Requirements and Analysis Document for get_rect()

Version: 1.0.1
Date: 29/3 2016
Author: Felix Jansson, Jesper Lindström, Samuel Håkansson, Simon Sundström

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 an entertaining 2D platform game with a story based on the life of an IT student at Chalmers.

1.2 General characteristics of application

The application will be a desktop application with no support for other platforms. The game is a standalone without network and multiplayer features. The user will play in real time and the game is built on the idea of “quests” given by NPC interaction. The player will be given rewards for completing objects such as weapons, spells, gear or item to help defeat the enemies. This could also be found throughout the game. To measure progress and gate content the game will have a level indicator in form of a modular staff built by soda cans and duct tape. This will also measure health.

1.3 Scope of application

The application is only supported for desktop where iOS, android and web are excluded. To interact are keyboard and mouse or Xbox controller supported. The game is a single player game where the user plays against the game itself without network or multiplayer features. The game is intended to be an immersive and fun experience. The game supports to play a save session over multiple sessions. At certain places the game is saved automatically.

1.4 Objectives and success criteria of the project

It will be a game containing an estimated 10 hours playtime. The game delivers a consistent story throughout the entire game. There will be great enemy and map variation.

1.5 Definitions, acronyms and abbreviations

RPG	Role Playing Game
NPC	Non-Player Character
FPS	Frames Per Second
AI	Artificial Intelligence
NA	Not Applicable
GUI	Graphical User Interface
JRE	Java Runtime Environment

2 Requirements

In this section we specify all requirements

2.1 Functional requirements

All Function

1. *Move.* A player should be able to move left and right.

2. *Interact with Objects.* A player should be able to interact with certain objects in the world.

3. *Attack.* A player should be able to attack enemies during the game

4. *Jump.* Allow player to move the character vertical upwards.

5. *Save game.* Allow player to save current game session local to play another time.

6. *Open stash.* A player should be able to open a stash to access it's arsenal.

7. *Player dies.* A player should be able to die and have the game restore to an earlier save point.

8. *Entity takes damage.* Certain entities should be able to take damage.

9. *Load game.* The game should be able to load a saved state.

10. *Switch active weapon.* A player should be able to switch between a ranged and melee weapon.

2.2 Non-functional requirements

2.2.1 Usability

The game will be fairly available for new players. There might be things that is common game logic that we will assume the player knows. We will do tests with both experienced users and inexperienced users. Language wise we will work towards one single language, Swedish. There will be an in-game display of controls and a way to teach users how to play by playing the game.

2.2.2 Reliability

The game should not crash more than once in an estimated 1 hour of playtime.

2.2.3 Performance

- 60fps
- 1080p
- Less than an average 5 seconds of loading time.
- Less than an average 0.5 seconds between user input and visual representation of the action.

2.2.4 Supportability

The program will be available for use with multiple controllers, for example Xbox controllers as well as keyboard. We are aiming to create a desktop application.
There should be automated test verifying all use cases. Code related to the GUI could be tested manually.

2.2.5 Implementation

The application will be written in Java because it is a wide language with many available libraries and plugin which will improve the end product. This will require users to have JRE installed and configured. The user will also need to download and install the game files.

2.2.6 Packaging and installation

The application will be delivered as a zip-archive containing;
1. A file for the application code (a standard Java jar-file).
2. All needed resources, internationalization and localization files, icons, etc.
3. Start programs (scripts) to start the game on the different platforms.
4. A README-file documenting installation and start of application.

2.2.7 Legal

There might be legal issues related to rights to music and trademarks. The real characters portrayed in-game will be asked for permission, before being included in the game.

2.3 Application models

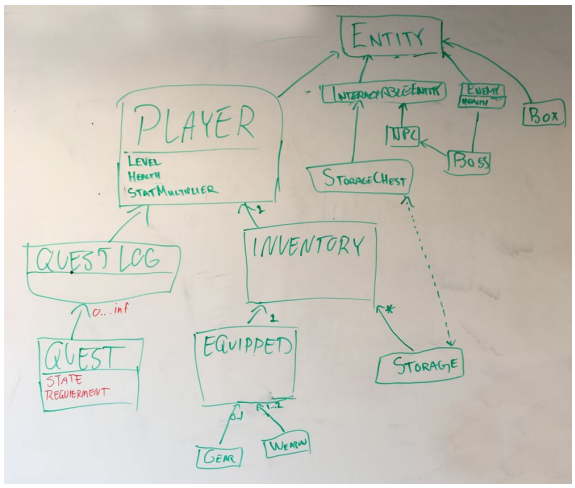
2.3.1 Use case model

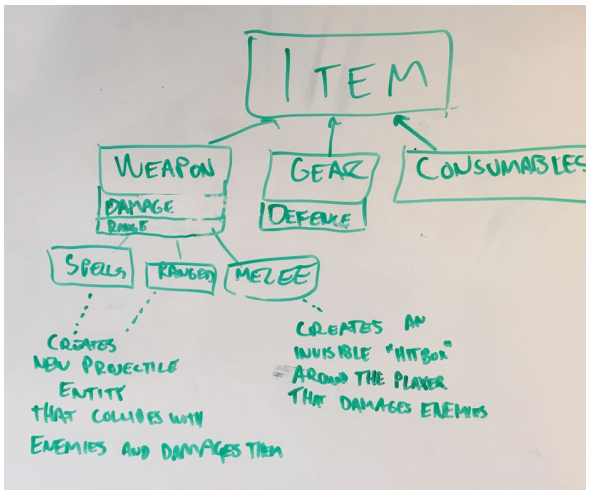
UML and a list of UC names (text for all in appendix)

2.3.2 Use cases priority

High: UC1, UC3, UC4, UC7, UC8
Medium: UC2, UC5, UC9
Low: UC6, UC10

2.3.3 Domain model





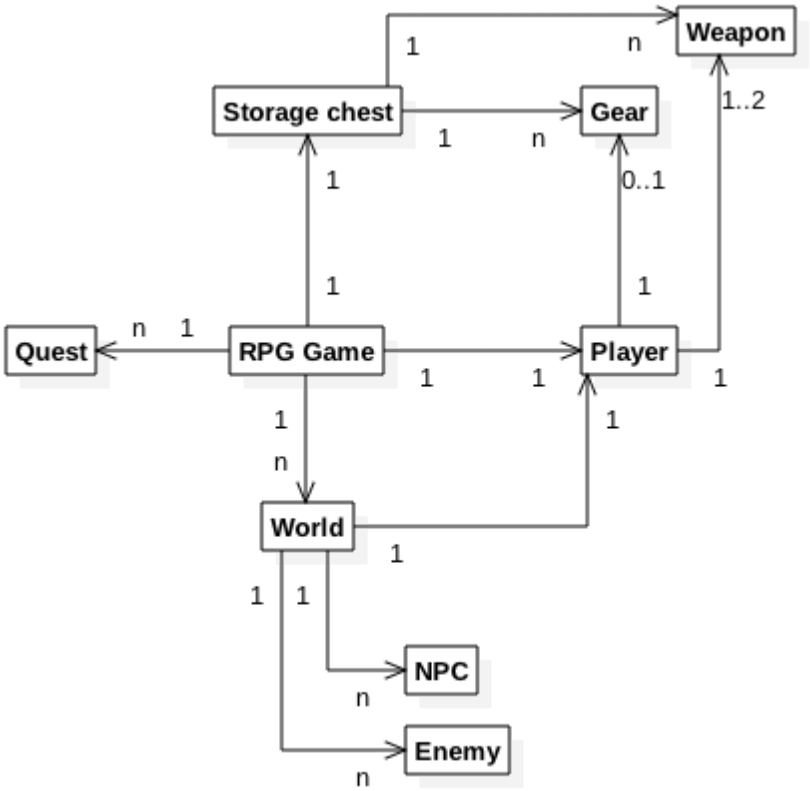
2.3.4 User interface

The application will not be themeable nor skinnable and have a minimum size of 1366x768. We will work towards the 16:9 dimension and if possible, 4:3 will be available.

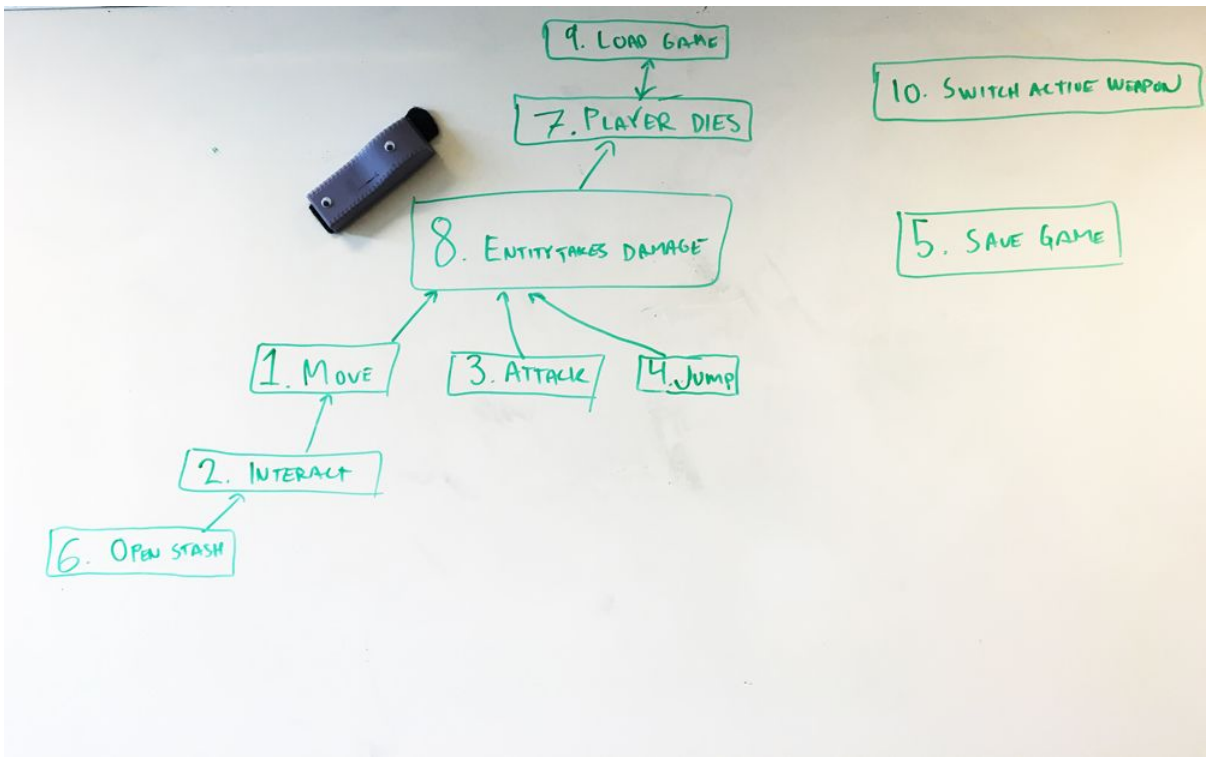
2.4 References

Appendix

Domain model



Use cases overview



Use cases

1. Use Case: Move

Summary: This is how the player moves around. This is only horizontally.

Priority: High
Extends:
Includes: Entity takes damage
Participators: Player

Normal Flow of events:

	Actor	System
1	Presses key to move in a horizontal direction.	
2		Moves the player in the direction corresponding to the input.

Alternate Flow of events:

Case 2a: Path obstructed

	Actor	System
2		An environment object is in the way and is blocking the path.
3		Dosen’t move the character.

Case 2b: Enemy in the way

	Actor	System
2		An enemy is in the way and is blocking the path.
3		Moves the player into the same tile as the enemy and damages the player.
4		Player Takes Damage *Includes UC 8, Entity takes damage*

2. Use Case: Interact

Summary: This will be your way to open doors, talk to NPC, change map, search containers, pickup items, basically interact with everything.

Priority: Medium
Extends:
Includes: Move
Participators: Player and an interactable entity.

Normal Flow of events:

	Actor	System
1	Player moves close to an interactable entity. *Includes: UC 1, Move*	
2		Entity indicates visually that it can be interacted with.
3	Player presses key to interact.	

Alternate flow of events

Case 1-3a: Player tries to interact with a non-interactable entity.

	Actor	System
1	Player moves close to a non-interactable entity. *Includes: UC 1, Move*	
2	Player presses key to interact	
3		Nothing happens.

3. Use Case: Attack

Summary: Player attacks enemies with weapons. There will be two types of weapons, melee and ranged. Melee is here seen as “normal flow”.

Priority: High
Extends:
Includes: Entity takes damage
Participators: Player and enemies

Normal Flow of events:

	Actor	System
1	Presses key to attack	
2		Visual representation of attack
		Entity (enemy) within area takes damage

Alternate Flow of events:

Case 2b: no enemy within range

	Actor	System
2		Nothing happens

4. Use Case: Jump

Summary: The players jumps.

Priority: High
Extends:
Includes: Entity takes damage
Participators: Player, Enemy

Normal Flow of events:

	Actor	System
1	Player presses key to jump.	
2		System moves the player up vertically a set height.

Alternate Flow of events:

Case 3-4a: Path obstructed

	Actor	System
3		An environment object is in the way and is blocking the path.
4		Stop the character's ascension.

Case 3-6b: Enemy in the way

	Actor	System
3		An enemy is in the way and is blocking the path.
4		Moves the player into the same tile as the enemy and damages the player.
5		Player Takes Damage
6		The Player continues with unchanged momentum.

5. Use Case: Save game

Summary: The progress will be saved at certain places.

Priority: Medium
Extends:
Includes:
Participators:

Normal Flow of events:

	Actor	System
1	Goes to save place	
2		Writes player info to file.
3		Notify player visually the game has been saved.

Alternate Flow of events:

Case 3-5a: Permissions

	Actor	System
3		The program does not have access to write on the hard drive.
4		Shows error message
5		Informs player that the save was not completed.

Case 3-5b: Out of Memory

	Actor	System
3		The system is out of memory
4		Shows error message
5		Informs player that the save was not completed.

6. Use Case: Interact with door

Summary: There is doors around the world and they will be interactable

Priority: Low
Extends: UC 2, Interact
Includes:
Participators: Player and a door.

Normal Flow of events:

	Actor	System
1	The player moves close to a door. *Includes UC 1, Move* *Replaces 1 in UC 2, Interact*	
4		The door is visually opened.
5		The player is moved to a new scene.

7. Use Case: Player dies

Summary: When the player dies the game will reload from the latest savepoint after showing a game over message.

Priority: High
Extends:
Includes: UC 8: Entity takes Damage, UC 9: Load Game
Participators: Player

Normal Flow of events:

	Actor	System
1		The player takes damage. *Includes UC 8, Entity takes damage*
2		The players health is lowered below 0
3		Displays a game over message.
4		Loads latest saved state. *Includes UC 9, Load Game*

8. Use Case: Entity takes damage

Summary: An entity is damaged by another entity.

Priority: High
Extends:
Includes:
Participators: 2 entities with damage and health.

Normal Flow of events:

	Actor	System
1		Entity is affected by a damage
2		Hit entity's health is reduced depending on damage and defence base values and multipliers.

Alternate Flow of events:

Case : Health reaches 0

	Actor	System
3		Hit entity's health is reduced below 0.
4		Entity's death animation is played.
5		Entity is removed from screen.

9. Use Case: Load game

Summary: If you have saved at some point earlier in time, you can load this “save file” and return to that state (progress, quests, items, location etc).

Priority: Medium
Extends:
Includes: UC 7: Player dies
Participators: Player

Normal Flow of events:

	Actor	System
1	Player presses “Continue” in the start menu	
2		The state at the time of the “save” is re-created: player progress, quests, location (scene), inventory.

Alternate Flow of events:

Case 1a: Player has died and is respawning

	Actor	System
1	Player dies. *Includes UC 7, Player dies.*	

Case 1b: Player wants to load a game but has no saved game.

	Actor	System
1		The “Continue” button in the start menu is hidden.

Case 2c: Load file is corrupt or gets removed (after you have loaded the game menu)

	Actor	System
2		An error message is displayed visually.

10. Use Case: Switch active weapon

Summary: You can equip two weapons at the same time, but you can only have one active at once. You can quickly switch the active weapon

Priority: Low
Extends:
Includes:
Participators: Player

Normal Flow of events:

	Actor	System
1	Player presses switch button	
2		Current weapon is switched for the secondary

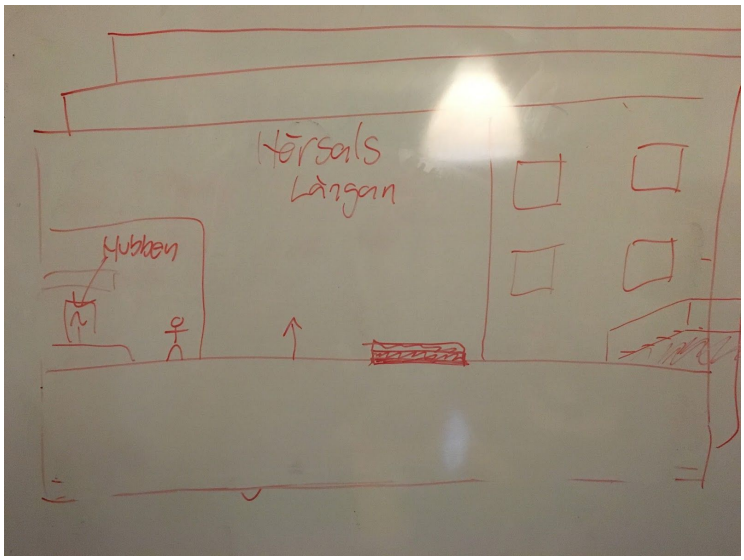
Alternate Flow of events:

Case 1-2a: Only one weapon equipped

	Actor	System
1	Player presses switch button	
2		Nothing happens. The current weapon remains active, since no other weapon is available.

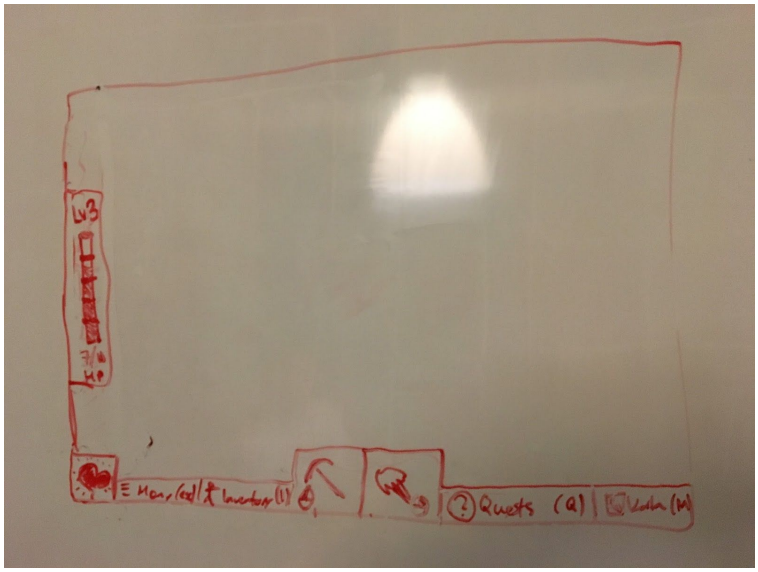
GUI

Hörsalslängan



Inventory

Game controller



Startmeny