veut dire que nous sommes proche de la cassure, on renverra alors une distance inférieure à d1 pour effectuer le lissage.

Cette technique permet "d'adoucir" la cassure, comme le montre la figure 11 page 10.

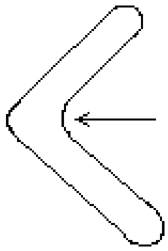

Figure 11: Résolution du problème de la cassure : l'angle a été remplacé par une courbe.

2 Fonctions de distribution de l'énergie

Comme nous l'avons vu, la fonction de distribution de l'énergie d'un émetteur permet de calculer en chaque point de l'espace une valeur qui est fonction de

la distance entre le point et l'émetteur. Cette valeur représente l'énergie que le point reçoit de l'émetteur.
Il est possible de définir cette fonction comme on le souhaite en respectant tout de même les lois physiques : la fonction doit être monotone décroissante et doit tendre vers 0 quand la distance entre le point et l'émetteur tend vers le rayon (voir section 1.4), par convention l'énergie vaut 0 pour un point situé à une distance d égale au rayon de l'émetteur. On peut comprendre facilement cette condition : l'émetteur est une source d'énergie, il est normal que plus on s'éloigne de cette source, plus l'énergie que l'on reçoit d'elle diminue. On peut imaginer par exemple une lampe, plus on s'éloigne d'elle et moins on est éclairé.
La figure 12 page 11 montre des exemples de courbes de fonctions d'énergie valides pour associer aux émetteurs.
Les abscisses représentent la distance du point à l'émetteur, les ordonnées la valeur de l'énergie (qui est fonction de la distance). R est le rayon d'influence de l'émetteur, l'énergie vaut 0 pour une distance égale à R. En 0, l'énergie est maximale (Em).

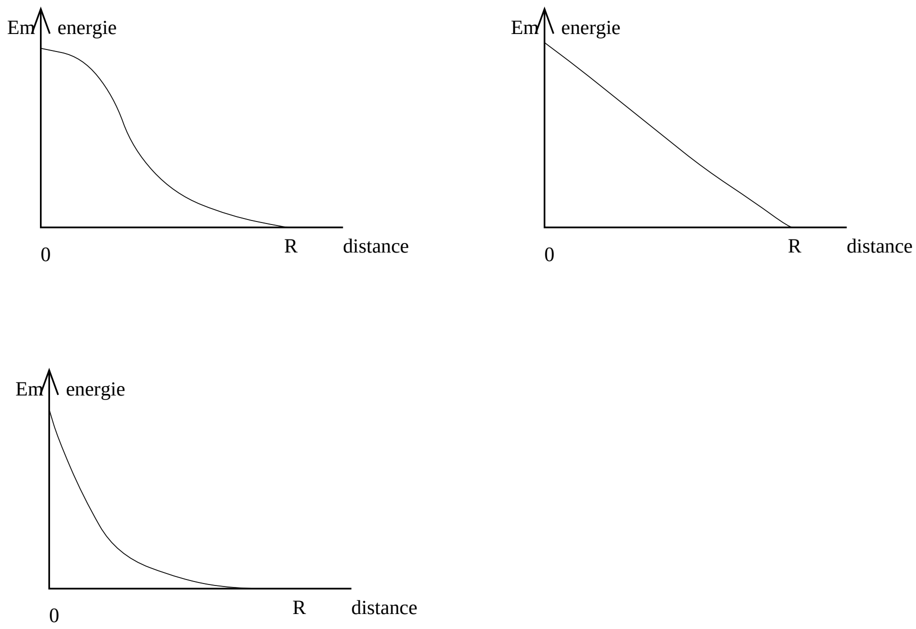

Figure 12: Exemples de fonctions d'énergie

Dans notre application, nous avons défini plusieurs fonctions d'énergie différentes que l'utilisateur peut associer aux émetteurs qu'il crée. Ces fonctions sont définies par une formule mathématique, nous donnons ci-dessous 2 de ces fonctions prédéfinies.

$$e(P) = \left\{ \quad (\frac{1-d}{R})^{(\frac{1}{|p|})} \right.$$

où e(P) est l'énergie que le point P reçoit de l'émetteur, d représente la distance de P à l'émetteur, p est la puissance de l'émetteur et R est le rayon de l'émetteur.

La figure 13 page 12 montre la courbe de cette fonction pour p = 0.5.

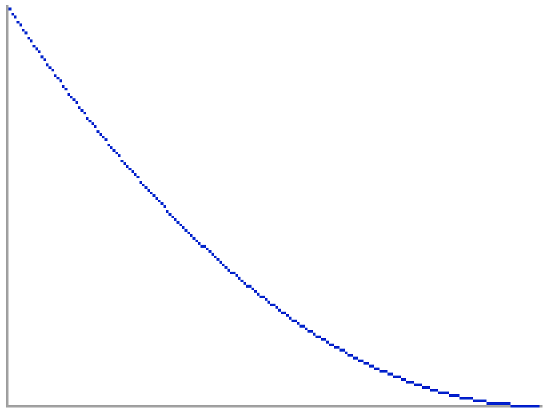

Figure 13: Une des fonctions d'énergie prédéfinies pour un émetteur de puissance 0.5.

$$e(P) = \begin{cases} p(1 - 3(\frac{d}{R})^2) & 0 \leq r_i \leq \frac{R}{3} \\ \frac{3p}{2}(1 - \frac{d}{R})^2 & \frac{R}{3} \leq d \leq R \\ 0 & R \leq d \end{cases}$$

où e(P) est l'énergie que le point P reçoit de l'émetteur, d représente la distance de P à l'émetteur, p est la puissance de l'émetteur et R est le rayon de l'émetteur. La figure 14 page 13 montre la courbe de cette fonction pour p = 0.5.

En plus de ces fonctions prédéfinies, l'éditeur de fonctions d'énergie du logiciel offre la possibilité de créer ses propres fonctions (voir la section ?? page ?? pour le mode d'utilisation). L'utilisateur doit dessiner la courbe de sa fonction (le logiciel l'empêche de définir une fonction non décroissante) et le programme calcule les valeurs d'énergie correspondant à la courbe. Ces valeurs sont stockées dans un tableau. On obtient ainsi une définition discrète de la fonction, on fera ensuite des interpolations pour calculer l'énergie sur toute la zone d'influence de l'émetteur.

Par ailleurs, nous avons besoin (en particulier pour le raytracing, voir section ?? page ??) de calculer les dérivées première et seconde des fonctions d'énergie. Pour les fonctions qui sont définies par une formule mathématique (fonctions prédéfinies), cela ne pose pas de problème : il s'agit d'un calcul de dérivé standard.

Les fonctions définies de manière discrète auront des dérivées première et seconde également définies par un ensemble de valeurs. On approxime la valeur de la dérivée en un point P en calculant la pente entre ce point P et le suivant

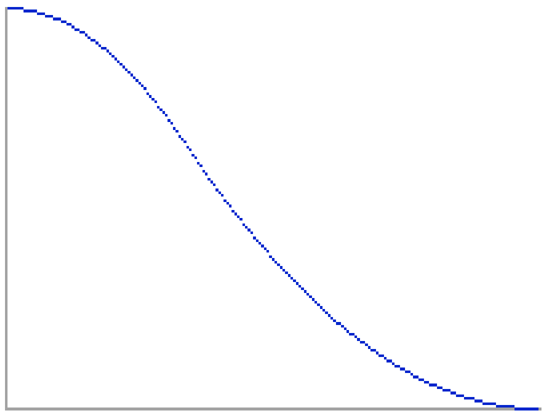

Figure 14: Une des fonctions d'énergie prédéfinies pour un émetteur de puissance 0.5.

(on pourrait affiner l'approximation en calulant la moyenne des pentes entre le point P et le point précédent, et entre le point P et le point suivant). On procéde de la même manière pour calculer les valeurs de la dérivée seconde à partir de la dérivée première.

2.1 Extension

Nous avons enrichi le concept des métaformes en modifiant la sémantique de la fonction de repartition d'énergie. Les fonctions de répartition d'energie "standard" donne des densités d'energie variant seulement en fonction de la distance du point à l'émetteur.
Nous avons rajouté la possibilité aux émetteurs d'émettre leur énergie de manière différente suivant la direction d'émission, on tient donc compte de l'angle entre le point "recepteur" et l'émetteur énergétique. On fait intervenir une fréquence angulaire pour moduler le nombre de "vagues" et une amplitude pour faire varier la taille de ces vagues. La figure 15 page 14 montre un exemple de ce que peut donner ce type d'effet, on y voit un émetteur point avec une fonction d'énergie standard à côté d'un émetteur point auquel on a modifié la fonction d'énergie comme nous venons de l'expliquer (le mode de rendu est *Phong*).

Cette technique permet avec des métaformes de base très simple de générer des objets très complexes et très beaux. Cette technique couplée à l'animation permet de créer des effets spectaculaires. On pourra rendre par exemple des chocs avec vibration sur la surface d'un corps mou par exemple.

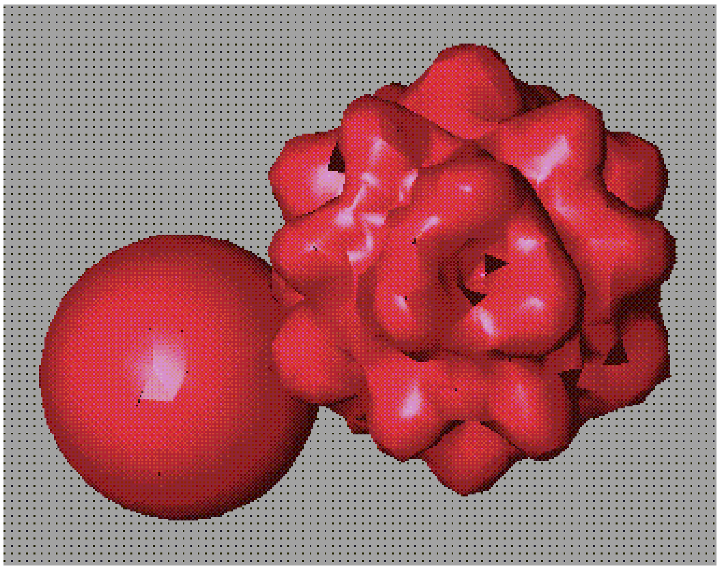

Figure 15: Un émetteur point standard et émetteur point auquel on a modifié la fonction d'énergie pour faire intervenir la direction d'émission en plus de la distance des points à l'émetteur

3 Algorithme de partitionnement de l'espace : le Marching Cube

3.1 Du Marching Cube ...

Pour visualiser nos métaformes à l'écran (dans les modes de rendu temps réel c'est à dire non *Raytracing*) il faut réussir à construire un réseau de facettes qui les approximent le plus fidèlement possible (une metaforme n'existant que par sa representation mathematique). Il n'y a pas de moyen simple d'y parvenir.
Une première approche pour l'obtention du réseau de facettes est l'algorithme du "Marching Cube" proposé en 1987 par Lorensen et Cline. Cette méthode se base sur une division de l'espace dans lequel il y a un objet à "triangulariser". Cet espace va être divisé en un certain nombre de cubes de tailles égales [1] comme le montre la figure 16 page 15.

Dans notre cas, nous allons calculer l'énergie dans les coins de chaque cube divisant l'espace. Voir la figure 17 page 15.

Si l'énergie d'un point P du cube est inférieure au seuil énergetique de la métaforme et que l'énergie d'un point Q, contigu à P, est supérieure à ce seuil alors on conclut qu'une facette passe entre ces deux points. On calcule un rapport entre les énergies et le seuil pour savoir de manière plus fine où passe la facette entre ces 2 points.
Cette approche présente des problèmes :

- légères turbulences non vues car trop petites par rapport à la taille des cubes

[1] Metagrid

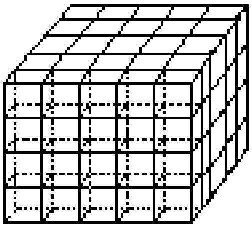

Figure 16: Une *metagrid*

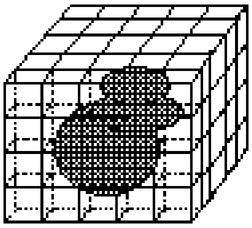

Figure 17: *Metagrid* englobant une métaforme

- si la métaforme est composée d'émetteurs relativement éloignés, les cubes sont gros ce qui fait diminuer le niveau de détails, dans des proportions parfois catastrophiques.

Le moyen de parer à cette situation serait d'augmenter le nombre de cubes dans l'espace de la métaforme mais cette technique serait beaucoup trop gourmande en calculs inutiles.

3.2 ... au Marching Cube Adaptatif

3.2.1 Méthode, algorithme

La solution se matérialise par un derivé de l'algorithme du Marching Cube : le Marching Cube Adaptatif (MCA), élaboré par Renben Shu, Chen Zhou, Mohan S. Kankanhalli (voir [RS]). Cette méthode se différencie du marching cube traditionnel par une approche récursive dans la division de l'espace, chaque cube pouvant diviser son espace en une autre série de cubes et ainsi de suite. Voir la figure 18 page 16.

Le MCA génère des facettes a un niveau n de recursivité puis regarde les angles formés entre les normales des 3 coins de chaque facette. Si les angles sont superieurs à une certaine valeur seuil on decide de descendre d'un niveau de recursivité pour affiner les facettes résultantes.
Algorithme:

1. ϵ = angle maximal toléré entre 2 normales

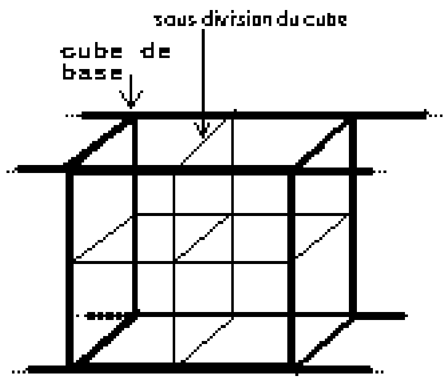

Figure 18: Approche récursive de la division de l'espace

2. construction de la cellule initiale qui sera en fait "collée" à la boite englobante

3. TraiteCellule(cellule initiale)

4. → Procedure TraiteCellule(Cellule)

5. on divise l'espace renfermé par la grille en un ensemble de sous-cellules (cubes) toutes de taille égale

6. pour toutes les sous-cellules

7. si la sous-cellule contient une surface

8. calcul de l'intersection de la surface avec les arêtes de la cellule et génération des facettes

9. si profondeur de recursivité maximale n'est pas atteinte

10. calcul des normales de tous les coins des facettes générées

11. pour toutes les sous-cellules

12. pour toutes les facettes de la sous cellule

13. soit n0, n1, n2 les normales de chaque coin de la facette

14. si (arccos $n_i * n((i+1) modulo3) < \epsilon) pour i \in (0, 1, 2)$

15. conserver la facette

16. sinon

17. TraiteCellule(sous-cellule)

18. fin si

19. fin pour tout

20. fin pour tout

21. fin si

22. → fin TraiteCellule

Le test employé en 7 est aussi utilisé pour l'algorithme de calcul d'intersection de rayon avec la métaforme et est décrit dans cette partie.

Cet algorithme entraîne un problème qu'illustre la figure 19 page 17 : il apparaît une cassure (*crack problem*) lors du maillage adaptatif lorsque une cellule de la grille de taille n est à proximité d'une cellule de taille $t < n$, la cassure est une discontinuité dans le réseau de facette, les facettes ne sont pas jointes et il en resulte des trous dans la forme genérée. Ce probleme n'est pas entièrement résolu en temps interactif.

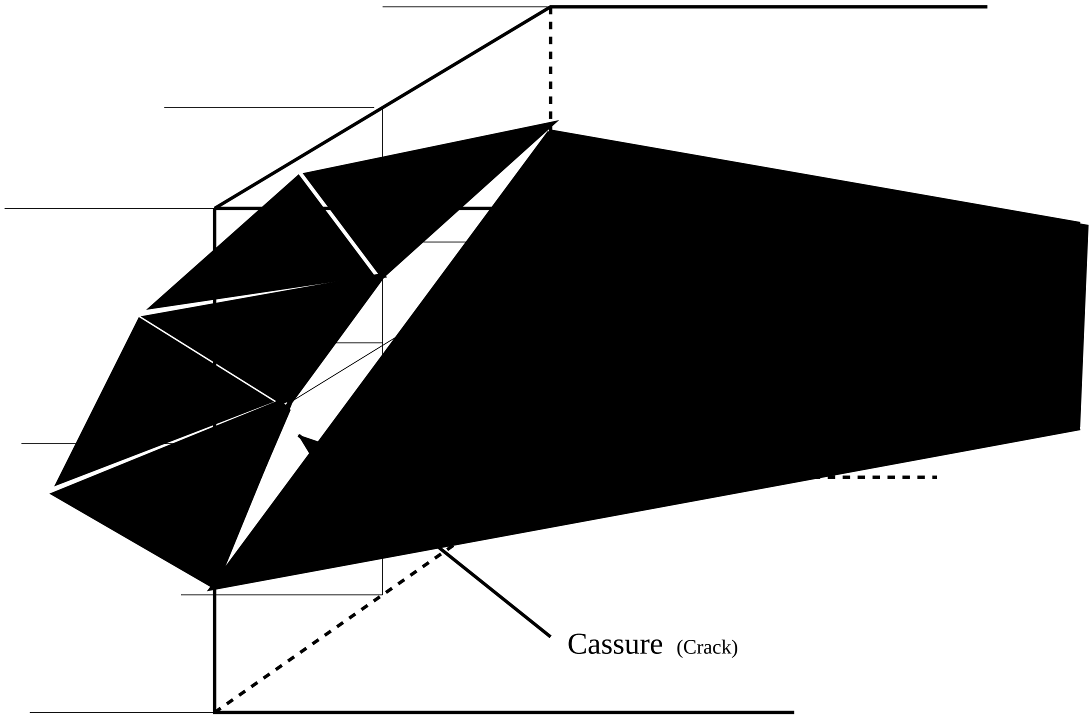

Figure 19: Le *crack problem*

3.2.2 Intérêts

Des cubes de petites tailles sont générés dans les endroits de la métaforme qui turbulent et qui ont donc besoin de plus de détails, pour les zones "calmes" des cubes plus gros suffisent.
Cette méthode présente plusieurs intérêts :

- une bonne rapidité d'éxecution.

- un nombre de facettes réduits, beaucoup aux endroits nécessaires, peu ailleurs.
- un niveau de détail relativement élevé.

4 Le mode de représentation réaliste : algorithme de lancé de rayons sur des surfaces implicites

4.1 Principes du lancé de rayons : le *raytracing*

4.1.1 Définition d'un rayon

Un rayon est défini par son point de départ (en général la caméra), sa direction et une liste d'objets avec lesquels il rentre en collision. : voir la figure 20 page 18.

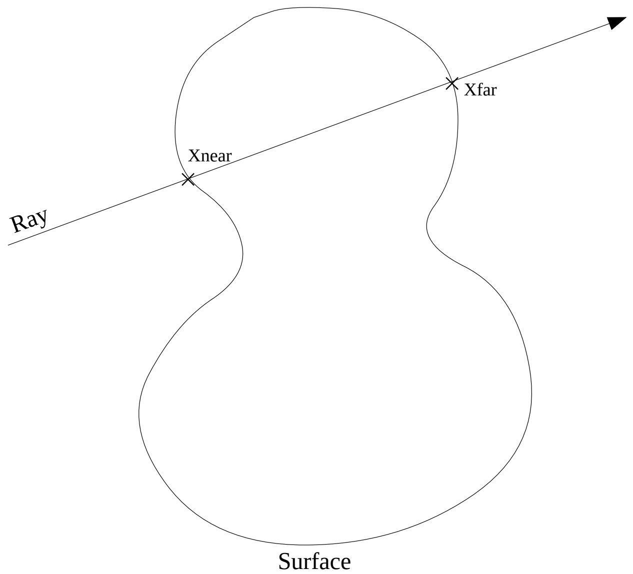

Figure 20: Intersection d'un rayon et de la surface de la métaforme

Cette liste d'intersections avec les objets contient le point d'intersection ainsi que la distance par rapport au point de départ du rayon. Ainsi, on peut trier la liste pour récuperer le premier objet intersecté par le rayon et calculer la couleur du point de collision.

4.2 Lancé de rayons sur les métaformes

Comme nous l'avons vu précédemment, les métaformes sont définies à l'aide d'une formule mathématique d'énergie dissipée en fonction de la distance. Notre but, pour calculer l'image finale, va donc être de trouver l'intersection entre les rayons envoyés de l'oeil vers les métaformes présentent dans l'espace de vision. Le principe utilisé pour trouver l'intersection entre un rayon et la métaforme

se base sur l'analyse numérique des fonctions définissant les émetteurs de la métaforme dans les points de l'espace. Ceci nous permet de pouvoir modeliser très précisemment la surface, dans ses moindres détails, ce qui n'est pas possible avec la représentation par facettes ou par des algorithmes simples d'illumination. L'analyse de la fonction se base sur un bornage de sa variation en certaines parties de l'espace. La valeur **L** sera utilisée plus tard, et correspond à la variation maximale d'énergie dans un cube de l'espace. Une autre valeur **G** sera utilisée plus tard et correspond à la variation maximale du gradiant d'énergie sur un intervalle donné.
La méthode pour trouver l'intersection entre la surface et le rayon se divise en trois parties :

1. Subdivision récursive de la surface pour chaque métaforme, à partir de sa métagrille[2] pour une représentation par octries[3] : voir figure 21 page 19

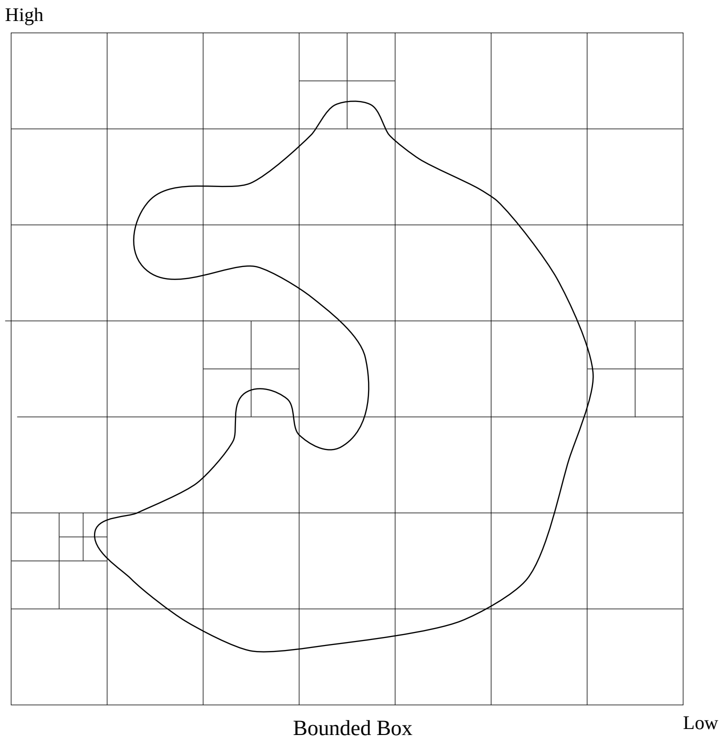

Figure 21: Construction d'une octrie par cube de la métagrille

2. Intersection du rayon avec les octants constituants les octries générées au dessus, puis choix de l'octant dans lequel le rayon va rencontrer la surface : voir figure 22 page 20.

3. Recherche de la première intersection entre le rayon et la surface dans l'octant : voir figure 23 page 21.

[2] *metagrid.*
[3] *octree.*

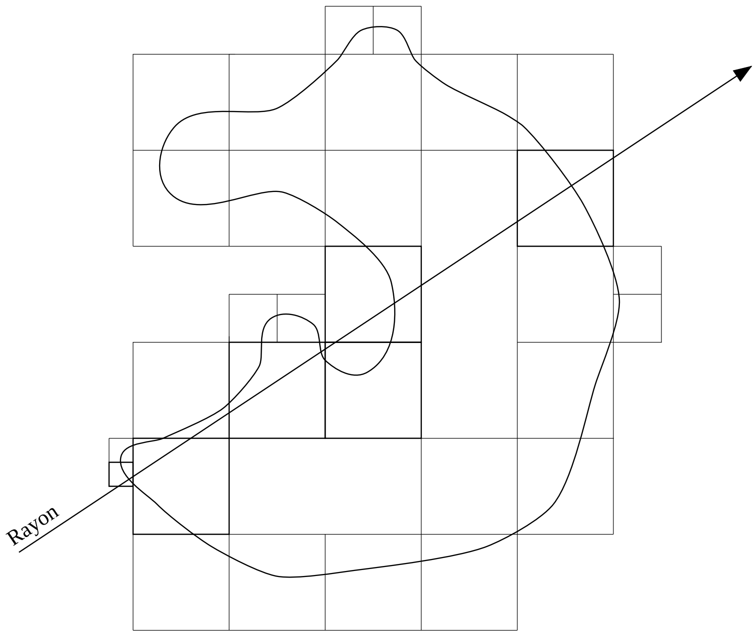

Figure 22: Octrie générée pour la surface et intersection du rayon avec les octants

4.2.1 Construction de l'octrie :

Une octrie est un arbre constitué d'octants, la racine de l'arbre correspond à un cube de la métagrille de la métaforme. Pour chaque cube de la métagrille, nous allons généner l'octrie correspondante à la subdivision de la surface dans ce cube. Chaque noeud de l'arbre peut être le père de huit fils, ces fils étant la division du cube du noeud en huit sous-cubes de résolution plus fine.
La construction de l'octrie nous permet d'optimiser les calculs pour l'intersection du rayon avec la surface puisqu'on est certain que la surface passe dans chaque octant de l'arbre (les feuilles de l'arbre en fait).
Pour trouver la subdivision de la surface en octant, il nous faut effectuer certains tests, voici l'algorithme :
Soit b le rayon d'un cube (c'est-à-dire la longueur de sa demi diagonale), et C son centre.

- Si $|f(C) - Treshold| > b * L$ alors il ne peut pas y avoir d'intersection car le cube est entièrement dans la surface ou à l'éxterieur.
- Si l'octant est d'une résolution assez fine, et que le test des valeurs énergétiques aux huit coins du cube nous montre que la surface passe forcément dans le cube, alors l'octant est rangé dans l'arbre sous forme de feuille, et les valeurs de L et G sont sauvegardées. Ce test permet l'arrêt de la récursivité dans la construction de l'arbre.

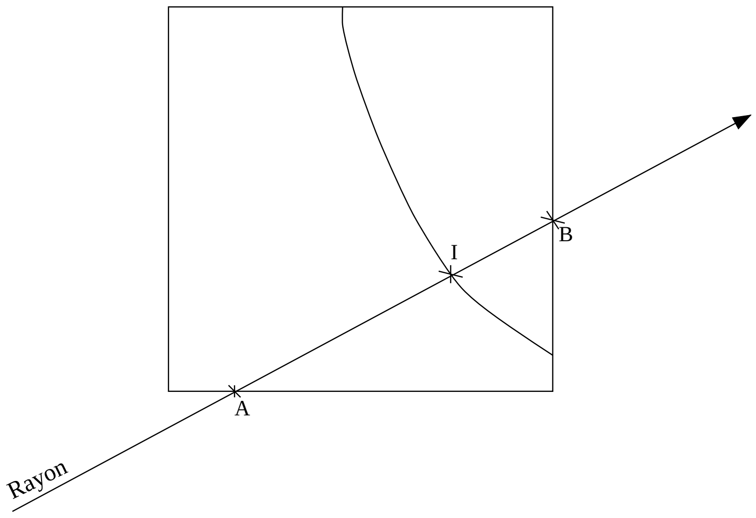

Figure 23: Intersection des rayons avec la métaforme dans l'octant.

- Si aucuns des critères ci-dessus n'ont été choisis alors le cube est divisé en huit sous cubes, et une exploration est lancée récursivement dans chacun des nouveaux cubes.

Le premier test utilise les propriétés des fonctions avec la constante de Lipschitz, pour borner la fonction et voir sa variation de valeurs dans un intervalle.

4.2.2 Chercher les octants intersectés par le rayon puis trouver le premier dans lequel le rayon va toucher la surface

Il suffit tout simplement dans cette partie là, de regarder, pour chaque octrie de chaque cube de la métagrille, si le rayon intersecte le cube ou pas, puis d'inserer dans une liste triée la position de l'intersection et la distance d'intersection sur le rayon.
Pour trouver ensuite dans quel octant de la liste le rayon va croiser la surface de la métaforme, il faut appliquer une serie de tests sur les octants dans l'ordre d'apparition sur le chemin du rayon.
Voici l'algorithme utilisé :

- Calculer la valeur de G dans la boîte correspondant à notre cellule d'octrie, puis regarder le résultat de $|g(tm)| > G$. La fonction g étant la fonction derivée de f dans la direction du rayon, et tm étant la position du centre du segment AB créé par le rayon dans la boîte. A et B sont les points d'intersection du rayon avec la boîte.

- Si c'est vrai alors on regarde les signes de $|f(A) - Treshold|$ et $|f(B) - Treshold|$ pour voir si d'une extrémité à l'autre on est dans la surface ou pas. Si les signes sont differents alors il n'y a qu'une seule intersection dans cette cellule entre le rayon et la surface, et on peut entammer un procédure de recherche dichotomique de la solution dans le cube.

- Si les signes sont égaux alors il ne peut pas y avoir d'intersection dans la cellule et dans ce cas la, nous passons à la suivante.
- Sinon, c'est que le premier test a échoué et dans ce cas, il peut y avoir plus d'une intersection entre le rayon et la surface. A ce moment là, on subdivise l'intervalle d'intersection en deux, puis on recommence récursivement les tests sur les deux nouveaux intervalles.

Cette méthode, optimisée grâce à l'utilisation d'un octrie pour approximer la surface, nous permet d'avoir un rendu très réaliste des objets de type métaforme ou plus généralement de type surfaces implicites.

5 Techniques pour l'Animation

Nous avons enrichi les objets de notre logiciel (métaformes, émetteurs, caméras) pour qu'ils puissent être animés. Pour cela nous utilisons une technique de clés et d'interpolation. A chaque objet du "monde" est associée une liste de clés (voir la section 1.8 page 33 qui explique comment créer une clé).

5.1 Les clés

Les clés sont des objets contenant un numero d'image et la description de l'objet pour cette image. La liste contient au minimum la clé référençant l'image 1, cette clé contient les attributs de l'objet au moment où on lance le mode d'animation. La clé décrit en fait un changement d'état de l'objet (le changement d'état peut être l'identité).

5.2 Interpolation

On pourra aisément trouver l'etat d'un objet entre deux clés à l'aide d'interpolations sur les attributs, ainsi nous disposons de tous les états intermédiaires de chaque objet entre les clés. Les interpolations sont pour l'instant linéaires mais on peut étendre pour générer toutes sortes d'effets animés comme les accélérations par exemple.

5.3 Animation

Pour toutes les images de l'animation, on demande à chaque objet de se mettre dans l'état logique par rapport au numéro de l'image souhaitée puis on demande la construction de l'image. Les objets calculent leur état pour cette image à l'aide de leur liste de clés.

6 Entrées / sorties

6.1 Les formats internes

Il est nécessaire de disposer d'un moyen qui permette de sauvegarder le travail effectué avec xblob. Supposons que l'on ait modélisé des métaformes, on peut souhaiter pouvoir les réutiliser plus tard d'où la nécessité de sauvegarder (et donc de charger ensuite) des métaformes. Cela est prévu dans Xblob. Pour cela nous disposons de 2 formats de fichiers propres à Xblob. Ce sont les formats RAW/MFA et BMF (Binary Metaform Format).

6.1.1 Le format script

Le premier format est un format ASCII qui décrit la scène (les objets s'y trouvant, leurs caractéristiques etc). Le principal avantage de ce format est qu'il permet à l'utilisateur de créer lui même un fichier de ce type. Il peut donc construire très facilement une scène complexe composée de plusieurs métaformes à l'aide d'un fichier script suivant la grammaire définie pour ce format. Cette grammaire est décrite en détails dans la section ?? page ??.
Voici un petit exemple de script dans ce format :

```
Scene 1 Description {
   Metaform 1
   {
      Treshold 0.8
      Source Point
      {
         Center 0 1 2
         Power 2
         Energy_F 1
         }
      }
      Source Segment
      {
         Start 0 6 4
         End 4 3 2
         Power 1
         Energy_F 1
         }
      }
   }
}
```

Ce script a pour effet de créer une nouvelle métaforme qui est composée de 2 émetteurs. Le premier de type point est placé aux coordonnées (0,1,2) du monde avec une puissance de 3 et un autre émetteur de type segment, dont le premier point est placé en (0,6,4), le second en (4,3,2) avec une puissance de 1. Ces 2 émetteurs ayant comme fonction d'énergie, la fonction prédéfinie numéro 1.

Le format interne d'animation Ce format permet également de définir des animations pour xblob. Le principe en est simple. En effet, il suffit de définir la scène aux différentes étapes de l'animation de manière exactement identique à celle décrite précédemment.
En voici un exemple :

```
Animation
{
   Moves 5
   Scene 1 Description
   {
      Metaform 1
      {
         Treshold 0.8
         Source Point
         {
            Center 0 1 2
            Power 3
            Energy_F 1
         }
      }
   }
   Scene 2 Description
   {
      Metaform 1
      {
         Treshold 0.6
         Source Point
         {
            Center 1 3 4
            Power 3
            Energy_F 1
         }
      }
   }
}
```

Ce script définit une animation en 5 mouvements avec au départ une métaforme d'iso valeur 0.8 composée d'un émetteur point placé en (0,1,2), à une métaforme d'iso valeur 0.6 composée d'un émetteur point placé en (1,3,4).

Grammaire du format RAW Description des différents types d'émetteurs et leurs caractéristiques:

- Point
  - Center : coordonnées du point
  - Radius : énergie émise par ce point
- Segment
  - Start : coordonnées du premier point
  - End : coordonnées du second point
- Triangle
  - P0 : coordonnées pour le premier point

  - P1 : coordonnées pour le deuxième point
  - P2 : coordonnées pour le troisième point

- Ligne brisée
  - P : coordonnées du premier point
  - P : coordonnées du deuxième point
  - P : coordonnées pour le troisième point
  - ...
  - P : coordonnées pour l'avant dernier point
  - P : coordonnées pour le dernier point

- Réseau de facettes (Mesh)
  - P0 : coordonnées du premier point (du 1er triangle)
  - P1 : coordonnées du deuxième point (du 1er triangle)
  - P2 : coordonnées du troisième point (du 1er triangle)
  - P0 : coordonnées du premier point (du 2ème triangle)
  - P1 : coordonnées du deuxième point (du 2ème triangle)
  - P2 : coordonnées du troisième point (du 2ème triangle)
  - ...
  - P0 : coordonnées du premier point (du nième triangle)
  - P1 : coordonnées du deuxième point (du nième triangle)
  - P2 : coordonnées du troisième point (du nième triangle)

Le *Treshold* décrit la valeur seuil (iso valeur) de la métaforme.

Format d'une scène (Grammaire pour le format RAW) :

```
SCENE :
            'Scene' int 'Description' '{' MF_In_Scene '}'
            ;

MF_In_Scene :
            Metaform
            MF_In_Scene Metaform
            ;

Metaform :
            'Metaform' int '{' 'Treshold' double Emittors_In_MF '}'
            ;

Emittors_In_MF :
            'Source' MF_Emit_Definition
            |Emittors_In_MF 'Source' MF_Emit_Definition
            ;
```

```
MF_Emit_Definition :
            Point_Emit_Definition
            |Segment_Emit_Definition
            |Triangle_Emit_Definition
            |Line_Emit_Definition
            |Mesh_Emit_Definition
            ;

Point_Emit_Definition :
            'Point' '{' Center Power Energy_F '}'
            ;

Segment_Emit_Definition :
            'Segment' '{' Start End Power Energy_F '}'
            ;

Triangle_Emit_Definition :
            'Triangle' '{' P0 P1 P2 Power Energy_F '}'
            ;

Line_Emit_Definition:
            'Line' '{' P P Maybe_More_Points Power Energy_F '}'
            ;

Mesh_Emit_Definition :
            'Mesh' '{' P0 P1 P2 Maybe_More_Triangles Power Energy_F '}'
            ;

Power :
            'Power' double
            ;

Center :
            'Center' Point3D
            ;

Start :
            'Start' Point3D
            ;

End :
            'End' Point3D
            ;

P :
            'P' Point3D
            ;

P0 :
            'P0' Point3D
```

```
            ;

P1 :
            'P1' Point3D
            ;

P2 :
            'P2' Point3D
            ;

Energy_F :
            'Energy_F' int
            |'User_Energy_Func' Energy_Points
            'Deriv_Energy_Func' Energy_Points
            'SDeriv_Energy_Func' Energy_Points
            ;

Maybe_More_Points:  * rien *
            | Maybe_More_Points P
            ;

Maybe_More_Triangles:  /* rien */

            |Maybe_More_Triangles P0 P1 P2
            ;

Point3D :
            double double double
            ;

Energy_Points :
            double
            |Energy_Points double
            ;

Grammaire pour le format RAW anime MAF

ANIMATION :
            'Animation' '{' 'Moves' int One_Or_More_Scenes '}'
            ;

One_Or_More_Scenes :
            SCENE
            |One_Or_More_Scenes SCENE
            ;
```

6.1.2 Le format binaire

Le deuxième format propre à xblob est un format binaire. Il est donc plus compact. Cela permet à plusieurs programmes xblob tournant en même temps d'échanger des informations et de modéliser simultanement une même scène sur 2 machines (Voir la section 3 page 35 modélisation en réseau).

6.2 Xblob communique avec le reste du monde : importation / exportation

Il y a de plus en plus de logiciels qui gèrent des objets en 3D et il est donc important que nous puissions échanger des objets avec ces différents logiciels. En effet, xblob est un logiciel de création de métaformes. Il n'est donc pas adapté à la création d'objets d'autres types (comme les cubes, tore etc).
Pour cette raison, nous l'avons doté de possibilités d'importation et d'exportation dans différents formats. Ainsi, nous pouvons utiliser des objets très complexes provenant d'autres programmes et les transformer en métaformes. A l'inverse, nous pouvons aussi donner à nos métaformes une représentation compréhensible par d'autres logiciels : sous forme d'un émetteur de type réseau de facettes. Ainsi, bien que les logiciels ne puissent pas modéliser de métaformes, ils pourront les visualiser.

Comme il n'existe pas de logiciels permettant de modéliser des métaformes, on utilise le principe suivant : la métaforme, une fois calculée est représentée par un flot de facettes, c'est donc cette représentation que nous allons convertir dans le format voulu. En effet, tous les logiciels de 3D offrent cette possibilité, c'est donc le moyen universel de convertir des objets hétéroclytes.
Xblob offre la possibilité d'exporter et d'importer les scènes au format 3DS (pour le logiciel 3D Studio). C'est un format binaire. 3D Studio est actuellement l'un des logiciel de 3D les plus utilisés.
Xblob permet aussi de lire des objets au format OBJ ainsi que de sauvegarder les métaformes dans ce format. Le format OBJ est un format ASCII très simple. Il se contente de spécifier le nombre de facettes dont l'objet dispose ainsi que les coordonées des différents sommets.
Le troisième format que l'on peut importer et dans lequel on peut exporter est VRML (Virtual Reality Modeling Language). C'est un format très à la mode, et qui se développe beaucoup actuellement. VRML est le résultat récent d'un essai pour définir un langage pour le Web, multi plate-formes de réalité virtuelle.

Partie III
Mode d'emploi, utilisation du logiciel Xblob

1 L'interface

L'interface est composée de trois parties visibles principales permettant d'accéder aux fonctionnalités du logiciel. Une fenêtre principale avec une barre de menu en haut et une zone de visualisation du monde, et une palette flottante d'options utiles au modelage des objets.

1.1 La barre de menu

La barre de menu contient tous les éléments de gestion du logiciel et d'options.
Le menu **File**[1] sert à la gestion des entrées/sorties sur fichier des scènes (voir section 2 35 pour des informations sur les formats utilisés et reconnus).
Le menu **View**[2] sert pour le choix entre les modes de vue utilisateurs prédéfinis et la vue caméra. Pour le mode de vue utilisateur, le logiciel offre le choix entre plusieurs vues planes comme la vue de face, de côté, ou de profil (voir section 1.4.1 page 30).

Le menu **Animation** permet de rentrer en mode de programmation d'animation. Ce sujet est décrit en detail dans la section 1.8 page 33.
Le menu **Network**[3] permet de passer en mode de modelage réseau avec plusieurs ordinateurs reliés. Ce sujet est traité dans la section 3 page 35.
Enfin, le menu **Help**[4] donne la boîte de dialogue de version du logiciel.

1.2 La fenêtre de visualisation

La fenêtre de visualisation sous la barre de menu est notre vision sur le monde de Xblob. C'est dans cette fenêtre que l'utilisateur va créer sa scène et placer ses objets. Sous cette fenêtre se trouve une série de trois boutons permettant de choisir un mode de déplacement ou de rotation des objets. Une ou deux directions peuvent être choisies simultanément étant donné que le monde de Xblob est en trois dimensions et que la souris ne nous permet facilement que de faire des mouvements dans deux dimensions en même temps.
La zone en dessous des boutons sert à afficher les messages d'aide en ligne du logiciel.

1.3 La *tool blob*

La *tool blob* est la palette volante d'outils permettant à l'utilisateur de modeler sa scène. Elle est divisée en trois parties :

[1] Fichier
[2] Vues
[3] Réseau
[4] Aide

1.3.1 Choix d'un émetteur

Le premier bouton sert à choisir le type de l'émetteur, parmi les cinq existants, que l'on veut créer dans la scène.

1.3.2 Choix d'un mode de rendu

Le deuxième bouton permet à l'utilisateur de choisir un mode de rendu, soit pour visualiser sa scène dans un mode de rendu réaliste (*gouraud, phong, raytracing*), soit pour le modelage plus ou moins rapide des objets dans la scène. Le premier mode *nothing* n'affiche pas la génération de la forme facettée et est par conséquent le plus rapide des modes de représentation.
Le mode *flat*, quant à lui, permet de voir la position des sources de lumière par rapport aux objets.

1.3.3 Choix de l'action à effectuer

Enfin, dans la partie basse de la boîte à outils, l'utilisateur va pouvoir choisir le type d'action qu'il désire effectuer dans la scène.
Trois types d'actions existent : création (voir section 1.5 page 31, édition (voir section 1.7 page 32), sélection (voir section 1.6 page 32). Le choix des objets sur lesquels se portent les actions édition et séléction est choisi par l'appui sur un des types d'objets listé au dessus du choix de l'action.
De plus, d'autres actions sont possibles à travers l'utilisation du troisième bouton de la souris dans la fenêtre de visualisation. L'appui sur ce bouton ouvre un menu dans lequel l'utilisateur aura le choix, suivant le type d'objet choisi, de le détruire ou bien d'éditer ses propriétés.

1.4 Composition d'une scène

1.4.1 Les différentes vues et la caméra

La vision du monde dans Xblob passe par la caméra, comme un cinéaste pour filmer ses scènes. En plus de la caméra, qui permet de visualiser la scène sous n'importe quel angle, Xblob est pourvu de vues utilisateurs planes prédéfinies qui offrent une aisance supplémentaire dans le modelage des objets dans la scène. Pour se déplacer dans le monde et établir une vue, l'utilisateur à la possibilité, soit d'agrandir ou rétrécir son champ de vision[5] (ce qui correspond à un zoom), soit d'effectuer une rotation de la caméra virtuelle[6] autour du centre du monde. La visualisation de l'image s'effectuera toujours à partir de la vue courante.

1.4.2 Les objets

Les objets pouvant être créés dans la scène sont des métaformes, constituées d'émetteurs. Au démarrage du logiciel, la scène est vide, et la caméra fixe le centre du monde.
Une métaforme est donc constituée d'émetteurs, de types variés.

[5] touche Shift enfoncé + bouton du milieu de la souris pressé + mouvement vertical de la souris.

[6] touche Shift enfoncé + mouvement de la souris.

Attributs d'une métaforme : Elle a comme attribut un seuil énergétique sur lequel on va placer sa peau. En fait, ce seuil représente la valeur dénergie dans l'espace où la surface de la forme sera dessinée.

Attributs des émetteurs : Un émetteur a comme attributs sa fonction d'énergie, donc la fonction de la distance qui donne la valeur de l'énergie dégagée par cet émetteur en un point P de l'espace. Si cette énergie est inférieure au seuil de la métaforme dans laquelle se trouve l'émetteur, alors le point est en dehors de la métaforme, dans le cas contraire, il est dans la métaforme. Un autre attribut de l'émetteur est sa puissance d'émission, plus celle-ci est importante, plus l'émetteur aura d'influence sur l'espace autour de lui.

1.5 La création d'objets dans la scène

La création d'objets dans la scène se fait tout d'abord par le choix du mode de création pour l'action dans la boîte à outils flottante, puis le choix d'un type d'émetteur en haut de la palette d'outils.
La création de l'émetteur se fait par un click sur le bouton gauche de la souris dans la fenêtre de visualisation, à l'endroit où l'on veut voir l'émetteur apparaître.

1.5.1 Emetteur point :

Pour l'émetteur point, un simple click suffit.

1.5.2 Emetteur segment :

Pour l'émetteur segment, le premier click place le point de départ du segment, et un second click le point de fin.

1.5.3 Emetteur triangle :

Pour l'émetteur triangle, le premier click place le premier sommet du triangle, puis les deux suivant placent les deux autres sommets du triangle.

1.5.4 Emetteurs lignes brisées et réseau de facettes :

Pour les émetteurs ligne et reseaux de facettes, le premier click commence la création de l'émetteur puis chaque click suivant rajoute un point à l'émetteur. Pour terminer la création, il suffit d'appuyer sur le bouton du milieu de la souris.

Si la scène est vide de tout objet, alors la création du premier émetteur entraînera automatiquement la création d'une métaforme pour le contenir.
Tout objet créé est d'office sélectionné (voir la section 1.6 page 32), pour rajouter un émetteur à une métaforme, il faut que celle-ci soit séléctionnée. Si aucune métaforme n'est séléctionnée dans la scène lors de la création d'un nouvel émetteur, une nouvelle métaforme est créée pour cet émetteur.
Les émetteurs dans une même métaforme rentrent en interaction, alors que des émetteurs dans deux métaforme distincts n'interagissent pas du tout ensemble.

1.6 La sélection des objets

Pour sélectionner des objets, il faut d'abord rentrer dans le mode selection d'objets correspondant à l'appui sur le bouton d'action "sélection". Ensuite, on choisi le type d'objet que l'on veut sélectionner en positionnant les boutons de choix soit sur "metaform", soit sur "emittor", soit sur "point" (point(s) constituant un émetteur).
Ensuite, pour le sélection des objets proprement dite, il suffit d'aller sur la fenêtre de sélection et d'appuyer soit sur le bouton du milieu de la souris pour sélectionner le type d'objet le plus proche de la souris, soit sur le bouton gauche en restant appuyer pour définir une zone de sélection des objets. Tout objet inclus entièrement dans cette zone se trouvera selectionné. Par défaut, la sélection d'objet entraîne la déselection de tous les objets précedemment selectionnés. Pour empêcher cela, il suffit de garder enfoncé la touche *ctrl* du clavier (sélection multiple).
Un bouton dans la boîte à outils permet de désélectionner tous les objets.
Les objets selectionnés sont dessinés d'une autre couleur dans la fenêtre de visualisation.

1.7 L'édition des objets et la modification de leurs attributs

1.7.1 Edition

Pour pouvoir éditer des objets dans le monde, c'est à dire pouvoir modifier leur positionnement dans l'espace, il faut se placer dans le mode d'édition dans la palette d'outils. Ensuite, on choisi le type d'objet que l'on désire modifier, de la même manière que pour la sélection, à un detail près, c'est que pour ce mode là, on peut effectuer les modification sur l'ensemble de la sélection plutôt que sur un type particulier d'émetteur.
Une fois le type d'objet selectionné pour la modification spatiale, le bouton gauche de la souris permettra de faire bouger l'objet dans le monde relativement à l'orientation des axes du monde.
Les trois boutons en dessous de la fenêtre de visualisation voient ici leur interêt puisqu'il permettent de choisir les axes de déplacement des objets, avec au maximum deux directions simultanémment.

1.7.2 Modification des attributs

Pour modifier les attributs d'une métaforme ou d'un émetteur, il faut cliquer avec le bouton droit dans la fenêtre de visualisation près de l'objet en question (c'est l'objet le plus proche du "click" souris qui est sélectionné). Si on veut modifier les attributs d'un ensemble d'objets dejà sélectionné (mode sélection courante), il suffit de cliquer avec le bouton droit n'importe où dans la fenêtre de visualisation.
Le "click" souris entraîne l'apparition d'une fenêtre qui permet de choisir l'attribut que l'on désire modifier.
Une fenêtre apparaît alors permettant de changer :

- La puissance d'un émetteur : entrer la valeur dans la case de saisie "Power", puis appuyer sur le bouton "Change". La puissance de l'émetteur sélectionné est modifiée, on voit la métaforme changer de forme.

- Le seuil énergétique d'une métaforme (celle sélectionnée ou celle contenant l'émetteur actuellement sélectionné) : entrer la valeur dans la case de saisie "Treshold", puis appuyer sur le bouton "Change". La métaforme change alors de forme.

- La fonction d'énergie d'un émetteur : appuyer sur le bouton "Change" de la partie "Energie function" de la fenêtre d'attributs, une nouvelle fenêtre apparaît alors, où l'on voit dans la partie supérieure la courbe de la fonction d'énergie actuellement associée à l'émetteur. Comme nous l'avons vu dans la section 2 page 10, le logiciel permet de choisir entre des fonctions d'énergies prédefinies et des fonctions que l'on peut soit même définir.

  – Pour choisir une fonction prédefinie, enfoncer le bouton de choix "Predefined functions", puis choisir dans le menu "Predefined" la fonction désirée. Sa courbe apparaît alors dans la partie supérieure de la fenêtre, et la métaforme change de forme prenant en compte la nouvelle fonction d'énergie pour le ou les émetteurs sélectionnés.
  – Procéder de la même manière pour choisir une fonction utilisateur (menu "User defined") dejà définie.
  – Pour définir une nouvelle fonction d'énergie, appuyer sur le bouton "Define a new function" puis dessiner la courbe de la fonction (le logiciel empêche, comme nous l'avons vu dans la section 2 page 10, de définir des fonctions non valides : c'est à dire non monotones décroissantes), ensuite, lorsque la courbe est satisfaisante, appuyer sur le bouton "Record" pour enregistrer la nouvelle fonction, le logiciel demande alors à l'utilisateur de donner un non à celle ci. pour enregistrer la

Remarque :
En ce qui concerne les modifications de la puissance et de la fonction d'énergie, les émetteurs modifiés dépendent du mode de selection :

- Pour le mode de sélection émetteur, l'émetteur modifié est celui qui est sélectionné.

- Pour le mode de sélection métaforme, tous les émetteurs de la métaforme sélectionnée sont modifiés.

- Pour le mode sélection courante, tous les émetteurs sélectionnés ainsi que tous les émetteurs de toutes les métaformes sélectionnées sont modifiées.

1.8 L'animation des éléments dans la scène

Notre logiciel offre la possiblité de créer des scènes animées. Le principe de l'animation se base sur une technique d'animation par cles et d'interpolation (voir section 5 page 22).

1.8.1 Exemple

On veut créer une animation de 50 images d'une métaforme grossissante et se deplaçant pendant 25 images vers le haut puis 25 images vers le bas. Pour cela, il faut se placer à l'image 25 et positionner la métaforme au dessus de sa position initiale, puis il faut aller à l'image 50, la positionner plus bas et mettre à ses émetteurs une puissance plus grande pour produire l'effet de grossissement. Ceci peut être réalisé de manière très simple, en effet il suffit de suivre les procédures ci-dessous. La figure 24 page 35 illustre cet exemple.

1. Se mettre en mode animation en cliquant dans le menu "Animation" sur "Animate". Une fenêtre apparaîtra contenant des outils de contrôle pour l'animation.

2. Le nombre d'images pour l'animation doit être mis à la valeur souhaitée (50 pour notre exemple) en cliquant sur le bouton change à côté de l'indicateur de nombre d'image.

3. Se placer sur l'image que l'on souhaite modifier (ici la 25) à l'aide de l'ascenceur horizontal contenu dans la fenêtre de contrôle d'animation.

4. Changer les valeurs des attributs à l'aide de l'interface, ici translater la métaforme vers le haut.

5. Enregistrer la modification (création d'une clé) avec le bouton "record" dans la fenêtre de contrôle d'animation.

6. Cliquer sur Animate pour lancer l'animation.

Sur notre exemple il faudrait encore, avant de cliquer sur Animate:

6. Se placer sur l'image 50 à l'aide de l'ascenceur horizontal contenu dans la fenetre de contrôle d'animation. Translater vers le bas et augmenter la puissance des émetteurs de la métaforme.

7. enregistrer la modification avec le bouton record dans la fenêtre de contrôle d'animation.

8. Cliquer sur Animate pour lancer l'animation.

1.8.2 Remarques

- Si on se positionne à l'aide de l'ascenceur horizontal sur une image qui n'a pas de clé associée et qu'elle se situe entre 2 cles, l'image vue sera l'image interpolée par rapport aux cles encadrantes (interpolation de tous les attributs).

- Si on se positionne, à l'aide de l'ascenceur horizontal, sur une image qui n'a pas de clé associée, et qu'il n'existe pas de clé référencant une image plus "lointaine" (de numéro supérieur à l'image courante) alors l'image vue sera l'image referencée par la clé la plus proche dans le passé (forcément!).

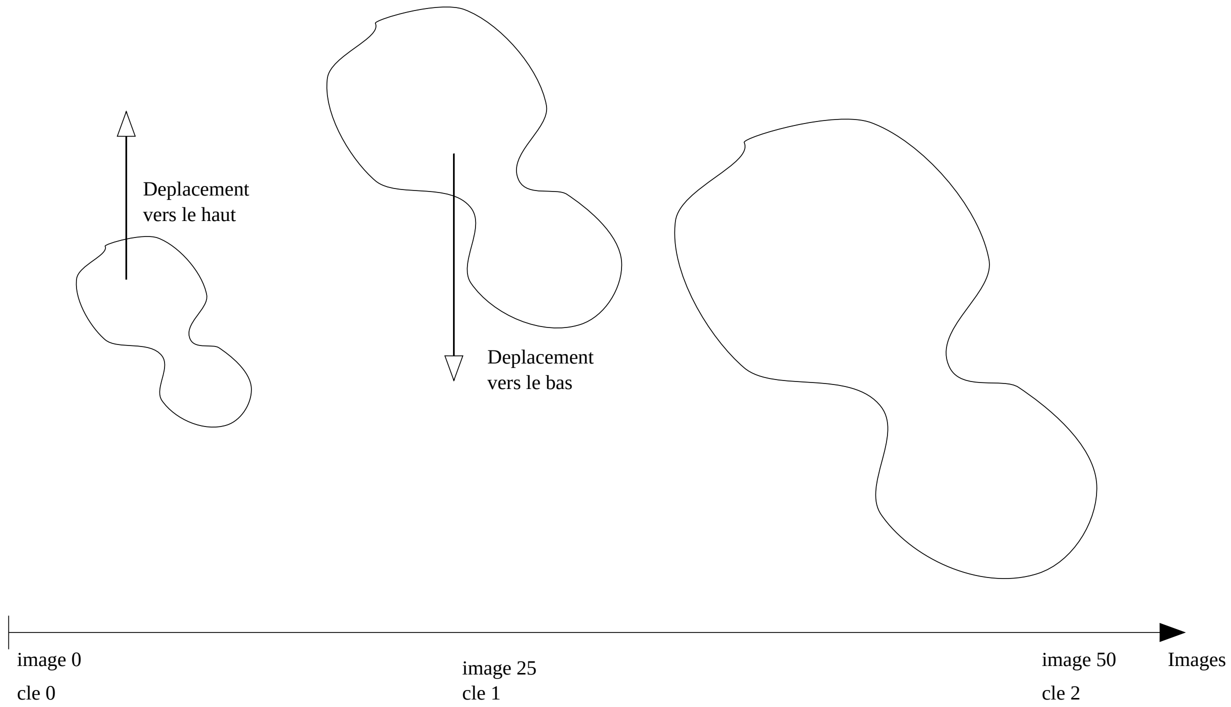

Figure 24: Illustration d'une animation

2 Importation et exportation, formats

La barre de menu de Xblob permet d'accéder aux fonctionnalités d'entrées/sorties. Les boutons "Load" et "Save" du menu "File" permettent de charger et sauvegarder au format interne de Xblob (Format RAW). Il suffit simplement de donner le nom du fichier lorque le sélecteur de fichier apparaît.
Il existe aussi les sous menus "Import" et "Export" qui permettent de sauvegarder la scène dans un format reconnu par d'autres logiciels. Il s'agit des formats 3DS, VRML et OBJ. Comme précédemment, il suffit de donner le nom du fichier que l'on veut charger/sauvegarder.

Remarque : il est recommandé de donner une extension correspondant au format du fichier. L'extension ".3DS" pour les fichiers de 3D Studio, ".WRL" pour les fichiers VRML et ".OBJ" pour les fichiers au format OBJ. Cela permet d'identifier le fichier immédiatement et de plus, lorque l'on invoque xblob avec l'option -import, il ne tient compte que de l'extension du fichier.

3 Modélisation d'une scène à plusieurs : la couche réseau

Nous avons rajouté une couche réseau sur notre logiciel Xblob. Elle permet une centralisation de la scène de travail.
Chaque instance du programme Xblob peut être ou client ou serveur. Le ser-

veur et tous ces clients vont pouvoir éditer la même scène en même temps, qui évoluera en parallele sur toutes les machines.
Utilisation :

- Serveur :
  Il faut qu'il y ait un serveur qui tourne sur une machine X. Pour mettre une instance de Xblob en mode serveur, il suffit d'appuyer sur le bouton “server” du menu “Network” de la barre de menu. Ensuite la scène éditee devient visible et éditable par tous les clients qui se connectent. C'est le serveur qui centralise et traite les demandes de modifications de la scène provenant des clients. Le serveur prévient tous les clients de la prise en compte d'une modification dont lui, ou un des clients est à l'origine.

- Client :
  Pour se connecter à un serveur, il suffit de cliquer sur le bouton “client” de la barre de menu. Une boite de dialogue apparaîtra et le nom de la machine hôte du serveur sera demandé. Ceci fait, le client se connecte au serveur et reçoit une message de bienvenue et la scène de travail dans l'état exact dans laquelle elle se trouve chez le serveur.

Remarque : seule la scène est commune, chaque instance de Xblob a ses propres caméras et peut se deplacer de manière indépendante dans la scène. Chaque instance a son propre mode de rendu (par exemple le serveur verra la scene en fil de fer alors qu'un de ses clients pourra visualiser cette scène en gouraud (ou tout autre mode)).

4 Installation du logiciel xblob

4.1 Installation

Le logiciel xblob est simple à installer. Il se présente sous la forme compressée d'une archive au format Tar. Sous UNIX, il suffit simplement de taper la commande suivante: 'gzip -d xblob.tgz — tar -xvf' Cela a pour conséquences de vous créer un répertoire dans lequel se trouvent toutes les sources du logiciel ainsi que le man et des fichiers d'informations. Il faut ensuite créer l'éxécutable en compilant ces sources, ceci grâce à la commande 'make'. Le fichier Meta-Form_Modeler des ressources Motif doit être copié (ou bien faire un lien) dans le répertoire $XAPPLRESDIR (souvent /.app-defaults). Une fois la compilation terminée, le programme xblob est créé et il est prêt à être utilisé.

4.2 Lancement du logiciel

On peut lancer xblob tout simplement en lançant la commande 'xblob'.
Xblob peut prendre plusieurs paramètres sur la ligne de commande. Voici les différentes options possibles : (les options suivies de OPTn signifie que ces options requièrent n arguments).

- -load file
  où file peut être soit du type RAW, soit du BMF ou encore du MAF (Metaform Animation Format).

- -import file
  où file est soit du type 3DS, soit VRML, soit OBJ
- -render file
  où file est le nom du fichier dans lequel sera sauvegardé le résultat
- -info
  pour avoir des informations sur xblob
- -version

Explication des différentes options Les options -load et -import permettent de charger dès le lancement de xblob un fichier. Ainsi la scène sera déjà composée d'une ou plusieurs métaformes suivant le contenu du fichier lu.
L'option 'render' doit impérativement être la première option de la ligne de commande. Celle-ci permet de ne pas lancer l'interface de xblob mais simplement de lire un fichier (option load ou import) et de calculer le rendu en raytracing. Le résulat sera stocké dans le fichier passé en paramètre pour cette option.
Par défaut xblob lancera son interface et permettra ainsi à l'utilisateur de modéliser à sa guise.

Partie IV
Conclusion

Nous avons beaucoup apprecié la réalisation de ce projet, il nous a permis d'appliquer les connaissances que nous avons aquises au cours de notre formation, de les confronter à un problème de grandeur réelle et ainsi de les consolider.
Il nous a aussi permis de nous organiser pour travailler tous en même temps sur l'élaboration d'un logiciel, de nous donner des méthodes de travail en groupe qui nous permettront plus tard de nous insérer plus facilement dans des équipes de travail.
Le fait de travailler sur des formes qui sont encore du domaine de la recherche et de les enrichir était très intéressant.
La réalisation de ce projet nous a permis de nous familiariser avec le monde de l'image de synthèse, et nous avons pu constater qu'il reste de nombreuses choses à réaliser, à trouver, à inventer dans ce domaine.
Nous aurions aimé avoir le temps d'en faire plus, notre désir est maintenant de continuer à développer notre logiciel pendant notre temps libre afin de l'enrichir et de l'améliorer.

Index

Références

[Bei] T. Beier. Practical uses for implicit surfaces in animation.

[Bli82] J. F. Blinn. A generalisation of algebraic surface drawing. *ACM Trans. Graphics*, 1982.

[Gra93] G. Graves. *The Magic of Metaballs*. Computer Graphics World, 1993.

[PT] B. A. Payne and A .W. Toga. Distance field manipulation of surface models.

[PW94] B. A. Payne and B. Wyvill. *Visual Simulation of Lightning*. Computer Graphics, 1994.

[RS] Mohan S. Kankanhalli Renben Shu, Chen Zhou. Adaptative marching cubes.

[RW90] T. Reed and B. Wyvill. A new method for 3d reconstruction from serial sections by computer graphics using "metaballs"'. *Proc. of The Australian Conference on Computer Graphics*, 1990.

[Wyva] B. Wyvill. The great train rubbery.

[Wyvb] B. Wyvill. Soft.

Rapport du projet de maîtrise année 1996/1997.
Nom du groupe: “les blobs”

Membres:

- Sébastien Schneider
  sschneid@clio.unice.fr
  http://www-mips.unice.fr/~sschneid

- Julien Hage
  jhage@clio.unice.fr
  http://www-mips.unice.fr/~jhage

- Olivier Nobili
  onobili@clio.unice.fr
  http://www-mips.unice.fr/~onobili

- Johanna Nobili
  jnobili@clio.unice.fr
  http://www-mips.unice.fr/~jnobili

Encadreur: Franck Diard
diard@essi.fr , http://www.essi.fr/~diard