Modélisation Géométrique : Blob Tree

Olivier Berthier Alexandre Vellutini Kevin Sauvageon

20 décembre 2013

Table des matières

1 Opérateurs

Liste des opérateurs disponibles :
— Blend : A+B (dans le code d'origine)
— Sub : A-B
— Union : max(A,B)
— Intersection : min(A,B)
— Xor :
if(A & B) A - B
else if(A) A
else B

2 Primitives

Liste des primitives disponibles :
— Sphere (dans le code d'origine)
— Boîte
— Cylindre
— Cône
— Tore

3 Algorithme d'érosion

L'algorithme d'érosion développé est basé sur une répartition aléatoire. Le placement des points d'érosion se fait par tirage de points à l'intérieur et à l'extérieur de l'objet. Chaque point tiré doit se trouver dans la boite englobante. On se base sur le signe de l'intensité pour savoir si le point est à l'intérieur ou à l'extérieur de l'objet. Ensuite on utilise une dichotomie pour obtenir un point le plus proche possible de la surface de l'objet. Il ne reste plus qu'à ajouter un blob d'intensité et de rayon aléatoires à l'arbre d'érosion. Pour obtenir une érosion plus crédible il faut aussi prendre en compte les paramètres d'influence et du rayon des points d'érosion en fonction de l'objet modélisé. Au final on fait une soustraction de l'arbre d'érosion et de l'arbre de l'objet.

Pour orienter l'érosion il suffit d'indiquer un vecteur de direction non null. L'algorithme est assez simple. Lorsque l'on tire le point à l'extérieur on va lui ajouter un vecteur dont la direction est celle indiquée et dont la longueur est la diagonale de la boite englobante de l'objet. Le point sera alors obligatoirement à l'extérieur de l'objet et du côté que l'on souhaite éroder. Une dichotomie permet d'obtenir un nouveau point d'érosion qui sera bien placé.

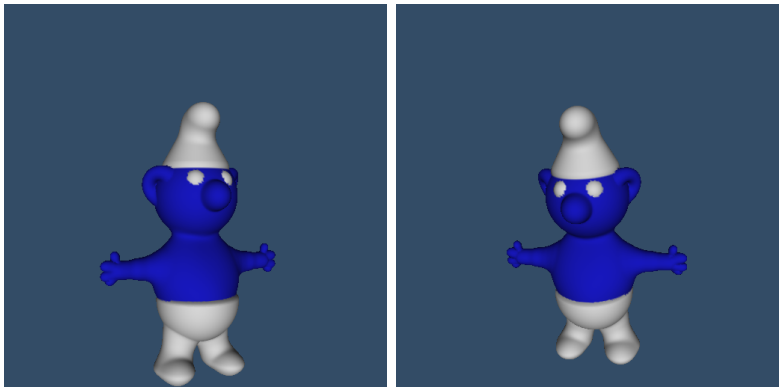
Enfin l'algorithme prend en compte l'érosion local. Une érosion local nécessite de préciser la zone que l'on souhaite éroder. Dans notre cas pour indiquer les zones d'érosion il suffit de définir un arbre de blob et de passer la boîte englobante à la fonction d'érosion. Les points intérieur seront alors tirés à l'intérieur de cet zone locale.

L'accessibilité n'a pas été gérée.

4 Ajouts

Le premier ajout est la gestion des couleurs par primitive. Lors de la construction de l'arbre il est possible de déclarer une couleur par blob. Cela permet de créer des objets de plusieurs couleurs. Il est aussi possible de déclarer une couleur pour l'érosion. Cette fonctionnalité augmente relativement la mémoire utilisée car il faut stocker la couleur pour chaque sommet de triangle.

Un système de création de scène a été mis en place afin de pouvoir avoir g´erer plusieurs objets d'arbres différents dans la scène. Il est possible d'appliquer une translation et une rotation aux objets. Cela permet de créer assez facilement des scènes un peu plus complexe, mais il faut prendre en compte l'utilisation mémoire de chaque objet.



(a) Schtroumpf (b) Schtroumpf

FIGURE 1 – Schtroumpf, 250, 4.89s

5 Objets blob

Voici les objets générés à partir d'arbres de primitives.

5.1 Objet Schtroumpf

Cet objet (Figure 1) n'est composé que d'un arbre de blob, il n'a pas d'érosion. En revanche l'arbre commençant à être assez complexe pour une programmation manuel, certains détails n'ont pas été ajoutés. Pour obtenir un personnage plus crédible et plus complexe il faudrait par exemple créer une méthode de génération à partir d'un squelette.

5.2 Objet Maison de Schtroumpf

Les maisons de schtroumpf (Figure 2) ont deux modèle de couleur. Les motifs sur le toit sont générés par une érosion orientée.

5.3 Village de Schtroumpf

Il s'agit la d'une scène qui dispatche N objets décrits ci-dessus alétoirement sur une surface (Figures 3, 4 et 5). Une condition de distance doit être respectée pour ne pas que les objets entrent en colision. Le terrain est aussi généré par un arbre de blob. L'objet bizzare au milieu de la scène est un puits.

6 Utilisation

Il n'y a pas besoin de librairies autre que OpenGL, la compilation sous GNU/Linux se fait via la commande 'make'. Pour lancer le programme il

(a) Maison Schtroumpf, polygonize : 250, érosion : 200, temps : 24.59s

(b) Maison Schtroumpf, polygonize : 250, érosion : 800, temps : 223.73s

(c) Maison Schtroumpf, polygonize : 125, érosion : 400, temps : 13.00s

(d) Maison Schtroumpf, polygonize : 125, érosion : 1000, temps : 79.90s

FIGURE 2 – Maisons Schtroumpf

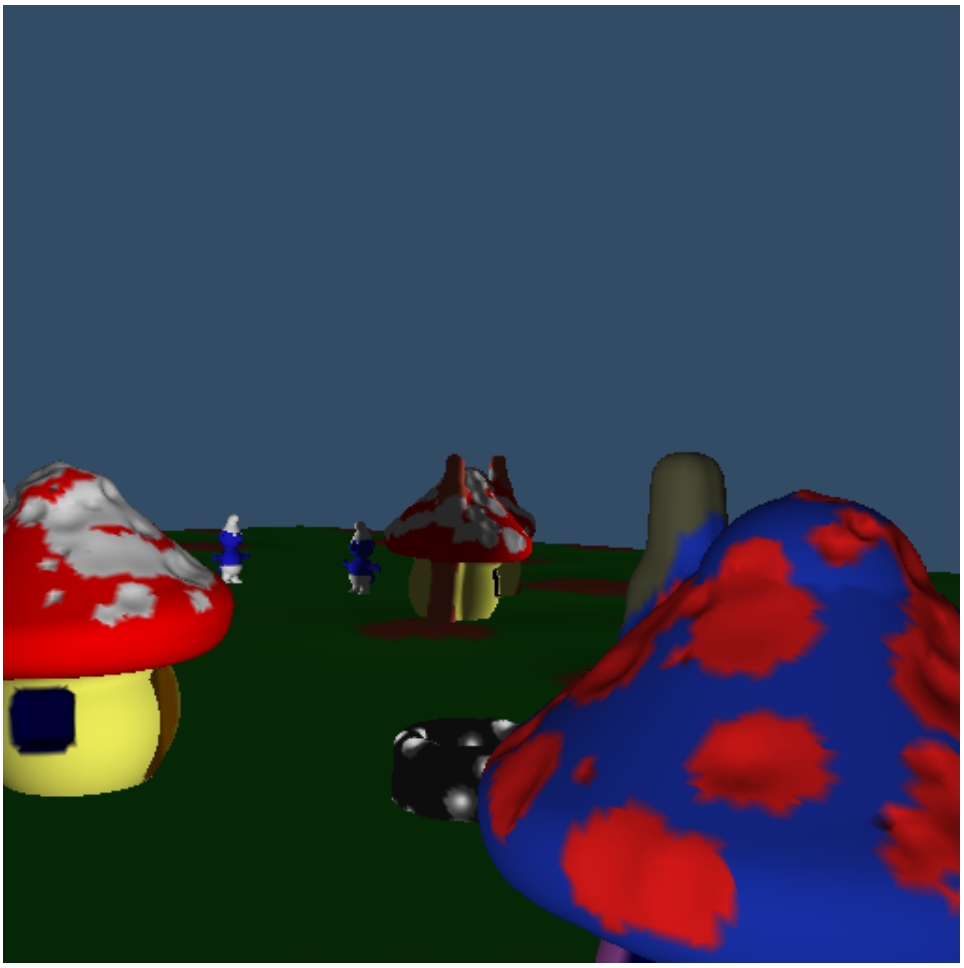


FIGURE 3 – Village Schtroumpf 20 objets, polygonize : 150, érosion : 100, temps : 54.47s

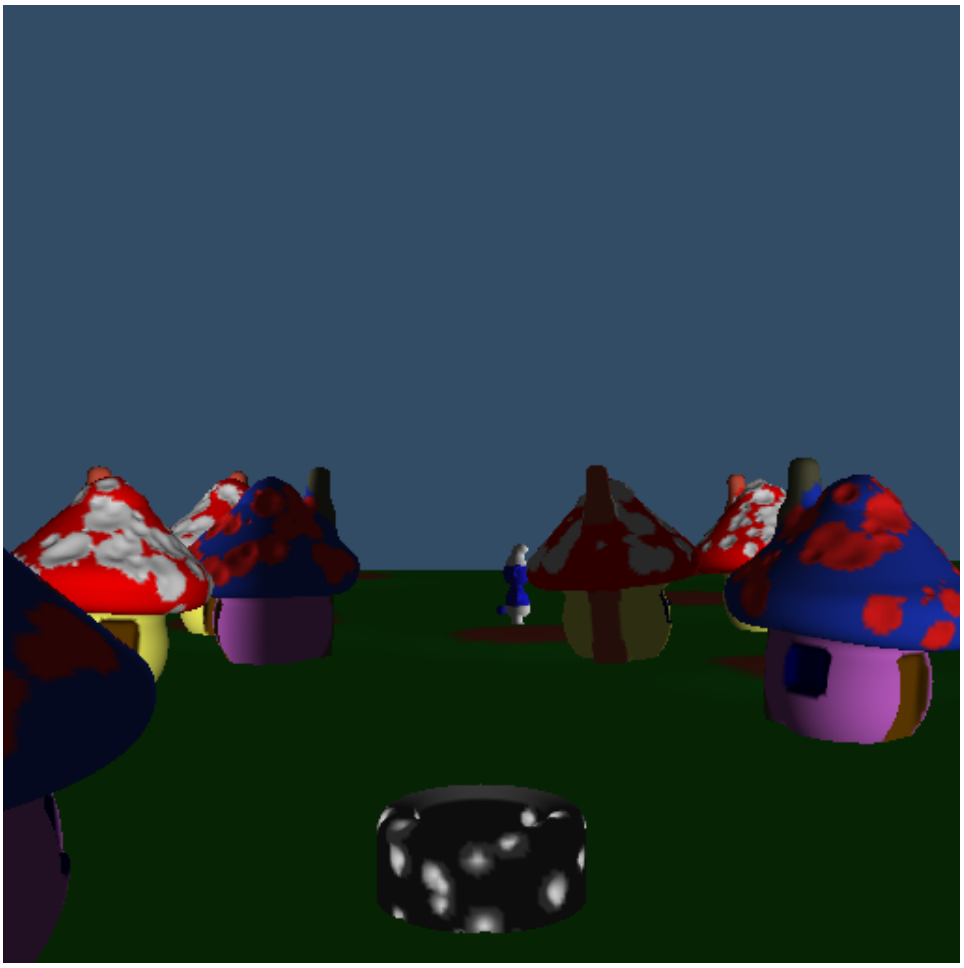


FIGURE 4 – Village Schtroumpf 40 objets, polygonize : 150, érosion : 100, temps : 99.33s

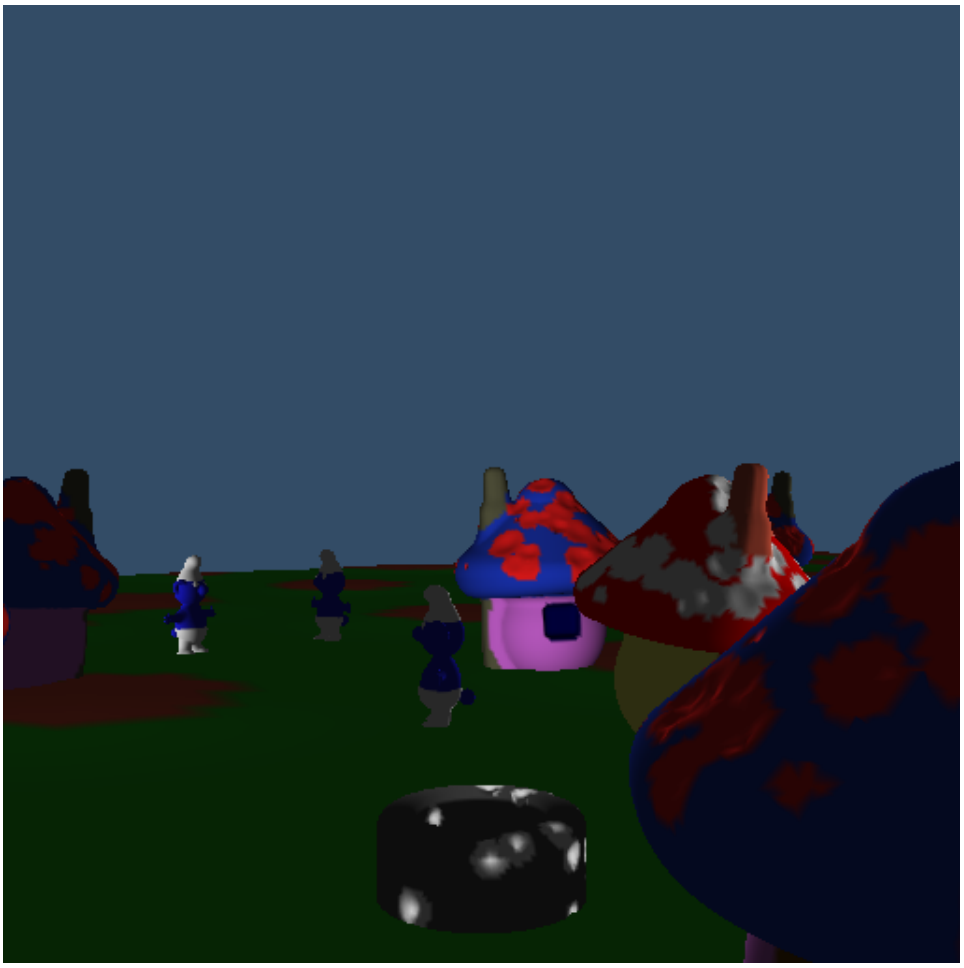


FIGURE 5 – Village Schtroumpf 40 objets, polygonize : 150, érosion : 100, temps : 99.33s

suffit d'utiliser la commade './blob'. Pour changer le niveau de polygonisation et d'érosion il faut changer les valeurs de POLYGONIZE_V et ERODE_V dans blobobject.h.