МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ

Ордена Трудового Красного Знамени

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский технический университет связи и информатики»

Математическая кибернетика и информационные технологии

Отчёт по лабораторной работе №5

по дисциплине «ИТиП» на тему:

«Алгоритм A*»

Выполнили: студент группы

БВТ2104

Старостин Игорь Дмитриевич

Руководитель:

Мосева Марина Сергеевна

Москва 2022

Цель работы: Изучить работу алгоритма A*. Научиться работать с чужим кодом. Довести до работоспособности приложенную программу.

Задание:

Реализовать метод equals класса Location

Реализовать метод hashCode класса Location

Реализовать метод numOpenWaypoints класса AStarState

Реализовать метод getMinOpenWaypoint класса AStarState

Реализовать метод addOpenWaypoint класса AStarState

Реализовать метод IsLocationClosed класса AStarState

Реализовать метод closeWaypoint класса AStarState

Скомпилировать и запустить программу

Ход работы:

1) Реализуем метод equals. Т. к нам может придти не только объект класса Location, но и null, то мы принимаем Object и внутри проверяем его на совпадение с Location.(См. Рис. 1).

```
public boolean equals(Object obj){
    //Мы получаем ссылку на объект, нам необходимо проверить является ли он экземпляром класса Location
    if (obj instanceof Location){
        Location loc = (Location) obj;
        //Если является - сравниваем его координаты
        return (loc.xCoord == xCoord && loc.yCoord == yCoord);
    }
    //Если не является - возвращаем false
    return false;
}
```

Рисунок 1 - Метод equals класса Location

2) Реализуем метод hashCode. Нам необходимо преобразовывать координаты в уникальное значение.(См.Рис.2).

```
public int hashCode(){
    //Возвращаем хэш-код, "контрольную сумму" объекта по его координатам.
    return (xCoord * xCoord) + (yCoord * (xCoord - 7));
}
```

Рисунок 2 - Метод hashCode класса Location.

3) Реализуем метод numOpenWaypoints, предварительно инициализировав два поля класса AStarState “хэш-карты” открытых и закрытых полей. В данном методе мы просто возвращаем длину “хэш-карты” открытых полей - openWaypoints.(См. Рис. 4).

```
public int numOpenWaypoints()
{
    return openWaypoints.size();
}
```

Рисунок 4 - Метод numOpenWaypoints класса AStarState.

4) Реализуем метод getMinOpenWaypoints, в нём программа итерирует перебирая все значения openWaypoints и находя среди них минимум по общей стоимости waypoint-a.(См. Рис. 5).

```
public Waypoint getMinOpenWaypoint()
{
    Waypoint minWaypoint = null;
    Waypoint comparableWaypoint = null;
    float minCost = Float.MAX_VALUE;
    //Итерируем по всем открытым точкам и постепенно находим минимум.
    for (int i = 0; i < openWaypoints.size(); i++){
        comparableWaypoint = (Waypoint) openWaypoints.values().toArray()[i];
        if(comparableWaypoint.getTotalCost() < minCost){
            minWaypoint = comparableWaypoint;
            minCost = comparableWaypoint.getTotalCost();
        }
    }
    return minWaypoint;
}
```

Рисунок 5 - Метод getMinOpenWaypoint класса AStarState.

5) Реализуем метод addOpenWaypoint, в котором добавляем waypoint в хэш-карту если его там нет или если он там присутствует, но стоимость пути до новой вершины меньше текущей.(См. Рис. 6).

```
public boolean addOpenWaypoint(Waypoint newWP)
{
    //Пробуем найти точку в открытых точках.
    Waypoint toCompare = openWaypoints.get(newWP.getLocation());
    //Если точка не найдена или она присутсвует, но её пред. значение меньше то добавляем/заменяем ее в коллекции.
    if(toCompare == null || toCompare.getPreviousCost() > newWP.getPreviousCost()){
        openWaypoints.put(newWP.getLocation(), newWP);
        return true;
    }
    return false;
}
```

Рисунок 6 - Метод addOpenWaypoint класса AStarState.

6) Реализуем метод isLocationClosed, нам необходимо проверить закрыта ли локация, т. е. мы проверяем находится ли она в хэш-карте закрытых вершин.(См. Рис. 7).

```
public boolean isLocationClosed(Location loc)
{
    return closedWaypoints.get(loc) != null;
}
```

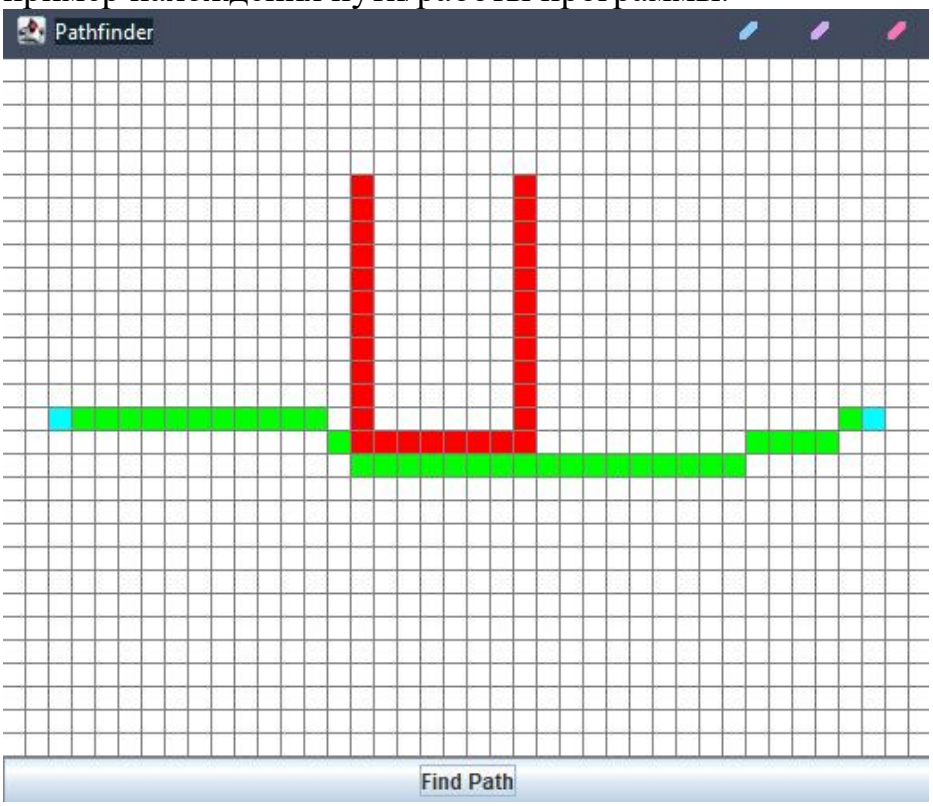
Рисунок 7 - Метод isLocationClosed класса AStarState.

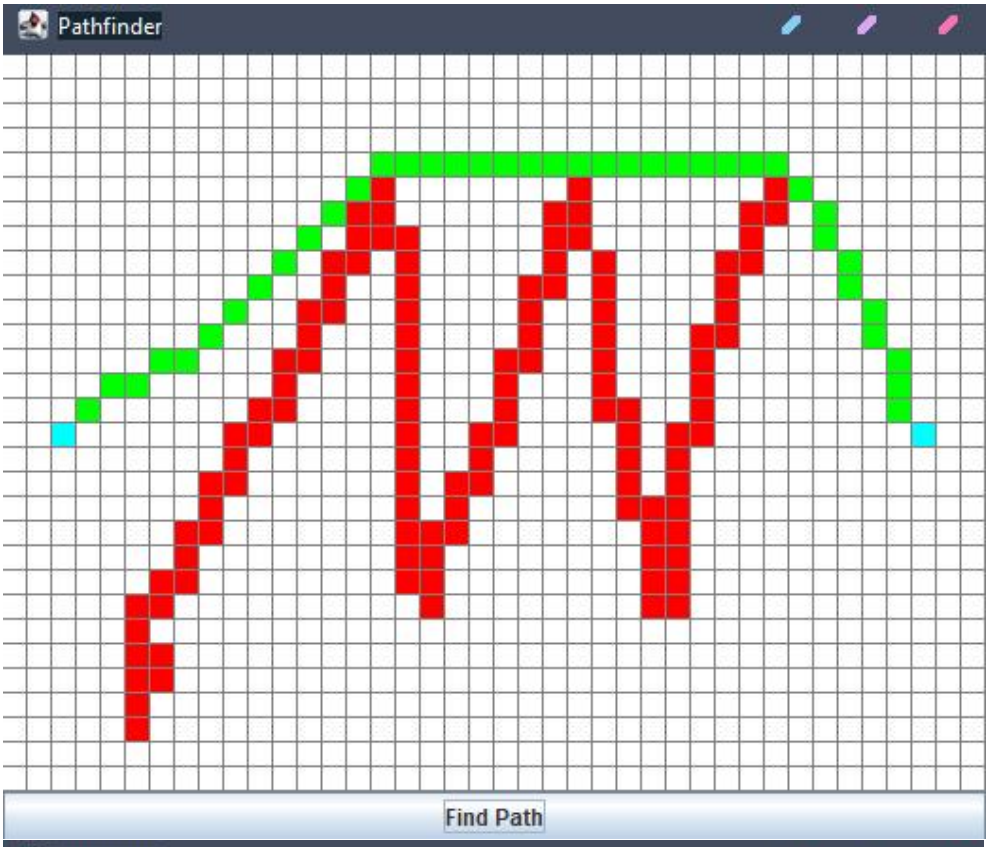
7) Реализуем метод closeWaypoint, нам необходимо удалить waypoint из хэш-карты открытых точек и переместить в хэш-карту закрытых точек.(См. Рис. 8).

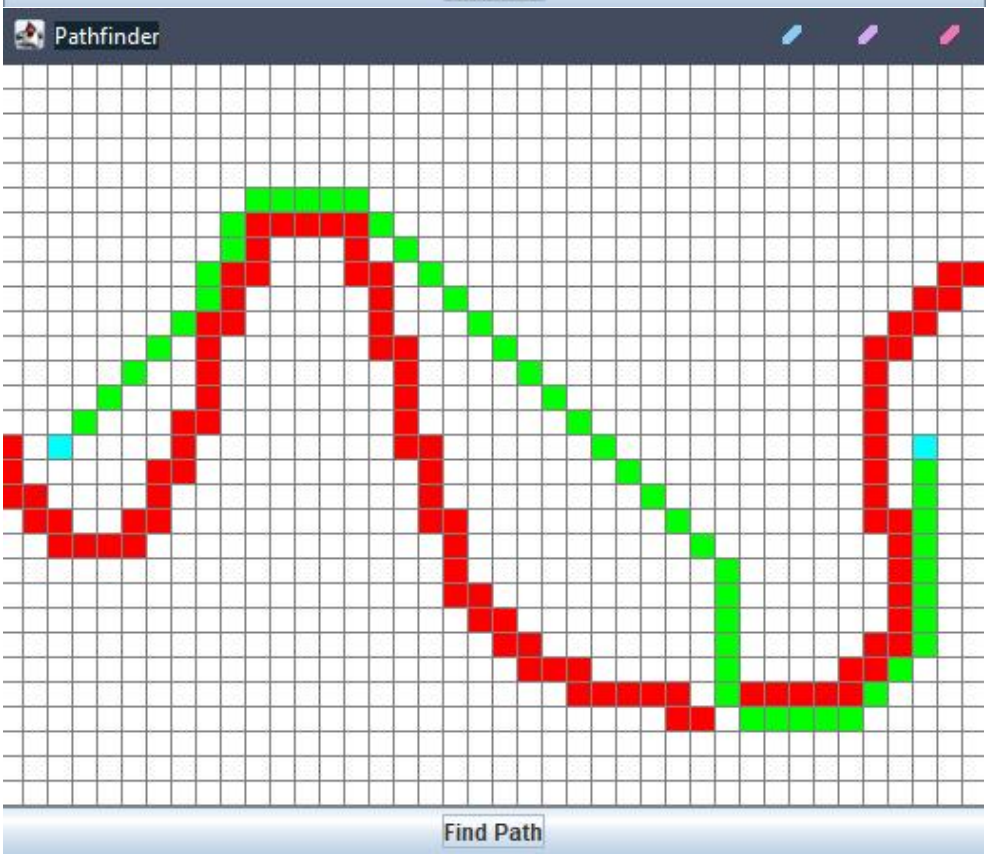
```
public void closeWaypoint(Location loc)
{
    //Получаем точку по хэшу локации из открытых точек
    Waypoint point = openWaypoints.get(loc);
    //Если point - null, то этой точки нет в открытых, следовательно дальше нечего делать
    if(point == null) return;
    //Удаляем точку из списка открытых
    openWaypoints.remove(loc);
    //Добавляем точку в список закрытых
    closedWaypoints.put(loc, point);
}
```

Рисунок 8 - Метод closeWaypoint класса AStarState.

8) Компилируем и запускаем программу, далее рис.9 - рис .11 пример нахождения пути/работы программы.







Рисунки 9-11 Результаты работы программы.

Вывод: Я изучил работу алгоритма A*, научился разбираться в чужом коде и доработал программу.