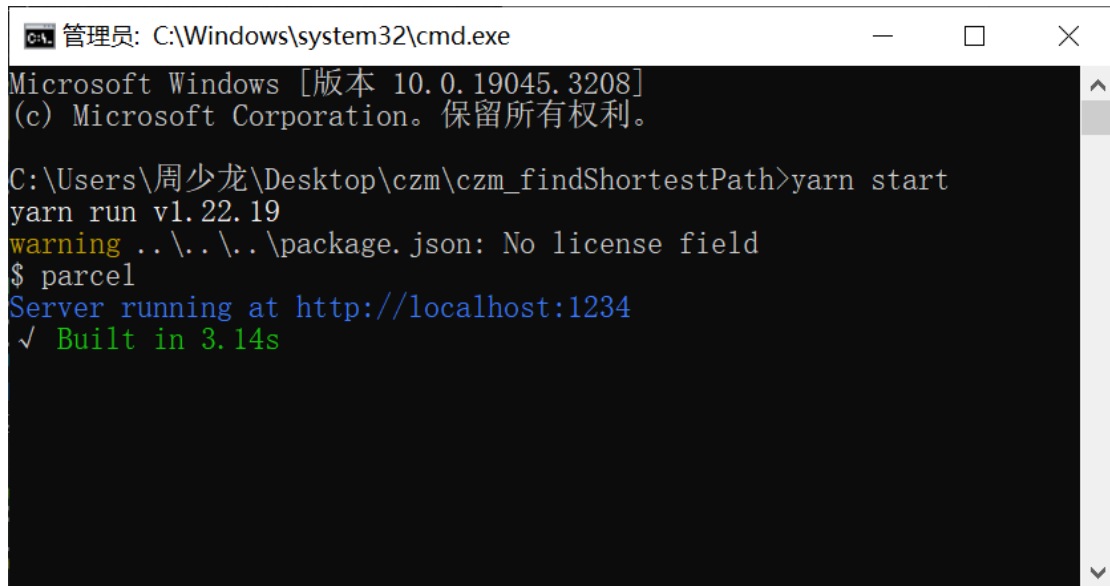


一、 启动方式：

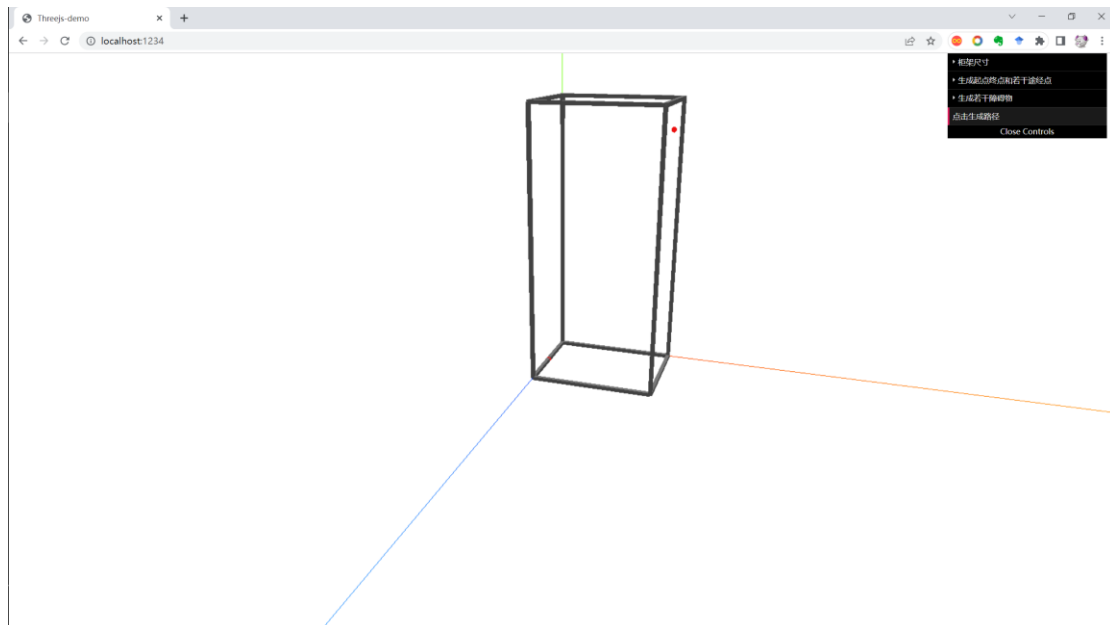
方式一：在\czm_findShortestPath 目录下打开 cmd，输入 “yarn start”，然后浏览器访问 localhost:1234，如下图：



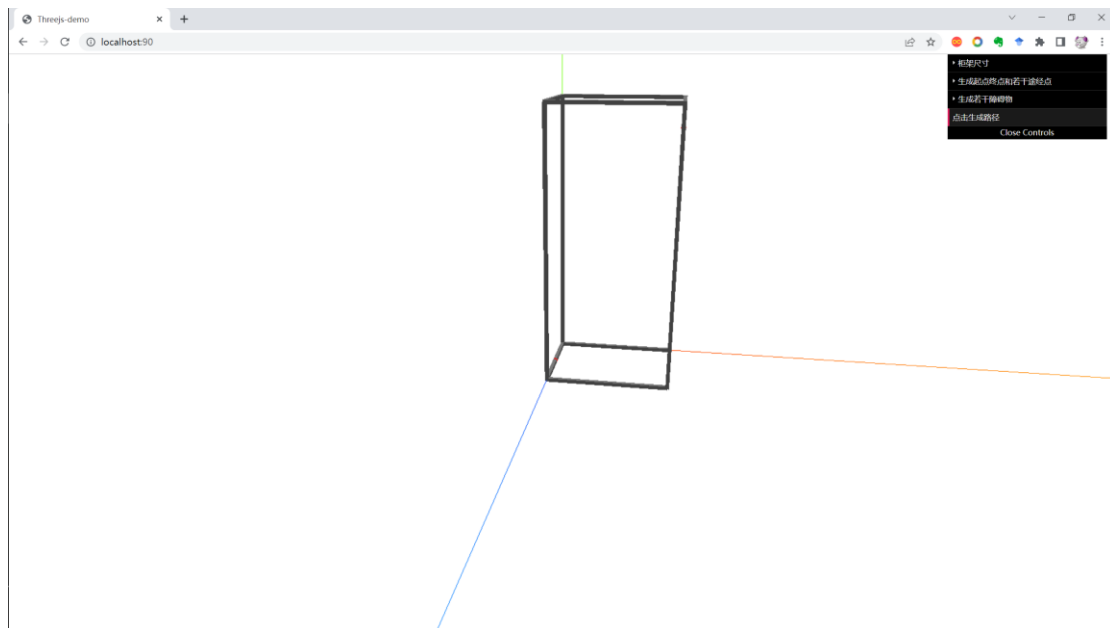
```
管理员: C:\Windows\system32\cmd.exe
Microsoft Windows [版本 10.0.19045.3208]
(c) Microsoft Corporation。保留所有权利。

C:\Users\周少龙\Desktop\czm\czm_findShortestPath>yarn start
yarn run v1.22.19
warning ..\..\..\package.json: No license field
$ parcel
Server running at http://localhost:1234
✓ Built in 3.14s
```

浏览器页面：



方式二：我们将项目打包部署在了 nginx 上，进入文件夹.\nginx-1.25.0-czm，然后点击 `nginx.exe` 启动项目，浏览器访问 `localhost:90`，如浏览器页面下图：



二、模型使用注意事项：

1. 初始框架长宽高设置如下：

| ▼ 框架尺寸 | |
|--------|----|
| 框架深 | 10 |
| 框架宽 | 8 |
| 框架高 | 22 |

请根据页面显示大小和实际框架大小选择度量单位，推荐使用“分米”作为度量单位。

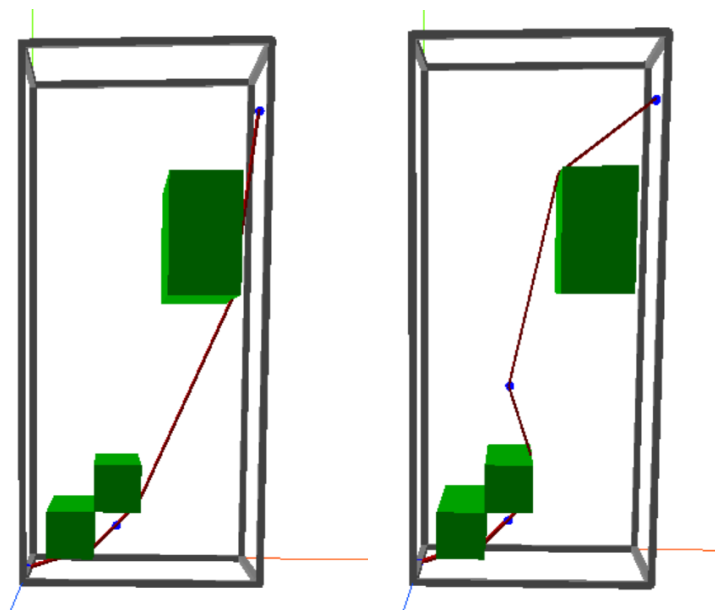
- 输入小数之后，GUI 可能会使输入的数取整，但是在页面中显示的物体仍然是实际输入的数值大小，这个可能是 GUI 的自带行为，对实际结果无影响。
- 我们假定障碍物为长、高、宽与 x 、 y 、 z 轴平行的长方体，用障碍物最靠近原点的顶点坐标作为障碍物的坐标，长为 x 方向上的长度，高为 y 方向上的长度，宽为 z 方向上的长度。
- 框架并没有被当做障碍物，如果需要将框架当做障碍物，请手动添加对应大小和位置的障碍物。
- 请在设置完所有障碍物后再设置安全距离，设置安全距离会使障碍物的长宽高向对应方向拓展一倍的安全距离，会在页面中显示为新的障碍物，此时新的障碍物的安全距离为 0，因为它们比实际设置的障碍物长宽高都要大一倍的安全距离。
- 生成的路径会根据生成途经点的顺序依次经过所有途经点。
- 起点、终点和途经点用蓝色的点表示，障碍物用绿色的长方体表示，生成的路径用红色长方体表示（注意，生成的是路径并不是排条，一是因为暂时没有考虑排条的厚度对于

路径点的影响，虽然这个影响不大；二是在折弯处本程序没有生成圆弧，也没有给出折弯处圆弧两端的两个点坐标，因为本程序主要功能是生成最短路径，只给出了折弯处的点坐标，如果需要圆弧两端端点需要后续工作完善）。

8. 我们没有在 GUI 中添加给起点、终点、途经点选择路径经过点时的方向这一选项，因为我们认为，如果要为这些点添加方向，可以让用户在添加途经点时将一个途经点添加为两个距离相近的途经点，比如用户本意是添加一个途经点(5, 5, 4)，经过这个点时路径的方向应该与 y 轴平行，那么用户可以添加两个途经点，分别为(5, 4.9, 4)和(5, 5.1, 4)，这样就能满足这个需求，当然这样会有两种情况：(5, 4.9, 4)-> (5, 5.1, 4)和(5, 5.1, 4)-> (5, 4.9, 4)，用户可以根据起点到原途经点的大方向来确定，拓展出来的两个途经点的先后顺序，借此来保证路径最短。
9. 我们的算法也不能确保路径的折弯夹角大于等于九十度，但是经过我们的观察，路径在绕过障碍物时一般不会有锐角折弯，只会两个途经点一来一回比较相近时路径才会产生锐角折弯。不过，我们可以通过将途经点按照事项 7 中拓展途经点的方法来避免产生锐角折弯。
10. 我们将障碍物简化为长方体，加上安全距离的限制之后的障碍物也是简化为更大的长方体，但以更精确的角度来看，加上安全距离之后的障碍物应该是边缘为圆弧的圆滑长方体，我们为了简单起见直接简化为了长方体，这样会使生成的最短路径长度稍稍大于实际的最短路径长度，误差不算大。
11. 打开浏览器控制台可以查看模型参数设置以及生成的最短路径上的所有坐标点。
12. 设置障碍物时请避免障碍物重叠，或者将起点、终点或者途经点设置在障碍物内部，本程序没有添加错误场景下的判断，请用户合理操作。
13. 建议：设置好所有点的坐标之后再去设置障碍物，且在设置好所有障碍物之后再去设置安全距离，否则出现意料之外的 bug，按照顺序操作则不会出现 bug。
14. 建议：每点击一次“点击生成路径”按钮之后建议刷新页面之后再进行下一次生成路径，否则可能会出现意料之外的 bug。

三、模型演示

如图所演示的，生成的红色路径依次经过蓝色途经点会绕过绿色障碍物，并且是最短路径。



四、项目介绍:

本项目实现了基于三维空间的结构化柜体内，点到点的最短路径的算法实现。抽象模型包含以下部分：起点、终点、若干途经点，若干障碍物，生成的路径要满足以下条件：

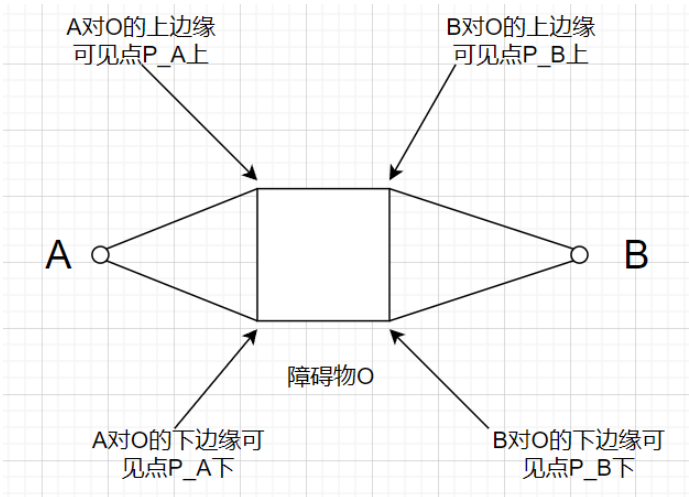
1. 路径从起点出发，依次通过途经点，最后到达终点，路径通过途经点的顺序是给定的；
2. 路径不能穿过障碍物，障碍物有安全距离，路径与障碍物之间距离大于等于安全距离；
3. 路径在折弯处夹角大于等于九十度（二维情况）；
4. 在经过起始点，途经点和终点时，必须由金属排横截面的中心位置点穿过，并且横截面与 XYZ 其中一个平面是保持平行；

核心算法思路：

该问题最核心的算法是**带障碍物情况下两点间最短距离的求解方法**，我们的算法参考了文献[1]中的方法，下面简要介绍我们算法的核心思想。

该问题中的障碍物全部简化为长方体，先考虑两个点在同一个 xy 平面内的情况，且仅有一个障碍物，如下图。图中给出了 A、B 两点对于障碍物 O 的上下边缘可见点。边缘可见点的定义为：在平面 P 上，若某个点 X 与 A 组成的线段，不穿过障碍物，则 X 是 A 的可见点。平面 P 上所有 A 可见点组成的集合定义为 A 的可见区域，记作 P_VA ；反之，定义为 A 的不可见区域，记为 P_NVA 。区域 P_VA 与区域 P_NVA 的边界线与障碍物 O 必然存在两个交点，这两个交点即为 A 对于 O 的边缘可见点（根据在 y 坐标的大小关系来定义上下边缘可见点）。

我们的算法主要根据 A 点 B 点与其对于一系列 AB 线段穿过的障碍物的上下边缘可见点的连线与其他障碍物是否相交的关系来迭代生成最短路径。详细伪代码请见文件“\算法伪代码以及说明.pdf”。



源代码见文件\czm_findShortestPath\src\app.js，本项目可视化部分调用 three.js 库完成，用 parcel 进行打包。

参考文献:

[1]陈智鹏,杨诗琴.带障碍物情况下两点间最短距离的求解方法[J].计算机工程,2010,36(16):171-173.