## 赛车功能实现

基本实现了一个赛车游戏的完整流程，而其中赛车功能的相关实现占较大的比重：赛车运动的控制(前进、后退、转弯、刹车)、氮气功能、复位功能、反向行驶检测、码表速度显示、相机跟随、按钮拖拽并保存位置、暂停游戏、伪漂移、小地图、地图选择、圈数检测、计时功能、开局倒计时、结算界面、选车界面旋转视角、选车界面拍照并保存本地相册、场景切换进度条、单人道具。

### 1.1 赛车运动的控制

1.1.1 赛车前进

|  |  |  |
| --- | --- | --- |
|  | 后轮驱动 | 四轮驱动 |
| 目前采用的方案 | × | √ |
| 初步结果 | 效果较差：力矩相同时赛车速度较小；加速度对力矩不敏感，力矩变化数倍，而加速度、最大速度变化不明显 | 目前为止，效果较好 |

1.1.2 赛车刹车

|  |  |  |
| --- | --- | --- |
|  | MiniGame | 真实世界 |
| 刹车方案 | 前轮刹车 | 前轮刹车 |
| 漂移方案 | 仅通过前轮刹车 + 转向无法实现转弯时漂移 解决方案：加入伪漂移 | 控制速度进弯 + 转向 + 拉手刹 + 调节方向控制车身角度 + 出弯松油门 + 调整方向加速进直道 |

1.1.3 赛车转弯

查阅相关资料，赛车的转向角度推荐为30度到45度，为了实现灵敏转向将最大转向角度设置为45度，比较两种转向方案。

|  |  |  |
| --- | --- | --- |
|  | 转动角度恒定 | 转动角度变化 |
| 目前采用的方案 | × | √ |
| 最大转动角度 | 45度 | 45度 |
| 转向效果 | 效果较差：速度很大时，方向的灵敏度增加，难以操纵 | 速度增加最大转向角随之减小，操纵手感远优于前者，但是不漂移高速过弯较难，但是这一点也符合预期效果 |
| 不同速度转动角度 | 45度 | 目前仅仅只是一个初步调节的公式，在后续开发中需要进行细致调节 |

转动角度变化方案中不同速度对应的转动角：

是转向角，是赛车速度，是赛车参考速度，是速度衰减系数。上述公式是经过理论推导后个人认为较为合理的公式，在后续开发中会进一步优化。本次Mini Game中取参考速度为90km/h，衰减系数取1，即线性衰减。转向角随速度变化如下图所示。

1.1.4 赛车后退

|  |  |  |
| --- | --- | --- |
|  | 赛车前进 | 赛车后退 |
| 目前采用的方案 | 向前的动力矩 | 赛车前进的力矩取负 |
| 相机跟随 | 速度大，赛车看起来远，速度小，赛车看起来近，可以体现出速度变化 | 赛车倒退速度过快时，会出现赛车已经退到视野之外 |
| 后续改进方案 | 保持不变 | 限制最大倒车速度、倒车动力矩；优化倒车时的相机跟随 |

1.1.5 车轮转动

将车轮转速RPM（每分钟转动圈数）转换为车轮每次Fixed Update转动的角度：

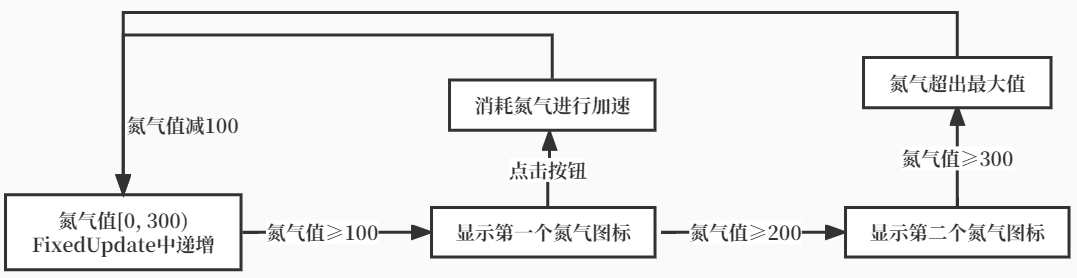
1.1.6 赛车的参数设置

|  |  |  |
| --- | --- | --- |
|  | 参数值 | 参数调节 |
| Mass | 1500 | 赛车质量在1000kg到1500kg较为合理 |
| Drag | 0.1 | 空气阻力可以设置较小：阻力可以通过不同地面对应不同摩擦系数设置，而非空气阻力 |
| Angular Drag | 5 | 转向阻力较大，减少赛车打转情况；但是也不能过大，否则会影响赛车转向 |
| 赛车重心 | 高度接近地面 | 虽然和实际不符，但是可以降低甚至完全避免侧翻等严重影响游戏体验的情况 |
| 赛车碰撞体 | 底盘略高于实际模型 | 赛车模型底盘一般较低，适当提高碰撞体底盘可以避免频繁底盘碰地，提升游戏体验 |
| 赛车材料 | 弹性系数0.5 | 虽然与实际不符，但是撞墙反弹方便玩家快速回归正常赛道，极大程度优化游戏体验 |
| 赛车底盘材料 | 弹性系数0 | 避免赛车底盘触地反弹的情况 |

1.1.7 Wheel Collider的参数设置

|  |  |  |
| --- | --- | --- |
|  | 参数值 | 参数调节 |
| Mass | 50 | 车轮质量在20kg到50kg较为合理 |
| Radius | 0.37 | 车轮半径必须和实际模型在大小和位置上完全重合，否则会出现颠簸等不良表现 |
| Suspension Distance | 0.1 | 赛车的悬挂距离，直接决定了车身的上下振动幅度，适当振动幅度可以使游戏更真实 |
| Spring | 1000 | 弹性系数过大振动不明显，弹性系数过小即使很小的冲击振动幅度也会达到振幅上限 |
| Damper | 3000 | 阻尼过小则振动时间过长，反之时间过短 |
| Forward Friction Stiffness | 3 | 提高赛车速度 |
| Sideways Friction Stiffness | 2.5 | 提高转向灵敏度 |

### 1.2 氮气功能



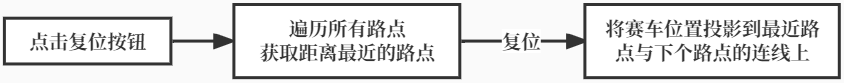
|  |  |  |
| --- | --- | --- |
|  | 目前采用的方案 | 后续改进 |
| 集气的方式 | 随时间进行集气 | 优化漂移后，可以根据漂移进行集气 |
| 氮气喷射 | 点击按钮喷射氮气 | 后续可以加入小喷功能 |

### 1.3 复位功能&反向行驶检测

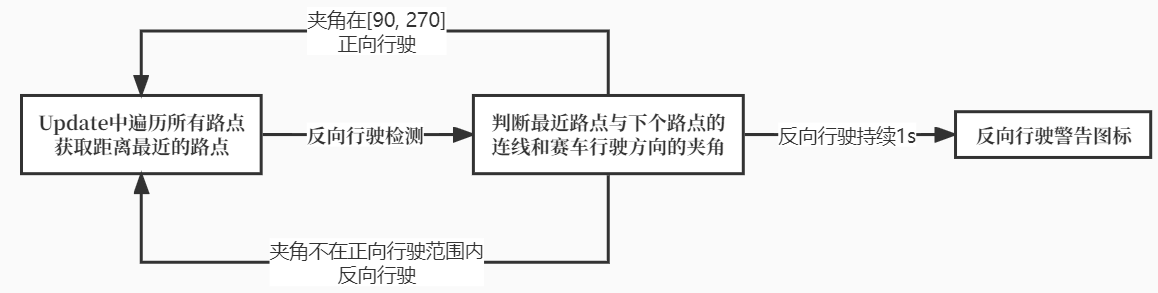
1.3.1 路点的标注

|  |  |  |
| --- | --- | --- |
| 路点标注 | 适用路段 | 目的 |
| 稀疏 | 直线路段、路线相距较远路段 | 提升性能：反向行驶检测恪尽职守地在Update中运行以提醒玩家是否反向行驶，过多的路点会在性能上产生负担 |
| 稠密 | 转弯路段、路线相距较近路段 | 否则复位时很可能无法复位到预期的路点，产生游戏bug |

1.3.2 车辆复位



1.3.3 反向行驶检测



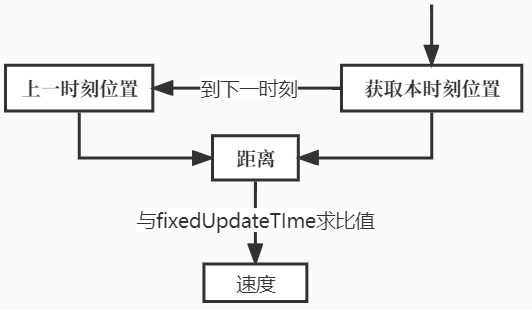
1.3.4 路点方法分析

基于路点的车辆复位及反向行驶检测存在诸多问题，与使用赛车通过面的方法进行比较。

|  |  |  |
| --- | --- | --- |
|  | 路点 | 赛车通过面/线段 |
| 空间性能 | 只需要存储点，空间性能较好 | 存储平面或者线段，略差于路点方法 |
| 时间性能 | 每次复位都要遍历所有点，性能差；更致命的是在反向行驶检测时需要在Update中时刻遍历所有点，严重影响性能 | 只需要比较最近一次经过的面即可，时间性能远远优于路点方法 |
| 准确度 | 由于路点默认设置在车道中心，当两条车道边缘处的距离小于车道边缘到车道中心的距离，那么无论是复位还是反向检测在车道边缘时都必然出现bug，因此路点方法在工程中没有任何实际应用价值 | 不会存在bug |

### 1.4 码表速度显示

1.4.1 计算速度



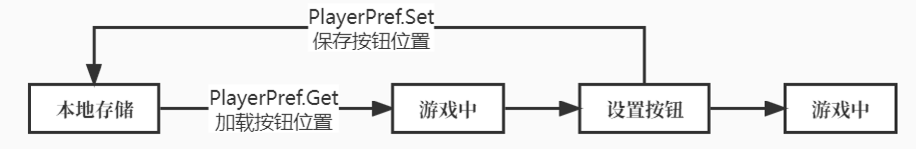
1.4.2 Lerp平滑码表盘指针转动

对Lerp，Slerp，Move Towards, Smooth Damp进行总结。

|  |  |
| --- | --- |
| Lerp | 线性插值，每次移动一定比例位置，用于线性运动  移动百分比不为1时，永远无法达到终点 |
| Slerp | 一般在四元数旋转操作中使用，球形插值不改变长度 |
| Move Towards | 限定每次移动最大速度，而非按比例移动  可以理解为匀速直线运动 |
| Smooth Damp | 平滑阻尼，可以设定大约达到目标所需的时间 |

### 1.5 按钮拖拽并保存设置

实际游戏中，不同玩家操作习惯不同，因此部分操作按钮设置为可以调节位置并存储在本地，在开始游戏时加载保存好的设置。



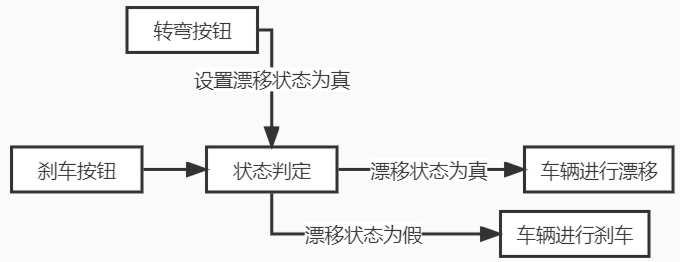
|  |  |
| --- | --- |
|  | PlayerPrefs |
| 优点 | Unity提供的公共类，存储比较方便 |
| 缺点 | 代码的相似度过高冗余、只支持三种数据类型的存储、数据容易被找到并修改保密性较差 |
| 改进方案 | 自定义数据库，但是实现起来较为麻烦  对PlayerPrefs进行封装：提供其他数据类型的接口包括用户自定义的类、定义算法对key和value的值进行加密 |

### 1.6 暂停游戏

|  |  |
| --- | --- |
| Time Scale | 时间流逝缩放比例，直接影响Time . time变化速度 |
| 暂停/继续游戏 | 设置Time Scale = 0 / 1 |
| Time Scale影响值 | time、delta Time、fixed Time、fixed Unscaled Delta Time |
| Time Scale不影响值 | Realtime Since Startup、unscaled Time、unscaled Delta Time、fixed Unscaled Time、fixed Delta Time |
| 影响游戏 | Fixed Update当Time Scale = 0时暂停  Update和Late Update没有停，仍然在渲染，但是其中依赖于Time的逻辑当Time Scale = 0时暂停  使用真实时间Real Time代替Time可以不受Time Scale影响 |

### 1.7 伪漂移

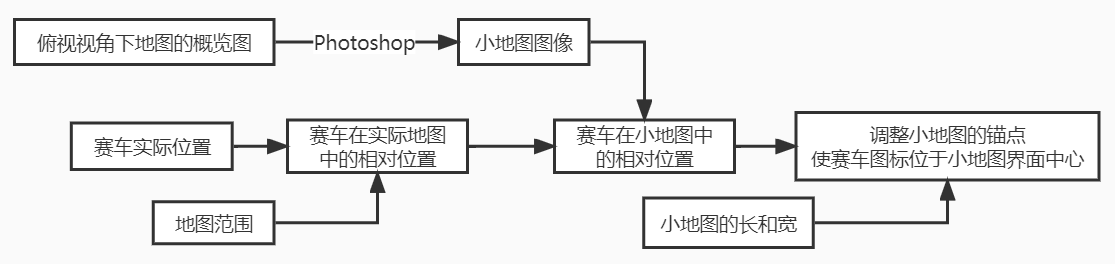
只通过前轮进行刹车转弯无法达到漂移的效果，因此需要在同时按刹车和转向键时，对赛车的车身进行旋转，达到伪漂移的效果。



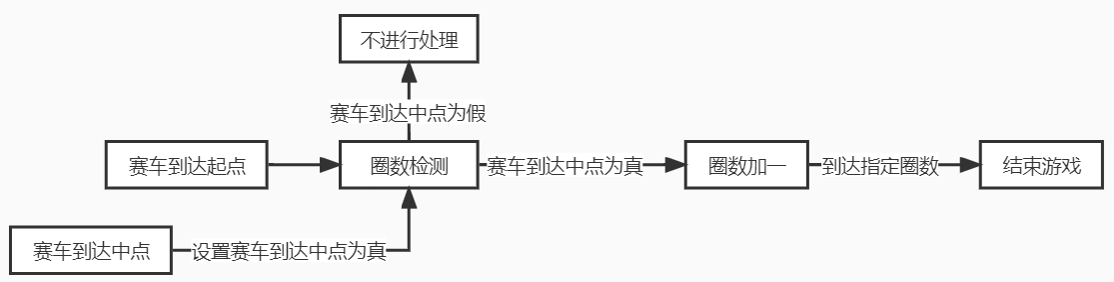
本次Mini Game中漂移不是重点，仅仅完成了简单的车身旋转，但是这个方案效果较差，与给赛车旋转力矩的方案进行比较。

|  |  |  |
| --- | --- | --- |
|  | 直接旋转车身 | 赛车旋转力矩 |
| 实现 | 较容易 | 较复杂 |
| 效果 | 很生硬不真实，效果较差 | 较真实 |
| 改进方案 | 可以考虑进行非线性旋转：旋转角速度先线性增加至峰值后再线性降低到零，然后由旋转角速度推算出车身的旋转角度，可以起到模拟旋转力矩的效果 | 只增加车身的旋转力矩可能很难实现较好的漂移效果，在后续开发中考虑加入更多的力矩及力以模拟更好的效果 |

### 1.8 小地图



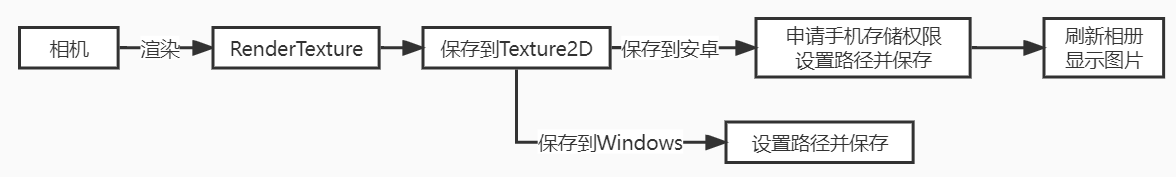
### 1.9 圈数检测



### 1.10 选车界面视角变换

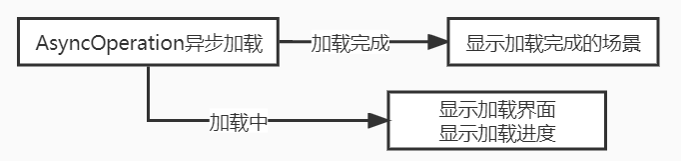
|  |  |  |
| --- | --- | --- |
| 屏幕触数 | 手指操作 | 视角变换 |
| 1 | 左右滑动 | 赛车为原点 赛车y轴正方向为旋转方向 旋转 |
| 1 | 上下滑动 | 赛车为原点 相机x轴正方向为旋转方向 旋转 |
| 2 | 指尖距离缩放 | 相机中心到赛车中心的距离增大/减小 |

### 1.11 选车界面拍照并保存到本地相册



|  |  |
| --- | --- |
| 手机直接截屏 | 相机渲染 |
| 只能保存整个屏幕的图像 | 可以选择渲染并保存的场景 |

### 1.12 场景切换进度条



|  |  |
| --- | --- |
| Async Operation | 决定加载场景操作是否完成 |
| Allow Scene Activation | 值为true 加载完成后自动跳转  值为false加载完成后可以通过按键控制，使其值为true，实现手动跳转 |

### 1.13 单人道具

|  |  |
| --- | --- |
| 直接使用赛车标签判定道具 | 由于赛车可能包含多个碰撞体，会出现道具多次拾取的现象 |
| 赛车设置包含特定标签的对象 | 此对象只有一个，只会拾取一次道具 |

|  |  |  |
| --- | --- | --- |
| 加速带道具 | 进入/离开 | 动力矩增大/恢复正常行驶动力矩 |
| 氮气道具 | 拾取 | 氮气值增加 道具消失一段时间 |
| 传送道具 | 拾取 | 赛车位置变换、行驶方向变换 |
| 跳跃道具 | 拾取 | 获得竖直方向速度进行跳跃  获得水平方向速度进行加速 |

### 1.14 其他功能

这些功能暂无要总结的技术：车辆选择、地图选择、开局倒计时、结算界面、相机跟随。

### 1.15 UI界面

Mini Game对UI的实现和已有资料基本相同，因此不进行单独总结。但是有两点需要格外注意：

1、UI在设置锚点、相对位置、自身大小时，注意对主流屏幕分辨率的适配；

2、尚待解决：对全面屏/异形屏（水滴屏、刘海屏等）的适配。

## 着色器

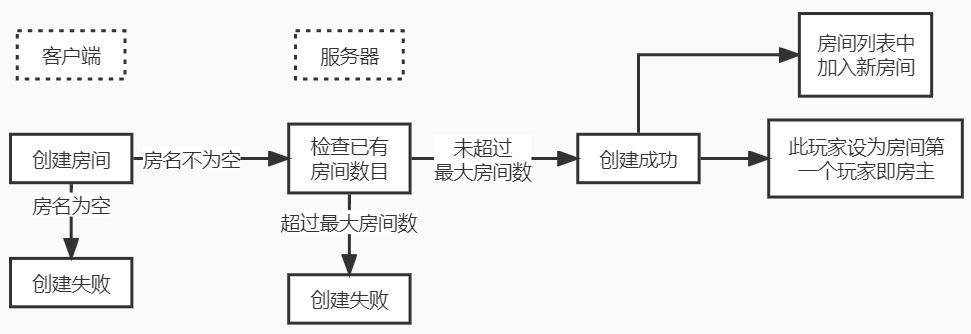
着色器实现功能较为简单，见shader文件。

## 联网功能

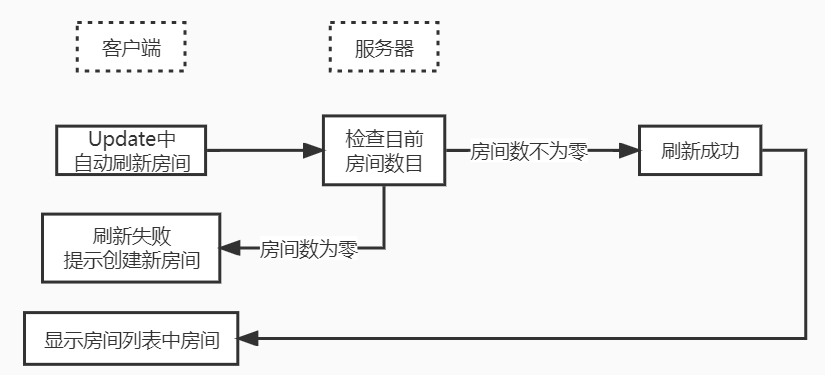
联网系统的搭建主要包括服务器和客户端两部分，并通过Google Protocol Buffer进行数据传输。完成系统搭建后，通过内网穿透将服务器的局域网IP映射到公网起到公网IP的作用，使得客户端可以直接连接服务器端。基于搭建的服务器-客户端框架，主要实现了如下功能：搭建简易的房间系统（创建房间、自动刷新房间、加入房间、房间内聊天、房主选择地图、房主开始游戏）、简单的位置同步、开局时简单的时间同步、小地图同步其他玩家、结算排名。

### 3.1 房间系统

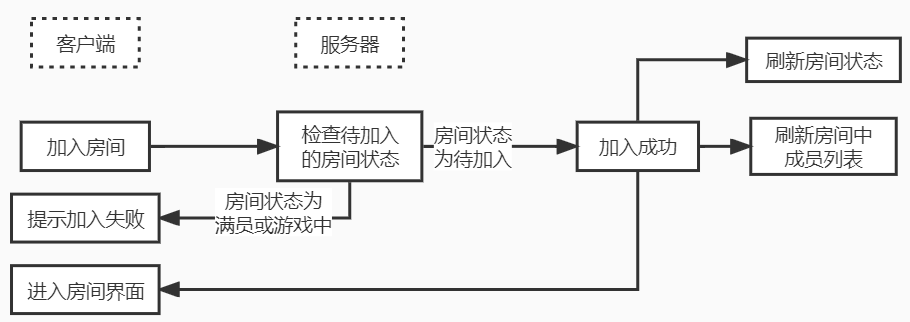
3.1.1 创建房间



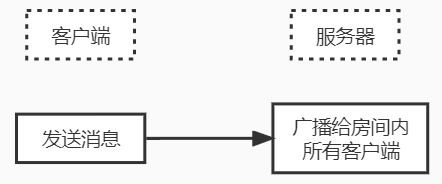
3.1.2 自动刷新房间



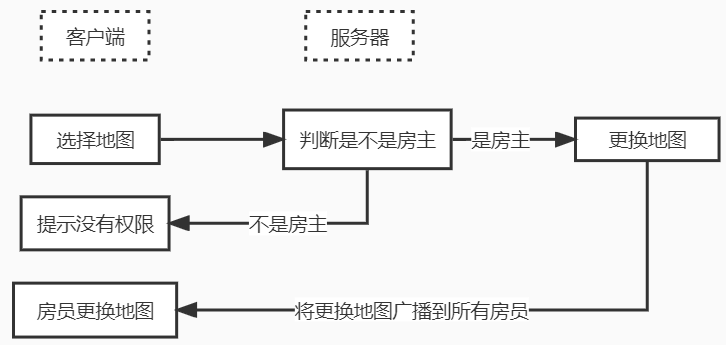
3.1.3 加入房间



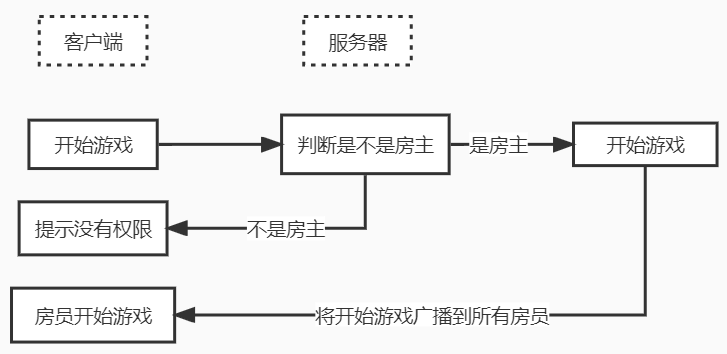
3.1.4 房间内聊天



3.1.5 房主选择地图

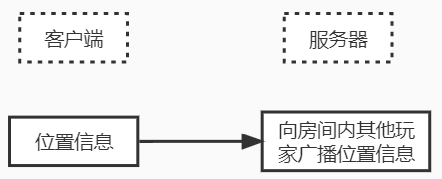


3.1.6 房主开始游戏

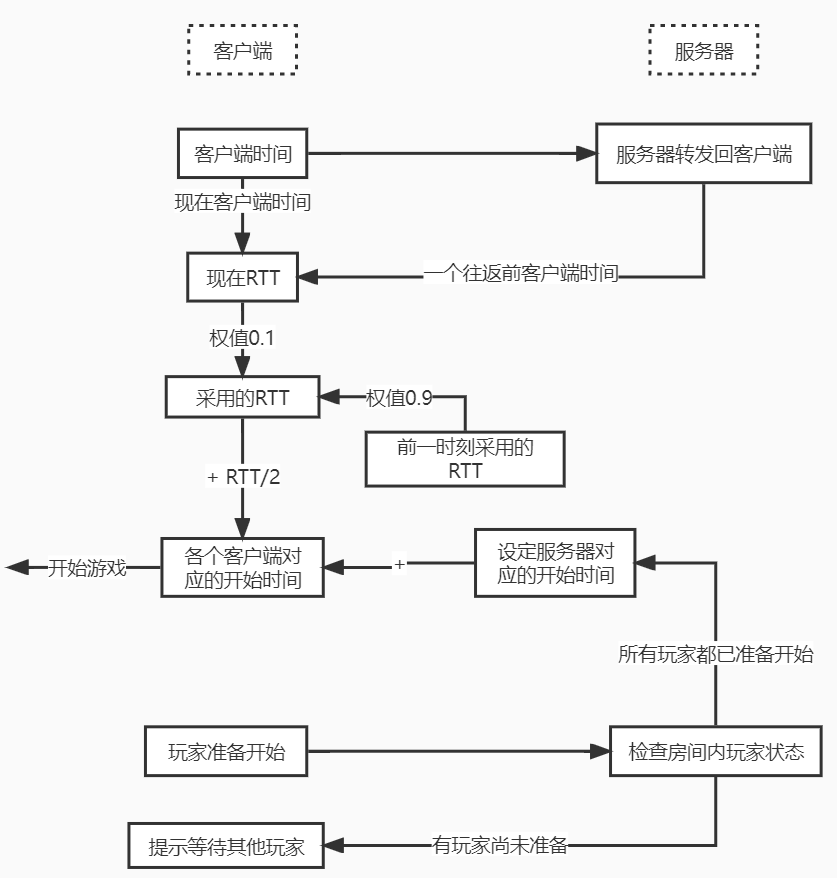


### 3.2 位置同步

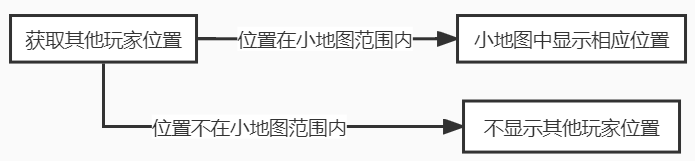
玩家之间的赛车同步目前暂时只采用了位置同步，会产生诸多问题：赛车位置同步延迟较大，赛车速度较快时会其他玩家同步的赛车出现较为明显的卡顿。在后续开发中考虑同步速度、角速度等多方面信息以提高赛车同步的实时性。



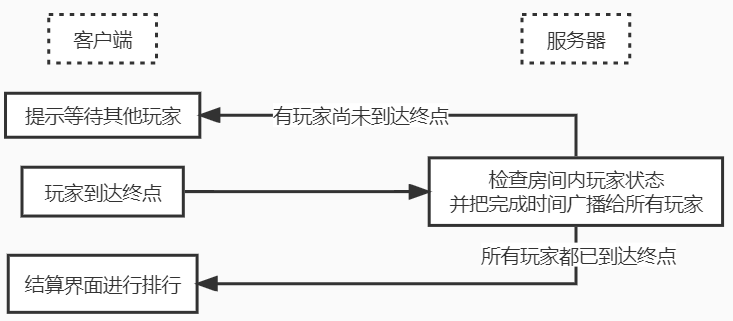
### 3.3 开局时间同步



### 3.4 小地图同步其他玩家



### 3.5 结算界面排名



## 四、 其他总结

### 4.1 Update与Fixed Update

Update是每帧更新而Fixed Update是固定间隔更新，Fixed Timestep就是Fixed Update的调用间隔。因为Fixed Update可以保持固定的时间间隔调用，因此某些需要在时间上精确模拟的任务就需要放在Fixed Update中。在本次游戏demo中，物理更新及一些需要用到Fixed Update Time的放在Fixed Update中，而大部分游戏逻辑以及UI相关放在Update中。对两种极端情况进行分析。

当Update中运算量暴增，即Update间隔拉长了许多。为了保证在两帧间隔较大的时候依然能精准模拟物理之类的运算，一帧之间会运行多次Fixed Update，将较大的间隔切分成多个固定的小时间段计算。但这种切分不能是无节制的，Fixed Update中的运算也会带来计算负荷，如果为了追上帧间隔而带来太多Fixed Update调用会让下一帧的时间更长，而更长的帧间距则需要更多的Fixed Update来追上，这会导致一个恶行循环，所以Fixed Update的调用会有一个时间限制，在Edit→Project Setting→Time中的Maximum Allowed Timestep选项。因此Fixed Update的固定时间间隔，并不是真正意义的固定的时间差。而是相对Update来说一个类似节拍器的存在。当两帧的时差过大时，Unity会“缩放”时间至设定的最大帧间隔Maximum Allowed Timestep，在游戏中看起来变得又卡又慢，但是这个设定可以有效防止在卡顿时游戏的物理系统彻底暴走的现象。

当Fixed Update中运算量暴增，会直接出现恶性循环：庞大的Fixed Update会直接拉大帧间隔，卡顿会比Update相同运算量情况下严重十几倍以上，因此需要尽量降低Fixed Update的负荷，否则会给游戏的运行带来严重的不稳定性以及卡顿。

### 4.2 GC垃圾回收

GC全称是garbage collection，即垃圾回收，这种机制主要作用在堆空间上。当堆内存上一个变量不再处于激活状态的时候，其所占用的内存并不会立刻被回收，不再使用的内存只会在GC的时候才会被回收。如果没有足够的存储单元，unity会触发垃圾回收来释放不再被使用的堆内存。GC操作是一步缓慢的操作，因此需要使得堆上创建的变量尽量处于激活状态以降低GC机制出发，查阅相关文档，可得降低GC的一些具体措施：

1. 使用对象池，尽量避免创建新对象；
2. string和StringBuilder的使用，降低重复修改字符串对性能的影响；
3. 减少装箱操作，因为装箱时必须创建一个新的对象；
4. 限制协程调用，因为start coroutine()实际上是new一个对象；
5. 用for代替foreach，foreach会涉及到迭代器的使用，5.5版本以前有装箱操作；
6. struct中不要有引用类型变量（struct是值类型，而如果struct中有引用类型的变量，GC会检查整个struct）；
7. 主动调用GC。