Introduction to Game Programming and Robotics

Unit # 1

Tentative Course Outline

- Visual Programming Language
  - Interaction with Robots through VPL
  - Competition
  - C# Code Generation
- Microsoft Robotics Development Studio
  - C# Overview
  - Concurrency and Coordination Runtime
  - Decentralized Software Services
  - Remote Webcam
  - Competitions: Robocup Soccer, Sumo Wrestler

Tentative Course Outline (Cont'd)

- Application of AI Techniques in Different Competitions
  - Neural Networks
  - Bayesian Networks
  - Evolutionary Algorithms
- Virtual Simulation Environment
- XNA Game Studio Express

Useful Information

- Course Website
  - http://cse460robotics.wikispaces.com/
- Books
  - Kyle Johns and Trevor Taylor, Professional Microsoft Robotics Developer Studio, 2008
  - Alexandre Lobao, Bruno Evangelista, and José de Farias, Beginning XNA 2.0 Game Programming: From Novice to Professional, 2008
  - Darryl Charles, Colin Fyfe, Daniel Livingstone and Stephen McGlinchey, Biologically Inspired Artificial Intelligence for Computer Games, 2008
  - Several other books mentioned on the course website

Online Resources

- Join the course wikispace and actively participate in discussions.
- Microsoft Robotics Studio Website for tutorials and discussion forums
  - http://msdn.microsoft.com/en-us/robotics/default.aspx
  - http://social.msdn.microsoft.com/Forums/en-US/category/robotics
  - http://msdn.microsoft.com/en-us/robotics/bb383569.aspx (Video Casts)

Software

- Download and install Microsoft Robotics Studio and Visual Programming Language
  - We will start with the Express edition but hopefully we will get the Standard Edition soon (through MSDN Academic Alliance)
  - Copy of Express Edition can be collected after the class
- Download and install XNA Game Studio Express
  - We won't start working on the XNA Game Studio Express before the 2nd Midterm.

Grading Policy (Tentative)

- Quiz 10%
- Midterms (15% each) 30%
- Final 30%
- Project/Competition 25%
- Class Participation 5%

Imagine Cup 2009 in Switzerland

Imagine Cup 2009 is not only a global competition. Microsoft Switzerland provided prizes for all Swiss Imagine Cup 2009 competitors and organized the local Imagine Cup 2009 finals on 8th May 2009.

Awards for all competitors!
All Swiss students who successfully competed (submitted entry as defined in each category's Round 1 instructions) in the Round 1 in any of the Imagine Cup 2009 categories received a Microsoft Press book:

- Categories 1 & 5: Programming Microsoft Visual C# 2008: The Language
- Categories 2 & 4: Programming Microsoft Robotics Studio
- Category 3: Microsoft XNA Game Studio 2.0: Learn Programming Now!
- Categories 6-9: Introducing Microsoft Silverlight 2.

The 35 first Swiss students who successfully competed (submitted entry as defined in Round 1 instructions) in Embedded Development category, as well as, the 35 Swiss competitors getting the highest scores in Round 1 of the Robotics & Algorithms category had also the chance to be awarded with a LEGO® MINDSTORMS® NXT to get quick start and hands-on experience with embedded development on Microsoft Robotics Studio and robotics hardware. See details: You Make IT Smart.

http://www.microsoft.com/switzerland/education/en/academiczone/professors/imaginecup.mspx

Project Expectations

- Implementation of New Robocup Soccer Algorithm (Project # 1)
- Development of an XNA-based Game (Project # 2)

- This is a tentative plan and as we move along more specific instructions will be provided.
- But the ultimate goal is to prepare students for Imagine Cup (it may take 1-2 years).

Demos

- WebCam
- Robocup Soccer
- Connectivity with Game Controller

Disclaimer

- Most of the materials and examples discussed in this lecture are taken from the following two sources:
  - http://msdn.microsoft.com/en-us/robotics/default.aspx (Tutorials)
  - Kyle Johns and Trevor Taylor, Professional Microsoft Robotics Developer Studio, 2008

Visual Programming Language

- The VPL is a new application development environment designed specifically to work with DSS services.
- Programs are defined graphically in data flow diagrams rather than the typical sequence of commands and instructions found in other programming languages.

First VPL Program: Hello World!

- Connection Dialog: This dialog enables a user to select which message is being output from the source and which message accepts the input in the destination activity.
  - From: DataValue
  - To: SayText
- Data Connections Dialog: This enables a user to map a particular data value in the message output from the source to a target value in the message input to the destination.
  - Value Box: Value
  - Target Box: SpeechText

First VPL Program: Hello World! (Cont'd)

- VPL automatically sends an input message to all top-level *Data* activities when the program starts.
- A *message* has one or more fields and each has a name and a type.
- When we specify values in the Data Connections dialog, we are actually selecting which field of the structure to send to a target value in the destination activity.

First VPL Program: Hello World! (Cont'd)

- Select the connection between the *Data* activity and the *TexttoSpeechTTS* activity. Change the *Value* property to *Length*.
- Now when we execute the diagram, the engine says 21.

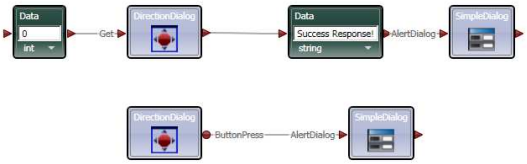
First VPL Program: Hello World! (Cont'd)

- In this example the *Data* activity builds a message that contains a value, which is a string and a value associated with the string, which is its length.
- This message is sent to the *TexttoSpeechTTS* activity. When the *TexttoSpeechTTS* activity receives a message on its *SayText* input pin, it speaks the text and then sends a success of failure message on its output pin.

Inputs, Outputs, and Notifications

- Each activity has one ore more inputs and one or more outputs.
- When there are multiple inputs or outputs, only a single pin is shown and the desired input or output is selected with the Connections dialog when a connection is made.
- Some activities have an additional output called a *notification*.

Second VPL Program



- Between Data → DirectionDialog
  - From: DataValue
  - To: Get
- Between DirectionDialog → Data
  - From: Get-Success
  - To: Create
- Between Data → SimpleDialog
  - From: DataValue
  - To: AlertDialog
- Between DirectionDialog → SimpleDialog (Notification)
  - From: ButtonPress
  - To: Alert Dialog

Second VPL Program (Cont'd)

- When we drag onto the diagram an activity that is already presented, VPL asks us whether we want a completely new activity or just another reference to the existing activity.
- If we specify that we want a new activity, the new activity is given a different name than the original one.

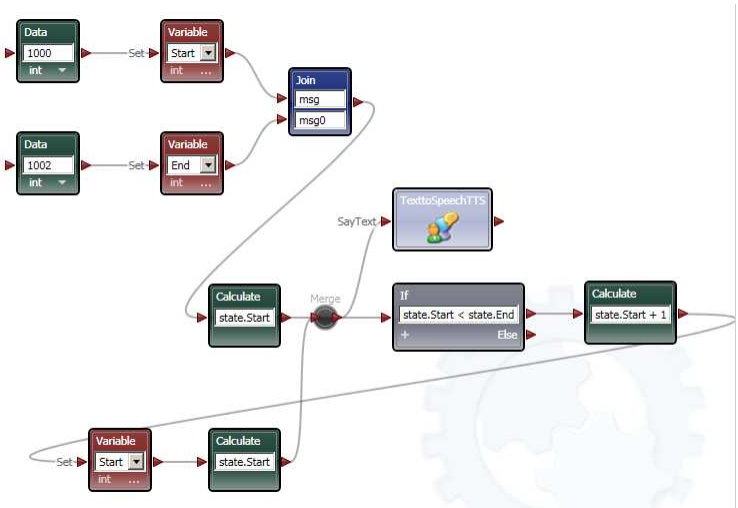
List of Basic Activities

Activity	Description
Variable	This activity is used to get or set the value of a state variable.
Calculate	This activity is used to calculate a value from one or more inputs.
Data	This activity is used to define static data that is used to an input to another activity.
Join	This activity waits to send an output message until it has received a message on all of its inputs. (similar to AND gate)
Merge	This activity forwards a message to its output from any of its inputs. (similar to OR gate)
If	This activity forwards a message on the first output that has an expression that evaluates to *true.*
Switch	This activity forwards a message on the first output that matches the input value.
List	This activity holds a list of data.
List Functions	This activity performs various operations on a list of data.

VPL Variables

- We use the *Variable* activity to define and set or get the value of a variable.

Looping and Conditions in VPL



The VPL Execution Model

- A diagram is inactive until a message enters it.
- The message can originate from a notification on an activity from a message that is injected into the inputs of a top-level *Data* activity.

Debugging a VPL Diagram

- *Note: The debugger works properly with Internet* Explorer *only.*
- We can run a diagram in debug mode by clicking Run → Debug Start.
- The debugger page is divided into four sections.
  - Diagram State: It shows values of all of the state variables in the diagram.
  - Current Node: It shows the currently executing activity, activities that have been scheduled for execution, and breakpoints.
  - Breakpoints: It shows the currently set breakpoints.
  - Pending Nodes: It shows all activities scheduled to be executed.

Homework

- Download and install MRDS
- Implement the programs discussed in today's class.