Requirements and Analysis Document for Panic on TDAncefloor

Version: 1.2

Date: 2017-05-11

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

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.

1.2 Definitions, acronyms and abbreviations

The user is the person playing the game.

The player is the character the user is controlling.

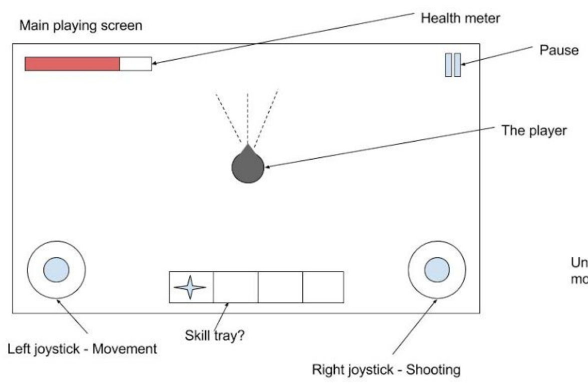
The inventory is a collection of items used to modify a character or a character’s attack(s).

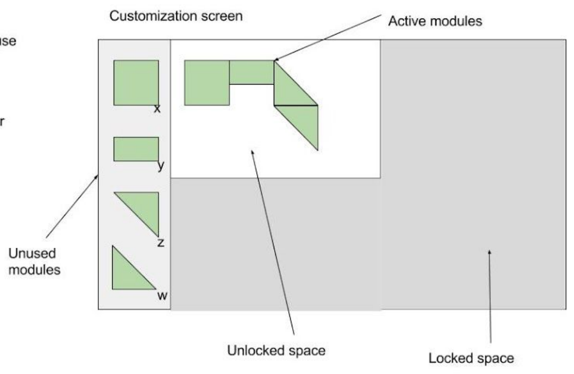
The storage is a collection of the player’s unused items.

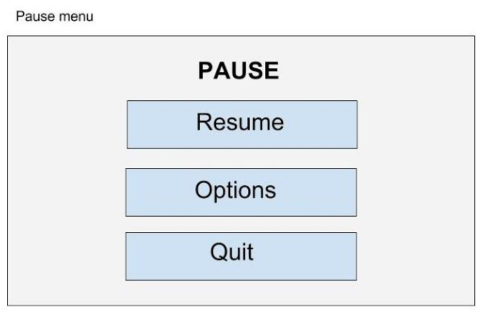
2 Requirements

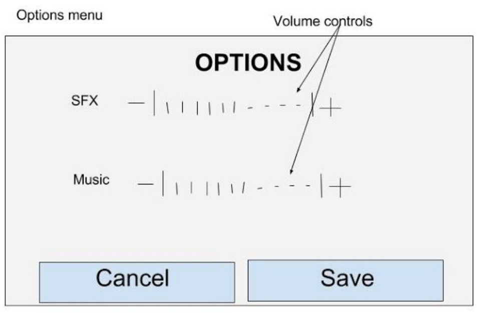
2.1 User interface

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
- All parts must be testable and have valid tests
- Should use immutable classes as much as possible
- Should run at 60 FPS

3 Use cases

3.0 UC: 0, RunGame

3.1 UC: 1, Movement

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

Priority: High

Extends: RunGame

Includes:

Participators: Actual player

3.1.1 Normal flow of events

	Actor	System
1.1	Pulls the left joystick in a direction.	
1.2		User move in that direction

3.1.2 Alternate flow

User gets hit, but continue moving

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

3.1.3 Exceptional flow 1

User get hits and dies.

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

3.1.4 Exceptional flow 2

User moves into a wall/object.

	Actor	System
4.1	Pulls the joystick in a direction	
4.2		User reaches a wall/object
4.3		User 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	Pulls the right joystick in a direction.	
1.2		A projectile spawns and a gun sound plays.
1.3		Projectile hits enemy and deals damage to it
1.4		Play hit animation

3.2.2 Alternate flow

Bullet hits an immovable object / world border

	Actor	System
2.1	Pulls the right joystick in a direction.	
2.2		A projectile spawns and a gun sound plays.
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 player 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 player 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 player presses the "Done"-button.	
1.4		The game proceeds

3.4 UC: 4, ItemCollect

Summary: When the player picks up an item

Priority: Medium

Extends: RunGame, Movement, EnemyDies

Includes: (other UCs)

Participators: Actual player

3.4.1 Normal flow of events

The player is close enough to an item to pick it up.

	Actor	System
1.1	Is close enough to an item.	

1.2		The item gets pulled to the player.
1.3		The player picks up the item.
1.4		Play ItemCollect sound.
1.5		The item gets removed from the map.
1.6		Item is added to player’s storage.
1.7		Display text depending on the type of item picked up.

3.5 UC: 5, pause menu

Summary: When the player taps the pause button

Priority: Medium

Extends: RunGame

Includes:

Participators: Actual player

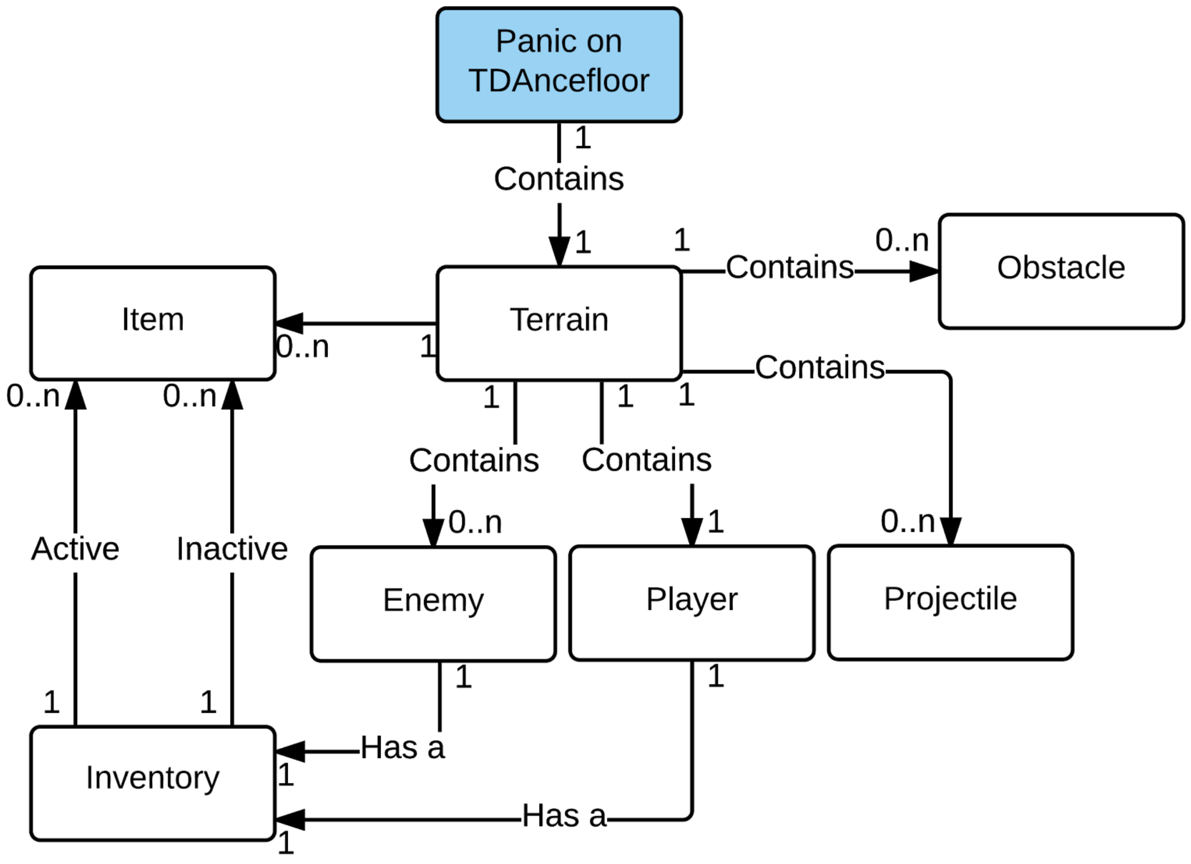
3.5.1 Normal flow of events

Pause menu opens and closes after user is done

	Actor	System
1.1	Taps the pause button	
1.2		Pauses game
1.3		Displays pause menu
1.4	Taps resume	
1.5		Hides the pause menu

1.6		Resumes 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