Requirements and Analysis Document for the Romijam project (RAD)

Version: 1.0

M. Phoohad S. d'Aubigné S. Nilsson F. Malmek

May 21, 2012

This version overrides all previous versions.

Contents

1 Introduction

This section gives a brief overview of the project.

1.1 Purpose of application

The purpose of this project is to create a fun and thrilling game that is easy to learn.

The game is supposed to provide a minor challenge and the possibility to become a skillful player.

1.2 General characteristics of application

The application will be a 2D desktop application with a GUI adapted for the Windows/Mac/Linux platforms without any network requirements.

The game will provide the possibility to play in either single- or multiplayer mode on the desktop using fixed keys for controlling each player-character's movements.

There will be several game-levels with different challenges, providing an interesting game experience. The game is lost when the last player-character in the level dies, and the game will then restart from the beginning of the current level.

The level is completed when the player reach the end of the level, and the game is completed if you complete all the levels.

1.3 Scope of application

The application is divided in parts. These parts are the main menu, select level menu, pause menu and normal gameplay. Hence application provides the possibility to pause the game and resume later in the same instance.

It is also runnable on several different operative systems. All it requires to run is support for Java's virtual machine.

1.4 Objectives and success criteria of the project

There should be at least three different game-levels with different obstacles and challenges to the game. It should be possible to navigate a player-character through each level by using a keyboard..

1.5 Definitions, acronyms and abbreviations

GUI: Graphical User Interface, in this context meaning both the navigation menu in the beginning and the graphical representation of the game while playing.

Player-character: A virtual representation of the players progress in the current game-level.

Game-level: The visual obstacles to be completed by the player.

Game over: If a game is over, you can not continue playing until you restart the game.

2 Requirements

In this section we specify all requirements

2.1 Functional requirements

The player should be able to perform these actions:

Start the game. Load progress and start where the player left off. Select available game-levels (i.e. already completed levels)

Control the movement of the character within the current game-level.

Die , which causes you to fail the game-level (i.e. game over).

Restart the current level.

2.2 Non-functional requirements

All the non-functional requirements are listed below under their appropriate subheading.

2.2.1 Usability

The usability is of high priority. The application must be usable by children, hence it must be easy to learn and use. Therefore the game will provide a well describing tutorial on how to play, which will prevent time-consuming learning processes.

2.2.2 Reliability

N/A

2.2.3 Performance

The game should be able to play in 60 FPS (frames per second) on a modern computer.

There should not be any freezing or crashing from normal use.

2.2.4 Supportability

There should be enough support in the implementation to allow for multi-player games (i.e. two-player functionality)—at least on the same screen.

The implementation should cater for the possibility of being used in other platforms (e.g. Web, Mobile). The effort needed to convert the application to each of these platforms should not be too big.

2.2.5 Implementation

The game will use separate frameworks for graphics and sound, so that the graphics run smoothly on any machine—utilizing the computer's graphics hardware.

2.2.6 Packaging and installation

The game will be runnable from it's project folder.

2.2.7 Legal

The game will not be using a trademarked name, and it will not be too similar to other copyrighted products. The game should be completely open-source and free for anyone to download and play. There might be some copyrighted graphic and sound material included in the game.

2.3 Application models

2.3.1 Use case model

UML and a list of UC names (text for all in appendix)

2.3.2 Use cases priority

High:

- Move
  - Go Right
  - Go Left
- Jump
  - Jump Right
  - Jump Left

Medium:

- Exit Game
- Crouch

Low:

- Select Level
- Start Level
- Lift Block
- Pause

2.3.3 Domain model

UML (see Appendix C)

2.3.4 User interface

One will navigate through the menu interface by using the keyboard arrow-keys. There will be three different options on the first view complemented with a picture.

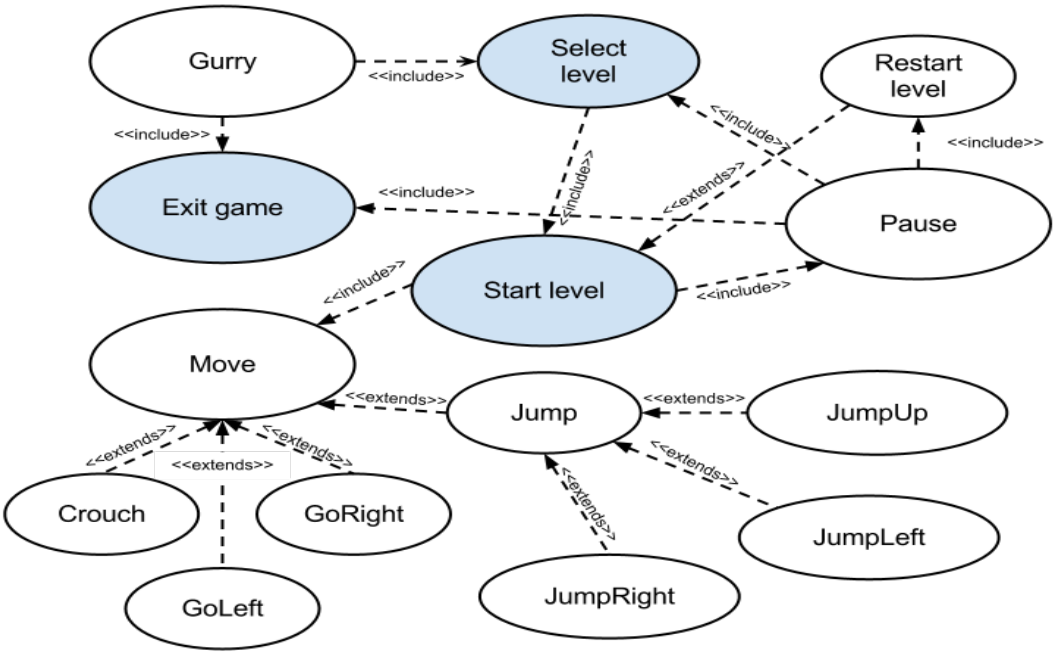
The game itself is based on a background with blocks that the player character plays through.

From the running stage, the player can pause and receive a Pause-view; disabling the running game and providing three options: Continue, Select level and Main Menu which will get him back to the start-view.

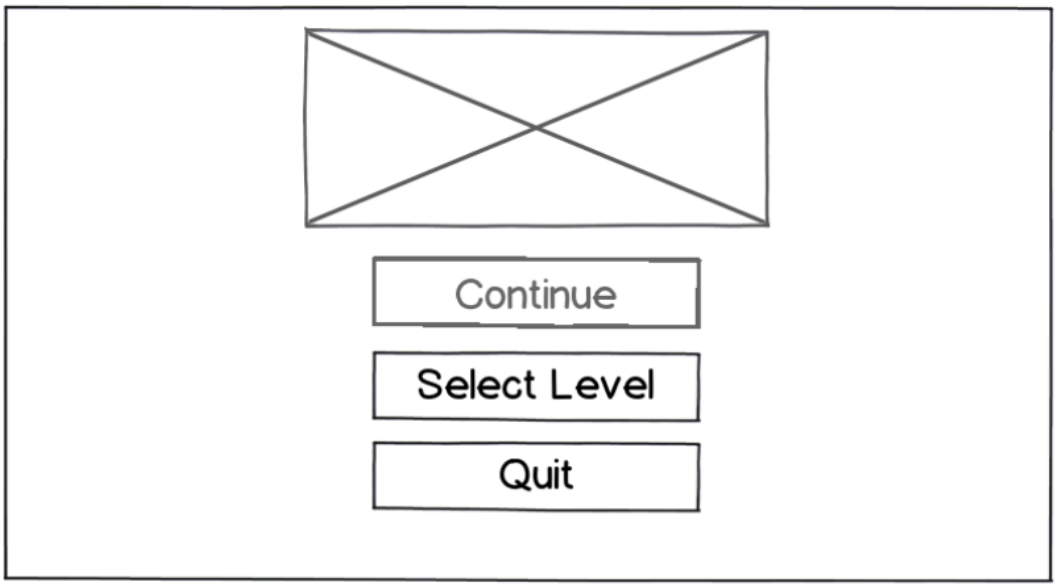
3 References

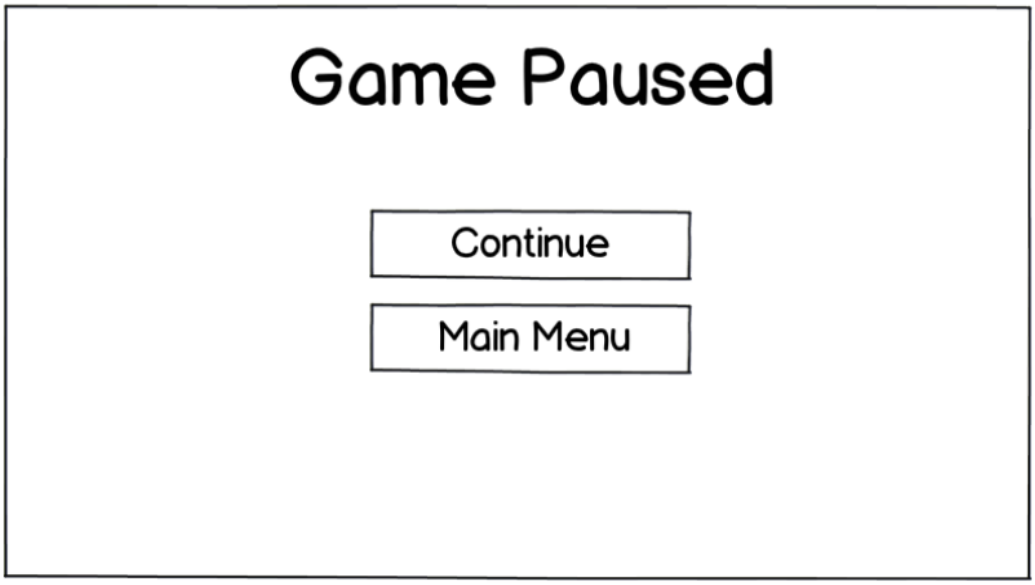
N/A

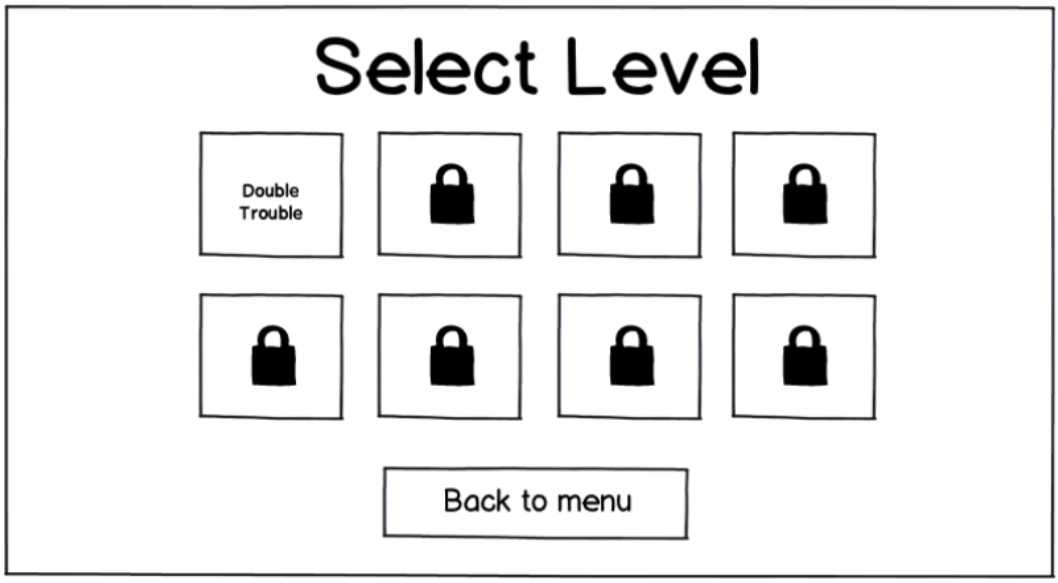
A Appendix: Use Case graph

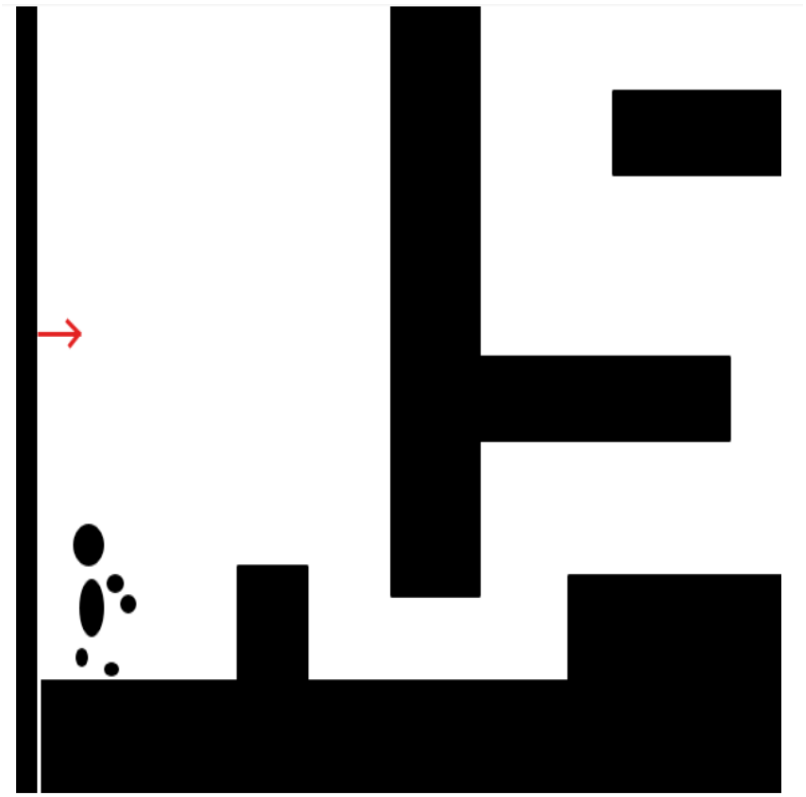


B Appendix: GUI

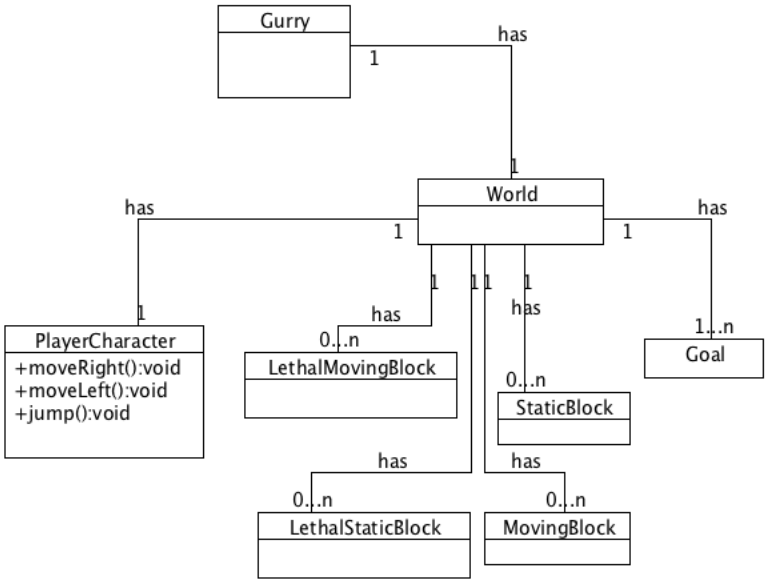





Select Level
Double Trouble
Back to menu



C Appendix: Domain Model



D Appendix: Use Cases

D.1 UC: Select Level

Short description: The user chooses which game-level the user wants to play
Priority: Low
Includes or extends: includes StartGame
Participating actors: The Actual Player (AP).
Normal flow of events

Actor	System
AP clicks a level image	
	The selected level is marked with an outline
	The “Start Game”button is activated
AP clicks “Start Game” button (see UC: StartGame)	

Alternative flow: No alternative
Exceptional flow: No exception

D.2 UC: Start Level

Short description: How the player starts a new game

Priority: Low
Extends or Includes: Includes Move, Pause
Participating actors: Actual player (AP).

Actor	System
AP clicks on “start level” button	
	Game loads the selected level.
	Application displays the level on screen and loops the level entities.

Exceptional flow: No exception

D.3 UC: Pause

Short description: Stops the game and shows a menu of options.
Priority: Low
Extends or includes: includes ExitGame, RestartLevel, SelectLevel
Participating actors: Actual player(AC). Game started, see UC: StartLevel.
Normal flow of events

Actor	System
AP presses the escape key	
	Game is freezed
	Application displays a paused view for player options (i.e Exit Game,Restart Level & Select Level. See their respective UCs).

D.4 UC: Move

Short description: How a user moves his player-character.
Priority: High
Extends or includes: Includes Crouch, GoRight, GoLeft, Jump
Participating actors: Actual player (AP). Game not paused, see UC: Pause
Normal flow of events

AP presses a keyboard key that moves the character	
	Player-character's position is changed.

Alternate flow: Touched lethal object
If new position is colliding with a lethal object, player-character dies.
Maybe add an alternate flow for getting crushed between two objects? Is this UC: Move?
Alternate flow: Colliding with an object
If new position is colliding with an object, the player-character does not change position.
Exceptional flow: No exception

D.5 UC: Go Left

Short description: The player-character moves left.
Priority: High
Extends or includes: Extends move.
Participating actors: Actual Player (AP).
Normal flow of events

AP presses the left arrow key	
	The player-character's horizontal position is moved left (x-value is decreased).

Alternate flow: No additional alternate flow, see UC: Move
Exceptional flow: No exception

D.6 UC: Go Right

Short description: The player-character moves right.
Priority: High
Extends or includes: Extends move.
Participating actors: Actual Player (AP).
Normal flow of events

AP presses the right arrow key	
	The player-character's horizontal position is moved right (x-value is increased).

Alternate flow: No additional alternate flow, see UC: Move
Exceptional flow: No exception

D.7 UC: Crouch

Short description: Player-character crouches down (vertically).
Priority: Medium
Extends or includes: Extends move.
Participating actors: Actual Player (AP). Game not paused, see UC: Pause
Normal flow of events

AP presses the left shift key or the down arrow key	
	The player-character's vertical height is lowered.

Alternate flows: No alternate flow
Exceptional flow: No exception

D.8 UC: Jump

Short description: The player-character jumps up.
Priority: High
Extends or includes: Extends move.
Participating actors: Actual Player (AP).
Normal flow of events

AP presses the spacebar	
	The player-character's vertical position is moved upwards(y-value is increased).

Alternate flow: No additional alternate flow, see UC: Move
Exceptional flow: No exception

D.9 UC: Jump Left

Short description: The player-character jumps, and moves left.
Priority: High
Extends or includes: Extends Jump.
Participating actors: Actual Player (AP).
Normal flow of events

AP is pressing the left arrow key while jumping	
	The player-character's horizontal position is moved left(x-value is decreased), and the vertical position is moved upwards(y-value is increased).

Alternate flow: No additional alternate flow, see UC: Jump
Exceptional flow: No exception

D.10 UC: Jump Right

Short description: The player-character jumps, and moves right.
Priority: High
Extends or includes: Extends Jump.
Participating actors: Actual Player (AP).
Normal flow of events

AP is pressing the right arrow key while jumping	
	The player-character's horizontal position is moved right(x-value is increased), and the vertical position is moved upwards(y-value is increased).

Alternate flow: No additional alternate flow, see UC: Jump
Exceptional flow: No exception

D.11 UC: Exit Game

Short description: How a player exit the game

Priority: Medium
Extends or Includes: N/A
Participating actors: Actual player (AP).

Actor	System
AP clicks on “exit game”	
	Game saves user progress and shuts down.

Exceptional flow: No exception