Using Neural Parallel Language in Distributed Game World Composing

Xizhi Li
The CKC honors school of Zhejiang University
Computer Science Department
email: LiXizhi@zju.edu.cn

Abstract:
Internet computer games are the best known and most complex distributed computer systems in the world. It usually deals with thousands or even millions of interactive entities among its distributed web servers. An evolutionary story and game world extensibilities are the key features of the next generation Internet and mobile games. Some properties of such games are high individual intelligence, the ability to learn from its environment, group-evolution, and tolerance of errors and uncertainties. Game-engine practitioners have used scripting technology to add soft computing capabilities to a variety of their engine modules; so that commercially released games will still enjoy a certain degree of online-reconfiguration. This article proposes a unified approach of using Neural Parallel Language (NPL) in a computer game's scripting engine. We have implemented a reduced version of NPL and our own 3D game engine in a pair. We will show the effectiveness of such a programming language methodology by means of our released game demo. As the web is coming more and more computable (the semantic web) and intelligent (agent technology), neural network based programming paradigm as described in this article is likely to become the solution to general purpose distributed software applications.

Key words: game engine, NPL, neuron network, programming language

1 Introduction

Soft Computing will play an important role on the programming paradigm of general purpose Internet applications. While some current effort on Semantic Web/Grid tells a computer agent exactly what to compute, Soft Computing tells it how to compute. A great deal of web technologies and recommended standards have been recently proposed to make the web more and more meaningful, interactive and intelligent. As envisaged by web service and ubiquitous computing research, software applications in the future are highly distributed and cooperative. Even though it has already been much easier to write a web application than before, current programming paradigm makes it difficult to design and implement software systems that will function gracefully in the dynamic network environment where they are situated. In other words, conventional programming gives programmers limited patterns in constructing next-generation computer software; and that the computing capabilities of the resulting software are automatically inadequate and fragile to the changing environment. It is hardly possible for the web to function as one giant brain without soft computing patterns being applied to various levels of programming paradigm. This article presents a Neural-Network based programming paradigm which will change the way that programmers model and implement their software applications.

If some web applications can still wait on old programming paradigms, there is one kind that can not. It is Internet computer game. Internet computer games are the best known and most complex distributed computer systems in the world. It usually deals with thousands or even millions of interactive entities among its distributed web servers. An evolutionary story and game world extensibilities are the key features of the next generation Internet and mobile games. Some properties of such games are high individual intelligence, the ability to learn from its environment, group-evolution, and tolerance of errors and uncertainties.

In conventional Internet RPG (Role Playing Game) games, each game-world objects like a Non-Player Character (NPC), Player-controlled Character(PC), boxes, doors, weapons, or even terrains is modeled as an abstract neuron. Both the timer(story-line), network command or the human players may stimulate some of its input fields. The stimuli might be generated by the game’s physics, network or GUI engine. During each simulation cycle on a local game engine, it executes any activated neurons (a program code associated with a certain game object), which usually read their stimuli, compute according to its current state, generate new stimuli to other neurons and sometimes even take actions (also a program code). On the whole, the driving force of a game engine is the constant firing of stimuli in a neuron network constructed by scripts.

Game-engine practitioners have used scripting technology to add soft computing capabilities to a variety of their engine modules; so that commercially released games will still enjoy a certain degree of online-reconfiguration. Unfortunately, there has been no unified approach for solving this problem. Instead, most games’ network engine[9] still reply on single Clients/Server (CS) architecture; a single one-table central database holds all the status of its game entities; all events and triggers in the virtual game world resides only on the same computer.

This article proposes a unified approach of using Neural Parallel Language (NPL) in a computer game's scripting engine. The conventional approach as described above are limited by the type of popular games such as MMORPG(Massively Multiplayer Online Role-Playing Game) and local networked RTS game (Real Time Strategy). The game world are either local to the game engine or contains only one server or a single session at any given time. Truly distributed Internet games are only at research level. In fact, researchers in the field of mobile computing, Augmented Reality (AR) or Virtual Reality (VR) are the major groups who are exploiting such possibilities in future computer games. One problem that must be dealt with by the AR or VR researchers is that the 3D world and stimuli information is gathered from a variety of physically distributed hardware by means of high-speed indoor/outdoor communication technologies such as blue-tooth and GPS, and that a synthesized virtual world must be constructed and simulated from these distributed sources.

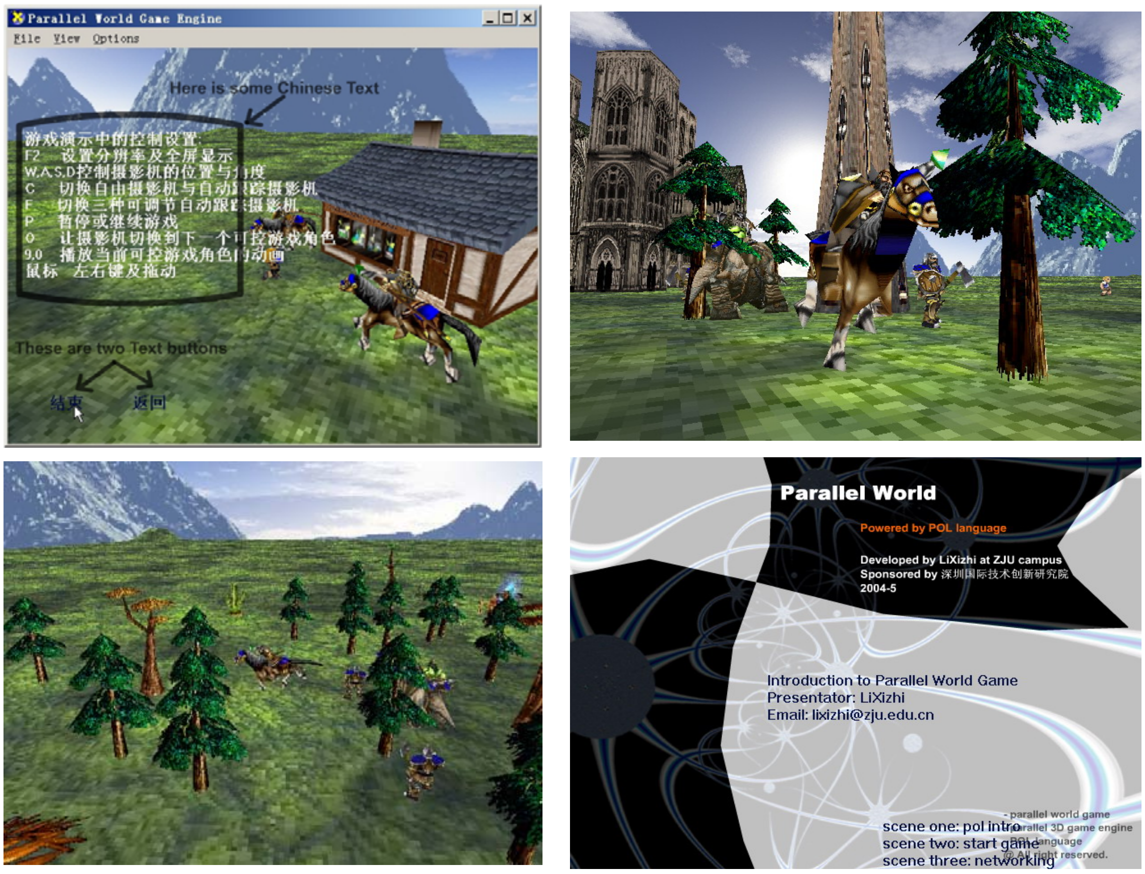
In our vision, Internet would reside inside a computer game, and computer game would become Internet. Like current web pages and hyperlinks, everyone would get a chance to develop his own 3D virtual world (game), link to other virtual worlds[4], and establish complex game-world objects interactions (logics) between these distributed worlds which are authored by different people. From the current technologies such as knowledge presentation on the Internet, 3DWeb technology[5][6], communication hardware and software, we concluded that such kind of world-constructing Internet game is not only feasible but is likely to become popular soon.

1.1 The NPL Methodology

In this article, we will propose a neural-network based programming paradigm (or NPL) for constructing highly distributed and dynamic software systems. We will demonstrate its usage through a distributed computer game called *Parallel World* which is our first formal application of Neuron Parallel Language (NPL). We also believe that: as the web is coming more and more computable (the semantic web) and intelligent, neural network based programming paradigm is likely to become the solution to general purpose distributed software applications (not just computer games).

The key idea of NPL is that software system in the future functions more like one giant brain spanning across the entire Internet. It could form new neuron connections, learn from experiences, remember patterns and perform many other functions resembling the human brain. Current object-oriented programming language lacks the directness in composing such kind of software systems, nor is any existing Neuron Network Simulation Language eligible for constructing commercial distributed software. NPL tries to solve these problem by means of (1) keeping all communication, network deployment and certifications details out of the program code, (2) presenting programmers a very clean neuron network based programming paradigm, (3) preserving all previous familiar paradigms such as object oriented or functional programming. By using NPL, software is constructed like designing a brain network. When writing or debugging their code, programmers do not need to concern about the actual physical environments where their software will be eventually deployed (e.g. a huge distributed virtual game world might exist over 1000 servers on the Internet and would be ever expanding). Instead, tasks concerning the actual hardware, communication protocol, certification and ontology (such as for exchanging meaningful information) will be specified not in program code, but in the visual compiler and runtime environment of the enabling language. For example, it is the task of the visual compiler to distribute/deploy an integral neuron network written in NPL to separate locations(runtimes) of the physical network. In theory, the only atomic structure that is unbreakable by NPL's visual compiler is a single Neuron.

We will see how the NPL will be used in constructing our game *Parallel World* which is a 3D Internet RPG game with game worlds hosted on multiple servers or the clients. The idea of *Parallel World* Game is to give each user the ability to compose their own 3D world (and story) which could link to and interact with game worlds on other game servers or other Internet users' computer. User's game worlds are not only distributed and linked like web pages, but could establish complex logics between game elements (such as characters, doors, terrains) from different network locations. This is due to neural channels between these game elements. Everything in *Parallel World* are scripted by NPL such as game menus, buttons, dialog boxes, sounds, 3D/2D graphics, animations, cinematic, communications with foreign neural networks, etc. See Fig 1.



游戏演示中的控制设置:
F2 设置分辨率及全屏显示
W.A.S.D控制摄影机的位置与角度
C 切换自由摄影机与自动跟踪摄影机
F 切换三种可调节自动跟踪摄影机
P 暂停或继续游戏
O 让摄影机切换到下一个可控游戏角色
9.0 播放当前可控游戏角色的动画
鼠标 左右键及拖动

Parallel World
Powered by POL language
Developed by LiXizhi at ZJU campus
Sponsored by 深圳国际技术创新研究院
2004-5

Introduction to Parallel World Game
Presentator: LiXizhi
Email: lixizhi@zju.edu.cn

scene one: pol intro
scene two: start game
scene three: networking

Figure 1. Screen shots from *Parallel World* game: Everything is scripted using NPL and might be deployed on as many computers as the number of neurons used in composing the game world itself.

Players might walk around, talking with other NPCs(None-player-characters), complete complex tasks all in an continuous infinitely large game world. It is like browsing 3D web pages[6] on the Internet, however, it is more interactive, purposeful and fun.

1.2 An analogy to Imagery

Behind graphics and sounds which are brought by *ParaEngine* (name of our 3D game engine associated with NPL), there is a huge neuron network functioning on the Internet. In order to better illustrate the relationships between the game engine, world logics, and the Neuron Networks; I will draw an analogy of the programming paradigm of NPL to a theory (hypothesis) of the human brain concerning Imagery [2] on cognitive science. In my own version of this theory, it states that human imagination and visual/auditory perceptions are in essence the same thing in our conscious mind, and that they are both the input and output of the unconscious mind which does the work of recognition, memorization and deduction. The cycle of imagination and the subconscious forms mostly a closed loop when we are asleep, and a biased loop (by what we perceive) when we are awake. See Fig 2.

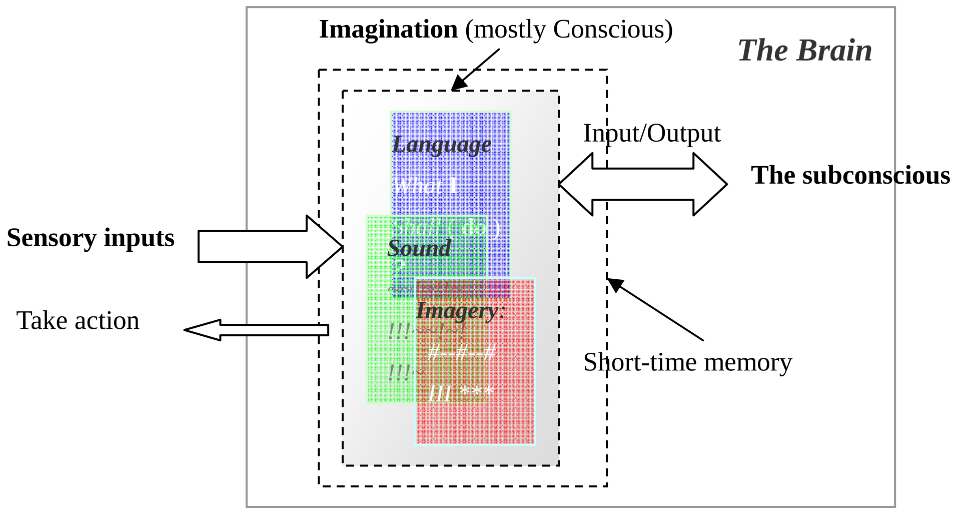


Figure 2. Human Brain: The Imagery-subconscious loop

Metaphorically speaking, the Imagination can be thought of as a multimedia, virtual reality "theatre" [7], where stories about the body and the self are played out. These stories,

- are influenced by the present situation according to perception,
- elicit the subconscious activities accordingly,
- and thereby influence the decisions taken by action.

In our programming paradigm, we use Neuron Parallel Language (NPL) to construct neuron network that maps to both the imagination and subconscious part of the human brain. However, the imagination represented by the state of a Neuron Network can not be visualized such as those seen in Fig 1 by itself. Instead the responsible neurons must tell the Game Engine game-related information upon activation whenever an internal state has been reached. This is done through a set of game specific API called Host API. We use a dual programming language model in our architecture, please see the following sections for more information. The 3D game engine will then present the Imagination in cutting-edge multimedia forms to the user. The user might interact with these objects. The game engine then translate such interaction (sensory inputs) to valid neuron stimuli. And the whole system will be functioning as seen in Fig 1 with text, buttons, graphics and sounds.

The neuron network is simulated by the NPL runtime environment on many distributed servers on the Internet, and there might be even more client computers that are connecting and disconnecting their local neuron networks with each other and some game servers. Each client computer has their own local game engine visualizing a certain configuration of the Imagination. So if we compare the entire game world on the Internet to one huge brain, then this brain's Imagination was visualized in many different ways concurrently. So if we count the conscious of each human player, then the actual brain model will become one huge subconscious on the network and many conscious on the terminals.

1.3 Related works

The relevant works falls into two categories. One is from Neuron Network Simulation researches on methodology (such as discreet event simulation paradigm [8]) and tools (GENESIS [3]). The other is from projects on Networked Virtual Environment[1] (Net-VR) (such as DIVE, NPSNET-V [4]) which have overlapping visions with us. First of all, NPL is not another Neuron Network Simulation toolkits. The essence of NPL is a neuron-network based programming paradigm for general purpose (commercial) software development such as computer games. In NPL, traditional (fixed) neuron prototype as in NN simulator is expanded broadly to be more like a software agent. Current NPL implementation as in our game engine concerns more on the following issues: neuron input fields specification (Ontology), cross-network neural message transmission (XML) and security(NPL runtime certification), automatic neuron network memory management (database support) and learning, neuron network deployment, dynamic neural connection establishment on the network, etc. Secondly, NPL is not a component or a library that is released for an existing language platform such as Java or C++. For example, there are many Distributed Virtual Environment implementations which are built directly on top of Java. They took advantages of the java platform, however, they sacrificed the directness in expressing a neuron network in a clean NN language. Java and many other popular programming platforms are designed with one dominant top-level paradigm: that is Object-Oriented. In this paradigm, everything is modeled as data and function pair. To construct a neuron network in Java, one must indirectly call functions through an object-oriented software interface. This leads to complex interface design and is difficult to use and upgrade. In NPL, we will have a clean Neuron network based syntax which resembles the actual network and hide all communication details from its code. By doing so, there will be no complex object interfaces that programmer needs to learn. NPL principle is that: as software is becoming more and more intelligent, so shall our programming language. In reality, the current NPL implementation adopts a dual-programming model [10]. In this model, there are two distinct language systems: one is host language, the other is extension language. These two language system could communicate at runtime through user-defined Host API. In our game engine, the host language is C++, and the extension language is NPL. C++ is chosen for its high performance and DirectX SDK support as required in our game engine.

Other related works include Augmented Reality and Virtual Reality projects such as Pacman [11],. However, the main focus of these projects is on framework design that maps hardware to software. In most cases, a single Client/Server architecture is used. In NPL and our Game Engine, our focus is mainly on an evolving game world or a computing environment (NN network) that will be as distributed as web pages.

2 Neural Parallel Language and ParaEngine

In order to create games that support distributed game world composing functionality on the Internet, we must design and implement (1) a new 3D game engine that supports these game features (2) a programming paradigm that could describe(compose) such game world logic and (3) integrate the programming paradigm into the game engine. We have implemented our own 3D game engine which is called **ParaEngine** in the following text. And the programming paradigm is actually inside a new programming language we devised (but not implemented completely) called Neural Parallel Language or **NPL** for short. The integration is illustrated in **Parallel World**[1]**,** a game demo we released.

This article mainly deals with the second problem: the NPL programming paradigm. As described previously, RPG game engine is, in essence, powered by a neuron network which are mapped to game world objects such as NPC, PC, doors and terrains. In conventional methodology, the entire neuron network must be constructed by scripts beforehand and simulated on a single computer. In NPL, however, neuron network can be constructed on a single computer, but run on arbitrary computers so long as atomic neuron resides in one runtime(computer). Existing neuron network can be arbitrarily extended by each hosting node(computer) or a central authorized server. Basic functions of neuron network will be internally supported by the language runtime environment, such as pattern learning and database-assisted memory. Memory of neuron network in the game implementation ensures that even when a client is disconnected from the Internet, it can still play in a certain kind of off-line mode, because both the game scene and some simple logic can be remembered by the local neuron network. Sometimes, neuron network is explicitly designed to take advantage of this feature. For instance, some game world data will be transmitted only if the client can not recollect (remember) a proper version of it.

This section to be extended for long paper preparation: Topics that will cover is (1) NPL paradigm, (2) ParaEngine, (3) Design Neural Network for Parallel World Game using NPL (4) demo code. The entire game demo (text in Chinese) can be downloaded from my website, see the footnote.

3 Conclusions

Future software would be designed under the premise that all front end software run in a distributed environment and (co)operate in a manner similar to neural networks. Computer game is one good example. It would play an important role in joining the real world with the virtual world. It is likely that the entire Internet would be inside one huge game world. Another possibility is that: the model of neuron network would not only be applied in micro scale but also in macro scale applications. As the web is coming more and more computable (the semantic web) and intelligent (agent technology, MAS), neural network based programming paradigm as described in this article is likely to become the solution to general purpose distributed software applications. NPL is one recent effort in bring general purpose programming to a level that most resembles our human brain.

[1] ParaEngine, Parallel World demo can be downloaded from my website: http://www.lixizhi.net/projects.htm#paraengine For more information not covered in this literature, one can find in the author’s website or by emailing him.

To editors: This article is only recently written and needs to be refined. If you think it proper to be included in your conference, I shall be glad to refine it as soon as possible. Thank you.
Email: LiXizhi@zju.edu.cn

Reference:

[1] Singhal, S., and Zyda, M. (1999). Networked Virtual Environments: Design and Implementation, ACM Press.

[2] Henry C.Ellis and R.ReedHunt. Fundamentals of Cognitive Psychology. MC Graw Hill. ISBN: 0-697-10543-1

[3] GENESIS : The GEneral NEural Simulation System Version 2.2.1. http://www.genesis-sim.org/GENESIS

[4] Capps, M.; McGregor, D.; Brutzman, D.; Zyda, M. NPSNET-V: A New Beginning for Dynamically Extensible Virtual Environments. IEEE Computer Graphics and Applications 20(5): 12-15 (2000).

[5] Jed Hartman and Josie. The VRML 2.0 Handbook: Building Moving Worlds on the Web Wernecke (1996) Addison-Wesley. ISBN 0-201-47944-3.

[6] Web3D Consortium. http://www.web3d.org/

[7] Murray Shanahan. The Imaginative Mind A Precis. Conference on Grand Challenges for Computing Research (gcconf04)

[8] Simon J.E. Taylor, Boon Ping Gan, Steffen Straßburger and Alexander Verbraeck. HLA-CSPIF Panel on Commercial Off-the-Shelf Distributed Simulation. Winter Simulation Conference 03'.

[9] Joseph Manojlovich, Phongsak Prasithsangaree, Stephen Hughes, Jinlin Chen, and Michael Lewis. UTSAF: A Multi-Agent-Based Framework for Supporting Military-Based Distributed Interactive Simulations in 3D Virtual Environments. Winter Simulation Conference 03'.

[10] R. Ierusalimschy, L. H. de Figueiredo, W. Celes. Lua-an extensible extension language. Software: Practice & Experience 26 #6 (1996) 635-652.

[11] Adrian David Cheok, Siew Wan Fong, Kok Hwee Goh, Xubo Yang, Wei Liu, Farzam Farzbiz, and Yu Li. Human Pacman: A Mobile Entertainment System with Ubiquitous Computing and Tangible Interaction over a Wide Outdoor Area. Mobile HCI 2003, LNCS 2795, pp. 209–224, 2003.

[12] Ruck Thawonmas and Takeshi Yagome. Application of the Artificial Society Approach to Multiplayer Online Games: A Case Study on Effects of a Robot Rental Mechanism. ADCOG 04'

[13] Rogelio Adobbati, Andrew N. Marshall, Andrew Scholer, Sheila Tejada, Gal Kaminka, Steven Schaffer, and Chris Sollitto. Gamebots: A 3D Virtual World Test-Bed For Multi-Agent Research, Proceedings of the International Conference on Autonomous Agents (Agents-2001) Workshop on Infrastructure for Agents, MAS, and Scalable MAS, Montreal, Canada, 2001.

[14] Tveit, A., Rein, O., Iversen, J.V. and Matskin, M., Scalable Agent-Based Simulation of Players in Massively Multiplayer Online Games, Proc. The 8th Scandinavian Conference on Artificial Intelligence (SCAI2003), Bergen, Norway, Nov., 2003.

[15] Manninen T. Interaction in Networked Virtual Environments as Communicative Action - Social Theory and Multi-player Games. In proceedings of CRIWG2000 Workshop, October 18-20, Madeira, Portugal, IEEE Computer Society Press

[16] Bowman, D. A., and Hodges, L. F. (1999). "Formalizing the Design, Evaluation, and Application of Interaction Techniques for Immersive Virtual Environments." Journal of Visual Languages and Computing, 10, 37-53.