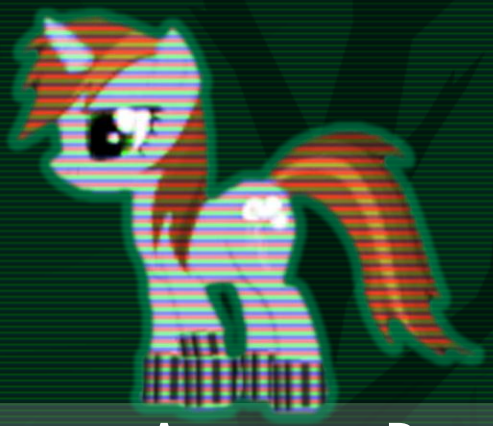
5/20/2012



Group 19

Requirements and Analysis Document for Fallout Equestria

Version: 2.0 | Lukas Kurtyan , Pontus Pall, Gustav Alm Rosenblad, Joakim Johansson

Requirements and Analysis Document for Fallout Equestria

Version: 2.0

Date: 5/20/2012

Authors:

Lukas Kurtyan
Pontus Pall
Gustav Alm Rosenblad
Joakim Johansson

This version overrides all previous versions.

Table of Contents

1 Introduction

"War, war never changes... ponies do".

1.1 Purpose of application

Create an awesome post-apocalyptic multiplayer action role playing pony game. The project aims to create a computer game based on the novel *Fallout: Equestria*[1]. The game itself is furthermore loosely inspired by pen and paper rpg's that have been based on the same novel.

1.2 General characteristics of application

The application will be a desktop, multi-player application with a graphical user interface for the Windows platform.

The game will be in ***2.5D***. The application will be a real time networked multiplayer game. The characters (ponies) in the game will be customizable. Gameplay mostly consists of running around in a party and trying to survive in the hostile environment. The particular action style is a mix of the popular games *The Realm of the mad god*[2] and *Magicka*[3]. The game ends if all party members die.

1.3 Scope of application

The game supports both Singleplayer and Multiplayer. Since friendship is magic, the single player version is just as hard as the multiplayer one. The world in use is the world of the host hosting the multiplayer session. Players should also be able to chat with each other.

1.4 Objectives and success criteria of the project

1. The game should have nice looking 2D-graphics.
2. Primitive AI should be supported in the game.
3. The game should be able to play in multiplayer mode. Chatting through text should work. This is on different computers so networking will have to be implemented.
4. Before the actual game starts, every player should be able to customize their own character. This includes things like colors, tails and hairstyles.

1.5 Definitions, acronyms and abbreviations

In alphabetical order:

- 2.5D – Even though the game basically is 2-dimensional, characters move in a third dimension when jumping and have 3-dimentional collision boxes.
- Archetype – A blue print for a type of entity.
- Assets – Things that are loaded from hard drive, e.g. sound, textures, maps etc.
- Behavior – Defines how an entity behaves.

[1] http://www.equestriadaily.com/2011/04/story-fallout-equestria.html
[2] http://www.realmofthemadgod.com/
[3] http://en.wikipedia.org/wiki/Magicka

- Client – A computer connected to a host.
- Component – A class defining a property of an entity.
- Entity – A public key to a database containing components.
- GUI, graphical user interface
- Host - The computer, to which the other computers are connected. Acts as a server. The place where logic is done, and the game is run.
- Java, programming language
- Look and feel – A XML-file and a large texture containing all the normal backgrounds, highlights, etc. that make up the GUI.
- LWJGL – A graphics library for java enabling access to Open GL.
- Screen – A layer shown on the computer display.

2 Requirements

2.1 Functional requirement

The player should be able to:

- Play alone against computer controlled opponents.
- Play with friends in a network (against computer controlled opponents).
- Create a custom pony character.
- Explore the map.
- Walk, jump, shoot.
- Host a game.
- Join a hosted game

For additional information, please see appendix A, product backlog.

2.2 Non-functional requirements

2.2.1 Usability

No tutorial should be needed in order to play the game since standard keys are used to navigate etc. Players familiar with this kind of game should know what to do and how to do it immediately.

2.2.2 Reliability

The reliability of the networked game is as reliable as the network itself.

2.2.3 Performance

The game should be able to maintain at least 30 frames per second on a moderately new computer.

2.2.4 Supportability

No components in the project depend on each other. Everything not part of the game graphics, the entity framework, most of the entity systems, etc., can be directly transferred into any other project. If you want to change how the game does graphics that would be harder since it’s highly

dependent on the LWJGL and the graphics libary. As long as one uses LWJGL though, the project is highly extendable.

Dynamic content (entities) are extendable through components, behaviors, and entity systems. For the dynamic part of the design, see entity framework.

The GUI can be extended by adding GUI-controls or combining those already existing. E.g. a spinner can be created by using two buttons and a text label. It is furthermore very easy to change the overall look and feel of the GUI by loading a different look and feel or writing custom GUI renderers.

2.2.5 Implementation

Java is required in order to play the game. Window users with a graphics card supporting Open GL 3.3 are supported.

2.2.6 Packaging and installation

N/A

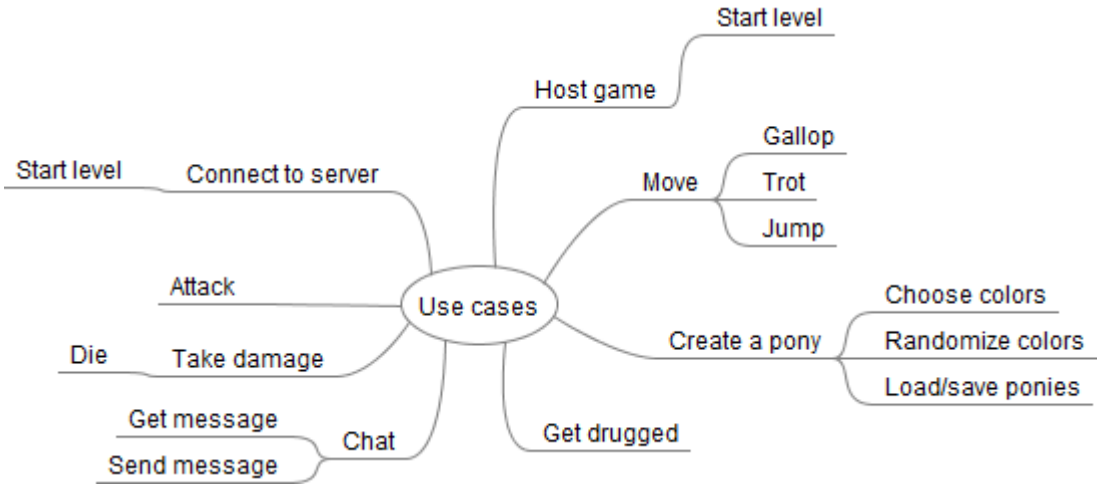
2.2.7 Legal

This game contains a lot of copyright infringements. E.g. Fallout belongs to Bethesda Softworks and My Little Pony belongs to Hasbro.

2.3 Application models

2.3.1 Use case model

Individual use cases to be found in appendix

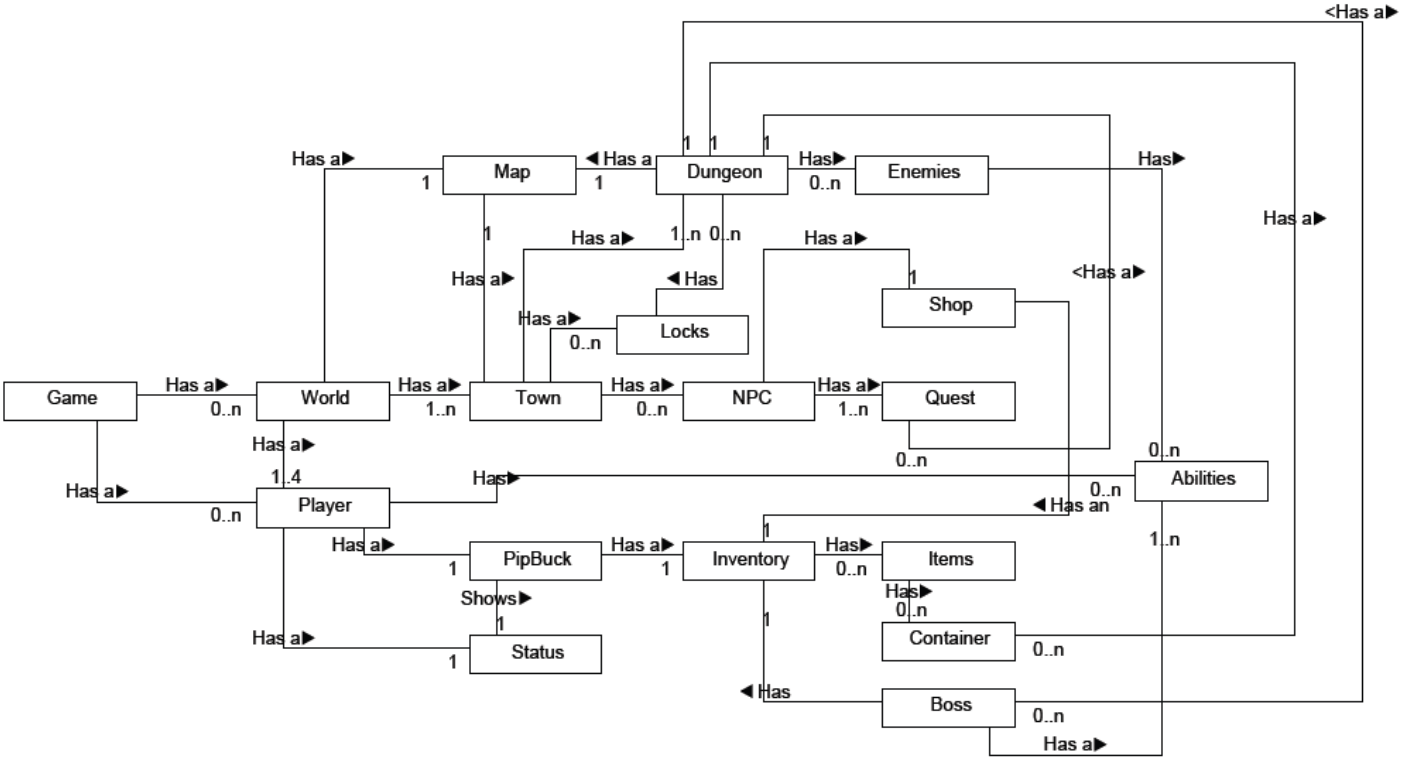


2.3.2 Use cases priority

Please see appendix A, product backlog.

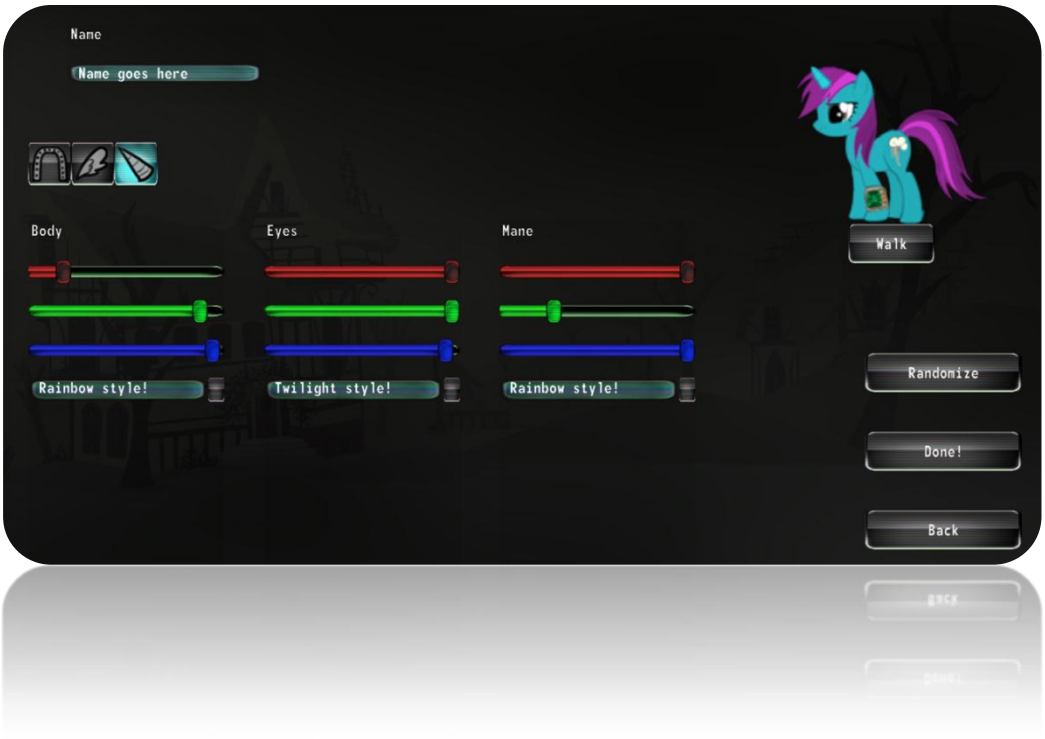
2.3.3 Domain model

Below is our original domain model. However, it is no longer applicable to our project since our project is now rathe aimed on developing a game engine than filling the game with content. It is still included in order to show what kind of functionality our engine could provide if extended.



2.3.4 User interface

This application uses a GUI written completely by us, as a part of the project. All the controls are written from scratch and are easily customizable with a custom look and feel.



3.0 References

Referenced sources can be found as footnotes throughout the document.

APPENDIX

Use case texts:

Chat
Short description: Write in the chat for other players in party to see.
Priority: Medium
Extends or includes: -
Participating actors: Player and other players in party.

Flow of events:

Actor	System
Writes text in the input field of the chat window in GUI.	
	Result is shown for all party members in chat window in GUI.

Connect to server
Short description: Connect to a server in order to play multiplayer.
Priority: High
Extends or includes: -
Participating actors: Player.

Flow of events:

Actor	System
Chooses multiplayer mode in menu	
	Shows a screen giving the actor two choices; to host a game or to join a game hosted by another player.
Chooses to join a game.	
	Shows a list of running servers on the network
Chooses a server to connect to and presses “Connect to server”	
	Shows the lobby screen with all the connected players

Host a game
Short description: Host a game in order to play multiplayer.
Priority: High
Extends or includes: -
Participating actors: Player.

Flow of events:

Actor	System
Chooses multiplayer	

mode in menu	
	Shows a screen giving the actor two choices; to host a game or to join a game hosted by another player.
Chooses to host a game	
	Shows the lobby screen where the rest of the players will appear as they connect.

Move
Short description: Trot (i.e. walk) around the map.
Priority: High
Extends or includes: -
Participating actors: Player.

Flow of events:

Actor	System
Presses one of the movement keys (W/A/S/D)	
	Changes the animation from *idle* to *walk* and changes the position according to the input.

Jump
Short description: The player jumps into the air.
Priority: Medium
Extends or includes: -
Participating actors: Player.

Flow of events:

Actor	System
Presses the jump key (spacebar)	
	Changes the animation from *idle/walk* to *jump* and sets the velocity upwards (along the z-axis).

Shoot
Short description: The player shoots a bullet.
Priority: Medium
Extends or includes: -
Participating actors: Player.

Flow of events:

Actor	System
Presses the left mouse button	
	Spawns a bullet entity in front of the player with a velocity heading away from the player.

Create a random pony
Short description: Creating a pony character with randomized characteristics.
Priority: Medium
Extends or includes: -
Participating actors: Player.

Flow of events:

Actor	System
From the main menu, the actor chooses “My little ponies”	
	Shows a screen with previously created ponies, as well as buttons for selecting, deleting, and creating new ponies.
Clicks the “Create a new pony”-button	
	Shows a screen with controls for editing pony characteristics, a preview of the pony, and buttons for saving and randomizing the pony.
Clicks the “Randomize-button”	
	Updates the preview and controls with a random pony.
Clicks the “Done”-button	
	The characteristics of the pony is saved in the XML-format.

Get delirious
Short description: The character in the game gets delirious.
Priority: Requirement
Extends or includes: Includes StatusChange,
Participating actors: Player or Enemy.

Flow of events:

Actor	System
Drinking alcohol or getting poisoned by enemy.	
	Screen becomes blurry, “S” turns to “shh” and “...hick” gets randomly inserted in the chat.