Ch 11: Scripting with Lua

Quiz # 5
Discussion
Data-Driven Software Design

Outside Materials
Why Data-Driven S/W Design?

✦ Non-Programmers
✦ Change in the code with no compilation
✦ More flexibility and Extensibility
Function: Approach 1

Action Func decision (state)

{
    if (state is X) then action = a
    if (state is Y) then action = b
    return action;
}
Function: Approach 2

Action Func decision (state)
{
    Lookup Table (State, action);
    return action;
}

//table here is defined by the user
Examples in Games
Game Design Example: user interface

Figure 2.4: State representation of Quantum Software's game.
Game Design Example: user interface

✧ What are the mappings in this interface?
✧ How do you abstract them?
✧ Do you think it is a good idea to abstract them? Why?
Game Design Example: Balance

- Manipulating numbers
- Introducing chance
- Manipulating rules
- Use trade-off matrix
- Encoding the game as another balanced game, e.g. Rock, Paper, Scissors
Game Design Example: Balance

笼
Are strategies that gives you a win no matter what.

笼 E.g.

<table>
<thead>
<tr>
<th></th>
<th>Wife Birthday</th>
<th>Not Wife’s Birthday</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buy Flowers</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>Don’t Buy Flowers</td>
<td>-100</td>
<td>0</td>
</tr>
</tbody>
</table>
Game Design Example: Balance

✦ What are the mappings in this?
✦ How do you abstract them?
✦ Do you think it is a good idea to abstract them? Why?
Game Design Example: AI

Execution Flow of an AI Engine

- Sense
- Think
- Act

Finite-state machines
Decision trees
Neural nets
Fuzzy logic
Rule-based systems
Planning systems
Example FSM

Events:
E = Enemy Seen
S = Sound Heard
D = Die

Action (callback) performed when a transition occurs
Mirage: Agent Model
Figure 6.1 Interactive Narrative Architecture

- **Story Engine**
  - dramatic intensity, presentation plan, anticipated camera movements

- **Director Agent**
  - lighting parameters, dramatic intensity, anticipated camera movements
  - character behavioral goals
  - camera behavioral goals

- **ELE**
  - Position, Orientation, Attenuation, Penumbra, Umbra, color

- **Camera System**
  - Position, Orientation, Field of view

- **Character Controller System**
  - Character
  - Behavior
  - Actor System
    - Position, Orientation, motion no.
Action Selection – Behavior Beats

**Trigger:** a goal that triggers the character behavior

**Preconditions:** defines the context that enables this behavior

**Postconditions:** defines the actions or side effects of the behavior

Sub-goals: define the sub-problems that need to be solved for the behavior to succeed:

- collection of character goals that need to be solved in sequence or parallel
- collection of character goals that need to be solved in sequence or parallel
- a combination of both
Agent Behavior – Simple Behavior

**Trigger:** a goal that triggers the character action

**Preconditions:** defines the context that enables this action

**Postconditions:** defines the actions or side effects of the action

**Action:** represented as Action + Adverb describes how the agent performs the action, encoding the subtext
Behavior Selection – Reactive Planning

1. choose behavior plan given user stereotype, character goal, failed behaviors

2. for each time tick
   ✤ monitor user action assessing current behavior
   ✤ if failure limit reached, fail behavior and go to step 1
   ✤ Update user model
**Algorithm**

```
Active Plan Tree

Goal:
open <door>
success-test:
<door> is open

Plan(sequential):
1. get <key> from <purse>
2. unlock <door> with <key>
3. open <door>
context-condition:
    have <purse> or have <key>

Goal:
get <key> from <purse>
success-test:
    have <key>

Goal:
unlock <door> with <key>
success-test:
    <door> is unlocked

Goal:
open <door>
success-test:
    <door> is open
```
How would you enable this?
Enabling Data-Driven Approaches

- Use tables and dynamic constructed lists
- Use scripting language (Lua)
- Use XML
Serialization

“Serialization is the process of converting a set of object instances that contain references to each other into a linear stream of bytes, which can then be sent through a socket, stored to a file, or simply manipulated as a stream of data”

-O’Rielly’s book
Formatters in .NET

- **Binary Formatter**
  
  ```csharp
  using System.Runtime.Serialization.Formatters;
  ```

- **SOAP Formatter**
  
  ```csharp
  using System.Runtime.Serialization;
  ```

- **XML formatter**
  
  ```csharp
  using System.Xml.Serialization;
  ```
Streams

- MemoryStream
  using System.IO;
- BufferedStream
  using System.IO;
- FileStream
  using System.IO;
The Class Definition:

```csharp
[Serializable]
public class player
{
    private int score;
    public int Score
    {
        get { return score; }
        set { score = value; }
    }
}
```

The Class Definition:

```csharp
[XmlRoot("Players")]
public class player
{
    [XmlElement ("Score")]
    private int score;
    public int Score
    {
        get { return score; }
        set { score = value; }
    }
}
```
FileStream mys = File.Create("Mydata.xml");

XmlSerializer x = new XmlSerializer
(typeof(player), "Player");
x.Serialize(mys, p);

mys.Close();

The Class Definition:
[XmlRoot("Players")]
public class player
{
    [XmlElement("Score")]
    private int score;
    public int Score
    {
        get { return score; }
        set { score = value; }
    }
}
Serialization in .NET (XML)

```csharp
XmlSerializer x = new XmlSerializer(
typeof (player), "Player");
FileStream myStream = File.OpenRead(name);
player p = (player) x.Deserialize
   (myStream);
myStream.Close();
```

The Class Definition:

```csharp
[XmlRoot("Players")]
public class player
{
    [XmlElement ("Score")]  
    private int score;

    public int Score
    {
        get { return score; }
        set { score = value; }
    }
}
```
<?xml version="1.0" encoding="utf-8" ?>

<Group>
    <Games>
        <Game>
            <Title>"Dragon Age"</Title>
            <Genre>"RPG"</Genre>
            <ReleaseDate>2011-10-11T00:00:00</ReleaseDate>
        </Game>
        <Game>
            <Title>"Assassins Creed"</Title>
            <Genre>"ActionAdventure"</Genre>
            <ReleaseDate>2011-11-15T00:00:00</ReleaseDate>
        </Game>
    </Games>
</Group>
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Xml.Serialization;
using System.IO;

namespace SerializationExercise
{
    class Program
    {

        static void Main(string[] args)
        {
            Group test;
            //List<Game> games;
            XmlSerializer mySerializer = new XmlSerializer(typeof(Group));
            FileStream myFileStream = new FileStream("../Games.xml", FileMode.Open);
            test = (Group)mySerializer.Deserialize(myFileStream);

            for (int i = 0; i < test.Games.Count; i++)
            {
                Console.WriteLine("game title is " + test.Games[i].Title);
            }
            Console.ReadLine();
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Xml.Serialization;
using System.Text;

namespace SerializationExercise
{
    [Serializable]
    public class Game
    {
        [XmlElement("Title")]
        public string Title
        {
            get;
            set;
        }

        [XmlElement("Genre")]
        public string Genre
        {
            get;
            set;
        }

        [XmlElement("ReleaseDate")]
        public DateTime ReleaseDate
        {
            get;
            set;
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Xml.Serialization;
using System.Text;
	namespace SerializationExercise
{
    [XmlRoot("Group")]
    public class Group
    {
        //Game[] games;
        [XmlArray("Games")]
        [XmlArrayItem("Game")]
        public List<Game> Games
        {
            get; set;
        }
    }
}
Class Assignment

- Take the class assignment you did in the last class, for the script for the camera
- Use an XML representation to script transitions for the camera
- Serialize this into a class
- Then use it to move the camera around
Assignment 3

- Let's take a look