# Kinect 深度資訊應用

王泓權
臺科大人工智慧研究中心 助理教授

# 大綱

- 深度影像初體驗
- Kinect 相關類別或成員介紹
- 深度資料介紹
- 深度量測範例

# 深度影像初體驗

- Kinect 感應器的深度影像串流是由 DepthImageStream 物件控制
- 深度影格資料存於 DepthImageFrame 物件

# 深度影像初體驗

- Step 01: 建立新專案-DepthImageDisplay
- Step 02: 建立使用者介面

```
<Window x:Class="DepthImageDisplay.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="480" Width="640" Loaded="Window_Loaded_1">
    <Grid>
        <Image x:Name="DepthImage"/>
    </Grid>
</Window>
```

# 深度影像初體驗

- Step 03: 初始化作業，登錄 StatusChanged 事件及定義其事件處理常式 KinectSensor_StatusChanged
- Step 04: 串流初始化、啟動與停止作業

# 深度影像初體驗

```
        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 short[] _imageDataByte;            //定義影像暫存資料陣列
        //InitialStream函式
        private void InitialStream()
        {
            this._myKinect.DepthStream.Enable();      //要求Kinect感應器產生深度資料串流
            //建立WriteableBitmap物件及影像區域
            this._writeableBitmap = new WriteableBitmap(_myKinect.DepthStream.FrameWidth,
_myKinect.DepthStream.FrameHeight, 96, 96, PixelFormats.Gray16, null);
            this._imageRect = new Int32Rect(0, 0, _myKinect.DepthStream.FrameWidth,
_myKinect.DepthStream.FrameHeight);
            //指定Image控制項影像來源為WriteableBitmap物件,使Image控制項顯示內容隨
WriteableBitmap物件內容改變
            DepthImage.Source = this._writeableBitmap;
            //用影格資料像素資料大小)定義暫存資料陣列長度
            _imageDataByte = new short[this._myKinect.DepthStream.FramePixelDataLength];
            //註冊Kinect_DepthFrameReady事件處理函式
            this._myKinect.DepthFrameReady += Kinect_DepthFrameReady;
            this._myKinect.Start();  //啟動Kinect感應器硬體
        }
```

# 深度影像初體驗

```
//UninitialStream函式
private void UninitialStream()
{
   //取消ColorFrameReady事件註冊
   this._myKinect.DepthFrameReady -= Kinect_DepthFrameReady;
   this._myKinect.Stop();  //停止Kinect感應器硬體工作
   this._myKinect = null;  //取消對Kinect感應器硬體參考
}
```

- 在程式原 using 敘述中增加以下敘述

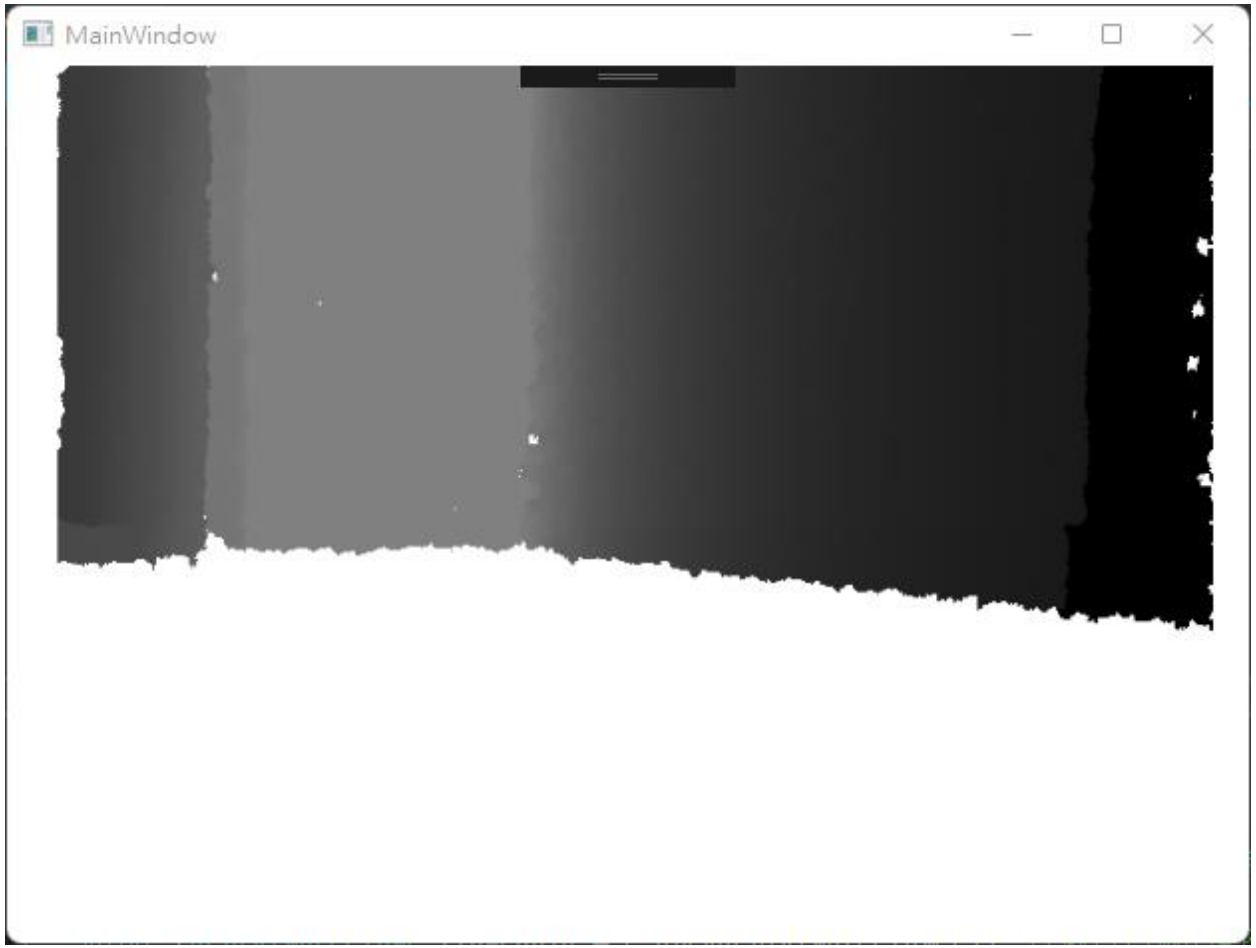
```
using System.Windows.Media.Imaging;
```

# 深度影像初體驗

- Step05: 深度影格備妥事件處理常式

```
    //影格備妥事件處理常式
    private void Kinect_DepthFrameReady(object sender, DepthImageFrameReadyEventArgs
e)
    {
        //取得傳遞的影格資料
        using (DepthImageFrame frameData = e.OpenDepthImageFrame())
        {
            if (frameData == null)  //如果影格資料不存在,直接離開事件處理函式
            {
                return;
            }
            //將影格資料複製到資料陣列
            frameData.CopyPixelDataTo(this._imageDataByte);
            this._writeableBitmap.WritePixels(this._imageRect, this._imageDataByte,
frameData.Width * frameData.BytesPerPixel, 0);
        }
    }
```

# 測試



# Kinect 相關類別或成員介紹

- KinectSensor
  - DepthStream: 屬性
  - MapDepthFrameToColorFrame: 方法
  - MapDepthToColorImagePoint: 方法
  - MapDepthToSkeletonPoint: 方法
  - MapSkeletonPointToColor: 方法
  - MapSkeletonPointToDepth: 方法
  - DepthFrameReady: 事件
  - SkeletonFrameReady: 事件

# Kinect 相關類別或成員介紹

- DepthImageFrameReadyEventArgs
  - OpenDepthImageFrame: 方法
- SkeletonFrameReadyEventArges
  - OpenSkletonFrame: 方法
- DepthImageStream
  - Format: 屬性
  - MaxDepth: 屬性
  - MinDepth: 屬性
  - NominalDiagonalFieldOfView: 屬性

# Kinect 相關類別或成員介紹

- DepthImageStream
  - NominalFocalLengthInPixels: 屬性
  - NominalHorizontalFieldOfView: 屬性
  - NominalVerticalFieldOfView: 屬性
  - Range: 屬性
  - TooFarDepth: 屬性
  - TooNearDepth: 屬性
  - UnKnownDepth: 屬性
  - Enable: 方法
  - OpenNextFrame: 方法

# Kinect 相關類別或成員介紹

- DepthImageFrame
  - PlayerIndexBitmask: 欄位
  - PlayerIndexBitmaskWidth: 欄位
  - Format: 屬性
  - PixelDataLength: 屬性
  - CopyPixelDataTo: 方法
  - MapFromSkeletonPoint: 方法
  - MapToColorImagePoint: 方法
  - MapToSkeletonPoint: 方法

# 深度資料介紹

- Kinect 感應器提供了第3個維度(深度)的資料，讓程式設計者設計空間變得更大
- Kinect SDK 並未提供直接取得紅外線資料串流的能力，而是提供由 Kinect SDK 處理紅外線產生的深度影像
- DepthImageStream(深度影像資料串流)與DepthImageFrame(深度影格)

# 深度資料介紹

- 在深度影格中，每一個像素雖然佔了16個位元(格是為Gray16)，但實際包含的深度值只佔其中較高的13個位元，最低的3個位元存放的是玩家編號(玩家是指易經被辨識的人體)

# 深度資料介紹

- 要得到一個像素的深度值，可以透過位元右移運算子(>>) DepthImageFrame物件的PlayerIndexBitmaskWidth 欄位，將玩家編號位元移除

```
depth=depthImagePixelArray[pixel]>>
    DepthImageFrame.PlayerIndexBitmaskWidth;
```

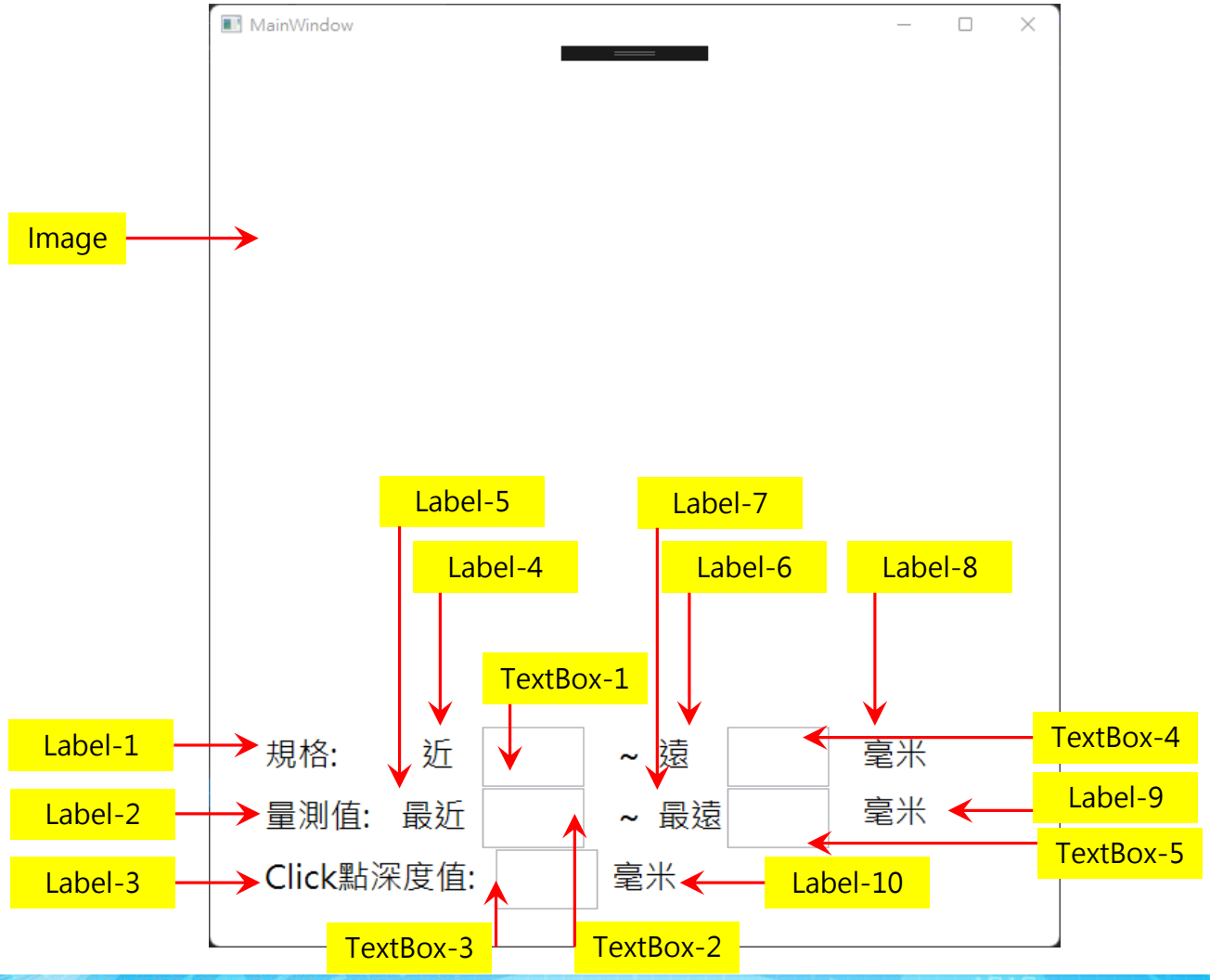
# 深度資料介紹

- Kinect 感應器正常狀態下深度感測範圍是從800毫米到4000毫米
- 深度影像是以灰階呈現，距離 Kinect 感應器越近(深度值越小)，像素顏色越暗，距離越遠像素灰階越淡(越接近白色)
- Kinect SDK 最多可以辨識出6位玩家
  - 未啟用骨架串流(SkeletonStream)，深度資料中的玩家編號一律為0
  - 玩家編號只是用以區分 Kinect 感應器視野範圍內不同人體，本身不具備辨識玩家的功能
  - 在啟用骨架串流情況下，玩家編號為0的像素，表示此深度像素不屬於任何玩家

# 深度量測範例

- Step 01: 建立新專案-DepthMeasurement
- Step 02: 建立使用者介面，1個 Image 控制項、10個 Label 控制項、5個 TextBox 控制項

# 深度量測範例


MainWindow
Image
Label-5
Label-7
Label-4
Label-6
Label-8
TextBox-1
Label-1
規格:
近
~ 遠
毫米
TextBox-4
Label-2
量測值:
最近
~ 最遠
毫米
Label-9
TextBox-5
Label-3
Click點深度值:
毫米
Label-10
TextBox-3
TextBox-2

# 深度量測範例

- Step 03:初始化作業，登錄 StatusChanged 事件及定義其事件處理常式 KinectSensor_StatusChanged
- Step 04: 串流初始化、啟動與停止作業

# 深度量測範例

```
        //InitialStream函式
        private void InitialStream()
        {
            this._myKinect.DepthStream.Enable();  //要求Kinect感應器產生深度資料串流
            //建立WriteableBitmap物件及影像區域
            this._writeableBitmap = new WriteableBitmap(_myKinect.DepthStream.FrameWidth,
_myKinect.DepthStream.FrameHeight, 96, 96, PixelFormats.Gray16, null);
            this._imageRect = new Int32Rect(0, 0, _myKinect.DepthStream.FrameWidth,
_myKinect.DepthStream.FrameHeight);
            //指定Image控制項影像來源為WriteableBitmap物件,使Image控制項顯示內容隨
WriteableBitmap物件內容改變
            DepthImage.Source = this._writeableBitmap;
            //用影格資料像素資料大小)定義暫存資料陣列長度
            _imageDataArray = new short[this._myKinect.DepthStream.FramePixelDataLength];
            //註冊Kinect_DepthFrameReady事件處理函式
            this._myKinect.DepthFrameReady += Kinect_DepthFrameReady;
            this._myKinect.Start();  //啟動Kinect感應器硬體
            //以TextBox控制項顯示Kinect感應器最近與最遠感測範圍
            SNearTxt.Text=this._myKinect.DepthStream.MinDepth.ToString();
            SFarTxt.Text = this._myKinect.DepthStream.MaxDepth.ToString();
        }
```

# 深度量測範例

## Step 05: 深度影格備妥事件處理常式

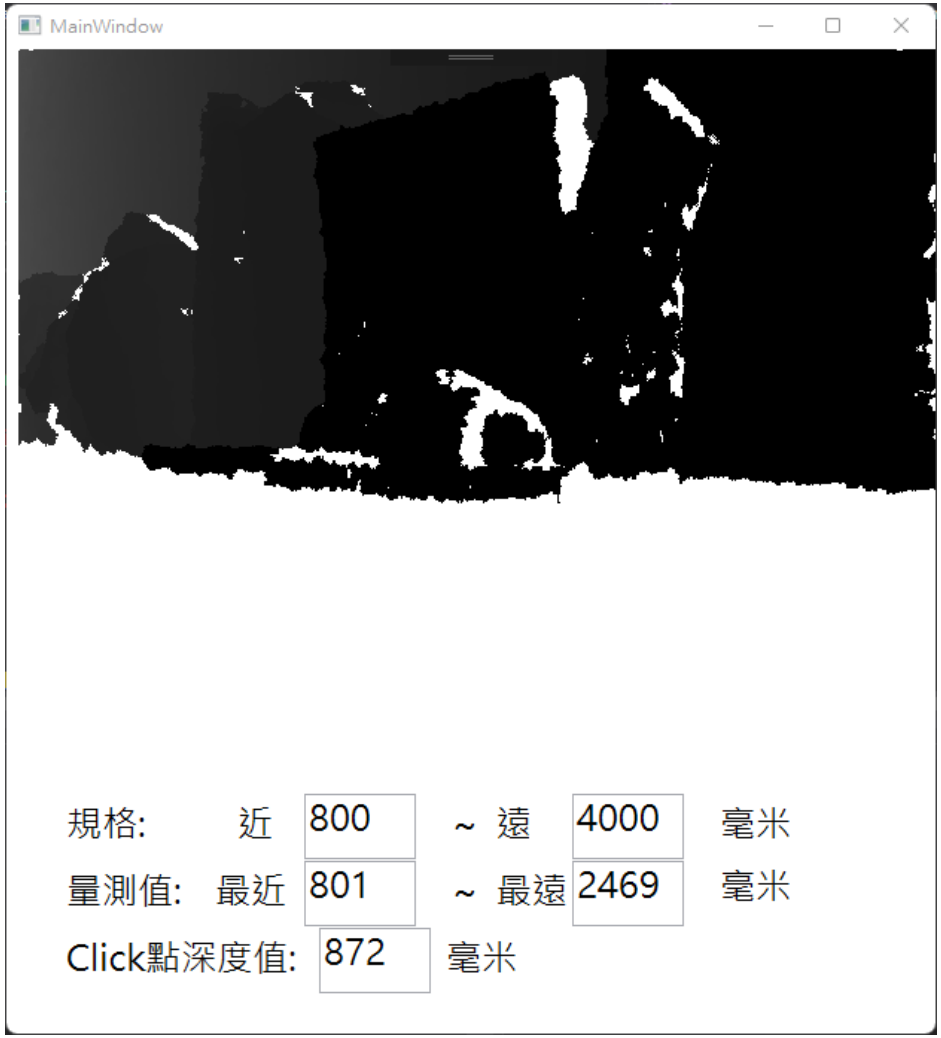
```
//影格備妥事件處理常式
private void Kinect_DepthFrameReady(object sender, DepthImageFrameReadyEventArgs e)
{
    int minlDepth = 4000;  //預設最近深度
    int maxDepth = 0;     //預設最遠深度
    using (DepthImageFrame frameData = e.OpenDepthImageFrame())  //取得傳遞的影格資料
    {
        if (frameData == null)  //如果影格資料不存在,直接離開事件處理函式
        {
            return;
        }
        //將影格資料複製到資料陣列
        frameData.CopyPixelDataTo(this._imageDataArray);
        //找出目前深度影格中最近與最遠的深度
        for (int i = 0; i < this._imageDataArray.Length; i++)
        {
            int depthValue = this._imageDataArray[i] >> DepthImageFrame.PlayerIndexBitmaskWidth; //取得目前像素深度資料
            //判斷目前像素深度是否為新的最近或最遠深度
            if (depthValue > 0 && depthValue < minlDepth) minlDepth = depthValue;
            if (depthValue > maxDepth) maxDepth = depthValue;
        }
        this._writeableBitmap.WritePixels(this._imageRect, this._imageDataArray, frameData.Width * frameData.BytesPerPixel,
0);
        //以TextBox控制項顯示目前深度影格中最近與最遠的深度
        MNearTxt.Text = minlDepth.ToString();
        MFarTxt.Text = maxDepth.ToString();
    }
}
```

# 深度量測範例

- Step 06: 用滑鼠點擊深度影像，指定要量測深度值的位置

```
        //滑鼠按下左鈕事件處理常式
        private void DepthImage_MouseLeftButtonDown(object sender,
System.Windows.Input.MouseButtonEventArgs e)
        {
            Point p = e.GetPosition(DepthImage);  //取得滑鼠敲擊點
            //如果資料陣列存有深度影像資料
            if (this._imageDataArray != null && this._imageDataArray.Length > 0)
            {
                int width = _myKinect.DepthStream.FrameWidth;  //取得深度影格寬度
                int arrayIndex = (int)(p.X + ((int)p.Y * width)); //計算滑鼠敲擊點在深度影像資料陣列的索
引
                int depth = this._imageDataArray[arrayIndex] >>
DepthImageFrame.PlayerIndexBitmaskWidth;  //取得滑鼠敲擊點深度資料
                DepthTxt.Text = depth.ToString();  //顯示滑鼠敲擊點深度值
            }
        }
```

# 測試


MainWindow
規格: 近 800 ~ 遠 4000 毫米
量測值: 最近 801 ~ 最遠 2469 毫米
Click點深度值: 872 毫米