***Multimedia Systems –***

**C-WG3#1**

**Project Plan**

File: Project Plan template.doc Last Modified: 28.11.2009 Approval Date:

Created: 1.11.2009 Number of Pages: 10 Approved By:

**PURPOSE OF THIS DOCUMENT**

This document contains the project plan for project C-WG3#1 – Cloud Gaming. The project is done for Multimedia systems-course in University of Oulu.

**REVISION HISTORY**

|  |  |  |  |
| --- | --- | --- | --- |
| Revision # | Date | Author(s) | Comments |
| 0.5 | 1.11.2009 | C-WG3#1 | Initial version |
| 1.0  1.1 | 2.11.2009  28.11.2009 | C-WG3#1  C-WG3#1 | More spesific version  Small fixes, typos corrected |

**ABBREVIATIONS**

|  |  |
| --- | --- |
| MMS | Multimedia systems |
| Yukon | Framework to capture video from OpenGL games. |
| HTTP | Hyper-text Transport Protocol |
| UDP | User Datagram Protocol |
| VLC | Videolan Client, an open source video streaming solution |

**REFERENCES**

Yukon Framework - <https://devel.neopsis.com/projects/yukon/>

LibLVC with Qt Example - <http://wiki.videolan.org/LibVLC_SampleCode_Qt>

**TABLE OF CONTENTS**

1 INTRODUCTION 4

2 Project DESCRIPTION 5

2.1 Main Tasks 5

2.2 Organization and resources 6

2.3 Scheduling 6

2.4 Methods and tools 6

3 TASK descriptions 7

3.1 T1: Capturing and streaming video 7

3.2 T2: Receiving and showing video stream 7

3.3 T3: Sending game controls to server 8

3.4 T4: Receiving controls and relaying them to the game. 8

3.5 T5: Putting all together 9

# INTRODUCTION

Our aim is to create a solution for playing games with cloud architecture. The game is run on a server instead of your own personal computer, and its video is streamed to your computer over the net. Your game controls will be also forwarded in the process. The playing experience is aimed to match what you would have if the game was run locally. Cloud architecture makes possible to play power consuming games on a relatively powerless computer, so you wouldn't have to invest in state-of-art hardware anymore, because video rendering is done on server. Possible future developments could also allow the game to access many servers at once to receive more calculation power if needed by the game.

In this particular project we decided to use Yukon framework to capture video from OpenGL – based games, such as DarkPlaces Quake open source engine on Linux. Video is then encoded and transmitted as a video stream from the server. If the client wants to play, he starts our project application which then connects to the stream and allows the user to control the game remotely. The game experience should be sufficiently good and offer an plausible alternative for gaming on your personal computer. There will be some amount of latency between the user and game because of the video stream and controls transmitted via network, but we aim to keep it to a minimal. Otherwise playing could become if not impossible, really hard. Our application will be demoed by a front-person-shooter game, Quake. Because of the genre, response times are critical. We will be using an open source clone engine of the game, allowing it to run on Linux platform and offer control receiving functions to be incorporated in the game. In the server end will basically be running a framework script which starts the game, command receiving and video relay. Client application will be built on Qt – base, which allows possible future portability to other platforms, such as mobile phones and other operating systems.

# Project DESCRIPTION

## Main Tasks

More specific review of project tasks is given in Chapter 3 of this document.

T1: Capturing and streaming video

Server is able to capture video from game and encode it for streaming over net to client.

T2: Receiving and showing video stream

Client can remotely connect to video stream and display it in a application window. Possible

Connect/disconnect feature to start-up and stop streaming in the server end.

T3: Sending game controls to server.

Client sends game controls to server as UDP data packets. At least keyboard input will be available in the application.

T4: Receiving controls and relaying them to the game.

Server receives controls and forwards them to game window. Client won't be able to access anything other than the game itself.

T5: Putting everything together

Testing possible use scenarios and optimizing game experience.

## Organization and resources

Responsible supervisor

Otso Kassinen ([otso.kassinen@ee.oulu.fi](mailto:otso.kassinen@ee.oulu.fi))

Project manager

Lauri Majamaa ([laurimaj@ee.oulu.fi](mailto:laurimaj@ee.oulu.fi))

Project team and its participation in main tasks

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | T1 | T2 | T3 | T4 | T5 |
| Lauri Majamaa | o | o | o | o | o |
| Tuomas Vähänen | x | o |  | x | o |
| Antti Väyrynen |  | x | o |  | x |
| Antti Lampela |  | x | x | o | o |
| Pasi Keski-Korsu |  |  | x | x | x |

**x** = main responsibility

o = assist if needed

## Scheduling

Project time span: 19.10.2009 – 9.12.2009

Main project events and deadlines:

|  |  |
| --- | --- |
| Date | Description |
| 19.10. | Project starts |
| 27.10. | Meeting project owner (supervisor) |
| 2.11. | Mid-way milestone and presentation**,** project plan ready |
| TBD | Project preview to the supervisor ( to be decided) |
| 2-9.12. | Final presentation |
| 9.12. | Material returned to the supervisor (CD/DVD) |

## Methods and tools used

Design and programming methodology: Agile coding – SCRUM

Programming environments: C++, Qt

Version control: SVN

Documentation: Open Office 3.0, Office 2007

Document management and distribution: www.ee.oulu.fi/~laurimaj/MMJ/

# TASK descriptions

## T1: Capturing and streaming video

General description

In this task we are going to use Yukon framework or some equivalent to capture OpenGL video output from the game. Video is then encoded with H.263 codec or another lightweight and streamed via HTTP or UDP protocol over the net.

Work allocation

Tuomas Vähänen

Result

Video stream of the game can be remotely accessed.

Subtasks

|  |  |  |  |
| --- | --- | --- | --- |
| Subtask | Description | Deadline | Resources |
| 1.1 | Working video capture | 13.11. |  |
| 1.2 | Stream accessible from client | 20.11 |  |
| 1.3 | Optimization | 2.12. |  |

## T2: Receiving and showing video stream

General description

Client application connects to a video stream on the server and displays it in the application window. Game video should be an equivalent to running the game on the client instead of server.

Work allocation

Antti Lampela, Antti Väyrynen

Result

Stream can be viewed directly from client application.

Subtasks

|  |  |  |  |
| --- | --- | --- | --- |
| Subtask | Description | Deadline | Resources |
| 2.1 | Connect to video stream | 13.11. |  |
| 2.2 | Video displayed correctly in application window | 20.11. |  |

## T3: Sending game controls to server

General description

Client program reads keyboard input and possible mouse and then sends them as UDP data packets over the net to the server. Controls should be forwarded in as real time as possible for satisfying game experience.

Work allocation

Antti Lampela, Pasi Keski-Korsu

Result

Game controls can be sent to server.

Subtasks

|  |  |  |  |
| --- | --- | --- | --- |
| Subtask | Description | Deadline | Resources |
| 3.1 | Reading input | 13.11. |  |
| 3.2 | Using UDP socket to send commands to server | 20.11. |  |
| 3.3 | Optimization | 2.12. |  |

## T4: Receiving controls and relaying them to the game.

General description

Server reads UDP Socket for incoming data packets containg game commands from the client. Commands are then relayed to game program. Client will have command access to the game only, and not be able to mess with the server.

Work allocation

Antti Lampela, Tuomas Vähänen

Result

Game controls can be received and they can be forwarded into the game.

Subtasks

|  |  |  |  |
| --- | --- | --- | --- |
| Subtask | Description | Deadline | Resources |
| 3.1 | Reading socket and unpacking commands | 13.11 |  |
| 3.2 | Commands forwarded to game | 20.11 |  |
| 3.3 | Optimization | 2.12. |  |

## T5: Putting all together

General description

Testing both video streaming and game controls forwarding together. At this point we will see if there is something to improve in our design, for example control response times and video lag. Specifications and other documents will also be finalized.

Work allocation

All project members

Result

Working client and server and enjoyable gameplay.

Subtasks

|  |  |  |  |
| --- | --- | --- | --- |
| Subtask | Description | Deadline | Resources |
| 3.1 | Testing | 27.11. |  |
| 3.2 | Optimization | 2.12. |  |