Lex et Yacc, projet d'étudiants de L3 en 2008–2009

D. Michelucci

Les étudiants auteurs de ce projet sont P. Albert et J. Aceituno. Merci à eux. Ce projet est disponible sur le site de l'UFR.

1 makefile

```
LDFLAGS=$(shell sh ./Makefile.ldflags)
CPPFLAGS=$(shell sh ./Makefile.cflags)

objects = src/scene.o src/structures.o
program = scene

all: lexou $(program)

$(program): CC = $(CXX) $(CPPFLAGS)
$(program): $(objects)
	$(CXX) $(LDFLAGS) $(objects) -o $(program)

lexou:  src/scene.lex src/scene.yacc
	cd src
	lex -osrc/lex.yy.c src/scene.lex
	yacc -o src/y.tab.c src/scene.yacc

clean:
	rm -f $(objects)
	rm -f $(program)
	rm -f src/y.tab.c src/lex.yy.c
```

2 Makefile.cflags

```
#!/bin/sh

if [ $(uname -s) = "Darwin" ]; then
        echo "-g -DMAC_OS_X";
else
        echo "-g -DLINUX";
fi
```

3 Makefile.ldflags

```
#!/bin/sh

if [ $(uname -s) = "Darwin" ]; then
        echo "-framework OpenGL -framework GLUT -ll -lc -lpthread -DMAC_OS_X";
else
        echo "-lGL -lGLU -lglut -lpthread -DLINUX";
fi
```

4 src/scene.h

```
#ifndef __SCENE_H
#define __SCENE_H

#include "structures.h"

void afficheRepere();
void* creerScene(void*);
void clavier(unsigned char touche,int x,int y);
void reshape(int x,int y);
void mousemotion(int x,int y);
void mouse(int button, int state,int x,int y);
void affichage();


float *getCaract(Figure* f, int attribut, int caracteristique);
void setCaracts(Figure *f, int attribut, float valeur, float valeur2, float valeur3);

int yylex();
int yyparse();
int yyerror(char *);
#endif
```

5 src/scene.lex

```
%option yylineno

espace          [ \t]+
chiffre          [0-9]
nombre          ({chiffre}+("."{chiffre}*)?)
lettre          [a-zA-Z]

%x              COMMENTAIRE

%%

"#"                     { BEGIN(COMMENTAIRE); }
<COMMENTAIRE>\n         { BEGIN(INITIAL); }
<COMMENTAIRE>.          { ; }

{espace}          ;

"quitter"       { return QUITTER; }
variables       { return LISTEVARIABLES; }
figures         { return LISTEFIGURES; }
afficher        { return AFFICHER; }

cube            { yylval.inb=defcube;return FORME; }
sphere          { yylval.inb=defsphere;return FORME; }
the             { yylval.inb=defthe;return FORME; }
tetraedre       { yylval.inb=deftetraedre;return FORME; }
cylindre        { yylval.inb=defcylindre;return FORME; }
variable        { return NEW_VARIABLE; }

taille          { yylval.inb=deftaille;return CARACT; }
position        { yylval.inb=defposition;return CARACT; }
rotation        { yylval.inb=defrotation;return CARACT; }
couleur         { yylval.inb=defcouleur;return CARACT; }
groupe          { return GROUPE; }
dissocier       { return DISSOCIER; }
enlever         { return ENLEVER; }

pause           { return PAUSE; }

angle           { yylval.inb=defangle;return ATTRIBUT; }
x               { yylval.inb=defx;return ATTRIBUT; }
y               { yylval.inb=defy;return ATTRIBUT; }
z               { yylval.inb=defz;return ATTRIBUT; }
r               { yylval.inb=defr;return ATTRIBUT; }
v               { yylval.inb=defv;return ATTRIBUT; }
b               { yylval.inb=defb;return ATTRIBUT; }

{lettre}({lettre}|{chiffre}|"_")*     {
    // On regarde si c'est le nom d'une figure
```

```
    Figure *f = scene.chercher(yytext);
    if(f != NULL) {
        // Ok !
        yylval.fig = f;
        return FIGURE;
    }
    else {
        Variable *v = variables.chercher(yytext);
        if(v != NULL) {
            // Ok !
            yylval.vari = v;
            return VARIABLE;
        }
        else {
            // Ce n'est ni une figure, ni une variable. C'est un nom.
            strncpy(yylval.nom, yytext, 255);
            return NOM;
        }
    }
}

{nombre}        { yylval.nb = atof(yytext); return NOMBRE; }

"++"    { return PLUPLU; }
"--"    { return MOMO; }
"+="    { return PLUSEGAL; }
"\-="   { return MOINSEGAL; }
"*="    { return MULEGAL; }
"\/="   { return DIVEGAL; }
"+"     { return PLUS; }
"\-"    { return MOINS; }
"*"     { return MUL; }
"\/"    { return DIV; }
"="     { return EGAL; }
"\^"    { return PUISS; }
"("     { return LPAREN; }
")"     { return RPAREN; }
;       { return PV; }
,       { return VIRGULE; }
"."     { return POINT; }
.       { yyerror("caractere_invalide"); }

%%
```

6 src/scene.yacc

```
%union { float nb; char nom[256]; int inb; Figure *fig; Variable *vari; float *reference; }
%token <nb> NOMBRE
%token <nom> NOM
%token <inb> CARACT
%token <inb> FORME
%token <inb> ATTRIBUT
%token <fig> FIGURE
%token <vari> VARIABLE
%token LPAREN RPAREN PV VIRGULE POINT
%token ERREUR
%token PLUS MOINS MUL DIV PUISS EGAL PLUSEGAL MOINSEGAL DIVEGAL MULEGAL
%token QUITTER LISTEVARIABLES AFFICHER LISTEFIGURES DISSOCIER GROUPE PAUSE NEW_VARIABLE ENLEVER
%token PLUPLU MOMO

/* Changement de la prÃ©cÃ©dence des opÃ©rateurs */
%left     MOINS
%left     PLUS
%left     DIV
%left     MUL
%right     PUISS

%type <nb> ligne
%type <nb> expression
%type <reference> variable

%start programme
```

```
%%

programme : programme ligne
          | ligne ;

ligne :   PV {}
    | error PV { yyerrok; }
    | DISSOCIER FIGURE PV
      {
          Figure *f = $2;
          if(f->groupe) {
              Groupe *g = (Groupe*)f;
              scene.inserer(g->f1);
              scene.inserer(g->f2);
              scene.enlever(f);
          } else yyerror("on ne dissocie qu'un groupe");
      }
    | ENLEVER FIGURE PV
      {
          Figure *f = $2;
          scene.enlever(f);
      }
    | ENLEVER VARIABLE PV
      {
          Variable *f = $2;
          variables.enlever(f);
      }
    | GROUPE NOM LPAREN FIGURE VIRGULE FIGURE RPAREN PV
      {
          Figure *f1 = $4;
          Figure *f2 = $6;
          Figure* c = new Groupe(f1,f2);
          strcpy(c->nom, $2);
          scene.inserer(c);
          scene.enlever(f1);
          scene.enlever(f2);
      }
    | NEW_VARIABLE NOM PV
      {
          Variable* v = new Variable($2, 0.0);
          variables.inserer(v);
      }
    | NEW_VARIABLE NOM EGAL expression PV
      {
          Variable* v = new Variable($2, $4);
          variables.inserer(v);
      }
    | FORME NOM PV
      {
          Figure* c;

          switch($1) {
              case defcube :  c = new Cube(); break;
              case defsphere :  c = new Sphere();  break;
              case defthe :  c = new The();  break;
              case deftetraedre :  c = new Tetraedre();  break;
              case defcylindre :  c = new Cylindre();  break;
          }

          strcpy(c->nom, $2); scene.inserer(c);
      }
    | PAUSE PV { usleep(100000); }
    | PAUSE expression PV {
        if($2 > 100) sleep((int)(floor($2/100))); else usleep(10000 * (int)floor($2)); }
    | FIGURE POINT CARACT LPAREN expression RPAREN
      {
          setCaracts($1, $3, $5, $5, $5);
      }
    |   FIGURE POINT CARACT LPAREN expression VIRGULE expression VIRGULE expression RPAREN
      {
          setCaracts($1, $3, $5, $7, $9);
      }
    |  variable EGAL expression PV { $$ = $3; *($1) = $3; printf("%f", $3); }
```

```
    |   variable PLUSEGAL expression PV { $$ = $3; *($1) += $3; printf("%f", *($1)); }
    |   variable MOINSEGAL expression PV { $$ = $3; *($1) -= $3; printf("%f", *($1)); }
    |   variable MULEGAL expression PV { $$ = $3; *($1) *= $3; printf("%f", *($1)); }
    |   variable DIVEGAL expression PV { $$ = $3; *($1) /= $3; printf("%f", *($1)); }
    |   LISTEVARIABLES PV { variables.lister(); }
    |   LISTEFIGURES PV     { scene.lister(); }
    |   QUITTER PV { exit(0); }
    |   expression PV        { $$ = $1; printf("%f\n", $$); }
    |   AFFICHER PV {
            #ifdef MAC_OS_X
                yyerror("commande indisponible sur Mac OS X");
            #endif
            #ifdef LINUX
                pthread_create(&fred, &attr, creerScene, NULL);
            #endif
        }
    ;

expression :    NOMBRE      { $$ = $1; }
    | variable PLUPLU       { $$ = (*($1))++; }
    | variable MOMO         { $$ = (*($1))--; }
    | PLUPLU variable       { $$ = ++(*($2)); }
    | MOMO variable         { $$ = --(*($2)); }
    | variable              { $$ = *($1); }
    | LPAREN expression RPAREN    { $$ = $2; }
    | expression PLUS expression  { $$ = $1 + $3; }
    | MOINS expression            { $$ = -$2; }
    | expression MOINS expression { $$ = $1 - $3; }
    | expression MUL expression   { $$ = $1 * $3; }
    | expression DIV expression   { $$ = $1 / $3; }
    | expression PUISS expression { int i, acc = 1;
            for(i=0; i<$3; i++) acc *= $1; $$ = acc; }
    ;

variable :    FIGURE POINT CARACT POINT ATTRIBUT    {
        float *v = getCaract($1, $3, $5);
        if(v != NULL) $$ = v;
        else yyerror("attribut inexistant"); }
    | VARIABLE  { $$ = &($1->valeur); }
    ;

%%
#include "lex.yy.c"

int yyerror(char *s) {
    printf("Erreur (%d) : %s.\n", yylineno, s);
    return 0;
}
```

7 src/structures.h

```
#ifndef _STRUCTURES_H
#define _STRUCTURES_H

#ifdef MAC_OS_X
#include <OpenGL/gl.h>
#include <OpenGL/glu.h>
#include <GLUT/glut.h>
#endif
#ifdef LINUX
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/freeglut.h>
#endif

enum doudli { deftaille, defposition , defrotation, defcouleur , defcube ,
deftetraedre, defcylindre, defsphere, defthe , defangle,
defx , defy , defz ,defr ,defv ,defb, defvariable };

class Couleur {
public: float r, v, b; Couleur(); Couleur(float, float, float);
        Couleur(const Couleur&); };
```

```
class Point {
public: float x, y, z; Point();
        Point(float, float, float); Point(const Point&); };

class Figure {
public: char nom[256]; Couleur couleur;
        Point position; Point taille; Point rotation; bool groupe;
        float angleRotation;
        Figure(); void applyCouleur(); void dessiner(); virtual void draw() = 0; };

class Cube : public Figure {
public: void draw(); };

class Sphere : public Figure {
private: GLUquadricObj *ptr;
public: Sphere(); void draw(); };

class Cylindre : public Figure {
private: GLUquadricObj *ptr;
public: Cylindre(); void draw(); };

class The : public Figure {
public: void draw(); };

class Tetraedre : public Figure {
public: void draw(); };

class Groupe : public Figure {
public: Figure *f1, *f2; Groupe(Figure*, Figure*); void draw(); };

class Scene {
private: Figure **listeFigures;
public: int nbFigures; Point centre; float rayon; float thedistance;
        float azimuth; float hauteur; int moderendu; int width; int height;
        Scene(); void inserer(Figure*); void dessiner(); void lister();
        void enlever(Figure*); Figure* chercher(char *nom); };

class Variable {
public:
        char nom[256]; float valeur; Variable(char nomm[256], float); };

class Variables {
public:
        Variable **listeVariables; int nbVariables; Variables(); void inserer(Variable*);
        Variable* chercher(char *); void enlever(Variable*); void lister(); };
#endif
```

8 src/structures.cpp

9 exemples/echiquier

```
cube plateau;
plateau.taille(8, 8, 0.1);
plateau.couleur(1); # blanc

# Et c'est parti pour 32 cases noires !
# On utilise le groupe pour ne pas avoir a renommer tout le temps ses variables

cube c1; c1.couleur(0); c1.taille(1, 1, 0.1); c1.position(-3.5, -3.5, 0.02);
groupe plateauEchecs(plateau, c1);
cube c1; c1.couleur(0); c1.taille(1, 1, 0.1); c1.position(-3.5, -1.5, 0.02);
groupe plateau(plateauEchecs, c1);
cube c1; c1.couleur(0); c1.taille(1, 1, 0.1); c1.position(-3.5, 0.5, 0.02);
groupe plateauEchecs(plateau, c1);
cube c1; c1.couleur(0); c1.taille(1, 1, 0.1); c1.position(-3.5, 2.5, 0.02);
groupe plateau(plateauEchecs, c1);
cube c1; c1.couleur(0); c1.taille(1, 1, 0.1); c1.position(-1.5, -3.5, 0.02);
groupe plateauEchecs(plateau, c1);
cube c1; c1.couleur(0); c1.taille(1, 1, 0.1); c1.position(-1.5, -1.5, 0.02);
groupe plateau(plateauEchecs, c1);
cube c1; c1.couleur(0); c1.taille(1, 1, 0.1); c1.position(-1.5, 0.5, 0.02);
```

```
groupe plateauEchecs(plateau, c1);
cube c1; c1.couleur(0); c1.taille(1, 1, 0.1); c1.position(-1.5, 2.5, 0.02);
groupe plateau(plateauEchecs, c1);
cube c1; c1.couleur(0); c1.taille(1, 1, 0.1); c1.position(0.5, -3.5, 0.02);
groupe plateauEchecs(plateau, c1);
cube c1; c1.couleur(0); c1.taille(1, 1, 0.1); c1.position(0.5, -1.5, 0.02);
groupe plateau(plateauEchecs, c1);
cube c1; c1.couleur(0); c1.taille(1, 1, 0.1); c1.position(0.5, 0.5, 0.02);
groupe plateauEchecs(plateau, c1);
cube c1; c1.couleur(0); c1.taille(1, 1, 0.1); c1.position(0.5, 2.5, 0.02);
groupe plateau(plateauEchecs, c1);
cube c1; c1.couleur(0); c1.taille(1, 1, 0.1); c1.position(2.5, -3.5, 0.02);
groupe plateauEchecs(plateau, c1);
cube c1; c1.couleur(0); c1.taille(1, 1, 0.1); c1.position(2.5, -1.5, 0.02);
groupe plateau(plateauEchecs, c1);
cube c1; c1.couleur(0); c1.taille(1, 1, 0.1); c1.position(2.5, 0.5, 0.02);
groupe plateauEchecs(plateau, c1);
cube c1; c1.couleur(0); c1.taille(1, 1, 0.1); c1.position(2.5, 2.5, 0.02);
groupe plateau(plateauEchecs, c1);
cube c1; c1.couleur(0); c1.taille(1, 1, 0.1); c1.position(-2.5, -2.5, 0.02);
groupe plateauEchecs(plateau, c1);
cube c1; c1.couleur(0); c1.taille(1, 1, 0.1); c1.position(-2.5, -0.5, 0.02);
groupe plateau(plateauEchecs, c1);
cube c1; c1.couleur(0); c1.taille(1, 1, 0.1); c1.position(-2.5, 1.5, 0.02);
groupe plateauEchecs(plateau, c1);
cube c1; c1.couleur(0); c1.taille(1, 1, 0.1); c1.position(-2.5, 3.5, 0.02);
groupe plateau(plateauEchecs, c1);
cube c1; c1.couleur(0); c1.taille(1, 1, 0.1); c1.position(-0.5, -2.5, 0.02);
groupe plateauEchecs(plateau, c1);
cube c1; c1.couleur(0); c1.taille(1, 1, 0.1); c1.position(-0.5, -0.5, 0.02);
groupe plateau(plateauEchecs, c1);
cube c1; c1.couleur(0); c1.taille(1, 1, 0.1); c1.position(-0.5, 1.5, 0.02);
groupe plateauEchecs(plateau, c1);
cube c1; c1.couleur(0); c1.taille(1, 1, 0.1); c1.position(-0.5, 3.5, 0.02);
groupe plateau(plateauEchecs, c1);
cube c1; c1.couleur(0); c1.taille(1, 1, 0.1); c1.position(1.5, -2.5, 0.02);
groupe plateauEchecs(plateau, c1);
cube c1; c1.couleur(0); c1.taille(1, 1, 0.1); c1.position(1.5, -0.5, 0.02);
groupe plateau(plateauEchecs, c1);
cube c1; c1.couleur(0); c1.taille(1, 1, 0.1); c1.position(1.5, 1.5, 0.02);
groupe plateauEchecs(plateau, c1);
cube c1; c1.couleur(0); c1.taille(1, 1, 0.1); c1.position(1.5, 3.5, 0.02);
groupe plateau(plateauEchecs, c1);
cube c1; c1.couleur(0); c1.taille(1, 1, 0.1); c1.position(3.5, -2.5, 0.02);
groupe plateauEchecs(plateau, c1);
cube c1; c1.couleur(0); c1.taille(1, 1, 0.1); c1.position(3.5, -0.5, 0.02);
groupe plateau(plateauEchecs, c1);
cube c1; c1.couleur(0); c1.taille(1, 1, 0.1); c1.position(3.5, 1.5, 0.02);
groupe plateauEchecs(plateau, c1);
cube c1; c1.couleur(0); c1.taille(1, 1, 0.1); c1.position(3.5, 3.5, 0.02);
groupe plateau(plateauEchecs, c1);


# Ensuite on amene des pieces futuristes
the theBlanc;
theBlanc.taille(0.4);
theBlanc.position(3.5,1.5,0.4);
theBlanc.couleur(1.0, 0.3, 0);

tetraedre tetraNoir;
tetraNoir.position(-3.5,-1.5,0.3);
tetraNoir.couleur(0.0, 0.6, 1);

sphere truc_Moche;
truc_Moche.position(0.5,-1.5,0.4);
truc_Moche.taille(0.6, 0.4, 0.4);
truc_Moche.couleur(0.6, 1, 00000);
```

10 exemples/histoire_d_une_superbanane

11 exemples/polsuper2000