МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по практической работе №1

по дисциплине «Интеллектуальные системы»

Тема: «Определение местонахождения игрока на поле»

Студент гр. 8382 _____ Мирончик П.Д.

Преподаватель _____ Беляев С.А.

Санкт-Петербург

2022

ЗАДАНИЕ

Необходимо разработать программу, имитирующую игрока виртуального футбола. Программа должна решать следующие задачи:

1. Получить из командной строки начальные координаты игрока и число (скорость вращения) и командой `move` переместить игрока в заданные координаты.

2. После получения сообщения о начале игры вращать игрока с заданной скоростью (поворот на каждом такте). Следует отметить, что движение игрока начнется только после команды рефери `play_on`.

3. По информации, полученной от сервера (сообщение `see`) на каждом такте, вычислить координаты игрока и вывести их в консоль (игрок не должен заранее знать свои координаты, они должны быть вычислены по ориентирам).

4. Разместить на поле дополнительного игрока из команды противника.

5. В процессе вычисления координат игрока (п. 3) одновременно вычислять координаты игрока противника (если он находится в поле зрения).

ХОД РАБОТЫ

1. Вычисление позиции игрока по двум флагам

1.1. Расчет координат

Для используемого алгоритма требуется знать 2 флага (объекта с известными координатами) и расстояние до них. На рис. 1 показаны флаги *F1* и *F2*, расстояния до этих флагов *d1* и *d2*, расстояние между флагами *d* – это известные величины.

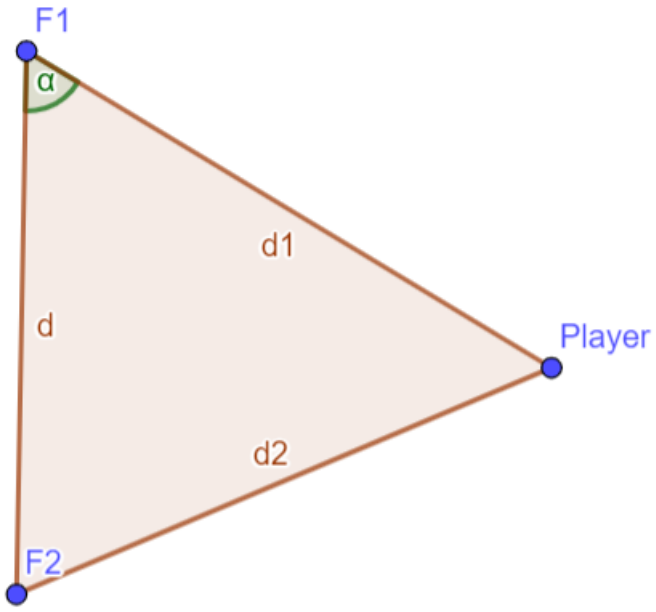

Рис. 1 – Определение координат игрока.

На данном этапе очевидна проблема выбора стороны – игрок может быть как справа, так и слева от выбранной пары флагов (для удобства предположим, что флаг *F1* находится строго выше *F2* на одной вертикальной оси – это не принципиально для вычислений и нужно лишь для визуализации алгоритма). Для решения этой проблемы воспользуемся параметром `direction`, т.е. углом, под которым игрок видит флаги. Если $F2.direction < F1.direction$, то флаг F2 находится левее флага F1 и игрок находится справа от пары флагов; в противном случае игрок находится слева.

Далее выполняется поиск угла α соответствующего углу *F1* треугольника при помощи теоремы косинусов:

$$\cos\alpha = \frac{d_2^2 - {d_1}^2 - d^2}{2d_1 d}$$

С использованием известных координат флагов *F1* и *F2* вычисляется нормированный вектор $\bar{f} = \frac{F2-F1}{|F2-F1|}$.

Затем требуется повернуть вектор $\bar{f}$ на угол α против часовой стрелки, если игрок находится справа, или по часовой стрелке, если игрок слева – для этого можно использовать матрицу поворота в двухмерном пространстве.

Полученный вектор будет нормированным вектором, сонаправленным с вектором $(F1, Player)$.

В конце остается лишь умножить полученный вектор на расстояние *d1* и прибавить его к координатам флага *F1* – полученная точка и будет являться позицией игрока.

1.2. Выбор лучшего значения

В большинстве случаев игрок видит больше двух флагов. В этом случае выполняются следующие действия:

1. Флаги сортируются по расстоянию до них.
2. Из полученного списка попарно выбираются флаги и для них считается позиция игрока (для 1 и 2, 2 и 3, 3 и 4 и т.д. флагов).
3. Из рассчитанных позиций выбирается позиция с наименьшей ошибкой.

Ошибка считается как сумма разниц между полученным расстоянием (расстояние между координатами флага и игроком) и видимым расстоянием, которое приходит с сервера. Такой метод расчета ошибки показал несколько лучшие результаты в сравнении с функцией ошибки, которая возвращает наибольшую разницу между полученным и видимым значениями расстояний.

2. Определение координат объектов

Расчет координат игроков и мяча одинаков, поэтому будет использован один и тот же алгоритм.

2.1. Поиск направления взгляда

Определим реальное направление взгляда игрока – нормированный вектор.

Для этого выберем флаг и найдем вектор $\overline{PF}$, где P – игрок, F – выбранный флаг, после чего нормируем полученный вектор. Направлением

взгляда в таком случае будет вектор, повернутый на угол *F.direction* (если поворот выполняется против часовой стрелки).

2.2. Выбор лучших координат.

Обычно мы видим сразу несколько флагов, находящихся на разной дистанции и с разным направлением. Для выбора лучшего из них воспользуемся функцией ошибки, схожей с примененной в п. 1.2.

1. Найдем рассчитанное направление всех флагов. Этим направлением будет угол между вектором взгляда и вектором $\overline{PF'}$, где F' - очередной флаг.

2. Сравним полученные направления с видимыми данными (что пришли в сообщении `see` сервера) и найдем сумму разностей. Эта сумма и будет ошибкой.

Очевидно, лучшим флагом будет являться тот, у которого получилась меньшая ошибка.

2.3. Определение координат объекта

Для определения координат объекта осталось лишь повернуть полученный нулевой вектор (вектор взгляда) на известное значение направления объекта, умножить на расстояние до объекта и прибавить к текущей позиции игрока. Полученные координаты будут искомой позицией объекта.

3. Передача параметров в программу

Для обработки аргументов программы используется модуль `yargs`. С его помощью определены следующие опции:

`--team` – строка, название команды.

`--coords` – два числа, разделенные символом «:». Координаты игрока, куда он будет размещен на старте командой `move`.

`--turn` – угол, на который игрок должен выполнять поворот на каждом такте. Если указан 0

`--log` – Boolean. нужно ли выводить в консоль информацию о видимых игроку объектах (мяч и другие игроки) и позицию самого игрока.

ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы были изучены способы определения координат объектов в условиях неопределенности восприятия окружающего мира.

Написана программа, осуществляющая взаимодействие с сервером RoboCop Soccer Simulation, выполняющая подключение, начальное размещение игрока и управление поворотом игрока после начала игры. Программа также обрабатывает ответы сервера, анализирует получаемые сообщения и выполняет расчет координат объектов, основываясь на полученных данных и учитывая возможную неточность этих данных.

Для более удобного управления программой добавлена возможность настраивать отдельные опции при запуске с использованием аргументов командной строки.