JAWS *Language*

Justin Lu | jl1439@columbia.edu [leader]
Andy Guo Xin | gl350@columbia.edu
Winston Chao | wc386@columbia.edu
Shoaib Anwar | sa597@columbia.edu

1.1 Introduction

Anyone who grew up in the mid-eighties to early-nineties is probably familiar with the genre of video games known as space shooters. While playing titles such as Galaxian and Galaga, most of us could not help but wonder how cool it would be if we were able to create our own space shooter type game without having to know complex graphics programming.

The JAWS language is the culmination of that dream, a simple programming language designed to allow users to easily create a vast array of space shooter type games. One of the main goals of JAWS was to allow for the creation of space shooter games without needing knowledge of graphics programming. The purpose of JAWS is to free game designers from that burden so that they can focus on the design of gameplay and depth in space shooter games.

1.2 JAWS - Main Features

Simplicity

The JAWS language is designed so that casual programmers or non-programmers with a good grasp of logic can be able to make space shooters. If the language wasn't designed for ease of use, then it would be obsolete since a space shooter could just be programmed from the ground up. Why JAWS is useful for the casual programmer is because it uses modern, easy to understand syntax. The structure of the language is designed to mimic most languages of today such as JAVA and C++. This will eliminate the need to learn a whole new set of rules for the semantics of the language.

Ease of Use

JAWS totally eliminates the need of any knowledge of graphics programming by incorporating most of the techniques used right into the language's design. The one biggest hump to any programmer when wanting to program a game is having to program the graphics. JAWS allows its user to bypass this by having most of the necessary tools in space shooter graphics, logic, and physics built-in as predefined functions.

For one, collision detection will be integrated into JAWS' object oriented nature. Objects such as ships, enemies, will contain flags or methods which allow to the user to access distance from each other or whether two objects are colliding. The collision detection only serves to inform the user how far two objects are from each other or whether they are colliding, it is up to the user to take that information and decide how they want their game to react to it. JAWS was designed this way so as to not make the gameplay too preset.

Game physics will also be incorporated as part of the language so the user does not have to deal with that aspect in creating a space shooter. Movement patterns of objects on the screen can be specified in the syntax. For example, enemies can be designated to move in circles, zig-zags, etc.

Speed and acceleration of enemies or the player can also be easily assigned. However, JAWS will also allow for the advanced user to specify custom game physics if he or she so wishes.

Object-Oriented

JAWS is designed as an object-oriented language to further extend the idea of simplicity of the language. Object orientation facilitates the re-use of code which can be extremely desirable in creating a space shooter game. For example, entire levels can be generalized by specifying general attributes such as number of enemies, locations, and etc. but can be re-used to make multiple levels by changing the individual AI or difficulty. Each item in JAWS will be designed to incorporate object-orientation.

Portability

Since JAWS will be implemented using JAVA, it automatically inherits the portability of JAVA. Space shooters will be able to be generated on any platform and played on any other platform. This helps users who wish for ease of distribution of their space shooter games.

Customizable

Even though JAWS provides default graphics, AI, movement patterns to speed up the process of putting together a space shooter, the language gives the user the flexibility to fully customize almost every feature. The player's ship, enemies, and surroundings can be changed to images that the user can specify. Furthermore, predefined AI for enemies can be reprogrammed by the user if the user is interested in making AI especially easy or hard. With the power of a customizable language, the space shooters that are created don't need to all look the same.

Built-in Sound Support

A very important feature of any game is sound. Sound effects and background music help to enhance the atmosphere of a game. JAWS will have built-in support to allow users to specify .midi background music, as well as have default sound effects for collisions, movement, and etc. Of course, the sound effects will be easily customized if the user so desires.

1.3 Syntax Sample:

Declaring an enemy object:

```
Enemy(name:badguy type:1 hp:100 movement:circle);
```

Detecting Collisions:

```
//where badguy is the name of an enemy
//player is the name of the player's ship
//~ is the distance operator


if (badguy~player == 0)
{
   label1.display("YOU DIED");
   player.lives -= 1;
}
```

CHAPTER 2 - Tutorial

JAWS is a programming language with the intention of facilitating the creations of vast array of space shooter type games without the programmer having to deal with the low level details of graphics programming.

The JAWS language and its syntax are very similar to that of Java, experienced java programmers should find it rather easy to program in JAWS.

2.1 Hello World Example:

```
// This program just prints on screen "helloWorld"
JAWS: helloworld;

// the declaration block
declarations
{
        // define the map
        map myWorld;

        // define the entity
        entity myLabel[1];
        myLable(){
                label;
        }

        // define string
        String myString = "Hello World!";

        // make the entity's label refer to myString
        myLabel[1].label = myString;

}

engine
{
        main(){
                // no need for conditionals, just prints out the label of text: "hello world!"
        }

}
```

2.2 Another Example:

// The above code will create a player and a monster (enemy ship) in which the player
// will have keyboard user interface and monster with a default falling straight down /
// towards the bottom of the screen

```
JAWS: steviebattle;

declarations
{
	map a;

	entity player[1];
	player()
	{
		_width = 40;
		_height = 40;
		 _x = 0;
		_y = 400;
		_image = "player.gif";
		_ai = "keyboard";
	}

	entity monster[1];
	monster()
	{
		 _width = 40;
		_height = 40;
		_x = 0;
		_y = 0;
		_image = "stevie.gif";
		//ai = "keyboard";
	}

	movement monster(){
		y -= 3;
	}


}

engine
{
	// nothing is needed
}
```

2.4 Yet Another example

// This program show 5 enemy ship with default AI moving downwards and the player with ability to shoot or collide with the enemy ships.

```
JAWS: metroids;

declarations
{
	map a;

	entity player[1];
	player()
	{
		_width = 40;
		_height = 40;
		_x = 0;
		_y = 400;
		_image = "player.gif";
		_ai = "keyboard";
	}

	entity monster[1];
	monster()
	{
		_width = 40;
		_height = 40;
		_x = 0;
		_y = 0;
		_image = "enemy. gif";
		//ai = "keyboard";
	}

	movement monster(){
		y -= 3;
	}

	entity monster2[1];
	monster()
	{
		_width = 40;
		_height = 40;
		_x = 75;
		_y = 0;
		_image = "enemy.gif";
		//ai = "keyboard";
```

```
	}

	movement monster(){
		y -= 3;
	}

	entity monster3[1];
	monster()
	{
		_width = 40;
		_height = 40;
		_x = 150;
		_y = 0;
		_image = "enemy.gif";
	}

	movement monster(){
		y -= 3;
	}

	entity monster4[1];
	monster()
	{
		_width = 40;
		_height = 40;
		_x = 225;
		_y = 0;
		_image = "enemy.gif";
		//ai = "keyboard";
	}
	movement monster(){
		y -= 3;
	}

	entity label1[1];
	label1()
	{
		_width = 100;
		_height = 50;
		_x = 200;
		_y = 400;
		_label = "DONT CRASH!";
	}

}
```

```
engine
{
	if player[0] ~ monster[0] == 0
	{
		label1[0]'s  label = "LOSER!";
	}

	if player[0] ~ monster2[0] == 0
	{
		label1[0]'s  label = "LOSER!";
	}

	if player[0] ~ monster3[0] == 0
	{
		label1[0]'s  label = "LOSER!";
	}

	if player[0] ~ monster4[0] == 0
	{
		label1[0]'s  label = "LOSER!";
	}

}
```

2.3 Program Structure

JAWS source file are divided into two parts, the declaration block and the engine block.

All variable, constants must be declared in the declaration block.

```
declarations
{
	// all declaration statements goes into here
}
```

The engine block is used for programmatically specifying the run time rules of the game.

```
engine
{
	main()
	{
		// this block repeats during run time to check for game conditions.
		// this is where winning conditions are written if any at all
	}
}
```

JAWS *Language Reference Manual*

1. Introduction

The JAWS programming language was designed to provide a platform for casual and experienced game designers to rapidly prototype and develop space shooter genre video games. The main focus behind the language was to eliminate the necessity for mastery of complex graphics and game physics implementation to allow the game designer to solely focus on game play design. Thus, the syntax and structure of the JAWS language strives for simplicity and ease of use.

2. Lexical Conventions

2.1 Comments

Comments are segments of code that the programmer wants the compiler to ignore. Comments in the JAWS programming language are limited to one per line. Comments must begin with "//" and everything else on that line up to the carriage return "/n" is treated as a comment.

2.2 Whitespace

In addition to comments, blanks, tabs, and carriage returns are ignored except where they are used to separate tokens. Tokens must be separated by at least one of the above.

2.3 Identifiers

Identifiers consist of a sequence of letters and digits and must start with a letter. In addition to letters, the underscore "_" can also be used and counts as an alphabetic. Identifiers represent the names that are given to declared data types that are used. It is suggested that identifiers begin with a lowercase letter. However, this is not enforced syntax.

2.4 Keywords

The following list of identifiers are reserved in the JAWS language and may not be used otherwise.

int	entity
string	movement
boolean	attack
double	point
constant	map
if	show
else	hide
for	true
to	false
by	
while	

2.5 Constants

JAWS allows the use of 3 types of constants: int, string, and double. Constants are defined by typing the "constant" prefix before an identifier. For example, to define a constant called "HIGH_SCORE":

```
constant string HIGH_SCORE = 100;
```

Constants retain their initial values and may not be changed during the execution of the program. It is suggested that constants be named with all uppercase letters to distinguish them from identifiers. However, this is not enforced syntax.

2.6 Separators
The characters used as separators in JAWS are:
( ) { } , ;

2.7 Tokens
Tokens are separated into identifiers, constants, keywords, operators, and other separators. Tokens are defined greedily meaning the token is the longest un-terminated recognized token type. All tokens must be separated by whitespace.

3. Data Types
There are nine types of data in JAWS: int, double, string, boolean, point, entity, movement, attack, and map.

3.1.1 int
Ints will be 32-bit two's compliant integers.

3.1.2 double
Doubles will be 64-bit double precision floating point numbers.

3.1.3 string
Strings will be a character string beginning with a " and ending with a ". JAWS will not support the single char data type so for single characters strings can be used in the same way.

3.1.4 boolean
Booleans will be a true or false value.

3.1.5 point
Since locations and points are so prevalent in JAWS, the point type is standard. Points consist of two ints, an x and a y, which uniquely define the point. The distance operator may be applied to point types, facilitating the calculation of distance between points and entities. Point type attributes may be directly accessed using the 's operator.

Points are declared like a normal type and its x and y are defaulted to 0. Point has a built-in initializer that takes in an int x and an int y which may be called to initialized x and y. Here is an example:

```
// point p will be at x = 0, y = 0
point p;

// point p will be at x = 6, y = 4
point p = point(6,4);
```

Point Type Attributes:

Type	Name	Description
int	x	x-coordinate of the point
int	y	y-coordinate of the point

3.1.6 entity

Entities are objects that interact with each other during game play that are displayed to the user. Each entity has a set of attributes that describe fully the entity's behavior and look. Entities can be players, monsters, powerups, labels, and anything else that could appear on the map. It is up to the programmer to correctly define their behavior.

Entity types in JAWS are all implemented as arrays. This facilitates the easy duplication of a multiplicity of entities without repeated declaration and initialization. When an entity class is created, all entities in its array are created with the same values. Each entity in the object array may be referenced and accessed individually thereafter. When an entity is referenced with an index specified, only that specific entity in the array is changed. When an index is left out, all entities in the array are affected.

Entity classes have special built-in fucntions which facilitate game programming. The ~ (distance) operator can be applied on two entity classes and returns the distance between those two entities. Entities also have show and hide functions. Show is called to display the entity and to load its initializer. Hide is called to unload the entity from display.

All attributes in a created entity class are set to default but can be initialized to different values in its initializer. To define the entity's initializer, define a function with the name of the entity without a return type.

The syntax for declaring an entity class and its initializer is as follows:

```
// entity declaration
entity name[multiplicity];

// entity initializer
name()
{
      [attribute] = newvalue;          // initializing attributes
      .
      .
      [type] name = value;       // adding user-defined variables
      .
      .
}
```

An example of creating a monster type called small_alien of multiplicity 5, and setting its constructor to lay them across horizontally, and creating user defined variables called hitpoints and pointValue:

```
entity small_alien[5];

small_alien()
{
      // individually set each small_alien's location
      for i = 0 to 5 by 1
      {
            small_alien[i]'s location = point(i*2, 0);
      }

      // set all small_alien's hitpoints to 10
      int hitpoints = 10;
```

```
      int pointValue = 100;
      show();
}
```

Entity Type Attributes:

Type	Name	Description
int	height	the height (in cells) of the entity on the map
int	width	the width (in cells) of the entity on the map
point	location	the location of the entity
string	image	address of a supported image file to be used
attack	weapon	the attack class that the entity is equipped with
movement	ai	the movement class to used to describe this entity
string	label	a string to display using the entity as a label

Entity Type Functions:

Name	Description
~	calculates the distance between two entities
show()	displays the entity on screen and executes the entity's initializer, the entity's movement ai also starts execution
hide()	unloads the entity from the screen and all execution associated to that entity is halted

3.1.7 movement

The movement type basically describes how an entity moves about the screen and how it determines its moves either randomly or based upon certain conditions. Movement types are defined individually and may take in parameters. Within the movement type is the description of the movement.

Movement types exist as attributes in entities. The location of the entity can be referred to by the movement type directly. Any other locations that the movement type wants to interact with must be passed in as parameters.

The movement type is a loop which keeps running for the specified entity. That means that defined movements are always looped and executed until that entity is unloaded with the hide command.

The syntax for declaring a movement class and its body is as follows:

```
movement name(type1 param1, ... , typeN paramN)
{
      // movement body
}
```

An example of a movement called retreat which makes this entity move north when the passed in entity is within 10 units of it. Otherwise, the entity keeps moving south.

```
movement charge(point oppLocation)
{
      if ( (location ~ oppLocation) < 10)
      {
          location's incy(1);
      }
      else
```

```
    {
        location's decy(1);
    }

}
```

JAWS implements a built-in movement type called keyboard, which moves an entity based on keyboard input. It will be of type movement and be called keyboard.

3.1.8 attack

Attack types represent an entity's offensive arsenal. It describes both how that entity's attack looks like, and also when and how it attacks.

Attack Type Attributes:

Type	Name	Description
string	image	the location of the image to be used
int	width	width in cells of the attack
int	height	height in cells of the attack

An attack type also contains three functions, two of which must be defined by the user, the shooting() and collision() functions. The last is the initializer which can be left blank. Shooting() describes the rate, speed, and direction of the attack. Collision() defines what action to take when this attack collides with the passed in entity.

```
attack name
{
    // initializer
    name() {
    [attribute] = newvalue;
    .
    .
    .
    }
    shooting() {

    }
    collision(entity source) {

    }
}
```

3.1.9 map

Map types represent individual levels. Each map can be specified with a width and height, and also a background image.

Map Type Attributes:

Type	Name	Description
int	width	width of the map
int	height	height of the map
string	background	image to be used for the map

Map Type Functions:

Name	Description
show	loads the map onto the display screen
hide	unloads the map from the display screen

Map types also have initializers with the same name. Map types are declared and initialized as follows:

```
map level1;

// initializer
level1()
{
      width = num;
      height = num;
      background = "image1.jpg";
}
```

3.2 Type Casting and Interaction

3.2.1 int and double

JAWS automatically type casts between int and double in assignments. An int may be defined using a double and vice versa. When declaring an int with a double, the result is always rounded down, meaning the precision is thrown away. For example, int a = 3.9 will leave a with the value of 3.

Int and double interaction during other operators is described below in the operators section.

3.2.2 string, int, and double

JAWS automatically type casts between strings and ints and doubles in assignments. Strings declared with an int or double will be accepted and converted to the corresponding character string. Int and doubles that are assigned strings will also automatically convert given that the string is in valid int or double form.

```
int i = "4";          // valid
double d = "5.2";     // valid
string s = 3;         // valid
string s = 3.2;       // valid

int i = "4.0";        // invalid
double d = "5.a";     // invalid
```

String interaction with other operators is described below in the operators section.

3.2.3 point, entity, movement, ai, map

Point, entity, movement, ai, and map types may not be used interchangeably and cannot be cast from one type to another. They all represent distinct data and no similarities exist enough between them to warrant type casting from one type to another. These types may be passed into functions where their individual attributes may be modified.

Interaction of these types with other operators is described below in the operators section.

4. Expressions

4.1 Expressions
General expressions refer to a combination of identifiers, keywords and operators that usually equate to a data type.

4.1.1 Mathematical Expressions
Mathematical expressions are combinations of identifiers and operators that take on the value of either an int or a double after evaluation. For example:

```
4.3         // valid mathematical expression
5 + 7 * 6   // valid mathematical expression

int a = 5; // invalid mathematical expression
```

4.1.2 Boolean Expressions
Boolean expressions are combinations of identifiers and operators that evaluate to a true or false value.

4.2 Statements
Statements are a combination of primary expressions, operators, and keywords usually ending in a semicolon. Statements usually constitute a line of code in JAWS.

5. Operators
The supported operators in jaws are assignment, additive, multiplicative, boolean, relational, equality, concatenation, distance, and class accessor.

5.1.1 Assignment Operators
Assignment operators assign an identifier to a new value. Some automatic type casting occurs between int, double, and strings. No type casting occurs for the other data types. Type casting is described in depth in the previous section.

The supported assignment operators are:
```
=      *=
+=     /=
-=
```

5.1.2 Additive/Multiplicative Operators
Additive and multiplicative operators may only be applied to int, double, and mathematical expressions. The + operator when applied to strings is a concatenation operator, not additive or multiplicative. If the operators are applied to all ints, the result will be an int. If the operators are applied to all doubles, the result will be a double. If the operators are applied to both ints and doubles, the result will be a double. For example:

The supported additive and multiplicative operators are:
```
+      *
-      /
%
```

5.1.3 Relational Operators

Relational operators may only be applied to int, double, and mathematical expressions. When relational operators are applied to both double and int, the comparisons will be done using the double value of the int.

The supported relational operators are:

```
>       >=
<       <=
```

5.1.4 Boolean/Negation Operators

Boolean and negation operators may only be applied to booleans and boolean expressions. Boolean operators return a boolean.

The supported relational operators are:

```
&&      !
||
```

5.1.5 Equality Operators

Equality operators may only be applied to data of the same types. Equality operators return a boolean. Equality applied to int, double, string, boolean, and point will be evaluated according to data. Equality in entity, movement, ai, attack, and map types are not evaluated based on their attributes but according to address.

The supported equality operators are:

```
==
!=
```

5.1.6 String Concatenation Operator

The string concatenation operator + can only be applied to strings. No automatic casting is done when using this operator.

```
"abc" + "def"          // valid
"abc" + 5              // invalid
```

5.1.7 Entity Distance Operator

The entity distance operator ~ can only be applied to the point and entity data types. The return value is a double with the absolute distance between the two arguments.

5.1.8 Type Accessor

The 's type accessor is used to access attributes and functions within the point, entity, movement, attack and map classes and thus can only be applied to those types.

```
class's attribute
class's function( )
```

5.2 Operator Precedence

The order in which operators are applied is described below, with operators at the top evaluated first, and operators on the same line evaluated according to their left to right order in the expression. Note that parentheses ( ) may be applied to give precedence to their contents in expressions. Precedence is listed from highest priority to lowest priority

Priority	Operator	Function
1	`'s`	type accessor
	`()`	function call
	`[]`	entity array index
2	`!`	negation
3	`~`	distance
4	`* / %`	multiplication, division, modulus
5	`+ -`	addition, subtraction
	`+`	string concatenation
6	`> >=`	greater, greater/equal
	`< <=`	less, less/equal
7	`==`	equal
	`!=`	not equal
8	`&&`	logical and
9	`||`	logical or
10	`=`	assignment
	`+= -=`	
	`*= /=`	

6. Declarations

6.1 User defined variables

Primitive data types int, double, string, boolean are declared…

Without initialization:

```
type name;
```

With initialization:

```
type name = initial-value;
```

Uninitialized int default to 0, double to 0.0, stings to “”, boolean to true.

6.2 User defined functions

User defined functions are declared as follows:

```
return-type function-name(type1 param1, … , typeN paramN)
{
      //function body
      return retValue;
}
```

The function must include a return statement returning an expression of the specified return type.

7. Conditionals and Loops

7.1 Coniditonal If…Else

Conditional if statements must have their then and else blocks separated by brackets { }. They should follow the form:

```
if boolean-expression
{
      //code block
}
else
{
      //code block
}
```

7.2 While Loops
While loops check the specified condition before the execution of the block and keep repeating as long as the condition is true. While loops follow the form:

```
while boolean-expression
{
      //code block
}
```

7.3 For Loops
For loops take an int, an end int, and an optional increment

```
for int to endInt by increment
{
      //code block
}
```

8. Coordinate System

The coordinate system in JAWS will center the point with x = 0 and y = 0 to the bottom left of the screen and increasing towards the top and right. This will model normal geometric coordinate systems and provide ease of use for the programmer.

9. Program Structure

The structure of a JAWS program consists of two main parts, the declarations and the engine. All programs in JAWS must be written in this context and all code should fall under either the declarations or engine sub parts.

```
// My JAWS Program
declarations
{
      // declarations
}

engine
{
      // engine code
```

```
}
```

9.1 Declarations
All constants, variables, entity, movement, attack, point, map types must be declared in the declarations segment of the program. The order in which they are declared is ignored so they can be declared in any order.

9.2 Engine Code
The engine segment contains all code relating to how the game is run during play. It is subdivided into two parts, the main loop and user-defined functions. The main loop is the game loop which keeps executing during gameplay. Here is how it is broken down:

```
engine
{
      main()
      {
            //main game play code
      }

      // user-defined functions
      // .
      // .
      // .
      // etc
}
```

9.1 A Sample JAWS Program
Here we try to illustrate the basic look and feel of a JAWS program with a sample of a complete and functional JAWS program. This program will simulate a game which

```
// Sample JAWS Program
// By Justin Lu
// "Attack of the Aliens" v1.0

declarations
{
      // lets create the map for our game
      map mainLevel;

      mainLevel()
      {
            // it will be a 50 x 50 map and
            //use a background called "bg.jpg"
            width = 50;
            height = 50;
            background = "bg.jpg";

      }

      // lets create an entity for our controllable player
```

```
entity player[1];
player()
{
      // our player will be 2x2 cells in size
      height = 2;
      width = 2;

      // lets make him start at the lower center of our map
      location = point(width/2 , 0);

      // we will equip him with a basic gun attack class,
      // to be defined later
      weapon = gun;

      // we will use a built-in default movement
      // called keyboard to allow input
      ai = keyboard;

}

// now lets create some enemy aliens, lets make 5
entity alien[5];

alien()
{
      // they will be 2x2 in size
      width = 2;
      height = 2;

      // lets lay them out across the top of the map
      for int i = 0 to 5 by 1
      {
            alien[i]'s location = make point( mainLevel's width/5 *
            i , mainLevel's height);
      }

      // we will set their AI to dumb_ai,
      // a movement class we will define
      ai = dumb_ai;
}

// we will now create the attack class gun that the player uses
attack gun
{
      gun()
      {
            image = "cool_laser.jpg";
            width = 1;
            height = 1;
      }
      shooting()
      {
```

```
                // we will use a builtin function
                // which uses keyboard input
                // when enter is pressed, our attack will go upwards
                if keyboardInput() == "/n"
                {
                      y++;
                }
          }
          collision(entity source)
          {
                // when our attack collides
                // with enemies of type alien,
                // lets unload them
                if source == alien
                {
                      source's hide();
                }
          }
     }

     // lets set our enemy AI now
     // it will be a dumb AI which just moves the enemy downwards
     movement dumb_ai( )
     {
          location's y -= 1;
     }
}

engine
{
     main()
     {
     // our game will have no engine code, lets just let it run
     // simple enough game, that all actions can be defined in
declarations
     // if we wanted to extend the game play,
     // we would need to add engine code
     }
}
```

CHAPTER 4 – Project Planning

4.1 Project Planning

To avoid major fall backs, our group decided to plan ahead and set up incremental milestones. Throughout the semester we had weekly (Mondays) status meeting to update on our individual responsibilities. We also planned biweekly meetings with our supervising TA who gave us invaluable direction in the early stages of the project.

4.2 Team Responsibilities

Taking from our past cs group project experiences and our intro to economics classes, we realized that certain part of the project is better done by an individual and other parts done jointly by members, and so we setup an initial plan for the division of labor:

Shoaib Anwar	Backend Design and Implementation Library classes Design and Implementation Presentation Powerpoint Creation
Winston Chao	Grammar Design Lexer/Parser Testing: Test suite design
Andy Lin	Backend Implementation Library classes Design and Implementation Documentation
Justin Lu	Grammar Design Lexer/Parser AST Parser / Code Generation

Note: All members were involved in incremental component testing as well as project wide integration testing.

4.3 Programming Style Guideline

All members must adhere to the following coding guideline during all phases of the project. Team members should try to follow the guidelines we strictly as possibly since it will benefit readability of the code for third party reader. Code should be written in a clear layout that will be easy to read. Avoid writing blocks all in one line. Comments are required for logical decision and initial declaration of logic variables.

4.3.1 Code Spacing

Line spacing should be uniform. Do no add unnecessary line spaces as it creates confusion for the reader. All horizontal spacing for nesting loops and variables requires using tabs.

4.4 Software Development Environment

The project was initially developed on clic lab computers using Java 1.4 with emacs and Antlr 2.7.x on Redhat Linux, we later moved our meetings to a lab full of windows xp and decided to use windows based text editors/IDE that each one of use were familiar with: TextPad, EditPlus, Netbeans. We continued to use Java 1.4 and Antlr 2.7.x.

4.4.1 Front End:

- ANTLR for Lexer, Parser, Walker for code generation and semantic checking.

4.4.2 Back End:

- EditPlus and Netbeans for library classes implemented in java

4.4.3 Test suite:

- java used to batch process test inputs
- Editplus to write unique test cases
- Shell to run testing

4.5 Project Log

4.5.1 Initial Projected Timeline:

9.11.2003	Come up with the basic ideas of our Language
9.20.2003	Finalize out Language ideas and White Paper
9.23.2003	Submit White Paper and start LRM
10.16.2003	Begin designing test suite and sample code
10.23.2003	Group status update on: lexer/parser, LRM and test suite design
10.27.2003	Finalize LRM and submit by midnight
11.4.2003	Parser Complete
11.11.2003	Code Generation Complete
11.18.2003	Static Semantics Complete
12.2.2003	Testing Complete
12.8.2003	Project Complete

4.5.2 Actual Timeline (actual progress log):

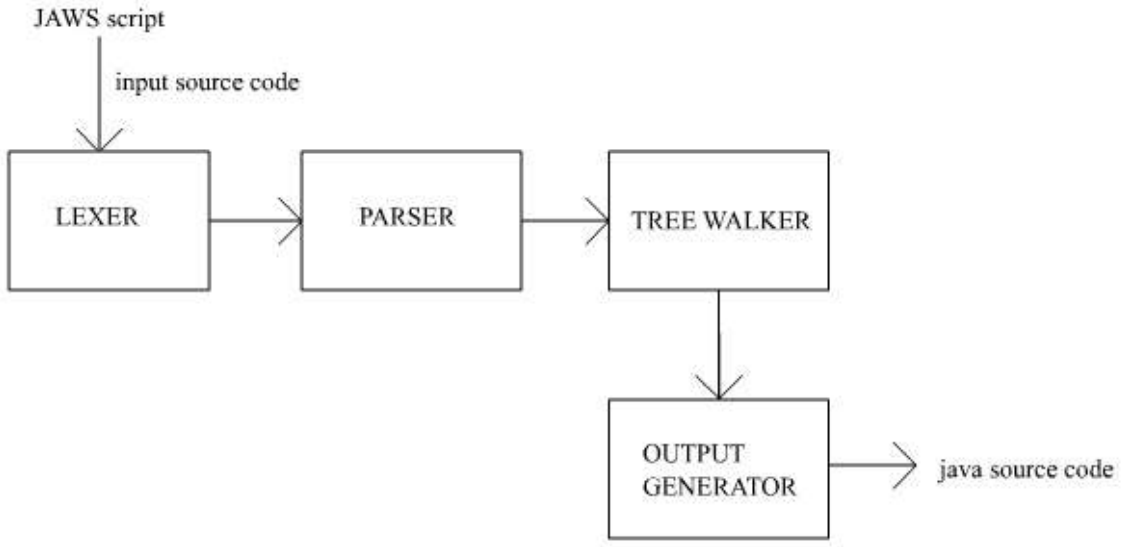
9.14.2003	Come up with the basic ideas of our Language
9.22.2003	Finalized out Language ideas and White Paper
9.23.2003	Submitted White Paper
10.2.2003	Started working on LRM
10.17.2003	LRM revision meeting
10.25.2003	Group met to update on: lexer/parser, LRM
10.27.2003	Finalized LRM and submitted
11.15.2003	Update meeting on Lexer/Parser
11.29.2003	Lexer/Parser and Code Generation Testing
12.2.2003	Begin implementation of backend/library classes
12.5.2003	Simple testing cases created
12.15.2003	Integration of components
12.16.2003	Testing parser complete
12.17.2003	Project Complete

CHAPTER 5 - Architectural Design

This section will give a brief overview of the design of the JAWS language and the required third party resources.

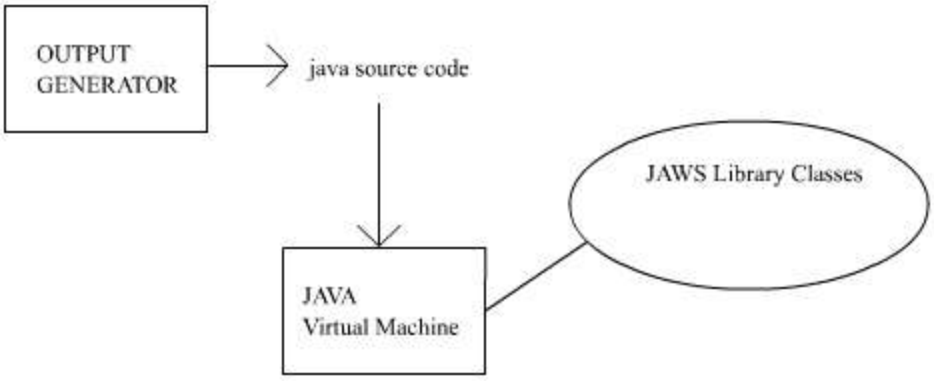
5.1 Block Diagram

The pipeline through which JAWS source code is convert into compilable and runnable java source code is via the block diagram below: the lexer, the parser, the output generator. The diagram shows the relationship and the sequence of processing between the components.



The JAWS source code is feed into the lexer, which produces a collection of tokens according to the JAWS language reference manual in Chapter 3. The collection of tokens is then feed into the parser for syntax checking. Upon successful pass of the parser, an abstract syntax tree of the tokens is feed into the tree walker which in turn stores the data in the output generator which then produces the java source code.

5.2 Runtime Environment:



Once the java source code is produced, the file is executed in java virtual machine which makes use of the JAWS library classes.

CHAPTER 6 - Testing Plan

6.1 Testing overview

The goal of software test is to find faults made by the program during its execution. In our case, we are also interested in testing the output different stages of our project, lexer, parser, walker and the java code generated by code generator. Although it is impossible to test for all array of input into our components, we emphasized exposing bugs in early stage of our development via the use of incremental test, and special case tests.

6.2 Antlr Testing

To test our lexer and parser for syntax parsing functionalities, we created batch JAWS created test programs that encompass all predefined pre-defined syntax of our language in our Language Reference Manual as well as user defined function/structures. In addition we intentionally created obvious bugs in order to make sure the parser is functioning as expected.

6.3 Java Testing

Due to lack of time we only implemented sample code testing. Basically we ran all our sample codes to test the output of the code generator by compiling, running the generated code. Functionalities were observed graphically.

CHAPTER 7 – Lessons Learned

7.1 Individual Responses

7.1.1 Justin's Lesson

I learned a great deal about programming language structure and the difficulties of specifying a complete and nonambiguous grammar during the course of this project. Although at first it seemed like a daunting task, after designing and creating an actual language, the feeling was quite great. First and foremost, I learned that creating a language is no easy task. Working simultaneously on the parser and walker greatly eased the process since changes to the language could be made while parsing and walking. It was definitely an interesting experience to make JAWS.

7.1.2 Andy's Lesson

The main lesson learned from the project was we should have learned to manage our time more efficiently and should really start as early as possible. It's always better to have more free time in the end for adding possible features instead of testing and debugging. I also learn about the basic steps required to make a translator.

7.1.3 Winston's Lesson

I think the biggest lesson for this project is to really get organized and have certain goals set in mind at a relatively early point in the development of your language. It definitely helps to have something to aim for as you progress through the coding. In addition, it really pays to have some sort of realization of the structure before you get far along in your code.

7.1.4 Shoaib's Lesson

It was very difficult to evenly distribute the work among the members. We sould have had more clearly defined roles

7.2 Advice for future groups

We recommend that groups should really start their project early and should adhere to the goals and treat the deadlines like scripts from the bible. That way you will be glad a few days before the project is due and will save you much frustration.

Appendix – Code Listing

Grammar File
jaws.g

antlr Generated Files
JAWSParser
JAWSTokenTypes
JAWSWalker
JAWSWalkerTokenTypes
JAWSLexer

Backend Classes
BulletController
Entity
JAWSmovement
JAWSpoint
KeyReleasedListener
Map

JAWS Translation/Compiler
JAWSOutputter
JAWSBoolean
JAWSDataType
JAWSDouble
JAWSEntity
JAWSInt
JAWSString

JAWS Executable
Jawsc

//--

```
// The JAWS scanner
// justin
//--------------------------------------------------------------------
class JAWSLexer extends Lexer;

options {
  charVocabulary = '\3'..'\377';
  exportVocab = JAWS;
  testLiterals=false;
  k=2;
}

{
        int nr_error = 0;
        public void reportError( String s ){
                super.reportError( s );
                nr_error++;
        }
        public void reportError( RecognitionException e ){
                super.reportError( e );
                nr_error++;
        }
}

// Single-line comments
COMMENT
  :  "//" (~('\n'|'\r'))*
     { $setType(Token.SKIP); }
  ;

// Literals
protected DIGIT : '0'..'9';
protected ALPHA : 'a'..'z' | 'A'..'Z' | '_';


NUMBER : (DIGIT)+ (DOT (DIGIT)+)?  ;

// string literals
STRING
  : ""!
    ( "" ""!
    | ~(""|'\n'|'\r')
    )*
    ( ""!
    | // nothing -- write error message
    )
  ;


// Keywords are literals in the parser grammar

// Operators
DOT      : '.'  ;
COLON            : ':'  ;
SEMI             : ';'  ;
COMMA            : ','  ;
```

```
EQUALS          : '='  ;
LBRACE                  : '{'  ;
RBRACE          : '}'  ;
LBRACKET        : '['  ;
RBRACKET        : ']'  ;
LPAREN          : '('  ;
RPAREN          : ')'  ;
NOT_EQUALS      : "!="  ;
LT      : '<'  ;
LTE     : "<=" ;
GT      : '>'  ;
GTE             : ">="  ;
PLUS            : '+'  ;
MINUS           : '-'  ;
MULT            : '*'  ;
DIV     : '/'  ;
DISTANCE        : '~'  ;
MOD             : '%'      ;
PLUS_EQUALS : "+=" ;
MINUS_EQUALS            : "-="     ;
MULT_EQUALS             : "*="     ;
DIV_EQUALS      : "/="     ;
EQUALS_EQUALS           : "=="     ;
AND             : "&&"   ;
OR              : "||"       ;
NOT             : '!'        ;
TYPE_ACCESSOR           : "'s"      ;

// Whitespace -- ignored
WS
  : ( ' '
    | '\t'
    | '\f'

    // handle newlines
    | ('\n' | ('\r' '\n') => '\r' '\n' | '\r')
      { newline(); }

    )
    { $setType(Token.SKIP); }
  ;


//an identifier.  Note that testLiterals is set to true!  This means
//that after we match the rule, we look in the literals table to see
//if it's a literal or really an identifer
IDENT
  options {testLiterals=true;}
            : ('a'..'z'|'A'..'Z'|'_') ('a'..'z'|'A'..'Z'|'0'..'9'|'_')*
            ;


class JAWSParser extends Parser;

options {
            k = 2;
```

```
	buildAST = true;
	exportVocab = JAWS;
}

tokens {
	PROGRAM;
	ARGS;
	CODEBLOCK;
	VAR;
	FORCON;
}




{
	int nr_error = 0;
	public void reportError( String s ) {
		super.reportError( s );
		nr_error++;
	}
	public void reportError( RecognitionException e ) {
		super.reportError( e );
		nr_error++;
	}
}

program
	:	"JAWS"! COLON! IDENT SEMI! declarBlock engineBlock EOF!
		{ #program = #([PROGRAM,"PROGRAM"], program); }
	;

declarBlock
	:	"declarations"^ LBRACE! (declarations)* RBRACE!
	;

declarations
	: intDecl
	| doubleDecl
	| stringDecl
	| booleanDecl
	| pointDecl
	| entityDecl
	| movementDecl
	| attackDecl
	| mapDecl
	//| const
	//| funcDecl
	;

/*
funcDecl
	: "func"^
	  ("int"|"double"|"string"|"boolean"|"point"|"entity"|"movement"|"attack"|"map"|"void")
	  IDENT LPAREN! (args)? RPAREN! LBRACE! codeBlock RBRACE!
	;
```

```
*/

intDecl
	: "int"^ IDENT ( EQUALS! ( NUMBER | IDENT ) )? SEMI!
	;

doubleDecl
	: "double"^ IDENT ( EQUALS! ( NUMBER | IDENT ) )? SEMI!
	;

stringDecl
	: "string"^ IDENT ( EQUALS! ( STRING | IDENT ) )? SEMI!
	;

booleanDecl
	: "boolean"^ IDENT ( EQUALS! ( "true" | "false" | IDENT ) )? SEMI!
	;

pointDecl
	: "point"^ IDENT ( EQUALS! "point"! LPAREN! NUMBER COMMA! NUMBER RPAREN! )?
SEMI!
	;

entityDecl
	: "entity"^ IDENT LBRACKET! NUMBER RBRACKET! SEMI!
	  (entityInit)?
	;

entityInit
	: IDENT! LPAREN! RPAREN! LBRACE!
	  ( mwidth)?
	  (mheight)?
	  (ex)?
	  (ey)?
	  (eimage)?
	  (eweapon)?
	  (eai)?
	  (elabel)?
/*
	  ("width"^ EQUALS! NUMBER SEMI!)?
	  ("height"^ EQUALS! NUMBER SEMI!)?
	  ("x_val"^ EQUALS! NUMBER SEMI!)?
	  ("y_val"^ EQUALS! NUMBER SEMI!)?
	  ("image"^ EQUALS! STRING SEMI!)?
	  ("weapon"^ EQUALS! STRING SEMI!)?
	  ("ai"^ EQUALS! STRING SEMI!)?
	  ("label"^ EQUALS! STRING SEMI!)?
*/

	   RBRACE!
	;

/*ewidth
	: "width"^ EQUALS! NUM SEMI!
	;
```

```
eheight
	: "height"^ EQUALS! NUM SEMI!
	;

*/



ex
	: "_x"^ EQUALS! NUMBER SEMI!
	;

ey
	:  "_y"^ EQUALS! NUMBER SEMI!
	;

eimage
	:  "_image"^ EQUALS! STRING SEMI!
	;

eweapon
	:  "_weapon"^ EQUALS! STRING SEMI!
	;

eai
	:  "_ai"^ EQUALS! STRING SEMI!
	;

elabel
	:  "_label"^ EQUALS! STRING SEMI!
	;

movementDecl
	:  "movement"^ IDENT LPAREN! (args)? RPAREN!
	  LBRACE! codeBlock RBRACE!
	;

args
	: parameter (COMMA! parameter)*
	  { #args = #([ARGS,"ARGS"], args); }
	;


parameter
	: ("int"|"double"|"string"|"boolean"|"point"|"entity"|"movement"|"attack"|"map")
	  IDENT
	;

attackDecl
	: "attack" IDENT LBRACE! codeBlock RBRACE!
	;

mapDecl
	: "map"^ IDENT SEMI! (mapInit)?
	;
```

```
mapInit
	: IDENT! LPAREN! RPAREN! LBRACE!
	  (mwidth)?
	  (mheight)?
	  (mbg)?
	  RBRACE!
	;

mwidth
	:  "_width"^ EQUALS! NUMBER SEMI!
	;

mheight
	:  "_height"^ EQUALS! NUMBER SEMI!
	;

mbg
	:  "_background"^ EQUALS! STRING SEMI!
	;

engineBlock
	:    "engine"^ LBRACE! (statement)* RBRACE!
	;

statement
	: assignmentStatement
	| ifStatement
	| whileStatement
	| forStatement
	//| funcCallStatement
	//| returnStatement
	;

/*
returnStatement
	: "return"^ (expression)? SEMI!
	;


funcCallStatement
	: funcCall SEMI!
	;

funcCall
	: IDENT LPAREN! (args)? RPAREN!
	  { #funcCall = #([FUNCCALL,"FUNCALL"], funcCall); }
	;
*/

assignmentStatement
	: variable
	  ( EQUALS^ | PLUS_EQUALS^ | MINUS_EQUALS^ | MULT_EQUALS^ | DIV_EQUALS^ )
	  expression SEMI!
	;

variable
```

```
        : IDENT ( LBRACKET! NUMBER RBRACKET!)? ( TYPE_ACCESSOR IDENT)*
          { #variable = #([VAR,"VAR"],variable); }
        ;

expression
        : logicExpression ( OR^ logicExpression)*
        ;

logicExpression
        : logicExpression2 ( AND^ logicExpression2)*
        ;

logicExpression2
        : (NOT^)? relationExpression
        ;

relationExpression
        : arithmeticExpression (
          (GTE^ | LTE^ | GT^ | LT^ | EQUALS_EQUALS^ | NOT_EQUA LS^ ) arithmeticExpression)?
        ;

arithmeticExpression
        : arithmeticExpression2 ( (PLUS^ | MINUS^) arithmeticExpression2 )*
        ;

arithmeticExpression2
        : arithmeticExpression3 ( (MULT^ | DIV^ | MOD^) arithmeticExpression3 )*
        ;

arithmeticExpression3
        : variable_r ( DISTANCE^ variable_r)*
        ;

variable_r
        : variable
        //| funcCall
        | NUMBER
        | STRING
        | "true"
        | "false"
        | LPAREN! expression RPAREN!
        ;

ifStatement
        : "if"^ expression LBRACE! codeBlock RBRACE!
          (options {greedy=true;} : "else"! LBRACE! codeBlock RBRACE!)?
        ;

codeBlock
        : (statement)*
          { #codeBlock = #([CODEBLOCK,"CODEBLOCK"], codeBlock); }
        ;

whileStatement
        : "while"^ expression LBRACE! codeBlock RBRACE!
        ;
```

```
forStatement
	: "for"^ forCondition LBRACE! codeBlock RBRACE!
	;

forCondition
	: IDENT EQUALS! NUMBER "to"! NUMBER ("by"! NUMBER)?
	  { #forCondition = #([FORCON,"FORCON"],forCondition); }
	;



//JAWS tree walker
class JAWSWalker extends TreeParser;

options {
	importVocab = JAWS;
	k=2;
}

{

	JAWSOutputter jo = new JAWSOutputter();
	String namep = "";
}




program
{ String title;}
	: #(PROGRAM title=id declarBlock engineBlock)
	  { jo.setName(title);jo.outputFile(title+".java"); }
	;

declarBlock
	: #("declarations" (declarations)*)
	;

declarations
{String s; String c;}
	: #("int" IDENT NUMBER
	  {
		double tmp = Double.parseDouble(#NUMBER.getText());
		JAWSInt a = new JAWSInt( (int)tmp );
		a.setName(#IDENT.getText());
		jo.addInt(a);
	  } )
	| #("double" IDENT NUMBER
	  {
		double tmp = Double.parseDouble(#NUMBER.getText());
		JAWSDouble a = new JAWSDouble( tmp );
		a.setName(#IDENT.getText());
		jo.addDouble(a);

	  } )
```

```
| #("string" IDENT STRING
  {
            String tmp = #STRING.getText();
            JAWSString a = new JAWSString( tmp );
            a.setName(#IDENT.getText());
            jo.addString(a);

  } )
| #("boolean"{boolean tmp;} IDENT ("false"{tmp = false;}|"true"{tmp = true;})
  {

            JAWSBoolean a = new JAWSBoolean(tmp);
            a.setName(#IDENT.getText());
            jo.addBoolean(a);
  } )
| #("point" s=pred:id
  {
            JAWSpoint p;
            AST next = pred.getNextSibling();
            if (next == null)
               p = new JAWSpoint(0,0,s);
            else
            {
               int a = Integer.parseInt(next.getText());
               next = next.getNextSibling();
               int b = Integer.parseInt(next.getText());
               p = new JAWSpoint(a,b,s);
            }
            jo.addPoint(p);

  } )
| #("map" s=pred2:id
  {
            Map m = new Map();
            m.name=s;
            AST next = pred2.getNextSibling();
            while (next != null)
            {
               if ( (next.getText()).equals("_width") )
               {

                        AST child = next.getFirstChild();
                        m.width = Integer.parseInt( child.getText() );
               }
               else if ( (next.getText()).equals("_height") )
               {

                        AST child = next.getFirstChild();
                        m.height = Integer.parseInt( child.getText() );
               }
               else if ( (next.getText()).equals("_background") )
               {

                        AST child = next.getFirstChild();
                        m.file = new java.io.File( child.getText() );
               }
```

```
            next = next.getNextSibling();
        }
        jo.addMap(m);
  })
| #("entity" s=id c=pred3:num
  {
        JAWSEntity je = new JAWSEntity();
        je.name = s;
        je.mult = Integer.parseInt(c);
        AST next = pred3.getNextSibling();
        while(next!=null)
        {
          if ( (next.getText()).equals("_width") )
          {

                AST child = next.getFirstChild();
                je.width = Integer.parseInt( child.getText() );
          }
          else if ( (next.getText()).equals("_height") )
          {

                AST child = next.getFirstChild();
                je.height = Integer.parseInt( child.getText() );
          }
          else if ( (next.getText()).equals("_x") )
          {

                AST child = next.getFirstChild();
                je.x = Integer.parseInt( child.getText() );
          }
          else if ( (next.getText()).equals("_y") )
          {

                AST child = next.getFirstChild();
                je.y = Integer.parseInt( child.getText() );
          }
          else if ( (next.getText()).equals("_image") )
          {

                AST child = next.getFirstChild();
                je.image = child.getText() ;
          }
          else if ( (next.getText()).equals("_weapon") )
          {

                AST child = next.getFirstChild();
                je.weapon = child.getText() ;
          }
          else if ( (next.getText()).equals("_ai") )
          {

                AST child = next.getFirstChild();
                je.ai = child.getText() ;
          }
          else if ( (next.getText()).equals("_label") )
          {
```

```
                        AST child = next.getFirstChild();
                        je.label = child.getText() ;
                    }
                    next = next.getNextSibling();
                }
                jo.addEntity(je);
            })

        ;

engineBlock
{ String st=""; }
        : #("engine" (st=pred:statement)?
          { if (!st.equals("")) {
            jo.addEngineStatement(st);


            AST next = pred.getNextSibling();
            while (next != null)
            {
                st = statement(next); jo.addEngineStatement(st);
                next = next.getNextSibling();
            }
            }

          }
          )
        | #("int" IDENT NUMBER
          {
                double tmp = Double.parseDouble(#NUMBER.getText());
                JAWSInt a = new JAWSInt( (int)tmp );
                a.setName(#IDENT.getText());
                jo.addInt(a);
          } )
        | #("double" IDENT NUMBER
          {
                double tmp = Double.parseDouble(#NUMBER.getText());
                JAWSDouble a = new JAWSDouble( tmp );
                a.setName(#IDENT.getText ());
                jo.addDouble(a);

          } )
        | #("string" IDENT STRING
          {
                String tmp = #STRING.getText();
                JAWSString a = new JAWSString( tmp );
                a.setName(#IDENT.getText());
                jo.addString(a);

          } )
        | #("boolean"{boolean tmp;} IDENT ("false"{tmp = false;}|"true"{tmp = true;})
          {

                JAWSBoolean a = new JAWSBoolean(tmp);
                a.setName(#IDENT.getText());
```

```
                jo.addBoolean(a);
            } )
          ;

statement returns [String r]
{String v; String e; String cb; r= "";}
          : #(EQUALS v=variable e=expression
            {r = v + " = " + e + ";";}
            )
          | #("if" e=expression cb=pred:codeBlock
            {
              r = "if "+e+"\r\n{\r\n"+cb+"}\r\n";
              AST next = pred.getNextSibling();
              if (next != null)
              {
                r+= "else {\r\n"+codeBlock(next)+"}\r\n";
              }
            }
            )
          | #("while" e=expression cb=codeBlock)
            {
              r = "while" + e + "\r\n{\r\n"+cb+"}\r\n";
            }
          | #("for" e=forCondition cb=codeBlock)
            {
              r = e + "\r\n{\r\n"+cb+"}\r\n";
            }
          ;

forCondition returns [String r]
{r = ""; String a; String b; String c;}
          : #(FORCON a=id b=num c=pred:num)
            {
              String inc = "1";
              r = "for (int "+a+"="+b+";"+a+"<="+c+";"+a+"+=";
              AST next = pred.getNextSibling();
              if (next != null)
              {
                inc = num(next);
              }
              r += inc+")";
            }
          ;

id returns [String r]
{r = ""; }
          : IDENT { r = #IDENT.getText(); }
          ;

num returns [String r]
{r = ""; }
          : NUMBER { r = #NUMBER.getText(); }
          ;

variable returns [String r]
{r = ""; String s; String a;}
```

```
        : #(VAR s=pred:id)
          {
            r = s;

            AST next = pred.getNextSibling();

            if (next != null)
            {
               if ( !(next.getText()).equals("'s") )
                     {
                        r += "["+next.getText()+"]";

                        next = next.getNextSibling();
                        //System.out.println(r);
                     }

                     while ( (next != null) && (next.getText()).equals("'s") )
                     {
                        r += ".";
                        next = next.getNextSibling();
                        r += next.getText();
                        next = next.getNextSibling();
                     }
            }
          }


        ;

expression returns [String r]
{r = ""; String a; String b;}
        : IDENT { r = #IDENT.getText(); }
        | NUMBER { r = #NUMBER.getText(); }
        | STRING { r = "\""+#STRING.getText()+"\""; }
        | "true" { r = "true"; }
        | "false" { r = "false";}
        | #(VAR a=pred:expression
          {
            r = a;
            AST next = pred.getNextSibling();
            if (next != null)
               r += "["+ next.getText() +"]" ;
          }
          )
        | #(DISTANCE a=expression b=expression
          {
            r = "( JAWSpoint.distance("+a+","+b+") )";
          })
        | #(MULT a=expression b=expression
          {
            r = "("+a + " * " + b+")";
          })
        | #(DIV a=expression b=exp ression
          {
            r = "("+a + " / " + b+")";
          })
```

```
            | #(MOD a=expression b=expression
              {
                r = "("+a + " % " + b+")";
              })
            | #(PLUS a=expression b=expression
              {
                r = "("+a + " + " + b+")";
              })
            | #(MINUS a=expression b=expression
              {
                r = "("+a + " - " + b+")";
              })
            | #(GTE a=expression b=expression
              {
                r = "("+a + " >= " + b+")";
              })
            | #(LTE a=expression b=expression
              {
                r = "("+a + " <= " + b+")";
              })
            | #(GT a=expression b=expression
              {
                r = "("+a + " > " + b+")";
              })
            | #(LT a=expression b=expression
              {
                r = "("+a + " < " + b+")";
              })
            | #(EQUALS_EQUALS a=expression b=expression
              {
                r = "("+a + " == " + b+")";
              })
            | #(NOT_EQUALS a=expression b=expression
              {
                r = "("+a + " != " + b+")";
              })
            | #(NOT a=expression
              {
                r = "(!"+a+ ")";
              })
            | #(AND a=expression b=expression
              {
                r = "("+a + " && " + b+")";
              })
            | #(OR a=expression b=expression
              {
                r = "("+a + " || " + b+")";
              })
            ;

codeBlock returns [String r]
{r = ""; String s;}
            : #(CODEBLOCK s=pred:statement)
              {
                 r += ( s + "\r\n" );
                 AST next = pred.getNextSibling();
```

```
            while (next != null)
            {
                    s = statement(next);
               r += ( s + "\r\n" );
               next = next .getNextSibling();

            }
          }
        ;



// $ANTLR 2.7.2: "jaws.g" -> "JAWSParser.java"$

import antlr.TokenBuffer;
import antlr.TokenStreamException;
import antlr.TokenStreamIOException;
import antlr.ANTLRException;
import antlr.LLkParser;
import antlr.Token;
import antlr.TokenStream;
import antlr.RecognitionException;
import antlr.NoViableAltException;
import antlr.MismatchedTokenException;
import antlr.SemanticException;
import antlr.ParserSharedInputState;
import antlr.collections.impl.BitSet;
import antlr.collections.AST;
import java.util.Hashtable;
import antlr.ASTFactory;
import antlr.ASTPair;
import antlr.collections.impl.ASTArray;

public class JAWSParser extends antlr.LLkParser       implements JAWSTokenTypes
 {

        int nr_error = 0;
        public void reportError( String s ) {
                super.reportError( s );
                nr_error++;
        }
        public void reportError( RecognitionException e ) {
                super.reportError( e );
                nr_error++;
        }

protected JAWSParser(TokenBuffer tokenBuf, int k) {
  super(tokenBuf,k);
  tokenNames = _tokenNames;
  buildTokenTypeASTClassMap();
  astFactory = new ASTFactory(getTokenTypeToASTClassMap());
}

public JAWSParser(TokenBuffer tokenBuf) {
  this(tokenBuf,2);
}
```

```
protected JAWSParser(TokenStream lexer, int k) {
 super(lexer,k);
 tokenNames = _tokenNames;
 buildTokenTypeASTClassMap();
 astFactory = new ASTFactory(getTokenTypeToASTClassMap());
}

public JAWSParser(TokenStream lexer) {
 this(lexer,2);
}

public JAWSParser(ParserSharedInputState state) {
 super(state,2);
 tokenNames = _tokenNames;
 buildTokenTypeASTClassMap();
 astFactory = new ASTFactory(getTokenTypeToASTClassMap());
}

        public final void program() throws RecognitionException, TokenStreamException {

                returnAST = null;
                ASTPair currentAST = new ASTPair();
                AST program_AST = null;

                try {      // for error handling
                        match(LITERAL_JAWS);
                        match(COLON);
                        AST tmp65_AST = null;
                        tmp65_AST = astFactory.create(LT(1));
                        astFactory.addASTChild(currentAST, tmp65_AST);
                        match(IDENT);
                        match(SEMI);
                        declarBlock();
                        astFactory.addASTChild(currentAST, returnAST);
                        engineBlock();
                        astFactory.addASTChild(currentAST, returnAST);
                        match(Token.EOF_TYPE);
                        program_AST = (AST)currentAST.root;
                        program_AST = (AST)astFactory.make( (new
ASTArray(2)).add(astFactory.create(PROGRAM,"PROGRAM")).add(program_AST));
                        currentAST.root = program_AST;
                        currentAST.child = program_AST!=null
&&program_AST.getFirstChild()!=null ?
                                        program_AST.getFirstChild() : program_AST;
                        currentAST.advanceChildToEnd();
                        program_AST = (AST)currentAST.root;
                }
                catch (RecognitionException ex) {
                        reportError(ex);
                        consume();
                        consumeUntil(_tokenSet_0);
                }
                returnAST = program_AST;
        }
```

```
public final void declarBlock() throws RecognitionException, TokenStreamException {

	returnAST = null;
	ASTPair currentAST = new ASTPair();
	AST declarBlock_AST = null;

	try {	// for error handling
		AST tmp68_AST = null;
		tmp68_AST = astFactory.create(LT(1));
		astFactory.makeASTRoot(currentAST, tmp68_AST);
		match(LITERAL_declarations);
		match(LBRACE);
		{
		_loop61:
		do {
			if ((_tokenSet_1.member(LA(1)))) {
				declarations();
				astFactory.addASTChild(currentAST, returnAST);
			}
			else {
				break _loop61;
			}

		} while (true);
		}
		match(RBRACE);
		declarBlock_AST = (AST)currentAST.root;
	}
	catch (RecognitionException ex) {
		reportError(ex);
		consume();
		consumeUntil(_tokenSet_2);
	}
	returnAST = declarBlock_AST;
}

public final void engineBlock() throws RecognitionException, TokenStreamException {

	returnAST = null;
	ASTPair currentAST = new ASTPair();
	AST engineBlock_AST = null;

	try {	// for error handling
		AST tmp71_AST = null;
		tmp71_AST = astFactory.create(LT(1));
		astFactory.makeASTRoot(currentAST, tmp71_AST);
		match(LITERAL_engine);
		match(LBRACE);
		{
		_loop113:
		do {
			if ((_tokenSet_3.member(LA(1)))) {
				statement();
				astFactory.addASTChild(currentAST, returnAST);
			}
			else {
```

```
                                        break _loop113;
                                }

                        } while (true);
                        }
                        match(RBRACE);
                        engineBlock_AST = (AST)currentAST.root;
                }
                catch (RecognitionException ex) {
                        reportError(ex);
                        consume();
                        consumeUntil(_tokenSet_0);
                }
                returnAST = engineBlock_AST;
}

public final void declarations() throws RecognitionException, TokenStreamException {

                returnAST = null;
                ASTPair currentAST = new ASTPair();
                AST declarations_AST = null;

                try {      // for error handling
                        switch ( LA(1)) {
                        case LITERAL_int:
                        {
                                intDecl();
                                astFactory.addASTChild(currentAST, returnAST);
                                declarations_AST = (AST)currentAST.root;
                                break;
                        }
                        case LITERAL_double:
                        {
                                doubleDecl();
                                astFactory.addASTChild(currentAST, returnAST);
                                declarations_AST = (AST)currentAST.root;
                                break;
                        }
                        case LITERAL_string:
                        {
                                stringDecl();
                                astFactory.addASTChild(currentAST, returnAST);
                                declarations_AST = (AST)currentAST.root;
                                break;
                        }
                        case LITERAL_boolean:
                        {
                                booleanDecl();
                                astFactory.addASTChild(currentAST, returnAST);
                                declarations_AST = (AST)currentAST.root;
                                break;
                        }
                        case LITERAL_point:
                        {
                                pointDecl();
                                astFactory.addASTChild(currentAST, returnAST);
```

```
                declarations_AST = (AST)currentAST.root;
                break;
            }
            case LITERAL_entity:
            {
                entityDecl();
                astFactory.addASTChild(currentAST, returnAST);
                declarations_AST = (AST)currentAST.root;
                break;
            }
            case LITERAL_movement:
            {
                movementDecl();
                astFactory.addASTChild(currentAST, returnAST);
                declarations_AST = (AST)currentAST.root;
                break;
            }
            case LITERAL_attack:
            {
                attackDecl();
                astFactory.addASTChild(currentAST, returnAST);
                declarations_AST = (AST)currentAST.root;
                break;
            }
            case LITERAL_map:
            {
                mapDecl();
                astFactory.addASTChild(currentAST, returnAST);
                declarations_AST = (AST)currentAST.root;
                break;
            }
            default:
            {
                throw new NoViableAltException(LT(1), getFilename());
            }
            }
        }
        catch (RecognitionException ex) {
            reportError(ex);
            consume();
            consumeUntil(_tokenSet_4);
        }
        returnAST = declarations_AST;
}

public final void intDecl() throws RecognitionException, TokenStreamException {

        returnAST = null;
        ASTPair currentAST = new ASTPair();
        AST intDecl_AST = null;

        try {      // for error handling
            AST tmp74_AST = null;
            tmp74_AST = astFactory.create(LT(1));
            astFactory.makeASTRoot(currentAST, tmp74_AST);
            match(LITERAL_int);
```

```
		AST tmp75_AST = null;
		tmp75_AST = astFactory.create(LT(1));
		astFactory.addASTChild(currentAST, tmp75_AST);
		match(IDENT);
		{
		switch ( LA(1)) {
		case EQUALS:
		{
			match(EQUALS);
			{
			switch ( LA(1)) {
			case NUMBER:
			{
				AST tmp77_AST = null;
				tmp77_AST = astFactory.create(LT(1));
				astFactory.addASTChild(currentAST, tmp77_AST);
				match(NUMBER);
				break;
			}
			case IDENT:
			{
				AST tmp78_AST = null;
				tmp78_AST = astFactory.create(LT(1));
				astFactory.addASTChild(currentAST, tmp78_AST);
				match(IDENT);
				break;
			}
			default:
			{
				throw new NoViableAltException(LT(1), getFilename());
			}
			}
			}
			break;
		}
		case SEMI:
		{
			break;
		}
		default:
		{
			throw new NoViableAltException(LT(1), getFilename());
		}
		}
		}
		match(SEMI);
		intDecl_AST = (AST)currentAST.root;
	}
	catch (RecognitionException ex) {
		reportError(ex);
		consume();
		consumeUntil(_tokenSet_4);
	}
	returnAST = intDecl_AST;
}
```

```
public final void doubleDecl() throws RecognitionException, TokenStreamException {

	returnAST = null;
	ASTPair currentAST = new ASTPair();
	AST doubleDecl_AST = null;

	try {	// for error handling
		AST tmp80_AST = null;
		tmp80_AST = astFactory.create(LT(1));
		astFactory.makeASTRoot(currentAST, tmp80_AST);
		match(LITERAL_double);
		AST tmp81_AST = null;
		tmp81_AST = astFactory.create(LT(1));
		astFactory.addASTChild(currentAST, tmp81_AST);
		match(IDENT);
		{
		switch ( LA(1)) {
		case EQUALS:
		{
			match(EQUALS);
			{
			switch ( LA(1)) {
			case NUMBER:
			{
				AST tmp83_AST = null;
				tmp83_AST = astFactory.create(LT(1));
				astFactory.addASTChild(currentAST, tmp83_AST);
				match(NUMBER);
				break;
			}
			case IDENT:
			{
				AST tmp84_AST = null;
				tmp84_AST = astFactory.create(LT(1));
				astFactory.addASTChild(currentAST, tmp84_AST);
				match(IDENT);
				break;
			}
			default:
			{
				throw new NoViableAltException(LT(1), getFilename());
			}
			}
			}
			break;
		}
		case SEMI:
		{
			break;
		}
		default:
		{
			throw new NoViableAltException(LT(1), getFilename());
		}
		}
		}
```

```
                match(SEMI);
                doubleDecl_AST = (AST)currentAST.root;
        }
        catch (RecognitionException ex) {
                reportError(ex);
                consume();
                consumeUntil(_tokenSet_4);
        }
        returnAST = doubleDecl_AST;
}

public final void stringDecl() throws RecognitionException, TokenStreamException {

        returnAST = null;
        ASTPair currentAST = new ASTPair();
        AST stringDecl_AST = null;

        try {      // for error handling
                AST tmp86_AST = null;
                tmp86_AST = astFactory.create(LT(1));
                astFactory.makeASTRoot(currentAST, tmp86_AST);
                match(LITERAL_string);
                AST tmp87_AST = null;
                tmp87_AST = astFactory.create(LT(1));
                astFactory.addASTChild(currentAST, tmp87_AST);
                match(IDENT);
                {
                switch ( LA(1)) {
                case EQUALS:
                {
                        match(EQUALS);
                        {
                        switch ( LA(1)) {
                        case STRING:
                        {
                                AST tmp89_AST = null;
                                tmp89_AST = astFactory.create(LT(1));
                                astFactory.addASTChild(currentAST, tmp89_AST);
                                match(STRING);
                                break;
                        }
                        case IDENT:
                        {
                                AST tmp90_AST = null;
                                tmp90_AST = astFactory.create(LT(1));
                                astFactory.addASTChild(currentAST, tmp90_AST);
                                match(IDENT);
                                break;
                        }
                        default:
                        {
                                throw new NoViableAltException(LT(1), getFilename());
                        }
                        }
                        }
                        break;
```

```
                }
                case SEMI:
                {
                        break;
                }
                default:
                {
                        throw new NoViableAltException(LT(1), getFilename());
                }
                }
                }
                match(SEMI);
                stringDecl_AST = (AST)currentAST.root;
        }
        catch (RecognitionException ex) {
                reportError(ex);
                consume();
                consumeUntil(_tokenSet_4);
        }
        returnAST = stringDecl_AST;
}

public final void booleanDecl() throws RecognitionException, TokenStreamException {

        returnAST = null;
        ASTPair currentAST = new ASTPair();
        AST booleanDecl_AST = null;

        try {      // for error handling
                AST tmp92_AST = null;
                tmp92_AST = astFactory.create(LT(1));
                astFactory.makeASTRoot(currentAST, tmp92_AST);
                match(LITERAL_boolean);
                AST tmp93_AST = null;
                tmp93_AST = astFactory.create(LT(1));
                astFactory.addASTChild(currentAST, tmp93_AST);
                match(IDENT);
                {
                switch ( LA(1)) {
                case EQUALS:
                {
                        match(EQUALS);
                        {
                        switch ( LA(1)) {
                        case LITERAL_true:
                        {
                                AST tmp95_AST = null;
                                tmp95_AST = astFactory.create(LT(1));
                                astFactory.addASTChild(currentAST, tmp95_AST);
                                match(LITERAL_true);
                                break;
                        }
                        case LITERAL_false:
                        {
                                AST tmp96_AST = null;
                                tmp96_AST = astFactory.create(LT(1));
```

```
                                        astFactory.addASTChild(currentAST, tmp96_AST);
                                        match(LITERAL_false);
                                        break;
                                }
                                case IDENT:
                                {
                                        AST tmp97_AST = null;
                                        tmp97_AST = astFactory.create(LT(1));
                                        astFactory.addASTChild(currentAST, tmp97_AST);
                                        match(IDENT);
                                        break;
                                }
                                default:
                                {
                                        throw new NoViableAltException(LT(1), getFilename());
                                }
                                }
                                }
                                break;
                        }
                        case SEMI:
                        {
                                break;
                        }
                        default:
                        {
                                throw new NoViableAltException(LT(1), getFilename());
                        }
                        }
                        }
                        match(SEMI);
                        booleanDecl_AST = (AST)currentAST.root;
                }
                catch (RecognitionException ex) {
                        reportError(ex);
                        consume();
                        consumeUntil(_tokenSet_4);
                }
                returnAST = booleanDecl_AST;
        }

        public final void pointDecl() throws RecognitionException, TokenStreamException {

                returnAST = null;
                ASTPair currentAST = new ASTPair();
                AST pointDecl_AST = null;

                try {      // for error handling
                        AST tmp99_AST = null;
                        tmp99_AST = astFactory.create(LT(1));
                        astFactory.makeASTRoot(currentAST, tmp99_AST);
                        match(LITERAL_point);
                        AST tmp100_AST = null;
                        tmp100_AST = astFactory.create(LT(1));
                        astFactory.addASTChild(currentAST, tmp100_AST);
                        match(IDENT);
```

```
                {
                switch ( LA(1)) {
                case EQUALS:
                {
                        match(EQUALS);
                        match(LITERAL_point);
                        match(LPAREN);
                        AST tmp104_AST = null;
                        tmp104_AST = astFactory.create(LT(1));
                        astFactory.addASTChild(currentAST, tmp104_AST);
                        match(NUMBER);
                        match(COMMA);
                        AST tmp106_AST = null;
                        tmp106_AST = astFactory.create(LT(1));
                        astFactory.addASTChild(currentAST, tmp106_AST);
                        match(NUMBER);
                        match(RPAREN);
                        break;
                }
                case SEMI:
                {
                        break;
                }
                default:
                {
                        throw new NoViableAltException(LT(1), getFilename());
                }
                }
                }
                match(SEMI);
                pointDecl_AST = (AST)currentAST.root;
        }
        catch (RecognitionException ex) {
                reportError(ex);
                consume();
                consumeUntil(_tokenSet_4);
        }
        returnAST = pointDecl_AST;
}

public final void entityDecl() throws RecognitionException, TokenStreamException {

        returnAST = null;
        ASTPair currentAST = new ASTPair();
        AST entityDecl_AST = null;

        try {     // for error handling
                AST tmp109_AST = null;
                tmp109_AST = astFactory.create(LT(1));
                astFactory.makeASTRoot(currentAST, tmp109_AST);
                match(LITERAL_entity);
                AST tmp110_AST = null;
                tmp110_AST = astFactory.create(LT(1));
                astFactory.addASTChild(currentAST, tmp110_AST);
                match(IDENT);
                match(LBRACKET);
```

```
                AST tmp112_AST = null;
                tmp112_AST = astFactory.create(LT(1));
                astFactory.addASTChild(currentAST, tmp112_AST);
                match(NUMBER);
                match(RBRACKET);
                match(SEMI);
                {
                switch ( LA(1)) {
                case IDENT:
                {
                        entityInit();
                        astFactory.addASTChild(currentAST, returnAST);
                        break;
                }
                case RBRACE:
                case LITERAL_int:
                case LITERAL_double:
                case LITERAL_string:
                case LITERAL_boolean:
                case LITERAL_point:
                case LITERAL_entity:
                case LITERAL_movement:
                case LITERAL_attack:
                case LITERAL_map:
                {
                        break;
                }
                default:
                {
                        throw new NoViableAltException(LT(1), getFilename());
                }
                }
                }
                entityDecl_AST = (AST)currentAST.root;
        }
        catch (RecognitionException ex) {
                reportError(ex);
                consume();
                consumeUntil(_tokenSet_4);
        }
        returnAST = entityDecl_AST;
}

public final void movementDecl() throws RecognitionException, TokenStreamException {

        returnAST = null;
        ASTPair currentAST = new ASTPair();
        AST movementDecl_AST = null;

        try {      // for error handling
                AST tmp115_AST = null;
                tmp115_AST = astFactory.create(LT(1));
                astFactory.makeASTRoot(currentAST, tmp115_AST);
                match(LITERAL_movement);
                AST tmp116_AST = null;
                tmp116_AST = astFactory.create(LT(1));
```

```
                astFactory.addASTChild(currentAST, tmp116_AST);
                match(IDENT);
                match(LPAREN);
                {
                switch ( LA(1)) {
                case LITERAL_int:
                case LITERAL_double:
                case LITERAL_string:
                case LITERAL_boolean:
                case LITERAL_point:
                case LITERAL_entity:
                case LITERAL_movement:
                case LITERAL_attack:
                case LITERAL_map:
                {
                        args();
                        astFactory.addASTChild(currentAST, returnAST);
                        break;
                }
                case RPAREN:
                {
                        break;
                }
                default:
                {
                        throw new NoViableAltException(LT(1), getFilename());
                }
                }
                }
                match(RPAREN);
                match(LBRACE);
                codeBlock();
                astFactory.addASTChild(currentAST, returnAST);
                match(RBRACE);
                movementDecl_AST = (AST)currentAST.root;
        }
        catch (RecognitionException ex) {
                reportError(ex);
                consume();
                consumeUntil(_tokenSet_4);
        }
        returnAST = movementDecl_AST;
}

public final void attackDecl() throws RecognitionException, TokenStreamException {

        returnAST = null;
        ASTPair currentAST = new ASTPair();
        AST attackDecl_AST = null;

        try {      // for error handling
                AST tmp121_AST = null;
                tmp121_AST = astFactory.create(LT(1));
                astFactory.addASTChild(currentAST, tmp121_AST);
                match(LITERAL_attack);
                AST tmp122_AST = null;
```

```
                tmp122_AST = astFactory.create(LT(1));
                astFactory.addASTChild(currentAST, tmp122_AST);
                match(IDENT);
                match(LBRACE);
                codeBlock();
                astFactory.addASTChild(currentAST, returnAST);
                match(RBRACE);
                attackDecl_AST = (AST)currentAST.root;
        }
        catch (RecognitionException ex) {
                reportError(ex);
                consume();
                consumeUntil(_tokenSet_4);
        }
        returnAST = attackDecl_AST;
}

public final void mapDecl() throws RecognitionException, TokenStreamException {

        returnAST = null;
        ASTPair currentAST = new ASTPair();
        AST mapDecl_AST = null;

        try {      // for error handling
                AST tmp125_AST = null;
                tmp125_AST = astFactory.create(LT(1));
                astFactory.makeASTRoot(currentAST, tmp125_AST);
                match(LITERAL_map);
                AST tmp126_AST = null;
                tmp126_AST = astFactory.create(LT(1));
                astFactory.addASTChild(currentAST, tmp126_AST);
                match(IDENT);
                match(SEMI);
                {
                switch ( LA(1)) {
                case IDENT:
                {
                        mapInit();
                        astFactory.addASTChild(currentAST, returnAST);
                        break;
                }
                case RBRACE:
                case LITERAL_int:
                case LITERAL_double:
                case LITERAL_string:
                case LITERAL_boolean:
                case LITERAL_point:
                case LITERAL_entity:
                case LITERAL_movement:
                case LITERAL_attack:
                case LITERAL_map:
                {
                        break;
                }
                default:
                {
```

```
                        throw new NoViableAltException(LT(1), getFilename());
                }
                }
                }
                mapDecl_AST = (AST)currentAST.root;
        }
        catch (RecognitionException ex) {
                reportError(ex);
                consume();
                consumeUntil(_tokenSet_4);
        }
        returnAST = mapDecl_AST;
}

public final void entityInit() throws RecognitionException, TokenStreamException {

        returnAST = null;
        ASTPair currentAST = new ASTPair();
        AST entityInit_AST = null;

        try {      // for error handling
                match(IDENT);
                match(LPAREN);
                match(RPAREN);
                match(LBRACE);
                {
                switch ( LA(1)) {
                case LITERAL__width:
                {
                        mwidth();
                        astFactory.addASTChild(currentAST, returnAST);
                        break;
                }
                case RBRACE:
                case LITERAL__x:
                case LITERAL__y:
                case LITERAL__image:
                case LITERAL__weapon:
                case LITERAL__ai:
                case LITERAL__label:
                case LITERAL__height:
                {
                        break;
                }
                default:
                {
                        throw new NoViableAltException(LT(1), getFilename());
                }
                }
                }
                {
                switch ( LA(1)) {
                case LITERAL__height:
                {
                        mheight();
                        astFactory.addASTChild(currentAST, returnAST);
```

```
		break;
}
case RBRACE:
case LITERAL__x:
case LITERAL__y:
case LITERAL__image:
case LITERAL__weapon:
case LITERAL__ai:
case LITERAL__label:
{
		break;
}
default:
{
		throw new NoViableAltException(LT(1), getFilename());
}
}
}
{
switch ( LA(1)) {
case LITERAL__x:
{
		ex();
		astFactory.addASTChild(currentAST, returnAST);
		break;
}
case RBRACE:
case LITERAL__y:
case LITERAL__image:
case LITERAL__weapon:
case LITERAL__ai:
case LITERAL__label:
{
		break;
}
default:
{
		throw new NoViableAltException(LT(1), getFilename());
}
}
}
{
switch ( LA(1)) {
case LITERAL__y:
{
		ey();
		astFactory.addASTChild(currentAST, returnAST);
		break;
}
case RBRACE:
case LITERAL__image:
case LITERAL__weapon:
case LITERAL__ai:
case LITERAL__label:
{
		break;
```

```
}
default:
{
	throw new NoViableAltException(LT(1), getFilename());
}
}
}
{
switch ( LA(1)) {
case LITERAL__image:
{
	eimage();
	astFactory.addASTChild(currentAST, returnAST);
	break;
}
case RBRACE:
case LITERAL__weapon:
case LITERAL__ai:
case LITERAL__label:
{
	break;
}
default:
{
	throw new NoViableAltException(LT(1), getFilename());
}
}
}
{
switch ( LA(1)) {
case LITERAL__weapon:
{
	eweapon();
	astFactory.addASTChild(currentAST, returnAST);
	break;
}
case RBRACE:
case LITERAL__ai:
case LITERAL__label:
{
	break;
}
default:
{
	throw new NoViableAltException(LT(1), getFilename());
}
}
}
{
switch ( LA(1)) {
case LITERAL__ai:
{
	eai();
	astFactory.addASTChild(currentAST, returnAST);
	break;
}
```

```
                    case RBRACE:
                    case LITERAL__label:
                    {
                            break;
                    }
                    default:
                    {
                            throw new NoViableAltException(LT(1), getFilename());
                    }
                    }
                    }
                    {
                    switch ( LA(1)) {
                    case LITERAL__label:
                    {
                            elabel();
                            astFactory.addASTChild(currentAST, returnAST);
                            break;
                    }
                    case RBRACE:
                    {
                            break;
                    }
                    default:
                    {
                            throw new NoViableAltException(LT(1), getFilename());
                    }
                    }
                    }
                    match(RBRACE);
                    entityInit_AST = (AST)currentAST.root;
            }
            catch (RecognitionException ex) {
                    reportError(ex);
                    consume();
                    consumeUntil(_tokenSet_4);
            }
            returnAST = entityInit_AST;
}

public final void mwidth() throws RecognitionException, TokenStreamException {

            returnAST = null;
            ASTPair currentAST = new ASTPair();
            AST mwidth_AST = null;

            try {      // for error handling
                    AST tmp133_AST = null;
                    tmp133_AST = astFactory.create(LT(1));
                    astFactory.makeASTRoot(currentAST, tmp133_AST);
                    match(LITERAL__width);
                    match(EQUALS);
                    AST tmp135_AST = null;
                    tmp135_AST = astFactory.create(LT(1));
                    astFactory.addASTChild(currentAST, tmp135_AST);
                    match(NUMBER);
```

```
                match(SEMI);
                mwidth_AST = (AST)currentAST.root;
        }
        catch (RecognitionException ex) {
                reportError(ex);
                consume();
                consumeUntil(_tokenSet_5);
        }
        returnAST = mwidth_AST;
}

public final void mheight() throws RecognitionException, TokenStreamException {

        returnAST = null;
        ASTPair currentAST = new ASTPair();
        AST mheight_AST = null;

        try {      // for error handling
                AST tmp137_AST = null;
                tmp137_AST = astFactory.create(LT(1));
                astFactory.makeASTRoot(currentAST, tmp137_AST);
                match(LITERAL__height);
                match(EQUALS);
                AST tmp139_AST = null;
                tmp139_AST = astFactory.create(LT(1));
                astFactory.addASTChild(currentAST, tmp139_AST);
                match(NUMBER);
                match(SEMI);
                mheight_AST = (AST)currentAST.root;
        }
        catch (RecognitionException ex) {
                reportError(ex);
                consume();
                consumeUntil(_tokenSet_6);
        }
        returnAST = mheight_AST;
}

public final void ex() throws RecognitionException, TokenStreamException {

        returnAST = null;
        ASTPair currentAST = new ASTPair();
        AST ex_AST = null;

        try {      // for error handling
                AST tmp141_AST = null;
                tmp141_AST = astFactory.create(LT(1));
                astFactory.makeASTRoot(currentAST, tmp141_AST);
                match(LITERAL__x);
                match(EQUALS);
                AST tmp143_AST = null;
                tmp143_AST = astFactory.create(LT(1));
                astFactory.addASTChild(currentAST, tmp143_AST);
                match(NUMBER);
                match(SEMI);
                ex_AST = (AST)currentAST.root;
```

```
		}
		catch (RecognitionException ex) {
			reportError(ex);
			consume();
			consumeUntil(_tokenSet_7);
		}
		returnAST = ex_AST;
}

public final void ey() throws RecognitionException, TokenStreamException {

		returnAST = null;
		ASTPair currentAST = new ASTPair();
		AST ey_AST = null;

		try {	// for error handling
			AST tmp145_AST = null;
			tmp145_AST = astFactory.create(LT(1));
			astFactory.makeASTRoot(currentAST, tmp145_AST);
			match(LITERAL__y);
			match(EQUALS);
			AST tmp147_AST = null;
			tmp147_AST = astFactory.create(LT(1));
			astFactory.addASTChild(currentAST, tmp147_AST);
			match(NUMBER);
			match(SEMI);
			ey_AST = (AST)currentAST.root;
		}
		catch (RecognitionException ex) {
			reportError(ex);
			consume();
			consumeUntil(_tokenSet_8);
		}
		returnAST = ey_AST;
}

public final void eimage() throws RecognitionException, TokenStreamException {

		returnAST = null;
		ASTPair currentAST = new ASTPair();
		AST eimage_AST = null;

		try {	// for error handling
			AST tmp149_AST = null;
			tmp149_AST = astFactory.create(LT(1));
			astFactory.makeASTRoot(currentAST, tmp149_AST);
			match(LITERAL__image);
			match(EQUALS);
			AST tmp151_AST = null;
			tmp151_AST = astFactory.create(LT(1));
			astFactory.addASTChild(currentAST, tmp151_AST);
			match(STRING);
			match(SEMI);
			eimage_AST = (AST)currentAST.root;
		}
		catch (RecognitionException ex) {
```

```
                reportError(ex);
                consume();
                consumeUntil(_tokenSet_9);
        }
        returnAST = eimage_AST;
}

public final void eweapon() throws RecognitionException, TokenStreamException {

        returnAST = null;
        ASTPair currentAST = new ASTPair();
        AST eweapon_AST = null;

        try {      // for error handling
                AST tmp153_AST = null;
                tmp153_AST = astFactory.create(LT(1));
                astFactory.makeASTRoot(currentAST, tmp153_AST);
                match(LITERAL__weapon);
                match(EQUALS);
                AST tmp155_AST = null;
                tmp155_AST = astFactory.create(LT(1));
                astFactory.addASTChild(currentAST, tmp155_AST);
                match(STRING);
                match(SEMI);
                eweapon_AST = (AST)currentAST.root;
        }
        catch (RecognitionException ex) {
                reportError(ex);
                consume();
                consumeUntil(_tokenSet_10);
        }
        returnAST = eweapon_AST;
}

public final void eai() throws RecognitionException, TokenStreamException {

        returnAST = null;
        ASTPair currentAST = new ASTPair();
        AST eai_AST = null;

        try {      // for error handling
                AST tmp157_AST = null;
                tmp157_AST = astFactory.create(LT(1));
                astFactory.makeASTRoot(currentAST, tmp157_AST);
                match(LITERAL__ai);
                match(EQUALS);
                AST tmp159_AST = null;
                tmp159_AST = astFactory.create(LT(1));
                astFactory.addASTChild(currentAST, tmp159_AST);
                match(STRING);
                match(SEMI);
                eai_AST = (AST)currentAST.root;
        }
        catch (RecognitionException ex) {
                reportError(ex);
                consume();
```

```
                consumeUntil(_tokenSet_11);
        }
        returnAST = eai_AST;
}

public final void elabel() throws RecognitionException, TokenStreamException {

        returnAST = null;
        ASTPair currentAST = new ASTPair();
        AST elabel_AST = null;

        try {      // for error handling
                AST tmp161_AST = null;
                tmp161_AST = astFactory.create(LT(1));
                astFactory.makeASTRoot(currentAST, tmp161_AST);
                match(LITERAL__label);
                match(EQUALS);
                AST tmp163_AST = null;
                tmp163_AST = astFactory.create(LT(1));
                astFactory.addASTChild(currentAST, tmp163_AST);
                match(STRING);
                match(SEMI);
                elabel_AST = (AST)currentAST.root;
        }
        catch (RecognitionException ex) {
                reportError(ex);
                consume();
                consumeUntil(_tokenSet_12);
        }
        returnAST = elabel_AST;
}

public final void args() throws RecognitionException, TokenStreamException {

        returnAST = null;
        ASTPair currentAST = new ASTPair();
        AST args_AST = null;

        try {      // for error handling
                parameter();
                astFactory.addASTChild(currentAST, returnAST);
                {
                _loop98:
                do {
                        if ((LA(1)==COMMA)) {
                                match(COMMA);
                                parameter();
                                astFactory.addASTChild(currentAST, returnAST);
                        }
                        else {
                                break _loop98;
                        }

                } while (true);
                }
                args_AST = (AST)currentAST.root;
```

```
                        args_AST = (AST)astFactory.make( (new
ASTArray(2)).add(astFactory.create(ARGS,"ARGS")).add(args_AST));
                        currentAST.root = args_AST;
                        currentAST.child = args_AST!=null &&args_AST.getFirstChild()!=null ?
                                args_AST.getFirstChild() : args_AST;
                        currentAST.advanceChildToEnd();
                        args_AST = (AST)currentAST.root;
                }
                catch (RecognitionException ex) {
                        reportError(ex);
                        consume();
                        consumeUntil(_tokenSet_13);
                }
                returnAST = args_AST;
        }

        public final void codeBlock() throws RecognitionException, TokenStreamException {

                returnAST = null;
                ASTPair currentAST = new ASTPair();
                AST codeBlock_AST = null;

                try {      // for error handling
                        {
                        _loop148:
                        do {
                                if ((_tokenSet_3.member(LA(1)))) {
                                        statement();
                                        astFactory.addASTChild(currentAST, returnAST);
                                }
                                else {
                                        break _loop148;
                                }

                        } while (true);
                        }
                        codeBlock_AST = (AST)currentAST.root;
                        codeBlock_AST = (AST)astFactory.make( (new
ASTArray(2)).add(astFactory.create(CODEBLOCK,"CODEBLOCK")).add(codeBlock_AST));
                        currentAST.root = codeBlock_AST;
                        currentAST.child = codeBlock_AST!=null
&&codeBlock_AST.getFirstChild()!=null ?
                                codeBlock_AST.getFirstChild() : codeBlock_AST;
                        currentAST.advanceChildToEnd();
                        codeBlock_AST = (AST)currentAST.root;
                }
                catch (RecognitionException ex) {
                        reportError(ex);
                        consume();
                        consumeUntil(_tokenSet_12);
                }
                returnAST = codeBlock_AST;
        }

        public final void parameter() throws RecognitionException, TokenStreamException {
```

```
returnAST = null;
ASTPair currentAST = new ASTPair();
AST parameter_AST = null;

try {      // for error handling
           {
           switch ( LA(1)) {
           case LITERAL_int:
           {
                      AST tmp166_AST = null;
                      tmp166_AST = astFactory.create(LT(1));
                      astFactory.addASTChild(currentAST, tmp166_AST);
                      match(LITERAL_int);
                      break;
           }
           case LITERAL_double:
           {
                      AST tmp167_AST = null;
                      tmp167_AST = astFactory.create(LT(1));
                      astFactory.addASTChild(currentAST, tmp167_AST);
                      match(LITERAL_double);
                      break;
           }
           case LITERAL_string:
           {
                      AST tmp168_AST = null;
                      tmp168_AST = astFactory.create(LT(1));
                      astFactory.addASTChild(currentAST, tmp168_AST);
                      match(LITERAL_string);
                      break;
           }
           case LITERAL_boolean:
           {
                      AST tmp169_AST = null;
                      tmp169_AST = astFactory.create(LT(1));
                      astFactory.addASTChild(currentAST, tmp169_AST);
                      match(LITERAL_boolean);
                      break;
           }
           case LITERAL_point:
           {
                      AST tmp170_AST = null;
                      tmp170_AST = astFactory.create(LT(1));
                      astFactory.addASTChild(currentAST, tmp170_AST);
                      match(LITERAL_point);
                      break;
           }
           case LITERAL_entity:
           {
                      AST tmp171_AST = null;
                      tmp171_AST = astFactory.create(LT(1));
                      astFactory.addASTChild(currentAST, tmp171_AST);
                      match(LITERAL_entity);
                      break;
           }
           case LITERAL_movement:
```

```
                {
                        AST tmp172_AST = null;
                        tmp172_AST = astFactory.create(LT(1));
                        astFactory.addASTChild(currentAST, tmp172_AST);
                        match(LITERAL_movement);
                        break;
                }
                case LITERAL_attack:
                {
                        AST tmp173_AST = null;
                        tmp173_AST = astFactory.create(LT(1));
                        astFactory.addASTChild(currentAST, tmp173_AST);
                        match(LITERAL_attack);
                        break;
                }
                case LITERAL_map:
                {
                        AST tmp174_AST = null;
                        tmp174_AST = astFactory.create(LT(1));
                        astFactory.addASTChild(currentAST, tmp174_AST);
                        match(LITERAL_map);
                        break;
                }
                default:
                {
                        throw new NoViableAltException(LT(1), getFilename());
                }
                }
                }
                AST tmp175_AST = null;
                tmp175_AST = astFactory.create(LT(1));
                astFactory.addASTChild(currentAST, tmp175_AST);
                match(IDENT);
                parameter_AST = (AST)currentAST.root;
        }
        catch (RecognitionException ex) {
                reportError(ex);
                consume();
                consumeUntil(_tokenSet_14);
        }
        returnAST = parameter_AST;
}

public final void mapInit() throws RecognitionException, TokenStreamException {

        returnAST = null;
        ASTPair currentAST = new ASTPair();
        AST mapInit_AST = null;

        try {      // for error handling
                match(IDENT);
                match(LPAREN);
                match(RPAREN);
                match(LBRACE);
                {
                switch ( LA(1)) {
```

```
case LITERAL__width:
{
        mwidth();
        astFactory.addASTChild(currentAST, returnAST);
        break;
}
case RBRACE:
case LITERAL__height:
case LITERAL__background:
{
        break;
}
default:
{
        throw new NoViableAltException(LT(1), getFilename());
}
}
}
{
switch ( LA(1)) {
case LITERAL__height:
{
        mheight();
        astFactory.addASTChild(currentAST, returnAST);
        break;
}
case RBRACE:
case LITERAL__background:
{
        break;
}
default:
{
        throw new NoViableAltException(LT(1), getFilename());
}
}
}
{
switch ( LA(1)) {
case LITERAL__background:
{
        mbg();
        astFactory.addASTChild(currentAST, returnAST);
        break;
}
case RBRACE:
{
        break;
}
default:
{
        throw new NoViableAltException(LT(1), getFilename());
}
}
}
match(RBRACE);
```

```
                mapInit_AST = (AST)currentAST.root;
        }
        catch (RecognitionException ex) {
                reportError(ex);
                consume();
                consumeUntil(_tokenSet_4);
        }
        returnAST = mapInit_AST;
}

public final void mbg() throws RecognitionException, TokenStreamException {

        returnAST = null;
        ASTPair currentAST = new ASTPair();
        AST mbg_AST = null;

        try {      // for error handling
                AST tmp181_AST = null;
                tmp181_AST = astFactory.create(LT(1));
                astFactory.makeASTRoot(currentAST, tmp181_AST);
                match(LITERAL__background);
                match(EQUALS);
                AST tmp183_AST = null;
                tmp183_AST = astFactory.create(LT(1));
                astFactory.addASTChild(currentAST, tmp183_AST);
                match(STRING);
                match(SEMI);
                mbg_AST = (AST)currentAST.root;
        }
        catch (RecognitionException ex) {
                reportError(ex);
                consume();
                consumeUntil(_tokenSet_12);
        }
        returnAST = mbg_AST;
}

public final void statement() throws RecognitionException, TokenStreamException {

        returnAST = null;
        ASTPair currentAST = new ASTPair();
        AST statement_AST = null;

        try {      // for error handling
                switch ( LA(1)) {
                case IDENT:
                {
                        assignmentStatement();
                        astFactory.addASTChild(currentAST, returnAST);
                        statement_AST = (AST)currentAST.root;
                        break;
                }
                case LITERAL_if:
                {
                        ifStatement();
                        astFactory.addASTChild(currentAST, returnAST);
```

```
                    statement_AST = (AST)currentAST.root;
                    break;
                }
                case LITERAL_while:
                {
                    whileStatement();
                    astFactory.addASTChild(currentAST, returnAST);
                    statement_AST = (AST)currentAST.root;
                    break;
                }
                case LITERAL_for:
                {
                    forStatement();
                    astFactory.addASTChild(currentAST, returnAST);
                    statement_AST = (AST)currentAST.root;
                    break;
                }
                default:
                {
                    throw new NoViableAltException(LT(1), getFilename());
                }
                }
            }
            catch (RecognitionException ex) {
                reportError(ex);
                consume();
                consumeUntil(_tokenSet_15);
            }
            returnAST = statement_AST;
        }

        public final void assignmentStatement() throws RecognitionException, TokenStreamException {

            returnAST = null;
            ASTPair currentAST = new ASTPair();
            AST assignmentStatement_AST = null;

            try {      // for error handling
                variable();
                astFactory.addASTChild(currentAST, returnAST);
                {
                switch ( LA(1)) {
                case EQUALS:
                {
                    AST tmp185_AST = null;
                    tmp185_AST = astFactory.create(LT(1));
                    astFactory.makeASTRoot(currentAST, tmp185_AST);
                    match(EQUALS);
                    break;
                }
                case PLUS_EQUALS:
                {
                    AST tmp186_AST = null;
                    tmp186_AST = astFactory.create(LT(1));
                    astFactory.makeASTRoot(currentAST, tmp186_AST);
                    match(PLUS_EQUALS);
```

```
                    break;
                }
                case MINUS_EQUALS:
                {
                    AST tmp187_AST = null;
                    tmp187_AST = astFactory.create(LT(1));
                    astFactory.makeASTRoot(currentAST, tmp187_AST);
                    match(MINUS_EQUALS);
                    break;
                }
                case MULT_EQUALS :
                {
                    AST tmp188_AST = null;
                    tmp188_AST = astFactory.create(LT(1));
                    astFactory.makeASTRoot(currentAST, tmp188_AST);
                    match(MULT_EQUALS);
                    break;
                }
                case DIV_EQUALS:
                {
                    AST tmp189_AST = null;
                    tmp189_AST = astFactory.create(LT(1));
                    astFactory.makeASTRoot(currentAST, tmp189_AST);
                    match(DIV_EQUALS);
                    break;
                }
                default:
                {
                    throw new NoViableAltException(LT(1), getFilename());
                }
                }
                }
                expression();
                astFactory.addASTChild(currentAST, returnAST);
                match(SEMI);
                assignmentStatement_AST = (AST)currentAST.root;
            }
            catch (RecognitionException ex) {
                reportError(ex);
                consume();
                consumeUntil(_tokenSet_15);
            }
            returnAST = assignmentStatement_AST;
}

public final void ifStatement() throws RecognitionException, TokenStreamException {

            returnAST = null;
            ASTPair currentAST = new ASTPair();
            AST ifStatement_AST = null;

            try {      // for error handling
                AST tmp191_AST = null;
                tmp191_AST = astFactory.create(LT(1));
                astFactory.makeASTRoot(currentAST, tmp191_AST);
                match(LITERAL_if);
```

```
                    expression();
                    astFactory.addASTChild(currentAST, returnAST);
                    match(LBRACE);
                    codeBlock();
                    astFactory.addASTChild(currentAST, returnAST);
                    match(RBRACE);
                    {
                    switch ( LA(1)) {
                    case LITERAL_else:
                    {
                            match(LITERAL_else);
                            match(LBRACE);
                            codeBlock();
                            astFactory.addASTChild(currentAST, returnAST);
                            match(RBRACE);
                            break;
                    }
                    case RBRACE:
                    case IDENT:
                    case LITERAL_if:
                    case LITERAL_while:
                    case LITERAL_for:
                    {
                            break;
                    }
                    default:
                    {
                            throw new NoViableAltException(LT(1), getFilename());
                    }
                    }
                    }
                    ifStatement_AST = (AST)currentAST.root;
            }
            catch (RecognitionException ex) {
                    reportError(ex);
                    consume();
                    consumeUntil(_tokenSet_15);
            }
            returnAST = ifStatement_AST;
}

public final void whileStatement() throws RecognitionException, TokenStreamException {

            returnAST = null;
            ASTPair currentAST = new ASTPair();
            AST whileStatement_AST = null;

            try {      // for error handling
                    AST tmp197_AST = null;
                    tmp197_AST = astFactory.create(LT(1));
                    astFactory.makeASTRoot(currentAST, tmp197_AST);
                    match(LITERAL_while);
                    expression();
                    astFactory.addASTChild(currentAST, returnAST);
                    match(LBRACE);
                    codeBlock();
```

```
                astFactory.addASTChild(currentAST, returnAST);
                match(RBRACE);
                whileStatement_AST = (AST)currentAST.root;
        }
        catch (RecognitionException ex) {
                reportError(ex);
                consume();
                consumeUntil(_tokenSet_15);
        }
        returnAST = whileStatement_AST;
}

public final void forStatement() throws RecognitionException, TokenStreamException {

        returnAST = null;
        ASTPair currentAST = new ASTPair();
        AST forStatement_AST = null;

        try {      // for error handling
                AST tmp200_AST = null;
                tmp200_AST = astFactory.create(LT(1));
                astFactory.makeASTRoot(currentAST, tmp200_AST);
                match(LITERAL_for);
                forCondition();
                astFactory.addASTChild(currentAST, returnAST);
                match(LBRACE);
                codeBlock();
                astFactory.addASTChild(currentAST, returnAST);
                match(RBRACE);
                forStatement_AST = (AST)currentAST.root;
        }
        catch (RecognitionException ex) {
                reportError(ex);
                consume();
                consumeUntil(_tokenSet_15);
        }
        returnAST = forStatement_AST;
}

public final void variable() throws RecognitionException, TokenStreamException {

        returnAST = null;
        ASTPair currentAST = new ASTPair();
        AST variable_AST = null;

        try {      // for error handling
                AST tmp203_AST = null;
                tmp203_AST = astFactory.create(LT(1));
                astFactory.addASTChild(currentAST, tmp203_AST);
                match(IDENT);
                {
                switch ( LA(1)) {
                case LBRACKET:
                {
                        match(LBRACKET);
                        AST tmp205_AST = null;
```

```
            tmp205_AST = astFactory.create(LT(1));
            astFactory.addASTChild(currentAST, tmp205_AST);
            match(NUMBER);
            match(RBRACKET);
            break;
}
case SEMI:
case EQUALS:
case LBRACE:
case RPAREN:
case NOT_EQUALS:
case LT:
case LTE:
case GT:
case GTE:
case PLUS:
case MINUS:
case MULT:
case DIV:
case DISTANCE:
case MOD:
case PLUS_EQUALS:
case MINUS_EQUALS:
case MULT_EQUALS:
case DIV_EQUALS:
case EQUALS_EQUALS:
case AND:
case OR:
case TYPE_ACCESSOR:
{
            break;
}
default:
{
            throw new NoViableAltException(LT(1), getFilename());
}
}
}
{
_loop120:
do {
            if ((LA(1)==TYPE_ACCESSOR)) {
                        AST tmp207_AST = null;
                        tmp207_AST = astFactory.create(LT(1));
                        astFactory.addASTChild(currentAST, tmp207_AST);
                        match(TYPE_ACCESSOR);
                        AST tmp208_AST = null;
                        tmp208_AST = astFactory.create(LT(1));
                        astFactory.addASTChild(currentAST, tmp208_AST);
                        match(IDENT);
            }
            else {
                        break _loop120;
            }

} while (true);
```

```
                        }
                        variable_AST = (AST)currentAST.root;
                        variable_AST = (AST)astFactory.make( (new
ASTArray(2)).add(astFactory.create(VAR,"VAR")).add(variable_AST));
                        currentAST.root = variable_AST;
                        currentAST.child = variable_AST!=null
&&variable_AST.getFirstChild()!=null ?
                                variable_AST.getFirstChild() : variable_AST;
                        currentAST.advanceChildToEnd();
                        variable_AST = (AST)currentAST.root;
                }
                catch (RecognitionException ex) {
                        reportError(ex);
                        consume();
                        consumeUntil(_tokenSet_16);
                }
                returnAST = variable_AST;
        }

        public final void expression() throws RecognitionException, TokenStreamException {

                returnAST = null;
                ASTPair currentAST = new ASTPair();
                AST expression_AST = null;

                try {      // for error handling
                        logicExpression();
                        astFactory.addASTChild(currentAST, returnAST);
                        {
                        _loop123:
                        do {
                                if ((LA(1)==OR)) {
                                        AST tmp209_AST = null;
                                        tmp209_AST = astFactory.create(LT(1));
                                        astFactory.makeASTRoot(currentAST, tmp209_AST);
                                        match(OR);
                                        logicExpression();
                                        astFactory.addASTChild(currentAST, returnAST);
                                }
                                else {
                                        break _loop123;
                                }

                        } while (true);
                        }
                        expression_AST = (AST)currentAST.root;
                }
                catch (RecognitionException ex) {
                        reportError(ex);
                        consume();
                        consumeUntil(_tokenSet_17);
                }
                returnAST = expression_AST;
        }

        public final void logicExpression() throws RecognitionException, TokenStreamException {
```

```
	returnAST = null;
	ASTPair currentAST = new ASTPair();
	AST logicExpression_AST = null;

	try {	// for error handling
		logicExpression2();
		astFactory.addASTChild(currentAST, returnAST);
		{
		_loop126:
		do {
			if ((LA(1)==AND)) {
				AST tmp210_AST = null;
				tmp210_AST = astFactory.create(LT(1));
				astFactory.makeASTRoot(currentAST, tmp210_AST);
				match(AND);
				logicExpression2();
				astFactory.addASTChild(currentAST, returnAST);
			}
			else {
				break _loop126;
			}

		} while (true);
		}
		logicExpression_AST = (AST)currentAST.root;
	}
	catch (RecognitionException ex) {
		reportError(ex);
		consume();
		consumeUntil(_tokenSet_18);
	}
	returnAST = logicExpression_AST;
}

public final void logicExpression2() throws RecognitionException, TokenStreamException {

	returnAST = null;
	ASTPair currentAST = new ASTPair();
	AST logicExpression2_AST = null;

	try {	// for error handling
		{
		switch ( LA(1)) {
		case NOT:
		{
			AST tmp211_AST = null;
			tmp211_AST = astFactory.create(LT(1));
			astFactory.makeASTRoot(currentAST, tmp211_AST);
			match(NOT);
			break;
		}
		case NUMBER:
		case STRING:
		case LPAREN:
		case IDENT:
```

```
                case LITERAL_true:
                case LITERAL_false:
                {
                        break;
                }
                default:
                {
                        throw new NoViableAltException(LT(1), getFilename());
                }
                }
                }
                relationExpression();
                astFactory.addASTChild(currentAST, returnAST);
                logicExpression2_AST = (AST)currentAST.root;
        }
        catch (RecognitionException ex) {
                reportError(ex);
                consume();
                consumeUntil(_tokenSet_19);
        }
        returnAST = logicExpression2_AST;
}

public final void relationExpression() throws RecognitionException, TokenStreamException {

        returnAST = null;
        ASTPair currentAST = new ASTPair();
        AST relationExpression_AST = null;

        try {      // for error handling
                arithmeticExpression();
                astFactory.addASTChild(currentAST, returnAST);
                {
                switch ( LA(1)) {
                case NOT_EQUALS:
                case LT:
                case LTE:
                case GT:
                case GTE:
                case EQUALS_EQUALS:
                {
                        {
                        switch ( LA(1)) {
                        case GTE:
                        {
                                AST tmp212_AST = null;
                                tmp212_AST = astFactory.create(LT(1));
                                astFactory.makeASTRoot(currentAST, tmp212_AST);
                                match(GTE);
                                break;
                        }
                        case LTE:
                        {
                                AST tmp213_AST = null;
                                tmp213_AST = astFactory.create(LT(1));
                                astFactory.makeASTRoot(currentAST, tmp213_AST);
```

```
                    match(LTE);
                    break;
                }
                case GT:
                {
                    AST tmp214_AST = null;
                    tmp214_AST = astFactory.create(LT(1));
                    astFactory.makeASTRoot(currentAST, tmp214_AST);
                    match(GT);
                    break;
                }
                case LT:
                {
                    AST tmp215_AST = null;
                    tmp215_AST = astFactory.create(LT(1));
                    astFactory.makeASTRoot(currentAST, tmp215_AST);
                    match(LT);
                    break;
                }
                case EQUALS_EQUALS:
                {
                    AST tmp216_AST = null;
                    tmp216_AST = astFactory.create(LT(1));
                    astFactory.makeASTRoot(currentAST, tmp216_AST);
                    match(EQUALS_EQUALS);
                    break;
                }
                case NOT_EQUALS:
                {
                    AST tmp217_AST = null;
                    tmp217_AST = astFactory.create(LT(1));
                    astFactory.makeASTRoot(currentAST, tmp217_AST);
                    match(NOT_EQUALS);
                    break;
                }
                default:
                {
                    throw new NoViableAltException(LT(1), getFilename());
                }
                }
                }
                arithmeticExpression();
                astFactory.addASTChild(currentAST, returnAST);
                break;
            }
            case SEMI:
            case LBRACE:
            case RPAREN:
            case AND:
            case OR:
            {
                break;
            }
            default:
            {
                throw new NoViableAltException(LT(1), getFilename());
```

```
                }
                }
                }
                relationExpression_AST = (AST)currentAST.root;
        }
        catch (RecognitionException ex) {
                reportError(ex);
                consume();
                consumeUntil(_tokenSet_19);
        }
        returnAST = relationExpression_AST;
}

public final void arithmeticExpression() throws RecognitionException, TokenStreamException {

        returnAST = null;
        ASTPair currentAST = new ASTPair();
        AST arithmeticExpression_AST = null;

        try {     // for error handling
                arithmeticExpression2();
                astFactory.addASTChild(currentAST, returnAST);
                {
                _loop135:
                do {
                        if ((LA(1)==PLUS||LA(1)==MINUS)) {
                                {
                                switch ( LA(1)) {
                                case PLUS:
                                {
                                        AST tmp218_AST = null;
                                        tmp218_AST = astFactory.create(LT(1));
                                        astFactory.makeASTRoot(currentAST,
tmp218_AST);
                                        match(PLUS);
                                        break;
                                }
                                case MINUS:
                                {
                                        AST tmp219_AST = null;
                                        tmp219_AST = astFactory.create(LT(1));
                                        astFactory.makeASTRoot(currentAST,
tmp219_AST);
                                        match(MINUS);
                                        break;
                                }
                                default:
                                {
                                        throw new NoViableAltException(LT(1),
getFilename());
                                }
                                }
                                }
                                arithmeticExpression2();
                                astFactory.addASTChild(currentAST, returnAST);
                        }
```

```
                                        else {
                                                break _loop135;
                                        }

                                } while (true);
                                }
                                arithmeticExpression_AST = (AST)currentAST.root;
                        }
                        catch (RecognitionException ex) {
                                reportError(ex);
                                consume();
                                consumeUntil(_tokenSet_20);
                        }
                        returnAST = arithmeticExpression_AST;
                }

                public final void arithmeticExpression2() throws RecognitionException, TokenStreamException {

                        returnAST = null;
                        ASTPair currentAST = new ASTPair();
                        AST arithmeticExpression2_AST = null;

                        try {      // for error handling
                                arithmeticExpression3();
                                astFactory.addASTChild(currentAST, returnAST);
                                {
                                _loop139:
                                do {
                                        if ((LA(1)==MULT||LA(1)==DIV||LA(1)==MOD)) {
                                                {
                                                switch ( LA(1)) {
                                                case MULT:
                                                {
                                                        AST tmp220_AST = null;
                                                        tmp220_AST = astFactory.create(LT(1));
                                                        astFactory.makeASTRoot(currentAST,
tmp220_AST);
                                                        match(MULT);
                                                        break;
                                                }
                                                case DIV:
                                                {
                                                        AST tmp221_AST = null;
                                                        tmp221_AST = astFactory.create(LT(1));
                                                        astFactory.makeASTRoot(currentAST,
tmp221_AST);
                                                        match(DIV);
                                                        break;
                                                }
                                                case MOD:
                                                {
                                                        AST tmp222_AST = null;
                                                        tmp222_AST = astFactory.create(LT(1));
                                                        astFactory.makeASTRoot(currentAST,
tmp222_AST);
                                                        match(MOD);
```

```
                                                break;
                                        }
                                        default:
                                        {
                                                throw new NoViableAltException(LT(1),
getFilename());
                                        }
                                        }
                                        }
                                        arithmeticExpression3();
                                        astFactory.addASTChild(currentAST, returnAST);
                                }
                                else {
                                        break _loop139;
                                }

                        } while (true);
                        }
                        arithmeticExpression2_AST = (AST)currentAST.root;
                }
                catch (RecognitionException ex) {
                        reportError(ex);
                        consume();
                        consumeUntil(_tokenSet_21);
                }
                returnAST = arithmeticExpression2_AST;
        }

        public final void arithmeticExpression3() throws RecognitionException, TokenStreamException {

                returnAST = null;
                ASTPair currentAST = new ASTPair();
                AST arithmeticExpression3_AST = null;

                try {      // for error handling
                        variable_r();
                        astFactory.addASTChild(currentAST, returnAST);
                        {
                        _loop142:
                        do {
                                if ((LA(1)==DISTANCE)) {
                                        AST tmp223_AST = null;
                                        tmp223_AST = astFactory.create(LT(1));
                                        astFactory.makeASTRoot(currentAST, tmp223_AST);
                                        match(DISTANCE);
                                        variable_r();
                                        astFactory.addASTChild(currentAST, returnAST);
                                }
                                else {
                                        break _loop142;
                                }

                        } while (true);
                        }
                        arithmeticExpression3_AST = (AST)currentAST.root;
                }
```

```
		catch (RecognitionException ex) {
			reportError(ex);
			consume();
			consumeUntil(_tokenSet_22);
		}
		returnAST = arithmeticExpression3_AST;
}

public final void variable_r() throws RecognitionException, TokenStreamException {

		returnAST = null;
		ASTPair currentAST = new ASTPair();
		AST variable_r_AST = null;

		try {	// for error handling
			switch ( LA(1)) {
			case IDENT:
			{
				variable();
				astFactory.addASTChild(currentAST, returnAST);
				variable_r_AST = (AST)currentAST.root;
				break;
			}
			case NUMBER:
			{
				AST tmp224_AST = null;
				tmp224_AST = astFactory.create(LT(1));
				astFactory.addASTChild(currentAST, tmp224_AST);
				match(NUMBER);
				variable_r_AST = (AST)currentAST.root;
				break;
			}
			case STRING:
			{
				AST tmp225_AST = null;
				tmp225_AST = astFactory.create(LT(1));
				astFactory.addASTChild(currentAST, tmp225_AST);
				match(STRING);
				variable_r_AST = (AST)currentAST.root;
				break;
			}
			case LITERAL_true:
			{
				AST tmp226_AST = null;
				tmp226_AST = astFactory.create(LT(1));
				astFactory.addASTChild(currentAST, tmp226_AST);
				match(LITERAL_true);
				variable_r_AST = (AST)currentAST.root;
				break;
			}
			case LITERAL_false:
			{
				AST tmp227_AST = null;
				tmp227_AST = astFactory.create(LT(1));
				astFactory.addASTChild(currentAST, tmp227_AST);
				match(LITERAL_false);
```

```
                    variable_r_AST = (AST)currentAST.root;
                    break;
                }
                case LPAREN:
                {
                    match(LPAREN);
                    expression();
                    astFactory.addASTChild(currentAST, returnAST);
                    match(RPAREN);
                    variable_r_AST = (AST)currentAST.root;
                    break;
                }
                default:
                {
                    throw new NoViableAlt Exception(LT(1), getFilename());
                }
                }
        }
        catch (RecognitionException ex) {
                reportError(ex);
                consume();
                consumeUntil(_tokenSet_23);
        }
        returnAST = variable_r_AST;
}

public final void forCondition() throws RecognitionException, TokenStreamException {

        returnAST = null;
        ASTPair currentAST = new ASTPair();
        AST forCondition_AST = null;

        try {      // for error handling
                AST tmp230_AST = null;
                tmp230_AST = astFactory.create(LT(1));
                astFactory.addASTChild(currentAST, tmp 230_AST);
                match(IDENT);
                match(EQUALS);
                AST tmp232_AST = null;
                tmp232_AST = astFactory.create(LT(1));
                astFactory.addASTChild(currentAST, tmp232_AST);
                match(NUMBER);
                match(LITERAL_to);
                AST tmp234_AST = null;
                tmp234_AST = astFactory.create(LT(1));
                astFactory.addASTChild(currentAST, tmp234_AST);
                match(NUMBER);
                {
                switch ( LA(1)) {
                case LITERAL_by:
                {
                    match(LITERAL_by);
                    AST tmp236_AST = null;
                    tmp236_AST = astFactory.create(LT(1));
                    astFactory.addASTChild(currentAST, tmp236_AST);
                    match(NUMBER);
```

```
                                        break;
                                }
                                case LBRACE:
                                {
                                        break;
                                }
                                default:
                                {
                                        throw new NoViableAltException(LT(1), getFilename());
                                }
                                }
                                }
                                forCondition_AST = (AST)currentAST.root;
                                forCondition_AST = (AST)astFactory.make( (new
ASTArray(2)).add(astFactory.create(FORCON,"FORCON")).add(forCondition_AST));
                                currentAST.root = forCondition_AST;
                                currentAST.child = forCondition_AST!=null
&&forCondition_AST.getFirstChild()!=null ?
                                        forCondition_AST.getFirstChild() : forCondition_AST;
                                currentAST.advanceChildToEnd();
                                forCondition_AST = (AST)currentAST.root;
                        }
                        catch (RecognitionException ex) {
                                reportError(ex);
                                consume();
                                consumeUntil(_tokenSet_24);
                        }
                        returnAST = forCondition_AST;
                }


                public static final String[] _tokenNames = {
                        "<0>",
                        "EOF",
                        "<2>",
                        "NULL_TREE_LOOKAHEAD",
                        "COMMENT",
                        "DIGIT",
                        "ALPHA",
                        "NUMBER",
                        "STRING",
                        "DOT",
                        "COLON",
                        "SEMI",
                        "COMMA",
                        "EQUALS",
                        "LBRACE",
                        "RBRACE",
                        "LBRACKET",
                        "RBRACKET",
                        "LPAREN",
                        "RPAREN",
                        "NOT_EQUALS",
                        "LT",
                        "LTE",
                        "GT",
```

```
		"GTE",
		"PLUS",
		"MINUS",
		"MULT",
		"DIV",
		"DISTANCE",
		"MOD",
		"PLUS_EQUALS",
		"MINUS_EQUALS",
		"MULT_EQUALS",
		"DIV_EQUALS",
		"EQUALS_EQUALS",
		"AND",
		"OR",
		"NOT",
		"TYPE_ACCESSOR",
		"WS",
		"IDENT",
		"PROGRAM",
		"ARGS",
		"CODEBLOCK",
		"VAR",
		"FORCON",
		"\"JAWS\"",
		"\"declarations\"",
		"\"int\"",
		"\"double\"",
		"\"string\"",
		"\"boolean\"",
		"\"true\"",
		"\"false\"",
		"\"point\"",
		"\"entity\"",
		"\"_x\"",
		"\"_y\"",
		"\"_image\"",
		"\"_weapon\"",
		"\"_ai\"",
		"\"_label\"",
		"\"movement\"",
		"\"attack\"",
		"\"map\"",
		"\"_width\"",
		"\"_height\"",
		"\"_background\"",
		"\"engine\"",
		"\"if\"",
		"\"else\"",
		"\"while\"",
		"\"for\"",
		"\"to\"",
		"\"by\""
	};

	protected void buildTokenTypeASTClassMap() {
		tokenTypeToASTClassMap=null;
```

```
};

private static final long[] mk_tokenSet_0() {
	long[] data = { 2L, 0L};
	return data;
}
public static final BitSet _tokenSet_0 = new BitSet(mk_tokenSet_0());
private static final long[] mk_tokenSet_1() {
	long[] data = { -9106841396496564224L, 3L, 0L, 0L};
	return data;
}
public static final BitSet _tokenSet_1 = new BitSet(mk_tokenSet_1());
private static final long[] mk_tokenSet_2() {
	long[] data = { 0L, 32L, 0L, 0L};
	return data;
}
public static final BitSet _tokenSet_2 = new BitSet(mk_tokenSet_2());
private static final long[] mk_tokenSet_3() {
	long[] data = { 2199023255552L, 832L, 0L, 0L};
	return data;
}
public static final BitSet _tokenSet_3 = new BitSet(mk_tokenSet_3());
private static final long[] mk_tokenSet_4() {
	long[] data = { -9106841396496531456L, 3L, 0L, 0L};
	return data;
}
public static final BitSet _tokenSet_4 = new BitSet(mk_tokenSet_4());
private static final long[] mk_tokenSet_5() {
	long[] data = { 9079256848778952704L, 24L, 0L, 0L};
	return data;
}
public static final BitSet _tokenSet_5 = new BitSet(mk_tokenSet_5());
private static final long[] mk_tokenSet_6() {
	long[] data = { 9079256848778952704L, 16L, 0L, 0L};
	return data;
}
public static final BitSet _tokenSet_6 = new BitSet(mk_tokenSet_6());
private static final long[] mk_tokenSet_7() {
	long[] data = { 8935141660703096832L, 0L};
	return data;
}
public static final BitSet _tokenSet_7 = new BitSet(mk_tokenSet_7());
private static final long[] mk_tokenSet_8() {
	long[] data = { 8646911284551385088L, 0L};
	return data;
}
public static final BitSet _tokenSet_8 = new BitSet(mk_tokenSet_8());
private static final long[] mk_tokenSet_9() {
	long[] data = { 8070450532247961600L, 0L};
	return data;
}
public static final BitSet _tokenSet_9 = new BitSet(mk_tokenSet_9());
private static final long[] mk_tokenSet_10() {
	long[] data = { 6917529027641114624L, 0L};
	return data;
}
```

```
public static final BitSet _tokenSet_10 = new BitSet(mk_tokenSet_10());
private static final long[] mk_tokenSet_11() {
	long[] data = { 4611686018427420672L, 0L};
	return data;
}
public static final BitSet _tokenSet_11 = new BitSet(mk_tokenSet_11());
private static final long[] mk_tokenSet_12() {
	long[] data = { 32768L, 0L};
	return data;
}
public static final BitSet _tokenSet_12 = new BitSet(mk_tokenSet_12());
private static final long[] mk_tokenSet_13() {
	long[] data = { 524288L, 0L};
	return data;
}
public static final BitSet _tokenSet_13 = new BitSet(mk_tokenSet_13());
private static final long[] mk_tokenSet_14() {
	long[] data = { 528384L, 0L};
	return data;
}
public static final BitSet _tokenSet_14 = new BitSet(mk_tokenSet_14());
private static final long[] mk_tokenSet_15() {
	long[] data = { 2199023288320L, 832L, 0L, 0L};
	return data;
}
public static final BitSet _tokenSet_15 = new BitSet(mk_tokenSet_15());
private static final long[] mk_tokenSet_16() {
	long[] data = { 274877409280L, 0L};
	return data;
}
public static final BitSet _tokenSet_16 = new BitSet(mk_tokenSet_16());
private static final long[] mk_tokenSet_17() {
	long[] data = { 542720L, 0L};
	return data;
}
public static final BitSet _tokenSet_17 = new BitSet(mk_tokenSet_17());
private static final long[] mk_tokenSet_18() {
	long[] data = { 137439496192L, 0L};
	return data;
}
public static final BitSet _tokenSet_18 = new BitSet(mk_tokenSet_18());
private static final long[] mk_tokenSet_19() {
	long[] data = { 206158972928L, 0L};
	return data;
}
public static final BitSet _tokenSet_19 = new BitSet(mk_tokenSet_19());
private static final long[] mk_tokenSet_20() {
	long[] data = { 240551217152L, 0L};
	return data;
}
public static final BitSet _tokenSet_20 = new BitSet(mk_tokenSet_20());
private static final long[] mk_tokenSet_21() {
	long[] data = { 240651880448L, 0L};
	return data;
}
public static final BitSet _tokenSet_21 = new BitSet(mk_tokenSet_21());
```

```
	private static final long[] mk_tokenSet_22() {
		long[] data = { 242128275456L, 0L};
		return data;
	}
	public static final BitSet _tokenSet_22 = new BitSet(mk_tokenSet_22());
	private static final long[] mk_tokenSet_23() {
		long[] data = { 242665146368L, 0L};
		return data;
	}
	public static final BitSet _tokenSet_23 = new BitSet(mk_tokenSet_23());
	private static final long[] mk_tokenSet_24() {
		long[] data = { 16384L, 0L};
		return data;
	}
	public static final BitSet _tokenSet_24 = new BitSet(mk_tokenSet_24());

	}


// $ANTLR 2.7.2: "jaws.g" -> "JAWSParser.java"$

public interface JAWSWalkerTokenTypes {
	int EOF = 1;
	int NULL_TREE_LOOKAHEAD = 3;
	int COMMENT = 4;
	int DIGIT = 5;
	int ALPHA = 6;
	int NUMBER = 7;
	int STRING = 8;
	int DOT = 9;
	int COLON = 10;
	int SEMI = 11;
	int COMMA = 12;
	int EQUALS = 13;
	int LBRACE = 14;
	int RBRACE = 15;
	int LBRACKET = 16;
	int RBRACKET = 17;
	int LPAREN = 18;
	int RPAREN = 19;
	int NOT_EQUALS = 20;
	int LT = 21;
	int LTE = 22;
	int GT = 23;
	int GTE = 24;
	int PLUS = 25;
	int MINUS = 26;
	int MULT = 27;
	int DIV = 28;
	int DISTANCE = 29;
	int MOD = 30;
	int PLUS_EQUALS = 31;
	int MINUS_EQUALS = 32;
	int MULT_EQUALS = 33;
	int DIV_EQUALS = 34;
	int EQUALS_EQUALS = 35;
```

```
	int AND = 36;
	int OR = 37;
	int NOT = 38;
	int TYPE_ACCESSOR = 39;
	int WS = 40;
	int IDENT = 41;
	int PROGRAM = 42;
	int ARGS = 43;
	int CODEBLOCK = 44;
	int VAR = 45;
	int FORCON = 46;
	int LITERAL_JAWS = 47;
	int LITERAL_declarations = 48;
	int LITERAL_int = 49;
	int LITERAL_double = 50;
	int LITERAL_string = 51;
	int LITERAL_boolean = 52;
	int LITERAL_true = 53;
	int LITERAL_false = 54;
	int LITERAL_point = 55;
	int LITERAL_entity = 56;
	int LITERAL__x = 57;
	int LITERAL__y = 58;
	int LITERAL__image = 59;
	int LITERAL__weapon = 60;
	int LITERAL__ai = 61;
	int LITERAL__label = 62;
	int LITERAL_movement = 63;
	int LITERAL_attack = 64;
	int LITERAL_map = 65;
	int LITERAL__width = 66;
	int LITERAL__height = 67;
	int LITERAL__background = 68;
	int LITERAL_engine = 69;
	int LITERAL_if = 70;
	int LITERAL_else = 71;
	int LITERAL_while = 72;
	int LITERAL_for = 73;
	int LITERAL_to = 74;
	int LITERAL_by = 75;
}

// $ANTLR 2.7.2: "jaws.g" -> "JAWSWalker.java"$

import antlr.TreeParser;
import antlr.Token;
import antlr.collections.AST;
import antlr.RecognitionException;
import antlr.ANTLRException;
import antlr.NoViableAltException;
import antlr.MismatchedTokenException;
import antlr.SemanticException;
import antlr.collections.impl.BitSet;
import antlr.ASTPair;
import antlr.collections.impl.ASTArray;
```

```
public class JAWSWalker extends antlr.TreeParser       implements JAWSWalkerTokenTypes
 {


        JAWSOutputter jo = new JAWSOutputter();
        String namep = "";
public JAWSWalker() {
        tokenNames = _tokenNames;
}

        public final void program(AST _t) throws RecognitionException {

                AST program_AST_in = (AST)_t;
                String title;

                try {      // for error handling
                        AST __t154 = _t;
                        AST tmp1_AST_in = (AST)_t;
                        match(_t,PROGRAM);
                        _t = _t.getFirstChild();
                        title=id(_t);
                        _t = _retTree;
                        declarBlock(_t);
                        _t = _retTree;
                        engineBlock(_t);
                        _t = _retTree;
                        _t = __t154;
                        _t = _t.getNextSibling();
                        jo.setName(title);jo.outputFile(title+".java");
                }
                catch (RecognitionException ex) {
                        reportError(ex);
                        if (_t!=null) {_t = _t.getNextSibling();}
                }
                _retTree = _t;
        }

        public final String  id(AST _t) throws RecognitionException {
                String r;

                AST id_AST_in = (AST)_t;
                r = "";

                try {      // for error handling
                        AST tmp2_AST_in = (AST)_t;
                        match(_t,IDENT);
                        _t = _t.getNextSibling();
                        r = tmp2_AST_in.getText();
                }
                catch (RecognitionException ex) {
                        reportError(ex);
                        if (_t!=null) {_t = _t.getNextSibling();}
                }
                _retTree = _t;
                return r;
```

```
}

public final void declarBlock(AST _t) throws RecognitionException {

	AST declarBlock_AST_in = (AST)_t;

	try {	// for error handling
		AST __t156 = _t;
		AST tmp3_AST_in = (AST)_t;
		match(_t,LITERAL_declarations);
		_t = _t.getFirstChild();
		{
		_loop158:
		do {
			if (_t==null) _t=ASTNULL;
			if ((_tokenSet_0.member(_t.getType()))) {
				declarations(_t);
				_t = _retTree;
			}
			else {
				break _loop158;
			}

		} while (true);
		}
		_t = __t156;
		_t = _t.getNextSibling();
	}
	catch (RecognitionException ex) {
		reportError(ex);
		if (_t!=null) {_t = _t.getNextSibling();}
	}
	_retTree = _t;
}

public final void engineBlock(AST _t) throws RecognitionException {

	AST engineBlock_AST_in = (AST)_t;
	AST pred = null;
	String st="";

	try {	// for error handling
		if (_t==null) _t=ASTNULL;
		switch ( _t.getType()) {
		case LITERAL_engine:
		{
			AST __t169 = _t;
			AST tmp4_AST_in = (AST)_t;
			match(_t,LITERAL_engine);
			_t = _t.getFirstChild();
			{
			if (_t==null) _t=ASTNULL;
			switch ( _t.getType()) {
			case EQUALS:
			case LITERAL_if:
			case LITERAL_while:
```

```
                    case LITERAL_for:
                    {
                            pred = _t==ASTNULL ? null : (AST)_t;
                            st=statement(_t);
                            _t = _retTree;
                            break;
                    }
                    case 3:
                    {
                            break;
                    }
                    default:
                    {
                            throw new NoViableAltException(_t);
                    }
                    }
                    }
                    if (!st.equals("")) {
                            jo.addEngineStatement(st);


                            AST next = pred.getNextSibling();
                            while (next != null)
                            {
                                    st = statement(next); jo.addEngineStatement(st);
                                    next = next.getNextSibling();
                            }
                            }


                    _t = __t169;
                    _t = _t.getNextSibling();
                    break;
            }
            case LITERAL_int:
            {
                    AST __t171 = _t;
                    AST tmp5_AST_in = (AST)_t;
                    match(_t,LITERAL_int);
                    _t = _t.getFirstChild();
                    AST tmp6_AST_in = (AST)_t;
                    match(_t,IDENT);
                    _t = _t.getNextSibling();
                    AST tmp7_AST_in = (AST)_t;
                    match(_t,NUMBER);
                    _t = _t.getNextSibling();

                                    double tmp =
Double.parseDouble(tmp7_AST_in.getText());
                                    JAWSInt a = new JAWSInt( (int)tmp );
                                    a.setName(tmp6_AST_in.getText());
                                    jo.addInt(a);

                    _t = __t171;
                    _t = _t.getNextSibling();
                    break;
```

```
                        }
                        case LITERAL_double:
                        {
                                AST __t172 = _t;
                                AST tmp8_AST_in = (AST)_t;
                                match(_t,LITERAL_double);
                                _t = _t.getFirstChild();
                                AST tmp9_AST_in = (AST)_t;
                                match(_t,IDENT);
                                _t = _t.getNextSibling();
                                AST tmp10_AST_in = (AST)_t;
                                match(_t,NUMBER);
                                _t = _t.getNextSibling();

                                                double tmp =
Double.parseDouble(tmp10_AST_in.getText());
                                                JAWSDouble a = new JAWSDouble( tmp );
                                                a.setName(tmp9_AST_in.getText());
                                                jo.addDouble(a);


                                _t = __t172;
                                _t = _t.getNextSibling();
                                break;
                        }
                        case LITERAL_string:
                        {
                                AST __t173 = _t;
                                AST tmp11_AST_in = (AST)_t;
                                match(_t,LITERAL_string);
                                _t = _t.getFirstChild();
                                AST tmp12_AST_in = (AST)_t;
                                match(_t,IDENT);
                                _t = _t.getNextSibling();
                                AST tmp13_AST_in = (AST)_t;
                                match(_t,STRING);
                                _t = _t.getNextSibling();

                                                String tmp = tmp13_AST_in.getText();
                                                JAWSString a = new JAWSString( tmp );
                                                a.setName(tmp12_AST_in.getText());
                                                jo.addString(a);


                                _t = __t173;
                                _t = _t.getNextSibling();
                                break;
                        }
                        case LITERAL_boolean:
                        {
                                AST __t174 = _t;
                                AST tmp14_AST_in = (AST)_t;
                                match(_t,LITERAL_boolean);
                                _t = _t.getFirstChild();
                                boolean tmp;
                                AST tmp15_AST_in = (AST)_t;
```

```
                                match(_t,IDENT);
                                _t = _t.getNextSibling();
                                {
                                if (_t==null) _t=ASTNULL;
                                switch ( _t.getType()) {
                                case LITERAL_false:
                                {
                                        AST tmp16_AST_in = (AST)_t;
                                        match(_t,LITERAL_false);
                                        _t = _t.getNextSibling();
                                        tmp = false;
                                        break;
                                }
                                case LITERAL_true:
                                {
                                        AST tmp17_AST_in = (AST)_t;
                                        match(_t,LITERAL_true);
                                        _t = _t.getNextSibling();
                                        tmp = true;
                                        break;
                                }
                                default:
                                {
                                        throw new NoViableAltException(_t);
                                }
                                }
                                }


                                                JAWSBoolean a = new JAWSBoolean(tmp);
                                                a.setName(tmp15_AST_in.getText());
                                                jo.addBoolean(a);

                                _t = __t174;
                                _t = _t.getNextSibling();
                                break;
                        }
                        default:
                        {
                                throw new NoViableAltException(_t);
                        }
                        }
                }
                catch (RecognitionException ex) {
                        reportError(ex);
                        if (_t!=null) {_t = _t.getNextSibling();}
                }
                _retTree = _t;
}

public final void declarations(AST _t) throws RecognitionException {

                AST declarations_AST_in = (AST)_t;
                AST pred = null;
                AST pred2 = null;
                AST pred3 = null;
```

```
String s; String c;

try {      // for error handling
    if (_t==null) _t=ASTNULL;
    switch ( _t.getType()) {
    case LITERAL_int:
    {
        AST __t160 = _t;
        AST tmp18_AST_in = (AST)_t;
        match(_t,LITERAL_int);
        _t = _t.getFirstChild();
        AST tmp19_AST_in = (AST)_t;
        match(_t,IDENT);
        _t = _t.getNextSibling();
        AST tmp20_AST_in = (AST)_t;
        match(_t,NUMBER);
        _t = _t.getNextSibling();

                                    double tmp =
Double.parseDouble(tmp20_AST_in.getText());
                                    JAWSInt a = new JAWSInt( (int)tmp );
                                    a.setName(tmp19_AST_in.getText());
                                    jo.addInt(a);

        _t = __t160;
        _t = _t.getNextSibling();
        break;
    }
    case LITERAL_double:
    {
        AST __t161 = _t;
        AST tmp21_AST_in = (AST)_t;
        match(_t,LITERAL_double);
        _t = _t.getFirstChild();
        AST tmp22_AST_in = (AST)_t;
        match(_t,IDENT);
        _t = _t.getNextSibling();
        AST tmp23_AST_in = (AST)_t;
        match(_t,NUMBER);
        _t = _t.getNextSibling();

                                    double tmp =
Double.parseDouble(tmp23_AST_in.getText());
                                    JAWSDouble a = new JAWSDouble( tmp );
                                    a.setName(tmp22_AST_in.getText());
                                    jo.addDouble(a);


        _t = __t161;
        _t = _t.getNextSibling();
        break;
    }
    case LITERAL_string:
    {
        AST __t162 = _t;
        AST tmp24_AST_in = (AST)_t;
```

```
		match(_t,LITERAL_string);
		_t = _t.getFirstChild();
		AST tmp25_AST_in = (AST)_t;
		match(_t,IDENT);
		_t = _t.getNextSibling();
		AST tmp26_AST_in = (AST)_t;
		match(_t,STRING);
		_t = _t.getNextSibling();

				String tmp = tmp26_AST_in.getText();
				JAWSString a = new JAWSString( tmp );
				a.setName(tmp25_AST_in.getText());
				jo.addString(a);


		_t = __t162;
		_t = _t.getNextSibling();
		break;
	}
	case LITERAL_boolean:
	{
		AST __t163 = _t;
		AST tmp27_AST_in = (AST)_t;
		match(_t,LITERAL_boolean);
		_t = _t.getFirstChild();
		boolean tmp;
		AST tmp28_AST_in = (AST)_t;
		match(_t,IDENT);
		_t = _t.getNextSibling();
		{
		if (_t==null) _t=ASTNULL;
		switch ( _t.getType()) {
		case LITERAL_false:
		{
			AST tmp29_AST_in = (AST)_t;
			match(_t,LITERAL_false);
			_t = _t.getNextSibling();
			tmp = false;
			break;
		}
		case LITERAL_true:
		{
			AST tmp30_AST_in = (AST)_t;
			match(_t,LITERAL_true);
			_t = _t.getNextSibling();
			tmp = true;
			break;
		}
		default:
		{
			throw new NoViableAltException(_t);
		}
		}
		}
```

```
                                JAWSBoolean a = new JAWSBoolean(tmp);
                                a.setName(tmp28_AST_in.getText());
                                jo.addBoolean(a);

            _t = __t163;
            _t = _t.getNextSibling();
            break;
}
case LITERAL_point:
{
            AST __t165 = _t;
            AST tmp31_AST_in = (AST)_t;
            match(_t,LITERAL_point);
            _t = _t.getFirstChild();
            pred = _t==ASTNULL ? null : (AST)_t;
            s=id(_t);
            _t = _retTree;

                                JAWSpoint p;
                                AST next = pred.getNextSibling();
                                if (next == null)
                                   p = new JAWSpoint(0,0,s);
                                else
                                {
                                   int a = Integer.parseInt(next.getText());
                                   next = next.getNextSibling();
                                   int b = Integer.parseInt(next.getText());
                                   p = new JAWSpoint(a,b,s);
                                }
                                jo.addPoint(p);


            _t = __t165;
            _t = _t.getNextSibling();
            break;
}
case LITERAL_map:
{
            AST __t166 = _t;
            AST tmp32_AST_in = (AST)_t;
            match(_t,LITERAL_map);
            _t = _t.getFirstChild();
            pred2 = _t==ASTNULL ? null : (AST)_t;
            s=id(_t);
            _t = _retTree;

                                Map m = new Map();
                                m.name=s;
                                AST next = pred2.getNextSibling();
                                while (next != null)
                                {
                                   if ( (next.getText()).equals("_width") )
                                   {

                                         AST child = next.getFirstChild();
                                        m.width = Integer.parseInt( child.getText() );
```

```
                                                    }
                                                    else if ( (next.getText()).equals("_height") )
                                                    {

                                                            AST child = next.getFirstChild();
                                                            m.height =
Integer.parseInt( child.getText() );
                                                    }
                                                    else if ( (next.getText()).equals("_background") )
                                                    {

                                                            AST child = next.getFirstChild();
                                                            m.file = new java.io.File( child.getText() );
                                                    }
                                                    next = next.getNextSibling();
                                                }
                                                jo.addMap(m);

                                        _t = __t166;
                                        _t = _t.getNextSibling();
                                        break;
                                }
                                case LITERAL_entity:
                                {
                                        AST __t167 = _t;
                                        AST tmp33_AST_in = (AST)_t;
                                        match(_t,LITERAL_entity);
                                        _t = _t.getFirstChild();
                                        s=id(_t);
                                        _t = _retTree;
                                        pred3 = _t==ASTNULL ? null : (AST)_t;
                                        c=num(_t);
                                        _t = _retTree;

                                                JAWSEntity je = new JAWSEntity();
                                                je.name = s;
                                                je.mult = Integer.parseInt(c);
                                                AST next = pred3.getNextSibling();
                                                while(next!=null)
                                                {
                                                    if ( (next.getText()).equals("_width") )
                                                    {

                                                            AST child = next.getFirstChild();
                                                           je.width = Integer.parseInt( child.getText() );
                                                    }
                                                    else if ( (next.getText()).equals("_height") )
                                                    {

                                                            AST child = next.getFirstChild();
                                                           je.height = Integer.parseInt( child.getText() );
                                                    }
                                                    else if ( (next.getText()).equals("_x") )
                                                    {

                                                            AST child = next.getFirstChild();
```

```
                                                    je.x = Integer.parseInt( child.getText() );
                                                }
                                                else if ( (next.getText()).equals("_y") )
                                                {

                                                    AST child = next.getFirstChild();
                                                    je.y = Integer.parseInt( child.getText() );
                                                }
                                                else if ( (next.getText()).equals("_image") )
                                                {

                                                    AST child = next.getFirstChild();
                                                    je.image = child.getText() ;
                                                }
                                                else if ( (next.getText()).equals("_weapon") )
                                                {

                                                    AST child = next.getFirstChild();
                                                    je.weapon = child.getText() ;
                                                }
                                                else if ( (next.getText()).equals("_ai") )
                                                {

                                                    AST child = next.getFirstChild();
                                                    je.ai = child.getText() ;
                                                }
                                                else if ( (next.getText()).equals("_label") )
                                                {

                                                    AST child = next.getFirstChild();
                                                    je.label = child.getText() ;
                                                }
                                                next = next.getNextSibling();
                                            }
                                            jo.addEntity(je);

                                    _t = __t167;
                                    _t = _t.getNextSibling();
                                    break;
                                }
                                default:
                                {
                                    throw new NoViableAltException(_t);
                                }
                                }
                        }
                        catch (RecognitionException ex) {
                                reportError(ex);
                                if (_t!=null) {_t = _t.getNextSibling();}
                        }
                        _retTree = _t;
}

public final String  num(AST _t) throws RecognitionException {
                String r;
```

```
		AST num_AST_in = (AST)_t;
		r = "";

		try {	// for error handling
			AST tmp34_AST_in = (AST)_t;
			match(_t,NUMBER);
			_t = _t.getNextSibling();
			r = tmp34_AST_in.getText();
		}
		catch (RecognitionException ex) {
			reportError(ex);
			if (_t!=null) {_t = _t.getNextSibling();}
		}
		_retTree = _t;
		return r;
}

public final String  statement(AST _t) throws RecognitionException {
		String r;

		AST statement_AST_in = (AST)_t;
		AST pred = null;
		String v; String e; String cb; r= "";

		try {	// for error handling
			if (_t==null) _t=ASTNULL;
			switch ( _t.getType()) {
			case EQUALS:
			{
				AST __t177 = _t;
				AST tmp35_AST_in = (AST)_t;
				match(_t,EQUALS);
				_t = _t.getFirstChild();
				v=variable(_t);
				_t = _retTree;
				e=expression(_t);
				_t = _retTree;
				r = v + " = " + e + ";";
				_t = __t177;
				_t = _t.getNextSibling();
				break;
			}
			case LITERAL_if:
			{
				AST __t178 = _t;
				AST tmp36_AST_in = (AST)_t;
				match(_t,LITERAL_if);
				_t = _t.getFirstChild();
				e=expression(_t);
				_t = _retTree;
				pred = _t==ASTNULL ? null : (AST)_t;
				cb=codeBlock(_t);
				_t = _retTree;

					r = "if "+e+"\r\n{\r\n"+cb+"}\r\n";
					AST next = pred.getNextSibling();
```

```
                                        if (next != null)
                                        {
                                                r+= "else {\r\n"+codeBlock(next)+"}\r\n";
                                        }

                                _t = __t178;
                                _t = _t.getNextSibling();
                                break;
                        }
                        case LITERAL_while:
                        {
                                AST __t179 = _t;
                                AST tmp37_AST_in = (AST)_t;
                                match(_t,LITERAL_while);
                                _t = _t.getFirstChild();
                                e=expression(_t);
                                _t = _retTree;
                                cb=codeBlock(_t);
                                _t = _retTree;
                                _t = __t179;
                                _t = _t.getNextSibling();

                                        r = "while" + e + "\r\n{\r\n"+cb+"}\r\n";

                                break;
                        }
                        case LITERAL_for:
                        {
                                AST __t180 = _t;
                                AST tmp38_AST_in = (AST)_t;
                                match(_t,LITERAL_for);
                                _t = _t.getFirstChild();
                                e=forCondition(_t);
                                _t = _retTree;
                                cb=codeBlock(_t);
                                _t = _retTree;
                                _t = __t180;
                                _t = _t.getNextSibling();

                                        r = e + "\r\n{\r\n"+cb+"}\r\n";

                                break;
                        }
                        default:
                        {
                                throw new NoViableAltException(_t);
                        }
                        }
                }
                catch (RecognitionException ex) {
                        reportError(ex);
                        if (_t!=null) {_t = _t.getNextSibling();}
                }
                _retTree = _t;
                return r;
        }
```

```
public final String  variable(AST _t) throws RecognitionException {
	String r;

	AST variable_AST_in = (AST)_t;
	AST pred = null;
	r = ""; String s; String a;

	try {      // for error handling
		 AST __t186 = _t;
		 AST tmp39_AST_in = (AST)_t;
		 match(_t,VAR);
		 _t = _t.getFirstChild();
		 pred = _t==ASTNULL ? null : (AST)_t;
		 s=id(_t);
		 _t = _retTree;
		 _t = __t186;
		 _t = _t.getNextSibling();

				r = s;

				AST next = pred.getNextSibling();

				if (next != null)
				{
				   if ( !(next.getText()).equals("'s") )
					{
					   r += "["+next.getText()+"]";

					   next = next.getNextSibling();
					   //System.out.println(r);
					}

					while ( (next != null) && (next.getText()).equals("'s") )
					{
					   r += ".";
					   next = next.getNextSibling();
					   r += next.getText();
					   next = next.getNextSib ling();
					}
				}

	}
	catch (RecognitionException ex) {
		reportError(ex);
		if (_t!=null) {_t = _t.getNextSibling();}
	}
	_retTree = _t;
	return r;
}

public final String  expression(AST _t) throws RecognitionException {
	String r;

	AST expression_AST_in = (AST)_t;
	AST pred = null;
```

```
r = ""; String a; String b;

try {      // for error handling
            if (_t==null) _t=ASTNULL;
            switch ( _t.getType()) {
            case IDENT:
            {
                        AST tmp40_AST_in = (AST)_t;
                        match(_t,IDENT);
                        _t = _t.getNextSibling();
                        r = tmp40_AST_in.getText();
                        break;
            }
            case NUMBER:
            {
                        AST tmp41_AST_in = (AST)_t;
                        match(_t,NUMBER);
                        _t = _t.getNextSibling();
                        r = tmp41_AST_in.getText();
                        break;
            }
            case STRING:
            {
                        AST tmp42_AST_in = (AST)_t;
                        match(_t,STRING);
                        _t = _t.getNextSibling();
                        r = "\""+tmp42_AST_in.getText()+"\"";
                        break;
            }
            case LITERAL_true:
            {
                        AST tmp43_AST_in = (AST)_t;
                        match(_t,LITERAL_true);
                        _t = _t.getNextSibling();
                        r = "true";
                        break;
            }
            case LITERAL_false:
            {
                        AST tmp44_AST_in = (AST)_t;
                        match(_t,LITERAL_false);
                        _t = _t.getNextSibling();
                        r = "false";
                        break;
            }
            case VAR:
            {
                        AST __t188 = _t;
                        AST tmp45_AST_in = (AST)_t;
                        match(_t,VAR);
                        _t = _t.getFirstChild();
                        pred = _t==ASTNULL ? null : (AST)_t;
                        a=expression(_t);
                        _t = _retTree;

                                    r = a;
```

```
                        AST next = pred.getNextSibling();
                        if (next != null)
                           r += "["+ next.getText() +"]" ;

                _t = __t188;
                _t = _t.getNextSibling();
                break;
}
case DISTANCE:
{
                AST __t189 = _t;
                AST tmp46_AST_in = (AST)_t;
                match(_t,DISTANCE);
                _t = _t.getFirstChild();
                a=expression(_t);
                _t = _retTree;
                b=expression(_t);
                _t = _retTree;

                        r = "( JAWSpoint.distance("+a+","+b+") )";

                _t = __t189;
                _t = _t.getNextSibling();
                break;
}
case MULT:
{
                AST __t190 = _t;
                AST tmp47_AST_in = (AST)_t;
                match(_t,MULT);
                _t = _t.getFirstChild();
                a=expression(_t);
                _t = _retTree;
                b=expression(_t);
                _t = _retTree;

                        r = "("+a + " * " + b+")";

                _t = __t190;
                _t = _t.getNextSibling();
                break;
}
case DIV:
{
                AST __t191 = _t;
                AST tmp48_AST_in = (AST)_t;
                match(_t,DIV);
                _t = _t.getFirstChild();
                a=expression(_t);
                _t = _retTree;
                b=expression(_t);
                _t = _retTree;

                        r = "("+a + " / " + b+")";

                _t = __t191;
```

```
		_t = _t.getNextSibling();
		break;
	}
	case MOD:
	{
		AST __t192 = _t;
		AST tmp49_AST_in = (AST)_t;
		match(_t,MOD);
		_t = _t.getFirstChild();
		a=expression(_t);
		_t = _retTree;
		b=expression(_t);
		_t = _retTree;

				r = "("+a + " % " + b+")";

		_t = __t192;
		_t = _t.getNextSibling();
		break;
	}
	case PLUS:
	{
		AST __t193 = _t;
		AST tmp50_AST_in = (AST)_t;
		match(_t,PLUS);
		_t = _t.getFirstChild();
		a=expression(_t);
		_t = _retTree;
		b=expression(_t);
		_t = _retTree;

				r = "("+a + " + " + b+")";

		_t = __t193;
		_t = _t.getNextSibling();
		break;
	}
	case MINUS:
	{
		AST __t194 = _t;
		AST tmp51_AST_in = (AST)_t;
		match(_t,MINUS);
		_t = _t.getFirstChild();
		a=expression(_t);
		_t = _retTree;
		b=expression(_t);
		_t = _retTree;

				r = "("+a + " - " + b+")";

		_t = __t194;
		_t = _t.getNextSibling();
		break;
	}
	case GTE:
	{
```

```
		AST __t195 = _t;
		AST tmp52_AST_in = (AST)_t;
		match(_t,GTE);
		_t = _t.getFirstChild();
		a=expression(_t);
		_t = _retTree;
		b=expression(_t);
		_t = _retTree;

				r = "("+a + " >= " + b+")";

		_t = __t195;
		_t = _t.getNextSibling();
		break;
	}
	case LTE:
	{
		AST __t196 = _t;
		AST tmp53_AST_in = (AST)_t;
		match(_t,LTE);
		_t = _t.getFirstChild();
		a=expression(_t);
		_t = _retTree;
		b=expression(_t);
		_t = _retTree;

				r = "("+a + " <= " + b+")";

		_t = __t196;
		_t = _t.getNextSibling();
		break;
	}
	case GT:
	{
		AST __t197 = _t;
		AST tmp54_AST_in = (AST)_t;
		match(_t,GT);
		_t = _t.getFirstChild();
		a=expression(_t);
		_t = _retTree;
		b=expression(_t);
		_t = _retTree;

				r = "("+a + " > " + b+")";

		_t = __t197;
		_t = _t.getNextSibling();
		break;
	}
	case LT:
	{
		AST __t198 = _t;
		AST tmp55_AST_in = (AST)_t;
		match(_t,LT);
		_t = _t.getFirstChild();
		a=expression(_t);
```

```
		_t = _retTree;
		b=expression(_t);
		_t = _retTree;

				r = "("+a + " < " + b+")";

		_t = __t198;
		_t = _t.getNextSibling();
		break;
	}
	case EQUALS_EQUALS:
	{
		AST __t199 = _t;
		AST tmp56_AST_in = (AST)_t;
		match(_t,EQUALS_EQUALS);
		_t = _t.getFirstChild();
		a=expression(_t);
		_t = _retTree;
		b=expression(_t);
		_t = _retTree;

				r = "("+a + " == " + b+")";

		_t = __t199;
		_t = _t.getNextSibling();
		break;
	}
	case NOT_EQUALS:
	{
		AST __t200 = _t;
		AST tmp57_AST_in = (AST)_t;
		match(_t,NOT_EQUALS);
		_t = _t.getFirstChild();
		a=expression(_t);
		_t = _retTree;
		b=expression(_t);
		_t = _retTree;

				r = "("+a + " != " + b+")";

		_t = __t200;
		_t = _t.getNextSibling();
		break;
	}
	case NOT:
	{
		AST __t201 = _t;
		AST tmp58_AST_in = (AST)_t;
		match(_t,NOT);
		_t = _t.getFirstChild();
		a=expression(_t);
		_t = _retTree;

				r = "(!"+a+ ")";

		_t = __t201;
```

```
                    _t = _t.getNextSibling();
                    break;
                }
                case AND:
                {
                    AST __t202 = _t;
                    AST tmp59_AST_in = (AST)_t;
                    match(_t,AND);
                    _t = _t.getFirstChild();
                    a=expression(_t);
                    _t = _retTree;
                    b=expression(_t);
                    _t = _retTree;

                            r = "("+a + " && " + b+")";

                    _t = __t202;
                    _t = _t.getNextSibling();
                    break;
                }
                case OR:
                {
                    AST __t203 = _t;
                    AST tmp60_AST_in = (AST)_t;
                    match(_t,OR);
                    _t = _t.getFirstChild();
                    a=expression(_t);
                    _t = _retTree;
                    b=expression(_t);
                    _t = _retTree;

                            r = "("+a + " || " + b+")";

                    _t = __t203;
                    _t = _t.getNextSibling();
                    break;
                }
                default:
                {
                    throw new NoViableAltException(_t);
                }
                }
            }
            catch (RecognitionException ex) {
                reportError(ex);
                if (_t!=null) {_t = _t.getNextSibling();}
            }
            _retTree = _t;
            return r;
}

public final String  codeBlock(AST _t) throws RecognitionException {
            String r;

            AST codeBlock_AST_in = (AST)_t;
            AST pred = null;
```

```
		r = ""; String s;

		try {	// for error handling
			AST __t205 = _t;
			AST tmp61_AST_in = (AST)_t;
			match(_t,CODEBLOCK);
			_t = _t.getFirstChild();
			pred = _t==ASTNULL ? null : (AST)_t;
			s=statement(_t);
			_t = _retTree;
			_t = __t205;
			_t = _t.getNextSibling();

					r += ( s + "\r\n" );
					AST next = pred.getNextSibling();
					while (next != null)
					{
						s = statement(next);
					   r += ( s + "\r\n" );
					   next = next.getNextSibling();

					}

		}
		catch (RecognitionException ex) {
			reportError(ex);
			if (_t!=null) {_t = _t.getNextSibling();}
		}
		_retTree = _t;
		return r;
}

public final String  forCondition(AST _t) throws RecognitionException {
		String r;

		AST forCondition_AST_in = (AST)_t;
		AST pred = null;
		r = ""; String a; String b; String c;

		try {	// for error handling
			AST __t182 = _t;
			AST tmp62_AST_in = (AST)_t;
			match(_t,FORCON);
			_t = _t.getFirstChild();
			a=id(_t);
			_t = _retTree;
			b=num(_t);
			_t = _retTree;
			pred = _t==ASTNULL ? null : (AST)_t;
			c=num(_t);
			_t = _retTree;
			_t = __t182;
			_t = _t.getNextSibling();

				String inc = "1";
				r = "for (int "+a+"="+b+";"+a+"<="+c+";"+a+"+=";
```

```
                                AST next = pred.getNextSibling();
                                if (next != null)
                                {
                                        inc = num(next);
                                }
                                r += inc+")";

        }
        catch (RecognitionException ex) {
                reportError(ex);
                if (_t!=null) {_t = _t.getNextSibling();}
        }
        _retTree = _t;
        return r;
}


public static final String[] _tokenNames = {
        "<0>",
        "EOF",
        "<2>",
        "NULL_TREE_LOOKAHEAD",
        "COMMENT",
        "DIGIT",
        "ALPHA",
        "NUMBER",
        "STRING",
        "DOT",
        "COLON",
        "SEMI",
        "COMMA",
        "EQUALS",
        "LBRACE",
        "RBRACE",
        "LBRACKET",
        "RBRACKET",
        "LPAREN",
        "RPAREN",
        "NOT_EQUALS",
        "LT",
        "LTE",
        "GT",
        "GTE",
        "PLUS",
        "MINUS",
        "MULT",
        "DIV",
        "DISTANCE",
        "MOD",
        "PLUS_EQUALS",
        "MINUS_EQUALS",
        "MULT_EQUALS",
        "DIV_EQUALS",
        "EQUALS_EQUALS",
        "AND",
        "OR",
```

```
		"NOT",
		"TYPE_ACCESSOR",
		"WS",
		"IDENT",
		"PROGRAM",
		"ARGS",
		"CODEBLOCK",
		"VAR",
		"FORCON",
		"\"JAWS\"",
		"\"declarations\"",
		"\"int\"",
		"\"double\"",
		"\"string\"",
		"\"boolean\"",
		"\"true\"",
		"\"false\"",
		"\"point\"",
		"\"entity\"",
		"\"_x\"",
		"\"_y\"",
		"\"_image\"",
		"\"_weapon\"",
		"\"_ai\"",
		"\"_label\"",
		"\"movement\"",
		"\"attack\"",
		"\"map\"",
		"\"_width\"",
		"\"_height\"",
		"\"_background\"",
		"\"engine\"",
		"\"if\"",
		"\"else\"",
		"\"while\"",
		"\"for\"",
		"\"to\"",
		"\"by\""
	};

	private static final long[] mk_tokenSet_0() {
		long[] data = { 116530640358211584L, 2L, 0L, 0L};
		return data;
	}
	public static final BitSet _tokenSet_0 = new BitSet(mk_tokenSet_0());
	}


// $ANTLR 2.7.2: "jaws.g" -> "JAWSParser.java"$

public interface JAWSWalkerTokenTypes {
	int EOF = 1;
	int NULL_TREE_LOOKAHEAD = 3;
	int COMMENT = 4;
	int DIGIT = 5;
	int ALPHA = 6;
```

```
int NUMBER = 7;
int STRING = 8;
int DOT = 9;
int COLON = 10;
int SEMI = 11;
int COMMA = 12;
int EQUALS = 13;
int LBRACE = 14;
int RBRACE = 15;
int LBRACKET = 16;
int RBRACKET = 17;
int LPAREN = 18;
int RPAREN = 19;
int NOT_EQUALS = 20;
int LT = 21;
int LTE = 22;
int GT = 23;
int GTE = 24;
int PLUS = 25;
int MINUS = 26;
int MULT = 27;
int DIV = 28;
int DISTANCE = 29;
int MOD = 30;
int PLUS_EQUALS = 31;
int MINUS_EQUALS = 32;
int MULT_EQUALS = 33;
int DIV_EQUALS = 34;
int EQUALS_EQUALS = 35;
int AND = 36;
int OR = 37;
int NOT = 38;
int TYPE_ACCESSOR = 39;
int WS = 40;
int IDENT = 41;
int PROGRAM = 42;
int ARGS = 43;
int CODEBLOCK = 44;
int VAR = 45;
int FORCON = 46;
int LITERAL_JAWS = 47;
int LITERAL_declarations = 48;
int LITERAL_int = 49;
int LITERAL_double = 50;
int LITERAL_string = 51;
int LITERAL_boolean = 52;
int LITERAL_true = 53;
int LITERAL_false = 54;
int LITERAL_point = 55;
int LITERAL_entity = 56;
int LITERAL__x = 57;
int LITERAL__y = 58;
int LITERAL__image = 59;
int LITERAL__weapon = 60;
int LITERAL__ai = 61;
int LITERAL__label = 62;
```

```
	int LITERAL_movement = 63;
	int LITERAL_attack = 64;
	int LITERAL_map = 65;
	int LITERAL__width = 66;
	int LITERAL__height = 67;
	int LITERAL__background = 68;
	int LITERAL_engine = 69;
	int LITERAL_if = 70;
	int LITERAL_else = 71;
	int LITERAL_while = 72;
	int LITERAL_for = 73;
	int LITERAL_to = 74;
	int LITERAL_by = 75;
}


// $ANTLR 2.7.2: "jaws.g" -> "JAWSLexer.java"$

import java.io.InputStream;
import antlr.TokenStreamException;
import antlr.TokenStreamIOException;
import antlr.TokenStreamRecognitionException;
import antlr.CharStreamException;
import antlr.CharStreamIOException;
import antlr.ANTLRException;
import java.io.Reader;
import java.util.Hashtable;
import antlr.CharScanner;
import antlr.InputBuffer;
import antlr.ByteBuffer;
import antlr.CharBuffer;
import antlr.Token;
import antlr.CommonToken;
import antlr.RecognitionException;
import antlr.NoViableAltForCharException;
import antlr.MismatchedCharException;
import antlr.TokenStream;
import antlr.ANTLRHashString;
import antlr.LexerSharedInputState;
import antlr.collections.impl.BitSet;
import antlr.SemanticException;

public class JAWSLexer extends antlr.CharScanner implements JAWSTokenTypes, TokenStream
 {

	int nr_error = 0;
	public void reportError( String s ){
		super.reportError( s );
		nr_error++;
	}
	public void reportError( RecognitionException e ){
		super.reportError( e );
		nr_error++;
	}
public JAWSLexer(InputStream in) {
	this(new ByteBuffer(in));
```

```
}
public JAWSLexer(Reader in) {
	this(new CharBuffer(in));
}
public JAWSLexer(InputBuffer ib) {
	this(new LexerSharedInputState(ib));
}
public JAWSLexer(LexerSharedInputState state) {
	super(state);
	caseSensitiveLiterals = true;
	setCaseSensitive(true);
	literals = new Hashtable();
	literals.put(new ANTLRHashString("map", this), new Integer(65));
	literals.put(new ANTLRHashString("to", this), new Integer(74));
	literals.put(new ANTLRHashString("_y", this), new Integer(58));
	literals.put(new ANTLRHashString("for", this), new Integer(73));
	literals.put(new ANTLRHashString("_height", this), new Integer(67));
	literals.put(new ANTLRHashString("_label", this), new Integer(62));
	literals.put(new ANTLRHashString("movement", this), new Integer(63));
	literals.put(new ANTLRHashString("false", this), new Integer(54));
	literals.put(new ANTLRHashString("true", this), new Integer(53));
	literals.put(new ANTLRHashString("_x", this), new Integer(57));
	literals.put(new ANTLRHashString("engine", this), new Integer(69));
	literals.put(new ANTLRHashString("boolean", this), new Integer(52));
	literals.put(new ANTLRHashString("string", this), new Integer(51));
	literals.put(new ANTLRHashString("_image", this), new Integer(59));
	literals.put(new ANTLRHashString("entity", this), new Integer(56));
	literals.put(new ANTLRHashString("_width", this), new Integer(66));
	literals.put(new ANTLRHashString("attack", this), new Integer(64));
	literals.put(new ANTLRHashString("while", this), new Integer(72));
	literals.put(new ANTLRHashString("JAWS", this), new Integer(47));
	literals.put(new ANTLRHashString("point", this), new Integer(55));
	literals.put(new ANTLRHashString("_ai", this), new Integer(61));
	literals.put(new ANTLRHashString("_background", this), new Integer(68));
	literals.put(new ANTLRHashString("if", this), new Integer(70));
	literals.put(new ANTLRHashString("_weapon", this), new Integer(60));
	literals.put(new ANTLRHashString("int", this), new Integer(49));
	literals.put(new ANTLRHashString("double", this), new Integer(50));
	literals.put(new ANTLRHashString("declarations", this), new Integer(48));
	literals.put(new ANTLRHashString("else", this), new Integer(71));
	literals.put(new ANTLRHashString("by", this), new Integer(75));
}

public Token nextToken() throws TokenStreamException {
	Token theRetToken=null;
tryAgain:
	for (;;) {
		Token _token = null;
		int _ttype = Token.INVALID_TYPE;
		resetText();
		try {   // for char stream error handling
			try {   // for lexical error handling
				switch ( LA(1)) {
				case '0':  case '1':  case '2':  case '3':
				case '4':  case '5':  case '6':  case '7':
				case '8':  case '9':
```

```
{
		mNUMBER(true);
		theRetToken=_returnToken;
		break;
}
case '.':
{
		mDOT(true);
		theRetToken=_returnToken;
		break;
}
case '"':
{
		mSTRING(true);
		theRetToken=_returnToken;
		break;
}
case ':':
{
		mCOLON(true);
		theRetToken=_returnToken;
		break;
}
case ';':
{
		mSEMI(true);
		theRetToken=_returnToken;
		break;
}
case ',':
{
		mCOMMA(true);
		theRetToken=_returnToken;
		break;
}
case '{':
{
		mLBRACE(true);
		theRetToken=_returnToken;
		break;
}
case '}':
{
		mRBRACE(true);
		theRetToken=_returnToken;
		break;
}
case '[':
{
		mLBRACKET(true);
		theRetToken=_returnToken;
		break;
}
case ']':
{
		mRBRACKET(true);
```

```
            theRetToken=_returnToken;
            break;
}
case '(':
{
            mLPAREN(true);
            theRetToken=_returnToken;
            break;
}
case ')':
{
            mRPAREN(true);
            theRetToken=_returnToken;
            break;
}
case '~':
{
            mDISTANCE(true);
            theRetToken=_returnToken;
            break;
}
case '%':
{
            mMOD(true);
            theRetToken=_returnToken;
            break;
}
case '&':
{
            mAND(true);
            theRetToken=_returnToken;
            break;
}
case '|':
{
            mOR(true);
            theRetToken=_returnToken;
            break;
}
case '\':
{
            mTYPE_ACCESSOR(true);
            theRetToken=_returnToken;
            break;
}
case '\t':  case '\n':  case '\u000c':  case '\r':
case ' ':
{
            mWS(true);
            theRetToken=_returnToken;
            break;
}
case 'A':  case 'B':  case 'C':  case 'D':
case 'E':  case 'F':  case 'G':  case 'H':
case 'I':  case 'J':  case 'K':  case 'L':
case 'M':  case 'N':  case 'O':  case 'P':
```

```
case 'Q':  case 'R':  case 'S':  case 'T':
case 'U':  case 'V':  case 'W':  case 'X':
case 'Y':  case 'Z':  case '_':  case 'a':
case 'b':  case 'c':  case 'd':  case 'e':
case 'f':  case 'g':  case 'h':  case 'i':
case 'j':  case 'k':  case 'l':  case 'm':
case 'n':  case 'o':  case 'p':  case 'q':
case 'r':  case 's':  case 't':  case 'u':
case 'v':  case 'w':  case 'x':  case 'y':
case 'z':
{
        mIDENT(true);
        theRetToken=_returnToken;
        break;
}
default:
        if ((LA(1)=='/') && (LA(2)=='/')) {
                mCOMMENT(true);
                theRetToken=_returnToken;
        }
        else if ((LA(1)=='!') && (LA(2)=='=')) {
                mNOT_EQUALS(true);
                theRetToken=_returnToken;
        }
        else if ((LA(1)=='<') && (LA(2)=='=')) {
                mLTE(true);
                theRetToken=_returnToken;
        }
        else if ((LA(1)=='>') && (LA(2)=='=')) {
                mGTE(true);
                theRetToken=_returnToken;
        }
        else if ((LA(1)=='+') && (LA(2)=='=')) {
                mPLUS_EQUALS(true);
                theRetToken=_returnToken;
        }
        else if ((LA(1)=='-') && (LA(2)=='=')) {
                mMINUS_EQUALS(true);
                theRetToken=_returnToken;
        }
        else if ((LA(1)=='*') && (LA(2)=='=')) {
                mMULT_EQUALS(true);
                theRetToken=_returnToken;
        }
        else if ((LA(1)=='/') && (LA(2)=='=')) {
                mDIV_EQUALS(true);
                theRetToken=_returnToken;
        }
        else if ((LA(1)=='=') && (LA(2)=='=')) {
                mEQUALS_EQUALS(true);
                theRetToken=_returnToken;
        }
        else if ((LA(1)=='=') && (true)) {
                mEQUALS(true);
                theRetToken=_returnToken;
        }
```

```
                                        else if ((LA(1)=='<') && (true)) {
                                                mLT(true);
                                                theRetToken=_returnToken;
                                        }
                                        else if ((LA(1)=='>') && (true)) {
                                                mGT(true);
                                                theRetToken=_returnToken;
                                        }
                                        else if ((LA(1)=='+') && (true)) {
                                                mPLUS(true);
                                                theRetToken=_returnToken;
                                        }
                                        else if ((LA(1)=='-') && (true)) {
                                                mMINUS(true);
                                                theRetToken=_returnToken;
                                        }
                                        else if ((LA(1)=='*') && (true)) {
                                                mMULT(true);
                                                theRetToken=_returnToken;
                                        }
                                        else if ((LA(1)=='/') && (true)) {
                                                mDIV(true);
                                                theRetToken=_returnToken;
                                        }
                                        else if ((LA(1)=='!') && (true)) {
                                                mNOT(true);
                                                theRetToken=_returnToken;
                                        }
                                else {
                                        if (LA(1)==EOF_CHAR) {uponEOF(); _returnToken =
makeToken(Token.EOF_TYPE);}
                                else {throw new NoViableAltForCharException((char)LA(1),
getFilename(), getLine(), getColumn());}
                                }
                                }
                                if ( _returnToken==null ) continue tryAgain; // found SKIP token
                                _ttype = _returnToken.getType();
                                _returnToken.setType(_ttype);
                                return _returnToken;
                        }
                        catch (RecognitionException e) {
                                throw new TokenStreamRecognitionException(e);
                        }
                }
                catch (CharStreamException cse) {
                        if ( cse instanceof CharStreamIOException ) {
                                throw new
TokenStreamIOException(((CharStreamIOException)cse).io);
                        }
                        else {
                                throw new TokenStreamException(cse.getMessage());
                        }
                }
        }
}
```

```
	public final void mCOMMENT(boolean _createToken) throws RecognitionException,
CharStreamException, TokenStreamException {
		int _ttype; Token _token=null; int _begin=text.length();
		_ttype = COMMENT;
		int _saveIndex;

		match("//");
		{
		_loop4:
		do {
			if ((_tokenSet_0.member(LA(1)))) {
				{
				match(_tokenSet_0);
				}
			}
			else {
				break _loop4;
			}

		} while (true);
		}
		if ( inputState.guessing==0 ) {
			_ttype = Token.SKIP;
		}
		if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
			_token = makeToken(_ttype);
			_token.setText(new String(text.getBuffer(), _begin, text.length()-_begin));
		}
		_returnToken = _token;
	}

	protected final void mDIGIT(boolean _createToken) throws RecognitionException,
CharStreamException, TokenStreamException {
		int _ttype; Token _token=null; int _begin=text.length();
		_ttype = DIGIT;
		int _saveIndex;

		matchRange('0','9');
		if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
			_token = makeToken(_ttype);
			_token.setText(new String(text.getBuffer(), _begin, text.length()-_begin));
		}
		_returnToken = _token;
	}

	protected final void mALPHA(boolean _createToken) throws RecognitionException,
CharStreamException, TokenStreamException {
		int _ttype; Token _token=null; int _begin=text.length();
		_ttype = ALPHA;
		int _saveIndex;

		switch ( LA(1)) {
		case 'a':  case 'b':  case 'c':  case 'd':
		case 'e':  case 'f':  case 'g':  case 'h':
		case 'i':  case 'j':  case 'k':  case 'l':
		case 'm':  case 'n':  case 'o':  case 'p':
```

```
		case 'q':  case 'r':  case 's':  case 't':
		case 'u':  case 'v':  case 'w':  case 'x':
		case 'y':  case 'z':
		{
			matchRange('a','z');
			break;
		}
		case 'A':  case 'B':  case 'C':  case 'D':
		case 'E':  case 'F':  case 'G':  case 'H':
		case 'I':  case 'J':  case 'K':  case 'L':
		case 'M':  case 'N':  case 'O':  case 'P':
		case 'Q':  case 'R':  case 'S':  case 'T':
		case 'U':  case 'V':  case 'W':  case 'X':
		case 'Y':  case 'Z':
		{
			matchRange('A','Z');
			break;
		}
		case '_':
		{
			match('_');
			break;
		}
		default:
		{
			throw new NoViableAltForCharException((char)LA(1), getFilename(),
getLine(), getColumn());
		}
		}
		if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
			_token = makeToken(_ttype);
			_token.setText(new String(text.getBuffer(), _begin, text.length()-_begin));
		}
		_returnToken = _token;
	}

	public final void mNUMBER(boolean _createToken) throws RecognitionException,
CharStreamException, TokenStreamException {
		int _ttype; Token _token=null; int _begin=text.length();
		_ttype = NUMBER;
		int _saveIndex;

		{
		int _cnt9=0;
		_loop9:
		do {
			if (((LA(1) >= '0' && LA(1) <= '9'))) {
				mDIGIT(false);
			}
			else {
				if ( _cnt9>=1 ) { break _loop9; } else {throw new
NoViableAltForCharException((char)LA(1), getFilename(), getLine(), getColumn());}
			}

			_cnt9++;
		} while (true);
```

```
		}
		{
		if ((LA(1)=='.')) {
			mDOT(false);
			{
			int _cnt12=0;
			_loop12:
			do {
				if (((LA(1) >= '0' && LA(1) <= '9'))) {
					mDIGIT(false);
				}
				else {
					if ( _cnt12>=1 ) { break _loop12; } else {throw new
NoViableAltForCharException((char)LA(1), getFilename(), getLine(), getColumn());}
				}

				_cnt12++;
			} while (true);
			}
		}
		else {
		}

		}
		if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
			_token = makeToken(_ttype);
			_token.setText(new String(text.getBuffer(), _begin, text.length()-_begin));
		}
		_returnToken = _token;
	}

	public final void mDOT(boolean _createToken) throws RecognitionException,
CharStreamException, TokenStreamException {
		int _ttype; Token _token=null; int _begin=text.length();
		_ttype = DOT;
		int _saveIndex;

		match('.');
		if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
			_token = makeToken(_ttype);
			_token.setText(new String(text.getBuffer(), _begin, text.length()-_begin));
		}
		_returnToken = _token;
	}

	public final void mSTRING(boolean _createToken) throws RecognitionException,
CharStreamException, TokenStreamException {
		int _ttype; Token _token=null; int _begin=text.length();
		_ttype = STRING;
		int _saveIndex;

		_saveIndex=text.length();
		match("");
		text.setLength(_saveIndex);
		{
		_loop16:
```

```
		do {
			if ((LA(1)=='') && (LA(2)=='')) {
				match('');
				_saveIndex=text.length();
				match('');
				text.setLength(_saveIndex);
			}
			else if ((_tokenSet_1.member(LA(1)))) {
				{
				match(_tokenSet_1);
				}
			}
			else {
				break _loop16;
			}

		} while (true);
		}
		{
		if ((LA(1)=='')) {
			_saveIndex=text.length();
			match('');
			text.setLength(_saveIndex);
		}
		else {
		}

		}
		if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
			_token = makeToken(_ttype);
			_token.setText(new String(text.getBuffer(), _begin, text.length()-_begin));
		}
		_returnToken = _token;
	}

	public final void mCOLON(boolean _createToken) throws RecognitionException,
CharStreamException, TokenStreamException {
		int _ttype; Token _token=null; int _begin=text.length();
		_ttype = COLON;
		int _saveIndex;

		match(':');
		if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
			_token = makeToken(_ttype);
			_token.setText(new String(text.getBuffer(), _begin, text.length()-_begin));
		}
		_returnToken = _token;
	}

	public final void mSEMI(boolean _createToken) throws RecognitionException,
CharStreamException, TokenStreamException {
		int _ttype; Token _token=null; int _begin=text.length();
		_ttype = SEMI;
		int _saveIndex;

		match(';');
```

```
		if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
			_token = makeToken(_ttype);
			_token.setText(new String(text.getBuffer(), _begin, text.length()-_begin));
		}
		_returnToken = _token;
	}

	public final void mCOMMA(boolean _createToken) throws RecognitionException,
CharStreamException, TokenStreamException {
		int _ttype; Token _token=null; int _begin=text.length();
		_ttype = COMMA;
		int _saveIndex;

		match(',');
		if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
			_token = makeToken(_ttype);
			_token.setText(new String(text.getBuffer(), _begin, text.length()-_begin));
		}
		_returnToken = _token;
	}

	public final void mEQUALS(boolean _createToken) throws RecognitionException,
CharStreamException, TokenStreamException {
		int _ttype; Token _token=null; int _begin=text.length();
		_ttype = EQUALS;
		int _saveIndex;

		match('=');
		if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
			_token = makeToken(_ttype);
			_token.setText(new String(text.getBuffer(), _begin, text.length()-_begin));
		}
		_returnToken = _token;
	}

	public final void mLBRACE(boolean _createToken) throws RecognitionException,
CharStreamException, TokenStreamException {
		int _ttype; Token _token=null; int _begin=text.length();
		_ttype = LBRACE;
		int _saveIndex;

		match('{');
		if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
			_token = makeToken(_ttype);
			_token.setText(new String(text.getBuffer(), _begin, text.length()-_begin));
		}
		_returnToken = _token;
	}

	public final void mRBRACE(boolean _createToken) throws RecognitionException,
CharStreamException, TokenStreamException {
		int _ttype; Token _token=null; int _begin=text.length();
		_ttype = RBRACE;
		int _saveIndex;

		match('}');
```

```
		if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
			_token = makeToken(_ttype);
			_token.setText(new String(text.getBuffer(), _begin, text.length()-_begin));
		}
		_returnToken = _token;
	}

	public final void mLBRACKET(boolean _createToken) throws RecognitionException,
CharStreamException, TokenStreamException {
		int _ttype; Token _token=null; int _begin=text.length();
		_ttype = LBRACKET;
		int _saveIndex;

		match('[');
		if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
			_token = makeToken(_ttype);
			_token.setText(new String(text.getBuffer(), _begin, text.length()-_begin));
		}
		_returnToken = _token;
	}

	public final void mRBRACKET(boolean _createToken) throws RecognitionException,
CharStreamException, TokenStreamException {
		int _ttype; Token _token=null; int _begin=text.length();
		_ttype = RBRACKET;
		int _saveIndex;

		match(']');
		if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
			_token = makeToken(_ttype);
			_token.setText(new String(text.getBuffer(), _begin, text.length()-_begin));
		}
		_returnToken = _token;
	}

	public final void mLPAREN(boolean _createToken) throws RecognitionException,
CharStreamException, TokenStreamException {
		int _ttype; Token _token=null; int _begin=text.length();
		_ttype = LPAREN;
		int _saveIndex;

		match('(');
		if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
			_token = makeToken(_ttype);
			_token.setText(new String(text.getBuffer(), _begin, text.length()-_begin));
		}
		_returnToken = _token;
	}

	public final void mRPAREN(boolean _createToken) throws RecognitionException,
CharStreamException, TokenStreamException {
		int _ttype; Token _token=null; int _begin=text.length();
		_ttype = RPAREN;
		int _saveIndex;

		match(')');
```

```
		if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
			_token = makeToken(_ttype);
			_token.setText(new String(text.getBuffer(), _begin, text.length()-_begin));
		}
		_returnToken = _token;
	}

	public final void mNOT_EQUALS(boolean _createToken) throws RecognitionException,
CharStreamException, TokenStreamException {
		int _ttype; Token _token=null; int _begin=text.length();
		_ttype = NOT_EQUALS;
		int _saveIndex;

		match("!=");
		if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
			_token = makeToken(_ttype);
			_token.setText(new String(text.getBuffer(), _begin, text.length()-_begin));
		}
		_returnToken = _token;
	}

	public final void mLT(boolean _createToken) throws RecognitionException,
CharStreamException, TokenStreamException {
		int _ttype; Token _token=null; int _begin=text.length();
		_ttype = LT;
		int _saveIndex;

		match('<');
		if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
			_token = makeToken(_ttype);
			_token.setText(new String(text.getBuffer(), _begin, text.length()-_begin));
		}
		_returnToken = _token;
	}

	public final void mLTE(boolean _createToken) throws RecognitionException,
CharStreamException, TokenStreamException {
		int _ttype; Token _token=null; int _begin=text.length();
		_ttype = LTE;
		int _saveIndex;

		match("<=");
		if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
			_token = makeToken(_ttype);
			_token.setText(new String(text.getBuffer(), _begin, text.length()-_begin));
		}
		_returnToken = _token;
	}

	public final void mGT(boolean _createToken) throws RecognitionException,
CharStreamException, TokenStreamException {
		int _ttype; Token _token=null; int _begin=text.length();
		_ttype = GT;
		int _saveIndex;

		match('>');
```

```
		if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
			_token = makeToken(_ttype);
			_token.setText(new String(text.getBuffer(), _begin, text.length()-_begin));
		}
		_returnToken = _token;
	}

	public final void mGTE(boolean _createToken) throws RecognitionException,
CharStreamException, TokenStreamException {
		int _ttype; Token _token=null; int _begin=text.length();
		_ttype = GTE;
		int _saveIndex;

		match(">=");
		if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
			_token = makeToken(_ttype);
			_token.setText(new String(text.getBuffer(), _begin, text.length()-_begin));
		}
		_returnToken = _token;
	}

	public final void mPLUS(boolean _createToken) throws RecognitionException,
CharStreamException, TokenStreamException {
		int _ttype; Token _token=null; int _begin=text.length();
		_ttype = PLUS;
		int _saveIndex;

		match('+');
		if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
			_token = makeToken(_ttype);
			_token.setText(new String(text.getBuffer(), _begin, text.length()-_begin));
		}
		_returnToken = _token;
	}

	public final void mMINUS(boolean _createToken) throws RecognitionException,
CharStreamException, TokenStreamException {
		int _ttype; Token _token=null; int _begin=text.length();
		_ttype = MINUS;
		int _saveIndex;

		match('-');
		if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
			_token = makeToken(_ttype);
			_token.setText(new String(text.getBuffer(), _begin, text.length()-_begin));
		}
		_returnToken = _token;
	}

	public final void mMULT(boolean _createToken) throws RecognitionException,
CharStreamException, TokenStreamException {
		int _ttype; Token _token=null; int _begin=text.length();
		_ttype = MULT;
		int _saveIndex;

		match('*');
```

```
		if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
			_token = makeToken(_ttype);
			_token.setText(new String(text.getBuffer(), _begin, text.length()-_begin));
		}
		_returnToken = _token;
	}

	public final void mDIV(boolean _createToken) throws RecognitionException,
CharStreamException, TokenStreamException {
		int _ttype; Token _token=null; int _begin=text.length();
		_ttype = DIV;
		int _saveIndex;

		match('/');
		if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
			_token = makeToken(_ttype);
			_token.setText(new String(text.getBuffer(), _begin, text.length()-_begin));
		}
		_returnToken = _token;
	}

	public final void mDISTANCE(boolean _createToken) throws RecognitionException,
CharStreamException, TokenStreamException {
		int _ttype; Token _token=null; int _begin=text.length();
		_ttype = DISTANCE;
		int _saveIndex;

		match('~');
		if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
			_token = makeToken(_ttype);
			_token.setText(new String(text.getBuffer(), _begin, text.length()-_begin));
		}
		_returnToken = _token;
	}

	public final void mMOD(boolean _createToken) throws RecognitionException,
CharStreamException, TokenStreamException {
		int _ttype; Token _token=null; int _begin=text.length();
		_ttype = MOD;
		int _saveIndex;

		match('%');
		if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
			_token = makeToken(_ttype);
			_token.setText(new String(text.getBuffer(), _begin, text.length()-_begin));
		}
		_returnToken = _token;
	}

	public final void mPLUS_EQUALS(boolean _createToken) throws RecognitionException,
CharStreamException, TokenStreamException {
		int _ttype; Token _token=null; int _begin=text.length();
		_ttype = PLUS_EQUALS;
		int _saveIndex;

		match("+=");
```

```
		if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
			_token = makeToken(_ttype);
			_token.setText(new String(text.getBuffer(), _begin, text.length()-_begin));
		}
		_returnToken = _token;
	}

	public final void mMINUS_EQUALS(boolean _createToken) throws RecognitionException,
CharStreamException, TokenStreamException {
		int _ttype; Token _token=null; int _begin=text.length();
		_ttype = MINUS_EQUALS;
		int _saveIndex;

		match("-=");
		if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
			_token = makeToken(_ttype);
			_token.setText(new String(text.getBuffer(), _begin, text.length()-_begin));
		}
		_returnToken = _token;
	}

	public final void mMULT_EQUALS(boolean _createToken) throws RecognitionException,
CharStreamException, TokenStreamException {
		int _ttype; Token _token=null; int _begin=text.length();
		_ttype = MULT_EQUALS;
		int _saveIndex;

		match("*=");
		if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
			_token = makeToken(_ttype);
			_token.setText(new String(text.getBuffer(), _begin, text.length()-_begin));
		}
		_returnToken = _token;
	}

	public final void mDIV_EQUALS(boolean _createToken) throws RecognitionException,
CharStreamException, TokenStreamException {
		int _ttype; Token _token=null; int _begin=text.length();
		_ttype = DIV_ EQUALS;
		int _saveIndex;

		match("/=");
		if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
			_token = makeToken(_ttype);
			_token.setText(new String(text.getBuffer(), _begin, text.length()-_begin));
		}
		_returnToken = _token;
	}

	public final void mEQUALS_EQUALS(boolean _createToken) throws RecognitionException,
CharStreamException, TokenStreamException {
		int _ttype; Token _token=null; int _begin=text.length();
		_ttype = EQUALS_EQUALS;
		int _saveIndex;

		match("==");
```

```
		if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
			_token = makeToken(_ttype);
			_token.setText(new String(text.getBuffer(), _begin, text.length()-_begin));
		}
		_returnToken = _token;
	}

	public final void mAND(boolean _createToken) throws RecognitionException,
CharStreamException, TokenStreamException {
		int _ttype; Token _token=null; int _begin=text.length();
		_ttype = AND;
		int _saveIndex;

		match("&&");
		if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
			_token = makeToken(_ttype);
			_token.setText(new String(text.getBuffer(), _begin, text.length()-_begin));
		}
		_returnToken = _token;
	}

	public final void mOR(boolean _createToken) throws RecognitionException,
CharStreamException, TokenStreamException {
		int _ttype; Token _token=null; int _begin=text.length();
		_ttype = OR;
		int _saveIndex;

		match("||");
		if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
			_token = makeToken(_ttype);
			_token.setText(new String(text.getBuffer(), _begin, text.length()-_begin));
		}
		_returnToken = _token;
	}

	public final void mNOT(boolean _createToken) throws RecognitionException,
CharStreamException, TokenStreamException {
		int _ttype; Token _token=null; int _begin=text.length();
		_ttype = NOT;
		int _saveIndex;

		match('!');
		if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
			_token = makeToken(_ttype);
			_token.setText(new String(text.getBuffer(), _begin, text.length()-_begin));
		}
		_returnToken = _token;
	}

	public final void mTYPE_ACCESSOR(boolean _createToken) throws RecognitionException,
CharStreamException, TokenStreamException {
		int _ttype; Token _token=null; int _begin=text.length();
		_ttype = TYPE_ACCESSOR;
		int _saveIndex;

		match("'s");
```

```
		if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
			_token = makeToken(_ttype);
			_token.setText(new String(text.getBuffer(), _begin, text.length()-_begin));
		}
		_returnToken = _token;
	}

	public final void mWS(boolean _createToken) throws RecognitionException,
CharStreamException, TokenStreamException {
		int _ttype; Token _token=null; int _begin=text.length();
		_ttype = WS;
		int _saveIndex;

		{
		switch ( LA(1)) {
		case ' ':
		{
			match(' ');
			break;
		}
		case '\t':
		{
			match('\t');
			break;
		}
		case '\u000c':
		{
			match('\f');
			break;
		}
		case '\n':  case '\r':
		{
			{
			boolean synPredMatched53 = false;
			if (((LA(1)=='\r') && (LA(2)=='\n'))) {
				int _m53 = mark();
				synPredMatched53 = true;
				inputState.guessing++;
				try {
					{
					match('\r');
					match('\n');
					}
				}
				catch (RecognitionException pe) {
					synPredMatched53 = false;
				}
				rewind(_m53);
				inputState.guessing--;
			}
			if ( synPredMatched53 ) {
				match('\r');
				match('\n');
			}
			else if ((LA(1)=='\n')) {
				match('\n');
```

```
                        }
                        else if ((LA(1)=='\r') && (true)) {
                                match('\r');
                        }
                        else {
                                throw new NoViableAltForCharException((char)LA(1), getFilename(),
getLine(), getColumn());
                        }

                        }
                        if ( inputState.guessing==0 ) {
                                newline();
                        }
                        break;
                }
                default:
                {
                        throw new NoViableAltForCharException((char)LA(1), getFilename(),
getLine(), getColumn());
                }
                }
                }
                if ( inputState.guessing==0 ) {
                        _ttype = Token.SKIP;
                }
                if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
                        _token = makeToken(_ttype);
                        _token.setText(new String(text.getBuffer(), _begin, text.length()-_begin));
                }
                _returnToken = _token;
        }

        public final void mIDENT(boolean _createToken) throws RecognitionException,
CharStreamException, TokenStreamException {
                int _ttype; Token _token=null; int _begin=text.length();
                _ttype = IDENT;
                int _saveIndex;

                {
                switch ( LA(1)) {
                case 'a': case 'b': case 'c': case 'd':
                case 'e': case 'f': case 'g': case 'h':
                case 'i': case 'j': case 'k': case 'l':
                case 'm': case 'n': case 'o': case 'p':
                case 'q': case 'r': case 's': case 't':
                case 'u': case 'v': case 'w': case 'x':
                case 'y': case 'z':
                {
                        matchRange('a','z');
                        break;
                }
                case 'A': case 'B': case 'C': case 'D':
                case 'E': case 'F': case 'G': case 'H':
                case 'I': case 'J': case 'K': case 'L':
                case 'M': case 'N': case 'O': case 'P':
                case 'Q': case 'R': case 'S': case 'T':
```

```
		case 'U': case 'V': case 'W': case 'X':
		case 'Y': case 'Z':
		{
			matchRange('A','Z');
			break;
		}
		case '_':
		{
			match('_');
			break;
		}
		default:
		{
			throw new NoViableAltForCharException((char)LA(1), getFilename(),
getLine(), getColumn());
		}
		}
		}
		{
		_loop57:
		do {
			switch ( LA(1)) {
			case 'a': case 'b': case 'c': case 'd':
			case 'e': case 'f': case 'g': case 'h':
			case 'i': case 'j': case 'k': case 'l':
			case 'm': case 'n': case 'o': case 'p':
			case 'q': case 'r': case 's': case 't':
			case 'u': case 'v': case 'w': case 'x':
			case 'y': case 'z':
			{
				matchRange('a','z');
				break;
			}
			case 'A': case 'B': case 'C': case 'D':
			case 'E': case 'F': case 'G': case 'H':
			case 'I': case 'J': case 'K': case 'L':
			case 'M': case 'N': case 'O': case 'P':
			case 'Q': case 'R': case 'S': case 'T':
			case 'U': case 'V': case 'W': case 'X':
			case 'Y': case 'Z':
			{
				matchRange('A','Z');
				break;
			}
			case '0': case '1': case '2': case '3':
			case '4': case '5': case '6': case '7':
			case '8': case '9':
			{
				matchRange('0','9');
				break;
			}
			case '_':
			{
				match('_');
				break;
			}
```

```
                        default:
                        {
                                break _loop57;
                        }
                        }
                } while (true);
                }
                _ttype = testLiteralsTable(_ttype);
                if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
                        _token = makeToken(_ttype);
                        _token.setText(new String(text.getBuffer(), _begin, text.length()-_begin));
                }
                _returnToken = _token;
        }


        private static final long[] mk_tokenSet_0() {
                long[] data = new long[8];
                data[0]=-9224L;
                for (int i = 1; i<=3; i++) { data[i]=-1L; }
                return data;
        }
        public static final BitSet _tokenSet_0 = new BitSet(mk_tokenSet_0());
        private static final long[] mk_tokenSet_1() {
                long[] data = new long[8];
                data[0]=-17179878408L;
                for (int i = 1; i<=3; i++) { data[i]=-1L; }
                return data;
        }
        public static final BitSet _tokenSet_1 = new BitSet(mk_tokenSet_1());
        }
```

BACKEND LIBRARY CLASSES

```
/*
 * JAWSmovement.java
 *
 */

/**
 *
 * @author  Andy
 */

import javax.swing.Timer;
import java.awt.event.*;
import javax.swing.*;
import java.awt.*;
import java.util.*;

public class BulletController implements ActionListener{

        public boolean running = false;
        Vector entities = new Vector();
        Vector monsters = new Vector();
```

```
        Map map = null;
  public BulletController(Map m){
                map = m;
        }


  public void move(int delay){
          running = true;
    new Timer(delay, this).start();
  }


  public void actionPerformed(ActionEvent e) {
                for (int i=0;i<entities.size() ;i++ ){

                        // lookp through the vector and adjust
                        Entity entity = (Entity)entities.elementAt(i);

                        if(entity.y >=  0){
                                //System.out.println("Madeit");
                                entity.y = entity.y - 5;
                                entity.setLocation(entity.x,entity.y);
                                for (int z=0;z<monsters.size(); z++){

        if(JAWSpoint.collision((Entity)monsters.elementAt(z),entity,map)){
                                                        //collided
                                                        System.out.println("HITSSSSSSSSSS");
                                                        map.remove((Component)monsters.elementAt(z));
                                                        monsters.removeElementAt(z);
                                                        map.repaint();
                                                }

                                }
                        }  else{
                                map.remove((Component)entity);
                        }
                }

  }


        public void add(Entity en){
                entities.add(en);
        }

        public void addMonster(Entity en){
                System.out.println("Monster Added");
                monsters.add(en);
        }
}



/*
 * Entity.java
 */
```

```
/**
 *
 * @author Shoaib, Andy
 *
 */


import javax.swing.*;
import java.awt.*;
import java.awt.Image;
import java.io.*;
import java.awt.Graphics2D;
import java.awt.image.BufferedImage;
import java.awt.event.KeyListener;
import java.awt.Rectangle;
import java.awt.geom.*;

public class Entity extends JComponent{

   /** Creates a new instance of Entity */
    BulletController myC;



   public void addMon(Entity en)
   {
           myC.addMonster(en);
   }

   public Entity(String s, int w, int h, int sx, int sy, Map m, String l) {

                       myC = new BulletController(m);
      //Create a file object from the specified path and get the Image from
      //it
      string  = s;
       map = m; //set the id number
      file = new File(string);

      image = new ImageIcon(file.getAbsolutePath()).getImage();

       //this.setPreferredSize(new Dimension(w,h));
       width = w;
      height = h;

       this.setBorder(new javax.swing.border.LineBorder(new java.awt.Color(255, 0, 255)));
       this.setVisible(true);
       x  = sx;
       y  = sy;
             label = l;
          this.setBounds(x,y,width,height);
      rec.setBounds(x,y,width,height);
   }

   public void attack(){
    /*Enter this entities attack information here */
```

```
 Entity bullet[] = new Entity[1];
 bullet[0] =  new Entity("/o.gif",5,5,this.x,this.y,map,"x");
 map.add(bullet[0]);
 bullet[0].setBounds(this.x, this.y, 5,5);

 JAWSmovement movement = new JAWSmovement("bullet",bullet[0],map);

          myC.add(bullet[0]);
          if (!myC.running)
          {
                    System.out.println("In here");
                    myC.move(5);
          }
}
public void paint(Graphics g){
 g.setColor(Color.red);
 if(file.exists()){
 g.drawImage(image, 0, 0, width, height,this);
 }
 else
 {
    g.setColor(Color.white);
    g.drawString(label,0,15);
    //g.drawString(string,0,0);
 }
 rec.setBounds(x,y,width,height);
 //g.drawRect(0,0, width,height);
}

/*variable declarations*/

 File file;
 Image image;
 boolean keyboard;
int width;
int height;
 int x;
int y;
Point p;
Map map;
public int id;
 public Rectangle rec = new Rectangle();
String string;
 String label;
 }
 // andy and shoaib

 import javax.swing.Timer;
 import java.awt.event.*;
 import javax.swing.*;
 import java.awt.*;
 import java.io.*;
 import java.awt.image.BufferedImage;
 import java.util.Vector;
 import java.awt.event.KeyListener;
 public class JAWSmovement implements ActionListener{
```

```
    //pass in a single entity that will move a certain way
    public JAWSmovement(String s, Entity e, Map m){
    entity = e;
    name = s;
    single = true;
    map = m;
    }

    public void move(int delay){
       t = new Timer(delay, this);
          t.start();
    }


    public void actionPerformed(ActionEvent e) {

                        entity.y += 3;
                        entity.setLocation(entity.x,entity.y);
                                map.repaint();
                                //map.numBullets--;


    }

    /* variable declarations */
    Entity[] entities;
    Entity entity;
    String name;
    boolean single;
    Map map;
    Timer t;
         boolean n = true;
}
/*
 * Point.java
 *
 */

/**
 *
 * @author  Shoaib Anwar
 */


import java.lang.Math;

class JAWSpoint {

    /** Creates a new instance of Point */
    public JAWSpoint(int a, int b, String n) {
    x = a;
    y = b;
    name = n;
    }
```

```
//distance between entity and entity
public static double distance(Entity a, Entity b){
//distance = sqrt((y2 - y1) + (x2 - x1))
    if(a!=null && b!=null){
double y1 = a.rec.getCenterY();
double y2 = b.rec.getCenterY();

double x1 = a.rec.getCenterX();
double x2 = b.rec.getCenterX();

return Math.sqrt(Math.pow((y2 - y1),2)+Math.pow((x2-x1),2));
    }
    else return -1;
}

//distance between entity and point
public static double distance(Entity a, JAWSpoint b){
    if(a!=null && b!=null){
double y1 = a.rec.getCenterY();
int y2 = b.y;

double x1 = a.rec.getX();
 int x2 = b.x;

return Math.sqrt(Math.pow((y2 - y1),2)+Math.pow((x2-x1),2));
    }
    else return -1;
}


public static boolean collision(Entity a, Entity b,Map m){
    //System.out.println("in here");
if(a!=null && b!=null  && a != b){

    if(a.rec.intersects(b.rec)){
        //System.out.println("Collision");
        return true;
    }
    else
        return false;
}
else return false;
}

public static double collision1(Entity a, Entity b){
    //System.out.println("in here");
if(a!=null && b!=null  && a != b){

    if(a.rec.intersects(b.rec)){
        //System.out.println("Collision");
        return 0;
    }
    else
        return 1;
```

```
    }
    else return 1;
    }

    /*variable declarations */
    int x;
    int y;
    String name;

    public int getX()
    {
        return x;
    }

    public int getY()
    {
        return y;
    }

    public String getName ()
    {
        return name;
    }
}


import java.awt.event.KeyListener;
import javax.swing.*;
import java.awt.*;
import java.io.*;

class KeyReleasedListener implements KeyListener{

        public KeyReleasedListener(Entity entity, Map m, JFrame f){
            a = entity;
            map = m;
                    fref = f;

        }
            /*Checks to see if this entity will be out of bounds if moved */
    public boolean willBeOutOfBounds(int a, int b, int c, boolean add){
    boolean check = false;

    if(add){
        if(a + b > c){
            check = true;
        }
    }
    else{
            if(a - b < c){
                check = true;
            }
        }

    return check;
    }
```

```
    public void keyPressed(java.awt.event.KeyEvent e) {

          //if the entity won't be out of bounds move it

      fref.repaint();

      if(e.getKeyCode()==e.VK_RIGHT){
        if(!willBeOutOfBounds(a.x,a.width+5,map.width,true)){
        a.x = a.x +5;
        a.setLocation(a.x,a.y);
        }

      }
      else
      if(e.getKeyCode()==e.VK_LEFT){
        if(!willBeOutOfBounds(a.x,5,0,false)){
        a.x = a.x-5;
        a.setLocation(a.x,a.y);
        }
      }
      else
      if(e.getKeyCode()==e.VK_UP){
         if(!willBeOutOfBounds(a.y,5,0,false)){
         a.y = a.y -5;
         a.setLocation(a.x,a.y);
         }
      }
      else
      if(e.getKeyCode()==e.VK_DOWN){
         if(!willBeOutOfBounds(a.y,a.width+5,map.height,true)){
         a.y = a.y +5;
         a.setLocation(a.x,a.y);

         }

      }


      //user fires gun
      if(e.getKeyCode()==e.VK_SPACE){

       a.attack();
      }

    }

    public void keyReleased(java.awt.event.KeyEvent e) {

}

    public void keyTyped(java.awt.event.KeyEvent e) {
    }

   /*variable declarations */
        Entity a;
```

```
        Map map;
           JFrame fref;


 }



/*
 * Map.java
 *
 */

/**
 *
 * @author  Shoaib Anwar
 */



import javax.swing.*;
import java.awt.*;
import java.awt.Image;
import java.io.*;
import java.awt.event.KeyListener;
import java.awt.event.*;
import java.util.Vector;

public class Map extends JPanel implements ActionListener{

    /** Creates a new instance of Map if an Image is specified*/
    public Map(String s) {

        //Create a file object from the specified path and get the Image
        file = new File(s);
        ic = new ImageIcon(file.getAbsolutePath());
        System.out.println(file.getAbsolutePath());
        image = ic.getImage();
        width = ic.getIconWidth();
        height = ic.getIconHeight();
        this.setSize(width,height);
        this.setPreferredSize(new java.awt.Dimension(width,height));
        this.setLayout(null);
        this.setVisible(true);
        this.setOpaque(true);
        entities = new Vector();
          name = "blank";


    }

    /** Creates a new instance of Map if an Image isn't specified but size is*/
    public Map(int w, int h){

       //default is to set the background to black
       this.setBackground(Color.black);
```

```
    this.setSize(w,h);
     this.setPreferredSize(new java.awt.Dimension(w,h));
    this.setLayout(null);
    this.setVisible(true);
    this.setOpaque(true);
     width = w;
    height = h;
    entities = new Vector();

  }

  /** Creates an new instance of Map if nothing is specified */
  public Map(){

      this.setBackground(Color.black);
      this.setPreferredSize(new java.awt.Dimension(300,500));
      this.setLayout(null);
      this.setVisible(true);
      this.setOpaque(false);
      width = 300;
      height = 500;
      entities = new Vector();

   }

   public void collisionCheck(int delay){
     new Timer(delay, this).start();
  }


  public void actionPerformed(ActionEvent e) {

  }

  public void addToVector(Entity a){
   entities.add(a);
  }
  /* variable declarations */

  ImageIcon ic;
  File file;
  Image image = new ImageIcon("x.gif").getImage();
  int height;
  int width;
  Vector entities;
  String name;
  int numBullets;
  /*the entities will be this height */



}
```

JAWS Translation/Compiler

```
// justin

import java.util.*;
import java.io.*;

public class JAWSOutputter
{

   public Vector ints;
   public Vector doubles;
   public Vector strings;
   public Vector booleans;
   public Vector entities;
   public Vector movements;
   public Vector attacks;
   public Vector maps;
   public Vector points;

   public String engine;
   public String name;

   public JAWSOutputter()
   {
       ints = new Vector();
       doubles = new Vector();
       strings = new Vector();
       booleans = new Vector();
       entities = new Vector();
       movements = new Vector();
       attacks = new Vector();
       maps = new Vector();
       points = new Vector();

       engine = "//engine code\r\n";
       name = "JAWSProgram";
   }

   public void setName(String n)
   {
       name = n;
   }

   public void addInt(JAWSInt x)
   {
       ints.addElement(x);
   }

   public void addDouble(JAWSDouble x)
   {
       doubles.addElement(x);
   }

   public void addString(JAWSString x)
   {
       strings.addElement(x);
```

```
}

public void addBoolean(JAWSBoolean x)
{
    booleans.addElement(x);
}

public void addPoint(JAWSpoint x)
{
    points.addElement(x);
}

public void addMap(Map x)
{
    maps.addElement(x);
}

public void addEntity(JAWSEntity x)
{
    entities.addElement(x);
}

public void addEngineStatement(String x)
{
    engine += (x+"\r\n");
}

public String outEngine()
{
    String s = engine;
    s+= "}}\r\n";
    s+= "class ThreadController implements Runnable{\r\n";
    s+= name + " j;\r\n";
    s+= "Thread T;\r\n";
    s+= "public ThreadController("+name+" jaws){\r\n";
    s+= "j = jaws;T = new Thread(this);T.start();}\r\n";
    s+= "public void run() {\r\n";
    s+= "while (T!=null){j.repeat();}}\r\n";
    s+= "public void kill(){\r\n";
    s+= "if (T!=null) { T=null; }}}\r\n";


    return s;
}

public String outInts()
{
    String s = "//declared int's\r\n";
    for (int i = 0; i < ints.size(); i++)
    {
       JAWSInt jint = (JAWSInt)ints.elementAt(i);
       s += "int "+jint.getName()+" = "+ jint.getInt() + ";\r\n";

    }
    return s;
}
```

```
public String outDoubles()
{
    String s = "//declared doubles\r\n";
    for (int i = 0; i < doubles.size(); i++)
    {
       JAWSDouble jd = (JAWSDouble)doubles.elementAt(i);
       s += "double "+jd.getName()+" = "+ jd.getDouble() + ";\r\n";
    }
    return s;
}

public String outStrings()
{
    String s = "//declared Strings\r\n";
    for (int i = 0; i < strings.size(); i++)
    {
       JAWSString js = (JAWSString)strings.elementAt(i);
       s += "String "+js.getName()+" = \""+ js.getString() + "\";\r\n";
    }
    return s;

}

public String outBooleans()
{
    String s = "//declared booleans\r\n";
    for (int i = 0; i < booleans.size(); i++)
    {
       JAWSBoolean jb = (JAWSBoolean)booleans.elementAt(i);
       s+= "boolean " +jb.getName()+" = "+jb.getBoolean()+ ";\r\n";
    }
    return s;
}

public String outPoints()
{
    String s = "//declared JAWSpoints\r\n";
    for (int i = 0; i < points.size(); i++)
    {
       JAWSpoint jp = (JAWSpoint)points.elementAt(i);
       s+= "JAWSpoint "+jp.getName()+" = new JAWSpoint("+
              jp.getX()+","+jp.getY()+","+jp.getName()+");\r\n";
    }
    return s;
}

public String outMaps()
{
    String s = "//declared Maps\r\n";
    for (int i = 0; i < maps.size(); i++)
    {
       Map m = (Map)maps.elementAt(i);
       int w = m.width;
       int h = m.height;
       if (w == -1)
```

```
                w = 300;
            if (h == -1)
                h = 500;
            File f = m.file;
            if (f != null)
            {
                s += "Map "+m.name+" = new Map(\""+f.toString()+"\");\r\n";
            }
            else
            {
                s += "Map "+m.name+" = new Map("+w+","+h+");\r\n";
            }

        }
        return s;
    }

    public String outEntities()
    {
        String map = ( (Map)maps.elementAt(0) ).name;
        String s = "//declared entities\r\n";
        for (int i = 0; i <entities.size(); i++)
        {
            JAWSEntity je = (JAWSEntity)entities.elementAt(i);
            s+= "Entity[] "+je.name+" = new Entity"+"["+je.mult+"];\r\n";

        }

        return s;
    }

    public String outHeader()
    {
        String s = "//JAWS generated output Game\r\n"+
                    "import javax.swing.*;\r\n"+
                    "import java.awt.*;\r\n" +
                    "import java.io.*;\r\n"+
                    "import java.awt.image.BufferedImage;\r\n"+
                    "import java.util.Vector;\r\n"+
                    "import java.awt.event.KeyListener;\r\n"+
                    "public class "+name+" extends JPanel{\r\n"+
                    "public Vector entities = new Vector();\r\n"+
                    "JFrame fref;\r\n";
        return s;
    }


    public String outCons()
    {
        String map = ( (Map)maps.elementAt(0) ).name;
        String s = "//JAWS Constructor\r\n"+
                    "public "+name+" (JFrame f) {\r\n"+
                    "fref = f;\r\n";

        for (int i = 0; i < maps.size(); i++)
        {
```

```
        Map m = (Map)maps.elementAt(i);
        s+= m.name+".setVisible(true);\r\n";
        s+= m.name+".setOpaque(true);\r\n";

    }

    for (int i = 0; i < entities.size(); i++)
    {

        JAWSEntity je = (JAWSEntity)entities.elementAt(i);
        for (int j = 0; j < je.mult; j++)
        {
                s+=je.name+"["+j+"] = new Entity(\""+je.image+"\","+je.width+","+je.height+","
                +je.x+","+je.y+","+map+",\""+je.label+"\");\r\n";
                s+="entities.add("+je.name+"["+j+"]);\r\n";
                s+=map+".add("+je.name+"["+j+"]);\r\n";
                s+=je.name+"["+j+"].setBounds("+je.x+","+je.y+","+je.width+","+je.height+");\r\n";


                if (je.ai.equals("keyboard"))
                {
                   s+="KeyReleasedListener kl"+i+" = new KeyReleasedListener("
                            +je.name+"["+j+"],"+ map+",fref);\r\n";
                   s+="fref.addKeyListener(kl"+i+");\r\n";
                }

        }



    }


    s+= "}"  ;
    return s;
}

public String outMain()
{
    String map = ( (Map)maps.elementAt(0) ).name;
    String s = "public static void main(String[] args){\r\n"+
                "\r\n"+
                "\r\n"+
                "JFrame frame = new JFrame(\"Space Shooter\");\r\n"+
                "frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);\r\n"+
                "\r\n"+
                "//make new jaws program and add its map to the content pane\r\n"+
                name+" j = new "+name+"(frame);\r\n"+
                "frame.setContentPane(j."+map+");\r\n"+
                "frame.setResizable(false);\r\n"+
                "frame.pack();\r\n"+
                "\r\n"+
                "//show frame\r\n"+
                "frame.setVisible(true);\r\n"+
                "ThreadController myCon = new ThreadController(j);\r\n"+
                "}\r\n"+
```

```
                    "public void repeat() {\r\n";
        return s;
    }


    public static void main(String[] args)
    {

        //outputFile("test.txt");
    }


    public void outputFile(String filename)
    {
        //System.out.println(output1());
        try {
            FileWriter fw = new FileWriter(filename);
            PrintWriter pw = new PrintWriter(new BufferedWriter(fw));
            pw.println(outHeader());
            pw.println(outInts());
            pw.println(outDoubles());
            pw.println(outStrings());
            pw.println(outBooleans());
            pw.println(outPoints());
            pw.println(outMaps());
            pw.println(outEntities());
            pw.println(outCons());
            pw.println(outMain());
            pw.println(outEngine());

            pw.close();
        }
        catch (Exception e)
        {}
    }

}


public class JAWSBoolean extends JAWSDataType
{

    public boolean data;

    public JAWSBoolean(boolean x)
    {
        data = x;
    }

    public String getType()
    {
        return "boolean";
    }

    public boolean getBoolean()
    {
```

```
        return data;
    }
}


public class JAWSDataType
{

    String name;

    public JAWSDataType()
    {
        name = null;
    }

    public JAWSDataType(String s)
    {
        name = s;
    }

    public String getType()
    {
        return "abstract";
    }

    public void setName(String s)
    {
        name = s;
    }

    public String getName()
    {
        return name;
    }

}
```

```
public class JAWSDouble extends JAWSDataType
{

   public double data;

   public JAWSDouble(double x)
   {
        data = x;
   }

   public String getType()
   {
        return "double";
   }

   public double getDouble()
   {
        return data;
   }

}




public class JAWSEntity
{

        String name="";
        int height=0;
        int width=0;
        int mult=0;
        String image="";
        String label="";
        String weapon="";
        String ai="";
        int x=0;
        int y=0;

        public JAWSEntity()
        {

        }


}

public class JAWSInt extends JAWSDataType
{

   public int data;

   public JAWSInt(int x)
   {
        data = x;
   }
```

```
   public String getType()
   {
       return "int";
   }

   public int getInt()
   {
       return data;
   }

}


public class JAWSString extends JAWSDataType
{

   public String data;

   public JAWSString(String x)
   {
       data = x;
   }

   public String getType()
   {
       return "string";
   }

   public String getString()
   {
       return data;
   }

}
```

JAWS Executable

```
// justin

import java.io.*;
import antlr.CommonAST;

public class jawsc {

  public static void main(String[] args) {
    // Use a try/catch block for parser exceptions
    try {
      // if we have at least one command-line argument
      if (args.length > 0 ) {
        //System.err.println("Parsing...");

        // for each directory/file specified on the command line
        for(int i=0; i< args.length;i++)
```

```
      doFile(new File(args[i])); // parse it
    }
    else
      System.err.println("Usage: java jtest <directory name>");

  }
  catch(Exception e) {
    System.err.println("exception: "+e);
    e.printStackTrace(System.err);   // so we can get stack trace
  }
}


// This method decides what action to take based on the type of
//   file we are looking at
public static void doFile(File f) throws Exception {
  // If this is a directory, walk each file/dir in that directory
  if (f.isDirectory()) {
    String files[] = f.list();
    for(int i=0; i < files.length; i++)
      doFile(new File(f, files[i]));
  }

  // otherwise, if this is a java file, parse it!
  else
    parseFile(new FileInputStream(f));

}

// Here's where we do the real work...
public static void parseFile(InputStream s) throws Exception {
  try {
    // Create a scanner that reads from the input stream passed to us
    JAWSLexer lexer = new JAWSLexer(s);

    // Create a parser that reads from the scanner
    JAWSParser parser = new JAWSParser(lexer);

    // start parsing at the compilationUnit rule
      System.err.println("Parsing...");

    parser.program();

    if (parser.nr_error > 0)
    {
        System.out.println("Unable to compile");
        return;
    }

    CommonAST tree = (CommonAST)parser.getAST();

    System.out.println(tree.toStringList());

    JAWSWalker walker = new JAWSWalker();
    walker.program(tree);
```

```
    }
    catch (Exception e) {
      System.err.println("parser exception: "+e);
      e.printStackTrace();   // so we can get stack trace
    }
  }
}
```