**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

Курсовая РАБОТА

**по дисциплине «Системы реального времени на основе LINUX»**

Тема: Использование пакетов навигации в ROS

Студенты группы 1303

Бабикова Е.

Дегтярь Р.

Преподаватель

Филатов А.

Санкт-Петербург

2016

**Исходная постановка задачи**

Избегая Всевидящего Ока Саурона, необходимо пронести кольцо всевластия и выбросить его в Ородруин.

**Исходные данные**

На неизвестной карте с кучей препятствий, которую периодически осматривает Всевидящее Око, необходимо пересечь Мордор и дойти до Ородруина.

Ограничения на исходные данные:

* Око имеет радиус обзора. В нём нельзя оставаться дольше какого-то фиксированного времени.
* У вас есть запас сил. Он не бесконечен.
* Карта Мордора плоская, но имеет непреодолимые препятствия, которые необходимо обойти.
* Из-за препятствий всевидящее Око имеет слепые зоны.

**Усложнения**

На карте присутствует Голлум, мечтающий отнять у вас свою прелесть. Вы успешно от него отбиваетесь (телепортируя его в произвольную точку карты) но это отнимает у вас выносливость.

Око Саурона осматривает карту произвольно (в произвольный момент времени может смениться ускорение осмотра, скорость поворота сектора обзора соответственно пересчитывается).

**Краткие сведения о Navigation Stack**

В данной курсовой работе необходимо ознакомиться с пакетом ROS Navigation Stack. Этот пакет позволяет проложить маршрут робота от начальной точки до целевой на карте с различными непроходимыми препятствиями. Стоит заметить, что разработчики этого пакета не дают никаких гарантий, что построенный маршрут будет оптимальным.

Для того, чтобы использовать Navigation Stack в своем проекте необходимо обеспечить:

* отправку сообщений LaserScan с информацией о препятствиях;
* отправку сообщений с одометрией персонажа;
* отправку сообщений со структурой дерева tf.

Следующий рисунок отображает общий механизм взаимодействия программы с Navigation Stack:

Odometry

LaserScan

tf tree

Twist

Goal

Предоставив Navigation Stack актуальные данные и целевую точку на карте, программа получает в ответ последовательность сообщений типа Twist, следуя

которым можно добраться до указанной цели.

**Описание структуры программы**

Исходя из поставленной задачи программа должна содержать как минимум две сущности:

* персонажа, несущего кольцо;
* око Саурона, осматривающее карту Мордора.

Помимо интерактивных сущностей необходимо обеспечить визуализацию. Для визуализации был выбран Irrlicht – кроссплатформенный графический 3D движок, написанный на С++. У него есть все необходимые возможности для реализации нашей задачи, а также к нему прилагается полная документация и большое количество различных примеров. Irrlicht позволит создать виртуальный мир и отслеживать расстояние от персонажа до окружающих его препятствий, что в свою очередь позволит формировать сообщения LaserScan для Navigation Stack.

Око Саурона может быть выделено в отдельный компонент, в рамках которого генерируются псевдослучайные числа, на основании которых строится траектория движения ока.

Таким образом, если Navigation Stack запущен и правильно сконфигурирован, задача сводится к реализации трех ROS нод:

* controller – обеспечивает взаимодействие с Navigation Stack;
* scene – обеспечивает визуализацию событий, происходящих в Мордоре;
* sauron – обеспечивает формирование траектории движения ока.

**Конфигурирование Navigation Stack**

Наиболее сложной частью работы является настройка пакета Navigation Stack. Для правильного планирования путей на неизвестной карте необходимо запустить:

* move\_base – основной компонент Navigation Stack;
* gmappign – пакет, используемый move\_base для построения карты окружающего мира на основе сообщений LaserScan и Odometry;
* пользовательскую программу, предоставляющую данные о роботе в формате сообщений LaserScan, Odometry, Pose и StampedTransform.

При запуске ноды move\_base необходимо загрузить ROS-параметры, описывающие различные характеристики процесса планирования пути. Наиболее важными и непосредственно влияющими на работу Navigation Stack являются следующие параметры:

* размер и разрешение costmap;
* описание габаритов робота;
* частота обновления costmap и частота обмена сообщениями;
* описание сенсора.

Важно добиться соответствия между глобальной и локальной costmap. Если этого не сделать, планирование пути будет ошибочным, что приведет к столкновению с препятствиями.

**Заключение**

Пакет Navigation Stack является неоптимальным, но достаточно надежным и относительно простым в использовании средством планирования маршрутов роботов.