Project Multitris

Group 23
Marcus Dicander
Måns Olson
Tomas Alaeus
Daniel Boström
Oscar Olsson

Requirements Document (RD)

Table of Contents

1. Preface

1.1. Expected readership

Architecture engineers, developers, end users.

1.2. Version history

Version	Summary	Date	Authors
1.0	First version of the Requirements Document (RD)	2007-12-19	Oscar Olsson, Marcus Dicander, Tomas Alaeus, Daniel Boström, Måns Olson

2. Introduction

2.1. User base and project goal

Multitris aims to renew the well-known Tetris (1) gameplay, primarily by enabling web-based multiplayer games. Tetris is a puzzle game where the player controls falling pieces of different shape by rotating them and moving them sideways. The idea is to place them so that complete rows are formed. When a row is complete it is removed. Pieces that are not part of complete rows remain on the board. When the pile of pieces reaches a certain height the game is over, and a score is given to the player. As the game progresses, the game speed increases. Multitris will let the players play together on a single, widened game board. Each player will get their own pieces. The goal will be to co-operate, and create rows over the entire game board. On top of this, we will provide additional functionality such as “powerups”, special bricks that can be activated to change game behaviour and alter the pile of pieces. For example, a powerup could clear a row or remove all pieces from the board. A singleplayer mode will also be included in the game.

There are current multiplayer versions of Tetris, such as Tetrinet (2). There are several limitations to these implementations, the major one being lack of interactivity between players. While players can collect and use powerups, there is no other interactivity. The game boards for each player are separate, so players' pieces cannot affect each other. Multitris will differ from current multiplayer Tetris versions by allowing players to play on the same board. Co-operation is required to efficiently complete rows. The game will be distributed via the web, and the user will simply click a link on a web page to start the game. The user will then have to grant the system permission to run on his computer. The user also has the option to download the client for offline use.

We target *casual gamers*, i.e. people who play games on a non-regular basis. Our intended user is a male between 15-35 with access to a computer network. Ideally, the user has enough technical knowledge to play games via his browser, and will have a Java Runtime Environment JRE (3) installed on his machine. The user runs either Windows XP, Mac OS X, Linux and has a computer that supports hardware-accelerated graphics. Additionally, the user should be able to read and understand English. Familiarity with the game Tetris is also recommended. The user should ideally use the Internet (reads blogs, search portals, or chats) to find web games.

2.2. Main system uses

The main use of the system is recreational. Primarily, the game will be used as a distraction when the user has a short period of time available.

Usage narratives

- Martin is working in his office. Being a casual gamer, he finds Multitris through a game portal during his lunch break. He clicks a link on a web page to start the game. A menu is shown, and Martin selects the singleplayer option. Having played Tetris before, Martin is familiar with the gameplay. When Martin starts the game he instinctively reaches for the cursor keys, and finds that they control the game as one would expect. However, he shortly notices special blocks that seem to affect gameplay, so he brings up the help menu. In the help menu, he reads about the different powerups, and then continues playing. During the game, he stacks blocks falling from the top of the screen on each other, and tries to minimize the height of the block pile. After five minutes, the game speed has increased to the point where it requires his full attention. After a few more minutes the speed is so high he has no choice but to desperately stack pieces wherever they fit. Soon, the pile reaches the top of the screen and the game ends. He then begins a new game, and after playing a couple of times, he shuts the game down and resumes his work.

- After work, Martin decides to try the multiplayer version of the game. He launches the game, and views a list of available multiplayer games. He finds a four-player game hosted by another user, "John", and joins the game. When two more users have joined, the game is full and John starts the game session. Martin and John co-operate well, but the other users have less experience with Tetris and leave holes in the brick pile. Thus, few lines are removed and the game is over quickly. They decide to try again, and this time the players have understood the need to construct rows together. The game session lasts for roughly five minutes, and when it is over Martin quits Multitris.

- John is a student who is studying from home. He decides to play Multitris, which he has been playing on several occasions. He starts the game and selects the option to host an Internet game session. He enters his name and the number of players, and then presses OK. While he is waiting for other players to join, he switches applications and resumes his studies. A short while later he switches back to Multitris, finds that the game is full, and starts the game session.

2.3. System environment and context

Multitris can be used for both multiplayer and singleplayer games. The system is ideally used in a network of computers, where the option to play multiplayer games is enabled. It is also suitable as a distraction on laptops, used for example while travelling. Possible situations are on a train, boat, aircraft or bus.

It is possible to play the game on either Windows XP, Mac OS X or Linux. The game will be distributed via Java Webstart (a standard way of distributing Java applications via the web). This solution automatically also provides a standalone, downloadable client that can be played offline.

We will assume that our users have at least a 1.0 Ghz Pentium 4 processor or equivalent, and a graphics chipset that supports the Open Graphics Library, OpenGL (4). OpenGL is a cross-platform interface for hardware accelerated graphics. We also assume that they have JRE 5 or higher installed. For exact specifications, see "8.1. Minimal system environment specifications" and "8.2. Optimal system environment specifications".

Starting a singleplayer session is a quick process which will take no more than a minute. Multiplayer sessions, on the other hand, depend on other users and so might take longer to start. Typically, the user has a short period of time (perhaps 30 minutes) for playing games.

2.4. The scope of the system

Multitris is a multiplayer Tetris-like game. Below is a table that outlines the features of the system:

Table 1.

Topic	**In**	**Out**
Singleplayer support	X	
Multiplayer support over a network	X	
Hardware-accelerated graphics	X	
Web-based client (Java Webstart)	X	
Central server	X	
Standalone client	X	
Savable game states		X
Multiplatform (Win, Mac, Linux)	X	
Support for phones and PDAs		X
Multilingual user interface		X
Interactive tutorial		X
Tetris-like gameplay	X	
Local chat functions	X	
Global chat functions		X

2.5. Main factors

First, the game needs to provide an interesting distraction for the users. By using well-known gameplay from Tetris we have a good base, that is decidedly popular. The popularity is confirmed by Wikipedia, that says that a variant of Tetris “[...] is available for nearly every video game console and computer operating system” (1). Our most important contribution to this kind of gameplay is the added multiplayer support.

Second, we want to make sure that a multiplayer game session can be started quickly. This means that we need to provide an efficient way of finding people to play with, and that multiplayer games are simple to set up. This will require a server that manages current games and keeps track of the different players. A list of open games will be available through the user's client.

Further, we need to ensure that the user's system is capable of running the game. The system needs to fulfill the requirements outlined in the “System environment and context” section above.

It is also important that the game has an interface that appeals to the user. This is a major problem for all games and will require both time and effort.

2.6. Technologies and risks

We intend to use the Java platform, with the Lightweight Java Game Library (LWJGL, 5) wrapper. This is a tool for developing games in Java and provides access to the OpenGL library. One possible risk with this is the group's lack of experience with OpenGL graphics.

Another possible risk is network communications when hosting games. The host computer will be required to have a specific port open, which is a a potential security threat. Many users will be behind a firewall and not know how to open ports, and these users might not be able to host games. Also, any Java Webstart

program that requests full permission (access to the client's system and network) has to be signed with a certificate.

The game can be launched via any web browser with a Java Webstart plugin. As for operating systems, cross-platform compability is always an issue. However, OpenGL and LWJGL are confirmed to work on three major operating systems, namely Windows XP, Mac OS X and Linux. We will therefore be able to make the game run on these platforms.

3. Glossary

Table 2.

Term	Explanation
Java	A platform-independent programming language.
JRE 1.5	Java Runtime Environment 1.5, which provides a virtual machine for running Java.
LWJGL	Lightweight Java Game Library, a library for games creation in Java.
Java Webstart	A standardized way to launch Java applications from the Internet.
OpenGL	Open Graphics Library, a standard for hardware accelerated graphics.
Central server	The central server is a server that keeps a list of game servers waiting for players to join. There is only one central server.
Game server	The game server is the program that handles the game lobby and game sessions. There is one game server for each game being played.
Game	A game is the time period from the time that the server is launched until it is shut down. This includes the lobby in which players can talk and wait for other players, and any game sessions.
Game session	A game session is the time period when the user actually plays the game.
Game lobby	The game lobby is a process that allows players to communicate and wait for others during games, between game sessions. In a sense, it is a virtual meeting room.
Game board	A playing field with a grid of a given size that can hold bricks.
Brick	A fundamental game element which can connect to other bricks.
Piece	Four connected bricks that can be controlled by a player. Pieces can also collide with each other and bricks.
Powerup	A special brick that can be activated by a player to alter the game board or gameplay.

4. User requirements definition

4.1. Functional requirements

4.1.1. Game properties

#1 Game pieces
The user shall be able to control several different kinds of pieces.

#2 Boundaries
There shall be boundaries limiting piece movement.

#3 Falling pieces
The pieces shall move downwards with a given speed.

#4 Score
The system shall keep and display a total score.

#5 Get help
The user shall be able to access game instructions.

#6 Menu system
The user shall be able to navigate a menu system to access the different game functions.

4.1.2. Game sessions

#7 Start singleplayer game session
The user shall be able to start a singleplayer game session.

#8 Host multiplayer game
The user shall be able to host a network game.

#9 Host private game
The user shall be able to host a private network game.

#10 Join multiplayer game
The user shall be able to join a network game.

#11 Chat with other users
The user shall be able to chat with other users when in a game.

4.1.3. Piece movement

#12 Move piece vertically
The user shall be able to increase vertical piece movement downwards.

#13 Move piece horizontally
The user shall be able to control the pieces' horizontal movement.

#14 Rotate piece
The user shall be able to rotate the piece that he controls.

#15 Collide with other player-controlled bricks
In multiplayer sessions, different players' pieces shall be able to collide.

4.1.4. Brick placement

#16 Piece control
The user shall control one piece at a time.

#17 Fixate piece
A piece that cannot move downwards shall be fixated to the board.

#18 Remove row
When a row is completed, it shall be removed from the game board.

#19 Finishing a game session
When a piece gets fixated above the top boundary, the game session shall end.

4.1.5. Powerups

#20 Obtain powerup
The user shall be able to obtain powerups.

#21 Create powerup
The system shall introduce powerup bricks when rows are removed.

#22 Use powerup
Players shall be able to activate obtained powerups.

4.2. Non-functional requirements

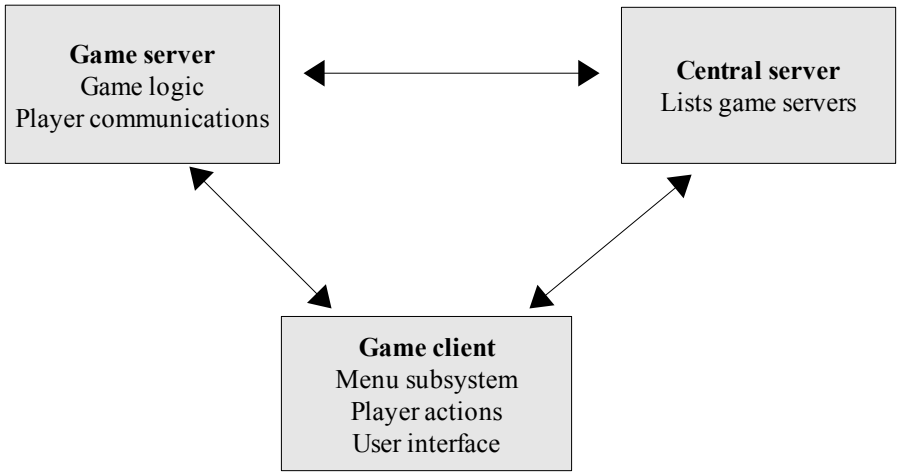
#23 Documentation
The game shall have documentation describing the game objective, how to start a singleplayer game session, join a session, host a session and navigate pieces. Further, it shall explain the different powerups.

#24 Graphics acceleration
The user shall be able to use graphics acceleration hardware for playing the game.

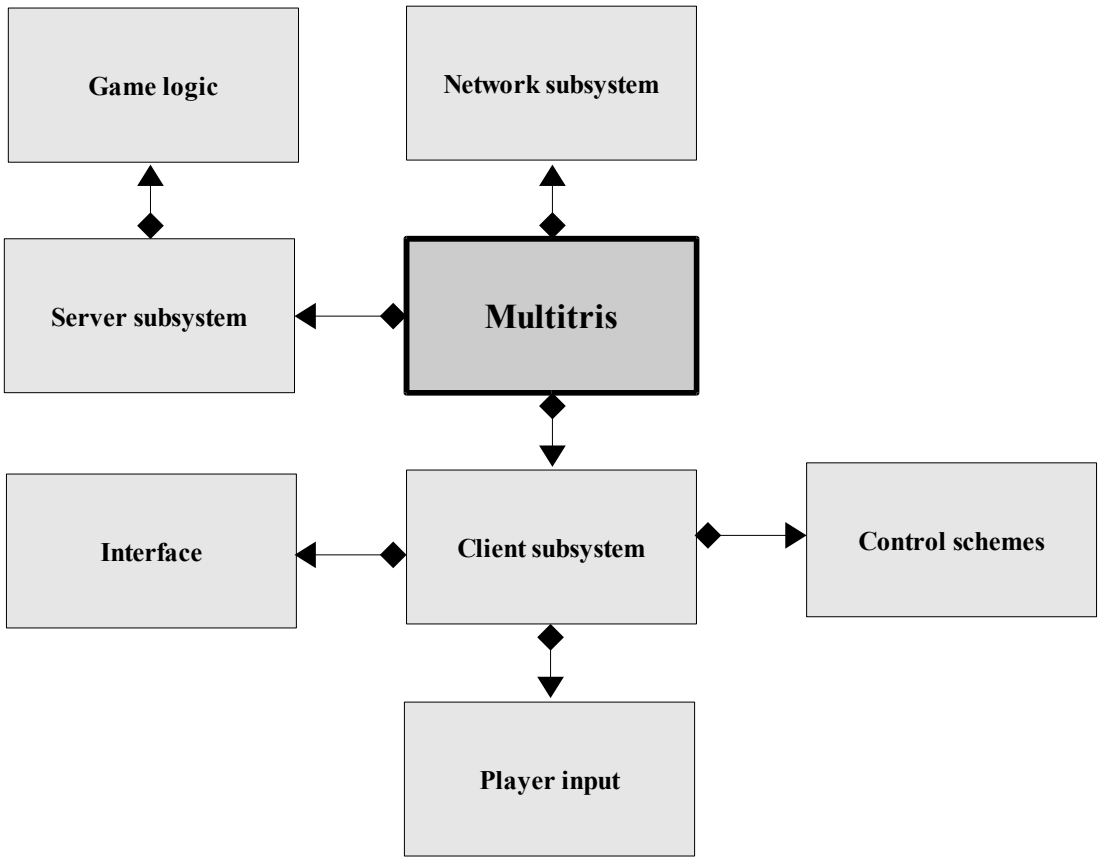
#25 Inputs
The system shall support two input devices – a keyboard and the X-Box 360 game controller.

5. System architecture

5.1. System architecture overview



5.2. Subsystems overview



6. System requirements specification

All requirements below are numbered corresponding to the user requirements as outlined above.

6.1. Functional requirements

The functional requirements described below are sorted into groups, which are referenced from the different requirements.

6.1.1. Game properties

#1 Game pieces
Requirement: There shall be seven different pieces which can be placed. Each piece consists of four parts called bricks.
Rationale: The pieces vary in shape, thereby adding an element of strategy to the game.
Referenced requirements: None.

#2 Boundaries
Requirement: There shall be boundaries that restricts piece movement. Specifically, there is the top, lower and side boundaries. No piece may pass any boundary other than the top one.
Rationale: The game board is defined by boundaries.
Test: Starting a game session and verifying that pieces can not be moved outside boundaries.
Referenced requirements: The *Piece movement* group, *Game pieces.*

#3 Falling pieces
Requirement: Any piece controlled by a player shall move downward with a given speed, that is increased when a piece gets fixated.
Rationale: The user should have limited time to decide how to place the piece.
Test: Starting a game session and verifying that pieces fall when controlled by a player.
Referenced requirements: The *Piece movement* group, *Game pieces*, *Fixate piece*.

#4 Score
Requirement: The system shall keep and display a total score that is altered when pieces are fixated or pieces collide.
Rationale: The user wants to obtain as high a score as possible.
Test: Starting a game session and verifying that the score is altered upon the mentioned actions.
Referenced requirements: *Fixate pieces*, *Collide with other player-controlled pieces*, *Game pieces.*

#5 Get help
Requirement: The system shall provide a manual for the user. The manual shall describe how to start, stop and play a game session, as well as how to navigate pieces.
Rationale: The user may want to read a manual on how to play the game.
Test: Reading through the documentation and verifying that it includes a description of said topics.
Referenced requirements: *Game pieces.*

#6 Menu system
Requirement: The system shall allow the user to choose from one of the following options through a menu system: Singleplayer, Host game, Join game, Help, Exit game.
Rationale: To select different functions of the game.
Referenced requirements: The *Game sessions* group

6.1.2. Game sessions

#7 Start singleplayer game session
Requirement: The player shall be able to start a singleplayer game session.
Rationale: The user may want to play singleplayer games, or may not be able to access a network.
Test: Starting a singleplayer session.
Referenced requirements: None.

#8 Host multiplayer game
Requirement: The system shall allow a user to start a game that other users are able to join.
Rationale: Users may want to play a game together. Further, it is the key idea in this game.
Test: Starting a network game session and letting other users join it.
Referenced requirements: None.

#9 Host private game
Requirement: The user shall be able to host a private network game. This should be the same as hosting a standard network game, except that the game will not show up in the game list for other users. However, the central server should still keep track of the game to allow the desired users to join it by entering its name.
Rationale: Users may want to play with friends only.
Test: Starting a private network game and making sure it does not show up in the game list, then joining the game via the network.
Referenced requirements: None.

#10 Join multiplayer game
Requirement: The system shall allow users to join previously hosted games.
Rationale: The user may want to join an already hosted game.
Test: Letting another user start a network game session and joining it.
Referenced requirements: None.

#11 Chat with other users
Requirement: The user shall be able to chat with other users when in a game. At any time during a game, a user should be able to send a message that will be displayed to all participating users.
Rationale: The users may want to communicate with each other during games.
Test: Starting a network game and making sure that the user can send messages to other users.
Referenced requirements: None.

6.1.3. Piece movement

#12 Move piece vertically
Requirement: The user shall be able to increase the downward speed of the current piece by a constant factor.
Rationale: The user may want to increase downwards brick movement speed in order to get a new piece faster.
Test: Starting a game session and making sure the piece can be accelerated downwards.
Referenced requirements: *Falling pieces*, *Piece control*, *Game pieces*.

#13 Move piece horizontally
Requirement: The user shall be able to move the piece horizontally with constant steps.
Rationale: The user may want to move pieces so that rows can be completed, allowing for longer games.
Test: Starting a game session and making sure the piece can be moved horizontally.
Referenced requirements: None.

#14 Rotate piece
Requirement: The user shall be able to rotate the piece he controls, either clockwise or counter-clockwise.
Rationale: The user may want to rotate the pieces so that rows can be completed, allowing for longer games.
Test: Starting a game session and making sure the piece can be rotated.

Referenced requirements: None.

#15 Collide with other player-controlled bricks
Requirement: Two non-fixed bricks that collide shall either be removed or moved apart. If they are removed, each player shall gain a new piece and a fixed score shall be subtracted from the total score.
Rationale: There must be a behavior for pieces that collide.
Test: Starting a multiplayer game session with another player, and testing piece collision behaviour.
Referenced requirements: *Piece control*, the *Piece movement* group, *Fixate piece*, *Score*, *Game pieces*.

6.1.4. Brick placement

#16 Piece control
Requirement: When game session starts the user shall receive a brick to control. After a brick has been fixated the user shall receive a new brick.
Rationale: Rows are formed by controlling single bricks and placing them where they fit.
Test: Starting the game session and making sure one gets new pieces in the beginning and each time a brick is fixated.
Referenced requirements: *Fixate piece*, the *Piece movement* group, *Game pieces*.

#17 Fixate piece
Requirement: When the vertical movement is obstructed by a fixated piece or the lower boundary, the piece shall stop moving and be taken out of player control.
Rationale: Bricks need to be fixated to form complete rows.
Test: Starting a game session and making sure pieces get fixated when they can no longer move downwards.
Referenced requirements: *Falling pieces*, the *Piece movement* group, *Boundaries*, *Game pieces*.

#18 Remove row
Requirement: Complete brick rows reaching between the two side boundaries shall be removed and points shall be added to the total score.
Rationale: The player wants to form complete rows in order to prolong the game, allowing for higher score.
Test: Starting a game, completing a row, and making sure that the row is removed.
Referenced requirements: *Score*, *Boundaries*.

#19 Finishing a game session
Requirement: When a piece gets fixated above the top boundary, the game session shall be terminated and the total score presented to the user. In a multiplayer game session, all participating players shall be presented with the total score.
Rationale: The game must come to an end.
Test: Starting a game session, and stacking pieces until one gets fixated above the top boundary.
Referenced requirements: *Fixate piece*, *Boundaries*, the *Game sessions* group, *Game pieces*.

6.1.5. Powerups

#20 Obtain powerup
Requirement: The user shall be able to obtain powerups contained in completed rows.
Rationale: Powerups add variety and an extra element of strategy to the game.
Test: Starting a game session, and making sure one can obtain powerups contained in completed rows.
Referenced requirements: *Remove row*.

#21 Create powerup
Requirement: When a row is removed, a random brick on the game board shall be replaced with a powerup brick. If there are no bricks on the game board, nothing happens.
Rationale: Powerups add variety and an extra element of strategy to the game.
Test: Starting a game session, and making sure powerups are introduced to the game board when a row is

removed.
Referenced requirements: *Remove row*.

#22 Use powerup
Requirement: At any time during a game session, the player shall be able to activate a powerup that he has obtained. This will affect gameplay or the game board. A list of powerups with corresponding effects is given in Table 3.
Rationale: The Powerups add variety and an extra element of strategy to the game.
Referenced requirements: *Obtain powerup*.

Table 3.

Powerup	**Effect**
Clear row	Removes all bricks in the bottom row from the game board.
Clear partial row	Removes all bricks in a given part of the bottom row from the game board.
Clear field	Removes all bricks on the game board.
Mirror flip	Allows the player to flip pieces horizontally.
Gravity	Pulls all bricks on the board as far down as possible, which removes any gaps.
No gravity	Removes gravity for the current piece, removing its falling speed.
Foresight	Shows upcoming pieces to the player.

6.2. Use Cases (UCs)

6.2.1. UC1: Use menu system

Primary Actor: User.
Stakeholders:
- User: Wants the system to activate a game function.
Precondition: The application is started.
Minimal Guarantee: None.
Success Guarantee (postconditions): The requested function is activated.

Basic Flow:
1. System presents user with a choice between playing a singleplayer game, joining a game, hosting a game, getting help, or exiting the game.
2. User makes his choice.
3. System activates the requested function.

Extensions:
3a. The user has selected to play a singleplayer game.
    3a.1. System starts a game session (user plays game).
3b. The user has selected to join a game.
    3b.1. User joins game.
3c. The user has selected to host a game.
    3c.1. User hosts game.
3d. The user has selected to get help.
    3d.1. System presents user with the documentation for the game.
3e. The user exits the game.
    3e.1. Shut down the application.
    3e.2. Terminate process.

Special requirements:
For extensions 3b and 3c, access to a computer network is required.

Technology and data variation:
None.

Frequency of occurrence:
At least once each time application is started.

Trigger:
User accesses menu system.

6.2.2. UC2: Host game

Primary Actor: User.
Stakeholders:
- User: Wants the system to host and start a network game session.
Precondition: A network connection is available.
Minimal Guarantee: System remains running.
Success Guarantee (postconditions): System hosts and starts a network game session.

Basic Flow:
1. User inputs nickname and number of players.
2. User requests to open game for new players.
3. System opens game for new players.
4. User waits for additional players to join the game.
5. User requests to start game session.
6. System closes game for new players.
7. System starts game session (user plays game).

Extensions:
1a. User input is invalid.
    1a.1. System notifies user.
    1a.2. System requests new input.
4a. User requests to close game.
    4a.1. System closes game.
    4a.2. System routes user to menu system.
    4a.3. Terminate process.

Special requirements:
None.

Technology and data variation:
None.

Frequency of occurrence:
Unspecified.

Trigger:
User requests to host game.

6.2.3. UC3: Join game

Primary Actor: User.
Stakeholders:
- User: Wants the system to join a network game and game session.
Precondition: A network connection is available.
Minimal Guarantee: System remains running.
Success Guarantee (postconditions): System joins a network game and game session.

Basic Flow:
1. User inputs nickname.
2. System presents user with a list of games hosted by other players.
3. User requests to join a game.
4. User waits for a game session to start.
5. System starts a game session (user plays game).

Extensions:
1a. User input is invalid.
    1a.1. System notifies user.
    1a.2. System requests new input.
1-3a. User requests to abort joining game.
    1-3a.1. System routes user to menu system.
    1-3a.2. Terminate process.
2a. There is no connection to the master server list.
    2a.1. System notifies user.
3a. System fails to join game.
    3a.1. System notifies user.
    Continue at step 2.

Special requirements:
None.

Technology and data variation:
None.

Frequency of occurrence:
Unspecified.

Trigger:
User requests to join game.

6.2.4. UC4: Play game

Primary Actor: A user playing the game.
Stakeholders:
- User: Wants to be presented with a total score.
Precondition: A game session is active.
Minimal Guarantee: System ends game session and presents the user with a total score.
Success Guarantee (postconditions): System ends game session and presents the user with a total score.

Basic Flow:
1. User receives a piece.
2. User navigates the piece.
3. System fixates the piece.
Repeat from 1 unless a brick is fixated above the top boundary.
4. System ends session.
5. The user is presented with a total score.

Extensions:
1-3a. At any time, user requests to exit game.
    1-3a.1. Continue at step 4.
2a. User uses a powerup.
    2a.1. System applies a powerup effect.
2b. User's piece collides with another player's piece.
    2b.1a. System removes the pieces.
        2b.1a.1. System subtracts points from the user's total score.
    2b.1b. User moves piece away.
        2b.1b.1. Nothing happens.
3a. The piece does not complete a row.
    3a.1. System calculates a score bonus.
3b. The piece completes a row.
    3b.1. System removes the row.
    3b.2. System adds any powerup bricks in the row to the player's powerup list.
    3b.3. System replaces a random brick on the game board with a powerup.
    3b.4. System calculates a score bonus.

Special requirements:
None.

Technology and data variation:
None.

Frequency of occurrence:
Once per game session.

Trigger:
System starts game session.

6.3. Non-functional requirements

#23 Documentation
Requirement: The game shall have documentation describing the game objective, how to start a singleplayer game session, join a session, host a session and navigate pieces. Further, it shall explain the different powerups.
Test: Going through the documentation, and verifying that it contains said items.

#24 Graphics acceleration
Requirement: We are limited to using the Open Graphics Library via the Lightweight Java Game Library wrapper (see the "2.6. Technologies and risks"above) for developing the game.
Test: Not relevant.

#25 Inputs
Requirement: The user shall support two input devices – a keyboard and the X-Box 360 game controller.
Test: Playing the game with said input devices.

7. System evolution

There are certain fundamental assumptions on which the system is based. These include:
- The operating system on which the system is run supports Java.
- For the clients, support for OpenGL via LWJGL is also assumed.
- A Java Runtime Environment of version 1.5 or higher is installed.
- The TCP/IP protocol is used for network communication.
- For users wishing to host games, the proper network port is open.

As the system environment evolves, these are some of the changes to the system we may have to make:
- New operating systems may not support OpenGL, in which case a different rendering engine may have to be used.
- New hardware may not support OpenGL, in which case a different rendering engine may have to be used.

As the user's needs change, these are some changes to the system we may hae to make:
- Users may want to communicate globally, in which case a chat subsystem may have to be added to the central server. Initially, we will only allow players to communicate within games.
- Users may not want to launch a separate game, in which case conversion to a Java Applet may be required. Java Applets are played from within the web browser.

8. Appendices

8.1. Minimal system environment specifications

Supported operating systems: Windows XP, Mac OS X, Linux.
1.0+ GHz Intel Pentium processor or equivalent
512 MB of RAM memory
NVIDIA GeForce 4MX or equivalent
50 MB free hard drive space
JRE installed, version 1.5 or higher

8.2. Optimal system environment specifications

Supported operating systems: Windows XP, Mac OS X, Linux.
2.0+ GHz Intel Pentium processor or equivalent
1 GB of RAM memory
NVIDIA GeForce FX 7600 or equivalent
50 MB free hard drive space
JRE installed, version 1.5 or higher

8.3. References

(1) Tetris – http://en.wikipedia.org/wiki/Tetris
(2) Tetrinet – http://en.wikipedia.org/wiki/TetriNET
(3) The Java Runtime Environment, JRE – http://java.sun.com/
(4) OpenGL – http://www.opengl.org/
(5) Lightweight Java Game Library – http://www.lwjgl.org/ (a Java library for making games)