OKTAL SYNTHETIC ENVIRONMENT

INSA TOULOUSE

Rapport de stage, Juin-Septembre 2005

Rendu de forêts en temps réel

Maître de stage : Paul Pitot

Nicolas Bonneel
5 GMM

Remerciements

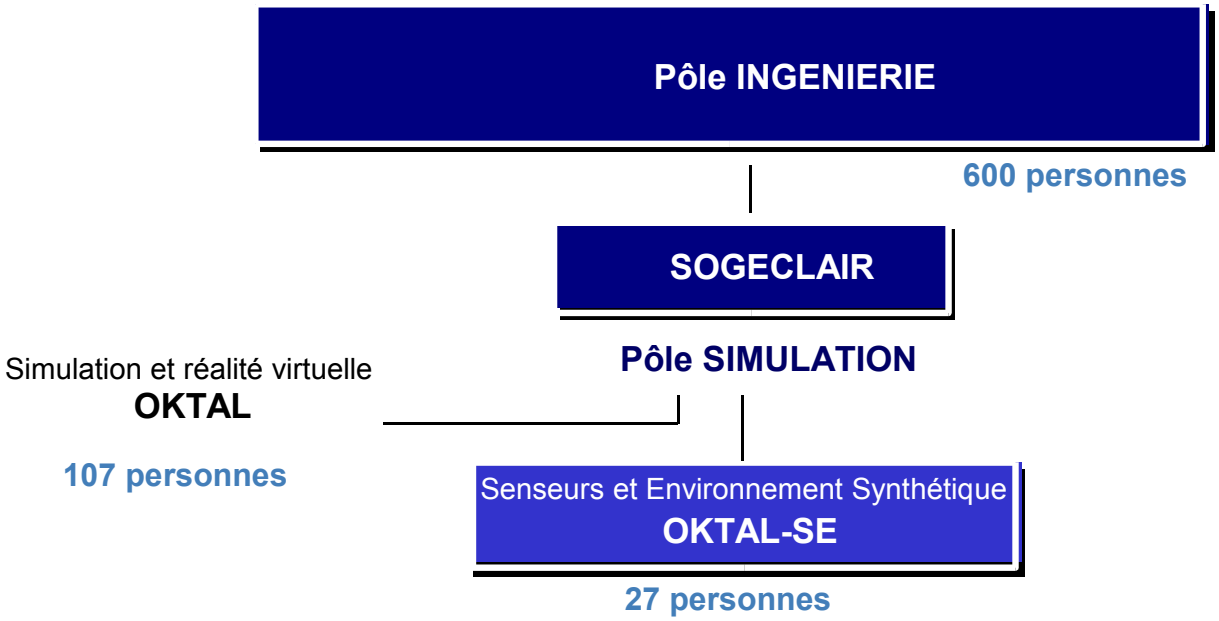
Je remercie tout particulièrement Jean Latger, Directeur Général d'Oktal SE pour m'avoir accepté au sein de son entreprise de fin Juin à début Septembre 2005, et Paul Pitot, Directeur Scientifique et responsable du stage, pour tout le temps qu'il m'a accordé pour m'aider à réaliser les objectifs du stage.

Sommaire

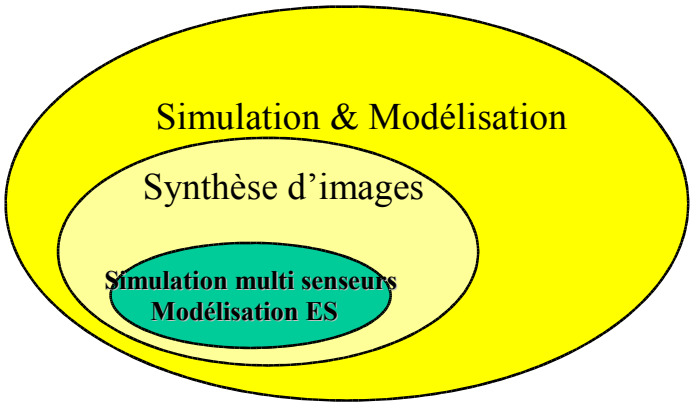
A- Présentation de l'entreprise

I- Identité

OKTAL-SE fait partie du Groupe SOGECLAIR, spécialisée dans les environnements synthétiques et la simulation multi senseurs.



Créée le 2 Février 2001 à Toulouse, OKTAL-SE possède un capital de 754 k€, et un chiffre d'affaire de 2,2 M€ en 2004.



OKTAL-SE est une petite structure réactive spécialisée dans la simulation à forte valeur ajoutée scientifique basée sur les technologies de synthèse d'images appliquée au domaine de la simulation multi senseurs et de la modélisation de l'environnement synthétique (E&S)

OKTAL-SE est un acteur engagé dans le marché de la réalité virtuelle et la simulation. Au sein de cet énorme domaine technique, OKTAL-SE adresse tout particulièrement le segment de niche des logiciels de simulation multi senseurs d'environnements synthétiques. L'offre d'OKTAL-SE cible les domaines de l'optronique, de l'électromagnétisme et de l'acoustique et se compose d'un ensemble d'outils logiciels associés à des prestations de services d'accompagnement (i.e., conseil, études, maintenance, support technique, etc..).

II- Compétences générales

L'offre OKTAL SE s'adresse aux secteurs de la Défense ainsi qu'aux marchés de l'automobile, des télécommunications, du spatial, de l'aménagement du territoire et de l'aviation civile. Dans ce segment, OKTAL SE est reconnu comme un acteur possédant un savoir faire pointu et fortement différencié.
OKTAL-SE possède aujourd'hui 3 noyaux de compétence forts et reconnus :

Logiciels de simulation senseurs

- Catalogue de produits sur étagère
- Formation sur mesure
- Études basées sur ses produits
- Session de formation générale
- Maintenance

Services spécialisés

- Études scientifiques sur la simulation
- Régie à haute valeur ajoutée
- Développement logiciels spécifiques
- Intégration logicielle
- Intégration matérielle

Modélisation d'environnement synthétique

- Catalogue de terrains et d'objets 3D
- Production de terrains 3D sur demande
- Caractérisation physique de terrains et d'objets 3D
- Modélisation de l'atmosphère

III- Les clients d'Oktal-SE

Il existe 4 types de clients :

- Les grands industriels (de la défense ou civils) :
  Ils interviennent comme intégrateur de leurs solutions dans des systèmes complexes et à forte valeur ajoutée :

  France Télécom R&D
  Renault Recherche
  MBDA (fournisseur de systèmes d'armes)
  SAGEM (fournisseurs de senseurs)
  Dassault
  SNECMA
  EADS
  OKTAL
  BGT
  LFK

- Les ministères de la Défense français ou étrangers à travers leurs centres d'expertise technique et leurs services programme :

  SPART
  SPAé
  SPOTI
  CELAR,ETBS, ETAS, CTA, CEG, LRBA,CEV
  WTD81
  FMV
  MOD Corée du Sud

- Les sociétés ou centres de recherche civils ou militaires français et étrangers :

  ONERA
  FOI
  FGAN
  CEA
  CNES
  ISL

- Les organismes pouvant prendre en charge des programmes de recherche français ou européens :

  L'ESA/ESTEC
  La communauté européenne à travers ses programmes FP xxx
  Le ministère de la recherche à travers des procédures types RNRT, RNTL
  Le ministère des transports à travers les procédures PREDIT

B- Déroulement du stage

I- Problématique

Un logiciel permettant de reconstituer le quart Sud-Est de la France en 3D à partir de données diverses (lignes de niveaux, contours de surfaciques comme les lacs ou de linéiques comme les rivières...) a été développé par Oktal SE. Ce terrain 3D a été commandé dans le but de simuler des survols en hélicoptère pour l'entraînement de pilotes. Il s'agit du projet AGETIM.

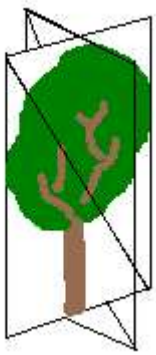
Le premier but du stage a été d'intégrer à ce programme la gestion de forêts denses visuellement acceptables pour du survol et aux performances suffisamment correctes pour pouvoir être intégrées à une zone couvrant tout le quart Sud-Est de la France.
Un second objectif fut de trouver d'autres représentations de forêts de meilleure qualité mais pour des simulations géographiquement plus restreintes.
Le langage utilisé est le C++.

II- Représentations de forêts

Dans un premier temps, plusieurs représentations de forêts ont été testées afin d'évaluer les performances ainsi que l'aspect visuel qu'elles donneraient.

1- Les arbres en croix

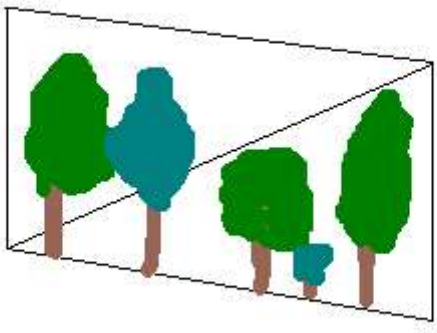
Il s'agit d'une représentation qui créé une nouvelle géométrie pour chaque arbre, et lui associe une texture d'arbre contenant de la transparence. Elle est donc adaptée pour le rendu d'arbres vus de près mais ne conviendra pas pour des forêts à grande échelle. En effet, chaque arbre comporte 4 triangles (voir schéma), et une forêt peut contenir plusieurs dizaines de milliers d'arbres (1 km² de forêt en comptant 1 arbre pour 20m² contiendra 50 000 arbres soit 200 000 polygones). En revanche, 4 polygones par arbre nous permet d'obtenir le détail des arbres depuis n'importe quelle direction de regard.



2- Les lisières

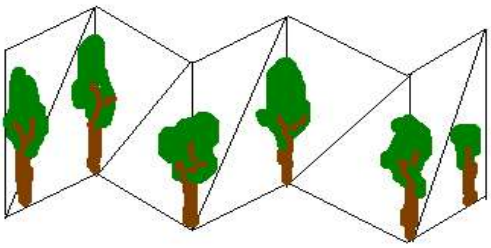
Une représentation sous forme de « rideaux » d'arbres ou lisières a été essayée. Plusieurs types de lisières ont été testées :

- Des lisières planes :
  Elles permettent d'économiser un nombre important de polygones : Plusieurs arbres sont placés sur une seule face plane texturée. Ces lisières ne peuvent être vues que si l'observateur se situe suffisamment face à elles, d'où l'intérêt d'en mettre un grand nombre dans tous les sens dans la scène, et si l'observateur est trop proche il s'apercevra de la supercherie.
  Un ensemble de textures génériques est utilisée : chaque lisière ne possède pas sa propre texture car la mémoire texture des cartes graphiques est limitée (≈ 128 Mo pour l'ordre de grandeur) et le reste du paysage utilise déjà plusieurs Mo de textures.

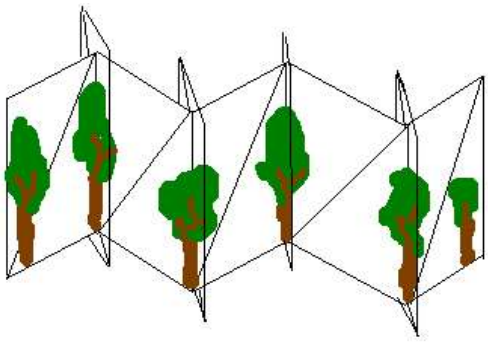


- Des lisières en « accordéon » :
  Un « accordéon » est créé et un arbre est créé sur chaque arête verticale (à part la première et la dernière pour lesquelles l'arbre est décalé). On peut ou non rajouter une face supplémentaire à chaque arête pour créer un aspect arbre en croix.

  Elles n'offrent pas un réel gain en nombre de polygones par rapport aux arbres en croix, mais permettent de créer des « strips » : Les différents polygones partagent les mêmes sommets deux à deux, ce qui est réputé pour optimiser les performances du rendu par le moteur 3D : OpenGL. En revanche on a remarqué qu'il y a trop de sur-écriture : dans une direction donnée il apparaît trop de polygones à rendre en même temps à cause de la transparence de beaucoup d'entre eux due à l'espace entre chaque arbre, ce qui ralentit beaucoup les performances.

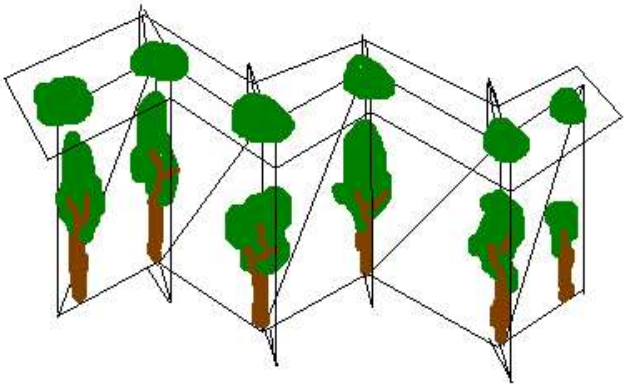


Lisière sans face de croix



Lisière avec face de croix

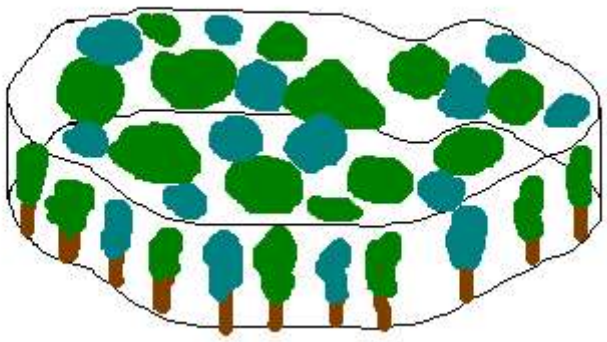
De plus, on peut rajouter une série de faces de canopée pour complexifier un peu le modèle, mais les performances sont insuffisantes.



On s'appliquera à garder un espacement constant entre chaque arbre, ce qui permet de créer des géométries d'accordéon différentes tout en conservant une unique texture générique pour toute la forêt (ou un ensemble de textures génériques).

3- La boîte à chaussure

Il s'agit de représenter la canopée de la forêt globalement, ainsi qu'une lisière sur toute la bordure de la forêt. La canopée pourra être positionnée à une hauteur inférieure à celle de la lisière, ce qui permet de faire dépasser les arbres sur le contour de la forêt.
La boîte à chaussure permet un réel gain de performance puisqu'il y a autant de faces de canopée que de faces de terrain sous cette canopée. De plus, si la canopée est suffisamment dense en arbres, on peut supprimer complètement les faces de terrain sous la canopée, ce qui nous permet d'avoir une forêt pour un coût presque nul (seules les faces des lisières externes sont ajoutées). Cette représentation est suffisante pour un point de vue situé à plusieurs centaines de mètres de la forêt.



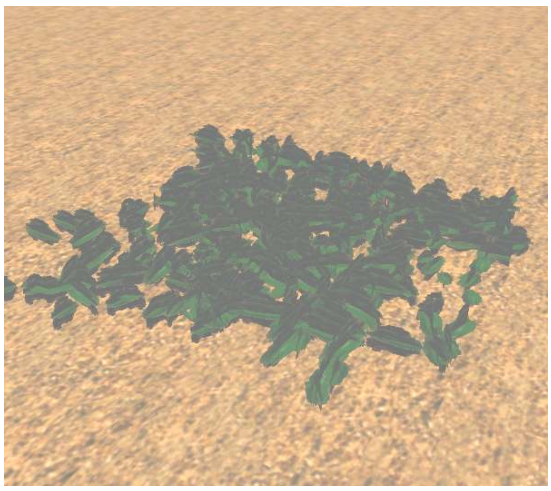
Il faut faire attention à détecter correctement le contour réel de la forêt car le terrain est découpé en « zones » (opération de « hersage »), et une forêt pourra donc être scindée. Le contour de la forêt n'est donc pas le contour tel qu'il apparaît mais tel qu'il est fourni avant l'opération de hersage.
Les lisières n'appartenant pas au contour réel de la forêt mais appartenant au contour apparent devront être tronquées à la hauteur de la canopée pour que le hersage ne soit pas révélé par un quadrillage de lisières dépassant de la canopée. On utilise donc des faces qui ne dépassent pas de la canopée pour boucher les trous qui pourraient être visibles entre deux niveaux de détails consécutifs.

De plus, des tests où la canopée est en relief pour épouser la forme des cimes des arbres avec une gestion automatique des niveaux de détails par le moteur ROAM (gestion des niveaux de détail de manière continue) ont été réalisés, ainsi que des tests avec plusieurs couches de canopées, mais les performances obtenues ne nous ont pas paru suffisantes.

4- La texture

Une représentation sous forme de texture pleine où seule la texture de la canopée mélangée à celle d'un sous-bois remplace celle du terrain a été testée. Cette représentation est suffisante pour une vue très éloignée de la forêt.

Pour les représentations sous forme de lisières et d'arbres en croix, on génère un patch de forêt : On génère une zone de 200m*200m, par exemple, de forêt qui sera répétée. De cette manière, la génération d'un patch ne dépend pas de la forme de la forêt, et on pourra par la suite référencer ces patchs au lieu de les instancier (Ce travail sera à faire lorsque qu'un visualiseur de terrain pouvant planter des forêts en temps réel sera disponible. Nous nous contenterons pour le moment de copier et d'altimétriser les lisières et arbres un par un sur le terrain avant la phase de rendu.).
Les lisières pourront déborder un peu du patch pour une meilleure continuité visuelle lorsque l'on posera les patches côtes à côtes.



Patch de forêt (lisières droites + faces de canopée)

Par ailleurs, toutes les normales sont positionnées artificiellement vers le haut afin que l'éclairement soit uniforme lors de la visualisation, l'éclairement réel des arbres étant déjà donné par leurs textures. Cela permet d'éviter aux faces opposées à la direction de la source lumineuse de ne pas être éclairées du tout et d'apparaître noires au lieu d'avoir la texture d'arbre d'origine.

III- Les niveaux de détail

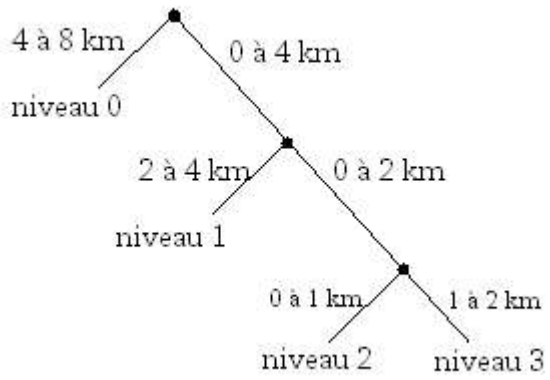
Une façon de réduire le nombre de polygones affichés en même temps est de procéder à la réalisation de niveaux de détail différents pour la gestion des forêts. Il s'agit d'utiliser des représentations différentes si l'on se situe à faible distance de la forêt ou à grande distance. D'une manière générale, les différentes représentations sont précalculées : il s'agit donc de niveaux de détail statiques. Certaines méthodes utilisent des subdivisions de polygones de manière continue pour affiner le maillage lors du rendu : il s'agit d'une gestion ROAM des niveaux de détail.

Nous travaillons ici dans la version statique des niveaux de détail puisque le visualiseur 3D qui sera utilisé dans le cadre du projet AGETIM ne gère pas encore les niveaux de détail continus.

Une gestion de différents niveaux de détail pour l'affichage du terrain était déjà codé. Nous avons choisi de découpler les niveaux de détail du terrains de ceux de la forêt : en effet, nous pouvons avoir besoin d'un niveau de détail très fin lorsque l'on est très très près des arbres sans pour autant avoir besoin que le maillage du terrain continue de se raffiner.
En pratique, chaque type de forêt possède une gestion indépendante de ses niveaux de détail.

Chaque niveau est affiché sur une distance couvrant la moitié de celle du niveau précédent, le dernier niveau étant le plus fin, et le premier niveau, le niveau 0, étant le plus grossier. La distance calculée est définie par la distance entre l'observateur et le centre de la zone à afficher.
Par exemple, s'il n'y a que 4 niveaux de détail et que l'on démarre avec une zone que l'on considérera faire 8 km, le niveau 3 s'affichera de 0 à 1 km, le niveau 1 s'affichera de 1 km à 2 km (un niveau étant défini de 0 à 1 km, le niveau 2 ne s'affichera pas sur cet intervalle), le niveau 1 de 2 km à 4 km et le niveau 0 de 4km à 8km.

Ce système est codé par un arbre décisionnel, qui donnerait dans l'exemple précédent :



Dans le cadre du projet de reconstruction du quart Sud-Est de la France, nous avons choisi de ne conserver que deux niveaux de détail :
- La boite à chaussure pour des distances d'observation faibles
- La texture plaquée au sol pour les grandes distances.

Pour les projets de plus petite dimensions géographiques, nous avons choisi les niveaux de détail suivant :

- D'abord les arbres en croix lorsque l'on est très proche de la forêt (niveau de détail 5)
- Lisières planes ainsi que canopée ensuite (niveau 4)
- Puis on ne conserve qu'une boîte à chaussure (niveaux 3 et 2)
- Et enfin on garde uniquement la texture de canopée pour les très grandes distances (niveaux 1 et 0)

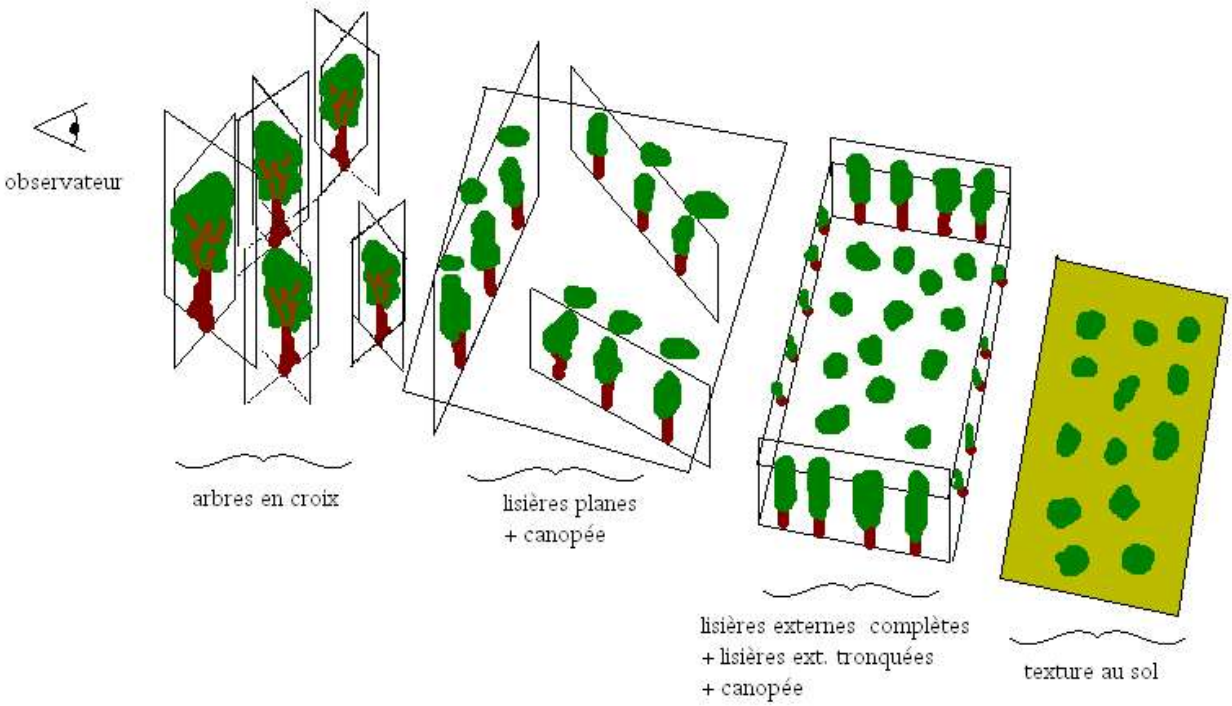


Schéma récapitulatif des représentations choisies dans le cadre de forêts élaborées

IV- Le shader

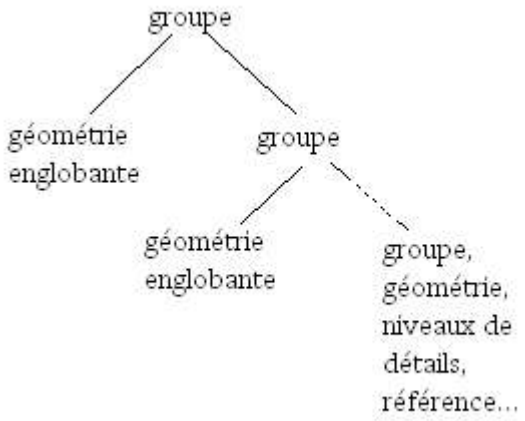
Pour faciliter la tâche des infographistes, une forêt est considérée comme une « texture » appliquée au sol. Le but est de rendre cette texture comme une forêt en 3D. Il s'agit donc d'une texture 3D ayant des propriétés particulières : elle sera rendue par un « shader ».

Par ailleurs, il nous a paru judicieux de pouvoir utiliser cette même méthode de rendu pour pouvoir planter d'autres types d'éléments que les forêts : pouvoir utiliser des textures « villes 3D », pouvoir planter des cailloux, de l'herbe, des barrières...
Les normes d'Oktal SE étant anglaises, le nom du shader est donc le 3DCoverageShader.

Pour rester le plus générique possible, ce shader a pour propriété un ensemble d'opérations. Une classe abstraite CoverageOperation est donc créée. Elle possède une fonction virtuelle pure qui permet d'effectuer le rendu de la texture 3D, et l'attribut « level » déterminant le niveau de détail auquel cette opération s'applique.
Le shader quant à lui possède comme attribut la hauteur maximale de la forêt (ou une hauteur quelconque de référence) ainsi qu'une liste de CoverageOperation.

Un ensemble d'opérations dérivant de cette classe a été construit :

- CoverageOperationCover : Permet de « planter » n'importe quel objet tridimensionnel de référence (ou pattern) répétitivement sur toute la zone à texturer. Possède les attributs : taille de la zone, le nom du fichier contenant le pattern (« file ») et le nom de l'objet pattern (« object ») dans ce fichier, un fichier pouvant contenir plusieurs patterns différents aux noms différents.
  Le pattern est altimétrisé. Le pattern est composé d'éléments insécables afin de ne pas couper des lisières au milieu d'un arbre, et afin de réduire naturellement la densité des arbres en bordure de forêt.
  Il est de la forme :



  La géométrie englobante pouvant être un polygone (le groupe sera présent si tous ses segments sont dans la surface à traiter), ou un ensemble de points (le groupe sera présent si tous les points sont dans la surface à traiter).

  Dans le cas des forêts, la géométrie à copier est un groupe de lisières, chacune possédant comme géométrie englobante une série de points correspondant à la position de chaque arbre. Les arbres ne pourront donc pas être tronqués.

- CoverageOperationDuplicate : Permet de dupliquer le sol texturé et de l'élever afin de former une canopée. Il possède les attributs : le pourcentage de la hauteur maximale (« heightScale », = hauteur d'élévation).

- CoverageOperationSide : Permet de créer des lisières sur le bord de la forêt. Il possède les attributs : le pourcentage de la hauteur maximale (= hauteur de la lisière), un booléen (« closingSide ») définissant s'il s'agit des lisières sur le contour du hersage ou sur le contour réel de la forêt qu'il faut créer, un nom de fichier contenant un pattern de lisière : le profil de la lisière et sa texture, ainsi que le nom de l'objet pattern dans ce fichier.

- CoverageOperationTexturize : Permet de texturer le sol avec une texture différente. Cela permet de changer de texture pour des niveaux de détails différents. Possède les attributs : un nom de fichier d'un pattern et le nom du pattern contenant une face de référence avec la texture à appliquer (ainsi que son mapping etc..).

- CoverageOperationHollow : Permet de supprimer le sol. Ne contient aucun paramètre. Pour un niveau de détail donné, cette opération doit être effectuée en dernier.

Des outils de sérialisation et désérialisation sous forme XML (la bibliothèque XIO d'Oktal SE) permettent ainsi d'écrire un ensemble de ces classes dans un fichier qui définira la texture et les opérations à appliquer.

Le « rendu » consiste, à partir du fichier contenant le maillage du terrain, d'y ajouter le maillage des représentations voulues.

A noter qu'il n'est pas nécessaire de répéter les opérations identiques d'un niveau de détail à un autre niveau directement supérieur : un niveau dont la description est absente sera interprété comme le niveau précédent. Cela est utile lorsque nous avons une forêt grossière au niveau 0, que cette représentation est maintenue jusqu'au niveau 4 par exemple, pour continuer sur une représentation plus fine au niveau 5 : dans ce cas, il ne sera pas nécessaire de répéter la description du niveau 0 aux niveaux 1, 2, 3 et 4.

Nous obtenons ainsi, pour le cas des forêts « réalistes », une texture qui sera enregistrée sous la forme :

```
<sdm4::3DCoverageShader key="3d coverage shader #0">
        <ElementList key="coverageOperations">

                <sdm4::CoverageOperationCover>
                        <IntProperty key="level">
                                5
                        </IntProperty>
                        <StringProperty key="file">
                                crosses.bdd
                        </StringProperty>
                        <StringProperty key="object">
```

```
                crosses
        </StringProperty>
        <DoubleProperty key="size">
                160
        </DoubleProperty>
</sdm4::CoverageOperationCover>

<sdm4::CoverageOperationTexturize>
        <IntProperty key="level">
                5
        </IntProperty>
        <StringProperty key="file">
                gabarits.bdd
        </StringProperty>
        <StringProperty key="object">
                solforet
        </StringProperty>
</sdm4::CoverageOperationTexturize>


<sdm4::CoverageOperationCover>
        <IntProperty key="level">
                4
        </IntProperty>
        <StringProperty key="file">
                lisieressimples.bdd
        </StringProperty>
        <StringProperty key="object">
                lisieres_simples
        </StringProperty>
        <DoubleProperty key="size">
                160
        </DoubleProperty>
</sdm4::CoverageOperationCover>

<sdm4::CoverageOperationDuplicate>
        <IntProperty key="level">
                4
        </IntProperty>
        <StringProperty key="file">
                gabarits.bdd
        </StringProperty>
        <StringProperty key="object">
                canopee
        </StringProperty>
        <DoubleProperty key="heightScale">
                0.4
        </DoubleProperty>
</sdm4::CoverageOperationDuplicate>

<sdm4::CoverageOperationTexturize>
```

```
        <IntProperty key="level">
                4
        </IntProperty>
        <StringProperty key="file">
                gabarits.bdd
        </StringProperty>
        <StringProperty key="object">
                solforet
        </StringProperty>
</sdm4::CoverageOperationTexturize>

<sdm4::CoverageOperationSide>
        <IntProperty key="level">
                2
        </IntProperty>
        <StringProperty key="file">
                gabarits.bdd
        </StringProperty>
        <StringProperty key="object">
                lisiere
        </StringProperty>
        <BooleanProperty key="closingSide">
                FALSE
        </BooleanProperty>
</sdm4::CoverageOperationSide>

<sdm4::CoverageOperationDuplicate>
        <IntProperty key="level">
                2
        </IntProperty>
        <StringProperty key="file">
                gabarits.bdd
        </StringProperty>
        <StringProperty key="object">
                canopee
        </StringProperty>
        <DoubleProperty key="heightScale">
                0.6
        </DoubleProperty>
</sdm4::CoverageOperationDuplicate>

<sdm4::CoverageOperationTexturize>
        <IntProperty key="level">
                2
        </IntProperty>
        <StringProperty key="file">
                gabarits.bdd
        </StringProperty>
        <StringProperty key="object">
                solforet
        </StringProperty>
```

```
                </sdm4::CoverageOperationTexturize>

        </ElementList>

        <DoubleProperty key="height">
                        20.000000
        </DoubleProperty>

        <StringProperty key="Property Layer Name">
        </StringProperty>

</sdm4::3DCoverageShader>
```

V- L'utilitaire

Un utilitaire pour créer les patterns a été créé. Celui-ci permet, grâce à des fonctions de lisières simples ou en accordéon, de canopées etc.. de construire les fichiers utilisables par le shader.

On lui fournit un fichier contenant la liste des matériaux de la scène afin d'y ajouter les matériaux de lisières etc.. On lui fournit aussi la liste des textures d'arbres individuels (vus de profil et vus de dessus) et leur dimension, ainsi que la probabilité que chaque texture apparaisse, le nombre de types de lisières différentes souhaitées (deux types de lisières sont différents si les textures qu'elles utilisent sont différentes), le nombre d'arbres total dans le patch.
Quelques paramètres supplémentaires sont facilement réglables dans le code : le nombre d'arbres par lisière, la distance moyenne entre chaque arbre, l'intervalle aléatoire dans lequel est choisi la taille de l'arbre, l'angle maximum entre les arbres des lisières en accordéon...

Une première fonction permet de déterminer l'implantation de tous les arbres ainsi que leurs types, puis on appelle les différentes fonctions codées qui généreront un patch de lisières, ou bien un patch d'arbres en croix etc... ainsi que les textures correspondantes.

L'implantation des arbres est faite de manière à former des lisières en accordéon dont l'angle entre chaque arbre ne dépasse pas un certain seuil, afin de pouvoir construire les lisières « planes » en projetant les arbres de la lisière en accordéon sur le plan médian.
- Les lisières « planes » sont ainsi construites en gardant la géométrie de la face qui contient le premier et le dernier arbre, et en appliquant la texture de la lisière en accordéon (on néglige la déformation). En revanche, pour construire la géométrie englobante de la lisière simple, on projette la position des arbres sur la face.
- Les arbres en croix sont séparés en deux fonctions : la première permettant de créer la géométrie des faces parallèles à la lisière, une deuxième créant la géométrie des faces situées à la bissectrice de chaque angle formé par les accordéons. Cela permet de réutiliser cette deuxième fonction afin de créer des faces supplémentaires aux lisières en accordéon pour ajouter des faces de croix.

C- Bilan

I- Améliorations futures

Plusieurs fonctionnalités n'ont pas été implémentées par manque de temps et permettraient d'avoir un rendu meilleur.

1- Le mapping

Tout comme les autres matériaux 2D, les matériaux 3D peuvent avoir leur propre mapping. Cela permettrait par exemple, grâce à un mapping linéaire, d'implanter des barrières le long des routes grâce à ce même CoverageShader.
Il s'agit donc de modifier les coordonnées horizontales x et y (l'altitude étant le z) interprétées comme des coordonnées de texture UV grâce au mapping inverse, et de planter les objets ainsi déformés à leurs nouvelle position.
Un librairie de gestion des repères est déjà codée et pourra simplifier la mise en oeuvre de ce mapping.

2- Les arbres « géométriques »

Dans les représentations testées, seuls des arbres sous formes de textures étaient envisageables. Nous ne sommes donc pas descendu à une description géométrique réelle de la forme des arbres (tronc, branches, feuilles...).
En revanche, des librairies telles que SpeedTree sont spécialisées dans ce type de représentation, mais n'ont pas des performances suffisantes pour gérer des forêts complètes et denses. Il est donc envisageable d'intégrer un moteur de type SpeedTree au niveau de détail le plus fin afin de rendre les forêts réellement pénétrables.
Le shader tel qu'il est codé devrait avoir une souplesse suffisante pour y intégrer cette représentation.

3- Diversification des patches

Pour diversifier la forêt, il pourrait être utile de choisir aléatoirement les patches parmi un ensemble de patches prédéfinis. Cela pourrait être une option du CoverageOperationCover que de choisir parmi plusieurs patterns au lieu de ne recopier qu'un seul type de pattern.
On observerait alors moins de répétitivité dans la forêt.

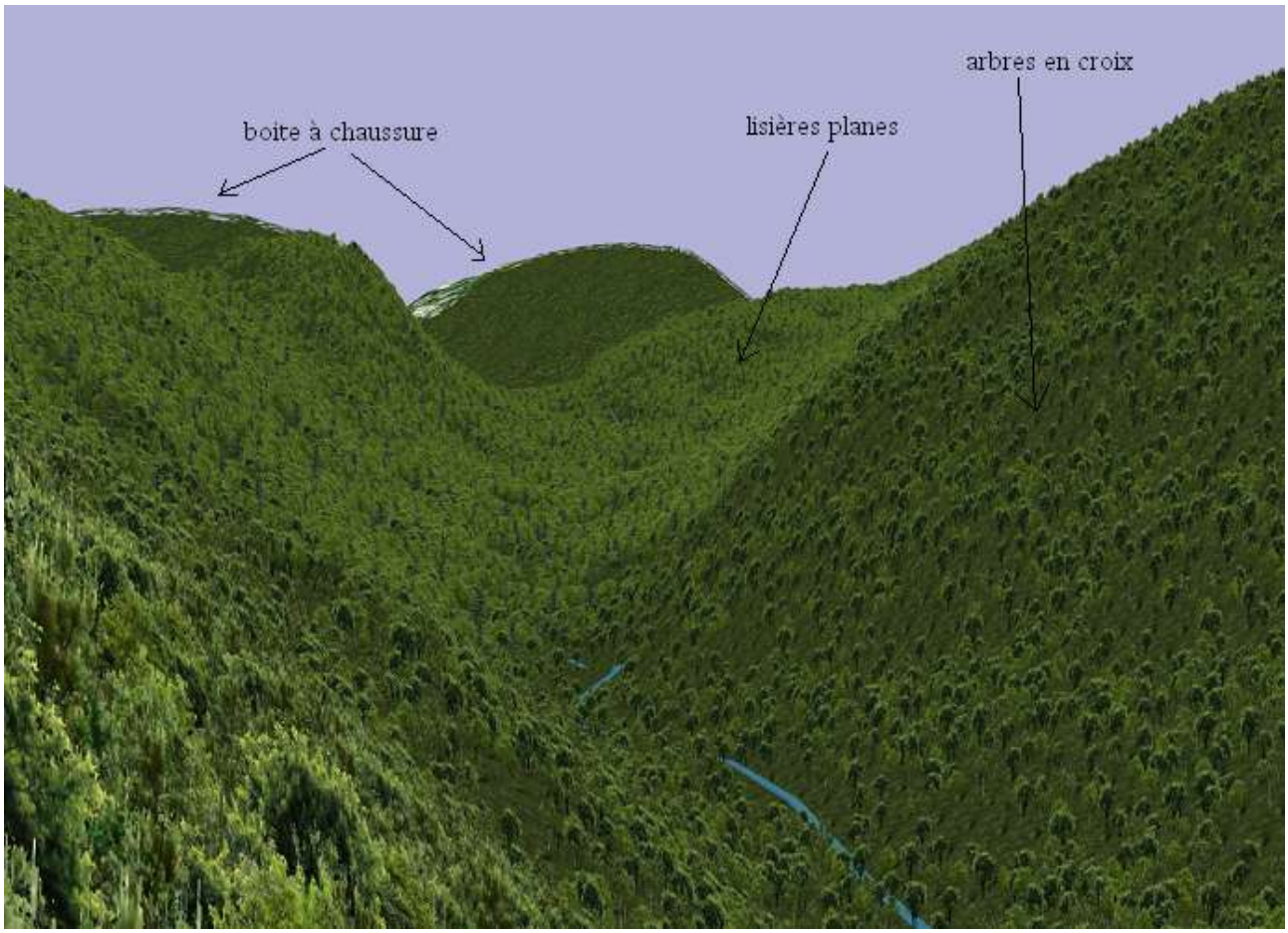
4- Une opération « eau plate »

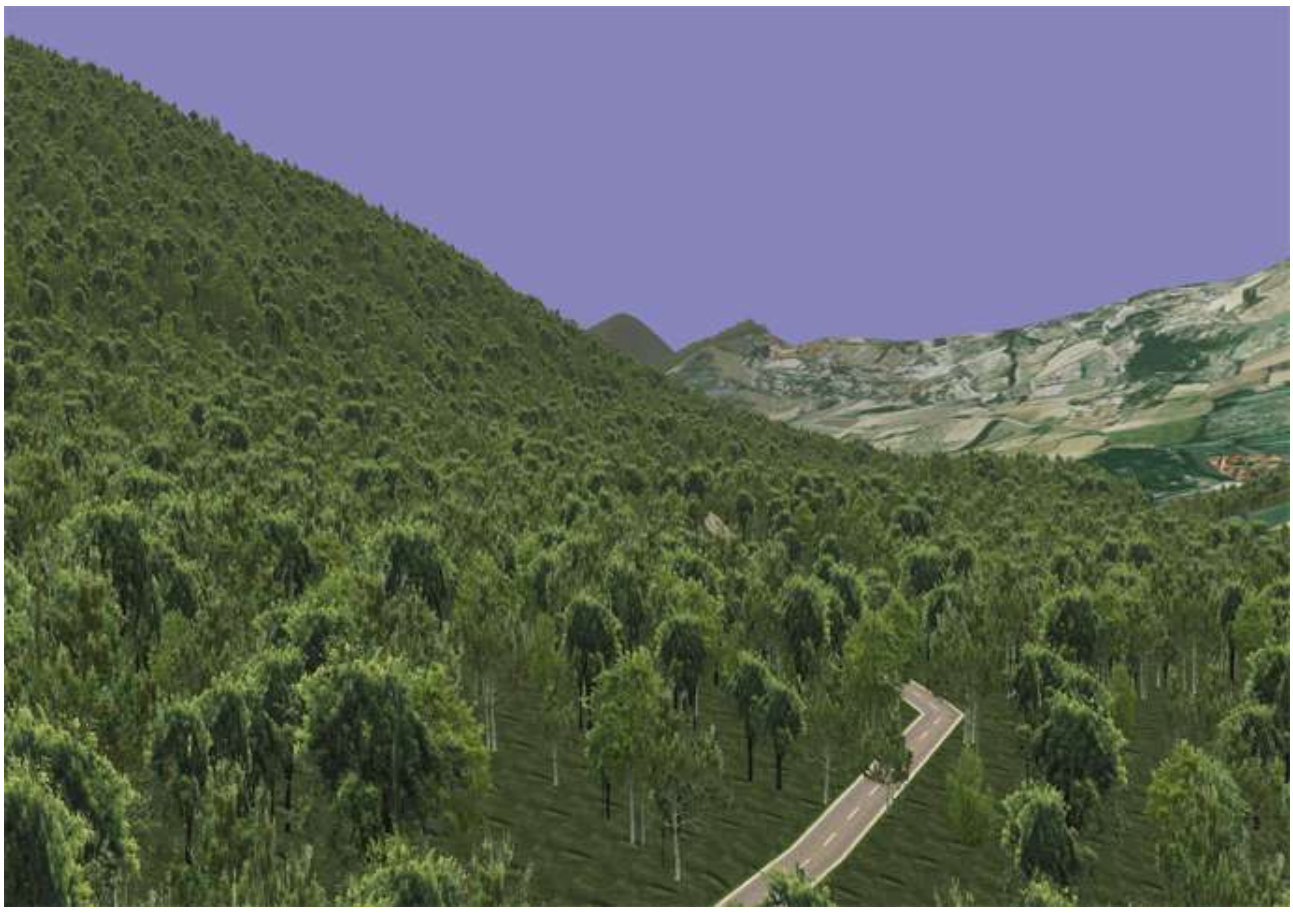
Une opération intéressante dans le cadre du projet AGETIM, bien que ne concernant pas les forêts mais le 3DCoverageShader en général, est une opération qui permettrait de rendre le terrain parfaitement plat et horizontal afin de pouvoir créer des lacs.

II- Résultats

Les performances obtenues pour les forêts de type boîte à chaussure pour le projet AGETIM sont suffisantes puisque le coût d'une boîte à chaussure est presque nul si on enlève le sol sous la canopée. Nous obtenons donc une simulation temps réel, de qualité assez médiocre lorsque la forêt est vue de près puisque les arêtes de la boîte sont visibles facilement, et que le survol de la canopée ne laisse pas apparaître de parallaxe avec le sol.

Pour la représentation plus réaliste de la forêt, nous obtenons des simulations proches du temps réel lors de l'affichage de la forêt sur une surface d'écran restreinte, et des performances plutôt mauvaises en plein écran. Il s'agirait de problèmes de fill-rate (une surface à texturer et remplir trop importante pour la carte graphique). En revanche la qualité visuelle de ces forêts est correcte.





III- Conclusion

Ce stage m'a permis d'apprendre beaucoup de techniques utilisées dans le domaine de la 3D, tels que les niveaux de détail (ROAM et statiques) et divers algorithmes de calculs géométriques (calcul de normales de polygones complexes, calcul d'un contour d'un ensemble de faces, ...).
Il m'a permis d'étudier toute la problématique associée aux niveaux de détail statiques : les transitions qui ne doivent pas se voir, le hersage du maillage et la continuité entre les zones, ...
Enfin, j'ai pu améliorer mes connaissances en C++ en manipulant la STL, et la programmation objet d'un manière générale, dans une ambiance de travail agréable grâce aux nombreuses heures de conseils que mon tuteur m'a accordées.

Le sujet du stage n'a, quant à lui, que partiellement été rempli puisqu'une représentation satisfaisante du point de vue performances n'a pas vraiment été trouvée (elle ne convient pas à du temps réel en plein écran), mais le shader programmé durant ce stage est assez souple pour pouvoir accueillir de futures représentations de forêts plus optimales. Le sujet reste donc ouvert.