天津大学

《计算机图形学》实验报告

学　　号 3022244277

姓　　名 丁成功

学　　院 智能与计算学部

专　　业 计算机科学与技术

年　　级 2022级

任课教师 徐庆、刘世光

2024年　12　月　15　日

1　场景搭建

列出使用的第三方素材，行人和车辆动画实现（可选）等场景搭建细节

（1）场景的第三方素材

Demo City By Versatile Studio (Mobile Friendly) | 3D 都市 | Unity Asset Store

场景简要介绍：

该包包含一个即用型场景，其中包含夜城的一小部分和一条长高速公路。

所有照明均已放置、配置和烘焙。

还内置了遮挡剔除以进行优化。

照明、反射和材料旨在模拟潮湿天气。

其中本次实验主要使用了其中的夜晚的城区部分。

（2）人物的动画实现

静态人物共有 48 个，均匀分布在城区的各个部分，有各个年龄段的人物，其中给每个静态人物都设置了动画，使得画面更加自然、生动。

图表 1 部分静态人物的动作

动态行走人物有 39 个，利用 NaveMesh 功能实现人物的导航功能，每个人物均添加了 Animator ，保证了行走过程中均有行走动画，其中动态行走人物到达目的地以后，会短暂的停歇，为此添加了人物的动画切换，利用脚本和 Animator 的 bool 参数判断人物是否行走，保证了动作切换过渡过程的自然。

（3）车辆的动画实现

车辆的动画实现难点主要在于规划路线，使得车辆按照既定的路线行进，并添加了碰撞检测，在速度较低的情况下可以躲避障碍物。

图表 2 部分车辆的规划路线

a. 首先创建了 path.cs 脚本用于在场景视图中可视化路径。

通过绘制Gizmo来展示路径的节点和连接线

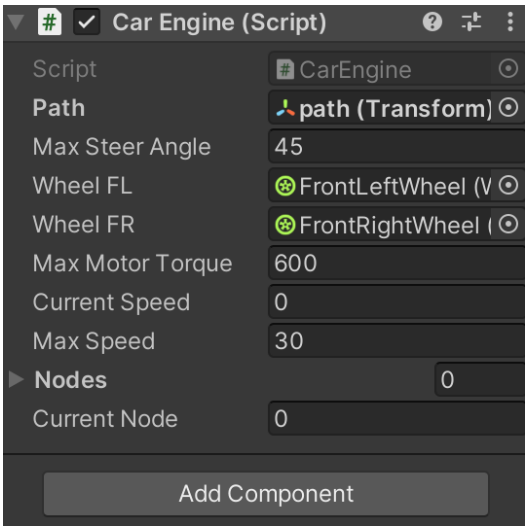
关键特点：

使用OnDrawGizmos()方法在Unity编辑器中绘制路径

自动获取当前GameObject下的所有子节点作为路径节点

支持创建闭合路径（首尾相连）

b. 为了车辆可以物理化的移动，创建了 CarEngine.cs 脚本，



图表 3 CarEngine 有关属性

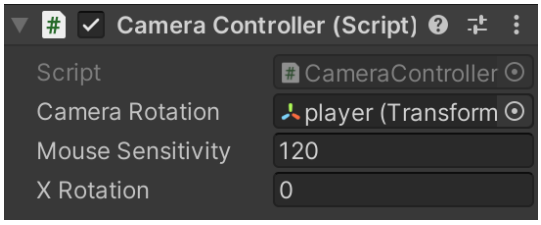
这个脚本具有以下功能：

路径跟随：自动沿预定义的路径行驶，可以在Unity编辑器中通过创建路径节点来自定义行驶路线，到达最后一个节点后自动返回起点（循环路径）。
驾驶控制：动态计算转向角度，根据当前速度控制电机扭矩，限制最大速度。

2 操控方式实现

操控方式主要是以一个小人作为player，小人是以第一人称视角，键盘可以控制移动方向，鼠标控制视角，键盘和鼠标共同交互决定了人物的移动。
实现的方式：

a. 小人的视角设计，给小人添加了一个camera，位置大概在小人的眼睛的位置，并且给该camera添加了一个脚本cameraController，该脚本的功能是：



视角控制系统：

实现第一人称视角的鼠标控制，支持水平（左右）和垂直（上下）视角旋转，关键特性，鼠标灵敏度可调节，垂直视角角度限制在-90°到90°，使用Time.deltaTime保证旋转速度与帧率无关

旋转机制：

水平旋转：绕Y轴旋转整个相机对象

垂直旋转：绕本地X轴旋转相机，使用Quaternion.Euler进行角度转换

输入处理：使用Input.GetAxis获取鼠标移动量，乘以灵敏度和时间增量调整旋转速度。

b. Player的移动方式：

给player添加了一个playerMove脚本，用于实现利用键盘控制小人的移动。此外，也给小人添加了walkshift脚本，主要用于player行走和静止时的动画切换，让小人也有相应的动画。

3　交互方式实现

鼠标移动能改变视角朝向，键盘操作小人行走。该两个关于小人的交互已经在操控方式中进行了详细说明。接下来进行AR窗口有关的详细说明。

切换打开或关闭AR窗口和切换展示AR窗口的不同引导式：

对于AR窗口，主要利用和键盘的交互实现：

a. 对前进的路线进行引导线的提示，方便用户跟随前进（如连贯导航线等）

   前进路线的交互使用了N（代表navigation），按下N则会显示前进路线的引导线的提示，使得更方便player跟进。该控制方式的实现是在line.cs脚本里面添加了按键控制的代码，line.cs是主要用来绘画连贯导航线的。

b. 每隔一段距离在道路上（尤其是路口）提示前进方向（如指示路标等）

c. 在AR窗口上提示当前前进方向（如指南针的形式等）

   使用了UI的canvas功能，当按下C（代表compass），会显示指南针功能；如果指南针已经显示，按下C，指南针AR窗口则会消失。

   该交互的控制使用了脚本compasscontrol.cs，用于控制指南针的实现。

4　引导方式实现（重点）

详述每一种实现的引导方式，思考不同引导方式之间的优劣

a. 对前进的路线进行引导线的提示，方便用户跟随前进（如连贯导航线等）

   首先给player添加了一个line Renerer的组件，同时创建了一个材质球material，材质球主要作用是将引导的方式表示为箭头的形式，更加美观。

图表 4 路线引导

此外，还添加了line.cs脚本，实现了一个使用 NavMeshAgent 和 LineRenderer 的功能，主要用于绘制一个可见路径，并能够通过按下 "N" 键来切换路径的可见性。以下是代码实现的功能和逻辑概要：

主要功能

路径计算和可视化：

该脚本使用 NavMeshAgent 来计算从当前对象到目标位置的导航路径。

路径通过 LineRenderer 可视化，LineRenderer 会根据路径的每个角点绘制一条线。

路径可见性切换：

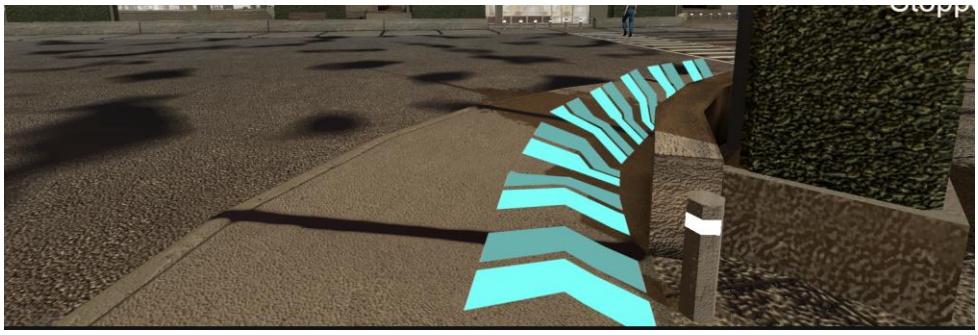
通过按下 "N" 键来切换路径是否可见。如果路径当前可见，按下 "N" 键会使其隐藏；如果路径隐藏，按下 "N" 键则会显示路径。

这是通过 isPathVisible 这个布尔值和 LineRenderer.enabled 属性来控制的

路径绘制限制：

路径的可视化是有距离限制的，最大可见路径长度由 visiblePathDistance 控制。如果路径的某一段距离超出了可见范围，只有路径的前部分会被绘制，剩余的部分会被截断。路径点的位置会被调整高度 (pathHeightOffset)，确保路径不会被遮挡。

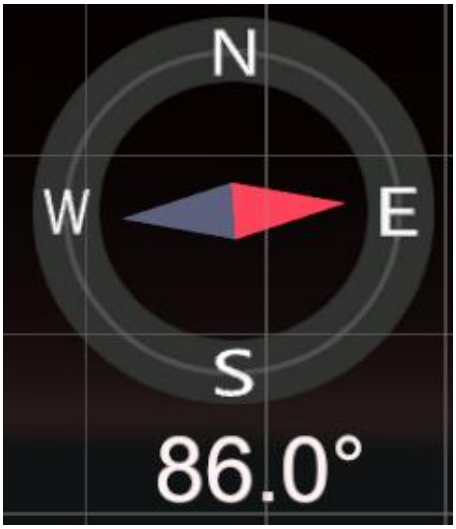
b. 每隔一段距离在道路上（尤其是路口）提示前进方向（如指示路标等）



图表 5 在拐弯处添加了提示路标

具体实现方法是在 line.cs 脚本中根据导航判断拐点，利用一系列拐点生成一个曲线，利用曲线line renderer渲染实现路口的路标。逻辑上和第一种引导方式没有本质区别，主要是判断路口的逻辑。运行过程中进行判断，如果是路口，触发脚本，根据脚本渲染指示牌。

c. 在AR窗口上提示当前前进方向（如指南针的形式等）



图表 6 指南针的实现

指南针的实现依靠了canvas，实现了一个基本的指南针系统导航，显示玩家的朝向、移动方向以及角度。该指南针通过旋转图像、显示角度和箭头指示器来提供方向感知。

主要功能：

1. 计算玩家的移动方向和角度

根据玩家的 Transform（位置）计算移动方向，并将其转化为相对于世界坐标系正北方向的角度。

2. 更新UI显示

指南针旋转：根据玩家的移动方向，旋转指南针图像，使其指向玩家的移动方向。

方向箭头颜色：根据玩家是否移动，改变方向箭头的颜色（活动时为 activeColor，停止时为 inactiveColor）。

角度文本显示：显示玩家当前的角度，或者当玩家停止时显示 "Stopped"。

3. 动态更新和优化

如果玩家移动的距离超过设定的阈值 (movementThreshold)，则更新方向角度；否则，不更新。通过一个阈值来防止在极小的位移时频繁更新，提高性能。

4. 界面可定制

是否显示角度文本：通过 showDegreeText 控制是否在屏幕上显示角度

旋转模式：通过 useRotationMode 控制是否使用旋转模式（让指南针图像本身旋转指示方向）。

Compass.cs代码实现详细说明

1. 初始化组件

玩家位置（playerTransform）：如果没有手动指定 playerTransform（玩家的位置），则默认使用当前物体（通常是脚本附加的物体）的 Transform 作为玩家的位置。

初始化UI元素：检查 degreeText 是否存在，并根据 showDegreeText 显示或隐藏角度文本。

2. 更新逻辑

玩家移动方向：

通过计算玩家当前的位置与上一个位置之间的差异，获得玩家的移动方向。movementDirection 计算玩家的朝向方向（单位向量），然后通过 Mathf.Atan2 获取该方向相对于正北方向的角度。使用 Mathf.Rad2Deg 将角度从弧度转换为度。

3. 角度标准化：

确保计算出的角度在 0-360 度之间。如果角度为负，则加上 360 度，确保角度始终为正。

4. 判断玩家是否在移动：

通过 Vector3.Distance 计算玩家当前位置与上一个位置之间的距离，若该距离超过 movementThreshold，则认为玩家正在移动。如果玩家正在移动，则计算并更新角度信息，否则显示为停止状态。

5. UI更新

指南针旋转：如果启用旋转模式（useRotationMode），指南针图像会根据计算出的角度进行旋转。通过 compassImage.rotation = Quaternion.Euler(0, 0, -angle) 来旋转图像。箭头颜色：箭头指示器的颜色会根据玩家是否在移动而改变。如果移动则设置为 activeColor（活动颜色），否则为 inactiveColor（非活动颜色）。角度文本：如果设置了显示角度文本（showDegreeText），并且玩家正在移动，则更新文本显示当前的角度。如果停止移动，则显示 "Stopped"。

思考三种引导方式的优劣：

1. 前进路线的引导线提示（如连贯导航线）

优点：

直观易懂：用户可以清晰地看到前进路线，尤其在复杂的环境中，可以帮助用户知道正确的行进方向。

连续性：沿着引导线前进时，可以避免用户迷失，特别是在没有明确路标的情况下。

清晰的视觉反馈：通过不同颜色或风格的引导线，可以明确标示出正确的路线，适合新手或需要精确导航的用户。

缺点：

空间占用：引导线可能会占用一定的屏幕或现实世界的视野，影响用户的整体体验，特别是在狭窄或复杂的环境中。

动态变化：当环境发生变化（如障碍物或道路突变）时，引导线可能需要实时更新，增加系统复杂度和计算负担。

适应性差：对于用户的个性化需求，如偏好某些特定路径或道路，它不够灵活。

2. 道路上（特别是路口）距离提醒（如指示路标、路口标识）

优点：

时效性强：用户能在即将到达转弯或路口时提前收到提示，有足够的时间做出反应。

精确性高：通过显示特定的地理标识或指示，能够非常精确地告诉用户需要改变方向，减少误判。

减少认知负担：用户只需要关注标识，不需要时刻注视屏幕或其他视觉线索，能够减少注意力的分散。

缺点：

适用性受限：如果环境中没有显著的地标或路口，或是用户处于一个较为广阔的区域（如广场、大型公园等），这种方式可能不太有效。

视觉障碍：在某些场景中（如雨雪天气、夜间），路标和指示可能不太明显或难以辨认。

3. AR窗口中的前进方向提示（如指南针或动态箭头）

优点：

实时性强：可以在用户视野中实时显示当前的前进方向，且不需要用户频繁切换视线或焦点，增强了现实感和沉浸感。

方向感强：尤其是在开放或不规则环境中，指南针或动态箭头可以帮助用户迅速明确方向，避免迷失。

空间感更好：与现实世界的场景结合，玩家能够在虚拟信息与实际场景之间获得流畅过渡，提高导航的准确性和亲和力。

缺点：

视觉干扰：如果提示过多或显示方式不合适，可能会导致用户的视线分散，增加视觉疲劳。

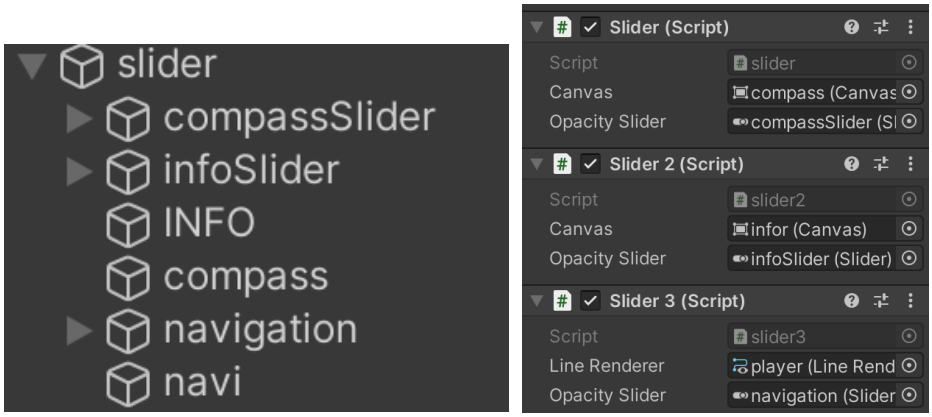
不够直观，引导的方式可能会让玩家难以适应。

5 配置方式实现

1. 透明度的配置实现

按下P键可以实现透明度的设置。

图表 7 透明度的配置



图表 8 利用canvas的slider组件

利用canvas的slider组件结合脚本控制每一个slider，实现透明度的设置。

具体脚本逻辑：

1. 变量声明

canvas: 用来调节透明度的 Canvas。

opacitySlider: 控制透明度的 Slider 控件。

canvasImage: 用来修改透明度的 Image 组件，通常该组件是 Canvas 子物体中的一个 Image。

2. Start 方法

获取 canvas 上的 Image 组件，假设 Canvas 上有一个子物体包含了 Image 组件。

将 opacitySlider 的初始值设置为当前 Image 组件的透明度值（即 canvasImage.color.a）。

为 opacitySlider 添加一个监听器，每当滑块值发生变化时调用 OnSliderValueChanged 方法。

3. OnSliderValueChanged 方法

每当滑动条的值变化时，这个方法会被调用。

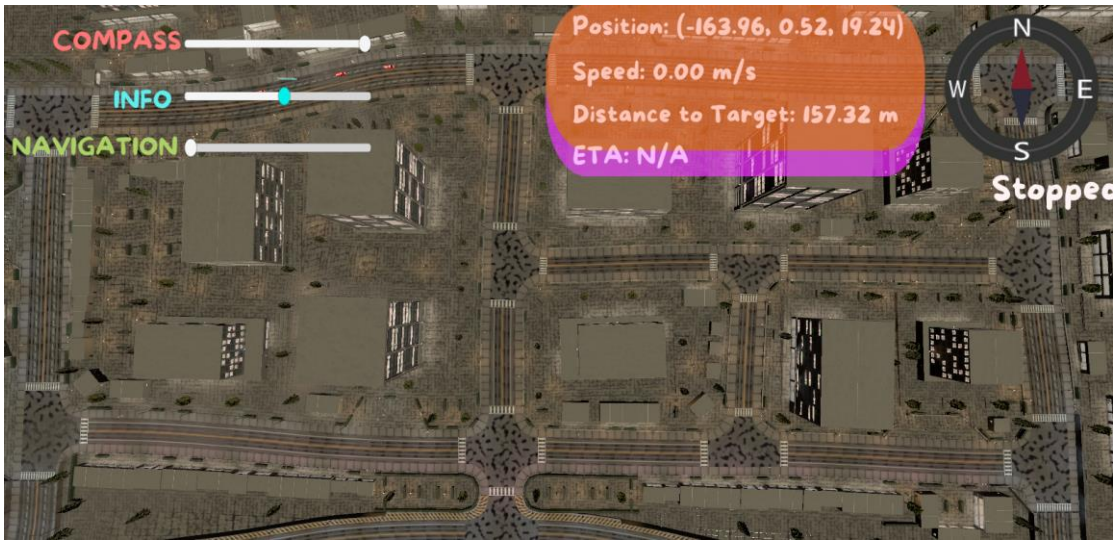
方法内首先获取当前 Image 组件的颜色（包括透明度）。

更新 Image 的透明度（alpha 值）为滑块的当前值。

最后，将新的颜色值赋回 canvasImage.color，更新 Image 的透明度。

2. 配置目的地

按下M键，会显示小地图，左键按下，则会获取该点的坐标，并且赋给空物体destination。



图表 9 小地图配置目的地

6　其他AR窗口信息呈现

选择的AR窗提示信息：目标点距离、方位、当前速度、预计时间：

图表 10 AR窗口显示信息

1. 目标点距离

为什么选择：目标点的距离是一个直观且非常有用的信息，可以帮助用户了解自己与目的地的远近。这对于导航、路线规划和定位系统尤为重要。

应用场景：在导航或路线引导中，目标点距离能清晰地告知用户需要走多远才能到达目标。

2. 方位

为什么选择：方位信息可以帮助用户确定自己与目标点的相对位置，避免迷失方向。在AR中，通常通过箭头或者方向指示来表示方位。

应用场景：例如，在室内导航或户外旅行时，方位帮助用户保持正确的行进方向。

3. 当前速度

为什么选择：显示当前的速度可以帮助用户实时了解自己的行进情况，尤其在需要快速做出决策的情况下（如交通工具的速度监控或徒步行进时）。

应用场景：在骑行、驾车或行走过程中，AR提供当前速度有助于用户掌控进度并调整策略。

4. 预计时间

为什么选择：预计到达目标的时间能让用户更好地规划自己的行程。尤其在长时间旅行中，知道预计的到达时间有助于调整节奏或安排其他活动。

应用场景：在长途行驶或导航中，显示预计时间帮助用户设置合理的期望并避免延误。

7 总结

本实验通过综合运用多种引导方式、配置选项和实时信息提示，构建了一个具有高度互动性和可定制性的AR导航系统。用户不仅能够通过视觉信息获得引导，还能根据需求调节AR窗口的显示方式和透明度，从而提升了导航体验。通过流畅的行人和车辆动画，整个场景更加生动真实，增强了沉浸感。未来还可以根据具体应用场景进一步优化和扩展系统功能，例如加入更多的实时交通信息和个性化的路线推荐。

通过该实验，我也更加深入了解了课上的计算机图形学技术，在Unity中可以看到实物的建模使用了多边形网格的方法；还了解了各种光照技术；在设置材质球的过程中给我也体会到了着色技术；给人物和车辆添加动画时了解了动画模拟，这次实验让我从利用unity的过程中体会了计算机图形学的算法在实际中的应用，也加深了我对计算机图形学算法的理解。