CSCI 345
Assignment 2
Due: Midnight, April 29

Introduction

For the remainder of the course, we are going to design and implement a simple engine for Interactive Fiction, otherwise known as “Adventure Games”. If you're not familiar with the genre, I suggest you read the Wikipedia article.[1] Quoting from that article:

> Text adventures are one of the oldest types of computer games and form a subset of the adventure genre. The player uses text input to control the game, and the game state is relayed to the player via text output.
>
> Input is usually provided by the player in the form of simple sentences such as "get key" or "go east", which are interpreted by a text parser. Parsers may vary in sophistication; the first text adventure parsers could only handle two-word sentences in the form of verb-noun pairs. Later parsers, such as those built on Infocom's ZIL (Zork Implementation Language), could understand complete sentences. Later parsers could handle increasing levels of complexity parsing sentences such as "open the red box with the green key then go north". This level of complexity is the standard for works of interactive fiction today.
>
> Despite their lack of graphics, text adventures include a physical dimension where players move between rooms. Many text adventure games boasted their total number of rooms to indicate how much gameplay they offered. These games are unique in that they may create an illogical space, where going north from area A takes you to area B, but going south from area B did not take you back to area A. This can create mazes that do not behave as players expect, and thus players must maintain their own map. These illogical spaces are much more rare in today's era of 3D gaming, and the Interactive Fiction community in general decries the use of mazes entirely, claiming that mazes have become arbitrary 'puzzles for the sake of puzzles' and that they can, in the hands of inexperienced designers, become immensely frustrating for players to navigate.

For the purposes of our game, we will confine ourselves to commands of one or two words. We will implement rooms and related concepts.

The purpose of this assignment is to draw two UML diagrams of the classes we will have in our game and to prepare for the next assignment.

Problem Statement

For this assignment, you are to draw two UML diagrams. The first diagram will cover classes that are important to your command parser. The second diagram will cover classes representing the game.

1 https://en.wikipedia.org/wiki/Interactive_fiction, downloaded on October 19, 2012

Diagram 1 - Command Interpretation

Your command interpreter will require the following components:

1. A CommandParser which gets the commands from the user and parses them. The command parser:
   a) reads a line of text from the user and splits the line into individual *word*s.
   b) matches the words against all words that are known to the program. (See below for word matching). If there are words that are not known to the interpreter, an error message is produced and the next command is read. If more than one known word matches, that is also an error, except that an exact match (see below) takes precedence over a prefix match.
   c) checks the words against the list of all *action*s. (See actions, below.) The words match the action if there are one or two words and the action has the same number (one or two) of non-null vocabulary terms and each word is in the corresponding vocabulary term.
   d) If there is an action matching the entered words, that action is executed. Otherwise, the user gets an error message.
2. A class maintaining global objects. The purpose of this class is to act as a place to hold all of the global variables, attributes, etc. that are required by a game. Among the attributes that are present in the globals are:
   a) allWords: A collection of all the words known to the game. Used by the command interpreter.
   b) allActions: A collection of all the actions known to the game.
   c) thePlayer.
3. Actions:
   a) Actions can have one or two vocabulary terms associated with them. These vocabulary terms are used to match the action against the available vocabulary items.
   b) Actions have an execute method which is invoked by the command interpreter if this is the only applicable action for the current command.
4. Vocabulary Terms. These are sets of Words. A word matches a vocabulary term if the word is a member of the set of words.
5. Words:
   a) Words have an associated string which represents the word that is matched.
   b) Words can be matched in one of two ways, (1) an exact match, the entered word must exactly match the word, or (2) a prefix match, the entered word is a prefix of the word.

Diagram 2 - Game Objects

A number of the game objects have names which are Strings. Eventually we will require that all the names are unique. The following game objects are part of the game:

1. Rooms. Rooms have names. Rooms are always *container*s of Game Items. Rooms have a set of paths that lead from this room to other rooms.
2. Paths go from one room to another. Paths do not have names. However, paths are identified by a Vocabulary Term.
3. Game Items are inanimate objects that are scattered about the game. Game Items have names. Game Items are always found in containers. A number of different kinds of objects can be containers. Game Items can also be containers, such as a treasure chest that might hold other Game Items. However, not all Game Items are containers.
4. Movers. Movers are Game Objects that move about the rooms of the game. Movers always have a location, which is a room, in the game. Movers are always containers, that is movers can carry Game Items. There are two kinds of movers, (1) the player, (2) other movers that are managed by the game.

Example

Words known to the game include "go", "gold", "move", "south", "north", "norte".

The words "go" and "move" are part of a move vocabulary term. The words "north" and "norte" are part of a north vocabulary term. The commands "go north" and "move norte" would both match an action identified by the move and north vocabulary terms.

A path that goes north is identified by the north vocabulary term.

If the player is in a room with a north path, the command "go north" would determine that the word "north" is one of the words in the north vocabulary term, which means the player should move to the other end of the north path.

Assignment

1. On moodle I have placed two sample UML class diagrams (one simple, one with descrptive comments) that describe part of the Ropes assignment. I used the UMLet application (www.umlet.com) to draw these diagrams. (I demonstrated the UMLet application in class.) I've included notes in the diagram showing the components of the diagram.
2. You are to pick a UML tool and use it to create two UML class diagrams for the two diagrams described above. You should read the descriptions carefully to identify the classes, attributes, methods, and relationships implied by the description.

Hints

1. Read all of the description above carefully. Almost everything (but not absolutely everything!) there reflects a class, method, or attribute that should be included in your diagrams. It's your job to look at the description above and translate it into a class diagram.
2. Don't worry about the fact that the globals (diagram 1) clearly have a reference to Player (diagram 2) and that Paths (diagram 2) reference Vocabulary Terms (diagram 1). We'll get to that later.
3. Don't discard your diagrams. You should be expecting to update them as the quarter progresses.

What to Submit

This assignment is due by midnight, April 29. Make a zip file with your two diagrams. The diagrams should be in one of the following formats: .jpg, .gif, .png, .pdf. That will almost certainly mean that you will have to export the diagrams from your UML tool. Make sure to save the original file that you used to generate the diagrams. Also, please review your exported diagram before submitting it to make sure that everything looks the way you expect it to.

Grading

Grading for this lab is as follows:

- Diagram 1 50 points
- Diagram 2 50 points

Total: 100 points. Partial credit will be given.