ANNEXES

Bibliographie

[AA94] : ARQUÈS (D.), AUBERT (S.). - “Modélisation de scènes tri-dimensionnelles pour l’animation en synthèse d’images : notion d’arbres ECSG et compilation”, Journées AFIG, Toulouse, 1994, p. 273-284.

[AR94] : ARQUÈS (D.), RIS (Ph.). - “Méthode de parallélisation du lancer de rayon intégrant une acquisition dynamique de la connaissance topologique de la scène”, Revue Internationnale de CFAO et d’Infographie, Actes de MICAD, 1994.

[AR93] : ARQUÈS (D.), RIS (Ph.). - “Définition et implantation d'une nouvelle méthode de parallélisation appliquée au lancer de rayon”, Journées AFIG-GROPLAN, Bordeaux, 1993.

[Af91] : Association Française des Utilisateurs d’Unix. Dossier Spécial Benchmarks, mars 1991, 169 p.

[AK87] : ARVO (J.), KIRK (D.). - “Fast ray tracing by ray classification”. - Computer Graphics, 21(4), juillet 1987, p. 55-64.

[AK89] : ARVO (J.), KIRK (D.). -"A Survey Of Ray Tracing Acceleration Techniques" p. 201-262 In An Introduction To Ray Tracing / Ed. A. S. Glassner. London : Academic Press Limited, 1989. - 327 p.

[Am84] : AMANATIDES (J.). - “Ray tracing with cones”. - Computer Graphics, 18(3), 1984, p. 129-135.

[Ap68] : APPEL (A.). - "Some techniques for shading machine renderings of solids." - AFIPS, Spring Joint Computer conf. 1968, p. 32-37.

[APB87] : ARNALDI (B.), PRIOL (T.), BOUATOUCH (K.). - "A new space subdivision method for ray tracing CSG modelled scenes." - The visual computer. - Springer-Verlag, 3, 1987, p. 98-108.

[APR95] : ARQUÈS (D.), PIRANDA (B.), RIS (Ph.). - "Protocole de test pour analyser les performances en lancer de rayon et modèle statistique simplifié", Journées AFIG, Marseille, 22-24 nov. 1995, p.69-80.

[BA94] : BERGAD (K.), ATAMENIA (A.). - "Approche discrète du lancer de rayon", Actes des journées AFIG-GROPLAN, Toulouse, 30 nov. - 2 dec. 1994, p.31-42.

[BB94] : BACHE (R.), BAZZANA (G.) - "Software Metrics for product assesment". Mac Graw-Hill, Maidenhead (U.K.), 1994, 249 pages.

[BBN92] : BARRY (C. A.), BALAKRISHNAN (N.), NAGARAJA (H.N.). - "A first course in order statistics".J. Wiley & Sons, INC, New-York, 1992, 279 pages.

[BBP89] : BADOUEL (D.), BODIN (F.), PRIOL (T.). -"Lancer de rayon : approches parallèles". Publication interne n°452, IRISA RENNES, janvier 1989.

[BBP90] : BADOUEL (D.), BOUATOUCH (K.), PRIOL (T.). -"Ray tracing on distributed memory parallel computers : stategies for distributing computations and data". - Rapport de recherche n°1163, INRIA RENNES, février 1990.

[BDMV88] : BOUVILLE (C.), DUBOIS (J.L.), MARSHALL (I.), VIAUD (M.L.). - "Monté-Carlo integration applied to illumination model". Actes d'Eurographics'88, 1988.

[BERKELEY] : Cours de physique de Berkeley volume 2. - Armand Colin, Collection U. 1984.

[Bl82] : BLINN (J.F.) - "Light reflection fonctions for simulation of clouds and dusty surfaces". Siggraph'82 conference proceedings, 16(3), july 1982, p. 21-29.

[Bo79] : Encyclopédie Bordas, Sciences Pures, Bordas, 1979.

[BP90] : BADOUEL (D.), PRIOL (T.) - "Vm_pRay An Efficient Ray Tracing Algorithm On A Distributed Memory Parallel Computer". INRIA Rennes, Rapport De

Recherche n°1198.

[BS93] : BARTH (W.), STURZLINGER (W.). - "Efficient ray tracing for Bezier and B-spline surfaces". Computers & Graphic, 17(4), juillet 93, p. 423-430

[BSS93] : BLASI (P.), Le SAEC (B.), SCHLICK (C.). - "A rendering algorithm for discrete volume density objets". Actes of Eurographics93, Sept. 1993, p. 201-210.

[BWJ84] : BRONSVOORT (W.F.), VAN WIJK (J.J.), JANSEN (F.W.). "Two methods for improving the efficiency of ray casting in solid modeling". - Computer Aided Design, 16, 1984, p. 51-55.

[Ca94] : CALLET (P.). - "De l'effet de la mouture des pigments sur la couleur des poudres et des peintures", International Journal of CADCAM and computer graphics, 9(6), 1994, p. 831-845.

[CDHMS89] : CROW (F.C.), DEMOS (G.), HARDY (J.), McLAUGHLIN (J.), SIMS (K.) - "3D image synthesis on the connection machine" in Parallel processing for computer vision and display / sous la direction de DEW, EARNSHAW, HEYWODD : Addison Wesley, 1989.

[CDP95] : CAZALS (F.), DRETTAKIS (G.), PUECH (C.). "Filtering, clustering and hierarchy construction : a new solution for ray-tracing complex scenes", Actes d'Eurographics95, 14(3), 1995, p. 371-382.

[Ch92] : CHAILLOU (Ch.). - "Architectures des systèmes pour la synthèse d'images". Dunod, Paris, 192 p.

[CM89] : CHANDY (K.M.), MISRA (J.) - "Parallel program design : a foundation". Addison Wesley, 1989, 516 p.

[CT82] : COOK (R.L.), TORRANCE (K.E.). -"A réflectance model for computer graphics". - ACM trans. graph., 1, 1982, p. 7-24.

[Co86] : COOK (R.L.). "Stochastic sampling in computer graphics". ACM Trans. Graph, 5(1), janvier 1986.

[CPC84] : COOK (R.L.), PORTER (T.), CARPENTER (L.). "Distributed ray tracing". - Computer Graphics (Siggraph'84 proceedings), 18(3), juin 1984, p. 137-145.

[CS93] : COHEN (D.), SHAKED (A.). - Photo-realistic imaging of digital terrains",

Actes d'Eurographics93, Sept. 1993, p. 363-373

[CW88] : CLEARY (J.G.), WYVILL (M.). - "Analysis Of An Algorithm For Fast Ray Tracing Using Uniform Space Subdivision". The Visual Computer, 4(2), July 1988.

[CW93] : COHEN (M.F.), WALLACE (J.R.). - "Radiosity and realistic image synthesis". Londres : Academic Press limited, 1993. 381 p.

[CWBV86] : CLEARY (J.G.), WYVILL (B.), BIRTHWISTLE (G.M.), VATTI (R.) - "Multiprocessor ray tracing", Computer graphics forum, 5(1), march 1986, p. 3-12

[DK85] : DADOUN (N.), KIRKPATRICK (D.G.). "The geometry of beam tracing algorithm". - Proceedings Eurographics'85, juin 1985, p. 55-61.

[DS84] : DIPPE (M.), SWENSEN (J.). - "An adaptative subdivision algorithm and parallel architectrue for realistic image synthesis". Siggraph'84 conference proceedings, 18(3), 1984, p. 149-158.

[FDFH93] : FOLEY (J.), van DAM (A.), FEINER (S.), HUGHES (J.). - "Computer Graphics : principles and practice", Addison Wesley, 1993, 1175 p.

[FP81] : FUCHS (H.), POULTON (J.). "Pixel-planes : a VLSI oriented design for a raster graphics engine". - VLSI Design, 20(3), 1981.

[Fr85] : FRANKLIN (R.) - "Problems with raster graphics algorithms" in "Data Structures For Raster Graphics", EurographicSeminars, Springer-Verlag, 1985, p. 1-8.

[FTI86] : FUJIMOTO (A.), TANAKA (T.), IWATA (.), "ARTS : accelerated ray tracing system". - IEEE Computer Graphics Applications, 6(4), avril 1986, p. 16-26.

[GA93] : GARGANTINI (I.), ATKINSON (H.H.). - "Ray tracing an octree : numerical evaluation of the first intersection". Computer Graphics Forum, 12(4), Oct. 1993, p. 199-210.

[Gl84] : GLASSNER (A. S.). - "Space subdivision for fast ray tracing". - IEEE Computer Graphics Applications, 4(10), octobre 1984, p.15-22.

[Gl89] : An introduction to ray tracing / sous la direction de Andrew S. Glassner. - Londres : Academic press limited, 1989, 327 p.

[Go71] : GOURAUD (Henri). - "Continuous shading of curved surfaces". IEEE Transaction on computers, 20(6), 1971, p. 623-629.

[GP89] : GREEN (S.A), PADDON (D.J). - "Exploiting coherence for multiprocessor ray tracing". IEEE computer graphics and application, 9(11), nov. 1989, p. 12-26.

[GP90] : GREEN (A.), PADDON (D.J.) - "A highly flexible multiprocessor solution for ray tracing". The Visual Computer, 1990, p. 62-73.

[GS85] : GOLDSMITH (J.), SALMON (J.). - "A ray tracing system for the hypercube". - Caltech Concurrent Computing Project Memorandum HM154, California Institude of Technology, 1985.

[GS87] : GOLDSMITH (J.), SALMON (J.). - "Automatic creation of object hierarchies for ray tracing". IEEE Computer Graphics Application, 7(5), May 1987, p. 14-20.

[He86] : HERSCOVICI (A.) - "Introduction aux grands ordinateurs scientifiques". Paris : Eyrolles, 1986, 170 p.

[He89] : HECKBERT (P. S.). - "Writing a ray tracer". In An introduction to ray tracing, ed. A. S. Glassner. - Londres : Academic press limited, 1989, p. 263-293.

[HG83] : HALL (R.A.), GREENBERG (D.). - "A testbed for realistic image synthesis". - ACM trans. graph., 3, 1983, p.10-20.

[HG86] : HAINES (E.A.), GREENBERG (D.P.) - "The light buffer : a shadow testing accelerator". IEEE Comp. Graph. Appl., 7(5), Sept. 1986, p. 6-16.

[HG94] : HUMBERT (P.), GARDAN (Y.). - "Utilisation des arbres de rayons pour le rendu de séquence d'animation". Actes de Groplan 94, Toulouse, 1994, p. 261-272.

[Ka83] : KAJIYA (J.T.). - "New techniques for ray tracing procedurally defined objects" - Computer Graphics, 17(3), 1983, p. 91-102.

[KH84] : KAJIYA (J.T.), VON HERZEN (B.P.). - "Ray tracing volume densities". - Computer Graphics, 18(3), 1984, p. 129-134.

[HH84] : HECKBERT (P.S.), HANRAHAN (P.). - "Beam tracing polygonal objects". -

Computer Graphics, 18(3), juillet 1984, p. 119-127.

[Ka85] : KAPLAN (M.R.). - "Space tracing, a constant time ray tracer. State of the art in image synthesis". - Siggraph'85 course notes, 11, juillet 1985.

[Ka86] : KAJIYA (J.T.). - "The rendering equation". - Computer graphics, 20(4), Août 1986, p. 143-150.

[KE84] : KEDEM (G.), ELLIS (J.L.). "The ray casting machine". - Proceedings of ICCD'84, Ray NY, oct 1984, p.533-554.

[KH95] : KEATES (M.), HUBBOLD (R.J.). - "Interactive ray tracing on a virtual shared-memory". Computer graphics forum, 14(4), Oct. 1995, p. 189-202.

[KK86] : KAY (T.L.), KAJIYA (J.). - "Ray tracing complex scenes". Computer Graphics, 20(4), Août 1986, p. 269-278.

[KNS87] : KABAYASHI (H.), NAKAMURA (T.), SHIGEI (Y.). - "Parallel processing of an object space image synthesis using ray-tracing", The Visual Computer, 3(1), 1987, p. 12-22.

[La95] : LAOUBI (A.). - "Radiosité et lancer de rayon pour un modèle pour un modèle global d'illumination". - [1995]. - 81 f. dactyl. Rapport de DEA : Université de Marne-La-Vallée.

[LDC93] : MAUREL (H.), DUTHEN (Y.), CAUBET (R.). - "A 4D ray tracing". Proceedings of Eurographics93, Sept. 1993, p. 285-294.

[Le92] : LEFER (W.). - "Parallélisation du lancer de rayon : une solution au problème de l'équilibre de la charge". Actes de GROPLAN 92, Nantes, 1992, p. 163-170.

[LOKSO83] : NISHIMURA (H.), OHNO (H.), KAWATA (T.), SHIRAKAWA (I.), OMURA (K.). - "LINKS-1 : a parallel pipelined multimicrocomputer system for image cration." - Proceedings of th 10th symposium on computer architecture, SIGARCH, 1983, p. 387-394.

[LRU85] : LEE (M.), REDNER (R.A.), USELTON (S.P.). -"Statistically optimized sampling for distributed ray tracing.". - Computer Graphics, 19(3), juillet 1985, p. 61-67.

[Lu92] : LUCAS (M.). - "La Parallélisation De La Synthèse D'images Réalistes".

Revue Internationale De C.F.A.O. Et D'infographie, 7(1), 1992, p.41-50.

[Ma81] : MAX (N.L.). - "Vectorized procedural models for natural terrain : waves and islands in the sunset". Computer graphics, 15(3), Août 1981,p. 317-324.

[MDC93] : MAUREL (H.), DUTHEN (Y.), CAUBET (R.) - "A 4D Ray Tracing". Computer graphics forum, Actes d'Eurographics'93, Barcelone, septembre 1993, 12 (3), p. 285-294.

[MGAG93] : MICHELIN (S.), GUALTIERO (M.), ARQUES (D.), GROSSETIE (J.C.). -"Radiosity technique using a new expression of the form factor". Computer Graphics Forum, actes d'Eurograpics'93, 12 (3), 1993, p.421-432.

[Mo81] : MORAVEC (Hans P.). - "3D graphics and the wave theory". - Computer graphics, 15(3), Aug. 1981, p. 291-296.

[OM87] : OHTA (M.), MAEKAWA (M.) - "Ray coherence theorem and constant time ray tracing algorithm". Computer Grapics 1987 (Proc. Of CG International'87) , ed. T.L Kunni, 1987, p. 303-314.

[PAGM90] : PEROCHE (B.), ARGENCE (J), GHAZANFARPOUR (D.), MICHELUCCI (D.). - "La synthèse d'images". Hermes, 1990, 295 p.

[PB85] : PLUNKETT (D.J.), BAILEY (M.J.). - "The vectorization of a ray tracing algorithm for improved execution speed.". - IEEE computer graphics application, 5(8), Août 1985, p. 52-60.

[PB89] : PRIOL (T.), BOUATOUCH (K.). -"Ray tracing on parallel computers : a performance study". - Publication interne n°451, IRISA RENNES, janvier 1989, 20 p.

[Pe89] : PERLIN (K.) - "Hypertexture", SIGGRAPH'89 conference procedings, 23(3), 1989, p. 253-262.

[Ph75] : PHONG (B. T.). - "Illumination for computer generated pictures". Communication of the ACM. 18(6), juin 1975, p. 311-317.

[Pi91] : PITOT (P.). - "Conception et réalisation d'une machine parallèle dédiée à la synthèse d'images réalistes : la machine VOXAR". Thèse de doctorat, Université Paul Sabatier, Toulouse, 1991, 195 p.

[Pi94] : Piranda (B.) - "Optimisation de l'algorithme du lancer de rayon dans le

traitement de scènes définies par des arbres CSG". Rapport de DEA, Besançon, 1994, 129 p.

[Pr89] : Priol (T.). -"Lancer De Rayon Sur Des Architectures Parallèles : Étude Et Mise En Œuvre". Thèse Soutenue À L'université Rennes I, 1989, 136 p.

[Pu86] : PURGATHOFER (W.). - "A statistical method for adaptive stochastic sampling", p. 145-152 in : Proceedings of Eurographic'86 / sous la direction de A.A.G. Requicha. - ELSEVIER (North Holland), 1986.

[RA94] : RIS (Ph.), ARQUÈS (D.) - "Parallel ray tracing based upon a multilevel topological knowledge acquisition of the scene ", Computer Graphics Forum, Eurographics'94, Oslo, Septembre 1994, p. 221-232.

[Ri91] : RIS (Ph.) - "Signification des critères de performance d'un programme: étude après parallélisation d'un programme de lancer de rayon" dossier spécial benchmarks, AFUU, 1991, p. 66-79.

[Ro82] : ROTH (S.D.). - "Ray casting for modeling solids". - Computer graphics image process, 18, 1982, P. 109-144.

[Ro94] : ROST (R. A.) - "Reading the fine print : what benchmarks don't tell you". Computer Graphics Proceedings, ACM Siggraph, Orlando, July 1994, p. 497-498.

[RW80] : RUBIN (S.), WHITTED (T.). -"A three dimensional representation for fast rendering of complex scenes". - Computer graphics, 14(3), juillet 1980, p. 110-116.

[SC95] : STOLTE (N.), CAUBET (R.). - "Discrete Ray-tracing of huge voxel spaces". Actes d'Eurographics95, 14(3), 1995, p 383-394.

[SDB86] : SPEER (L.R.), DEROSE (T.D.), BARSKY (B.A.). - "A theoritical and empirical analysis of coherent ray tracing." . - Computer generated images (Proc. of graphics interface'85), 27-31 mai 1986, p. 11-25.

[Si89] : SILLION (François). - "Simulation de l'éclairage pour la synthèse d'images : réalisme et interactivité". - [1989]. - 172 f. dactyl. Thèse spécialité informatique : PARIS-SUD centre d'ORSAY.

[SS90] : SEQUIN (C.H.), SMYRL (E.K.). - "Parametrized ray-tracing3. Computer graphics, vol 23, n°3, juillet 1990, p. 308-314.

[SSS74] : SUTHERLAND (I.E.), SPROULL (R.F.), SCHUMACKER (R.A.). -"A characterization of ten hidden surface algorithms". - Computer surv., 6(1), Mars 1974, p. 1-55.

[STN87] : SHINYA (M.), TAKAHASHI (T.), NAITO (S.). -"Principles and applications of pencil tracing". - Computer graphics, 21(4), juillet 1987, p. 45-54.

[VLG93] : ROBIN (V.), GARDAN (Y.), LANUEL (Y.). - "Optimisation du tracé de rayon : une approche originale et adaptative entre le ray-tracing et le beam-tracing". Actes de MICAD'93, Paris, 1993, p. 19-39.

[Wh80] : WHITTED (T.). - "An improved illumination model for shaded display". - Communication of ACM, 23 (6), 1980, p.343-349.

[WKS86] : WYVILL (G.), KUNII (T.L.), SHIRAL (Y.). -"Space division for ray tracing in CSG". - IEEE Computer graphics applications, 6(4), avril 1986, p. 28-34.

[YMS83] : YAU (), MANN-MAY (), SRIHARI (S.N.). -"A hierarchical data structure for multi-dimensional digital images". - Communication of ACM, 26(7), juillet 1983, p. 504-515.

ANNEXE I

Exemple de programmes EMILE

I) Le presse-papiers

```
#include "global.h"
#include "creer_obj.h"
#include "objets.h"
#include "define.h"
#include "visu.h"

void initialise_var_globales(void)
{ /* definition du parallelisme : variables globales */
  ambiant.r = 20.0 ; ambiant.v = 10.0 ; ambiant.b = 20.0 ;

#ifndef SWAP_PARAL
/* version parallélisme simulé du programme */
  nb_processus = 64 ;
  nb_site=1 ;
  strcpy(nom_site[0],"graphique1.univ-mlv.fr") ;
  strcpy(repertoire[0],"/home/pri/lr/scene");
  strcpy(nom_image[0],"pressp.rgb");
  premiere_ligne[0] =   0 ;
  premiere_colonne[0] = 0 ;
  derniere_ligne[0] =   64 ;
  derniere_colonne[0] = 64 ;
#else
```

```
/* version parallélisme vrai du programme */
  nb_processus = 64 ;
  nb_site=5 ;
  /* g2 */
  strcpy(nom_site[0],"graphique2.univ-mlv.fr") ;
  strcpy(repertoire[0],"/home/pri/dg");
  strcpy(nom_image[0],"pressp0.rgb");
  premiere_ligne[0] =   0 ;   premiere_colonne[0] = 0 ;
  derniere_ligne[0] =   64 ;  derniere_colonne[0] = 12 ;
  /* g1 */
  strcpy(nom_site[1],"graphique1.univ-mlv.fr") ;
  strcpy(repertoire[1],"/home/pri/lr/paral9.1/g1");
  strcpy(nom_image[1],"pressp1.rgb");
  premiere_ligne[1] =   0 ;  premiere_colonne[1] = 12 ;
  derniere_ligne[1] =   64 ; derniere_colonne[1] = 18 ;
  /* g3 */
  strcpy(nom_site[2],"graphique3.univ-mlv.fr") ;
  strcpy(repertoire[2],"/home/pri");
  strcpy(nom_image[2],"pressp2.rgb");
  premiere_ligne[2] =   0 ;  premiere_colonne[2] = 18 ;
  derniere_ligne[2] =   64 ; derniere_colonne[2] = 35 ;
  /* g4 */
  strcpy(nom_site[3],"graphique4.univ-mlv.fr") ;
  strcpy(repertoire[3],"/home/pri");
  strcpy(nom_image[3],"pressp3.rgb");
  premiere_ligne[3] =   0 ;  premiere_colonne[3] = 35 ;
  derniere_ligne[3] =   64 ; derniere_colonne[3] = 47 ;
  /* cray */
  strcpy(nom_site[4],"cauchy.univ-mlv.fr") ;
  strcpy(repertoire[4],"/u/pri");
  strcpy(nom_image[4],"pressp4.rgb");
  premiere_ligne[4] =   0 ;  premiere_colonne[4] = 47 ;
  derniere_ligne[4] =   64 ; derniere_colonne[4] = 64 ;
#endif
}
/* sources lumineuses */
void source(double x, double y, double z, double R, double G,
double B, obj_pt lum)
```

```
{ echelle(lum, 5.1, 0.1, 0.1);  lumiere(lum, (short)true);
  icolor(lum,R,G,B);  translation(lum, x, y, z);
}

void boules(double niv, int flag, double cx, double cy, double cz,
double r, obj_pt* sc, obj_pt ob )
{ double cr, v, b, nr;

  initialise(&ob) ;
  type(ob, sphere) ;
  echelle (ob, r, r, r) ;
  translation (ob, cx, cy, cz) ;

  cr= fabs(23.0*niv + 10.0*((double) rand()) / (double) RAND_MAX);
  v = fabs(50.0+5*niv + 10.0*((double)rand())/ (double) RAND_MAX);
  b = fabs(90.0-30.0*niv+10.0*((double)rand())/(double)RAND_MAX) ;

  icolor (ob, cr, v, b);   dcolor (ob, cr*0.4, v*0.4, b*0.4);
  phong  (ob, (float)40.0); nphong (ob, (short)100);
  rcolor (ob, 80.0, 80.0, 80.0);
  ajoute (ob, sc) ;

  if( niv > 1.0 )
  { nr = r/2.5 ; niv-- ;
    if( flag != 1 ) boules( niv, 2, cx+r+nr, cy, cz, nr, sc, ob) ;
    if( flag != 2 ) boules( niv, 1, cx-r-nr, cy, cz, nr, sc, ob) ;
    if( flag != 3 ) boules( niv, 4, cx, cy+r+nr, cz, nr, sc, ob) ;
    if( flag != 4 ) boules( niv, 3, cx, cy-r-nr, cz, nr, sc, ob) ;
    if( flag != 5 ) boules( niv, 6, cx, cy, cz+r+nr, nr, sc, ob) ;
    if( flag != 6 ) boules( niv, 5, cx, cy, cz-r-nr, nr, sc, ob) ;
  }
}

/*************************************/
void main(unsigned int argc, char **argv)
{char nom_fichier[255] ;
 obj_pt scene ;
 obj_pt ecran, lum, o1 ;
```

```
 int i ;

 init_defaut() ;
 strcpy(nom_fichier, argv[1]) ;
 initialise_var_globales() ;
 initialise(&ecran) ; initialise(&lum) ; initialise(&o1) ;

 /* sphereflakes */
boules((double) 3.0, 0,(double) 0.0, (double) 0.0, (double) 0.0,
(double)4.0, &scene, o1);
 /* cube transparent englobant le sphereflakes */
 initialise(&o1) ; type(o1, cube) ;
 echelle(o1,10.0, 10.0, 10.0);
 indice_refraction(o1, (float)1.05);
 tcolor(o1, 90.0, 90.0, 90.0);
 icolor(o1, 5.0,  5.0,  5.0); dcolor(o1 ,9.0, 3.0, 3.0);
 phong(o1, (float)50.0); nphong(o1, (short)90) ;
 ajoute(o1, &scene) ;

 /* support sous le cube */
 initialise(&o1) ;  type(o1, cube) ;
 echelle(o1, 20.0, 20.0, 1.0) ;
 translation(o1, 0.0, 0.0, -11.001);
 icolor(o1, 5.0, 35.0, 15.0) ; dcolor(o1, 20.0, 24.0, 20.0);
 phong(o1, (float)50.0); nphong(o1, (short)20);
 ajoute(o1, &scene) ;

 type(lum, sphere) ;
 source(-40.0, -40.0, 40.0, 150.0, 150.0, 150.0, lum);
 ajoute(lum, &scene) ;

/* définition de la caméra */
 type(ecran, camera) ;
 rotationz(ecran, 30.0) ;
 rotationx(ecran, -45.0) ;
 translation(ecran, 35.0, -35.0, 30.0) ;
 ajoute(ecran, &scene) ;
/* nb_site est une variable globale initialisee a 1 */
```

```
 for (i=0 ; i<nb_site ; i++) sauver_scene(scene, nom_fichier, i) ;
 libere(&ecran) ; libere(&lum) ; libere(&o1) ;
 libere_scene(&scene) ;
 exit(0) ;
}
```

II) Les objets canoniques

```
#include "global.h"
#include "creer_obj.h"
#include "objets.h"
#include "define.h"
#include "visu.h"
#include <math.h>
#include <stdlib.h>

void tourne (obj_pt o, float ligne, int num, obj_pt* scene)
{ obj_pt o1 ;
  float i, k ;
  int  j, t ;

  echelle (o, 0.05, 0.05, 0.1) ;
  i = 0.0 ; j=0 ; k = (0.5 + ligne)*7.0 ;
  initialise(&o1) ; (*o1) = (*o) ;
  rotationy (o1, 67.0*sin(10.0*ligne-i*j) ) ;
  t=1 ;
  while (i<4.0)
  { icolor (o1, 25.0*i+55.0*fabs(sin(10.5*(ligne+i))),
            20.0*i+60.0*fabs(cos(33.3*(ligne*(i+1)))),
            20.0*i+60.0*fabs(sin(20.2*(ligne/(i+1.0))))  ) ;
    translation(o1, -0.45+0.3*i, 0.2, ligne );
    nphong(o1,10+50*j);
         phong(o1,10.0+45.0*fabs(sin(ligne+i+j*t)) ) ;
    dcolor (o1, 5.0*(i+1)+20.0*fabs(sin(ligne+i*j)),
            5.0*(i+1)+20.0*fabs(sin(ligne*i+j)),
            5.0*(i+1)+20.0*fabs(sin(ligne-i*j)) ) ;
    if (num+t == 16) texture(o1,2) ;
    else texture(o1, num+t) ;
```

```
    if (i==3.0) type(o1,cone) ;
    ajoute(o1, scene) ;

    i = i+1.0 ; j = 1+2 ; t = t+1 ;
    initialise(&o1) ; (*o1) = (*o) ;
    rotationz(o1, 66.0*sin(0.5*(ligne-i*j+i))) ;
    rotationy(o1, 66.0*sin((ligne*(1.0+i*j)+j))) ;
    rotationx(o1, 66.0*sin(0.2*(ligne-i-j))) ;
  }
  libere(&o1) ;
}

/*****************************************/
void main(unsigned int argc, char **argv)
{ char nom_fichier[255] ;
  obj_pt scene, ob1, ecran ;

  strcpy(nom_fichier, argv[1]) ;
  nb_processus =  64 ;
  premiere_ligne[0] =   0 ;  premiere_colonne[0] = 0 ;
  derniere_ligne[0] =   64 ; derniere_colonne[0] = 64 ;
  ambiant.r = ambiant.v = ambiant.b = 20.0 ;

  init_defaut() ;
  strcpy(nom_site[0],"graphique1") ;
  strcpy(repertoire[0],"/home/pri/lr/scene");
  strcpy(nom_image[0],"objets.rgb");

  /* position de l'oeil */
  initialise(&ecran) ; type(ecran, camera) ;
  rotationy(ecran, 45.0) ;
  translation(ecran, -0.01, -0.35, 0.001) ;
  ajoute(ecran, &scene) ;

  initialise(&ob1); type (ob1, cube); tourne (ob1,0.45,1,&scene) ;
  initialise(&ob1); type (ob1, sphere); tourne(ob1,0.15,5,&scene);
  initialise(&ob1);type(ob1,cylindre); tourne(ob1,-0.15,9,&scene);
  initialise(&ob1);type(ob1,parabole);tourne(ob1,-0.45,12,&scene);
```

```
  /*mirroir du fond*/
  initialise(&ob1) ; type(ob1, cube) ;
  echelle(ob1, 0.6, 0.01, 0.6) ;
  icolor (ob1, 10.0, 10.0, 10.0); rcolor (ob1, 90.0, 90.0, 90.0);
  translation(ob1, 0.0, 0.6, 0.0 );
  dcolor (ob1, 5.0, 5.0, 5.0) ;
  phong(ob1, 90.0) ; nphong (ob1, 150) ;
  ajoute (ob1, &scene) ;

  initialise(&ob1) ; type (ob1, sphere) ; lumiere (ob1, true) ;
  echelle (ob1, 0.0001, 0.0001, 0.0002) ;
  icolor  (ob1, 40.0, 100.0, 40.0) ;         /* vert */
  translation(ob1, 0.5, -1.0, -0.245) ;
  ajoute(ob1, &scene) ;

  initialise(&ob1) ; type (ob1, sphere) ; lumiere (ob1, true) ;
  echelle (ob1, 0.0001, 0.0001, 0.0002) ;
  icolor  (ob1, 40.0, 40.0, 100.0) ;         /* bleu */
  translation(ob1, -0.5, -1.0, 0.35) ;
  ajoute(ob1, &scene) ;

  initialise(&ob1) ; type (ob1, sphere) ; lumiere (ob1, true) ;
  echelle (ob1, 0.0001, 0.0001, 0.0002) ;
  icolor  (ob1, 100.0, 40.0, 40.0) ;         /* rouge */
  translation(ob1, -0.2, -1.0, -0.45) ;
  ajoute(ob1, &scene) ;

  sauver_scene(scene, nom_fichier, 0) ;
  libere(&ob1) ; libere(&ecran) ;
  libere_scene(&scene) ;
  exit(0) ;
}
```

III) La scène aléatoire

```
#include "global.h"
#include "creer_obj.h"
#include "objets.h"
```

```
#include "define.h"
#include "visu.h"
/*************************************/
/*    parametres statistiques         */
/*************************************/
#define ECHANTILLON          2500
#define _sqr(a) (a)*(a)

static double mon_random(double borne)
{ double alea=drand48() ;
  return( borne * alea ) ;
}

/*******************************************/
void initialise_var_globales(void)
{ /* definition du parallelisme : variables globales */
  nb_processus = 16 ;
  ambiant.r = 30.0 ; ambiant.v = 30.0 ; ambiant.b = 30.0 ;

  nb_site=1 ;
  strcpy(nom_site[0],"graphique1.univ-mlv.fr") ;
  strcpy(repertoire[0],"/home/pri/lr/scene");
  premiere_ligne[0] =   0 ;  premiere_colonne[0] = 0 ;
  derniere_ligne[0] =   16 ; derniere_colonne[0] = 16 ;
}

/*************************************/
void main(unsigned int argc, char **argv)
{ char nom_fichier[255] ;
 obj *scene, *ob1 ;
 obj *ecran ;
 int i, mon_type, tirage ;
 float dx, dy, dz ;
 short source=0 ;

 init_defaut() ; strcpy(nom_fichier, argv[1]) ;
 initialise_var_globales() ;
```

```
initialise(&ecran) ;
/******* camera *********/
type(ecran, camera) ;
echelle(ecran, 1.0, 3.0, 1.0) ;
translation(ecran, 0.0, -3.5, 0.0) ;
ajoute(ecran, &scene) ;

for (tirage=0.0 ; tirage<ECHANTILLON ; tirage++)
{ dy = 0.01+mon_random(0.15) ;
  dx = 0.01+mon_random(0.15) ;
  dz = 0.01+mon_random(0.15) ;

  initialise(&ob1) ;
  mon_type = (int) mon_random(4.0) ;
  switch(mon_type)
  { case 0 : type(ob1, cube) ; break ;
    case 1 : type(ob1, sphere) ; break ;
    case 2 : type(ob1, cylindre) ; break ;
    case 3 : type(ob1, cone) ; break ;
    default : type(ob1, parabole) ; break ;
  }
  echelle(ob1, dx, dy, dz) ;

  dy = mon_random(100.0) ;
  dx = mon_random(100.0) ;
  dz = mon_random(100.0) ;  icolor (ob1, dz, dy, dx) ;

  dy = mon_random(70.0) ;
  dx = mon_random(70.0) ;
  dz = mon_random(70.0) ;  dcolor (ob1, dz, dy, dx) ;

  dx = mon_random(100.0) ;  phong (ob1, dx) ;
  dx = mon_random(200.0) ;  nphong (ob1, (int) dx) ;

  dy = mon_random(90.0) ;
  dx = mon_random(90.0) ;
  dz = mon_random(90.0) ;
```

```
   /* lumiere, refraction */
   mon_type = (int) mon_random(16.8) ;
   switch (mon_type)
   {  case 1 : if (source<10) {lumiere(ob1,(short)1); source ++ ;}
      break ;
      case 0 : rcolor (ob1, dz, dy, dx) ; break ;
      case 2 : tcolor (ob1, dz, dy, dx) ;
               dx = 1.0+mon_random (1.5) ; indice_refraction(ob1,
dx) ;
      break ;
      case 3 : tcolor (ob1, dz, dy, dx); rcolor (ob1, dz, dy, dx);
               dx = 1.0+mon_random (1.5) ; indice_refraction(ob1,
dx) ;
      break ;
      default : break ;
   }
   /* rotation */
   mon_type = (int) mon_random(2.0) ;
   switch (mon_type)
   {  case 0 :  dx=random(90.0); dy=random(90.0); dz=random(90.0);
            rotationx(ob1,dx);     rotationy(ob1,dy);     rotationz
(ob1,dz) ;
            break ;
      default : break ;
   }

   dy = mon_random(2.0) ;
   dx = mon_random(2.0) -1.0 ;
   dz = mon_random(2.0) -1.0 ;
   translation(ob1, dx, dy, dz);

   ajoute (ob1, &scene) ;
 }

 /* nb_site est une variable globale initialisee a 1 */
 for (i=0 ; i<nb_site ; i++) sauver_scene(scene, nom_fichier, i) ;

 libere(&ecran) ;  libere(&ob1) ;
```

```
 libere_scene(&scene) ;
 exit(0) ;
}
```

Annexe II

Exemple de configuration du réseau

La configuration des machines est décrite via un petit langage. Le signe "$" indique que la ligne est en commentaire

```
$ Première partie
Topology
  $ déclaration des noeuds du réseau : 2*7 machines
  $ ligne SUD
  node=free4      ;
  node=free5      ;
  node=graphique3 ;
  node=free2      ;
  node=graphique4 ;
  node=graphique1 ;
  node=graphique2 ;
  $ ligne NORD    ;
  node=cauchy     ;
  node=free1      ;
  node=graphique0 ;
  node=onyx       ;
  node=free3      ;
  node=graphique6 ;
  node=bio1       ;
end
```

```
$ Seconde partie : on déclare les actions
begin

  $ début des déclaration concernant le premier noeud
  Computer bio1
  $  send *.c ;
  $ lors de la compilation, on envoie les fichiers *.c
  $   directory /home/pri;
  $ Le répertoire d'arrivée est /home/pri
  $  send *.h ;
  $ lors de la compilation, on envoie les fichiers *.h
  $   directory /home/pri ;
    send ./bio1/* ;
  $ le fichier executable a été stocké sous ./bio1, on l'envoi
     directory /home/pri ;
    run makefile ;
  $ on exécute le scripte qui s'appelle makefile et qui va
  $ selon les cas compiler et/ou executer le programme
  EndComputer
  $ fin des déclaration du premier noeud

  Computer local
    run make ;
    directory /home/pri/lr/paral9.1/g1 ;
  EndComputer

  Computer graphique2
  $  send *.c ;
  $   directory /home/pri/dg;
  $  send *.h ;
  $   directory /home/pri/dg ;
    send ./g2/* ;
     directory /home/pri/dg ;
    run makefile ;
  EndComputer

  Computer graphique3
```

```
$  send *.c ;
$    directory /home/pri;
$  send *.h ;
$   directory /home/pri ;
  send ./g3/* ;
    directory /home/pri ;
  run makefile ;
EndComputer

Computer graphique4
$  send *.c ;
$    directory /home/pri;
$  send *.h ;
$   directory /home/pri ;
  send ./g4/* ;
    directory /home/pri ;
   run makefile ;
EndComputer

Computer graphique0
$  send *.c ;
$    directory /home/pri;
$  send *.h ;
$   directory /home/pri ;
  send ./g0/* ;
    directory /home/pri ;
   run makefile ;
EndComputer

Computer onyx
$  send *.c ;
$    directory /usr/people2/thesard/pri ;
$  send *.h ;
$    directory /usr/people2/thesard/pri ;
  send ./f6/* ;
   directory /usr/people2/thesard/pri  ;
   run makefile ;
EndComputer
```

```
Computer graphique6
$  send *.c ;
$    directory /home/pri;
$  send *.h ;
$   directory /home/pri ;
  send ./g6/* ;
    directory /home/pri ;
   run makefile ;
EndComputer

Computer cauchy
$  send *.c ;
$    directory /u/pri;
$  send *.h ;
$   directory /u/pri ;
  send ./cray/* ;
   directory /u/pri ;
   run makefile ;
EndComputer

Computer free1
$  send *.c ;
$    directory /usr/people/pri;
$  send *.h ;
$    directory /usr/people/pri;
  send ./f1/* ;
   directory /usr/people/pri ;
   run makefile ;
EndComputer

Computer free2
$  send *.c ;
$    directory /usr/people/pri;
$  send *.h ;
$    directory /usr/people/pri;
  send ./f2/* ;
   directory /usr/people/pri ;
```

```
   run makefile ;
EndComputer

Computer free3
$  send *.c ;
$    directory /usr/people/pri;
$  send *.h ;
$    directory /usr/people/pri;
  send ./f3/* ;
   directory /usr/people/pri ;
   run makefile ;
EndComputer

Computer free4
$  send *.c ;
$    directory /usr/people/pri;
$  send *.h ;
$    directory /usr/people/pri;
  send ./f4/* ;
   directory /usr/people/pri ;
   run makefile ;
EndComputer

Computer free5
$  send *.c ;
$    directory /usr/people/pri;
$  send *.h ;
$    directory /usr/people/pri;
  send ./f5/* ;
   directory /usr/people/pri ;
   run makefile ;
EndComputer

Computer cauchy
  send ./cray/* ;
  directory /u/pri ;
  run makefile ;
EndComputer
```

```
end
```

Résumé

Cette thèse sur le lancer de rayon parallèle et l'analyse des performances se décompose en trois parties :

1) Nous présentons une formalisation du lancer de rayon à l'aide de quatre graphes : le graphe des objets réels (objets de la scène et boîtes englobantes), graphe des objets virtuels (rayons), graphe des processus (description algorithmique du programme) et le graphe des machines. Ce formalisme permet de décrire les différentes optimisations à travers la structuration des graphes. Une nouvelle taxinomie des méthodes d'optimisation tant séquentielles que parallèles est déduite.

2) Une optimisation parallèle du lancer de rayon basée sur l'exploitation d'informations topologiques déduites des calculs est présentée. Cette méthode utilise un ensemble de règles logiques agissant au niveau des processus, ainsi que des messages échangés entre ceux-ci. Ces règles optimisent principalement les calculs sur les rayons primaires mais permettent également d'optimiser ceux concernant les rayons secondaires et d'ombre grâce à l'ajustement des boîtes englobantes. Elles sont accompagnées d'une seconde série de règles simples redéfinissant la notion de cohérence ainsi que d'une troisième applicable à l'animation.

3) La dernière partie est consacrée à l'analyse des performances selon deux approches. Dans un premier temps, le lancer de rayon est analysé pour en extraire un certain nombre de paramètres caractérisant les performances calculatoires. Un protocole de tests est défini : ces tests sont indépendants de la machine et reproductibles. Dans un second temps, une version simplifiée de notre lancer de rayon est décrite mathématiquement par des moyens statistiques. Il est alors par exemple possible de donner une formule mathématique du gain de cette méthode et de rechercher grâce au calcul des variations le meilleur découpage écran maximisant ce gain. Cette approche encore limitée aux rayons primaires semble ouvrir de nouvelles pistes de recherche.

Summary

This Ph.D. thesis presents our work in parallel ray-tracing. We can distinguish three parts in this work :

1) A ray-tracing normalization based upon four graphs is presented : graph of real objects (scene objects and bounding boxes), graph of virtual objects (rays), graph of processes (algorithmical description of the ray tracer) and the graph of computer. This normalization makes the description of optimization easier and is suitable for a new classification of classical methods, as well as new and parallel methods.

2) A parallel optimization based upon dynamic acquisition of topological knowledge is presented. The method uses a set of logical rules computed by each process. The processes exchange messages that allow the reduction of ray/object intersection computation. Two other set of rules about coherence and animation are given too.

3) The last part deals with performance analysis. In a first time we analyze ray-tracing in order to extract relevant parameters. A test protocol is defined. These tests are not dependent on a computer and are repeatable. In a second time, a simple release of our ray tracer is mathematically described with statistics. It is then possible to give an equation of benefit for instance. The best tilling of screen can be found with a variation calculation of this equation . For the moment, this approach is only suitable for primary rays by it seems to open a new research domain.

Mots clés

Rendu réaliste, Lancer de rayon, Parallélisme, Calculs distribués, Analyse des performances, Modélisation statistique, Calcul des variations, Taxinomie des optimisations.