附錄 B

appendix B

B.1 主要手勢的內部手勢辨識

函式名稱

P5Bend_Init()　判斷 P5 手套彎曲感應器(Bend sensor)是否以在執行
P5Bend_SetClickSensitivity()　判斷彎曲感應器目前的彎曲程度
P5Bend_Process()　執行每個影格(frame)的彎曲感應器彎曲程度之前後比對

B.1.1　手指彎曲初始設定之程式碼

```
int nDebounceCounter[5] = {0, 0, 0, 0, 0};
int nLastFingerValue[5];
unsigned char nClickSpot[5];
void BendsPage::OnTimer(UINT nIDEvent)
{
        unsigned char value;
        if(P5.m_P5Devices != NULL)
        {
                value=((unsigned char)P5.m_P5Devices[0].m_byBendSensor_Data[P5_INDEX]);
                m_progIndex.SetPos(63-value);
                value=((unsigned char)P5.m_P5Devices[0].m_byBendSensor_Data[P5_MIDDLE]);
                m_progMiddle.SetPos(63-value);
                value=((unsigned char)P5.m_P5Devices[0].m_byBendSensor_Data[P5_RING]);
                m_progRing.SetPos(63-value);
                value=((unsigned char)P5.m_P5Devices[0].m_byBendSensor_Data[P5_PINKY]);
                m_progPinky.SetPos(63-value);
                value=((unsigned char)P5.m_P5Devices[0].m_byBendSensor_Data[P5_THUMB]);
                m_progThumb.SetPos(63-value);
                if( this->IsWindowVisible() )
                {
                        if(P5.m_P5Devices[0].m_byButtons[0] && !bButPressed)
                        {
                                if(nResetCtr%2)
                                {
                                        if(P5.m_P5Devices != NULL)
                                        {
                                                P5.P5_SaveBendSensors(0);
                                                m_txtBendsMsg.SetWindowText("Fingers has been Saved");
                                        }
                                }
                                else
                                {
                                        if(P5.m_P5Devices != NULL)
                                        {
                                                P5.P5_CalibrateBendSensors(0);
                                        m_txtBendsMsg.SetWindowText("Make a fist and hit 'A' button on P5 Device");
                                        }
                                }
                          nResetCtr ++;
                          bButPressed = TRUE;
                        }
```

```
                    if(!P5.m_P5Devices[0].m_byButtons[0])
                    {
                              bButPressed = FALSE;
                    }
          }
     }
     CPropertyPage::OnTimer(nIDEvent);
}
```

B.1.2 進行彎曲判斷程式描述

```
BOOL BendsPage::OnInitDialog()
{
     CPropertyPage::OnInitDialog();
     SetTimer(1234, 10, NULL);
     m_progThumb.SetRange(0,63);
     m_progThumb.SetStep(1);
     m_progIndex.SetRange(0,63);
     m_progIndex.SetStep(1);
     m_progMiddle.SetRange(0,63);
     m_progMiddle.SetStep(1);
     m_progRing.SetRange(0,63);
     m_progRing.SetStep(1);
     m_progPinky.SetRange(0,63);
     m_progPinky.SetStep(1);
}
```

B.1.3 彎曲程度之比對的完整程式描述

```
void P5Bend_Process()
{
     static int firsttime = 1;
     if (myP5Devices != NULL)
     {
          if (firsttime==1)
          {
                    for (int i=0; i<5; i++)
                              nLastFingerValue[i] = myP5Devices->m_P5Devices[0].m_byBendSensor_Data[i];
                    firsttime = 0;
          }
          else
          {
                    for (int i=0; i<5; i++)
                    {
                    int delta = myP5Devices->m_P5Devices[0].m_byBendSensor_Data[i]-nLastFingerValue[i];
                              nP5ClickEdge[i] = 0;
                              if (bP5ClickLevel[i]==false)
                              {
                                        if (delta>nBendSensitivity[i])
                                        {
                                                  bP5ClickLevel[i] = true;
                                                  nP5ClickEdge[i] = 1;
                                        nClickSpot[i] = myP5Devices->m_P5Devices[0].m_byBendSensor_Data[i];
                                        }
                              }
                              else
                              {
```

```
                    //If finger is already pressed, see if it is unclicked
                    if (myP5Devices->m_P5Devices[0].m_byBendSensor_Data[i]<nClickSpot[i])
                    {
                        //Finger returned to old position, so unclick it
                        bP5ClickLevel[i] = false;
                        nP5ClickEdge[i] = -1;
                    }
                }
                nLastFingerValue[i] = myP5Devices->m_P5Devices[0].m_byBendSensor_Data[i];
            }
        }
    }
}
```

B.2 主要手勢的外部手勢辨識

函式名稱

P5Motion_Init()	判斷 P5 手套是否在偵測範圍爲內
P5Motion_FilterXYZ()	取得 P5 手套的空間參數
P5Motion_FilterYPR ()	取得 P5 手套的旋轉參數
P5Motion_SetClipRegion()	定義游標的範圍
P5Motion_InvertMouse()	定義對游標的空間軸線位置
P5Motion_Process()	執行每個影格(frame)的感應器(sensor)前後位置比對

宣告游標位置的參數　X,Y,Z

```
int nXPos = 0, nYPos = 0, nZPos = 0;
```

宣告游標位置的的初始更新率

```
float fXMickey = 0.0f, fYMickey = 0.0f, fZMickey = 0.0f;
```

設定手掌位置的偵測範圍

```
int nxclipstart = 0, nxclipend = 1024, nyclipstart = 0, nyclipend = 1024, nzclipstart = 0, nzclipend = 1024;
void P5Motion_SetClipRegion(int xstart, int xend, int ystart, int yend, int zstart, int zend)
{
    nxclipstart = xstart;
    nxclipend = xend;
    nyclipstart = ystart;
    nyclipend = yend;
    nzclipstart = zstart;
    nzclipend = zend;
}
```

設定空間三個軸項基準位址

```
int nxinvert = 0, nyinvert = 0, nzinvert = 0;
void P5Motion_InvertMouse(int xaxis, int yaxis, int zaxis)
{
    nxinvert = xaxis;
    nyinvert = yaxis;
    nzinvert = zaxis;
}
```

B.2.1 如何取得 P5 手套空間參數的完整程式描述

```
float fXPos[P5MOTION_XYZFILTERSIZE], fYPos[P5MOTION_XYZFILTERSIZE], fZPos[P5MOTION_XYZFILTERSIZE];
float fFilterX, fFilterY, fFilterZ;
void P5Motion_FilterXYZ()
{
    static int firsttime = 1;
    if (myP5Device != NULL)
```

```
        {
                if (firsttime==1)
                {
                        for (int i=0; i<P5MOTION_XYZFILTERSIZE; i++)
                        {
                                fXPos[i] = myP5Device->m_P5Devices[bP5Id].m_fx;
                                fYPos[i] = myP5Device->m_P5Devices[bP5Id].m_fy;
                                fZPos[i] = myP5Device->m_P5Devices[bP5Id].m_fz;
                        }
                        firsttime = 0;
                }
                else
                {
                        #define FLUSH_SETPOINT  30.0f
                        float xflushsize, yflushsize, zflushsize;
                        int i, j;
xflushsize = fabs(myP5Device->m_P5Devices[bP5Id].m_fx - fXPos[P5MOTION_XYZFILTERSIZE-1])/2.0f;
                        xflushsize *= P5MOTION_XYZFILTERSIZE/FLUSH_SETPOINT;
                        xflushsize = floor(xflushsize+1.0f);
                        if (xflushsize>(P5MOTION_XYZFILTERSIZE-1))
                                xflushsize = P5MOTION_XYZFILTERSIZE-1;
                        yflushsize = fabs(myP5Device->m_P5Devices[bP5Id].m_fy -
                      fYPos[P5MOTION_XYZFILTERSIZE-1])/2.0f;
                        yflushsize *= P5MOTION_XYZFILTERSIZE/FLUSH_SETPOINT;
                        yflushsize = floor(yflushsize+1.0f);
                        if (yflushsize>(P5MOTION_XYZFILTERSIZE-1))
                       yflushsize = P5MOTION_XYZFILTERSIZE-1;
                        zflushsize = fabs(myP5Device->m_P5Devices[bP5Id].m_fz -
                      fZPos[P5MOTION_XYZFILTERSIZE-1])/2.0f;
                        zflushsize *= P5MOTION_XYZFILTERSIZE/FLUSH_SETPOINT;
                        zflushsize = floor(zflushsize+1.0f);
                        if (zflushsize>(P5MOTION_XYZFILTERSIZE-1))
                                zflushsize = P5MOTION_XYZFILTERSIZE-1;
                        for (j=0; j<(int)(xflushsize); j++)
                        {
                                for (i=0; i<(P5MOTION_XYZFILTERSIZE-1); i++)
                                {
                                        fXPos[i] = fXPos[i+1];
                                }
                                fXPos[P5MOTION_XYZFILTERSIZE-1] =
                      myP5Device->m_P5Devices[bP5Id].m_fx;
                        }
                        for (j=0; j<(int)(yflushsize); j++)
                        {
                                for (i=0; i<(P5MOTION_XYZFILTERSIZE-1); i++)
                                {
                                        fYPos[i] = fYPos[i+1];
                                }
                                fYPos[P5MOTION_XYZFILTERSIZE-1] =
                      myP5Device->m_P5Devices[bP5Id].m_fy;
                        }
                        for (j=0; j<(int)(zflushsize); j++)
                        {
                                for (i=0; i<(P5MOTION_XYZFILTERSIZE-1); i++)
                                {
                                        fZPos[i] = fZPos[i+1];
                                }
                                fZPos[P5MOTION_XYZFILTERSIZE-1] = myP5Device->m_P5Devices[bP5Id].m_fz;
                }
            }
            fFilterX = 0.0f;
            fFilterY = 0.0f;
            fFilterZ = 0.0f;
            for (int i=0; i<P5MOTION_XYZFILTERSIZE; i++)
            {
                fFilterX += fXPos[i]/2.0f;
                fFilterY += fYPos[i]/2.0f;
                fFilterZ += fZPos[i]/2.0f;
            }
            fFilterX /= P5MOTION_XYZFILTERSIZE;
            fFilterY /= P5MOTION_XYZFILTERSIZE;
            fFilterZ /= P5MOTION_XYZFILTERSIZE;
        }
}
```

B.2.2 如何取得 P5 手套旋轉角度的完整程式描述

```
float fYaw[P5MOTION_YPRFILTERSIZE], fPitch[P5MOTION_YPRFILTERSIZE], fRoll[P5MOTION_YPRFILTERSIZE];
float fFilterYaw, fFilterPitch, fFilterRoll;
void P5Motion_FilterYPR()
{
        static int firsttime = 1;
        if (myP5Device != NULL)
        {
            if (firsttime==1)
            {
                for (int i=0; i<P5MOTION_YPRFILTERSIZE; i++)
                {
                    fYaw[i] = myP5Device->m_P5Devices[bP5Id].m_fyaw;
                    fPitch[i] = myP5Device->m_P5Devices[bP5Id].m_fpitch;
                    fRoll[i] = myP5Device->m_P5Devices[bP5Id].m_froll;
                }
                firsttime = 0;
            }
            else
            {
                for (int i=0; i<(P5MOTION_YPRFILTERSIZE-1); i++)
                {
                    fYaw[i] = fYaw[i+1];
                    fPitch[i] = fPitch[i+1];
                    fRoll[i] = fRoll[i+1];
                }
                fYaw[P5MOTION_YPRFILTERSIZE-1] = myP5Device->m_P5Devices[bP5Id].m_fyaw;
                fPitch[P5MOTION_YPRFILTERSIZE-1] = myP5Device->m_P5Devices[bP5Id].m_fpitch;
                fRoll[P5MOTION_YPRFILTERSIZE-1] = myP5Device->m_P5Devices[bP5Id].m_froll;
            }
            fFilterYaw = 0.0f;
            fFilterPitch = 0.0f;
            fFilterRoll = 0.0f;
            for (int i=0; i<P5MOTION_YPRFILTERSIZE; i++)
            {
                fFilterYaw += fYaw[i];
                fFilterPitch += fPitch[i];
                fFilterRoll += fRoll[i];
            }
            fFilterYaw /= P5MOTION_YPRFILTERSIZE;
            fFilterPitch /= P5MOTION_YPRFILTERSIZE;
            fFilterRoll /= P5MOTION_YPRFILTERSIZE;
        }
}
```

B.2.3 感應器前後位置比對的完整程式描述

```
void P5Motion_Process()
{
    static float fLastXpos = 0.0f, fLastYpos = 0.0f, fLastZpos = 0.0f;

    P5Motion_FilterXYZ();
    if ((myP5Device->m_P5Devices[0].m_byBendSensor_Data[P5_RING]<30) &&
        (myP5Device->m_P5Devices[0].m_byBendSensor_Data[P5_PINKY]<30))
    {
        //apply axis invertion if required and calculate the delta from last frame
        fXMickey = (fLastXpos - fFilterX) * nxinvert;
        fYMickey = (fLastYpos - fFilterY) * nyinvert;
        fZMickey = (fLastZpos - fFilterZ) * nzinvert;
        #define COEFF1      0.0042f
        #define COEFF2      1.2403f
        fXMickey = COEFF1*fXMickey*fXMickey*fXMickey +
     COEFF2*fXMickey;
        fYMickey = COEFF1*fYMickey*fYMickey*fYMickey +
     COEFF2*fYMickey;
        fZMickey = COEFF1*fZMickey*fZMickey*fZMickey +
     COEFF2*fZMickey;
        if ( fabs(fXMickey) > 3.0f)
            fXMickey *= 4.0f;
        else
            fXMickey *= 2.0f;
        if ( fabs(fYMickey) > 3.0f)
            fYMickey *= 4.0f;
        else
            fYMickey *= 2.0f;
        if ( fabs(fZMickey) > 3.0f)
            fZMickey *= 4.0f;
        else
            fZMickey *= 2.0f;
        nXPos += (int)(fXMickey);
        nYPos += (int)(fYMickey);
        nZPos += (int)(fZMickey);
        if (nXPos > nxclipend)
            nXPos = nxclipend;
        else if (nXPos < nxclipstart)
            nXPos = nxclipstart;
        if (nYPos > nyclipend)
            nYPos = nyclipend;
        else if (nYPos < nyclipstart)
            nYPos = nyclipstart;
        if (nZPos > nzclipend)
            nZPos = nzclipend;
        else if (nZPos < nzclipstart)
            nZPos = nzclipstart;
    }
    fLastXpos = fFilterX;
```

```
        fLastYpos = fFilterY;
        fLastZpos = fFilterZ;

      P5Motion_FilterYPR();
        static bool bgrabstate = false;
        static float fZeroYawPos, fZeroPitchPos, fZeroRollPos;
        #define BEND_THRESHOLD 20
        if ((myP5Device->m_P5Devices[0].m_byBendSensor_Data[P5_THUMB]>BEND_THRESHOLD)&&
                (myP5Device->m_P5Devices[0].m_byBendSensor_Data[P5_INDEX]>BEND_THRESHOLD) &&
                (myP5Device->m_P5Devices[0].m_byBendSensor_Data[P5_MIDDLE]>BEND_THRESHOLD)&&
                (myP5Device->m_P5Devices[0].m_byBendSensor_Data[P5_RING]>BEND_THRESHOLD) &&
                (myP5Device->m_P5Devices[0].m_byBendSensor_Data[P5_PINKY]>BEND_THRESHOLD))
        {
                if (bgrabstate == false)
                {
                        bgrabstate = true;
                        fZeroYawPos = fFilterYaw;
                        fZeroPitchPos = fFilterPitch;
                        fZeroRollPos = fFilterRoll;
                }
        }
        else
        {
                bgrabstate = false;
        }
        if (bgrabstate == true)
        {
                fAbsYawPos = fFilterYaw;
                fAbsPitchPos = fFilterPitch;
                fAbsRollPos = fFilterRoll;
                #define YPR_ROTSPEED      0.5f
                if ( fFilterRoll > (fZeroRollPos+30.0f) )
                        fRelRollPos += YPR_ROTSPEED;
                else if ( fFilterRoll<(fZeroRollPos-30.0f) )
                        fRelRollPos -= YPR_ROTSPEED;
                if (fRelRollPos > 180.0f)
                        fRelRollPos = -180.0f;
                else if (fRelRollPos < -180.0f)
                        fRelRollPos = 180.0f;
                if (fFilterYaw > (fZeroYawPos+25.0f))
                        fRelYawPos += YPR_ROTSPEED;
                else if (fFilterYaw < (fZeroYawPos-25.0f))
                        fRelYawPos -= YPR_ROTSPEED;
                if (fRelYawPos > 180.0f)
                        fRelYawPos = -180.0f;
                else if (fRelYawPos < -180.0f)
                        fRelYawPos = 180.0f;
                if (fFilterPitch > (fZeroPitchPos+25.0f))
                        fRelPitchPos += YPR_ROTSPEED;
                else if (fFilterPitch < (fZeroPitchPos-25.0f))
                        fRelPitchPos -= YPR_ROTSPEED;
                if (fRelPitchPos > 180.0f)
                        fRelPitchPos = -180.0f;
                else if (fRelPitchPos < -180.0f)
                        fRelPitchPos = 180.0f;
}
}
```