# "Input here – Execute there" through networks: the case of gaming[1]

Christos Bouras Vassilis Poulopoulos Ioannis Sengounis Vassilis Tsogkas
Research Academic Computer Technology Institute, Riga Feraiou 61, 26221, Patras, Greece
Computer Engineering and Informatics Department, University of Patras, 26500, Rion, Greece
bouras@cti.gr poulop@cti.gr jns@sch.gr tsogkas@cti.gr

***Abstract* —As the evolution of computer technology introduces new advances in networks (among others), online gaming becomes a new trend. Following the trends of our era, Games At Large introduces an innovative platform for running interactive, rich content multimedia applications over a WAN Network. Games@Large project's vision is to provide a new system architecture for Interactive Multimedia which will enhance existing CE devices such as, Set Top Boxes (STB) and other devices, which are lacking both the CPU power and the graphical performance to provide a rich user experience. This paper presents the controllers' sub-system of the innovative mechanism that will be implemented in the context of Games at Large project. More specifically, it presents the general architecture of the complete system and focuses on the "capturing" and "execution of commands" modules at the client and server side. The client software captures the input from the input devices, sends the commands over a WAN and the server is responsible for receiving and executing the commands to the correct application.**

***Index Terms*—remote control, online gaming, remote command execution, input device capturing**

## I. Introduction

COMPUTER games constitute nowadays one of the most dynamic and fastest changing technological area, both in terms of market evolution and technology development [1]. In this area, as the computer games are evolving and online activities and gaming become parts of our life, the need of interaction within a client – server architecture becomes very intense. In this document we present a mechanism for transferring input commands from any device, acting as the client, to execution commands at the corresponding program of the server. The purpose of this mechanism is to be able to control a program that runs on the centralized server from a remote operating system. This mechanism is created within the scope of the Games at Large project. Meeting the demand of highly interactive multimedia systems with low cost end devices (CE), requires a radical change in the system's architecture. Games@Large project intends to design a platform for running interactive rich content multimedia applications. Games@Large vision is to provide a new system architecture for Interactive Multimedia which will enhance existing CE devices such as, Set Top Boxes (STB) and other devices which are lacking both the CPU power and the graphical performance, to provide a rich user gaming experience.

In this paper we present the general architecture of the sub-system that controls the input of the devices and their server side execution. More specifically, we examine how, input is able to be captured by any input device, commands are sent over the network and finally, commands are executed at the target software of the server.

## II. Games At Large

The Games@Large[1] mission is to develop a new method for ubiquitous video games through unique technology to transfer graphical data while reducing latency and ensuring QoS in a cost-effective manner. Main focus will be given on studying and supporting the use of video games within four different focus areas: User's home, Hotels, Internet Café, Elderly Houses. A multi-layer approach will cut horizontally across the Games@Large focus areas, aiming to assess the conditions under which a Games@Large platform may frame within and improve the state of the art of each business domain, through performing the following, logically consecutive activities: collecting user requirements, researching and developing common Technologies, implementing and integrating those technologies within the required Servers and prototype CE Devices, running technology verification and Training and evaluating all results.

## III. General Architecture

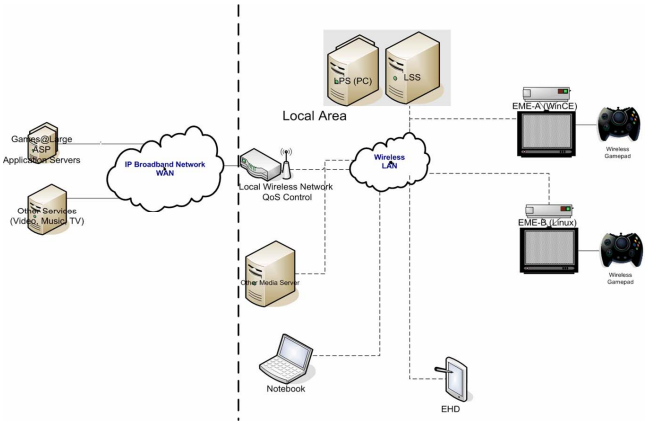


**Figure 1: System General Architecture**

The Games@Large system architecture scheme is depicted in Figure. 1. It consists of the service provider side, on the left, and the end-users' side, which are connected through a broadband distribution network (the Internet). The

[1] Games@Large Project, IST-1-038453-IP, Funded by EU, http://www.gamesatlarge.eu

infrastructure for end-user environment will be formed by the wireless LAN (802.11 a/g/n or other methods to be investigated within the project). This local network connects devices that are involved in the multimedia chain, either as media storage/processing servers, or as consumer user terminals [2].

## IV. Client Side And Server Side

The infrastructure that we are developing, as depicted in figure 2, executes the following procedure:

- Registers the input controllers on the client side.
- Captures all input from the controller devices.
- Formats input in the appropriate style for transferring.
- Transfers the controller's input to the server using a socket connection.
- Executes the input commands locally on the server.

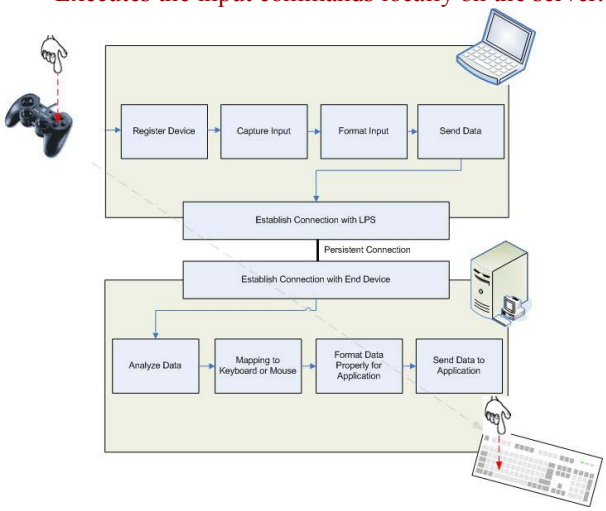


**Figure 2: Graphical representation of the input controller's data transmission**

When the client program starts, it initiates the device discovery procedure, which may be offered either by a separate architectural module, e.g. the device discovery module which uses UPnP, or by a system call causing the discovery for input devices attached to the system. It is essential afterwards, that the results of the device discovery are registered in our program so that we are aware of the existing input devices marking out several other non-existing.

The next step of the procedure is to capture the input coming from the input controllers. This is achieved by recording the key codes coming from the input devices. Input devices such as mice or keyboards are interrupt-driven while with joysticks or joy pads the polling method is used for reading. The previous means that whenever an input event is caused by a keyboard or a mouse, an interrupt message is sent to the message queue of our program; then it is translated and finally recorded. However, the polling case of joysticks or joy-pads means that these devices have to be polled by a program's thread in order to sense motion or button presses. The polling period has to be small enough to capture any input, but not too small to monopolize the system's CPU. A period of 10ms seems to be a wise trade off.

After an input key code has been captured, the transmission of it takes place. This is achieved using an already open socket connection with the server side. Data is transmitted through the socket in the form of a string with a certain communication protocol.

The socket connection can either be of TCP or UDP protocol. Since UDP emphasizes on real time, low latency transmission, it is preferable for this transmission. Even if some key codes are lost in the process of transmitting them over the network, there is no real loss since there is a flow of key codes that can overcome this possible threat. However, in real life, error prune networks, such as WiFi's, the TCP protocol is preferred avoiding the possible game experience fall caused by lost controller's packets transmission.

Since the key codes have arrived at the server side, they are executed at the running game instance. At this point, there needs to be a distinction between the different types of transmitted key codes. There are basically three types of possible input device's data transmission. Commands may be coming from: (a) keyboard, (b) mouse, (c) joystick / joy-pad device or (d) any other HID input device.

In the first case, the server has to recognize the virtual key code, or the "pressed" / "released" event of a keyboard button, then do a possible mapping to some other key code, based on the game and user profile, and finally deliver it to the active application window for execution.

In the case of mouse input, the server has to recognize the virtual key code or the "pressed" / "released" event of a mouse button, recognize any mouse wheel event or any mouse movement (absolute or relative), then do a possible mapping to some other key code, based on the game and user profile, and finally deliver the key code to the active application window for execution.

In the case of joystick/joy-pad input, the server recognizes the state of the joystick/joy-pad device, maps the state to the appropriate keystrokes using the xml – mapping file of the particular game-joystick/joy-pad combination. In this way, we are able to emulate the joystick/joy-pad input using pure keystrokes–mouse movements that represent the actual behavior of the input device. Finally the key code is delivered to the active application window for execution.

For any other HID input device the system treat the input similarly to the joystick/joy-pad. The only prerequisite is the existence of a mapping file in order to convert the commands to keyboard and mouse instructions..

## References


[1] G. O. Young, "Synthetic structure of industrial plastics (Book style with paper title and editor)," in *Plastics*, 2nd ed. vol. 3, J. Peters, Ed. New York: McGraw-Hill, 1964, pp. 15–64.

[2] Y. Tzruya, A. Shani, F. Bellotti and A. Jurgelionis. "Games At Large: A new platform for ubiquitous gaming", BroadBand Europe 2006, Geneva, Switzerland, December 11 – 14, 2006

[3] "Games at Large: Project Summary", 20 December 2006, http://www.gamesatlarge.eu