Hands-on Robotics with Programming

Mar 15 2010

Shih-Chung Kang
Assistant Professor
Computer-Aided Engineering (CAE)
Department of Civil Engineering
National Taiwan University

Today's Topics

1. **Arbiter class overview**
2. **Introduction to VPL (Virtual Programming Language)**
3. **Simulation of LEGO Robot**
4. **Introduction to DSS (Decentralized Software Services)**
5. **Program LEGO Robot Motion Control DSS**

1.Arbiter class overview (1/7)

- **Arbiter.Recive:**
  Creating "Receive Task".
  When a "Port" gets message, executes "Handler".

```
var myport = new Port<int>();
var mytask = Arbiter.Receive(true, myport, i
=> Console.WriteLine(i));
```

1.Arbiter class overview (2/7)

- **Arbiter.Activate:**
  Creating "DispatcherQueue " to execute "Tasks".

```
var taskQueue = new
DispatcherQueue("myqueue", new Dispatcher());
var myport = new Port<int>();
Arbiter.Activate(taskQueue,
Arbiter.Receive(true, myport, msg =>
Console.WriteLine(msg)));
```

1.Arbiter class overview (3/7)

- **Arbiter.Choice:**
  Like switch case. Executing one "Handler".

```
Arbiter.Choice(myport, intmsg =>
  Console.WriteLine(intmsg), stringmsg =>
  Console.WriteLine(stringmsg));

Arbiter.Choice(
    Arbiter.Receive<int>(false, myport,
  intmsg => Cosnole.WriteLine(intmsg)),
    Arbiter.Receive<string>(false, myport,
  strmsg => Console.WriteLine(strmsg)));
```

1.Arbiter class overview (4/7)

- **Arbiter.FromHandler,**
  Arbiter.FromIteratorHandler:
  Turning "delegate function " into "Task".
  "Task" is used for "DispatcherQueue".

```
var task = Arbiter.FromHandler( () =>
  { Console.WriteLine("Hello World"); });
```

FromIteratorHandler: IEnumerator<ITask>

1.Arbiter class overview (5/7)

Arbiter.Interleave:
ServiceBehavior.Concurrent (Read),
ServiceBehavior.Exclusive (Write),
ServiceBehavior.TearDown (Stop).

```
// activate an Interleave Arbiter to coordinate how the handlers of the service
// execute in relation to each other and to their own parallel activations
Arbiter.Activate(_taskQueue,
    Arbiter.Interleave(
    new TeardownReceiverGroup(
    // one time, atomic teardown
        Arbiter.Receive<Stop>(false, _mainPort, StopHandler)
    ),
    new ExclusiveReceiverGroup(
    // Persisted Update handler, only runs if no other handler running
        Arbiter.Receive<UpdateState>(true, _mainPort, UpdateHandler)
    ),
    new ConcurrentReceiverGroup(
    // Persisted Get handler, runs in parallel with all other activations of itself
    // but never runs in parallel with Update or Stop
        Arbiter.Receive<GetState>(true, _mainPort, GetStateHandler)
    ))
);
```

1.Arbiter class overview (6/7)

- **Arbiter.JoinedReceive:**
  Wait until both "Ports" get message.

```
Arbiter.JoinedReceive<int, string>(false,
  myport, myport2, (intmsg,strmsg) =>
  Console.WriteLine(intmsg+","+strmsg))
```

1.Arbiter class overview (7/7)

- **Arbiter.MultipleItemReceive:**
  Wait until "Port" get enough messages.

```
Arbiter.MultipleItemReceive(true, portInt, 5,
    intArray =>
  Console.WriteLine(string.Join(",", new
  List<int>(intArray).ConvertAll<string>(i
  => i.ToString()).ToArray()));
```

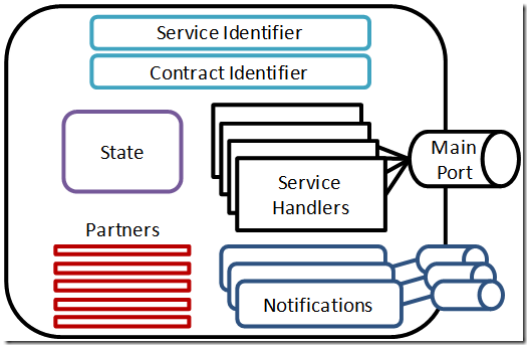
2.Introduction to VPL (Visual Programming Language)

- 2010 #4 VisualProgrammingLanguage.ppt

3.Simulation of LEGO Robot

- 2010 #4 SimulationOfLegoRobot.ppt

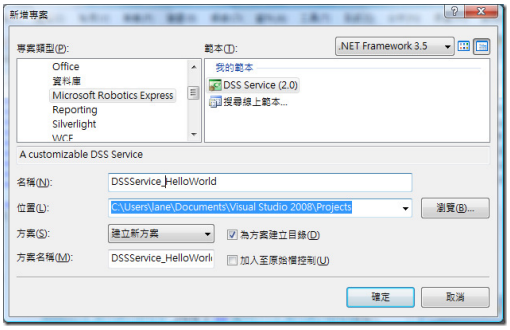
4.Introduction to DSS (Decentralized Software Services)



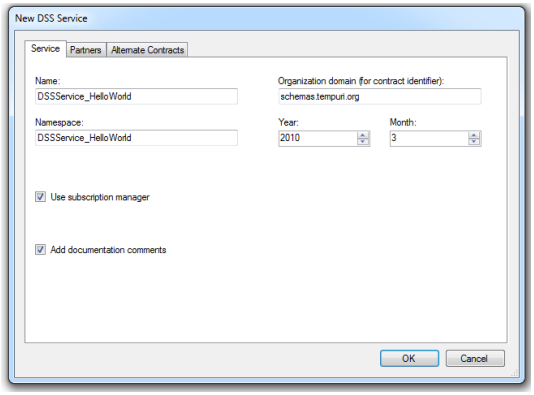
4.Introduction to DSS Example 1: Hello World

4.DSS, Example 1 (1/5)

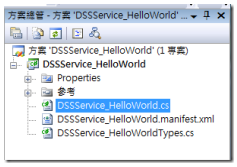
- "Hello World" again.
- Visual Studio 2008



4.DSS, Example 1 (2/5)



4.DSS, Example 1 (3/5)



DSSService_HelloWorldTypes.cs

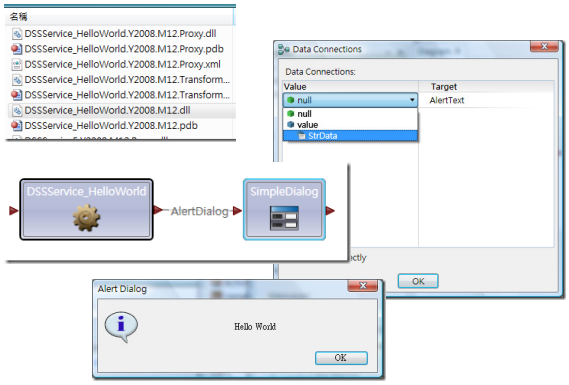
```
/// <summary>
/// DSSService_HelloWorld state
/// </summary>
[DataContract]
public class DSSService_HelloWorldState
{
        [DataMember]
        public string StrData { get; set; }
}
```

4.DSS, Example 1 (4/5)

DSSService_HelloWorld.cs

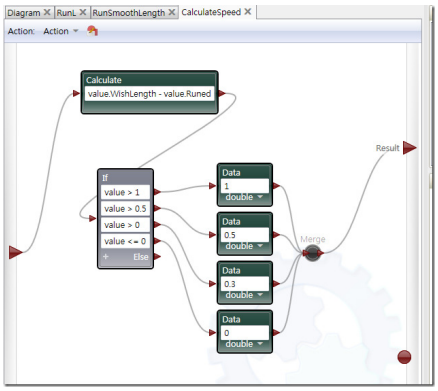
```
/// <summary>
/// Service start
/// </summary>
protected override void Start()
{

    //
    // Add service specific initialization here
    //

    base.Start();
    _state.StrData = "Hello World";
}
```

4.DSS, Example 1 (5/5)



4.Introduction to DSS
Example 2: Patrol Robot

4.DSS, Example 2 (1/6)



4.DSS, Example 2 (2/6)

- **CalculateSpeedService**Types.cs

```
[DataContract]
public class CalculateSpeedInput
{
    [DataMember]
    public double WishLength { get; set; }

    [DataMember]
    public double StageRunned { get; set; }
}

[DataContract]
public class CalculateSpeedOutput
{
    [DataMember]
    public double Speed { get; set; }
}
```

4.DSS, Example 2 (3/6)

- CalculateSpeedServiceTypes.cs

```
[DisplayName("CalculateSpeedToRun")]
[Description("Calculate Speed to run")]
public class CalculateSpeed : Submit<CalculateSpeedInput,
   PortSet<CalculateSpeedOutput, Fault>>
{

}
```

4.DSS, Example 2 (4/6)

- CalculateSpeedServiceTypes.cs

Original:

```
public class CalculateSpeedServiceOperations :
PortSet<DsspDefaultLookup, DsspDefaultDrop, Get, Subscribe>
```

New:

```
public class CalculateSpeedServiceOperations :
PortSet<DsspDefaultLookup, DsspDefaultDrop, Get, Subscribe,
CalculateSpeed>
```

4.DSS, Example 2 (5/6)

- CalculateSpeedService.cs

```
[ServiceHandler(ServiceHandlerBehavior.Concurrent)]
public IEnumerator<ITask> CalculateSpeedHandler(CalculateSpeed cs)
{
    double lengthleft = cs.Body.WishLength - cs.Body.StageRunned;
    double calculatedspeed = 0;
    if (lengthleft > 1)
        calculatedspeed = 1;
    else if (lengthleft > 0.5)
        calculatedspeed = 0.5;
    else if (lengthleft > 0)
        calculatedspeed = 0.3;
    else if (lengthleft <= 0)
        calculatedspeed = 0;
     cs.ResponsePort.Post(new CalculateSpeedOutput() { Speed =
   calculatedspeed });
    yield break;
}
```

4.DSS, Example 2 (6/6)