<^> TRIFORCE <^>

Requirements Workflow

by Gabe Pike

CMPS 335

Spring 2012

Team 10

Other Team Member: Brandon Hinesley

Table of Contents

Section 1: Requirements workflow
1.1 Requirements Workflow....3
1.1.1 Initial Understanding and Glossary....3
1.1.2 Initial Business Model....8
1.1.3 Initial Requirements....19
1.1.4 Details Requirements....19
1.1.5 Iterated Use-Case Diagrams and Descriptions....19
1.2 Walkthrough....30
1.3 Revised Requirements After Walkthrough....31

Section 1.1: Requirements Workflow

1.1.1 Initial understanding of the project

General

Triforce is a remake of the classic SNES game, Tetris Attack. It is called Triforce in part because of the core game mechanic where one must align at least three matching blocks. In the game, a player controls a cursor that lets him swap two adjacent blocks horizontally. Combinations of 3 or more blocks in a row will break the blocks, causing the blocks above them to fall. The game mechanics allow the player to use their skills and reaction time to gain great rewards. The game mechanics are flexible and allow for several fun modes of play. Figure 1 shows a screenshot of gameplay in Tetris Attack.

Figure 1: This screenshot shows a game with two players. Each player controls the white cursor on their grid of blocks. A player presses a button two swap the blocks inside the cursor.

Menu

The user experience should begin with opening the game to the main menu. The menu will contain buttons for Play, Settings, Help, Credits/About, and Quit. The Settings button will open a new menu to configure settings such as themes, sounds, and difficulty. Help will tell the user how to play. When the user selects Play, they will be taken to another menu to select what mode they want to play. When they select the mode, they will be taken to the actual game. It will start after a timer of three seconds. The gameplay area will consist of a grid of blocks and a cursor for the player to control, as seen in Figure 1. The player uses the keyboard or mouse to move the cursor to an adjacent block horizontally or vertically. The player must break blocks by aligning three or more of the same color vertically or horizontally.

Controls/Input

The controls to Triforce are rather simple. The user only needs to concern himself with the following input (input that will use the same binding is put in the same bullet point):

- Navigate the menus; Move the cursor in four directions
- Select a menu item; Swap blocks
- Boost the speed of play
- Pause/Unpause; Exit

We want to support multiple methods of input. It will be possible to use a keyboard, mouse, or Xbox 360 controller to control everything in the game. This will be done by making generic bindings that are not specific to any kind of input device. Then, we will map each input method from each input device to the generic bindings.

Gameplay and Mechanics

Triforce has potential for several modes of play. Due to time constraints, we will not likely be able to implement them all. At a minimum, we want to implement Endurance Mode, where a player must survive as long as he can while blocks are pushed onto the play area at accelerating speeds. If time permits, we will also implement Versus Mode. In Versus Mode, two players build up large combos and chains, and drop them on each other. It could be played on the input device or over a network connection. Due to time constraints and experience, it is unlikely we will implement any sort of artifical intelligence.

The game play of Triforce looks rather simple to a player, but its inner workings are actually rather complex. The main game mechanics are combos and chains. The mechanics of combos are the easiest to explain. A combo is achieved by aligning at least three matching blocks together. However, you can combine sets of blocks by positioning things right before moving them. Below are some examples of possible combos.

Figure 2: 3x Combo

Figure 3: 4x Combo

Figure 4: 6x Combo (2 sets of blocks)

Chains are more difficult to pull off. Simple chains require a bit of planning, but it is possible to pull off some very complex, well-thought-out chains. A chain is when at least one combo causes blocks to fall directly into another combo. The figures below feature some examples of chains in ascending difficulty.

Figure 5: Basic Chain

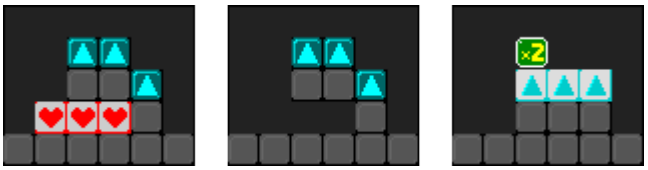


Figure 6: Raised Skill Chain

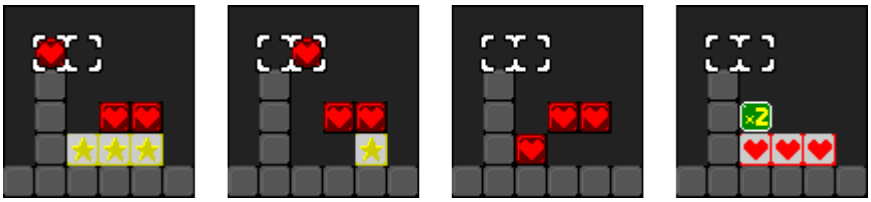


Figure 7: Advanced Chain

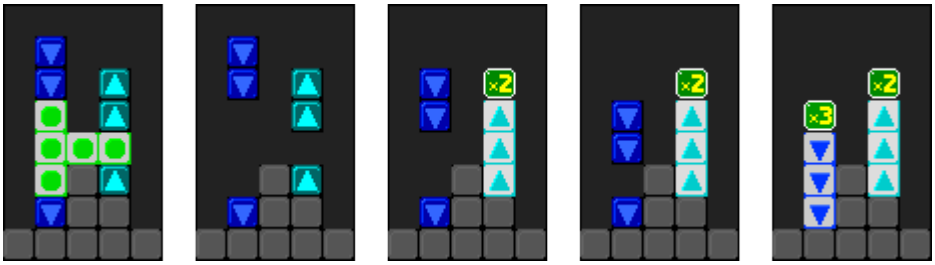


Figure 8: Time-Lag Skill Chain

Tools, Libraries, and Version Control:

We will be using Visual Studio 2010 as our development environment during the production Triforce. The game will require the GLUT library for graphics and sounds, and the 2DGraphics library. Due to some limitations of the 2DGraphics library, we will be adding several features to the header file. It will run in a Windows environment and compiled an x86 architecture. We are going to manage our source code and resources with Git and use Github to host our Git repository. We chose Git because both of us know the benefits of version control, and I have experience with Git. Github was chosen because of its many rich features for developers, such its bug/enhancement ticket system, commit tracking tools, wiki, and online code viewing.

Glossary

Combo: A game mechanic where a player aligns at least three matching blocks

Chain: A game mechanic that requires a player to use smart timing and positining to cause combos to generate more combos from falling blocks.

OpenGL: An open-source, cross-platform API for creating 2D and 3D computer graphics.

GLUT (OpenGL Utility Toolkit): A closed-source procedural library written in C that contains utilities for OpenGL programs and input devices.

2DGraphics Library: A closed-source interface to GLUT that contains library functions useful for games, created by Dr. Arif Wani.

Version Control: Management of changes to documents, computer programs, or other collections of information.

Repository: A data structure, usually stored on a server, that contains a set of files and directories, historical record of changes in the repository, set of commit objects, and a set of references to commit objects, called heads.

Git: A version control system that is fast, local, and distributed.

Github: A popular website for hosting Git repositories that is free to use.

Commit: A single revision in a revision control system.

Visual Studio 2010: An integrated development environment for Windows.

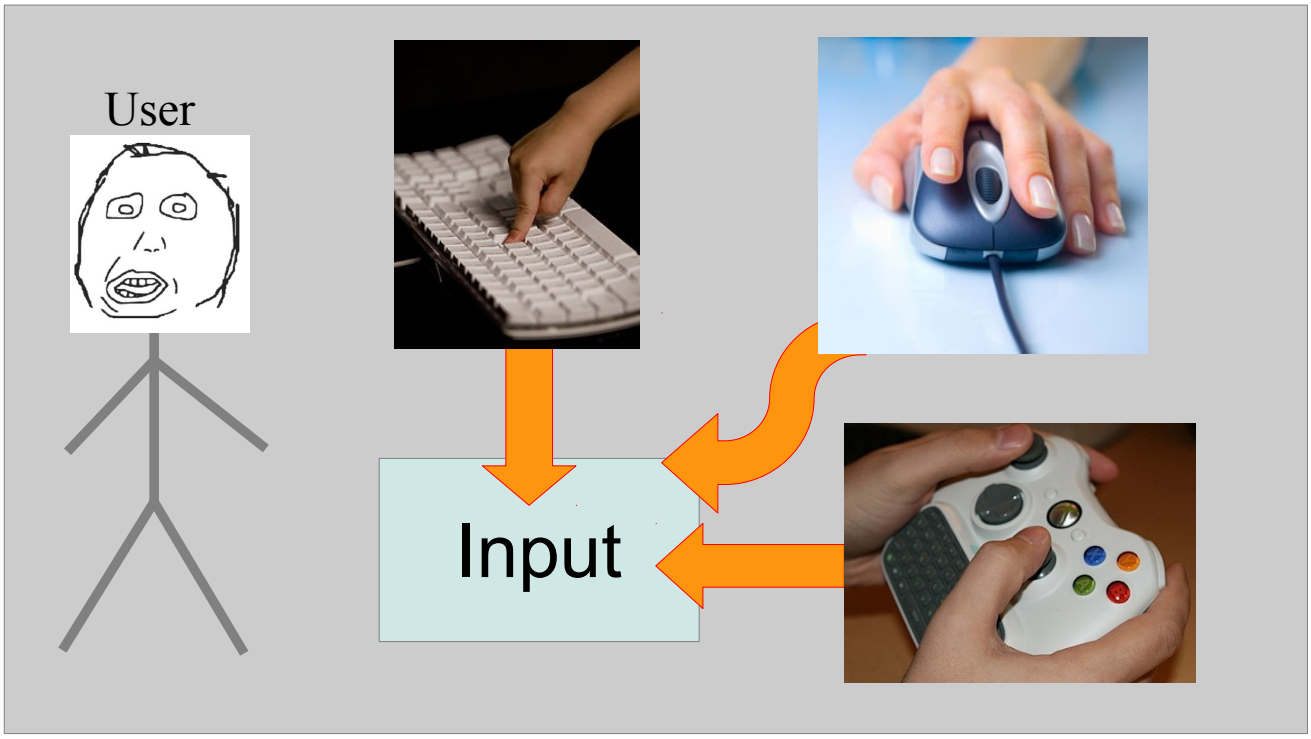
1.1.2 Initial Business Model

When a user runs the game, the first thing that happens is that the configuration file is loaded from disk. The configuration file contains key bindings for input, the preferred default theme, and theme directory location. After loading the configuration file, the initial menu is displayed to the user. The initial menu will display a background image and the buttons used to transition to other states of the game. The menu contains the following buttons: Play, Settings, Help, About, and Quit.

Use-Case 1: User runs the executable

Brief Description: When the user runs the programs, a configuration file is loaded and the main menu is displayed.

Use-Case 2: User activates an input event



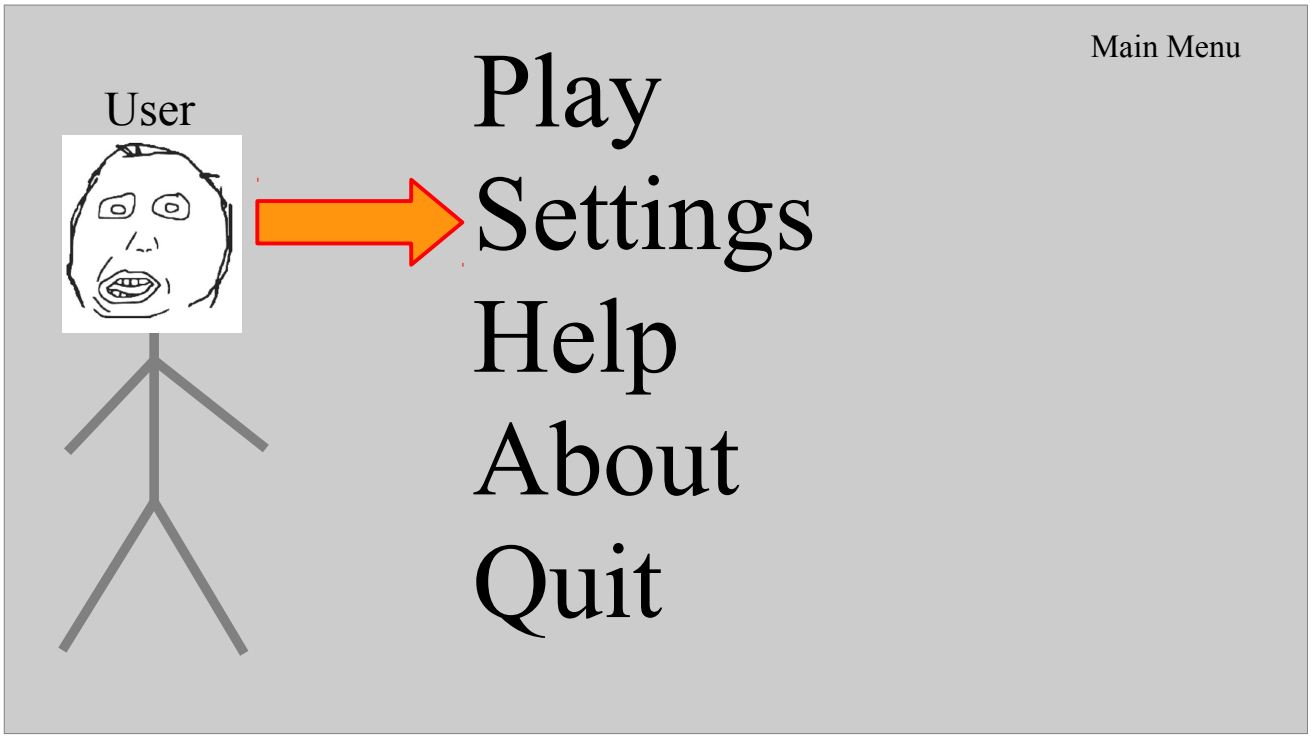
Brief Description: When a user activates a binding on an input device, it is handed by a generic interface via mappings and callback routines.

Use-Case 3: User selects Play

Brief Description:

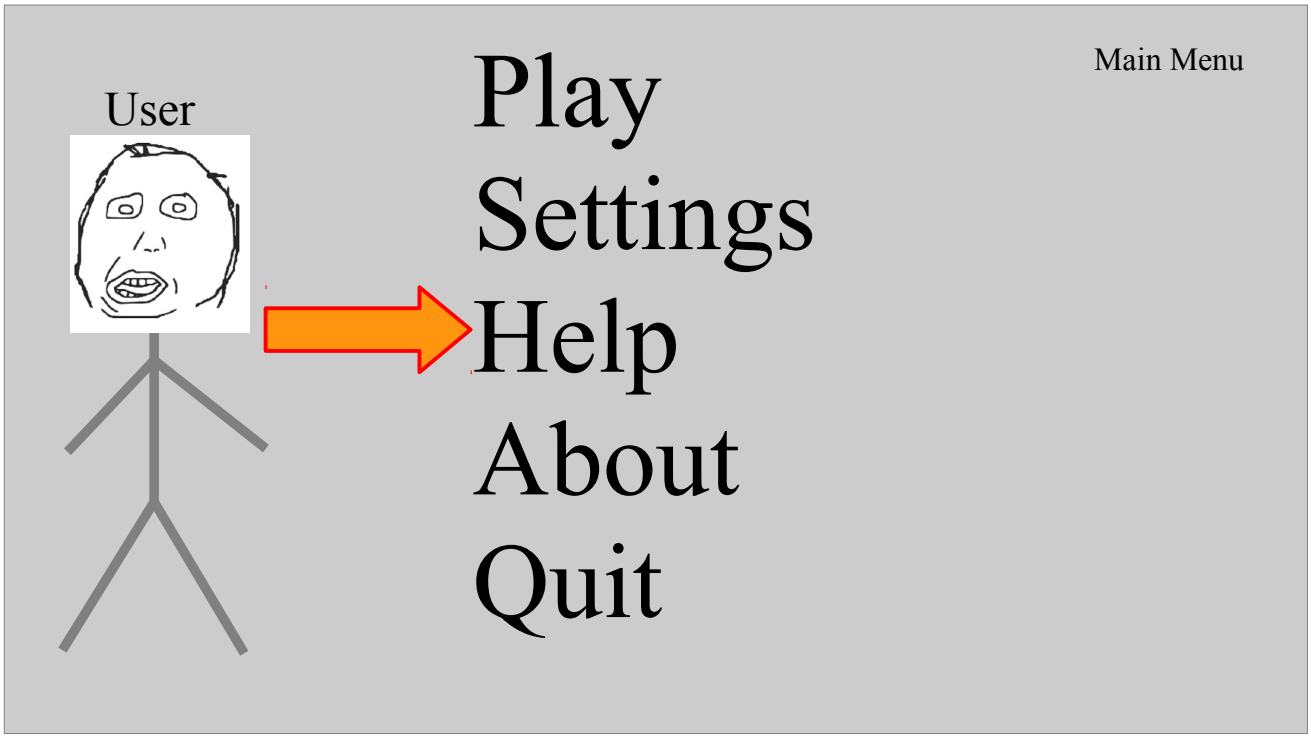
The user selects play on the main menu and the game begins.

Use-case 4: User selects Settings



Brief Description: The user selects Settings and the settings menu is displayed. On the settings menu, the user can change controls, audio, theme, and game play settings.

Use-case 5: User selects Help



Brief Description: The Help button will display a small tutorial about how to play.

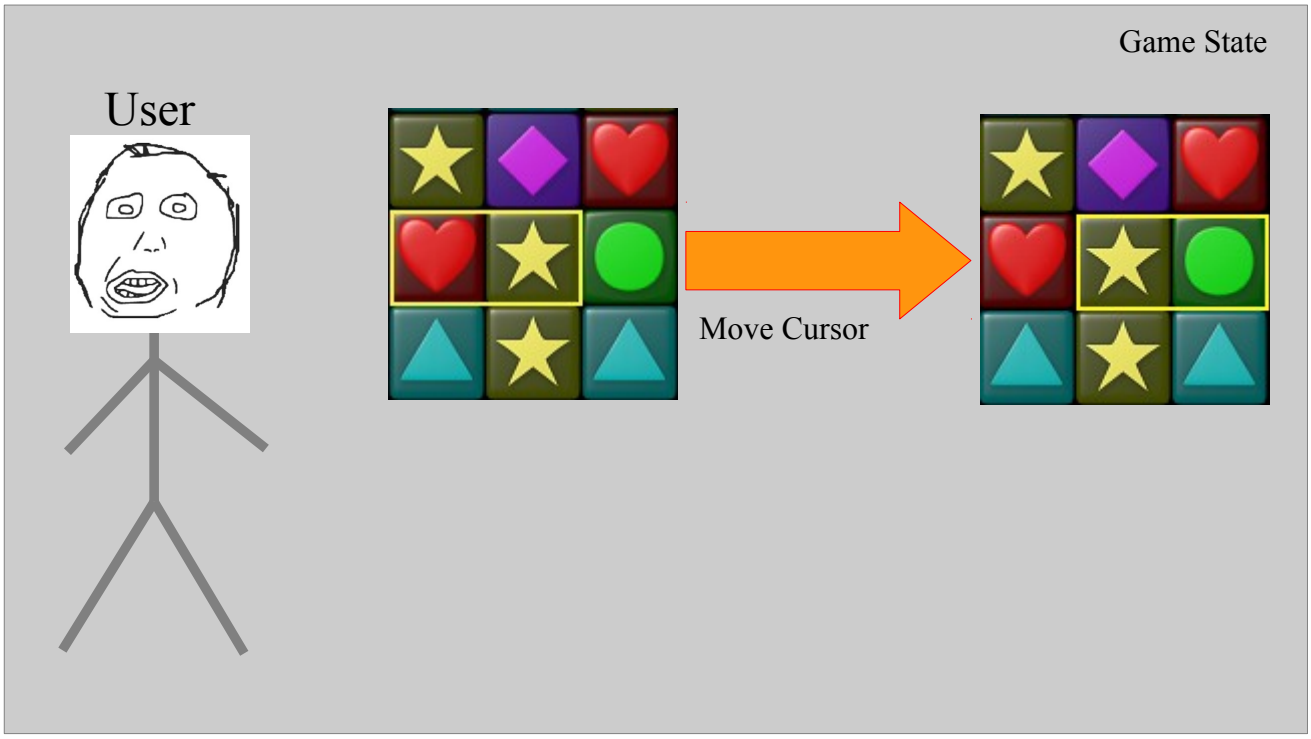
Use-case 6: User selects About

Brief Description: The About button will display information about the authors, licensing information, and a brief description of the game.

Use-Case 7: Use selects Quit

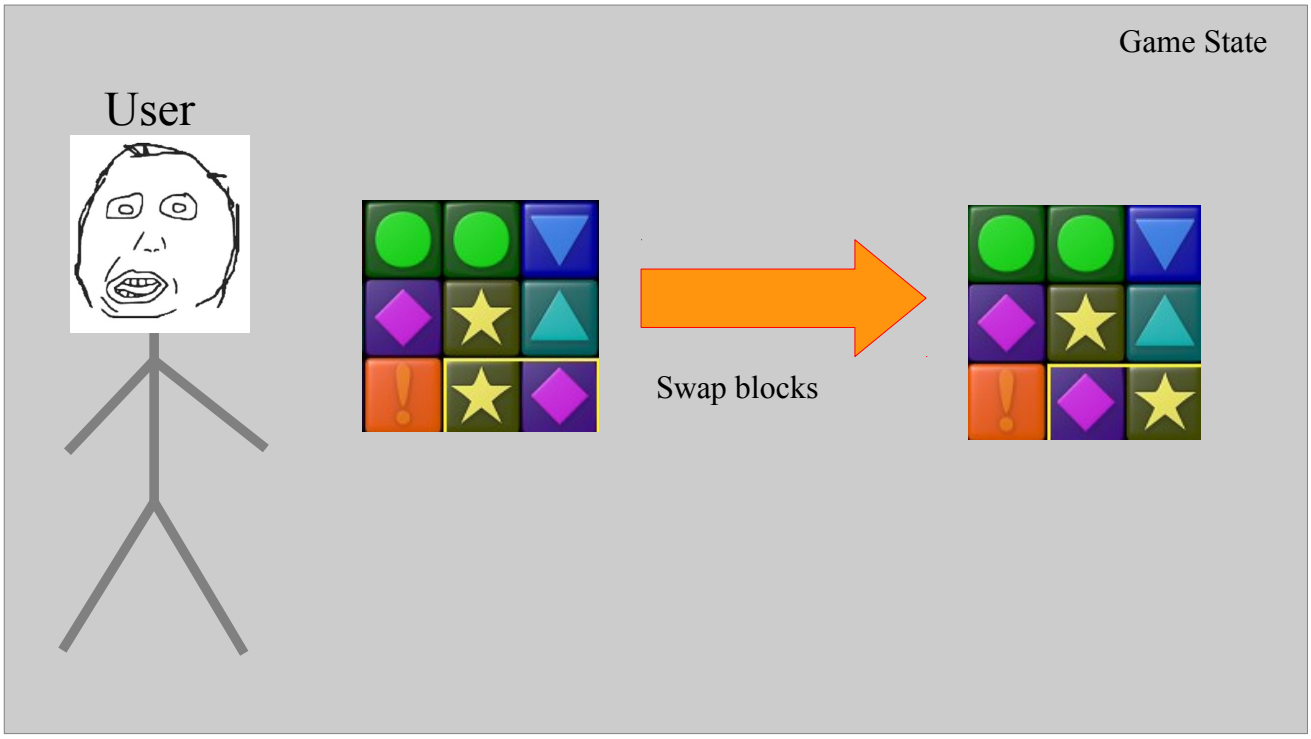
Brief Description: The game will exit when the user selects Quit.

User-Case 8: User moves the cursor



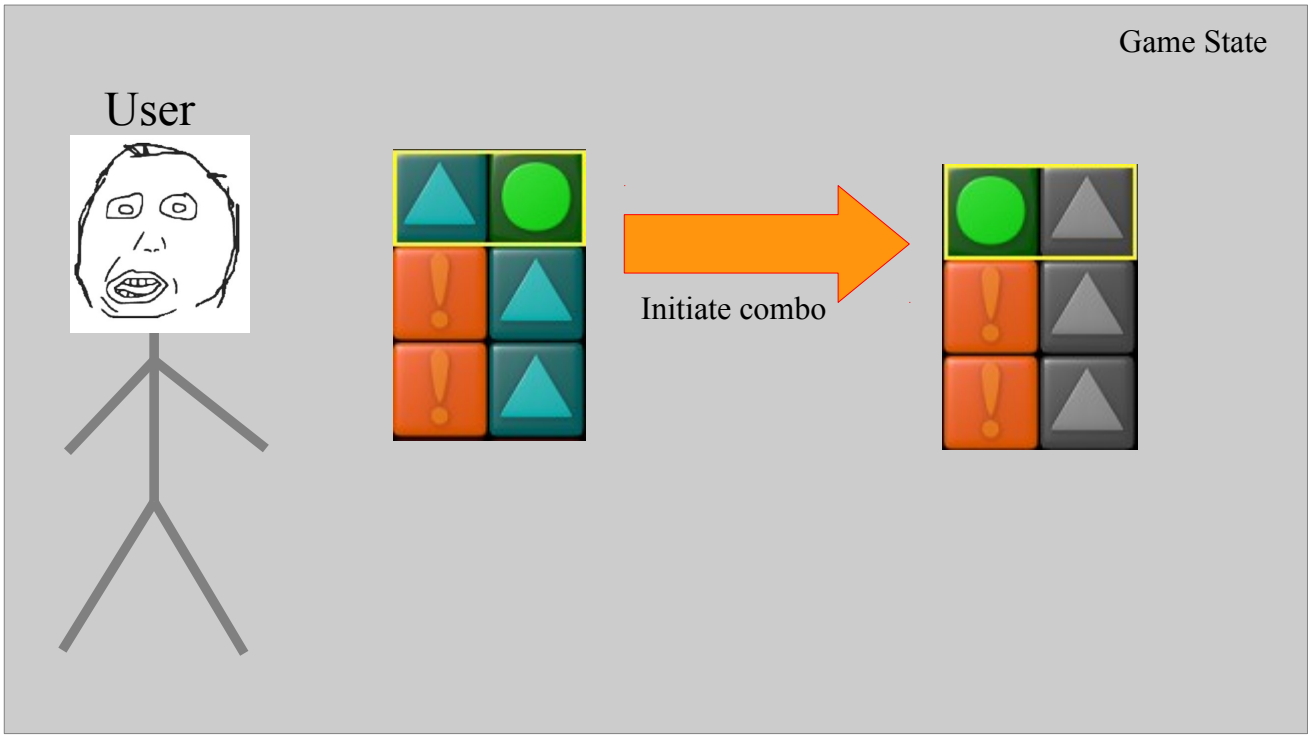
Brief Description: A user moves the cursor and its position is updated to a new cell.

User-Case 9: User swaps blocks



Brief Description: When the user swaps blocks, their positions in the grid are swapped and combo and fall detection must occur.

User-Case 10: User activates a combo



Brief Description: If a user activates a combo, the blocks will momentarily pause, then break, then cause all blocks above them (if any) to fall.

1.1.3 Initial Requirements

1.1.4 Detailed Requirements

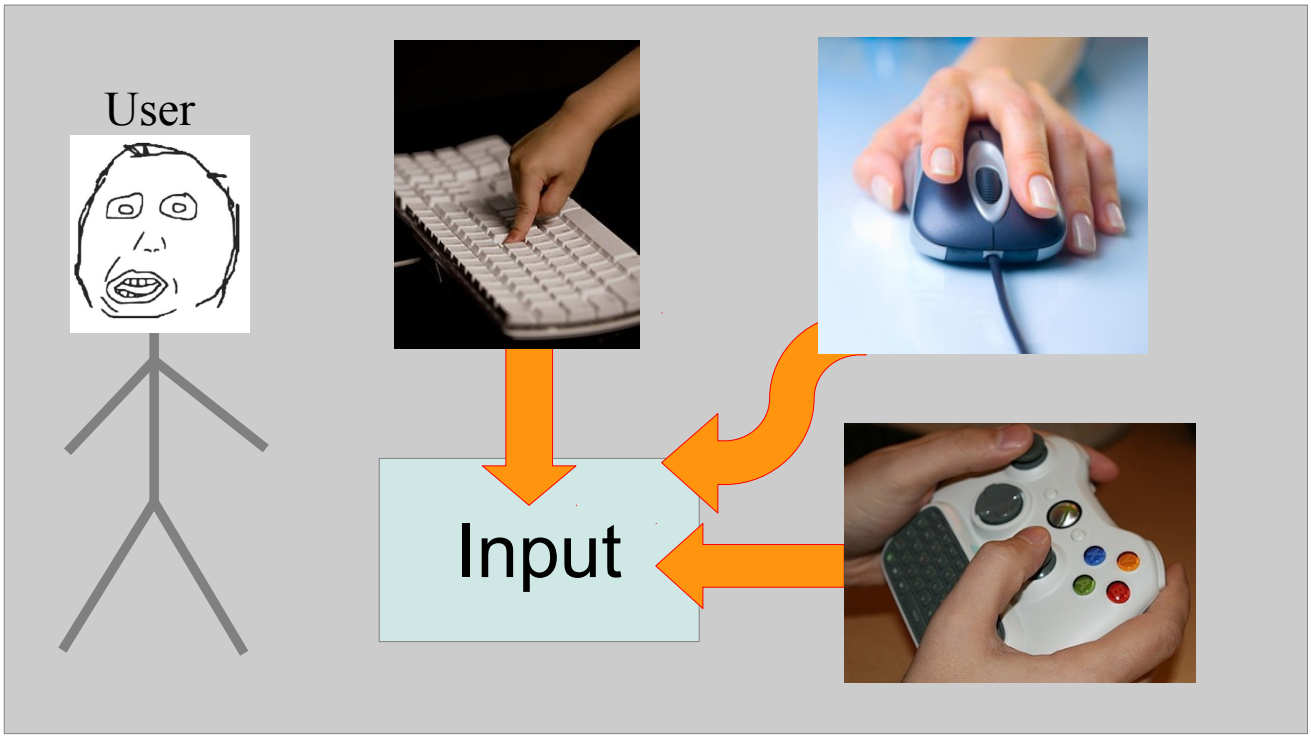
1.1.5 Iterated case-use diagrams

Use-Case 1: User runs the executable

Step-By-Step Description:

1. Run the executable program
2. Configuration file is loaded. (load sane defaults if no config file)
3. initialize menu objects
4. load resources for main menu
5. Change game state to Menu
6. Current menu is set to MainMenu
7. Display the menu buttons
8. Accept Input
9. Handle Input

Use-Case 2: User activates an input event



Step-By-Step Description:

When a user activates an input event, such as from a keyboard, mouse, or Xbox 360 controller, it is handled by a single interface. This interface is a pseudo-device that maintains keybindings for the actual device. For instance, the 'A' key, 'A' button, and left-click can all map to a binding that handles selecting menu items and swapping blocks.

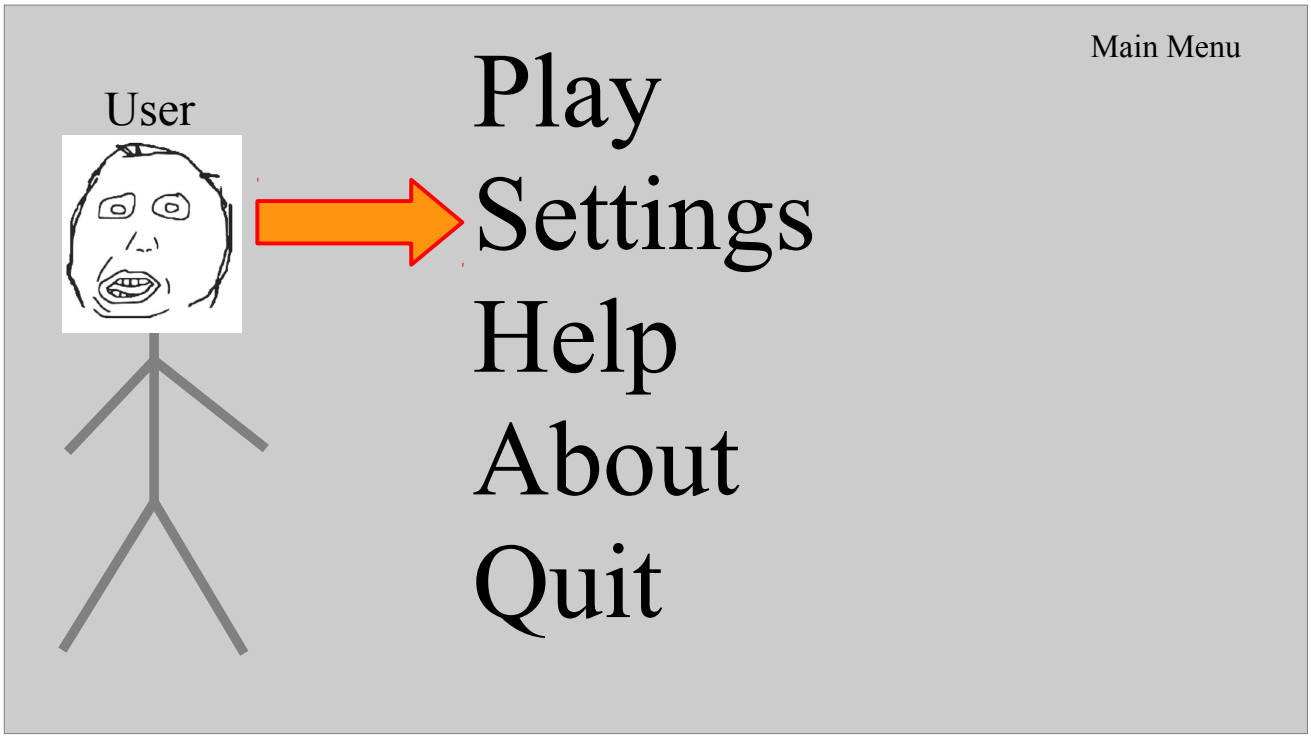
1. User activates a registered key binding
2. Current context passes it to the Input interface.
3. Input interface convert it to a generic key binding
4. Handle action via the registered callback function

Use-Case 3: User selects Play

Step-By-Step Description:

1. User selects Play
2. Input interface handles action via callback for Play
3. Callback function sets game state to Play
4. Game state is changed to Play
5. Initialized Game objects
6. Load resources for game objects
7. Begin a countdown
8. Start the game

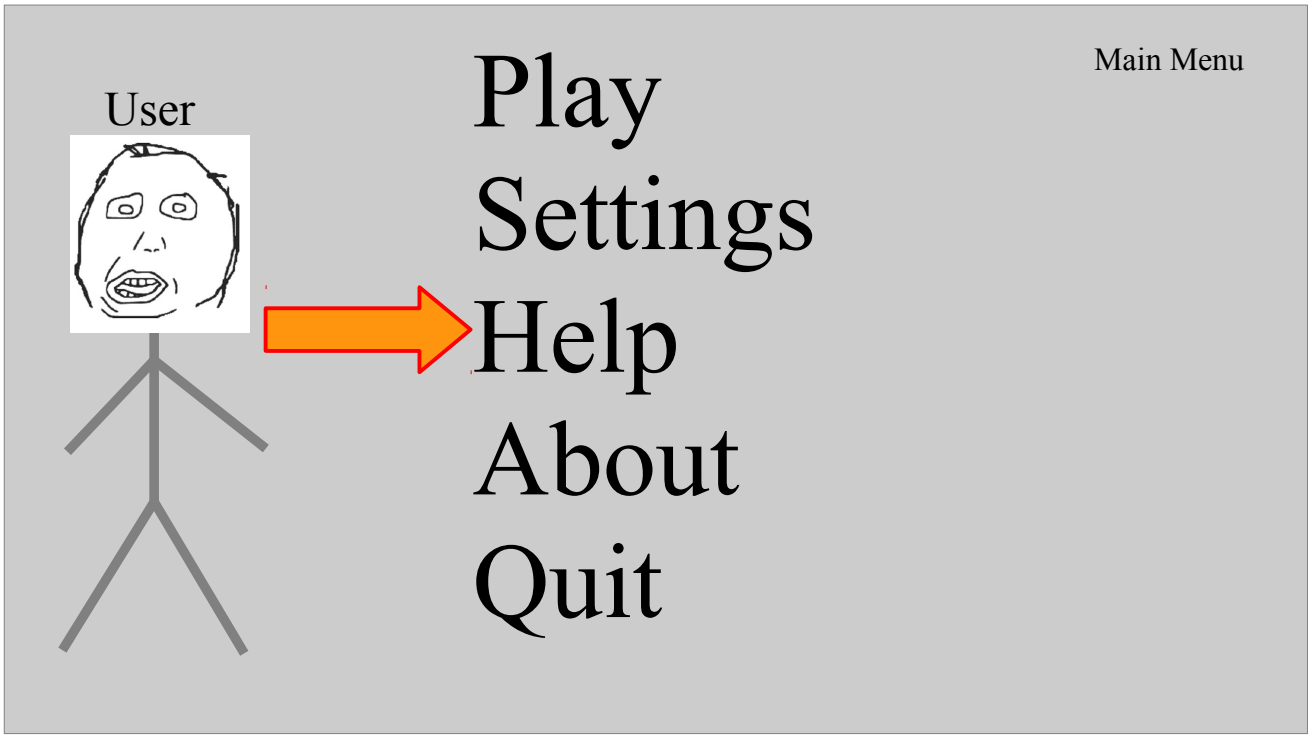
Use-case 4: User selects Settings



Step-By-Step Description:

1. User selects Settings
2. CurrentMenu is set to Settings
3. Settings menu objects are initialized
4. Settings menu resources are loaded
5. Control passed to the user
6. User clicks Save to save settings to configuration file
7. User clicks Back to go back to main menu

Use-case 5: User selects Help



Step-By-Step Description:

1. User selects Help
2. CurrentMenu is set to Help
3. Objects are initialized
4. Resources are loaded
5. Small tutorial is displayed, aided with text and graphics
6. User can click Back to go to main menu

Use-case 6: User selects About

Step-By-Step Description:

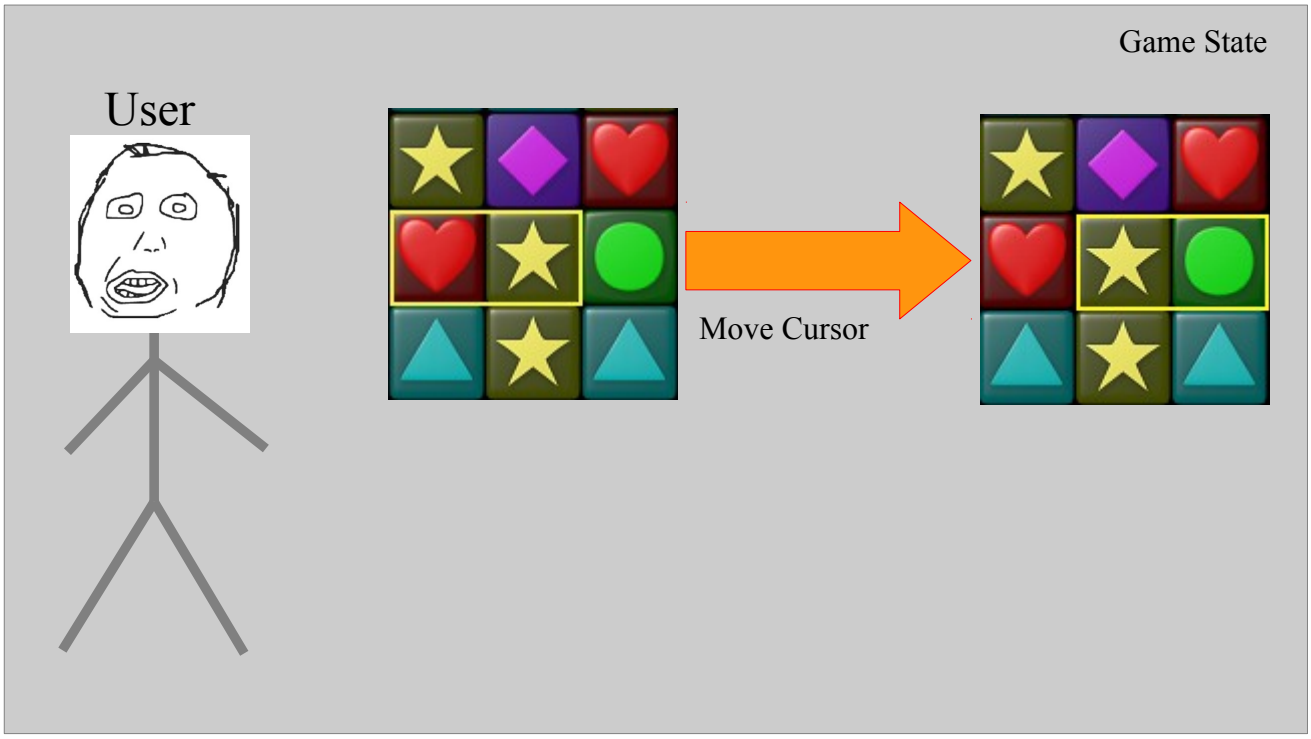
1. User selects About
2. CurrentMenu is set to About
3. Objects are initialized
4. Resources are loaded
5. Brief description of game, licensing/copywrite info, and credits are displayed
6. User can click Back to go to main menu

Use-Case 7: Use selects Quit

Step-By-Step Description:

1. User selects Quit
2. Prompt user, asking if they are sure (if time permits)
3. Free objects and resources
4. Exit the program

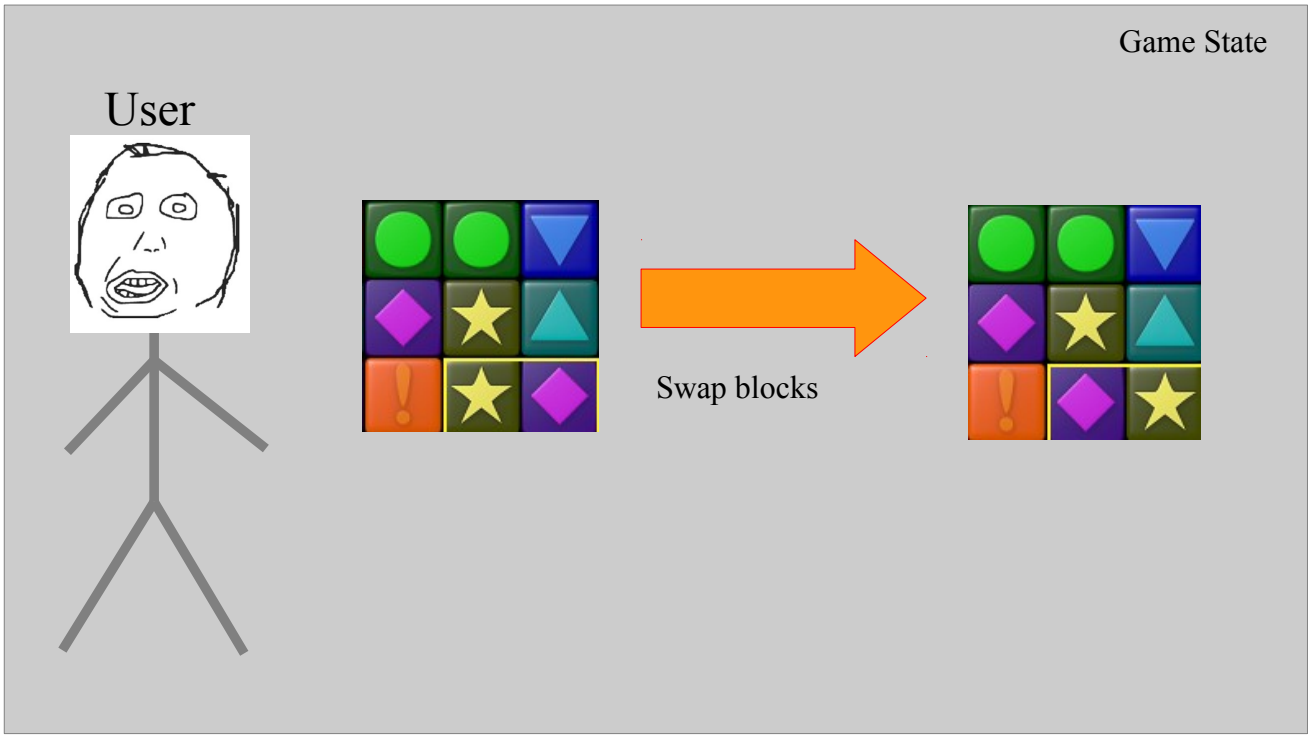
User-Case 8: User moves the cursor



Step-By-Step Description:

1. Check boundaries: Make sure cursor does not leave the game area
2. Calculate new position as an offset of the Grid's position, given the row, column, and block length

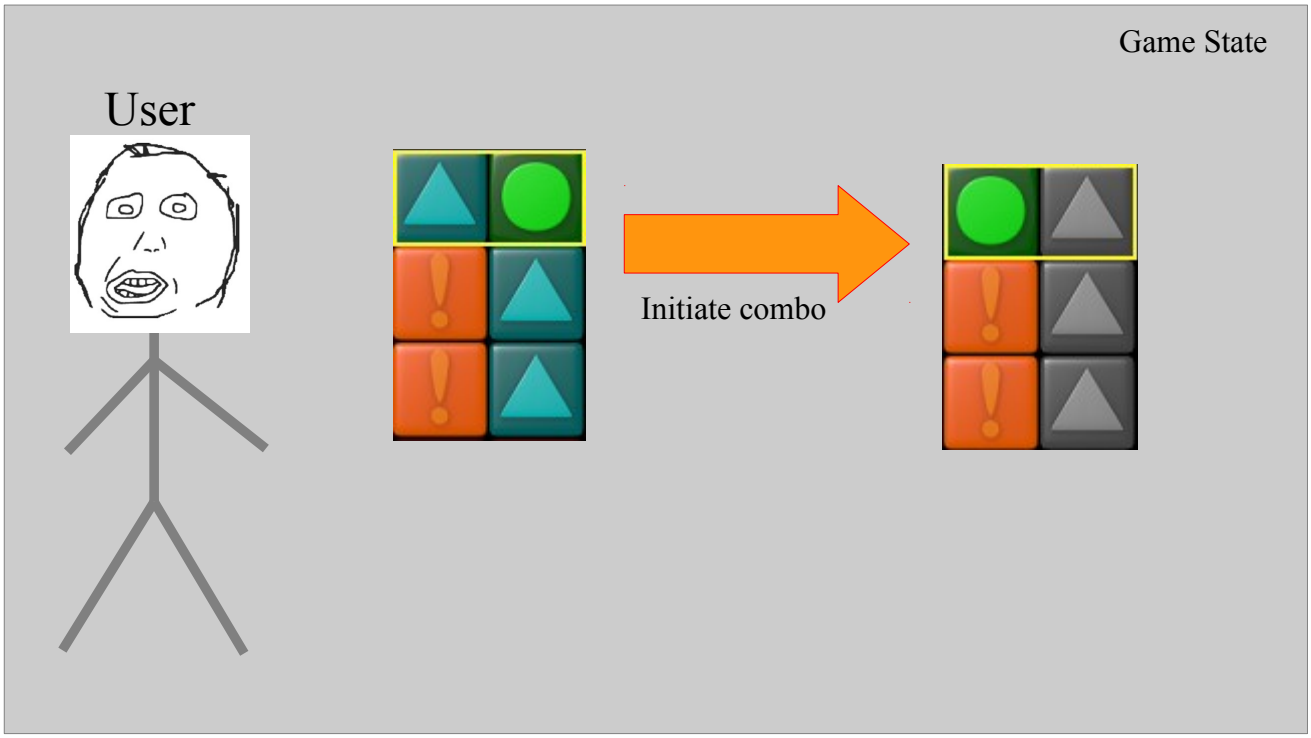
User-Case 9: User activates a combo



Step-By-Step Description:

1. Swap the positions of the blocks in the Grid data structure
2. Detect possible new combos
3. Detect possible new falling states

User-Case 10: User activates a combo



Step-By-Step Description:

1. Calculate the interval that the combo must last, based on the number of blocks broken
2. Set the Blocks being broken to a Combo state
3. Set the Grid to a Combo state
4. Break the blocks after the interval elapses
5. Set blocks above the broken combo (if any) to a Falling state
6. Detect Combos on newly fallen blocks

1.2 Walkthrough

-By Brandon Hinesley

- Spell check (all I saw was artifical -> artificial)
- Explain how blocks flow from bottom up, since this is not typical of block games.
- Need to cover garbage blocks.
- Do not need use-case for starting the game and some menu options
- Draw use-case diagrams correctly, according to software engineering standards

1.3 Revised Requirements after walkthrough

1.3.1 Initial understanding of the project

General

Triforce is a remake of the classic SNES game, Tetris Attack. It is called Triforce in part because of the core game mechanic where one must align at least three matching blocks. In the game, a player controls a cursor that lets him swap two adjacent blocks horizontally. Combinations of 3 or more blocks in a row will break the blocks, causing the blocks above them to fall. The game mechanics allow the player to use their skills and reaction time to gain great rewards. The game mechanics are flexible and allow for several fun modes of play. Figure 1 shows a screenshot of gameplay in Tetris Attack.

Figure 9: This screenshot shows a game with two players. Each player controls the white cursor on their grid of blocks. A player presses a button two swap the blocks inside the cursor.

Menu

The user experience should begin with opening the game to the main menu. The menu will contain buttons for Play, Settings, Help, Credits/About, and Quit. The Settings button will open a new menu to configure settings such as themes, sounds, and difficulty. Help will tell the user how to play. When the user selects Play, they will be taken to another menu to select what mode they want to play. When they select the mode, they will be taken to the actual game. It will start after a timer of three seconds. The gameplay area will consist of a grid of blocks and a cursor for the player to control, as seen in Figure 1. The player uses the keyboard or mouse to move the cursor to an adjacent block horizontally or vertically. The player must break blocks by aligning three or more of the same color vertically or horizontally.

Controls/Input

The controls to Triforce are rather simple. The user only needs to concern himself with the following input (input that will use the same binding is put in the same bullet point):

- Navigate the menus; Move the cursor in four directions
- Select a menu item; Swap blocks
- Boost the speed of play
- Pause/Unpause; Exit

We want to support multiple methods of input. It will be possible to use a keyboard, mouse, or Xbox 360 controller to control everything in the game. This will be done by making generic bindings that are not specific to any kind of input device. Then, we will map each input method from each input device to the generic bindings.

Gameplay and Mechanics

Triforce has potential for several modes of play. Due to time constraints, we will not likely be able to implement them all. At a minimum, we want to implement Endurance Mode, where a player must survive as long as he can as the game gets increasingly more difficult. If time permits, we will also implement Versus Mode. In Versus Mode, two players play against each other by building chains of combos to build garbage blocks that are dropped on the enemy (explained below). In theory, two players could play on one display with separate input (or sharing) input devices, or play over a network connection. Other modes of play allow a user to

play against enemy agents. However, due to time constraints and experience, it is unlikely we will implement any modes that employ artificial intelligence.

In the modes of play we plan to implement, the challenge is to break or “clear” blocks as rows of blocks are pushed into the play area. The blocks are pushed up from the bottom. It starts out (configurably) slow, and the rate that blocks are push gradually accelerates. In Endurance mode, the player must survive as long as possible. In versus mode, the players have to deal with the challenge of the accelerating blocks in addition to clearing garbage blocks dropped by the opponent.

The game play of Triforce looks rather simple to a player, but its inner workings are actually rather complex. The main game mechanics are combos and chains. The mechanics of combos are the easiest to explain. A combo is achieved by aligning at least three matching blocks together. However, you can combine sets of blocks by positioning things right before moving them. Below are some examples of possible combos.

Figure 10: 3x Combo

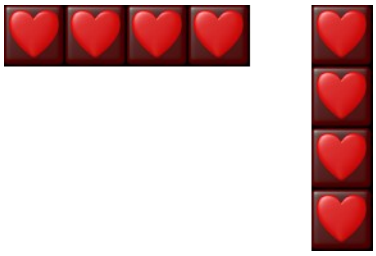


Figure 11: 4x Combo

Figure 12: 6x Combo (2 sets of blocks)

Chains are more difficult to pull off. Simple chains require a bit of planning, but it is only

possible to pull off large and complex chains with practice and skill. A chain is when at least one combo causes blocks to fall directly into another combo. The figures below feature some examples of chains in ascending difficulty.

Figure 13: Basic Chain

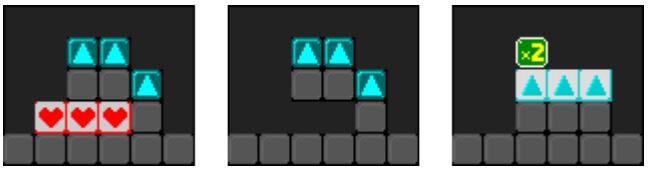


Figure 14: Raised Skill Chain

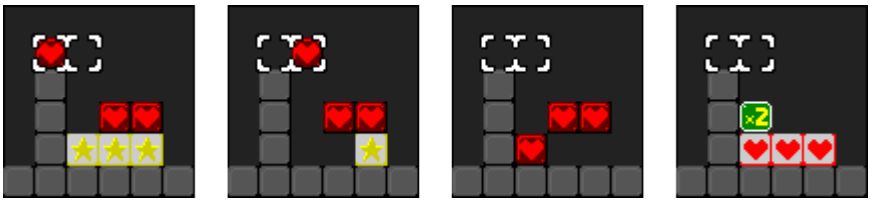


Figure 15: Advanced Chain

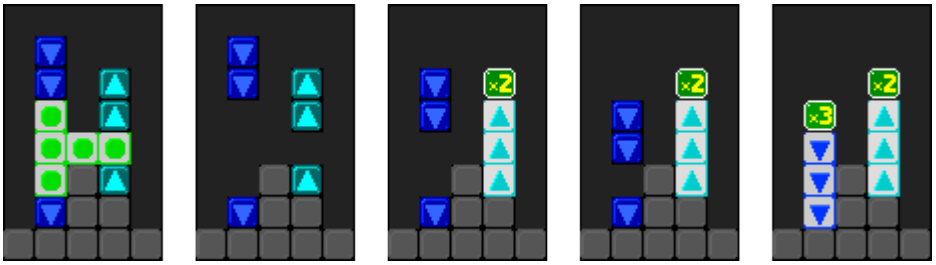


Figure 16: Time-Lag Skill Chain

Another important mechanic to cover is garbage blocks. Garbage blocks are formed in Versus mode when a player makes a chain. The amount of garbage blocks generated is directly related to the size of the chain. An example of garbage blocks can be seen in Figure 1. The garbage blocks are the set of blue blocks with an angry face in the middle. It was generated by the player on the left and dropped on the player on the right from above. When a combo is generated adjacent to a set of garbage blocks, all of the garbage blocks break and randomly generate new regular blocks.

Tools, Libraries, and Version Control:

We will be using Visual Studio 2010 as our development environment during the production Triforce. The game will require the GLUT library for graphics and sounds, and the 2DGraphics library. Due to some limitations of the 2DGraphics library, we will be adding several features to the header file. It will run in a Windows environment and compiled an x86 architecture. We are going to manage our source code and resources with Git and use Github to host our Git repository. We chose Git because both of us know the benefits of version control, and I have experience with Git. Github was chosen because of its many rich features for developers, such its bug/enhancement ticket system, commit tracking tools, wiki, and online code viewing.

Glossary

Combo: A game mechanic where a player aligns at least three matching blocks

Chain: A game mechanic that requires a player to use smart timing and positining to cause combos to generate more combos from falling blocks.

Garbage Blocks: Blocks generated by a player in Versus mode by creating chains. Larger chains creater more garbage blocks.

OpenGL: An open-source, cross-platform API for creating 2D and 3D computer graphics.

GLUT (OpenGL Utility Toolkit): A closed-source procedural library written in C that contains utilities for OpenGL programs and input devices.

2DGraphics Library: A closed-source interface to GLUT created by Dr. Arif Wani that contains library functions useful for game development.

Version Control: Management of changes to documents, computer programs, or other collections of information.

Repository: A data structure, usually stored on a server, that contains a set of files and directories, historical record of changes in the repository, set of commit objects, and a set of references to commit objects, called heads.

Git: A version control system that is fast, local, and distributed.

Github: A popular website for hosting Git repositories that is free to use.

Commit: A single revision in a revision control system.

Visual Studio 2010: An integrated development environment for Windows.

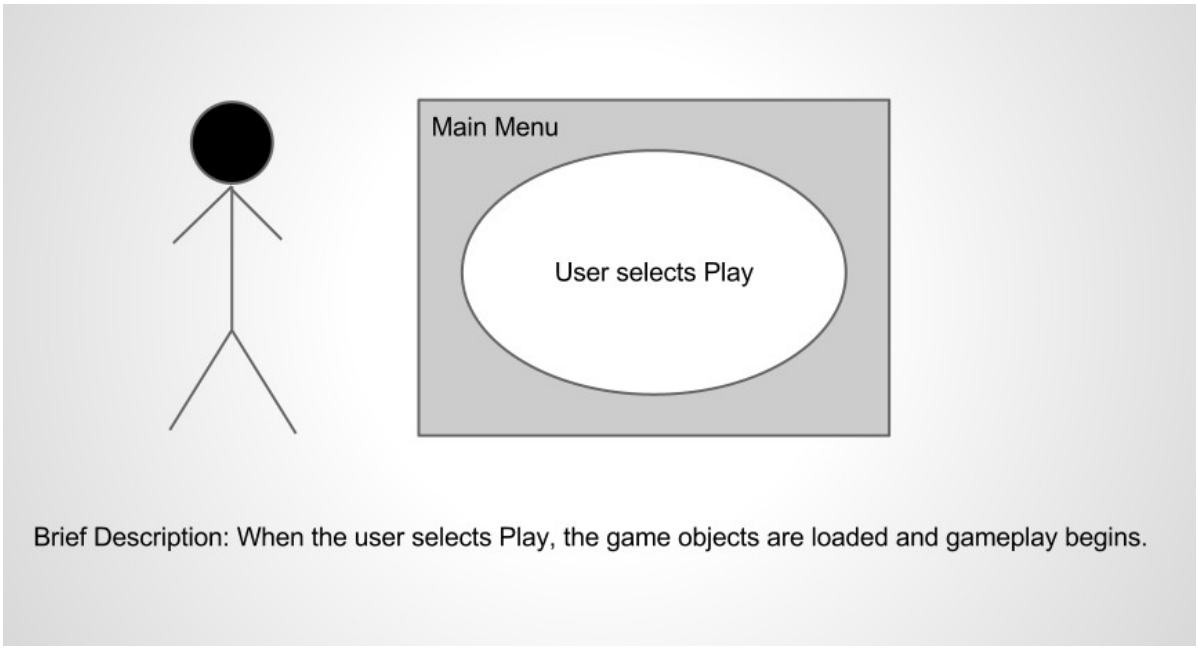
1.3.2 Initial Business Model

When a user runs the game, the first thing that happens is that the configuration file is loaded from disk. The configuration file contains key bindings for input, the preferred default theme, and theme directory location. After loading the configuration file, the initial menu is displayed to the user. The initial menu will display a background image and the buttons used to transition to other states of the game. The menu contains the following buttons: Play, Settings, Help, About, and Quit.

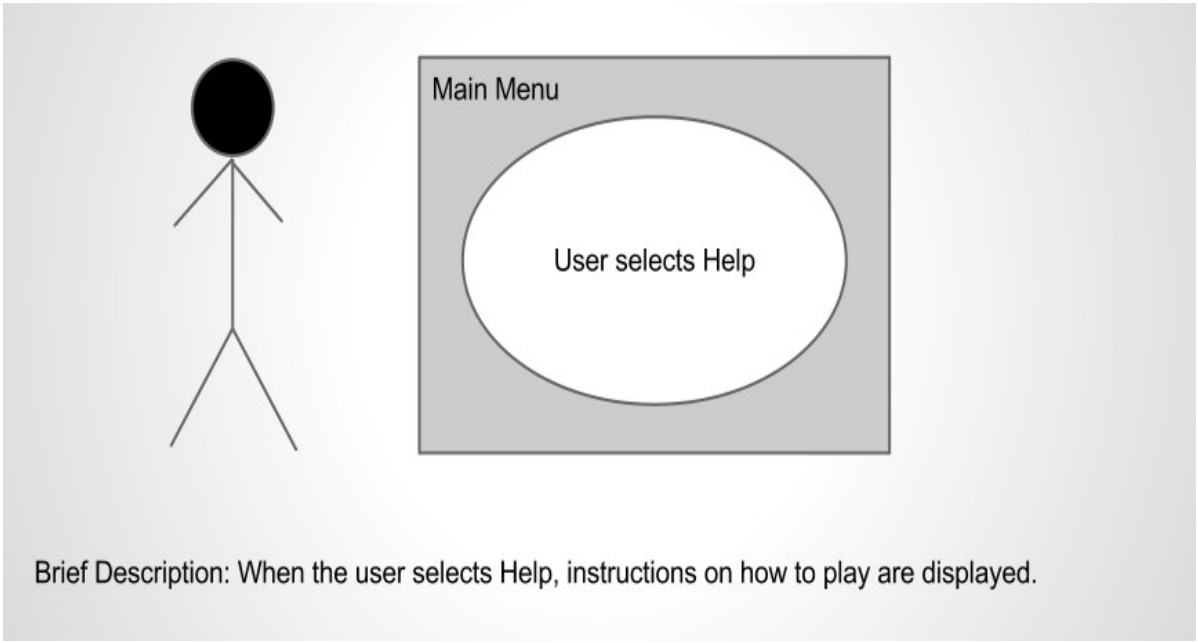
We plan to support multiple input devices. A single interface will control the mappings from input devices to a pseudo input device.

The actual gameplay will have several different states, including Play, Pause, Combo, and Countdown. Individual blocks will also need to have their own states, including Enabled, Disabled, Inactive, Combo, Falling.

Use Cases

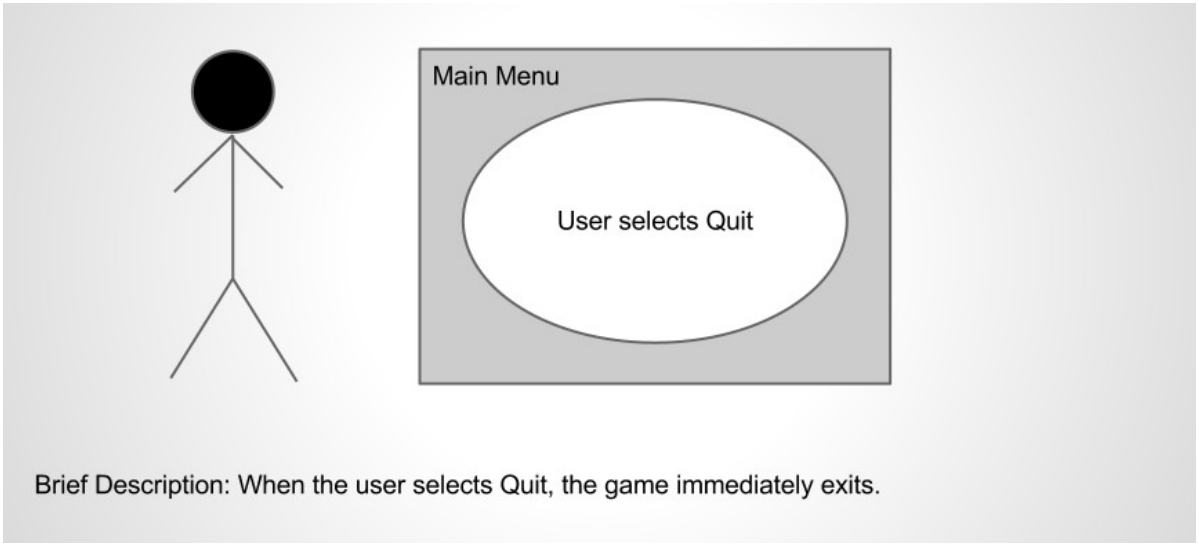


Brief Description: When the user selects Play, the game objects are loaded and gameplay begins.

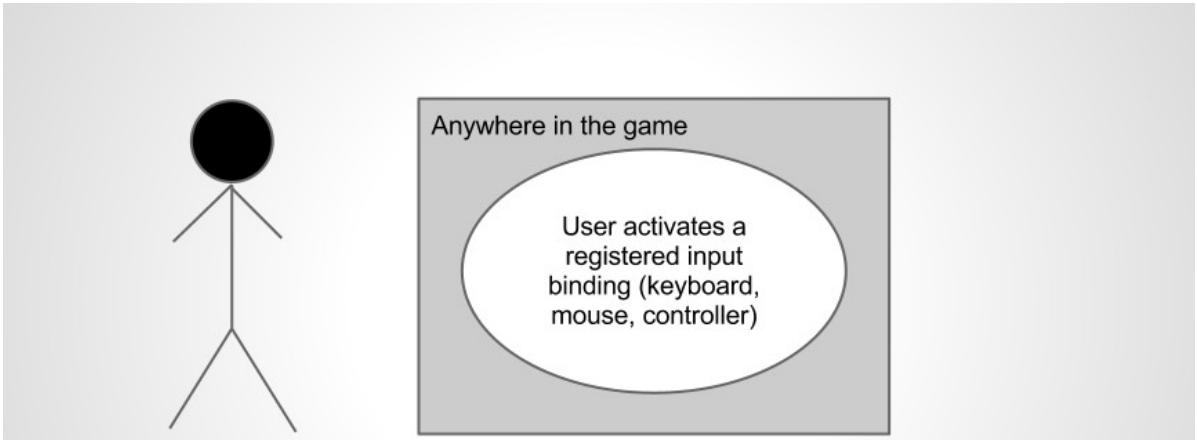


Brief Description: When the user selects Help, instructions on how to play are displayed.

Use Cases

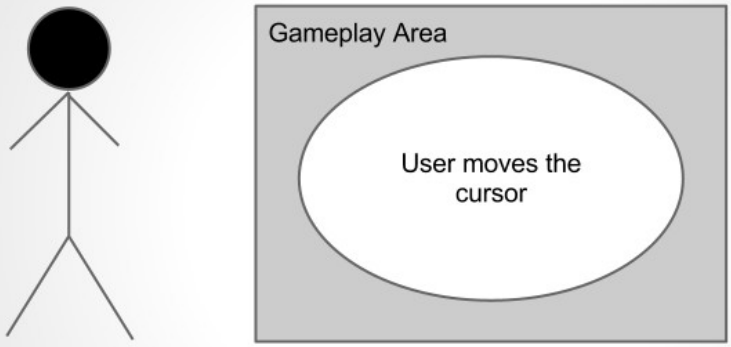


Brief Description: When the user selects Quit, the game immediately exits.

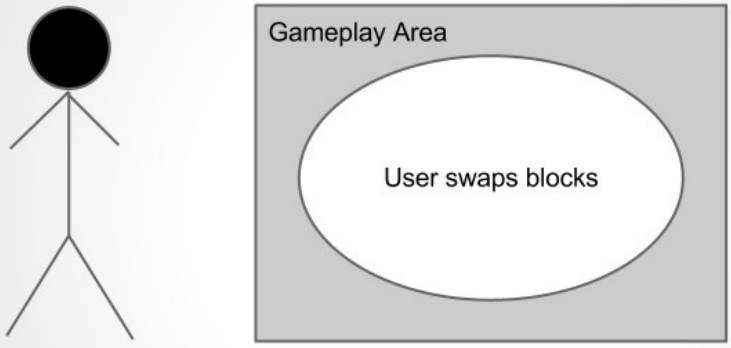


Brief Description: When the user activates a registered input binding, such as pressing/holding a key, clicking the mouse, or pressing a button on an xbox controller, its registered callback function is called.

Use Cases

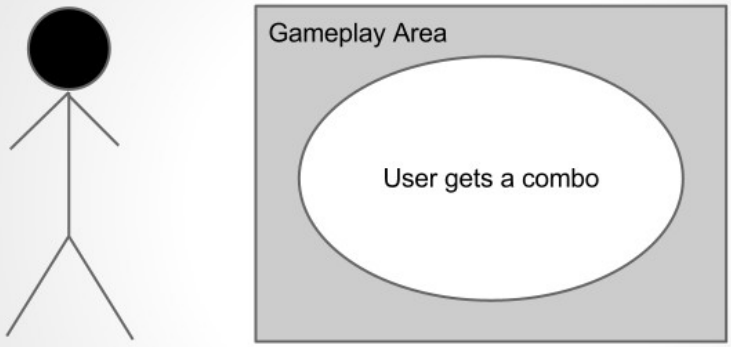


Brief Description: When the user moves the cursor, the cursor's position is updated one block in the direction the user requests, and the graphic is redrawn.

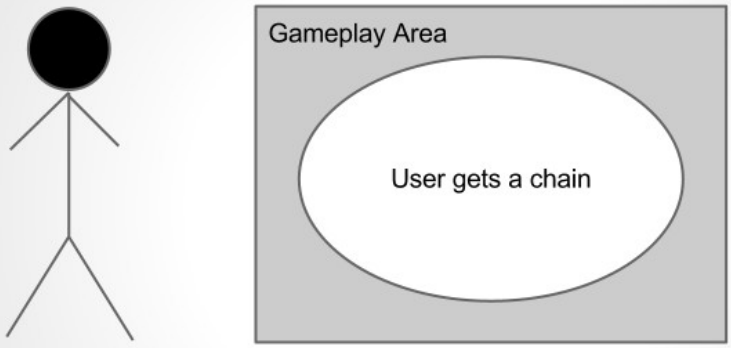


Brief Description: When the user swaps the two blocks, the graphics and states of the selected blocks are swapped with each other.

Use Cases

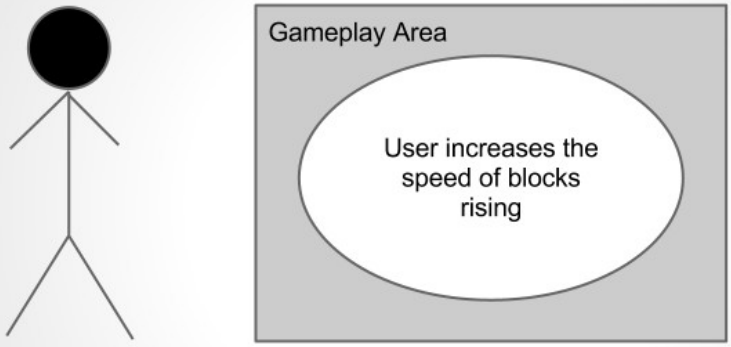


Brief Description: When the user gets a combo, blocks will temporarily stop rising. An animation is played, and then the blocks break. The blocks above the cleared combo will fall down.

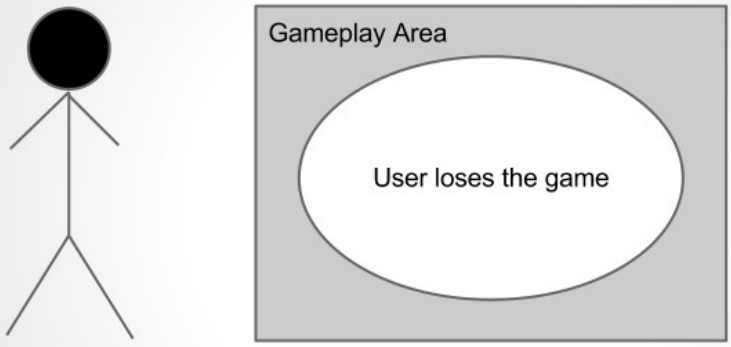


Brief Description: When the user gets a chain, it will act just like a regular combo. However, a bonus will be displayed to indicate that a chain occurred.

Use Cases



Brief Description: The user can activate a key binding that forces blocks to rise at a faster rate. This rate will be constant, and the current speed will be temporarily replaced with this constant while the binding is activated.



Brief Description: The user loses the game when a block hits the top of the grid.

1.3.3 Obtain Initial Requirements

The user begins the game and is presented with the main menu. The options are Play, Help, and Quit. The user begins the game by selecting Play. A tutorial is displayed if he selects Help. The game exits if he selects Quit. The game must support multiple input devices, so multiple bindings must map to a single action.

The user has a limited amount of actions. While in game, the user can:

- move in the cursor up, down, left, and right
- temporarily increase the speed that blocks rise into the play area
- swap blocks

When the user swaps blocks, it has a possibility of causing a combo, thus changing the state of the game.

1.3.4 Obtain Detailed Requirements

Input:

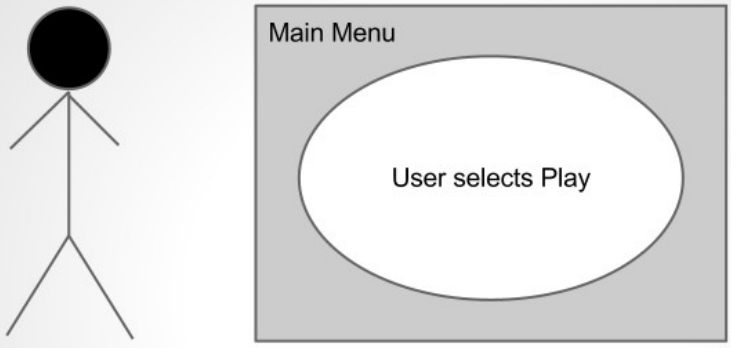
- Since multiple input devices must be supported, a generic interface must be able to handle all of them.
- Different states must be recognized, such as when a binding is pressed, held down, and released.
- More details in Brandon's report

Game Mechanics:

- The cursor moves one space at a time.
- The cursor cannot move outside of the grid.
- When a block comes into contact with the top of the grid, the player loses the game.
- A timer will keep track of the time played and be displayed on the screen.
- The score will be displayed on the screen.
- Chains must be kept track of between combos and fall events.
- At least 3 blocks form a combo, but it is possible to have up to 14 in a single combo.
- After a combo clears, an algorithm must detect which blocks need to fall down.
- When a falling block lands, an algorithm must detect if a combo occurs.
- When a combo >=4 or a chain occurs, it shall display an animated bonus where the event happened.
- Combos will multiply the number of blocks in the combo by a constant and add it to the score.
- Chains will act as multipliers to the combo that occurs and drastically increase the score.

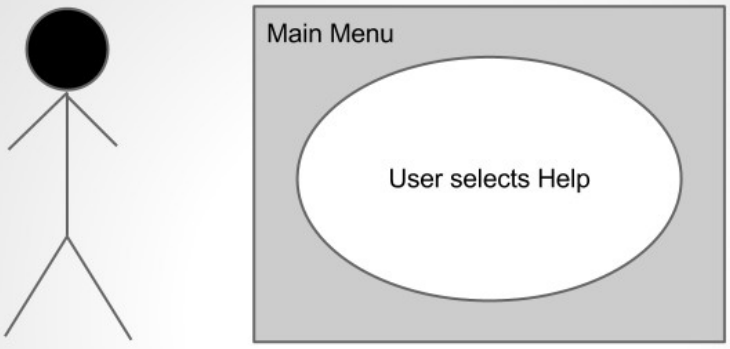
1.3.5 Obtain Detailed Requirements

Use Cases



Updated Description: When the user selects Play, the game state is changed. The grid, cursor, blocks, timer, and score are initialized.

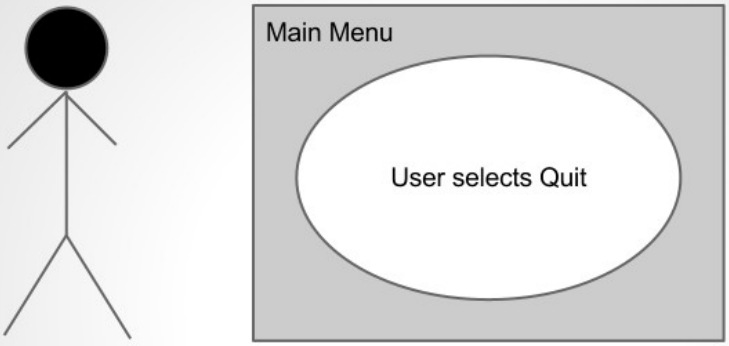
Step-by-step:
1. User selects Play
2. Change game state to PLAY
3. Initialize grid, cursor, blocks, timer, and score



Updated Description: When the user selects Help, instructions on how to play are displayed.

Step-by-step:
1. User selects Help
2. Display text/graphics at the correct position that explain how to play.

Use Cases



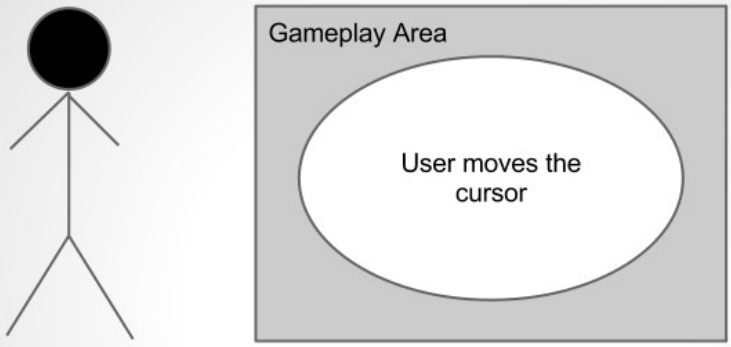
Updated Description: When the user selects Quit, the game immediately exits.

Step-by-step:
1. User selects Quit
2. Game exits

Updated Description: When the user activates a registered input binding, and the binding is in a registered state (e.g. on_press, is_down, _on_release), its registered callback method is invoked.

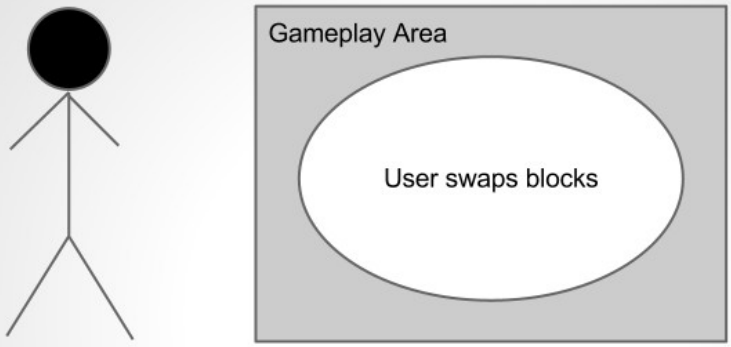
Step-by-step:
1. Map a keybinding to an action and an input state
2. User activates keybinding
3. Callback method is invoked

Use Cases



Updated Description: When the user moves the cursor, the cursor's position is updated one block in the direction the user requests, and the graphic is redrawn.

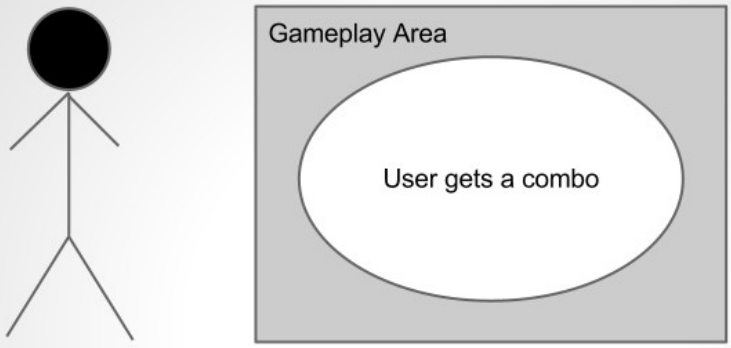
Step-by-step:
1. User input calls cursor move method.
2. Calculate cursor's new coordinates based on the direction it is moved.
3. Do not let cursor escape bounds of grid.
4. Redraw the cursor.



Updated Description: When the user swaps the two blocks, the graphics and states of the selected blocks are swapped with each other. Combo detection and fall detection algorithms are executed.

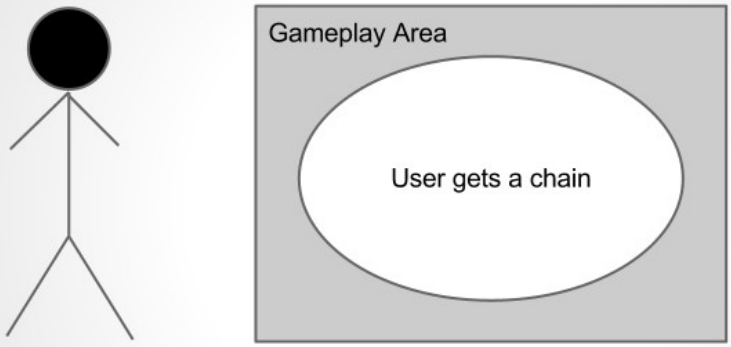
1. User input calls swap block method.
2. The graphics and states of the selected blocks are swapped with each other.
3. Detect combos for both blocks.
4. Detect falls for both blocks

Use Cases



Updated Description: When the user gets a combo, the blocks in the combo go into a combo state. All blocks will temporarily stop rising. A combo with 4 or more blocks displays a bonus animation. The blocks in the combo display a special effect. Detect if any blocks need to fall.

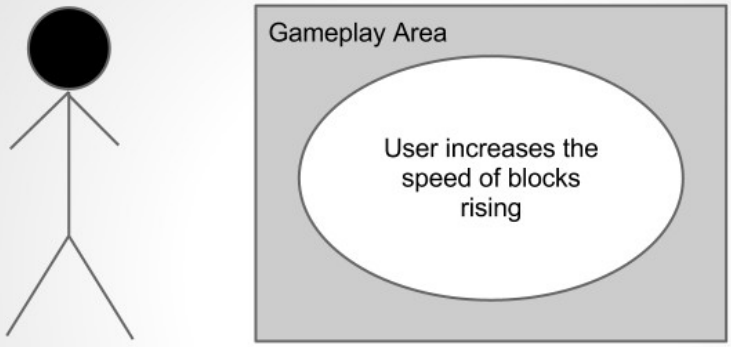
1. Combo is detected
2. Set state of Blocks in the combo to COMBO and state of Grid to COMBO
3. Display animated bonus if combo has 4 or more blocks in it.
4. Display different graphic while blocks are in COMBO state and pause the rising of new blocks while grid is in COMBO state
5. After time for combo has elapsed, set state of blocks in combo to DISABLED, and Grid state to PLAY
6. Detect blocks that need to fall



Updated Description: When the user gets a chain, it will act just like a regular combo. However, a bonus will be displayed to indicate that a chain occurred.

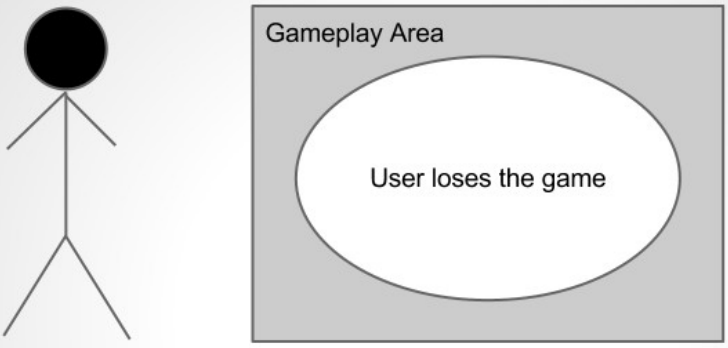
Step-by-step:
Same as combo, except for step 3.
3. Display animated bonus if combo has 4 or more blocks in it, and display an animated bonus for the chain.

Use Cases



Updated Description: The user can activate a key binding that forces blocks to rise at a faster rate. This rate will be constant, and the current speed will be temporarily replaced with this constant while the binding is activated.

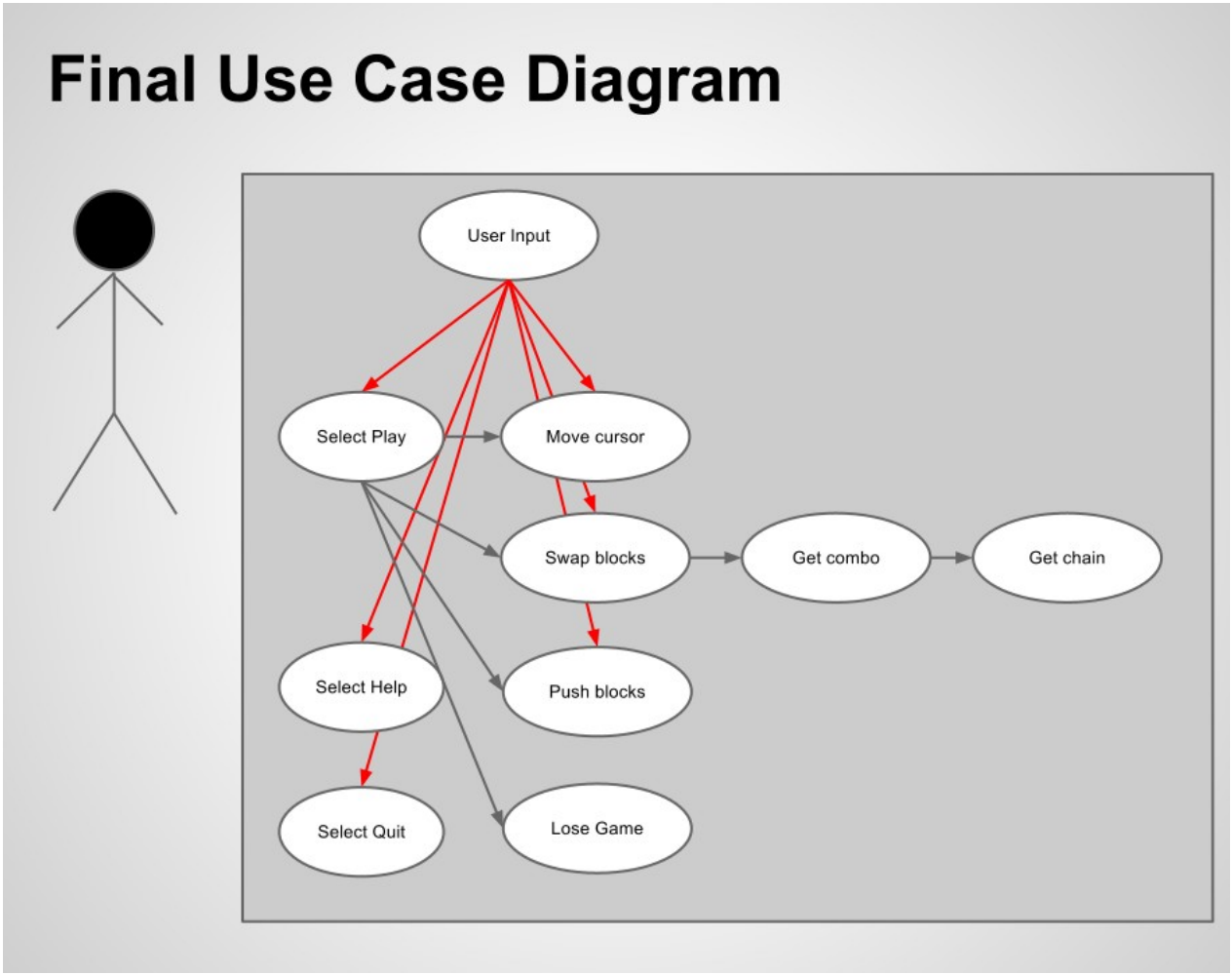
1. User input invokes method to make blocks rise faster.
2. Grid's state changed to PUSH
   - while in PUSH state, automatic rising of blocks is disabled.
3. Blocks are pushed up by 1 pixel on a short, constant interval while the input binding is active.
4. When user releases the button, the Grid's state is changed to PLAY.



Brief Description: The user loses the game when a block hits the top of the grid.

1. Block hitting the top of the Grid is detected.
2. Grid state changes to GAMEOVER.
   - While in GAMEOVER state, blocks stop pushing up.
3. Initialize a countdown timer.
4. Display graphics for game over screen.
5. Go back to main menu.

Use Cases


Final Use Case Diagram
User Input
Select Play
Move cursor
Swap blocks
Get combo
Get chain
Select Help
Push blocks
Select Quit
Lose Game