Miranda Castle, Karen Hernandez, Truc Le, Julia Waggoner

3 . 5 . 2015

Object Oriented Software Development

Professor Scott Thede

**Use Cases for Text Adventure Game**

**I. Use Case: Movement**

1. Game prompts user to enter a command.
2. Player enters a movement command, such as “north,” to move north. Other cardinal directions are accepted. “Back” is also an option to move player back to the immediate previous space.
3. Game matches player’s command to a set of valid commands.
4. Game executes the command to alter the environment and player’s placement accordingly.
5. Game notifies player about the command’s result (the command’s effect on the game environment).

**Variation #1**

1.1 Player enters movement command that is not appropriate to player’s current environment (trying to swim with no water, for example).

1.2 Game notifies player of the invalid command and prompts player to re-enter movement command.

1.3 Player enters another command.

**Variation #2**

2.1 Player enters a command that does not exist.

2.2 Game notifies player of the invalid nature of the command.

2.3 Player enters another command.

**II. Use Case: Look at Items**

1. Player enters “look” to find all items that are in the current environment.
2. Game prints name and type of items in the environment.
3. Player enters “look at” + item to examine a specific item.
4. Game prints description of the item that player specified to look at.

**Variation #1**

1.1 Player enters “look” but there are no items in the current environment.

1.2 Game notifies the player that there are no items in the current environment.

1.3 Player enters another command.

**Variation #2**

1.1 Player enters “look” in an environment where seeing isn’t possible, such as a dark room.

1.2 Game prints a message explaining that the user cannot find items in the dark.

1.3 Player enters a new command.

**Use Case III: Store Items**

1. Player enters “take” to take the item that player is looking at.
2. Game saves that item into player’s inventory, a list of items that were previously taken.
3. Player enters “inventory” to check player’s inventory.
4. Game prints out all the items that are in player’s inventory.
5. Player continues game.

**Variation #1**

2.1 Player chooses not to take the items that are in the current environment.

2.2 The items stay in the environment and nothing gets added to the players current inventory.

2.3 Return to step 8.

**Variation #2**

3.1 Player enters “take” an item that is not possible to take.

3.2 Game prints a message explaining why player cannot take that item.

3.3 Return to step 8.

**III. Use Case: Save and Exit Game**

1. Player enters “save.”
2. Game saves the current environment, inventory, and progress.
3. Player enters “exit.”
4. Game prints “Are you sure?”
5. Player enters “yes.”
6. Game closes.

**Variation #1**

1.1 Player exits the game without entering save.

1.2 Game restores to last saved version when player starts the game again.

**Variation #2**

2.1 Game prints “Are you sure?”

2.2 Player enters “No.”

2.3 Game continues at current point.