10 Submission 2 - Full Soccer Team

10.1 Introduction

The last submission for this assignment involves you taking what you done for the previous submission and expanding it to develop the artificial intelligence of a whole team which is capable of playing soccer. This means that you need to expand your team to handle different game modes, consider opponents positions, generate formations all in aid to develop a team capable of beating your friends and more importantly your enemies.

10.2 Implementation Details

The RoboCup code base comes down to making a choice between two actions; walking to a target or kicking to a target. Ultimately your team needs to perform the decision making to handle making that choice. Therefore, no matter what the game state is your code inside the `select_skill` function needs to return `kickTarget` or `move` with the appropriate parameters.

Feel free to create and add any additional `.py` files as long as they exist somewhere inside the project folder.

Things to consider:

1. Your code only has a limited amount of time to perform task (). If your code exceeds this time, it will not crash but rather the players will fall over frequently or behave abnormally.
2. Make sure you have behaviours that can handle the different game modes, we recommend a simple function that can handle the different groups of game modes.
3. In the first submission you looked to find an optimal assignment of players to positions, now you will need to consider how can we create a set of positions. To this end you we recommend considering the following:
   - Hardcoding an intelligent formation
   - Perhaps your formation should be dependent on where the ball is
4. Another aspect revolves around decision making, what should each team member do and when. This can be handled by asking questions of the world. For example `am I the closest player to the ball` or `is there a teammate nearby`. Incorporating these types of decision making mechanism can be achieved using the following:
   - Finite State Machines
   - Decision Trees
   - If Statements ;)

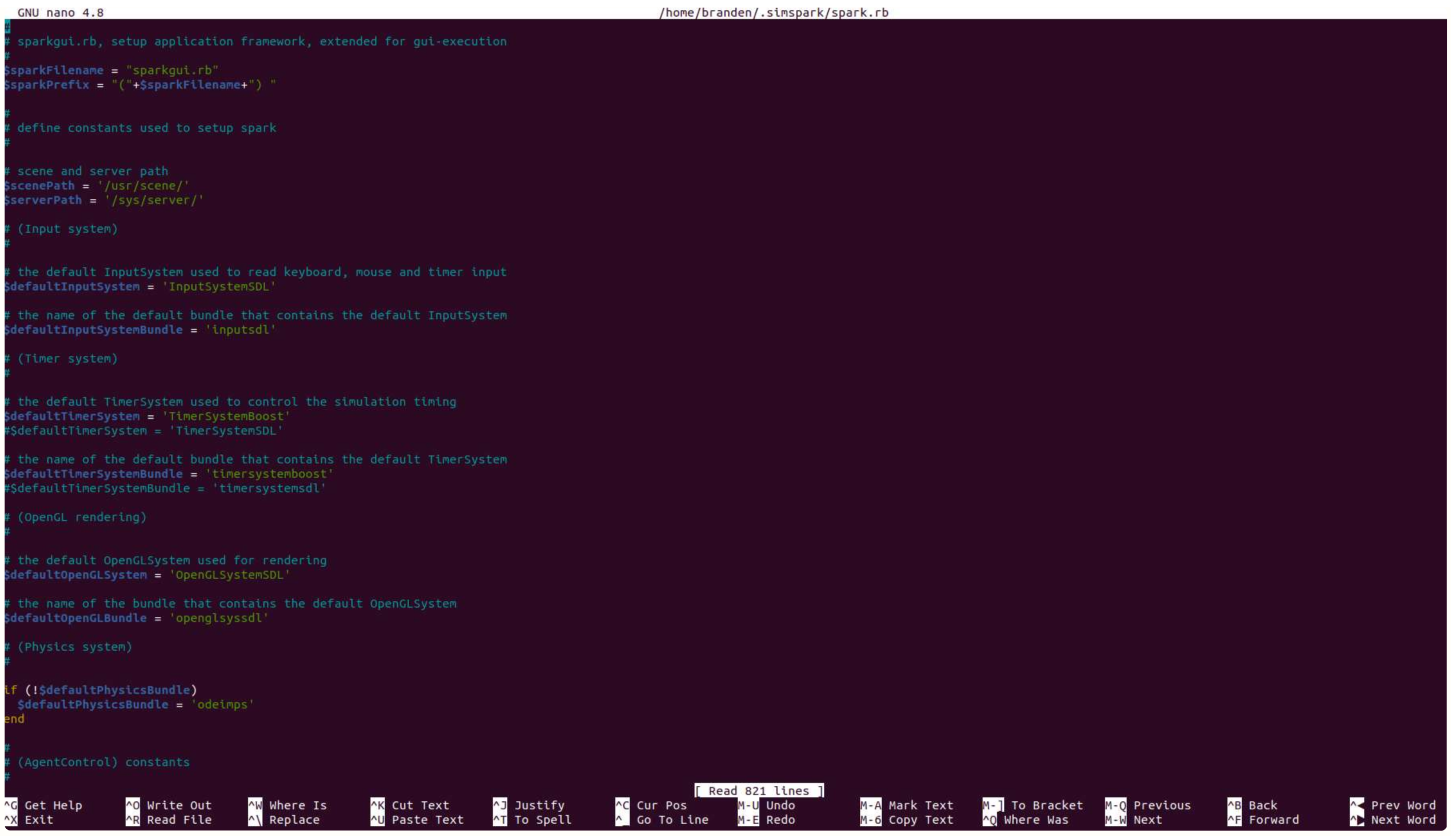
10.3 Fixing Falling Over

The reason why your players are falling over is that your code is too slow to finish in a single server tick. One solution which we will allow is to run the server in asyn mode. Which means the server will wait for all players to send a message before moving on.

To enable this you need to follow the following steps:

1. Inside the same terminal that you would use to run the server type the following command:

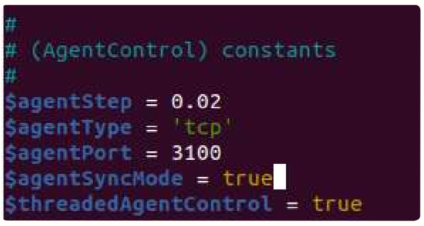
```
nano ~/.simspark/spark.rb
```



```
  GNU nano 4.8                                                /home/branden/.simspark/spark.rb
#
# sparkgui.rb, setup application framework, extended for gui-execution
#
$sparkFilename = "sparkgui.rb"
$sparkPrefix = "("+$sparkFilename+") "

#
# define constants used to setup spark
#

# scene and server path
$scenePath = '/usr/scene/'
$serverPath = '/sys/server/'

# (Input system)
#

# the default InputSystem used to read keyboard, mouse and timer input
$defaultInputSystem = 'InputSystemSDL'

# the name of the default bundle that contains the default InputSystem
$defaultInputSystemBundle = 'inputsdl'

# (Timer system)
#

# the default TimerSystem used to control the simulation timing
$defaultTimerSystem = 'TimerSystemBoost'
#$defaultTimerSystem = 'TimerSystemSDL'

# the name of the default bundle that contains the default TimerSystem
$defaultTimerSystemBundle = 'timersystemboost'
#$defaultTimerSystemBundle = 'timersystemsdl'

# (OpenGL rendering)
#

# the default OpenGLSystem used for rendering
$defaultOpenGLSystem = 'OpenGLSystemSDL'

# the name of the bundle that contains the default OpenGLSystem
$defaultOpenGLBundle = 'openglsyssdl'

# (Physics system)
#

if (!$defaultPhysicsBundle)
  $defaultPhysicsBundle = 'odeimps'
end

#
# (AgentControl) constants
#
                                            [ Read 821 lines ]
^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     M-U Undo       M-A Mark Text   M-] To Bracket  M-Q Previous   ^B Back        ^◀ Prev Word
^X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spell    ^_ Go To Line  M-E Redo       M-6 Copy Text   ^Q Where Was    M-W Next       ^F Forward     ^▶ Next Word
```

This will open the following looking file

2. Use the arrow keys to navigate down the file until you get to `$agentSyncMode = false`. You will need to change this to TRUE as can be seen below



```
#
# (AgentControl) constants
#
$agentStep = 0.02
$agentType = 'tcp'
$agentPort = 3100
$agentSyncMode = true
$threadedAgentControl = true
```

Note for the following instructions the precise key commands will depend on which program you are using as well as if you are using Vscode from inside WSL. The main thing you need to do is execute `Write-Out` followed by `Exit`.

3. Then type `Ctrl-O` to save. It will ask if you are sure then press `Enter`.

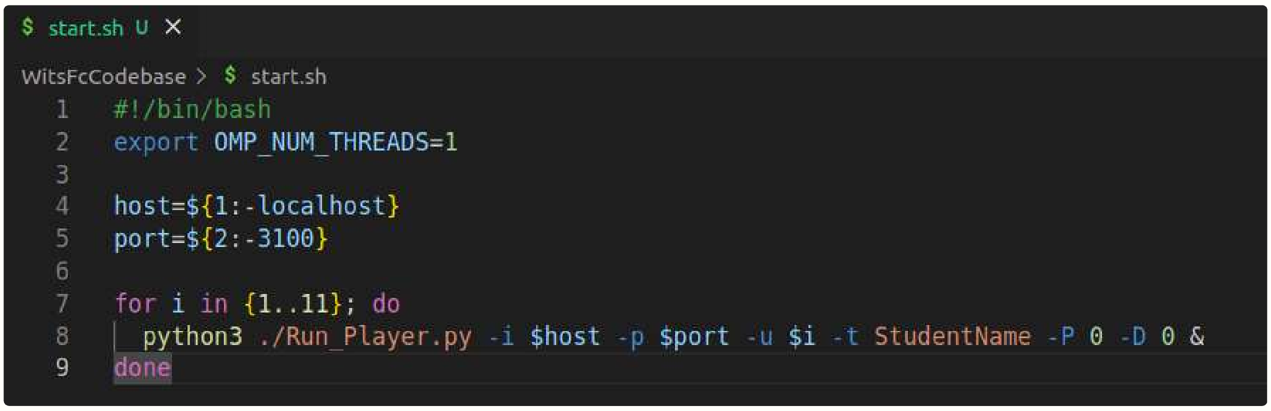
4. Lastly type `Ctrl-X` to close the file.

Now you should be able to run your agents normally, if they are still falling over then there is an issue with your code.

10.4 Submission Requirements

To submit your code

1. First you need to change the team_name. To do this open `start.sh` and modify the string that says `StudentName`. This can be found on line 8, simply replace that with your team name. For simplicity, please use your student number as the teamname.



```
1  #!/bin/bash
2  export OMP_NUM_THREADS=1
3
4  host=${1:-localhost}
5  port=${2:-3100}
6
7  for i in {1..11}; do
8    python3 ./Run_Player.py -i $host -p $port -u $i -t StudentName -P 0 -D 0 &
9  done
```

1. Right click on the `WitsFcCodebase` folder and zip it

2. upload to moodle submission link

10.5 Tournament Structure

Once your submission has been made your team will be added into the pool of other teams. The tournament uses a Swiss style structure which is a popular format used in various competitive settings, including chess, eSports, card games, and academic competitions. It is designed to pair participants against others with similar performance levels, ensuring a fair and competitive experience for all players throughout the tournament.

10.5.1 Key Features of the Swiss Tournament Structure:

1. **Fixed Number of Rounds**:
   - The tournament consists of a predetermined number of rounds, regardless of the number of participants. Typically, the number of rounds is chosen based on the number of participants to ensure that a clear winner can be determined.
2. **No Elimination**:
   - Unlike knockout tournaments, no player is eliminated after a loss. All participants play in each round, allowing everyone to compete in all rounds of the tournament.
3. **Pairing Based on Performance**:
   - In the first round, participants are usually paired randomly or based on seeding (if prior rankings are available). In subsequent rounds, players are paired against opponents with similar scores (e.g., a player with 3 wins and 1 loss will be paired with another player who has 3 wins and 1 loss).
   - The goal is to match players who are performing similarly, ensuring more balanced and competitive matches as the tournament progresses.
4. **Scoring System**:
   - Participants earn points based on their performance in each round (e.g., 1 point for a win, 0.5 points for a draw, and 0 points for a loss). The total score after all rounds determines the final rankings.
   - Tiebreakers, such as the Buchholz system (sum of opponents' scores), may be used to rank players with identical scores.
5. **Final Standings**:
   - After the predetermined number of rounds, participants are ranked based on their total score. The player with the highest score is declared the winner. If there is a tie, tiebreaker methods are applied to determine the final standings.

« 9 Submission 1 - Role Assignment

"**COMS3005A RoboCup Assignment**" was written by Branden Ingram.

This book was built by the bookdown R package.