Integrating and using IronPython in a XNA Game

Federico Cicchi

Pycon3 Conference

10th May 2009 - Florence

What is this talk about?

- Python in the Game Industry
- Introduce .NET, Iron Python and XNA Game Studio
- Demo session: How use Python inside a XNA game
- Demo session: A Python console running inside a XNA game

Please ask questions if I'm not clear!

Python in the Game Industry

"Changing scripting languages is the last thing you want to contemplate more than six months into a project like this, but by that point we had begun to realize our scripting language was not going to get the job done (...). We finally made the painful, but necessary, decision to switch, and ported the game to Python. Things have been great ever since." (Mike Goslin, Disney, about Toontown OnLine game)

Title	Developer	Publication
Severance: Blade of Darkness	Rebel Act Studio	2001
Eve OnLine	CCP Games	2003
Toontown OnLine	Disney	2003
Temple of Elemental Evil	Troika Games	2003
Civilization IV	Firaxis Games	2005
Battlefield 2 (http://bf2tech.org/)	Digital Illusion	2005
Pirates of the Caribbean OnLine	Disney	2007

What is this talk about?

- ✓ Python in the Game Industry

- → **Introduce .NET, Iron Python and XNA Game Studio**

- Demo session: How use Python inside a XNA game

- Demo session: A Python console running inside a XNA game

.NET Framework

- Microsoft Programming Framework for Windows (latest version 3.5)

- Not tied to a particular programming language, although the default language is C#.

- Powerful application runtime (CLR), Automatic garbage collection, powerful JIT compiler

- Cross-platform with Mono (http://mono-project.com)

- Microsoft .NET 4.0 announced last September
  - Full support for IronPython, IronRuby and F#

IronPython

- Implementation of the Python programming language running under .NET
- Originally created by Jim Hugunin, now being developed by a Microsoft team
- Well integrated with the rest of the .NET Framework
- Open source (Microsoft Public License)
- Run also on Mono
- Latest stable version 2.0.1 (targeted CPython 2.5.2 for compatibility)

Installing IronPython

- Download and Install .NET 2.0 SP1 redistributable (from Microsoft)

- Download and Install IronPython Installer (Latest stable version 2.0.1)
  - It also installs CPython standard modules and packages that work under IronPython.

- http://www.codeplex.com/IronPython

Using .NET Assemblies

- Standard .NET APIs can be imported directly by IronPython.
- We need a *reference* to the assembly by using `clr.AddReference()`

```
import clr
clr.AddReference('System.Windows.Forms')
```

- Then, the module can then be imported.

```
from System.Windows.Forms import Application,Form

f = Form()
f.Text = 'Hello World!'
Application.Run(f)
```

XNA Game Studio

- A set of tools based on supported versions of Microsoft Visual Studio.
- Allows building games for Microsoft Windows, XBOX 360 and Zune.
- Includes the XNA Framework, a set of managed libraries based on .NET 2.0 and designed for game development.
- XNA libraries can be imported directly by IronPython as other .NET libraries.

```
import clr
clr.AddReference('Microsoft.Xna.Framework')
clr.AddReference('Microsoft.Xna.Framework.Game')
from Microsoft.Xna.Framework import *
```

Installing XNA Game Studio

- Download and Install XNA Game Studio from http://creators.xna.com

What is this talk about?

- ✓ Python in the Game Industry
- ✓ Introduce .NET, Iron Python and XNA Game Studio
- → **Demo session: How use Python inside a XNA game**
- Demo session: A Python console running inside a XNA game

Set up our game project

- Open VisualStudio and create a new Windows Game project.

- Give a name to the project (for example “PyconGame”).

- Compile and Run.

- For our examples, we have included some common elements usually present in every game:
  - A game object class to manage 3d models, texture and rendering.
  - A camera class to view our game object.

The Python Interpreter (1)

- Add a new element: XNA Game Studio 3.0 → Game Component
  - Give a name to the component (for example PythonInterpreter.cs)

- Add these IronPython .DLL references to the project:
  - `IronPython.dll`
  - `IronPython.Modules.dll`
  - `Microsoft.Scripting.dll`
  - `Microsoft.Scripting.Core.dll`

The Python Interpreter (2)

- Our ingredients from `Microsoft.Scripting.Hosting`:

```
ScriptRuntime m_RunTime;
ScriptEngine m_Engine;
ScriptScope m_Scope;
```

- Our recipe:

```
m_RunTime = Python.CreateRuntime();
m_Engine = Python.GetEngine(m_RunTime);
m_Engine.SetSearchPaths(new string[] { "C:\\Program Files\\IronPython 2.0.1\\",
                                       "C:\\Program Files\\IronPython 2.0.1\\Lib\\"});
m_Scope = m_Engine.CreateScope();
```

- Add the interpreter to our game class:

```
private PythonInterpreter m_Interpreter;
```

```
m_Interpreter = new PythonInterpreter(this);
Components.Add(m_Interpreter);
```

The Python Interpreter (3)

- How execute Python code:

```
ScriptSource CreateScriptSourceFromFile(string path, Encoding encoding,
                                        SourceCodeKind kind);

ScriptSource CreateScriptSourceFromString(string code, SourceCodeKind kind);
```

- Example:

```
string code =
@"import clr
clr.AddReference('Microsoft.Xna.Framework')
clr.AddReference('Microsoft.Xna.Framework.Game')
from Microsoft.Xna.Framework import *
clr.AddReference('PyconGame')
from PyconGame import *"

ScriptSource src = m_Engine.CreateScriptSourceFromString(code, SourceCodeKind.Statements);

source.Execute(m_Scope);
```

The Python Interpreter (4)

- How reference and use XNA object in IronPython:

```
void SetVariable(string name, object value);

bool ContainsVariable(string name);
object GetVariable(string name);
T GetVariable<T>(string name);

public bool RemoveVariable(string name);
```

- For example, inside the Game class:

```
m_Interpreter.m_Scope.SetVariable("game", this);

string code = "print game.m_Ship.Name”;

ScriptSource src =
m_Engine.CreateScriptSourceFromString(code,SourceCodeKind.SingleStatement);

source.Execute(m_Scope);
```

Moving our ship with Python

- Script `shipControl.py`

```
import math
rotation = Vector3(0,0.05,0)

def Rotate(s, left):
  if left:
    s.m_WorldRotation += rotation
  else:
    s.m_WorldRotation -= rotation

def Move(s, forward):
  if forward:
    s.m_Velocity += Vector3(-math.sin(s.m_WorldRotation.Y), 0, -math.cos(s.m_WorldRotation.Y))
  else:
    s.m_Velocity -= Vector3(-math.sin(s.m_WorldRotation.Y), 0, -math.cos(s.m_WorldRotation.Y))
```

- Example:

```
m_Interpreter.AddGlobal("s1", m_Ship);
m_Interpreter.ExecuteFromFile("shipControl.py");
//...
if (Keyboard.GetState().IsKeyDown(Keys.Left)) {
  m_Interpreter.Execute("Rotate(s1, 1)", SourceCodeKind.SingleStatement); }
```

A "game based" example

- Typical game life cycle: *Initialization, Updating loop, Finalization*
- Game component are usually updated every frame (1/60 sec)
- We implement an Update() function using python generators:

```
class RotateBehavior(object):
    def __init__(self, obj):
        self.__generator = self.__UpdateFunction()
        self.object = obj
    # Update() is called once per frame and tells the update function to resume.
    def Update(self):
        return self.__generator.next()
    def __UpdateFunction(self):
        dRot = Vector3(0.00, 0.02, 0.00)
        while 1:    # Loop forever
            yield
            self.object.m_WorldRotation += dRot
```

What is this talk about?

- Python in the Game Industry
- Introduce .NET, Iron Python and XNA Game Studio
- Demo session: How use Python inside a XNA game
- **Demo session: A Python console running inside a XNA game**

The Python Console

- It'a a XNA Game component, like the Python Interpreter.
- It's mainly composed by a background sprite and a string object.
- Useful for debugging and prototyping.

Redirect the output

- By default, IronPython output result is displayed in the Visual Studio Console.

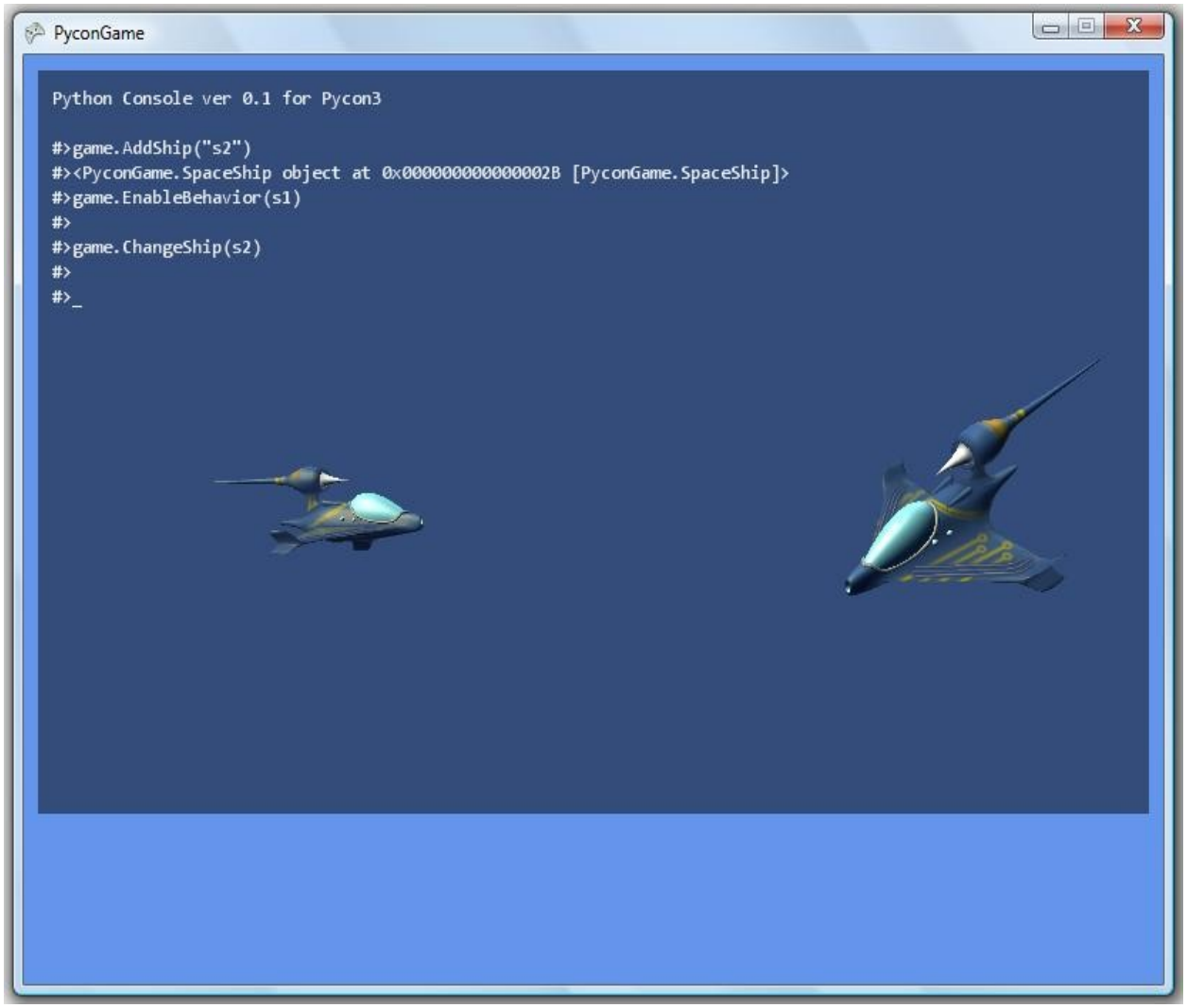
- But we can easily redirect it in the Python Interpreter component:

```
private System.IO.MemoryStream m_outputStream;
```

```
m_outputStream = new MemoryStream();
m_RunTime.IO.SetOutput(m_outputStream, new StreamWriter(m_outputStream));
```

```
private string ReadFromStream(MemoryStream ms)
{
    int length = (int)ms.Length;
    Byte[] bytes = new Byte[length];
    ms.Seek(0, SeekOrigin.Begin);
    ms.Read(bytes, 0, (int)ms.Length);
    string res = Encoding.GetEncoding("utf-8").GetString(bytes, 0, (int)ms.Length);
    ms.SetLength(0);
    ms.Position = 0;
    return res.Trim();
}
```

The Python Console



Thanks for listening!