Tactics 项目开发文档

一、项目文件

Tactics所使用的大部分资源文件都储存在Assets文件夹中。本部分将对Assets文件夹中的内容进行注释，方便后续开发者快速上手Tactics项目开发。

1. Scripts

该文件夹储存Tactics运行时所需要的游戏脚本（Untiy自带的脚本除外）。

大多数脚本都要挂载在相应的GameObject上，当该GameObject处于激活状态时脚本才会被执行。

1.1 ButtonManager

ButtonManager.cs

挂载该脚本的对象：MainMenu → Canvas → ButtonManager，Credit → ButtonManager

用途：主菜单场景和成员列表场景中按下不同按钮所执行的函数（主菜单中LoadGame按钮除外）

主菜单中：

Start按钮：QuickStart()，沿用用户上次的设置（如果没有则用默认值）快速进行一次仿真；用户上次的设置通过PlayerPrefs类储存。

GameSettings按钮：GameSet()，进入TrackSelect场景。

Credits按钮：Credits()，进入Credit场景。

Quit按钮：quit()，退出游戏。

成员列表场景中：

MainMenu按钮：MainMenu()，回到主菜单。

MainMenuLoadButton.cs

挂载该脚本的对象：MainMenu → Canvas → ButtonManager

用途：主菜单场景中按下Load Game按钮后，呼出存档窗口；在存档窗口按下Back按钮后，关闭存档窗口。

CompletePanelButton.cs

挂载该脚本的对象：RaceArea → Canvas → Panel complete → CompletePanelButtonManager

用途：在仿真结束界面，按下不同按钮所执行的函数。包括MainMenu()和Retry()。

GamePauseButton.cs

挂载该脚本的对象：RaceArea → Canvas → Panel Pause → PauseButtonManager

用途：在仿真暂停界面，按下不同按钮所执行的函数。

Continue按钮：ContinueRace()，继续仿真。

Save and Load按钮：SaveGame()，打开存档读档窗口。

Retry按钮：Retry()，重新开始。

MainMenu按钮：BacktoMainMenu()，回到主菜单界面。

GameSetting.cs

挂载该脚本的对象：TrackSelect → GameSettingManager

用途：在GameSetting界面，按下不同按钮所执行的函数。

同时，该脚本还储存着用户对仿真的各项设置。

除此之外，用户在进入TrackSelect场景后，还会调用该脚本的Start()函数，对部分参数进行初始化。

用户在进行部分设置后，需要通过PlayerPrefs类储存用户的历史选择。

Monitor的设置是单独实现在MonitorSetting.cs里的。

MonitorSetting.cs

挂载该脚本的对象：TrackSelect → Canvas →ButtonPanel → Panel → AddMonotors → MonitorDropDownManager

用途：实现用户添加监视器的功能。储存用户对监视器的设置。

用户对监视器的设置包括：监视器数目（≤3），监视器目标，监视器的监视角度，都是通过DropDown实现，而非Button。

用户在进入TrackSelect场景后，会调用该脚本的Start()函数，根据用户的过往设置初始化相关的DropDown。

之后在RaceArea场景中MonitorManager.cs脚本根据用户的设置对监视器进行初始化时，需要获取该脚本中储存的参数。

EscScene.cs

挂载该脚本的对象：TrackSelect → Esc， Credit → Esc

用途：在TrackSelect场景和Credit场景通过Esc键快速回到主菜单。

quitGame.cs

挂载该脚本的对象：MainMenu → QuitGame

用途：在主菜单界面，按下Esc时，关闭仿真器

quitRace.cs

挂载该脚本的对象：RaceArea → QuitRace

用途：在仿真场景下，按下Esc时，暂停仿真，并呼出暂停窗口

1.2 Race

StartManager.cs

挂载该脚本的对象：RaceArea → StartManager

用途：控制巡线开始。

进入巡线场景后，首先执行的便是该代码。阅读代码可以从这里入手。

需要完成以下操作：

1）初始化Cpp接口；

2）判断本次运行是否需要读档

如果需要读档，则根据存档里的数据设置相应的参数；

如果不需要读档，则只需要进行一些初始化操作；

3）开启设置车辆颜色的代码；（CarColor.SetActive(true)）

4）开启设置不同巡线模式下UI的代码；（ModeManager.SetActive(true)）

5）开启切换视角功能；（ViewModeManager.SetActive(true)）

6）开启车辆控制的相关代码；（CarControl.SetActive (true);）

6）开启计时；（LapTimer.SetActive (true)）

7）开启车辆Speed、Steer、CruiseError的实时显示；

8）巡线开始前的倒计时；

CarColorSetting.cs

挂载该脚本的对象：RaceArea → CarColor

用途：根据用户的选择，设置车辆的颜色。

对于每辆车，分别获取用户设置的颜色，再将对应颜色的车壳setActive(true)即可。

carControlActive.cs

挂载该脚本的对象：RaceArea → CarControlActive

用途：根据用户的选择，开启每辆车的控制代码。

如果本次运行为读档，则需要复现存档中的仿真结果，需要开启所有Car的LoadControl组件；

如果本次运行不是读档，则需要判断：

如果某辆车的ControlMethod为1，则为键盘控制，需要开启对应Car的CarUserControl组件；

如果某辆车的ControlMethod为2，则需要Cpp代码控制，将CallCppControl对象setActive，以此来调用Cpp代码。

GameModeManager.cs

挂载该脚本的对象：RaceArea → ModeManager

用途：根据用户选择的模式，设置巡线时的UI以及相关GameObject

首先将车辆损伤、车辆得分等数据置零。

若选择TimeMode（也即RaceMode），则需要将TimeMode相关的UI开启，显示巡线时间和车辆损伤。

若选择ScoreMode，则需要将ScoreMode相关的UI开启，显示每辆车所得分数和损伤。同时，还要将赛道上的奖励宝石setActive（也即ScoreObjects）

同时还要考虑参与玩家的数量不同导致的UI差别。

HalfPointTrigger.cs

挂载该脚本的对象：RaceArea → HalfPointTrigger

用途：记录车辆是否已经行驶了半圈，和LapComplete.cs配合使用。若车辆没有经过半途的检查点，即使车辆经过终点线也不会被判断为行驶了一圈。

每辆车对应一个叫HalfFlag的bool型变量，如果该车通过了半途的检查点则该变量值为true，否则为false。

LapComplete.cs

挂载该脚本的对象：RaceArea → LapCompleteTrigger

用途：检测车辆是否通过起点，与HalfPointTrigger.cs配合使用，判断车辆是否行驶完了一圈。

车辆通过终点检查点时，若检测到之前已经通过了半途检查点（HalfFlag为true），则判断该车行驶完了一圈，圈数＋1，并设置对应的HalfFlag和LapFlag，开始下一圈的计数。

若车辆行驶的圈数达到目标圈数，巡线结束，将RaceFinish对象SetActive。

LapTimeManager.cs

挂载该脚本的对象：RaceArea → LapTimeManager

用途：在TimeMode下，巡线计时，并在窗口左上角显示巡线时间。

CruiseData.cs

挂载该脚本的对象：： RaceArea → Car

用途：计算小车距离中心线的距离、赛道曲率等。

参数CarNum为当前小车的编号，不同Car中CruiseData组分的CarNum数值不同。

DistanceError储存小车距离中心线的距离；

Curvature储存当前位置赛道中心线曲率；

AngleError储存小车方向和赛道中心线方向的差距。

WaypointsData为当前赛道中心点的数据，数据为xml格式，是通过CarWaypoints插件生成的。该插件可以很方便地获取赛道中心路标点。

使用该插件时，需要将Editor文件夹中的CarWaypoints文件夹移动到Editor文件夹外；而使用完该插件后，需要将CarWaypoints文件夹放回Editor文件夹。该插件的使用教程可以在CarWaypoints文件夹中找到。

计算小车距离中心线的距离的方法：获取距离小车最近的两个路标点，以这两点作直线，求小车距离该直线的距离。

计算小车当前位置赛道中心线曲率的方法：获取离小车最近的一个路标点以及之后的第一个、第三个路标点，过这三个点做圆，该圆半径的倒数就为当前曲率。不过由于数值大小原因，返回的曲率值为半径倒数的十倍。

CollisionDamage.cs

挂载该脚本的对象：RaceArea → Car

用途：检测小车的碰撞和损伤。

当小车收到碰撞时，获取小车抵消该碰撞所需的冲力，并记录小车碰撞次数和碰撞损伤程度。

BlueScore.cs

挂载该脚本的对象：RaceArea → ScoreObjects → BlueScore

用途：在ScoreMode仿真中，当车辆触碰到蓝色宝石后，给该车辆加分。

RedScore.cs

挂载该脚本的对象：RaceArea → ScoreObjects → RedScore

用途：在ScoreMode仿真中，当车辆触碰到红色宝石后，给该车辆加分。

YellowScore.cs

挂载该脚本的对象：RaceArea → ScoreObjects → YellowScore

用途：在ScoreMode仿真中，当车辆触碰到黄色宝石后，给该车辆加分。

RaceFinish.cs

挂载该脚本的对象：RaceArea → RaceFinish

用途：巡线结束时执行的代码，主要功能为关闭车辆控制相关的代码，并呼出巡线结束的UI。其中ScoreMode和TimeMode的结束UI不完全相同。

1.3 CameraManage

CameraStable.cs

挂载该脚本的对象：RaceArea → Car → Cube

用途：稳定相机。相机为Cube的子对象，在Cube上附加该代码后， Cube在X方向和Z方向不偏转，相机的拍摄角度也因此稳定。

MonitorManager.cs

挂载该脚本的对象：RaceArea → Canvas → Monitors → MonitorManager

用途：根据用户的设置，仿真时在屏幕上方添加小窗口显示监视器中的画面。

ViewModeManager.cs

挂载该脚本的对象：RaceArea → ViewModeManager

用途：在仿真界面，用户按下c键可以切换视角，按下x键可以切换当前观察的车辆。

关于c键和x键的输入，可以在Unity开发界面点击Edit → Project Settings → Input Manager参考。

每辆车有四个摄像头，分别为普通视角、第一人称视角、远视角、俯视角，参数ViewMode表示当前视角；

用户还可以在不同车辆之间切换视角，参数CamNum表示当前观察的车辆编号。

1.4 Display

LapNumDisplay.cs

挂载该脚本的对象：RaceArea → Canvas → UIRight → PanelRight → LapNumDisplayManager

用途：巡线时，在窗口右上角显示当前视角对应车辆的已行驶圈数。

SpeedDisplay.cs

挂载该脚本的对象：RaceArea → Canvas → UIBottom → SpeedDisplay → SpeedDisplayManager

用途：获取每辆车的行驶速度，并显示在屏幕下方的UI上。

Speed数组储存各车辆的行驶速度，但屏幕下方只显示当前视角跟随的车辆的速度值。

SteerDisplay.cs

挂载该脚本的对象：RaceArea → Canvas → UIBottom → SteerDisplay → SteerDisplayManager

用途：获取每辆车的方向盘转角，并显示在屏幕下方的UI上。在第一人称视角模式下，还会显示方向盘的贴图。

和SpeedDisplay.cs不同（有一个Speed数组，分别储存每辆车的速度），只有一个steer参数储存当前视角所跟随车辆的方向盘转角。

ErrorDisplayManager.cs

挂载该脚本的对象：RaceArea → Canvas → UIBottom → ErrorDisplay → ErrorDisplayManager

用途：获取每辆车距离赛道中心线的距离，并显示在屏幕下方的UI上。

和SpeedDisplay.cs不同（有一个Speed数组，分别储存每辆车的速度），只有一个CruiseError参数储存当前视角所跟随车辆的巡线误差。

DamageDisplay.cs

挂载该脚本的对象：RaceArea → Canvas → UILeft → ……DamageDisplay → DamageDisplayManager

用途：仿真过程中，在窗口左上角显示各车辆的损伤程度。

ExtentOfDamage数组和CollisionNum数组分别储存各车辆的损伤程度和碰撞次数。

参数CarNum表示该脚本要显示几号车辆的损伤值。若脚本是显示一号车辆损伤程度的UI的组件，该值则设为0；若脚本是显示二号车辆损伤程度的UI的组件，该值则设为1……以此类推。

ScoreDisplay.cs

挂载该脚本的对象：RaceArea → Canvas → UILeft → ScoreModeDisplay → ScoreModeDisplay

用途：ScoreMode仿真过程中，储存各个车辆的得分，并在在窗口左上角显示各车辆的得分。

参数CarNum表示该脚本要显示几号车辆的得分。若脚本是显示一号车辆得分的UI的组件，该值则设为0；若脚本是显示二号车辆得分的UI的组件，该值则设为1……以此类推。

MiniMap.cs

挂载该脚本的对象：RaceArea → Canvas → UIRight → MiniMap → MarkPlayerCar

用途：控制小地图的车辆标识位置和方向。

参数CarNum表示该脚本控制的是几号车的小地图标志。

1.5 CarControl

CppControl.cs

挂载该脚本的对象：无

用途：实现Cpp代码和Unity的接口。

API的详细情况请参考CppControl.cs和CppCarControl.h中的定义。

CallCppControl.cs

挂载该脚本的对象：RaceArea → CppControl

用途：调用外部cpp代码，控制车辆运行。

为保证仿真确定性，不使用Update()函数而使用FixedUpdate()函数。

仿真开始时，在Start()内会调用CppControl.InitializeCppControl()，进行仿真初始化。

每次在FixedUpdate()中调用CppControl.CarControlCpp()函数。

InitializeCppControl()函数和CarControlCpp()函数定义在CppControl.cs脚本中，但实际内容在外部cpp代码中实现。

用户在CarControlCpp()中编写代码进行运算，将运算得到的控制参数（即steering、accel、footbrake、handbrake）通过CarMove函数传递到CppControl.cs脚本中的steering[]、accel[]、footbrake[]、handbrake[]数组中，再在CppControl.cs脚本中使用CarController.Move函数控制车辆运行。

CarController.cs（Unity自带）

挂载该脚本的对象：RaceArea → Car

用途：实现车辆模型和车辆控制，目前使用的是Unity自带的代码。

若需要修改车辆动力学模型，应该从这入手。

CarUserControl.cs（Unity自带）

挂载该脚本的对象：RaceArea → Car

用途：实现车辆的键盘控制，是Unity自带的代码，进行了一些修改，目前暂时没有使用。

CarControlKeyBoard.cs

挂载该脚本的对象：RaceArea → Car

用途：实现车辆的键盘控制，是根据Unity自带的代码CarUserControl.cs修改的。

参数CarNum表示该脚本控制的是几号车辆。若脚本为一号车辆的组件，则将该值设为0；若脚本为二号车辆的组件，则将该值设为1……以此类推。

1.6 Save

SaveTactic.cs

挂载该脚本的对象：无

用途：定义SaveTactic类，明确存档时需要储存哪些内容。

SaveCondition.cs

挂载该脚本的对象：无

用途：定义SaveCondition类，记录各个存档位的使用情况。

SaveConditionManager.cs

挂载该脚本的对象：RaceArea → Canvas → Panel Save → SaveConditionManager

用途：根据存档位使用情况，修改UI。

若存档没有被使用，在仿真场景中查看该存档则需要显示Save按钮以供用户存档；在主菜单中查看该存档则不能有任何按钮。

若存档已经被使用，在仿真场景中查看该存档则需要显示Save按钮和Load按钮，分别提供存档和读档的功能；在主菜单中查看该存档则需要显示Load按钮和Delete按钮，分别提供读档和删档功能。

SaveButton.cs

挂载该脚本的对象：RaceArea → Canvas → Panel Save → PanelRoll → Panel → Save Savebutton

用途：实现存档功能。

在存档窗口按下Save按钮后，储存本次仿真过程中对车辆输出的所有操作参数，用于在读档时复现。将存档所需的内容储存为一个SaveTactic类对象，并将其转化为二进制文件储存在相应的文件夹中，并修改存档使用情况。

LoadButton.cs

挂载该脚本的对象：RaceArea → Canvas → Panel Save → PanelRoll → Panel → Save Savebutton，MainMenu → Canvas → Panel Load → PanelRoll → Panel → Save Button → Delbutton/Loadbutton

用途：实现读档和删档功能。

在存档窗口按下Load按钮后，进入该存档对应的赛道场景中，并令参数LoadNum为对应的存档号（不等于0）。当LoadNum不等于0时，进入仿真场景后，StartManager.cs和carControlActive.cs脚本会读取对应的存档并进行相应的设置。

在存档窗口按下Delete按钮后，删除对应存档，并修改档位使用情况。

LoadControl.cs

挂载该脚本的对象：RaceArea → Car

用途：依照存档中的内容控制小车行驶。

RecordControllerOutput.cs

挂载该脚本的对象：RaceArea → RecordOutputofController

用途：使用动态数组ArrayList储存对各个小车输出的控制参数。

1.7 Others

SkyBoxRot.cs

挂载该脚本的对象：MainMenu和TrackSelect场景中的任意一个对象均可

用途：令MainMenu和TrackSelect场景中背景的天空缓缓旋转。

1.8 TrackData

该文件夹储存道路中心路标点文件，这些文件目前都是通过CarWaypoints插件生成的。这些数据在CruiseData.cs脚本中被使用。

1.9 WayPointData

CarWaypoints插件插件所需使用的部分脚本，可以参考CarWaypoints插件教程理解。教程可以在Editor/CarWaypoints文件夹中找到。

2. Scene

该文件夹存放Tactics中的各个场景。

（如果打开项目发现Unity的Scene窗口一片空白，点击该文件下的场景即可）

2.1 MainMenu

主菜单界面，是仿真器开始运行时最初进入的场景。

2.2 Credit

展示项目开发人员名单的界面。

2.3 TrackSelect

巡线设置界面。

2.4 RaceArea

各个巡线地图的场景。

3. Plugins

该文件夹命名固定为Plugins。

插件文件夹，Tactics的外部接口会放置在该文件中。

例如，要实现Tactics和Cpp代码的通讯，用户需要将Cpp代码生成的dll文件放进Plugins文件夹中，即可实现通讯。关于Cpp与Unity接口相关的代码详见CppControl.cs脚本。

4. Editor

该文件夹命名固定为Editor。

Editor文件夹可以在根目录下，也可以在子目录里，只要名子叫Editor就可以。Editor下面放的所有资源文件或者脚本文件都不会被打进发布包中，并且脚本也只能在编辑时使用。一般呢会把一些工具类的脚本放在这里。

目前使用的CarWaypoints插件，在使用时需要将CarWaypoints文件夹拖出Editor文件夹外，使用完成后需要将其放回Editor文件夹，以免被打进发布包中。

5. Resources和StreamingAssets

这两个也是命名固定的特殊文件夹，不过目前的开发过程并没有涉及到。

6. 储存素材的文件夹

6.1 Audio

该文件夹储存Tactics所用的音频素材。

6.2 Materials和Textures

这些文件夹储存Tactics所用的材质、纹理等素材。

6.3 Objects

该文件夹储存了Tactic所用模型，如树木、岩石、天线杆等。

6.4 Standard Assets

该文件夹储存了Tactic使用的Unity自带的标准素材。目前使用的车辆模型就是Unity自带的素材。

6.5 SkyBox

该文件夹储存了Tactic使用的天气盒素材。

6.6 Animation

该文件夹储存了Tactic使用的动画素材。

7. GameObjects

Tactics中的GameObject并不直接储存在Assets文件夹中，但GameObject是游戏引擎中非常重要的组成部分，仿真中一切实体或逻辑都是由一个个GameObject组成起来的。

在Unity开发界面的Hierarchy窗口中可以查看每个Scene中包含的GameObject。本节将对各个Scene中的GameObjects进行简要注释。

7.1 RaceArea

Car

即用户控制的车辆。

装载刚体组件（Rigidbody），还装载着CarController.cs、CarUserControl.cs、CarAudio.cs、ColisionDamage.cs、CruiseData.cs脚本。

其子对象中，Cube用于稳定摄像头；

SkyCar中为车辆的外观模型，CarColorSetting.cs脚本设置车辆颜色时，就是将对应颜色的车壳setActive，而不同颜色的车壳就包含在SkyCar中。

Canvas

用户看到的UI。

UILeft：左上角的UI，又分为TimeMode的UI和ScoreMode的UI。

UIRight：右上角的UI，主要显示当前圈数和小地图。

UIBottom：下方UI，主要显示当前车辆的速度、方向盘方向和巡线误差。

Monitors：当用户添加了监视器时，在上方显示的监视器窗口。

CountDown：巡线开始时的倒计时UI。

Panel Save：存档窗口的UI。

Panel complete：结束窗口的UI。

Panel Pause：暂停窗口的UI。

StartManager

控制仿真开始的GameObject，装载StartManager.cs脚本。

在执行完一定的操作后，会让对象ModeManager、LapTimeManager、CarControlActive、CarColor、ViewModeManager激活。

ModeManager

由StartManager激活，根据用户选择的模式（TimeMode或ScoreMode）设置UI。装载GameModeManager.cs。

LapTimeManager

由StartManager激活，计时。装载LapTimeManager.cs。

CarControlActive

由StartManager激活，根据用户选择的控制方式，分别激活每辆车的键盘控制功能或代码控制功能。装载CarControlActive.cs。

ViewModeManager

由StartManager激活。装载ViewModeManager.cs，实现用户切换视角和车辆的功能。

CarColor

由StartManager激活。装载CarColorSetting.cs，根据用户的选择设置各车辆的颜色。

CppControl

由ModeManager激活。当仿真要使用外部cpp代码进行车辆控制时，该对象会被激活。装载CallCppControl.cs脚本，实现cpp控制功能。

ScoreObjects

由ModeManager激活。ScoreMode中场地上的宝石。在ScoreMode下，该对象会被激活。

HalfPointTrigger

触发器，不渲染但是有碰撞箱。判断车辆是否通过半途检查点。装载HalfPointTrigger.cs脚本。

LapCompleteTrigger

触发器，不渲染但是有碰撞箱。判断车辆是否通过终点线。装载LapComplete.cs脚本。若车辆通过终点线，仿真结束，则会激活RaceFinish。

RaceFinish

由LapCompleteTrigger激活。当仿真结束时，该对象会被激活。装载RaceFinish.cs脚本，完成仿真结束时所需的操作。

QuitRace

装载QuitRace.cs，按下Esc后暂停游戏并呼出暂停窗口。

RaceScene

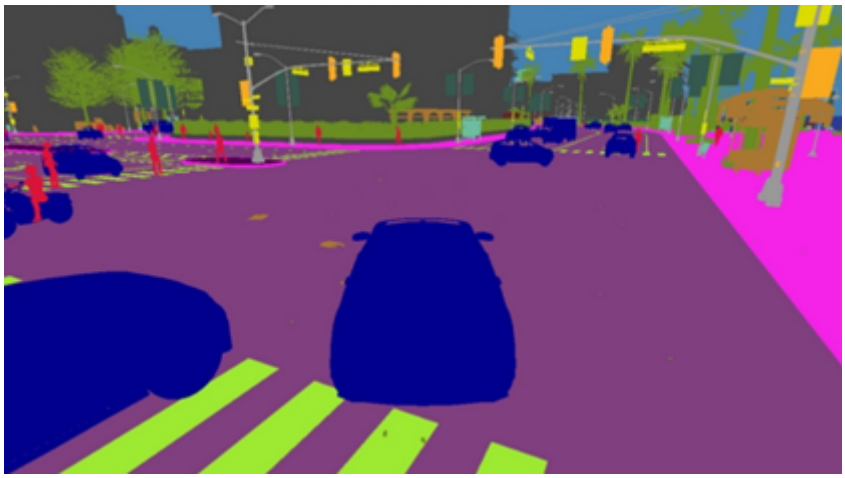
赛道的物理模型和环境的模型等。这里面的GameObject主要是场景中的地形和装饰等。

二、功能调研

Tactics的部分功能仍处在调研阶段，该部分会整理目前的调研结果。

1. 语义相机

主要调研了Carla的语义分割相机 Semantic segmentation camera。语义相机将视野中的所有物品分类并用相应的颜色表示，效果如下图所示。



当仿真器开始运行时，场景中的每个元素都会被赋予一个标签。物品通过其相对文件路径进行分类。例如，mesh若储存在Unreal/CarlaUE4/Content/Static/Pedestrians，则对应标签为Pedestrian。

服务器端提供一幅带有红色通道编码的标签信息的图像：红色值为x的像素属于标记为x的对象。这个原始的carla.Image可以在ColorConverter中的carla.CityScapesPalette的帮助下存储和转换，以应用标签信息并显示具有语义分割的图片。

Carla自带22种语义tag。官网给出了用户如何添加新tag的教程。

教程链接：https://carla.readthedocs.io/en/latest/tuto_D_create_semantic_tags/

已明确Carla是通过渲染层面的代码，获取了一个rgb图象，而这个图象的r通道的值为tag编号，最后根据该值设定颜色反馈给用户。

2. 地图编辑

2.1 SketchUp调研

2.1.1 简介

SketchUp由@Last Software公司开发，是一套直接面向设计方案创作过程的设计工具，其创作过程不仅能够充分表达设计师的思想而且完全满足与客户即时交流的需要，它使得设计师可以直接在电脑上进行十分直观的构思，是三维建筑设计方案创作的优秀工具。

其卖点就在于使用简便，人人都可以快速上手，且有大量免费的学习资源。无论草图、3D模型均可轻松完成。支持各种专业文件格式，让协作变得简单。

2.1.2 使用体验

与Unity兼容性

Unity的SketchUp接口：https://docs.unity.cn/cn/2018.4/Manual/HOWTO-ImportObjectSketchUp.html

Unity可以直接读取SketchUp 文件，实际使用起来相当方便；而其他的建模软件一般需要把模型转换为FBX格式，再导入Unity。对于过于复杂的建模，可能在导入时会出现破面、丢面的效果，不过仿真器一般并不需要太复杂的建模。

Unity直接读取SketchUp文件时，支持部分SketchUp特性：纹理和材质、组件定义和组、文件中每个场景的摄像机数据。

地图精度

高。数据精度可到小数点后六位。

建图难易度

使用SketchUp建模非常方便，通过简单的快捷键和推拉等操作就可以完成建模。

系统兼容性，要求的电脑配置

要求64位操作系统，不支持32位；

SketchUp2021不支持MacOS 10.13 (High Sierra)、Mac OS X 10.12 (Mojave)、Windows7、Windows8、Windows Vista 及更早版本、Mac OS X 10.12 (Sierra) 及更早版本、Linux、虚拟化环境、远程桌面连接、Boot Camp、Parallels和VMWare环境。

2.1.3 研究领域的使用情况

在建筑设计、室内设计、园林设计、住宅设计、城市规划、木工设计、3D打印等方面都有广泛的应用。

暂时没发现有仿真器使用该软件。

2.1.4 数据格式

一般储存为.skp文件，是SketchUp的专用格式，可以直接被SketchUp和Uniy读取。

也可储存为.fbx文件。Autodesk FBX文件格式是一种常用的3D数据交换格式，用于 3D 编辑器和游戏引擎之间。支持 3D 模型、场景层次、材质照明、动画、骨骼、蒙皮、混合形状、 NURBS 曲面和曲线。使用二进制格式，快速又高效。

2.2 地图编辑器例程

该例程给地图编辑器功能的实现提供了一个专业的思路。虽然该例程使用的是老版本的Unity，但可以使用Unity2021.3.6f1c1 打开，只需要在首次打开时等待项目更新即可。

交大云盘：

链接:https://jbox.sjtu.edu.cn/l/n1E0pP

提取码: lrur

百度网盘：

链接：https://pan.baidu.com/s/16B1TXVxe_lRIxyp4Gmi6Kw

提取码：dgvw