Warcraft TD

Requirements Description
Group 23

Authors:
Håvard Geithus
Sondre Løberg Sæter
Nicolai Meltveit,
Hallvard Andreas Eriksen
Håkon Drolsum Røkenes

COTS: Android-SDK

Quality Attributes:
Modifiability
Testability
Usability

April 8, 2011

Contents

1 Introduction

This document contains the requirements for the game named WARCRAFT TD. For the sake of brevity, it will from this point on be referred to as *the system* or *the game*. Let's give it a short introduction.

The game belongs to a well known computer game genre which is called *tower defence*. This is a subgenre of real-time strategy games where the goal is to try to stop enemies from crossing the map by building towers which shoot at them as they pass. Enemies and towers usually have varied abilities and costs. When an enemy is defeated, the player earns money or points, which are used to buy or upgrade towers. The gameplay is characterised by the positioning of static units by the player to defend against mobile enemy units who are trying to get from a start point to an end point. They try do so in a periodic fashion called *waves*. Having played numerous games from this genre one realizes that the challenge of surviving contributes alot to the fun. If it's easy, then it's boring. Also the difficulty should increase in a somewhat linear fashion as waves are killed. Because of this, there will be a need to easily change the parts of the system from which the user experience manifests itself. This justifies the choice of the first quality attribute, namely *modifiability*. Another important aspect of games are that they should behave in the same way between different playthroughs. Also the occurence of strange events due to faults will have a negative impact on gameplay. The system should therefore be easy to test to enable the development of a consistent gameplay experience. This justifies the second quality attribute; *testability*. The third and last quality attribute is *usability*. However, this will only be an implicit topic of this document.

2 Functional Requirements

Every requirement is given an ID so one can refer to it at a later point. In addition, every functional requirement is given a priority (LOW, MEDIUM or HIGH). We partition the functional requirements into gameplay requirements and graphical user interface requirements.

2.1 Gameplay Requirements

- **FR.1: Stop mobs using towers (Priority: High)**
  Mobs[1] will come streaming towards the goal, and the player should stop them before they arrive. To achieve this he must build defence towers.

[1] Plural, stands for Mobile Objects. Typically used in MMORPG games and stands for attackable non playing objects. Usually monsters.

- **FR.2: Non-overlapping waves (Priority: High)**
  The mobs come in waves. Before a wave can start, the previous wave has to be finished, that is; either killed or that the mobs have reached the goal.

- **FR.3: Next wave after time period (Priority: High)**
  When a wave is finished, the next one will arrive when the player presses the *start wave* button, or a certain time period has elapsed.

- **FR.4: Increasing mob strength (Priority: Medium)**
  The mobs become stronger every wave, and they can have different properties (e.g. armor type) and abilities (e.g. flying).

- **FR.5: Earning gold (Priority: High)**
  The player earns gold by killing mobs and completing levels.

- **FR.6: Winning the game (Priority: Medium)**
  In total there are 25 waves of mobs. If the player survives these he has completed the game.

- **FR.7: High score (Priority: Low)**
  There are different high scores availible (e.g. shortest completion time, least number of mobs passed through.

- **FR.8: Player health (Priority: High)**
  The player has 20 health poins (HP) and loses 1 HP for every mob that arrives at the goal.

- **FR.9: Damage test level (Priority: Low)**
  After these 25 ordinary levels there will be a bonus level referred to as the *damage test*, where an immensely powerful mob is let into the game. No one has ever slain it before.

- **FR.10: Upgradable towers (Priority: Low)**
  Towers can be upgraded to increase its usefulness. There are 3 upgrade levels for every tower.

2.2 Graphical user interface requirements

- **FR.11: Menu for tower building (Priority: High)**
  The tower buying menu contain all the different towers that can be built. The player presses the tower icon to bring up the menu, presses the icon of a tower (the menu then disappears automatically) then presses and releases his finger at the desired location on the map. This is where the tower is placed. The current player gold is updated. Note: If the tower is not affordable, then it's grayed out in the menu and its cost is written in red numbers.

- **FR.12: Menu for selling and upgrading towers (Priority: Low)**
  If the user presses one of his towers, a small tower action menu is shown. Here one can upgrade the tower or sell it. Note: if the tower is sold, then only a certain percentage of its purchase cost is returned to the player.

- **FR.13: Status bar (Priority: High)**
  In the upper right corner there is an overview of current HP, resources (e.g. gold), score and time.

3 Quality Requirements

Having discussed the functional requirements of the system, it is now time to look at the system from another point of view, namely by its quality requirements. As noted earlier, the important quality requirements for this system are *modifiability* and *testability*. To chatacterize these requirements it is customary to use what is called quality attribute scenarios. They demonstrate what quality means for the system. Below follows an important selection of quality attribute scenarios given in tabular form.

3.1 Modifiability

3.1.1 Modifiability requirements

- QR.1: Easy to change tower properties.
- QR.2: Adding/removing a tower to/from the game should be relatively easy.
- QR.3: Easy change enemy properties.
- QR.4: Adding/removing waves to/from the game should be relatively easy.
- QR.5: Simple, easy and fast to make new maps.

3.1.2 Modifiability scenarios

MS.1:	Changing tower properties
Portion of Scenario	**Possible Values**
Source	Developer
Stimulus	Wants to change some properties of a tower in the game (typically to tweak its behavior to improve gameplay or graphics).
Artifact	Code
Environment	At design time
Response	Changes are made, and game still works
Response Measure	Done within one hour

Portion of Scenario	**Possible Values**
MS.2:	Adding new tower
Source	Developer
Stimulus	Wants to add a completely new tower (its attributes and graphics are given for all three upgrade levels).
Artifact	Code
Environment	At design time
Response	Changes are made, and game still works
Response Measure	Done within four hours

Portion of Scenario	**Possible Values**
MS.3:	Adding map
Source	Developer
Stimulus	Wants to add a map to the game (using the Tiled editor).
Artifact	Code
Environment	At design time
Response	The map is created using the Tiled Editor and it can be easily imported into the game
Response Measure	Done within two hours (only complex maps should take two hours)

3.2 Testability

3.2.1 Testability requirements

- QR.6: Easy to access component state (internal monitoring).
- QR.7: Log relevant information.

3.2.2 Testability scenarios

Portion of Scenario	**Possible Values**
TS.1:	Testing status bar
Source	System tester
Stimulus	Wants to test the status bar
Artifact	The code
Environment	At design time
Response	Gets gold from killing enemies. Spends gold when purchasing towers or upgrades.
Response Measure	The gold earned or spent should be correct (in accordance with the predefined value)

Portion of Scenario	**Possible Values**
TS.2:	Starting game
Source	Player
Stimulus	Wants to start a new game from the main menu
Artifact	The game
Environment	At testing time. The application itself is running.
Response	The first level starts.
Response Measure	Should not take more than 3 seconds.

Portion of Scenario	**Possible Values**
TS.3:	Entering high score list
Source	Player
Stimulus	Wants to get onto high score table
Artifact	The game
Environment	At testing time
Response	Get through the whole game
Response Measure	Get onto high score table (if good enough)

Portion of Scenario	**Possible Values**
TS.4:	Building towers
Source	Player
Stimulus	Wants to build a tower
Artifact	The game
Environment	At testing time
Response	Tower is shown in the game
Response Measure	Tower is functional (true/false)

4 COTS (Commercial off-the-shelf)

When making applications for mobile clients running Andriod, there are certain constraints that one have to be aware of. These constraints are both the technical specifications of the clients and the fact that Android is, for us, a new platform. The specifications of the different clients running Android may also vary alot. Therefore we have to make it work over a range of models. There are for instance a limited amount of RAM availible. The HTC Hero, which many bought, has only 288MB RAM. The screen resolution is also a constraint we have to think about, since the Android phones have a screen resolution of about 320x480 [1]. The newer phones have better specifications, but one shouldn't make the application only for the last generation of devices.

Since the platform is new, there is alot to read up on. One can only learn so much, given limitations in time. We might also encounter constraints due to the sheep library that we are going to use when making this game.

5 Issues

No issues as of now.

6 Changes

Fixed misspelling on front page. Defined IDs for all requirements and scenarios. Assigned priorities to every functional requirement.

References

[1] HTC. Htc hero specification, 2011. [Online; accessed 8-April-2011].