北京航空航天大學
BEIHANG UNIVERSITY

实验报告

内容（名称）：社会力模型仿真

院（系）名称	计算机学院
专业名称	计算机科学与技术
指导教师	宋晓
学号及姓名	16061200 陈治齐
	16061101 张哲维
	16061106 吴　枫
	16061107 蒋　锋

2018 年 12 月

一、 实验内容

近年来，国内外大城市突发事件频发，如何在紧急情况下进行有效的人群疏散已经引起各国的关注。我国正处于社会转型期，城市化进程加快使得城市内人口和建筑物密度急剧增加，公共建筑物内的人群聚集问题更加突出。因此，研究房间内的人群疏散问题以及如何合理安排房间布局、有效组织人群疏散具有重要的现实意义。

本报告基于 Helbing 提出的社会力模型，建立单房间疏散场景，实现图形化仿真，研究场景、人群大小、出口大小等因素对疏散时间的影响。

二、 数学模型

核心公式：

$$\mathrm{m_i}\frac{dv_i}{dt} = m_i\frac{v_i^0(t)e_i^0(t) - v_i(t)}{\tau} + \sum_{j(\neq i)} f_{ij} + \sum_{W} f_{iW}$$

式中，

$v_i^0(t)$是期望速度，程序中所有人取相同值$2.0\mathrm{m/s}$。$e_i^0(t)$是期望方向，根据实验中的场景较为规则特点，采用 A*算法的变种 JPS 算法寻找期望方向。$v_i(t)$是当前速度，τ是特征时间，取固定值$0.5\mathrm{m/s}$。f_{ij}是行人间的排斥力，使用下式计算：

$$\mathrm{f_{ij}} = \mathrm{A_i} * \frac{r_{ij} - d_{ij}}{B_i} * \overrightarrow{n_{ij}}$$

式中，

r_{ij}是两个圆半径之和，d_{ij}是两圆的距离，$\overrightarrow{n_{ij}}$是力的单位方向向量。参数$\mathrm{A_i}$和B_i取固定值，分别是$2000\mathrm{N}$和$0.08\mathrm{m}$。

f_{iW}是人与墙的排斥力，使用下式计算：

$$\mathrm{f_{iW}} = \mathrm{A_i} * \frac{r_i - d_{iW}}{B_i} * \overrightarrow{n_{iW}}$$

三、 编程实现与调试过程

1、BasicClasses.py

首先，构造了几个基本的类，来对对象进行仿真，Vector2D 来表示位置、速度、力等；Circle 来模拟行人；Box 实质上是矩阵，用来模拟障碍物、墙以及目的地；Scene 则是整块“画布”，用来保存静态的障碍物、墙和目的地以及动态的人的位置。

类	属性	函数
Vector2D 二维向量	x：横坐标 y：纵坐标	__init__(self, x, y)
		__add__(self, other)
		__sub__(self, other)
		__mul__(self, scalar)
		__rmul__(self, scalar)
		__truediv__(self, scalar)
		norm(self)
		__str__(self)
		get_x(self)
		get_y(self)
		set_x(self, x)
		set_y(self, y)
Circle 行人	pos：位置向量 vel：当前速度 next_pos：下一位置 next_vel：下一时刻速度 mass：质量 radius：圆半径或人半肩宽	__init__(self, x, y, vx, vy, mass, scene = None)
		get_radius(self)
		distance_to(self, other)
		is_intersect(self, other)
		ped_repulsive_force(self)
		wall_repulsive_force(self)
		desired_force(self)
		get_force(self)

		accleration(self)
		compute_next(self, scene)
		update_status(self)
Box 障碍物	p1：对角点 1 p2:对角点 2	__init__(self, x1, y1, x2, y2)
		__init__(self, x1, y1, x2, y2)
		scale(self, factor)
		is_intersect(self, other)
		is_in(self, pos)
		center(self)
		width(self)
		height(self)
Scene 场景	boxes：障碍物和墙 dests：目标位置 peds：行人	__init__(self, dests=None, peds=None, boxes=None)
		load(self, path)
		all_peds_arrived(self)
		update(self)
		save(self, path)

2、PathFinder.py

利用 A*算法进行寻路。

类	函数
Node 节点	__init__(self, x, y)
	__init__(self, scene)
	build_nodes(self)
	update_nodes(self, start)
	is_walkable_at(self, x, y)

AStarPathFinder A*算法	jump(self, cx, cy, dx, dy, start, goal)
	find_neighbors(self, node)
	neighbors(self, node)
	identify_successors(self, node, open_list, start, goal)
	heuristic_estimate(self, start, goal)
	dist_between(self, node1, node2)
	get_lowest(self, open_set)
	construct_path(self, goal)
	a_star(self, start, goal)
	get_node(self, pos)
其他	path_finder_init(scene)
	get_direction(scene, source)

通过等式：F = G + H

式中，

G = 从起点 A，沿着产生的路径，移动到网格上指定方格的移动耗费。

H = 从网格上当前方格移动到终点 B 的预估移动耗费。

在本次仿真中，H 值使用曼哈顿方法，它计算从当前格到目的格之间水平和垂直的方格的数量总和，忽略对角线方向，是对剩余距离的一个估算，而非实际值，这也是这一方法被称为启发式的原因。

通过以上等式，可以选择路径中下一步是哪个方向，从而达到寻路的目的。

关键代码：

```
    def a_star(self, start, goal):

"""http://theory.stanford.edu/~amitp/GameProgramming/ImplementationNo
tes.html"""
        open_list = [start]
        start.g = 0
        start.f = start.g + self.heuristic_estimate(start, goal)
        start.open = True
        while len(open_list) != 0:
            current = self.get_lowest(open_list)
            if current == goal:
                self.construct_path(goal)
                return
            open_list.remove(current)
            current.open = False
            current.closed = True
            self.identify_successors(current, open_list, start, goal)
        # 无路可走
```

图 1　a_star(self, start, goal)

```
    def get_lowest(self, open_set):
        lowest = float("inf")
        lowest_node = None
        for node in open_set:
            if node.f < lowest:
                lowest = node.f
                lowest_node = node
        return lowest_node
```

图 2　get_lowest(self, open_set)

3、Gui.py

GUI 是基于 python 的 tkinter 包实现的。首先，将场景中的障碍物（灰色）及目的地（黑色）加入画布中，开始仿真后，基于每个人当前位置和上一个位置绘制出运动轨迹。同时，gui 允许拖动场景中的障碍物及目标位置来构成新的场景，并可将新的场景保存下来，也可加载之前所保存的场景。

类	函数
SfmGui	__init__(self, scene, epoch)
	begin_simulate(self)
	begin_simulate_btn(self, event)
	color_list_init(self, num)
	get_color(self, index)
	get_click(self, event)
	click_release(self, event)
	reset_scene(self, event)
	change_scene(self, scene)
	save(self, event, path)
	load(self, event, path)
	init_canvas(self)
	bind_btn(self)
	add_box(self, box, fill="black")
	add_person(self, ped, fill="black")
	add_dest(self, dest, fill="black")
	move_box(self, box, x, y)
其它	_async_raise(tid, exctype)
	stop_thread(thread)

图 3　　GUI 效果图

四、 程序运行结果分析

1、程序运行结果

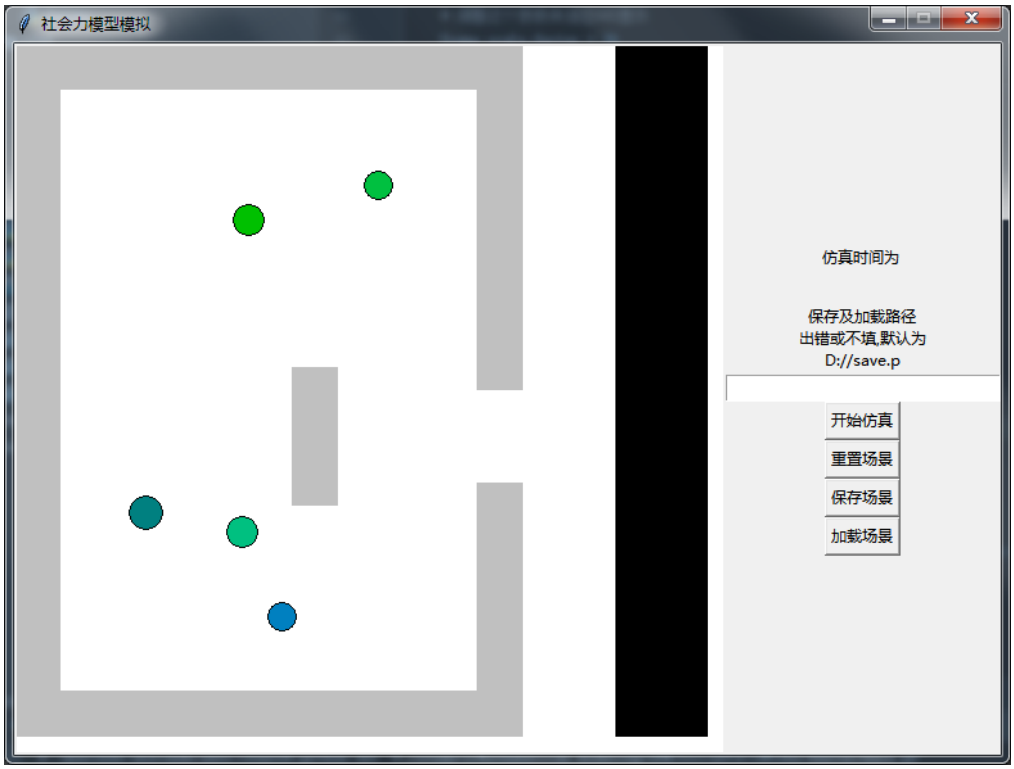

图 4　　5 人纵向障碍物初始状态

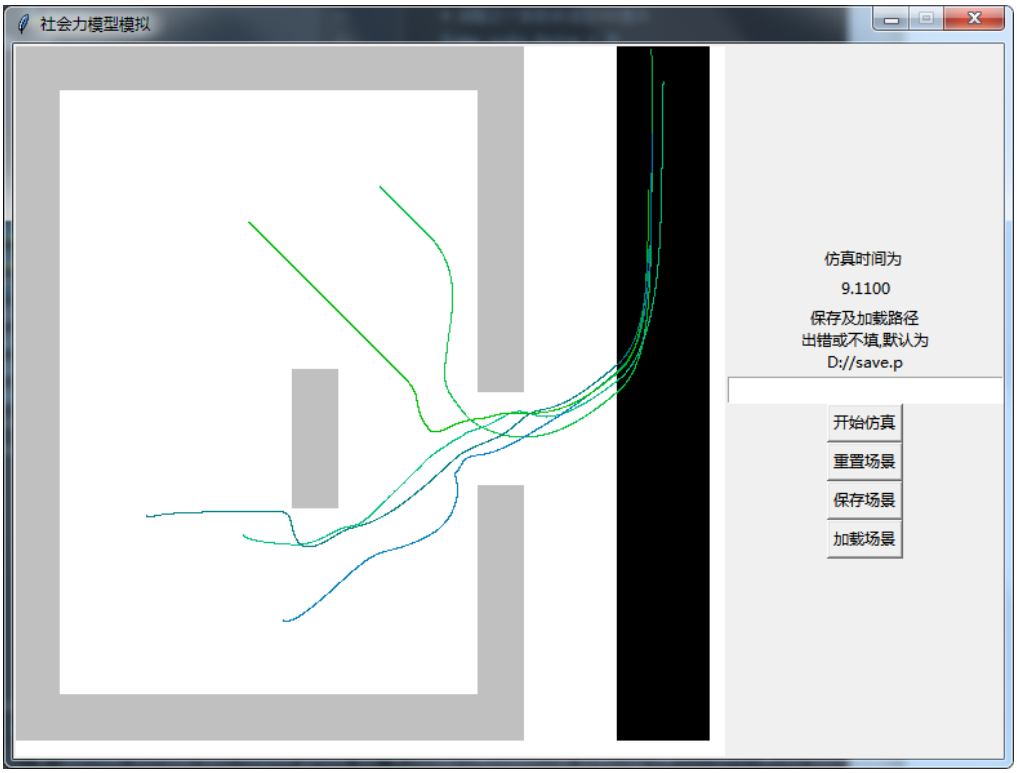

图 5　　5 人纵向障碍物终止状态

2、结果分析

1. 相同环境下，不同人数模拟情况

组号	障碍物方向	出口大小	人数	仿真时间 t	平均时间△t
1	横向	2	5	7.965	7.407
2				7.100	
3				7.155	
4			10	9.450	9.113
5				8.865	
6				9.025	
7			15	11.320	10.693
8				10.725	
9				10.035	
10	纵向		5	7.665	6.867
11				6.570	
12				6.365	
13			10	8.125	8.133
14				8.265	
15				8.010	
16			15	9.565	9.935
17				10.095	
18				10.145	

表 1　不同人数下的疏散时间

由测试数据可知，在障碍物方向与出口大小相同的情况下，人数增加可导致疏散时间的增加，且增加幅度较平均。

2. 相同人数下，横向纵向障碍物的模拟情况

组号	人数	出口大小	障碍物方向	仿真时间 t	平均时间△t
1	5	2	横向	7.515	6.943
2				6.155	
3				7.160	
4			纵向	6.545	6.883
5				6.080	
6				8.025	
7	10		横向	9.125	9.155
8				8.975	
9				9.365	
10			纵向	8.625	8.543
11				8.585	
12				8.420	
13	15		横向	10.420	10.477
14				10.775	
15				10.235	
16			纵向	9.735	10.135
17				10.155	
18				10.515	

表 2　　不同方向障碍物下的疏散时间

由测试数据可知，在人数与出口大小相同的情况下，障碍物的方向这一因素并不是十分影响疏散时间，通过几组数据可以初步认为纵向的障碍物的疏散时间略小于横向障碍物，但由于数据较小，随机性较大，因此并不具有普遍规律。

3. 相同人数下，不同出口大小的模拟情况

组号	障碍物方向	人数	出口大小	仿真时间 t	平均时间△t
1	横向	10	2	9.125	9.155
2				8.975	
3				9.365	
4			4	7.640	7.493
5				7.535	
6				7.305	
7			6	7.260	6.967
8				6.935	
9				6.705	
10	纵向		2	8.735	8.458
11				8.465	
12				8.175	
13			4	7.825	7.580
14				7.360	
15				7.555	
16			6	7.090	7.068
17				7.180	
18				6.935	

表 3　　不同出口大小下的疏散时间

由测试数据可知，在人数与障碍物方向相同的情况下，出口大小越大，疏散的时间会越小，但这一减小趋势会随着出口大小的增大而渐渐降低，即当出口大小已经较大的时候，再增大出口，对疏散时间的减少效果并不是十分明显。

五、 总结

通过社会力模型，我们组对单房间疏散这一场景有了初步的认知，也探讨了“房间人群的大小”、“障碍物方向”以及“出口大小”这三个因素对疏散速度的影响，更重要的是对数学建模来抽象社会问题，对其进行仿真有了初步的认知。我们也体会到了这是一种成本较低、而收益较高的研究方法，为我们日后的学习提供了新的思路。

同时，本次实验也存在着一些不足和遗憾，如当房间内人数较多的时候，仿真时间过长，未来如果对算法进行改进，能够仿真房间内一百乃至几百人的场景，得到的结论将会更具备普遍性；没有尝试添加多个障碍物，构造较为复杂的场景；可以就“出口位置”的设计进行深入研究，探讨在什么样的位置设立多大的出口可以达到更高的疏散效率，这样的结论会更有意义一些。