1. 实验环境
   1. 环境版本

Arena修订过后的最终版本

* 1. 难度等级

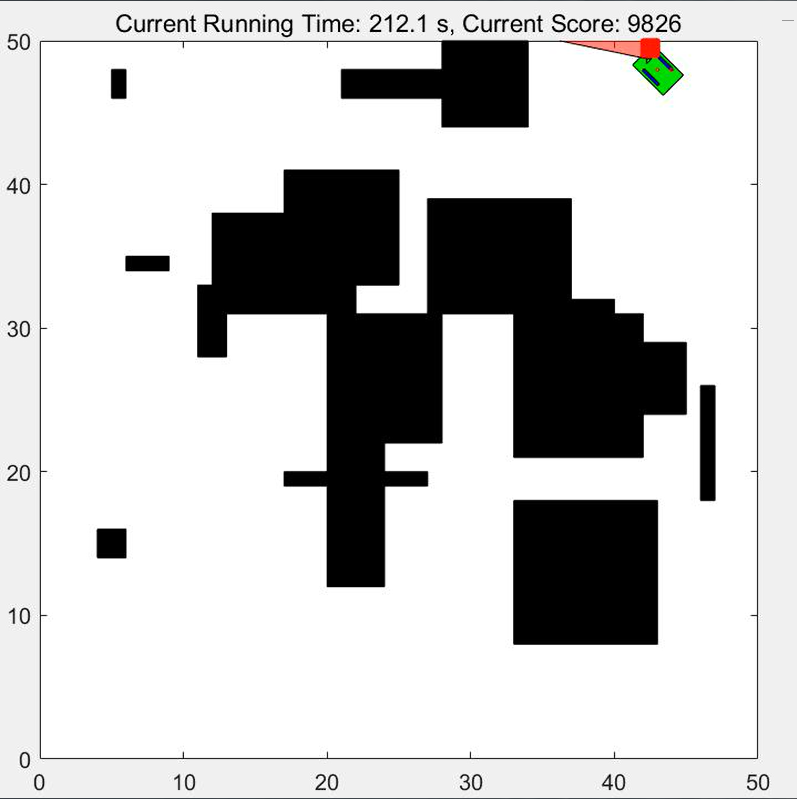
全局地图未开：globalview=0；

未加智能车：smartAgent=0。

值得一提的是，虽然没有对有障碍小车的情况进行相应的算法设计，但是在小车数量较少并且没有“倒霉”到小车完全将路堵死的情况下，依然可以跑到终点。

* 1. 已实现的地图

除ranmap之外的8张地图均可以跑通。但是对于c1地图，由于不知道全局地图信息，需要重复探索进行试错，有几次进入死路需要掉头，因此在150s内无法到达终点。在调整时间限制之后，于212.1s到达终点（如下图）。由于部分地图的计算时间会相对较长，我们设置了参数record=1，将每一张地图的运动过程录制下来一并发送，方便进行老师助教检查。



1. 设计思路
   1. 路径规划

我们借鉴了工作目录下面的AStar算法，在其基础上修改了其中的问题并进行了优化。我们设有open、close两个列表，使用8-邻域、启发算法来寻找路径。我们先学习了当下一系列路径规划算法，如混合Astar、DWA、Dstar等，受到Dstar的启发，我们尝试**从终点到当前位置寻找路径**，发现相比于从当前位置到终点的路径规划效果要稳定很多。

考虑到小车具有大小，实际寻找到的路径会出现贴着障碍物的情况，因此我们首先尝试了计算机图形学中的膨胀操作，将障碍物沿着边缘膨胀一格，在一部分地图上获得了不错的效果，但是也随之诞生了一系列问题，首先是当小车贴着障碍物行走时，小车自身的坐标点会被膨胀的障碍物所遮挡，导致路径搜索出现问题；另外是在部分地图如b2，会出现膨胀之后终点被障碍物遮挡导致报错的情况。对于后者，我们想到可以在膨胀之后将终点所在的坐标点设置为0，但是这种方法鲁棒性不够强。

在这种前提下，我们想到了对于路径规划中的代价算法增加**惩罚项**，也就是在每次计算代价时根据待考察路径点旁的障碍数量进行一定的惩罚，增加路径代价，也就在存在其他比较“安全”的路径时摒弃这条“崎岖小路”。在加入之后我们进行实验，发现加入后的运算量十分庞大，当前环境很长时间才能迭代一步。因此我们考虑仿照卷积神经网络使用卷积来**加快计算**。由于每次搜索路径时探索到的地图在当前搜索时不会改变，因此可以首先利用7\*7的卷积核对探测到的地图进行卷积，获取每一个点7\*7邻域内的障碍物数量矩阵传入寻路算法中，在启发式搜索计算代价时，只需调用卷积得到的惩罚矩阵，将当前点的障碍物数量乘上权重，作为惩罚项加在原来的cost上。应用了这种算法之后，我们发现几乎在每张地图都能获得很好的效果，巧妙地解决了小车与障碍物的碰撞问题。并且由于惩罚项的加入，搜索到的路径非常直观的具有壁障的效果，在两侧都存在障碍物时，会稳妥地选择从正中间通过。

然而，我们发现有些窄路即使规划的路线处于正中央，由于小车本身具有一定体积，因此也不容易通过容易发生碰撞。因此我们摒弃了之前的膨胀操作转而使用**闭运算**（先膨胀后腐蚀）。对于障碍物实现闭运算之后，宽度很窄的路径其两边的障碍物会连接在一起，而其他障碍物并不会受到闭运算的影响，这很好地实现了我们的预期要求。不仅如此，加入闭运算之后可以减小卷积核的大小为5\*5，达到与7\*7一样的效果，减小了运算量。

* 1. 控制算法

获取了小车的路径之后我们设计控制算法来控制小车按照预定的轨迹行进，借鉴预瞄法路径跟踪，我们利用路径上的下一个点与小车当前坐标点之间的夹角来控制小车的“方向盘”。我们首先获取小车的转向角alpha，并将其归一化到0到2\*pi之间。受PID算法的启发，当角度较大时转向的速度应该也越大，因此我们设计的转向速度控制函数正比于转向角(tan(alpha))。同时考虑到有些地图的路径通过对小车的身位控制非常苛刻，为了安全考虑我们采取了转弯时先改变方向，再直行的方法。

在编写控制算法时，我们一开始并没有考虑到发现前方有大量障碍物时路径规划会出现巨大突变的情况，获取了小车的转向角之后，并没有判断其是锐角还是钝角，导致当转向角为钝角时，小车会沿着大角度转弯，并且当小车朝向与轨迹刚好相反时，小车会认为不用转向了（此时alpha为pi，转向角速度为k\*tan(pi)=0），因此甚至会产生与路径背道而驰的情况。在弄清了实验环境以后，我们清楚了角速度与小车朝向变化的关系。在此基础上我们设置当转向角为钝角时的控制量与锐角时的控制量刚好相反。调整之后，小车完美沿着规划的路径行驶。

1. 程序亮点
2. 启发算法中加入了惩罚项，大大提升了小车的鲁棒性，代替了较多人考虑的膨胀思路，更能适配众多类型的地图。
3. 闭运算的加入使得小车能很好地规避较窄的路径，与惩罚项的结合会产生一加一大于二的效果。
4. 对于小车的控制十分稳定，很少出现颠簸晃动的现象，同时小车转向一定是沿着转向角较小的方向转动。
5. 本次代码的实现完全自主实现，结合了本节课程以及其他课程所学知识并综合应用，不存在抄袭现象。
6. 改进空间

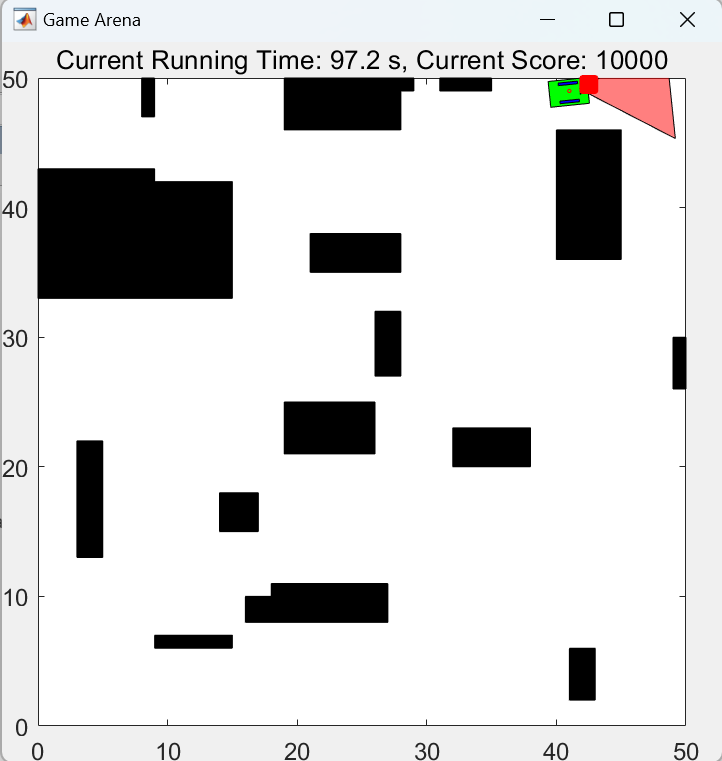
没有全局地图的情况下，小车在部分地图下计算时间会很长，导致运行速度较慢，原因应该是路径规划算法每次都要重新规划，而地图为50\*50，计算量较大，且有些可重复利用数据没有充分利用，后续可以进行优化。

1. 实验结果

由于部分地图的计算时间会相对较长，我们设置了参数record=1，将每一张地图的实验过程录制下来一并发送，方便进行老师助教检查。所有地图的movie都是由最终的版本一次性跑出来的，没有进行针对某地图的特别修改，可以验证算法的鲁棒性。

对于c1map，由于在150s时无法到达终点，我们增大了运行时间来将小车的运动探索过程录制下来，以便于展示我们的代码效果。

对于随机地图我们也进行了测试，在系统配置里设置random=1，在障碍物生成合理的情况下，我们的普适性也很强。下附一张随机地图下的运行结果。



对于有智能小车的情况，我们并没有专门设计算法来躲避小车，因此如果小车数量较多，则成功与否需要凭借一定的“运气”。随手跑的一次有8个智能小车的amap结果如下，全过程在视频“a-8agent.avi”中呈现。

