Tache 7 Partie 1

File contour.h:

...

Bezier3 approx\_bezier3(Contour c, int j1, int j2);

double distance\_point\_bezier3(Point P1, Bezier3 b3, double ti);

Contour simplification\_douglas\_peucker\_bezier3(Contour C, int j1, int j2,double d);

void create\_postscript\_contours\_bezier3(Liste\_Contours c, char \*file\_name, int hauteur, int largeur);

...

Source code de contour.c modifié :

...

void contours\_data\_bezier3(Liste\_Contours c)

{

Cellule\_Liste\_Contours \*el;

el = c.first;

int nb = 0;

int nb\_beziers = 0;

while (el != NULL)

{

nb++;

Cellule\_Liste\_Point \*e;

e = (el->data).first;

while (e != NULL)

{

e = e->suiv;

e = e->suiv;

e = e->suiv;

e = e->suiv;

nb\_beziers++;

}

el = el->suiv;

}

printf("Nombre des contours: %d\n", nb);

printf("Nombre des bezier totals: %d\n", nb\_beziers);

printf("\n");

}

...

Bezier3 approx\_bezier3(Contour c, int j1, int j2)

{

Bezier3 b3;

int n = j2 - j1;

Tableau\_Point T = sequence\_points\_liste\_vers\_tableau(c);

Point C0, C3;

C0 = T.tab[j1];

C3 = T.tab[j2];

if (n == 1)

{

Point C1, C2;

C1 = set\_point((2 \* C0.x + C3.x) / 3, (2 \* C0.y + C3.y) / 3);

C2 = set\_point((C0.x + 2 \* C3.x) / 3, (C0.y + 2 \* C3.y) / 3);

// Declaration de la courbe bezier

b3.A = C0;

b3.B = C1;

b3.C = C2;

b3.D = C3;

return b3;

}

else if (n == 2)

{

Point C1, C2, P1;

P1 = T.tab[j1 + 1];

C1 = set\_point((4 \* P1.x - C3.x) / 3, (4 \* P1.y - C3.y) / 3);

C2 = set\_point((4 \* P1.x - C0.x) / 3, (4 \* P1.y - C0.y) / 3);

// Declaration de la courbe bezier

b3.A = C0;

b3.B = C1;

b3.C = C2;

b3.D = C3;

return b3;

}

else if (n > 2)

{

double n\_double;

n\_double = (double)(n);

// Calcul a et b

double a, b, lambda;

a = (-15 \* n\_double \* n\_double \* n\_double + 5 \* n\_double \* n\_double + 2 \* n\_double + 4) / (3 \* (n\_double + 2) \* (3 \* n\_double \* n\_double + 1));

b = ((10 \* n\_double \* n\_double \* n\_double - 15 \* n\_double \* n\_double + n\_double + 2) / (3 \* (n\_double + 2) \* (3 \* n\_double \* n\_double + 1)));

lambda = (70 \* n\_double) / (3 \* (n\_double \* n\_double - 1) \* (n\_double \* n\_double - 4) \* (3 \* n\_double \* n\_double + 1));

// définir la fonction alpha(i) a faire

double x = 0.0;

double y = 0.0;

Point id;

double i\_dbl, alpha;

for (int i = 1; i < n; i++)

{

i\_dbl = (double)(i);

alpha = (6 \* i\_dbl \* i\_dbl \* i\_dbl \* i\_dbl) - (8 \* n\_double \* i\_dbl \* i\_dbl \* i\_dbl) + (6 \* i\_dbl \* i\_dbl) - (4 \* n\_double \* i\_dbl) + (n\_double \* n\_double \* n\_double \* n\_double) - (n\_double \* n\_double);

//FIXED

id = T.tab[j1 + i];

x = x + alpha \* (id.x);

y = y + alpha \* (id.y);

}

double res\_x, res\_y;

res\_x = a \* ((double)C0.x) + lambda \* x + b \* (double)(C3.x);

res\_y = a \* ((double)C0.y) + lambda \* y + b \* (double)(C3.y);

Point C1, C2;

C1 = set\_point(res\_x, res\_y);

x = 0;

y = 0;

for (int i = 1; i < n; i++)

{

i\_dbl = n\_double - (double)(i);

alpha = (6 \* i\_dbl \* i\_dbl \* i\_dbl \* i\_dbl) - (8 \* n\_double \* i\_dbl \* i\_dbl \* i\_dbl) + (6 \* i\_dbl \* i\_dbl) - (4 \* n\_double \* i\_dbl) + (n\_double \* n\_double \* n\_double \* n\_double) - (n\_double \* n\_double);

id = T.tab[j1 + i];

x = x + alpha \* ((double)id.x);

y = y + alpha \* ((double)id.y);

}

res\_x = b \* ((double)C0.x) + lambda \* x + a \* (double)(C3.x);

res\_y = b \* ((double)C0.y) + lambda \* y + a \* (double)(C3.y);

C2 = set\_point(res\_x, res\_y);

b3.A = C0;

b3.B = C1;

b3.C = C2;

b3.D = C3;

return b3;

}

else

{

printf("Error with the approximation to courbe Bezier3");

return b3;

}

}

double distance\_point\_bezier3(Point P1, Bezier3 b3, double ti)

{

double result;

Point A;

A = calcul\_ct\_bezier3(b3, ti);

result = distance(P1, A);

return result;

}

Contour simplification\_douglas\_peucker\_bezier3(Contour C, int j1, int j2, double d)

{

int n = j2 - j1;

// Creation de la courbe de Bezier

Bezier3 b3;

b3 = approx\_bezier3(C, j1, j2);

Tableau\_Point T = sequence\_points\_liste\_vers\_tableau(C);

// Variable initialisations

double distance, ti;

double max\_distance = 0; // dmax

int far\_away, j;

for (int i = j1 + 1; i < j2; i++)

{

j = i - j1;

ti = (double)(j) / (double)(n);

distance = distance\_point\_bezier3(T.tab[i], b3, ti);

if (max\_distance < distance)

{

max\_distance = distance;

far\_away = i;

}

}

if (max\_distance <= d)

{

Contour L;

L = creer\_liste\_Point\_vide();

ajouter\_element\_liste\_Point(&L, b3.A);

ajouter\_element\_liste\_Point(&L, b3.B);

ajouter\_element\_liste\_Point(&L, b3.C);

ajouter\_element\_liste\_Point(&L, b3.D);

return L;

}

else

{

Contour L1;

L1 = creer\_liste\_Point\_vide();

L1 = simplification\_douglas\_peucker\_bezier3(C, j1, far\_away, d);

Contour L2;

L2 = creer\_liste\_Point\_vide();

L2 = simplification\_douglas\_peucker\_bezier3(C, far\_away, j2, d);

return concatener\_liste\_Point(L1, L2);

}

}

void create\_postscript\_contours\_bezier3(Liste\_Contours c, char \*file\_name, int hauteur, int largeur) // Mode remplisage uniquement

{

// Extension managment

char \*no\_extension = strtok(file\_name, ".");

char \*with\_extension = malloc(strlen(no\_extension) + 4);

strcpy(with\_extension, no\_extension);

strcat(with\_extension, ".eps"); // concantenation

FILE \*fptr;

fptr = fopen(with\_extension, "w");

if (fptr == NULL)

{

printf("EPS File Error!");

exit(1);

}

fprintf(fptr, "%%!PS-Adobe-3.0 EPSF-3.0\n");

fprintf(fptr, "%%%%BoundingBox: %d %d %d %d\n", 0, 0, largeur, hauteur);

fprintf(fptr, "\n");

Cellule\_Liste\_Contours \*al;

al = c.first;

while (al != NULL)

{

Cellule\_Liste\_Point \*el;

el = (al->data).first;

Bezier3 b3;

b3.A = el->data;

el = el->suiv;

b3.B = el->data;

el = el->suiv;

b3.C = el->data;

el = el->suiv;

b3.D = el->data;

fprintf(fptr, "%.3f %.3f moveto ", b3.A.x, hauteur - b3.A.y);

fprintf(fptr, "%.3f %.3f %.3f %.3f %.3f %.3f curveto ", b3.B.x, hauteur - b3.B.y, b3.C.x, hauteur - b3.C.y, b3.D.x, hauteur - b3.D.y);

el = el->suiv;

while (el != NULL)

{

b3.A = el->data;

el = el->suiv;

b3.B = el->data;

el = el->suiv;

b3.C = el->data;

el = el->suiv;

b3.D = el->data;

fprintf(fptr, "%.3f %.3f %.3f %.3f %.3f %.3f curveto ", b3.B.x, hauteur - b3.B.y, b3.C.x, hauteur - b3.C.y, b3.D.x, hauteur - b3.D.y);

el = el->suiv;

}

fprintf(fptr, "\n 2.0 setlinewidth");

fprintf(fptr, "\n");

al = al->suiv;

}

fprintf(fptr, "fill\n");

fprintf(fptr, "\n");

fprintf(fptr, "\n");

fprintf(fptr, "showpage\n");

fclose(fptr);

return;

}

Source code de partie 2.1 (test program : test\_approx3.c) :

#include <stdint.h>

#include <string.h>

#include<stdlib.h>

#include "contour.h"

#include "image.h"

int main(int argc, char \*\*argv)

{

//Test no 11

printf("Starting Test 11\n");

printf("For n = 1\n");

Contour c;

c = creer\_liste\_Point\_vide();

int i = 0;

while (i<=1)

{

Point A;

double x, y;

printf("x pour point A:\n");

scanf("%lf", &x);

printf("y pour point A:\n");

scanf("%lf", &y);

A = set\_point(x, y);

ajouter\_element\_liste\_Point(&c,A);

printf("================================================\n");

i++;

}

Bezier3 b3;

int j1, j2;

printf("j1:\n");

scanf("%d", &j1);

printf("j2:\n");

scanf("%d", &j2);

b3 = approx\_bezier3(c, j1, j2);

printf("--------------------------------\n");

printf("C0: (%f, %f)\n", b3.A.x, b3.A.y);

printf("C1: (%f, %f)\n", b3.B.x, b3.B.y);

printf("C2: (%f, %f)\n", b3.C.x, b3.C.y);

printf("C3: (%f, %f)\n", b3.D.x, b3.D.y);

printf("\n\n");

//Test no 12

printf("Starting Test 12\n");

printf("For n = 2\n");

c = creer\_liste\_Point\_vide();

i = 0;

while (i<=2)

{

Point A;

double x, y;

printf("x pour point A:\n");

scanf("%lf", &x);

printf("y pour point A:\n");

scanf("%lf", &y);

A = set\_point(x, y);

ajouter\_element\_liste\_Point(&c,A);

printf("================================================\n");

i++;

}

printf("j1:\n");

scanf("%d", &j1);

printf("j2:\n");

scanf("%d", &j2);

b3 = approx\_bezier3(c, j1, j2);

printf("--------------------------------\n");

printf("C0: (%f, %f)\n", b3.A.x, b3.A.y);

printf("C1: (%f, %f)\n", b3.B.x, b3.B.y);

printf("C2: (%f, %f)\n", b3.C.x, b3.C.y);

printf("C3: (%f, %f)\n", b3.D.x, b3.D.y);

printf("\n\n");

//Test no 13

printf("Starting Test 13\n");

printf("For n = >=3\n");

c = creer\_liste\_Point\_vide();

i = 0;

while (i<=8)

{

Point A;

double x, y;

printf("x pour point A:\n");

scanf("%lf", &x);

printf("y pour point A:\n");

scanf("%lf", &y);

A = set\_point(x, y);

ajouter\_element\_liste\_Point(&c,A);

printf("================================================\n");

i++;

}

printf("j1:\n");

scanf("%d", &j1);

printf("j2:\n");

scanf("%d", &j2);

b3 = approx\_bezier3(c, j1, j2);

printf("--------------------------------\n");

printf("C0: (%f, %f)\n", b3.A.x, b3.A.y);

printf("C1: (%f, %f)\n", b3.B.x, b3.B.y);

printf("C2: (%f, %f)\n", b3.C.x, b3.C.y);

printf("C3: (%f, %f)\n", b3.D.x, b3.D.y);

return 0;

}

Nouveau Makefile :

#############################################################################

# Fichier Makefile

# UE MAP401 - DLST - UGA - 2022/2023

#############################################################################

# compilateur C

CC = clang

# chemin d'acces aux librairies (interfaces)

INCDIR = .

# chemin d'acces aux librairies (binaires)

LIBDIR = .

# options pour l'�dition des liens

LDOPTS = -L$(LIBDIR) -lm

# options pour la recherche des fichiers .o et .h

INCLUDEOPTS = -I$(INCDIR)

# options de compilation

COMPILOPTS = -g -Wall $(INCLUDEOPTS)

# liste des executables

EXECUTABLES = test\_image test\_geom test\_contour test\_postscript test\_mask test\_simplification test\_approx test\_degree2 test\_degree3 test\_approx3

#############################################################################

# definition des regles

#############################################################################

########################################################

# la r�gle par d�faut

all : $(EXECUTABLES)

########################################################

# regle generique :

# remplace les regles de compilation separee de la forme

# module.o : module.c module.h

# $(CC) -c $(COMPILOPTS) module.c

%.o : %.c %.h

@echo ""

@echo "---------------------------------------------"

@echo "Compilation du module "$\*

@echo "---------------------------------------------"

$(CC) -c $(COMPILOPTS) $<

########################################################

# regles explicites de compilation separee de modules

# n'ayant pas de fichier .h ET/OU dependant d'autres modules

image.o : image.c image.h types\_macros.h

@echo ""

@echo "---------------------------------------------"

@echo "Compilation du module image"

@echo "---------------------------------------------"

$(CC) -c $(COMPILOPTS) $<

test\_image.o : test\_image.c image.h types\_macros.h

@echo ""

@echo "---------------------------------------------"

@echo "Compilation du module test\_image"

@echo "---------------------------------------------"

$(CC) -c $(COMPILOPTS) $<

geom2d.o : geom2d.c geom2d.h contour.h

@echo ""

@echo "---------------------------------------------"

@echo "Compilation du geom2d"

@echo "---------------------------------------------"

$(CC) -c $(COMPILOPTS) $<

test\_geom.o : test\_geom.c geom2d.h

@echo ""

@echo "---------------------------------------------"

@echo "Compilation du module test\_geom"

@echo "---------------------------------------------"

$(CC) -c $(COMPILOPTS) $<

contour.o : contour.c contour.h image.h geom2d.h

@echo ""

@echo "---------------------------------------------"

@echo "Compilation du module contour"

@echo "---------------------------------------------"

$(CC) -c $(COMPILOPTS) $<

sequence\_point.o : sequence\_point.c sequence\_point.h geom2d.h

@echo ""

@echo "---------------------------------------------"

@echo "Compilation du module sequence\_point"

@echo "---------------------------------------------"

$(CC) -c $(COMPILOPTS) $<

test\_contour.o : test\_contour.c contour.h image.h

@echo ""

@echo "---------------------------------------------"

@echo "Compilation du module test\_contour"

@echo "---------------------------------------------"

$(CC) -c $(COMPILOPTS) $<

test\_postscript.o : test\_postscript.c contour.h image.h

@echo ""

@echo "---------------------------------------------"

@echo "Compilation du module test\_postscript"

@echo "---------------------------------------------"

$(CC) -c $(COMPILOPTS) $<

test\_mask.o : test\_mask.c contour.h image.h

@echo ""

@echo "---------------------------------------------"

@echo "Compilation du module test\_mask"

@echo "---------------------------------------------"

$(CC) -c $(COMPILOPTS) $<

test\_simplification.o : test\_simplification.c contour.h image.h

@echo ""

@echo "---------------------------------------------"

@echo "Compilation du module test\_simplification"

@echo "---------------------------------------------"

$(CC) -c $(COMPILOPTS) $<

test\_approx.o : test\_approx.c contour.h image.h

@echo ""

@echo "---------------------------------------------"

@echo "Compilation du module test\_approx"

@echo "---------------------------------------------"

$(CC) -c $(COMPILOPTS) $<

test\_approx3.o : test\_approx3.c contour.h image.h

@echo ""

@echo "---------------------------------------------"

@echo "Compilation du module test\_approx3"

@echo "---------------------------------------------"

$(CC) -c $(COMPILOPTS) $<

test\_degree2.o : test\_degree2.c contour.h image.h

@echo ""

@echo "---------------------------------------------"

@echo "Compilation du module test\_degree2"

@echo "---------------------------------------------"

$(CC) -c $(COMPILOPTS) $<

test\_degree3.o : test\_degree3.c contour.h image.h

@echo ""

@echo "---------------------------------------------"

@echo "Compilation du module test\_degree3"

@echo "---------------------------------------------"

$(CC) -c $(COMPILOPTS) $<

########################################################

# regles explicites de creation des executables

test\_image : test\_image.o image.o

@echo ""

@echo "---------------------------------------------"

@echo "Creation de l'executable "$@

@echo "---------------------------------------------"

$(CC) $^ $(LDOPTS) -o $@

test\_geom : test\_geom.o geom2d.o

@echo ""

@echo "---------------------------------------------"

@echo "Creation de l'executable "$@

@echo "---------------------------------------------"

$(CC) $^ $(LDOPTS) -o $@

test\_contour : test\_contour.o contour.o image.o geom2d.o sequence\_point.o

@echo ""

@echo "---------------------------------------------"

@echo "Creation de l'executable "$@

@echo "---------------------------------------------"

$(CC) $^ $(LDOPTS) -o $@

test\_postscript : test\_postscript.o contour.o image.o geom2d.o sequence\_point.o

@echo ""

@echo "---------------------------------------------"

@echo "Creation de l'executable "$@

@echo "---------------------------------------------"

$(CC) $^ $(LDOPTS) -o $@

test\_mask : test\_mask.o contour.o image.o geom2d.o sequence\_point.o

@echo ""

@echo "---------------------------------------------"

@echo "Creation de l'executable "$@

@echo "---------------------------------------------"

$(CC) $^ $(LDOPTS) -o $@

test\_simplification : test\_simplification.o contour.o image.o geom2d.o sequence\_point.o

@echo ""

@echo "---------------------------------------------"

@echo "Creation de l'executable "$@

@echo "---------------------------------------------"

$(CC) $^ $(LDOPTS) -o $@

test\_approx : test\_approx.o contour.o image.o geom2d.o sequence\_point.o

@echo ""

@echo "---------------------------------------------"

@echo "Creation de l'executable "$@

@echo "---------------------------------------------"

$(CC) $^ $(LDOPTS) -o $@

test\_approx3 : test\_approx3.o contour.o image.o geom2d.o sequence\_point.o

@echo ""

@echo "---------------------------------------------"

@echo "Creation de l'executable "$@

@echo "---------------------------------------------"

$(CC) $^ $(LDOPTS) -o $@

test\_degree2 : test\_degree2.o contour.o image.o geom2d.o sequence\_point.o

@echo ""

@echo "---------------------------------------------"

@echo "Creation de l'executable "$@

@echo "---------------------------------------------"

$(CC) $^ $(LDOPTS) -o $@

test\_degree3 : test\_degree3.o contour.o image.o geom2d.o sequence\_point.o

@echo ""

@echo "---------------------------------------------"

@echo "Creation de l'executable "$@

@echo "---------------------------------------------"

$(CC) $^ $(LDOPTS) -o $@

# regle pour "nettoyer" le r�pertoire

clean:

rm -fR $(EXECUTABLES) \*.o

Results of the test\_approx3:

**Starting Test 11**

For n = 1

x pour point A:

0

y pour point A:

0

================================================

x pour point A:

1

y pour point A:

0

================================================

j1:

0

j2:

1

--------------------------------

C0: (0.000000, 0.000000)

C1: (0.333333, 0.000000)

C2: (0.666667, 0.000000)

C3: (1.000000, 0.000000)

**Starting Test 12**

For n = 2

x pour point A:

0

y pour point A:

0

================================================

x pour point A:

1

y pour point A:

0

================================================

x pour point A:

1

y pour point A:

1

================================================

j1:

0

j2:

2

--------------------------------

C0: (0.000000, 0.000000)

C1: (1.000000, -0.333333)

C2: (1.333333, 0.000000)

C3: (1.000000, 1.000000)

**Starting Test 13**

For n = >=3

x pour point A:

0

y pour point A:

0

================================================

x pour point A:

1

y pour point A:

0

================================================

x pour point A:

1

y pour point A:

1

================================================

x pour point A:

1

y pour point A:

2

================================================

x pour point A:

2

y pour point A:

2

================================================

x pour point A:

3

y pour point A:

2

================================================

x pour point A:

3

y pour point A:

3

================================================

x pour point A:

4

y pour point A:

3

================================================

x pour point A:

5

y pour point A:

3

================================================

j1:

0

j2:

8

--------------------------------

C0: (0.000000, 0.000000)

C1: (1.737287, 0.929380)

C2: (1.844176, 3.489158)

C3: (5.000000, 3.000000

Partie 2.2 :

Table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Original | D=1 | D=3 | D=10 | D=30 |
|  |  |  |  |  |
| Asterix3  Nombre des contours: 32  Nombre des segments totals: 12926 | Nombre des bezier totals: 648 | Nombre des bezier totals: 242 | Nombre des bezier totals: 135 | Nombre des bezier totals: 58 |
|  |  |  |  |  |
| lettre-L-cursive  Nombre des contours: 3  Nombre des segments totals: 4228 | Nombre des bezier totals: 157 | Nombre des bezier totals: 32 | Nombre des bezier totals: 22 | Nombre des bezier totals: 13 |
|  |  |  |  |  |
| ColombesDeLaPaix  Nombre des contours: 106  Nombre des segments totals: 21764 | Nombre des bezier totals: 1155 | Nombre des bezier totals: 451 | Nombre des bezier totals: 230 | Nombre des bezier totals: 138 |