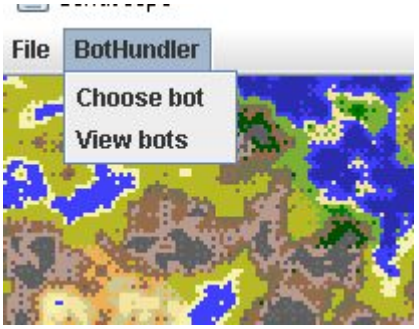
Руководство к пользованию

Чтобы запустить программу нужна установленная java (лучше 1.8). Далее выбираете релиз тут и запускаете с помощью команды:

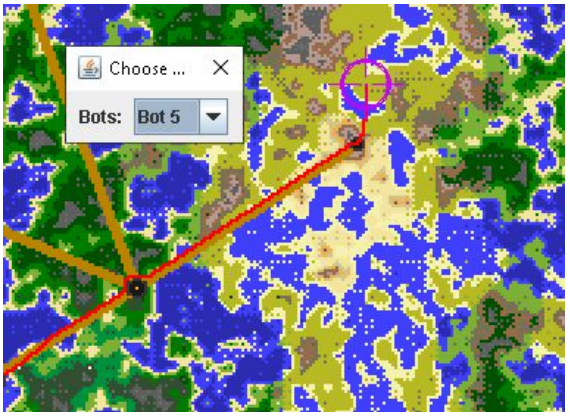
> java -Xms4096m -Xmx8192m -jar release.jar

Алгоритмы поиска (особенно Дейкстра и алгоритм Ли для взвешенных графов) “жрут” очень много процессорного времени, поэтому программа может работать медленно.

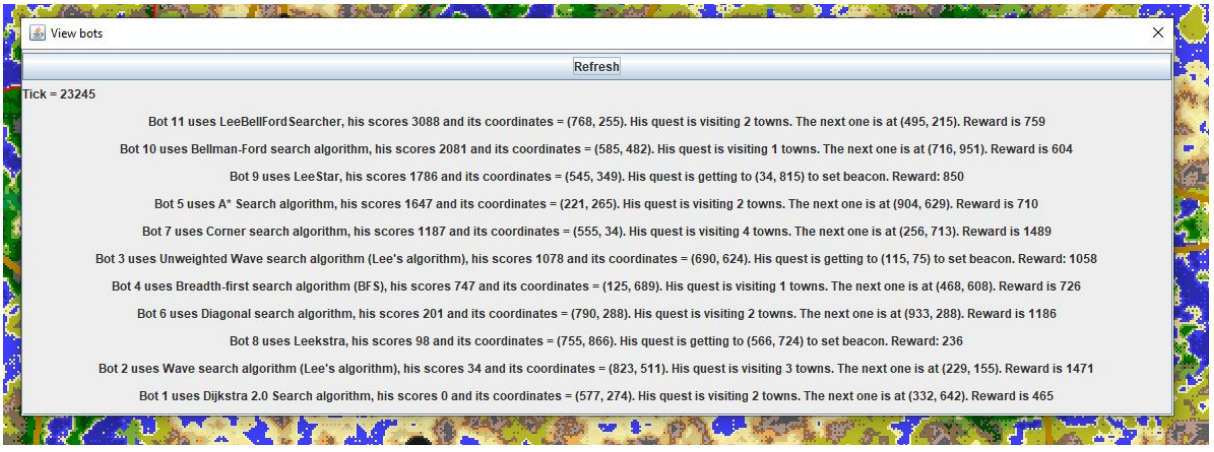
Вверху в меню баре вы можете выбрать “Choose bot”, чтобы наблюдать за каким-то определённым ботом.



На карте он будет выделен “прицелом” и будет показан его путь.



Вы можете посмотреть информацию о ботах, выбрав “View bots”. Откроется отсортированный по очкам список с информацией о ботах.



Добавление своих алгоритмов поиска

Если Вы хотите проверить, как будет работать другой алгоритм (например, если Вы напишете свой), то добавить его в нашу программу будет относительно просто, стоит лишь придерживаться паре моментов:

- Самое важное - расположение файла с алгоритмом. Своего “поисковика путей” необходимо расположить в папке ZPG/GameLogic/Searchers/
- Нужно реализовать интерфейс IDeWaySearcher или другими словами нужно обязательно сделать метод search()
- Метод search() должен возвращать очередь, где будут точки для посещения. Стартовая точка не должна входить в эту очередь, а финальная должна.
- Наличие проверки, что точка старта и точка финиша - точки, которые есть на карте, и в пути не должно быть начальной точки. Они не должны быть за ее пределами. Ну и, конечно, не забудьте, что алгоритму поиска обязательно нужно передать саму карту (сделать это можно в конструкторе)

```
if(map.checkCoordsLegal(start) == false ||
map.checkCoordsLegal(end) == false)
            throw new IllegalArgumentException("start = " + start
+ ", end = " + end);
```

- Переопределите метод toString(), указав строку с названием Вашего алгоритма, которая будет отображаться в статистике (чтобы его можно было найти в списке всех), в данном месте:

```
@Override
public String toString()
{
    return "Diagonal search algorithm";
}
```

- Также не забудьте добавить Ваш алгоритм в GameHundler в chooseSearchAlg. Там Вы можете, раскомментировать строку int what = … указать, что только Ваш алгоритм будет выбираться. Ах да, при добавлении алгоритма, не забудьте увеличить значение в nextInt

```java
private IDeWaySearcher chooseSearchAlg()
{
    Random r = new Random();
    int what = r.nextInt(11);
    //int what = 6;
    IDeWaySearcher res = null;
    switch(what)
    {
        case 0:
            res = new BreadthFirstSearcher(this.map);
            break;
        case 1:
            /*if(DepthFirstSearcher.getNums() == -1)
                res = new DepthFirstSearcher(this.map);
            else
                res = new BreadthFirstSearcher(this.map);*/
            res = new AStarSearcher(this.map);
            break;
        case 2:
            res = new LeeSearcher(this.map);
            break;
        case 3:
            res = new newDijkstra(this.map);
            break;
        case 4:
            res = new DiagonalSearcher(this.map);
            break;
        case 5:
            res = new CornerSearcher(this.map);
            break;
        case 6:
            res = new LeekstraSearcher(this.map);
            break;
        case 7:
            res = new LeeStarSearcher(this.map);
            break;
        case 8:
            res = new UnweightedLeeSearcher(this.map);
            break;
        case 9:
            res = new BellmanFordSearcher(this.map);
            break;
        case 10:
            res = new LeeBellFordSearcher(this.map);
            break;
        default:
            System.out.println("Failed successfully in chooseSearchAlg");
```

- Гарантированно создать бота со своим алгоритмом можно используя следующий шаблон, добавив его в public GameHundler(Runnable print):

```
bots.add(new Bot(sPoint.rndPoint(0, map.getMaxSize()-1), new ИмяВашегоАлгоритма(this.map)));
```

При желании можно взять уже созданные алгоритмы и модифицировать их на свой вкус или же взять их за основу для своего.

Если непонятно (что скорее всего так и есть), то можно посмотреть видео: https://www.youtube.com/watch?v=nbzez83hppM

Если вам нужно будет использовать множество при написании вашего алгоритма, то рекомендуем пользоваться классом SetBit. Вот некоторые методы:

- add - добавить элемент в множество
- remove - удалить элемент из множества
- contains - проверить содержится ли элемент в множестве
- Чтобы создать объект, нужно знать максимальное кол-во элементов, поэтому предлагаем пользоваться такой конструкцией: new SetBit(map.getMaxLineNum()+1)

Для организации очереди рекомендуем использовать LinkedList, так как очень часто нужно будет добавлять точки в структуру.