玩家編號與入侵偵測

王泓權
臺科大人工智慧研究中心 助理教授

玩家編號與入侵偵測

- Kinect SDK 可以分析 Kinect 感應器傳回的深度影像資料，辨識出其中屬於人體形狀的像素，而且同時可以辨識6個人體，並為這些人體提供玩家編號(1~6)，此編號放在深度像素的前3個位元
- 利用 DepthImageFrame.PlayerIndexBitmask 為位元遮罩，就能從深度資料中擷取出此玩家編號加以利用，但是要同時啟用深度串流(DepthImageStream)與骨架串流(SkeletonStream)，玩家編號才會出現在深度資料中

玩家編號與入侵偵測

- 這個範例中，將以不同顏色標示出深度影像中的玩家，並在玩家身體任何一部為接近到距 Kinect 感應器1.5公尺以內的時候，將這部分像素改為紅色顯示以作為警告
- Step 01: 建立新專案-NearWarning，在專案中增加對 Kinect SDK 參考
- Step 02: 使用者介面、MainWindow()、KinectSeneors_StatuaChanged()、UninitialStream()、Window_Loade_1()

玩家編號與入侵偵測

- Step 03: 在 InitialStream() 函式內新增 Kinect 感應器提供骨架串流的程式

```
this._myKinect.SkeletonStream.Enable();
```

玩家編號與入侵偵測

Step 04: 影格備妥事件處理常式

```
        private KinectSensor _myKinect;
        private WriteableBitmap _writeableBitmap; //定義Image影像來源的WriteableBitmap物件
        private Int32Rect _imageRect;                         //定義影像區域矩形
        private byte[] _colorDataByte;                        //定義影像暫存資料陣列
        private int _warningDistance = 1500;          //警告距離
        //影格備妥事件處理常式-
        private void Kinect_DepthFrameReady(object sender, DepthImageFrameReadyEventArgs
e)
        {
            using (DepthImageFrame frameData = e.OpenDepthImageFrame())  //取得傳遞的影格資
料
            {
                if (frameData == null)  //如果影格資料不存在,直接離開事件處理函式
                {
                    return;
                }
```

玩家編號與入侵偵測

```
               //定義影像暫存資料陣列
              short[] depthDataArray = new
short[this._myKinect.DepthStream.FramePixelDataLength];
              //將影格資料複製到資料陣列
              frameData.CopyPixelDataTo(depthDataArray);

              int brightPosition = 0;
              for (int i = 0; i < depthDataArray.Length; i++)
              {
                 int playerNumber = depthDataArray[i] & DepthImageFrame.PlayerIndexBitmask;
                 int depth = depthDataArray[i] >> DepthImageFrame.PlayerIndexBitmaskWidth;  //
                 //if (playerNumber != 0)   //(1)
                 if (depth > 0 && playerNumber != 0)   //(1)
                 {
                    if (depth < _warningDistance)  //進入警告範圍的像素以紅色顯示
                    {
                       this._colorDataByte[brightPosition] = 0;
                       this._colorDataByte[brightPosition + 1] = 0;
                       this._colorDataByte[brightPosition + 2] = 255;
                    }
```

玩家編號與入侵偵測

```
else //進入警告範圍的像素依玩家編號顯示不同顏色
{
   switch (playerNumber)
   {
      case 1:  //藍色
         this._colorDataByte[brightPosition] = 255;
         this._colorDataByte[brightPosition + 1] = 0;
         this._colorDataByte[brightPosition + 2] = 0;
         break;
      case 2:   //綠色
         this._colorDataByte[brightPosition] = 0;
         this._colorDataByte[brightPosition + 1] = 255;
         this._colorDataByte[brightPosition + 2] = 0;
         break;
      case 3:   //青色
         this._colorDataByte[brightPosition] = 255;
         this._colorDataByte[brightPosition + 1] = 255;
         this._colorDataByte[brightPosition + 2] = 0;

         break;
```

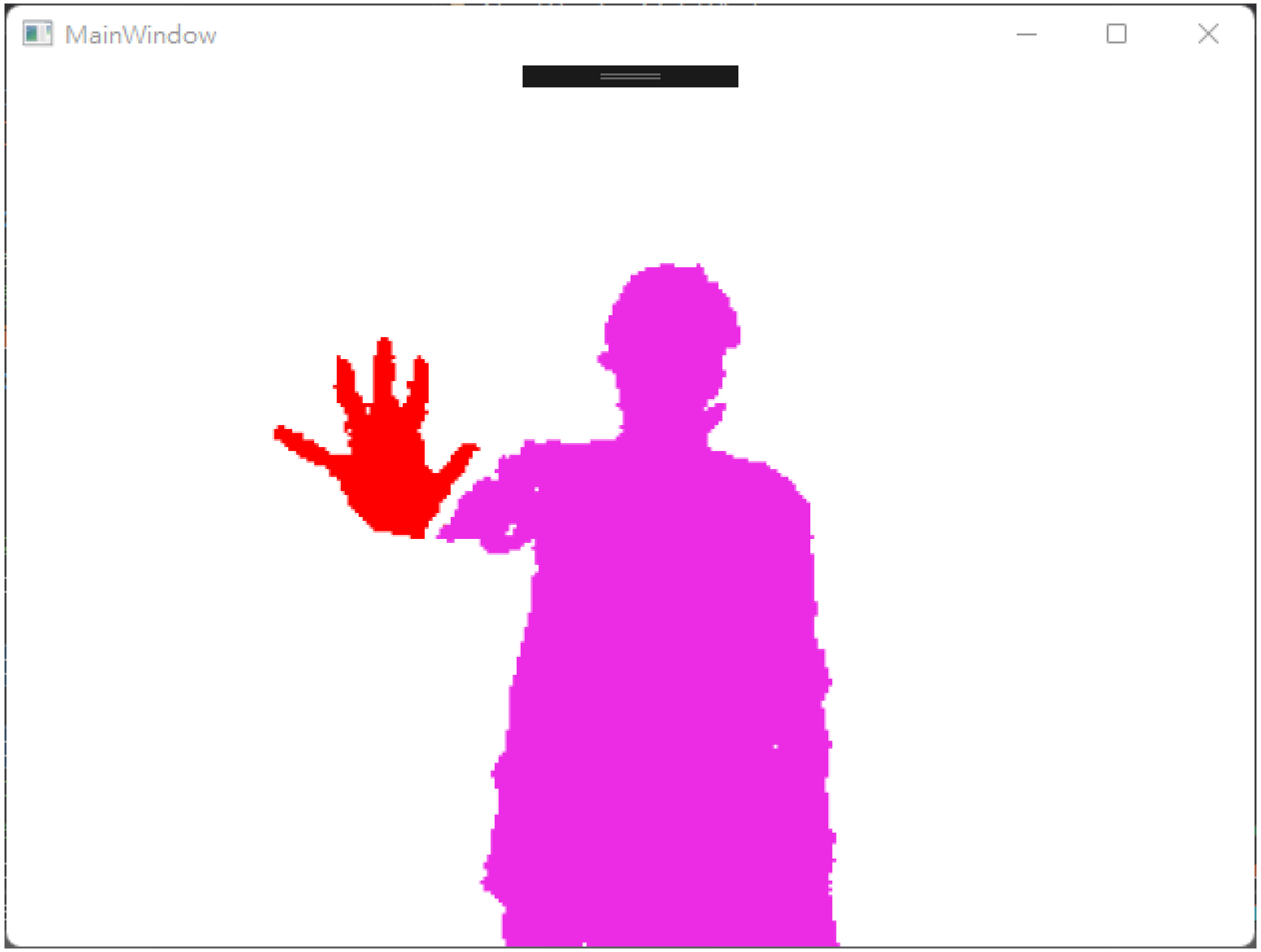
玩家編號與入侵偵測

```
                    case 4:  //黃色
                        this._colorDataByte[brightPosition] = 0;
                        this._colorDataByte[brightPosition + 1] = 255;
                        this._colorDataByte[brightPosition + 2] = 255;
                        break;
                    case 5:  //紫色
                        this._colorDataByte[brightPosition] = 229;
                        this._colorDataByte[brightPosition + 1] = 44;
                        this._colorDataByte[brightPosition + 2] = 236;
                        break;
                    case 6:  //橘色
                        this._colorDataByte[brightPosition] = 236;
                        this._colorDataByte[brightPosition + 1] = 135;
                        this._colorDataByte[brightPosition + 2] = 44;
                        break;
                }
            }
        }
```

玩家編號與入侵偵測

```
                else  //無玩家編號像素以白色顯示
                {
                    this._colorDataByte[brightPosition] = 255; //B
                    this._colorDataByte[brightPosition + 1] = 255; //G
                    this._colorDataByte[brightPosition + 2] = 255; //R;
                }
                brightPosition += 4;
            } // for
            this._writeableBitmap.WritePixels(this._imageRect, this._colorDataByte,
frameData.Width * 4, 0);
        }
    }
```

測試


MainWindow