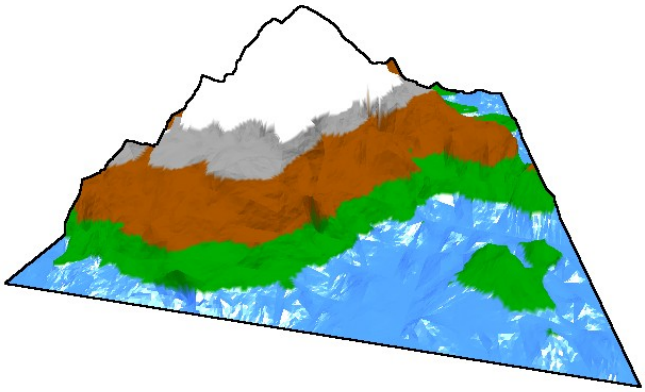
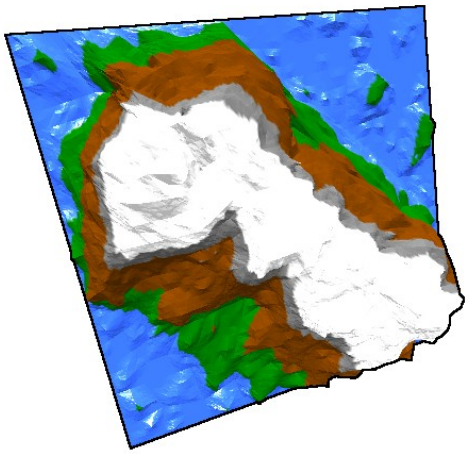
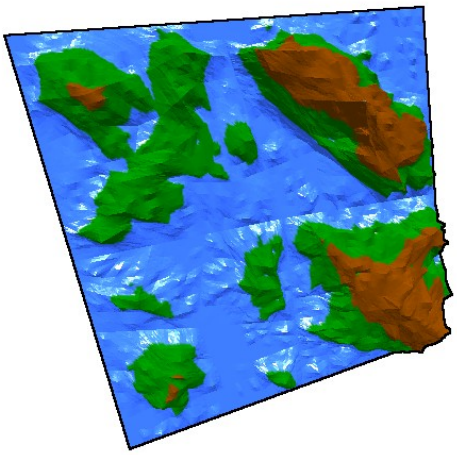
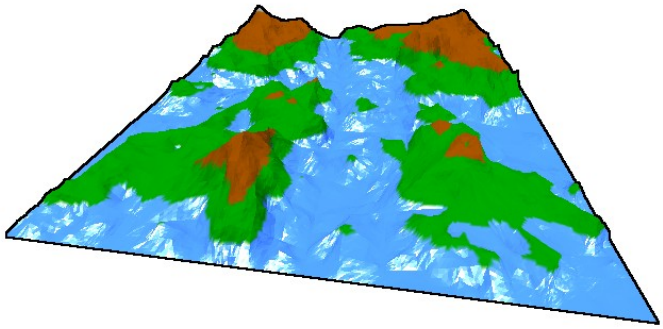
Bertrand Chazot, casier 29
Micro-projet de C n°41

# *Montagne Fractale*









# ***Cahier des charges***

Ce projet consiste à générer un terrain -constitué de facettes (polygones)- dans l'espace et à le visualiser en coloriant chaque facettes en fonction de son altitude (mer, plaine, montagne, neige...) et de son orientation par rapport à la source lumineuse (soleil).

Le logiciel procédera au calcul des coordonnées des différents sommets grâce à une division récursive du plan et à une pondération de la hauteur (qui décroit à chaque itération). Ensuite le terrain sera affiché, chaque polygone ayant une couleur fonction de son altitude et de son orientation à la source lumineuse.

Le but de ce programme est donc d'afficher un terrain montagneux, "d'apparence naturelle", éclairé par le soleil.

L'utilisateur pourra choisir le nombre d'itérations dans le découpage du terrain, la position de la source lumineuse ainsi que l'influence de l'aléatoire dans la découpe du terrain. Il obtiendra un rendu direct dans le fenêtre du logiciel.

Le programme sera réalisé en OpenGL, on utilisera GLUT pour créer les fenêtres.

## *Structures de données globales :*

Ce programme nécessite deux structures :

- une structure Point3d, contenant les coordonnées d'un point dans l'espace,
- une structure Configuration, contenant toutes les options de configuration du programme (nombre d'itérations, hauteur maximal, surface de la montagne, dépendance à l'aléatoire...)

Le programme utilisera une matrice *plan* (matrice carré de taille $2^{nombre\_itérations}+1$) pour stocker le découpage aléatoires du plan.

Il y aura en plus deux variables globales –nécessaires, car elles doivent être utilisées dans des fonctions ne pouvant avoir d'arguments (cela est imposé par OpenGL). Il y aura une variable global *conf* de type Configuration, ainsi qu'une variable *laMontagne* de type Gluint, contenant la *display list*, c'est à dire les instructions nécessaires à OpenGL pour afficher la montagne.

Le logiciel pourra lire un fichier pour remplir *conf*, fichier de structure semblable à Configuration.

Ex :

LARGEUR_FENETRE 480.0
LONGUEUR_FENETRE 640.0

NB_ITERATIONS 8
HAUTEUR_MAX 15.0
COTE_MONTAGNE 15.0
POSITION_SOLEIL 0 1 1
VARIATION_POINTS_DE_SEGMENT(POURCENTAGE) 35 65
VARIATION_CENTRE_DE_GRAVITE(POURCENTAGE) 45 55

Les deux variations caractérisent le caractère aléatoire de la découpe (nul si 50 50). Elles correspondent à la distance de laquelle le nouveau point peut s'éloigner respectivement du centre du segment ou du centre de gravité.

## *Structure globale du logiciel :*

Ce logiciel comporte trois parties :

- Une partie lecture remplissant *conf* soit à partir du fichier *config*, soit si celui-ci est absent en demandant les informations à l'utilisateur.
- Une partie remplissant *plan* qui découpe une partie du plan récursivement et affecte à chaque point du plan une hauteur. C'est la seule partie réellement spécifique aux montagnes fractales.
- Une dernière partie, qui affiche la montagne grâce à OpenGL après avoir créé la *display list*. Cette partie gérera les couleurs, l'éclairement et les évènements claviers (gestion de la caméra, possibilité de générer une nouvelle montagne…).

Le *main* sera constitué exclusivement de l'appel de la fonction lecture, de l'initialisation de la fenêtre avec GLUT et des différentes fonctions de rappels (callback) ainsi que de l'appel de la fonction de traitement des événements.

# *Modules du logiciel :*

## **Partie 1 :** Lecture

### Lecture :

Elle remplit *conf* avec, le cas échéant, les données du fichier, sinon en les demandant à l'utilisateur.

**Procedure** *lecture*()

## **Partie 2 :** Découpage

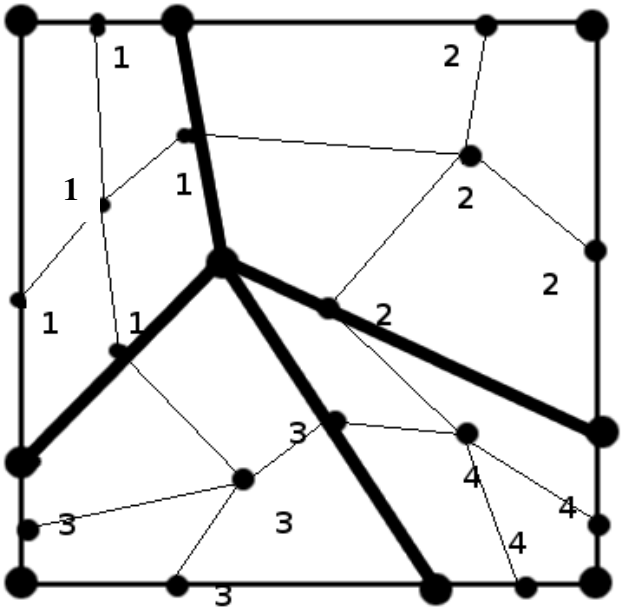
### DecoupPlan :

Fonction qui crée et initialise la matrice *plan*, puis qui appelle la fonction *decoupe*. Elle retourne *plan*.

**Fonction** *decoupPlan*()

```
Soit n
n ← (2^NB_ITERATIONS)+1

Soit  plan, matrice carré de taille n  de Point3d

plan[0][0]={X_MIN,Y_MIN,0}
plan[n-1][n-1]={X_MAX,Y_MAX,0}
plan[n-1][0]={X_MIN,Y_MAX,0}
plan[0][n-1]={X_MAX,Y_MIN,0}

decoupe(plan,0,n-1,0,n-1)

retourner plan
```

### Decoupe :




*Vu de dessus*

Fonction récursive qui prend en arguments le plan et les limites de la matrice où elle doit travailler.

Elle crée des nouveaux points en fonction de l'endroit où elle travaille : il y a une priorité sur le haut et la gauche (cf. dessin).

Pour cela elle appelle les fonctions pointDeSeg (pour les bords) et pointDeQuad (pour le centre).

Ensuite, elle calcule les hauteurs de ces points grâce aux fonctions hauteur et hauteurCentre.

Finalement, s'il reste des cases de la matrice à remplir, elle s'appelle elle-même sur chaque subdivision.

**Procedure** *decoupe*(plan, minI, maxI, minJ, maxJ, flags);

```
Soit milieuI, milieuJ
milieuI ← (maxI+minI)/2
milieuJ ← (maxJ+minJ)/2

// gestion du découpage et de la hauteur :

Si minJ==0 // on est en haut.
        plan[milieuI][minJ] ← pointDeSeg(plan[minI][minJ],plan[maxI][minJ])
        plan[milieuI][maxJ].z ← hauteur(plan[minI][maxJ].z,plan[maxI][maxJ].z)

Si minI==0 //on est à gauche.
        plan[minI][milieuJ] ← pointDeSeg(plan[minI][minJ],plan[minI][maxJ])
        plan[minI][milieuJ].z ← hauteur(plan[minI][minJ].z,plan[minI][maxJ].z)


plan[milieuI][maxJ] ← pointDeSeg(plan[minI][maxJ],plan[maxI][maxJ])
plan[maxI][milieuJ] ← pointDeSeg(plan[maxI][minJ],plan[maxI][maxJ])

plan[milieuI][milieuJ] ← pointDeQuad(plan[minI][minJ],plan[maxI][minJ],
                                     plan[maxI][maxJ],plan[minI][maxJ])

plan[milieuI][maxJ].z ← hauteur(plan[minI][maxJ].z,plan[maxI][maxJ].z)
plan[maxI][milieuJ].z ← hauteur(plan[maxI][minJ].z,plan[maxI][maxJ].z)

plan[milieuI][milieuJ].z ← hauteurCentre(plan[minI][minJ],plan[maxI][minJ],plan[maxI]
                                  [maxJ],plan[minI][maxJ])

// appels récursifs :

 si (maxI-minI)>2
        decoupe(plan,minI,milieuI,minJ,milieuJ) //partie haut gauche
        decoupe(plan,milieuI,maxI,minJ,milieuJ) //partie haut droite
        decoupe(plan,minI,milieuI,milieuJ,maxJ) //partie bas gauche
        decoupe(plan,milieuI,maxI,milieuJ,maxJ)  //partie bas droite
```

nb : Ci-dessous, une lettre majuscule représente un point.

## PointDeSeg :

Elle retourne un point situé sur le segment [A,B], à proximité du milieu du segment (décalé d'une distance aléatoire, dépendant de *conf*).

**Fonction** *pointDeSeg*(A, B)

```
Soit k, flottant aléatoire compris dans l'intervalle défini par VARIATION_POINTS_DE_SEGMENT
k ← k/100

retourner A translaté de k*AB
```

PointDeQuad :

Elle retourne un point situé à proximité du centre de gravité de ABCD, approximé par un parallélogramme, décalé d'une distance aléatoire, dépendant de *conf*.

**Fonction** *pointDeQuad*(A, B, D)

Soient k,l, flottants aléatoires compris dans l'intervalle défini par VARIATION_CENTRE_DE_GRAVITE
k ← k/100
l ← l/100

retourner A translaté de k*AB et de l*AD

Hauteur :

Retourne une hauteur z dépendant de la hauteur de A et B, de la pente de AB, de *conf* et d'une variable aléatoire.

**Fonction** *hauteur*(A,B)

Soit T la valeur du sinus de AB avec l'horizontale
Soit k, flottant aléatoire compris entre -1 et 1

Soit H = (A.z+B.z)/2+k*T

Si H<0
    H ← 0
Si H>HAUTEUR_MAX
    H ← HAUTEUR_MAX

retourner H

HauteurCentre :

Elle retourne une hauteur z qui dépend de la surface de ABCD et de la moyenne des hauteurs de A, B, C, D, de *conf* ainsi que d'une variable aléatoire k.

**Fonction** *hauteurCentre*(A, B, C, D)

Soit S l'aire de ABCD
Soit k, flottant aléatoire compris entre -1 et 1

retourner (A.z+B.z+C.z+D.z)/4 + k*sqrt(S)

## **Partie 3 :** Affichage avec OpenGL

Montagne :

Fonction appelant decoupPlan pour remplir la *display list*. Elle redécoupe chaque quadrilatère de *plan* en deux triangles, leur attribue une couleur en fonction de leur altitude et calcule leur vecteur normal, nécessaire pour la gestion de la luminosité grâce à des fonctions auxiliaires.

**Procedure** *montagne*()

VextexColor :

Associe un type de matériau (eau, plaine, roche, roche grise, neige) en fonction de la hauteur à chaque point en appelant la fonction associée.

**Procedure** *vertexColor*(A)

Initialisation :

Initialisation d'OpenGL, création de la *display list* avec appel de *montagne*. Création du "soleil".

**Procedure** *init*();

Affichage :

Affichage de la *display list*.

**Procedure** *affichage*();

Reshape :

Fonction appelée à chaque déplacement / redimensionnement de la fenêtre.

**Procedure** *reshape*(largeur, hauteur)

Gestion du clavier :

Fonction gérant les évènements claviers et permettant de demander de tracer une nouvelle montagne, de modifier *conf*, de faire pivoter la montagne selon x, y et z...

**Procedure** *clavier*(touche, positionSourisX, positionSourisY)
**Procedure** *clavierSpecial*(touche, positionSourisX, positionSourisY)

Materiaux :

Gère la façon dont réagit chaque matériau à la lumière : couleur réfléchie et reflets.

**Procedure** *eau*()
**Procedure** *plaine*()
**Procedure** *roche*()
**Procedure** *rocheGrise*()
**Procedure** *neige*()