设计报告

**南京理工大学计算机科学与工程学院**

**C++课程设计 报告**

**班 级 22404班 \_**

**学生姓名 杨靖翔 \_\_**

**学 号 922106840441 \_\_**

**起止时间 9.1-9.24 \_\_**

**指导教师 娄震 \_ \_\_**

**南京理工大学计算机科学与工程学院制**

目录

1课题一：博弈系统 1

1.1课题功能 2

1.2开发平台 2

1.3AI算法设计 2

1.3.1总体流程图 2

1.3.2相关数据结构 6

1.3.3详细设计 10

1.4测试与运行 11

1.5调试记录 11

1.6版本记录 12

1.1课题功能

使用即时战略游戏搭建的博弈系统，测试学生的C++编程水平。该博弈系统是基于《帝国时代1》（微软1997年发行的即时战略游戏）改编的Qt项目。原游戏从模拟人类文明出现的石器时代开始进行，需要玩家搜集资源以生产各种单位及建筑物。当玩家搜集足够的资源，及建筑特定的建筑物之后，就可以升级至下一个文明时代。本系统1.0版本复刻了原游戏石器时代和工具时代的部分内容和操作，并开发了通过程序控制游戏过程的接口（以下简称为AI接口）。

通过AI接口编写一系列代码控制游戏过程，并完成指定的游戏目标。AI程序在博弈系统中运行的成绩，将作为C++课程设计个人项目评分的基础。

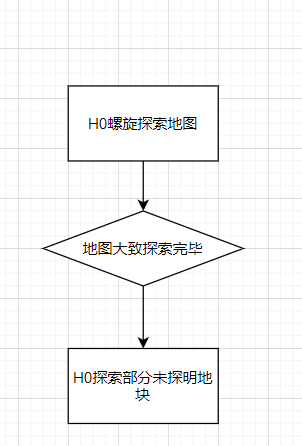
1.2 开发平台

Qt是一款跨平台的应用程序开发框架，被广泛用于开发图形用户界面（GUI）和嵌入式系统。Qt最初由挪威的Trolltech公司开发，现在由Digia公司维护和开发。Qt的目标是提供一种简单、直观、高效的开发环境，使开发人员能够快速构建出功能丰富、可靠稳定的应用程序。

其次，Qt提供了丰富的功能和组件，使开发人员能够轻松构建出各种复杂的应用程序。Qt包含了大量的预定义控件和类，例如按钮、文本框、列表框等，开发人员可以直接使用这些控件来构建用户界面。此外，Qt还提供了强大的图形渲染引擎和图形效果，使应用程序界面更加美观和吸引人。

另外，Qt还提供了一套完善的工具和开发环境，帮助开发人员进行代码编辑、调试和测试。Qt Creator是Qt的官方集成开发环境（IDE），它提供了直观的界面和丰富的功能，使开发人员能够更加高效地开发和调试应用程序。

1.3 AI算法设计

1.3.1 总体流程图

1.3.1.1 地图探索

H0（AIGame.human中下标为0的对象）以城镇中心为圆心，逆时针螺旋旋转探索地图

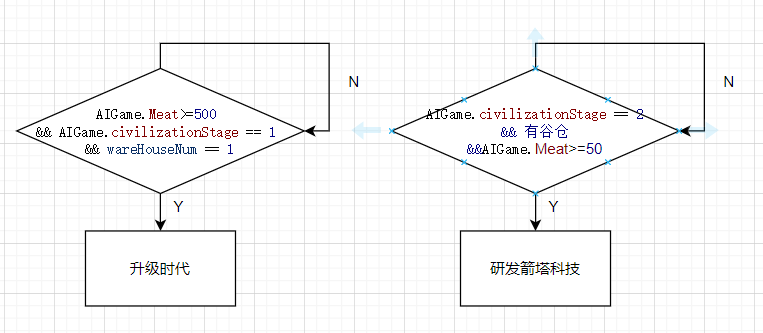
螺旋探索地图探明超过80%时，H0寻找部分未探明地块并探明。

图示

描述已自动生成1.3.1.2 人口

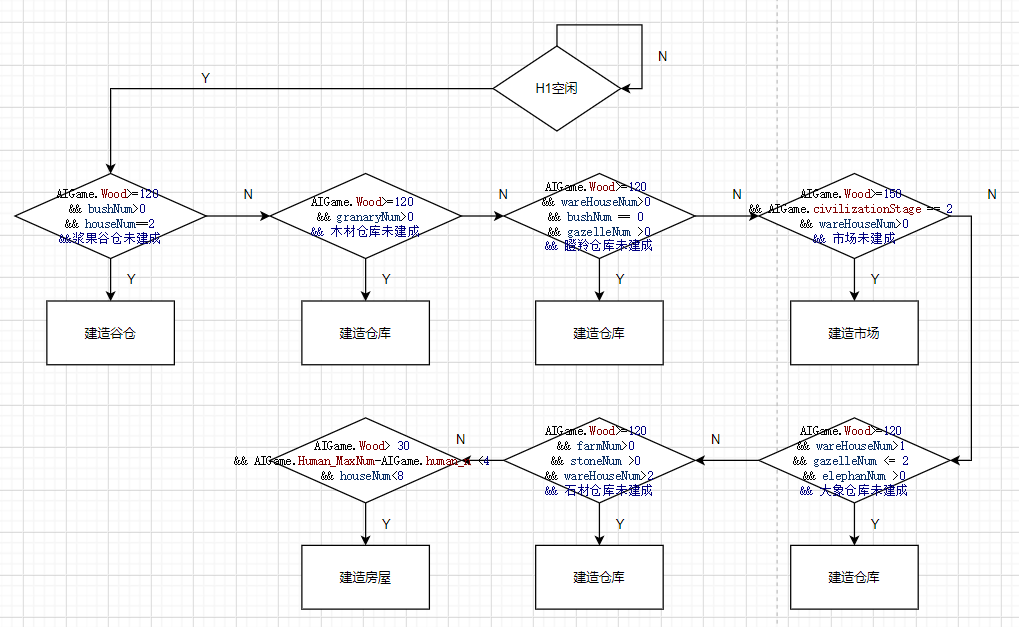
Civilization为1时，若人口小于12且meat大于50则生产人口

Civilization为2时，若人口小于12且meat大于50则生产人口

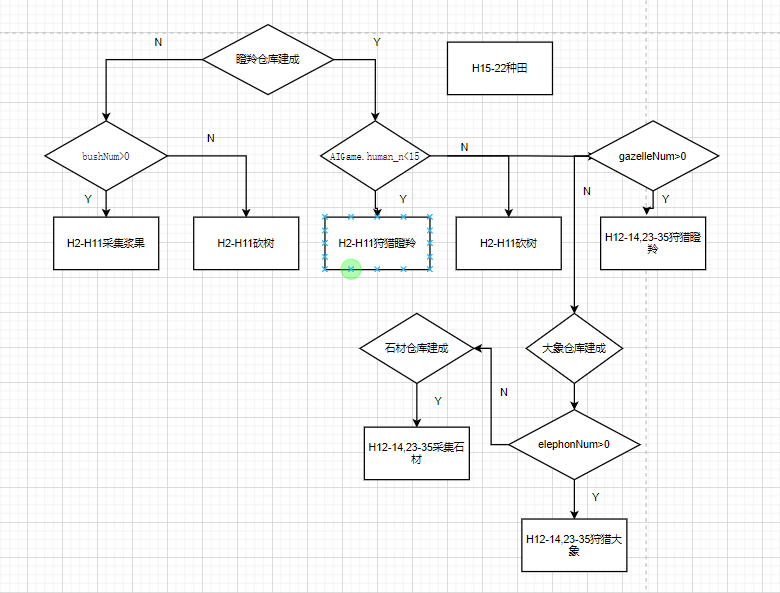
 1.3.1.3 科技

Civilization为1时，若meat>500且至少有一个仓库，则城镇中心升级时代

Civilization为2时，若meat>50则谷仓研发箭塔科技

1.3.1.4 建造

所有建造完成后，stone>150则建造箭塔

1.3.1.5 资源

木材>1050后，H2-11采集石材，农田耕种完后，H15-22采集木材

1.3.2 相关数据结构

1.3.2.1 mapBlock

1. 功能：用于标记地块

2. 属性：UL坐标，是否已探测，类型，是否可通过，是否危险，A\*算法相关FGH和指针

1. **struct** mapBlock
2. {
3. mapBlock() {}
4. mapBlock(**int** L,**int** U)
5. {
6. **this**->blockL=L;
7. **this**->blockU=U;
8. }
10. **int** blockL;
11. **int** blockU;
13. **bool** isDetect=**false**;
14. **int** type = -2;   //用资源/建筑Type标记，-2为空地，-3为不可建造，human为114
15. **bool** isPass=**true**;
16. **bool** mark = **false**;
17. **bool** isDanger=**false**;
18. **bool** isWork=**false**;
19. **int** SN;
21. **int** F=0;
22. **int** G=0;
23. **int** H=0;
24. mapBlock\* parent=NULL;
25. };

1.3.2.2 resData

1. 功能：获得资源地块的信息

2. 属性：SN，距离，坐标

1. **struct** resData
2. {
3. resData() {}
4. **int** SN;
5. **int** distance;
6. **bool** isCollect=**false**;
7. **int** blockL;
8. **int** blockU;
9. };

1.3.3 详细设计

1.3.3.1 地图信息更新

1. 功能：每帧调用，获得地图信息

1. //——————————————————————————更新地图信息————————————————————————————
2. **void** AI::updateMap()
3. {

6. //初始化
7. **for**(**int** i=0;i<72;i++)
8. {
9. **for**(**int** j=0;j<72;j++)
10. {
11. **if**(i == 0 || j == 0 || i == 71 || j == 71)
12. map[i][j].isPass = **false**;
13. **else**
14. map[i][j].isPass = **true**;
15. map[i][j].isDanger = **false**;
16. map[i][j].type = -2;    //标为空地
17. }
18. }
19. //视野标记
20. **for**(**int** i=0;i<AIGame.human\_n;i++)
21. {
22. **int** blockL=AIGame.human[i].BlockL;
23. **int** blockU=AIGame.human[i].BlockU;
24. //标记类型
25. **for**(**int** j=0;j<3;j++)
26. {
27. **for**(**int** k=0;k<3;k++)
28. {
29. **if**(map[blockL][blockU].type == -2)
30. map[blockL][blockU].type = -3;
31. }
32. }
33. map[blockL][blockU].type = 114;
34. //标记可视范围
35. **for**(**int** j=0;j<7;j++)
36. {
37. **for**(**int** k=0;k<7;k++)
38. {
39. **if**((j ==0 && k == 0) || (j==6 && k==0) || (j==0 && k==6) || (j==6 && k==6))
40. **continue**;
41. map[blockL+j-3][blockU+k-3].isDetect=**true**;
43. }
44. }
45. }
47. //建筑地块标记
48. **for**(**int** i=0;i<AIGame.building\_n;i++)
49. {
50. **int** blockL=AIGame.building[i].BlockL;
51. **int** blockU=AIGame.building[i].BlockU;
52. **int** blockType=AIGame.building[i].Type;
53. **int** blockSN=AIGame.building[i].SN;
54. **int** lengthOfSide=AIGame.building[i].Foundation+2;
55. //标记类型/可通过/SN
56. **for**(**int** j=0;j<lengthOfSide;j++)
57. {
58. **for**(**int** k=0;k<lengthOfSide;k++)
59. {
60. map[blockL+k][blockU+j].isDetect=**true**;
61. map[blockL+k][blockU+j].isPass=**false**;
62. map[blockL+k][blockU+j].type=blockType;
63. map[blockL+k][blockU+j].SN=blockSN;
64. }
65. }
66. }
67. //资源标记
68. **for**(**int** i=0;i<AIGame.resource\_n;i++)
69. {
70. **int** blockL=AIGame.resource[i].BlockL;
71. **int** blockU=AIGame.resource[i].BlockU;
72. **int** blockType=AIGame.resource[i].Type;
73. **int** blockSN=AIGame.resource[i].SN;
75. //标记可通过/危险标识
76. **switch** (blockType) {
77. **case** RESOURCE\_BUSH:             //树，浆果丛占地1
78. map[blockL][blockU].isPass=**false**;
79. **break**;
80. **case** RESOURCE\_TREE:
81. //3\*3不可建筑标记，1不可通过标记
82. **for**(**int** i=0;i<3;i++)
83. {
84. **for**(**int** j=0;j<3;j++)
85. {
86. **if**(map[blockL-1+i][blockU-1+j].type == -2)map[blockL-1+i][blockU-1+j].type=-3;
87. }
88. }
89. map[blockL][blockU].isPass=**false**;
90. **break**;
91. **case** RESOURCE\_STONE:            //石头占地1但生成方式有问题
92. //4\*4不可建筑标记，3\*3不可通过标记
93. **for**(**int** j=0;j<3;j++)
94. {
95. **for**(**int** k=0;k<3;k++)
96. {
97. map[blockL-1+j][blockU-1+k].isPass=**false**;
98. }
99. }
100. **for**(**int** i=0;i<4;i++)
101. {
102. **for**(**int** j=0;j<4;j++)
103. {
104. **if**(map[blockL-1+i][blockU-1+j].type == -2)map[blockL-1+i][blockU-1+j].type=-3;
105. }
106. }
107. **break**;
108. **case** RESOURCE\_LION:
109. **for**(**int** j=0;j<7;j++)
110. {
111. **for**(**int** k=0;k<7;k++)
112. {
113. **if**((j ==0 && k == 0) || (j==6 && k==0) || (j==0 && k==6) || (j==6 && k==6))
114. **continue**;
115. **if**(blockL+j-3>=0 && blockL+j-3<=71 && blockU+k-3>=0 && blockU+k-3<=71)
116. {
117. map[blockL+j-3][blockU+k-3].isDanger=**true**;
118. }
119. }
120. }
121. **break**;
122. **case** RESOURCE\_GAZELLE:
123. //4\*4不可通过标记
124. **for**(**int** j=0;j<5;j++)
125. {
126. **for**(**int** k=0;k<5;k++)
127. {
128. **if**(blockL-2+k<0 || blockL-2+k>72 || blockU-2+k<0 || blockU-2+k>72)**continue**;
129. **if**(map[blockL-2+j][blockU-2+k].type == -2)
130. map[blockL-2+j][blockU-2+k].isPass=**false**;
131. }
132. }
133. **break**;
134. **case** RESOURCE\_ELEPHANT:
135. //3\*3不可建筑标记
136. **for**(**int** j=0;j<7;j++)
137. {
138. **for**(**int** k=0;k<7;k++)
139. {
140. **if**(blockL-3+k<0 || blockL-3+k>71 || blockU-3+k<0 || blockU-3+k>71)**continue**;
141. **if**(map[blockL-3+j][blockU-2+k].type == -2)
142. map[blockL-3+j][blockU-2+k].type=-3;
143. }
144. }
145. **break**;
146. **default**:
147. map[blockL][blockU].isPass=**true**;
148. **break**;
149. }
150. //标记类型/SN
151. map[blockL][blockU].type = blockType;
152. map[blockL][blockU].SN = blockSN;
153. }

156. }

1.3.3.2 最近资源收集

1. 功能：计算距离收集站最近的资源位置，并指定人员工作

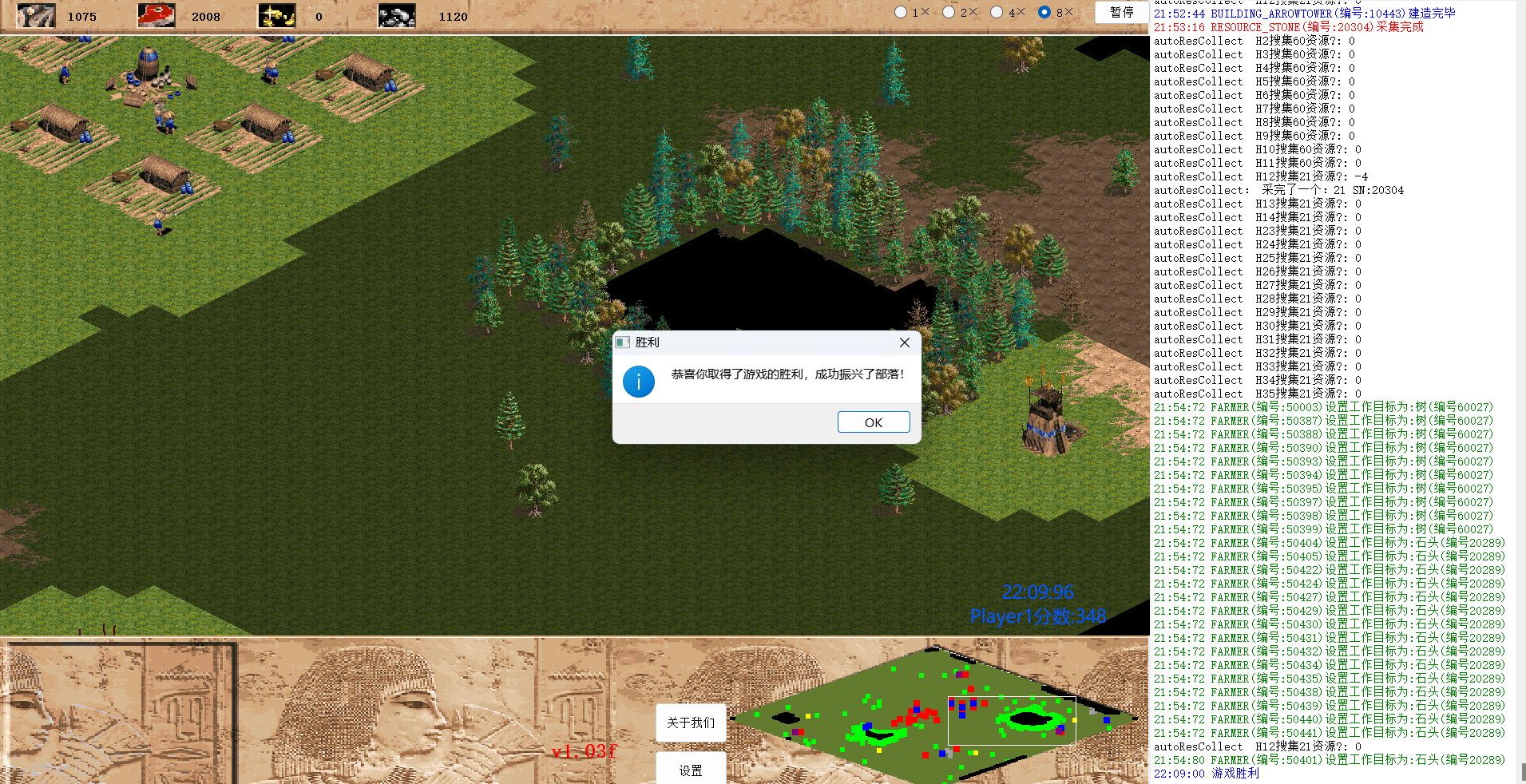
1. //指定收集范围，部分单位，自动收集
2. **bool** AI::autoResCollect(list<AI::resData> &resList, AI::resData &closeRes, **int** CenterBL, **int** CenterBU, **int** radius, **int** type, **int** Hstart,**int** Hend)
3. {
4. //在外界初始化列表
6. //人员指定
7. **int** hI = AIGame.human\_n;
8. **if**(hI>Hend+1) hI = Hend+1;
10. **for**(**int** i=Hstart;i<hI;i++)
11. {
12. **if**(AIGame.human[i].NowState == 0 || AIGame.human[i].NowState == 11)
13. {
14. **int** isOK= HumanAction(AIGame.human[i].SN,closeRes.SN);
15. emit AIDebugText("autoResCollect  H"+QString::number(i)+"搜集"+QString::number(type)+"资源?: "+QString::number(isOK));
16. **if**(isOK == -4)
17. {
18. emit AIDebugText("autoResCollect： 采完了一个："+QString::number(type)+" SN:"+QString::number(closeRes.SN));
19. //采完了更新地点
20. resList = getNearbyRes(CenterBL,CenterBU,radius,type);
21. **if**(resList.empty() == **true**)
22. {
23. emit AIDebugText("autoResCollect： 无可用资源");
24. **return** **false**;
25. }**else**
26. {
27. closeRes=getCloseRes(resList);
28. }
29. }
30. }
31. }
32. **return** **true**;
33. }

1.3.3.3 A\*寻路算法

1. 功能：自动避障，寻找最近路线

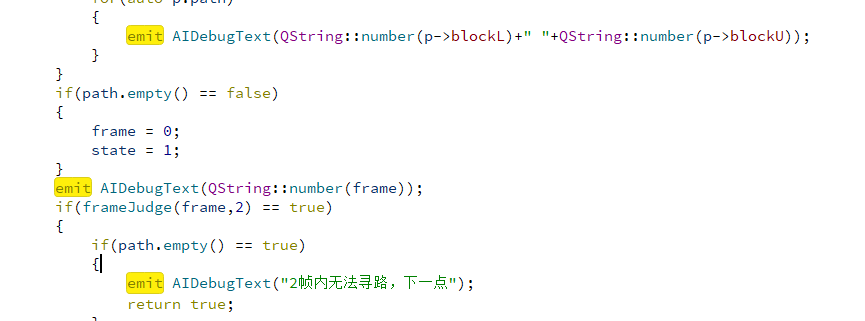
1. //计算G
2. **int** calG(mapBlock \*temp\_start, mapBlock \*point);
3. //计算H
4. **int** calH(mapBlock \*end, mapBlock \*point);
5. //计算F
6. **int** calF(mapBlock \*point);
8. //从list中获得F最小的点
9. mapBlock \*getLeastFPoint();
11. //判断是否在list中
12. mapBlock \*isInList(**const** list<mapBlock \*> &thelist, **const** mapBlock \* point) **const**;
14. //判断两点间可否互相抵达
15. **bool** isCanReach(**const** mapBlock \* point,**const** mapBlock \* target,**bool** isIgnoreCorner) **const**;
17. //获得周围点的vector
18. vector<mapBlock \*> getSurroundPoints(**const** mapBlock \*point,**bool** isIgnoreCorner);
20. //寻找前进路径
21. mapBlock \* findPath(mapBlock &startPoint,mapBlock &endPoint,**bool** isIgnoreCorner);
22. //====获得路径指针
23. list<AI::mapBlock \*> AI::getPath(AI::mapBlock &startPoint, AI::mapBlock &endPoint, **bool** isIgnoreCorner)
24. {
25. mapBlock \*result = findPath(startPoint,endPoint,isIgnoreCorner);
26. list<mapBlock \*> path;
28. **while**(result)
29. {
30. path.push\_front(result);
31. result  = result->parent;
32. }
34. //清空列表
35. openList.clear();
36. closeList.clear();
37. **return** path;
38. }

1.4 测试运行



1.5 调试记录

1. 打印获得下一个前进点进行调试



2. 获得资源中心位置进行调试

图形用户界面, 文本

描述已自动生成

3. 获得收集站中心位置进行算法调试

图形用户界面

低可信度描述已自动生成

4. 获得农民工作状态进行策略调整

文本

描述已自动生成

1.6 版本记录

V0.1：更改资源收集策略

V0.2：更新资源中心点计算方法

V0.3：增加26人口时击杀狮子的队伍

V0.4：修复了Release版本无法运行的BUG

V0.5：简化了部分函数逻辑

图形用户界面, 应用程序

描述已自动生成

图形用户界面, 文本

描述已自动生成