

Wild Graphies

Numéro 4

La récursivité dans la génération d'arbres.

Mai 1994

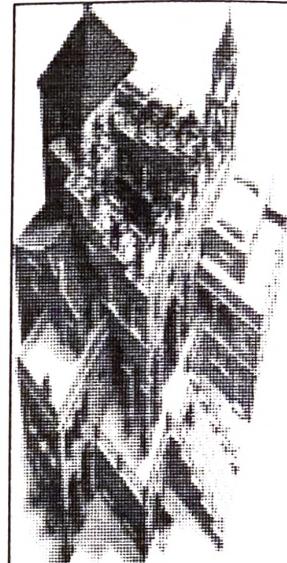
EDITO

Une idée a germé dans la tête d'un d'entre vous... Un jour, il s'est dit pourquoi pas, ...oui pourquoi pas. Et la réponse c'est vous qui lui donnerez : Pourquoi ne pas réaliser un slideshow des membres de Wild Graphics ? Celui-ci contiendrait les meilleures images de chacun à la fois dans les domaines de l'image de synthèse, des fractales, ou du dessin fait main...

Quelques personnes ont déjà répondu par l'affirmative à la proposition faite, j'attend donc l'avis de chacun maintenant que l'annonce est faite complètement par le biais de la newsletter. Ceux qui le désirent peuvent même déjà envoyer leur création. Des précisions supplémentaires devront être faites au niveau de l'enrobage du slideshow telles que la musique ou la réalisation.

Ce quatrième numéro de Wild Graphics contient un dossier sur la génération des arbres par la récursivité, un avis sur le livre 3D Computer Graphics, la découverte d'un CD ROM dédié au raytracing, et la rubrique questions/réponses.

-Nicolas Mougel



L'artiste MC Escher

LA RECURSIVITE ET LES ARBRES

ARBRE RECURSIF.

J'avais réalisé, il y a quelque temps, un algo simple, en BASIC AMOS, qui traçait un arbre récursif, mais sans faire appel à aucune fonction récursive. La méthode est assez bestiale, la voici donc : Le principe est de créer une matrice $[2^N, 2^N]$ pour un arbre à N générations et donc chaque branche a 2 subdivisions. Chaque couple [I,J] représente un embranchement (un noeud, une articulation ...)

Pour donner à l'arbre un aspect vivant on associe à chaque couple [I,J] une série d'attributs (longueur de chaque branche partant du noeud, angle...) que chaque branche va transmettre aux noeuds donc aux branches suivantes. On peut ensuite modifier ces attributs lors de la création d'un nouveau noeud. Nicolas a vu le résultat, en 2D, il pourra vous en parler.

Le principe : On balaye la matrice, de J=0 à J=N, où J représente la génération courante, et pour

chaque couple on détermine s'il appartient à l'arbre :

```
Si ( I<2^J ) /* Si [I,J] e génération J */
{
    [I,J] appartient à l'arbre;
    longueur[I,J] = k * longueur[INT(I/2),J-1];
    Si (I pair) /* si branche de gauche */
        angle[I,J] = k * angle[INT(I/2),J-1];
    Sinon /* branche de droite */
        angle[I,J] = -k * angle[INT(I/2),J-1];
}
/* INT : Partie entière (ie INT(1.5) => 1) */
```

L'idée serait bien sûr d'étendre ce principe à un générateur de scripts Real3D/POV/... Le seul problème, de taille, c'est que ma méthode ne fonctionne qu'en 2D... On peut toujours créer un attribut profond, ou rajouter un angle (!?) ... Avis aux amateurs.

François

ARBRE RECURSIF (II).

J'en profite pour compléter l'article de François en montrant comment faire l'arbre récursif en question mais en utilisant la possibilité qu'offre le langage C de faire des fonctions récursives.

Quelques commentaires sur l'arbre généré : Chaque branche fille possède au moins 50% de la longueur de sa branche mère, les 50% restants étant aléatoires. L'angle de la branche fille de gauche dépasse les 90°, celle de droite varie entre 0 et 90°. L'angle d'une branche fille récupère au moins 30% de la valeur de l'angle de la branche mère, les 30 autres % étant aléatoires et les 40% restants perdus.

```
/* Tree.c : Exemple d'utilisation de la récursivité */

```

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>

#include <intuition/intuition.h>
#include <graphics/gfxbase.h>
#include <exec/exec.h>
#include <dos/dos.h>

#ifndef PI
#define PI 3.1415926
#endif

float Randm(float);
void CreateBranch(float, float, float, float);
void OpenAll(void);
void CloseAll(char *);

struct IntuitionBase *IntuitionBase = NULL;
struct GfxBase *GfxBase = NULL;

struct Screen *MyScreen = NULL;
struct NewScreen MyNewScreen=
{ 0, 0, 640, 200, 4, 0, 1, HIRES,
CUSTOMSCREEN, NULL, "Tree", NULL,
NULL };

UWORD DriPens = ~0;

struct Window *MyWindow = NULL;
struct NewWindow MyNewWindow=
{ 0, 0, 640, 200, 0, 1, BACKDROP |
BORDERLESS, NULL, NULL,
NULL, NULL, NULL, 0, 0, 0, 0,
CUSTOMSCREEN };

main()
{
    float x=320.0, y=10.0, length=50.0,
alpha=90.0;

    OpenAll();
```

```

SetAPen(MyWindow->RPort, 1);
 srand(MyScreen->MouseX+MyScreen->MouseY);

 CreateBranch(x, y, length, alpha);

 Delay(50*3);
 CloseAll("");
}

/* Crée une Branche */

void CreateBranch(float x, float y, float length,
float alpha)
{
    if (length<=5.0)
        return;
    SetAPen(MyWindow->RPort, 1);
    Move(MyWindow->RPort, (int) x, (int) (200-y));
    x=x+length*cos((alpha*PI)/180);
    y=y+length*sin((alpha*PI)/180);
    Draw(MyWindow->RPort, (int) x, (int) (200-y));
    length= (0.5 + Randm(0.5))*length;

    /* Branche de gauche */
    CreateBranch(x, y, length, (float) (90+alpha-
(0.3+Randm(0.3))*alpha));
    /* Branche de droite */
    CreateBranch(x, y, length, (float) (alpha-
(0.3+Randm(0.3))*alpha));
}

```

```

/* Cette fonction retourne une valeur aléatoire
entre 0 et n-1. */

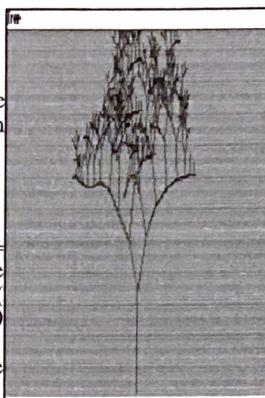
float Randm(float n)
{
    float r;
    r = ((float) rand()) / ((float) RAND_MAX );
    n *= r;

    return (n);
}

/* OpenAll() : Ouvre
bibliothèques, écran
et fenêtre */

void OpenAll(void)
{
    IntuitionBase =
(struct IntuitionBase *)
OpenLibrary(
"intuition.library", 0
);
    if (IntuitionBase
== NULL)
        CloseAll("Unable to open intuition.library.");
    GfxBase = (struct GfxBase *) OpenLibrary(
"graphics.library", 0 );
    if (GfxBase == NULL)
        CloseAll("Un able to open
graphics.library.");
    MyScreen = OpenScreenTags(
&MyNewScreen, SA_Pens, &DriPens,
SA_Depth, 4, TAG_DONE );

```



```

if (MyScreen == NULL)
    CloseAll("Unable to open the screen.");

MyNewWindow.Screen = MyScreen;
MyWindow = (struct Window *) OpenWindow(&MyNewWindow);
if (MyWindow == NULL)
    CloseAll("Enable to open the window.");
}

/* CloseAll() : Ferme fenêtre, écran et
bibliothèques */

void CloseAll(char *message)
{
    if (*message)
        printf("%s\n", message );
    if (MyWindow)
        CloseWindow( MyWindow );
    if (MyScreen)
        CloseScreen( MyScreen );
    if (GfxBase)
        CloseLibrary( (struct Library *) GfxBase );
    if (IntuitionBase)
        CloseLibrary( (struct Library *) IntuitionBase );
    Exit(0);
}

```

Si vous en avez la curiosité vous pouvez essayer d'affectioner des variations de couleurs aux branches, des feuilles aux branches terminales et de changer les caractéristiques de génération de l'arbre.

-Nicolas

QUESTIONS/ REPONSES

Question de WaiYip : Comment utilise-t-on les courbes Béziers? A quoi correspondent les 16 vecteurs? A quoi servent les valeurs de type, flatness, u_steps et v_steps? Quels sont les effets de chaque option les uns sur les autres et seul? Existe-t-il un utilitaire qui nous permet de bien les utiliser? Quel est l'indice de réfraction du verre? Je cherche aussi des informations sur les objectifs des appareils photo, comment choisir les lentilles pour avoir le même effet qu'un objectif fish-eye (épaisseur des lentilles, convergente, divergente,...)? Et s'il y a un moyen d'obtenir le même effet sur POV, comment le faire?

Remarque: Vous-avez tous remarqué, lecteurs assidus que notre cher fanzine (*NdN : Newsletter...*) n'a pas de logo, alors si vous avez une super idée, et bien faite nous la parvenir. Et si nous avons assez de réponses, nous nous aiderez à en sélectionner un sinon nous allons devoir garder celui qui est en tête de ce fanzine (hélas...).

Question de Francois pour Arnaud : A quoi ressemble le langage E ?

Question de Arnaud : Est-ce que quelqu'un connaît une formule ou un algo rapide pour passer d'une palette RGB à une palette HSV ? J'ai bien un algo mais il est si lennnnnnnnnt :-)

NEWS

3D COMPUTER GRAPHICS, by ALAN WATT:
J'ai dernièrement acheté ce bouquin qui me semblait, à juste titre, particulièrement complet et fourni concernant la synthèse d'images. En effet, bien que n'ayant parcouru le livre que très rapidement, j'y ai

trouvé moult renseignements sur toutes les techniques les plus recentes en matière de synthèse. Ainsi, lors du salon d' IMAGINA 94, Noel nous avait parlé à Nicolas et moi de la représentation en BICUBIC PATCH, technique qui est abordée dans 3DCG. Toutes les étapes de la création d'images synthétiques sont expliquées: les transformations dans l'espace (rot., bending, twist ...), les modèles d'illumination (Phong, Gouraud, Fresnet ...), les textures 3D,

l'anti_alias ou l'anti_mipmap ou l'anti_alpha ou l'anti_alpha_mipmap hiérarchisée. Ceci n'est bien sûr qu'un aperçu: le bouquin est très vaste, mais chaque notion est accompagnée d'explications pratiques (principe, formules, algorithme, photos...). Pour tout vous dire, en une matinée j'ai pu réaliser, en AMOS, un mini-ray-tracer qui génère une sphère, éclairée, bump-mappée, avec turbulence, texture, specularité et diffusion ...

3D COMPUTER GRAPHICS (2nd ED.) by ALAN WATT, ADDISON-WESLEY, ISBN 0-201-6318615

Francois

LES RENDEZ-VOUS d'IMAGINA.

Du Mercredi 13 au Dimanche 24 Avril se déroulaient à Paris, au Carré Séita, des projections de films d'image de synthèse. Les séquences provenaient du festival d'Imagina, aussi bien de cette année que des années précédentes, et étaient proposées toutes les 1h30 de 15h00 à 20h30 7j/7. L'entrée était gratuite et ouverte à tout public, autant dire que pour les passionnés d'images c'était une heureuse occasion de (re)découvrir une partie d'Imagina et une variété de création via 200 séquences projetées pendant 12 jours.

Petit plus : A l'entrée il était possible de participer à Interactive Plant Growing, une borne interactive où 5 plantes étaient présentes devant un grand écran. Le

principe : à chaque touché que vous faites sur une des plantes, un dessin équivalent au type de la plante s'inscrit au fur et à mesure sur l'écran. Retirez vos doigts et l'évolution de la plante s'arrête. On obtient ainsi au bout de 5 minutes un véritable jardin de plantes virtuelles unique. Attention, le cactus a un rôle un peu spécial... Bravo à Christa Sommerer et Laurent Mignonneau pour cette réalisation originale.

Nicolas

RAYTRACING C.D. ROM:

Après avoir lu l'article sur les petits logiciels (il en reste encore tout plein, l'article ne montre que les plus intéressants) qui nous facilite l'utilisation des utilitaires de ray-tracing qui utilisent des scripts, vous devez vos dire que se les procurer tous ou une partie de ces logiciels doit finalement coûter cher et être difficile à trouver. Heureusement que D.P. TOOL CLUB à penser à nous, cher passionné du raytracing , car il nous a concocté un super C.D. ROM qui regroupe tous les logiciels de raytracing du D.P.: Director 3D, SM Tracer, Luxart, POV v2.0 et non v2.1 comme dans la pub, Polyray, vivid... Mais aussi tous les logiciels qui les complètent : 3DS2POV, PV3D, Moray, Pdots, PovCad, PovLandscape Generator, PrePov, Raw2Pov et bien d'autre encore plus ou moins connue. Bien sur il y a des scripts, des animations (les plus volumineux) et des images de Mike Miller, Truman Brown... Et ce n'est pas encore fini, il y a encore des logiciels de traitement d'image comme Piclab, ImProcess, Image Alchey, GWS, Svga, NéoPaint..., des logiciels de morphing (Dmorph...), de fractales (Fractint...), des sources pour les programmeurs... C'est un C.D. ROM que je recommande pour tous, car il est très complet et on doit normalement trouver son bonheur (650 Mo) pour la modique somme de 139 Fr, plus les frais d'envois. Il s'agit de la deuxième édition qui prend de l'âge: 25/10/93.

WaiYip

Ont contribué à ce numéro : WaiYip Cheng, Arnaud Danassié et François Gutherz. Merci à eux.
Au prochain numéro.

**Wild Graphics, 6 avenue de la Chasse,
77500 Chelles**