Project Plan: Multiplayer XKobo clone (Mobo)

CO3015 COMPUTER SCIENCE PROJECT | DEPARTMENT OF COMPUTER SCIENCE, UNIVERSITY OF LEICESTER, OCTOBER 2015

CHRISTOPHER JOHN COLA

INTRODUCTION

XKobo is an action space-shooter style game released in 1995 by Akira Higuchi for UNIX-based operating systems. The goal of the XKobo is to eliminate all fortresses in each level. Fortresses are comprised of interconnected nodes surrounding a core, all of which need to be fired upon and destroyed in order to eliminate the fortress itself. The fortresses defend themselves by firing at the player from the nodes in return and thus the game's challenge is born out of the player needing to skillfully dodge enemy fire whilst also attempting to destroy all the nodes of a fortress.

AIM

This project's aim is to produce a game characteristically similar to XKobo but with the addition of multiplayer online features. These features will take the form of either cooperative or competitive play, where players are able see and interact each other ingame. One of the main focuses is to produce an multiplayer experience that is synchronised and fluid where problems with latency are mitigated using networking techniques such as input prediction and lag compensation. The project also aims to extend the base game along with adding the necessary features and polished user experience one might expect of a modern day title. The game will be programmed using the Microsoft XNA framework which will facilitate development of common game features like graphics and has some inbuilt networking features.

OBJECTIVES

- ❏ Familiarise with C# and chosen game framework.
- ❏ Research online game networking and evaluate libraries available.
- ❏ Utilise 2D graphics and game logic to produce a game similar to XKobo.
- ❏ Develop a client and server with network protocols.
- ❏ Perform usability testing to gauge the quality of the user experience.

For the first objective, it will be necessary to familiarise myself with the C# programming language and the framework chosen as quick as possible. Likely having not previously used either, I intend to begin the project by learning the basics of the framework and to build my confidence in the language by programming minor functions of the game that will nevertheless be required in the final product such as the player movement, GUI menus or fortress generation. Following shortly afterwards, I hope to create a stripped down (in comparison to the final product) but playable game that will form the basis for creating the multiplayer networking component.

Of high importance will be researching modern multiplayer networking techniques for client-server architectures. Finding the optimal solution to designing and developing synchronous multiplayer within the scope of the project and one that is possible given the technical limitations will be critical to the success of the project.

FUNCTIONAL REQUIREMENTS

ID	Requirement Description	MoSCoW Rating
FR-001	Controllable by mouse and keyboard.	MUST
FR-002	Controllable by gamepad.	COULD
FR-003	Allow users to change resolution.	SHOULD
FR-004	Have a title screen with buttons with navigation to starting a game, viewing instructions, settings, credits and exiting.	MUST
FR-005	Have a submenu for starting a game that allows a user to select between starting a single-player and online game.	MUST
FR-006	Allow users to change user-specific settings.	MUST
FR-007	Allow users to view instructions for the game.	MUST
FR-008	Allow users to view credits for the game.	MUST
FR-009	Allow users to exit the game.	MUST
FR-010	Search for available servers and display them.	MUST
FR-011	Allow users to host a game.	MUST
FR-012	Allow users to join a pre-existing game.	MUST
FR-013	Allow users to text chat in the lobby and ingame.	COULD
FR-020	Allow users to move up, down, left and right using the arrow keys on the keyboard.	MUST
FR-021	Allow users to aim and shoot using the mouse.	MUST
FR-022	Display an in-game HUD with health and score.	MUST
FR-023	Display a minimap showing the player, enemies and hazards.	MUST
FR-023	Detect collisions between projectiles and entities and act accordingly.	MUST
FR-024	Procedurally generate 'fortresses' based on an algorithm and a size.	MUST
FR-025	Procedurally generate player hazards such as asteroids.	SHOULD
FR-026	Track player position so 'fortresses' can aim and shoot at the player.	MUST
FR-027	Advance the player to the next level when all the fortresses on the current level have been destroyed.	MUST
FR-028	Allow users in an online game see each other moving independently.	MUST
FR-029	Allow players to pick up additional weapons and power=ups.	SHOULD
FR-030	Track player statistics such as kills, projectiles fired and score.	SHOULD

The development of the project will begin with the design of the architecture of the client-server system, constituent components, interaction and behaviour. This includes a component decomposition to define the classes and respective methods and relationships between them. The client will take the form of the game itself containing the presentation, application and business logic and a local database model for storing single player specific data. The client can therefore described as a ‘fat’ client as it will provide a significant amount of functionality (all the game logic) without the server. The main tasks for the client will be developing the user interface and graphics of the game, creating the game logic such as movement of the player, generation of fortresses, basic AI for non-player characters and the shooting and projectile mechanics. Of less initial importance but required for the final build will be menus for starting the game, user-centric configuration options such as difficulty and name setting and finally technical configuration options such as graphics fidelity and game resolution.

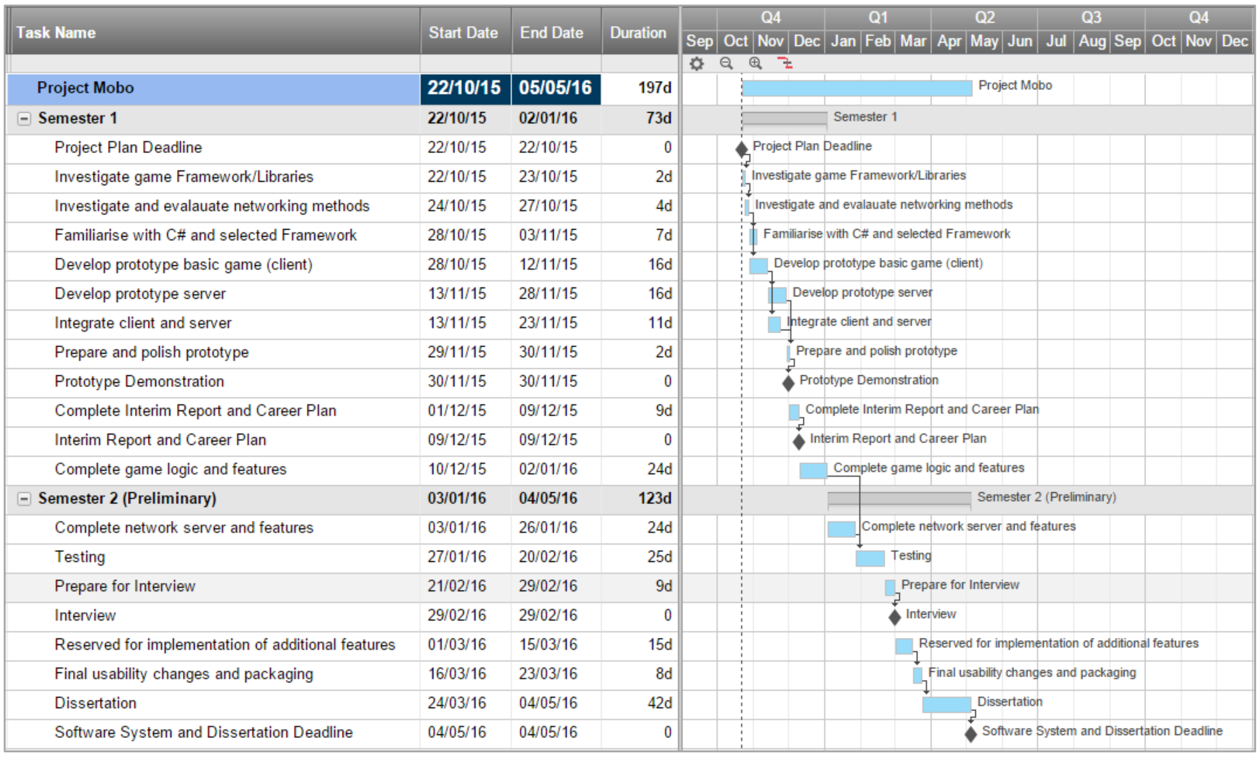
The primary role of the server is to enable online multiplayer. The server will provide positional tracking of elements such as players, enemies and projectiles to the clients. The server should also provide functionality for single player such as the ability to track player statistics between clients and allowing users to log in and continue where they left off should they play the game on different systems. Players will be able to host or join rooms which they can choose to password protect if desired. The server will comprise of a database management system such as MySQL connected to an exposed API that the client will interact with.

During an online game, players will be able to see each other and move around independently of each other. Actions taken by a player, such as shooting and picking up power-ups should be seen and registered by other online players. Since tracking with absolute accuracy is impossible over a network, the client should request an update of each other player’s position a fixed number of times per unit time. This networking system should be smooth; other players should not be moving in ‘jerky’ movements or intermittently, instead the movements should be interpolated.

Testing will take place throughout the development of the project to ensure validity of the various functionality of the game. Unit tests will be written alongside the code and regularly ran to ensure changes made do not impair functionality elsewhere. Once both the client and server have been developed in suitable capacity, integration testing will help discover possible bugs that arise from message transfer between systems. Stress testing will also be carried out to ensure the system can operate under reasonable load. Since the end product is a user-facing game, a usability and HCI survey will be important to highlight usability problems and issues with the user-experience.

Finally, once development and testing have been completed an evaluation report will be written highlighting issues, decisions made and why, and how these related to the project’s goals. It will also include technical analysis for how particular functions of the game work, for example the netcode, and detailed analysis of the usability and HCI testing carried out during the testing phase.

PLANNING AND TIMELINE



Task Name	Start Date	End Date	Duration
Project Mobo	22/10/15	05/05/16	197d
Semester 1	**22/10/15**	**02/01/16**	**73d**
Project Plan Deadline	22/10/15	22/10/15	0
Investigate game Framework/Libraries	22/10/15	23/10/15	2d
Investigate and evalauate networking methods	24/10/15	27/10/15	4d
Familiarise with C# and selected Framework	28/10/15	03/11/15	7d
Develop prototype basic game (client)	28/10/15	12/11/15	16d
Develop prototype server	13/11/15	28/11/15	16d
Integrate client and server	13/11/15	23/11/15	11d
Prepare and polish prototype	29/11/15	30/11/15	2d
Prototype Demonstration	30/11/15	30/11/15	0
Complete Interim Report and Career Plan	01/12/15	09/12/15	9d
Interim Report and Career Plan	09/12/15	09/12/15	0
Complete game logic and features	10/12/15	02/01/16	24d
Semester 2 (Preliminary)	**03/01/16**	**04/05/16**	**123d**
Complete network server and features	03/01/16	26/01/16	24d
Testing	27/01/16	20/02/16	25d
Prepare for Interview	21/02/16	29/02/16	9d
Interview	29/02/16	29/02/16	0
Reserved for implementation of additional features	01/03/16	15/03/16	15d
Final usability changes and packaging	16/03/16	23/03/16	8d
Dissertation	24/03/16	04/05/16	42d
Software System and Dissertation Deadline	04/05/16	04/05/16	0

The Gantt chart shows the project schedule over the two semesters. As mentioned in the objectives, the project begins initially with research into graphical libraries and networking. The duration for these investigatory tasks is the maximum possible, and will likely not take as long. If suitable libraries or methods are found early, implementation will begin early. The prototype, due to be completed by late November, will resemble a stripped down version of the final game, with enough functionality to be able to showcase basic networking such as messaging and movement synchronisation. In addition it would be desirable to have hosting and joining games in place for more than two players.

The project is not without risks. The first semester will be very development heavy, with relatively short sprints of development allowed. There is a large risk of tasks taking too long, especially the development of the server of which less than three weeks has been allocated. Another potential issue is that the chosen game library or framework might already have some functionality for network play and therefore it may not be as easy to control lower level components without programming network features in the way the framework requires. This potential loss of control makes finding suitable technologies critical for the project to proceed in time with the schedule.

BIBLIOGRAPHY

1. Lecky-Thompson, Guy W. Fundamentals of Network Game Development. Herndon, VA, USA: Course Technology / Cengage Learning, 2008.
2. Source Multiplayer Networking. Valve Developer Community Wiki, Accessed 19/10/2015, https://developer.valvesoftware.com/wiki/Source_Multiplayer_Networking
3. Rabin, Steve. Introduction to Game Development (2nd Edition). Herndon, VA, USA: Course Technology / Cengage Learning, 2009.
4. Mount David. Varshney Amitabh. Networking and Multiplayer Games Chapter 9 University of Maryland 2011