Requirements and Analysis Document for Panic on TDAncefloor

Version: 1.2

Date: 2017-05-11

Author: Farzad Besharati, Gustav Lahti, Rikard Teodorsson, Magnus Wamby

This version overrides all previous versions.

1 Introduction

This application exists purely to entertain its users. The application is a top-down shooter where the user has the ability to change their attacks by connecting modules together. These modules are dropped by enemies when killed.

1.1 Definitions, acronyms and abbreviations

The user is the person playing the game.

The player is the character the user is controlling.

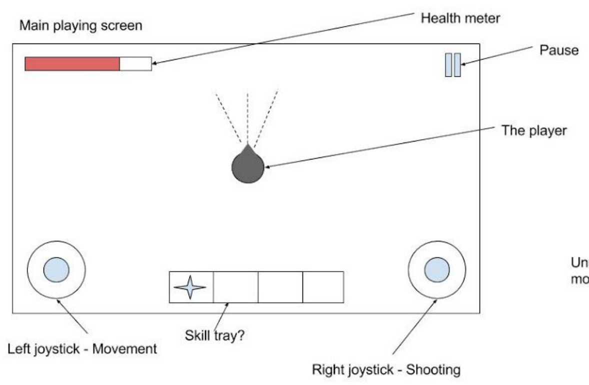
The inventory is a collection of items used to modify a character or a character's attack(s).

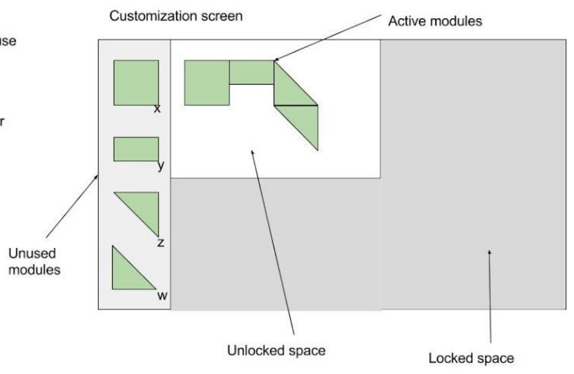
The storage is a collection of the player's unused items.

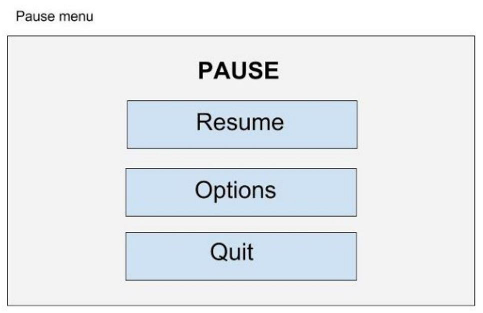
2 Requirements

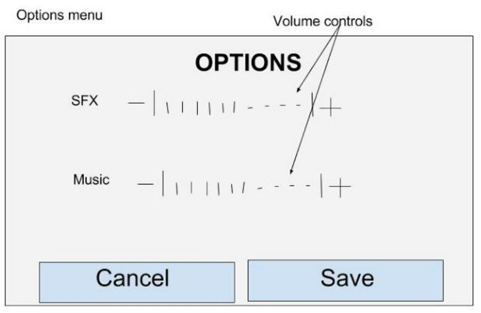
2.1 User interface

Skill tray may or may not be part of the final product.









2.2 Functional requirements

What will the user be able to do ? Write a list of use case names (id's) in the language of the customer. The specific flows for each use case is recorded below. Specify a use cases in priority order.

The user will be able to:

- Start the game
- Quit the game
- Move player (use case 1, Movement)
- Fight enemies
    - Attack (use case 2, Shoot)
        - Kill enemies (use case 3, EnemyDies)
    - Take damage (use case 4, DamageTaken)
        - Die
- Pause (use case 7, PauseMenu)
    - Return to the game
    - Quit to main menu
    - Change options
        - Change volume
- Pick up items (use case 6, ItemCollect)

- Customize the player by equipping items (use case 5, CharacterCustomization)
    - Modify attacks
    - Modify player
- Progress with increasing difficulty

2.3 Non-functional requirements

Any special considerations besides functionality? Usability, reliability, performance, supportability, legal, implementation, ... NOTE: Testability mandatory (must have tests)

- Must follow MVC
- The Model must be testable and have valid tests that all pass
- Should run at 60 FPS

3 Use cases

3.0 UC: 0, RunGame

Summary: The user starts up the game and is greeted with the main menu, where a variety of options exist.

Priority: High

Extends:

Includes: All other use cases

Participators: Actual player

3.0.1 Normal flow of events

User gets to the main menu and immediately starts the game

	Actor	System
1.1	User starts the application	
1.2		Shows the main menu
1.3	User picks the option named “Start”	
1.4		Game starts

3.0.2 Alternate flow of events

User gets to the main menu and fiddles with the options, then runs the game

	Actor	System
2.1	User starts the application	
2.2		Shows the main menu
2.3	User picks the option named “Settings”	
2.4		Shows options menu

2.5	User modifies the options and presses the button labeled “Start”	
2.6		The game starts with the chosen settings

3.1 UC: 1, Movement

Summary: The user pulls the leftmost joystick and the game checks if something is in the way before moving the player.

Priority: High

Extends: RunGame

Includes:

Participators: Actual player

3.1.1 Normal flow of events

	Actor	System
1.1	User pulls the left joystick in a direction.	
1.2		Player moves in that direction

3.1.2 Alternate flow

Player gets hit, but continues moving

	Actor	System
2.1	User pulls the joystick in a direction	
2.2		Player gets hit by an enemy
2.3		Player takes damage but keeps moving in the same

		direction.

3.1.3 Exceptional flow 1

Player get hits and dies.

	Actor	System
3.1	User pulls the joystick in a direction	
3.2		Player gets hit by an enemy
3.3		Player's hp reaches 0 and dies

3.1.4 Exceptional flow 2

Player moves into a wall/object.

	Actor	System
4.1	User pulls the joystick in a direction	
4.2		Player reaches a wall/object
4.3		Player stops moving.

3.2 UC: 2, Shooting at enemy

Summary: When you pull the rightmost virtual joystick the character shoots in the same direction.

Priority: High

Extends: RunGame

Includes: 3, EnemyDies

Participators: Actual player

3.2.1 Normal flow of events

Bullet hits an enemy

	Actor	System
1.1	The user pulls the right joystick in a direction.	
1.2		A projectile spawns
1.3		Projectile hits enemy and deals damage to it
1.4		Destroy projectile

3.2.2 Alternate flow

Bullet hits an immovable object / world border

	Actor	System
2.1	The user pulls the right joystick in a direction.	
2.2		A projectile spawns
2.3		Projectile hits an object that can't take damage
2.4		Destroy projectile

3.3 UC: 3, Customization Step

Summary: the process in-between challenges wherein the player is given an opportunity to change their character.

Priority: medium

Extends: RunGame

Includes: Change/Add Item

Participators: the player

3.3.1 Normal flow of events

The user changes one or more items in their Inventory, e.g. moving an item from the Storage to the Inventory and vice verse.

	Actor	System
1.1	The user changes one or more items, via a drag-and-drop interface.	
1.2		The system changes the player’s abilities to reflect this.
1.3	The user presses the “Done”-button.	
1.4		The game proceeds

3.4 UC: 4, pause menu

Summary: When the player taps the pause button

Priority: Medium

Extends: RunGame

Includes:

Participators: Actual player

3.4.1 Normal flow of events

Pause menu opens and closes after user is done

	Actor	System
1.1	User taps the pause button	

1.2		Pauses game
1.3		Displays pause menu
1.4	User taps resume	
1.5		Hides the pause menu
1.6		Resumes the game

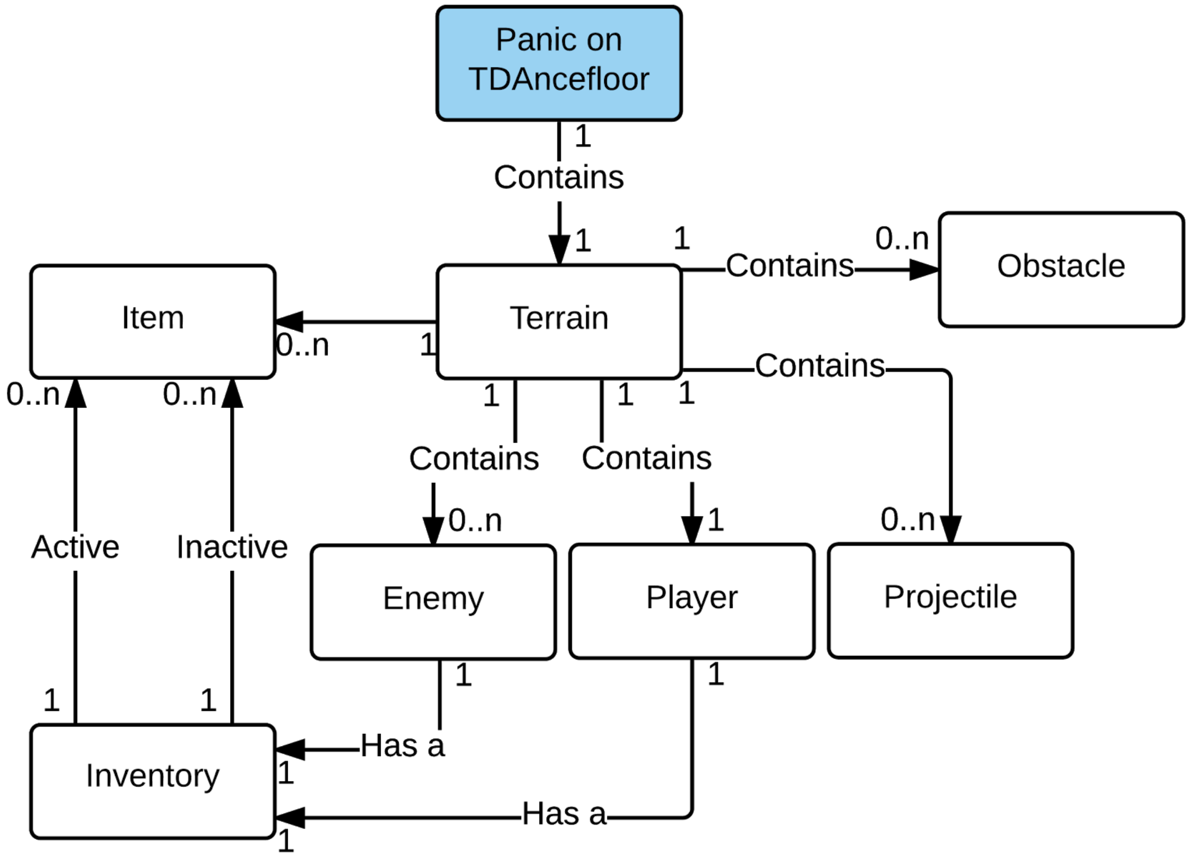
3.4.2 Alternate flow

The user changes the music volume

	Actor	System
2.1	User taps the pause button	
2.2		Pauses game
2.3		Displays pause menu
2.4	User taps Options	
2.5		Display options menu
2.6	User changes music volume	
2.7		Change music volume
2.8	User taps return	
2.9		Displays pause menu
2.10	User taps resume	

2.11		Hide the pause menu
2.12		Resume the game

4 Domain model



4.1 Class responsibilities

4.1.1 System

The System is the top-level object which delegates what should be done in the game.

4.1.2 Terrain

The Terrain is the area in which the game is played.

4.1.3 Obstacle

The Obstacle class represents different kinds of obstacles in the Terrain that neither the Player or Enemies can cross or Attack through.

4.1.4 Player

The Player is the representation of the user's character. It has an Inventory, health and can move.

4.1.5 Enemy

The Enemy class is a representation of the computer controlled characters and works in a similar fashion to the Player class.

4.1.6 Inventory

The Inventory class contains all Items that the Player or an Enemy are currently using.

4.1.7 Storage

The Storage contains all of the Player's unused items, i.e. Items in the Storage do not affect the Player's Attacks.

4.1.8 Item

The Item class represents a single item, contained in an inventory or on the playing field. Items affect the Attacks of the owner or the owner in general (extra health, faster movement speed, etc). Enemies occasionally drop the items they are carrying when destroyed. The Player can pick up Items that are lying on the field when close to them.

4.1.9 Attack

The Attack class handles the attacks used by the Player and Enemy classes. Attacks can be modified by the items found in a character's Inventory.

5 References