Requirements and Analysis Document for Panic on TDAncefloor

Version: 1

Date: 2017-03-28

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.

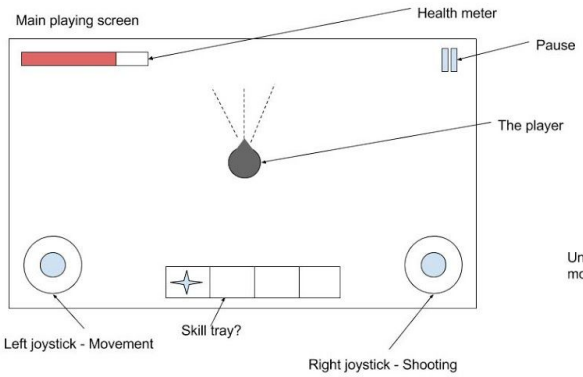
The inventory is a collection of items used to modify a character or a character's attack(s).

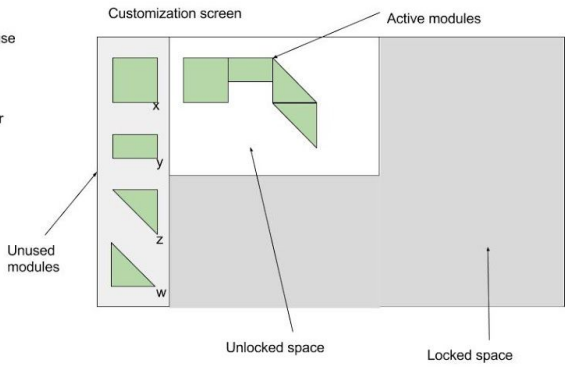
The storage is a collection of the player's unused items.

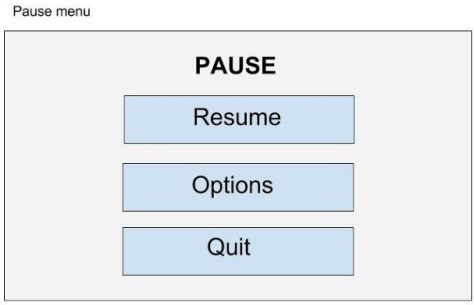
2 Requirements

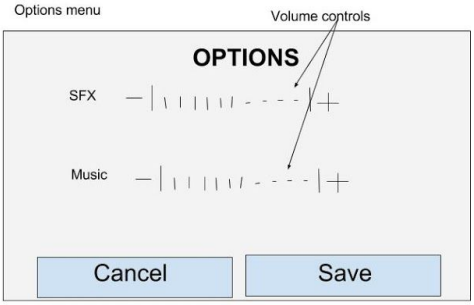
2.1 User interface

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

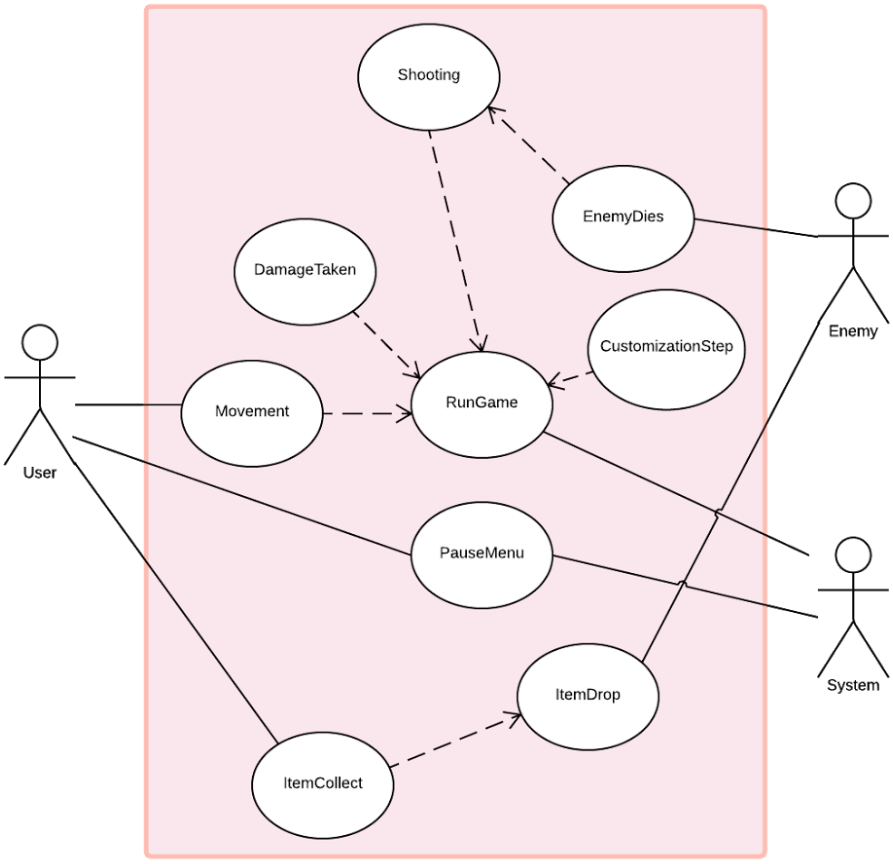
2.3 Non-functional requirements

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

Schematical overview



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: 4, DamageTaken

Participators: Actual player

3.1.1 Normal flow of events

	Actor	System
1	Pulls the left joystick in a direction.	
2		Checks if a wall is in the way.
3		Checks if an enemy is in the way.
4		Checks if an item is within pickup range.
5		The player moves in that direction

3.1.2 Alternate flow 1

Player blocked by wall

	Actor	System
2.1	Pulls the joystick in a direction	
2.2		Checks if a wall is in the way.
2.3		The player moves alongside the wall based on the angle between the joystick and the wall.

3.1.3 Alternate flow 2

Player blocked by enemy

	Actor	System

3.1	Pulls the left joystick in a direction	
3.2		Checks if a wall is in the way.
3.3		Checks if an enemy is in the way.
3.4		The player reacts according to use case 4, DamageTaken.

3.1.4 Alternate flow 3

Player close to item

	Actor	System
4.1	Pulls the left joystick in a direction	
4.2		Checks if a wall is in the way.
4.3		Checks if an enemy is in the way.
4.4		Checks if an item is within pickup range.
4.5		See use case 6, ItemCollect.

3.2 UC: 2, Shooting

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 does not hit an enemy

	Actor	System
1	Pulls the right joystick in a direction.	
2		A projectile spawns and a gun sound plays.
3		Checks if the projectile hit an enemy.
4		No target hit.

3.2.2 Alternate flow 1

Bullet hits enemy who survives

	Actor	System
2.1	Pulls the right joystick in a direction.	
2.2		A projectile spawns and a gun sound plays.
2.3		Checks if projectile hit an enemy.
2.4		The projectile hits an enemy, play hit animation.
2.5		Calculates new health of enemy.

3.2.3 Alternate flow 2

Bullet hits enemy who dies

	Actor	System
3.1	Pulls the right joystick in a direction.	
3.2		A projectile spawns and a gun sound plays.
3.3		Checks if the projectile hit an enemy.
3.4		The projectile hits an enemy, play hit animation.
3.5		Calculates new health of enemy. Health is less than or equal to 0.
3.6		Enemy health is set to 0.
3.7		See use case 3, EnemyDies.

3.3 UC: 3, enemy dies

Summary: When an enemy's HP is less or equal to zero

Priority: High

Extends: RunGame, Shoot

Includes: (other UCs)

Participators: An enemy

3.3.1 Normal flow of events

Enemy dies. There are still other enemies remaining.

	Actor	System

1		Play enemyDies animation.
2		Check if item drops (See UC 8, Item Drops)
4		Remove enemy from game.
5		Check if any enemies are alive.
6		Nothing happens.

3.3.2 Alternate flow 1

Enemy dies. Last enemy in the round

	Actor	System
1		Play enemyDies animation.
2		Check if item drops (See UC 8, Item Drops)
4		Remove enemy from game.
5		Check if any enemies are alive.
6		All enemies are dead.
7		All remaining items not picked up are pulled to the player.
8		See use case 6, ItemCollect
9		Shows text “Wave cleared” on screen.
10		Changes screen to Character customization.

3.4 UC: 4, DamageTaken

Summary: Give audio-visual feedback when the player character is damaged.

Priority: High

Extends: RunGame

Includes:

Participators: Actual player

3.4.1 Normal flow of events

	Actor	System
1	Player fails to avoid an enemy's harmful action and takes damage	
2		Subtract enemy damage from the player's health bar.
3		Check if player health is above 0.
4		Display the player's new health using the health bar.
5		Play sound effect.
6		Temporary color change.

3.4.2 Alternate Flow 1

The player is hit by a powerful attack

	Actor	System
2.1	Player fails to avoid an enemy's harmful action and takes damage	
2.2		Subtract enemy damage from the player's health bar.

2.3		Check if player health is above 0.
2.4		Display the player's new health using the health bar.
2.5		Play different sound effect.
2.6		Screen shake.
2.7		Temporary color change (lasts for a longer time).

3.4.4 Flow 3 - The player dies

	Actor	System
3.1	Player fails to avoid enemy's harmful action and takes damage.	
3.2		Subtract enemy damage from the player's health bar.
3.3		Check if player health is above 0.
3.4		Set the player's health to 0.
3.5		Display the player's new health using the health bar.
3.6		Play PlayerDies sound effect.
3.7		Screen shake.
3.8		See use case X, GameOver

3.5 UC: 5, 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

3.5.1 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	The player changes one or more items, via a drag-and-drop interface.	
2		The system changes the player's abilities to reflect this.
3	The player presses the "Done"-button.	
		The game proceeds with the next wave.

3.6 UC: 6, ItemCollect

Summary: When the player picks up an item

Priority: Medium

Extends: RunGame, Movement, EnemyDies

Includes: (other UCs)

Participators: Actual player

3.6.1 Normal flow of events

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

	Actor	System

1	Is close enough to an item.	
2		The item gets pulled to the player.
3		The player picks up the item.
4		Play ItemCollect sound.
5		The item gets removed from the map.
6		Item is added to player's storage.
7		Display text "Character item *ItemName* picked up".

3.6.2 Alternate flow

Flow 2 The player is close enough to an item to pick it up. The item is for weapon customization.

	Actor	System
2.1	Is close enough to an item	
2.2		The item gets pulled to the player
2.3		The player picks up the item
2.4		Play ItemCollect sound
2.5		The item gets removed from the map
2.6		Item is added to player's storage
2.7		Display text "Weapon item *ItemName* picked up"

3.7 UC: 7, pause menu

Summary: When the player taps the pause button

Priority: Medium

Extends: RunGame

Includes:

Participators: Actual player

3.7.1 Normal flow of events

Flow 1 Open pause menu and close it again, no changes

	Actor	System
1	Taps the pause button	
2		Pauses game
3		Displays pause menu
4	Taps resume	
5		Hides the pause menu
6		Resumes the game

3.7.2 Alternate flow

Flow 2 Open pause menu, change volume and resume the game

	Actor	System
2.1	Taps the pause button	
2.2		Pauses game

2.3		Displays pause menu
2.4	Taps Options	
2.5		Displays the options menu
2.6	Changes a volume slider	
2.7.1	Taps Save	
2.7.1.2		Saves new values
2.7.2	Taps Cancel	
2.8		Hides Options menu
2.9	Taps resume	
2.10		Hides the pause menu
2.11		Resumes the game

3.7.3 Alternate flow

Flow 3 Open pause menu and quit the game

	Actor	System
2.1	Taps the pause button	
2.2		Pauses game
2.3		Displays pause menu
2.4	Taps Quit	

2.5		Stops the game
2.6		Displays main menu

3.8 UC: 8, Item Drops

Summary: the process where an item drop after an enemy is killed

Priority: low

Extends: RunGame, EnemyDies

Includes: Change/Add Item

Participators: An Enemy

3.8.1 Normal Flow

No item drops

	Actor	System
1		Checks drop table
2		Checks item
3		Drops nothing

3.8.2 Alternate flow

An item drops

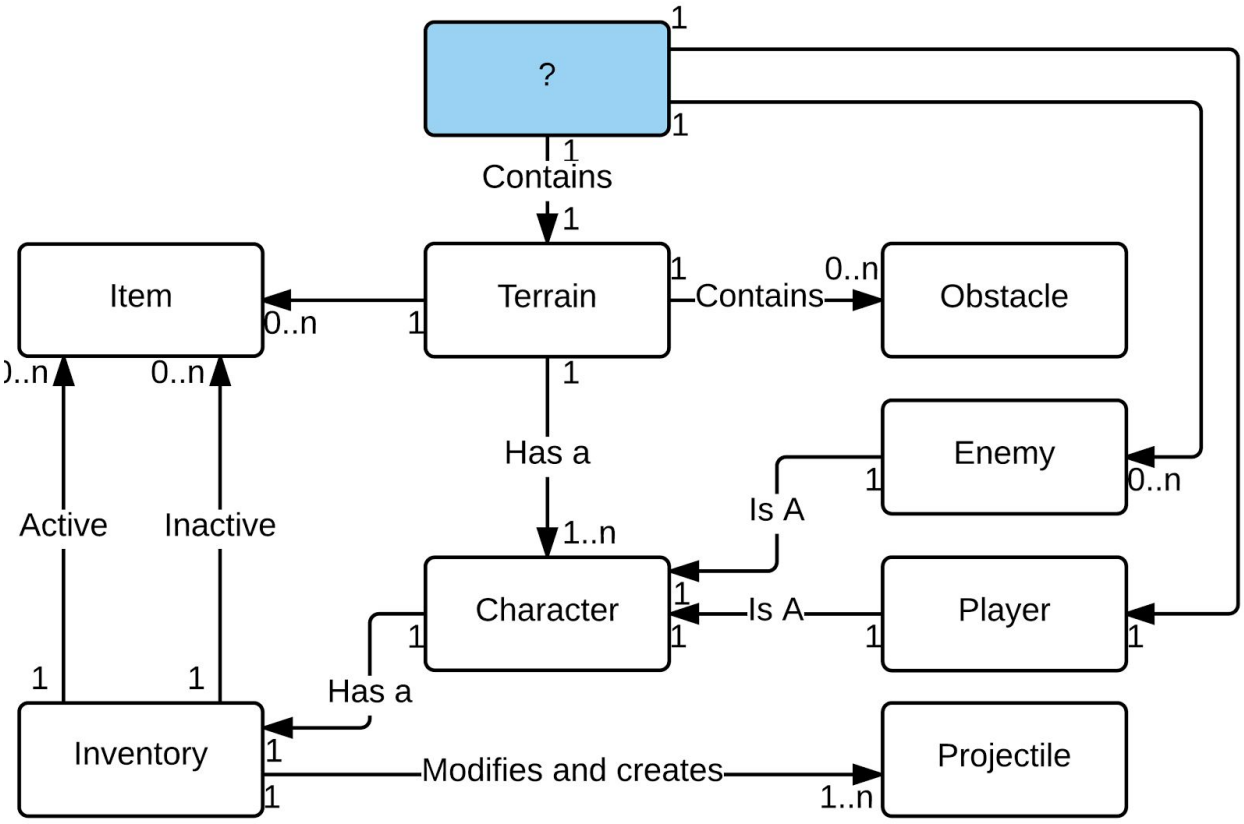
	Actor	System
2.1		Checks drop table
2.2		Checks item
2.3		Drops item
2.4		Display item on screen

3.8.3 Alternate flow

A rare and above item drops

	Actor	System
2.1		Checks drop table
2.2		Checks item
2.3		Drops item
2.4		Play sound depending on rarity
2.5		Display item on screen with different colors depending on rarity

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