How to video360 for dummies

# Intro

[Zet om in engels – moet da in het engels?]

Het project bestaat ten eerst uit de sensor, de Kinect. Deze sensor registreert de handelingen en stuurt ze vervolgens door naar het programma. Het programma zal deze handelingen omzetten in signalen voor de Raspberry Pi, welke de camera aanstuurt. De beelden worden via de Raspberry Pi teruggestuurd naar het programma en getoond op het beeldscherm. Dit alles gebeurd over een intern netwerk.

Een eerste uitbreiding die we hiervoor hebben is dat we over het internet zouden werken. Een 2de uitbreiding die verwezenlijkt kan worden is vanaf dat we een werkende module over het internet hebben, er meerdere samenstellen zodat de gebruiker vanop één locatie meerdere plaatsen kan bezichtigen.

# Equipment

[Zet om in engels – moet da in het engels?]

De te gebruiken ontwikkeltaal voor de Kinect is C#, het programma zal worden geschreven in Visual Studio 2012. De code zal bijgehouden worden in een gezamenlijke GIT. Hierin zal ook de code voor de Raspberry Pi module te vinden zijn. Dit deel bestaat dan uit Linux commando’s.

+

Kinect, sdk Kinect, raspberry pi beta, video module, stappenmotor, visual studio 2012

# Set-up

Fotos + hoe maak je alles aan elkaar vast

# Raspberry Pi

## What is Raspberry Pi

## Why use Raspberry Pi

## Code

## Commands and used programs

# C#

## What is C#

## Why use C#

## Code

I started from SkeletonBasics, a tutorial provided by Microsoft itself.

Altered or self-written code:

/// <summary>

/// Event handler for Kinect sensor's SkeletonFrameReady event

/// </summary>

/// <param name="sender">object sending the event</param>

/// <param name="e">event arguments</param>

private void SensorSkeletonFrameReady(object sender, SkeletonFrameReadyEventArgs e)

{

Skeleton[] skeletons = new Skeleton[0];

using (SkeletonFrame skeletonFrame = e.OpenSkeletonFrame())

{

if (skeletonFrame != null)

{

skeletons = new Skeleton[skeletonFrame.SkeletonArrayLength];

skeletonFrame.CopySkeletonDataTo(skeletons);

}

}

using (DrawingContext dc = this.drawingGroup.Open())

{

// Draw a transparent background to set the render size

dc.DrawRectangle(Brushes.Black, null, new Rect(0.0, 0.0, RenderWidth, RenderHeight));

if (skeletons.Length != 0)

{

foreach (Skeleton skel in skeletons)

{

RenderClippedEdges(skel, dc);

if (skel.TrackingState == SkeletonTrackingState.Tracked)

{

// Create the thread object, passing in the Alpha.Beta method

// via a ThreadStart delegate. This does not start the thread.

this.DrawBonesAndJoints(skel, dc);

this.CheckDirection(skel, dc);

}

else if (skel.TrackingState == SkeletonTrackingState.PositionOnly)

{

dc.DrawEllipse(

this.centerPointBrush,

null,

this.SkeletonPointToScreen(skel.Position),

BodyCenterThickness,

BodyCenterThickness);

}

}

}

// prevent drawing outside of our render area

this.drawingGroup.ClipGeometry = new RectangleGeometry(new Rect(0.0, 0.0, RenderWidth, RenderHeight));

}

}

/// <summary>

/// Handles the checking or unchecking of the seated mode combo box

/// </summary>

/// <param name="sender">object sending the event</param>

/// <param name="e">event arguments</param>

private void CheckBoxSeatedModeChanged(object sender, RoutedEventArgs e)

{

if (null != this.sensor)

{

if (this.checkBoxSeatedMode.IsChecked.GetValueOrDefault())

{

this.sensor.SkeletonStream.TrackingMode = SkeletonTrackingMode.Seated;

}

else

{

this.sensor.SkeletonStream.TrackingMode = SkeletonTrackingMode.Default;

}

}

}

public static DateTime lastHit = new DateTime();

public static Boolean mSend = false;

/// <summary>

/// Custom function - check direction hands

/// </summary>

/// <param name="skeleton">skeleton to draw</param>

/// <param name="drawingContext">drawing context to draw to</param>

private void CheckDirection(Skeleton skeleton, DrawingContext drawingContext)

{

Point handLeft = new Point(skeleton.Joints[JointType.HandLeft].Position.X, skeleton.Joints[JointType.HandLeft].Position.Y);

Point handRight = new Point(skeleton.Joints[JointType.HandRight].Position.X, skeleton.Joints[JointType.HandRight].Position.Y);

Point hipCenter = new Point(skeleton.Joints[JointType.HipCenter].Position.X, skeleton.Joints[JointType.HipCenter].Position.Y);

Thread t1 = new Thread(delegate()

{

TelnetChar("R", drawingContext);

});

Thread t2 = new Thread(delegate()

{

TelnetChar("L", drawingContext);

});

Thread t3 = new Thread(delegate()

{

TelnetChar("M", drawingContext);

});

/\*

\* Met deze code zal hij max 2/3 keer per seconde een L/R sturen

\* En na 10 seconden in activiteit stuurt hij een M

\*/

if (handLeft.X <= hipCenter.X - 0.4 && lastHit.Add(new System.TimeSpan(0, 0, 0, 0, 500)) <= DateTime.Now)

{

FormattedText left = new FormattedText("L send! ",

new CultureInfo("en-us"),

FlowDirection.LeftToRight,

new Typeface(new FontFamily("Arial"), FontStyles.Normal,

FontWeights.Normal, new FontStretch()),

16D,

Brushes.White);

drawingContext.DrawText(left, new Point(300, 50));

t2.Start();

lastHit = DateTime.Now;

mSend = false;

}

if (handRight.X >= hipCenter.X + 0.4 && lastHit.Add(new System.TimeSpan(0, 0, 0, 0, 500)) <= DateTime.Now)

{

FormattedText left = new FormattedText("R send!",

new CultureInfo("en-us"),

FlowDirection.LeftToRight,

new Typeface(new FontFamily("Arial"), FontStyles.Normal,

FontWeights.Normal, new FontStretch()),

16D,

Brushes.White);

drawingContext.DrawText(left, new Point(300, 50));

t1.Start();

lastHit = DateTime.Now;

mSend = false;

}

if (lastHit.Add(new System.TimeSpan(0, 0, 10)) <= DateTime.Now && !mSend)

{

FormattedText left = new FormattedText("Turn Mid!",

new CultureInfo("en-us"),

FlowDirection.LeftToRight,

new Typeface(new FontFamily("Arial"), FontStyles.Normal,

FontWeights.Normal, new FontStretch()),

16D,

Brushes.White);

drawingContext.DrawText(left, new Point(300, 50));

t3.Start();

lastHit = DateTime.Now;

mSend = true;

}

}

/// <summary>

/// Telnet connection

/// </summary>

private void TelnetChar(string charToSend, DrawingContext drawingContext) {

//create a new telnet connection to hostname "gobelijn" on port "23"

/\*TelnetConnection tc = new TelnetConnection("192.168.1.26", 8888);

string prompt = "";

// while connected

if(tc.IsConnected)

{

prompt = charToSend;

tc.WriteLine(prompt);

string temp = tc.Read();

}\*/

using (StreamWriter w = File.AppendText("logFile.txt"))

{

Log(charToSend + " send to raspberry through telnet.", w);

}

Thread.Sleep(100);

// 500 is 5sec

// 25 is 0.25sec

}

public static void Log(string logMessage, TextWriter w)

{

// w.Write("\nLog Entry : ");

w.WriteLine("\n{0}", DateTime.Now);

w.WriteLine("\r : {0}", logMessage);

}

}

}

# Extra features

Zoals foto trekken indien na 18uur + beweging

Wireless camera

…

# Results

Img + foto van opstelling + programma

# Demo

Plaats link hier

# Conclusion

Wat hebben we geleerd / wat kon er beter