Requirements and Analysis Document for Blockster

Contents

Version: 2.1

Date 2014-05-25

Author Eric Bjuhr, Oskar Jönefors, Emilia Nilsson, Joel Tegman

This version overrides all previous versions.

1 Introduction

This section gives a brief overview of the project.

1.1 Purpose of application

The project aims to create a computer game based of an already existing game known as *Block Dude*, but with somewhat different rules and more details. *Block Dude* was released on the calculator *Texas Instruments TI-83+*. The game genre is a puzzle and platform game.

1.2 General characteristics of application

The application will be a desktop, stand-alone (non-networked) game that will be keyboard-controlled. The user must apply logic and manipulate blocks in the environment to complete each level. The game will let you control two different characters, switching between them at will. Grabbing blocks and moving them around will be the way to finish the game, as there will be obstacles that will require the player to do so. There will also be cases where the player is stuck and can not clear the game because of certain wrong choices that were made. When that happens, the player can reset the level and start over again. Because of this, basic concept of the game will be very similar to the original *Block Dude* game, but with an enhanced and colorized GUI and a different ruleset. Reaching the goal will end the level.

1.3 Scope of application

The application will be single player only. The user can switch between two characters who each need to reach their respective goals. The game level can be restarted at will.

1.4 Objectives and success criteria of the project

The player should be able to move around, climb, lift, grab, place, push and pull blocks. It should be possible to complete and restart a level.

1.5 Definitions, acronyms and abbreviations

Level - A map/stage/course that the player is supposed to finish.
GUI - Graphical user interface.
Blocks - The obstacles in the level that will be of different types, some static, and some moveable.
Java - Platform-independent programming language.
UML - Unified Modeling Language, diagram showing relationships between classes and packages and their properties.
UC - Use Case - actions that the user can perform.
Visual framework - a graphical library helping and making it possible to draw the game on the screen.

2 Requirements

2.1 Functional requirements

Player will be able to:

1. Start a new game.
2. Maneuver the characters according to game logic. (right / left, climb blocks, no jumping)
3. Push and pull certain blocks along the ground.
4. Lift blocks and move around with them.
5. Place lifted blocks.
6. Shift which character to control.

2.2 Non-functional requirements

2.2.1 Usability

The game will be very easy to use. The only actions a player can make is to move sideways, climb a block, lift and place a block and push and pull a block. There will be a *How to play*-section in the README-file (that will be included when installing the game, see *Packaging and installation*) that will explain the controls and game rules.

2.2.2 Reliability

Normal gameplay should be possible without problems. However, if the user has malicious intent, problems may occur.

2.2.3 Performance

Blockster will be a simple 2D game and will not require any powerful hardware to run smoothly.

2.2.4 Supportability

There will be unit tests covering all of the classes in the core package to make sure it functions properly.
The visual framework will be exchangeable.

2.2.5 Implementation

The Blockster game will be written in *Java*.

2.2.6 Packaging and installation

The installation process will consist of cloning the codebase from a git repository and building it with maven. After the application has been built, it can be executed with the maven exec plugin, or by running the generated jar file. Specific commands can be found in the README supplied with the game.

2.2.7 Legal

The name *Blockster* is already used for existing game titles. Whether it is trademarked or not will not be investigated. There are legal issues regarding rights to the *Block Dude* game and trademark but that will not be covered here.

2.3 Application models

2.3.1 Use case model

A complete list of use cases will be included in the appendix.

2.3.2 Use cases priority

1. Move
2. Grab
3. Push/Pull
4. Lift
5. Climb
6. Switch character
7. Restart stage

2.3.3 Domain model

See *Appendix*

2.3.4 User interface

The GUI view will be locked upon the currently active character, always keeping it in the center of the screen, only showing a small piece on the entire map at a time in a mini map. When pressing the *switch* key, the screen moves from the active character to the other.

2.4 References

Block Dude game (as a part of the puzzle pack): http://www.detachedsolutions.com/puzzpack/

APPENDIX

GUI

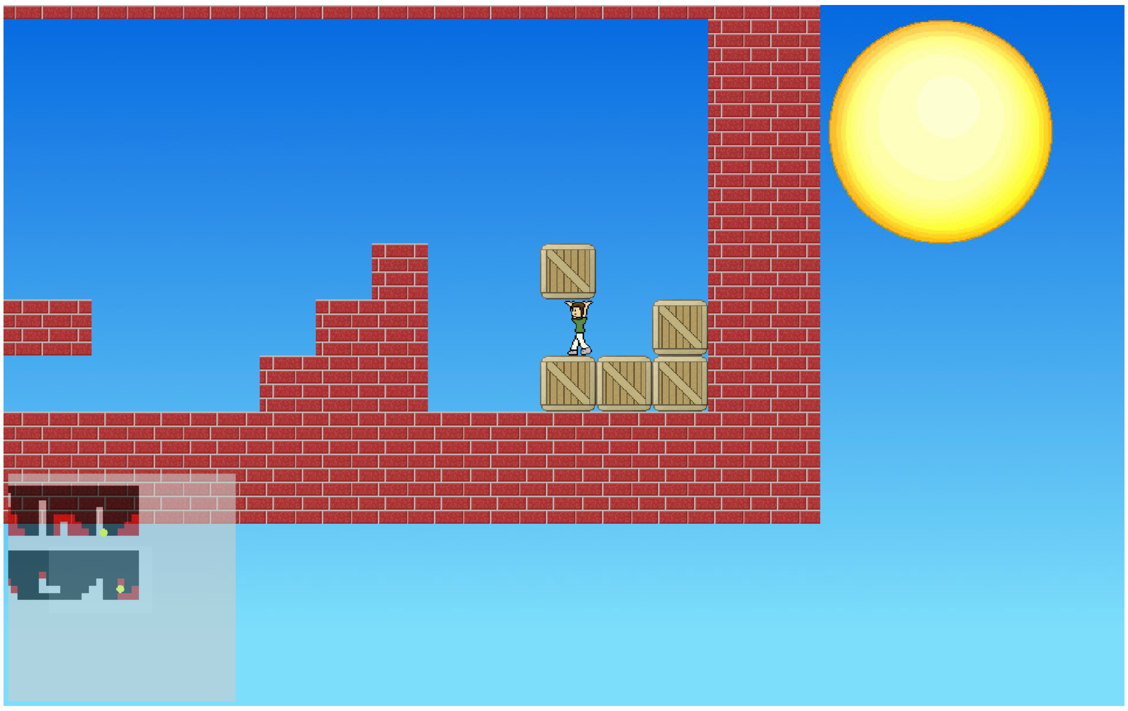

Figure 1: The figure shows the GUI, consisting of a game view and a minimap. The game view represents a player lifting up one of the blocks in the map using one of the playable characters.

Domain model

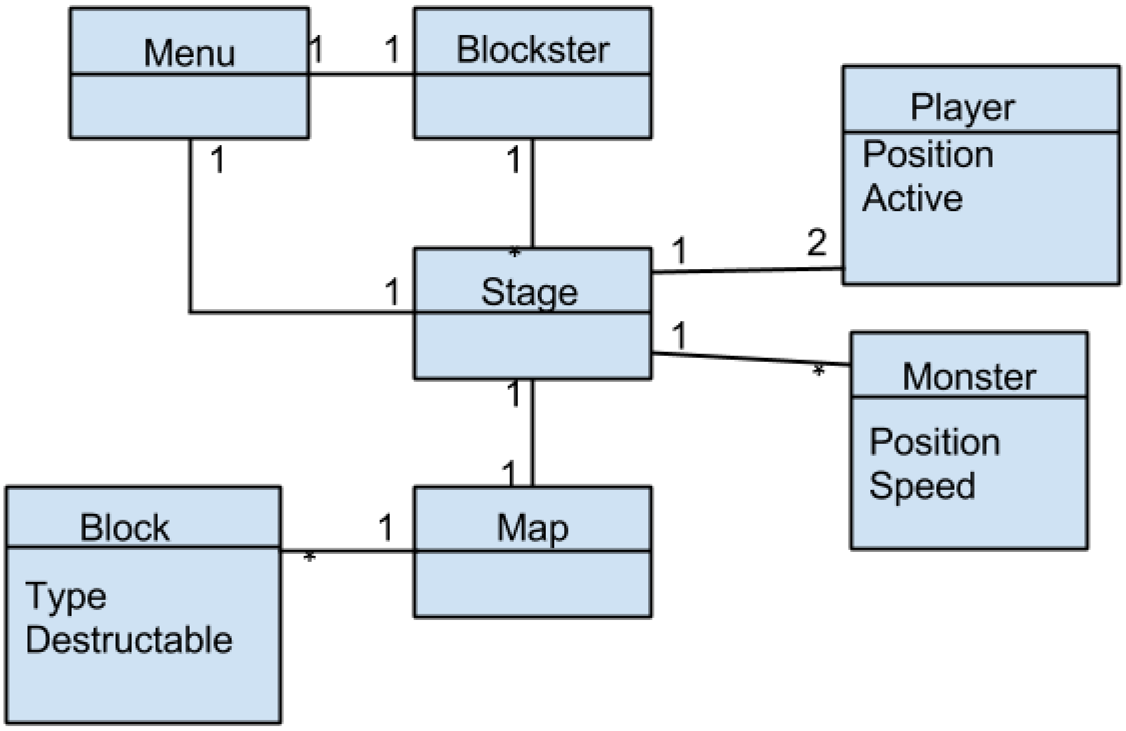

Figure 2: The figure shows the initial domain model and the concept of the game. Monsters, destructible and the game menu blocks were never implemented.

Use cases

Use Case: Move

Summary: This is how the player move the character or characters. A UC Switch may depend which character the player moves around.
Priority: High
Extends: -
Includes: -
Participators: The player.

Normal flow of events:
The player moves around freely from left to right.

	Actor	System
1	Clicks or holds either of the the side arrows on the keyboard.	
2		Moves the character in the direction of the clicked arrow key.

Alternate flows:
Flow 2.1 The character stands next to a block and tries to move in that direction.

	Actor	System
2.1.1		The character will have a collision with the block.
2.1.2		The character's position will be set to the previous one before the collision.

Flow 2.2 The character has no block to support it’s next steps.

	Actor	System
2.2.1		The character falls down.
2.2.2		The character lands on a block further down.

Flow 2.5 The character bumps into a monster. Not implemented.

	Actor	System
2.5.1		The character dies.
2		The stage restarts.

Flow 2.6 The character steps on top of a button. Not implemented.

	Actor	System
2.6.1		An event linked to the button occurs (for example: a hatch opens/closes).

Flow 2.7 The character steps on top of a trampoline. Not implemented.

	Actor	System
2.7.1		The character jumps up two blocks high.
2.7.2	Presses either left or right arrow key.	
2.7.3		See 2.1/2.3, 2.2 to see what will happen.

Exceptional flow:
A block in movement falls upon the character. Not implemented.

	Actor	System
2.8.1		The character dies a painful death.
2.8.2		The stage restarts.

User Case: Grab

Summary: This is how the player will grab onto blocks being able to push or pull them.
Priority: High.
Extends: -
Includes: -
Participators: The player, the block.

Normal flow of events:
The player stands next to a block and grabs it.

	Actor	System
1	Grabs the block with a *grab* key.	
2		Shows animation for the character grabbing the block.
3	Pushes the block by using either left or right arrow key.	
4		Changes the position for the block to the pushed area.
5		Changes the position of the character to the area where the pushed block last stood.
6		Shows animation for the character and block being pushed, one block in length.
7	Pulls the block by using either left or right arrow key.	
8		Changes the position of the character to the position of the direction of the pull.
9		Changes the position of

		the block to the position of the direction of the pull.
10		Shows animation for the character and block being pulled, one block in length.

Alternate flows:
Flow 3.1 The block that is supposed to be pushed has another block blocking it’s way.

	Actor	System
3.1.1		1. If it’s a different block from the one pushed, nothing will happen. 2. If it’s the same block like the one pushed, they will both be pushed.
3.1.2		Changes position of the block first in line one block forward.
3.1.3		Changes position of the next block in line one block forward.
3.1.4		Changes position of the character one block in length forward.

Flow 7.1 The character has no ground to support him behind.

	Actor	System
7.1.1		Nothing will happen.

Exceptional flow:
N/A

User Case: Lift

Summary: This is how the player will lift up blocks being able to move them around.
Priority: High.
Extends: -
Includes: -
Participators: The player, the block.

Normal flow of events:
The player stands next to a block and picks it up.

	Actor	System
1	Presses a *lift* key.	
2		The block is taken out of the map
3		Animation shows the character lifting the block up and having it over his head.
4	Presses the *lift* key again.	
5		The block is brought back to the map, placed with a position in front of the character.
6		Shows animation for the character putting down the block.

Alternate flows:
Flow 1.1 The block is a non-liftable block.

	Actor	System
1.1.1		Nothing will happen.

Flow 4.1 The character has a block in front of him already.

	Actor	System
4.1.1		The block is brought back to the map, placed with a position in front of the character and on top of the block. If there is two (or more) blocks on top of each other in front of the character then nothing will happen.

Exceptional flow:
N/A

User Case: Climb

Not implemented

Summary: This is how the player will climb blocks and other obstacles.
Priority: High.
Extends: -
Includes: -
Participators: The player, the application.

Normal flow of events:
The player stands next to a block and climbs it.

	Actor	System
1	The player presses the *Climb* button which is the up arrow.	
2		The character will move it’s position to on top of the block in question.

Alternate flows:
Flow 2.1 The character holds a block.

	Actor	System
2.1.1		The character climbs on top of the block.
2.1.2		The character carries the other block with him.

Flow 2.2 The block has another block on top of it.

	Actor	System
2.4.1		The character won’t move (it will be too high to reach).

Exceptional flow:
N/A

User Case: Quit stage

Not implemented

Summary: This is how the player will be able to quit the current stage in order to get to the game menu. UC New game or possibly UC Select stage preceded this UC.
Priority: Medium.
Extends: -
Includes: -
Participators: The player, the application.

Normal flow of events:
The player goes into a paus menu choosing to quit the stage.

	Actor	System
1	The player presses the menu button.	
2		Menu is displayed.
3	The player selects “Quit stage”	
4		Quits the stage.
5		Loads the game screen.

Alternate flows:
N/A
Exceptional flow:
N/A

User Case: Restart stage

Summary: This is how the player will restart the stage. Possibly because the player cannot get any further and is stuck.
Priority: Medium.
Extends: -
Includes: -
Participators: The player, the application.
Normal flow of events:
The player restarts the game.

	Actor	System
1	The player presses the *Restart stage* button that will be the *R* key.	
2		The stage restarts.

Alternate flows:
N/A
Exceptional flow:
N/A

User Case: Switch character

Summary: The player switches from one of the playable characters to the other.
Priority: Medium.
Extends:
Includes:
Participators: The player, the application.
Normal flow of events:
The player switches to the other character to control.

	Actor	System
1	The player presses the *Switch character* button.	
2		The stage camera glides in to show the other playable character.
3		The other character becomes controllable.

Alternate flows:
N/A
Exceptional flow:
N/A

User Case: Select stage

Not implemented

Summary: This is how the user selects a particular stage to play.
Priority: Medium.
Extends:
Includes:
Participators: The player, the application.
Normal flow of events:
The user chooses a standard stage.

	Actor	System
1	At the main menu, the user selects the *Stage* entry.	
2		A listing of stages is shown. Locked stages are clearly marked.
3	The user selects a stage and presses the *confirm* button.	
4		The stage starts.

Alternate flows: The player chooses a custom stage

	Actor	System
1	At the main menu, the player selects the *Stage* entry.	
2		A listing of stages is shown. Locked stages are clearly marked. Custom stages have their own category and are never locked.
3	The user selects a custom stage and presses the *confirm* button.	

4		The stage loads.

Exceptional flow:
N/A

User Case: Change volumes

Not implemented

Summary: The player changes the music or sound effects volume.
Priority: Low
Extends:
Includes:
Participators: The player, the application
Normal flow of events: The player changes the sound from the main menu.

	Actor	System
1	At the main menu, the player selects the *Settings* entry.	
2		A number of options are shown, two of which are volume sliders for music and sound effects.
3	The user adjusts the sliders.	
4	The user presses *confirm*	
5		The player is taken back to the main menu.

Alternate flows: The player changes the volume from within a stage.

	Actor	System
1	In the middle of a stage, the player presses the *menu button*.	
2		A menu is shown as a stage overlay. One of the options is *settings*.
3	The user selects the *settings* entry and presses the *confirm* button.	

4		A number of options are shown, two of which are volume sliders for music and sound effects.
3	The user adjusts the sliders.	
4	The user presses *confirm*	
5		The player is taken back to the stage menu, and can resume playing by pressing the menu button again.

Exceptional flow:
N/A

User Case: Change controls

Not implemented

Summary: The user remaps controls to his or her liking.
Priority: Low
Extends:
Includes:
Participators: The player, the application.
Normal flow of events:

	Actor	System
1	At the main menu, the player selects the *Settings* entry.	
2		A number of options are shown, one of which is *change controls*.
3	The player selects the *change controls* entry and presses the *confirm* button.	
4		The view shows a listing of all the mapped controls in the game.
5	The player selects a control she wants to remap, and presses the *confirm* button.	
6		A prompt is shown saying: *Press the key you would like to assign*.
7	The player presses a key.	
8		The key value of the control is changed to reflect the new key.
9	The player selects the *save settings* entry at the bottom of the screen.	

10		The player is taken back to the *settings* menu, and can go back to the main menu from there.

Alternate flows: The player changes the controls from within a stage.

	Actor	System
1	In the middle of a stage, the player presses the *menu* button.	
2		A menu is shown as a stage overlay. One of the entries is *Settings.*
3	The player selects the *Settings* entry.	
4		A number of options are shown, one of which is *change controls*.
5	The player selects the *change controls* entry and presses the *confirm* button.	
6		The view shows a listing of all the mapped controls in the game.
7	The player selects a control she wants to remap, and presses the *confirm* button.	
8		A prompt is shown saying: *Press the key you would like to assign*.
9	The player presses a key.	
10		The key value of the control is changed to

		reflect the new key.
11	The player selects the *save settings* entry at the bottom of the screen.	
12		The player is taken back to the *settings* menu, and can go back to the stage menu from there.

Exceptional flow:
N/A