Requirements and Analysis Document for Visualista (RAD)

Table of Contents

1 Introduction
- 1.1 Purpose of application
- 1.2 General characteristics of application
- 1.3 Scope of application
- 1.4 Objectives and success criteria of the project
- 1.5 Definitions, acronyms and abbreviations

2 Requirements
- 2.1 Functional requirements
- 2.2 Non-functional requirements

Version: 0.3

Date: 2014-03-27

Author: Erik Risfelt & Markus Bergland

This version overrides all previous versions.

1 Introduction

This section gives a brief overview of the project.

1.1 Purpose of application

The Visualista project aims to create a toolbox for a single artist, and give that user virtual tools to create a Visual Novel. For definition of a Visual Novel, see definitions, acronyms and abbreviations.

The application also aims to supply a graphical and logical interface from which a pre-created Visual Novel can be played by a single player.

1.2 General characteristics of application

The Application will be a desktop, stand alone (non-networked), single player application that will run on Windows, Mac and Linux platforms which support Java.

The application will supply tools to create a visual novel, this includes the ability to create and delete novels, create and delete scenes as well as create and delete actors. A user may only edit one scene at a time, and the application will only be able to load one Visual Novel at a time. All graphical content of the Visual Novel must be provided by the user. All actions within the Visual Novel must be defined by the user. There will be a very limited (>15) amount of preset actors for the user to use within their Visual Novel. The application will support save and load functions for a Visual Novel. The application will support a graphical interface (GUI) to use with the tools. There will be mouse driven as well as keyboard controlled parts of the application.

The application will support a graphical interface to run a Visual Novel. There will be logical interpretations of actions defined in a Visual Novel. While playing a Visual Novel, any dynamic modifications of a scene will be temporarily saved.

1.3 Scope of application

The application will not support any online features.

The application will not support logical clauses (“if-clauses”) for actions defined by the user. The application will not support dynamic values in actions, with the exception of using an actor’s current position within a grid. The application will not use timed events, and will not be able to queue actions with a delay in between.

The application will not support saving the actions taken during a the playing of a visual novel.

See Possible Future directions for a widened scope.

1.4 Objectives and success criteria of the project

1. The user can create a Visual Novel, and within this create and define scenes, actors and actions.
2. The user can save an edited Visual Novel, and later open it within the application, either to play it or to edit it.
3. The user can play a pre-created Visual Novel, and within in it have expected results when clicking actors with defined actions.

1.5 Definitions, acronyms and abbreviations

For GUI definitions, please refer to the GUI sketch further below.

The definition of Visual Novel may vary depending on the source, but for the ease of the project the following definition is provided and to be used within the project.

- Visual Novel - A graphical interpretations of a story defined by a user, with limited user interaction and in most cases support for changing the story clientside. To be compared with a single player game without a goal, hence not meeting the real criteria for being defined as a game.

- GUI - Graphical user interface
- Java - Platform independent programming language
- JRE - Java Runtime Environment. The additional software required to run Java applications.

- Novel - A gathering of multiple scenes. Could be compared with a project within other applications. A novel is the highest member in the hierarchy of a Visual Novel and contains the Metadata for all other members of the hierarchy.
- Scene - A scene contain a grid, a text field as well as a background image. A novel contains at least one scene, and a scene's grid is at least 1x1 large, but is not required to have a defined image or text. When a Visual Novel is played, one and only one scene can be active at a time, and is what is shown to the user.
- Grid - A grid is a *n* x *n* graphical interpretations of tiles, and is contained by a scene. A grid is always square, and need to be at least 1x1. Without a grid, a scene cannot exist. A grid holds no data on what its tiles contains, and have no actions bound to it.
- Tile - A tile is a single square with a defined size, currently set as 32 x 32 pixels, and contains an actor. Tiles may not under any circumstances lack an actor, and the definition "empty tile" is in fact a tile holding what is defined as the Shell Actor. A tile is responsible for pass on when it is clicked by the user, and to hold the data of its actor.

- Actor - An actor is a graphical representation of an object, and carries a number of actions. An actor is defined as placed once it is represented in a tile, and is otherwise uncallable. Actors are not responsible for supplying the data to the director, but are responsible for handing the actions to a tile upon a request.
- Action - An action is contained within the actor, defined by the user creating or editing and can do one of the four following things:
    - Clear a tile. This is defined as setting the current actor in that tile to the Shell Actor.
    - Set the actor of a tile at a specified location to a predefined actor. The location may only be specified as an exact value, or the value of the tile containing the actor containing this action.
    - Update the text area of a predefined scene. This may be updated to an empty string, effectively clearing it. To see the definition of the text area, see the GUI sketch.
    - Change the current scene to another scene. This dynamically saves the current state of the current scene before changing.
- Director - A director is responsible to carry out actions upon request, and to be a link between data for scenes and tiles. This is required due to the possibility for actions to refer to tiles in another scene, and even the text area of those scenes. A director does not have a graphical interpretation, and is what is commonly referred to as a controller in programming definitions.

2 Requirements

In this section we specify all requirements

2.1 Functional requirements

Create a list of high level functions here (from the use cases).
Create, save, open, rename Novels
Create, save, open (multiple), rename scenes belonging to a novel.
Also assocciate a text and a backdrop to a scene.
Create actors with an image, an action sequence and a name assocciated with them.
And then place them in a grid within a scene.
Lastly to clear a tile and to play a novel previously created.

2.2 Non-functional requirements

Possible NA (not applicable).

2.2.1 Usability

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

GUI

Domain model

Use case texts