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

Table of Contents

1 Introduction 2
1.1 Purpose of application 2
1.2 General characteristics of application....2
1.3 Scope of application....2
1.4 Objectives and success criteria of the project....2
1.5 Definitions, acronyms and abbreviations....3
2 Requirements 3
2.1 Functional requirements....3
2.2 Non-functional requirements....3
2.2.1 Usability....4
2.2.2 Reliability....4
2.2.3 Performance....4
2.2.4 Supportability....4
2.2.5 Implementation....4
2.2.6 Packaging and installation....4
2.2.7 Legal....5
2.3 Application models....5
2.3.1 Use case models....5
2.3.2 Use cases priority....5
2.3.3 Domain model....5
2.3.4 User interface....5
2.4 References....5

Version: 0.4
Date: 2012-05-09
Group: 8
Authors: Daniel Jonsson, Johan Grönvall, Victor Rådmark, Johan Rignäs

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 it is easy to use.

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 to slay 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 an "upgrade tree" with upgrades that can be unlocked by spending upgrade points. These upgrades give the character stronger weapons and new abilities. Upgrade points will be given to the player when he 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 weapons in the game that the player should be able to switch between and use to damage and kill the enemies.
4. The player should be able to lose the game by getting his character killed, either by getting hit by enemies or by 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 level 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

NA

2.2.3 Performance

To increase the performance of the game we only draw tiles that are visible on the screen. This makes it possible for map makers to do a lot bigger maps then they otherwise would not be able to do.

2.2.4 Supportability

The implementation will be done in a support friendly manner. It will be easy to add more items, weapons and enemies to the game. And the map editor will make it easy for everyone to create a map on their own, without having to read a single line of code or having to recompile the application.

The aim is to make tests for most of the public methods in the model.

2.2.5 Implementation

The application is implemented in Java environment.

2.2.6 Packaging and installation

A stable version of the game can be found here:
[URL] (.jar)

If you rather prefer to have the latest version of the game you can get it from *github.com*:

- Clone the project with:
  git clone git@github.com:MaTachi/Robots-Stole-My-Girlfriend.git
- Open Eclipse.
- Import the project into Eclipse.
- Compile the project and run it.

How to run the Map Editor:

- The map editor can be found on the following URL: ***https://github.com/MaTachi/2D-Map-Editor***
- Create a map with the map editor.
- Put the map in Robots-Stole-My-Girlfriend/res/data/level/ and name it LevelX.xml.
- Start the game and the map should be available on the level selection screen.

2.2.7 Legal

There is some sound in the game that might be copyrighted. And some of the images used in the game are images we have found on the net and altered. This is not covered here.

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 priority use cases:

1. StartGame
2. Move
3. Jump
4. CompleteLevel
5. CollideWithEnvironment
6. Exit

See APPENDIX for complete map.

2.3.3 Domain model

UML, possible some text.

2.3.4 User interface

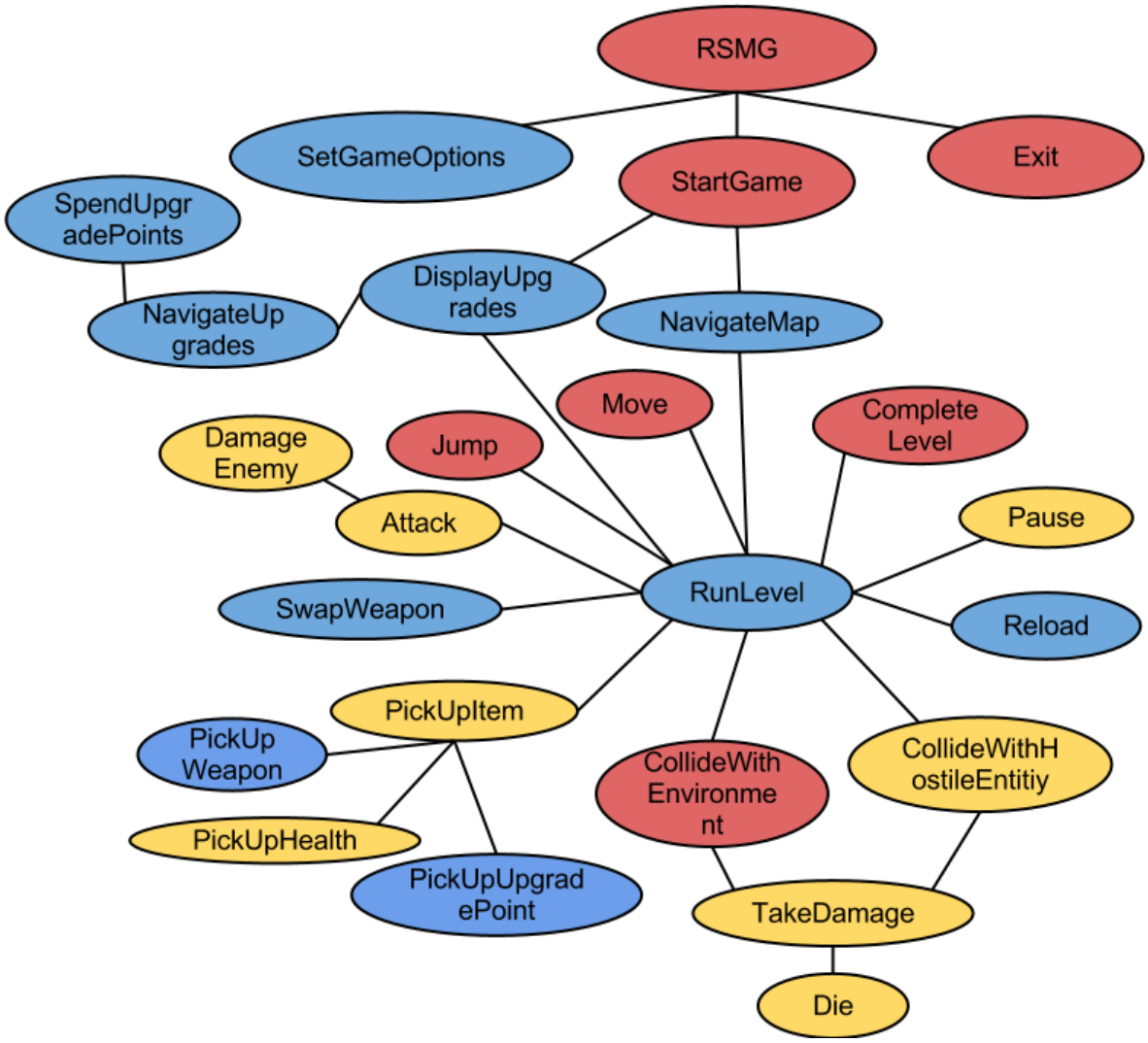
The application supports fullscreen but the default setting are 960*540. This is possible to switch between in the options menu or directly on the configuration file located in *res/data/config/config.xml*.

2.4 References

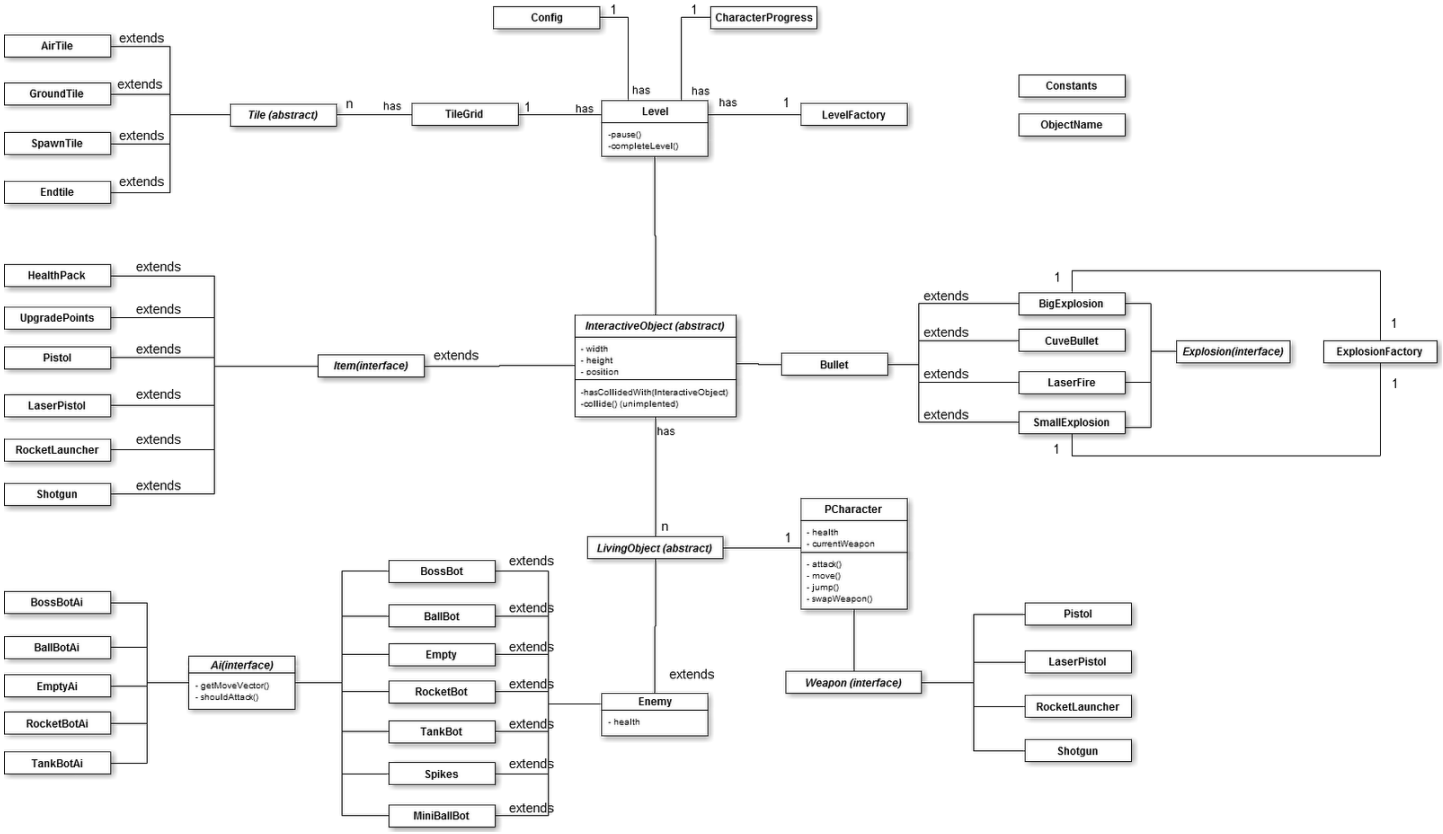
NA

APPENDIX

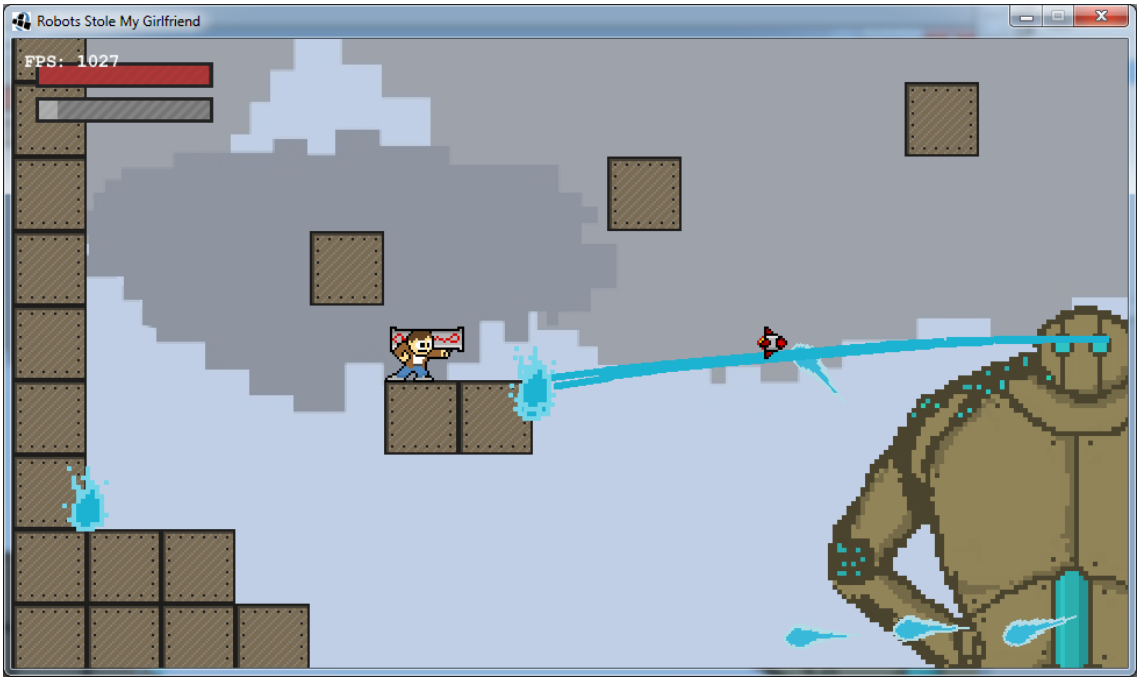
Use Cases Mindmap

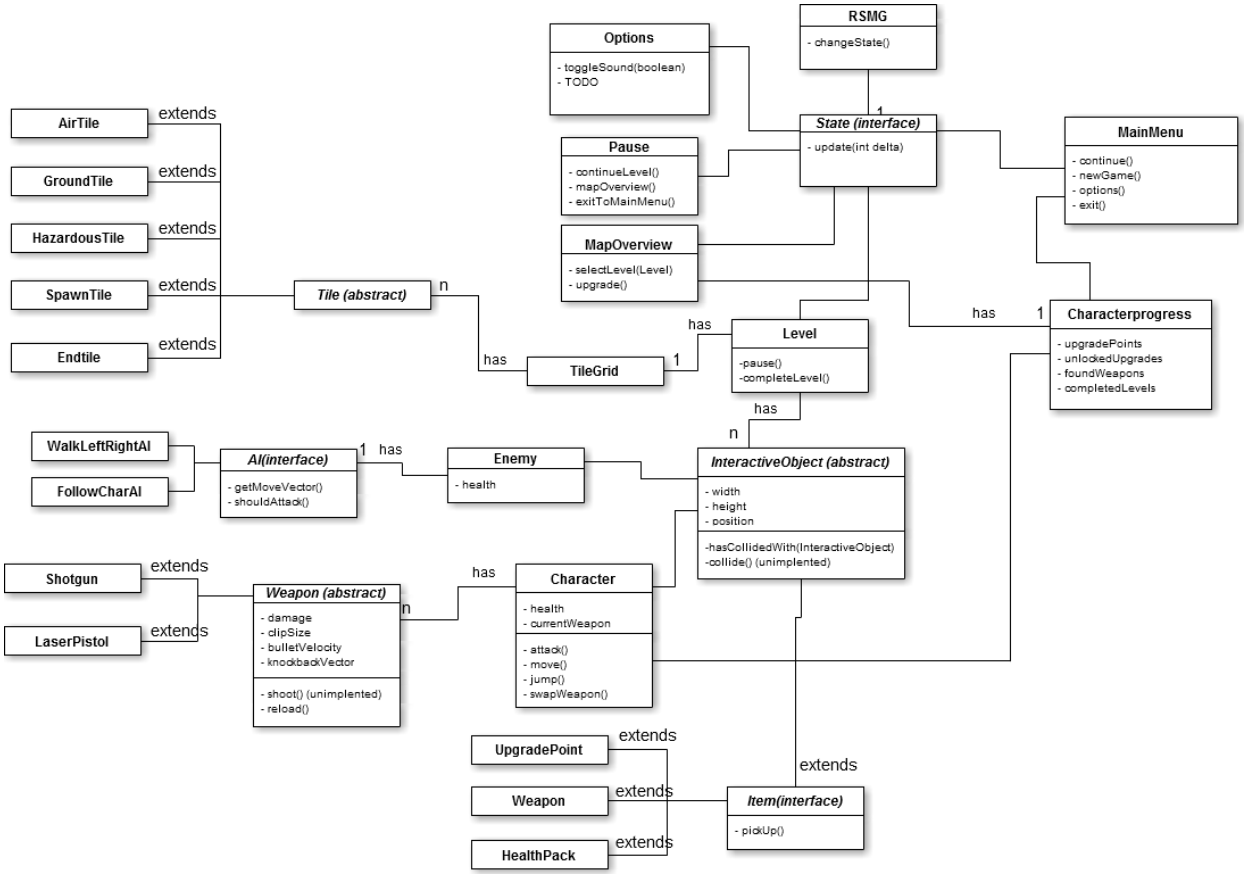


Domain model


Config
CharacterProgress
AirTile
GroundTile
SpawnTile
Endtile
extends
Tile (abstract)
TileGrid
Level
-pause()
-completeLevel()
LevelFactory
Constants
ObjectName
HealthPack
UpgradePoints
Pistol
LaserPistol
RocketLauncher
Shotgun
Item(interface)
InteractiveObject (abstract)
- width
- height
- position
-hasCollidedWith(InteractiveObject)
-collide() (unimplented)
Bullet
BigExplosion
CuveBullet
LaserFire
SmallExplosion
Explosion(interface)
ExplosionFactory
LivingObject (abstract)
PCharacter
- health
- currentWeapon
- attack()
- move()
- jump()
- swapWeapon()
BossBotAi
BallBotAi
EmptyAi
RocketBotAi
TankBotAi
Ai(interface)
- getMoveVector()
- shouldAttack()
BossBot
BallBot
Empty
RocketBot
TankBot
Spikes
MiniBallBot
Enemy
- health
Weapon (interface)
Pistol
LaserPistol
RocketLauncher
Shotgun
has
extends
1
n

GUI


Robots Stole My Girlfriend
FPS: 1027



Pic 2. Our domain model.