Requirements and Analysis Document for the game project Robots Stole My Girlfriend

Table of Contents

Version: 0.2
Date: 2012-03-13
Group: 8
Authors: Daniel Jonsson, Johan Grönvall, Johannes Ulén, Victor Rådmark

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 a two dimensional action oriented platforming video game.
The game will be of original design and is designed in such a way that even less-experienced players will be able to enjoy it.

1.2 General characteristics of application

The game will be a desktop, standalone (non-networked) game with a graphical user interface for the Windows/Mac/Linux platforms.

It will be played in real-time and the actual player controls a single in-game character by using his keyboard. The character will be able to move to the left and to the right, and he will also be able to jump upon platforms. The game will have at least three levels which the player will be able to complete one after another. Only level one will be unlocked from start, and to unlock a map the player has to complete the previous one. These levels can be completed by managing to move the character to the finishing line. To make it harder for the player to do this, the levels will also contain computer controlled enemies. The player can then either try to avoid these enemies or attempt slaying them using his weapons. These weapons can be found on the ground during gameplay, and will then be stacked up and be at the player's disposal for the rest of the game.

To give the game an extra aspect and to increase the replay value of the levels, we will have a "upgrade tree" with upgrades that can be unlocked by spending upgrade points. These upgrades can give the character stronger weapons and new abilities. Upgrade points will be given when the player unlocks levels, they can be dropped by some monsters and they can be found on hidden places in the levels. Spending upgrade points is done between levels on the

level selection screen.

1.3 Scope of application

The game will only have a single player mode.
The levels will be unlocked one after another in a linear fashion.
The player will only be able to shoot straight forward.
We won't use a physics engine.
The levels won't be hardcoded. Instead we will store them as XML files in a directory in the game folder.
The player will only be able to have one saved game at a time.

1.4 Objectives and success criteria of the project

1. It should be possible to complete levels (of which there should be at least three), receive an upgrade point for doing so and then select a new level on the level selection screen.
2. The player should be able to spend upgrade points in order to advance his character.
3. There should be at least two different types of weapons in the game that the player should be able to switch between and then use to damage and kill his enemies.
4. The player should be able to lose the game by getting his character killed, either by getting hit by enemies or environmental hazards (such as spike traps).

1.5 Definitions, acronyms and abbreviations

- Upgrade tree, a set of upgrades, divided into "branches", that can be enabled by spending upgrade points. By spending points in one branch the player will be able to unlock further (more advanced) upgrades in said branch. These trees are usually called talent trees or skill trees in games that use them but in those cases they only concern just that - talents and skills, which is why we used a different term, since we also have weapon upgrades in our tree. See *pic 1* in the appendix for a better understanding of how they usually looks like in games.

2 Requirements

In this section we specify all requirements

2.1 Functional requirements

The player should be able to;

1. Start a new game.
2. Continue an earlier game.
3. Game progess (completed levels, found weapons, unlocked upgrades and upgrade points) is saved automatically.
4. Select which level he wants to play from a map overview screen.
5. Start and play the selected level.

    a. Move
    b. Jump
    c. Attack and kill enemies
    d. Reload weapon
    e. Interact with the environment
    f. Pause
    g. Pick up items (upgrade points, weapons and health boxes)
    h. Take damage and die
    i. Swap weapons
6. Unlock upgrades with his upgrade points.
7. Exit the game.

2.2 Non-functional requirements

NA (not applicable).

2.2.1 Usability

Usability is high priority. It should be easy for users to start the game and levels within a very short period.

The game will use a common control scheme to make it easy for people to play the game. The game will also give feedback to all of the player's actions, so he easily can interpret the state of the game.

…...

2.2.2 Reliability

2.2.3 Performance

2.2.4 Supportability

2.2.5 Implementation

2.2.6 Packaging and installation

2.2.7 Legal

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

A list

2.3.3 Domain model

UML, possible some text.

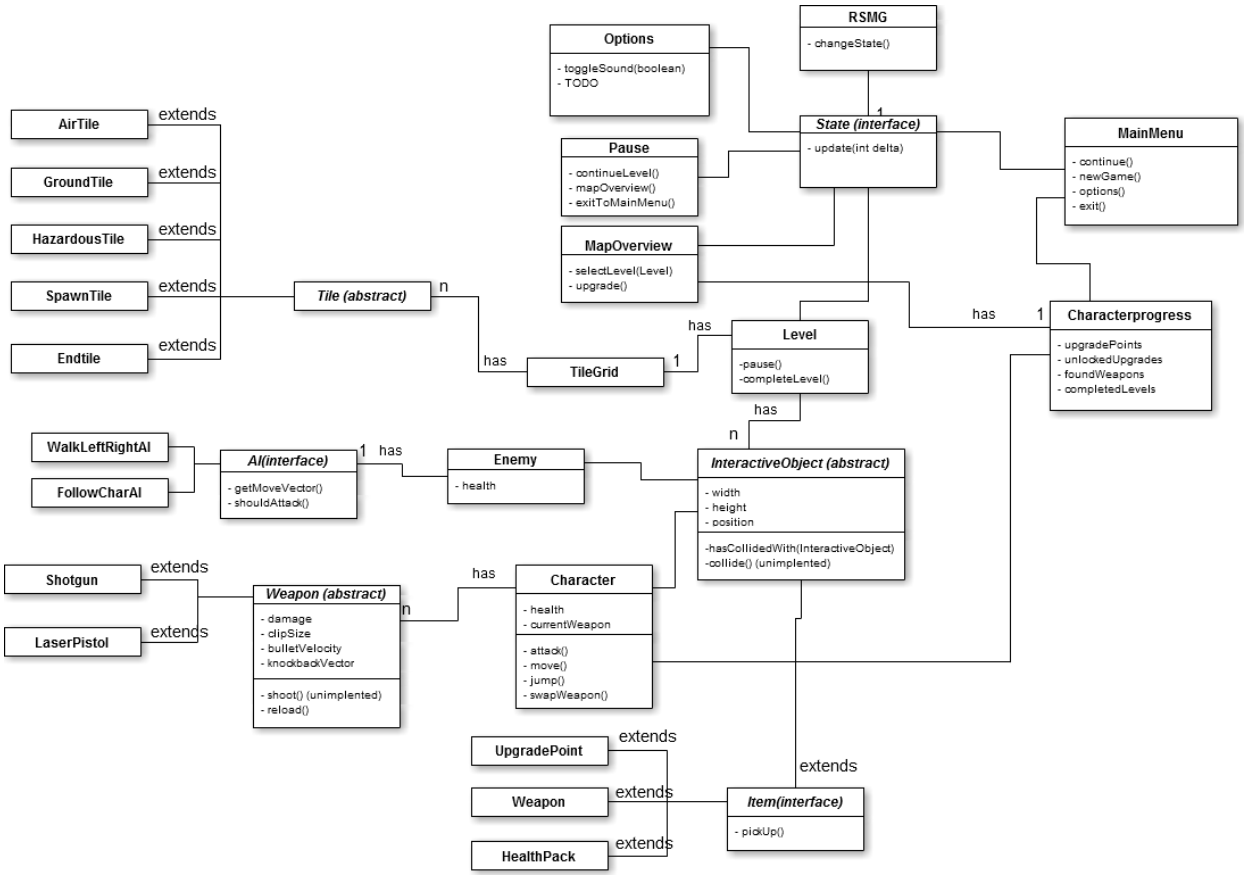
2.3.4 User interface

Text to motivate a picture.

2.4 References

APPENDIX

Pic 1. Upgrade/Skill tree from the PC game Diablo II.



Pic 2. Our domain model.

Use case texts
GUI