PQ Labs

上海品奇数码科技有限公司

PQ Labs 触摸桌 SDK

C/C++版使用手册

v 2.0

版本说明

版本	日期	备注
1.0	2018-11	
2.0	2019-10	Fix bug. 增加获取 platform 框信息接口。

目录

1. 简述

上海品奇数码科技有限公司（PQ Labs Shanghai）一直致力于为客户提供一流的多点触摸解决方案。自 2007 年以来，品奇数码就不断的推动多点触摸技术在各个行业的应用，目前，品奇数码的客户遍及展览展示、教育、军事、房地产等诸多行业，极大的拓展了多点触摸技术的应用范围，成为国内多家知名或重大活动的多点触摸解决方案。

触摸桌 SDK 可提供由触摸桌识别的预置物件（水杯、TAG）或后期定制的非预设物件（自定义商品等）的详细信息，并附带配置自定义非预设物件的小工具。本文为 C/C++ 版本的 SDK 使用手册。

2. 流程图

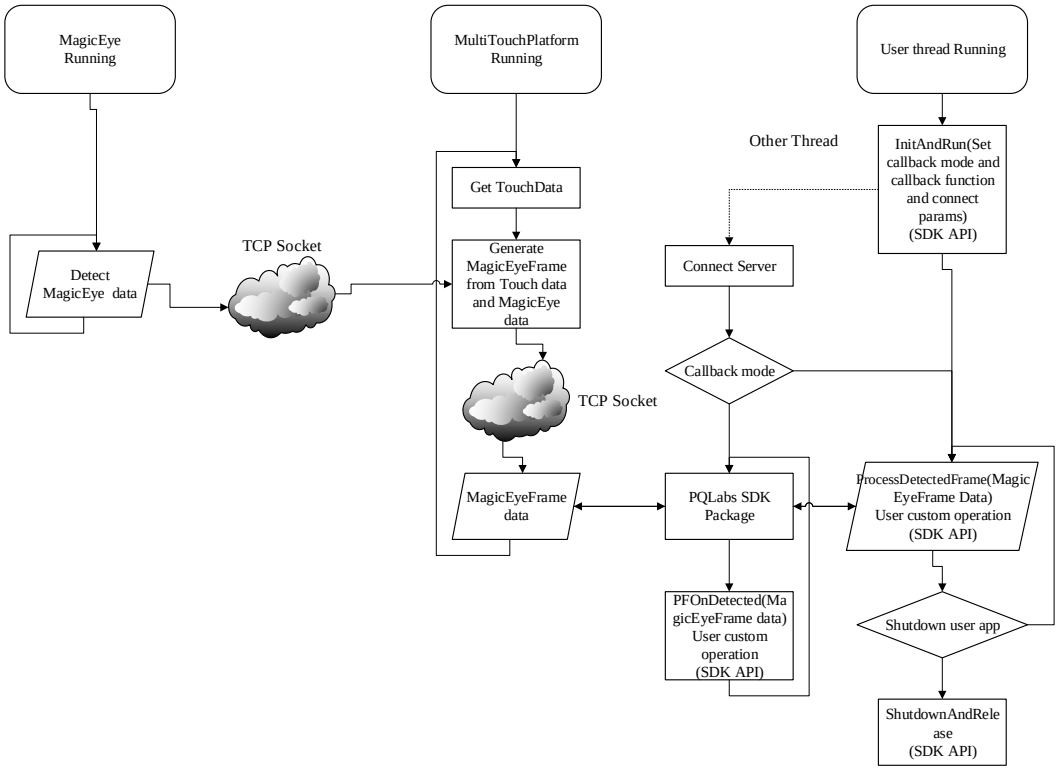

图 1 SDK 流程图

本 SDK 依赖于本公司的触摸平台软件 MultiTouchPlatform 及本公司的物体识别软件 Light-FieldCapture，以上软件为 SDK 提供了 TAG 识别及物体识别、物体定位功能，SDK 和以上软件间采用 TCP SOCKET 进行通信。SDK 采用动态链接库的方式提供，并且内部采用多线程方式实现，用户可选用异步回调函数或同步函数获取 TAG 及被识别的物体的信息，开发人员只要了解一些 C/C++编程知识的，均可快速上手并开发。

3. 开发步骤

这里以 Visual Studio 2017 作为开发 IDE 为例，对使用 SDK 开发 Application 过程中的几个关键步骤进行说明。

3.1 准备

a) 下载并安装本公司触摸桌驱动软件

下载地址：

b) 下载并安装本公司的 Light-FieldCapture 物体识别软件

下载地址：

c) 下载品奇触摸桌开发包 C/C++版本

下载地址：

解压 zip 文件后得到的 PQTouchTableSDK 文件夹。

- bin、include、lib 文件夹分别为 SDK dll、SDK 头文件及 SDK lib 存放目录。
- SampleCode 文件夹：含 SDK 的示例项目，下文会对该 SampleCode 进行详细说明。
- PQLabs TouchTable SDK Reference.doc：SDK 参考手册，内含本 SDK 的所有函数、结构体等详细定义。

d) 下载并安装 Visual Studio 2017

微软下载地址：

e) 运行 MultiTouchPlatform 及 Light-FieldCapture 软件，

3.2 Visual Studio 工程设置

这里以简单的 Console 程序为例，运行 Visual Studio，

a） 新建一个 Console Project，这里取工程名为 TestSDK_CPP，工程文件夹为 SampleCode，保存 solution 为 TestSDK；

b） 在 Visual Studio 中，右键单击 TestSDK 工程，在“Properties”中，找到”VC++ Directories”-> “Include Directories”一栏，填入”;..\..\..\include”，指定头文件搜索路径中加入“include”文件夹，以便在编译时能找到"PQDetectionSDKTool.h"头文件；找到”VC++ Directories”-> “Library Directories”一栏，填入”;..\..\..\lib”，指定库文件搜索路径中加入“lib”文件夹，以便能在链接时找到“PQDetectionSDKTool.lib”库；

c） 右键单击 TestSDK 工程，依次点击”Add”->”New Item…”->”C++ File(.cpp)”，修改文件名为 TestSDK_CPP.cpp，点击 Add。

d） 在 TestSDK_CPP.cpp 中添加下面两行，引用 PQDetectionSDKTool.h”及链接” PQDetectionSDKTool.lib”库。

```
#include "PQDetectionSDKTool.h"
#include <stdio.h>
#include <Windows.h>

#pragma comment(lib, "PQDetectionSDKTool.lib")
```

3.3 物体识别数据处理

在TestSDK_CPP.cpp中添加如下行，此OnDetected函数参数magicEyeFrame为当前帧中检测到的物体的详细信息。用户可以在此函数中写自己需要的功能。示例中为将MagicEyeFrame中的物体识别信息打印出来。

在TestSDK_CPP.cpp中添加如下行

```
void OnDetected(DetectionData detectionData)
{
  //show tags detail
  for (int i = 0; i < detectionData.tagsCount; i++)
  {
       printf("tagId: %d, \ttagType: %d, \ttagAction: %d, \tcX: %.2f,
\tcY: %.2f\n", detectionData.tags[i].id, detectionData.tags[i].type,
```

```
detectionData.tags[i].status, detectionData.tags[i].cx,
detectionData.tags[i].cy);
  }
  //show objects detail
  for (int i = 0; i < detectionData.objectsCount; i++)
  {
      printf("eyesid: %d, \ttype: %d, \ttype_description: %s \tstatus: %d,
\tcX: %.2f, \tcY: %.2f\n", detectionData.objects[i].id,
detectionData.objects[i].type, detectionData.objects[i].type_description,
detectionData.objects[i].status, detectionData.objects[i].cx,
detectionData.objects[i].cy);
  }
  printf("-------------------------------------------\r\n");
}
```

下面是两种可选方式获取物体识别信息

对物体识别数据的获取有两种方式，第一种为异步回调函数。此模式下 OnDetected 会被 SDK 内的子线程更新数据时调用

```
int main()
{
    int re = InitAndRun("127.0.0.1", 53607, CALLBACK_MODE, OnDetected);
    if (re != 0)
    {
        printf("InitAndRun Error. Error code:%d\r\n", re);
    }
    getchar();
    ShutdownAndRelease();
}
```

第二种为同步函数获取物体识别信息，即用户在自己的线程中主动调用 ProcessDetectedData(DetectionData *pDetectionData);此函数来获取当前物体识别信息。注意：在此主动模式下，用户必须尽可能高频率的调用本函数。触摸桌的触摸功能帧率永远小于等于此函数调用频率，若用户停止调用本函数且SDK与MultiTouchPlatform连接未中断时，会导致触摸桌触摸无响应。

```
int main()
{
    int re = InitAndRun("127.0.0.1", 53607, DEFAULT_MODE, 0);
    if (re != 0)
    {
        printf("InitAndRun Error. Error code:%d\r\n", re);
```

```
            getchar();
            return re;
        }
        while (true)
        {
            DetectionData detectionData;
            re = ProcessDetectedData(&detectionData);
            if(re == 0)
                OnDetected(detectionData);
        }
        ShutdownAndRelease();
    }
```

3.4 退出时的处理

在退出程序前或停止使用 SDK 功能时可调用 `ShutdownAndRelease();`函数。等待函数返回 `true` 时即可。

4. 完整的示例代码

TestSDK_CPP.cpp

```
#include "PQDetectionSDKTool.h"
#include <stdio.h>
#include <Windows.h>

#pragma comment(lib, "PQDetectionSDKTool.lib")

void OnDetected(DetectionData detectionData)
{
    //show tags detail
    for (int i = 0; i < detectionData.tagsCount; i++)
    {
        printf("tagId: %d, \ttagType: %d, \ttagAction: %d, \tcX: %.2f, \tcY: %.2f\n",
detectionData.tags[i].id, detectionData.tags[i].type, detectionData.tags[i].status,
detectionData.tags[i].cx, detectionData.tags[i].cy);
    }
    //show objects detail
    for (int i = 0; i < detectionData.objectsCount; i++)
    {
        printf("eyesid: %d, \ttype: %d, \ttype_description: %s \tstatus: %d, \tcX: %.2f,
\tcY: %.2f\n", detectionData.objects[i].id, detectionData.objects[i].type,
detectionData.objects[i].type_description, detectionData.objects[i].status,
detectionData.objects[i].cx, detectionData.objects[i].cy);
    }
    printf("-----------------------------------------------\r\n");
}

// callback mode
int main()
{
    int re = InitAndRun("127.0.0.1", 53607, CALLBACK_MODE, OnDetected);
    if (re != 0)
    {
        printf("InitAndRun Error. Error code:%d\r\n", re);
    }
    getchar();
    ShutdownAndRelease();
}
```

```
//// non callback mode
//int main()
//{
//   int re = InitAndRun("127.0.0.1", 53607, DEFAULT_MODE, 0);
//   if (re != 0)
//   {
//       printf("InitAndRun Error. Error code:%d\r\n", re);
//       getchar();
//       return re;
//   }
//   while (true)
//   {
//       DetectionData detectionData;
//       re = ProcessDetectedData(&detectionData);
//       if(re == 0)
//           OnDetected(detectionData);
//   }
//   ShutdownAndRelease();
//}
```