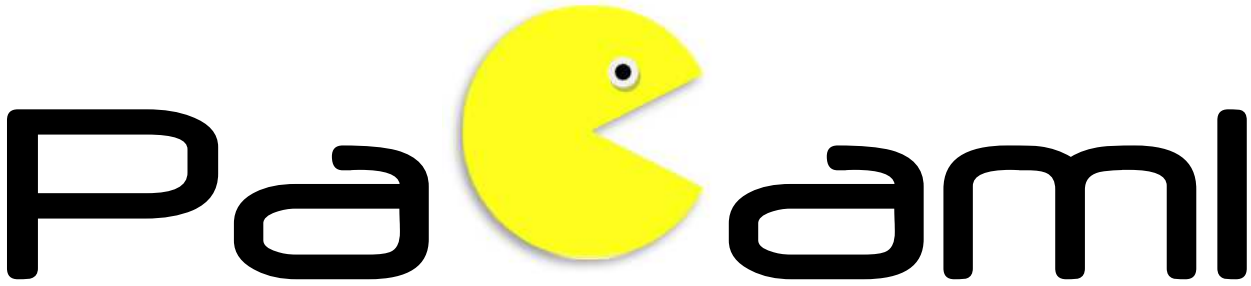


Pac-Man Game Programming Language

Final Report

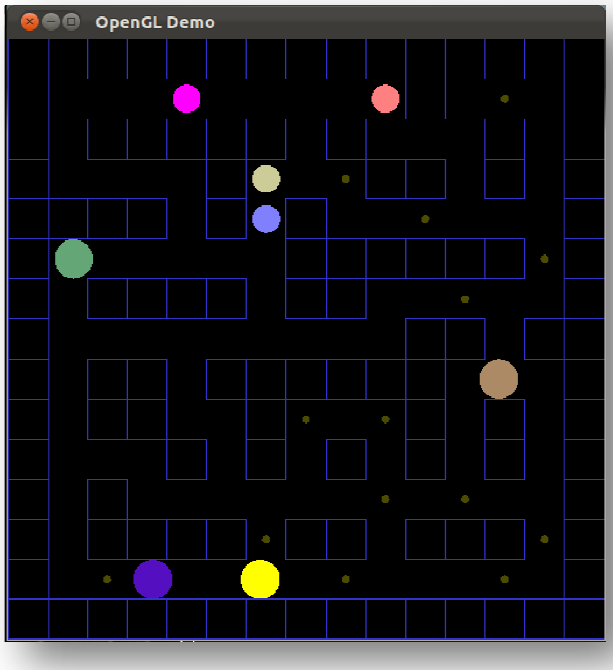
12.22.2011

Chun-Kang Chen(cc3260)

Hui-Hsiang Kuo (hk2604)

Shuwei Cao (sc3331)

Wenxin Zhu (wz2203)



Thanks to our teammates!

Contents

Chapter 1

An Introduction to PaCaml

PaCaml is a game programming language which facilitates the design of elements in PAC-MAN scene, such as maps, characters, items, and winning conditions. It is designed especially for those who have wished to be a “God” in the world of Pac-Man. PaCaml helps recall a generations of 1980’s of the best of their childhood memories.

1.1Background

PAC-MAN is an arcade game immensely popular since its original release and is considered as an icon of 1980’s popular culture [1]. It has made a great impact on a generation of people and is still appealing to the public for today [2]. However, the game scenes of each stage, the size of the map, winning condition and the number of ghosts, etc. in PAC-MAN is all pre-designed and players are unable to create or change them. It could be a great fun if the players can design a PAC-MAN based on their own favorite. Therefore, we introduce PaCaml, a PAC-MAN game design language, which enables users to create their favorite scenes and design whatever they like to make PAC-MAN more interesting.

1.2Description

The language PaCaml (PAC-MAN + Ocaml) is developed using Ocaml language. It is text-based and java like but has its own primitives. The language has a ‘main’ function like C and each statement ends with a semicolon. The basic idea and rules in PAC-MAN will not be changed. However, the language allows users to design mazes, number and distribution of eating-dots, number and speed of the ghosts, location of the power pellets, figure of each character, etc. Users can also design the condition of scoring and set the winning condition using the language.

1.3Features

Simple

We aim to design a language which can be fast and easily learned by users. As most of programmers today have dealt with Java, we design PaCaml and make it very similar to Java in syntax and expressions. We omit some data types which are not necessary in PaCaml such as any floating point related types, boolean, etc. We also add some game related functions like setMap, setPlayer which make the language easy to program.

Object-oriented

PaCaml inherits the technique of object-oriented design from Java which focuses on the data (objects) and on the interfaces to it.

Statically-scoped

PaCaml is statically-scoped which means that the environment in PaCaml which is extended (and becomes current) when a procedure is called is the environment in which the procedure was created.

Applicative-ordered

PaCaml uses applicative-order during the evaluation.

Robust

As we have said, the programmer will be the “God” in the world of Pac-Man, which means they can design the Pac-Man as they wish. Users are able to design the strategy of the ghosts, color of characters, etc.

Interesting

Our goal is to make Pac-Man programming as interesting as possible. Unlike other languages which might be tedious during programming, PaCaml is designed to be friendly interacted with programmers.

1.4A Quick Look at PaCaml Code

```
int main()
{
  /* create a barrier, a ghost, and add them to the map */
  point p;
  player pacman;
```

```
        item bar;
        item ghost;

        // add a barrier
        p = getAvailablePoint();   //randomly choose a point
        bar.i_type = _BARRIER;     //set type
        bar.i_point = p;           //set location
        addBarrier(bar);           //add to the map

        // add a ghost
        p = getAvailablePoint();   //randomly choose a point
        ghost.i_type = _GHOST;     //set type
        ghost.i_point = p;         //set location
        addBarrier(ghost);         //add to the map


        /*  create a player of this game */
        pacman.p_point = getAvailablePoint(); //randomly choose a point
        setPlayer(pac);
      }
```

Chapter 2

Language Tutorial

PaCaml is a simple game programming language with lots of funs. Users probably don't need too much time learning PaCaml if they are familiar with modern programming languages like Java or C, as PaCaml shares many of conventions with those languages. This chapter will provide users with some key features in creating and designing an interesting Pac-man game.

2.1 Sample Program

```
/* Recursive function for computing the Fibonacci numbers */

int fib( int x )
{
        if (x < 2)
       {
              return 1;
       }
       else
       {
              return fib(x-1) + fib(x-2);
       }
}


/*  Iterative function for computing the Greatest Denominator between two
integers  */

int gcd(int a, int b)
{
       while (a != b)
       {
              if(a > b)
              {
                     a = a - b;
              }
              else
              {
                     b = b - a;
              }
       }
       return a;
}
```

```
/*  The main function that is called initially whenever the program is executed
*/

item createItem( int type, int x, int y)
{
	item i;
	point p;
	p.x = x;
	p.y = y;
	i.i_type = type;
	i.i_point = p;
	return i;
}

void main()
{
	player pacman;
	item ghost1;
	item barrier1;
	item gift1;

		// initiate a pacman
	pacman.p_point = getAvailablePoint();
	setPlayer(pacman);

		// initiate a ghost
	ghost1 = createItem(_GHOST, 5, 5);
	addGhost(ghost1);

		// initiate a barrier
	barrier1 = createItem(_BARRIER, 8 , 8);
	addBarrier(barrier1);

		// initiate a gift
	gift1 = createItem(_GIFT, 8 , 9 );
	addGift(gift1);

	play();
}
```

This section gives a tutorial of how to use PaCaml.

2.2 How to Compile and Run the Sample Program

- If you are using Linux like Ubuntu, simply run the terminal window. Ocaml package is required to run PaCaml, please type "`sudo apt-get install ocaml`" to download and install the package.
- LablGL library is also required for running the game. Please type "`sudo apt-get install liblablgl-ocaml-dev`" for installation.

- Type "`make`" in the source directory to compile the sample program.
- Open a plain text editor and paste the above sample code into it. Save the file with the file name "sample.pacaml" under the same directory. Please notify that ".pacaml" is the suffix required for the PaCaml source file.
- Run the sample program by typing "`./pacmal < sample.pacaml`"

2.3 How to Write a PaCaml Source File

All PaCaml code requires a "`main`" function to call other defined or built-in functions and also establish logics and iterations. "`main`" function usually has a type of "`void`" which means the function doesn't return any result.

PaCaml is designed to be the java-like language. It heritages most of the features found in Java. "`//`" denotes the single line comment and "`/*….*/`" denotes the comment enclosed. "`;`" is used to end the statement. Blocks of statements are placed in "`{…}`".

Besides some common types like "`int`", "`string`", PaCaml has four game specified types (Map, Item, Point, Player).

Type	Properties
Map	Map is the abstract container of the game, including player and items. In other words, it is the world of Pac-Man.
Item	Roles accessed through the data type Item are ghosts, gifts, and barriers. Basic properties contained in an item are location, and image. Other important properties include type (i.e., type of the object, such as ghost, gift, and barrier), effect (which changes the status of PacMan), speed, status, move function, escape function, move counter, and maximum of move counter.
Point	Point is a 2D geometry concept, with two components, x, and y, which together depict the position of a player's or an item's.
Player	Properties are defined in Player, including position, speed, status (normal, dead, invincible, slow, fast, and fighting, and the valid remaining time of the current status (0 means forever).

An easy way to play with PaCaml is to use the built-in functions. The following table lists all the built-in functions in PaCaml.

Built-in Functions	Effect
main	Call other defined or built-in functions and also establish logics and iterations.
setPlayer	Move the player to a new point in the map. If the point is used, return error. The default status of player: speed: 5, mode: normal, color: (1.0, 1.0, 0.0)(yellow)
addGhost	Add a ghost onto a point in the map. If the point is used, return error. The default status of ghost: speed: 5, mode: easy, color: random
addBarrier	Construct barrier in the map. If the point is used, return error.
addGift	Add a new gift onto a point in the map. If the point is used, return error.
getMapWidth	Return the width of current map.
getMapHeight	Return the height of current map.
getMapItemName	Return an item on a specific location, or null if there is no item.
isPointAvailable	Check if the point of the current map is available. Return 1 if the point is available, or 0 if this point has been assigned to another item.
getAvailablePoint	Return one randomly chosen available point, or null if there is no available point on this map.
print	Print a string on standard output. Print(point) will return the coordinate of the point
play	Command to run the game

PaCaml provides a short-cut for game design. Game types and game levels are pre-defined in type of item in PaCaml for users so that users can easily set the mode and give the item its type. However, users are also able to define the type and level themselves.

Game types are defined in type of item in PaCaml.

Game Type	Effect
_GHOST	Item type set to “ghost”
_BARRIER	Item type set to “barrier”
_GIFT	Item type set to “gift”

Game levels are defined in type of item in PaCaml.

Game Level	Effect
_EASY	Easy mode for ghost
_NORMAL	Normal mode for ghost
_HARD	Hard mode for ghost
_GIFT_SPEEDUP	Pac-Man will be speeded up after eating the gift
_GIFT_SLOWDOWN	Pac-Man will be slowed down after eating the gift
_GIFT_KILLER	Pac-Man will be given the power of eating ghosts after eating the gift
_GIFT_SCORE	Score will be gained after Pac-Man eating the gift

A sample code is given below for tutorial:

```
// define type of item
item createItem( int type, int level, int x, int y)
{
	item i;
	point p;
	p.x = x;
	p.y = y;
	i.i_type = type;
	i.level = level;
	i.i_point = p;
	return i;
}

void main() {


//pre-define player, item ghost1 and gift1
	player pacman;
	item ghost1;
	item gift1;

//set player and place it on any available point on the map
	pacman.p_point = getAvailablePoint();
	setPlayer(pacman);


// add a ghost to point (5,5) and set its mode to “hard”
	ghost1 = createItem(_GHOST, _HARD, 5, 5);
	addGhost(ghost1);

// add a gift to point (8, 9) and set its function to “speed up”
	gift1 = createItem(_GIFT, _GIFT_SPEEDUP, 8 , 9 );
	addGift(gift1);
// add barrier
	addBarrier(6, 5);
	addBarrier(7, 5);

	play();
}
```

2.4 Other Functionalities

2.4.1 Functions

```
int plus( int a, int b )
```

```
{
   return a + b;
}
int minus( int a, int b)
{
   return a - b;
}
void main()
{
   print(plus( 1, 2 ));
   print(minus( 1, 2 ));
}
```

The above program shows how the function is defined in PaCaml. Here we have defined two functions “plus” and “minus” which execute the corresponding mathematical operations respectively.

2.4.2 Arithmetic Functionality

```
/*    Test the integer arithmetic functionality*/

void main()
{
    //Test point arithmetic
    point p1;
    point p2;
    p1.x = 4;
    p1.y = 3;
    p2.x = 1;
    p2.y = 2;


    print("Plus");
    print(p1+p2);
    print("Minus");
    print(p1-p2);

    print("Multiply");
    print(p1*p2);

    print("Divide");
    print(p1/p2);
    //Test integer arithmetic
    print("Integer");
    print( 1 + 5 );
    print( 5 - 2 );
    print( 4 * 2 );
    print( 1 / 2 );
    print( 5 % 3 );

    //Order of operations with integer arithmetic
    print( 1 + 2 * 3 + 4 );
    print( ( 1 + 2 ) * ( 3 + 4 ) );
}
```

The above program shows the arithmetic functionality in PaCaml. Users can try the program to check the results. As PaCaml doesn't have floating type, decimal values are not supported in the calculation. Only integers are returned.

2.4.3 Assignment

```
/* Test the functionality of assignment */
void main()
{
   int a;
   int b;
   int c;
   point p;
   a = 1;

   //Assigning variables
   print(a);
   print(b=a);

   //Assigning members
   p.x = 6;
   p.y = 2;
   print(p.x);
   print(p.y);

   //Successive assignment
   print(a = b = c = p.x = p.y = 13);
   print(a);
   print(b);
   print(c);
   print(p.x);
   print(p.y);

   //Testing additional assignment operators
   print(a += 1);
   print(a++); //++ is equivalent to x += 1;
   print(a -= 1);
   print(a--); //-- is equivalent to x -= 1;
   print(a *= 10);
   print(a /= 2);
}
```

The above program shows the assignment functionality in PaCaml. Users can try the program to check the results. Assignment is mainly used when placing points in PaCaml.

2.4.4 Recursive

```
/* Test recursive  */
int fib( int x )
{
    if (x < 2)
    {
        return 1;
    }
    else
    {
        return fib(x-1) + fib(x-2);
    }
}
```

```
void main()
{
   print(fib(0));
   print(fib(1));
   print(fib(2));
   print(fib(3));
   print(fib(4));
   print(fib(5));
}
```

The above program shows the recursive functionality in PaCaml. Users can try the program to check the results.

2.4.5 Loop

```
/*    Testing the for statement in m. */

void main()
{
      int i;

      for( i = 0; i < 10; i++)
      {
            print(i);
      }
}
```

```
/*    Test the while statement. */

void main()
{
      int i;
      i = 0;
      while( i < 10 )
      {
            print(i);
            i++;
      }
}
```

The above programs show the loop functionality in PaCaml. The first program is the "for" loop and the second one is the "while" loop. Users can try the program to check the results.

2.4.6 Logic

```
/*    Test the logic functionality  */

void main()
{

      print("NOT");
      print( !(1==1) );
      print( !(1==2) );

      print("AND");
      print( (1==1) && (1==1) );   // ( true && true)
```

```
            print( (1==1) && (1==2) );   // ( true && false)
            print( (1==2) && (1==1) );   // ( false && true)
            print( (1==2) && (1==2) );   // ( false && false)
            print("OR");
            print( (1==1) || (1==1) );   // ( true || true)
            print( (1==1) || (1==2) );   // ( true || false)
            print( (1==2) || (1==1) );   // ( false || true)
            print( (1==2) || (1==2) );   // ( false || false)

            print("NotEqual");
            print( 1 != 2 );
            print( 1 != 1 );
            print("less");
            print( 1 < 2 );
            print( 2 < 1 );
            print("LessEqual");
            print( 1 <= 2 );
            print( 1 <= 1 );
            print( 2 <= 1 );
            print("Greater");
            print( 1 > 2 );
            print( 2 > 1 );
            print("GreaterEqual");
            print( 1 >= 2 );
            print( 1 >= 1 );
            print( 2 >= 1 );

            print("Order");
            print( !(1==1) || (1==1) );
            print( !( (1==1) || (1==1) ) );
      }
```

The above program shows the logic functionality in PaCaml. PaCaml supports all the logics defined in Java. Users can try the program to check the results.

2.4.7 Condition

```
      /*    Test the if statement */

      void main()
      {
            if(1)
            {
                  print( 0 );
            }

            if(0)
            {
                  print( 1 );
            }
            else
            {
                  print( 2 );
            }
      }
```

The above program shows the condition functionality in PaCaml. Users can try the program to check the results.

Chapter 3

Language Reference Manual

3.1 Lexical conventions

3.1.1 Comments

A block of comment in PaCaml is delimited by "/*" and "*/". Single-line specific format, i.e., "//", is also provided. Identifying a comment block is a non-greedy operation. In other words, the first "*/" found after "/*" terminates the comment block.

No other constraint is given, considering characters and positions of comments.

Example:

Single-line comment:

```
// This line is not parsed by the interpreter
```

Block of comment:

```
/* These lines
are not to be parsed by
the PaCaml interpreter */
```

3.1.2 Naming

(1) Variable names are case-sensitive. A variable's name can be any legal identifier — an unlimited-length sequence of Unicode letters and digits, beginning with a letter or the underscore character "_".

(2) Subsequent characters may be letters, digits, or underscore characters.

(3) No duplicate names are allowed in the fields including variables and functions.

(4) Keywords (defined in 2.3) are reserved.

3.1.3 Keywords

Keywords are reserved identifiers. The following are the keywords in PaCaml:

int, char, string, Map, Player, Item, Barrier, Gift, Ghost, Color

return, continue, break, if, else, do, while, for

pre-defined functions

3.2 Data Types

Two groups of data types are defined in PaCaml. Primitive data types include integer, character, string, and etc. Game specific data types include map, item, and etc. These two groups are introduced in section 3.1 and section 3.2 respectively.

PaCaml is statically-typed, which means that all variables must first be declared before they can be used.

3.2.1 Primitive Data Types

(1) Integer (int)

The int data type is a 32-bit signed numeric variant. Negative numbers are signified by the minus sign ('-'), while no white space is allowed between the minus sign and the integral number.

Please notice that a point ('.') is prohibited in integer declaration, even if the value equals to an integer. In other words, 1 is a legal declaration while 1.0 and 1. are illegal. Default value given to an uninitialized variable is 0.

Examples:

```
0, 2, 256, -102
int a;
a = 10;
```

In addition to numeric variant, an integer can be used as a presentation of a Boolean value. As convention, 1 stands for true and 0 (or null) stands for false.

Example:

```
int flag;
flag = 1;
if (flag) { … }
```

3.2.2 Derived Data Types

(1)String

Array of characters is denoted as string. It is defined by double quotes “”.

Example:

```
“abcd”, “aAbB”
```

Declaration:

```
String str;
str = “abcd”;
```

3.2.3 Game-specific Types

(1)Map

Map is the abstract container of the game, including player and items. In other words, it is the world of Pac-Man.

(2) Player

Pac-Man is the only object accessed through the data type Player.

Properties are defined in Player, including location, speed, status (normal, dead, invincible, slow, and fast, and the valid remaining time of the current status (0 means forever). However, users are only allowed to set the value of a player’s location.

Example:

```
player pacman;
point p;
p.x = 10;
p.y = 4;
pacman.p_point  =  p;   // put the pacman on (10,4)
```

(3) Item

Roles accessed through the data type Item are ghosts, gifts, and barriers.

Basic properties contained in an item are location, color. Other important properties include type (i.e., type of the object, such as ghost, gift, and barrier), effect (which changes the status of PacMan), speed, status, duration, move direction, difficulty level, move counter.

The difficulty level is used to determine what move function a ghost uses. There are three levels: _EASY, _NORMAL, and _HARD. The EASY level simply uses random function to decide the next move; the Normal level and HARD level calculate the distance between the PacMan and the ghosts.

There are four types of gifts in this game: SPEEDUP, SLOWDOWN, KILLER, and SCORE. Once the PacMan eats the gift, the status of the PacMan will be changed, and will last the value of duration.

	Ice pellet (SLOWDOWN)	Slow Down
	Fire pellet (SPEEDUP)	Speed up
	Power pellet (KILLER)	Allow Pacman can eat the ghosts
	Score pellet (SCORE)	Get more score than small dots

Example:

```
/*  Set type of an item */
item i;
i.i_type = _GHOST;      // set this item as a ghost
i.i_type = _BARRIER;  // set this item as a barrier
i.i_type = _GIFT;         //  set this item as a gift

/*   set the location of this item  */
point p;
p.x = 5;
p.y = 5;
i.i_point  =  p;   // put the item  on (5,5)

/*   Set the level of a ghost. This can only be used when the item is a ghost  */
i.level = _EASY;   //set the level to EASY
```

i.level = _NORMAL; //set the level to NORMAL
i.level = _HARD; //set the level to HARD

/* Set the type of a gift. This can only be used when the item is a gift */
i.level = _SPEEDUP; //set the type of the gift to SPEEDUP
i.level = _SLOWDOWN; //set the type of the gift to SLOWDOWN
i.level = _KILLER; //set the type of the gift to KILLER
i.level = _SCORE; //set the type of the gift to SCORE

(4) Point

Point is a 2D geometry concept, with two components, x, and y, which together depict the position of a player's or an item's.

Example:

```
point p;
p.x = 5;
p.y = 7;
```

3.3 Expressions

3.3.1 Unary operators

(1) – expression:

Numeric negation

(1) ! expression:

Logical negation

3.3.2 Additive operators

(1) expression + expression:

Numeric addition

(2) expression += expression

Numeric addition

(3) expression – expression:

Numeric subtraction

(4) expression – =expression:

Numeric subtraction

3.3.3 Multiplicative operators

(1) expression * expression:

(i) Numeric multiplication;

(ii) Expression of identifier * number creates the same identifier with different

(iii) index by n times and new identifiers inherit all the attributes. e.g. Copy the ghosts.

(2) expression / expression:

Numeric division

3.3.4 Relational operators

(1)expression < expression:

(i) Less-than;

(ii) Comparison of properties of two objects. The value is 1 if expr1 is less powerful than expr2. An example is given as:

```
 Player p;
Item i;
if (p.speed < item.speed) { … }
```

According to the default definition, slow < normal < fast, while weak < normal < aggressive < invincible.

(2) expression > expression:

(i) Greater-than;

(ii) Comparison of properties of two objects. The value is 1 if expr1 is more powerful than expr2.

(3) expression <= expression:

Similar to 4.4.1, except that comparison of properties also returns 1 if expr1 == expr2 (i.e., the two properties are of the same level. For details, please refer to 4.5.1).

(4) expression >= expression:

Similar to 4.4.12 except that comparison of properties also returns 1 if expr1 == expr2 (i.e., the two properties are of the same level. For details, please refer to 4.5.1).

3.3.5 Equality operators

(1)expression == expression:

(i) Equal-to

(ii) 1 if the two properties are of the same level, and 0 otherwise.

(2)expression != expression:

(i) Not-equal-to

(ii) 1 if the two properties are not of the same level, and 0 otherwise.

3.3.6 expression && expression:

Return 1 if and only if both values are 1; otherwise it return 0.

3.3.7 expression || expression:

Return 1 if at least one of the two values are 1; otherwise it returns 0.

3.4 Declarations

Declaration creates new variables with a specified type and an optional values by the following forms:

type_specifier identifier
identifier = initialization expression

where type_specifier is one of the following type: int, string, player, item, and point.

Example:

```
int num;
num = 14;
```

3.5 Statements and Blocks

3.5.1 Expression statement

Expression statement is a construct consisting of variables, operators and functions. It is like real word statements.

Examples:

```
int i;
i = 3;
point p;
p.x + i;
```

3.5.2 Conditional statement

Same as Java or C, conditional statement executes a certain code only if a particular condition is true.

Examples:

(1)

```
if (expression)
   {statement}
else
   {statement}
```

(2)

```
if (expression)
    {statement}
elseif (expression)
    {statement}
else
    {statement}
```

3.5.3 While statement

The while statement continually executes a block of statement while the condition is true.

Example:

```
while (expression)
{ statement }
```

3.5.4 Blocks

Block is a group of statements surrounded by a pair of braces. It is used in flow control sentences.

Example:

```
while (expression1)
{
    if (expression2)
       {statement1}

    else

        {statement2}

}
```

3.5.5 Return statement

A function returns to its caller by means of the return statement, which has the following form:

```
return ( expression ) ;
```

3.6 Function Definitions

3.6.1 Basic Functions

func return_data_type function_name (arg1,arg2, arg3)

```
{
    /*
    statement;
    */
}
```

3.6.2 Built-in Functions

(1)main

```
int main()
```

This function is the entry point for execution of the program. Each PaCaml program must define this function. A return value of zero indicates failure; otherwise, success.

(2)setPlayer

```
int setPlayer(Player player)
```

Set the PacMan of this game.

(3)addGhost

```
int addGhost (Item item)
```

Add user-defined ghost object to the current map. A return value of zero indicates failure; otherwise, success.

(4)addGift

```
int addGift(Item item)
```

Add a gift to the map. A return value of zero indicates failure; otherwise, success.

(5)addBarrier

```
int addBarrier(Item item)
```

Add a barrier to a specific location of the map. A return value of zero indicates failure; otherwise, success.

```
int addBarrier(int x, int y)
```

Add a barrier to [x, y] of the map. A return value of zero indicates failure; otherwise, success.

(6) getMapWidth

```
int getMapWidth()
```

Return the current width of the map.

(7) getMapHeight

```
int getMapHeight()
```

Return the current width of this map.

(8)getMapItemName

```
String getMapItemName(Point)
```

Return an item name on a specific location, or empty string if there is no item.

(9) isPointAvailable

```
int isPointAvailable(Point)
```

Check if the point of the current map is available.
Return 1 if the point is available, or 0 if this point has been assigned to another item.

(10) getAvailablePoint

```
point getAvailablePoint()
```

Return one randomly chosen available point, or null if there is no available point on this map.

(11)print

```
void print(String)
```

Print a string on standard output.

(12) play

```
void play()
```

Start to play Pac-Man.

Chapter 4

Project Plan

4.1 Project Management Process

Our team followed a combinational methodology of conventional project management approach and extreme project methodology when developing PaCaml. After the project was released, we held our first meeting to analyze the project requirement, brainstorm the project topics and achieve an initial plan. Although our initial project was not considered by the instructor and TA as the one matching the requirements very well, we revaluated the whole project and came up with a new proposal very soon.

We held one-hour weekly meeting to identify the project progress and assign new tasks according to the time line for the coming week. Three hours each week, usually on Tuesday, were reserved for group working and brainstorming. Email was the essential method for daily communication. We also used Dropbox to share valuable materials and references. Google code was implemented as the version control system.

A simple 'Hello World' programming task was assigned to every team member so that each one can get easily started and understand how Ocaml works. Concrete tasks were assigned to each group member each week according to each one's specialty and workload.

Testing was conducted along with the coding work at each stage of the project so that errors were caught at the early stage. After each part of the code was completed, each member was in charge of the corresponding testing for the whole system.

4.2 Programming Style Guide

- Code specified by the member itself needs to be commented (* comment *)
- Neat and clean coding style is required. i.e. indention, bracing, etc.
- Use the same text editor to make sure indention or word wrap will not be mixed up

- Update the code regularly with the whole team on SVN. Remember to give information of updated files
- Commit the code on SVN without any bugs or conflicts
- Filename should be consistent with the function of the code and named by lowercase letters, with spaces replaced by underscores (i.e. compile_and_run.sh)
- Variables and function names need to be clearly defined using conventional and proper name
- Ocaml source file should be consistent with sample given in lecture notes.
- All codes related with language should be stored in the directory named 'lang' and other codes related with graphics should be stored in the directory named 'show'

4.3 Project Time Management

4.3.1 Events List

Event No.	Event Name	Estimated Due Date
1	Team formation and brainstorming	13/09/2011
2	Learning Ocaml	31/10/2011
3	Project proposal (topic, title, goal, etc.)	28/09/2011
4	Language reference manual	31/10/2011
5	Language design (ast, scanner, parser, lexer, etc.)	22/11/2011
6	Graphics design (interface)	22/11/2011
7	Code merging and debugging	29/11/2011
8	Testing	06/12/2011
9	Presentation	22/12/2011
10	Final report	22/12/2011

4.3.1 Events and Time Estimation

Event ID	Predecessor ID	Time Estimates			Expected Time
		Earliest Time	Normal Time	Latest Time	
1	-	5	6	7	6
2	-	30	40	42	38.7
3	1	10	13	15	12.8
4	2	25	30	33	29.7
5	3	18	20	22	20
6	3	18	20	22	20
7	4,5	5	6	7	6
8	6	5	6	7	6
9	7	1	1	2	1.2
10	7	3	4	5	4

*Number represents for days.

*Expected time= (Earliest time + 4 × Normal time + Latest time)/6

4.3.2 Gantt Chart

ID	Task Name	Start	Finish	Duration	2011
					SEP OCT NOV DEC
1	Team formation and brainstorming	07/09/11	13/09/11	7	
2	Learning Ocaml	21/09/11	31/10/11	40	
3	Project proposal	20/09/09	28/09/11	9	
4	Language reference manual	29/09/11	31/10/11	8	
5	Language design	01/11/11	23/11/11	23	
6	Graphics design	01/11/11	22/11/11	22	

7	Code merging and debugging	01/12/11	06/12/11	7	
8	Testing	13/12/11	19/12/11	7	
9	Presentation	21/12/11	22/12/11	2	
10	Final report	12/12/11	22/12/11	11	

4.4 Roles and Responsibilities

Name	Roles and Responsibilities
Chun-Kang Chen	Responsible for game graphics, testing
Hui-Hsiang Kuo	Responsible for interpreter, testing
Wenxin Zhu	Team leader. Responsible for parser and scanner
Shuwei Cao	Responsible for parser and scanner in corporation with Wenxin Zhu

4.5 Software Development Environment

4.5.1 Operating System

Linux was used as the operating system of the project including the language and graphics part. However, the project has been tested both on Windows and Linux.

4.5.2 Version Control System

Google code project hosting server was used as the version control system to update and back up all the source codes.

4.5.3 Language Used

Language was developed using Ocaml. Ocaml OpenGL was implemented to develop graphic part of the language.

4.5.4 File Sharing

Dropbox was used for sharing files between team members. All documents including LRM, Report, etc were uploaded onto the cloud. Google documents were also implemented for editing the files online and brainstorming.

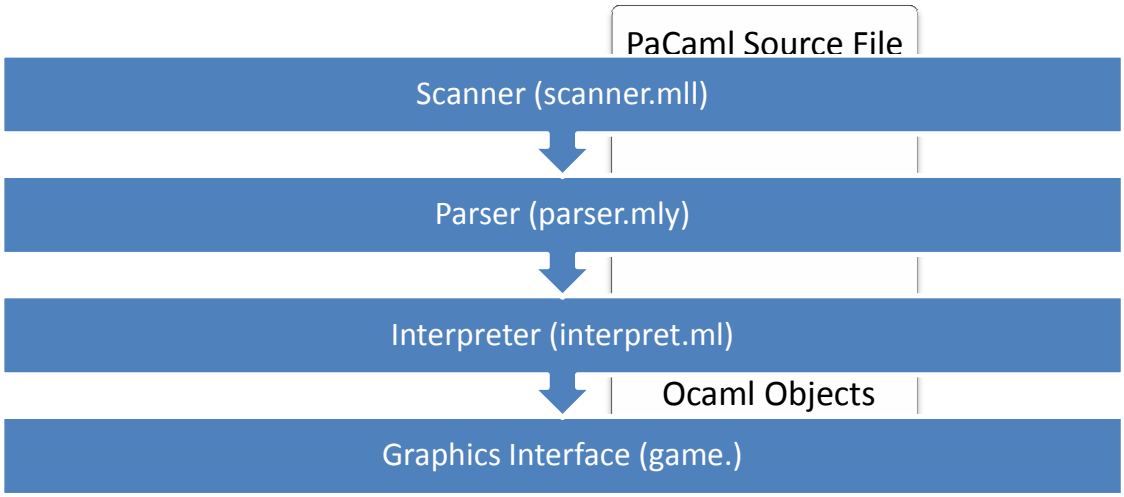
4.6 Project Log

Date	Task Done
09/09/2011	Team formed
13/09/2011	1st meeting, initial plan
22/09/2011	2nd meeting, project title confirmed
27/09/2011	3rd meeting, project proposal
28/09/2011	Project proposal submitted
04/10/2011	4th meeting, Ocaml learning
05/10/2011	Feedback from instructor and TA
11/10/2011	5th meeting project reevaluated, coming up with new ideas
17/10/2011	Meeting with instructor, confirmation with new topics
18/10/2011	6th new project proposal finalized, language reference manual brainstorming
19/10/2011	New project proposal submitted
25/10/2011	7th language reference manual finalized
31/10/2011	Language reference manual submitted
08/11/2011	8th meeting, project progress check, task assigned to each member
15/11/2011	9th meeting, project progress check, language part, graphics part
22/11/2011	10th meeting, project progress check, language part finished, graphics part finished
29/11/2011	11th meeting, code emerging, testing task assigned
06/12/2011	12th meeting, project progress check, testing
18/12/2011	13th meeting, testing finished, project report, project ppt
22/12/2011	Project presentation, project final report submitted

Chapter 5

Architectural Design

5.1 Block Diagram



5.2 Interfaces

All the components are written in Ocaml including the GUI, and that's the reason we decided to build an interpreter.

The source file will be a .pacaml file. This file will be sent as the standard input to the executable pacaml interpreter. This source file will be converted into tokens by the scanner, and parsed into the types. All of the value of these types will be assigned into one of our root data class – gamemap, which stores the Pac-Man, a list of ghosts, a list of barriers, and a list of gifts. While we parse the built-in function "play()," pacaml will call the gaming GUI function, and pass the gamemap as the argument for drawing the interactive user interface. All the classes – point, player, item, and gamemap – are built with the set and get function to retrieve each member of the class objects.

5.3 Task Distribution

Component(s)	Implementer(s)
ast.mli	Shuwei Cao, Wenxin Zhu
classes.ml	Hui-Hsiang Kuo
game.ml	Chun-Kang Chen
interpret.ml	Hui-Hsiang Kuo
pacaml.ml	Hui-Hsiang Kuo
parser.mly	Shuwei Cao, Wenxin Zhu
scanner.mll	Shuwei Cao, Wenxin Zhu
Tests	All members
Makefile	Hui-Hsiang Kuo

Chapter 6

Test Plan

6.1 Representative Source Code

The input of PaCaml is PaCaml source codes. The output of PaCaml compiler is OCaml objects. The objects are then processed by the graphics interface written in Ocaml with the library OpenGL.

Due to the layout discussed above, we are not able to present the output in codes in this chapter.

6.2 Test Suites

We chose the test cases starting from the MICROC test suite, and gradually increased the complexity. The test cases consist of arithmetic, logic, control structures, different data types, and built-in functions.

6.2.1 Arithmetic

Test case	Motivation	Output	Result
print(point(4,3) + point(1,2))	Point addition	(5,5)	Correct
print(point(4,3) - point(1,2))	Point subtraction	(3,1)	Correct
print(point(4,3) * point(1,2))	Point multiplication	(4,6)	Correct
print(point(4,3) / point(1,2))	Point division	(4,1)	Correct
print(1 + 5)	Integer addition	6	Correct
print(5 - 2)	Integer subtraction	3	Correct
print(4 * 2)	Integer multiplication	8	Correct
print(1 / 2)	Integer division	0	Correct
print(5 % 3)	Integer Modulus	2	Correct
print(1 + 2 * 3 + 4)	Order of operations	11	Correct
print((1 + 2) * (3 + 4))	Order of operations	21	Correct

6.2.2 Logic

Test case	Motivation	Output	Result
print(!(1==1))	NOT Operator	False	Correct
print(!(1==2))	NOT Operator	True	Correct
print((1==1) && (1==1))	AND Operator	True	Correct
print((1==1) && (1==2))	AND Operator	False	Correct
print((1==2) && (1==1))	AND Operator	False	Correct
print((1==2) && (1==2))	AND Operator	False	Correct
print((1==1) || (1==1))	OR Operator	True	Correct
print((1==1) || (1==2))	OR Operator	True	Correct
print((1==2) || (1==1))	OR Operator	True	Correct
print((1==2) || (1==2))	OR Operator	False	Correct
print(1 != 2)	NOT EQUAL	True	Correct
print(1 != 1)	NOT EQUAL	False	Correct
print(1 < 2)	LESS	True	Correct
print(2 < 1)	LESS	False	Correct
print(1 <= 2)	LESS EQUAL	True	Correct
print(1 <= 1)	LESS EQUAL	True	Correct
print(2 <= 1)	LESS EQUAL	False	Correct
print(1 > 2)	GREATER	False	Correct
print(2 > 1)	GREATER	True	Correct
print(1 >= 2)	GREATER EQUAL	False	Correct
print(1 >= 1)	GREATER EQUAL	True	Correct
print(2 >= 1)	GREATER EQUAL	True	Correct
print(!(1==1) || (1==1))	Order of Logic	True	Correct
print(!((1==1) || (1==1)))	Order of Logic	False	Correct

6.2.3 if

Test case	Motivation	Output	Result

if(1)
{
print(0);
}
 	
if(0)
{
print(1);
}
else
{
print(2);
}
	Test if and if/else	0
		2
			Correct

6.2.4 for

Test case	Motivation	Output	Result

int i;
 	
for(i = 0; i < 10; i++)
{
print(i);
}
	Test for loop	0
		1
		2
		3
		4
		5
		6
		7
		8
		9
			Correct

6.2.5 while

Test case	Motivation	Output	Result

int i;
i = 0;
 	
while(i < 10)
{
print(i);
i++;
}
	Test while loop	0
		1
		2
		3
		4
		5
		6
		7
		8
		9
			Correct

6.2.6 Functions

Test case	Motivation	Output	Result

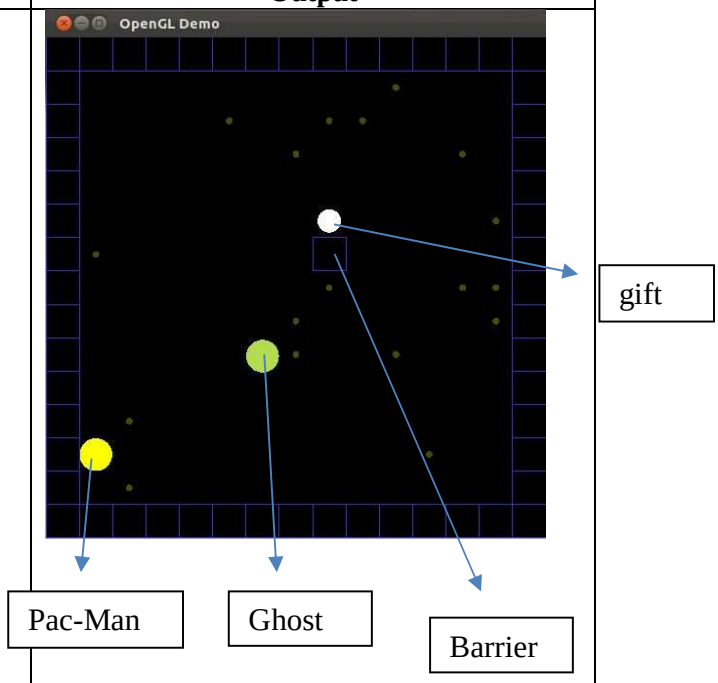
int plus(int a, int b)
{
return a + b;
}
 	
int minus(int a, int b)
{
return a - b;
}
 	
void main()
{
print(plus(1, 2));
print(minus(1, 2));
}
	Test function call	3
		-1
			Correct

6.2.7 Derived Types

Test case	Motivation	Output	Result
string str; str = “helloworld”; print(str);	Type of String	helloworld	Correct
point.x = 10;	The value of x is assigned to 10	10	Correct
point.x = 21;	Test out of bound	("invalid point x: 15. x must be between 1-13.")	Correct
item1.i_type = _GHOST;	Type field of item	This item is set to be a ghost	Correct
item1.level = _HARD;	Level field of item	The level of the ghost is set to be hard.	Correct

6.2.8 Pac-Man related functions

Test	Output

item createItem(int type, int x, int y)
{
item i;
point p;
p.x = x;
p.y = y;
i.i_type = type;
i.i_point = p;
return i;
}
 	
void main()
{
player pacman;
item ghost1;
item barrier1;
item gift1;
 	
pacman.p_point = getAvailablePoint();
setPlayer(pacman);
 	
ghost1 = createItem(_GHOST, 5, 5);
addGhost(ghost1);
 	
barrier1 = createItem(_BARRIER, 8 , 8);
addBarrier(barrier1);
 	
gift1 = createItem(_GIFT, 8 , 9);
addGift(gift1);
 	
play();
}
	

6.3 Testing Method & Testing Automation

To guarantee that the interpreter can work correctly, regression testing is performed as each piece of code is integrated. We initially chose test cases by selecting the basic features, like arithmetic, logic, and control features, and ensure that the interpreter works as expected. The next step was to assign and modify the value of each data type in different ways, and check if the interpreter still works as expected.

In the beginning of development, we test manually to ensure that all the output results of the interpreter are correct. In the later stages, we use a script written in Python to compare the expected output and actual output, and report the status of each test case with either PASS or NOTPASS.

Chapter 7

Lessons Learned

7.1 Lessons Learned

7.1.1 Chun-Kang Chen

Building a new language is a perfect way to deeply understand all the significant concepts of the PLT and, of course, Ocaml. I took a compiler class in the other university before; however, I never really created a language. The first assignment was really shock me -- I finished it before I understood it. But, that is a fundamental knowledge of a compiler.

I was assigned to be the guy to create the Pacman game in Ocaml. That's really much tougher than I imaged. Ocaml is the first functional language I learned, which is not as intuition as any other language. After the first part of the game is finished, I figured out the most difficult part of this project – team work. We have to discuss the interface between the interpreter and my game stuff. For connecting this part, I spend couple hours to modify the original code, and we made all interfaces for the following steps.

It makes us really excited while the first Pacman game built by "Pacaml". It's a great class project.

7.1.2 Hui-Hsiang Kuo

This project taught me many lessons in learning a new functional programming language, implementing a large scale software system, and working as a team. Although working on Ocaml programming was painful, but game design did make it interesting. For each regular meeting, we brainstormed with each other for getting the Pacaml in place. A good team is not that how smart members are, but how much they can dedicate to. I would like to say that I really enjoyed working with Chun-Kang, Wenxin, and Shuwei.

I was responsible for implementing the interpreter. However, the most difficult part of this project is not implementing the parser or interpreter, but deciding what the language should and shouldn't include. In the later stage of development, we found that we created lots of useless data types and unfriendly functions for Pacaml, and they were really too complicated for the end users. The process of improving the interpreter was with lots of challenges because we know that perfection is that there is nothing left to take away, and we almost achieved it.

The way to design a language is tough, but as a programmer, it is necessary to realize how compiler and interpreter work. Thanks to COMS4115. It's a great experience.

7.1.3 Wenxin Zhu

PaCaml is a fantastic practice to understand in the course Programming Languages and Translators. I find it quite helpful as we have gone through most of the concepts introduced in the course. What we have learnt in the lectures brings the theoretic background and basis, while what we have fulfilled in the project brings the experience which broadens my sight in programming languages and adds up to my knowledge.

I am responsible for scanning and parsing. The task was complicated as I had no idea what to begin with. The files that define the types, the parsing rules, and the tokens are relative to each another, and therefore, it is hard to finish one of them then go on with another. Fortunately, my teammates gave me a smart suggestion that I would better start with one or two function, say "print hello world", to understand how things work within the scanning and parsing files. As long as I was able to do the print function, I added more elements to the files and made it complete in the end. In the meantime, Shuwei has been a great help in fixing the bugs that I would hardly notice, as well as adding elements to the scanner and the parser. With Shuwei's efforts, the scanner and the parser function as we expected.

I have had a lot of fun in PaCaml. Thank you, my teammates!

7.1.4 Shuwei Cao

The most important thing I have learned from this project was that one should never ever say something is too hard for him before he has the courage of doing that. However, once you get started, you have no choice but keep your head up and hold your dream tight.

It is really a difficult time when you have no idea of how to do the whole thing. But, just be patient and little by little, your understanding is able to get improved. In the project, I was responsible for the parser and scanner in cooperation with Wenxin Zhu.

I have learned how to use Linux, how to use SVN, how to program using Ocaml, how to write the parser and scanner, how to do the testing… I would like to thank my teammates Chun-Kang Chen, Hui-Hsiang Kuo and Wenxin Zhu. It was a great experience working with you. I have learned a lot from your experience and teaching.

This project gives me a brand new idea of how programming language is like. It is really difficult but it is truly worthy.

7.2 Advice for Future Teams

The first as well as the most important step is to design the language. You four people are to spend one semester to realize the design in codes, and therefore, it is important that you have a cool idea to stick to, and cool teammates that agree on the idea. It is never over-prepared if you have a list of key question at hand to discuss within the team and with your TAs and professor. What is the interest of the team? What are the advantages that you expect in the new language? What are the functionalities? For the key functionalities, what are the details that you might want to agree on at the very beginning?

The second step is to set up a meeting time, an overall schedule, and always, always, keep up to it.

Now you have a cool idea. Yet it doesn't mean that the idea have to remain constant without any a bit of change. Prepare for the difficulties in that you are not sure to fix some problems considering time and efficiency issue. We suggest that you find reference from previous teams or your peer, and go talk to your TAs and professors for advice. Please keep in mind that compromises are to be made from time to time, since this is a one-semester project, and it is possible that the result is not perfect at the end. Think about the issue in this way: which do you prefer, to spend much time on dedicated details before you are sure to get things done, or to build and realize the frame first and then fill it up with the details?

The last piece of advice, guess what:

START EARLY.

Good luck, our fellow teams!

Reference

[1] *Pac-Man still going strong at 30*. (2010). Retrieved Dec 14, 2011, from UPI: http://www.upi.com/Entertainment_News/2010/05/22/Pac-Man-still-going-strong-at-30/UPI-74821274544243/

[2] Parish, J. (n.d.). *The Essential 50: Part 10-Pac-Man*. Retrieved Dec 14, 2011, from 1UP: http://www.1up.com/features/essential-50-pac-man

Appendix

Appendix A.1 – ast.ml

```
type op =
    Add | Sub | Mult | Div | Mod
    | And | Or | Eq | NEq | Less | LEq | Greater | GEq

type expr =
    Id of string (* foo *)
    | Member of string * string (* foo.bar *)
    | Integer of int (* 13 *)
    | Strings of string (* "helloworld" *)
    | GameType of string (* _HARD *)
    | Assign of expr * expr (* x = y *)
    | BinOp of expr * op * expr (* x + y *)
    | Not of expr (* !x *)
    | Call of string * expr list (* foo(x, y) *)
    | NoExpr (* for (;;) *)

type stmt = (* Statements *)
    Expr of expr (* foo = bar + 13; *)
    | Return of expr (* return 13; *)
    | Block of stmt list (* { ... } *)
    | If of expr * stmt * stmt (* if (foo == 13) {...} else {...} *)
    | For of expr * expr * expr * stmt (* for (i=0;i<10;i=i+1) { ... } *)
    | While of expr * stmt (* while (i<10) { ... } *)
type par_decl = {
    paramname : string; (* Variable Name*)
    paramtype : string; (* Variable Type *)
}
type var_decl = {
    varname : string; (* Variable Name *)
    vartype : string; (* Variable Type *)
}
type func_decl = {
    fname : string; (* Function Name*)
    rettype : string; (* Return Type *)
    formals : par_decl list; (* Argument Name *)
    locals : var_decl list; (* Local Variables *)
    body : stmt list;
}
(* types used for pacaml *)
type point = {
    mutable x : int;
    mutable y : int;
}
type player = {
    mutable speed : int;
    mutable status : string;
    mutable p_point : point;
    mutable remaining_time : int;
}
type item = {
    mutable i_type : int;
    mutable duration : int;
    mutable i_point : point;
    mutable color : float * float * float;
    mutable level : int;
}
type map = {
    mutable pacman : player;
    mutable list_item : item list;
    mutable width : int;
    mutable height : int;
}
type myType = Int of int | Bool of bool | String of string | Map of map | Player of
player | Item of item | Point of point

(* get the type of a variable *)
let getType v =
    match v with
        Int(v) -> "int"
    | Bool(v) -> "bool"
    | String(v) -> "string"
```

```
    | Map(v) -> "map"
    | Player(v) -> "player"
    | Item(v) -> "item"
    | Point(v) -> "point"
(* return  myType of a variable*)
let getInt v =
    match v with
        Int(v) -> v
    | _ -> 0

let getBool v =
    match v with
        Bool(v) -> v
    | Int(1) -> true
    | _ -> false
let getString v =
    match v with
        String(v) -> v
    | _ -> ""
let getMap v =
    match v with
        Map(v) -> v
    | _ -> {pacman= {speed=0;  status="normal";  p_point={x=750;
y=750};  remaining_time=0};list_item=[]; width=1500; height=1500}
let getPlayer v =
    match v with
        Player(v) -> v
    | _ -> {speed=0;  status="normal"; p_point={x=750; y=750} ; remaining_time=0}

let getItem v =
    match v with
        Item(v) -> v
    | _ -> {i_type=0; duration=0; i_point={x=0; y=0}; color=(1.,1.,1.) ; level=0}
let getPoint v =
    match v with
        Point(v) -> v
    | _ -> {x=0; y=0}
type program = var_decl list * func_decl list (* global vars, funcs *)
```

Appendix A.2 – scanner.mll

```
{ open Parser } (* token types *)
rule token = parse
    [' ' '\t' '\r' '\n'] { token lexbuf } (* white space *)
    | "/*" { comment lexbuf } (* comment *)
    | "//" { singlecomment lexbuf }
    | '(' { LPAREN }
    | ')' { RPAREN }
    | '{' { LBRACE }
    | '}' { RBRACE }
    | ';' { SEMI }
    | ',' { COMMA }
    | '.' { DOT }
    | '+' { PLUS }
    | '-' { MINUS }
    | '*' { TIMES }
    | '/' { DIVIDE }
    | '%' { MOD }
    | '"' { QUOTATION}
    | "+=" { PLUSEQ }
    | "-=" { MINUSEQ }
    | "*=" { TIMESEQ }
    | "/=" { DIVIDEEQ }
    | "%=" { MODEQ }
    | '=' { ASSIGN }
    | '!' { NOT }
    | "++" { PLUSPLUS }
    | "--" { MINUSMINUS }
    | "==" { EQ }
    | "!=" { NEQ }
    | '<' { LT }
    | "<=" { LEQ }
    | ">" { GT }
    | ">=" { GEQ }
    | "&&" { AND }
    | "||" { OR }
```

```
    | "if" { IF } (* keyword *)
    | "else" { ELSE }
    | "for" { FOR }
    | "while" { WHILE }
    | "return" { RETURN }
    | "void" { DATATYPE("void") }
    | "int" { DATATYPE("int") }
    | "string" { DATATYPE("string")}
    | "map" { DATATYPE("map") }
    | "player" { DATATYPE("player") }
    | "item" { DATATYPE("item") }
    | "point" { DATATYPE("point") }
    | ['0'-'9']+ as ints { INTEGER(int_of_string ints) } (* Integer *)
    | ['_']['a'-'z' 'A'-'Z' '_' '0'-'9']+ as gametype { GAMETYPE(gametype) }  (* typs for
pacman  *)
    | ['a'-'z' 'A'-'Z' '0'-'9' '_']+ as id { ID(id) }
    | eof { EOF } (* End of file *)
    | _ as char { raise (Failure("illegal character: " ^ Char.escaped char)) }
    and comment = parse
    "*/" { token lexbuf } (* End of comment *)
    | _ { comment lexbuf }

    and singlecomment = parse
    ['\r' '\n'] {token lexbuf } (* End of comment *)
    | _ {singlecomment lexbuf }
```

Appendix A.3 – parser.mly

```
%{ open Ast %}
%token LPAREN RPAREN LBRACE RBRACE LBRACKET RBRACKET
%token SEMI COMMA DOT QUOTATION
%token PLUS MINUS TIMES DIVIDE MOD PLUSPLUS MINUSMINUS
%token PLUSEQ MINUSEQ TIMESEQ DIVIDEEQ MODEQ
%token EQ NEQ LT LEQ GT GEQ AND NOT OR ASSIGN
%token IF ELSE ELSEIF FOR WHILE RETURN
%token INT VOID FLOAT BOOL
%token <int> INTEGER
%token <string> ID
%token <string> DATATYPE
%token <string> GAMETYPE
%token EOF
%left PLUSEQ MINUSEQ
%left TIMESEQ DIVIDEEQ MODEQ
%right ASSIGN
%left OR AND EQ NEQ
%left LT GT LEQ GEQ
%right NOT
%left PLUS MINUS
%left TIMES DIVIDE MOD
%left PLUSPLUS MINUSMINUS
%nonassoc NOELSE
%nonassoc ELSE
%nonassoc LPAREN
%start program
%type <Ast.program> program
%%
program:
/* nothing */ { [], [] }
| program vdecl { ($2 :: fst $1), snd $1 }
| program fdecl { fst $1, ($2 :: snd $1) }
fdecl:
    DATATYPE ID LPAREN formals_opt RPAREN LBRACE vdecl_list stmt_list RBRACE
        { {    fname = $2;
            rettype = $1;
            formals = $4;
            locals = List.rev $7;
            body = List.rev $8 } }
formals_opt:
    /* nothing */ { [] }
    | formal_list { List.rev($1) }
formal_list:
    param_decl { [$1] }
    | formal_list COMMA param_decl { $3 :: $1 }

vdecl_list:
    /* nothing */    { [] }
```

```
    | vdecl_list vdecl { $2 :: $1 }
vdecl:
    DATATYPE ID SEMI { { varname = $2; vartype = $1 } }

param_decl:
    DATATYPE ID
        { {    paramname = $2;
           paramtype = $1 } }
stmt_list:
    /* nothing */ { [] }
    | stmt_list stmt { $2 :: $1 }

stmt:
       expr SEMI { Expr($1) }
    | RETURN expr_opt SEMI { Return($2) }
    | LBRACE stmt_list RBRACE { Block(List.rev $2) }
    | IF LPAREN expr RPAREN stmt %prec NOELSE { If($3, $5, Block([])) }
    | IF LPAREN expr RPAREN stmt ELSE stmt %prec ELSE { If($3, $5, $7) }
    | FOR LPAREN expr_opt SEMI expr_opt SEMI expr_opt RPAREN stmt { For($3, $5, $7, $9) }
    | WHILE LPAREN expr RPAREN stmt { While($3, $5) }
expr_opt:
    /* nothing */ { NoExpr }
    | expr { $1 }
expr:
       ID { Id($1) }                                          // foo
    | ID DOT ID { Member($1, $3) }                            // player.speed
    | INTEGER { Integer($1) }                                 // 8
    | QUOTATION ID QUOTATION { Strings($2) }                  // "helloworld"
    | GAMETYPE { GameType($1) }                               // _HARD
    | expr PLUS expr { BinOp($1, Add, $3) }                   // x + y
    | expr MINUS expr { BinOp($1, Sub, $3) }                  // x - y
    | expr TIMES expr { BinOp($1, Mult, $3) }                 // x * y
    | expr DIVIDE expr { BinOp($1, Div, $3) }                 // x / y
    | expr ASSIGN expr { Assign($1,$3) }                      // x = y
    | expr PLUSEQ expr { Assign($1, BinOp($1, Add, $3)) }     // x += y
    | expr MINUSEQ expr { Assign($1, BinOp($1, Sub, $3)) }    // x -= y
    | expr TIMESEQ expr { Assign($1, BinOp($1, Mult, $3)) }   // x *= y
    | expr DIVIDEEQ expr { Assign($1, BinOp($1, Div, $3)) }   // x /=y
    | expr MODEQ expr { Assign($1, BinOp($1, Mod, $3)) }      // x %= y
    | expr MOD expr { BinOp($1, Mod, $3) }                    // x % y
    | expr AND expr { BinOp($1, And, $3) }                    // x && y
    | expr OR expr { BinOp($1, Or, $3) }                      // x || y
    | expr EQ expr { BinOp($1, Eq, $3) }                      // x == y
    | expr NEQ expr { BinOp($1, NEq, $3) }                    // x != y
    | expr LT expr { BinOp($1, Less, $3) }                    // x < y
    | expr LEQ expr { BinOp($1, LEq, $3) }                    // x <= y
    | expr GT expr { BinOp($1, Greater, $3) }                 // x > y
    | expr GEQ expr { BinOp($1, GEq, $3) }                    // x >= y
    | NOT expr { Not($2) }                                    // !x
    | expr PLUSPLUS { Assign($1, BinOp($1, Add, Integer(1))) }    // x++
    | expr MINUSMINUS { Assign($1, BinOp($1, Sub, Integer(1))) }  // x--
    | LPAREN expr RPAREN { $2 }                               // (x)
    | ID LPAREN actuals_opt RPAREN { Call($1, $3) }           // foo(...), function
call
actuals_opt:
    /* nothing */ { [] }
    | actuals_list { List.rev $1 }
actuals_list:
    expr { [$1] }
    | actuals_list COMMA expr { $3 :: $1 }
```

Appendix A.4 – interpret.ml

```
open Ast
open Printf
open Classes
open Game
module NameMap = Map.Make(struct
    type t = string
    let compare x y = Pervasives.compare x y
end)
module StringMap = Map.Make(String);;
let mymap = new gamemap;;
let pointMap = ref StringMap.empty;;
let avail_x = ref 0;;
let avail_y = ref 0;;
```

```
let num_items = ref 0;;
let gift_colors = [|(1.,0.5,0.5) ; (0.5,0.5,1.) ; (1.,0.,1.) ; (0.8,0.8,0.6)|];
exception ReturnException of myType * myType NameMap.t
(* Initiate variable *)
let init t =
  match t with
    "int" -> Int(0)
    | "bool" -> Bool(false)
    | "string" -> String("")
    | "map" -> Map({ pacman= {speed=5;  status="normal";  p_point={x=1;
y=1};  remaining_time=0}; list_item=[]; width=1500 ; height=1500})
    | "player" -> Player({speed=5;  status="normal";  p_point={x=1;
y=1} ;remaining_time=0})
    | "item" -> Item({i_type=0; duration=0;i_point={x=0; y=0}; color=(1.,1.,1.);
level=0})
    | "point" -> Point({x=0; y=0})
    | _ -> Int(0)
(* Main entry point: run a program *)
let run (vars, funcs) =
    (* Put function declarations in a symbol table *)
    let func_decls = List.fold_left
        (fun funcs fdecl -> NameMap.add fdecl.fname fdecl funcs)
        NameMap.empty funcs
    in

    (* Invoke a function and return an updated global symbol table *)
    let rec call fdecl actuals globals =

    (* Evaluate an expression and return (value, updated environment) *)
    let rec eval env = function
            Integer(i) -> Int i, env
        | GameType(i) ->
            (match i with
              "_EASY" -> Int 0, env
            | "_NORMAL" -> Int 1, env
            | "_HARD" -> Int 2, env
            | "_GHOST" -> Int 3, env
            | "_BARRIER" -> Int 4, env
            | "_GIFT" -> Int 5, env
            | "_GIFT_SPEEDUP" -> Int 6, env
            | "_GIFT_SLOWDOWN" -> Int 7, env
            | "_GIFT_KILLER" -> Int 8, env
            | "_GIFT_SCORE" -> Int 9, env
            | _ -> raise (Failure ("Invalid game type: " ^ i ^ ".")))
        | NoExpr ->  Int 1, env
        (* String *)
        | Strings(var) -> String var, env
        (* Identifier *)
        | Id(var) ->
            let locals, globals = env in
            if NameMap.mem var locals then
                (NameMap.find var locals), env
            else if NameMap.mem var globals then
                (NameMap.find var globals), env
            else raise (Failure ("Undeclared identifier: " ^ var))
        (* Assign value to member *)
        | Member(var, i) ->
            let v, env = eval env (Id var) in
                let vType = getType v in
                (match vType with
                  | "player" ->
                    (match i with
                      | "p_point" -> Point (getPlayer v).p_point
                      | _ -> raise (Failure ("Invalid member of Player: " ^ i)))
                  | "item" ->
                    (match i with
                        "i_type" -> Int (getItem v).i_type
                      | "duration" -> Int (getItem v).duration
                      | "i_point" -> Point (getItem v).i_point
                      | "level" -> Int (getItem v).level
                      | _ -> raise (Failure ("Invalid member of Item: " ^ i)))
                  | "point" ->
                    (match i with
                        "x" -> Int (getPoint v).x
                      | "y" -> Int (getPoint v).y
                      | _ -> raise (Failure ("Invalid member of Point: " ^
i)))
                  | _ -> raise (Failure ("Cannot access " ^ var ^ "." ^ i))), env
        (* Binary operator *)
        | BinOp(e1, op, e2) ->
            let v1, env = eval env e1 in
```

```
            let v2, env = eval env e2 in
            let v1Type = getType v1 in
            let v2Type = getType v2 in
            if v1Type = v2Type then
                (match op with
                    Add ->
                        if v1Type = "int" then
                             Int (getInt v1 + getInt v2)
                        else if v1Type = "point" then
                             if ((getPoint v1).x + (getPoint v2).x)>=1 && ((getPoint
v1).x + (getPoint v2).x)<=13
                                 && ((getPoint v1).y + (getPoint v2).y)>=1 && ((getPoint
v1).y + (getPoint v2).y)<=13 then
                                  Point({x=((getPoint v1).x + (getPoint v2).x);
y=((getPoint v1).y + (getPoint v2).y)})
                             else    raise (Failure ("Out of bound: The value of point.x
and point.y should be between 1-13"))
                        else raise (Failure ("Incorrect data type: " ^ v1Type ^ " + " ^
v2Type))
                    | Sub ->
                        if v1Type = "int" then
                             Int (getInt v1 - getInt v2)
                        else if v1Type = "point" then
                            if ((getPoint v1).x - (getPoint v2).x)>=1 && ((getPoint v1).x
- (getPoint v2).x)<=13
                                 && ((getPoint v1).y - (getPoint v2).y)>=1 && ((getPoint
v1).y - (getPoint v2).y)<=13 then
                                 Point({x=((getPoint v1).x - (getPoint v2).x);
y=((getPoint v1).y - (getPoint v2).y)})
                            else    raise (Failure ("Out of bound: The value of point.x
and point.y should be between 1-13"))
                        else raise (Failure ("Incorrect data type: " ^ v1Type ^ " - " ^
v2Type))
                    | Mult ->
                        if v1Type = "int" then
                             Int (getInt v1 * getInt v2)
                        else if v1Type = "point" then
                            if ((getPoint v1).x * (getPoint v2).x)>=1 && ((getPoint v1).x
* (getPoint v2).x)<=13
                                 && ((getPoint v1).y * (getPoint v2).y)>=1 && ((getPoint
v1).y * (getPoint v2).y)<=13 then
                                 Point({x=((getPoint v1).x * (getPoint v2).x);
y=((getPoint v1).y * (getPoint v2).y)})
                            else    raise (Failure ("Out of bound: The value of point.x
and point.y should be between 1-13"))
                        else raise (Failure ("Incorrect data type: " ^ v1Type ^ " * " ^
v2Type))
                    | Div ->
                        if v1Type = "int" then
                             Int (getInt v1 / getInt v2)
                        else if v1Type = "point" then
                            if ((getPoint v1).x / (getPoint v2).x)>=1 && ((getPoint v1).x
/ (getPoint v2).x)<=13
                                 && ((getPoint v1).y / (getPoint v2).y)>=1 && ((getPoint
v1).y / (getPoint v2).y)<=13 then
                                 Point({x=((getPoint v1).x / (getPoint v2).x);
y=((getPoint v1).y / (getPoint v2).y)})
                            else    raise (Failure ("Out of bound: The value of point.x
and point.y should be between 1-13"))
                        else raise (Failure ("Incorrect data type: " ^ v1Type ^ " / " ^
v2Type))
                    | Mod ->
                        if v1Type = "int" then
                             Int (getInt v1 mod getInt v2)
                        else raise (Failure ("Incorrect data type: " ^ v1Type ^ " % " ^
v2Type))
                    | And ->
                        if v1Type = "bool" then
                             Bool (getBool v1 && getBool v2)
                        else raise (Failure ("Incorrect data type: " ^ v1Type ^ " && " ^
v2Type))
                    | Or ->
                        if v1Type = "bool" then
                             Bool (getBool v1 || getBool v2)
                        else raise (Failure ("Incorrect data type: " ^ v1Type ^ " || " ^
v2Type))
                    | Eq ->
                        if v1Type = "int" then
                               Bool (getInt v1 = getInt v2)
                        else if v1Type = "map" then
                             Bool (getMap v1 = getMap v2)
```

```
                        else if v1Type = "player" then
                             Bool (getPlayer v1 = getPlayer v2)
                        else if v1Type = "item" then
                             Bool (getItem v1 = getItem v2)
                        else if v1Type = "point" then
                             Bool (getPoint v1 = getPoint v2)
                        else raise (Failure ("Incorrect data type: " ^ v1Type ^ " == " ^
v2Type))
                    | NEq ->
                        if v1Type = "int" then
                               Bool (getInt v1 != getInt v2)
                        else if v1Type = "map" then
                             Bool (getMap v1 != getMap v2)
                        else if v1Type = "player" then
                             Bool (getPlayer v1 != getPlayer v2)
                        else if v1Type = "item" then
                             Bool (getItem v1 != getItem v2)
                        else if v1Type = "point" then
                             Bool (getPoint v1 != getPoint v2)
                        else raise (Failure ("Incorrect data type: " ^ v1Type ^ " == " ^
v2Type))
                    | Less ->
                        if v1Type = "int" then
                               Bool (getInt v1 < getInt v2)
                        else raise (Failure ("Cannot compare " ^ v1Type ^ " < " ^
v2Type))
                    | LEq ->
                        if v1Type = "int" then
                               Bool (getInt v1 <= getInt v2)
                        else raise (Failure ("Cannot compare: " ^ v1Type ^ " <= " ^
v2Type))
                    | Greater ->
                        if v1Type = "int" then
                               Bool (getInt v1 > getInt v2)
                        else raise (Failure ("Cannot compare: " ^ v1Type ^ " > " ^
v2Type))
                    | GEq ->
                        if v1Type = "int" then
                               Bool (getInt v1 >= getInt v2)
                        else raise (Failure ("Cannot compare: " ^ v1Type ^ " >= " ^
v2Type))), env
          else raise (Failure ("Data type mismatch : " ^ v1Type ^ " and " ^ v2Type))
        (* Assignment *)
      | Assign(var, e) ->
          let var1, env = eval env var in
          let e1, (locals, globals) = eval env e in
          let v1Info =
              match var with
                    Id (i) -> ("id", (i, ""))
                  | Strings(i) -> ("strings", (i, ""))
                  | Member(i, j) -> ("member", (i, j))
                  | _ -> raise (Failure ("Left side must be an identifier or a
member")) in
              let v1IdType = fst v1Info in
              let v1Name = snd v1Info in
              let v1Type = (* ("map", "locals") *)
                  (if NameMap.mem (fst v1Name) locals then
                      (getType (NameMap.find (fst v1Name) locals), "locals")
                  else if NameMap.mem (fst v1Name) globals then
                      (getType (NameMap.find (fst v1Name) globals), "globals")
                  else raise (Failure ("Undeclared identifier: " ^ fst
v1Name)))
              in
              let v1RetType = getType var1 in
              let e1Type = getType e1 in
              if v1RetType = e1Type then
                   match v1RetType with
                      "int" ->
                          if v1IdType = "id" then
                              (if snd v1Type = "locals" then
                                  e1, (NameMap.add (fst v1Name) e1 locals, globals)
                              else if snd v1Type = "globals" then
                                  e1, (locals, NameMap.add (fst v1Name) e1 globals)
                              else raise (Failure ("Fatal error")))
                          else if v1IdType = "member" then
                              if fst v1Type = "item"  then
                                  if snd v1Name = "i_type" then
                                      if getInt e1 = 3 || getInt e1 = 4  || getInt e1 =
5then

                                          if snd v1Type = "locals" then
                                              e1, (
```

```
                                                    ((getItem (NameMap.find (fst v1Name)
locals)).i_type <- getInt e1);
                                                    (locals, globals))
                                            else if snd v1Type = "globals" then
                                                e1, (
                                                    ((getItem (NameMap.find (fst v1Name)
globals)).i_type <- getInt e1);
                                                    (locals, globals))
                                            else raise (Failure ("Fatal error"))
                                        else raise (Failure ("Invalid value of i_type: "
^ string_of_int (getInt e1) ^ ". i_type must be _GHOST, _BARRIER, or _GIFT."))
                                    else if snd v1Name = "duration" then
                                        if getInt e1 >= 100 && getInt e1 <= 500 then
                                            if snd v1Type = "locals" then
                                                e1, (
                                                    ((getItem (NameMap.find (fst v1Name)
locals)).duration <- getInt e1);
                                                    (locals, globals))
                                            else if snd v1Type = "globals" then
                                                e1, (
                                                    ((getItem (NameMap.find (fst v1Name)
globals)).duration <- getInt e1);
                                                    (locals, globals))
                                            else raise (Failure ("Fatal error"))
                                        else raise (Failure ("Invalid value of duration:
" ^ string_of_int (getInt e1) ^ ". Duration must be between 100-500."))
                                    else if snd v1Name = "level" then
                                        if (getItem (NameMap.find (fst v1Name)
locals)).i_type = 3 then
                                            if getInt e1 = 0 || getInt e1 = 1 || getInt
e1 = 2 then
                                                if snd v1Type = "locals" then
                                                    e1, (
                                                        ((getItem (NameMap.find (fst
v1Name) locals)).level <- getInt e1);
                                                        (locals, globals))
                                                else if snd v1Type = "globals" then
                                                    e1, (
                                                        ((getItem (NameMap.find (fst
v1Name) globals)).level <- getInt e1);
                                                        (locals, globals))
                                                else raise (Failure ("Fatal
error"))
                                            else raise (Failure ("Invalid value of level:
" ^ string_of_int (getInt e1) ^ ". level must be _EASY(0), _MEDIAN(1), or
_HARD(2)"))
                                        else if (getItem (NameMap.find (fst v1Name)
locals)).i_type = 5 then
                                            if getInt e1 = 6 || getInt e1 = 7 || getInt
e1 = 8 || getInt e1 = 9 then
                                                if snd v1Type = "locals" then
                                                    e1, (
                                                        ((getItem (NameMap.find (fst
v1Name) locals)).level <- ((getInt e1)-5));
                                                        ((getItem (NameMap.find (fst
v1Name) locals)).color <- gift_colors.(((getInt e1)-6)));
                                                        (locals, globals))
                                                else if snd v1Type = "globals" then
                                                    e1, (
                                                        ((getItem (NameMap.find (fst
v1Name) globals)).level <- ((getInt e1)-5));
                                                        ((getItem (NameMap.find (fst
v1Name) locals)).color <- gift_colors.(((getInt e1)-6)));
                                                        (locals, globals))
                                                else raise (Failure ("Fatal
error"))
                                            else raise (Failure ("Invalid value of level:
" ^ string_of_int (getInt e1) ^ ". Level must be _GIFT_SPEEDUP, _GIFT_SLOWDOWN,
_GIFT_KILLER, _GIFT_SCORE"))
                                        else raise (Failure ("Invalid value of level:
Level of this item shouldn't be used."))
                                    else raise (Failure ("Fatal error"))
                                else if fst v1Type = "point" then
                                    if snd v1Name = "x" then
                                        if getInt e1 >= 1 && getInt e1 <= 13 then
                                            if snd v1Type = "locals" then
                                                e1, (
                                                    ((getPoint (NameMap.find (fst v1Name)
locals)).x <- getInt e1);
                                                    (locals, globals))
                                            else if snd v1Type = "globals" then
```

```
                                                e1, (
                                                    ((getPoint (NameMap.find (fst v1Name)
globals)).x <- getInt e1);
                                                    (locals, globals))
                                            else raise (Failure ("Fatal error"))
                                        else raise (Failure ("invalid point x: " ^
string_of_int (getInt e1) ^ ". x must be between 1-13."))
                                    else if snd v1Name = "y" then
                                        if getInt e1 >= 1 && getInt e1 <= 13 then
                                            if snd v1Type = "locals" then
                                                e1, (
                                                    ((getPoint (NameMap.find (fst v1Name)
locals)).y <- getInt e1);
                                                    (locals, globals))
                                            else if snd v1Type = "globals" then
                                                e1, (
                                                    ((getPoint (NameMap.find (fst v1Name)
globals)).y <- getInt e1);
                                                    (locals, globals))
                                            else raise (Failure ("Fatal error"))
                                        else raise (Failure ("invalid point y: " ^
string_of_int (getInt e1) ^ ". y must be between 1-13."))
                                    else raise (Failure ("Fatal error"))
                                else raise (Failure ("Cannot assign value to: " ^ fst
v1Type ^ "." ^ snd v1Name))
                            else raise (Failure ("Cannot assign value to: " ^ fst
v1Type))
                    | "string" ->
                        if v1IdType = "id" then
                            (if snd v1Type = "locals" then
                                e1, (NameMap.add (fst v1Name) e1 locals, globals)
                            else if snd v1Type = "globals" then
                                e1, (locals, NameMap.add (fst v1Name) e1 globals)
                            else raise (Failure ("Fatal error")))
                        else raise (Failure ("Cannot assign to: " ^ fst v1Type))
                    | "point" ->
                        if v1IdType = "id" then
                            (if snd v1Type = "locals" then
                                e1, (NameMap.add (fst v1Name) e1 locals, globals)
                            else if snd v1Type = "globals" then
                                e1, (locals, NameMap.add (fst v1Name) e1 globals)
                            else raise (Failure ("Fatal error")))
                        else if v1IdType = "member" then
                            if fst v1Type = "player" then
                                if snd v1Name = "p_point" then
                                    if getType e1 = "point" then
                                        if snd v1Type = "locals" then
                                            e1, (
                                                ((getPlayer (NameMap.find (fst v1Name)
locals)).p_point.x <- (getPoint e1).x);
                                                ((getPlayer (NameMap.find (fst v1Name)
locals)).p_point.y <- (getPoint e1).y);
                                                (locals, globals))
                                        else if snd v1Type = "globals" then
                                            e1, (
                                                ((getPlayer (NameMap.find (fst v1Name)
globals)).p_point.x <- (getPoint e1).x);
                                                ((getPlayer (NameMap.find (fst v1Name)
globals)).p_point.y <- (getPoint e1).y);
                                                (locals, globals))
                                        else raise (Failure ("Fatal error"))
                                    else raise (Failure ("Invalid value of i_point: " ^
getString e1 ^ ". Type of the argument should be point."))
                                else raise (Failure ("Fatal
error"))
                            else if fst v1Type = "item" then
                                if snd v1Name = "i_point" then
                                    if getType e1 = "point" then
                                        if snd v1Type = "locals" then
                                            e1, (
                                                ((getItem (NameMap.find (fst v1Name)
locals)).i_point.x <- (getPoint e1).x);
                                                ((getItem (NameMap.find (fst v1Name)
locals)).i_point.y <- (getPoint e1).y);
                                                (locals, globals))
                                        else if snd v1Type = "globals" then
                                            e1, (
                                                ((getItem (NameMap.find (fst v1Name)
locals)).i_point.x <- (getPoint e1).x);
                                                ((getItem (NameMap.find (fst v1Name)
locals)).i_point.y <- (getPoint e1).y);
```

```
                                                (locals, globals))
                                        else raise (Failure ("Fatal error"))
                                    else raise (Failure ("Invalid value of i_point: " ^
getString e1 ^ ". Type of argument should be point."))
                                else raise (Failure ("Fatal
error"))
                            else raise (Failure ("Cannot assign value to: " ^ fst
v1Type))
                        else raise (Failure ("Cannot assign value to: " ^ fst v1Type))
                    | _ -> (* player, point, point *)
                        if v1IdType = "id" then
                            (if snd v1Type = "locals" then
                                e1, (NameMap.add (fst v1Name) e1 locals, globals)
                            else if snd v1Type = "globals" then
                                e1, (locals, NameMap.add (fst v1Name) e1 globals)
                            else raise (Failure ("Fatal error")))
                        else raise (Failure ("Cannot assign value to: " ^ fst
v1Type))
                else if v1IdType = "id" then
                        raise (Failure ("Cannot assign: " ^ fst v1Type ^ " = " ^ e1Type))
                else if v1IdType = "member" then
                        raise (Failure ("Cannot assign: " ^ v1RetType ^ " = " ^ e1Type))
                else raise (Failure ("Fatal error"))
        | Not(var) ->
            let vnew, envnew = eval env var in
                if getType vnew = "bool" then
                    Bool(not (getBool vnew)), envnew
                else raise (Failure ("Data type mismatch: " ^ getType vnew))
    (* built-in sfunction setPlayer(player) *)
    | Call("setPlayer", [e]) ->
        let v, env = eval env e in
        let v1Info =
            match e with
                Id (i) -> ("id", (i, ""))
            | _ -> raise (Failure ("left side must be an identifier or a member"))
in
        let v1Name = snd v1Info in
            (if getType v = "player"  then
                let pacman = getPlayer(v) in
                (if((StringMap.mem (string_of_int(pacman.p_point.x) ^ "," ^
string_of_int(pacman.p_point.y)) !pointMap) = false) then
                    ignore(
                        let tmp = new player in
                        tmp#setSpeed 5;
                        tmp#getPoint#setX (pacman.p_point.x*100+50);
                        tmp#getPoint#setY (pacman.p_point.y*100+50);
                        mymap#setPlayer tmp;
                        pointMap := StringMap.add (string_of_int(pacman.p_point.x) ^ ","
^ string_of_int(pacman.p_point.y)) v1Name !pointMap;
                        num_items := !num_items +1;
                    )
                else
                    raise (Failure ("This point has been used (setPlayer(player))")));
            else
                raise (Failure ("type mismatch: !" ^ getType v)));
          Int(0), env

    (* built-in function addGhost(item) *)
    | Call("addGhost", [e]) ->
        let v, env = eval env e in
        let v1Info =
                match e with
                      Id (i) -> ("id", (i, ""))
                    | _ -> raise (Failure ("Left side must be an identifier or a
member")) in
        let v1Name = snd v1Info in
            (if getType v = "item"  then
                let ghost = getItem(v) in
                (if((StringMap.mem (string_of_int(ghost.i_point.x) ^ "," ^
string_of_int(ghost.i_point.y)) !pointMap) = false) then
                    ignore(
                        if ghost.i_type = 3 then
                        let tmp = new item in
                        tmp#setSpeed 5;
                        tmp#setType 1;
                        tmp#getPoint#setX (ghost.i_point.x*100+50);
                        tmp#getPoint#setY (ghost.i_point.y*100+50);
                        tmp#setLevel ghost.level;
                        tmp#setColor ((Random.float 1.), (Random.float 1.),
(Random.float 1.));
                        tmp#setRadius 0.49;
```

```
                            mymap#addGhost tmp;
                            pointMap := StringMap.add (string_of_int(ghost.i_point.x) ^
"," ^ string_of_int(ghost.i_point.y)) v1Name !pointMap;
                            num_items := !num_items +1;
                        else
                            raise (Failure ("Data type mismatch: This item is not
GHOST"));
                        )
                    else
                        raise (Failure ("This point has been used (addGhost(Item))")));
            else
                raise (Failure ("Data type mismatch: " ^ getType v)));
        Int(0), env

    (* built-in function addBarrier(item) *)
    | Call("addBarrier", [e]) ->
        let v, env = eval env e in
        let v1Info =
            match e with
                Id (i) -> ("id", (i, ""))
            | _ -> raise (Failure ("Left side must be an identifier or member access"))
in
        let v1Name = snd v1Info in
            (if getType v = "item"  then
                let barrier = getItem(v) in
                    (if((StringMap.mem (string_of_int(barrier.i_point.x) ^ "," ^
string_of_int(barrier.i_point.y)) !pointMap) = false) then
                        ignore(
                        if barrier.i_type = 4 then
                            let tmp = new item in
                            tmp#setType 0;
                            tmp#getPoint#setX (barrier.i_point.x);
                            tmp#getPoint#setY (barrier.i_point.y);
                            mymap#addBar tmp;
                            pointMap := StringMap.add (string_of_int(barrier.i_point.x) ^
"," ^ string_of_int(barrier.i_point.y)) v1Name !pointMap;
                            num_items := !num_items +1;
                        else
                            raise (Failure ("Data type mismatch: This item is not
BARRIER")))
                    else
                        raise (Failure ("This point has been used (addBarrier(Item))")));
            else
                raise (Failure ("Data type mismatch: " ^ getType v)));
        Int(0), env
    (* built-in function addBarrier(item) *)
    | Call("addBarrier", [e1 ; e2]) ->
        let v1, env = eval env e1 in
        let v2, env = eval env e2 in
            (if getType v1 = "int" && getType v2 = "int" then
                (if getInt v1 >=1 && getInt v1 <=13 && getInt v2 >=1 && getInt v2 <=13
then
                    ignore(
                    if ((StringMap.mem (string_of_int(getInt v1) ^ "," ^
string_of_int(getInt v2)) !pointMap) = false)
then
                        let tmp = new item in
                        tmp#setType 0;
                        tmp#getPoint#setX (getInt v1);
                        tmp#getPoint#setY (getInt v2);
                        mymap#addBar tmp;
                        pointMap := StringMap.add (string_of_int((getInt v1)) ^ "," ^
string_of_int((getInt v2))) ("barrier_tmp","") !pointMap;
                        num_items := !num_items +1;
                    else
                        raise (Failure ("This point has been used (addBarrier(int x,int
y))")))
                else
                    raise (Failure ("Value error: The value should be between 1 to 13.
( arg1=" ^ string_of_int(getInt v1) ^ ", arg2=" ^ string_of_int(getInt v2) ^ ".) ")))
            else
                raise (Failure ("Data type mismatch: arg1=" ^ getType v1 ^ ", arg2=" ^
getType v2 ^ ".")));
        Int(0), env
    (* built-in function addGift(item) *)
    | Call("addGift", [e]) ->
        let v, env = eval env e in
        let v1Info =
            match e with
                Id (i) -> ("id", (i, ""))
            | _ -> raise (Failure ("left side of assignment must be an identifier or
```

```
member access")) in
        let v1Name = snd v1Info in
            (if getType v = "item"  then
                let gift = getItem(v) in
                    (if((StringMap.mem (string_of_int(gift.i_point.x) ^ "," ^
string_of_int(gift.i_point.y)) !pointMap) = false) then
                        ignore(
                        if gift.i_type = 5 then
                            let tmp = new item in
                            tmp#setType 2;
                            tmp#getPoint#setX (gift.i_point.x*100+50);
                            tmp#getPoint#setY (gift.i_point.y*100+50);
                            tmp#setColor gift.color;
                            tmp#setLevel gift.level;
                            tmp#setRadius 0.35;
                            tmp#setDuration 200;
                            mymap#addGift tmp;
                            pointMap := StringMap.add (string_of_int(gift.i_point.x) ^
"," ^ string_of_int(gift.i_point.y)) v1Name !pointMap;
                            num_items := !num_items +1;
                        else
                            raise (Failure ("Data type mismatch: This item is not
GIFT")))
                    else
                        raise (Failure ("This point has been used (addGift)")));
            else
                raise (Failure ("Data type mismatch: " ^ getType v)));
          Int(0), env

    (* built-in function getMapWidth() *)
    | Call("getMapWidth", []) ->
        Int(15), env

    (* built-in function getMapHeight() *)
    | Call("getMapHeight", []) ->
        Int(15), env
    (* built-in function play() *)
    | Call("play", []) ->
        show_pacman  mymap;
        Int(0), env
    (* built-in function getMapItemName(point) *)
    | Call("getMapItemName", [e]) ->
        let v, env = eval env e in
            (if getType v = "point"  then
                let point = getPoint(v) in
                    (if((StringMap.mem (string_of_int(point.x) ^ "," ^
string_of_int(point.y)) !pointMap) = true) then
                        String(fst (StringMap.find (string_of_int(point.x) ^ "," ^
string_of_int(point.y)) !pointMap)),env
                    else
                        String(""),env)
            else
                raise (Failure ("Data type mismatch: " ^ getType v)));
    (* built-in function isPointAvailable(point) *)
    | Call("isPointAvailable", [e]) ->
        let v, env = eval env e in
            (if getType v = "point"  then
                let point = getPoint(v) in
                    (if((StringMap.mem (string_of_int(point.x) ^ "," ^
string_of_int(point.y)) !pointMap) = false) then
                        Bool(true),env
                    else
                        Bool(false),env)
            else
                raise (Failure ("Data type mismatch: " ^ getType v)));
    (* built-in function getAvailablePoint(point) *)
    | Call("getAvailablePoint", []) ->
        if !num_items < 169 then
            let r = ref 1 in
                while !r > 0 do
                    avail_x := (Random.int 13)+1;
                    avail_y := (Random.int 13)+1;
                    if((StringMap.mem (string_of_int(!avail_x) ^ "," ^
string_of_int(!avail_y)) !pointMap) = false) then
                        r :=0;
                done;
                Point({x=(!avail_x); y=(!avail_y)}) ,env
        else
            raise (Failure ("No available space."));
    (* built-in function print(DATATYPE) *)
    | Call("print", [e]) ->
```

```
            let v, env = eval env e in
                (if getType v = "int" then
                    print_endline (string_of_int (getInt v))
                else if getType v = "string" then
                    print_endline (getString v)
                else if getType v = "bool" then
                    print_endline (string_of_bool (getBool v))
                else if getType v = "point" then
                    print_endline ("(" ^ string_of_int (getPoint v).x  ^ " , " ^
string_of_int (getPoint v).y ^ ")")
                else
                    print_endline(getType v));
          Int(0), env

    | Call(f, actuals) -> (* check *)
        let fdecl =
            try NameMap.find f func_decls
            with Not_found -> raise (Failure ("Undefined function: " ^ f))
        in
        let actuals, env = List.fold_left
            (fun (actuals, env) actual ->
            let v, env = eval env actual in v :: actuals, env)
            ([], env) actuals
        in
        let (locals, globals) = env in
        try
            let globals = call fdecl (List.rev actuals) globals
            in Bool false, (locals, globals)
        with ReturnException(v, globals) -> v, (locals, globals)
    in
    (* Execute a statement and return an updated environment *)
    let rec exec env = function
        Block(stmts) -> List.fold_left exec env stmts
        | Expr(e) -> let _, env = eval env e in env
        | If(e, s1, s2) ->
            let v, env = eval env e in
            exec env (if getBool v != false then s1 else s2)
        | While(e, s) ->
            let rec loop env =
                let v, env = eval env e in
                if getBool v != false then loop (exec env s) else env
            in loop env
        | For(e1, e2, e3, s) ->
            let _, env = eval env e1 in
            let rec loop env =
                let v, env = eval env e2 in
                if getBool v != false then
                    let _, env = eval (exec env s) e3 in
                    loop env
                else
                    env
            in loop env
        | Return(e) ->
            let v, (locals, globals) = eval env e in
                if getType v = fdecl.rettype then
                    raise (ReturnException(v, globals))
                else
                    raise (Failure ("Function " ^ fdecl.fname ^ " returns " ^ getType v ^
" instead of " ^ fdecl.rettype))
    in (* done with below *)
    (* call: enter the function: bind actual values to formal args *)
    let locals =
        try List.fold_left2
            (fun locals formal actual ->
            NameMap.add formal.paramname actual locals)
        NameMap.empty fdecl.formals actuals
        with Invalid_argument(_) ->
            raise (Failure ("Wrong number of arguments to: " ^ fdecl.fname))
    in
    let locals = List.fold_left (* Set local variables to 0 *)
        (fun locals local -> NameMap.add local.varname (init local.vartype) locals)
        locals fdecl.locals
    in (* Execute each statement; return updated global symbol table *)
        snd (List.fold_left exec (locals, globals) fdecl.body)

    (* run: set global variables to 0; find and run "main" *)
    in let globals = List.fold_left
        (fun globals vdecl -> NameMap.add vdecl.varname (init vdecl.vartype) globals) (*
vdecl is the identifier, "X" *)
        NameMap.empty vars
    in try
```

```
        call (NameMap.find "main" func_decls) [] globals
    with Not_found ->
        raise (Failure ("Did not find the main() function"))
```

Appendix A.5 – game.ml

```
open Classes;;
(*  initial map  *)
let map_bar = [|[|1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1|];
                [|1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1|];
                [|1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1|];
                [|1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1|];
                [|1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1|];
                [|1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1|];
                [|1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1|];
                [|1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1|];
                [|1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1|];
                [|1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1|];
                [|1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1|];
                [|1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1|];
                [|1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1|];
                [|1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1|];
                [|1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1|]|]
;;
let gmap = new gamemap;;
let map_width = float(15);;
let map_height = float(15);;
let pi = 3.14159265;;
let r = 0.49;;
let v = 0.05;;
let width = 1.;;
let colorb = (0.2, 0.2, 0.8);;
let total_score = ref 0;;
let pacman = new player;;
let ghost_g = new item;;
let gift_g = new item;;
let px = ref 7.5;;
let py = ref 7.5;;
let vx = ref 0.0;;
let vy = ref 0.0;;
let next_dir = ref 0;;
let now_dir = ref 0;;
let colorp = (1., 1., 0.);;
let counter = ref 0;;
let circle x px py r = [(sin x) *. r +. px, (cos x) *. r +. py];;
(*  check if the pacman and ghosts hit the bar *)
let check_hit_bar_x x y =
        if map_bar.(truncate (y /. width)).(truncate (x /. width)) = 1 ||
            map_bar.(truncate ((y +. r) /. width)).(truncate (x /. width)) = 1 ||
            map_bar.(truncate ((y -. r) /. width)).(truncate (x /. width)) = 1 then
            true
        else false
;;
let check_hit_bar_y x y =
        if map_bar.(truncate (y /. width)).(truncate (x /. width)) = 1 ||
           map_bar.(truncate (y /. width)).(truncate ((x +. r) /. width)) = 1 ||
           map_bar.(truncate (y /. width)).(truncate ((x -. r) /. width)) = 1 then
            true
        else false
;;
(*  check if the item need to change the direction *)
let check_item_change_dir item =
    (*if (vx = 0.) && ((abs_float (y-.float(truncate(y)))) < 0.05) &&*)
    if (item#getNowDir = 0 || item#getNowDir = 2) && (item#getMoveCounter <= 0) &&
((((item#getPoint)#getX-50) mod 100) < item#getSpeed) &&
       (map_bar.((item#getPoint)#getY / 100 + 1).((item#getPoint)#getX / 100) != 1 ||
        map_bar.((item#getPoint)#getY / 100 - 1).((item#getPoint)#getX / 100) != 1) then
            true
    else if (item#getNowDir = 1 || item#getNowDir = 3) && (item#getMoveCounter <= 0) &&
((((item#getPoint)#getY-50) mod 100) < item#getSpeed) &&
       (map_bar.((item#getPoint)#getY / 100).((item#getPoint)#getX / 100 + 1) != 1 ||
        map_bar.((item#getPoint)#getY / 100).((item#getPoint)#getX / 100 - 1) != 1) then
            true
    else if ((item#getVelocity)#getX = 0) && ((item#getVelocity)#getY = 0) then
            true
    else false;
```

```
;;
(*  set the velocity of an item *)
let set_item_v vxx vyy item =
    (item#getVelocity)#setX vxx;
    (item#getVelocity)#setY vyy;
    item#setNowDir item#getNextDir;
;;
(*  read from keyboard *)
let keyboard ~key ~x ~y =
    match key with
        | 27 -> exit 0
        | 113 -> exit 0
        | _ -> ignore (Printf.printf "Key: %c %d pressed.\n%!" (char_of_int key) key)
;;
let special_key ~key ~x ~y =
    match key with
        | Glut.KEY_LEFT -> pacman#setNextDir 0
        | Glut.KEY_UP -> pacman#setNextDir 1
        | Glut.KEY_RIGHT -> pacman#setNextDir 2
        | Glut.KEY_DOWN -> pacman#setNextDir 3
        | _ -> print_endline("Special key pressed")
;;
(*  set the new location of an item *)
let move_item item =
    if (item#getNextDir = 0) &&
        not (check_hit_bar_x (((float((item#getPoint)#getX) -. (float(item#getSpeed))) /.
100.) -. item#getRadius)
                             (float((item#getPoint)#getY) /. 100.))
        then set_item_v (- item#getSpeed) 0 item
    else if (item#getNextDir = 1) &&
        not (check_hit_bar_y  (float((item#getPoint)#getX) /. 100.)
                             (((float((item#getPoint)#getY) +. (float(item#getSpeed))) /.
100.) +. item#getRadius))
        then set_item_v 0 item#getSpeed item
    else if (item#getNextDir = 2) &&
        not (check_hit_bar_x (((float((item#getPoint)#getX) +. (float(item#getSpeed))) /.
100.) +. item#getRadius)
                             (float((item#getPoint)#getY) /. 100.))
        then set_item_v item#getSpeed 0 item
    else if (item#getNextDir = 3) &&
        not (check_hit_bar_y  (float((item#getPoint)#getX) /. 100.)
                             (((float((item#getPoint)#getY) -. (float(item#getSpeed))) /.
100.) -. item#getRadius))
        then set_item_v 0 (- item#getSpeed) item
    else if (item#getNextDir = -1)
        then set_item_v 0 0 item;
    if (item#getVelocity)#getX > 0 &&
       (check_hit_bar_x ((float((item#getPoint)#getX) +. float((item#getVelocity)#getX))
/. 100. +. item#getRadius)
                        ((float((item#getPoint)#getY) +. float((item#getVelocity)#getY))
/. 100.) )
        then (item#getVelocity)#setX 0
    else if (item#getVelocity)#getX < 0 &&
       (check_hit_bar_x ((float((item#getPoint)#getX) +. float((item#getVelocity)#getX))
/. 100. -. item#getRadius)
                        ((float((item#getPoint)#getY) +. float((item#getVelocity)#getY))
/. 100.) )
        then (item#getVelocity)#setX 0
    else if (item#getVelocity)#getY > 0 &&
       (check_hit_bar_y ((float((item#getPoint)#getX) +. float((item#getVelocity)#getX))
/. 100.)
                        ((float((item#getPoint)#getY) +. float((item#getVelocity)#getY))
/. 100. +. item#getRadius) )
        then (item#getVelocity)#setY 0
    else if (item#getVelocity)#getY < 0 &&
       (check_hit_bar_y ((float((item#getPoint)#getX) +. float((item#getVelocity)#getX))
/. 100.)
                        ((float((item#getPoint)#getY) +. float((item#getVelocity)#getY))
/. 100. -. item#getRadius) )
        then (item#getVelocity)#setY 0;
    if ((item#getVelocity)#getX = 0) && ((item#getVelocity)#getY = 0) then item#setNowDir
(-1);
    (item#getPoint)#setX ((item#getPoint)#getX + (item#getVelocity)#getX);
    (item#getPoint)#setY ((item#getPoint)#getY + (item#getVelocity)#getY);
;;

(*  draw an item on the screen*)
let draw_item item =
    GlDraw.color (item#getColor);
    GlDraw.begins `polygon;
    for i=0 to 359 do
```

```
        List.iter GlDraw.vertex2 (circle (2. *. pi *. float i /. 360.)
(float((item#getPoint)#getX) /. 100.) (float((item#getPoint)#getY) /. 100.)
(item#getRadius))
    done;
    GlDraw.ends ();
;;
let r0 = ref 0;;
let r1 = ref 0;;
let r2 = ref 0;;
let r3 = ref 0;;
let r_array = [|r0; r1; r2; r3|];;
let d0 = ref 0;;
let d1 = ref 0;;
let d2 = ref 0;;
let d3 = ref 0;;
let dis_now = ref 0;;
let d_array = [|d0; d1; d2; d3|];;
let get_weight_random_dir ran =
    if ran < !r0 then 0
    else if ran < (!r0) + (!r1) then 1
    else if ran < (!r0) + (!r1) + (!r2) then 2
    else 3
;;
let set_bar_dir item =

    if  (check_hit_bar_x (((float((item#getPoint)#getX) -. (float(item#getSpeed))) /.
100.) -. item#getRadius)
                         (float((item#getPoint)#getY) /. 100.))
        then r0 := 1;
    if  (check_hit_bar_y  (float((item#getPoint)#getX) /. 100.)
                          (((float((item#getPoint)#getY) +. (float(item#getSpeed))) /.
100.) +. item#getRadius))
        then r1 := 1;
    if  (check_hit_bar_x (((float((item#getPoint)#getX) +. (float(item#getSpeed))) /.
100.) +. item#getRadius)
                         (float((item#getPoint)#getY) /. 100.))
        then r2 := 1;
    if  (check_hit_bar_y  (float((item#getPoint)#getX) /. 100.)
                          (((float((item#getPoint)#getY) -. (float(item#getSpeed))) /.
100.) -. item#getRadius))
        then r3 := 1;
;;
(*  calculate the distance between pacman and ghost *)
let calculate_dist item =
    d0 := abs((item#getPoint#getX - 100) - pacman#getPoint#getX) +
          abs((item#getPoint#getY) - pacman#getPoint#getY);
    d1 := abs((item#getPoint#getX) - pacman#getPoint#getX) +
          abs((item#getPoint#getY + 100) - pacman#getPoint#getY);
    d2 := abs((item#getPoint#getX + 100) - pacman#getPoint#getX) +
          abs((item#getPoint#getY) - pacman#getPoint#getY);
    d3 := abs((item#getPoint#getX) - pacman#getPoint#getX) +
          abs((item#getPoint#getY - 100) - pacman#getPoint#getY);
    dis_now := abs((item#getPoint#getX) - pacman#getPoint#getX) +
               abs((item#getPoint#getY) - pacman#getPoint#getY);
;;
(*  difficulty level *)
let lv2_ai item =
    r0 := 25;
    r1 := 25;
    r2 := 25;
    r3 := 25;
    if item#getNowDir >=0 && item#getNowDir <=3 then
        r_array.(item#getNowDir) := 5;
    calculate_dist item;
    for i=0 to 3 do
        if (!(d_array.(i)) < !dis_now) then
            r_array.(i) := 2000;
    done;
    set_bar_dir item;
    get_weight_random_dir (Random.int (!r0 + !r1 + !r2 + !r3))
;;
let lv1_ai item =
    r0 := 25;
    r1 := 25;
    r2 := 25;
    r3 := 25;
    if item#getNowDir >=0 && item#getNowDir <=3 then
        r_array.(item#getNowDir) := 5;
    calculate_dist item;
    for i=0 to 3 do
        if (!(d_array.(i)) < !dis_now) then
```

```
            r_array.(i) := 100;
    done;
    set_bar_dir item;
    get_weight_random_dir (Random.int (!r0 + !r1 + !r2 + !r3))
;;
let lv0_ai item =
    r0 := 25;
    r1 := 25;
    r2 := 25;
    r3 := 25;
    set_bar_dir item;
    get_weight_random_dir (Random.int (!r0 + !r1 + !r2 + !r3))
;;
let lv_array = [|lv0_ai; lv1_ai; lv2_ai|];;
let set_item_next_dir item =
    item#setMoveCounter 13;
    item#setNextDir (lv_array.(item#getLevel) item);
;;
(*  set the pacman as a global variable*)
let set_pacman player =
   pacman#setSpeed player#getSpeed;
   pacman#getPoint#setX player#getPoint#getX;
   pacman#getPoint#setY player#getPoint#getY;
   pacman#setColor player#getColor;
;;
(*  generate gifts  *)
let gene_gift mainMap x y =
        let gift = new item in
        gift#setType 2;
        (gift#getPoint)#setX ((x*100)+50);
        (gift#getPoint)#setY ((y*100)+50);
        gift#setColor (0.3, 0.3, 0.);
        gift#setRadius 0.1;
        mainMap#addGift gift
;;
let generate_g mainMap =
    let rec draw_into_map b_list =
        match b_list with
        []->()
        | hd::tl ->
            map_bar.(hd#getPoint#getY).(hd#getPoint#getX) <- 1;
            draw_into_map tl
    in draw_into_map mainMap#getBars;
    let x = ref 0 in
    let y = ref 0 in
    for i=1 to 20 do
        while map_bar.(!y).(!x) != 0 do
            x := 1 + (Random.int 13);
            y := 1 + (Random.int 13);
        done;
        gene_gift mainMap !x !y;
        x := 0;
        y := 0;
    done
;;
(* set the map as a global variable  *)
let set_map mainMap =
   gmap#setPlayer mainMap#getPlayer;
   gmap#assignGhosts mainMap#getGhosts;
   gmap#assignGifts mainMap#getGifts;
   gmap#assignBars mainMap#getBars;
   set_pacman mainMap#getPlayer;
   generate_g gmap;
;;
(*  print message and exit *)
let game_over message =
    print_string("Score: ");
    print_int(!total_score);
    print_endline("");
    print_endline(message);
    exit 0;
;;
(* control and draw pacman  *)
let show_pacman mainMap =

  set_map mainMap;
  ignore (Glut.init Sys.argv);
  Glut.initDisplayMode ~double_buffer:true ();
  Glut.initWindowSize ~w:512 ~h:512;
  ignore (Glut.createWindow ~title:"OpenGL Demo");
  GlClear.color (0.0, 0.0, 0.0);
```

```
GlDraw.shade_model `smooth;
let render () =
  GlMat.mode `projection;
  GlMat.load_identity ();
  GluMat.ortho2d ~x:(0., map_width) ~y:(0., map_height);
  GlMat.mode `modelview;
  GlMat.load_identity ();
  GlClear.clear [ `color ];

  let change_back p =
      p#setColor (1., 1., 0.);
      p#setSpeed 5;
      p#setStatus "normal";
  in let deal_status p =
      p#setTime (p#getTime - 1);
      if p#getTime <= 0 then
          change_back p;
  in let check_status p =
      if not (p#getStatus = "normal") then
          deal_status p;
  in check_status pacman;
  move_item pacman;
  let g4 g p =
      p#setStatus  "normal";
      total_score := !total_score + 100;
  in let g3 g p =
      p#setStatus  "killer";
      p#setTime  g#getDuration;
      p#setColor g#getColor;
  in let g1 g p =
      p#setSpeed  10;
      p#setStatus  "fast";
      p#setTime  g#getDuration;
      p#setColor g#getColor;
  in let g2 g p =
      p#setSpeed  1;
      p#setStatus  "slow";
      p#setTime  g#getDuration;
      p#setColor g#getColor;
  in let deal_gift g p =
      total_score := (!total_score + g#getScore);
      print_string("Score: ");
      print_int(!total_score);
      print_endline("");
      if g#getLevel = 1 then
          g1 g p
      else if g#getLevel = 2 then
          g2 g p
      else if g#getLevel = 3 then
          g3 g p
      else if g#getLevel = 4 then
          g4 g p;
      g#setLevel (- 1);
      if(List.length gmap#getGifts) = 1 then
          game_over("You Win!!!!");
  in let check_gift g p =
      if (abs (p#getPoint#getX - g#getPoint#getX) +
          abs (p#getPoint#getY - g#getPoint#getY) )
          < (truncate((p#getRadius +. g#getRadius) *. 100.))
          then
              deal_gift g p
      else
          draw_item g;
  in let rec check_gifts glist player =
      match glist with
      []->[]
      | g1::tl ->
          check_gift g1 player;
          if ((g1#getLevel) != -1) then
              g1::(check_gifts tl player)
          else
              check_gifts tl player
  in
  gmap#assignGifts (check_gifts gmap#getGifts pacman);
  draw_item pacman;
 let move_ghost ghost =
      if (check_item_change_dir ghost) then
          set_item_next_dir ghost;
      move_item ghost;
      ghost#setMoveCounter (ghost#getMoveCounter -1);
  in let rec move_ghosts mlist =
```

```
        match mlist with
        []->()
        | m1::tl ->
            move_ghost m1;
            draw_item m1;
            move_ghosts tl;
    in move_ghosts gmap#getGhosts;
    let kill_ghost m =
        m#setLevel (- 1);
        total_score := !total_score + (10 * m#getLevel);
    in let deal_ghost m p =
        if p#getStatus = "killer" then
            kill_ghost m
        else
            game_over("Game Over -- You Lose!!! -- You are killed by a ghost");
    in let check_ghost m p =
        if (abs (p#getPoint#getX - m#getPoint#getX) +
            abs (p#getPoint#getY - m#getPoint#getY) )
            < (truncate((p#getRadius +. m#getRadius) *. 100.))
            then
                deal_ghost m p;
    in let rec check_ghosts mlist player =
        match mlist with
        []->[]
        | m1::tl ->
            check_ghost m1 player;
            if ((m1#getLevel) != -1) then
                m1::(check_ghosts tl player)
            else
                check_ghosts tl player
    in
    gmap#assignGhosts (check_ghosts gmap#getGhosts pacman);
    GlDraw.color colorb;
    let draw_box x y width =
        GlDraw.begins `line_loop;
        List.iter GlDraw.vertex2 [x *. width, y *. width;
                                  (x +. width ) *. width, y *. width;
                                  (x +. width ) *. width, (y +.  width ) *. width;
                                  x *. width, (y +.  width ) *. width];
        GlDraw.ends ();
    in
    let rec print_map_raw r x y =
        if float(x) < map_width then
            if r.(x) = 1 then
                draw_box (float(x)) (float(y)) width;
        if float(x) < map_width then
            print_map_raw r (x+1) y;
    in let rec print_map_bar m y =
        if float(y) < map_height then
            print_map_raw m.(y) 0 y;
        if float(y) < map_height then
            print_map_bar m (y+1);
    in print_map_bar map_bar 0;
    Gl.flush ();
    Glut.swapBuffers () in
  Glut.displayFunc ~cb:render;
  Glut.idleFunc ~cb:(Some Glut.postRedisplay);
  Glut.keyboardFunc ~cb:(keyboard);
  Glut.specialFunc ~cb:(special_key);
  Glut.mainLoop ()
;;
```

Appendix A.6 – classes.ml

```
class point =
  object
     val mutable x = 0
     val mutable y = 0
     method setX new_x = x <- new_x
     method setY new_y = y <- new_y
     method getX = x
     method getY = y
  end;;
class player =
  object
    val mutable speed = 0
```

```
    val mutable status = "normal"
    val mutable p = new point
    val mutable v = new point
    val mutable color = (1.,1.,0.)
    val mutable remaining_time = 0
    val mutable next_dir = 0
    val mutable now_dir = 0
    val mutable move_counter = 0
    val mutable r = 0.49
    method setSpeed new_speed = speed <- new_speed
    method getSpeed = speed
    method setStatus new_status = status <- new_status
    method getStatus = status
    method setPoint new_point = p <- new_point
    method getPoint = p
    method setVelocity new_point = v <- new_point
    method getVelocity = v
    method setColor new_color = color <- new_color
    method getColor = color
    method setTime new_time = remaining_time <- new_time
    method getTime = remaining_time
    method setNextDir new_next_dir = next_dir <- new_next_dir
    method getNextDir = next_dir
    method setNowDir new_now_dir = now_dir <- new_now_dir
    method getNowDir = now_dir
    method setMoveCounter new_move_counter = move_counter <- new_move_counter
    method getMoveCounter = move_counter
    method setRadius new_radius = r <- new_radius
    method getRadius = r
  end;;
class item =
  object
    val mutable speed = 0
    val mutable i_type = 0
    val mutable duration = 0
    val mutable p = new point
    val mutable v = new point
    val mutable color = (1.,1.,1.)
    val mutable r = 0.0
    val mutable level = 0
    val mutable score = 10
    val mutable next_dir = 0
    val mutable now_dir = 0
    val mutable move_counter = 0
    method setSpeed new_speed = speed <- new_speed
    method getSpeed = speed
    method setType new_type = i_type <- new_type
    method getType = i_type
    method setDuration new_duration = duration <- new_duration
    method getDuration = duration
    method setPoint new_point = p <- new_point
    method getPoint = p
    method setVelocity new_point = v <- new_point
    method getVelocity = v
    method setColor new_color = color <- new_color
    method getColor = color
    method setNextDir new_next_dir = next_dir <- new_next_dir
    method getNextDir = next_dir
    method setNowDir new_now_dir = now_dir <- new_now_dir
    method getNowDir = now_dir
    method setMoveCounter new_move_counter = move_counter <- new_move_counter
    method getMoveCounter = move_counter
    method setRadius new_radius = r <- new_radius
    method getRadius = r
    method setLevel new_level = level <- new_level
    method getLevel = level
    method setScore new_score = score <- new_score
    method getScore = score
  end;;
class gamemap =
  object
    val mutable pacman = new player
    val mutable list_bar: item list = []
    val mutable list_ghost: item list = []
    val mutable list_gift: item list = []
    method setPlayer new_player = pacman <- new_player
    method getPlayer = pacman
    method addBar new_item = list_bar <- new_item::list_bar
    method getBars = list_bar
    method addGhost new_item = list_ghost <- new_item::list_ghost
    method getGhosts = list_ghost
```

```
    method addGift new_item = list_gift <- new_item::list_gift
    method getGifts = list_gift
    method assignGhosts new_list = list_ghost <- new_list;
    method assignBars new_list = list_bar <-new_list;
    method assignGifts new_list = list_gift <-new_list;
  end;;
```

Appendix A.7 – pacaml.ml

```
let _ =
  let lexbuf = Lexing.from_channel stdin in
  let program = Parser.program Scanner.token lexbuf in
  ignore (Interpret.run program)
```

Appendix B – Makefile

```
OBJS =    classes.cmo scanner.cmo game.cmo ast.cmo parser.cmo interpret.cmo pacaml.cmo
TARFILES = Makefile scanner.mll parser.mly \
      ast.mli myclasses.ml game.ml interpret.ml pacaml.ml \
pacaml : $(OBJS)
      ocamlc -I +lablGL lablgl.cma lablglut.cma str.cma unix.cma -o pacaml $(OBJS)

scanner.ml : scanner.mll
      ocamllex scanner.mll

parser.ml parser.mli : parser.mly
      ocamlyacc parser.mly
game.cmo: classes.cmo game.ml
      ocamlc -I +lablGL lablgl.cma lablglut.cma str.cma unix.cma classes.cmo game.ml
%.cmo : %.ml
      ocamlc -c $<

%.cmi : %.mli
      ocamlc -c $<
clean :
      rm -f pacaml parser.ml parser.mli scanner.ml a.out *.cmo *.cmi
ast.cmo:
ast.cmx:
interpret.cmo: ast.cmo
interpret.cmx: ast.cmx
pacaml.cmo: scanner.cmo parser.cmi interpret.cmo \
   ast.cmo
pacaml.cmx: scanner.cmx parser.cmx interpret.cmx \
   ast.cmx
parser.cmo: ast.cmo parser.cmi
parser.cmx: ast.cmx parser.cmi
scanner.cmo: parser.cmi
scanner.cmx: parser.cmx
parser.cmi: ast.cmo
```

Appendix C.1 – hello_world.pacaml

```
/*  Test print function */

void main()
{

    string str;
    str = "helloworld";
    print(str);

    print("Helloworld2");
}
```

Appendix C.2 – hello test_arith.pacamll

```
/*    Test the integer arithmetic functionality*/

void main()
{
    //Test point arithmetic
    point p1;
    point p2;
    p1.x = 4;
    p1.y = 3;
    p2.x = 1;
    p2.y = 2;


    print("Plus");
    print(p1+p2);
    print("Minus");
    print(p1-p2);

    print("Multiply");
    print(p1*p2);

    print("Divide");
    print(p1/p2);
    //Test integer arithmetic
    print("Integer");
    print( 1 + 5 );
    print( 5 - 2 );
    print( 4 * 2 );
    print( 1 / 2 );
    print( 5 % 3 );

    //Order of operations with integer arithmetic
    print( 1 + 2 * 3 + 4 );
    print( ( 1 + 2 ) * ( 3 + 4 ) );
}
```

Appendix C.3 – test_assignment.pacaml

```
/*     Test the functionality of assignment */
void main()
{
        int a;
        int b;
        int c;
        point p;
        a = 1;

        //Assigning variables
        print(a);
        print(b=a);

        //Assigning members
        p.x = 6;
         p.y = 2;
        print(p.x);
```

```
        print(p.y);

        //Successive assignment
        print(a = b = c = p.x = p.y = 13);
        print(a);
        print(b);
        print(c);
        print(p.x);
        print(p.y);

        //Testing additional assignment operators
        print(a += 1);
        print(a++);    //++ is equivalent to x += 1;
        print(a -= 1);
        print(a--);    //-- is equivalent to x -= 1;
        print(a *= 10);
        print(a /= 2);
        print(a %= 2);

        //Self-assignment
        print(a = a);
}
```

Appendix C.4 – test_fib.pacaml

```
/*      Test recursive  */
int fib( int x )
{
    if (x < 2)
    {
        return 1;
    }
    else
    {
        return fib(x-1) + fib(x-2);
    }
}
void main()
{
        print(fib(0));
        print(fib(1));
        print(fib(2));
        print(fib(3));
        print(fib(4));
        print(fib(5));
}
```

Appendix C.5 – test_for.pacaml

```
/*      Testing the for statement in m. */

void main()
{
        int i;

        for( i = 0; i < 10; i++)
        {
                print(i);
        }
}
```

Appendix C.6 – test_func.pacaml

```
/*      Testing function   */
int plus( int a, int b )
{
        return a + b;
}
```

```
int minus( int a, int b)
{
	return a - b;
}
void main()
{
	print(plus( 1, 2 ));
	print(minus( 1, 2 ));
}
```

Appendix C.7 – test_gcd.pacaml

```
/*	Testing a more complex function in m. */
int gcd(int a, int b)
{
	while (a != b)
	{
		if(a > b)
		{
			a = a - b;
		}
		else
		{
			b = b - a;
		}
	}

	return a;
}
void main()
{
	print(gcd(4, 14));
	print(gcd(6,18));
	print(gcd(169,65));
}
```

Appendix C.8 – hello test_if.pacaml

```
/*	Test the if statement */

void main()
{
	if(1)
	{
		print( 0 );
	}

	if(0)
	{
		print( 1 );
	}
	else
	{
		print( 2 );
	}
}
```

Appendix C.9 – test_logic.pacaml

```
/*	Test the logic functionality  */

void main()
{

	print("NOT");
	print( !(1==1) );
	print( !(1==2) );
```

```
        print("AND");
        print( (1==1) && (1==1) );   // ( true && true)
        print( (1==1) && (1==2) );   // ( true && false)
        print( (1==2) && (1==1) );   // ( false && true)
        print( (1==2) && (1==2) );   // ( false && false)
        print("OR");
        print( (1==1) || (1==1) );   // ( true || true)
        print( (1==1) || (1==2) );   // ( true || false)
        print( (1==2) || (1==1) );   // ( false || true)
        print( (1==2) || (1==2) );   // ( false || false)

        print("NotEqual");
        print( 1 != 2 );
        print( 1 != 1 );
        print("less");
        print( 1 < 2 );
        print( 2 < 1 );
        print("LessEqual");
        print( 1 <= 2 );
        print( 1 <= 1 );
        print( 2 <= 1 );
        print("Greater");
        print( 1 > 2 );
        print( 2 > 1 );
        print("GreaterEqual");
        print( 1 >= 2 );
        print( 1 >= 1 );
        print( 2 >= 1 );

        print("Order");
        print( !(1==1) || (1==1) );
        print( !( (1==1) || (1==1) ) );
}
```

Appendix C.10 – test_objects.pacaml

```
/*      Test pacaml built-in objects. */
item createItem( int type, int x, int y)
{
        item i;
        point p;
        p.x = x;
        p.y = y;
        i.i_type = type;
        i.i_point = p;
        return i;
}
void main()
{
        player pacman;
        item ghost1;
        item barrier1;
        item gift1;
        pacman.p_point = getAvailablePoint();
        setPlayer(pacman);
        ghost1 = createItem(_GHOST, 13, 5);
        addGhost(ghost1);
        barrier1 = createItem(_BARRIER, 8 , 8);
        addBarrier(barrier1);
        gift1 = createItem(_GIFT, 8 , 9 );
        addGift(gift1);
        play();
}
```

Appendix C.11 – test_pacman.pacaml

```
void main()
{
player pac;
item ghost1;
item ghost2;
item ghost3;
```

```
item gift1;
item gift2;
item gift3;
item gift4;
point p1;
addBarrier(1,10);
addBarrier(2,12);
addBarrier(2,10);
addBarrier(2,8);
addBarrier(2,6);
addBarrier(2,5);
addBarrier(2,3);
addBarrier(2,2);
addBarrier(3,12);
addBarrier(3,10);
addBarrier(3,8);
addBarrier(3,6);
addBarrier(3,5);
addBarrier(3,2);
addBarrier(4,12);
addBarrier(4,8);
addBarrier(4,4);
addBarrier(4,2);
addBarrier(5,11);
addBarrier(5,10);
addBarrier(5,8);
addBarrier(5,6);
addBarrier(5,2);
addBarrier(6,12);
addBarrier(6,6);
addBarrier(6,5);
addBarrier(6,4);
addBarrier(7,10);
addBarrier(7,9);
addBarrier(7,8);
addBarrier(7,6);
addBarrier(7,2);
addBarrier(8,12);
addBarrier(8,9);
addBarrier(8,8);
addBarrier(8,6);
addBarrier(8,4);
addBarrier(8,2);
addBarrier(9,11);
addBarrier(9,9);
addBarrier(9,6);
addBarrier(10,13);
addBarrier(10,11);
addBarrier(10,9);
addBarrier(10,7);
addBarrier(10,6);
addBarrier(10,5);
addBarrier(10,4);
addBarrier(10,2);
addBarrier(11,9);
addBarrier(11,7);
addBarrier(11,2);
addBarrier(12,12);
addBarrier(12,11);
addBarrier(12,9);
addBarrier(12,5);
addBarrier(12,4);
addBarrier(12,2);
addBarrier(13,7);
// set pacman
p1 = getAvailablePoint();     // randomly choose a point
print(isPointAvailable(p1));  //test isPointAvailable(Point)
pac.p_point = p1;
setPlayer(pac);
print(getMapItemName(p1));
// add Gift 1
p1 = getAvailablePoint();
gift1.i_type = _GIFT;
gift1.level = _GIFT_KILLER;
gift1.i_point = p1;
addGift(gift1);
print(getMapItemName(p1));
// add Gift 2
p1 = getAvailablePoint();
gift2.i_type = _GIFT;
gift2.level = _GIFT_SPEEDUP;
```

```
gift2.i_point = p1;
addGift(gift2);
print(getMapItemName(p1));
// add Gift 3
p1 = getAvailablePoint();
gift3.i_type = _GIFT;
gift3.level = _GIFT_SLOWDOWN;
gift3.i_point = p1;
addGift(gift3);
print(getMapItemName(p1));
// add Gift 4
p1 = getAvailablePoint();
gift4.i_type = _GIFT;
gift4.level = _GIFT_SCORE;
gift4.i_point = p1;
addGift(gift4);
print(getMapItemName(p1));
// add monster 1
p1 = getAvailablePoint();
ghost1.i_type = _GHOST;
ghost1.level = _EASY;
ghost1.i_point = p1;
addGhost(ghost1);
print(getMapItemName(p1));
// add monster 2
p1 = getAvailablePoint();
ghost2.i_type = _GHOST;
ghost2.level = _NORMAL;
ghost2.i_point = p1;
addGhost(ghost2);
print(getMapItemName(p1));
// add monster 3
p1 = getAvailablePoint();
ghost3.i_type = _GHOST;
//ghost3.level = _HARD;
ghost3.i_point = p1;
addGhost(ghost3);
print(getMapItemName(p1));
//start
play();
}
```

Appendix C.12 – test_while.pacaml

```
/*      Test the while statement. */

void main()
{
        int i;
        i = 0;
        while( i < 10 )
        {
                print(i);
                i++;
        }
}
```

Appendix D – SVN Log

---
r67 | soga1113 | 2011-12-22 04:49:48 -0400 (Thu, 22 Dec 2011) | 1 line

test case updated
---
r66 | soga1113 | 2011-12-22 04:19:40 -0400 (Thu, 22 Dec 2011) | 1 line

print point, point operators
---
r65 | soga1113 | 2011-12-21 22:40:45 -0400 (Wed, 21 Dec 2011) | 1 line

datatype string bug fixed; comment done
---
r64 | soga1113 | 2011-12-21 05:50:04 -0400 (Wed, 21 Dec 2011) | 8 lines

1. test case done

2. print("thisisstring") done; A string cannot contain space

3. not equal bug fixed

---
r63 | soga1113 | 2011-12-21 04:24:37 -0400 (Wed, 21 Dec 2011) | 1 line

error message checked
---
r62 | soga1113 | 2011-12-21 03:50:34 -0400 (Wed, 21 Dec 2011) | 1 line

---
r61 | soga1113 | 2011-12-21 01:46:45 -0400 (Wed, 21 Dec 2011) | 1 line

rename to .pacaml
---
r60 | soga1113 | 2011-12-21 01:45:02 -0400 (Wed, 21 Dec 2011) | 1 line

rename
---
r59 | cck73823@gmail.com | 2011-12-20 20:35:19 -0400 (Tue, 20 Dec 2011) | 1 line

add random points:317
---

r58 | soga1113 | 2011-12-20 20:22:18 -0400 (Tue, 20 Dec 2011) | 1 line

gift color fixed
---
r57 | soga1113 | 2011-12-20 20:18:44 -0400 (Tue, 20 Dec 2011) | 1 line

check number of items. should be less then 169.
---
r56 | soga1113 | 2011-12-20 19:54:28 -0400 (Tue, 20 Dec 2011) | 1 line

change gift color and type of gift
---
r55 | soga1113 | 2011-12-20 19:03:13 -0400 (Tue, 20 Dec 2011) | 1 line

bar index fixed
---
r54 | cck73823@gmail.com | 2011-12-20 19:01:43 -0400 (Tue, 20 Dec 2011) | 1 line

add bar into map
---
r53 | soga1113 | 2011-12-20 15:45:38 -0400 (Tue, 20 Dec 2011) | 1 line

item radius int to float
---
r52 | soga1113 | 2011-12-20 15:02:46 -0400 (Tue, 20 Dec 2011) | 1 line

reset myclass item radius and duration
---
r51 | soga1113 | 2011-12-20 14:59:26 -0400 (Tue, 20 Dec 2011) | 1 line

update gift radius and duration
---
r50 | cck73823@gmail.com | 2011-12-20 14:10:57 -0400 (Tue, 20 Dec 2011) | 1 line

add exit in game_over
---
r49 | cck73823@gmail.com | 2011-12-20 13:50:25 -0400 (Tue, 20 Dec 2011) | 1 line

add giftssvn up
---
r48 | soga1113 | 2011-12-20 06:40:32 -0400 (Tue, 20 Dec 2011) | 12 lines

1. point: not used as reference. the value of a point can
   be changed and reassign to another item or player

2. add a new function "addBarrier(int x, int y)." This function

is provided for easily adding barriers.

3. The difficulty level: _MEDIAN has been changed to _NORMAL

4. all the functions defined in the language manual are implemented.

---
r47 | soga1113 | 2011-12-20 04:18:02 -0400 (Tue, 20 Dec 2011) | 1 line

function getAvailablePoint() done
---
r46 | soga1113 | 2011-12-20 02:36:06 -0400 (Tue, 20 Dec 2011) | 1 line

random color fixed
---
r45 | soga1113 | 2011-12-20 02:33:57 -0400 (Tue, 20 Dec 2011) | 1 line

function isPointAvailable done
---
r44 | soga1113 | 2011-12-20 02:26:08 -0400 (Tue, 20 Dec 2011) | 1 line

point range updated
---
r43 | soga1113 | 2011-12-19 16:53:21 -0400 (Mon, 19 Dec 2011) | 1 line

check duplicate points
---
r42 | soga1113 | 2011-12-19 16:27:23 -0400 (Mon, 19 Dec 2011) | 1 line

scanner update
---
r41 | cck73823@gmail.com | 2011-12-19 16:23:43 -0400 (Mon, 19 Dec 2011) | 1 line

ghost level
---
r40 | cck73823@gmail.com | 2011-12-19 15:56:47 -0400 (Mon, 19 Dec 2011) | 1 line

combining
---
r39 | soga1113 | 2011-12-19 12:28:30 -0400 (Mon, 19 Dec 2011) | 1 line

update global map
---
r38 | soga1113 | 2011-12-19 04:59:32 -0400 (Mon, 19 Dec 2011) | 1 line

readme and makefile updated
---
r37 | soga1113 | 2011-12-19 04:50:02 -0400 (Mon, 19 Dec 2011) | 1 line

global var
---
r36 | soga1113 | 2011-12-19 02:10:10 -0400 (Mon, 19 Dec 2011) | 1 line

interpret
---
r35 | cck73823@gmail.com | 2011-12-19 00:40:23 -0400 (Mon, 19 Dec 2011) | 1 line

eat dots./compile_and_run.sh
---
r34 | cck73823@gmail.com | 2011-12-19 00:36:06 -0400 (Mon, 19 Dec 2011) | 1 line

modify myclasses add setGifts(LIST)
---
r33 | cck73823@gmail.com | 2011-12-18 23:37:18 -0400 (Sun, 18 Dec 2011) | 1 line

test_run
---
r32 | cck73823@gmail.com | 2011-12-18 20:30:30 -0400 (Sun, 18 Dec 2011) | 1 line

modify myclasses
---
r31 | cck73823@gmail.com | 2011-12-18 20:30:25 -0400 (Sun, 18 Dec 2011) | 1 line

using mainMap with seprate lists
---
r30 | cck73823@gmail.com | 2011-12-18 20:19:56 -0400 (Sun, 18 Dec 2011) | 1 line

using mainMap with seprate lists
---
r29 | cck73823@gmail.com | 2011-12-18 20:15:32 -0400 (Sun, 18 Dec 2011) | 1 line

modify myclasses
---
r28 | cck73823@gmail.com | 2011-12-18 20:10:12 -0400 (Sun, 18 Dec 2011) | 1 line

using mainMap with seprate lists
---
r27 | cck73823@gmail.com | 2011-12-18 20:10:02 -0400 (Sun, 18 Dec 2011) | 1 line

modify myclasses

---
r26 | cck73823@gmail.com | 2011-12-18 20:06:15 -0400 (Sun, 18 Dec 2011) | 1 line

modify myclasses
---
r25 | cck73823@gmail.com | 2011-12-18 19:58:51 -0400 (Sun, 18 Dec 2011) | 1 line

using classes
---
r24 | cck73823@gmail.com | 2011-12-18 19:24:05 -0400 (Sun, 18 Dec 2011) | 1 line

modify myclasses
---
r23 | cck73823@gmail.com | 2011-12-18 18:07:11 -0400 (Sun, 18 Dec 2011) | 1 line

modify myclasses
---
r22 | cck73823@gmail.com | 2011-12-18 15:45:30 -0400 (Sun, 18 Dec 2011) | 1 line

use monster class
---
r21 | cck73823@gmail.com | 2011-12-17 20:00:55 -0400 (Sat, 17 Dec 2011) | 1 line

modify myclasses
---
r20 | cck73823@gmail.com | 2011-12-17 19:56:43 -0400 (Sat, 17 Dec 2011) | 1 line

modify myclasses
---
r19 | cck73823@gmail.com | 2011-12-17 19:56:18 -0400 (Sat, 17 Dec 2011) | 1 line

modify myclasses
---
r18 | cck73823@gmail.com | 2011-12-17 19:50:26 -0400 (Sat, 17 Dec 2011) | 1 line

modify myclasses
---
r17 | cck73823@gmail.com | 2011-12-17 18:30:33 -0400 (Sat, 17 Dec 2011) | 1 line

change map from list to array
---
r16 | Vicky.Wenxin.Zhu@gmail.com | 2011-12-06 12:29:29 -0400 (Tue, 06 Dec 2011) | 7 lines

A lang/compile.ml

D lang/log
A lang/bytecode.ml
A lang/execute.ml
D lang/main.ml

---
r15 | cck73823@gmail.com | 2011-12-06 12:21:57 -0400 (Tue, 06 Dec 2011) | 1 line

add random monster
---
r14 | Vicky.Wenxin.Zhu@gmail.com | 2011-12-02 19:02:52 -0400 (Fri, 02 Dec 2011) | 6 lines

the following three files work, meaning, parsing is basically done.

lang/ast.mli
lang/parser.mly
lang/scanner.mll

---
r13 | Vicky.Wenxin.Zhu@gmail.com | 2011-11-30 11:25:28 -0400 (Wed, 30 Nov 2011) | 1 line

modified parser
---
r12 | soga1113 | 2011-11-30 03:04:55 -0400 (Wed, 30 Nov 2011) | 3 lines

classes needed for Pacaml

---
r11 | cck73823@gmail.com | 2011-11-22 12:17:11 -0400 (Tue, 22 Nov 2011) | 1 line

[*what] run run pacman
---
r10 | cck73823@gmail.com | 2011-11-22 10:43:44 -0400 (Tue, 22 Nov 2011) | 1 line

[*what] it runs better... but....
---
r9 | cck73823@gmail.com | 2011-11-22 10:19:06 -0400 (Tue, 22 Nov 2011) | 1 line

[*what] make the picman can run.... but....
---
r8 | cck73823@gmail.com | 2011-11-15 11:34:49 -0400 (Tue, 15 Nov 2011) | 1 line

remove some useless code of show/test.ml

---
r7 | cck73823@gmail.com | 2011-11-15 10:40:37 -0400 (Tue, 15 Nov 2011) | 2 lines

test

---
r6 | cck73823@gmail.com | 2011-11-15 10:37:46 -0400 (Tue, 15 Nov 2011) | 1 line

show test... please install the library: liblablgl-ocaml-dev
---
r5 | Vicky.Wenxin.Zhu@gmail.com | 2011-11-09 01:54:41 -0400 (Wed, 09 Nov 2011) | 1 line

add class: point
---
r4 | Vicky.Wenxin.Zhu@gmail.com | 2011-11-08 22:31:28 -0400 (Tue, 08 Nov 2011) | 1 line

print function accepts any string
---
r3 | Vicky.Wenxin.Zhu@gmail.com | 2011-11-08 22:15:45 -0400 (Tue, 08 Nov 2011) | 1 line

Hello World
---
r2 | Vicky.Wenxin.Zhu@gmail.com | 2011-11-08 22:12:44 -0400 (Tue, 08 Nov 2011) | 1 line

mkdir lang
---
r1 | (no author) | 2011-10-11 10:32:27 -0400 (Tue, 11 Oct 2011) | 1 line

Initial directory structure.
