Homework 1 (with additional explanations and new additional explantions)

Intelligent Robotics 1,

ECE 478/578

Deadline, **October 21**

This Homework is Group Project
Will be incorporated into your final project

About Homework 1 for all students

1. Students who work with Richard Armstrong on Mr. Jeeves are not required to do these homeworks.
2. You discuss all your mini-projects with Rick.

1. Two homeworks are mandatory for all other students.
2. Because of additional Intel Projects, the homework deadlines will be extended.
3. You work in your initial teams or teams of your choice.
4. You can use work of other students as part of your project but you must specifically mention it.
5. I am grading based on results and not good will of a student.

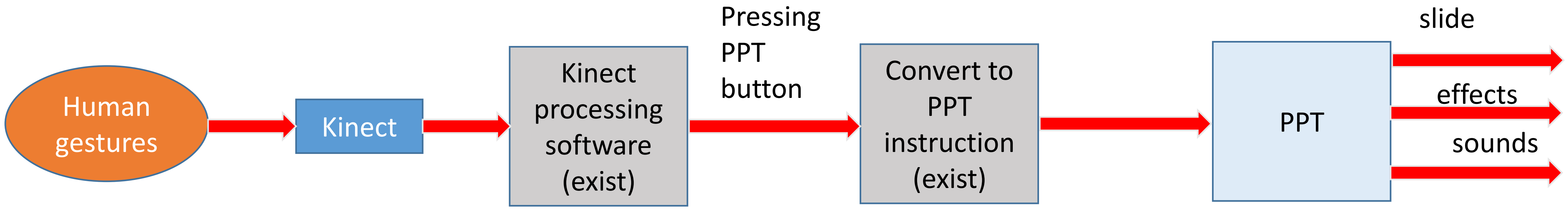
- First two phases of Homework 1 are mandatory
- Third first of Homework 1 is for volunteers only.
- In this homework 1 you will learn:
  1. Use of Kinect to control a robot, to create commands and data for a robot.
  2. The concept of state machine in robotics.
  3. The concept and use of fuzzy logic in robotics
  4. Using Powerpoint for scenario prototyping.
  5. Dialogs with robots.

Explanation to the **first phase** of Homework 1

1. Install Kinect and Kinect libraries. GLUT, etc. Find a page on Internet, plus pages with Kinect applications.
2. Read about KINECT and using Kinect to control mouse motions.
3. (independent on Kinect) Create Powerpoint presentation with effects, figures, pictures, videos, etc
   - A) Presentation about Copernicus – for those in Copernicus group
   - B) Presentation about Frankenstein Monster and PSU – for those in Monster Group and in stage design group.
   - C) Presentation with the “Quantum Debate” play – for those in Quantum Debate Play robots (Einstein, Newton, Marie Curie, Cat and Bohr).
4. (independent on Kinect) These slides can be done with jumps and transitions, as in my PPT slides demonstrated in class, but must have better effects and be more theatrical. Record your voice – we need more male voices. These voices and other sounds should be incorporated into the PPT presentation.
5. (independent on Kinect) Add questions and answers in Romanian, Spanish, Russian, Chinese, Indian, etc. (we have Latin and German already). Do not spend too much time, do this only if you are familiar with these languages.
6. (integration of Kinect and PPT) Use Kinect-based control of PPT slides software from my class and/or from internet. The Kinect is used to simulate the motion of the mouse.
   - a. You control it up, down, left, right, etc. You see the cursor when you move it on the screen.
   - b. Then you control clicking a mouse. Kinect interface can simulate all possible motions of the mouse.
   - c. You can create a user interface with many buttons, for binary, symbolic, integer or any data – for instance the control types.
   - d. This way, you can select for your robot the input and controlling data:
     - a. Input of left and right sensor like in Braitenberg Vehicles
     - b. Type of Braitenberg Vehicle (robot) such as Shy or Aggressive
     - c. Values of parameters.
     - d. Actions to move, grab, click, etc.
   - e. See the separate files for explanation of the KINECT software. Read them, or at least be aware what information is in which file.

the first phase of Homework 1

In the simplest variants the gestures simulate actions of a computer mouse

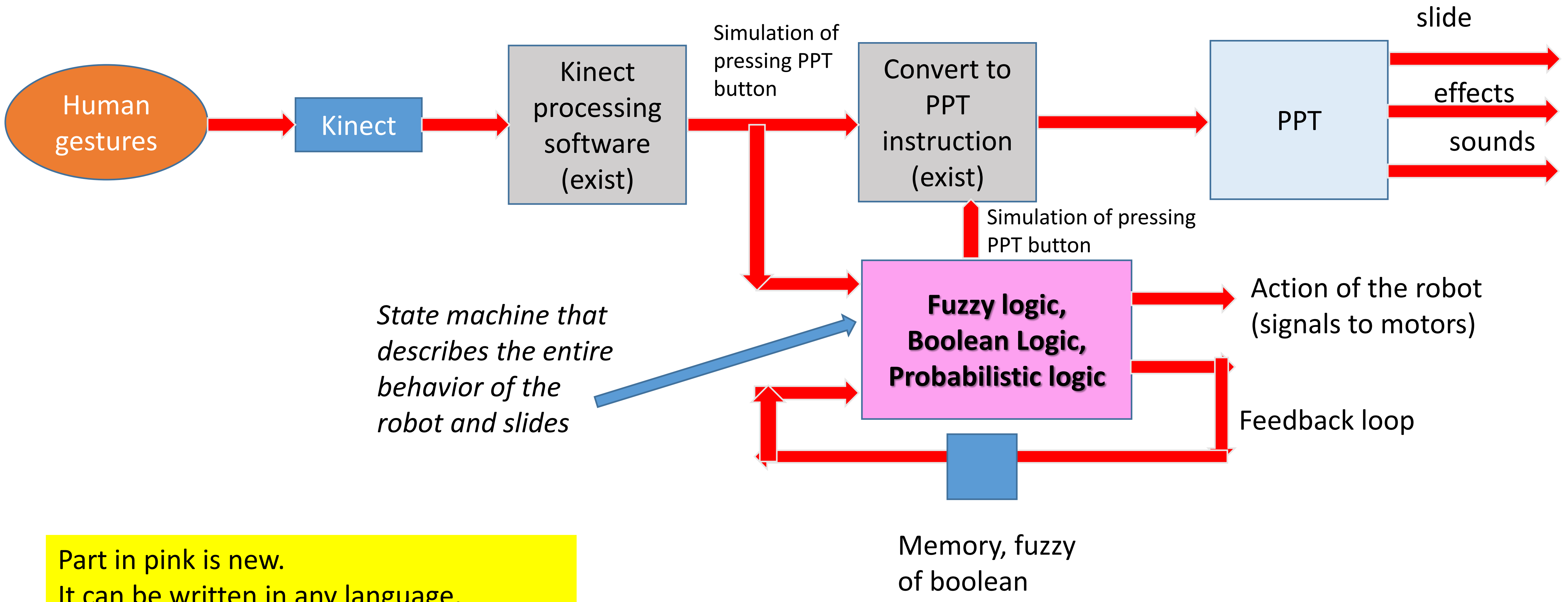

Human gestures
Kinect
Kinect processing software (exist)
Pressing PPT button
Convert to PPT instruction (exist)
PPT
slide
effects
sounds

In a more advanced variant the gestures are emotional human gestures that are interpreted by a state machine

Explanation to the SECOND phase of Homework 1

1. Download and install KINECT. Be sure that it works correctly using their test software.
2. Complete and test the first phase of the homework, as explained above. You cannot complete your project without understanding how Kinect is included in this project.
3. (independent on Kinect) Create in your software a state machine that describes the behavior of the robot, robots or/and the entire theatre presentation.
   a. This machine can be deterministic, probabilistic of fuzzy.
   b. Or mixed.
   c. It can have several machine communicating with one another. It can be programmed in any language.
4. The entire system can be written in any language of your choice, or mixture of languages. It should use PPT, and Kinect software.
5. The block-diagram of the system is shown in next slide.
6. Next three slides present some variants of the state machine. You are not restricted to them. You can create arbitrary state machine.
7. Additional slides demonstrate fuzzy and other state machines for a robot.
8. Videotape your work. Write a document – report.

Complete software for the second phase of the Homework 1 (minimum variant)


Human gestures
Kinect
Kinect processing software (exist)
Simulation of pressing PPT button
Convert to PPT instruction (exist)
PPT
slide
effects
sounds
Simulation of pressing PPT button
Fuzzy logic, Boolean Logic, Probabilistic logic
Action of the robot (signals to motors)
State machine that describes the entire behavior of the robot and slides
Feedback loop
Memory, fuzzy of boolean

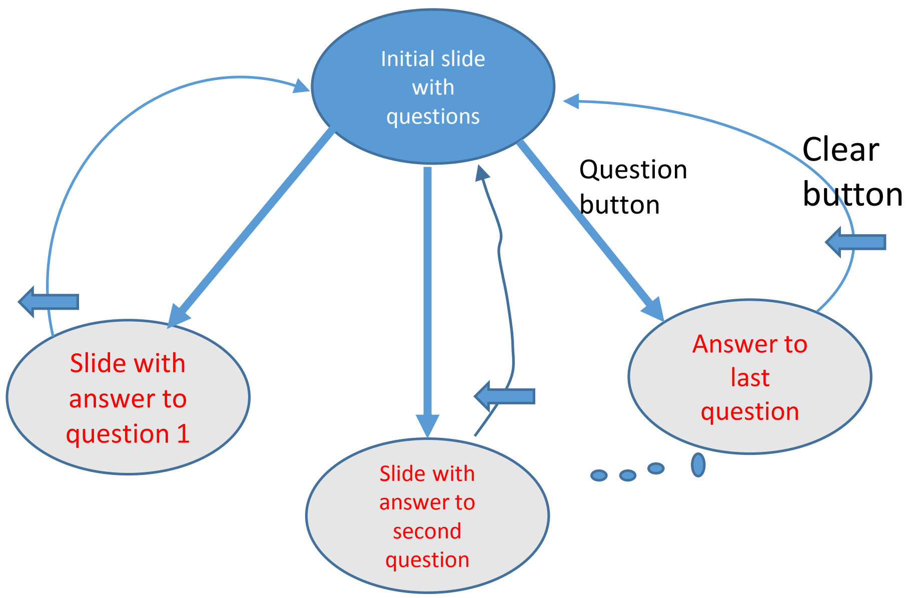
Part in pink is new.
It can be written in any language, perhaps the easiest is to use the same language in which Kinect processing and conversion was written (color grey)

Additional explanation

1. Observe that the only input to a robot is Kinect.
2. Thus the robot is controlled by one or more of the following
    a. Human gestures with arms
    b. Human gestures with ”skeleton”
    c. Human pressing buttons or using any “mechanical interface”
    d. Position of other robots as seen on Kinect.
    e. Position of people as seen by Kinect
    f. Quality of human motion in exercise, as evaluated by some objective criteria
    g. Any other idea taken from literature about uses of Kinect.
    h. Any of your ideas, be creative.!
    i. The robot can be physical :
        a. any robot in the lab,
        b. Lego robot on your own,
        c. what you want.
    j. The robot can be simulated :
        a. Simple ad hoc graphics like in Robo-Logo or Lego, turtle graphics.
        b. Any existing software for simulating andvanced robots (needs some simple integration).

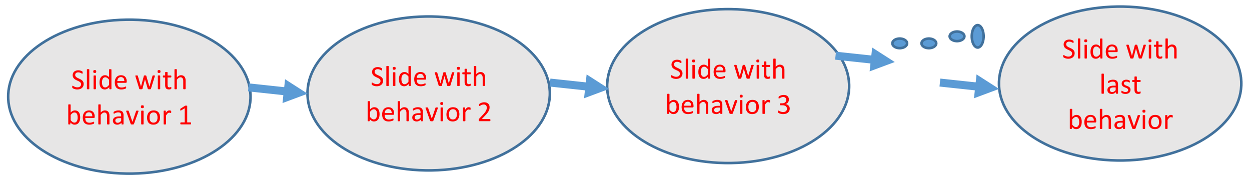
Simplified State Machines of robot plays

M1. State Machine of Copernicus


Initial slide with questions
Question button
Clear button
Slide with answer to question 1
Slide with answer to second question
Answer to last question

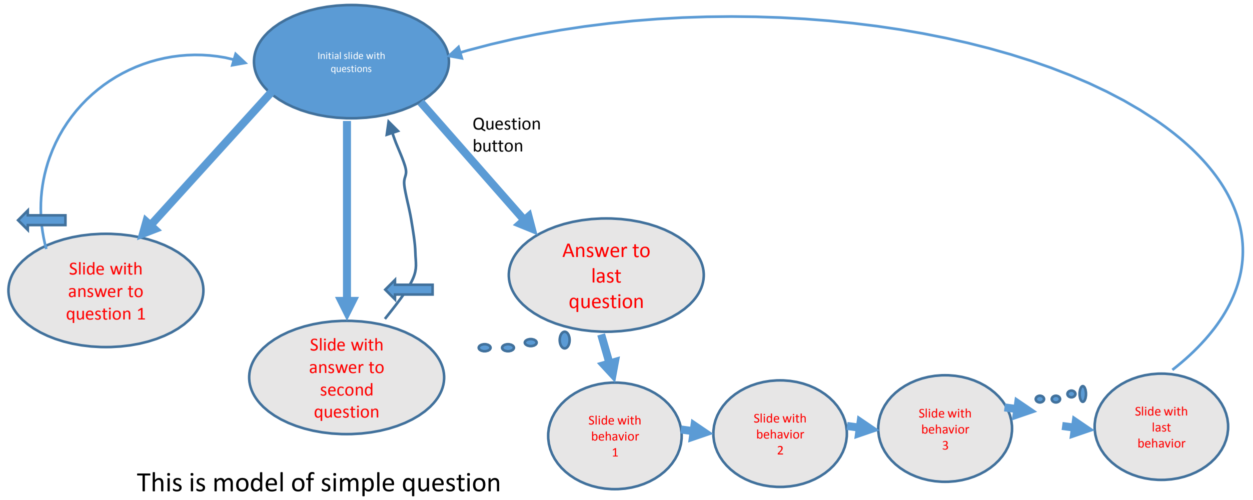
This is model of simple question answering

M2. State Machine of Regular Linear play



This is model of simple linear play like in standard theatre and Great Quantum Debate play

Combined M1 and M2. Answer to a question is a Regular Linear play

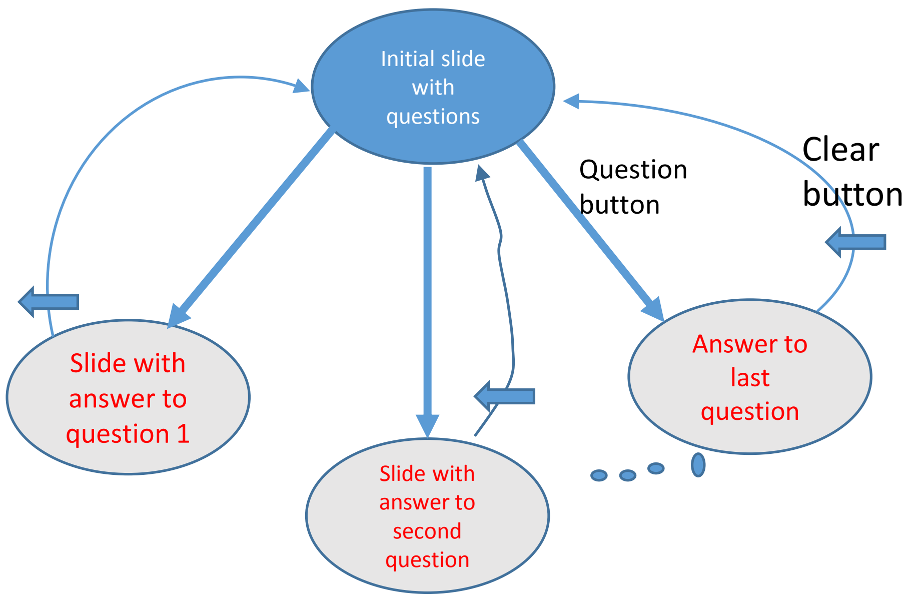


This is model of simple question answering

Other combinations are of course possible

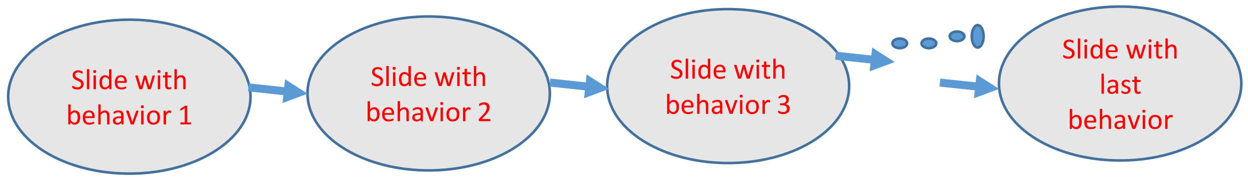
Simplified State Machines of robot plays

M1. State Machine of Copernicus


Initial slide with questions
Question button
Clear button
Slide with answer to question 1
Slide with answer to second question
Answer to last question

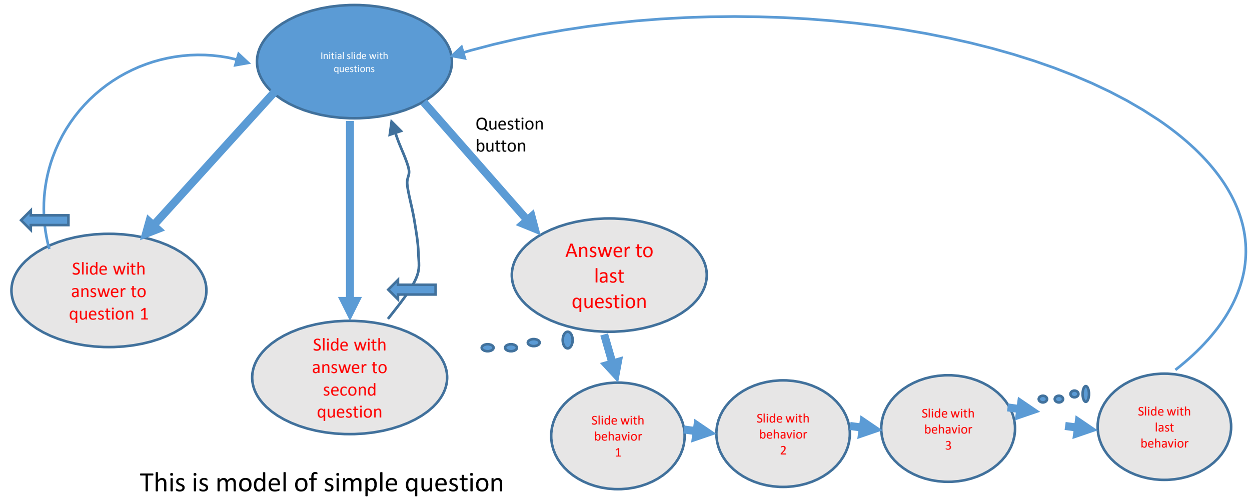
This is model of simple question answering

M2. State Machine of Regular Linear play



This is model of simple linear play like in standard theatre and Great Quantum Debate play

Combined M1 and M2. Answer to a question is a Regular Linear play



This is model of simple question answering

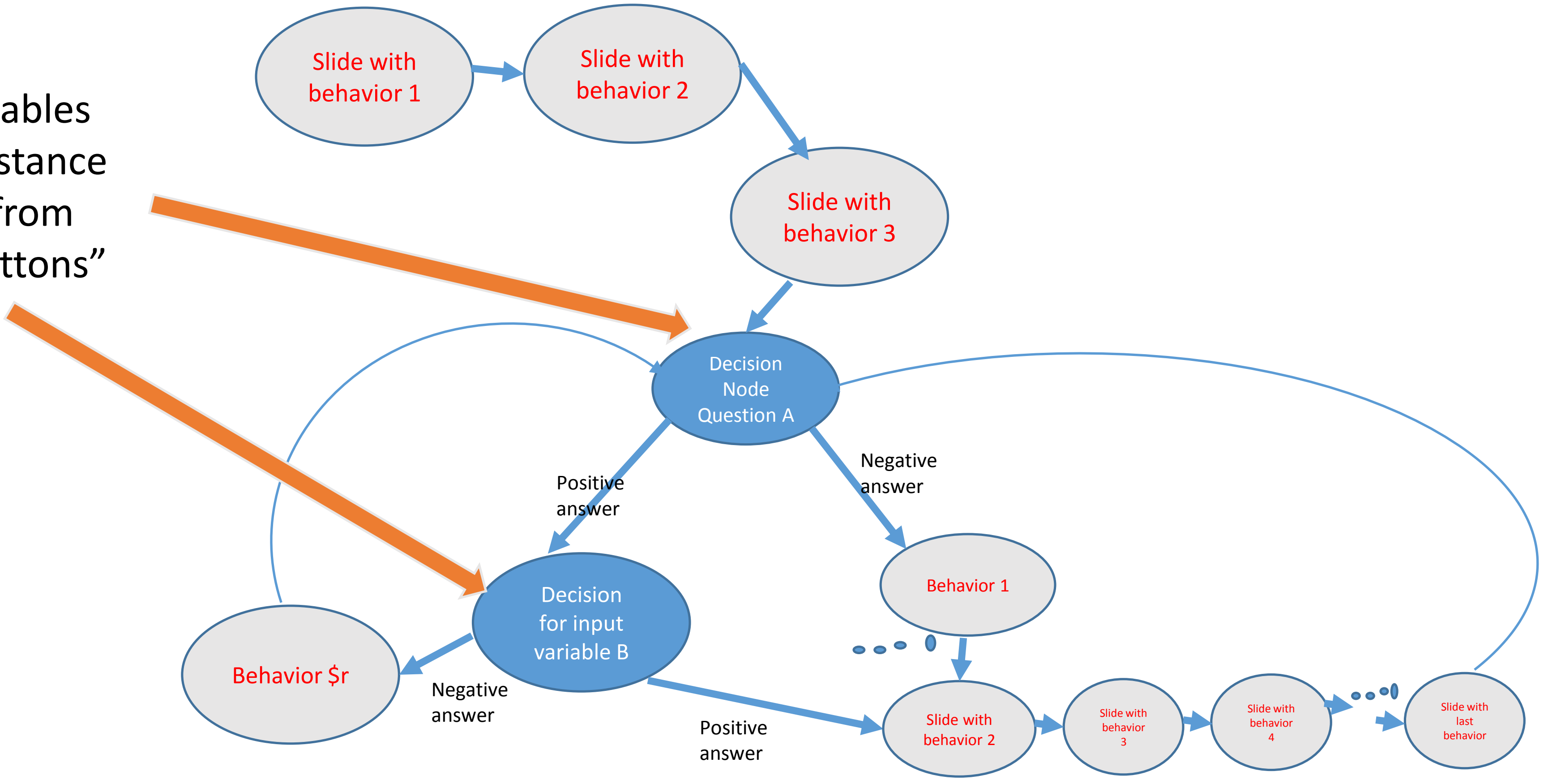
Other combinations are of course possible

Simplified State Machine of interactive robot play

M3. State Machine with interaction (similar to one explained in class for Paradise Lost)

This is equivalent to Finite State Machine or general Flowchart

The decision variables are inputs, for instance from sensors or from “pressing PPT buttons”



Remarks

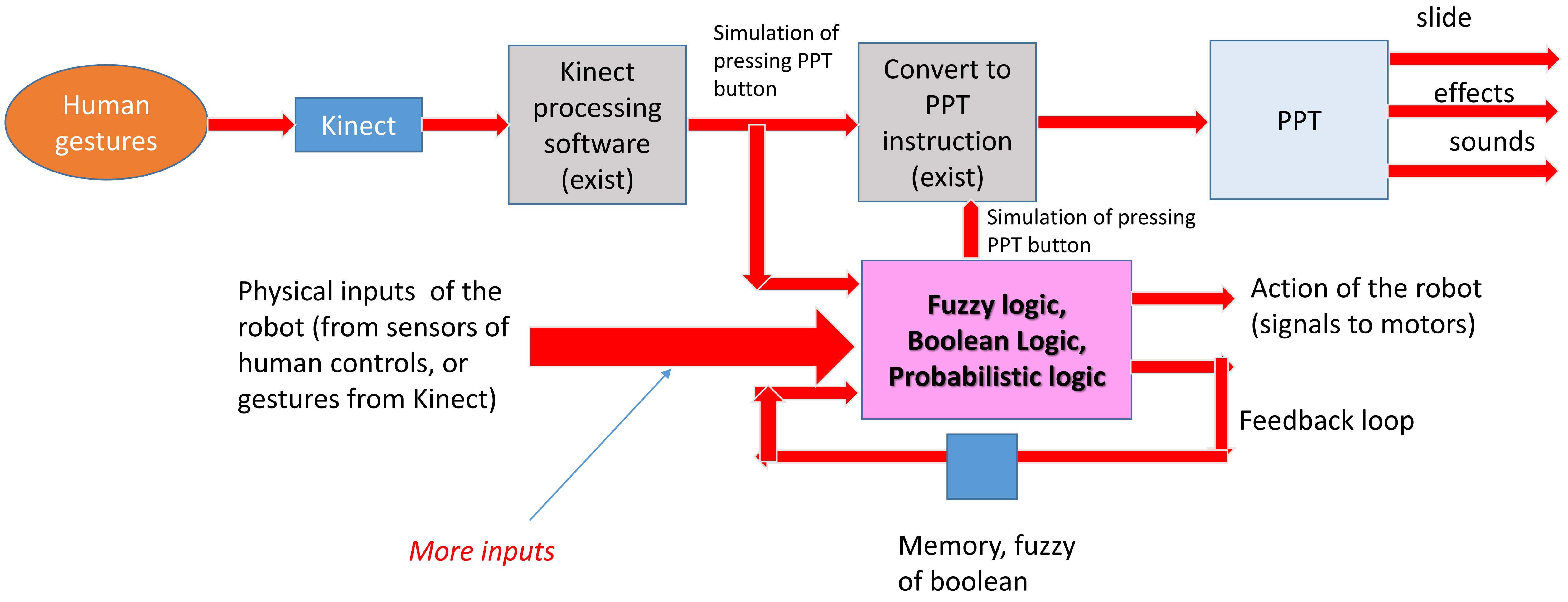
1. There are many people working on the same part of homework (like all people from Copernicus team)
2. The homeworks for all students (Copernicus team, quantum debate teams and Monster team) in the class are quite similar.
3. Software for Kinect exists and can be also found in internet.
4. Partial credit will be given for partial solutions.
5. Creativity will be rewarded.
6. Ask me questions if not clear.

Explanation of the third phase of Homework 1 (NOT MANDATORY)

Additional work for people who already work on projects (not required now, but will count for the next homework and your project)

1. Assume any robot from the lab, it can be particularly your robot from your project (this would save you time later on).
2. The robot **must have some input(s) other than KINECT**. This can be touch sensors, sonars, light sensors, etc. Your choice.
3. The robot should have some effectors.
4. It can be a Lego robot of your design. See my slides.
5. The robot's architecture can be any architecture discussed so far in our class, for instance a Braitenberg Vehicle.
6. The robot behavior is changed by gestures from KINECT. For instance, you can use your hand gestures or facial gestures to change the behavior of a Braitenberg Robot from Aggressive to Shy.
7. All your work should be videotaped.

Complete software for the additional work for the Homework 1


slide
Human gestures
Kinect
Kinect processing software (exist)
Simulation of pressing PPT button
Convert to PPT instruction (exist)
PPT
effects
sounds
Simulation of pressing PPT button
Physical inputs of the robot (from sensors of human controls, or gestures from Kinect)
Fuzzy logic, Boolean Logic, Probabilistic logic
Action of the robot (signals to motors)
Feedback loop
More inputs
Memory, fuzzy of boolean

List of useful documents on this page for Week2

1. Scripts of plays and conversations.
2. Report about KINECT to PPT interface done in our class.
3. Other reports about KINECT to PPT interfaces.
4. Other reports about controlling robots with KINECT.
5. Class material about fuzzy logic.
6. Material (PPT slides and text) about using a simple quantum(probabilistic) control of a Braitenberg robot in RobotC language.

Kinect Controlled Slide show presentation

Kinect Controlled Slide show presentation

- **Kinect Interactions** – allow developers to create interactive applications with a new, well-designed and ergonomically friendlier interaction language than before.
- The Kinect SDK can now detect a “push towards the screen” gesture to activate buttons, and this is the recommended “activate” gesture now instead of the old “hover, wait until the circle fills and its pressed” gesture.
- The latest K4W SDK solves this issue by differentiating between an open and a closed hand.
- When you close your hand, you “grip” whatever’s underneath it (such as a scroll viewer), and then you can move your hand to scroll.
- If you release the scroller while your hand is still moving, the scrolling keeps happening – similarly to the inertia scroll you are already used to on touch devices.
- You can close your hand again at any time, and stop the inertia – again, as you expected.

Software that you have to install:

- - Kinect SDK 1.7
- - Kinect for Windows Developer Toolkit

In the MainView.xaml

1. In the MainView.xaml, add the following namespace and add the KinectSensorChooserUI control to the main grid:

```
xmlns:k=http://schemas.microsoft.com/kinect/2013
<Grid>
   <k:KinectSensorChooserUI HorizontalAlignment="Center" VerticalAlignment="Top"
Name="sensorChooserUi" />
</Grid>
```

Initialize KinectSensorChooser

2. We also need to initialize the KinectSensorChooser from the code behind file Add a private field to the MainWindow code behind file.

```
private KinectSensorChooser sensorChooser;
```

Set Up an event handler for the OnLoaded

3. Set up an event handler for the OnLoaded event in the MainWindow constructor (or in XAML, as you wish):

```
public MainWindow()
{
    InitializeComponent();
    Loaded += OnLoaded;
}
```

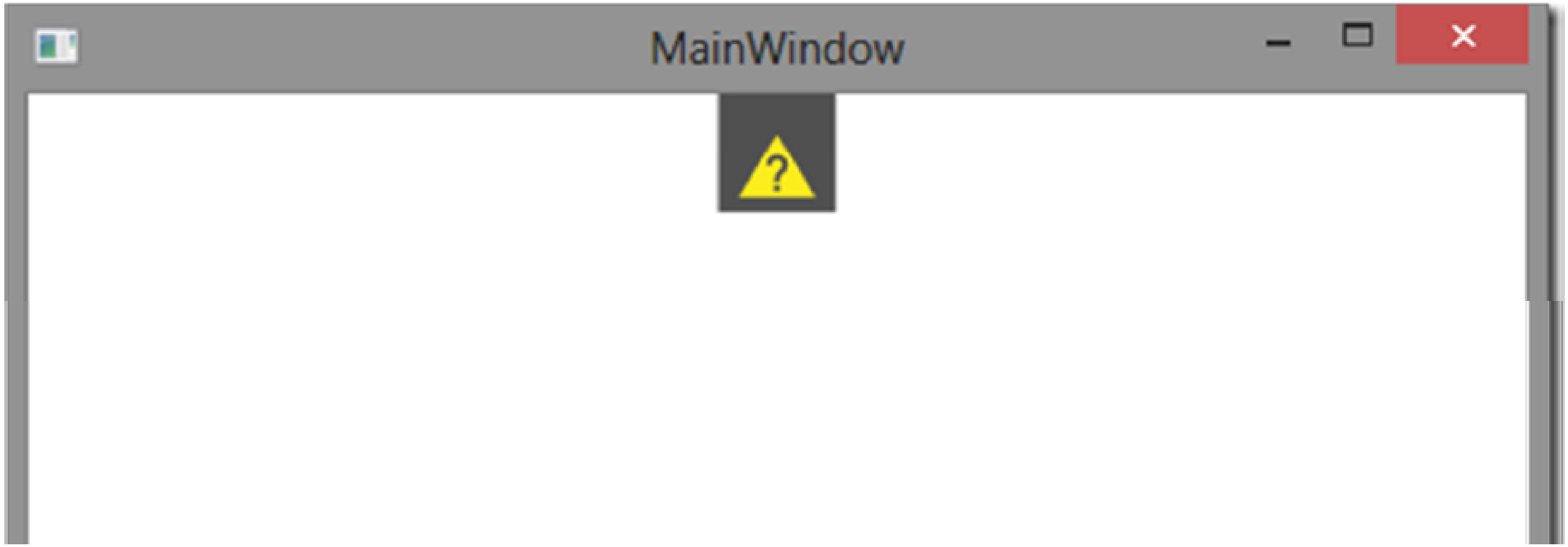
Create the OnLoaded event handler:

4. create the OnLoaded event handler:

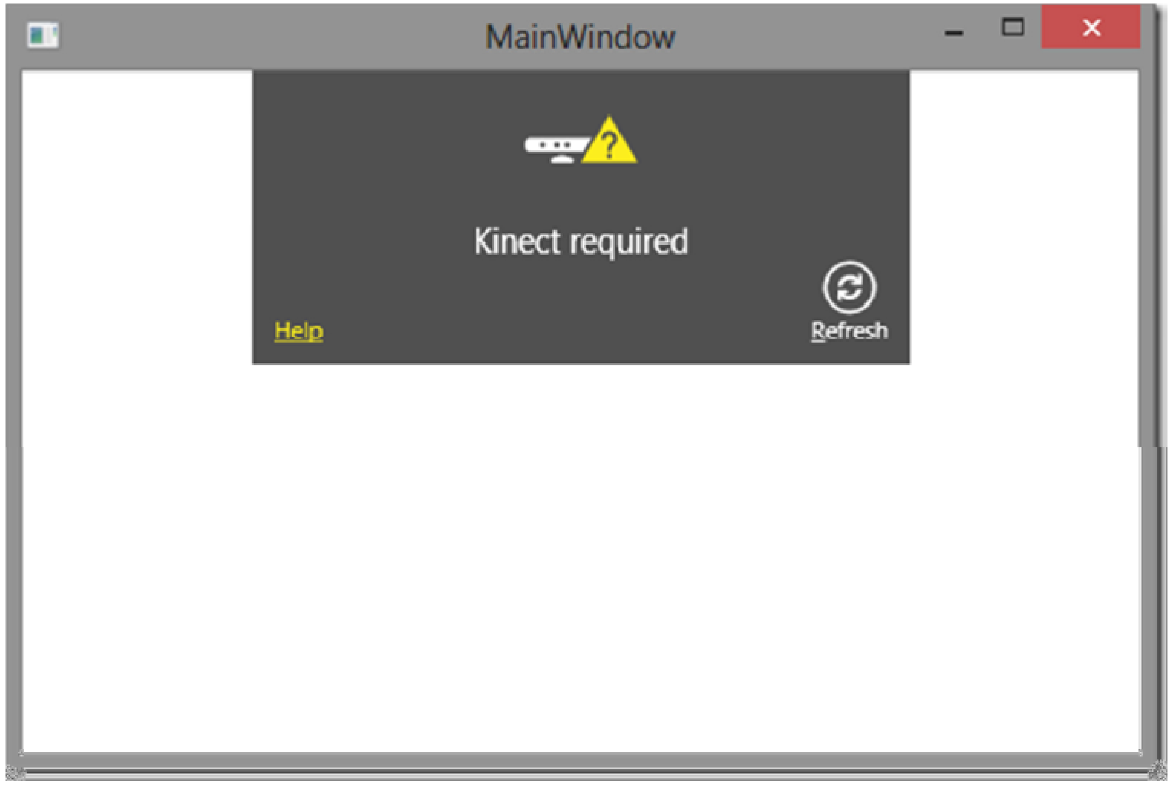
```
private void OnLoaded(object sender, RoutedEventArgs routedEventArgs)
{
    this.sensorChooser = new KinectSensorChooser();
    this.sensorChooser.KinectChanged += SensorChooserOnKinectChanged;
    this.sensorChooserUi.KinectSensorChooser = this.sensorChooser;
    this.sensorChooser.Start();
}
```

If a sensor is chosen or got initialized, the code above will invoke the SensorChooserOnKinectChanged event handler.

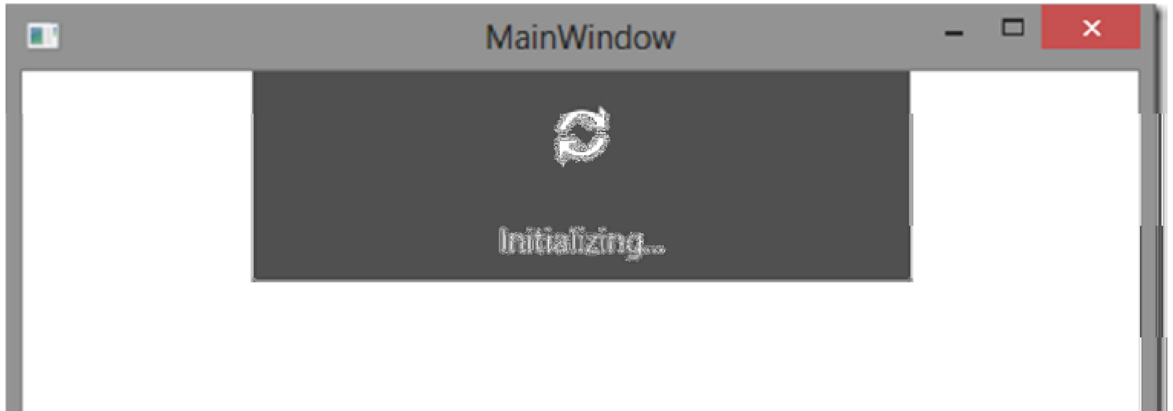
If there are no Kinects connected to the computer:



Kinect required



Once the Kinect is connected, it initializes it.



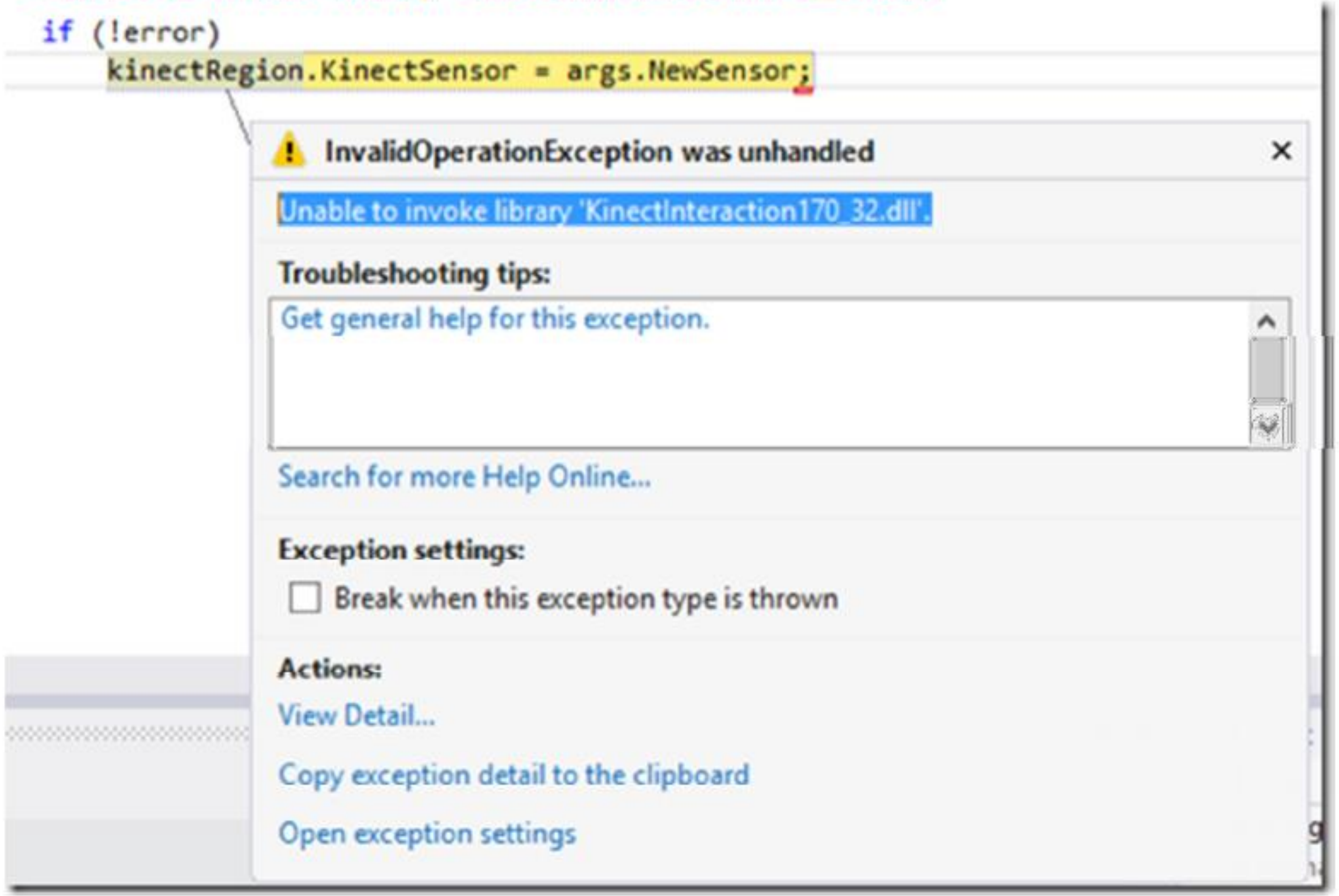
KinectRegion

- KinectRegion is the key element of Kinect Interactions on WPF:
- It is the screen area where interactive elements are placed and can be manipulated.
- KinectRegion is also responsible for displaying and moving the hand cursor.
- An application can have multiple KinectRegions, but they cannot be nested.
- Each KinectRegion can have its own respective Kinect sensor.

```
<k:KinectRegion Name="kinectRegion">
</k:KinectRegion>
```

Error message:

“Unable to invoke library 'KinectInteraction170_32.dll'.”



```
if (!error)
    kinectRegion.KinectSensor = args.NewSensor;
```

InvalidOperationException was unhandled

Unable to invoke library 'KinectInteraction170_32.dll'.

Troubleshooting tips:

Get general help for this exception.

Search for more Help Online...

Exception settings:

☐ Break when this exception type is thrown

Actions:

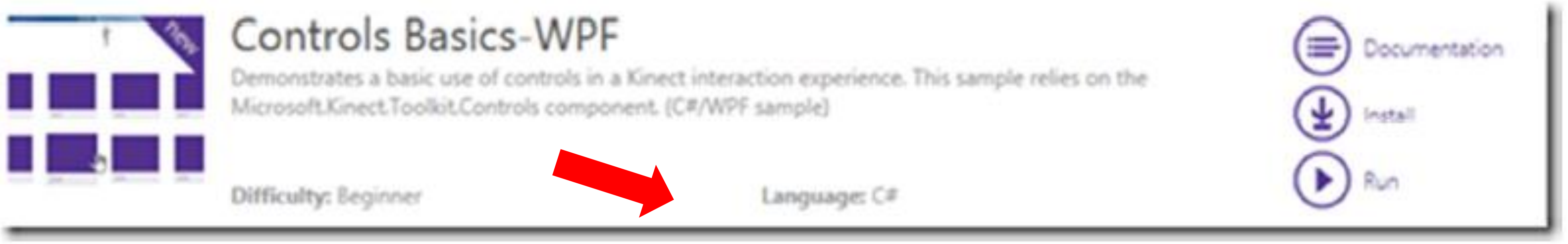
View Detail...

Copy exception detail to the clipboard

Open exception settings

Solution

- This is a common problem with Kinect.
- The dll mentioned can be a bit tricky to find.
- The simplest way is to install the Control Basics – WPF source code from the Developer Tookit Browser (press the Install button)
- Or just download the sample project for this article.


Controls Basics-WPF

Demonstrates a basic use of controls in a Kinect interaction experience. This sample relies on the Microsoft.Kinect.Toolkit.Controls component. (C#/WPF sample)

Difficulty: Beginner

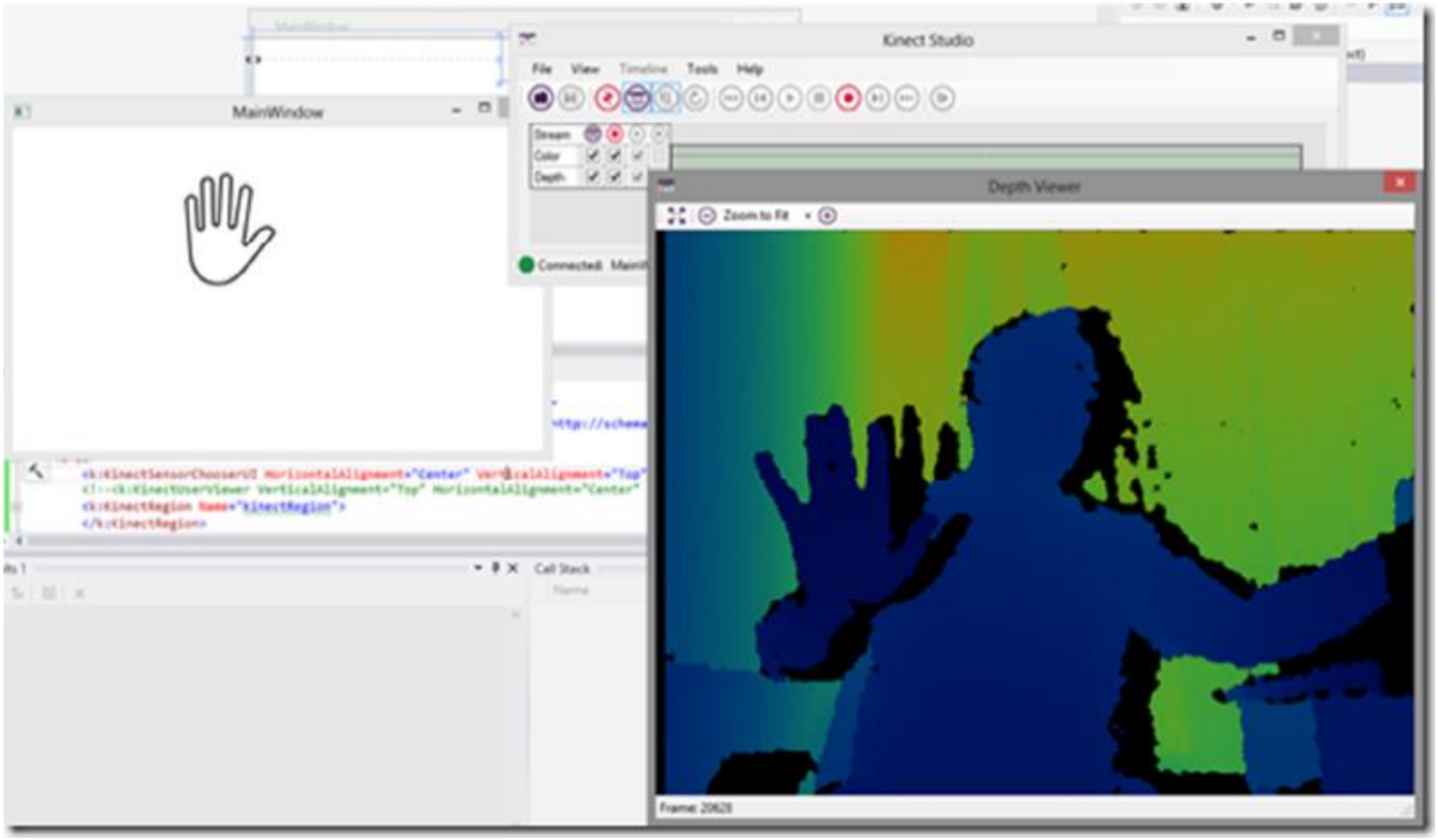
Language: C#

Kinect Controlled Slide Show Presentation

- In the AnyCPU.Debug folder of the sample's installation directory, copy the KinectInteraction170_32.dll and KinectInteraction170_64.dll files to the project's output directory.

- NOTE:
- You may have to wait as much as 20 seconds for the Kinect Sensor to initialize.
- During this process, your application will not respond and will show a wait cursor if you move the mouse over it.

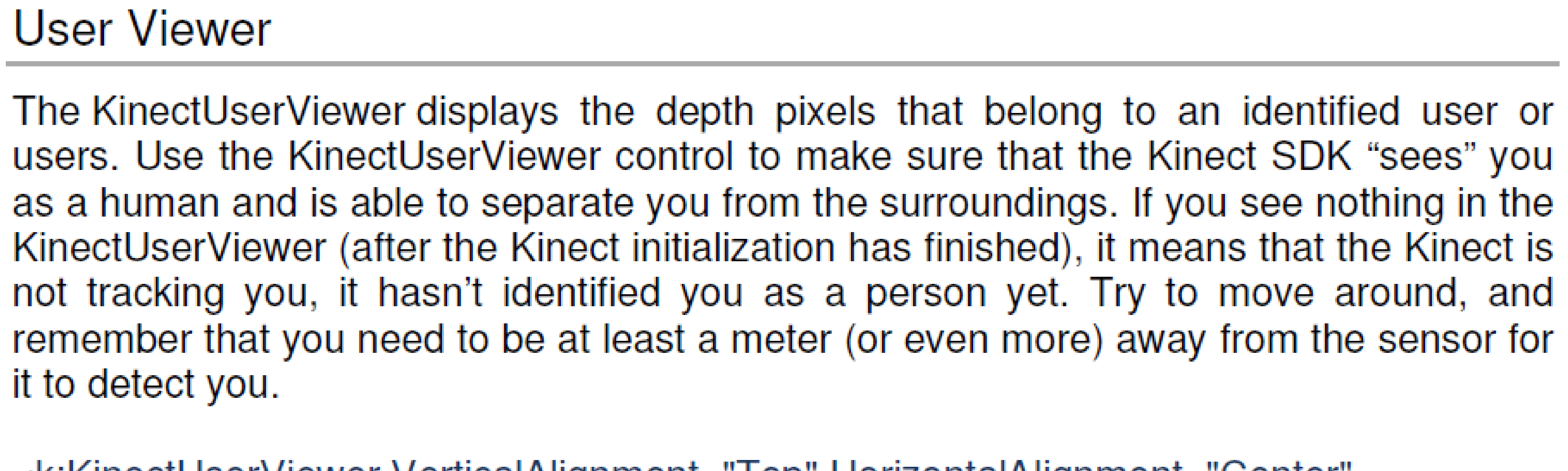
Better hand recognition using Kinect:



```
<k:KinectSensorChooserUI HorizontalAlignment="Center" VerticalAlignment="Top"
<!--<k:KinectUserViewer VerticalAlignment="Top" HorizontalAlignment="Center"
<k:KinectRegion Name="kinectRegion">
</k:KinectRegion>
```

User Viewer

- The Kinect

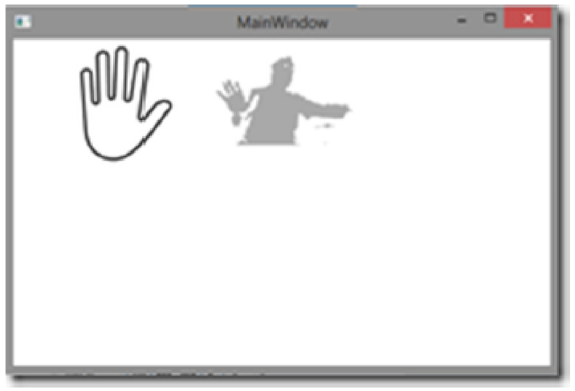


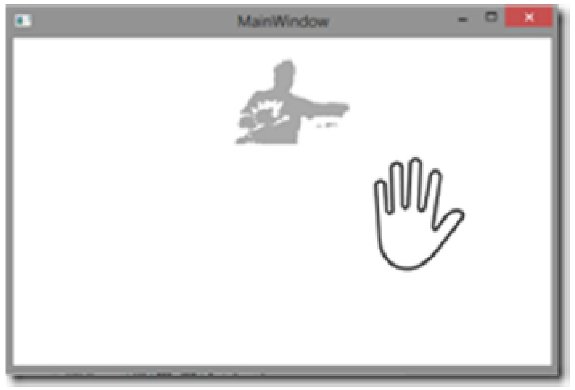
User Viewer

The KinectUserViewer displays the depth pixels that belong to an identified user or users. Use the KinectUserViewer control to make sure that the Kinect SDK “sees” you as a human and is able to separate you from the surroundings. If you see nothing in the KinectUserViewer (after the Kinect initialization has finished), it means that the Kinect is not tracking you, it hasn’t identified you as a person yet. Try to move around, and remember that you need to be at least a meter (or even more) away from the sensor for it to detect you.

```
<k:KinectUserViewer VerticalAlignment="Top" HorizontalAlignment="Center"
```

Screenshots:





Interactive Controls

Interactive Controls

KinectTileButton:
The KinectTileButton is one of the simplest controls available. It resembles a Windows 8-style tile with a template-able label and a Content that is just like the content of the regular Button control.

In this project, we are using this to display the slides on the bottom of the screen.
Below is the source code to declare the tile buttons dynamically and bind it to the slide show.