**Unity3D教程：制作简单小汽车游戏**

Posted on 2013年05月15日 by U3d / [Unity3D 基础教程](http://www.unitymanual.com/category/manual/unity3d-%e5%9f%ba%e7%a1%80%e6%95%99%e7%a8%8b)/被围观 112 次

1.现实中的车不是那么好漂移的，漂移需要轮胎与地面的低摩擦和良好的悬挂系统配合。

2.游戏里完全模拟力学引擎也是不现实的，因为漂移在现实中需要高超的技能，你不能指望玩家能有专业的技能水平去操控或者改装你的车吧，万一拐弯忘拉手刹，或者拉太久。而且完全的物理模拟，不仅费人脑去设计，也费电脑资源去运算。再说了，开发的不是微软模拟飞行这样的专业级模拟游戏。

3.在无数次失败的代码后发现，高速移动的赛车忽然间的转弯是会侧翻的，这个不是游戏引擎问题，我查了很多的汽车侧翻事故，这个东西是客观存在的，也就是说能漂移的汽车，需要有稳定精确的汽车架构，很低的底盘(我试过这个，但是在u3d里不管用，重心和底盘都快贴到地上了，依然高速运动下转弯会侧翻)，专业级的悬挂系统(这很重要，这也是游戏里模拟不出来的)，每个部件精确的调试，还有一大堆的汽车辅助系统(这个后面说)。

4.侧翻不可避免，但游戏里不允许玩家稍微操控失误就侧翻，这会大大影响游戏性。我试过用加大重力的方法来防止侧翻，但是加大重力也就加大了摩擦力，使用物理引擎去模拟漂移就更加的困难。

5.自动稳定杆系统（ASBS）能有效防止侧翻，但是为了一款游戏有必要这么麻烦么？

经过几天的研究，我研究出几套漂移的解决方案（我开发采用的是最后一种，重点介绍最后一种）

方案一：整个赛车干脆都用translate做坐标移动，这个就是无视wheelcollider物理引擎的做法，新手适用，简单方便，但之后的开发上会有很多问题需要解决（如果你想解决这些问题的话）。

方案二：用wheelcollider的悬挂设置和摩擦力设置做。这个应该是最标准的，但是我一直没研究出来他的Forward Friction和Sideways Friction怎么设置，并且这些设置有什么效果（官方的赛车例子根本不会漂移）。当然这个做法的移动是使用wheelcollider的motorTorque做移动的。

方案三：轮子一样还用wheelCollider，但汽车移动的动力不使用轮子的motorTorque，而是在汽车的rigidbody上施加Force来传动（官方的例子是这样的，这也是令我匪夷所思的，只能说明官方也知道他的wheelcollider引擎还有问题）。用Force有个好处，你可以在拐弯的时候做受力分析，计算出最后侧滑的合力方向和大小，实时计算赛车动力方向和侧滑判定。这个唯一的问题就是汽车稳定性问题，在高速运动的汽车中很难保证他的稳定，自己开发ASBS系统也不太可能吧。

方案四：这个是我的方法，用wheelcollider和translate混合处理。首先一点重要的，所有汽车移动，转弯都必须使用wheelcollider提供的函数（至于为什么参看我关于wheelcollider的日志），用wheelcollider方法可以保证车子正常运动，但不会漂移（要么过快车速侧翻，要么就直接转弯过去了），判断漂移只需要判断玩家点击手刹的时候，计算车子拉手刹一瞬间的转弯角和惯性方向，使用transform.translate来实现甩尾，有效规避掉不必要的摩擦判断和物理判断。

由于手刹后前后轮抱死产生的摩擦和离心力的作用改变了整辆车合力的方向，所以导致汽车运动方向的改变，产生漂移。那我们假设他拉得不是手刹，而只是普通的制动闸，只有制动减速效果，所以先用wheelcollider做正常的拐弯减速运动，然后单独写一个traslate按预计漂移方向做物体移动，并判断当手刹按钮按下后再进行强行物体移动加以模拟减速移动来产生漂移的摩擦衰减效果（transform的坐标移动是不受物理引擎影响的，所以写起来就很简单。）

这样最后综合起来的效果就能产生漂移效果的最初框架了。玩家操作简单，开发者开发也简单，何乐而不为。最后剩下的工作就是要去慢慢调试一些数据来改变用户体验了。

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |
| --- | --- |
| 001 | [html] view plaincopyprint? |
| 002 |  |
| 003 | **var** rearWheel1:WheelCollider; |
| 004 |  |
| 005 | **var** rearWheel2:WheelCollider; |
| 006 |  |
| 007 | **var** frontWheel1:WheelCollider; |
| 008 |  |
| 009 | **var** frontWheel2:WheelCollider; |
| 010 |  |
| 011 | **var** wheelFL : Transform; |
| 012 |  |
| 013 | **var** wheelFR : Transform; |
| 014 |  |
| 015 | **var** wheelRL : Transform; |
| 016 |  |
| 017 | **var** wheelRR : Transform; |
| 018 |  |
| 019 | **var** steer\_max = 20; |
| 020 |  |
| 021 | **var** motor\_max = 10; |
| 022 |  |
| 023 | **var** brake\_max = 100; |
| 024 |  |
| 025 | **private** **var** steer = 0; |
| 026 |  |
| 027 | **private** **var** forward = 0; |
| 028 |  |
| 029 | **private** **var** back = 0; |
| 030 |  |
| 031 | **private** **var** motor = 0; |
| 032 |  |
| 033 | **private** **var** brake = 0; |
| 034 |  |
| 035 | **private** **var** reverse = **false**; |
| 036 |  |
| 037 | **private** **var** speed = 0; |
| 038 |  |
| 039 | function Start() { |
| 040 |  |
| 041 | rigidbody.centerOfMass = Vector3(0, -0.05, 0); |
| 042 |  |
| 043 | } |
| 044 |  |
| 045 | function FixedUpdate () { |
| 046 |  |
| 047 | speed = rigidbody.velocity.sqrMagnitude; |
| 048 |  |
| 049 | steer = Mathf.Clamp(Input.GetAxis(“Horizontal”), -1, 1); |
| 050 |  |
| 051 | forward = Mathf.Clamp(Input.GetAxis(“Vertical”), 0, 1); |
| 052 |  |
| 053 | back = -1 \* Mathf.Clamp(Input.GetAxis(“Vertical”), -1, 0); |
| 054 |  |
| 055 | **if**(speed == 0) { |
| 056 |  |
| 057 | **if**(back > 0) { reverse = **true**; } |
| 058 |  |
| 059 | **if**(forward > 0) { reverse = **false**; } |
| 060 |  |
| 061 | } |
| 062 |  |
| 063 | **if**(reverse) { |
| 064 |  |
| 065 | motor = -1 \* back; |
| 066 |  |
| 067 | brake = forward; |
| 068 |  |
| 069 | } **else** { |
| 070 |  |
| 071 | motor = forward; |
| 072 |  |
| 073 | brake = back; |
| 074 |  |
| 075 | } |
| 076 |  |
| 077 | rearWheel1.motorTorque = motor\_max \* motor; |
| 078 |  |
| 079 | rearWheel2.motorTorque = motor\_max \* motor; |
| 080 |  |
| 081 | rearWheel1.brakeTorque = brake\_max \* brake; |
| 082 |  |
| 083 | rearWheel2.brakeTorque = brake\_max \* brake; |
| 084 |  |
| 085 | frontWheel1.steerAngle = steer\_max \* steer; |
| 086 |  |
| 087 | frontWheel2.steerAngle = steer\_max \* steer; |
| 088 |  |
| 089 | wheelFL.localEulerAngles.y = steer\_max \* steer; |
| 090 |  |
| 091 | wheelFR.localEulerAngles.y = steer\_max \* steer; |
| 092 |  |
| 093 | wheelFR.Rotate(frontWheel1.rpm \* -6 \* Time.deltaTime,0, 0); |
| 094 |  |
| 095 | wheelFL.Rotate(frontWheel2.rpm \* -6 \* Time.deltaTime,0, 0); |
| 096 |  |
| 097 | wheelRR.Rotate(rearWheel1.rpm \* -6 \* Time.deltaTime,0, 0); |
| 098 |  |
| 099 | wheelRL.Rotate(rearWheel2.rpm \* -6 \* Time.deltaTime,0, 0); |
| 100 |  |
| 101 | } |