Project Hangman

Dustin Payne
PayneGames

7/2/2019

Logo

Contents

1 Revision History

Date	Version	Description	Author
7/2/19	A.1	Early “alpha” iteration	Dustin Payne

2 General Information

Project Summary	
Project Name	Project ID
Hangman Project	dp222gr.hangman
Project Manager	Main Client / Audience
Dustin Payne	Students (age 18-30 typically)
Key Stakeholders	
Project manager, developer, graphical asset designer, end-user, test users	
Executive Summary	
This project will result in a nice looking, easy to play version of the game Hangman for the purpose of demonstrating ability to execute a software development strategy.	

3 Vision

The vision of this project is to create a version of the popular game “Hangman.” It will be developed in Java and utilize JavaFX to create a GUI. The game will generate a phrase from a premade list of phrases that belong to different categories. The player will select letters to try to fill in the blanks of the phrase. If the player selects a letter that is not in the phrase, a piece of a hanging stickman will be added to a representation of a stickman being hanged. The objective of the player is to complete the whole phrase before the stickman is completed.

The GUI will have a menu on top where the player can select options for the game and 26 buttons on the bottom that represent each letter of the English alphabet. The middle of the screen will display the phrase with guessed letters shown and unguessed letters represented by underscores. The player will be able to select from various categories that the game will generate a phrase from. There are only two planned categories so far: movies and tv series. More categories will hopefully be added as development continues.

My game goes well beyond the scope of what is required and will offer a few features that most others’ projects will not. In particular, my game will offer a complete graphical experience with no outdated, hard to use console implementation. My game will also feature sounds and animations. Simply put, this will be a real game.

Vision Reflection: It is a bit hard to make the game hangman sound exciting but compared to the minimum game (i.e. a console-based game), my features are pretty exciting. The template stated the vision should aid in a shared understanding between all project participants of what the game should be like, so I described how I wanted the game to work. This makes it read a bit dull because the game hangman is not a very exciting one. Hopefully I can reach the goals set for features because I don’t know how to animate yet.

4
Project Plan

The development of the game will start with building the framework, namely: creating letter and phrase objects with their relevant methods and building the game window that gives the ability to set a phrase, guess letters, and reset the game. The next step will be to develop graphics to represent the number of guesses left and add categories. After testing, the game is basically done but the maintenance stage will enable PayneGames to add additional categories.

4.1 Introduction

Simply put, this is the game Hangman.

4.2 Justification

This game will be made to allow players to use a well-designed, easy-to-use GUI to play Hangman using phrases taken from a pool of popular and/or classic TV shows and movies.

4.3 Stakeholders

1. Project manager – wants to make a clear plan with attainable goals so that time can be managed
2. Developer – wants to write good code that is easy to maintain and update with future features.
3. Graphical asset designer – wants to make the game nice to look at and easy to understand
4. Test users – anyone willing to play the game as it's being developed. They can offer useful feedback on what changes to make or features to add.
5. End user – every step of this project should enhance the gameplay experience for the end user. Everything should be done with them in mind.

4.4 Resources

1. 8 weeks of time to plan, develop, test, and polish the game
2. Eclipse IDE

3. GIMP to make some graphics using resources from gameart2d.com
4. RStudio to retrieve a list of popular and well-known movies from INDB's .tsv files that they make publicly available.
5. One acer laptop for writing code

4.5 Hard- and Software Requirements

The hardware requirements are minimal. Any computer with java installed should be able to run it.

4.6 Overall Project Schedule

The first iteration will be delivered before 12:00 8/2/2019. The second will be delivered during week 8 of 2019. The third will be delivered during week 10 of 2019. The final release will be delivered during week 12 of 2019.

4.7 Scope, Constraints and Assumptions

The scope of this project is narrowed strictly to developing the game and delivering it for grading. No marketing or maintenance will be done, nor will it be released for sale. The only constraint is time. Since this project will produce no expenses or income, there are no financial constraints. This project is being produced on the assumption that the player has a mouse.

Iterations

Plan for four iterations, including this. This is a fine-grained plan on what is to be done in each iteration and with what resources. To begin with, this is a plan of what we *expect* to do, update this part with *additions* (never remove anything) when plans do not match up with reality. Also make time estimates for the different parts.

In this course the overall planning has in some ways already been decided, so use the template to provide more details on specific tasks that define *your* project. Remember that you can plan to add features to any of the phases as long as the main focus is also met.

The first assignment is to complete iteration one.

5.1 Iteration 1

1) Write project documentation		est. time:
a) Write general information and vision:		35 minutes
b) Write project plan:		45 minutes
c) Write iterations:		60 minutes
d) Write risk analysis:		20 minutes
e) Write reflection:		10 minutes
	Total:	170 minutes
2) Develop game		est. time:
a) Write code for phrase.java and letter.java		30 minutes
b) Write code for GUI		120 minutes
c) Retrieve IMDB data and build categories		30 minutes
	Total:	180 minutes
	Iteration total:	350 minutes

5.2 Iteration 2

1) Make UML diagrams		est. time:
a) Make class diagram:		30 minutes
b) Write use case scenarios:		60 minutes
c) Make use case diagram:		60 minutes
d) Make state diagram:		90 minutes
	Total:	240 minutes
2) Alter code		est. time:
a) Fix any dependencies revealed in 1.		180 minutes
	Total:	180 minutes
	Iteration total:	420 minutes

5.3 Iteration 3

1) Write testing documentation		est. time:
a) Write manual tests:		60 minutes

b) Write automated tests:		60 minutes
c) Format tests into document:		120 minutes
	Total:	240 minutes
	Iteration total:	240 minutes

5.4 Iteration 4

1) Polish the game		est. time:
a) Develop graphical assets:		180 minutes
b) Add sounds:		120 minutes
c) Format buttons:		30 minutes
	Total:	330 minutes

2) Complete documentation		est. time:
a) Polish documentation:		90 minutes
b) Update tests:		60 minutes
	Total:	150 minutes
	Iteration total:	480 minutes

Risk Analysis

6.1 List of risks

The risks are minimal, but the main concern is missing a deadline. While this is unlikely, it would not be surprising if it happened. As a consequence, the whole project would be delayed to the retake deadlines. Another risk is getting sick. If I get sick it could reduce the amount of effective time I have due to loss of productivity

6.2 Strategies

The strategy being implemented (or was planned on being implemented) is to complete tasks at least a few days ahead of schedule. I cannot prevent illness, but I can mitigate the risk by trying to stay healthy.