Making IT
good for society

BCS Digital Industries Apprenticeship

Software Development Technician
Project B - Maze Game

Version 1.2

November 2018

Change History

Any changes made to the project shall be clearly documented with a change history log. This shall include the latest version number, date of the amendment and changes made. The purpose is to identify quickly what changes have been made.

Version Number and Date	Changes Made
V1.0 04/06/18	Document created.
V1.1 02/07/18	Submission email address amended
Version 1.2 November 2018	Declaration template removed. To be supplied in a separate document.

Project Overview and Objectives

You work for Olde Worlde Phunne, a computer gaming company. They want to increase the number of visitors to their website by offering a free maze game, which potential customers can get access to by signing up to their website. Your manager would like you to design, build, and test an initial version of this component – the "Maze Game". This initial version may either itself be a website or a stand-alone program to be downloaded from the website.

An apprentice will need to:

1. **Review all the key information and create a design for the maze game;**
2. **Construct the maze game in accordance with the design;**
3. **Test that the maze game meets its requirements;**
4. **Document what you built.**

Project Outputs and Deliverables

Once completed, to demonstrate completion of the tasks you are asked to provide a series of outputs that should be submitted together with the synoptic project declaration.

Deliverable	Output	Evidence
Design	Create documentation to describe what the maze game will do and how it will work. This is likely to include: • Any assumptions made about the requirements or changes made to the requirements; • Sketches of the user interface; • Brief explanations describing what each element of the user interface does; • A specification of the format of data used to configure the game.	Word or PDF documents or similar
Construction	Write a program that implements the maze game. • Your program should be logically structured. • It should follow good coding practices.	Files containing program code
Test	Create and execute a set of tests that demonstrate that the program meets its requirements. • The tests may be manual or automated. • The tests may be written before, after, or at the same time as the program code. • For each test you should document its expected outcome and the actual result.	Any suitable format e.g. textual documents, spreadsheets, program code.
Document	Document the results of your work. • Discuss any limitations of your design and/or implementation. • Propose future improvements. • Create a user guide.	Word or PDF document or similar. A video might be suitable for a user guide.

Project Information and Equipment

To complete this project, you will need to review all the information in the bullet list below. The information can be found in the Appendix and will enable you to deliver the key outputs and deliverables for this project as detailed in the table above.

- Background information.
- Conceptual models.
- Use cases.

In addition, you will be provided with access to a virtual platform or alternatively if a virtual platform is not available, your training provider and or employer will provide you with all resources required to complete your project including:

- computer equipment with access to the Internet;
- an appropriate software development environment;
- suitable document preparation software.

Apprenticeship Competencies Covered

Competency Standard
Logic: writes simple code for discrete software components following an appropriate logical approach to agreed standards (whether for web, mobile or desktop applications)
Data: makes simple connections between code and defined data sources as specified.
Test: functionally tests that the deliverables for that component have been met or not.
Analysis: follows basic analysis models such as use cases and process maps.
Quality: follows organisational and industry good coding practices (including those for naming, commenting etc.).
Communication: clearly articulates the role and function of software components to a variety of stakeholders (including end users, supervisors etc.).
User Interface: develops user interfaces as appropriate to the organisations development standards and the type of component being developed.

Appendix – Background Information and Business Requirements

The following additional information has been provided to help you with the completion of the project.

Background information

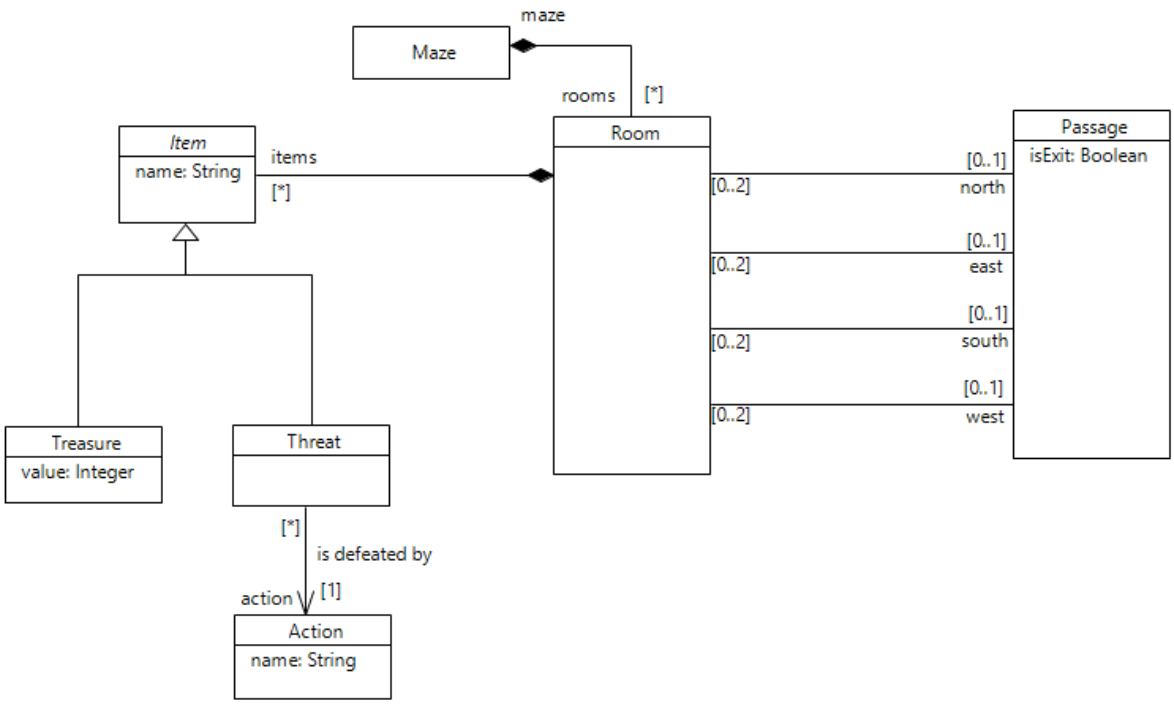
- Olde Worlde Phunne develops computer games for a variety of platforms, specialising in games that hark back to the past.
- The company wishes to increase traffic to its website by offering a simple maze game free of charge. Your task is to design, built and test the first version of this game.
- The game may be a separate website accessible from the company's main website, or a downloadable executable to run on any execution environment of your choice, on a desktop computer or a mobile computing device.
- The user interface may have any style that you like, including command-line, point-and-click, and any combination of these and other techniques.
- When the game is started, the first action must be to initialise the maze's structure of rooms, passages and items from a configuration, represented by data in a text file.
    - You will need to define one or more such configuration files.
    - Your program must be able to check that a configuration file is valid, and report if not, with a statement of what is wrong.
- Once the game is configured, the program should deposit the player at a random room in the game; then play proceeds according to the user's actions specified below.
- The purpose of the game is to finish the maze with as much wealth as possible.
- The game should eventually be accessible only to registered customers, but you are not required at this stage to handle the authentication of users.
- The game should be designed and built to production standards.
- You are advised to get a basic version of the game working first, before adding richer features.

Conceptual models

The UML models below show the main concepts used in the application and their relationships.

The Maze

Data used to configure the game should correspond to this conceptual model and will define a Maze and its initial contents of Rooms, Passages and Items.

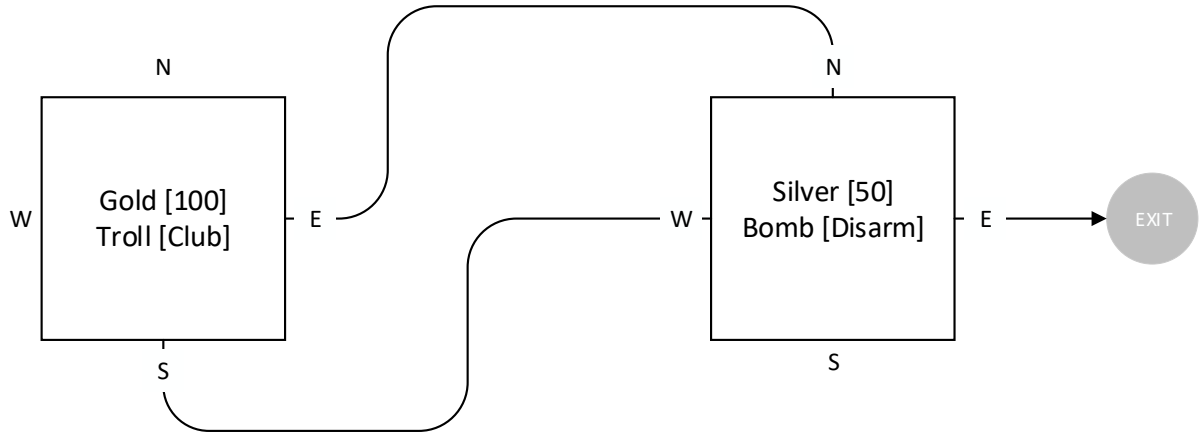


The following statements clarify and expand upon the meaning of this model.

- A Maze has a set of any number of Rooms.
- Rooms are connected by means of Passages.
- Each Room may have between 1 and 4 Passages leaving it, in the directions North, East, South, or West.
- Each Passage with isExit == false will connect two Rooms, which might be the same.
- Such Passages are bi-directional, i.e. will take you back to where you came from.
- Each Passage with isExit == true will connect to exactly one Room.
- There can be only one Room with an exit Passage.
- Each Room contains a set of Items, where an Item is either a Treasure or a Threat, and has a name.
- Each Treasure has an Integer value, e.g. Gold (value 100).
- Each Threat may be defeated by one Action, e.g. a Troll may be defeated by the action called Club.
- Once a Treasure has been collected or a Threat defeated, it will disappear from the Room for the remainder of the game.

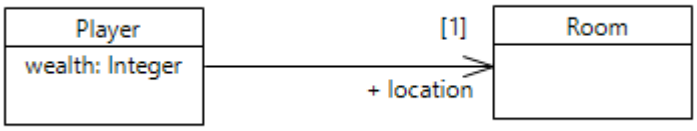
Example

The diagram below illustrates a simple configuration of a Maze with two Rooms, two non-exit Passages, each Room containing one Treasure and one Threat, and one Room having the exit Passage.



The Play

The state of the game during play should correspond to the conceptual model below, which is an extension of the previous model.

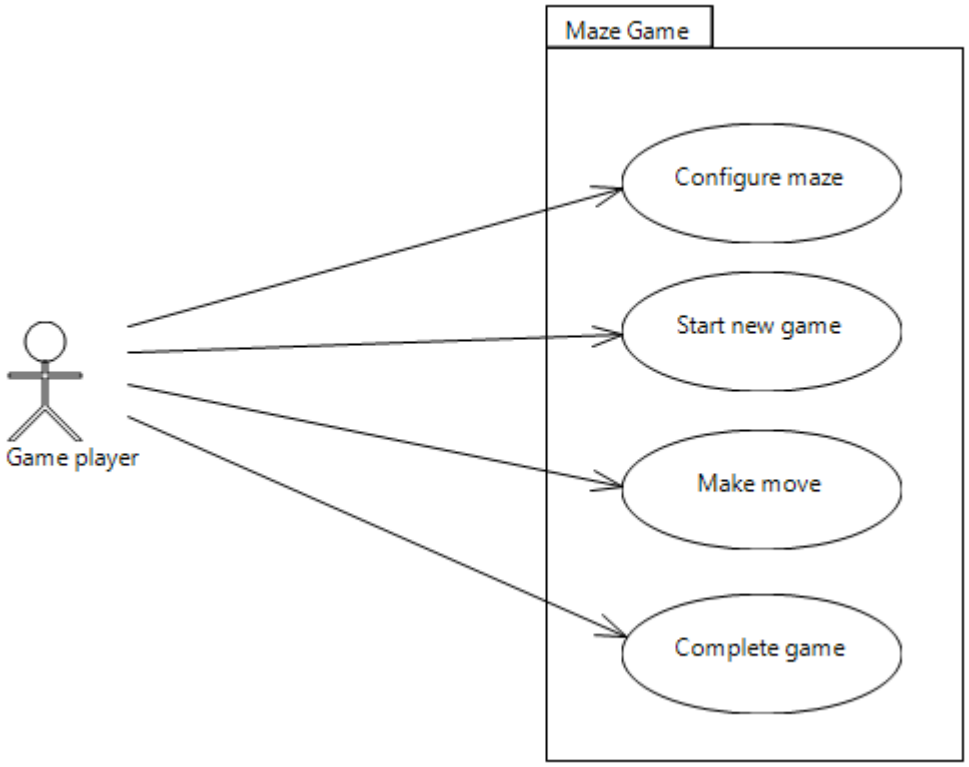


The following statements clarify and expand upon the meaning of this model.

- When game play is started after configuration, the Player should be deposited in a random Room.
- As play progress, the Player moves from Room to Room through Passages, and carries out actions in the current Room, which may cause the Player to gain or lose wealth.
- When the Player takes the exit Passage, play finishes.

Use Case Model

The UML Use Case model below shows the interactions that a user may have with the game.



Configure maze:

- Specify text file that contains maze configuration data.
- Read and validate the configuration file.
- If valid, construct program data to represent the maze.
- If not valid, report errors in the file.

Start new game:

- This may only occur with a valid game configured.
- A new Player is created with no wealth and deposited in a random Room in the Maze.
- The game will report to the user which Threats and Treasures are to be seen in the current Room.
- A game may be started at any time during play and will reset the maze to its original configuration.

Make move:

- The user can see what Treasures and Threats are in the current Room.
- The user may pick up a Treasure, in which case the Treasure is removed from the room and the Treasure's value added to the Player's wealth.
- The user may deposit wealth in the Room in the form of coins (i.e. instances of Treasure with name == "Coin"), each of value 1. Depositing a coin subtracts 1 from the Player's wealth. Depositing coins may be useful to recognizably mark Rooms that have been visited.
- The user must eliminate all Threats before leaving the Room.
- A Threat is eliminated by carrying out the Action that defeats it. The user must discover which Action this is and must specify which Threat is being acted upon.
- Once there are no more Threats the user may leave the room in one of the directions in which there is a Passage.
- If the user chooses a direction in which there is no Passage the program will announce "You may not go that way".
- If the chosen Passage is an exit the game completes.

Complete game:

- The Player has found the exit Passage.
- The game reports the Player's wealth to the user.
- The user may now initialize a new configuration, start a new game with the existing configuration, or quit.

On completion, please upload documentation relating to the project deliverables and a completed project declaration (provided separately) to the relevant folder location as specified by your training provider. Alternatively, please send to epateam@bcs.uk