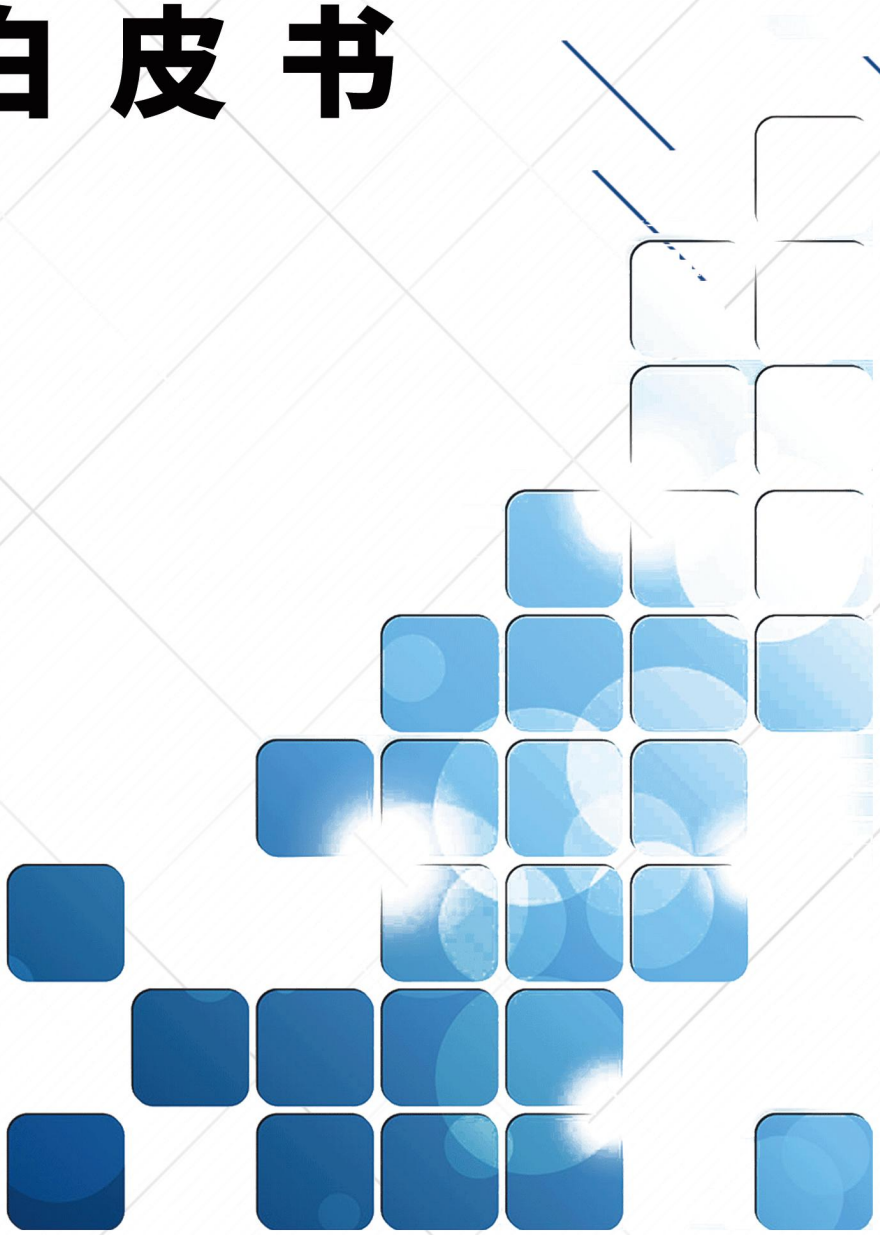
蜂鸟视图

蜂鸟云产品
白 皮 书



目　录

1. 概述

本文面向蜂鸟云产品的所有用户群体，撰写本文的目的在于方便用户在选购蜂鸟云产品及相关服务前，能够简要了解使用或购买和理解本产品前所需的相关知识背景；同时，了解本产品所之相关的技术领域的知识该要。蜂鸟云（英文：FengMap Cloud），指代蜂鸟视图科技有限公司的室内空间云服务平台产品。

1.1. 从楼层（建筑）平面图到室内地图

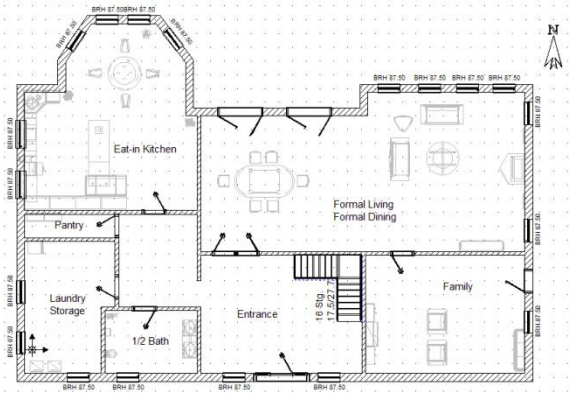
在建筑学和建筑工程中，平面图（Floor Plan）是一种按比例绘制的图画，从上面看，显示了房间、空间、交通模式和一个结构中的其他物理特征之间的关系。

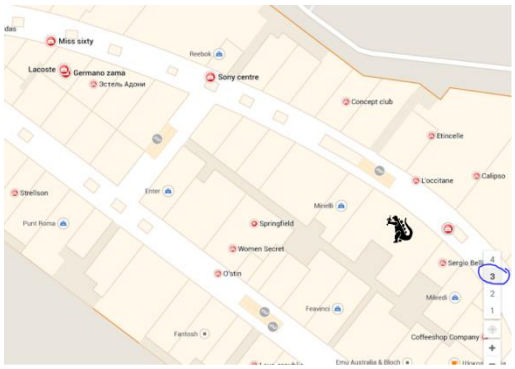
通常在墙壁之间画出尺寸，以说明房间的大小和墙壁的长度。平面图还可能包括水槽、热水器、炉子等装置的细节。楼层平面图可能包括施工说明，以明确装饰、施工方法，或电气项目的符号。

它也被称为平面图，是一个通常以 4 英尺（1.2 米）的楼面高度投影的测量平面，与之相对的是立面图，它是从建筑物的侧面沿其高度投影的测量平面，或者是建筑物沿轴线切割以显示内部结构的剖面或横截面。

楼层平面图经过数字化的再加工后，形成的一种新的数据结构，我们称之为室内地图。其特性不仅包括以表述空间特征的几何图形符号，同时，还包括用于室内楼层的复杂的路网信息。

在通常情况下，楼层平面图在计算机中一般以 CAD 的数据格式进行存储，在建筑工程期间，建筑设计人员通过测绘手段和 Auto CAD 工具，对建筑物的楼内进行制图；后期，通过其他专业软件，将原始的 CAD 楼层平面图，进行再加工，赋予其更加美观的图形符号化以及具备导航能力的道路路网数据，将其转换为室内地图（Indoor Map）。





1.2. 室内场景的实时定位

室内定位系统（Indoor Positioning System，以下简称 IPS）是一个设备网络，用于在 GPS 和其他卫星技术缺乏精度或完全失效的地方定位人或物体，如多层建筑内、机场、小巷、停车场和地下位置。

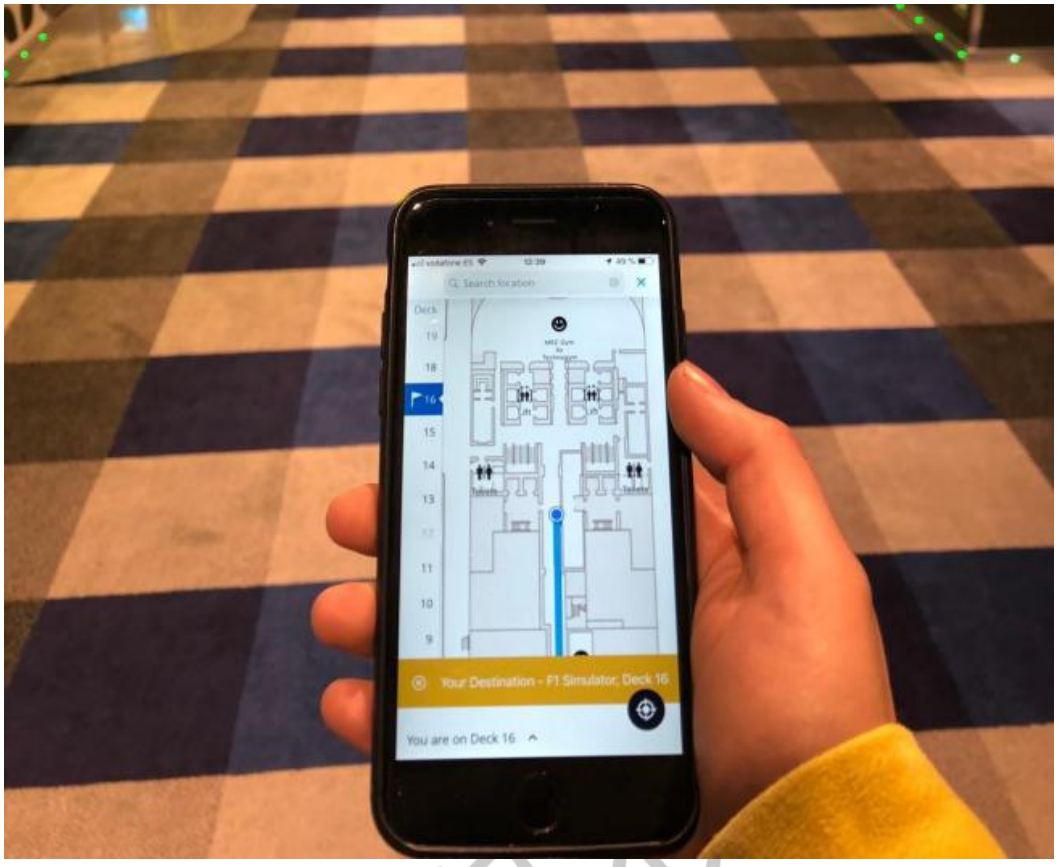
大量的技术和设备被用来提供室内定位，从重新配置已经部署的设备，如智能手机、WIFI 和蓝牙天线、数码相机和时钟；到专门建造的装置，将继电器和信标战略性地置于整个定义的空间。灯光、无线电波、磁场、声学信号和行为分析都被用于 IPS 网络。部分 IPS 可以达到 2 厘米的位置精度，这与在户外可以达到 2 厘米精度的启用 RTK 的 GNSS 接收机相当。IPS 使用不同的技术，包括对附近锚节点有已知固定位置的节点，如它们或者主动定位移动设备和标签，或者为设备提供环境位置或环境背景以获得感应。 IPS 的本地化性质导致了设计的分散，系统使用各种光学、无线电、甚至声学技术。

1.3. 应用场景

1.3.1. 室内寻路

寻路（或称寻路）包含了人（和动物）在物理室内空间中确定方向和从一个地方到另一个地方的所有方式。现代寻路系统已经开始纳入对人们迷路原因的研究，他们对标识牌的反应以及如何改进这些系统。

在医院等公共建筑中，室内寻路通常由信息亭、室内地图和建筑目录来帮助。室内寻路在办公楼中也同样重要。这样的空间涉及到游客正常词汇之外的领域，显示出对一套通用的独立于语言的符号的需求。为手持移动设备提供室内地图正变得普遍，数字信息亭系统也是如此。其他经常使用的寻路辅助工具是使用颜色编码和标牌集群，用来将信息分成一个层次，防止信息过载的问题。最近的一些机场航站楼包括天花板设计和地板图案，鼓励乘客沿着所需的方向流动。



1.3.2. 轨迹跟踪及回放

1.3.3. 基于 LBS 的服务

基于位置的服务（Location-Based Service，LBS）又称适地性服务、行动定位服务、位置服务、置于位置的服务，是通过移动运营商的无线电通讯网络（如 GSM 网、LTE 网）或外部定位方式（如 GPS）获取移动终端用户的位置消息（地理坐标）。在 GIS 平台的支持下，为用户提供相应服务的一种增值业务。

位基服务可以应用于不同的领域，例如：健康、工作、个人生活等。此服务可以用来辨认一个人或物的位置，例如发现最近的提款机或朋友同事的目前的位置，也能透过客户目前所在的位置提供直接的手机广告，并包括个人化的天气消息提供，甚至提供本地化的游戏。

2. 关键技术

2.1. 三维图形渲染技术

2.1.1. 基于 Web 端的三维图形渲染技术——WebGL

WebGL 是一种 JavaScript API，用于在不使用插件的情况下在任何兼容的网页浏览器中呈现交互式 2D 和 3D 图形。WebGL 完全集成到浏览器的所有网页标准中，可将影像处理和效果的 GPU 加速使用方式当做网页 Canvas 的一部分。WebGL 元素可以加入其他 HTML 元素之中并与网页或网页背景的其他部分混合[3]。WebGL 程序由 JavaScript 编写的句柄和 OpenGL Shading Language（GLSL）编写的着色器代码组成，该语言类似于 C 或 C++，并在电脑的图形处理器（GPU）上执行。

目前，WebGL 在最新的浏览器中被广泛支持。然而，其可用性取决于其他因素，如 GPU 支持。WebGL 官方网站提供了一个简单的测试页。而第三方网站提供了更详细的消息（如浏览器使用的渲染器以及可用的扩展）

2.1.2. 非 Web 场景下的三维图形渲染技术——OpenGL 和 Direct3D

OpenGL（Open Graphics Library）是用于渲染 2D、3D 矢量图形的跨语言、跨平台的应用程序编程接口。它由近 350 个不同的函数调用组成，用来从简单的图形比特绘制复杂的三维景象。而另一种与之对应的程序接口系统是仅用于 Microsoft Windows 上的 Direct3D。OpenGL 常用于 CAD、虚拟现实、科学可视化程序和电子游戏开发。

与 WebGL 相比，OpenGL 和 Direct3D 技术则更加面向底层，在操作系统级别和显示设备进行通信及控制，能够提供更加高效的性能和更优异的效果。

2.1.3. 新一代的三维图形渲染技术——云渲染

云服务正在全球范围演变为基础设施，应用上云大势所趋。云服务提供商加速边缘云部署，云节点位置不断下移，与终端用户的距离正在缩短，为各行各业应用上云提供便利和可能。作为显卡领域的主导厂商，英伟达早早的开始在 GPU 虚拟化上进行技术投入，并发布了 NVIDIA GRID 系列产品。而 AMD 和 Intel 也不甘落后，在 GPU 虚拟化上陆续推出产品，

GPU 虚拟化步入成熟。

基于云服务的渲染技术目前主要应用于云游戏领域，云渲染将内容的存储、计算和渲染都转移到云端，实时的游戏画面串流到终端进行显示，最终呈现到用户眼中。云游戏又被称为 GaaS（Game as a Service），它将游戏体验变成了一种服务，提供给广大消费者用户，解决了用户不断购买或升级终端的困扰，也避免了下载和更新内容的繁琐，在成本、时间、 内容、维护等方面提升了游戏体验的易用性。

相比客户端渲染技术，云渲染的技术方案则将实施成本分摊到了服务器硬件部分，实施开销很大一部分是硬件或云服务的使用成本。在现阶段技术加持下，其私有化成本较客户端渲染技术高出不少。

2.2. 室内定位技术

在室内环境无法使用卫星定位时，使用室内定位技术作为卫星定位的辅助定位，解决卫星信号到达地面时较弱、不能穿透建筑物的问题。最终定位物体当前所处的位置。常见的室内无线定位技术还有：Wi-Fi、蓝牙、红外线、超宽带、RFID、ZigBee 等。

2.2.1. 基于蓝牙信标（iBeacon）的定位

蓝牙通讯是一种短距离低功耗的无线传输技术，在室内安装适当的蓝牙局域网接入点后，将网络配置成基于多用户的基础网络连接模式，并保证蓝牙局域网接入点始终是这个微网络的主设备。这样通过检测信号强度就可以获得用户的位置信息。

蓝牙定位主要应用于小范围定位，例如：单层大厅或仓库。对于持有集成了蓝牙功能移动终端设备，只要设备的蓝牙功能开启，蓝牙室内定位系统就能够对其进行位置判断。不过，对于复杂的空间环境，蓝牙定位系统的稳定性稍差，受噪声信号干扰大。在目前主流市场中，基于蓝牙信标的方案，造价相对低廉，实施成本更低。

2.2.2. 基于无线网络（WIFI）的定位

通过无线接入点（包括无线路由器）组成的无线局域网络(WLAN)，可以实现复杂环境中的定位、监测和追踪任务。它以网络节点（无线接入点）的位置信息为基础和前提，采用经验测试和信号传播模型相结合的方式，对已接入的移动设备进行位置定位，最高精确度大

约在 1 米至 20 米之间。如果定位测算仅基于当前连接的 Wi-Fi 接入点，而不是参照周边 Wi-Fi 的信号强度合成图，则 Wi-Fi 定位就很容易存在误差（例如：定位楼层错误）。

另外，Wi-Fi 接入点通常都只能覆盖半径 90 米左右的区域，而且很容易受到其他信号的干扰，从而影响其精度，定位器的能耗也较高。

2.2.3. 基于超宽频（UWB）的定位

超宽带技术与传统通信技术的定位方法有较大差异，它不需要使用传统通信体制中的载波，而是通过发送和接收具有纳秒或纳秒级以下的极窄脉冲来传输数据，可用于室内精确定位，例如：战场士兵的位置发现、机器人运动跟踪等。

超宽带系统与传统的窄带系统相比，具有穿透力强、功耗低、抗多径效果好、安全性高、系统复杂度低、能够提高精确定位精度等优点，通常用于室内移动物体的定位跟踪或导航。

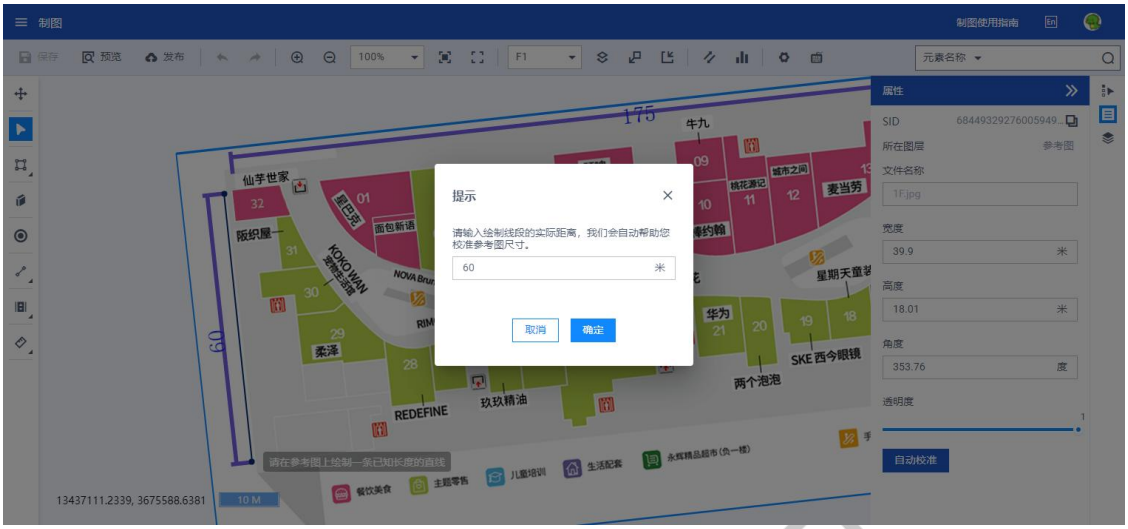
3. 产品能力

3.1. 室内空间制图

3.1.1. 参考图及配准

蜂鸟地图室内空间制图的核心方式是通过“描绘”的方式，进行基础的室内空间制图作业。参考图是制图过程中重要的一个资源，用户在制图之前，要选定合适的参考图作为绘制线画图的“基础底图”。导入参考图后，可以通过矢量绘制的方式，在参考图之上进行“描绘”。

同时，为保证绘制出来的矢量地图和实际建筑空间尺寸的一致性，要对参考图进行一步配准的工作，通过选定参考图上的一段线段长度，并给予其实际空间长度，工具会自动将参考图缩放至一个正确的缩放比例，用户在该缩放比例下绘制的矢量图将符合实际的空间尺寸。



3.1.2. 建筑物楼层命名和管理

对于室内地图来说，楼层是对建筑的细分，同时又是对要素在空间上的汇总，即一个建筑由多个楼层组成，每个楼层上又被划分为多种要素。

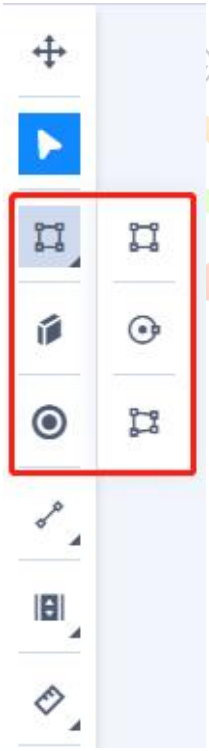
为此，蜂鸟制图为用户提供单独的楼层管理模块，用于管理建筑的楼层信息，包含添加楼层、删除楼层、调整楼层顺序，以及对楼层属性的添加和修改。添加楼层时，考虑到室内上下楼层基本一致的特殊结构，我们会默认为用户创建一个与 F1 层范围一致的楼层，用户根据当前层的参考图进行微调。楼层顺序通常与现实中的顺序一致：最底层为 F1，依次往上 F2，F3...。楼层属性包括楼层名称、楼层别名，楼层名称通常与平面图或电梯位置处楼层名称的标识一致，别名通常是大众对建筑认知的名称，不用的命名，在应用时按需求显示楼层名称或别名。

3.1.3. 地图要素绘制

地图要素是对地图中最小单元的表达，一个地图要素由要素形状和要素属性组成，本章节主要讲述要素形状的绘制。根据要素的表达形式，其表达形式有：面要素、点要素、线要素。

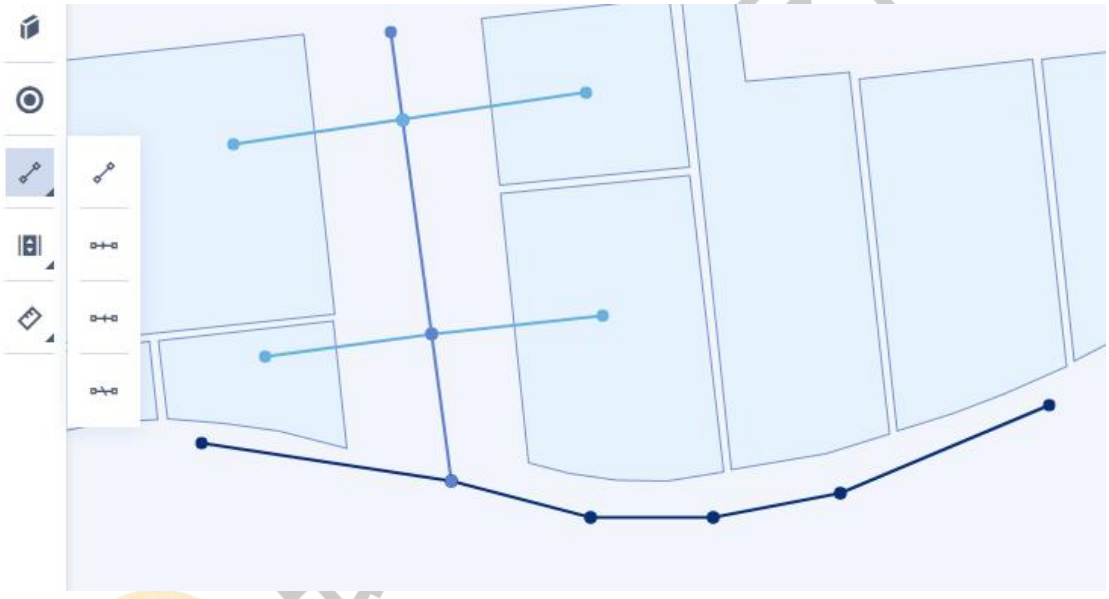
面要素用于表达地图上的区域，区域可以是现实中存在的实体区域，如商场中的店铺、展会中的展区等；也可以是虚拟的、人为定义的、用于体现空间上的范围，如停车场中的区域，开放式办公区里的部门，应用渲染时以三维面块的形式展现在地图上来体现室内的布局和分布。制图中为用户提供了绘制矩形、绘制圆形、绘制自定义多边形、绘制墙的工具。矩形绘制功能，提供了绘制指定宽高的矩形的能力；圆形绘制工具，提供了绘制指定半径的圆的能力；使用圆形和矩形绘制工具，既能提升制图要素的精度又能使绘制成果更美观；墙绘制工具，是利用墙的平行结构，固有的属性信息，将其抽象打包为一个固定的制图组件，一方面既能自定义墙体厚度，另一方面又能在绘制形状的同时

点要素是对一些常见公共设施的抽象，如服务台、电梯、卫生间，在制图时以点的形式表示，应用渲染时以图标的方式展示在地图上便于快速指引或查看其对应位置。面要素绘制工具又细分为墙体绘制工具、矩形绘制工具、圆形绘制工具、多边形绘制工具等。已绘制的工具支持形状编辑、移动位置、拉伸变形、拆分合并、线段转弧等多种操作。



3.1.4. 通行路网绘制和通行设施管理

通行路网和通行节点作为路径规划的主要依据，需要在制图层面绘制维护。蜂鸟制图为用户提供快捷的路网绘制工具及通行节点管理工具。通行路网包含路网等级和路网方向两个属性。路网级别是指显示中建议人们行走的道路的权重，例如商场中从正门进入后大厅的路通常为主干道路，通往卫生间、偏僻小店的路则为次要道路，进入店铺的路则为小路；路网方向是指实际场景管理方建议或规定的人流行径方向，如在机场、火车站等安检/检票的部分，路径则为单向通行。路径规划在通过识别不同的道路权重可以为用户提供优先主路的规划策略，路径方向为用户提供便捷规范的行径路线。

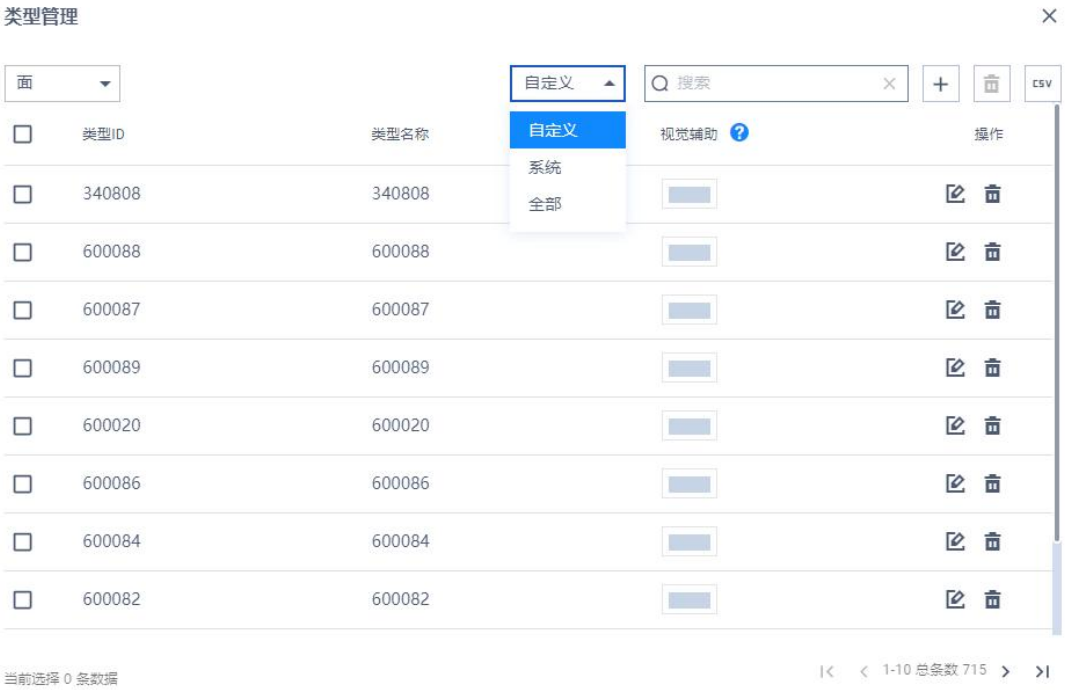


通行设施包含楼梯、直梯、扶梯三种类型，蜂鸟制图将一组同类型的设施定义为一个通行设施。一个设施抽象为一个通行节点，多个处于同一位置、且上下可通行的通行节点组成一组通行设施。通行设施包含通行设施的名称、类型、向上可到达的楼梯、向下可到达的楼梯以及是否为无障碍。通过先创建一组通行设施，再在每个可到达楼层地图上对应的位置标记通行设施节点，并将其绑定为对应的通行设施来实现跨楼层的路径规划。通行节点为具体到某一楼层的设施点，配置时，可通过批量打点的功能快速的，为其所能到达的所有楼层同一位置添加同样的设施点，以大大提升用户的制图效率。

3.1.5. 地图要素类型管理

地图要素类型，是地图数据不可或缺的属性之一，它是从应用的角度将地图按照不同的功能进行的划分，如购物类、餐饮类、冷饮类等等。用户可以根据自己的场景，结合实际应用需求，将地图上的要素进行分类，便于后续的按类型检索、按类型设置颜色等一系列通过

类型对地图的使用。而类型管理，为用户提供了当前账户下分类的一个汇总，以及对各类型进行增删改查的操作。类型管理按类型的来源又分类系统类型和自定义类型。系统类型为制图工具内置的类型，通常为一些常见较基础的类型，如墙、综合类型、运动户外、鞋包配饰；自定义类型为用户自己定义的类型。每个类型包含类型 ID、类型名称、视觉辅助。数据中以类型 ID 进行展示；类型名称用于标识类型 ID；视觉辅助用于为不同的类型设置颜色，以方便识别。



3.1.6. 地图要素属性管理

各个地图要素都有其对应的属性信息，包含中英文名称、类型、高度、显示级别等等字段，后期应用阶段用于搜索/显示使用。不同图层的要素包含不同的属性信息。属性信息支持单个要素修改，也支持批量元素修改。

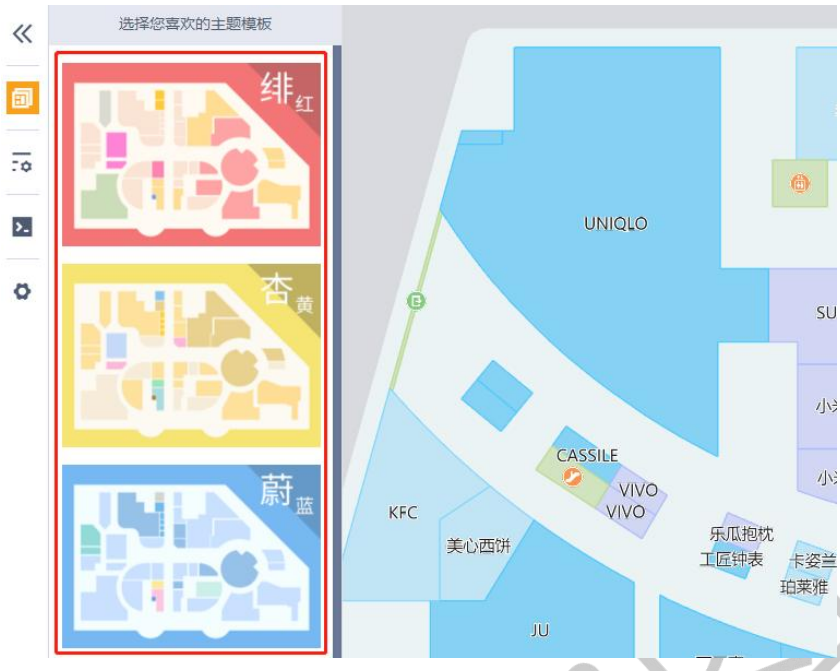
字段名称	字段含义	示例内容
SID	唯一索引	5795180201939
类型	类型 ID	300000
中文	中文名称	耐克

名称		
英文名称	英文名称	Nike
高度	面的拉伸高度	2m
显示级别	地图元素开始渲染和消失渲染的级别	0~29
面积	元素面积	48.36 ㎡
通行状态	路径规划时的通行状态	可到达且允许经过
标注高度	标注的三维高度	2.2m
坐标X	x	13383210
坐标Y	y	3533284

3.1.7. 地图主题样式配置及管理

地图绘制完成后，可以在主题编辑器中为地图进行样式配置。

主题模板：主题编辑器内置 6 种模板供用户选择。选择模板，可一键为地图配置颜色。



图：主题模板

除了主题模板，还提供了**详细配置**，包含全局配置和楼层配置。

全局配置包含：地图背景配置、按图层配置和按类型配置。

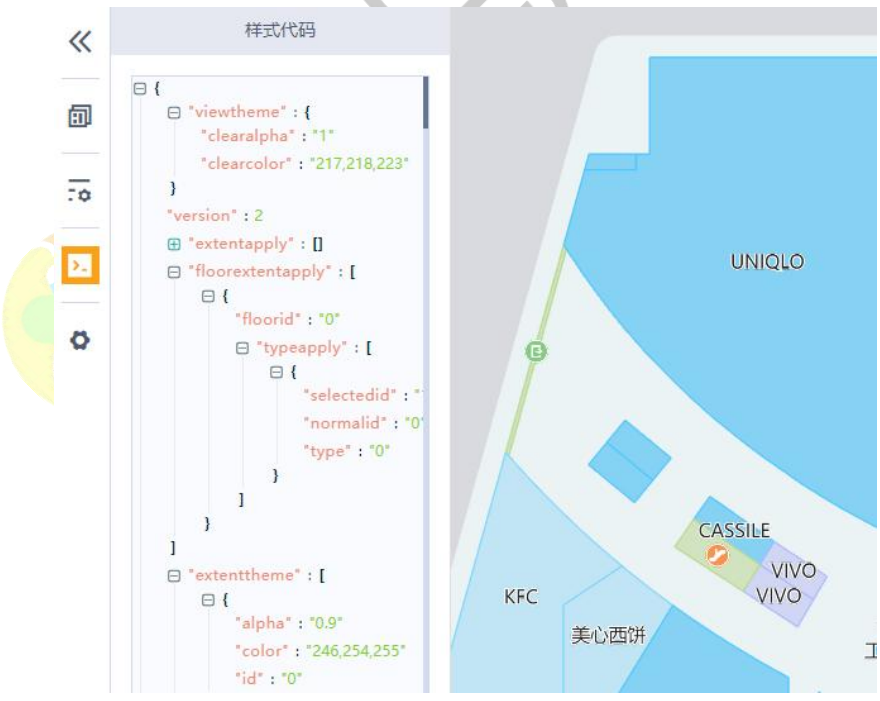
楼层配置包含：单面配置、单标注配置和单图标配置。

图：详细配置

图标管理：在制图编辑器绘制图标时，如果图标类型选择的是系统类型，在主题编辑器中会自动匹配系统类型默认图标。如果选择的是自定义类型，当个人图标库中没有自定义类型对应图标时，显示蜂鸟默认图标。有对应类型图标时，加载个人图标库图标。

图：图标管理-系统图标

在主题样式配置中，单个面元素配置的优先级高于按类型配置面图层的优先级。对单个面元素进行设置后，会覆盖按类型配置的样式。如果已保存单个面元素的配置内容，要恢复到按类型配置面图层的统一样式，需要使用【清除配置】功能。清除对单个面元素的配置，恢复到按类型配置面元素。单标注和单图标配置的优先级同理。



图：样式代码

样式代码：代码内容为当前主题的 theme 配置，根据当前配置的样式，实时变动。

图：全局设置

全局设置：对 UI 显示、地图控制和撤销回退的设置。

导入视觉辅助样式：视觉辅助样式来源于制图工具中的类型配色，导入样式后，会对类型配置中的面图层的填充颜色进行覆盖。如果对单个面元素的填充颜色进行了配置，则配置过面的颜色不会被覆盖。

样式刷：选中已配置过的元素，给没有赋值的元素进行快速赋值。选中一个面元素，获取选中 store 面和面上 label 的样式配置（顶面贴图不复制），作为基准样式，点击样式刷，再选中目标 store 面，把样式配置赋值给目标。使用样式刷时，基准面元素上没有 label，赋值时，就不改变目标面元素上的 label 配置。如果基准面上有 label，被赋值的面上没有 label，则给目标面上的空 label 赋值，当面上有 label 时，与复制面上的 label 配置一致。

连续刷：选中基准面，点击样式刷，再按下 Ctrl 键，点击一个目标面，渲染后，再点击下一个面，依次操作。

一个地图可以创建多个主题文件，也就是可以为一个地图配置多种样式，进行数据调用时可以指定主题 ID 进行访问即可。

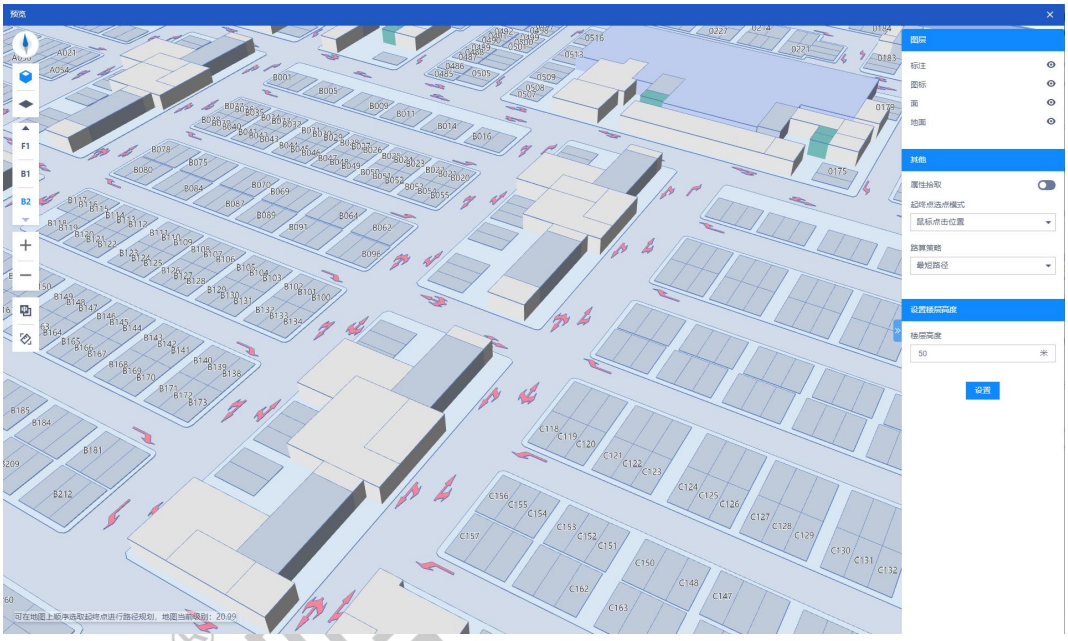
3.1.8. 保存和发布

编辑后的数据点击保存后能保存当前的修改记录，而不影响在线调用时的成果。而点击

发布后，平台会先保存，再将当前状态的数据发布到在线调用/数据下载时所访问的服务器上，此时如果有应用在线调用该数据，则会直接影响该应用。

3.1.9. 地图数据预览

预览主要用于对编辑数据的检查和三维效果的查看。通过预览能查看已绘制数据的二/三维展示效果、属性信息、路径规划等是否符合预期，如不符合，能随时再进入编辑界面调整数据。



3.1.10. 地图数据导出和下载

发布后的数据支持下载和导出。下载的数据分为 2 个版本，v1 版和 v2 版，v2 为 v1 版本的升级版，直接体现是数据量较小，不同的版本对应不同的 sdk。如 v1 版适用于 JS sdk2.4 以前的版本，v2 版则主要适用于 JS SDK v2.4 及之后的版本。下载的数据为*.fmap 格式的压缩包，主要用于 sdk 的离线加载使用。导出主要是指对面元素属性信息的导出，包含面元素的中心点坐标，方便用户后续与自有平台的数据对接。

3.2. 空间业务数据融合

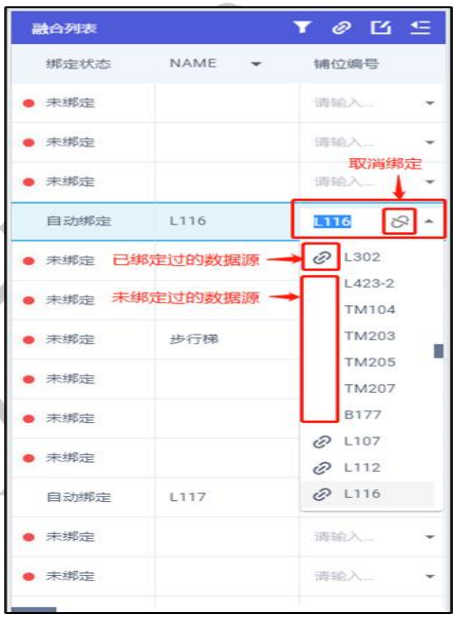
3.2.1. 数据源管理

蜂鸟制图为用户提供最基础的地图数据支撑，业务数据则是对地图数据的扩展延伸，同时也是应用层面为管理层做出决策的主要依据。而业务数据大多存储于业务侧服务器或内网服务，很难将其与地图相结合、发挥其价值，为此，蜂鸟为用户提供了存储、管理或连接业务数据的工具——数据源管理，主要解决数据存储、管理和连接的问题，为地图和业务数据做关联提供基础支撑。

数据源管理导入数据源和编辑数据源两部分。导入的数据源支持：静态数据、API 数据、空表数据。其中可除 API 数据源为外链数据不可编辑外，其余可通过数据源编辑工具进行编辑。如下为数据源编辑的主界面，包含对数据源的增删改查及导出功能。

3.2.2. “数图融合”

数据融合即为地图数据与业务数据（数据源）建立关联。基于数据源的最终应用方式，数据融合方式可以分类两类：扩展属性融合和叠加图层融合。扩展属性融合是指将数据源作为地图元素的附加属性进行展示，使地图信息更丰富。多以地图元素的详情信息，或以专题图形式展现；叠加图层融合是将数据源作为蜂鸟地图标准图层外的自定义图层存在，内容包含几何形状和属性信息，以覆盖物形式叠加在地图上展示。叠加融合即为数据源赋予/绘制一种几何形状，如点、线、面，其是将数据源与图形的融合。



扩展属性的融合原理是基于字段的融合，通常是将地图的 SID 和数据源的唯一 ID 进行绑定关联的过程，最终得到关联后的融合成果。如字面意思，扩展属性类的融合成果主要作为蜂鸟地图标准字段的补充字段，在地图上以店铺的详情信息进行展示。

叠加图层的融合原理主要是结合地图，给数据源绘制空间数据，包括点、线、面；操作时需要先选中要绘制的数据源，点击绘制按钮开始绘制。已绘制的数据支持移动、形状编辑的操作。

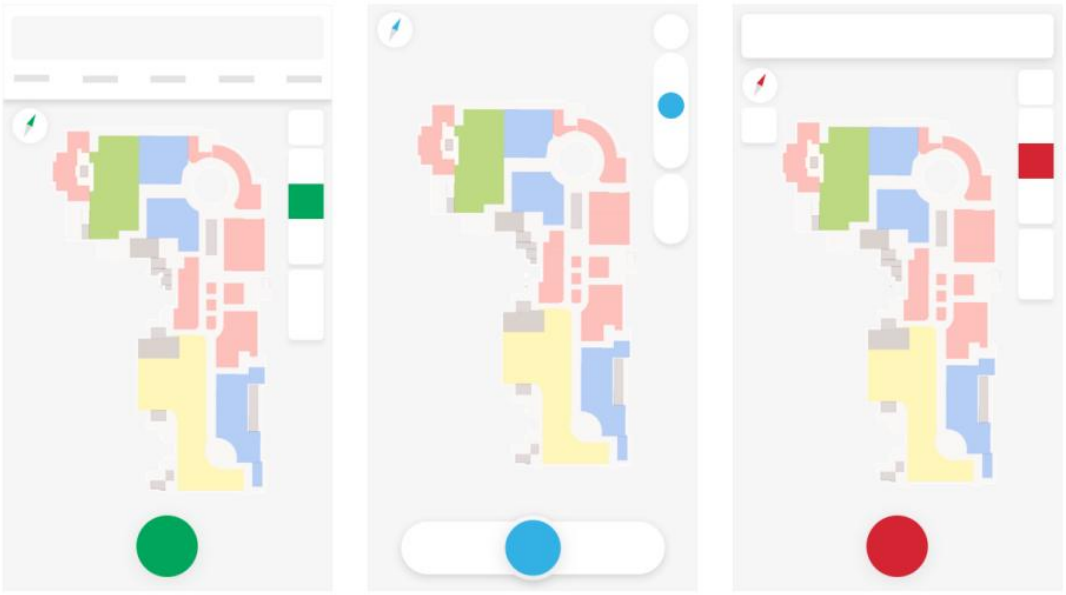
3.2.3. 融合数据的发布和使用

融合结束需要发布形成融合成果，融合成果以 API 方式提供给用户进行使用。通过 Open API 的 AK 和 SK 进行签名方式的验证，以确保数据的安全访问，访问内容包含查询地图楼层信息的 API 和获取指定融合数据的 API，用户按需接入后进行访问。

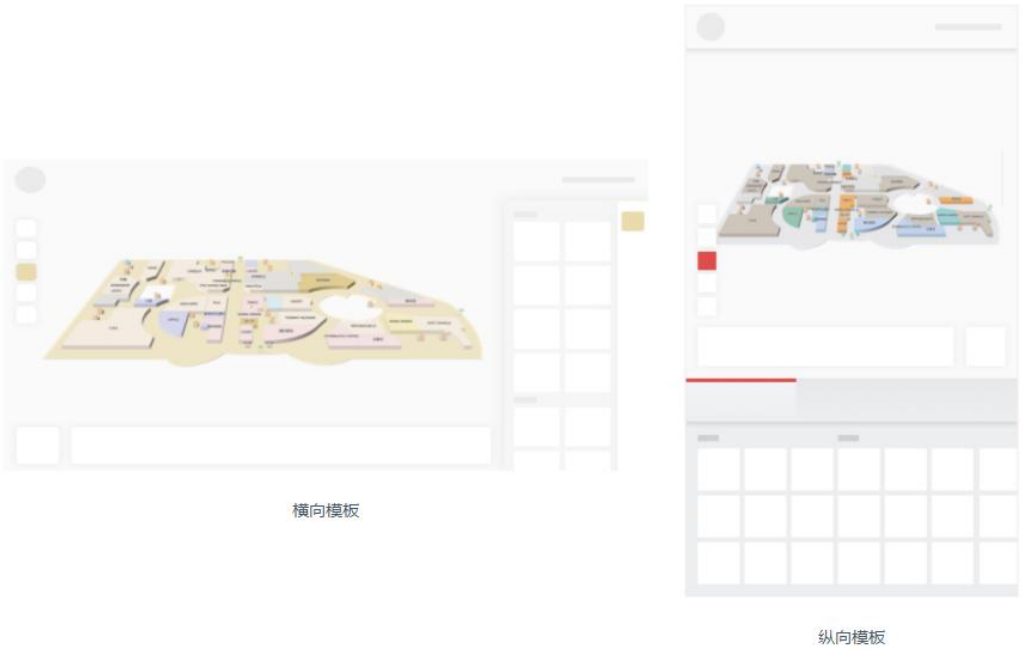
3.3. 低代码快速应用交付

3.3.1. 面向场景的应用模板

蜂鸟云微程序为用户提供了手机端和导览端两类模板。手机端模板适用于手机端导航导览；导览端模板适用于商场大屏端展示。系统内置了 4 个手机端模板和 2 个导览端模板。进入编辑器，使用模板，可以一键配置程序布局和样式。



图：手机端模板



图：导览端模板

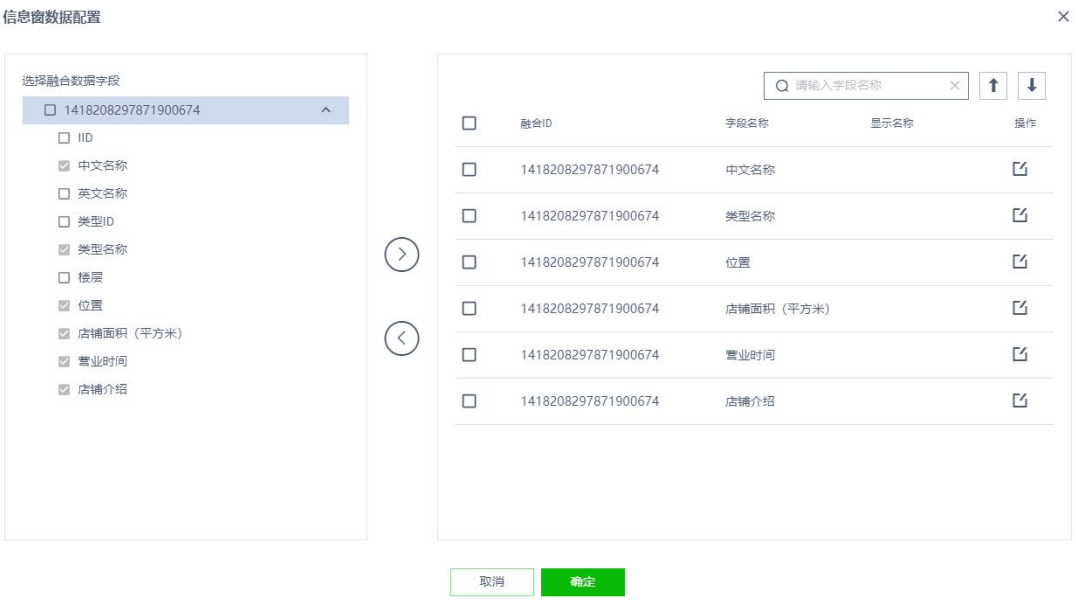
3.3.2. 在应用中使用自己制作的室内地图

创建时，可以选择已发布的地图创建手机端或导览端程序，把制作好的室内地图加载到创建的应用中。未发布的地图不能创建微程序。

3.3.3. 在应用中使用融合数据

在微程序中，通过 Web API 的方式调用数据融合服务。Web API 的 Key 可在开发控制台中创建获取。手机端配置中**信息窗和自定义图层**两个组件可以使用融合数据；导览端配置中**信息窗**组件可以使用融合数据。

手机端-信息窗：可展示融合数据的扩展型属性值和叠加图层的属性值。配置如下：



图：信息窗数据配置

信息窗数据配置弹窗中，左侧为当前地图关联的融合数据字段，右侧为已选择的要展示的融合字段。配置后效果如下图所示：

图：手机端-信息窗

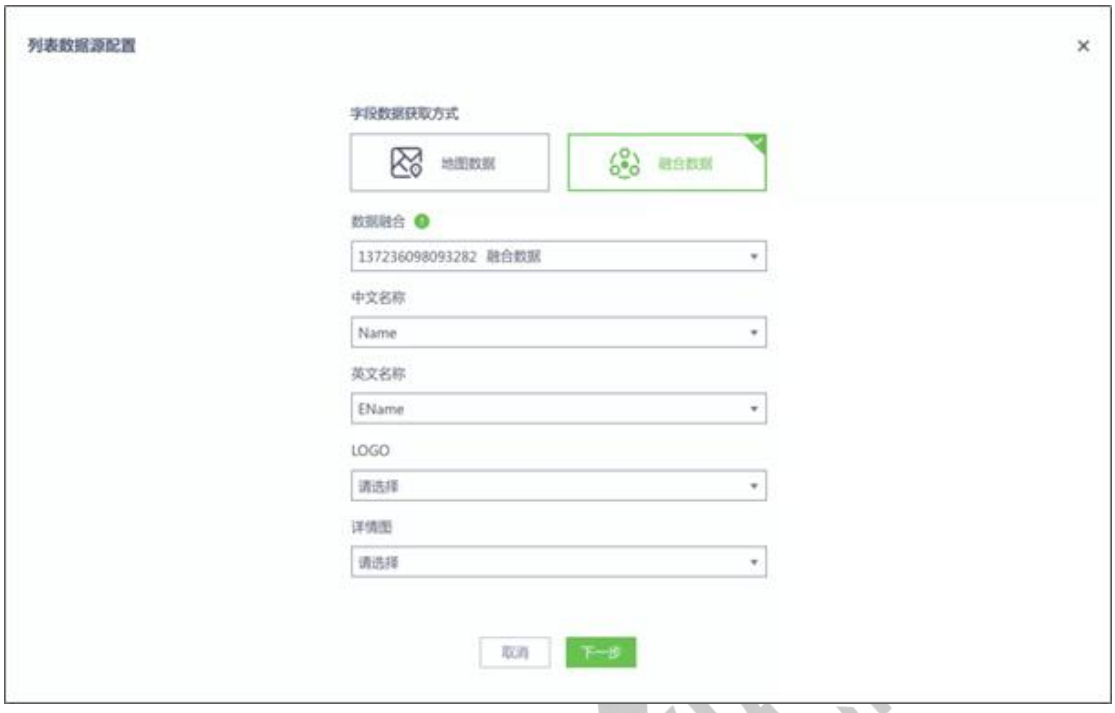
手机端-自定义图层：对叠加型融合数据的配置。叠加型融合数据分为三种：点、线、面。点类型数据，在预览时，地图上显示配置的图标。线和面类型数据，在预览时，地图上显示数据融合工具中绘制的线和面。

图：自定义图层数据配置

在数据配置弹窗中勾选融合数据的属性字段，预览时，点击地图上的点或面，被勾选的属性字段会展示在信息窗中；如果没有勾选任何属性，则不展示。配置后如下图所示：

图：手机端-自定义图层

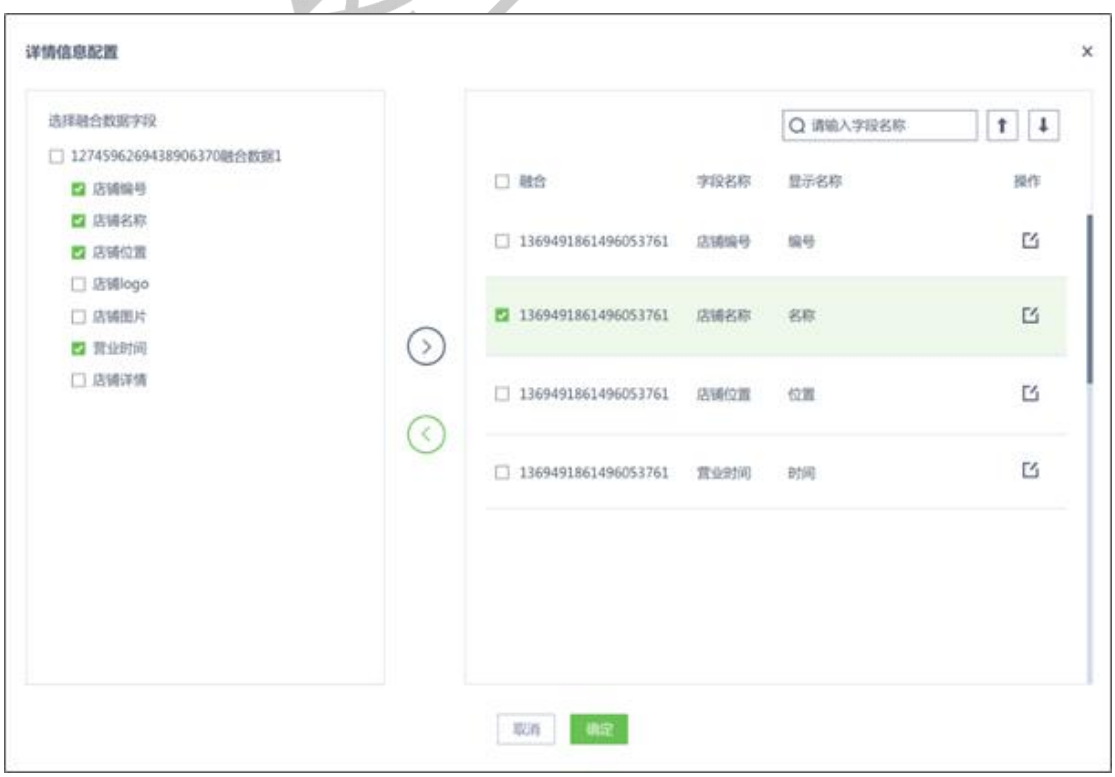
导览端-信息窗：数据配置包含列表信息和详情信息配置。



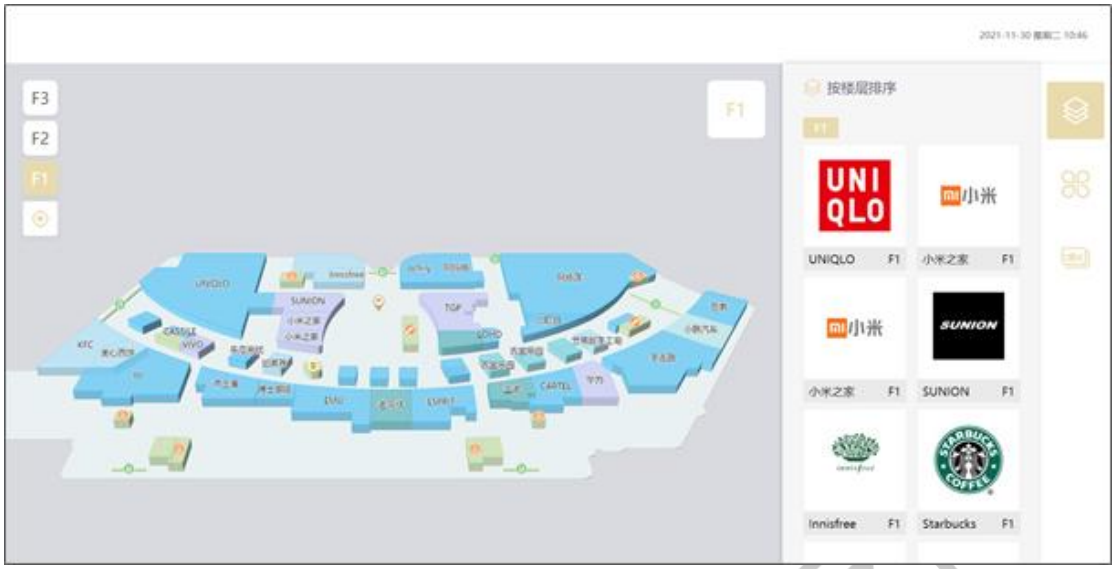
图：列表数据源配置

列表信息是指列表中要展示的店铺信息，数据可以来源于地图数据或融合数据。

详情信息是指信息窗中店铺的详情信息。详情信息配置是对在信息窗中要展示哪些融合字段的配置。如下图所示：

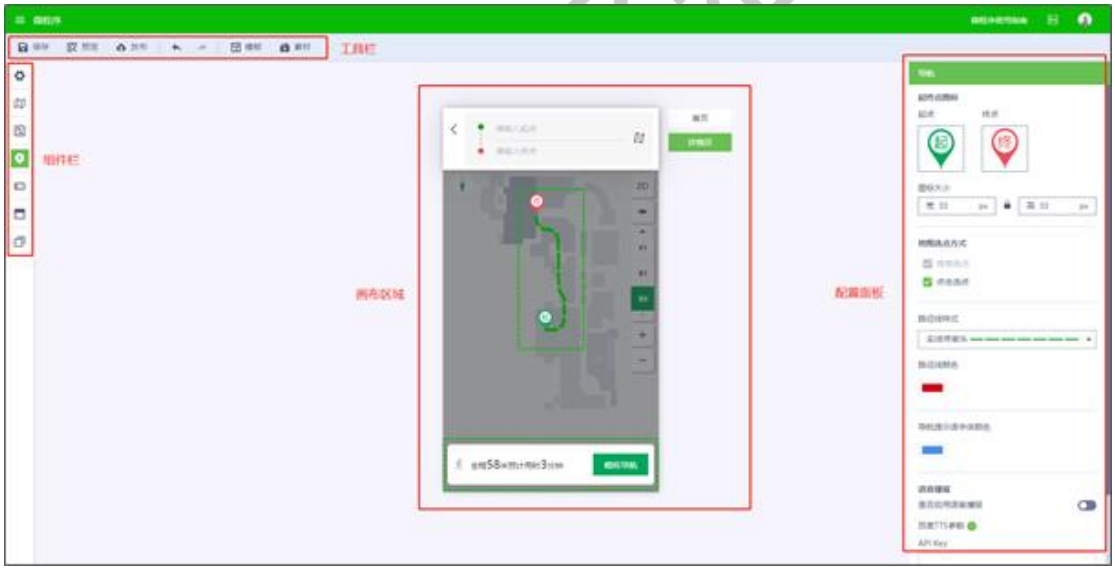


图：详情信息配置



图：导览端-列表信息

3.3.4. 无代码化的应用界面和功能配置



图：手机端-编辑页面

上图为手机端编辑页面，分为 4 个区域，上：工具栏；左：组件栏；中：画布区域；右：配置面板。工具栏：包含保存、预览、发布、撤销回退、模板、素材管理。组件栏：包含地图、楼层控件、导航、搜索栏、信息窗、自定义图层。画布区域：高亮显示配置组件在页面上的内容和位置。配置面板：显示选中组件的配置项。

地图组件：对地图进行进行配置，包含初始化地图显示设置、地图选点图标配置。

楼层控件：对指北针和楼层控件的配置。

导航组件：对导航起终点、地图选点方式、路径线和语音播报的配置。

搜索栏组件：对搜索框和按类型搜索的设置。

信息窗组件：对信息窗数据配置。信息窗中的数据包含地图基础信息（元素名称、所在楼层）和融合数据的扩展型属性值和叠加图层的属性值。

自定义图层组件：对叠加型融合数据的配置。配置包含入口设置、自定义图层抽屉和自定义图层数据配置。

图：导览端-编辑页面

上图为导览端编辑页面，分为 4 个区域，上：工具栏；左：组件栏；中：画布区域；右：配置面板。工具栏：包含保存、预览、发布、撤销回退、模板、素材管理、导览屏管理。组件栏：包含地图、楼层控件、信息窗、图例、标题栏、图片。画布区域：高亮显示配置组件在页面上的内容和位置。配置面板：显示选中组件的配置项。

地图组件：对地图进行配置。包含初始化地图显示设置和图标设置。

楼层控件：对楼层按钮和楼层标识的配置。配置包含楼层样式、展示位置、最多展示楼层数据、聚焦楼层文字颜色、聚焦楼层背景颜色、复位图标设置、按钮间距和圆角设置，以及楼层标识的字号、颜色和透明度的设置。

信息窗组件：对信息窗内展示内容的配置。信息窗内展示的内容包含品牌列表、店铺详情、路线指引。

图例组件：对地图上面元素的类型配色和 POI 类型说明。

标题栏组件：对 LOGO、标题和显示时间进行配置。

图片组件：用户可以上传自己需要的图片，可以是移动端程序二维码或公众号图片等。

3.3.5. 应用多平台发布

移动导航应用程序配置完成后，会生成 H5 链接。使用方式大致有 3 种：

（1）直接使用 H5 链接，进行模拟导航。

（2）将 H5 链接嵌入到自己的应用程序中。

（3）作为微信小程序组件的 src 属性，以外链的方式进行加载，嵌入到微信小程序中。

导览屏应用程序配置完成后，每个导览屏均对应生成一个 H5 链接。导览屏应用程序主要应用于商场的导览屏客户端，给予商场顾客查询店铺和路线指引的功能。用户可以将导览屏应用程序的 H5 链接进行部署，或嵌入到自己的应用程序中，让其成为自己应用功能的一部分。

3.4. 室内三维地图二次开发

3.4.1. 多平台的二次开发包

在蜂鸟云产品套件中，我们提供了不同平台下供开发者使用的应用程序开发套件（SDK）。

FengMap SDK for JavaScript，基于浏览器的 Web 解决方案，采用 WebGL 渲染技术，在现代浏览器下，提供了更加高效的三维场景渲染解决方案。不仅支持常用的前端开发框架，诸如，VUE、React 等，在 WebView 的支持下，能够无障碍的将三维地图场景集成入您的移动端应用程序中。

FengMap SDK for MiniProgram，在更加统一的微信体系下，蜂鸟云提供了原生支持微信小程序开发环境的应用程序开发套件，供广大微信小程序开发者使用。

FengMap SDK for Mobile 提供了更加原生的支持，使蜂鸟地图在在 Android 和 iOS 环境中得以完美支持。面向各大基于 Android 生态的设备生产厂商开发者提供了完整的技术解决方案。

3.4.2. 在应用程序中使用蜂鸟地图

引入蜂鸟地图核心模块，在您的地图展示页面文件中添加如下代码完成对地图的初始化。

在地图初始化完毕之后，构造地图对象。备注：地图加载必须设置宽、高。示例代码如下：

```
var map;

var options = {
    appName: '蜂鸟研发 SDK_2_0',
    key: '57c7f309aca507497d028a9c00207cf8',
    mapID: '1321274646113083394',
    container: document.getElementById('fengmap'),
    mapURL: './data/',
    themeURL: './data/theme/',
    themeID: '1351477465818427393',
 }

 map = new fengmap.FMMap(options);
```

3.4.3. 在蜂鸟地图上添加标注

蜂鸟地图提供根据地图坐标点绘制要素功能。可绘制要素分为文字要素（FMTextMarker）、图片要素（FMImageMarker）、线要素（FMFMLineMarker）、多边形要素（FMPolygonMarker）、自定义要素（FMDomMarker）、定位要素（FMLocationMarker）、动态模型要素（FMDynamicModel）、几何体要素（FMExtrudeMarker）以及热力图绘制（FMHeatMap）。

添加图片要素

图片要素是用户在指定坐标点位置添加自定义的图片。示例代码如下：

```
map.on('loaded', function() {
/* 构造 Marker */
```

```
        var marker = new fengmap.FMImageMarker({
            url: 'images/red.png',
            x: getRandomCoords(map.getBound()).x,
            y: getRandomCoords(map.getBound()).y
        });
        var level = map.getLevel()
        var floor = map.getFloor(level);
        marker.addTo(floor);

        /* 移除 Marker 请调用 marker.remove() */

        setInterval(function() {
            if (!onAnimate) {
                /* 改变 Marker 位置 */
                marker.moveTo({
                    x: getRandomCoords(map.getBound()).x,
                    y: getRandomCoords(map.getBound()).y,
                    animate: false,
                    finish: function() {
                        onAnimate = false;
                    }
                });
            }
        }, 500)
    });
```

添加文字要素

文字要素是用户在指定坐标点位置添加自定义的文字。示例代码如下：

```
    map.on('loaded', function() {
        /* 构造 Marker 并添加到地图上 */
        textMarker = new fengmap.FMTextMarker({
            text: 'Hello FengMap',
            x: map.getCenter().x,
            y: map.getCenter().y
        });
        var level = map.getLevel()
        var floor = map.getFloor(level);
        textMarker.addTo(floor);

        /* 要从地图上移除 Marker 可以调用 textMarker.remove() */
```

```
            var onAnimate = false;
            setInterval(function() {
                if (onAnimate == false) {
                    onAnimate = true;
                    /* 动画移动 Marker 位置 */
                    textMarker.moveTo({
                        x: getRandomCoords(map.getBound()).x,
                        y: getRandomCoords(map.getBound()).y,
                        animate: true,
                        duration: 2,
                        finish: function() {
                            /* 动态改变文字内容 */
                            textMarker.text = textMarker.x.toString() + ',' +
textMarker.y.toString();
                            /* Marker 内容更新后必须调用 update() 方法使其生效 */
                            textMarker.update();
                            onAnimate = false;
                        }
                    });
                }
            }, 500)
        });
```

添加线要素

支持绘制线要素操作，用户可根据坐标点集绘制折线。注：线是针对整个地图的，线段的点可以是不同楼层的点。示例代码如下：

```
    map.on('loaded', function() {

            /* 构造线对象 */
            var segment = new fengmap.FMSegment();
            const height = 3;
            segment.points = [{
                x: 12619607.715459315,
                y: 2621869.145377799,
                z: height
            }, {
                x: 12619615.015019862,
                y: 2621866.573966666,
```

```
            z: height
        }, {
            x: 12619611.016728846,
            y: 2621855.267326041,
            z: height
        }];
        /* 使用 level 参数定义要在哪个楼层上绘制线覆盖物 */
        segment.level = map.getLevel();
        /* 构造线 Marker */
        var lineMarker = new fengmap.FMLineMarker({
            segments: [segment]
        });
        /* 向指定的地图添加 Marker，和 LocationMarker 一样，LineMarker 只能向 Map 添加，而不是其他类型 Marker 那样，向楼层对象添加，LineMarker 添加后无法修改其所在楼层 */
        lineMarker.addTo(map);

        /* lineMarker.remove() */
    });
```

添加多边形要素

支持绘制多边形要素操作，用户根据坐标点集绘制多边形、圆形、矩形。注：多边形是针对楼层的，每一楼层可包含多个多边形。示例代码如下：

```
map.on('loaded', function() {

        var level = map.getLevel()
        var floor = map.getFloor(level);

        /* 矩形 */
        var rect_width = 4.5;
        var rect_height = 6;
        var rect_center = {
            x: 12619620,
            y: 2621864
        };
        /* 使用 FMCalculator 的矩形构造器可以快速构造出矩形的几何形状坐标集合 */
        var rectangleOption = {
            points: fengmap.FMCalculator.rectangleBuilder(rect_width, rect_height, rect_center),
```

```
            x: 12619620,
            y: 2621864,
            height: 1
        };
        /* 构造 Marker */
        var rectangle = new fengmap.FMPolygonMarker(rectangleOption);
        /* 添加到地图的指定楼层上 */
        rectangle.addTo(floor);

        /* 圆形 */
        var circle_radius = 1.5;
        // segments 影响形成的圆形的边数和顶点数量，数值越大圆形在视觉上越 '平滑'
        var circle_segments = 25;
        var circle_center = {
            x: 12619625,
            y: 2621871
        };
        /* 使用 FMCalculator 的矩形构造器可以快速构造出圆形的几何形状坐标集合 */
        var circleOption = {
            points: fengmap.FMCalculator.circleBuilder(circle_radius, c
ircle_center, circle_segments),
            x: 12619625,
            y: 2621871
        }
        var circle = new fengmap.FMPolygonMarker(circleOption);
        circle.addTo(floor);

        /* 任意多边形 */
        var polygonOption = {
            x: 12619625,
            y: 2621871,
            height: 1,
            points: [{
                x: 12619609,
                y: 2621872
            }, {
                x: 12619617,
                y: 2621875
            }, {
                x: 12619618,
                y: 2621868
            }],

        }
```

```
        var polygon = new fengmap.FMPolygonMarker(polygonOption);
        polygon.addTo(floor);
        /* 要移除 Marker 请调用 remove 方法 */
        /* polygon.remove() */
        /* circle.remove() */
        /* rectangle.remove() */
    });
```

添加自定义要素

可根据实际业务需求，在地图指定位置添加自定义 dom 要素。 示例代码如下：

```
map.on('loaded', function() {
        /* 构造 Marker */
        var domMarker = new fengmap.FMDomMarker({
            x: 12619607,
            y: 2621869,
            content: '<p class="my-popup">Hello</p>'
        });
        var level = map.getLevel()
        var floor = map.getFloor(level);
        domMarker.addTo(floor);

        /* 要移除 marker 只需调用 domMarker.remove() */

        setInterval(function() {
            if (!onAnimate) {
                /* 改变 marker 位置的方法 */
                domMarker.moveTo({
                    x: getRandomCoords(map.getBound()).x,
                    y: getRandomCoords(map.getBound()).y,
                    animate: true,
                    duration: 2,
                    finish: function() {
                        onAnimate = false;
                    }
                });
            }
        }, 500)
    });
```

添加热力图

热力图功能是将业务数据展示在地图上。热力图是根据由多个地图坐标点及对应点的 value 值生成的，可反映某种特性分布的图形。 示例代码如下：

```
var heat;
map.on('loaded', function() {
    /* 构造热力 */
    heat = new fengmap.FMHeatMap(map);
    /* 添加热力的数据源 */
    heat.addDataSource(createFakeData());
    /* 将热力添加到地图的指定楼层上，添加后立刻显示 */
    heat.addTo(map.getFloor(map.getLevel()));
    setInterval(
        function() {
            var fakeData = createFakeData();
            /* 清除热力图上的所有数据源，注意，如果不清除数据源，则新的数据源会持续向原有数据源进行追加 */
            heat.clearDataSource();
            /* 向热力图追加数据源 */
            heat.addDataSource(fakeData);
            /* 追加数据源后，调用 update() 使其生效。该方法会使用较高资源对热力图进行重绘，请在适当时机调用，避免过于频繁的调用带来性能影响 */
            heat.update();
        }, 300
    )
})

function createFakeData() {
    var heatDataSource = [];
    for (var index = 0; index < 10; index++) {
        var heatPoint = getRandomCoords(map.getBound());
        heatPoint.value = Math.random() * 100;
        heatDataSource.push(heatPoint);
    }
    return heatDataSource;
}
```

添加定位点要素

定位点要素是用户在指定坐标点位置添加自定义的图片要素。图片要素针对地图，一个地图可以添加多个定位要素。一般定位要素用于导航过程中位置的标识。 示例代码如下：

```
    map.on('loaded', function() {
        /* 构造 Marker */
        var locationMarker = new fengmap.FMLocationMarker({
            x: 12619620,
            y: 2621871,
            url: './images/bluedot.png',
            level: map.getLevel(),
            size: 12
        });
        /* 和其他 Marker 不同，LocationMarker 只能向 map 添加，添加后，可通过 Move
To 方法改变楼层 */
        locationMarker.addTo(map);

        setInterval(function() {
                if (!onAnimate) {
                    locationMarker.moveTo({
                        x: getRandomCoords(map.getBound()).x,
                        y: getRandomCoords(map.getBound()).y,
                        /* 要切换楼层时，通过设置 level 属性来改变 locationMarker
的所在楼层 */
                        //level: locationMarker.level + 1,
                        animate: true,
                        duration: 2,
                        finish: function() {
                            onAnimate = false;
                        }
                    });
                }
        }, 500)
        /* locationMarker.remove() */
    });
```

添加动态模型要素

动态模型要素是用户在指定坐标点位置添加自定义的 3D 模型要素。3D 模型要素针对地

图，一个地图可以添加多个模型要素，并且支持播放模型动画。 示例代码如下：

```
map.on('loaded', function() {
        /**
         * 向地图场景中添加模型文件
         */
        var marker = new fengmap.FMDynamicModel({
            url: 'images/duck.glb',
            x: 12619607,
            y: 2621869,
            callback: function() {
                console.log('ok');
            }
        });
        var level = map.getLevel()
        var floor = map.getFloor(level);
        /* 将 Marker 添加到地图的指定楼层上 */
        marker.addTo(floor);

        setInterval(function() {
            if (!onAnimate) {
                /* 改变 Marker 的位置 */
                marker.moveTo({
                    x: getRandomCoords(map.getBound()).x,
                    y: getRandomCoords(map.getBound()).y,
                    animate: true,
                    duration: 2,
                    finish: function() {
                        onAnimate = false;
                    }
                });
            }
        }, 500)
    });
```

添加几何体要素

支持绘制几何体拉伸要素操作，用户可以自定义多边形坐标点集，绘制拉伸几何体。注：几何体是针对楼层的，每一楼层可包含多个几何体。示例代码如下：

```
map.on('loaded', function() {
      addMarkerFunc();
```

```
    });

    function addMarkerFunc() {
      var floor = map.getFloor(map.getLevel());
      createMarker(getRandomRectangle(floor));
    }

  function createMarker(coords) {

       var floor = map.getFloor(map.getLevel());

       marker = new fengmap.FMExtrudeMarker({
         opacity: 0.5,
         color: "#ff0000",
         extrudeHeight: 10,
         height: 5,
         edgeMode: fengmap.EdgeMode.ALL,
         edgeColor: "#ffff00",
         edgeOpacity: 1.0,
         points: coords,
       });

       marker.addTo(floor);
  }

    function getRandomRectangle(floor) {
         let center = floor.center;
         let w = 10;
         let h = 10;
         let p1 = {
             x: center.x - w / 2,
             y: center.y - h / 2
         };
         let p2 = {
             x: center.x + w / 2,
             y: center.y - h / 2
         };
         let p3 = {
             x: center.x + w / 2,
             y: center.y + h / 2
         };
         let p4 = {
             x: center.x - w / 2,
             y: center.y + h / 2
```

```
        };
        let end = {
            x: center.x - w / 2,
            y: center.y - h / 2
        };
        return [p1, p2, p3, p4, end]
    }
```

3.4.4. 在蜂鸟地图上进行要素搜索

蜂鸟地图要素搜索提供多种获取 POI 数据的接口，支持对地图数据基础模型（FMModel）、文字标注（FMLabel）、公共设施(FMFacility)、地板（FMExtent）、3D 模型（FMExternalModel）等查询地图要素的调用接口说明。

要素搜索

根据查询条件搜索查询地图要素。不添加查询条件时，会返回 levels 、type 设置范围内的全部数据内容。示例代码如下：

```
    var searchOption = {
            mapID: options.mapID,
            appName: options.appName,
            key: options.key
        }

    /* 由于 Analyser 和 Map 都是异步构造的函数，两个对象是同时进行初始化的，根据该变量判断是否两个对象是否都构造完成 */
    var readyObjectCount = 0,
        _searchResult;

    /* 查询的代码，必须放在分析器的构造完成的回调中执行 */
    function search() {
        var searchRequest = new fengmap.FMSearchRequest();
        searchRequest.levels = [1,2,3,4,5]
        searchRequest.type = fengmap.FMType.MODEL | fengmap.FMType.LABEL;
```

```
        /* name、ename、keyword 属性，fuzzy 为 true 则进行模糊查询,fuzzy 为 false 则
进行精确查询
        searchRequest.addCondition({
            'name':{'text': "满",fuzzy:true}
            });
        */

        /*searchRequest.addCondition({
            'ename':{'text': "adidas",fuzzy:true}
        });
        */

         /*searchRequest.addCondition({
             'keyword': { 'text': "三", fuzzy: true }
             });
         */

        /* searchRequest.addCondition({
            'FID': '820360010144'
        }); */

        /* searchRequest.addCondition({
            'typeID': [100000, 100100, 200200]
        }); */

        //周边查询，需要设置中心点和半径。
        /* searchRequest.addCondition({
            'circle': {'center':{'x':12619619.607053667,'y':2621894.340},'r
adius': 10 }
        }) */

        //多边形查询，需要设置坐标集合且顶点数不少于 3 个。
        /* searchRequest.addCondition({
            'polygon': [
                        { 'x': 12619619.607053667, 'y': 2621894.340 },
                        { 'x': 12619610.607053667, 'y': 2621893.340 },
                        { 'x': 12619611.607053667, 'y': 2621884.340 },
                        { 'x': 12619620.607053667, 'y': 2621885.340 },
                        { 'x': 12619619.607053667, 'y': 2621894.340 }
                    ]
        }) */

        // 计算包含坐标元素
        /* searchRequest.addCondition({
```

```
            'contain': {
                        'point': { x: 12619616.875869446, y: 2621887.309
82777 }
                    }
        }) */

    analyser.query(searchRequest, (result) => {
            console.log(result);
            _searchResult = result;
            readyObjectCount++;
            if (readyObjectCount == 2)
                markSearchResult();
            });
        }

    /* 构造初始化查询分析器，在回调中执行 Search 方法 */
    var analyser = new fengmap.FMSearchAnalyser(searchOption, function() {
        search();
    });

    /* 方法定义，在地图上把搜索结果标记出来 */
    function markSearchResult() {
        _searchResult.forEach(feature => {
            if (feature.type == fengmap.FMType.MODEL && feature.name) {
                var marker = new fengmap.FMImageMarker({
                    url: './images/red.png',
                    x: feature.center.x,
                    y: feature.center.y,
                    anchor: fengmap.FMMarkerAnchor.BOTTOM
                });
                var level = map.getLevel()
                var floor = map.getFloor(level);
                marker.addTo(floor);
                updateUI(feature.FID + ' ' + feature.name);
            }
        });
    }

    var map;
    map = new fengmap.FMMap(options);
    map.on('loaded', function() {
        readyObjectCount++;
        if (readyObjectCount == 2)
        markSearchResult();
```

```
});
```

周边搜索

周边搜索方法需提供查询范围中心点、查询范围半径等参数。示例代码如下：

```
var searchOption = {
          mapID: options.mapID,
          appName: options.appName,
          key: options.key
      }

var analyser = new fengmap.FMSearchAnalyser(searchOption, function() {
    var searchRequest = new fengmap.FMSearchRequest();
    searchRequest.levels = [2]
    searchRequest.type = fengmap.FMType.MODEL;
    searchRequest.addCondition({
        'circle': {
            'center': {
                'x': 12619620,
                'y': 2621864
            },
            'radius': 15
        }
    });

    analyser.query(searchRequest, (result) => {
        _searchResult = result;
        readyObjectCount++;
        if (readyObjectCount == 2) markSearchResult();
            });
        })

function markSearchResult() {
    console.log(_searchResult);
    //console.log(map.getNodes(_searchResult));
    _searchResult.forEach(feature => {
        if (feature.type == fengmap.FMType.MODEL && feature.name) {
            /* var marker = new fengmap.FMImageMarker({
                url: './images/red.png',
                x: feature.center.x,
                y: feature.center.y,
```

```
                anchor: fengmap.FMMarkerAnchor.BOTTOM
            });
            var level = map.getLevel()
            var floor = map.getFloor(level);
            marker.addTo(floor); */
            updateUI(feature.FID + ' ' + feature.name);
        }
    });
}

/* 没什么用的方法，纯粹为了更新界面的提示 */
function updateUI(msg) {
    var infoDiv = document.body.getElementsByClassName('info')[0];
    var li = document.createElement('li');
    li.innerText = msg;
    infoDiv.appendChild(li);
}

locationMarker.addTo(map);

readyObjectCount++;
if (readyObjectCount == 2)
    markSearchResult();
});
```

几何区域查询

查询几何区域，示例代码如下：

```
 /* 任意多边形 */
    var polygonOption = {
        height: 1,
        points: [{
            x: 12619617.362326605,
            y: 2621882.6430324786
        }, {
            x: 12619636.017352777,
            y: 2621884.395266865
        }, {
            x: 12619618.360931246,
            y: 2621840.2641846496
        }, {
```

```
            x: 12619602.616454065,
            y: 2621844.9678212376
        }],

    }

    var readyObjectCount = 0;
    var _searchResult;

    function markSearchResult(result) {
        result.forEach(feature => {
            if (feature.type == fengmap.FMType.MODEL && feature.name) {
                var marker = new fengmap.FMImageMarker({
                    url: './images/red.png',
                    x: feature.center.x,
                    y: feature.center.y,
                    anchor: fengmap.FMMarkerAnchor.BOTTOM
                });
                var level = map.getLevel()
                var floor = map.getFloor(level);
                marker.addTo(floor);
                updateUI(feature.FID + ' ' + feature.name);
            }
        });
    };
    var analyser, request;

    function initRequest() {
        var searchRequest = new fengmap.FMSearchRequest();
        searchRequest.levels = [2]
        searchRequest.type = fengmap.FMType.MODEL;
        searchRequest.addCondition({
            polygon: polygonOption.points
        });
        return searchRequest;
    }

    /* 初始化分析器，这里初始化分析器使用 Map 实例进行构造，因此 分析器必须在 Map
的 Loaded 事件回调中使用，如果希望并行使用，请采用 option 的配置方案。 */
    function initAnalyser() {
        analyser = new fengmap.FMSearchAnalyser({
            map: map
        }, function() {
            request = initRequest();
```

```
            analyser.query(request, (result) => {
                markSearchResult(result);
            });
        });
    }



    map = new fengmap.FMMap(options);
    map.on('loaded', function() {
        initAnalyser();
        var polygon = new fengmap.FMPolygonMarker(polygonOption);
        polygon.addTo(map.getFloor(2));
    });

    /* 没什么用的方法，纯粹为了更新界面的提示 */
    function updateUI(msg) {
        var infoDiv = document.body.getElementsByClassName('info')[0];
        var li = document.createElement('li');
        li.innerText = msg;
        infoDiv.appendChild(li);
    }
```

3.4.5. 在蜂鸟地图上进行路线规划

蜂鸟地图常用于各类信息的展示及高精度的定位导航应用。路径规划功能支持在有导航数据的地图中，根据起点和终点规划最优路径的功能，根据路径规划结果，进行第一人称或第三人称的模拟以及真实导航，返回路径经过的坐标点集，同时可根据结果集获取路径描述等信息。

路径规划分析

设置起、终点，在地图渲染后即 loaded 回调后，进行路径分析，分析成功，获取路径描述信息，并返回路径经过的坐标点集。示例代码如下：

```
    var route, count = 0;
```

```
var analyser = new fengmap.FMNaviAnalyser(options, () => {
      let naviRequest = {
          start: {
              level: 1,
              x: 12619612.556779679,
              y: 2621854.2805
          },
          dest: {
              level: 5,
              x: 12619608.229031052,
              y: 2621867.9186
          },
          mode: fengmap.FMNaviMode.MODULE_SHORTEST,
          priority: fengmap.FMNaviPriority.PRIORITY_DEFAULT
      }
      analyser.route(naviRequest, (result) => {
          route = result;
          count++;
          if (count === 2) {
              cb();
          }
      });
  })

  map.on('loaded', function() {
      count++;
      if (count === 2) {
          cb();
      }
  });

  var segments;
  var line;

  function cb() {
      console.log(route);
      segments = [];
      route.subs.forEach(leg => {
          if (leg.levels[0] === leg.levels[1]) {
              let segment = new fengmap.FMSegment();
              leg.waypoint.points.forEach(point => {
                  point.z = 3
              });
              segment.points = leg.waypoint.points;
```

```
                segment.level = leg.levels[0];
                segments.push(segment);
            }
        });
        console.log(segments);
        line = new fengmap.FMLineMarker({
            segments: segments
        })
        line.addTo(map)
    }
```

3.4.6. 在服务端使用路径计算和搜索

蜂鸟地图提供了一种全新的基于 Node.js 服务端的解决方案，地图与计算内核进行了解耦，用户可以根据自己实际的业务场景，不仅可以在前端通过本地 JavaScript 代码调用路径计算，或在 Web Worker 中调用路径计算，同时还可以在服务端以 Node.js 的方式将路径计算包装成为 restful 服务进行使用。服务端的调用方式能够极大的降低前端计算的压力，使用户能将路径计算和搜索服务部署在服务器上，由服务器完成耗时的计算工作，从而提升应用程序的用户体验和地图的交互体验。

路径计算

服务端路径计算， 示例代码如下：

```
var express = require('express');
var fengmap = require('./bin/fengmap.analyser.min');

var path = require('path');

var app = express();

var options = {

    mapID: '1321274646113083394',
    //必要，地图应用名称，通过蜂鸟云后台创建
```

```
        appName: '蜂鸟研发 SDK_2_0',
        //必要，地图应用密钥，通过蜂鸟云后台获取
        key: '57c7f309aca507497d028a9c00207cf8',
        //必要，node 环境下必须设置该参数
        mapURL: path.resolve(__dirname) + '/data/',
    }

    var analyser = new fengmap.FMNaviAnalyser(options, () => {

        app.listen(8989, '0.0.0.0', () => {
            console.log('route server listening on port 8989');
        });
    }, (error) => {
        console.log(error);
    });

    app.all('*', function(req, res, next) {
        res.header('Access-Control-Allow-Origin', '*');
        res.header('Access-Control-Allow-Headers', 'Content-Type, Content-
Length, Authorization, Accept, X-Requested-With, yourHeaderFeild');
        res.header('Access-Control-Allow-Methods', 'PUT, POST, GET, DEvarE,
 OPTIONS');
        if (req.method === 'OPTIONS') {
            res.send(200);
        } else {
            next();
        }
    });

    app.get('/route', function(req, res) {
        var query = req.query;
        var start = query.start.split(',');
        var dest = query.dest.split(',');
        var request = {
            start: {
                x: parseFloat(start[0]),
                y: parseFloat(start[1]),
                level: parseInt(start[2]),
            },
            dest: {
                x: parseFloat(dest[0]),
                y: parseFloat(dest[1]),
                level: parseInt(dest[2]),
            },
```

```
        mode: fengmap.FMNaviMode.MODULE_SHORTEST,
        priority: fengmap.FMNaviPriority.PRIORITY_DEFAULT
    }
    analyser.route(request, (result) => {
        res.send(result);
    }, () => {
        res.send('route failed');
    });

});
```

搜索查询

服务端搜索查询， 示例代码如下：

```
let express = require('express');
let fengmap = require('./bin/fengmap.analyser.min');

let path = require('path');

let app = express();


let options = {

    mapID: '1321274646113083394',
    //必要，地图应用名称，通过蜂鸟云后台创建
    appName: '蜂鸟研发 SDK_2_0',
    //必要，地图应用密钥，通过蜂鸟云后台获取
    key: '57c7f309aca507497d028a9c00207cf8',
    //必要，node 环境下必须设置该参数
    mapURL: path.resolve(__dirname) + '/data/',
}

let analyser = new fengmap.FMSearchAnalyser(options, () => {

    app.listen(8990, '0.0.0.0', () => {

        console.log('search server listening on port 8990');
```

```
        });

    }, (error) => {

        console.log(error);
    });

    app.all('*', function(req, res, next) {
        res.header('Access-Control-Allow-Origin', '*');
        res.header('Access-Control-Allow-Headers', 'Content-Type, Content-
Length, Authorization, Accept, X-Requested-With, yourHeaderFeild');
        res.header('Access-Control-Allow-Methods', 'PUT, POST, GET, DELETE,
 OPTIONS');
        if (req.method === 'OPTIONS') {
            res.send(200);
        } else {
            next();
        }
    });

    app.get('/search', function(req, res) {

        let request = new fengmap.FMSearchRequest();

        let query = req.query;

        // request.levels = [1, 2];//[1, 2, 3, 4, 5]
        request.type = fengmap.FMType.MODEL | fengmap.FMType.LABEL;


        request.addCondition(query);

        analyser.query(request, (result) => {

            res.send(result);

        }, () => {
            console.log('search failed');
        });
    });
```

3.4.7. 将室内定位的坐标转换为蜂鸟地图坐标

蜂鸟地图提供了经纬度和地图坐标之间的转换方法，蜂鸟地图为墨卡托坐标系米制坐标。

Fengmap 地图坐标系为全球坐系统，全球坐标系统下地理元素具有唯一的空间位置、真实的地理方向，多元素间有明确的相对位置关系。Fengmap 的标注物添加，更新需要使用 Fangmap 地图坐标，其他坐标系坐标需要进行正确的转化后才能正确的在地图上呈现。

坐标转换应用场景

1) 定位位置地图展示

Fengmap 地用应用场景较多，主要可以分为:园区、景区类室内外一体化场景、单一室内场景两种、一体化场景中定位技术呈现多样性:室外 GPS、室内 WIFI/iBeacon。单一室内主要使用 UWB/WIFI/iBeacon 等成熟室内定位技术较多。室外 GPS 定位坐标系统为 WGS84 星标系，可使用计算转化公式直接进行 Fengmap 主标转换。室内定位星标系多为相对坐标系。定位计算数据源一般为图片、CAD，返回坐标为对应定位原点的相对坐标，可理解转换原理后使用转换示例进行坐标转换。

2) 业务系统地图升级，老旧业务 POI 需要再升级后的蜂鸟地图上展示：

3D 室内地图应用发展之前，一些包含室内地图的项目中，地图部分往往使用图片，CAD 平面图较多。业务 POI 的位置信息均使用相对于数据原点的相对坐标存储。使用 Fengmap 三维地图升级系统后，老旧业务 POI 的位置信息不能直接使用，需要进行坐标转换，此种情况与定位场景中相对坐标转换方法相同。

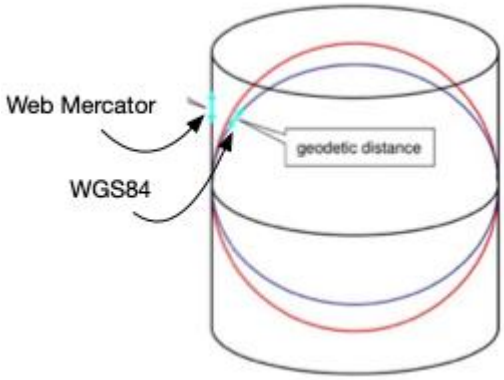
综上，我们主要说明两种情况下的 Fengmap 地图坐标转换原理与方法：①WGS84 坐标到 Fengmap 坐标转换；②相对坐标到 Fengmap 坐标转换。

坐标转换方法

1）WGS84 坐标到 Fengmap 坐标转换

WGS84 一种国际上采用的地心坐标系，坐标值为经纬度。

Fengmap 地图坐标系统为 WebMercator 投影坐标系，基准面为 WGS84，坐标单位为 m。



通用转化方法如下:

mercator 代指转化后的 WebMercator 坐标。lonlat 代指经纬度坐标，.lon 代指经度.lat 代指纬度。

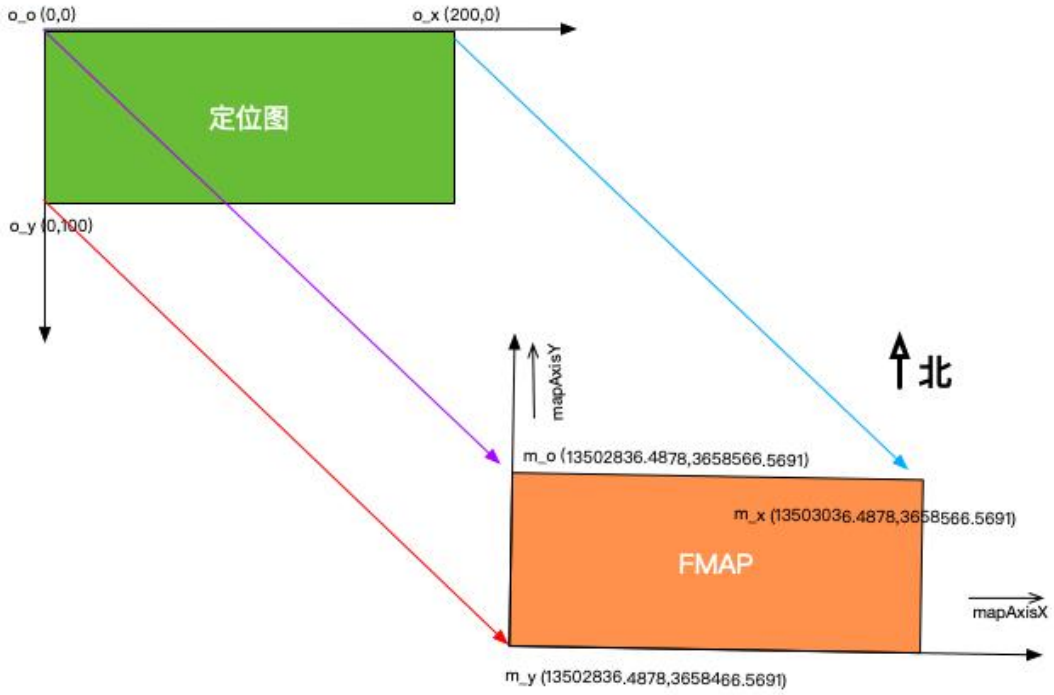
mercator.x=lonlat.lon*20037508.34/180;

mercator.y=(log(tan((90+lonlat.lat)*PI/360))/(PI/180))*20037508.34/180;

WGS84(或 bd09ll)到百度墨卡托(bd09mc)的转化请参考百度地图开放平台官网进行转换。参考:https://lbsyun.baidu.com/jsdemo.htm#TranslateoriTobd

2）相对坐标到 Fengmap 坐标转换

局部坐标的转换，主要处理好:原点坐标对应、X 轴终点坐标对应、Y 轴终点坐标对应，即可正确转换。



局部坐标的转换，主要处理好:原点坐标对应、X/Y 轴地图方向、坐标值比例三个对应关系，

即可正确转换。一般室内定位系统，使用图片作为定位计算数据，左上点为原点(0,0)，Y 轴正方向与 Fengmap 坐标系相反。转换计算的基本过程说明如下，不关心计算过程和原理可以只阅读下面的第 1、4 点;

a. 建立原始坐标系与目标坐标系的对应关系，原点，X、Y 轴终点的定位坐标与 Fengmap 坐标的映射。坐标转换 demo 中封装了坐标转换类 CoordTransformer。调用 init 方法建立映射:

```
    //创建转换器
    var trasformer = new CoordTransformer();

    //原始坐标系参数 三个角点
    var origonParas =[];
    origonParas[0]={'x':0,'y':0};//坐标角点 原点
    origonParas[1]={'x':200,'y':0};//X 轴终点原始坐标
    origonParas[2]={'x':0,'y':100};//Y 轴终点原始坐标

    //目标坐标系参数 三个角点
    var targetParas=[];
    targetParas[0]={'x':13502836.4878,'y':3658566.5691};//定位原点对应的地图坐标点
    targetParas[1]={'x':13503036.4878,'y':3658566.5691};//定位 X 轴终点对应的地图坐标
    targetParas[2]={'x':13502836.4878,'y':3658466.5691};//定位 Y 轴终点对应的地图坐标

    //转换器初始化
    trasformer.init(origonParas,targetParas);
```

b. 根据映射关系计算原始坐标轴对应的地理坐标轴 mapAxisX，mapAxisY 向量。

c. 计算原始坐标 X、Y 变化范围，Fengmap 坐标 X、Y 变化范围确定变化比例。

d. Fengmap 坐标轴向量单位化。

b、c、d 三个步骤在坐标转换类内部完成

```
    _oriOrigion =origonParas[0];//原始坐标原点
    _oriAxisX ={'x':origonParas[1].x - origonParas[0].x,'y':origonParas[1].y - origonParas[0].y};//原始坐标系 X 轴向量
    _oriAxisY ={'x':origonParas[2].x - origonParas[0].x,'y':origonParas[2].y - origonParas[0].y);//原始坐标系 Y 轴向量
```

```
    _oriRange ={'x':this.getVectorLen(_oriAxisX),'y':this.getVectorLen(_o
riAxisY)};//原始坐标系 XY 轴长度(模)
```

```
    _tarOrigion =targetParas[0];//目标坐标原点
    _tarAxisX ={'x':targetParas[1].x - targetParas[0].x,'y':targetParas[1].
y - targetParas[0].y};//目标坐标系 X 轴向量
    _tarAxisY ={'x':targetParas[2].x - targetParas[0].x,'y':targetParas[2].
y - targetParas[0].y);//目标坐标系 X 轴向量
    _tarRange ={'x':this.getVectorLen(_tarAxisX),'y':this.getVectorLen(_t
arAxisY)};//目标坐标系 X 轴向量
```

```
    //向量单位化
    _oriAxisX.x /=_oriRange.x; _oriAxisX.y /=_oriRange.x;
    _oriAxisY.x /=_oriRange.y; _oriAxisY.y /=_oriRange.y;
```

```
    _tarAxisX.x /=_tarRange.x; _tarAxisX.y /=_tarRange.x;
    _tarAxisY.x /=_tarRange.y; _tarAxisY.y /=_tarRange.y;
```

e. 坐标转换计算。调用坐标转换类的转换方法传入原始坐标，返回值为转换后的目标坐标。

```
    var targetCoord = trasformer.transform(origon):
```

f. 转换类内部计算方法说明:

```
    var offset = {'x':origon.x - _ori0rigion.x,'y':origon.y - _ori0rigion.
y};//原点坐标的偏移
    var offsetX = _oriAxisX.x*offset.x + _oriAxisX.y*offset.y;//原始坐标 X 轴
偏移向量
    var offsetY = _oriAxisY.x*offset.x + _oriAxisY.y*offset.y;//原始坐标 Y 轴
偏移向量
    var offstRatio = {'x':offsetX/_oriRange.x,'y':offsetY/_oriRange.y};//
原始坐标系 X、Y 轴长度比率
    var tar0ffset = {'x':offstRatio.x*_tarRange.x,'y':offstRatio.y*_tarRan
ge.y};//目标坐标系 X、Y 轴长度
    var tarCoord =
     {'x':_tar0rigion.x + _tarAxisX.x*tar0ffset.x+_tarAxisY.x*tar0ffset.y,
'y':_tar0rigion.y + tarAxisx.y*taroffset.x+_tarAxisY.y*taroffset.y);
```

公式说明:转换后坐标 = 目标坐标系原点坐标 + 目标坐标系 X 轴单位向量*目标坐标系 X 变化长度 + 目标坐标系 Y 轴单位向量*目标坐标系 Y 变化长度。

```
retum tarCoord;//返回目标坐标系坐标
```

注:原始坐标对应的地图坐标映射建立所需的地图坐标有两种获取方式① 原始范围小于等于地图制作范围，可以使用 SDK 提供坐标拾取接口获取所需地图坐标。②原始范围大于地图制作范围，需要专业数据校准工具来完成坐标获取工作。

3.5. 室内空间位置服务

3.5.1. 定位信标的管理及预部署

蜂鸟定位信标管理是通过“项目管理”的方式，对定位信标进行部署和管理。相关项目管理人员，可以在蜂鸟云平台上，对项目进行创建、编辑、删除等常规操作。在创建项目时，需要关联一张已经发布过的蜂鸟云地图，后续所有的定位相关部署操作，都会在该电子图纸上进行部署操作。通过项目管理中的“项目部署”入口，可以管理当前项目中的所有信标并对信标进行预部署。

预部署是指在绘制地图时标注并导出希望在相关位置进行点位部署而选中的坐标点，从而方便在移动端部署时，无需选择部署位置，直接在相关预部署点位绑定信标即可完成信标部署。

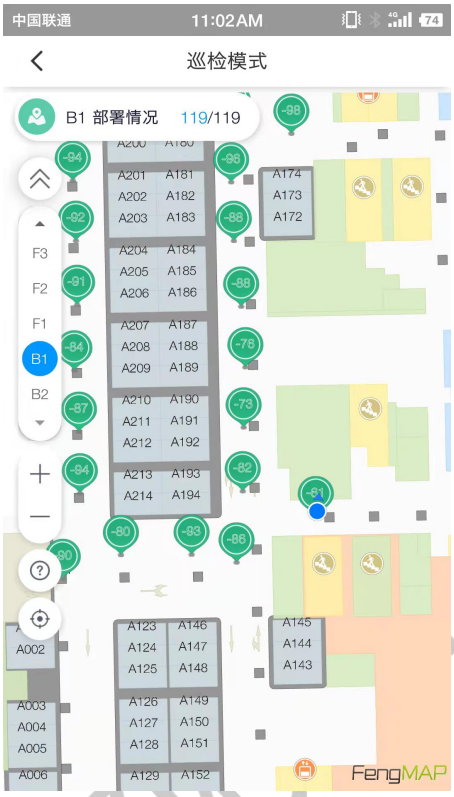
在蜂鸟云子账号管理模块，可以为项目添加指定施工或巡检人员，进行项目的线下部署或巡检。

3.5.2. 移动端信标部署及巡检

蜂鸟定位项目移动端提供了两个主要的功能，线下部署和巡检。

信标部署是指，在对应的项目当中的相关位置点击后，通过扫码识别信标上的二维码或条形码，或者手工输入的方式，把位置和信标进行绑定的操作。同时，我们也提供了在无网络或弱网络唤醒下，离线部署的能力。移动端本身也支持了多人协同部署端能力，在大型或多层项目中，多个施工人员可以同时进行项目部署。

巡检是指，在项目部署完一个阶段或项目完毕后，通过巡检功能检测项目定位情况、信标信号强度以及对应信标电池电量。



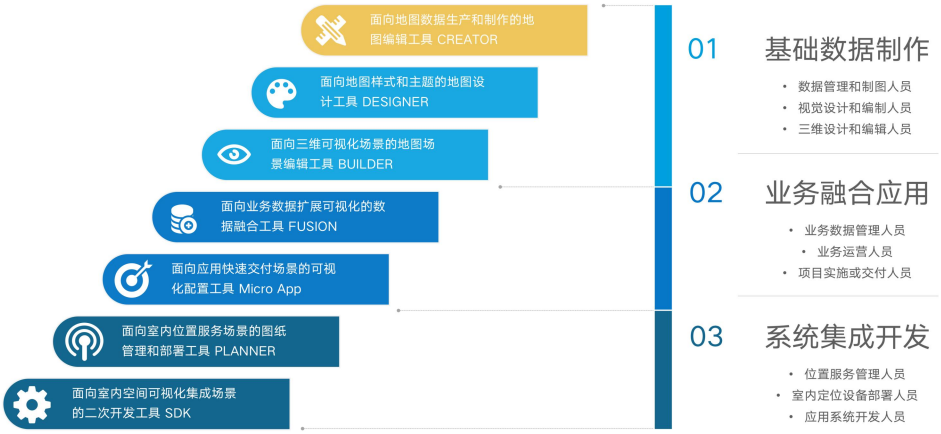
3.5.3. 在您的应用程序中使用室内定位服务

您可以通过在您的应用程序或小程序中整合蜂鸟云提供的定位 SDK，来实现室内实时定位的能力。通过实时定位能力，可以满足在您的业务场景中实时监控、轨迹回放、区域报警、室内导航、精准营销、反向寻车和数据分析等多种需求。帮助您实现精准导航、精准营销、进行客户大数据分析，为您的业务提供强有力的技术支撑。

4. 产品体系

4.1. 产品架构

平台组成



4.2. 地图编辑器（英文：FengMap Creator）

蜂鸟云地图编辑器 V2.0 是蜂鸟视图研发的自主绘图编辑器，是 V1.0 的全新升级。该编辑器面向企业和个人用户，用户可以简单、高效的绘制地图上的点、线、面等空间元素，并可以自行管理所拥有的私有地图数据，并对全量的地图数据进行修改和编辑。

4.3. 主题编辑器（英文：FengMap Designer）

Designer 是蜂鸟视图推出的一款自主配置地图主题样式的工具。主题编辑器中内置多种模板，通过选择模板，可一键为地图配置主题。在详细配置中，提供了按图层、按类型和按楼层配置方式，满足用户个性化定制需求。

4.4. 数据融合编辑器（英文：FengMap Fusion）

通过使用 Fusion，您可以将来自其他数据源的数据引入到您的蜂鸟云账号中，还可以将其与您已发布的地图数据进行空间关联，成为地图数据的扩展属性；或参考地图数据给数据

源赋予地理坐标，使其成为可叠加于地图的空间标注数据，从而实现”数图联动”。

4.5. 微应用（英文：FengMap MicroApp）

MicroApp 是蜂鸟视图推出的一款依托于低代码技术的高效、高性能的室内地图导航应用生成工具。微程序将繁琐的底层架构和基础配置抽象化为图形界面，通过行业化模板、拖放式组件和可视化配置快速构建应用。免去了代码编写工作，让开发者能更加专注于业务场景的实现，并进一步提升交付速度。

4.6. 室内位置规划服务（英文：FengMap Planner）

Planner 打通了制图到定位完整的工具链，让定位设备部署变得更加便捷、精准。 针对定位厂商和软件开发商，同时，Planner 提供了开放的定位接入服务，提供了标准化的产品，让用户可以便捷的接入完整的空间位置服务，扩展原有的业务系统。

4.7. 室内地图引擎（英文：FengMap SDK）

FengMap Engine 是由蜂鸟视图开发的，基于 OpenGL 和 WebGL 技术，让用户轻松构建诸如空间信息管理、建筑信息管理及三维空间数据可视化、导航等类型应用的多平台、综合型地图引擎。通过自主研发的 FMKernel 引擎，可以让开发者轻松构建运行在网页端、移动端等多平台共享地图应用程序开发。

北京市朝阳区安定路39号长新大厦14层
400-680-0432
www.fengmap.com
marcom@fengmap.com