CS2006 — *Core War*

Chengyi Lin, Ross Apted and Ben Lovell

December 12, 2011

Contents

Part 1

Running instructions

In order to run, run the executable ./Main [warrior 1] [warrior 2] file, which takes 2 arguments for two warrior files.

Part 2

Introduction

2.1 Aim

The aim of this Project is to implement a core war programming game in which two (or more) programs (known as warriors) battle to control a virtual machine. The warriors are written in a simple programming language called Redcode. The object of the game is to cause the opposing program(s) to terminate, leaving the winning warrior in control of the machine.

2.2 Requirements:

Primary requirements

- Implementing step, which executes each instruction in the virtual machine. — Done
- Implementing drawSystem in Display.hs which draws a visual representation of the core contents. —Done
- Implementing parsers to convert Strings to instructions. —Done
- Extend main so that it reads two files from the command line, parses them, adds them to the core and executes the warriors against each other. —Done

Other things for the project were also tackled:

Secondary Requirements

- Extend addWarrior so that it adds the warrior to a random location in the core. You will need to make sure that it does not overlap with another warrior. —Done

Part 3

Description

The program was separated into logically separate parts. This was to allow modularity in the scraping part and the user interface part.

On the one hand, there was the scraping part. This was modularised by using abstract classes and

3.1 base.py

....

3.2 session.py

....

3.3 Reddit.py

....

3.4 storyview.py

....

Part 4

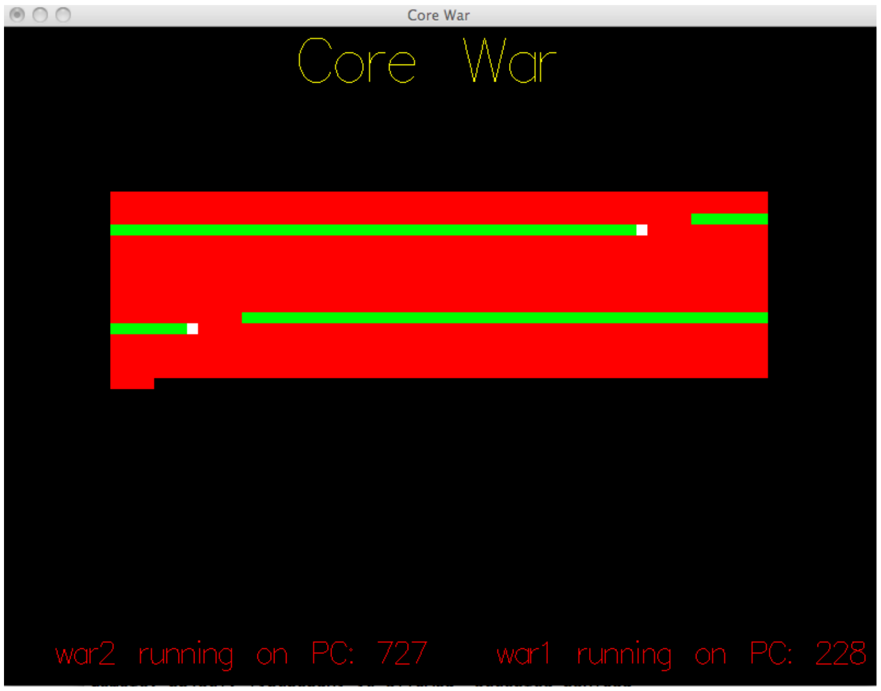
Testing

4.1 How we Tested

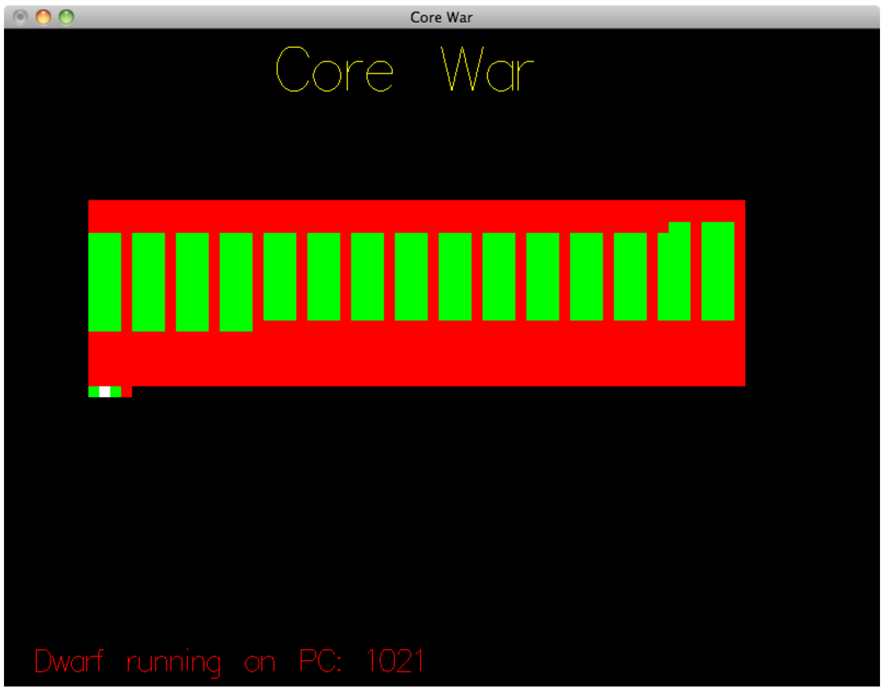
Used command line to run program tying all possible, cases and took screen shots.

4.2 Tests

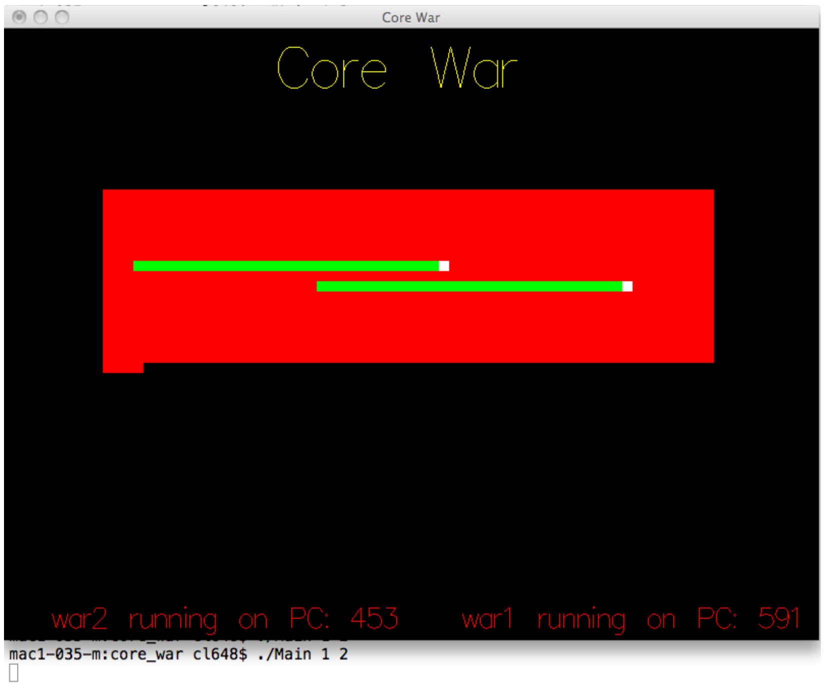
- Basic working, can load two warriors as arguments from command line and run them — Passed



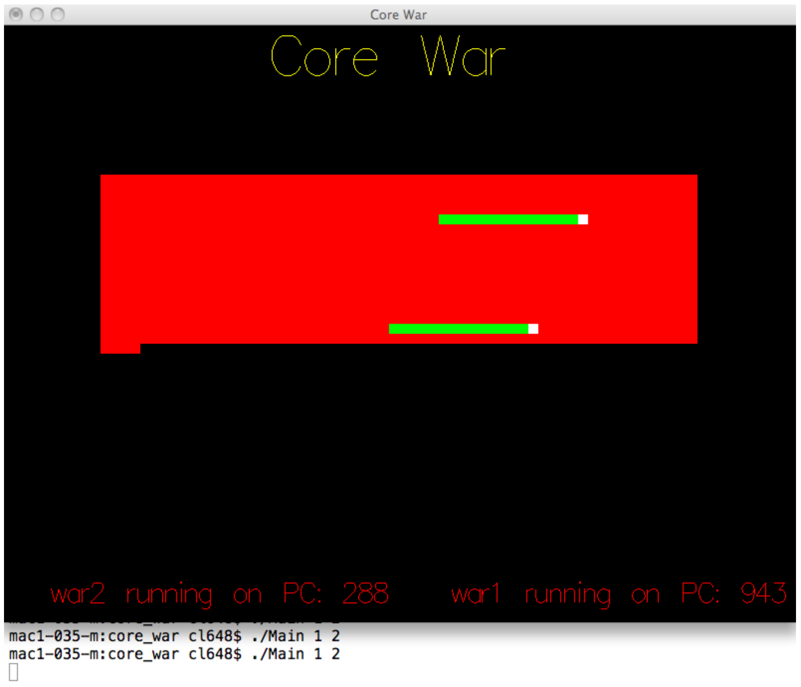
- Run the given test-state which contains two warriors: dwarf and imp, dwarf win — Passed



- Add warriors to random locations — Passed

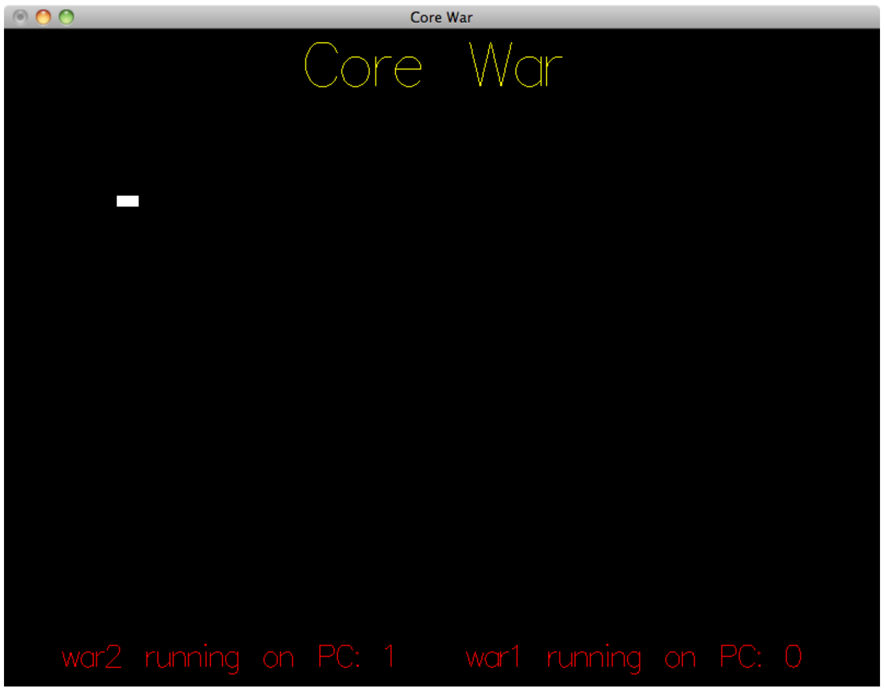


```
mac1-035-m:core_war cl648$ ./Main 1 2
```



```
mac1-035-m:core_war cl648$ ./Main 1 2
mac1-035-m:core_war cl648$ ./Main 1 2
```

- Never set two warriors to same location (Test by setting the core size to 2, and run program several times, and see if any errors happen) — Passed



war2 running on PC: 1 war1 running on PC: 0

Part 5

Problems encountered

Part 6

How we worked as a team

...

Part 7

Conclusion

We successfully implement the core war game, using haskell. And learning quite a lot for using gloss. We worked as a team and met the basic requirements and as well as some additional ones.

Part 8

References

...