# Kinect 人机交互实验报告

## 狂野飙车体感控制

## 1. 项目背景

《狂野飙车 8：极速凌云》（ Asphalt 8：Airborne）是一款由 Gameloft 制作发行的一款风靡全球的赛车游戏。该游戏最初只有手游版，利用陀螺仪和虚拟按键操作；后来也移植到了 PC 平台，利用鼠标和键盘进行操作。漂移和氮气加速系统是该游戏的一大特色。在转弯的时候，玩家可以踩刹车以进行漂移并搜集氮气，搜集一定氮气之后便能使用氮气加速。选择合适的时机激活氮气加速可以激发“完美氮气”以进行最大幅度的加速。此外，该游戏中的物理引擎也较为独特，通过合适的操作，玩家可以控制赛车在空中实现水平螺旋飞跃，翻滚等特技动作。本次实验，我们小组将尝试使用 Kinect 来完成模拟键盘和鼠标对该游戏的全程操作。

## 2. 目标与需求分析

PC 玩家对该游戏的操作主要分为两部分，一部分是游戏开始前的鼠标操作，用于选择车辆和关卡等，另一部分是游戏中的键盘操作，用于车辆的控制。鼠标操作主要涉及鼠标的移动和单击鼠标左键的动作。因此需要用 Kinect 对玩家手的位置和状态的进行追踪并完成对鼠标操作的转化。具体设计如下：

键盘映射：

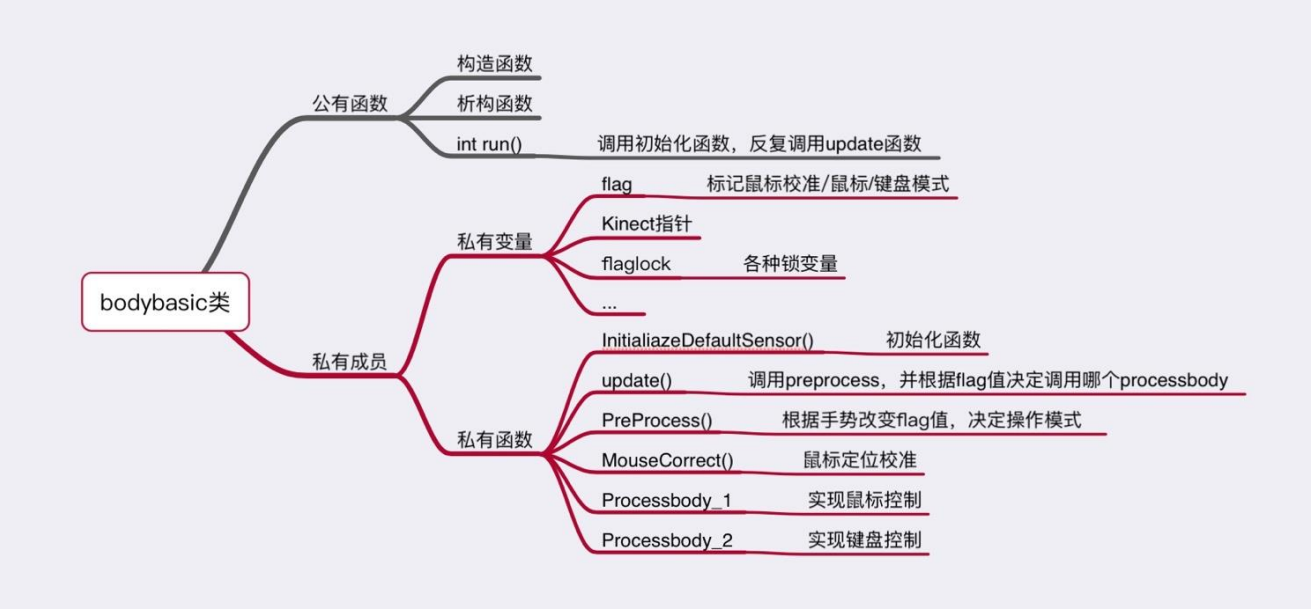
| 游戏功能 | 虚拟键位 | 人体姿势 |
|---|---|---|
| 左转 | A key | 左右手斜率满足左转阈值条件 |
| 右转 | D key | 左右手斜率满足右转阈值条件 |
| 刹车 | S key | 左脚前伸 |
| 氮气推进 | F key | 右脚前伸 |
| 视角切换 | C key | 左手张开 |
| 暂停游戏 | Esc key | 右手张开 |
| 复位 | R key | 双手置于胸前平行且垂直于地面 |

鼠标映射：右手高举进行鼠标模式和键盘模式的切换，左手高举进行鼠标校准，右手张开移动表示鼠标移动，右手闭合迅速张开表示鼠标点击，左右手同时张开且两手斜率满足一定阈值条件表示鼠标滚轮操作。

## 3. 系统结构

### 3.1 类的基本结构

为了实现对赛车的控制，我们设计了如下程序结构：




## 3.2 执行过程

在程序执行的过程中，我们通过调用 run()函数开始整个控制过程，在进入 run()函数之后，首先调用 InitializeDefultSensor()函数完成对传感器的初始化，然后循环调用 update()函数。在 update()函数中，首先调用 PreProcess()函数根据当前检测到的人体姿态确定 flag 的值。然后根据 flag 的值判断接下来的操作。若 flag=0，则调用 MouseCorrect()函数进入鼠标校准操作；若 flag=1，则调用函数 Processbody_1()进入鼠标控制模式；若 flag=2，则调用函数 Processbody_2()进入键盘控制模式。

# 4. 实现细节

## 4.1 鼠标校准

键盘控制对应类中的 MouseCorrect()函数，鼠标校准的基本思想是将显示器鼠标移动的像素区域与 Kinect 捕捉到的特定区域一一映射起来，防止 Kinect 捕捉右手位置时鼠标跟踪失败、偏移等问题。Kinect 识别到的人体关节点的三维信息保存在结构体 Joints 中，通过将右手举到左上方和右下方，利用这两个点的 x,y 坐标信息便能在空间确定一块特定区域。接下来只需要获取显示屏的分辨率信息，将像素点坐标和 Kinect 获取的区域映射起来即可。对于屏幕分辨率的获取需要 Windows 自带 API 函数 GetMonitorInfo（）和 MONITORINFO 类。

上述过程可以实现显示器的鼠标校准和坐标映射，但我们的实验中用到了多块显示屏的扩展模式，希望鼠标只在显示游戏的副屏内移动而不会进入主屏，这就需要对屏幕进行选择。这里同样用到了 GetMonitorInfo（）函数，由 GetMonitorInfo(hm[monitor_num], &mixTemp)，返回当前显卡驱动显示器的个数 monitor_num 及序号，对每个显示器进行遍历，引导用户选择正确的显示器即可。对显示器进行遍历选择时，通过 mouse_event()函数利用循环结构使鼠标在当前显示器中央从左向右进行移动，帮助用户进行显示器判断和选择。

鼠标校准时的语音操作提示主要利用了 PlaySound()函数，播放本地录制的相应音频文件。

## 4.2 鼠标控制

键盘控制对应类中的 Processbody_1()函数，本实验鼠标控制主要用于进入游戏和游戏中的车辆、赛道选择等。根据实际的需求需要实现鼠标的准确移动，鼠标点击和鼠标滚轮操作，Windows 提供了对鼠标进行操作的 API 函数 mouse_event()。由于 Kinect 对人体的检测是按帧数检测的，1s 内能够获取多帧信息，因此只需将 Kinect 检测到的右手位置与屏幕像素点实时映射即可实现鼠标跟随检测到的右手的坐标变化而移动，具体实现用到了 mouse_event(MOUSEEVENTF_ABSOLUTE | MOUSEEVENTF_MOVE, mx, my, 0, 0)。鼠标点击通过检测到的右手状态进行，右手闭合后张开点击一次，点击操作分解为鼠标按下和松开两个操作，用 mouse_event（）时用到了 mouse_event(MOUSEEVENTF_LEFTDOWN, 0, 0, 0, GetMessageExtraInfo())（鼠标左键按下），和鼠标左键松开操作：mouse_event(MOUSEEVENTF_LEFTUP, 0, 0, 0, GetMessageExtraInfo())。对于鼠标滚轮的操作需要先检测到左手处于张开状态，通过 Kinect 获取的左右手坐标计算斜率，满足阈值条件即进行鼠标滚轮滚动操作，鼠标滚轮操作主要用 mouse_event(MOUSEEVENTF_WHEEL, 0, 0, x, 0)(x 表示滚动的像素点位置变化，x>0 向上滚动，x<0 向下滚动)。

## 4.3 键盘控制

键盘控制对应类中的 Processbody_2()函数，通过人体姿势完成键盘操控，实现在游戏的过程中对赛车转弯、加速、刹车、复位以及切换视角、暂停游戏的控制。下面分别对每一操作进行详细的介绍：

1、转弯

我们模拟实际中人对方向盘的操控，通过检测左右手的位置来判断是否进行转弯以及左转还是右转。首先，我们计算两只手之间连线的斜率（xy 平面内），当斜率大于设定的阈值之后认为正在进行转弯操作，然后根据左右手之间的相对高度判断左转还是右转。然后在判定进行左转时按下”A”键（不松开），并设定锁变量不许再进行按键的操作；右转时按下”D”键（不松开），并设定锁变量不许再进行按键的操作；不进行转弯时松开键盘，并解除控制不许按键的锁变量。

之所以不直接根据斜率的正负进行判断，是因为在转弯的过程中如果转动方向盘的幅度过大（大于 90°），斜率的正负号将发生反转，这将会导致出现误判的情形，因此我们在判定的过程中采用了上述方案，引入左右手的标记避免了这种误判情况的发生。

2、加速和刹车

我们通过左右脚控制加速和刹车，左脚对应刹车，右脚对应加速。在具体实现中，我们分别计算左右脚到左右膝盖连线的斜率（此时是 yz 平面内的斜率），当斜率小于设定的阈值时，认为脚踩了下去，此时按下并立即松开对应的按键，其中左脚刹车对应“S”键，右脚加速对应“W”键，并在此过程中设定与转弯类似的锁变量。

3、复位

我们设定两手小臂在胸前竖直对应复位操作。通过检测左右手手掌和左右手肘之间连线的斜率（xy 平面），若两手斜率均大于阈值，则按下”R”键进行赛车复位；否则松开”R”键。

4、切换视角和暂停游戏

我们设定左手张开切换视角、右手张开暂停游戏。通过检测左右手的张开闭合状态，控制对应的按键完成操作。

若检测到左手张开并且当前锁变量为 0，则按下”C”键切换视角并设定锁变量为 1，当检测到左手闭合并且此时锁变量为 1 时松开”C”，并修改锁变量为 0。目的在于控制在按下”C”键的过程中不许再按”C”键，同时保证只有手张开后再闭合时才松开”C”键，这是因为在这里按下时间过短时系统会自动忽略此次按键操作。

若检测到右手张开并且当前锁变量为 0，则按下”Esc”键并送松开暂停游戏，同时设定锁变量为 1，避免在手张开的时间内持续按键，若检测到右手闭合，则设定锁变量为 0,允许进行”Esc”的按键操作。

# 5. 实验结果

实验较好地实现了利用 Kinect 对游戏的操作和鼠标控制，游戏操作反应良好，组员通过 Kinect 操作在游戏中拿了第一名，基本能实现键盘操作的效果。鼠标控制方面，受限于 Kinect 获取帧数的限制，基本能够稳定实现移动点击和滚动等基本操作，但是如果用于 FPS 射击游戏等对鼠标灵敏度要求较高的游戏，灵敏度可能比不上实际鼠标操作。此外语音提示能解决单屏幕下打开游戏无法看到操作提示的问题，屏幕校准时的检测能解决多屏扩展模式下对显示器的显示问题。总之本次实验较为成功，实现了不错的演示效果，极具趣味性，也收获了老师的好评。

# 附件清单

1. main.cpp main 函数
2. CBodyBasics.h 头文件
3. CBodyBasics.cpp 操作主体文件
4. Audio 语音提示文件夹