Tackling online game development problems with a novel network scripting language

Paul Sheppard – ITI Techmedia
George Russell – Codeplay Software
Alastair F Donaldson – Codeplay Software
Rich Rowan – Codeplay Software
Verena Achenbach – Codeplay Software
ITI Scotland is a member-based commercial organisation focused on driving sustainable economic growth in Scotland through ownership of commercially targeted R&D programmes which deliver world-class intellectual assets.

Established 2003

$800m funding over 10 years

$200m already committed to creating innovative, commercially-focused IP across 20 R&D programmes
R&D Programme
Online Games Development

- Online Games Programme began October 2005
- Completes in December 2008
- Budget of $10m
- 4 key strands to the programme
  - Network Scripting Language
  - Games Design Toolkit
  - Software Productivity Tools
  - Procedural Content Generation
Network Scripting Language (NSL)
To create a novel scripting language for writing bandwidth-efficient online game logic
Writing network applications is hugely complex.

- Latency
- Bandwidth Efficiency
- Concurrency
- Debugging
- Testing
Create a simple language that handles complexity

- Easy to use
- Object oriented
- Deterministic
- Provides network transparency
- Includes debug & testing tools
NSL Code has been designed to be easily understood by programmers

- Uses a Java-like syntax
- Game and world is composed of *objects*

```java
47 class Cell {
48 {
49     /* Top left corner, position in the game world. */
50     public var vec2 pos;
51     /* The size of a cell e.g. its edge length */
52     public var float size;
53     public var int row;
54     public var int column;
55     public var int num_cells_in_row;
56     public var Point2D[] contents;
57     /* Left to right, in rows, including self. 
58     TL, T, TR, L, SELF, R, BL, B, BR. 
59     We currently only support having 9!. 
60     */
61     public var Cell[] neighbours;
62 }
63     Cell constructor(vec2 p, float s, int r, int c, int num_in_row) {
64         pos = p;
65         size = s;
66         row = r;
67         column = c;
68         num_cells_in_row = num_in_row;
69     }
```
Easy to use Integration

- Easy to integrate with any game engine written in C++
- Could be implemented in other languages on other platforms
- Designed to work seamlessly on top of different networking libraries
Network Transparency

Standard Implementation

<table>
<thead>
<tr>
<th>Game (Logic, Physics, Graphics, etc)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network Layer</td>
</tr>
</tbody>
</table>
Network Transparency

NSL Implementation

- Engine (Physics, Graphics, etc.)
- Game Logic
- Runtime
- Network Layer
Deterministic

- NSL enforces applications to be deterministic
- Server ensures clients maintain consistent state
- All clients and server will obtain the same results for the same inputs regardless of location or time of computation
Bandwidth Efficiency
Concurrency

- Parallel by design, invisible to programmer
- Multi-threaded runtime
- Aim to achieve linear parallel performance scaling in most cases.
Testing

- Determinism enables us to
  - Save any frame
  - Replay then or later

- Programmer does not need to build a test harness. All handled by the default features of NSL.
Graphical Display of the Game's Objects

File  Run  Control  View
Results
Bandwidth Efficiency

State Transfer Bandwidth vs NSL for Object Replication

- 40 FPS
- 30 FPS
- 20 FPS
- 10 FPS
- NSL at 20 FPS
- NSL at 40 FPS

Number of Game Objects

KB/sec

Results
Bandwidth Efficiency
Results

Server-client communication

Server Network Traffic for Input and Forwarded Input

- Traffic Out with Visibility
- Traffic Out without Visibility
- Traffic In with Visibility
- Traffic In without Visibility

No. of Clients vs Bandwidth (KB/s)
Results
Concurrentlty Test

% Performance Improvement

No. of Cells

- 2 Threads
- 4 Threads
- 8 Threads

- 144 196 256 324 400 441 484 529 576 625 676
Conclusions

- Initial results show that NSL can result in efficient use of network bandwidth.
- NSL can exploit the power of multi-core processors automatically.
- Programming of game logic is simplified as the required knowledge of underlying network system is minimised.
ITI are looking for companies who are interested in licensing this research for use in commercial projects.

Please contact us if you are interested in learning more.
Q & A

paul.sheppard@ititechmedia.com

richard@codeplay.com