Sam Dickson
Niraj Venkat
David Zinn
Artemiy Oblassov

Summit: Product Backlog

Problem statement
Rarely do you see multiplayer games with a retro design as retro games used to be fairly solitary experiences meant for one. We aim to fix this problem and lead the way in a new trend to combine the old classics with the replayability of the new multiplayer games.

Purpose of this Document
The purpose of this design document is to provide key details of the *Summit* application. It will summarize its functional requirements, general priorities, outline its design, and discuss design issues.

Background
Within the growing games market there has been a trend towards multiplayer games amongst bigger name studios and a trend towards more retro like games in the independent market. We believe that this will create interest amongst players and spark a demand for fun, simple games for more than one person. This in turn will cause game development studios to try and cater to these wants and allow us to create a new demand while simultaneously giving us an edge in experience and establishing our name in a new exciting field.

Environment
The client will be developed game in Java using LWJGL (Lightweight Java Game Library). The server will similarly be written in Java with a small SQL database used to hold a list of high scores. We will also be using Junit for unit testing for all modules.

Reliability / Accessibility
The server should always be available and easy to contact as to allow players to participate in the game at their discretion.

Performance / Efficiency
Communication between client and server should be efficient and reliable to reduce any delays

Sam Dickson
Niraj Venkat
David Zinn
Artemiy Oblassov

between clients that would manifest itself as lag in the game.

Portability
Easily portable to any device that supports Java. Can be ported to Android easily and efficiently.

Security
The game client needs to handle UDP packets securely to prevent UDP packet spoofing that would allow a hacker to change the position and powerups of the player.

Testability
We designed this game to be fun and easy to test. We may use JUnit to test our Java components and keep track of bugs using a bug tracking system such as BugZilla.

Extensibility
With a well designed and reliable testing plan there should be no issues with adding new features, extending the game experience to the new limits. To go along with maintainability, the game needs to be extensible. It should be easy to add new elements to the game at any time. Our game should be easy to update, upgrade, and modify. In an ideal real-life situation, the game would also be available to sequels. The way we intend to implement the game will allow for more map variations in the future. This could also lead to allowing for more than four players per game, and possibly even different objectives or teams.

Functional Requirements

ID	Requirements	Estimated Hours
1	As a user I would like to join a game lobby	4
2	As a user I would like to create a game lobby	1
3	As a user I would like to start a game	24
4	As a user I would like to view the leaderboards from the main screen	0.5
5	As a user I would like to be able to interact with my character	4
6	As a user I would like to pick up and use powerups in the game	2
7	As a user I would like to interact with other characters	4
8	As a user I would like to interact with the environment in game	3
9	As a user I would like to customize a character model	1
	Total Hours:	1,000,000

Sam Dickson
Niraj Venkat
David Zinn
Artemiy Oblassov

Use Cases

1. Join a game lobby with friends

1. Click on “Find Lobby” button	2. Display the join game dialog box
3. Input the lobby name/password	3. Game room is displayed

2. Create a game lobby

1. Click on “Create Lobby” button	2. Create a lobby password
3. Game room is displayed	

3. Start a game

1. All players click on the “Ready” button	
2. Host clicks on “Start Game” button	3. The game starts

4. View leaderboards

1. Click on “Leaderboards” button	2. Leaderboard is displayed

5. Interact with my character

1. User presses space	2. Character jumps
3. User presses a	4. Character moves to the left
5. User presses d	6. Character moves to the right

6. Pick up powerups and use them

1. A powerup is spawned	2. Player moves character over powerup
3. Powerup effect occurs	

7. Interact with other characters

1. Player moves mouse to aim weapon	
2. Player clicks mouse button to fire weapon	3. Check to see if player is hit
	4a. Character is stunned if hit
	4b. Character continues as normal if not hit

8. Interact with the environment

1. Player lands on ledge	2. Ledge ability triggers if it has one

9. Customize a character model

1. Player chooses a character model	2. Image is updated to reflect choice
3. Player chooses a character color	3. Image is updated to reflect choice