Daniel Clayton

Drew Waller

**Due:** 10/30/2018

**Group:** Proj01 5

**Project 1 – Design**

**Use Cases**

**Use Case 1.** Playing\_Shelby\_Center (main menu)

**History** created 10/29/18 Daniel Clayton

**Description** Player selects the play game option from the start menu, after opening the game executable.

**Assumptions** Player always completes the game when it is started.

**Actors** Player (Primary)

Probability-based system (C++ Random number generator)

**Steps** 1. Player runs executable of program

2. Player selects play game option from text-input menu (Option ‘1’)

3. REPEAT

3.1 Player moves forward 1 position.

3.2 A randomly triggered event occurs (Either none, encounter, puzzle, or exchange.)

3.3 Player’s attributes are updated (Positively or negatively)

3.4 IN PARALLEL

3.4.1 Players attributes are checked to make sure they are non-negative

4. Player wins game after 20 successful non-negative iterations.

**Variations #1** Player may lose game, if any attribute falls to 0 or below.

**Non-Functional High Score Updates:** After a player completes the game, high score needs to be calculated and added to the high score list.

**Issues** Implementation is incomplete and needs to be applied. Classes must be implemented, and relationships established using a UML class diagram.

**Use Case 2.** High Score Menu

**History** created 10/29/18 Daniel Clayton

**Description** Player selects the “high scores” option from the text-input start menu after opening the game executable.

**Assumptions** There are prior existing high scores that can be displayed.

**Actors** Player (Primary)

Basic Input/Output (C++ cin and cout statements)

Dynamic database that updates after completed game iterations.

**Steps** 1. Player runs executable of program

2. Player selects “High Scores” option from the text-input menu.

3. REPEAT

3.1 Display the current element of the dynamically allocated array.

3.2 Concatenate a newline character onto the end of that element.

3.3 Advance to the next high score, up to a maximum of 10 high scores.

**Variations** There will be a different number of high scores, anywhere from 5 (pre-populated) high scores to no more than 10 of player-achieved high scores.

**Non-Functional** Every high score will be a simple display mechanism. No player interaction is required.

**Issues** Implementation is incomplete and will be established upon project coding portion.

**Use Case 3.** Triggering an event within the gameplay.

**History** created 10/29/18 Daniel Clayton

**History** edited and added material 10/30/2018 Drew Waller

**Description** Player is in progress moving through the hall and triggers an occurrence that is probability based upon movement. These occurrences are:

25% nothing happens.

30% player encounters a puzzle (can impact payer positively or negatively).

10% player encounters a professor (loses a random amount of time, gain intelligence).

15% player is attacked by grunt work (lose time and intelligence).

10% player must grade papers (lose time, gain money).

**Assumptions** All the player’s attributes are above 0.

**Actors** Player (Primary)

Probability-based system (C++ random number generator)

Set of Puzzles, Events, or Encounters.

**Steps** 1. Player is in-game and initiates a move command.

2a. System generates random integer between 0 – 100. if that integer is between (0 – 24)

2a.1. No changes occur, player just moved forward.

OR

2b. The randomly generated integer is between (25 - 54)

2b.1 Player gets a puzzle randomly assigned from a set of puzzles.

2b.2 If player answers correctly, gain prescribed attributes listed with puzzle.

OR

2c. Player is assigned a value within the “Encounter Professor” integer is between (55 - 64)

2c.1 Lose time

2c.2 Randomly (60% chance) gain intelligence at a static rate – e.g. 20 intel.

OR

2d. Player is assigned a value within the “Encounter Graduate” integer between (65 - 74)

2d.1 Loses time statically (10 time). No other gains.

OR

2e. Player is assigned a value within the “Grunt Work” integer between (75 - 89)

2e.1 Loses time and intelligence statically. (10 time and 10 intelligence.)

OR

2f. Player is assigned a value within the “Grade Papers” integer is between (90 - 99)

2f.1 Lose time (10 time) but gain money (50 money.)

3. Update character information

**Variations** The various outcomes that can occur from one move. (Listed above)

**Non-Functional** This occurs automatically. Player interaction will only be required to initiate the move and solve a puzzle if the outcome involves that.

**Issues** This needs to be implemented in full still with the code portion of the project.