

The Drone Language

A Stack-Based Imperative Language

Fall 2012

COMS 4115

George Brink

Shuo Qiu

Xiaotong Chen

Xiang Yao



**Content**

Chapter 1: Introduction & Purpose 5

1.1 Purpose & Background 5

1.2 The Drone Language 5

1.3 The Drone War Game 6

1.3.1 Game Overview 6

1.3.2 Arena 6

1.3.3 Drone 6

1.3.4 Bullet 7

1.3.5 Drone Actions 7

1.4 GUI 8

Chapter 2: Tutorial 9

2.1 Getting Started 9

2.2 Compiling and running Drones 10

2.3 Variable types 10

2.4 Functions 11

2.5 Labels and Jumps 11

2.6 GUI Outputs 13

2.6.1 Text Status of Drones 13

2.6.2 Arena GUI 13

Chapter 3: Reference Manual 14

3.1. Language Syntax 14

3.1.1 Keywords 14

3.1.2 Player defined names 14

3.1.3 Comments 14

3.1.4 Functions 15

3.1.5 Label 16

3.2. Fundamental Types 17

3.2.1 Integer 17

3.2.2 Boolean 17

3.2.3 Flags 17

3.3. Variables 17

3.4. Operators 18

3.4.1 Arithmetic operators 18

3.4.2 Logic operators 18

3.4.3 Logic constants 18

3.4.4 Conditions 18

3.4.5 Variable manipulation 18

3.4.6 Stack manipulation 19

3.5. Game specific functions 19

3.5.1 Move 19

3.5.2 Stop 19

3.5.3 Shoot 19

3.5.4 Look 19

3.5.5 isFoe 20

3.5.6 isAlly 20

3.5.7 isWall 20

3.5.8 wait 20

3.5.9 getHealth 20

3.5.10 random 20

3.6. Pseudo-commands 21

3.6.1 Conditions 21

3.6.2 Loops 21

Chapter 4: Drone-Basic 23

4.1 The conditional branching 23

4.2 The conditional loops 23

4.3 The counted loop 24

4.4 User procedures and functions 24

4.5 Variables 24

4.6 The game-related functions 24

4.7 The search procedure 25

4.8 Comparison between the base Drone language and Drone-Basic 25

Chapter 5: Project Plan 26

5.1 Process 27

5.1.1 Planning 27

5.1.2 Specification 27

5.1.3 Development 27

5.1.4 Testing 28

5.2 Programming Style Guide 28

5.2.1 General Programming Principles 28

5.2.2 Keep Testing Everything 29

5.2.3 Keep Communications 29

5.2.4 Using Version Control Tool 29

5.2.5 Mutual Code Review 30

5.2.6 Code Documentation & Comments 30

5.3 Project Timeline 30

5.4 Project Log 31

5.5 Team Responsibility 33

5.6 Development Environment 34

Chapter 6: Architecture Design 35

6.1 Design Overview 35

6.2 Interfaces Between the Components 37

6.2.1 Scanner (scanner.mll -Author: George) 37

6.2.2 Parser (parser.mly -Author: George, Xiaotong, Xiang) 37

6.2.3 AST (ast.ml -Author: George) 37

6.2.4 Arena (arena.ml -Author: George, Xiaotong, Xiang, Shuo) 37

6.2.5 Drone (drone.ml -Author: George, Xiaotong, Xiang, Shuo) 37

6.2.6 Bullet (bullet.ml -Author: Xiaotong, Xiang) 38

6.2.7 GUI (gui.ml -Author: Shuo, George) 38

6.2.8 Helper Funcs (utils.ml -Author: Xiaotong, Xiang) 38

6.2.9 Drone-Basic (scanner\_dbt.mll and parser\_dbt.mly –Author: George) 38

Chapter 7: Test Plan 39

7.1 Unit Testing 39

7.1.1 Integer 39

7.1.2 Comment 40

7.1.3 Variables 40

7.1.4 Arithmetic operators 41

7.1.5 Logic constants 42

7.1.6 Logic operators 42

7.1.7 Stack manipulation 44

7.1.8 Function 46

7.1.9 Label 47

7.1.10 Move 47

7.1.11 Stop 48

7.1.12 Shoot 48

7.1.13 Look 48

7.1.14 isFoe 49

7.1.15 isAlly 49

7.1.16 isWall 50

7.1.17 Wait 50

7.1.18 GetHealth 51

7.1.19 Random 51

7.1.20 Endless Loop 52

7.1.21 Conditional Loop 53

7.1.22 if 56

7.1.23 if-else 57

7.2 Integration Test: An Example Programs 58

7.2.1 Drone Berserk 58

7.2.2 Drone Rabbit 71

Chapter 8: Lessons Learned 80

8.1 George Brink 80

8.2 Shuo Qiu 80

8.3 Xiang Yao 81

8.4 Xiaotong Chen 81

Appendix 83

Source Code Listing 83

1. Scanner.mll 83

2. Parser.ply 85

3. AST.ml 88

4. Drone.ml 91

5. Arena.ml 100

6. main.ml 104

7. gui.ml 105

8. bullet.ml 108

9. utils.ml 110

10. scanner\_dbt.mll (George Brink’s individual contribution) 110

11. parser\_dbt.mly (George Brink’s individual contribution) 113

# Chapter 1: Introduction & Purpose

## 1.1 Purpose & Background

Drone War is a video game, which belongs to the “programming game” genre. As in all such games, the player has no direct influence on the course of the game. Instead, a player writes a program, which acts as an AI for the game characters and watch how those characters interact. The Drone War is based on a concept of a battle-royal between several drones (each with its own AI program). Drones are randomly dropped into the arena and fight with each other until only one is left or the time limit for the battle is exceeded.

Since the Drone War’s primary concept is a battle, the language for the AIs used in it should encourage writing fast, predictable, and efficient algorithms. On the other hand, the Drone War is essentially a game and its intended audience is as wide as possible, but not all potential players know the art of programming and have experience in playing with the programming game. So, in order to lower the threshold, the language for drones should be simple and it should have as few operators and concepts as possible.

To satisfy these requirements, the Drone Language was designed.

## 1.2 The Drone Language

Drone language is a stack-based imperative language. The stack accepts only integers, booleans, and flags. Integers can be used as arithmetic operands or parameters of the functions. Booleans are subject to stack manipulation operations and as parameter for conditional jump operators. Flags are subject to stack manipulation operations and special functions which check the flag is it of the expected kind and leave boolean true or false on the stack. Each word read from the source code is either a comment, integer, boolean, call to a user defined function, label, variable, or operator.

To make the Drone Language easier to use, we added conditional execution, endless loops and conditional loops. Those compound statements are considered to be “a syntactic sugar”. They are not executed directly, but translated into a set of labels and conditional jumps.

## 1.3 The Drone War Game

### 1.3.1 Game Overview

The battle in the Drone War game happens in fixed-size arena and with multiple drones acting individually, under control of AI programs written by players. Each AI file passed to the game from command line is considered to be individual drone (it is possible to run several drones against each other under the control of the same AI). Before the battle starts, drones can be separated into different teams and if drone’s AI is smart enough, several drones of the same team can help each other.

Drones can move around the Arena, look around and shoot Bullets. Bullets are flying to a specified distance in the specified direction and once distance is reached or Bullet hits the wall of the Arena, Bullet explodes. The explosion of the Bullet damages all Drones which are close enough. Once Drone life reach 0, it considered “dead”.

The concept of the “fighting machine” and simplicity of the Drone Language lead to the very strict unforgiveness of the errors in programming, any error in AI is considered to be a fatal one and if it happened, the drone instantly become “brain dead”. There is no graceful error handling in the drones’ AI. The drone which encounter such problem become frozen and while it is not technically dead yet, it does nothing for the remainder of the battle and become an easy prey for the opponents.

The flow of the battle is controlled by ticks. Each operation performed by the AI takes exactly one tick to complete. The moving of drones and bullets also happens under the same tick counter. That ensures that each drones are moving simultaneously and the AI which acts more efficiently has a better chance of winning against a not so efficient drones.

The battle continues until only one drone is left in play or battle for the predefined length of time.

### 1.3.2 Arena

Arena is a square of size 1000\*1000 units enclosed by impenetrable walls. Drone which hits the wall receive some damage. Bullet which hits the wall immediately explodes.

### 1.3.3 Drone

In the arena, Drone is represented as a land vehicle with a freely turning cannon (meaning a drone can move in one direction while shooting in another). Each Drone has 100 health points at the start of the battle. Once drone’s HP reaches 0, it cannot do anything and leaves its body in the arena.

At the start of the battle, drones are put on the Arena at random X and Y coordinates.

### 1.3.4 Bullet

Bullets are shot by drones. They are not controlled by players in any way. Bullet always flies until it reaches the specified distance or hit the wall of the arena.

Bullet’s explosion has a radius of 50 points and damage received by the drone inside the blast radius is proportional to the distance from the center of explosion. If a drone was hit directly it receives 50 points of damage. If distance to the epicenter was 1 point, drone receives 49 points of damage. Distance of 50 points or more is completely safe. A drone can be damaged by its own projectile if it blows up close enough. Bullet’s speed is 5 points per tick. A Bullet cannot travel for more than 500 points (half of the arena).

### 1.3.5 Drone Actions

#### 1.3.5.1 Move

Drone can move around the arena by issuing command: *move* with one parameter *direction*. Once the command is issued, the drone starts moving in the desired direction until next *move* command changes it or the *stop* command cancel the movement. Drone does not have “mass” so there is no need to worry about inertia. If drone hits the wall of the arena it loses 10 HP as a result of the hit. The movement speed is set to 1 unit per step.

#### 1.3.5.2 Look

Drone can see other drones and walls of the arena by issuing a command: *look* with one parameter *direction*. *Look* has an “angle of vision” with the side angle of 30 degrees. This means, the drone sees not just objects on the straight line but in the area of a triangle. The distance to the wall is calculated by the exact direction of the *look*.

The *look* command returns a list of tuples: [drone1 [drone2 ...]] wall

Where each tuple consists of a flag (what this tuple describes?), direction (exact direction the object), and distance (distance to the object). The ‘type’ flags can be one of FOE, ALLY, or WALL. The WALL tuple is always the last one in the list and acts as an indicator that there were no more drones seen in the given direction.

#### 1.3.5.3 Shoot

Drone can shoot by issuing command: *shoot* with two parameters *direction* and *distance*, which mean where and how far the bullet will fly before exploding. Drone can issue a *shoot* command once every 10 ticks. This timeout represents “gun is reloading” or “cooling off period”. If drone attempts to shoot more often, the *shoot* will return FALSE. If shooting was successful – TRUE.

NB: this return code does not tell was the target hit or not.

## 1.4 GUI

The GUI of the Drone War Game shows the state of the battle tick by tick, as well as stats of each drone on the battlefield.

The GUI representation of the arena depends on the size of the window and arena does not always look like square but it shows the correct position of each object.

The detail information of drones is displayed to the right of the arena. “The total ticks” shows the total number of ticks since the battle started. And “AI ticks” of each drone will show its live time.

The drones are drawn as a triangle with a line coming from its center. The direction that the acute angle pointing at is the moving direction of drone and the direction of the line is the drone’s gun direction. Also, drones in different teams will be displayed in different color with their names and health near them. When a drone is dead, a red cross will be shown over it.

The bullet in GUI is a black solid circle and when bullet explodes, it will be a red solid five-pointed star that we can easily find out whether a drone is damaged by this bullet.

# Chapter 2: Tutorial

## 2.1 Getting Started

The idea is to create a drone to beat others’. As to “write” a drone, you may need operations like: *dup*, *drop*, *dropall*, *swap*, *over*, or *rot* to manipulate the stack. Operations like *read* and *store* can help declare or use variables. Labels in conjunction with operations *Jump* and *jumpif* can help build a complex control flow (or you can choose the easier way: use *begin*, *while*, *again* to make loops and *if*, *else*, *endif* for branching; just like in any high-level programming language). Like in other languages *and*, *or*, and *not* are logic operators to deal with boolean. As stated, Drone language is a game language thus, there are several game oriented functions: *move*, *shoot*, *look*, *wait*, *getHealth*, *isFoe*, *isAlly*, *isWall*. By using these functions and operations above, a programmer can easily create a smart drone to fight with other drones.

Here is a simple example of a drone:

// This drone is a kind of wimp.

// It continues running from one wall to another until dead

begin // start of the main loop

0 360 random // randomly pick a direction

direction store // save the randomly picked value

direction read move // move to the direction

begin // move to the wall stop before hitting

direction read look // Look forward.

begin

iswall not while // If object is not a wall,

drop // ignore direction

drop // and distance to it.

again

drop // ignore direction to the wall

20 > while // If distance to the wall is more

again // than 20, then repeat the loop.

stop // Stop moving once wall was reached.

again // Repeat the main loop forever

## 2.2 Compiling and running Drones

In the Drone War, each Drone file contains a complete AI for exactly one drone. The files with a text of the AI should have an extension **.dt**.

To add Drones to the game, player just passes files with AIs to the game engine in the command line.

It is almost meaningless to pass single drone to the game, since it would be the only one on the arena and the battle finish immediately with a “winner by default”. Usually, the game starts with passing several drone files to the game engine:

./DroneWar drone1.dt drone2.dt drone3.dt drone4.dt

Teams can be specified by adding **-t** key between drone files:

./DroneWar drone1.dt drone2.dt -t drone3.dt drone4.dt

Here, the first two drones will fight for themselves, but third and fourth will be a team mates.

Beside the **-t** key, the game engine recognizes two other useful keys: **-D** and **-q**.

The **-D** enables the debug mode for all drones passed after it. For example:

./DroneWar drone1.dt drone2.dt -D drone3.dt drone4.dt

The first two drones will fight as usual, but for the last two, at the start of the battle, the game engine will create a **.dt.decompiled** file which will contain the exact list of bytecodes which would control the drone’s behavior. All labels will disappear and all jumps to labels will be converted into absolute jumps to the operation.

After the battle started, each drone in debug mode will add a line to the **.dt.debug** file. In this file the player will find which exact operation the drone was supposed to perform at some tick and what was the contents of the stack before the operation.

This mode is useful for finding errors in the AI and detailed understanding what are the high-level compound statements actually are.

The **-q** flag disables GUI completely. This can be used to considerably speed up a battle. For example if player wants to gather statistics on how better or worse one drone actually is. The individual battle cannot answer this question with a good degree of certainty because drones appear at random position in the arena and some possible randomness in the drones’ behavior.

## 2.3 Variable types

Integer, Boolean, and Flags are fundamental types in the Drone language. Integer and Boolean are the standard types which can be found in any other language. Flags, on the other hand, are specific to the DroneWar game and can be one of Foe, Ally or Wall. Flags have only one purpose, they are used to indicate what the drone sees. The Drone language has several functions which use flags as input and return a boolean type value to tell if top of the stack contains a flag of the specified kind, it’s, say did *look* function detected a foe or an ally.

Below are some examples about the usage of fundamental types:

// some arithmetic operations and compound statements

begin // start of the loop

a read // read contents of variable a

9 < while // if it is less then 9, stay in the loop

b read // read contents of variable b

2 + // add 2 to it and leave result on stack

store b // save top of the stack to variable b

a read 1 + a store // increment contents of a

again // repeat the loop

// example of a flag operation

60 look // fill the stack with tuples

isfoe // does the top of the stack contain FOE flag?

if // if yes

shoot // then shoot

else // if not

move // then move

endif // end of branching

## 2.4 Functions

Player can define functions by starting it with “sub” and ending with “endsub”. Between “sub” and “endsub” is the function’s body. All the names of functions are global. So, never try to define two functions using the same name. Also, one function cannot contain another. Functions can be called by simply using their names.

Below is an example of definition and call of a function:

sub increment // define a function called “increment”

1 + // the function will put 1 on the stack

// and add it to whatever was on top of

endsub // the stack before the call

a read 1 + // direct use of the 1 + operators

a read increment // call to a function

## 2.5 Labels and Jumps

Instead of (or in conjunction with) using high-level compound statements as in examples above, the same algorithms can be created by using labels and jumps.

Labels are defined by adding colon to the word and a program can do unconditional and conditional jumps to these labels.

The simple unconditional loop is a just a:

labelA: // define a point in the code

doSomething // call a function

labelA jump // and repeat it indefinetely

The conditional branching is a little more difficult:

a read 9 < // is contents of variable less than 9?

labelDone jumpif // if yes, then goto the label

doSomething // if not, do this

labelDone: // just a label

doSomethingElse // this will be done either immediately

// if a is less than 9, or after the

// the doSomething if variable a is

// equal or more than 9

Conditional loops are done by combining this two technics:

labelStart: // the start of the loop

a read 9 < // is contents of variable less than 9?

labelDone jumpif // if yes, then jump out of the loop

doSomething // if not, do this

labelStart jump // and repeat

labelDone: // once variable become less than 9,

// we exit the loop here

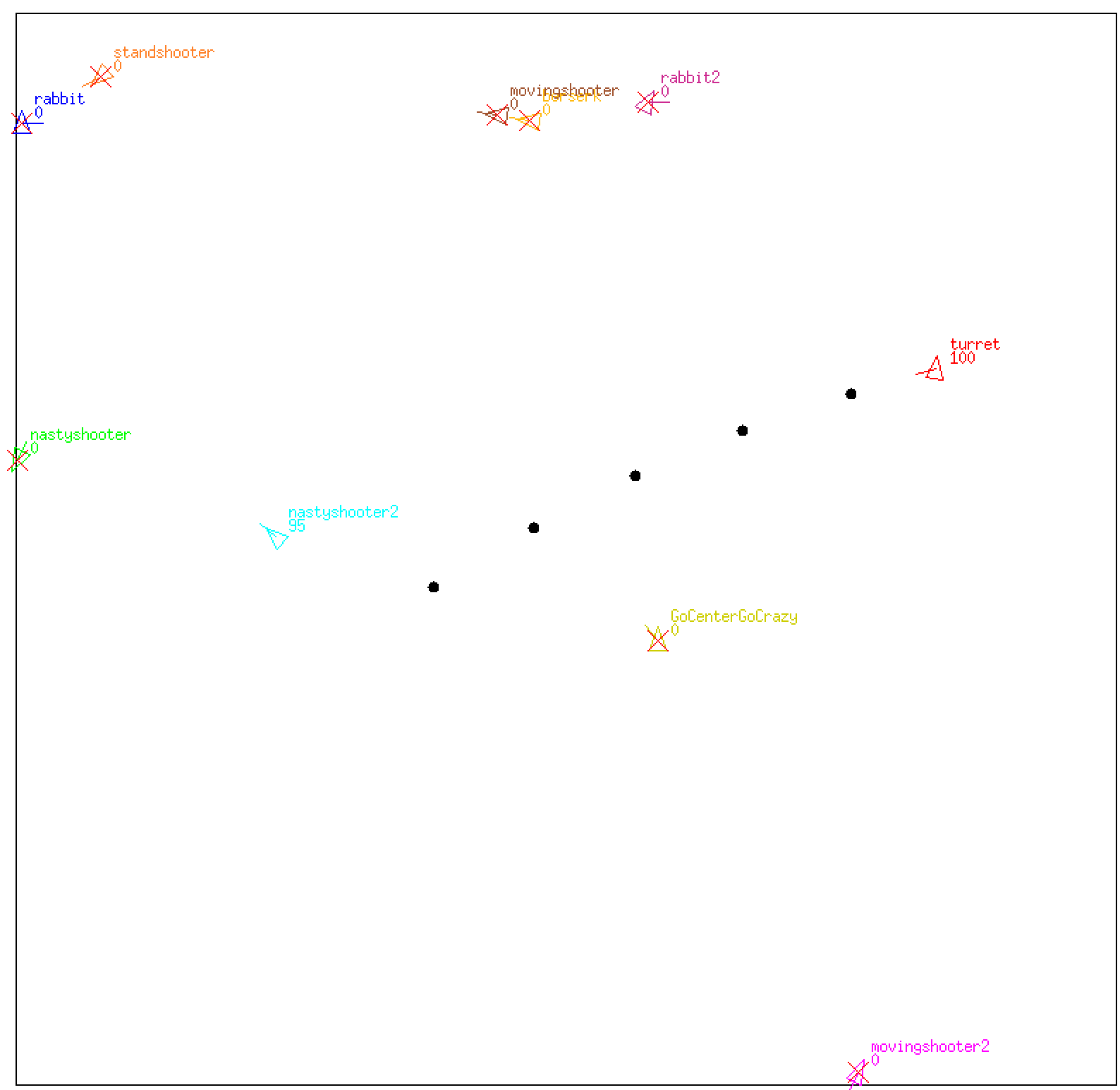
## 2.6 GUI Outputs

### 2.6.1 Text Status of Drones

The text part of the GUI shows total ticks each Drones runs as well as other information such as “Team ID”, “AI Ticks”, “Moving”, “Gun cooldown” and etc.

### 2.6.2 Arena GUI

Each Drone in the arena is displayed as a triangle with a “gun” on it, the direction of the “gun” shows in which direction the Drone is searching or shooting. And the direction of drone’s triangle shows where is it moving (or moved last if it is standing still right now). Bullets are displayed as black spots which moving faster than Drones. Once Bullet explodes, a “star” is displayed to show the range of damage. What’s more, on the top of each Drone, its name and health are displayed. Once Drones’ health becomes 0, there is a cross displayed over the Drone to show its death.



# Chapter 3: Reference Manual

## 3.1. Language Syntax

### 3.1.1 Keywords

Keywords used by the language are case insensitive (i.e Dup is the same as DUP or dup). The list of known keywords is:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| dup | drop | dropAll | swap | over | rot | read | store | jump | jumpIf |
| sub | endSub | move | stop | shoot | look | wait | getHealth | random | mod |
| isFoe | isAlly | isWall | and | or | not | if | else | endif |  |
| begin | while | again |  |  |  |  |  |  |  |

### 3.1.2 Player defined names

Unlike keywords (which are case insensitive), names defined by the player are case sensitive.

Those names are used as names for variables, labels, and user defined functions.

### 3.1.3 Comments

Single line comments, start with a word // and continue to the end of the line. E.g. each of the following lines contains a comment

// whole line can be a comment

2 2 + // or comment can start after some compilable words

// any word appeared after first // is still a comment

Multi line comments, start with a word /\* and continue to the first \*/ word. The nested comments are not supported.

/\* Inside here is

a comment \*/

### 3.1.4 Functions

#### 3.1.4.1 Structure of Function

User functions are marked with a word "Sub" followed by a function name, any number of commands and ends with "EndSub". It is not allowed to redefine any function or having a function inside a function.

Sub foo // correctly defined function

these words a body of a function

EndSub

Sub foo // this is an error: function redefinition

another words

EndSub

Sub foo 1 2 /\* correctly defined function,

words 1 and 2 are body of the function \*/

Sub bar // error: sub functions are not allowed

3 4

EndSub

EndSub

Sub myAdd + EndSub // correctly defined function

#### 3.1.4.2 Call of Function

The call to the user defined function is just its name. E.g. assuming we defined the function ‘myAdd’ as in the previous chapter, then the next two lines will do exactly the same:

2 2 +

2 2 myAdd

### 3.1.5 Label

#### 3.1.5.1 Structure of Label

Labels start with a letter followed by any number of letters, numbers, and ‘\_’ (underscore) symbols. Labels ended with a colon:

this\_is\_label:

this-is/not\_a.label:

123456: // also not a label

Of course, the white-space character split sequences of characters into sequence of words and the next line will be understood as four words and a label with the name 'label':

this is not a label:

#### 3.1.5.2 Unconditional and conditional jumps to the label

Operation "unconditional jump to the label" is marked by adding, "jump" to the name. The next line shows an unconditional jump to the labels defined in the previous example:

this\_is\_label jump

Conditional jump (marked jumpIf) checks the top of the stack first, if there was a true value, then the jump happens, if there was a false value, then jump does not happen and the execution is passed to the next operation after jumpif.

#### 3.1.5.3 Local & Global Labels

Label visibility is restricted to the function. For example:

Sub foo

2

lbl1: 2 +

lbl1 Jump // ok

lbl2 Jump // error

EndSub

lbl2: lbl1 JumpIf // error

Here, label *lbl1* is defined inside a function *foo* and jump to it is allowed. The label *lbl2* is defined in the main program and jump to it is allowed from anywhere from the main program, but not from the inside of user defined function. Conversely, the conditional jump to *lbl1* will fail since the label is defined inside of the function, but the jump is attempted from the main program.

## 3.2. Fundamental Types

### 3.2.1 Integer

Integer is word which consists solely from characters 0-9.

123 // one integer

1 2 3 // three integers

These words put the specified integer directly on the stack.

### 3.2.2 Boolean

Booleans are two words "true" and "false" which represent the logical values and are subject to logical operations and conditional jumps.

### 3.2.3 Flags

Flags are game specific type. They are produced by the function Look and explain what drone sees. There are four such flags: Foe, Ally, and Wall.

## 3.3. Variables

Variables are words started with a letter and any number of letters, digits and underscore symbols that directly followed by keywords "store" or "read". The first one takes the top of the stack and stores it into the variable (creating the variable in the process if necessary). The second one reads variable and puts its contents on the stack. E.g.

2 abc store

In this example, we assign 2 to a variable *abc* so that we can use it in the future.

abc read

Get the content of variable *abc* and push it into the stack. In this example, we push 2 to the stack because we assigned 2 to *abc* in the previous example.

Variables can contain any of the three fundamental types: integer, boolean or flag.

## 3.4. Operators

Operators are always taking some number of values from the stack and return some values back on the stack:

In the next examples, the top of the stack is considered to be on the left and the $ word symbolizes the end of stack

### 3.4.1 Arithmetic operators

+ b a $ -> (a + b) $

- b a $ -> (a - b) $

\* b a $ -> (a \* b) $

/ b a $ -> (a / b) $

mod b a $ -> (a mod b) $

^ b a $ -> (a ^ b) $

### 3.4.2 Logic operators

and b a $ -> (a and b) $

or b a $ -> (a or b) $

not a $ -> (not a) $

### 3.4.3 Logic constants

true $ -> true $

false $ -> false $

### 3.4.4 Conditions

= b a $ -> (a = b) $

< b a $ -> (a < b) $

> b a $ -> (a > b) $

### 3.4.5 Variable manipulation

name store a $ -> $

Store value into variable "name", create the variable if necessary. Always read the first on the stack and value it to “name”.

name read $ -> a $

Read value from variable "name". Die if such variable does not exist.

### 3.4.6 Stack manipulation

drop c b a $ -> b a $

dropall c b a $ -> $

dup c b a $ -> c c b a $

swap c b a $ -> b c a $

over c b a $ -> b c b a $

rot c b a $ -> a c b $

## 3.5. Game specific functions

### 3.5.1 Move

move direction $ -> $

Start moving in the specified direction

### 3.5.2 Stop

stop $ -> $

Stop moving

### 3.5.3 Shoot

shoot direction distance $ -> bool $

Shoot in the specified direction and distance. This function returns boolean value:

***true***  ***->*** shooting was successful and projectile is on its way

***false ->*** cannon did not have enough time to cool-down

### 3.5.4 Look

look direction $ -> flag1 dir1 dist1 ... WALL dir dist $

Look for other drones and walls in the specified direction. The function returns one or more triplets: type of the object, exact direction to it, and distance to the object. Type of the object is a flag from the set: FOE, ALLY, or WALL.

The WALL triplet is always the last one, so it can be used to detect an end of the look’s output.

### 3.5.5 isFoe

isFoe flag $ -> bool $

Checks is the top of the stack contains a flag FOE and returns corresponding boolean value.

### 3.5.6 isAlly

isAlly flag $ -> bool $

Checks is the top of the stack contains a flag ALLY and returns corresponding boolean value.

### 3.5.7 isWall

isWall flag $ -> bool $

Checks is the top of the stack contains a flag WALL and returns corresponding boolean value.

### 3.5.8 wait

wait ticks $ -> $

Be idle (do nothing) for specified number of ticks

### 3.5.9 getHealth

getHealth $ -> health $

Put current drone's health on the stack

### 3.5.10 random

random b a $ -> integer $

Make a random integer in the range [a,b] (inclusive) and return it.

## 3.6. Pseudo-commands

All operators and game-specific commands described in sections 4 and 5 take exactly are executed directly by the game engine and take one tick to perform. The next set of commands added for convenience. They are compiled by the translator into several simple operators and can take any number of additional ticks to complete.

### 3.6.1 Conditions

Conditional branching is done by the means IF/ELSE/ENDIF. The stack should contain a Boolean value before the IF. If this value is true, then the set of command which follows the IF would be executed. If the value is false, then control jumps to the set of commands after the keyword ELSE, or to the command which follows ENDIF, if the ELSE keyword is omitted. Nested IF branching is allowed. For example, shoot if the top of the stack contains the description of the enemy drone

isFoe if shoot endif

This code will be transformed by compiler into:

isFoe not endif\_label jumpIf shoot endif\_label:

### 3.6.2 Loops

#### 3.6.2.1 Endless loop

The endless loop is the most simple one, it is defined by keywords BEGIN and AGAIN:

begin 100 500 random 0 360 random shoot again

This will make the drone to shoot endlessly to a random distance in a random direction. This code is converted into a simple:

L1: 100 500 random 0 360 random shoot L1 jump

#### 3.6.2.2 Conditional loop

Conditional loops are defined by the same BEGIN and AGAIN keywords. Addition of the WHILE keyword allows to leave the endless loop if top of the stack is false when execution reach the WHILE keyword. For example, the cleanup after the LOOK command can be like this:

begin isEnd while drop drop again

This is code will be compiled into

L1: isend L2 jumpif drop drop L1 jump L2:

The WHILE keyword can appear anywhere inside the BEGIN-AGAIN block, this allows to create loops with post-conditions or even with conditions in the middle of the block:

begin dup isfoe shoot endif isend while drop drop again

Both types of loops can be nested.

# Chapter 4: Drone-Basic

The Drone-Basic language was designed as an afterthought for the Drone War project. The language itself is based in a Visual Basic and tweaked to allow special, game-related operations and concepts.

The Drone-Basic mostly follows the syntax of Visual Basic: One statement per line of source code, several statements can be grouped inside one compound statement, the language is completely case-insensitive, only single-line comments started with apostrophe, user procedures and functions.

### 4.1 The conditional branching

There are three types of conditional compound statements:

IF condition THEN statement

IF conditions THEN

statements

END IF

IF conditions THEN

statements

ELSE

statements

END IF

### 4.2 The conditional loops

There are two ways to do a conditional loops, with pre-condition and post-condition. Both variants can use the keyword WHILE (continue the loop, while the condition is true) and UNTIL (continue the loop until the condition become true):

DO [WHILE | UNTIL] condition

statements

LOOP

DO

statements

LOOP [WHILE | UNTIL] condition

All types of the conditional loop accept the EXIT DO statement which ends the loop immediately without checking the loop condition.

### 4.3 The counted loop

Just a regular FOR loop:

FOR variable=a TO b [STEP c]

statements

NEXT

The FOR loop also can be ended with EXIT FOR statement.

### 4.4 User procedures and functions

The user-defined procedures are following the Visual Basic's syntax:

SUB name(parameters)

statements

END SUB

Calls to a procedures are done with a special keyword CALL:

CALL name(arguments)

Unlike Visual Basic, the keyword CALL and parenthesis after the procedure name are necessary.

The user-defined functions are also following the Visual Basic's syntax:

FUNCTION name(parameters)

statements

name = result

END FUNCTION

The function have to have at least one assignment statement where the function name acts as a variable.

Parameters of procedures and functions are local to the procedures they are defined in. But since those variables are still has global bindings - only tail recursion in user procedures or functions are allowed. The use of other types of recursion can result in unpredictable behavior.

### 4.5 Variables

Variables in Drone-Basic are integer only (with the exception for the records returned by STARTSCAN and NEXTSCAN function, see below).

### 4.6 The game-related functions

Most of Drone-Basic game-related features are following the syntax for calling user-defined procedures and functions:

The Drone-Basic has next set of procedures (which require a CALL to be called):

CALL SLEEP(ticks)

CALL MOVE(direction)

CALL STOP()

The SHOOT(direction, distance) operation can act as both function and procedure. In case of a function – it returns a boolean value and can be used inside any conditional expression (in the IF or WHILE/UNTIL loops). The true returned by the SHOOT means that the bullet was shot successfully, the false – gun is still reloading. If the SHOOT is called as procedure – we ignore the result of the SHOOT.

The GETHEALTH() and RANDOM(min, max) are regular functions which can be used in any arithmetic expression.

### 4.7 The search procedure

The search procedure in the Drone-Basic language is the farthest operation from the classical Visual Basic. It consists of two functions:

The search procedure is started by the call to a function STARTSCAN(direction). This returns an object of the scan-result type:

var = STARTSCAN(direction)

The var here is not a single variable, but a collection of variables which represent a closes object the drone saw in the given direction. The next object the drone saw can be accessed by calling a function:

var = NEXTSCAN()

The NEXTSCAN function can be called several times until the list of objects seen by the first STARTSCAN operation is exhausted. The extra calls to NEXTSCAN can result in a drone coma with “Nothing to store” explanation.

The object read by STARTSCAN and NEXTSCAN is actually a structure with several elements:

obj.DIRECTION Integer. The exact direction to the object

obj.DISTANCE Integer. The exact distance to the object.

obj.ISFOE Boolean. The object is a drone and belongs to one of the opposing teams

obj.ISALLY Boolean. The object is a drone and belongs to the same team as the drone itself.

obj.ISWALL Boolean. The object is a wall.

obj.ISEND Boolean. The last object in the list of objects. After receiving such object it is not allowed to call NEXTSCAN.

### 4.8 Comparison between the base Drone language and Drone-Basic

Both languages allow the full control over drones. The text of the program in Drone-Basic is a little easier to understand since it is a higher level language. But as a downside after compilation to the IR it produces a less efficient code. Also while using the Drone-Basic, programmer is unable to access some of convenient functions (like dropall) which result in a necessity to write an extra code.

Here is an example of the same algorithm written in both languages:

drones/drone.dt

0 direction store

main\_loop:

dropall

direction read look

isFoe

shootIt jumpif

direction read 10 + direction store

main\_loop jump

shootIt:

dup direction store

shoot

10 wait

main\_loop jump

drones/drone.dbt

direction=0

start:

drone = startScan(direction)

if drone.isfoe then

direction = drone.direction

call shoot(drone.direction, drone.distance)

call sleep(10)

else

direction = direction + 10

end if

do until drone.isWall

drone = nextScan()

loop

goto start

# Chapter 5: Project Plan

## 5.1 Process

### 5.1.1 Planning

To make a decent design of the project, our team decided to start planning process earlier right after we learnt what need to be included as parts of a language, which we believed was the key to success. So as to make continuous progress and good communications within the team, we first set up a short meeting after each lecture, and could share the latest updates of the progress and discuss about what to do in the next few days. Besides, we set up Google Code with SVN so as to keep all source code in good shape and up to date for everybody in the team.

For the topic of the project, our team first agreed on the designing rules, which were “Interesting, Simple, and Efficient”.

### 5.1.2 Specification

An advantage of our team, was that we have an experienced leader who can always give advice, indicate what to do. After discussing about several ideas, we agreed on that a programming game would be attractive to most people and a stack-based language for it would fit our design rules best. Since a stack based language is easy to use and any user who knows nothing about programming would be able to make his or her own AI programs.

After handing in the proposal, we got help both from team leader and MICRO-C example of how to create a language starting from building Scanner, Parser and Ast. Based on the basic design of our language, we successfully agreed on and finished designing details for our game part.

### 5.1.3 Development

Mentioned before, our team used SVN version control tool during the development process to manage and keep every team member update up to date. So as to make sure each component of the project works correctly, our team applied a waterfall approach, in which each component was implemented and tested properly before we moved on to next step. First of all, our leader gave us an overview description of the language outline by creating Scanner and AST tree for us. Based on the language keywords and other basic information designed by our team leader, we discussed about the future work and made sure nothing was missing.

During when, Professor indicated that our language was somehow too hard to be understood and might need more high level “meat” such as conditional execution, endless loops as well as conditional loops.

After adding compound statements mentioned by professor into the language, our team continued to work on the compiler part of our project, which includes processing of all bytecodes, storing variables and subs. Some changes and corrections to the language design was implemented as a result of a more close work on the actual compiler and attempts to write a working drone AI. Tests also got implemented in this part.

After language part has been tested, our team then moved on to the game engine part of our project. Built in functions such as *move*, *look* and *shoot* were created according to the game description. Drone and Bullet were converted into object-oriented classes each containing local variables inside. However, applying our language to a real game was the most challenging part since multiple errors in different perspectives could happen.

When the game part was tested thoroughly, we created GUI part to make the game more interesting and completed all functionalities.

At this time, together with the initial development of GUI, the compiler from Drone-Basic was added. It was intended as a template for adding other languages of different styles and paradigms into the Drone War game. Unfortunately we did not have enough time to complete them to any degree of testing.

### 5.1.4 Testing

Since our team implemented a waterfall approach in the development process, every component was tested thoroughly before we moved on to next part. We developed and tested the project in order of Scanner, Parser, Ast, Engine, Arena and GUI. After unit testings, we implemented integration testing on the whole project by creating testing Drones, and which contained all possible syntax of our language. Based on the behaviours of Drones, our language could be tested in a large scale. However, multiple bugs and errors did happen during both unit testing and integration testing. Everyone in the team took part in the testing part and fixing the bugs.

## 5.2 Programming Style Guide

### 5.2.1 General Programming Principles

During the early meetings, our team made agreements on the designing and programming on the whole project, which were “Interesting”, “Simple” and “Efficient”. Later, after we learnt that stack-base language would be the best choice for us, we then decided to use a waterfall approach to make development of this project. So as to keep everyone in the team concentrate on the main ideas of, we made some extra programming principles to help us continuously make progress, which included “keep testing everything”, “keep communication”, “using version control tool”, “mutual code review”, and “code documentation”. Based on these rules, our team could better understand work as well as communicate with each other in a smooth way.

### 5.2.2 Keep Testing Everything

To development process in a waterfall approach, our team needed to make sure every functionality and every single method had been tested correctly before we move on to next step. Thus, our team designed and created several methods for unit testing for each component. After the game was finished, a number of testing Drones were created so as to implement integration testing. We all believed that testing everything from some time to time can always lead to better products in the future.

### 5.2.3 Keep Communications

Decided in the first meeting, our team scheduled a short meeting after each lecture, which could not only share ideas from the lecture, but also discuss about progress in the project. As usual, our team leader separated works into pieces and assigned to every team member according to our timeline, and this was also good timing to solve difficulties met during the development.

Beside meeting, our team kept emailing everyone about the latest progress had been made as well as TO-DO works for other. One could always send new ideas or possible improvements about the project.

### 5.2.4 Using Version Control Tool

Recommended by Professor Edwards, version control tool was always the best choice for a team work in programming. However, we strongly agreed with him since SVN made great help in our project. Each team member’s code could be merged and pushed to server any time and no one needed to worry about losing or ruining latest code.

### 5.2.5 Mutual Code Review

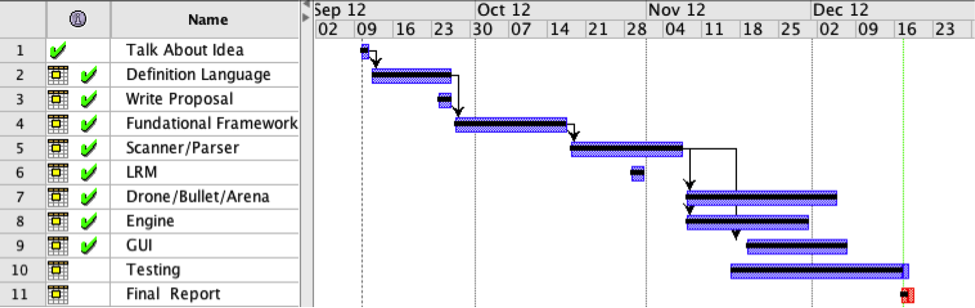
Another great principle of our team, once new progress was made, team leader as well as other team members would always do code review. Also, this was another way to reduce error and make improvement of the project.

### 5.2.6 Code Documentation & Comments

Because of the complexity of the functional language like OCaml it was always a problem to make other team member understand the code. And the fact that all team members were beginners to OCaml also hindered us a lot. As a solution mentioned by Professor Edwards, we always kept comments in the code up to date and wrote documentation for each new feature of the component. What’s more, multiple warning symbols were designed to let others know the information such as “something need to be improved”, “error might exists”, “TO-DO”, “not working” and etc.

## 5.3 Project Timeline

Major milestone and progress of the project is indicated as follow.



|  |  |
| --- | --- |
| Date | Milestone/Progress |
| 9/10/12 | Talk About Idea  Brainstorming |
| 9/26/12 | Language Definition  Syntax Definition  Game Definition |
| 9/29/12 | Language Proposal |
| 10/17/12 | Foundational Framework |
| 11/7/12 | Language Syntax & Semantic Analysis  Scanner  Parser  Abstract Syntax Tree |
| 11/15/12 | Engine Part Done  Drone  Arena  Bullet |
| 12/05/12 | GUI Part Started |

## 5.4 Project Log

The project log of our team is attached as follow, and please refer to SVN log for more detail information.

|  |  |
| --- | --- |
| Date | Milestone/Progress |
| 9/10/12 | Talk About Idea  Brainstorming |
| 9/12/12 | Decided on project topic  Of The Drone War |
| 9/16/12 | Agreed on a  Stack-based language |
| 9/24/12 | Started Game design  & Rules design |
| 9/26/12 | Language Definition  Syntax Definition  Game Definition |
| 9/29/12 | Language Proposal |
| 10/03/12 | First draft of Scanner, Parser  Created by team leader |
| 10/15/12 | AST tree generated |
| 10/17/12 | Foundational Framework |
| 10/22/12 | Studied MICRO-C compiler  New features to be considered  Based on it |
| 10/24/12 | Scanner, Parser and AST are finished  Start to work on Engine part |
| 10/29/12 | First draft of the Engine part byte code operations are finished;  Variables & Subs’ stacks and hash tables are created |
| 11/5/12 | Second draft of Engine is modified by team leader and multiple changes are implemented |
| 11/7/12 | Language Syntax & Semantic Analysis  Scanner  Parser  Abstract Syntax Tree tested and finished |
| 11/12/12 | Start to work on game helper functions and other classes such Bullet and Utils are created |
| 11/19/12 | Game helper functions are generated and first draft of the game is done |
| 11/21/12 | Several testing Drones are created and first integration testing implemented |
| 11/26/12 | Second draft of engine part is done |
| 12/5/12 | Engine Part done  Drone  Arena  Bullet |
| 12/10/12 | GUI Part Started |
| 12/14/12 | First draft of GUI-enabled game is done |
| 12/16/12 | GUI part is done and more testing drones are created to implement integration testing |
| 12/17/12 | Start to work on final report and final testing for the whole project |

## 5.5 Team Responsibility

As we have an experienced team leader, the project was separated into different components and each team member made contribution to it. After the project was finished, everyone implemented reports about his component in the representation slides as well as the final report. Here comes the assigned responsibilities of each team member:

|  |  |
| --- | --- |
| George | Team Leader  Created Scanner, Parser, AST  Engine parts’ features and improvement in every class  Code review and giving advice in every process of the project  Testing and modification in all perspectives  Unit testing & Integration testing  Basic-like translator (Individual) |
| Xiaotong | Parser: compound loops, conditional loops  Engine part: bytecode operations, variables & subs’ stack and hash table  Drones: objects variables, Drones operation functions  Arena: objects variables, Arena operation functions  Game: Bullet and Utils classes and other helper functions  Testing Drones created  Unit testing & Integration testing |
| Xiang | Parser: compound loops, conditional loops  Engine part: bytecode operations, variables & subs’ stack and hash table  Drones: objects variables, Drones operation functions  Arena: objects variables, Arena operation funcitons  Game: Bullet and Utils classes and other helper functions  Testing Drones created  Unit testing & Integration testing |
| Shuo | Parser: compound loops  Engine part: bytecode operations, debug  Drones: objects variables, Drones operation functions  Arena: objects variables, Arena operation funcitons  GUI: objects variables, Arena operation funcitons  Testing Drones created  Integration testing |

## 5.6 Development Environment

Operating systems: Mac OX, Windows XP, Windows 7, Linux Debian

Language: Objective Caml (OCaml)

Compiler: OCaml

Editors: Eclipse with OCaml Plugins, other various text editors

GUI: Ocaml Graphics Standard Library

Version Control: SVN, Google Code

Other tools: Google Docs, Emails,

# Chapter 6: Architecture Design

## 6.1 Design Overview

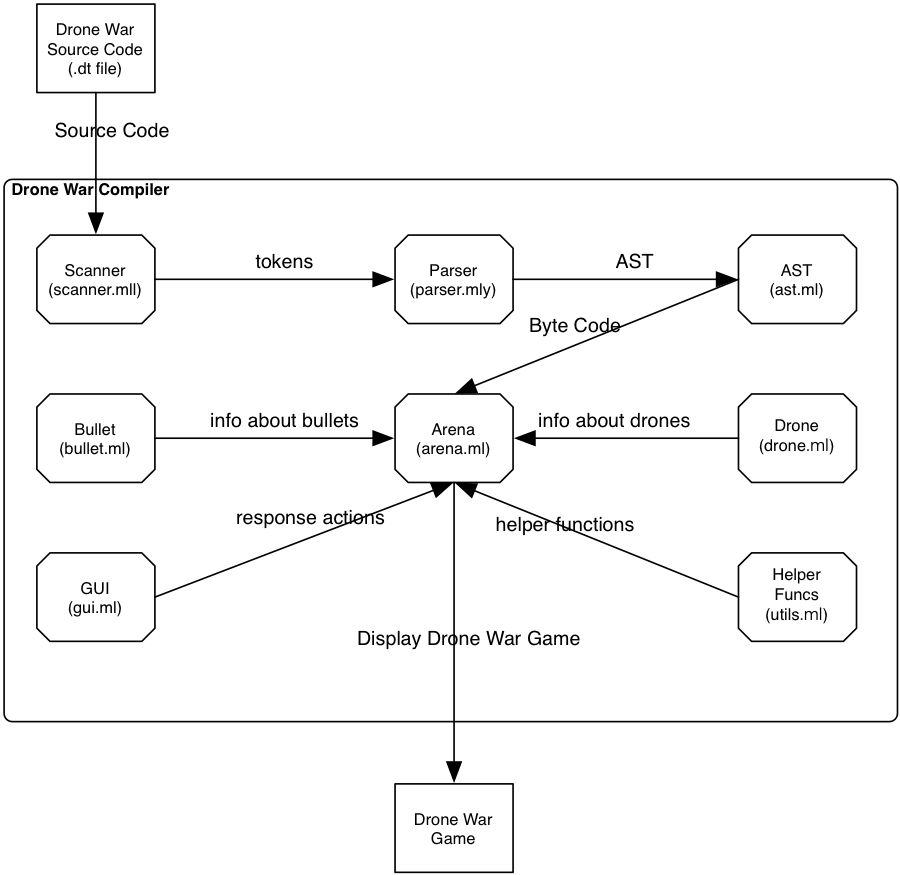
In our design, the Drone War project is composed of several components, which includes Syntax Analysis, Semantic Analysis, Compiler, Game Engine, GUI and other helper functions. When a qualified input file such as ***.dt*** or ***.dbt*** comes in, Compiler first links all needed files and sends it to Syntax Analysis part which contains the Scanner to get tokens. Secondly, tokens are passed to Semantics Analysis which contains Parser, so as to filter illegal tokens and store all necessary information like variables and subs. After that, arrays of bytecode are generated based on Abstract Syntax Tree and corresponding operations are to be implemented and stored in each Drone object’s stack. Thus, the Drone Language has successfully finished its work.

Followed by the language part, our Game Engine is going to take charge of all Drones in the arena. As controlled by arena object, all Drones will do exactly one tick in a single round, and all corresponding operations popped from the stack such as “look”, “move”, “shoot” will be implemented. After all Drones make a single tick, all updated information is to be stored. In this part all the operations are controlled by the arena and Game Engine functions from Drone.ml, Arena.ml, Bullet.ml and Utils.ml are called from time to time.

Once all updates to Arena’s objects (Drones or Bullets) are done, the Arena will call GUI part to visualize it on the graphic screen.

The last step in the main Arena’s loop is to check are there more than one Drone left alive? If not, then Arena considers that last Drone to be a winner and game ends. Another reason for exit from the main Arena’s loop is if total count of loops exceeds predefined constant. In that case, Arena presumes that the remaining live drones will not attack each other and the normal one-winner scenario is unachievable.

If Arena decided that the battle should continue, it again start requesting from the drones to execute one step of their AIs.



（Figure 5.1）

## 6.2 Interfaces Between the Components

### 6.2.1 Scanner (scanner.mll -Author: George)

The role of the scanner is to define what tokens are acceptable in our language. The scanner will go through the **.dt** file, which is our input source code, and recognize the stream of input file as tokens in our language or not. This component will convert all the input source code to tokens defined by us, so that it can reject the code that is not in the syntax of our language.

### 6.2.2 Parser (parser.mly -Author: George, Xiaotong, Xiang)

The role of the parser is to catch the tokens generated by the scanner. Although, we are sure these tokens are defined in our language, we still need to make sure these tokens together are meaningful, that is they can construct the AST defined by us. If these tokens don’t satisfy our grammar, parser will reject them。

### 6.2.3 AST (ast.ml -Author: George)

The role of AST is to define the structure of a program in our language. As stated above, during the execution of parser, it will check AST to see if the input tokens are valid or not. That is, the AST will be built during the parser. Also, from this component, we can get a list of bytecode defined in our language.

### 6.2.4 Arena (arena.ml -Author: George, Xiaotong, Xiang, Shuo)

Arena is the synchronizing piece of the game engine. It keeps track of all game objects (drones and bullets) and prompts them to do their assigned roles. The Drone object can request a creation of a Bullet object (shooting) and request the relative position of other drones (looking). These requests (as well as destruction of Bullets) are satisfied by the Arena.

Also Arena calls the GUI to show current state of the battle.

### 6.2.5 Drone (drone.ml -Author: George, Xiaotong, Xiang, Shuo)

As said in Arena, each drone is an “object”. It has many attributes like position, health, team and so on. The most important part of drone is that it contains two hash tables: *vars* to store the variables and *subs* to store the functions including the “main” (arrays of bytecode). This component is one of the most important parts of the engine. It includes many basic actions as well as the actual “drone’s CPU” which process the bytecodes.

### 6.2.6 Bullet (bullet.ml -Author: Xiaotong, Xiang)

Bullet is also a part of engine to represent the object of bullet. Compared with drone.ml, bullet.ml is very simple, it contains only the most basic information such as the position and the direction of a bullet.

### 6.2.7 GUI (gui.ml -Author: Shuo, George)

The role of GUI is to display the state of the game. This component is implemented by using the Graphics module in OCaml.

### 6.2.8 Helper Funcs (utils.ml -Author: Xiaotong, Xiang)

Utils.ml is a helper file for common functions. It contains several functions that can be called by arena and drone, such as the one help calculate the distance between two point, represented by (X,Y), common constants and such.

### 6.2.9 Drone-Basic (scanner\_dbt.mll and parser\_dbt.mly –Author: George)

Implementation of the Drone-Basic language.

# Chapter 7: Test Plan

## 7.1 Unit Testing

Mentioned above as a testing functionality, while compiling the Drone language, we provide a debug mode, in which two extra debugging files are created. ***<filename>.dt.decompiled*** is the file shows what bytecode is generated based on input file, while ***<filename>.dt.debug*** shows everything in the stack for each step. The unit testing cases for each component goes as followed:

### 7.1.1 Integer

test case:

1 2 3 4 5 6

generated byte code:

0: Int(1)

1: Int(2)

2: Int(3)

3: Int(4)

4: Int(5)

5: Int(6)

stack:

1 [ 0] Int(1) | EOS

2 [ 1] Int(2) | 1 EOS

3 [ 2] Int(3) | 2 1 EOS

4 [ 3] Int(4) | 3 2 1 EOS

5 [ 4] Int(5) | 4 3 2 1 EOS

6 [ 5] Int(6) | 5 4 3 2 1 EOS

result: successfully recognized the input integer.

### 7.1.2 Comment

test case:

6 // This is a comment 7

8

/\* this is also

9 a comment \*/

10

generated byte code:

0: Int(6)

1: Int(8)

2: Int(10)

stack:

1 [ 0] Int(6) | EOS

2 [ 1] Int(8) | 6 EOS

3 [ 2] Int(10) | 8 6 EOS

result: the comment parts have been ignored by the compiler.

### 7.1.3 Variables

test case:

36 var1 store

3 var1 read

generated byte code:

0: Int(36)

1: Store(var1)

2: Int(3)

3: Read(var1)

stack:

1 [ 0] Int(36) | EOS

2 [ 1] Store(var1) | 36 EOS

3 [ 2] Int(3) | EOS

4 [ 3] Read(var1) | 3 EOS

result: successfully set and get the value of a variable.

### 7.1.4 Arithmetic operators

test case:

1 2 + 1 - 1 \* 2 / 2 mod 4 ^

generated byte code:

0: Int(1)

1: Int(2)

2: Plus

3: Int(1)

4: Minus

5: Int(1)

6: Times

7: Int(2)

8: Divide

9: Int(2)

10: Mod

11: Int(4)

12: Power

stack:

1 [ 0] Int(1) | EOS

2 [ 1] Int(2) | 1 EOS

3 [ 2] Plus | 2 1 EOS

4 [ 3] Int(1) | 3 EOS

5 [ 4] Minus | 1 3 EOS

6 [ 5] Int(1) | 2 EOS

7 [ 6] Times | 1 2 EOS

8 [ 7] Int(2) | 2 EOS

9 [ 8] Divide | 2 2 EOS

10 [ 9] Int(2) | 1 EOS

11 [ 10] Mod | 2 1 EOS

12 [ 11] Int(4) | 1 EOS

13 [ 12] Power | 4 1 EOS

result: successfully calculate the result of arithmetic expression.

### 7.1.5 Logic constants

test case:

true

drop

false

drop

generated byte code:

0: Bool(true)

1: Drop

2: Bool(false)

3: Drop

stack:

1 [ 0] Bool(true) | EOS

2 [ 1] Drop | true EOS

3 [ 2] Bool(false) | EOS

4 [ 3] Drop | false EOS

result: successfully store or drop a bollean.

### 7.1.6 Logic operators

test case:

true true and

drop

true false and

drop

false false and

drop

true true or

drop

true false or

drop

false false or

drop

true not

drop

false not

drop

generated byte code:

0: Bool(true)

1: Bool(true)

2: And

3: Drop

4: Bool(true)

5: Bool(false)

6: And

7: Drop

8: Bool(false)

9: Bool(false)

10: And

11: Drop

12: Bool(true)

13: Bool(true)

14: Or

15: Drop

16: Bool(true)

17: Bool(false)

18: Or

19: Drop

20: Bool(false)

21: Bool(false)

22: Or

23: Drop

24: Bool(true)

25: Not

26: Drop

27: Bool(false)

28: Not

29: Drop

stack:

1 [ 0] Bool(true) | EOS

2 [ 1] Bool(true) | true EOS

3 [ 2] And | true true EOS

4 [ 3] Drop | true EOS

5 [ 4] Bool(true) | EOS

6 [ 5] Bool(false) | true EOS

7 [ 6] And | false true EOS

8 [ 7] Drop | false EOS

9 [ 8] Bool(false) | EOS

10 [ 9] Bool(false) | false EOS

11 [ 10] And | false false EOS

12 [ 11] Drop | false EOS

13 [ 12] Bool(true) | EOS

14 [ 13] Bool(true) | true EOS

15 [ 14] Or | true true EOS

16 [ 15] Drop | true EOS

17 [ 16] Bool(true) | EOS

18 [ 17] Bool(false) | true EOS

19 [ 18] Or | false true EOS

20 [ 19] Drop | true EOS

21 [ 20] Bool(false) | EOS

22 [ 21] Bool(false) | false EOS

23 [ 22] Or | false false EOS

24 [ 23] Drop | false EOS

25 [ 24] Bool(true) | EOS

26 [ 25] Not | true EOS

27 [ 26] Drop | false EOS

28 [ 27] Bool(false) | EOS

29 [ 28] Not | false EOS

30 [ 29] Drop | true EOS

result: successfully calculate the result of logic expression.

### 7.1.7 Stack manipulation

test case:

1 drop

1 2 3 dropall

1 dup

dropall

1 2 swap

dropall

1 2 3 over

dropall

1 2 3 rot

dropall

generated byte code:

0: Int(1)

1: Drop

2: Int(1)

3: Int(2)

4: Int(3)

5: Dropall

6: Int(1)

7: Dup

8: Dropall

9: Int(1)

10: Int(2)

11: Swap

12: Dropall

13: Int(1)

14: Int(2)

15: Int(3)

16: Over

17: Dropall

18: Int(1)

19: Int(2)

20: Int(3)

21: Rot

22: Dropall

stack:

1 [ 0] Int(1) | EOS

2 [ 1] Drop | 1 EOS

3 [ 2] Int(1) | EOS

4 [ 3] Int(2) | 1 EOS

5 [ 4] Int(3) | 2 1 EOS

6 [ 5] Dropall | 3 2 1 EOS

7 [ 6] Int(1) | EOS

8 [ 7] Dup | 1 EOS

9 [ 8] Dropall | 1 1 EOS

10 [ 9] Int(1) | EOS

11 [ 10] Int(2) | 1 EOS

12 [ 11] Swap | 2 1 EOS

13 [ 12] Dropall | 1 2 EOS

14 [ 13] Int(1) | EOS

15 [ 14] Int(2) | 1 EOS

16 [ 15] Int(3) | 2 1 EOS

17 [ 16] Over | 3 2 1 EOS

18 [ 17] Dropall | 3 2 2 1 EOS

19 [ 18] Int(1) | EOS

20 [ 19] Int(2) | 1 EOS

21 [ 20] Int(3) | 2 1 EOS

22 [ 21] Rot | 3 2 1 EOS

23 [ 22] Dropall | 2 3 2 2 1 EOS

result: successfully manipulate the stack by variable operators.

### 7.1.8 Function

test case:

2 2 foo

sub foo

2

2

+

endsub

generated byte code:

0: Int(2)

1: Int(2)

2: Call(foo)

sub foo

0: Int(2)

1: Int(2)

2: Plus

esub

stack:

1 [ 0] Int(2) | EOS

2 [ 1] Int(2) | 2 EOS

3 [ 2] Call(foo) | 2 2 EOS

4 foo[ 0] Int(2) | 2 2 EOS

5 foo[ 1] Int(2) | 2 2 2 EOS

6 foo[ 2] Plus | 2 2 2 2 EOS

result: successfully call the function.

### 7.1.9 Label

test case:

label:

main:

this\_is\_a\_label:

generated byte code:

-- nothing

stack:

-- nothing

### 7.1.10 Move

test case:

45 move

generated byte code:

0: Int(45)

1: Move

stack:

1 [ 0] Int(45) | EOS

2 [ 1] Move | 45 EOS

result: successfully move to direction 45.

### 7.1.11 Stop

test case:

45 move

stop

generated byte code:

0: Int(45)

1: Move

2: Stop

stack:

1 [ 0] Int(45) | EOS

2 [ 1] Move | 45 EOS

3 [ 2] Stop | EOS

result: stop moving.

### 7.1.12 Shoot

test case:

45 100 shoot

generated byte code:

0: Int(45)

1: Int(100)

2: Shoot

stack:

1 [ 0] Int(45) | EOS

2 [ 1] Int(100) | 45 EOS

3 [ 2] Shoot | 100 45 EOS

result: shoot to the direction of 45 and distance of 100.

### 7.1.13 Look

test case:

180 look

generated byte code:

0: Int(180)

1: Look

stack:

1 [ 0] Int(180) | EOS

2 [ 1] Look | 180 EOS

result: successfully return a list of Flags.

### 7.1.14 isFoe

test case:

100 look

isFoe

generated byte code:

0: Int(100)

1: Look

2: IsFoe

stack:

1 [ 0] Int(100) | EOS

2 [ 1] Look | 100 EOS

3 [ 2] IsFoe | Foe 71 252 Foe 121 426 Wall 100 416 EOS

result: successfully identify the FOE.

### 7.1.15 isAlly

test case:

100 look

isAlly

generated byte code:

0: Int(100)

1: Look

2: IsAlly

stack:

1 [ 0] Int(100) | EOS

2 [ 1] Look | 100 EOS

3 [ 2] IsAlly | Foe 71 252 Foe 121 426 Wall 100 416 EOS

result: successfully identify the Ally.

### 7.1.16 isWall

test case:

100 look

isWall

generated byte code:

0: Int(100)

1: Look

2: IsWall

stack:

1 [ 0] Int(100) | EOS

2 [ 1] Look | 100 EOS

3 [ 2] IsWall | Wall 100 108 EOS

result: successfully identify the Wall.

### 7.1.17 Wait

test case:

10 wait

generated byte code:

0: Int(10)

1: Wait

stack:

1 [ 0] Int(10) | EOS

2 [ 1] Wait | 10 EOS

3 waiting for 10 ticks

4 waiting for 9 ticks

5 waiting for 8 ticks

6 waiting for 7 ticks

7 waiting for 6 ticks

8 waiting for 5 ticks

9 waiting for 4 ticks

10 waiting for 3 ticks

11 waiting for 2 ticks

12 waiting for 1 ticks

result: this drone will be hang up.

### 7.1.18 GetHealth

test case:

100 health store

health read getHealth =

generated byte code:

0: Int(100)

1: Store(health)

2: Read(health)

3: GetHealth

4: Equal

stack:

1 [ 0] Int(100) | EOS

2 [ 1] Store(health) | 100 EOS

3 [ 2] Read(health) | EOS

4 [ 3] GetHealth | 100 EOS

5 [ 4] Equal | 100 100 EOS

result: successfully get the health of the drone.

### 7.1.19 Random

test case:

1 100 random

generated byte code:

0: Int(1)

1: Int(100)

2: Random

stack:

1 [ 0] Int(1) | EOS

2 [ 1] Int(100) | 1 EOS

3 [ 2] Random | 100 1 EOS

result: successfully generate a number between 1 and 100.

### 7.1.20 Endless Loop

test case:

begin

100 100 shoot

again

generated byte code:

0: Int(100)

1: Int(100)

2: Shoot

3: AbsJump(0)

stack:

1 [ 0] Int(100) | EOS

2 [ 1] Int(100) | 100 EOS

3 [ 2] Shoot | 100 100 EOS

4 [ 3] AbsJump(0) | true EOS

5 [ 0] Int(100) | true EOS

6 [ 1] Int(100) | 100 true EOS

7 [ 2] Shoot | 100 100 true EOS

8 [ 3] AbsJump(0) | false true EOS

9 [ 0] Int(100) | false true EOS

10 [ 1] Int(100) | 100 false true EOS

11 [ 2] Shoot | 100 100 false true EOS

12 [ 3] AbsJump(0) | false false true EOS

13 [ 0] Int(100) | false false true EOS

14 [ 1] Int(100) | 100 false false true EOS

15 [ 2] Shoot | 100 100 false false true EOS

16 [ 3] AbsJump(0) | true false false true EOS

17 [ 0] Int(100) | true false false true EOS

18 [ 1] Int(100) | 100 true false false true EOS

19 [ 2] Shoot | 100 100 true false false true EOS

20 [ 3] AbsJump(0) | false true false false true EOS

21 [ 0] Int(100) | false true false false true EOS

22 [ 1] Int(100) | 100 false true false false true EOS

23 [ 2] Shoot | 100 100 false true false false true EOS

24 [ 3] AbsJump(0) | false false true false false true EOS

25 [ 0] Int(100) | false false true false false true EOS

26 [ 1] Int(100) | 100 false false true false false true EOS

27 [ 2] Shoot | 100 100 false false true false false true EOS

28 [ 3] AbsJump(0) | true false false true false false true EOS

29 [ 0] Int(100) | true false false true false false true EOS

30 [ 1] Int(100) | 100 true false false true false false true EOS

31 [ 2] Shoot | 100 100 true false false true false false true EOS

32 [ 3] AbsJump(0) | false true false false true false false true EOS

33 [ 0] Int(100) | false true false false true false false true EOS

34 [ 1] Int(100) | 100 false true false false true false false true EOS

35 [ 2] Shoot | 100 100 false true false false true false false true EOS

36 [ 3] AbsJump(0) | false false true false false true false false true EOS

37 [ 0] Int(100) | false false true false false true false false true EOS

…

result: convert endless loop to jump and work correctly.

### 7.1.21 Conditional Loop

test case:

0 a store

begin

a read 1 + a store

a read 10 <

while

100 100 shoot

again

generated byte code:

0: Int(0)

1: Store(a)

2: Read(a)

3: Int(1)

4: Plus

5: Store(a)

6: Read(a)

7: Int(10)

8: Less

9: Not

10: AbsJumpIf(15)

11: Int(100)

12: Int(100)

13: Shoot

14: AbsJump(2)

stack:

1 [ 0] Int(0) | EOS

2 [ 1] Store(a) | 0 EOS

3 [ 2] Read(a) | EOS

4 [ 3] Int(1) | 0 EOS

5 [ 4] Plus | 1 0 EOS

6 [ 5] Store(a) | 1 EOS

7 [ 6] Read(a) | EOS

8 [ 7] Int(10) | 1 EOS

9 [ 8] Less | 10 1 EOS

10 [ 9] Not | true EOS

11 [ 10] AbsJumpIf(15) | false EOS

12 [ 11] Int(100) | EOS

13 [ 12] Int(100) | 100 EOS

14 [ 13] Shoot | 100 100 EOS

15 [ 14] AbsJump(2) | true EOS

16 [ 2] Read(a) | true EOS

17 [ 3] Int(1) | 1 true EOS

18 [ 4] Plus | 1 1 true EOS

19 [ 5] Store(a) | 2 true EOS

20 [ 6] Read(a) | true EOS

21 [ 7] Int(10) | 2 true EOS

22 [ 8] Less | 10 2 true EOS

23 [ 9] Not | true true EOS

24 [ 10] AbsJumpIf(15) | false true EOS

25 [ 11] Int(100) | true EOS

26 [ 12] Int(100) | 100 true EOS

27 [ 13] Shoot | 100 100 true EOS

28 [ 14] AbsJump(2) | true true EOS

29 [ 2] Read(a) | true true EOS

30 [ 3] Int(1) | 2 true true EOS

31 [ 4] Plus | 1 2 true true EOS

32 [ 5] Store(a) | 3 true true EOS

33 [ 6] Read(a) | true true EOS

34 [ 7] Int(10) | 3 true true EOS

35 [ 8] Less | 10 3 true true EOS

36 [ 9] Not | true true true EOS

37 [ 10] AbsJumpIf(15) | false true true EOS

38 [ 11] Int(100) | true true EOS

39 [ 12] Int(100) | 100 true true EOS

40 [ 13] Shoot | 100 100 true true EOS

41 [ 14] AbsJump(2) | true true true EOS

42 [ 2] Read(a) | true true true EOS

43 [ 3] Int(1) | 3 true true true EOS

44 [ 4] Plus | 1 3 true true true EOS

45 [ 5] Store(a) | 4 true true true EOS

46 [ 6] Read(a) | true true true EOS

47 [ 7] Int(10) | 4 true true true EOS

48 [ 8] Less | 10 4 true true true EOS

49 [ 9] Not | true true true true EOS

50 [ 10] AbsJumpIf(15) | false true true true EOS

51 [ 11] Int(100) | true true true EOS

52 [ 12] Int(100) | 100 true true true EOS

53 [ 13] Shoot | 100 100 true true true EOS

54 [ 14] AbsJump(2) | true true true true EOS

55 [ 2] Read(a) | true true true true EOS

56 [ 3] Int(1) | 4 true true true true EOS

57 [ 4] Plus | 1 4 true true true true EOS

58 [ 5] Store(a) | 5 true true true true EOS

59 [ 6] Read(a) | true true true true EOS

60 [ 7] Int(10) | 5 true true true true EOS

61 [ 8] Less | 10 5 true true true true EOS

62 [ 9] Not | true true true true true EOS

63 [ 10] AbsJumpIf(15) | false true true true true EOS

64 [ 11] Int(100) | true true true true EOS

65 [ 12] Int(100) | 100 true true true true EOS

66 [ 13] Shoot | 100 100 true true true true EOS

67 [ 14] AbsJump(2) | true true true true true EOS

68 [ 2] Read(a) | true true true true true EOS

69 [ 3] Int(1) | 5 true true true true true EOS

70 [ 4] Plus | 1 5 true true true true true EOS

71 [ 5] Store(a) | 6 true true true true true EOS

72 [ 6] Read(a) | true true true true true EOS

73 [ 7] Int(10) | 6 true true true true true EOS

74 [ 8] Less | 10 6 true true true true true EOS

75 [ 9] Not | true true true true true true EOS

result: convert conditional loop to jumpif and work correctly.

### 7.1.22 if

test case:

1 2 <

if

1

endif

generated byte code:

0: Int(1)

1: Int(2)

2: Less

3: Not

4: AbsJumpIf(6)

5: Int(1)

stack:

1 [ 0] Int(1) | EOS

2 [ 1] Int(2) | 1 EOS

3 [ 2] Less | 2 1 EOS

4 [ 3] Not | true EOS

5 [ 4] AbsJumpIf(6) | false EOS

6 [ 5] Int(1) | EOS

result: the if statement has been successfully convert to jumpif statement.

### 7.1.23 if-else

test case:

1 2 <

if

1

else

2

endif

generated byte code:

0: Int(1)

1: Int(2)

2: Less

3: Not

4: AbsJumpIf(7)

5: Int(1)

6: AbsJump(8)

7: Int(2)

stack:

1 [ 0] Int(1) | EOS

2 [ 1] Int(2) | 1 EOS

3 [ 2] Less | 2 1 EOS

4 [ 3] Not | true EOS

5 [ 4] AbsJumpIf(7) | false EOS

6 [ 5] Int(1) | EOS

7 [ 6] AbsJump(8) | 1 EOS

result: the if-else statement has been successfully convert to jumpif statement.

## 7.2 Integration Test: An Example Programs

To test the whole project thoroughly, we created a number of simple-minded Drones which can serve as examples for real players in creation of a really complicated AIs. In the example below, we show several drones written in the drone war language.

### 7.2.1 Drone Berserk

drones/berserk.dt

// This drone is very aggressive. It looks for any other drone,

// regardless of is it friend or foe, runs toward it and shoot.

0 direction store

main\_loop:

direction read look

// if drones sees a wall, that means it does not see

// any drone

isWall not sees\_a\_drone jumpif

stop // do not move if drone does not have a target

drop2 // if we sees a wall, then drop the distance

// to it (stack should be empty now)

0 360 random // get a random direction value

direction store // and the drone will look for the next

// target in this random direction

main\_loop jump

sees\_a\_drone:

dup direction store // store the direction to the drone

dup move // start moving toward the target

shoot // and shoot in the same direction

drop // ignore the result of shooting

// after charged to the nearest drone, we still

// have to cleanup data for other objects seen by look.

look\_cleanup:

swap drop // drop direction and

swap drop // distance to the next target

isWall main\_loop jumpif // if the last target was a wall

look\_cleanup jump // else repeat clean up process

// user function

// drop two values from the stack

sub drop2 drop drop endsub

This file will be complied into:

0: Int(0)

1: Store(direction)

2: Read(direction)

3: Look

4: IsWall

5: Not

6: AbsJumpIf(14)

7: Stop

8: Call(drop2)

9: Int(0)

10: Int(360)

11: Random

12: Store(direction)

13: AbsJump(2)

14: Dup

15: Store(direction)

16: Dup

17: Move

18: Shoot

19: Drop

20: Swap

21: Drop

22: Swap

23: Drop

24: IsWall

25: AbsJumpIf(2)

26: AbsJump(20)

sub drop2

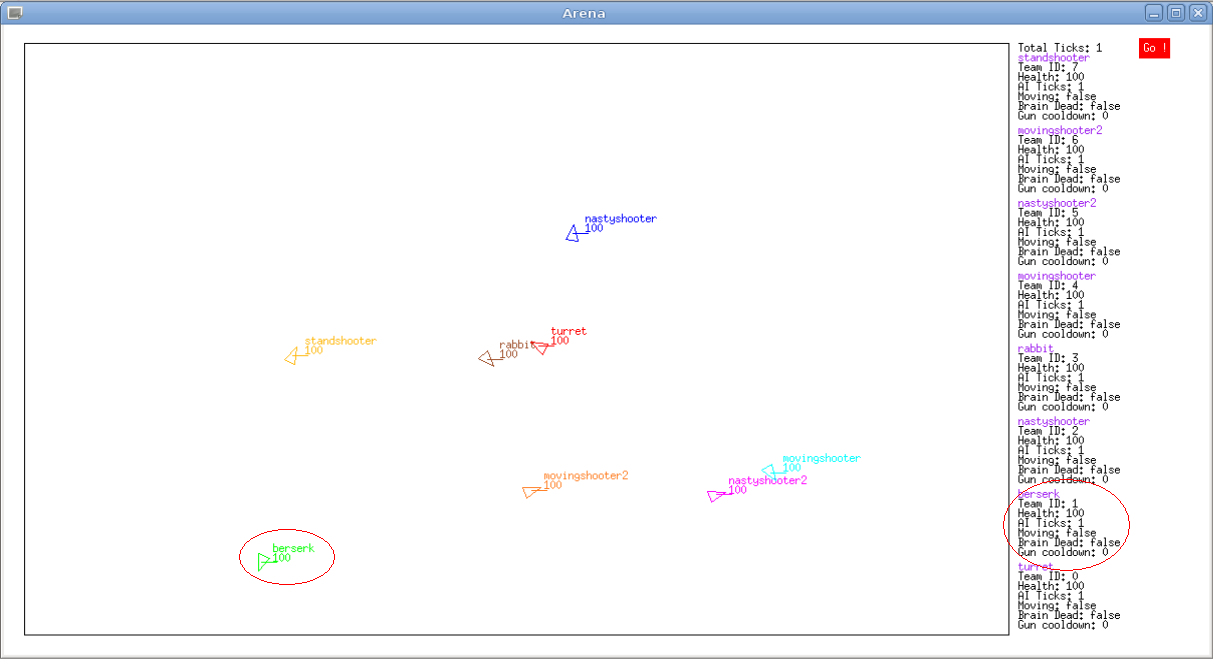
0: Drop

1: Drop

esub

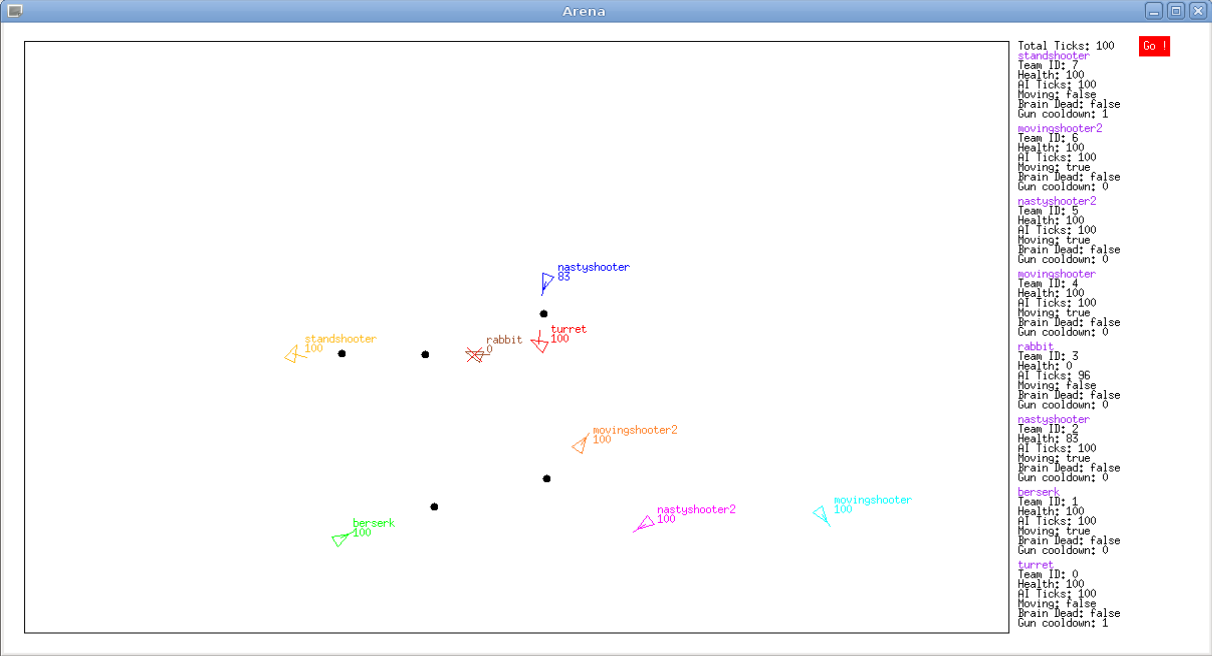
The running state of a game:

1. when it get started:



The stack:

1 [ 0] Int(0) | EOS



The stack:

1 [ 0] Int(0) | EOS

2 [ 1] Store(direction) | 0 EOS

3 [ 2] Read(direction) | EOS //set and get the original direction 0

4 [ 3] Look | 0 EOS //use look function to see the objects in direction

5 [ 4] IsWall | Foe 23 300 Foe 14 477 Foe 16 540 … //look result

6 [ 5] Not | false 23 300 Foe 14 477 Foe 16 540 … //find a drone

7 [ 6] AbsJumpIf(14) | true 23 300 Foe 14 477 Foe 16 540 … //jump

8 [ 14] Dup | 23 300 Foe 14 477 Foe 16 540 Wall … //duplicate

9 [ 15] Store(direction) | 23 23 300 Foe 14 477 Foe 16 540 … //store direction

10 [ 16] Dup | 23 300 Foe 14 477 Foe 16 540 Wall … //duplicate

11 [ 17] Move | 23 23 300 Foe 14 477 Foe 16 540 ...//move in direction

12 [ 18] Shoot | 23 300 Foe 14 477 Foe 16 540 Wall ...//shoot in directio

13 [ 19] Drop | true Foe 14 477 Foe 16 540 Wall 0 …//not care shoot result

14 [ 20] Swap | Foe 14 477 Foe 16 540 Wall 0 758 EOS

15 [ 21] Drop | 14 Foe 477 Foe 16 540 Wall 0 758 EOS

16 [ 22] Swap | Foe 477 Foe 16 540 Wall 0 758 EOS

17 [ 23] Drop | 477 Foe Foe 16 540 Wall 0 758 EOS

18 [ 24] IsWall | Foe Foe 16 540 Wall 0 758 EOS

19 [ 25] AbsJumpIf(2) | false Foe 16 540 Wall 0 758 EOS

20 [ 26] AbsJump(20) | Foe 16 540 Wall 0 758 EOS

21 [ 20] Swap | Foe 16 540 Wall 0 758 EOS

22 [ 21] Drop | 16 Foe 540 Wall 0 758 EOS

23 [ 22] Swap | Foe 540 Wall 0 758 EOS

24 [ 23] Drop | 540 Foe Wall 0 758 EOS

25 [ 24] IsWall | Foe Wall 0 758 EOS

26 [ 25] AbsJumpIf(2) | false Wall 0 758 EOS

27 [ 26] AbsJump(20) | Wall 0 758 EOS

28 [ 20] Swap | Wall 0 758 EOS

29 [ 21] Drop | 0 Wall 758 EOS

30 [ 22] Swap | Wall 758 EOS

31 [ 23] Drop | 758 Wall EOS

32 [ 24] IsWall | Wall EOS

33 [ 25] AbsJumpIf(2) | true EOS //check if we have drop all the look result

34 [ 2] Read(direction) | EOS //get the stored direction

35 [ 3] Look | 23 EOS //look at this direction

36 [ 4] IsWall | Foe 26 297 Foe 12 431 Foe 13 524 ...//find a drone

37 [ 5] Not | false 26 297 Foe 12 431 Foe 13 524 ...

38 [ 6] AbsJumpIf(14) | true 26 297 Foe 12 431 Foe 13 524 ...

39 [ 14] Dup | 26 297 Foe 12 431 Foe 13 524 Wall ...

40 [ 15]Store(direction) | 26 26 297 Foe 12 431 Foe 13 524 ...//store the new direction

41 [ 16] Dup | 26 297 Foe 12 431 Foe 13 524 Wall ...

42 [ 17] Move | 26 26 297 Foe 12 431 Foe 13 524 ...

43 [ 18] Shoot | 26 297 Foe 12 431 Foe 13 524 Wall ...

44 [ 19] Drop | true Foe 12 431 Foe 13 524 Wall 23 ...

45 [ 20] Swap | Foe 12 431 Foe 13 524 Wall 23 799 EOS

46 [ 21] Drop | 12 Foe 431 Foe 13 524 Wall 23 799 EOS

47 [ 22] Swap | Foe 431 Foe 13 524 Wall 23 799 EOS

48 [ 23] Drop | 431 Foe Foe 13 524 Wall 23 799 EOS

49 [ 24] IsWall | Foe Foe 13 524 Wall 23 799 EOS

50 [ 25] AbsJumpIf(2) | false Foe 13 524 Wall 23 799 EOS

51 [ 26] AbsJump(20) | Foe 13 524 Wall 23 799 EOS

52 [ 20] Swap | Foe 13 524 Wall 23 799 EOS

53 [ 21] Drop | 13 Foe 524 Wall 23 799 EOS

54 [ 22] Swap | Foe 524 Wall 23 799 EOS

55 [ 23] Drop | 524 Foe Wall 23 799 EOS

56 [ 24] IsWall | Foe Wall 23 799 EOS

57 [ 25] AbsJumpIf(2) | false Wall 23 799 EOS

58 [ 26] AbsJump(20) | Wall 23 799 EOS

59 [ 20] Swap | Wall 23 799 EOS

60 [ 21] Drop | 23 Wall 799 EOS

61 [ 22] Swap | Wall 799 EOS

62 [ 23] Drop | 799 Wall EOS

63 [ 24] IsWall | Wall EOS

64 [ 25] AbsJumpIf(2) | true EOS //check if we have drop all the look result

65 [ 2] Read(direction) | EOS//get the stored direction

66 [ 3] Look | 26 EOS //use look to see objects in direction

67 [ 4] IsWall | Foe 29 293 Foe 9 373 Foe 9 506 ...//look result

68 [ 5] Not | false 29 293 Foe 9 373 Foe 9 506 ...//find a drone

69 [ 6] AbsJumpIf(14) | true 29 293 Foe 9 373 Foe 9 506 ...//jump

70 [ 14] Dup | 29 293 Foe 9 373 Foe 9 506 Wall ...//duplicate

71 [ 15] Store(direction) | 29 29 293 Foe 9 373 Foe 9 506 ...//store direction

72 [ 16] Dup | 29 293 Foe 9 373 Foe 9 506 Wall ...//duplicate

73 [ 17] Move | 29 29 293 Foe 9 373 Foe 9 506 ...//move in direction

74 [ 18] Shoot | 29 293 Foe 9 373 Foe 9 506 Wall ...//shoot in directio

75 [ 19] Drop | true Foe 9 373 Foe 9 506 Wall 26 ...

76 [ 20] Swap | Foe 9 373 Foe 9 506 Wall 26 787 EOS

77 [ 21] Drop | 9 Foe 373 Foe 9 506 Wall 26 787 EOS

78 [ 22] Swap | Foe 373 Foe 9 506 Wall 26 787 EOS

79 [ 23] Drop | 373 Foe Foe 9 506 Wall 26 787 EOS

80 [ 24] IsWall | Foe Foe 9 506 Wall 26 787 EOS

81 [ 25] AbsJumpIf(2) | false Foe 9 506 Wall 26 787 EOS

82 [ 26] AbsJump(20) | Foe 9 506 Wall 26 787 EOS

83 [ 20] Swap | Foe 9 506 Wall 26 787 EOS

84 [ 21] Drop | 9 Foe 506 Wall 26 787 EOS

85 [ 22] Swap | Foe 506 Wall 26 787 EOS

86 [ 23] Drop | 506 Foe Wall 26 787 EOS

87 [ 24] IsWall | Foe Wall 26 787 EOS

88 [ 25] AbsJumpIf(2) | false Wall 26 787 EOS

89 [ 26] AbsJump(20) | Wall 26 787 EOS

90 [ 20] Swap | Wall 26 787 EOS

91 [ 21] Drop | 26 Wall 787 EOS

92 [ 22] Swap | Wall 787 EOS

93 [ 23] Drop | 787 Wall EOS

94 [ 24] IsWall | Wall EOS

95 [ 25] AbsJumpIf(2) | true EOS//check if we have drop all the look result

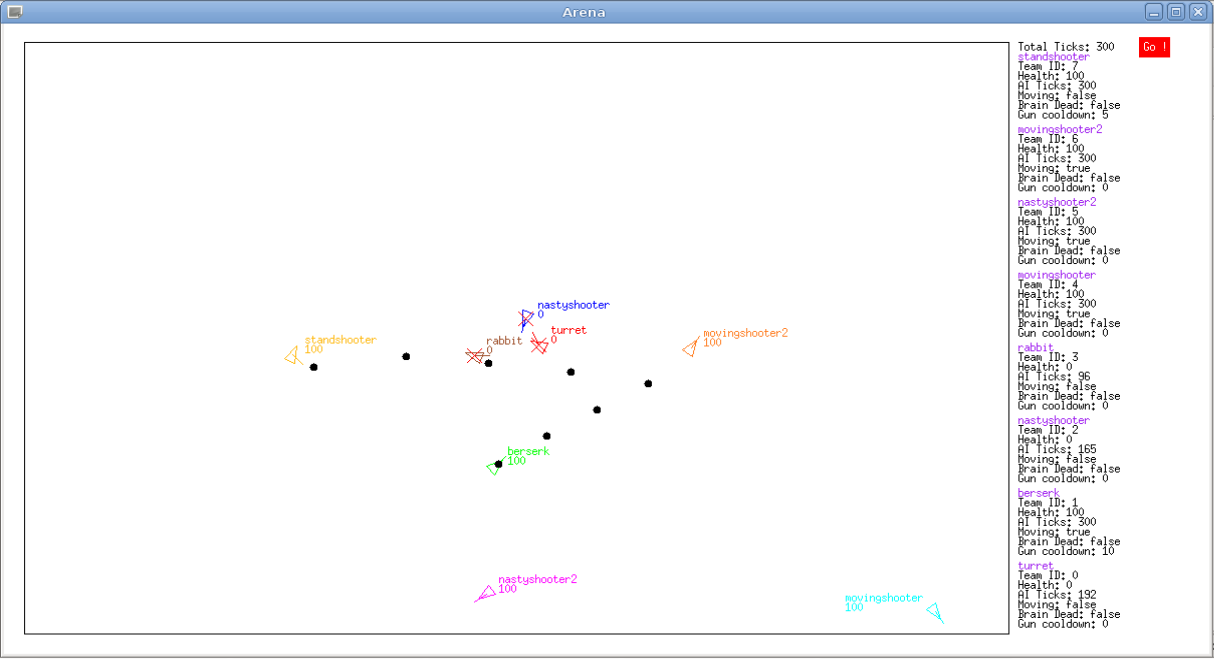
96 [ 2] Read(direction) | EOS//get the stored direction

97 [ 3] Look | 29 EOS

98 [ 4] IsWall | Foe 32 290 Foe 5 317 Foe 58 388 ...

99 [ 5] Not | false 32 290 Foe 5 317 Foe 58 388 ...

100 [ 6] AbsJumpIf(14) | true 32 290 Foe 5 317 Foe 58 388 ...



101 [ 14] Dup | 32 290 Foe 5 317 Foe 58 388 Foe ...

102 [ 15] Store(direction) | 32 32 290 Foe 5 317 Foe 58 388 ...

103 [ 16] Dup | 32 290 Foe 5 317 Foe 58 388 Foe ...

104 [ 17] Move | 32 32 290 Foe 5 317 Foe 58 388 ...

105 [ 18] Shoot | 32 290 Foe 5 317 Foe 58 388 Foe ...

106 [ 19] Drop | true Foe 5 317 Foe 58 388 Foe 5 ...

107 [ 20] Swap | Foe 5 317 Foe 58 388 Foe 5 492 ...

108 [ 21] Drop | 5 Foe 317 Foe 58 388 Foe 5 492 ...

109 [ 22] Swap | Foe 317 Foe 58 388 Foe 5 492 Wall ...

110 [ 23] Drop | 317 Foe Foe 58 388 Foe 5 492 Wall ...

111 [ 24] IsWall | Foe Foe 58 388 Foe 5 492 Wall 29 ...

112 [ 25] AbsJumpIf(2) | false Foe 58 388 Foe 5 492 Wall 29 ...

113 [ 26] AbsJump(20) | Foe 58 388 Foe 5 492 Wall 29 778 EOS

114 [ 20] Swap | Foe 58 388 Foe 5 492 Wall 29 778 EOS

115 [ 21] Drop | 58 Foe 388 Foe 5 492 Wall 29 778 EOS

116 [ 22] Swap | Foe 388 Foe 5 492 Wall 29 778 EOS

117 [ 23] Drop | 388 Foe Foe 5 492 Wall 29 778 EOS

118 [ 24] IsWall | Foe Foe 5 492 Wall 29 778 EOS

119 [ 25] AbsJumpIf(2) | false Foe 5 492 Wall 29 778 EOS

120 [ 26] AbsJump(20) | Foe 5 492 Wall 29 778 EOS

121 [ 20] Swap | Foe 5 492 Wall 29 778 EOS

122 [ 21] Drop | 5 Foe 492 Wall 29 778 EOS

123 [ 22] Swap | Foe 492 Wall 29 778 EOS

124 [ 23] Drop | 492 Foe Wall 29 778 EOS

125 [ 24] IsWall | Foe Wall 29 778 EOS

126 [ 25] AbsJumpIf(2) | false Wall 29 778 EOS

127 [ 26] AbsJump(20) | Wall 29 778 EOS

128 [ 20] Swap | Wall 29 778 EOS

129 [ 21] Drop | 29 Wall 778 EOS

130 [ 22] Swap | Wall 778 EOS

131 [ 23] Drop | 778 Wall EOS

132 [ 24] IsWall | Wall EOS

133 [ 25] AbsJumpIf(2) | true EOS

134 [ 2] Read(direction) | EOS

135 [ 3] Look | 32 EOS

136 [ 4] IsWall | Foe 35 287 Foe 60 354 Wall 32 764 EOS

137 [ 5] Not | false 35 287 Foe 60 354 Wall 32 764 EOS

138 [ 6] AbsJumpIf(14) | true 35 287 Foe 60 354 Wall 32 764 EOS

139 [ 14] Dup | 35 287 Foe 60 354 Wall 32 764 EOS

140 [ 15] Store(direction) | 35 35 287 Foe 60 354 Wall 32 764 EOS

141 [ 16] Dup | 35 287 Foe 60 354 Wall 32 764 EOS

142 [ 17] Move | 35 35 287 Foe 60 354 Wall 32 764 EOS

143 [ 18] Shoot | 35 287 Foe 60 354 Wall 32 764 EOS

144 [ 19] Drop | true Foe 60 354 Wall 32 764 EOS

145 [ 20] Swap | Foe 60 354 Wall 32 764 EOS

146 [ 21] Drop | 60 Foe 354 Wall 32 764 EOS

147 [ 22] Swap | Foe 354 Wall 32 764 EOS

148 [ 23] Drop | 354 Foe Wall 32 764 EOS

149 [ 24] IsWall | Foe Wall 32 764 EOS

150 [ 25] AbsJumpIf(2) | false Wall 32 764 EOS

151 [ 26] AbsJump(20) | Wall 32 764 EOS

152 [ 20] Swap | Wall 32 764 EOS

153 [ 21] Drop | 32 Wall 764 EOS

154 [ 22] Swap | Wall 764 EOS

155 [ 23] Drop | 764 Wall EOS

156 [ 24] IsWall | Wall EOS

157 [ 25] AbsJumpIf(2) | true EOS

158 [ 2] Read(direction) | EOS

159 [ 3] Look | 35 EOS

160 [ 4] IsWall | Foe 37 286 Foe 62 333 Wall 35 767 EOS

161 [ 5] Not | false 37 286 Foe 62 333 Wall 35 767 EOS

162 [ 6] AbsJumpIf(14) | true 37 286 Foe 62 333 Wall 35 767 EOS

163 [ 14] Dup | 37 286 Foe 62 333 Wall 35 767 EOS

164 [ 15] Store(direction) | 37 37 286 Foe 62 333 Wall 35 767 EOS

165 [ 16] Dup | 37 286 Foe 62 333 Wall 35 767 EOS

166 [ 17] Move | 37 37 286 Foe 62 333 Wall 35 767 EOS

167 [ 18] Shoot | 37 286 Foe 62 333 Wall 35 767 EOS

168 [ 19] Drop | true Foe 62 333 Wall 35 767 EOS

169 [ 20] Swap | Foe 62 333 Wall 35 767 EOS

170 [ 21] Drop | 62 Foe 333 Wall 35 767 EOS

171 [ 22] Swap | Foe 333 Wall 35 767 EOS

172 [ 23] Drop | 333 Foe Wall 35 767 EOS

173 [ 24] IsWall | Foe Wall 35 767 EOS

174 [ 25] AbsJumpIf(2) | false Wall 35 767 EOS

175 [ 26] AbsJump(20) | Wall 35 767 EOS

176 [ 20] Swap | Wall 35 767 EOS

177 [ 21] Drop | 35 Wall 767 EOS

178 [ 22] Swap | Wall 767 EOS

179 [ 23] Drop | 767 Wall EOS

180 [ 24] IsWall | Wall EOS

181 [ 25] AbsJumpIf(2) | true EOS

182 [ 2] Read(direction) | EOS

183 [ 3] Look | 37 EOS

184 [ 4] IsWall | Foe 39 285 Foe 64 312 Wall 37 762 EOS

185 [ 5] Not | false 39 285 Foe 64 312 Wall 37 762 EOS

186 [ 6] AbsJumpIf(14) | true 39 285 Foe 64 312 Wall 37 762 EOS

187 [ 14] Dup | 39 285 Foe 64 312 Wall 37 762 EOS

188 [ 15] Store(direction) | 39 39 285 Foe 64 312 Wall 37 762 EOS

189 [ 16] Dup | 39 285 Foe 64 312 Wall 37 762 EOS

190 [ 17] Move | 39 39 285 Foe 64 312 Wall 37 762 EOS

191 [ 18] Shoot | 39 285 Foe 64 312 Wall 37 762 EOS

192 [ 19] Drop | true Foe 64 312 Wall 37 762 EOS

193 [ 20] Swap | Foe 64 312 Wall 37 762 EOS

194 [ 21] Drop | 64 Foe 312 Wall 37 762 EOS

195 [ 22] Swap | Foe 312 Wall 37 762 EOS

196 [ 23] Drop | 312 Foe Wall 37 762 EOS

197 [ 24] IsWall | Foe Wall 37 762 EOS

198 [ 25] AbsJumpIf(2) | false Wall 37 762 EOS

199 [ 26] AbsJump(20) | Wall 37 762 EOS

200 [ 20] Swap | Wall 37 762 EOS

201 [ 21] Drop | 37 Wall 762 EOS

202 [ 22] Swap | Wall 762 EOS

203 [ 23] Drop | 762 Wall EOS

204 [ 24] IsWall | Wall EOS

205 [ 25] AbsJumpIf(2) | true EOS

206 [ 2] Read(direction) | EOS

207 [ 3] Look | 39 EOS

208 [ 4] IsWall | Foe 40 284 Wall 39 759 EOS

209 [ 5] Not | false 40 284 Wall 39 759 EOS

210 [ 6] AbsJumpIf(14) | true 40 284 Wall 39 759 EOS

211 [ 14] Dup | 40 284 Wall 39 759 EOS

212 [ 15] Store(direction) | 40 40 284 Wall 39 759 EOS

213 [ 16] Dup | 40 284 Wall 39 759 EOS

214 [ 17] Move | 40 40 284 Wall 39 759 EOS

215 [ 18] Shoot | 40 284 Wall 39 759 EOS

216 [ 19] Drop | true Wall 39 759 EOS

217 [ 20] Swap | Wall 39 759 EOS

218 [ 21] Drop | 39 Wall 759 EOS

219 [ 22] Swap | Wall 759 EOS

220 [ 23] Drop | 759 Wall EOS

221 [ 24] IsWall | Wall EOS

222 [ 25] AbsJumpIf(2) | true EOS

223 [ 2] Read(direction) | EOS

224 [ 3] Look | 40 EOS

225 [ 4] IsWall | Foe 41 283 Wall 40 753 EOS

226 [ 5] Not | false 41 283 Wall 40 753 EOS

227 [ 6] AbsJumpIf(14) | true 41 283 Wall 40 753 EOS

228 [ 14] Dup | 41 283 Wall 40 753 EOS

229 [ 15] Store(direction) | 41 41 283 Wall 40 753 EOS

230 [ 16] Dup | 41 283 Wall 40 753 EOS

231 [ 17] Move | 41 41 283 Wall 40 753 EOS

232 [ 18] Shoot | 41 283 Wall 40 753 EOS

233 [ 19] Drop | true Wall 40 753 EOS

234 [ 20] Swap | Wall 40 753 EOS

235 [ 21] Drop | 40 Wall 753 EOS

236 [ 22] Swap | Wall 753 EOS

237 [ 23] Drop | 753 Wall EOS

238 [ 24] IsWall | Wall EOS

239 [ 25] AbsJumpIf(2) | true EOS

240 [ 2] Read(direction) | EOS

241 [ 3] Look | 41 EOS

242 [ 4] IsWall | Foe 42 283 Wall 41 747 EOS

243 [ 5] Not | false 42 283 Wall 41 747 EOS

244 [ 6] AbsJumpIf(14) | true 42 283 Wall 41 747 EOS

245 [ 14] Dup | 42 283 Wall 41 747 EOS

246 [ 15] Store(direction) | 42 42 283 Wall 41 747 EOS

247 [ 16] Dup | 42 283 Wall 41 747 EOS

248 [ 17] Move | 42 42 283 Wall 41 747 EOS

249 [ 18] Shoot | 42 283 Wall 41 747 EOS

250 [ 19] Drop | true Wall 41 747 EOS

251 [ 20] Swap | Wall 41 747 EOS

252 [ 21] Drop | 41 Wall 747 EOS

253 [ 22] Swap | Wall 747 EOS

254 [ 23] Drop | 747 Wall EOS

255 [ 24] IsWall | Wall EOS

256 [ 25] AbsJumpIf(2) | true EOS

257 [ 2] Read(direction) | EOS

258 [ 3] Look | 42 EOS

259 [ 4] IsWall | Foe 43 282 Wall 42 742 EOS

260 [ 5] Not | false 43 282 Wall 42 742 EOS

261 [ 6] AbsJumpIf(14) | true 43 282 Wall 42 742 EOS

262 [ 14] Dup | 43 282 Wall 42 742 EOS

263 [ 15] Store(direction) | 43 43 282 Wall 42 742 EOS

264 [ 16] Dup | 43 282 Wall 42 742 EOS

265 [ 17] Move | 43 43 282 Wall 42 742 EOS

266 [ 18] Shoot | 43 282 Wall 42 742 EOS

267 [ 19] Drop | true Wall 42 742 EOS

268 [ 20] Swap | Wall 42 742 EOS

269 [ 21] Drop | 42 Wall 742 EOS

270 [ 22] Swap | Wall 742 EOS

271 [ 23] Drop | 742 Wall EOS

272 [ 24] IsWall | Wall EOS

273 [ 25] AbsJumpIf(2) | true EOS

274 [ 2] Read(direction) | EOS

275 [ 3] Look | 43 EOS

276 [ 4] IsWall | Foe 44 282 Wall 43 737 EOS

277 [ 5] Not | false 44 282 Wall 43 737 EOS

278 [ 6] AbsJumpIf(14) | true 44 282 Wall 43 737 EOS

279 [ 14] Dup | 44 282 Wall 43 737 EOS

280 [ 15] Store(direction) | 44 44 282 Wall 43 737 EOS

281 [ 16] Dup | 44 282 Wall 43 737 EOS

282 [ 17] Move | 44 44 282 Wall 43 737 EOS

283 [ 18] Shoot | 44 282 Wall 43 737 EOS

284 [ 19] Drop | true Wall 43 737 EOS

285 [ 20] Swap | Wall 43 737 EOS

286 [ 21] Drop | 43 Wall 737 EOS

287 [ 22] Swap | Wall 737 EOS

288 [ 23] Drop | 737 Wall EOS

289 [ 24] IsWall | Wall EOS

290 [ 25] AbsJumpIf(2) | true EOS

291 [ 2] Read(direction) | EOS

292 [ 3] Look | 44 EOS

293 [ 4] IsWall | Foe 44 282 Wall 44 732 EOS

294 [ 5] Not | false 44 282 Wall 44 732 EOS

295 [ 6] AbsJumpIf(14) | true 44 282 Wall 44 732 EOS

296 [ 14] Dup | 44 282 Wall 44 732 EOS

297 [ 15] Store(direction) | 44 44 282 Wall 44 732 EOS

298 [ 16] Dup | 44 282 Wall 44 732 EOS

299 [ 17] Move | 44 44 282 Wall 44 732 EOS

300 [ 18] Shoot | 44 282 Wall 44 732 EOS

### 7.2.2 Drone Rabbit

drones/berserk.dt

// The extremly harmless drone.

// It sits on one place and checks its health

// If damage detected - run somewhere for 0.1 seconds in hope to

// leave the the zone of danger. Then stop and wait until

// it again recieve some damage.

100 health store // set initial health to 100

main\_loop:

10 wait // wait for 10 ticks

health read getHealth =

// repeat indefinetely if no one harmed the drone

main\_loop jumpif

// what to do if drone recieved some damage

0 359 random // get a random value in the range 1-360

move // move in the random direction

10 wait // wait for 10 ticks

stop // stop

main\_loop jump // and go back to the beginning

This file will be compiled into:

0: Int(100)

1: Store(health)

2: Int(10)

3: Wait

4: Read(health)

5: GetHealth

6: Equal

7: AbsJumpIf(2)

8: Int(0)

9: Int(359)

10: Random

11: Move

12: Int(10)

13: Wait

14: Stop

15: AbsJump(2)

The state of the game:

1 [ 0] Int(100) | EOS

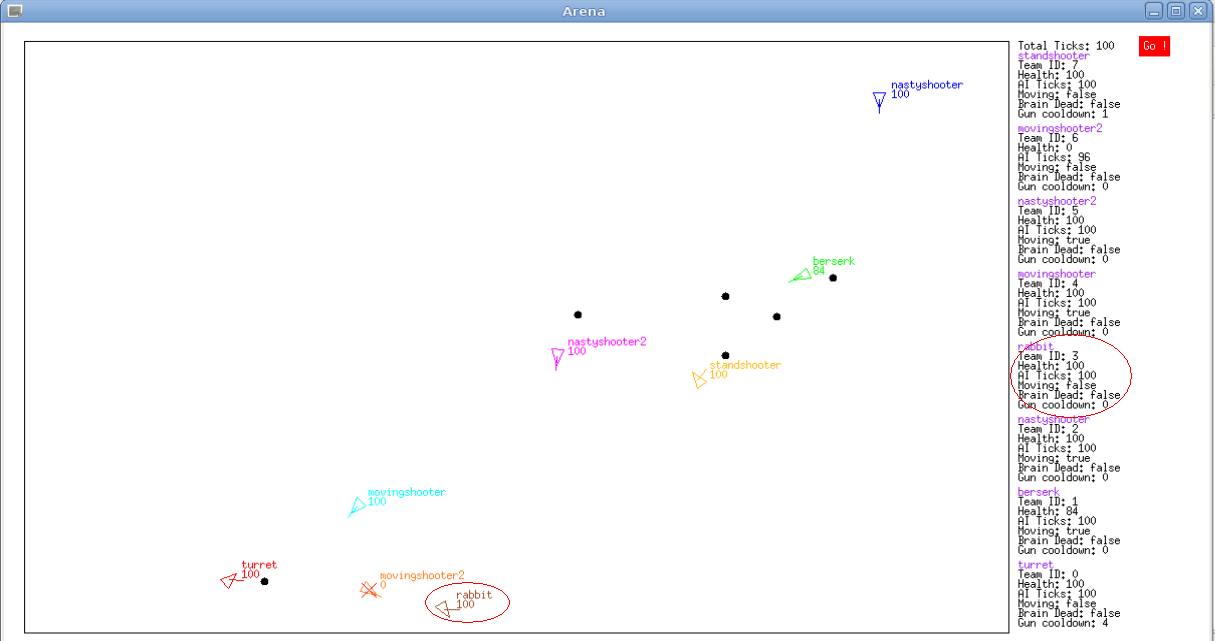
1 [ 0] Int(100) | EOS

2 [ 1] Store(health) | 100 EOS // store 100 in variable health

3 [ 2] Int(10) | EOS

4 [ 3] Wait | 10 EOS //wait ten ticks

5 waiting for 10 ticks

 6 waiting for 9 ticks

7 waiting for 8 ticks

8 waiting for 7 ticks

9 waiting for 6 ticks

10 waiting for 5 ticks

11 waiting for 4 ticks

12 waiting for 3 ticks

13 waiting for 2 ticks

14 waiting for 1 ticks

15 [ 4] Read(health) | EOS //get the stored health

16 [ 5] GetHealth | 100 EOS //get current health

17 [ 6] Equal | 100 100 EOS //check if it is damaged

18 [ 7] AbsJumpIf(2) | true EOS //no damaged

19 [ 2] Int(10) | EOS

20 [ 3] Wait | 10 EOS //wait ten ticks

21 waiting for 10 ticks

22 waiting for 9 ticks

23 waiting for 8 ticks

24 waiting for 7 ticks

25 waiting for 6 ticks

26 waiting for 5 ticks

27 waiting for 4 ticks

28 waiting for 3 ticks

29 waiting for 2 ticks

30 waiting for 1 ticks

31 [ 4] Read(health) | EOS//get the stored health

32 [ 5] GetHealth | 100 EOS//get current health

33 [ 6] Equal | 100 100 EOS//check if it is damaged

34 [ 7] AbsJumpIf(2) | true EOS//no damaged

35 [ 2] Int(10) | EOS//wait ten ticks

36 [ 3] Wait | 10 EOS

37 waiting for 10 ticks

38 waiting for 9 ticks

39 waiting for 8 ticks

40 waiting for 7 ticks

41 waiting for 6 ticks

42 waiting for 5 ticks

43 waiting for 4 ticks

44 waiting for 3 ticks

45 waiting for 2 ticks

46 waiting for 1 ticks

47 [ 4] Read(health) | EOS//get the stored health

48 [ 5] GetHealth | 100 EOS//get current health

49 [ 6] Equal | 100 100 EOS//check if it is damaged

50 [ 7] AbsJumpIf(2) | true EOS//no damaged

51 [ 2] Int(10) | EOS

52 [ 3] Wait | 10 EOS//wait ten ticks

53 waiting for 10 ticks

54 waiting for 9 ticks

55 waiting for 8 ticks

56 waiting for 7 ticks

57 waiting for 6 ticks

58 waiting for 5 ticks

59 waiting for 4 ticks

60 waiting for 3 ticks

61 waiting for 2 ticks

62 waiting for 1 ticks

63 [ 4] Read(health) | EOS

64 [ 5] GetHealth | 100 EOS

65 [ 6] Equal | 100 100 EOS

66 [ 7] AbsJumpIf(2) | true EOS

67 [ 2] Int(10) | EOS

68 [ 3] Wait | 10 EOS

69 waiting for 10 ticks

70 waiting for 9 ticks

71 waiting for 8 ticks

72 waiting for 7 ticks

73 waiting for 6 ticks

74 waiting for 5 ticks

75 waiting for 4 ticks

76 waiting for 3 ticks

77 waiting for 2 ticks

78 waiting for 1 ticks

79 [ 4] Read(health) | EOS

80 [ 5] GetHealth | 100 EOS

81 [ 6] Equal | 100 100 EOS

82 [ 7] AbsJumpIf(2) | true EOS

83 [ 2] Int(10) | EOS

84 [ 3] Wait | 10 EOS

85 waiting for 10 ticks

86 waiting for 9 ticks

87 waiting for 8 ticks

88 waiting for 7 ticks

89 waiting for 6 ticks

90 waiting for 5 ticks

91 waiting for 4 ticks

92 waiting for 3 ticks

93 waiting for 2 ticks

94 waiting for 1 ticks

95 [ 4] Read(health) | EOS

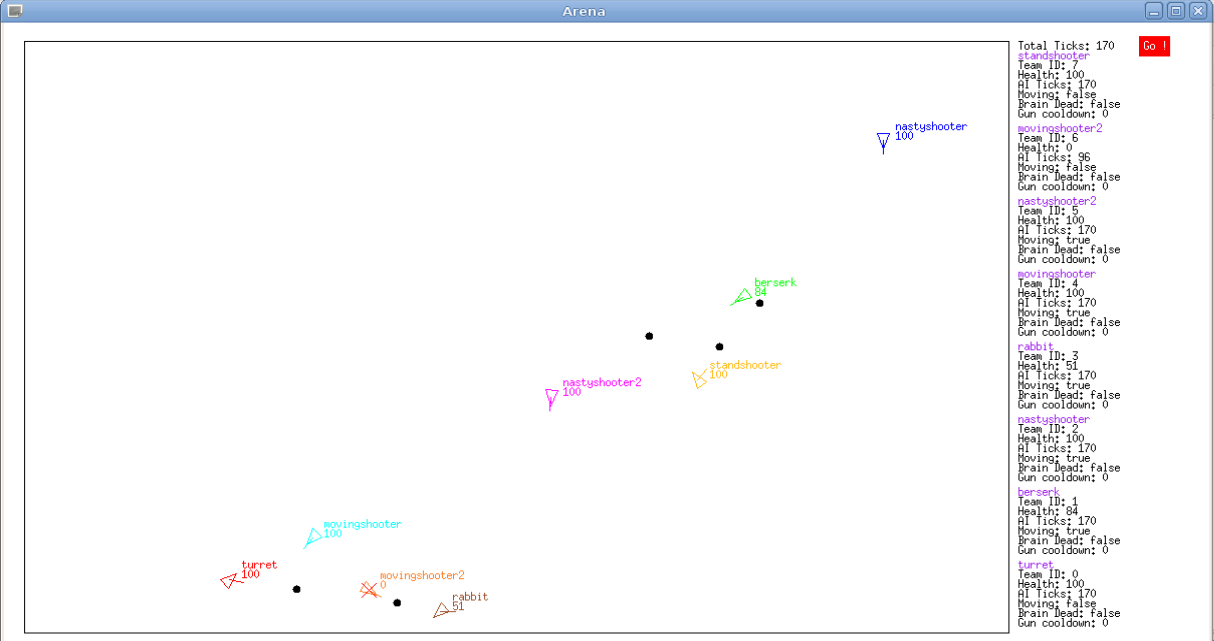
96 [ 5] GetHealth | 100 EOS

97 [ 6] Equal | 100 100 EOS

98 [ 7] AbsJumpIf(2) | true EOS

99 [ 2] Int(10) | EOS

100 [ 3] Wait | 10 EOS



101 waiting for 10 ticks

102 waiting for 9 ticks

103 waiting for 8 ticks

104 waiting for 7 ticks

105 waiting for 6 ticks

106 waiting for 5 ticks

107 waiting for 4 ticks

108 waiting for 3 ticks

109 waiting for 2 ticks

110 waiting for 1 ticks

111 [ 4] Read(health) | EOS

112 [ 5] GetHealth | 100 EOS

113 [ 6] Equal | 100 100 EOS

114 [ 7] AbsJumpIf(2) | true EOS

115 [ 2] Int(10) | EOS

116 [ 3] Wait | 10 EOS

117 waiting for 10 ticks

118 waiting for 9 ticks

119 waiting for 8 ticks

120 waiting for 7 ticks

121 waiting for 6 ticks

122 waiting for 5 ticks

123 waiting for 4 ticks

124 waiting for 3 ticks

125 waiting for 2 ticks

126 waiting for 1 ticks

127 [ 4] Read(health) | EOS

128 [ 5] GetHealth | 100 EOS

129 [ 6] Equal | 100 100 EOS

130 [ 7] AbsJumpIf(2) | true EOS

131 [ 2] Int(10) | EOS

132 [ 3] Wait | 10 EOS

133 waiting for 10 ticks

134 waiting for 9 ticks

135 waiting for 8 ticks

136 waiting for 7 ticks

137 waiting for 6 ticks

138 waiting for 5 ticks

139 waiting for 4 ticks

140 waiting for 3 ticks

141 waiting for 2 ticks

142 waiting for 1 ticks

143 [ 4] Read(health) | EOS

144 [ 5] GetHealth | 100 EOS

145 [ 6] Equal | 100 100 EOS

146 [ 7] AbsJumpIf(2) | true EOS

147 [ 2] Int(10) | EOS

148 [ 3] Wait | 10 EOS

149 waiting for 10 ticks

150 waiting for 9 ticks

151 waiting for 8 ticks

152 waiting for 7 ticks

153 waiting for 6 ticks

154 waiting for 5 ticks

155 waiting for 4 ticks

156 waiting for 3 ticks

157 waiting for 2 ticks

158 waiting for 1 ticks

159 [ 4] Read(health) | EOS//get the stored health

160 [ 5] GetHealth | 100 EOS//get current health

161 [ 6] Equal | 51 100 EOS//check if it is damaged

162 [ 7] AbsJumpIf(2) | false EOS//is damaged

163 [ 8] Int(0) | EOS

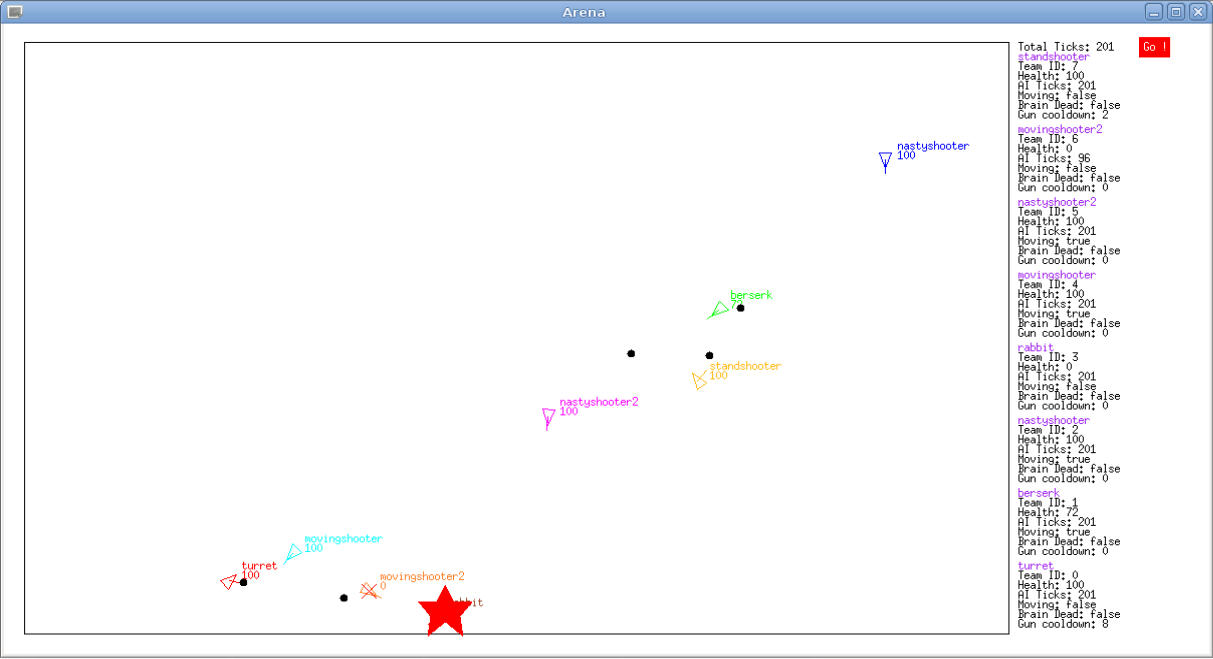
164 [ 9] Int(359) | 0 EOS

165 [ 10] Random | 359 0 EOS//get a random direction

166 [ 11] Move | 223 EOS//move towards this direction

167 [ 12] Int(10) | EOS

168 [ 13] Wait | 10 EOS//wait ten ticks



169 waiting for 10 ticks

170 waiting for 9 ticks

171 waiting for 8 ticks

172 waiting for 7 ticks

173 waiting for 6 ticks

174 waiting for 5 ticks

175 waiting for 4 ticks

176 waiting for 3 ticks

177 waiting for 2 ticks

178 waiting for 1 ticks

179 [ 14] Stop | EOS

180 [ 15] AbsJump(2) | EOS

181 [ 2] Int(10) | EOS

182 [ 3] Wait | 10 EOS

183 waiting for 10 ticks

184 waiting for 9 ticks

185 waiting for 8 ticks

186 waiting for 7 ticks

187 waiting for 6 ticks

188 waiting for 5 ticks

189 waiting for 4 ticks

190 waiting for 3 ticks

191 waiting for 2 ticks

192 waiting for 1 ticks

193 [ 4] Read(health) | EOS//get the stored health

194 [ 5] GetHealth | 100 EOS//get current health

195 [ 6] Equal | 15 100 EOS//check if it is damaged

196 [ 7] AbsJumpIf(2) | false EOS//is damaged

197 [ 8] Int(0) | EOS

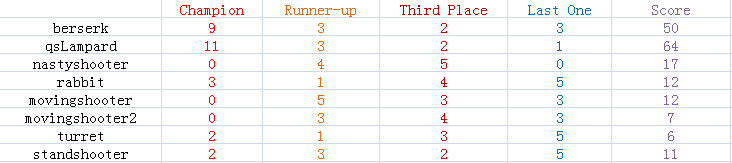
198 [ 9] Int(359) | 0 EOS

199 [ 10] Random | 359 0 EOS//get a random direction

200 [ 11] Move | 288 EOS//move towards this direction

201 [ 12] Int(10) | EOS

At last, I tested the game for 25 times and get the following table:



The score is 5\*Champion+3\*Runner-up+1\*Third Place-2\*Last One.

So if you can develop a good AI for your drone, you can make a difference and beat other drones!

# Chapter 8: Lessons Learned

## 8.1 George Brink

My previous understanding of programming languages was pure practical one. I knew how to write program, how to choose correct feature of the language depending on the task, how to choose the language for the task. The lectures taught me the correct names for many of those features and more theoretical understanding of the language translation process.

The OCaml requirement of the class proved to me that I have more imperative style of thinking than functional. For example, the procedures like recursion and list reversion are fairly complex and heavy processes in most imperative languages, and as a result it pained me to use them during writing the code in OCaml.

Another problem I encountered while working with OCaml is a too liberal use of currying. The OCaml does not actually check did it encounter a function or not a function which result in a very strange style of code. For example, if we need a function which should do something with two integer values we can write such function in a very simple manner:

let add x y = x+y;;

But the call to such function will be “add 1 2;;” which by the currying rule should be a “add(1(2))”. I doubt that any sane human being can realize that “1(2)” actually means “1 and 2”.

As a result, I am sure that after this class I will not use OCaml anymore.

## 8.2 Shuo Qiu

Stay in contact with your team throughout the semester, to make sure you are always up to speed.

Make sure you thoroughly understand the OCaml examples from class and try to search for other OCaml code for better understanding. Learning the basic OCaml language is hard, but class and object is much easier to use, just like Java or C.

Using OCaml in Linux is much easier in Window. Do not need complicated configuration compared with windows, Linux just needs you type in a command and everything is done.

Better understanding of lexing, scanner, parser and ast. I learned how a compiler works by participating in building them for our own language.

SVN really helps. Although I have troubles in submitting the files, I can always get the latest version of the project and find what has been changed.

## 8.3 Xiang Yao

In this class, I learnt the definition of a language as well as how to create and design a language on my own. During the development of the project, I learnt how a compiler was generated from beginning. First in the language part, I learnt what syntax analysis and semantic analysis were, and building Scanner, Parser as well as AST provided me more opportunities to widen my view from the inside of a compiler. After the language part, the Game Engine part gave me chance to apply a new designed language to real life, and it was in this way that I learnt how important the design was since using a language was sometimes even harder than creating one.

Besides the topic about Drone War, this class also helped me to learn and get familiar a new language of Ocaml, which was really complicated and hard to understand at beginning, however, we finally understood how powerful it was when applied to developing compilers. As a functional language, Ocaml helped to reduce a number of useless work when compared to other high level languages such as Java and C++.

Finally, I also learnt a lot from team leader as well as other team members during cooperations and brainstorming. I learnt that good management of time and human resource always resulted in better progress in any kind of project.

## 8.4 Xiaotong Chen

Firstly, because of this course, I can touch ocaml as the development language, which I never know before. The new language I learnt can be one of the most important parts in this class. As a functional language, its strictly requirement of “type” is the most impressed feature. To my surprise, all the statements have to “return” the same type. In additional, the “return” strategy is also a kind of strange: all the operation will return some type of value. For a simple example, after print, it will return -unit(), which basically indicates nothing to return. So we must take care of the return type when coding. By doing this, I gained my patience.

Secondly, it’s also about this language. At the very beginning, we have no idea about how to execute multiple lines of codes just like in Java or C. After carefully study the document of Ocaml, we finally know how to execute multiple lines of code: use begin and end. By doing this we can build a block of code that can be seen as an entire action.

What’s more, we design a game oriented language at the very beginning. We think like a designer instead of a programmer. This is the first time I have that kind of feel, that’s amazing. We design the game rules and operations of the drones. Then we construct the related scanner to analyze the source code. By doing this, we convert the source code to tokens. Next step is to analyze the tokens to check if these tokens are satisfied our grammar. Both parser and ast together can help us finish this step. With the definition of the “program” ast and parser can construct an AST after the grammar analysis. When finish these steps, which are general compiling processes, we convert source code to byte code. Then we implement the engine part that response the byte code. Finishing above steps will generate a game.

# Appendix

## Source Code Listing

### 1. Scanner.mll

{

**open** **Parser**;;

**open** **Lexing**;;

**let** debug=1;;

**let** incr\_lineno lexbuf =

**let** pos = lexbuf.lex\_curr\_p **in**

lexbuf.lex\_curr\_p <- { pos **with**

pos\_lnum = pos.pos\_lnum + 1;

pos\_bol = pos.pos\_cnum;

}

;;

**exception** **Unknown\_token** **of** string \* int \* int;;

**let** create\_hashtable size init =

**let** tbl = **Hashtbl**.create size **in**

**List**.iter (**fun** (key, data) -> **Hashtbl**.add tbl key data) init;

tbl

**let** keyword\_table =

create\_hashtable 8 [

( "dup", **DUP** );

( "drop", **DROP** );

( "dropall", **DROPALL** );

( "swap", **SWAP** );

( "over", **OVER** );

( "rot", **ROT** );

( "read", **READ** );

( "store", **STORE** );

( "jump", **JUMP** );

( "jumpif", **JUMPIF** );

( "sub", **SUB** );

( "endsub", **END\_SUB** );

( "if", **IF** );

( "else", **ELSE** );

( "endif", **END\_IF** );

( "begin", **BEGIN** );

( "while", **WHILE** );

( "again", **AGAIN** );

( "move", **MOVE** );

( "stop", **STOP** );

( "shoot", **SHOOT** );

( "look", **LOOK** );

( "wait", **WAIT** );

( "gethealth", **GETHEALTH** );

( "random", **RANDOM** );

( "isfoe", **ISFOE** );

( "isally", **ISALLY** );

( "iswall", **ISWALL** );

( "mod", **MOD** );

( "and", **AND** );

( "or", **OR** );

( "not", **NOT** );

];;

}

**let** digit = ['0'-'9']

**let** space = [' ' '\t']

**let** whitespace = [' ' '\t' '\r']

**let** notspace = [^ ' ' '\t' '\r' '\n']

**let** name = ['a'-'z' 'A'-'Z'] ['a'-'z' 'A'-'Z' '0'-'9' '\_']\*

rule token = parse

| '\n' { incr\_lineno lexbuf; token lexbuf }

| digit+ **as** str { **INTEGER** (int\_of\_string str) }

| '+' { **PLUS** }

| '-' { **MINUS** }

| '\*' { **TIMES** }

| '/' { **DIVIDE** }

| '^' { **POWER** }

| "true"|"false" **as** str { **BOOL**(bool\_of\_string str) }

| '=' { **EQUAL** }

| '<' { **LESS** }

| '>' { **GREATER** }

| ['a'-'z' 'A'-'Z']+ **as** str {**try**

**let** token = **Hashtbl**.find keyword\_table (**String**.lowercase str) **in**

token

**with** **Not\_found** -> **NAME** (str) }

| name ':' **as** str { **LABEL** (**String**.sub str 0 ((**String**.length str)-1) ) }

| name **as** str { **NAME** (str) }

| whitespace { token lexbuf }

| "//" { sinlge\_line\_comment lexbuf }

| "/\*" { multi\_line\_comment lexbuf }

| notspace \* **as** str { **raise** (**Unknown\_token** (str, lexbuf.lex\_curr\_p.pos\_lnum, lexbuf.lex\_start\_p.pos\_cnum-lexbuf.lex\_start\_p.pos\_bol +1) ) }

| eof { **EOF** }

**and** sinlge\_line\_comment = parse

| '\n' { **Lexing**.new\_line lexbuf; token lexbuf }

| \_ { sinlge\_line\_comment lexbuf }

**and** multi\_line\_comment = parse

| "\*/" { token lexbuf }

| '\n' { **Lexing**.new\_line lexbuf; multi\_line\_comment lexbuf }

| \_ { multi\_line\_comment lexbuf }

### 2. Parser.ply

%{

**open** **Ast**;;

**open** **Printf**;;

**open** **Lexing**;;

**open** **Utils**;;

**let** auto\_label\_counter = ref 0;;

**let** make\_label() =

incr auto\_label\_counter;

("-" ^ string\_of\_int(!auto\_label\_counter))

;;

%}

%token **SUB** **END\_SUB**

%token **IF** **ELSE** **END\_IF**

%token **BEGIN** **WHILE** **AGAIN**

%token **READ** **STORE**

%token **COLON**

%token **JUMP** **JUMPIF**

%token <string> **LABEL**

%token <string> **NAME**

%token <int> **INTEGER**

%token **PLUS** **MINUS** **TIMES** **DIVIDE** **MOD** **POWER**

%token **AND** **OR** **NOT**

%token <bool> **BOOL**

%token **EQUAL** **LESS** **GREATER**

%token **DROP** **DROPALL** **DUP** **SWAP** **OVER** **ROT**

%token **MOVE** **STOP** **SHOOT** **LOOK** **ISFOE** **ISALLY** **ISWALL** **WAIT** **GETHEALTH** **RANDOM**

%token **EOF**

%start drone

%**type** <**Ast**.sub list> drone

%%

drone:

program { **let** main\_sub = { name="--"; body = **List**.rev (fst $1); } **in**

main\_sub :: snd $1 }

program:

{ [], [] } /\* two lists **for** main body **of** the program **and** **for** functions defined by users \*/

| program operation { ($2 :: fst $1), snd $1 } /\* add operations **to** the body **of** the main program \*/

| program sub { fst $1, ($2 :: snd $1) } /\* add user **function** **to** the list **of** subs \*/

| program compaund\_statment { ($2 @ fst $1), snd $1 }

sub:

**SUB** **NAME** operations **END\_SUB** { { name = $2; body = **List**.rev $3; } } /\* store the **function** name **and** **function** operations between "sub" **and** "esub" \*/

operations:

{ [] }

| operations operation { **if** $2=**Nop** **then** $1 **else** $2 :: $1 }

| operations compaund\_statment { $2 @ $1 }

| operations error { **let** pos = **Parsing**.rhs\_start\_pos 2 **in**

**raise** (**Parse\_failure** ("Unrecognized tokens starting from line %d position %d**\n**", pos.pos\_lnum, (pos.pos\_cnum - pos.pos\_bol +1)));

}

compaund\_statment:

**IF** operations **END\_IF** { **let** lbl = make\_label() **in**

( **Label**(lbl):: $2 ) @ [ **JumpIf**(lbl) ; **Not** ]

}

| **IF** operations **ELSE** operations **END\_IF** { **let** lbl1 = make\_label() **and** lbl2= make\_label() **in**

( **Label**(lbl2):: $4) @ ( **Label**(lbl1)::(**Jump**(lbl2):: $2 )) @ [ **JumpIf**(lbl1) ; **Not** ]

}

| **BEGIN** operations **AGAIN** { **let** lbl=make\_label() **in**

(**Jump**(lbl)::$2) @ [**Label**(lbl)]

}

| **BEGIN** operations **WHILE** operations **AGAIN** {**let** lbl1 =make\_label() **and** lbl2 = make\_label() **in**

[**Label**(lbl1); **Jump**(lbl2)] @ $4 @ [ **JumpIf**(lbl1) ; **Not** ] @ $2 @ [**Label**(lbl2)]

}

operation:

**INTEGER** { **Int**($1) }

| **PLUS** { **Plus** }

| **MINUS** { **Minus** }

| **TIMES** { **Times** }

| **DIVIDE** { **Divide** }

| **MOD** { **Mod** }

| **POWER** { **Power** }

| **AND** { **And** }

| **OR** { **Or** }

| **NOT** { **Not** }

| **BOOL** { **Bool**($1) }

| **EQUAL** { **Equal** }

| **LESS** { **Less** }

| **GREATER** { **Greater** }

| **NAME** **READ** { **Read**($1) }

| **NAME** **STORE** { **Store**($1) }

| **DROP** { **Drop** }

| **DROPALL** { **Dropall** }

| **DUP** { **Dup** }

| **SWAP** { **Swap** }

| **OVER** { **Over** }

| **ROT** { **Rot** }

| **LABEL** { **Label**($1) }

| **NAME** **JUMP** { **Jump**($1) }

| **NAME** **JUMPIF** { **JumpIf**($1) }

| **NAME** { **Call**($1) }

| **MOVE** { **Move** }

| **STOP** { **Stop** }

| **SHOOT** { **Shoot** }

| **LOOK** { **Look** }

| **ISFOE** { **IsFoe** }

| **ISALLY** { **IsAlly** }

| **ISWALL** { **IsWall** }

| **WAIT** { **Wait** }

| **GETHEALTH** { **GetHealth** }

| **RANDOM** { **Random** }

### 3. AST.ml

**module** **StringMap** = **Map**.**Make**(**String**);;

**type** bytecode =

**Nop**

| **Int** **of** int

| **Plus**

| **Minus**

| **Times**

| **Divide**

| **Mod**

| **Power**

| **And**

| **Or**

| **Not**

| **Bool** **of** bool

| **Equal**

| **Less**

| **Greater**

| **Colon**

| **Store** **of** string

| **Read** **of** string

| **Label** **of** string

| **Drop**

| **Dropall**

| **Dup**

| **Swap**

| **Over**

| **Rot**

| **Jump** **of** string

| **JumpIf** **of** string

| **AbsJump** **of** int

| **AbsJumpIf** **of** int

| **Call** **of** string

| **Move**

| **Stop**

| **Shoot**

| **Look**

| **IsFoe**

| **IsAlly**

| **IsWall**

| **Wait**

| **GetHealth**

| **Random**

;;

**let** string\_of\_bytecode code =

**match** code **with**

**Nop** -> "" *(\* \*)*

| **Int**(x) -> "Int(" ^ (string\_of\_int x) ^ ")" *(\* 5, integer\*)*

| **Plus** -> "Plus" *(\* 1 2 +, addition of integers \*)*

| **Minus** -> "Minus" *(\* 1 2 -, subtraction of integers \*)*

| **Times** -> "Times" *(\* 1 2 \*, mutip of integers \*)*

| **Divide** -> "Divide" *(\* 1 2 /, division of integers \*)*

| **Mod** -> "Mod" *(\* 1 2 mod, take mod of 1 by 2 \*)*

| **Power** -> "Power" *(\* 1 2 ^, take the power of 1 by 2 \*)*

| **And** -> "And" *(\* bool1 bool2 and, return bool1 && bool2 \*)*

| **Or** -> "Or" *(\* bool1 bool2 or, return bool1 || bool2 \*)*

| **Not** -> "Not" *(\* bool1 not, return negation of bool1 \*)*

| **Bool**(b) -> "Bool(" ^ (string\_of\_bool b) ^ ")" *(\* true, boolean type true or false \*)*

| **Equal** -> "Equal" *(\* 2 2 =, equal \*)*

| **Less** -> "Less" *(\* 1 2 <, smaller \*)*

| **Greater** -> "Greater" *(\* 2 1 >, greater \*)*

| **Colon** -> "Colon" *(\* : , colon \*)*

| **Store**(var) -> "Store(" ^var ^ ")" *(\* 2 store , store the value of 2 \*)*

| **Read**(var) -> "Read(" ^var ^ ")" *(\* 2 read , read the value of 2 \*)*

| **Label**(name) -> "Label(" ^ name ^ ")" *(\* label1: , take the label of name label1 \*)*

| **Drop** -> "Drop" *(\* a b c -> a b, drop the first element in the stack \*)*

| **Dropall** -> "Dropall" *(\* a b c ->, drop all elements in the stack \*)*

| **Dup** -> "Dup" *(\* a b c -> a b c c, duplicate first element in the stack \*)*

| **Swap** -> "Swap" *(\* a b c -> a c b, swap the elements in the stack \*)*

| **Over** -> "Over" *(\* a b c -> a b c b \*)*

| **Rot** -> "Rot" *(\* a b c -> b c a \*)*

| **Jump**(name) -> "Jump(" ^ name ^ ")" *(\* label1 jump, jump the label names label1 \*)*

| **JumpIf**(name) -> "JumpIf(" ^ name ^ ")" *(\* label1 jumpif, condition jump\*)*

| **AbsJump**(addr) -> "AbsJump(" ^ (string\_of\_int addr) ^ ")"

| **AbsJumpIf**(addr) -> "AbsJumpIf(" ^ (string\_of\_int addr) ^ ")"

| **Call**(name) -> "Call(" ^ name ^ ")" *(\* call the function by the name \*)*

| **Move** -> "Move"

| **Stop** -> "Stop"

| **Shoot** -> "Shoot"

| **Look** -> "Look"

| **IsFoe** -> "IsFoe"

| **IsAlly** -> "IsAlly"

| **IsWall** -> "IsWall"

| **Wait** -> "Wait"

| **GetHealth** -> "GetHealth"

| **Random** -> "Random"

;;

**type** look\_flags =

**Foe** *(\* enemy type \*)*

| **Ally** *(\* friend type\*)*

| **Wall** *(\* boundary of arena \*)*

**type** operands =

**Undefined**

| **Integer** **of** int

| **Boolean** **of** bool

| **Flag** **of** look\_flags

**let** string\_of\_operand op =

**match** op **with**

**Undefined** -> "undef"

| **Integer**(x) -> string\_of\_int x

| **Boolean**(b) -> string\_of\_bool b

| **Flag**(f) -> **match** f **with**

**Foe** -> "Foe"

| **Ally** -> "Ally"

| **Wall** -> "Wall"

**type** sub = { *(\* function defined by user \*)*

name : string; *(\* function name \*)*

body : bytecode list; *(\* function body \*)*

}

**type** program = bytecode list \* sub list *(\* compiled, but not-linked program defintion returned from the parser \*)*

### 4. Drone.ml

**open** **Ast**;;

**open** **Parser**;;

**open** **Parser\_dbt**;;

**open** **Printf**;;

**open** **Utils**;;

**exception** **Error\_in\_AI** **of** string \* string \* int;;

**type** drone\_action =

**No\_Action**

| **Do\_Shoot** **of** int \* int

| **Do\_Look** **of** int

**class** drone =

**object** (self)

*(\* init the containers\*)*

**val** **mutable** subs = **Hashtbl**.create 16

**val** **mutable** vars : (string, **Ast**.operands) **Hashtbl**.t = **Hashtbl**.create 16

**val** **mutable** current\_sub = "--"

**val** **mutable** instruction\_pointer = 0

**val** **mutable** call\_stack: (string \* int) **Stack**.t = **Stack**.create ()

**val** **mutable** stack : (**Ast**.operands) **Stack**.t = **Stack**.create ()

*(\* variables to enable debug functionality \*)*

**val** **mutable** debug\_mode = false

**val** **mutable** debug\_out\_file = stderr *(\* channel for debug output \*)*

**val** **mutable** tick\_counter = 0 *(\* life-time ticks counter, used in debug output function \*)*

*(\* various members \*)*

**val** **mutable** drone\_name = "" *(\* name of the drone for GUI \*)*

**val** **mutable** team\_id = 0 *(\* id of the team this drone belongs to \*)*

*(\* variuables to describe current drone state \*)*

**val** **mutable** health = 100

**val** **mutable** direction\_of\_the\_body = 0 *(\* used by GUI to draw where the drone is moving if drone's image is not a circle \*)*

**val** **mutable** direction\_of\_the\_gun = 0 *(\* used by GUI to draw where the drone's gun is pointing (direction of the last SHOOT command \*)*

**val** **mutable** ticks\_to\_wait = 0 *(\* if non-zero, the AI will skip a step \*)*

**val** **mutable** moving = false *(\* does the drone moving or not? \*)*

**val** **mutable** brain\_dead = false *(\* will become true if at some step the drone caught an exception \*)*

**val** **mutable** reason\_for\_coma = "" *(\* explanation why AI died \*)*

**val** **mutable** x\_position = 0. *(\* used by other drones to determine the position in the arena can set maximum in Arena as Radius of the circle\*)*

**val** **mutable** y\_position = 0. *(\* used by other drones to determine the position in the arena 0-360\*)*

*(\* maxmium bullet load is 5 can be displayed in the GUI \*)*

**val** **mutable** bullet\_capacity = 5

**val** **mutable** has\_bullet = true

*(\* set to 10 each time drone shoots. drone cannot shoot until gun\_cooldown returns to zero \*)*

**val** **mutable** gun\_cooldown = 0

**method** get\_moving\_direction = direction\_of\_the\_body

**method** set\_moving\_direction dire = direction\_of\_the\_body <- dire

**method** get\_x\_position = x\_position

**method** set\_x\_position x = x\_position <- x

**method** get\_y\_position = y\_position

**method** set\_y\_position y = y\_position <- y

**method** get\_current\_sub = current\_sub;

**method** get\_direction\_of\_the\_gun = direction\_of\_the\_gun;

**method** get\_drone\_name = drone\_name

**method** is\_brain\_dead = brain\_dead

**method** is\_alive = (health > 0)

**method** get\_ai\_ticks = tick\_counter

**method** get\_health = health

**method** belongs\_to\_team id = team\_id <- id

**method** get\_team\_id = team\_id

**method** get\_moving\_status = moving

**method** set\_health h =

health <- max h 0;

moving <- moving && health>0

**method** get\_reason\_for\_coma = reason\_for\_coma

**method** get\_gun\_cooldown = gun\_cooldown

*(\* this method is called, by the engine's LOOK procedure \*)*

**method** found\_target dist dire flag=

**Stack**.push (**Integer** (dist)) stack;

**Stack**.push (**Integer** (dire)) stack;

**Stack**.push (**Flag** (flag)) stack

**method** move speed =

**if** moving **then**

**begin**

y\_position <- y\_position +. (float\_of\_int(speed) \*. (sin (float\_of\_int(direction\_of\_the\_body) \*. pi /. 180.)));

x\_position <- x\_position +. (float\_of\_int(speed) \*. (cos (float\_of\_int(direction\_of\_the\_body) \*. pi /. 180.)));

*(\* check did we hit a wall? \*)*

**if** x\_position > 1000. || x\_position < 0. || y\_position > 1000. || y\_position < 0. **then**

**begin**

self#set\_health (health - 10);

**if** x\_position > 1000. **then** x\_position <- 1000.;

**if** x\_position < 0. **then** x\_position <- 0.;

**if** y\_position > 1000. **then** y\_position <- 1000.;

**if** y\_position < 0. **then** y\_position <- 0.;

*(\* this is still debated, what to do after hiting the wall, stop or bounce from it? \*)*

*(\* direction\_of\_the\_body <- ((direction\_of\_the\_body + 180) mod 360; (\* bouncing adds more chaos to the battle \*) \*)*

moving <- false; *(\* stopping is more easy to predict and explain \*)*

**if** health=0 **then** moving <- false *(\* if drone died after hitting the wall, it definetely will not move anymore \*)*

**end**

**end**

**method** set\_debug\_output out\_file =

debug\_out\_file <- out\_file;

debug\_mode <- true

*(\* print out all operations in the container \*)*

**method** dump\_code body\_as\_array out\_file =

**let** command\_counter = ref 0 **in**

**Array**.iter (**fun** x ->

fprintf out\_file "%3d: %s**\n**" !command\_counter (string\_of\_bytecode x);

command\_counter := !command\_counter +1

) body\_as\_array

*(\* decompile the program into compilable text \*)*

**method** decompile out\_file =

**let** body = (**Hashtbl**.find subs "--") **in** self#dump\_code body out\_file;

**Hashtbl**.iter (**fun** name body ->

**if** not (name="--") **then** **begin**

fprintf out\_file "**\n**sub %s**\n**" name;

self#dump\_code body out\_file;

fprintf out\_file "esub**\n**"

**end**

) subs

*(\* takes a raw list of operators including a Label(name) operator,*

*Remove all label, put them into temporary hash table*

*Using this hash table satisfy all jump(name) and convert them to jump(address) \*)*

**method** link\_jumps body\_as\_list =

**let** lbls = **Hashtbl**.create 16 **in**

**let** no\_label = **List**.fold\_left (**fun** acc x ->

**match** x **with**

**Label**(name) ->

**if** **Hashtbl**.mem lbls name **then** **raise** (**Failure** ("Label "^name^" defined twice"))

**else** **Hashtbl**.add lbls name (**List**.length acc);

acc

| \_ -> x::acc

) [] body\_as\_list **in**

**let** abs\_jumps = **List**.map(**fun** x -> **match** x **with**

**Jump**(name) ->

**if** not (**Hashtbl**.mem lbls name) **then** **raise** (**Failure** ("Label "^name^" is not defined"));

**AbsJump**( **Hashtbl**.find lbls name )

| **JumpIf**(name) ->

**if** not (**Hashtbl**.mem lbls name) **then** **raise** (**Failure** ("Label "^name^" is not defined"));

**AbsJumpIf**( **Hashtbl**.find lbls name )

| \_ -> x ) no\_label **in**

**Array**.of\_list (**List**.rev abs\_jumps)

*(\* check existance of a called sub, complain if it is not defined \*)*

**method** check\_sub\_existance body =

**Array**.iter (**fun** x -> **match** x **with**

**Call**(name) -> **if** not (**Hashtbl**.mem subs name) **then** **raise** (**Failure** ("Sub "^name^" is not defined"))

| \_ -> ()

) body

*(\* Read the drone \*)*

**method** load file\_name =

drone\_name <- **Filename**.chop\_extension (**Filename**.basename file\_name);

**let** chan\_in = **Pervasives**.open\_in file\_name **in**

**let** lexbuf = **Lexing**.from\_channel chan\_in **in**

**let** program =

(**if** (**Filename**.check\_suffix file\_name ".dt" ) **then** **Parser**.drone **Scanner**.token lexbuf

**else** **if** (**Filename**.check\_suffix file\_name ".dbt" ) **then** **Parser\_dbt**.drone **Scanner\_dbt**.drone\_basic lexbuf

**else** ([])

) **in**

*(\* First convert all jumps to the label into absolute jumps \*)*

**List**.iter (**fun** sub -> **Hashtbl**.add subs sub.name (self#link\_jumps sub.body)) program;

*(\* Next step, check the existance of all called user funcitons \*)*

**Hashtbl**.iter (**fun** name body -> (self#check\_sub\_existance body)) subs;

*(\* Last step, set starting position for the drone \*)*

self#set\_x\_position (**Random**.float 1000.);

self#set\_y\_position (**Random**.float 1000.);

self#set\_moving\_direction (**Random**.int 360)

*(\* self#print\_current\_pos; \*)*

*(\* helping pop function which converts operand to integer \*)*

**method** pop\_int=

**if** **Stack**.is\_empty stack **then** self#freeze "Empty stack";

**match** (**Stack**.pop stack) **with**

**Integer** op-> op

| \_ -> self#freeze "Type mismatch"; 0

*(\* helping pop function which converts operand to bool \*)*

**method** pop\_bool=

**if** **Stack**.is\_empty stack **then** self#freeze "Empty stack";

**match** (**Stack**.pop stack) **with**

**Boolean** op -> op

| \_ -> self#freeze "Type mismatch"; false

*(\* helping pop function which converts operand to look\_flag \*)*

**method** pop\_flag=

**if** **Stack**.is\_empty stack **then** self#freeze "Empty stack";

**match** (**Stack**.pop stack) **with**

**Flag** op -> op

| \_ -> self#freeze "Type mismatch"; **Wall**

**method** step =

tick\_counter <- tick\_counter+1;

**if** gun\_cooldown>0 **then** gun\_cooldown <- gun\_cooldown-1;

**if** ticks\_to\_wait > 0 **then** **begin**

**if** debug\_mode **then** **begin**

fprintf debug\_out\_file "%4d waiting for %d ticks**\n**" tick\_counter ticks\_to\_wait;

**end**;

ticks\_to\_wait <- ticks\_to\_wait-1;

**No\_Action**

**end** **else** **begin**

**let** body = (**Hashtbl**.find subs current\_sub) **in**

**if** (**Array**.length body) = instruction\_pointer **then** **begin**

**if** **Stack**.is\_empty call\_stack **then** self#freeze "Main program terminated";

**let** return\_address = (**Stack**.pop call\_stack) **in** **begin**

current\_sub <- fst return\_address;

instruction\_pointer <- snd return\_address;

**end**;

**No\_Action**

**end** **else** **begin**

**if** debug\_mode **then** self#print\_current\_state;

**let** action = **match** **Array**.get body instruction\_pointer **with**

*(\* primitive types \*)*

**Int** (x) -> **Stack**.push (**Integer** x) stack; **No\_Action**

| **Bool**(x) -> **Stack**.push (**Boolean** x) stack; **No\_Action**

*(\* simple arithmetics \*)*

| **Plus** -> **let** op2=self#pop\_int **and** op1=self#pop\_int **in** **Stack**.push (**Integer** (op1 + op2)) stack; **No\_Action**

| **Minus** -> **let** op2=self#pop\_int **and** op1=self#pop\_int **in** **Stack**.push (**Integer** (op1 - op2)) stack; **No\_Action**

| **Times** -> **let** op2=self#pop\_int **and** op1=self#pop\_int **in** **Stack**.push (**Integer** (op1 \* op2)) stack; **No\_Action**

| **Divide** -> **let** op2=self#pop\_int **and** op1=self#pop\_int **in** **Stack**.push (**Integer** (op1 / op2)) stack; **No\_Action**

| **Mod** -> **let** op2=self#pop\_int **and** op1=self#pop\_int **in** **Stack**.push (**Integer** (op1 **mod** op2)) stack; **No\_Action**

| **Power** -> **let** op2=self#pop\_int **and** op1=self#pop\_int **in** **Stack**.push (**Integer** (int\_of\_float((float\_of\_int(op1)) \*\* (float\_of\_int(op2))))) stack; **No\_Action**

*(\* boolean arithmetics \*)*

| **And** -> **let** op2=self#pop\_bool **and** op1=self#pop\_bool **in** **Stack**.push (**Boolean** (op1 && op2)) stack; **No\_Action**

| **Or** -> **let** op2=self#pop\_bool **and** op1=self#pop\_bool **in** **Stack**.push (**Boolean** (op1 || op2)) stack; **No\_Action**

| **Not** -> **let** op=self#pop\_bool **in** **Stack**.push (**Boolean** (not op)) stack; **No\_Action**

*(\* conditions \*)*

| **Less** -> **let** op2=self#pop\_int **and** op1=self#pop\_int **in** **Stack**.push (**Boolean** (op1 < op2)) stack; **No\_Action**

| **Greater** -> **let** op2=self#pop\_int **and** op1=self#pop\_int **in** **Stack**.push (**Boolean** (op1 > op2)) stack; **No\_Action**

| **Equal** -> **let** op2=self#pop\_int **and** op1=self#pop\_int **in** **Stack**.push (**Boolean** (op1 = op2)) stack; **No\_Action**

*(\* call anothe sub\*)*

| **Call**(name) -> **begin**

**Stack**.push (current\_sub, (instruction\_pointer+1)) call\_stack;

current\_sub <- name;

instruction\_pointer <- -1

**end**;

**No\_Action**

*(\* variables \*)*

| **Store**(varName) -> **if** **Stack**.is\_empty stack **then** self#freeze "Nothing to store";

**let** op = **Stack**.pop stack **in** **Hashtbl**.replace vars varName op;

**No\_Action**

| **Read**(varName) -> **if** not (**Hashtbl**.mem vars varName) **then** self#freeze "Variable not defined";

**let** op = **Hashtbl**.find vars varName **in**

**Stack**.push op stack;

**No\_Action**

*(\* stack manipulation \*)*

| **Drop** -> ignore(**Stack**.pop stack); **No\_Action**

| **Dropall** -> **Stack**.clear stack; **No\_Action**

| **Dup** -> **let** op=**Stack**.top stack **in** **Stack**.push op stack; **No\_Action**

| **Swap** -> **let** op2=**Stack**.pop stack **and** op1=**Stack**.pop stack **in** **begin** **Stack**.push op2 stack; **Stack**.push op1 stack **end**; **No\_Action**

| **Over** -> **let** op2=**Stack**.pop stack **and** op1=**Stack**.top stack **in** **begin** **Stack**.push op1 stack; **Stack**.push op2 stack **end**; **No\_Action**

| **Rot** -> **let** op3=**Stack**.pop stack **and** op2=**Stack**.top stack **and** op1=**Stack**.top stack **in** **begin** **Stack**.push op2 stack; **Stack**.push op3 stack; **Stack**.push op1 stack **end**; **No\_Action**

*(\* game specific operations \*)*

| **Move** -> **let** direction=self#pop\_int **in** direction\_of\_the\_body <- direction; moving <- true; **No\_Action**

| **Stop** -> moving <- false; **No\_Action**

| **Shoot** -> **let** direction=self#pop\_int **and** distance=self#pop\_int **in**

direction\_of\_the\_gun <- direction;

**Stack**.push (**Boolean** (gun\_cooldown=0)) stack;

**if** gun\_cooldown>0

**then** **No\_Action**

**else** (gun\_cooldown<-10; **Do\_Shoot**(direction, distance))

| **Look** -> **let** direction=self#pop\_int **in**

direction\_of\_the\_gun <- direction **mod** 360;

**if** direction\_of\_the\_gun > 180 **then** direction\_of\_the\_gun <- direction\_of\_the\_gun-360;

**Do\_Look**(direction\_of\_the\_gun)

| **IsFoe** -> **let** flag=self#pop\_flag **in** **Stack**.push (**Boolean** (flag=**Foe**)) stack; **No\_Action**

| **IsAlly** -> **let** flag=self#pop\_flag **in** **Stack**.push (**Boolean** (flag=**Ally**)) stack; **No\_Action**

| **IsWall** -> **let** flag=self#pop\_flag **in** **Stack**.push (**Boolean** (flag=**Wall**)) stack; **No\_Action**

| **GetHealth** -> **Stack**.push (**Integer**(health)) stack; **No\_Action**

| **Wait** -> ticks\_to\_wait <- self#pop\_int; **No\_Action**

*(\* TO DO! get random int between min and max \*)*

| **Random** -> **let** max=self#pop\_int **and** min=self#pop\_int **in** **Stack**.push (**Integer**(**Random**.int (max - min + 1) + min)) stack; **No\_Action**

*(\* jumps \*)*

| **AbsJump**(x) -> instruction\_pointer <- x-1; **No\_Action**

| **AbsJumpIf**(x) -> **if** self#pop\_bool **then** instruction\_pointer <- x-1; **No\_Action**

| \_ -> **No\_Action**

**in**

instruction\_pointer <- instruction\_pointer+1;

action

**end**

**end**

**method** print\_current\_pos =

**begin**

print\_endline drone\_name;

print\_float x\_position;

print\_endline "";

print\_float y\_position;

print\_endline "";

print\_endline "Direction: ";

print\_int direction\_of\_the\_body;

print\_endline "";

print\_endline "Gun Direction: ";

print\_int direction\_of\_the\_gun;

print\_endline "";

print\_endline "Health: ";

print\_int health;

print\_endline "";

print\_endline "team\_id: ";

print\_int team\_id;

print\_endline "";

print\_endline "";

**end**

**method** freeze explanation =

brain\_dead <- true;

reason\_for\_coma <- explanation;

**raise** (**Error\_in\_AI** (explanation, current\_sub, instruction\_pointer));

**method** print\_current\_state =

**let** sub\_name = (**if** current\_sub="--" **then** "" **else** current\_sub) **in**

**let** body = (**Hashtbl**.find subs current\_sub) **in**

**let** bc = **Array**.get body instruction\_pointer **in**

fprintf debug\_out\_file "%4d %20s[%3d] %20s |" tick\_counter sub\_name instruction\_pointer (string\_of\_bytecode bc);

**let** stack\_copy = **Stack**.copy stack **in**

**let** cnt = ref 1 **in**

**while** (!cnt < 10) && (not (**Stack**.is\_empty stack\_copy)) **do**

**let** op = **Stack**.pop stack\_copy **in**

fprintf debug\_out\_file " %s" (string\_of\_operand op);

cnt := !cnt +1

**done**;

**if** (**Stack**.is\_empty stack\_copy) **then**

fprintf debug\_out\_file " EOS**\n**"

**else**

fprintf debug\_out\_file " ...**\n**";

flush debug\_out\_file

*(\* for each shoot update bullet capacity and push boolean on the stack \*)*

**method** update\_bullet\_load =

**begin**

*(\* shoot \*)*

**if** bullet\_capacity > 0

**then**

**begin**

bullet\_capacity <- bullet\_capacity - 1;

has\_bullet <- true;

**end**

*(\* no bullet \*)*

**else**

has\_bullet <- false;

**end**;

has\_bullet

**end**;;

### 5. Arena.ml

**open** **Drone**;;

**open** **Printf**;;

**open** **Bullet**;;

**open** **Ast**;;

**open** **Utils**;;

**open** **Gui**;;

**class** arena =

**object** (self)

**val** **mutable** drones : drone list = []

**val** **mutable** bullets : bullet list = []

**val** **mutable** arena\_gui = **new** gui

**val** **mutable** gui\_enabled = true

**val** **mutable** debug\_mode = false

**val** **mutable** look\_range = 30 *(\* +30 and -30 on the given degree \*)*

**val** **mutable** bullet\_speed = 5

**val** **mutable** drone\_speed = 1

**val** **mutable** area\_map\_x = 1000

**val** **mutable** area\_map\_y = 1000

**val** **mutable** team\_counter = 0

**val** **mutable** gathering\_team = false

**method** disable\_gui = gui\_enabled<-false

**method** set\_debug\_mode mode = debug\_mode <- mode

**method** load file\_name =

**let** d = **new** drone **in** **begin**

d#load file\_name;

d#belongs\_to\_team team\_counter;

**if** not gathering\_team **then** team\_counter <- team\_counter+1;

**if** debug\_mode **then** **begin**

**let** decompiled\_file = open\_out (file\_name ^ ".decompiled") **in**

d#decompile decompiled\_file;

close\_out decompiled\_file;

d#set\_debug\_output (open\_out (file\_name ^ ".debug"))

**end**;

drones <- d :: drones

**end**

**method** get\_drone\_count = **List**.length drones;

**method** add\_bullet dist dire shoot\_d =

**let** b = **new** bullet **in**

b#init shoot\_d#get\_x\_position shoot\_d#get\_y\_position dire dist;

bullets <- b :: bullets

**method** run =

**if** gui\_enabled **then** arena\_gui#drawArena;

**let** steps = ref 1 **in**

**while** (self#step > 1) && (!steps < 2000) **do**

incr steps

**done**;

printf "Results:**\n**";

**List**.iter (**fun** d ->

printf "%s: %s**\n**" d#get\_drone\_name

(**if** d#is\_brain\_dead **then**

("brain dead after " ^ (string\_of\_int d#get\_ai\_ticks) ^ " ticks with explanation: " ^ d#get\_reason\_for\_coma)

**else** **if** not d#is\_alive **then**

("died after " ^ (string\_of\_int d#get\_ai\_ticks) ^ " ticks")

**else**

("still alive with " ^ (string\_of\_int d#get\_health) ^ "% of health ")

)

) drones

*(\* get a distance to the wall in the exact direction of the drone's look \*)*

**method** look\_wall dire d\_look=

**let** x=d\_look#get\_x\_position **and** y=d\_look#get\_y\_position **in**

**let** md = dire **mod** 360 **in**

**let** rd = radian\_of\_degree md **in**

**let** dh = max (int\_of\_float ((0. -. x) /. (cos rd))) (int\_of\_float ((1000. -. x) /. (cos rd))) **in**

**let** dv = max (int\_of\_float ((0. -. y) /. (sin rd))) (int\_of\_float ((1000. -. y) /. (sin rd))) **in**

**let** dist = **if** md=0 || md=180 **then** dh

**else** **if** md=90 || md=270 **then** dv

**else** min dh dv **in**

d\_look#found\_target dist dire **Wall**

**method** explosion b d =

**let** d\_x=d#get\_x\_position **and** d\_y=d#get\_y\_position **and** exp\_x=b#get\_pos\_x **and** exp\_y=b#get\_pos\_y **in**

**let** dist = distance (d\_x, d\_y, exp\_x, exp\_y) **in**

**if** dist < 50 **then** d#set\_health (d#get\_health - 50 + dist)

**method** step =

**let** live\_drones = ref 0 **in** *(\* to check how many drones are still alive and kicking \*)*

**List**.iter (**fun** active\_drone ->

**if** (active\_drone#is\_alive) && (not active\_drone#is\_brain\_dead) **then** **begin**

incr live\_drones;

**try** (

**let** action = active\_drone#step **in**

**match** action **with**

**No\_Action** -> ()

| **Do\_Shoot**(direction, distance) -> self#add\_bullet distance direction active\_drone

| **Do\_Look**(direction) -> **begin**

self#look\_wall direction active\_drone; *(\* the wall is always visible, and it is always the farthest object from the active drone \*)*

**let** found\_drones = **List**.filter (**fun** d ->

**if** d==active\_drone **then** false *(\* the drone cannot see itself \*)*

**else** **if** not d#is\_alive **then** false *(\* ignore dead drones \*)*

**else** **begin** *(\* check if the drone is in the look range \*)*

**let** angle\_to\_drone = degree\_of\_radian (atan2 (d#get\_y\_position -. active\_drone#get\_y\_position) (d#get\_x\_position -. active\_drone#get\_x\_position) ) **in**

abs (direction - angle\_to\_drone) < look\_range

**end**

) drones **in**

*(\* sort all drones in the look range by the distance from the active drone \*)*

**let** sorted\_found\_drones = **List**.rev(self#sort\_by\_dist active\_drone found\_drones) **in**

*(\* add all found drones into the active drone's stack \*)*

**List**.iter (**fun** d -> active\_drone#found\_target (distance(active\_drone#get\_x\_position, active\_drone#get\_y\_position, d#get\_x\_position, d#get\_y\_position))

(degree\_of\_radian (atan2 (d#get\_y\_position -. active\_drone#get\_y\_position) (d#get\_x\_position -. active\_drone#get\_x\_position) ))

(**if** active\_drone#get\_team\_id=d#get\_team\_id **then** **Ally** **else** **Foe**)

) sorted\_found\_drones

**end**

)

**with** **Error\_in\_AI** (reason, sub, position) -> printf "Drone %s died at %s:%d with explanation: %s**\n**" active\_drone#get\_drone\_name sub position reason

**end**

) drones;

*(\* update position for all drones and bullets \*)*

**List**.iter (**fun** d -> d#move drone\_speed ) drones;

**List**.iter (**fun** b -> b#move bullet\_speed; **if** b#is\_exploded **then** **List**.iter(**fun** d -> self#explosion b d) drones) bullets;

*(\* List.iter (fun d -> d#print\_current\_pos ) drones; \*)*

**if** gui\_enabled **then** **begin**

arena\_gui#clear;

**List**.iter (**fun** d -> arena\_gui#drawDroneDetail (int\_of\_float d#get\_x\_position) (int\_of\_float d#get\_y\_position) (radian\_of\_degree d#get\_moving\_direction) (radian\_of\_degree d#get\_direction\_of\_the\_gun) d#get\_drone\_name d#get\_health d#get\_team\_id d#get\_ai\_ticks d#get\_moving\_status d#get\_reason\_for\_coma d#get\_gun\_cooldown) drones;

**List**.iter (**fun** b -> **if**(b#is\_exploded) **then** arena\_gui#drawExplode (int\_of\_float b#get\_pos\_x) (int\_of\_float b#get\_pos\_y) **else** arena\_gui#drawBullet (int\_of\_float b#get\_pos\_x) (int\_of\_float b#get\_pos\_y)) bullets;

arena\_gui#wait;

**end**;

*(\* remove all exploded bullets from the arena \*)*

bullets <- **List**.filter (**fun** b -> not b#is\_exploded) bullets;

!live\_drones

**method** ins d drone d\_list =

**let** **rec** insert d e elements =

**match** elements **with**

[] -> [e]

| head :: tail -> **if** distance (d#get\_x\_position, d#get\_y\_position, e#get\_x\_position, e#get\_y\_position) <=

distance (head#get\_x\_position, head#get\_y\_position, d#get\_x\_position, d#get\_y\_position)

**then** e :: elements

**else** head :: insert d e tail

**in**

insert d drone d\_list

**method** sort\_by\_dist d d\_list=

**let** **rec** sort d elements =

**match** elements **with**

[] -> []

| head :: tail -> self#ins d head (sort d tail)

**in**

sort d d\_list

**method** start\_a\_team =

team\_counter <- team\_counter+1;

gathering\_team <- true

**end**;;

### 6. main.ml

**open** **Arena**;;  
**open** **Printf**;;  
**open** **Utils**;;  
  
  
**let** main =  
 print\_string "The Drone War**\n**The class project for COMS W4115 Programming Languages and Translators**\n**Columbia University, Fall 2012**\n**\  
 Professor:**\t**Stephen A. Edwards**\n**\  
 Students:**\t**George Brink (gb2280)**\n**\  
 **\t\t**Xiang Yao (xy2191)**\n**\  
 **\t\t**Xiaotong Chen (xc2230)**\n**\  
 **\t\t**Shuo Qiu (sq2144)**\n\n**\  
 ";  
 **Random**.self\_init();  
 **let** cage = **new** arena **in**  
 **Array**.iter (**fun** parameter ->  
 **if** parameter.[0]='-' **then**  
 **begin**  
 **match** parameter.[1] **with**  
 'D' -> cage#set\_debug\_mode true  
 | 't' -> cage#start\_a\_team  
 | 'q' -> cage#disable\_gui  
 | \_ -> print\_endline ("Unknown option " ^parameter);  
 **end**  
 **else**  
 **if** (**Filename**.check\_suffix parameter ".dt" ) || (**Filename**.check\_suffix parameter ".dbt" ) **then**  
 **begin**  
 print\_string "Loading ";  
 print\_string parameter;  
 **try**  
 cage#load parameter;  
 printf " - ok**\n**"  
 **with**  
 **Failure** t -> printf " - failed**\n**%s**\n**" t  
 | **Parse\_failure**(t,l,c) -> printf " - failed**\n**%s at %d:%d**\n**" t l c  
 | **Sys\_error** t -> printf " - file error**\n**%s**\n**" t  
 **end**  
 ) **Sys**.argv;  
 **Random**.self\_init();  
 print\_string ("Loaded " ^ (string\_of\_int cage#get\_drone\_count) ^ " drones**\n**");  
 cage#run;  
 exit 0;;

### 7. gui.ml

**open** **Unix**;;  
  
**class** gui =  
**object** (self)  
 **val** **mutable** info\_x = 0  
 **val** **mutable** info\_y = 0  
 **val** **mutable** size\_x = 0  
 **val** **mutable** size\_y = 0  
 **val** **mutable** max\_x = 0  
 **val** **mutable** max\_y = 0  
 **val** **mutable** temp\_x = 0  
 **val** **mutable** temp\_y = 0  
 **val** **mutable** counter = 0  
  
  
  
  
 **method** drawArena=  
 **Graphics**.open\_graph "";  
 **Graphics**.set\_window\_title "Arena";  
 **Graphics**.display\_mode false;  
 **Graphics**.remember\_mode true;  
 self#clear  
  
 **method** translate x y=  
 temp\_x <- (20 + x \* size\_x / 1000);  
 temp\_y <- (20 + y \* size\_y / 1000);  
  
  
  
 **method** drawDrone x y z=  
 **Graphics**.set\_color (**Graphics**.blue);  
 self#translate x y;  
 **Graphics**.draw\_circle temp\_x temp\_y 6;  
 **Graphics**.moveto temp\_x temp\_y;  
 **Graphics**.lineto ( int\_of\_float(cos (z)\*.12.) +size\_x) (int\_of\_float(sin (z)\*.12.)+temp\_y);  
  
  
 **method** drawCircleDroneDetail x y z name health=  
 **Graphics**.set\_color (**Graphics**.blue);  
 self#translate x y;  
 **Graphics**.draw\_circle temp\_x temp\_y 6;  
 **Graphics**.moveto temp\_x temp\_y;  
 **Graphics**.lineto ( int\_of\_float(cos (z)\*.12.) +temp\_x) (int\_of\_float(sin (z)\*.12.)+temp\_y);  
 **if** (x+**String**.length(name))>1000 **then** **if** (y+30)>1000 **then** self#translate (x-50) (y-13) **else** self#translate (x-50) (y+13)  
 **else** **if** (y+30)>1000 **then** self#translate (x+13) (y-13) **else** self#translate (x+13) (y+13);  
 **Graphics**.moveto temp\_x temp\_y;  
 **Graphics**.draw\_string name;  
 *(\*self#drawDroneHealth name health;\*)*  
 **if** health=0 **then** (self#drawDroneDead x y);  
  
 **method** drawDroneDetail x y body\_direc gun\_direc name health team\_id ai\_ticks moving\_status reason\_for\_coma gun\_cooldown=  
 self#drawDroneColor team\_id;  
 self#translate x y;  
 self#drawDroneBody x y body\_direc; *(\*draw the body of the drone \*)*  
 **Graphics**.moveto temp\_x temp\_y;  
 **Graphics**.lineto ( int\_of\_float(cos (gun\_direc)\*.15.) +temp\_x) (int\_of\_float(sin (gun\_direc)\*.15.)+temp\_y); *(\*draw the gun of the drone \*)*  
 **if** (x+7\***String**.length(name))>1000 **then** **if** (y+30)>1000 **then** self#translate (x-7\***String**.length(name)) (y-23) **else** self#translate (x-7\***String**.length(name)) (y+16)  
 **else** **if** (y+30)>1000 **then** self#translate (x+13) (y-23) **else** self#translate (x+13) (y+16);  
 **Graphics**.moveto temp\_x temp\_y;  
 **Graphics**.draw\_string name; *(\*draw the name of the drone \*)*  
 **Graphics**.moveto temp\_x (temp\_y-10);  
 **Graphics**.draw\_string (string\_of\_int health); *(\*draw the name of the drone \*)*  
 self#drawDroneInfo name health team\_id ai\_ticks moving\_status reason\_for\_coma gun\_cooldown; *(\*draw the information of the drone \*)*  
 **if** health=0 **then** (self#drawDroneDead x y); *(\*draw the deadbody of the drone \*)*  
  
 **method** drawDroneBody x y body\_direc=  
 self#translate x y;  
 **let** pi = 4. \*. atan 1. **in**  
 **let** x1=int\_of\_float(cos (body\_direc)\*.10.) +temp\_x **in**  
 **let** y1=int\_of\_float(sin (body\_direc)\*.10.) +temp\_y **in**  
 **let** x2=int\_of\_float(cos (body\_direc +. (140.\*. pi /.180.))\*.10.) +temp\_x **in**  
 **let** y2=int\_of\_float(sin (body\_direc +. (140.\*. pi /.180.))\*.10.) +temp\_y **in**  
 **let** x3=int\_of\_float(cos (body\_direc +. (220.\*. pi /.180.))\*.10.) +temp\_x **in**  
 **let** y3=int\_of\_float(sin (body\_direc +. (220.\*. pi /.180.))\*.10.) +temp\_y **in**  
 **Graphics**.draw\_poly [|(x1,y1);(x2,y2);(x3,y3)|];  
  
 **method** drawDroneInfo name health team\_id ai\_ticks moving\_status reason\_for\_coma gun\_cooldown=  
 **Graphics**.set\_color (10494192);  
 info\_y <- (info\_y-15);  
 **Graphics**.moveto info\_x info\_y;  
 **Graphics**.draw\_string name;  
 **Graphics**.set\_color (**Graphics**.black);  
 info\_y <- (info\_y-10);  
 **Graphics**.moveto info\_x info\_y;  
 **Graphics**.draw\_string "Team ID: ";  
 **Graphics**.draw\_string (string\_of\_int team\_id);  
 info\_y <- (info\_y-10);  
 **Graphics**.moveto info\_x info\_y;  
 **Graphics**.draw\_string "Health: ";  
 **Graphics**.draw\_string (string\_of\_int health);  
 info\_y <- (info\_y-10);  
 **Graphics**.moveto info\_x info\_y;  
 **Graphics**.draw\_string "AI Ticks: ";  
 **Graphics**.draw\_string (string\_of\_int ai\_ticks);  
 info\_y <- (info\_y-10);  
 **Graphics**.moveto info\_x info\_y;  
 **Graphics**.draw\_string "Moving: ";  
 **Graphics**.draw\_string (string\_of\_bool moving\_status);  
 info\_y <- (info\_y-10);  
 **Graphics**.moveto info\_x info\_y;  
 **Graphics**.draw\_string "Reason for coma: ";  
 **if** reason\_for\_coma="" **then** **Graphics**.draw\_string "Not coma yet" **else** **Graphics**.draw\_string reason\_for\_coma;  
 info\_y <- (info\_y-10);  
 **Graphics**.moveto info\_x info\_y;  
 **Graphics**.draw\_string "Gun cooldown: ";  
 **Graphics**.draw\_string (string\_of\_int gun\_cooldown);  
  
 **method** drawDroneDead x y=  
 **Graphics**.set\_color (**Graphics**.red);  
 self#translate x y;  
 **Graphics**.moveto (temp\_x-7) (temp\_y+7);  
 **Graphics**.lineto (temp\_x+7) (temp\_y-7);  
 **Graphics**.moveto (temp\_x-7) (temp\_y-7);  
 **Graphics**.lineto (temp\_x+7) (temp\_y+7);  
  
 **method** drawDroneColor x=  
 **match** x **with**  
 0 -> **Graphics**.set\_color (**Graphics**.red)  
 | 1 -> **Graphics**.set\_color (**Graphics**.green)  
 | 2 -> **Graphics**.set\_color (**Graphics**.blue)  
 | 3 -> **Graphics**.set\_color (10506797)  
 | 4 -> **Graphics**.set\_color (**Graphics**.cyan)  
 | 5 -> **Graphics**.set\_color (**Graphics**.magenta)  
 | 6 -> **Graphics**.set\_color (16744228)  
 | 7 -> **Graphics**.set\_color (16759055)  
 | 8 -> **Graphics**.set\_color (13487360)  
 | 9 -> **Graphics**.set\_color (13445520)  
 | 10 -> **Graphics**.set\_color (12092939)  
 | 11 -> **Graphics**.set\_color (9005261)  
 | 12 -> **Graphics**.set\_color (9132544)  
 | 13 -> **Graphics**.set\_color (5577355)  
 | 14 -> **Graphics**.set\_color (128)  
 | \_ -> **Graphics**.set\_color (**Graphics**.black)  
  
 **method** drawBullet x y=  
 **Graphics**.set\_color (**Graphics**.black);  
 self#translate x y;  
 **Graphics**.fill\_circle temp\_x temp\_y 4;  
  
 **method** drawExplode x y=  
 self#translate x y;  
 **Graphics**.set\_color (14423100);  
 **Graphics**.draw\_circle temp\_x temp\_y 10;  
 **Graphics**.set\_color (15597568);  
 **Graphics**.draw\_circle temp\_x temp\_y 20;  
 **Graphics**.set\_color (15608876);  
 **Graphics**.draw\_circle temp\_x temp\_y 30;  
 **Graphics**.set\_color (15613952);  
 **Graphics**.draw\_circle temp\_x temp\_y 40;  
 **Graphics**.set\_color (15627776);  
 **Graphics**.draw\_circle temp\_x temp\_y 50;  
  
   
 **method** clear=  
 **Graphics**.clear\_graph ();  
 **Graphics**.set\_color (**Graphics**.black);  
 max\_x <- **Graphics**.size\_x();  
 max\_y <- **Graphics**.size\_y();  
 info\_x <-(max\_x-190);  
 info\_y <-(max\_y-25);  
 size\_x <-(max\_x-220);  
 size\_y <-(max\_y-40);  
 counter <- (counter+1);  
 **Graphics**.draw\_rect 20 20 size\_x size\_y;  
 **Graphics**.moveto info\_x (max\_y-30);  
 **Graphics**.draw\_string "Total Ticks: ";  
 **Graphics**.draw\_string (string\_of\_int counter);  
  
 **method** wait=  
 **Graphics**.synchronize();  
 *(\*let s = Graphics.wait\_next\_event [Graphics.Button\_down;Graphics.Key\_pressed] in if s.Graphics.button*  
 *then Graphics.set\_color (Graphics.red); \*)*  
 **let** tt = **Unix**.gettimeofday() **in**  
 **while** **Unix**.gettimeofday() < tt +. 0.05 **do** () **done**

### 8. bullet.ml

**open** **Utils**;;  
  
**class** bullet =  
 **object** (self)  
  
 **val** **mutable** direction = 0  
 **val** **mutable** x\_position = 0.  
 **val** **mutable** y\_position = 0.  
 **val** **mutable** distance\_to\_fly = 0  
 **val** **mutable** distance\_traveled = 0  
  
 **val** **mutable** start\_x\_position = 0.  
 **val** **mutable** start\_y\_position = 0.  
 **val** **mutable** exploded = false  
  
 **method** get\_pos\_x = x\_position  
  
 **method** get\_pos\_y = y\_position  
  
 **method** get\_direction = direction  
  
 **method** is\_exploded = exploded  
  
 **method** init x y dir dist =  
 start\_x\_position <- x;  
 x\_position <- x;  
 start\_y\_position <- y;  
 y\_position <- y;  
 direction <- dir;  
 distance\_to\_fly <- min dist 1000  
  
  
 **method** move speed =  
 y\_position <- y\_position +. (float\_of\_int(speed) \*. (sin (float\_of\_int(direction) \*. pi /. 180.)));  
 x\_position <- x\_position +. (float\_of\_int(speed) \*. (cos (float\_of\_int(direction) \*. pi /. 180.)));  
 distance\_traveled <- distance(x\_position, y\_position, start\_x\_position, start\_y\_position);  
 exploded <- (x\_position > 1000.) || (x\_position < 0.) || (y\_position > 1000.) || (y\_position < 0.);  
 **if** exploded  
 **then** self#update\_position\_if\_flew\_out\_of\_arena  
 **else** exploded <- distance\_traveled >= distance\_to\_fly  
  
  
 **method** update\_position\_if\_flew\_out\_of\_arena =  
 **begin**  
 **if** x\_position > 1000. **then** x\_position <- 1000.;  
 **if** x\_position < 0. **then** x\_position <- 0.;  
 **if** y\_position > 1000. **then** y\_position <- 1000.;  
 **if** y\_position < 0. **then** y\_position <- 0.;  
 **end**  
  
**end**;;

### 9. utils.ml

**exception** **Parse\_failure** **of** string \* int \* int;;  
  
**let** pi = 4. \*. atan 1.;;  
  
**let** distance(x1, y1, x2, y2) =  
 int\_of\_float(sqrt((x1 -. x2)\*.(x1 -. x2) +. (y1 -. y2)\*.(y1 -. y2)));;  
  
**let** radian\_of\_degree angle =  
 float\_of\_int(angle) \*. pi /. 180.;;  
  
**let** degree\_of\_radian angle =  
 int\_of\_float( angle \*. 180. /. pi );;

### 

### 10. scanner\_dbt.mll (George Brink’s individual contribution)

**open** **Parser\_dbt**;;  
**open** **String**;;  
**open** **Lexing**;;  
  
**let** create\_hashtable size init =  
 **let** tbl = **Hashtbl**.create size **in**  
 **List**.iter (**fun** (key, data) -> **Hashtbl**.add tbl key data) init;  
 tbl  
  
**let** keyword\_table =  
 create\_hashtable 8 [  
 ("if", **IF**);  
 ("then", **THEN**);  
 ("else", **ELSE**);  
 ("do", **DO**);  
 ("loop", **LOOP**);  
 ("while", **WHILE**);  
 ("until", **UNTIL**);  
 ("exit", **EXIT**);  
 ("sub", **SUB**);  
 ("function", **FUNCTION**);  
 ("call", **CALL**);  
 ("end", **END**);  
 ("for", **FOR**);  
 ("to", **TO**);  
 ("step", **STEP**);  
 ("next", **NEXT**);  
 ("goto", **GOTO**);  
 ("true", **BOOL**(true));  
 ("false", **BOOL**(false));  
 ("and", **AND**);  
 ("or", **OR**);  
 ("not", **NOT**);  
  
 ("sleep", **SLEEP**);  
 ("move", **MOVE**);  
 ("stop", **STOP**);  
 ("shoot", **SHOOT**);  
 ("rnd", **RANDOM**);  
 ("health", **HEALTH**);  
  
 ("startscan", **STARTSCAN**);  
 ("nextscan", **NEXTSCAN**);  
 (".iswall", **ISWALL**);  
 (".isfoe", **ISFOE**);  
 (".isally", **ISALLY**);  
 (".distance", **DISTANCE**);  
 (".direction", **DIRECTION**);  
 ]  
  
**exception** **Unknown\_token** **of** string \* int \* int;;  
  
**let** incr\_lineno lexbuf =  
 **let** pos = lexbuf.lex\_curr\_p **in**  
 lexbuf.lex\_curr\_p <- { pos **with**  
 pos\_lnum = pos.pos\_lnum + 1;  
 pos\_bol = pos.pos\_cnum;  
 }  
}  
  
**let** digit = ['0' - '9']  
**let** id = ['a'-'z' 'A'-'Z']['a'-'z' 'A'-'Z' '0'-'9']\* | '.' ['a'-'z' 'A'-'Z']+  
**let** space = [' ' '\t' '\r']  
**let** not\_space = [^ ' ' '\t' '\r']  
  
rule drone\_basic = parse  
 | digit+ **as** inum { **let** num = int\_of\_string inum **in** **INT** num }  
 | id **as** word { **try**  
 **let** token = **Hashtbl**.find keyword\_table (**String**.lowercase word) **in**  
 token  
 **with** **Not\_found** -> **ID** (**String**.lowercase word)  
 }  
 | '(' { **LPAREN** }  
 | ')' { **RPAREN** }  
 | ':' { **COLON** }  
 | ',' { **COMMA** }  
 | '+' { **PLUS** }  
 | '-' { **MINUS** }  
 | '\*' { **TIMES** }  
 | '/' { **DIVIDE** }  
 | '=' { **EQUAL** }  
 | "<>" { **NOT\_EQUAL** }  
 | '<' { **LESS** }  
 | "<=" { **LESS\_EQUAL** }  
 | ">" { **GREATER** }  
 | ">=" { **GREATER\_EQUAL** }  
  
 | '\'' [^ '\n']\* *(\* eat up one-line comments \*)*  
 | space *(\* eat up whitespace \*)*  
 { drone\_basic lexbuf }  
  
 | '\n' { incr\_lineno lexbuf; **CR** }  
  
*(\* | not\_space \* as str { raise (Unknown\_token (str, lexbuf.lex\_curr\_p.pos\_lnum, lexbuf.lex\_start\_p.pos\_cnum-lexbuf.lex\_start\_p.pos\_bol +1) ) } \*)*  
  
 | eof { **EOF** }

### 11. parser\_dbt.mly (George Brink’s individual contribution)

%{  
  
**open** **Ast**;;  
**open** **Printf**;;  
**open** **Lexing**;;  
**open** **Utils**;;  
  
**let** auto\_label\_counter = ref 0;;  
  
**let** make\_label() =  
 incr auto\_label\_counter;  
 ("-" ^ string\_of\_int(!auto\_label\_counter))  
 ;;  
  
  
**let** report\_error error\_starts\_at message =  
 **raise** (**Parse\_failure** (message, error\_starts\_at.pos\_lnum, (error\_starts\_at.pos\_cnum-error\_starts\_at.pos\_bol+1)))  
 ;;  
  
%}  
  
%token **CR**  
%token **IF** **THEN** **ELSE**  
%token **DO** **LOOP** **WHILE** **UNTIL** **EXIT**  
%token **SUB** **FUNCTION** **CALL**  
%token **END**  
%token **FOR** **TO** **STEP** **NEXT**  
%token **GOTO**  
%token <bool> **BOOL**  
%token <string> **ID**  
%token <int> **INT**  
%token **LPAREN** **RPAREN** **COLON** **COMMA**  
%token **PLUS** **MINUS** **TIMES** **DIVIDE**  
%token **EQUAL** **NOT\_EQUAL**  
%token **LESS** **GREATER** **LESS\_EQUAL** **GREATER\_EQUAL**  
%token **AND** **OR** **NOT**  
%token **SLEEP** **MOVE** **STOP** **SHOOT** **RANDOM** **HEALTH**  
%token **STARTSCAN** **NEXTSCAN**  
%token **ISWALL** **ISFOE** **ISALLY** **DISTANCE** **DIRECTION**  
%token **EOF**  
  
%left **AND** **OR** **NOT**  
%left **EQUAL** **NOT\_EQUAL**  
%left **LESS** **GREATER** **LESS\_EQUAL** **GREATER\_EQUAL**  
%left **PLUS** **MINUS**  
%left **TIMES** **DIVIDE**  
  
  
%start drone  
%**type** <**Ast**.sub list> drone  
  
%%  
  
drone:  
 program { **let** main\_sub = { name="--"; body = **List**.rev (fst $1); } **in**  
 main\_sub :: snd $1 }  
  
  
program: {[],[]} /\* at the begining we have nothing \*/  
 | program **CR** { $1 }  
 | program statement { ($2 @ fst $1), snd $1 }  
 | program compaund\_statement { ($2 @ fst $1), snd $1 }  
 | program sub { fst $1, ($2 :: snd $1) } /\* add user **function** **to** the list **of** subs \*/  
  
  
statements:  
 /\* nothing \*/ { [] }  
 | statements **CR** { $1 }  
 | statements statement { $2 @ $1 }  
 | statements compaund\_statement { $2 @ $1 }  
  
  
statement:  
 **ID** **EQUAL** math\_expr **CR** { **Store**($1) :: $3 }  
 | **EXIT** **DO** **CR** { [ **Jump**("--ExitDo") ] }  
 | **EXIT** **FOR** **CR** { [ **Jump**("--ExitFor") ] }  
 | **GOTO** **ID** **CR** { [ **Jump**($2) ] }  
 | **ID** **COLON** { [ **Label**($1) ] }  
 | **CALL** **ID** **LPAREN** parameters **RPAREN** **CR** { **Call**($2) :: $4 }  
 | **CALL** **SLEEP** **LPAREN** math\_expr **RPAREN** **CR** { **Wait** :: $4 }  
 | **CALL** **MOVE** **LPAREN** math\_expr **RPAREN** **CR** { **Move** :: $4 }  
 | **CALL** **STOP** **LPAREN** **RPAREN** **CR** { [ **Stop** ] }  
 | **CALL** **SHOOT** **LPAREN** math\_expr **COMMA** math\_expr **RPAREN** **CR** { **Drop** :: **Shoot** :: ($4 @ $6) }  
 | **ID** **EQUAL** **STARTSCAN** **LPAREN** math\_expr **RPAREN** **CR** { [ **Store**($1^".distance"); **Store**($1^".direction"); **Store**($1^".flag"); **Look** ] @ $5 }  
 | **ID** **EQUAL** **NEXTSCAN** **LPAREN** **RPAREN** **CR** { [ **Store**($1^".distance"); **Store**($1^".direction"); **Store**($1^".flag"); ] }  
 | error **CR** { report\_error (**Parsing**.rhs\_start\_pos 1) "Syntax error" }  
  
  
compaund\_statement:  
 **IF** condition **THEN** statement  
 { **let** lbl = make\_label() **in**  
 **Label**(lbl) :: ( $4 @ ( [ **JumpIf**(lbl) ; **Not** ] @ $2 ) )  
 }  
 | **IF** condition **THEN** **CR** statements **END** **IF**  
 { **let** lbl = make\_label() **in**  
 **Label**(lbl) :: ( $5 @ ( [ **JumpIf**(lbl) ; **Not** ] @ $2 ) )  
 }  
 | **IF** condition **THEN** **CR** statements **ELSE** **CR** statements **END** **IF**  
 { **let** lblTrue = make\_label() **in**  
 **let** lblEndIf = make\_label() **in**  
 **Label**(lblEndIf) :: ( $5 @ (**Label**(lblTrue) :: **Jump**(lblEndIf) :: ( $8 @ ( **JumpIf**(lblTrue) :: $2) ) ) )  
 }  
 | **DO** **WHILE** condition **CR** statements **LOOP**  
 { **let** lblStart = make\_label() **and** lblCheckCondition = make\_label() **and** lblDone = make\_label() **in**  
 **let** block = **List**.map (**fun** x -> **match** x **with** **Jump**("--ExitDo") -> **Jump**(lblDone) | \_ -> x) $5 **in**  
 **Label**(lblDone) :: **JumpIf**(lblStart) :: ($3 @ (**Label**(lblCheckCondition) :: (block @ [**Label**(lblStart); **Jump**(lblCheckCondition) ])))  
 }  
 | **DO** statements **LOOP** **WHILE** condition  
 { **let** lblStart = make\_label() **and** lblDone = make\_label() **in**  
 **let** block = **List**.map (**fun** x -> **match** x **with** **Jump**("--ExitDo") -> **Jump**(lblDone) | \_ -> x) $2 **in**  
 **Label**(lblDone) :: **JumpIf**(lblStart) :: ($5 @ (block @ [**Label**(lblStart)]))  
 }  
 | **DO** **UNTIL** condition **CR** statements **LOOP**  
 { **let** lblCheckCondition = make\_label() **and** lblDone = make\_label() **in**  
 **let** block = **List**.map (**fun** x -> **match** x **with** **Jump**("--ExitDo") -> **Jump**(lblDone) | \_ -> x) $5 **in**  
 **Label**(lblDone) :: **Jump**(lblCheckCondition) :: (block @ ( **JumpIf**(lblDone) :: ($3 @ [**Label**(lblCheckCondition)])))  
 }  
 | **DO** statements **LOOP** **UNTIL** condition  
 { **let** lblStart = make\_label() **and** lblDone = make\_label() **in**  
 **let** block = **List**.map (**fun** x -> **match** x **with** **Jump**("--ExitDo") -> **Jump**(lblDone) | \_ -> x) $2 **in**  
 **Label**(lblDone) :: **JumpIf**(lblStart) :: **Not** :: ($5 @ (block @ [**Label**(lblStart)]))  
 }  
 | **FOR** **ID** **EQUAL** math\_expr **TO** math\_expr **CR** statements **NEXT**  
 { **let** lblAgain = make\_label() **and** lblDone = make\_label() **in**  
 **let** block = **List**.map (**fun** x -> **match** x **with** **Jump**("--ExitFor") -> **Jump**(lblDone) | \_ -> x) $8 **in**  
 [**Label**(lblDone); **JumpIf**(lblAgain); **Less**] @ $6 @ [ **Store**($2); **Dup**; **Plus**; **Int**(1); **Read**($2)] @ block @ [**Label**(lblAgain); **Store**($2)] @ $4  
 }  
 | **FOR** **ID** **EQUAL** math\_expr **TO** math\_expr **STEP** math\_expr **CR** statements **NEXT**  
 { **let** lblAgain = make\_label() **and** lblDone = make\_label() **in**  
 **let** block = **List**.map (**fun** x -> **match** x **with** **Jump**("--ExitFor") -> **Jump**(lblDone) | \_ -> x) $10 **in**  
 [**Label**(lblDone); **JumpIf**(lblAgain); **Less**] @ $6 @ [ **Store**($2); **Dup**; **Plus**] @ $8 @ [**Read**($2)] @ block @ [**Label**(lblAgain); **Store**($2)] @ $4  
 }  
  
sub:  
 **SUB** **ID** **LPAREN** args **RPAREN** **CR** statements **END** **SUB** **CR**  
 { **let** read\_arguments = **List**.map (**fun** arg -> **Store**(arg)) $4 **in**  
 **let** sub\_body = **List**.map(**fun** x -> **match** x **with**  
 **Read**(name) -> **if** **List**.exists (**fun** arg -> arg=name) $4 **then** **Read**($2^"-"^name) **else** **Read**(name)  
 | **Store**(name) -> **if** **List**.exists (**fun** arg -> arg=name) $4 **then** **Store**($2^"-"^name) **else** **Store**(name)  
 | \_ -> x) ($7 @ read\_arguments) **in**  
 { name = $2; body = **List**.rev sub\_body; }  
 }  
 | **FUNCTION** **ID** **LPAREN** args **RPAREN** **CR** statements **END** **FUNCTION** **CR**  
 { **let** read\_arguments = **List**.map (**fun** arg -> **Store**(arg)) $4 **in**  
 **let** sub\_body = **List**.map(**fun** x -> **match** x **with**  
 **Read**(name) -> **if** **List**.exists (**fun** arg -> arg=name) $4 **then** **Read**($2^"-"^name) **else** **Read**(name)  
 | **Store**(name) -> **if** **List**.exists (**fun** arg -> arg=name) $4 **then** **Store**($2^"-"^name) **else** **if** name=$2 **then** **Store**($2^"-") **else** **Store**(name)  
 | \_ -> x) ($7 @ read\_arguments) **in**  
 { name = $2; body = **List**.rev (**Read**($2^"-") :: sub\_body); }  
 }  
  
args: { [] }  
 | **ID** { [ $1 ] }  
 | args **COMMA** **ID** { $3 :: $1 }  
  
  
parameters: { [] }  
 | math\_expr { $1 }  
 | parameters **COMMA** math\_expr { $3 @ $1 }  
  
condition:  
 logic\_expr { $1 }  
 | logic\_expr **AND** logic\_expr { **And** :: ($3 @ $1) }  
 | logic\_expr **OR** logic\_expr { **Or** :: ($3 @ $1) }  
 | **NOT** logic\_expr { **Not** :: $2 }  
 | error { report\_error (**Parsing**.rhs\_start\_pos 1) "Malformed logical expression" }  
  
  
logic\_expr:  
 **BOOL** { [ **Bool**($1) ] }  
 | **LPAREN** logic\_expr **RPAREN** { $2 }  
 | math\_expr math\_relation math\_expr { $2 @ ( $3 @ $1) }  
 | **SHOOT** **LPAREN** math\_expr **COMMA** math\_expr **RPAREN** { **Shoot** :: ($3 @ $5) }  
 | **ID** **ISFOE** { [ **IsFoe**; **Read**($1^".flag") ] }  
 | **ID** **ISALLY** { [ **IsAlly**; **Read**($1^".flag") ] }  
 | **ID** **ISWALL** { [ **IsWall**; **Read**($1^".flag") ] }  
  
  
math\_relation:  
 **EQUAL** { [ **Equal** ] }  
 | **NOT\_EQUAL** { [ **Equal** ; **Not** ] }  
 | **LESS** { [ **Less** ] }  
 | **GREATER** { [ **Greater** ] }  
 | **LESS\_EQUAL** { [ **Greater** ; **Not** ] }  
 | **GREATER\_EQUAL** { [ **Less** ; **Not** ] }  
  
  
math\_expr:  
 **INT** { [ **Int**($1) ] }  
 | **ID** **LPAREN** parameters **RPAREN** { **Call**($1) :: $3 }  
 | **ID** { [ **Read**($1) ] }  
 | math\_expr **PLUS** math\_expr { **Plus** :: ( $3 @ $1) }  
 | math\_expr **MINUS** math\_expr { **Minus** :: ( $3 @ $1) }  
 | math\_expr **TIMES** math\_expr { **Times** :: ( $3 @ $1) }  
 | math\_expr **DIVIDE** math\_expr { **Divide** :: ( $3 @ $1) }  
 | **LPAREN** math\_expr **RPAREN** { $2 }  
 | **RANDOM** **LPAREN** math\_expr **COMMA** math\_expr **RPAREN** { **Random** :: ($5 @ $3) }  
 | **HEALTH** **LPAREN** **RPAREN** { [ **GetHealth** ] }  
 | **ID** **DISTANCE** { [ **Read**($1^".distance") ] }  
 | **ID** **DIRECTION** { [ **Read**($1^".direction") ] }  
 | error { report\_error (**Parsing**.rhs\_start\_pos 1) "Malformed math expression" }