TDT4120
Software Architecture

Super Bros Requirements Document

Group X2

Milos Zlatkovic
Dimitry Kongevold
Emil Grunt
Nicolay Thafvelin
Julius Buset Asplin
Håvard Kindem

COTS: XNA

Primary quality attribute:
Modifiability
Secondary quality attribute:
Usability

Contents

1 Introduction

This document is a description of the product delivered by Team X2 in the course TDT4240 Software Architecture. This document describes in short what kind of game we are going to create, and some of its features. The document contains the requirements and constraints of the system, and what kind of quality attributes we are focusing on. In part of the functional requirements we will describe a function of our application. Functional requirements are supported by quality requirement. In part of quality requirements we will describe non-functional requirements like Modifiability, Performance, Portability, Testability and Usability. The goal of our project is to be completed all high priorities functional requirements. At the end of document, there will be list of references, issues and changes for this document.

1.1 About the game

Our game is much like the popular Nintendo game Super Smash Brothers, with some tweaks from other 2D fighting games like "Street Fighter", "Tekken" and "Capcom VS Marvel". The game is a 2D fighting game where you select a character and a map and play. The player can move freely at the selected game map. If the player is hit by an object or an attack the players damage/percentage is increased, which makes it easier to hit the player out of the map. If the player falls outside a given area of the map, the player dies. If the player has more lives left he is put back onto the course with his damage/percentage reset. The goal of the game is to hit the other players of the map as many times as possible. The game ends after a given time or if all the players except one has died a given number of times. The winner is the person that has the best kill, death ratio. The game is mainly played whit xbox 360 controllers, from 2-4 players. You can play the game with the keyboard if you're only two players.

Functional requirements 2

Priorities of the functional requirements distributed from Low to High. High means that functional requirements very important for the project. Project misses the point without that functional requirements with high priorities. Low means that functional requirements donŠt important for project. If we have a time we will do that functional requirements. Medium means that the functional requirements are important for the project but not like the high priorities.

2.1 Starting a game

Summary: Before you can start to play the players have to choose their character amongst the given characters, and then choose a map.

Rationale: The players navigate in the menu with the left stick, selecting with the A button. The players can select characters from an character selection menu, and a map from a map selection menu. When successful selecting character and map, the game should start.

2.2 Game-play

Summary: It should be possible to attack other character with your character, and move around. Target is to hit other players of the map.

Rationale: A player should be able to move the character around and hit other players characters which should result in the other characters percent/damage increasing. Higher percent/damage should results that attacks at the character applies more force in a direction, so it's easier to hit the player out of the map. The target of the game is to hit other players out of the map, and not get hit out yourself. The game is finished when the time is out or the all the players except one has used up a given number of lives. The winner is the person that whit the best kill death ratio in the time mode, or the the last one standing i life mode.

2.3 Physics requirement

Summary: The characters should be able to move and jump around at the map, whit realistic physics. Should also be possible to throw objects, attack other players (hit, shoot, special attacks), if it hits another player it should affect the player in realistic way.

Rationale: When the left stick on the controller is used it should control the character in the same direction as the stick is moved. If stick is moved upwards the character should jump, if down the player should crouch. Pressing the A or B buttons should cause the character to do an attack. You can combine A and B with the stick in a direction to do a different attack. A gravity will always pull the characters and objects down to the ground.

2.4 Character

Summary: The game should support different characters with different attributes and look.

Rationale: Each character should have a different look and and different attributes.
A character should have this attributes

- Body mass
- Strength
- Speed
- Special moves

It should be easy to add more characters without needing to recompile the game.

2.5 Map

Summary: The game should support different maps, with different architecture and look.

Rationale: Each map should have a different architecture and look. This means that the map has different platform around on the stage that the characters can stand on, the platforms should match the image in the background. E.g. if the image has a rooftop, it should be a platform her that the character can stand on. It should be easy to add more maps without needing to recompile the game.

2.6 Graphics requirement

Summary: The game is viewed in 2D from the side. In a game-play it should show the selected characters and map.

Rationale: The game should represent the *characters* with an animated sprite in 2D.
The game should represent the *map* with an static sprite in 2D with collision boxes for the platforms on the map. The game should represent the *objects* with an animated sprite in 2D.

2.7 Sound FX and Music

Summary: The game should have sound FX and music

Rational: Music should play in the background while playing.
Sound FX should be heard on:

- Character hit
- Character special-move
- Character picks up power-ups
- Character uses power-up
- Object is thrown and hits ground
- Menu navigation

2.8 Map objects and power-ups

Summary: Objects and power-ups should show up at the map. The character should be able to pick up objects and power-ups.

Rationale: It should show up objects and power-ups at random location on the map at random time throughout the game. The player should be able to pick up objects and use it in the way the object is intended to be used. The character can also throw the object at other players, which will cause damage/percentage. Power-ups can also be picked up by the character. The power-up should change some of the attributes to the character.

3 Quality requirements

Our games quality requirements

3.1 Modifiability

- Add new characters, maps, objects and power-ups without recompilation
- Functional components should be specified in file
- Major functional components should be replaceable
- Play via LAN without changing lots of code

3.1.1 Add new characters, maps, objects and power-ups without recompilation

- **Source**:Game developer, user
- **Stimulus**:Add more characters, maps, objects and power-ups to the game
- **Artifact**:Game system
- **Enviroment**:During development or after releaser
- **Response**:It should be possible to add more characters, maps, objects and power-ups without recompiling the game, Characters, maps, objects and power-ups should be read from XML
- **Response Measure**:The functionality should be added in 1-2 hours.

3.1.2 Functional components should be specified in file

- **Source**:Game developer
- **Stimulus**:Designs the game
- **Artifact**:Components
- **Enviroment**:During development
- **Response**:It should be possible to change which modules are being used without change code
- **Response Measure**:Read which models/components to use from file

3.1.3 Major functional components should be replaceable

- **Source**:Game developer
- **Stimulus**:Design
- **Artifact**:Game modules and components
- **Enviroment**:Before compilation/development
- **Response**:It should be possible to add/remove modules/components without changing other classes
- **Response Measure**:Classes still work if you disable modules/components

3.1.4 Play via LAN without changing lots of code

- **Source**:Game developer
- **Stimulus**:Play game via LAN
- **Artifact**:Game system
- **Enviroment**:During development
- **Response**:It should be possible to add more characters, maps, objects and power-ups without recompiling the game, Characters, maps, objects and power-ups should be read from XML
- **Response Measure**:The functionality should be added in 2-3 hours.

3.2 Performance

- The game should run at the highest frame rate the computer can manage
- The user should not experience delay in image, or slow response
- The game should not load

3.2.1 Frame rate

- **Source**:User
- **Stimulus**:Game play
- **Artifact**:Graphics interface
- **Enviroment**:Runtime
- **Response**:User sees animation smoothly
- **Response Measure**:The game should run at a high enough frame rate so the user doesn't feel like the pictures lag

3.2.2 Game delay

- **Source**:User
- **Stimulus**:Game play
- **Artifact**:Graphics interface
- **Enviroment**:Runtime
- **Response**:User should see the responses from the game immediately
- **Response Measure**:The game should respond to the users interference within a few milliseconds

3.2.3 Game loading

- **Source**:User
- **Stimulus**:Game play
- **Artifact**:Game logic
- **Enviroment**:Runtime
- **Response**:User should not need to wait for a long time for the map to be loaded
- **Response Measure**:Maps shouldn't take more than 5 sec to load

3.3 Portability

Support pc and xbox 360 platform

- **Source**:Developer
- **Stimulus**:Support both pc and xbox 360
- **Artifact**:Implementation
- **Enviroment**:Design
- **Response**:Should be possible to compile the game for both pc and xbox 360. The game works on both platforms
- **Response Measure**:The functionality should be added in 2-3 hours

3.4 Testability

- Beta testing
- To make the game testable, the system should have the possibility to control each component's internal state and inputs. And observe its outputs.
- Large functional components should be replaceable with mock implementations

3.4.1 Beta testing

- **Source**:Test users
- **Stimulus**:Beta launch of game
- **Artifact**:Game
- **Enviroment**:When game is in beta state
- **Response**:Full walkthrough of game with error capturing
- **Response Measure**:Test game

3.4.2 Verifying code

- **Source**:Developer
- **Stimulus**:Architecture
- **Artifact**:Code
- **Enviroment**:During creation
- **Response**:Code should be created in the correct way and tested
- **Response Measure**:Apply test-driven development to satisfy this requirement

3.5 Usability

- Easy to start-up game
- Easy to use character in game

3.5.1 Easy to start-up game

- **Source**:User
- **Stimulus**:Game start and game play
- **Artifact**:Game logic
- **Enviroment**:Initialization and runtime
- **Response**:The user should have no problems start the game, and starting a match at a map
- **Response Measure**:Should be possible to just click on the game .exe file to start the game, and easy to navigate menus to start a match

3.5.2 Easy to use character in a game

- **Source**:User
- **Stimulus**:Game play
- **Artifact**:Game logic
- **Enviroment**:Runtime
- **Response**:Should be easy for a new user to navigate on map and perform attack on other characters
- **Response Measure**:Easy controls for character, so you learn the controls fast

Testability: Easy to test basic features of the game, like hitting results in character flying and getting higher percentage.

Usability: Easy to start a game. Easy character movement Easy fighting

4 COTS

Components and Technical Constraints

4.1 XNA 4 Framework

XNA is an API which is designed specifically towards making games. XNA is the successor for Managed Direct X, which was a thin .NET wrapper for the DirectX. It differs in having a lot of built in utility classes that perform routine tasks in games such as drawing and sound playback, and infrastructure that addresses common problems such as device management and resource handling. The XNA Framework is also notable for allowing end users to write games that run on retail Xbox 360 and Zune hardware, a privilege traditionally reserved for officially approved developers with specialized development kits. The tailoring for game development has made XNA a highly efficient platform for prototyping and development, but it also introduces several constraints on the software developed with it. The game focus makes it unsuitable for use in non-game hardware accellerated applications. It is not available for hosting as a control in a GUI form, and is heavily tied to the basic architecture Microsoft has designed for XNA games. Hosting XNA inside a Winforms application requires substantial work. XNA provides an easy interface to the mouse, keyboard and gamepads. However, only Microsoft's own Xbox 360 gamepads are supported, although DirectX itself supports a much larger array of gamepads. Furthermore, the XNA Framework provides a facility known as the Content Pipeline, an integrated system for processing and loading game data during compile and runtime. Using it imposes constraints on how resource handling and management is performed within the application. The pipeline also has limited support for graphic and audio formats. There are also subtle platform differences, only a small subset of the .NET Framework is available on the Xbox 360 and some aspects of the CLR (like arithmetic performance and automated garbage collection) work differently from the Windows implementation. When supporting both platforms, these aspects need to be taken into account.

4.2 SWF2XNA Framwork

Swf2XNA allows the use of Flash assets, animation and layout in XNA games. Box2D shapes and joints are supported visually using the Box2d.XNA port. There is a C# library that resembles the Flash library in AS3 (Sprites, addChild, gotoAndStop, etc). Swf2XNA also includes a game framework based on XNA, allowing easy usage of XNA elements not normally found in Flash, such as shaders, particles, and regular 3D objects.

5 References

This is the references we used to complete this document

5.1 Books

1 "Software Architecture in Practice, Second Edition", Len Bass, Paul Clements, Rick Kazman, Addision-Wesley, 2003, ISBN 0-321-15495-9

2 "Game Architecture and Design - A New Edition", Andrew Rollings and Dave Morris

5.2 Webpages

3 https://github.com/debreuil/Swf2XNA

4 http://en.wikipedia.org/wiki/MicrosoftXNA

5 http://www.microsoft.com/NET/

6 Issues

- With modifiability in mind, the XNA framework only runs on Windowsbased computers, XBOX 360 and Windows Phone 7 (limited API).
- Restricted to using C#
- Restricted to using .wav files for audio
- Restricted to using common image-formats

7 Changes

02/03/2012
-Original document

27/03/2012
-Added Issues
- Added priorities for functional requirements
- Changed quality requirements
- References fixed

29/04/2012
Delivery version:
- Clarification of some sentences and spell checking