School of Computer Science

University of KwaZulu-Natal

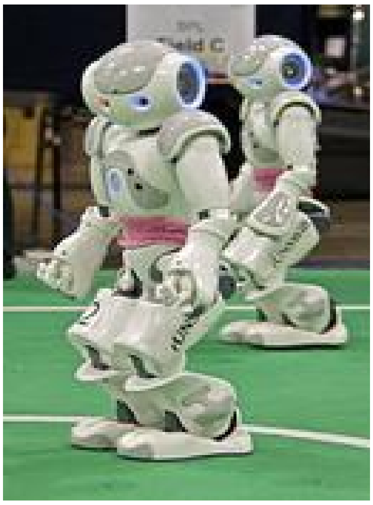
COMP102 W2

Group Assignment - Goal Keeping Agent

Your task is to use the python programming language to develop a **goal-keeper** that could be used within the context of the **RoboCup** competition.

http://en.wikipedia.org/wiki/RoboCup

RoboCup



“RoboCup is an annual international robotics competition founded in 1997. The aim is to promote robotics and AI research... ...a publicly appealing, but formidable challenge...”:

The official goal of the project:

"By the middle of the 21st century, a team of fully autonomous humanoid robot soccer players shall win a soccer game, complying with the official rules of FIFA, against the winner of the most recent World Cup.”

Unfortunately, the entire RoboCup challenge is very complex for an undergrad programming assignment, so we will focus only on one single aspect of football playing: **goal-keeping**. Furthermore, we will simplify this even more by restricting our model to '*spheres-in-a-2D-vacuum',* rather than '*humanoid-players-in-a-3D-world-with-gravity-and-inertia*', etc. At any rate, we will still have some fun!

Rather than playing with robots, which can be very expensive, we can focus on the game-playing model for RoboCup as a **turtle-based simulation**.

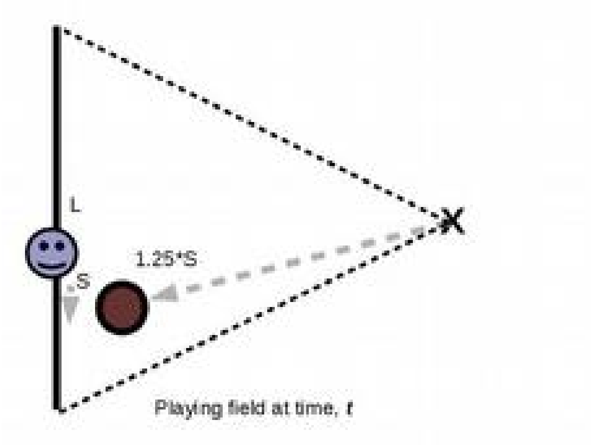
Group Assignment

Your group (3-5 members) are to develop a python turtle simulation of a goal-keeing agent. In simple terms, this agent has to fulfil the following objectives:

1. Observe the playing field (environment), in front of the goal posts for incoming shots.
2. When an incoming shot is perceived, the goal-keeper has to move from its current location to a location that will block the shot from scoring.

Your simulation should include a **2D graphical user interface** (gui), as well as **visualise** the ongoing state of the simulation – ball, goal-keeper, goal posts, and playing field.

The figure below depicts the playing field you are to simulate as an iterative **dynamic environment.** Use this diagram as a guideline for the visualisation aspect of the project:



Note the following simulation constraints:

1. The goal posts are represented by a straight line, length **L.**
2. A **goal** is scored when the **centre** of the ball touches the goal line.
3. The goal keeper may move **left**, or **right**, along the line, at a **constant** speed, **S**.
4. There is **only one ball** (though it may be reused after a goal or save).
5. The ball may only come from one **fixed point**, **X**, in the playing field, but can approach any point along the goal line – i.e. at different angles. The point **X** is perpendicular to the centre of the goal line, and is situated a distance of **L** from the goal line.
6. The ball can travel at one of several different **constant** speeds. The minimum is **0.5*S**, followed by **0.75*S**, **S**, **1.25*S, 1.75*S,** and a maximum speed of **2*S**.
7. During each iteration of the **dynamic** simulation, the **goal-keeper** and the **ball** object may advance one discrete step along their respective paths, until one of the following conditions occur:
    a) **Save** – the keeper reaches the point of intersection between ball trajectory and goal line first.
    b) **Goal** – the ball reaches the point of intersection between ball trajectory and goal line first.

The simulation should start with the keeper at a random point on the goal line. During each run, the ball should approach iteratively (in steps), from a random angle (toward the goal line). For the next run, the keeper should remain at the save/goal location, and the ball returned to point X.

Submission Requirements

Your group is to submit the following in a zip file:

1. zip file containing your source code.
2. A document desribing your simulation, including the **logic**, **environment specifications**, and **simple demo** in the form of a **user guide**.