Nicholas Grogg

Dr. Edwards

CS151

8-22-2017

**Project 1: Part 2: The Plan**

**Overview**

My game will run in a while loop with an string variable for storing input. It will use a 2d array for locations. Each array slot will have a number that corresponds to it's properties. I will have a character .cpp file and .h file that stores the character's inventory and allows the character to interact with the game. I will also have a location .cpp and .h file that sets the game's map and map properties such as locked doors, and items that can be interacted with. Overall the objective of the game will be to get to the exit within a certain amount of time.

**Objective of the game**

You are on a sinking ship, you have a limited amount of time (turns) to get to the exit. Depending on which difficulty you choose, you set the time (turn) limit. In order to get off the ship you need to find three keys and the exit before the timer reaches zero and the ship sinks. You will actually need all three keys to reach the exit. There are also collectibles that tell you where items are and give you bonus points

The three difficulties are easy, medium and hard. They will have sarcastic names for easy (akin to Wolfenstein's “Can I play daddy?” setting) and geeky references for the other difficulties.

**Character**

The character is a class with both a .cpp file and a .h file. The constructor for the character will have bool variables for the inventory, a bool value for the quit state, a bool value for the victory state, an int value for the score, an int value that is set at difficulty choice for the turns remaining, a string value for the player assigned name.

In total, my character class will have the following variables:

* inventory bool variables
* quit state bool value (for game while loop)
* bool value for the victory state (for game while loop)
* int variable for score
* int variable for the turns remaining
* string variable for name

There will be accessor functions for the inventory, score, and turns remaining. Your name will be used fairly sporadically. Primarily at the beginning and end. There will also be mutator functions for the name, inventory, score, quit bool, victory bool, and turns remaining.

In total my character class will have at least the following functions:

* Mutator/Accessor functions for inventory bool variables
* Mutator/Accessor for the string name variable
* Mutator/Accessor functions for the quit bool and victory bool
* Mutator/Accessor functions for the int turns remaining variable

**Map**

The game will be designed to prevent the user from leaving the bounds of the map, usually with condescending remarks. An example of leaving the map would be going to cell [-1][5]. The map will be within a 2d array, with specific numbers giving each cell properties. I will have properties for traversable areas, non-traversable areas, locked doors, keys and the collectible hint items mentioned below. This will all happen within a .cpp file and a .h file like the character. I will have a variable for the x array value, a variable for the y array value, a property variable for the actual cell of the array, and the 2d array itself. I will also have mutator and accessor functions for variables in the array.

I will use the following location properties:

* 1: Traversable terrain
  + Player can move here, although there isn't anything to see here.
* 2: Inaccessible terrain
  + Player cannot move here
* 3-5: Gold/Silver/Bronze locked door
  + Area cannot be passed until user has the appropriate key.
* 6-8: Gold/Silver/Bronze key
  + Key to appropriate door
* 9-11: Gold/Silver/Bronze hint collectible
  + Tells you where the appropriate key is
* 12: Exit
  + The exit, get here to win the game.

My location class will at least include:

* Variables
  + int variable for x array plane
  + int variable for y array plan
  + int property variable for each array value
  + 2d int array[13][7]
* Functions
  + Mutator/Accessor location function (to work with character function)
    - View current player location
    - Set player location
  + Mutator/Accessor location property
    - used to see current location property
    - Or change location property once user picks up an item.
  + Mutator function to set all locations to neutral property
    - Used for default constructor.

**Interactive items**

I generally don't like useless collectibles included just for points, so any items I implement will be used to progress the game. I likely will have several keys needed to get to each area. Rather than force the user to use a use command the interactive items will store a boolean value that allows the player to move into the next area. A example would be the game not letting you into an area without the red key initially, but once you find the key you can just go into the area.

I plan to implement:

* Keys: What retro style game is complete without arbitrary keys?
  + Gold key: needed to open the gold door
  + Silver key: needed to open the silver door
  + Bronze key: needed to open the bronze door.
* Hints: Tells you where an item is, also gives points
  + Exit location: what x/y is the exit on?
  + Gold location: what x/y is the gold key on?
  + Silver location: what x/y is the silver key on?
  + Bronze location: what x/y is the bronze key on?

**Commands**

I want to allow commands to be non-case sensitive, this may change depending on time constraints. Ideally I want 'Move forward' to execute the same way as 'move Forward' or 'move forward'.

I plan to use the following commands:

* help/Help: display game commands and format (command + argument)
* move/Move: move in a direction, accepts the following parameters
  + forward/Forward: move forward, increase y array value by one
  + backward/Backward: move backwards, decrease y array value by one
  + left/Left: move left, decrease x array value by one
  + right/Right: move right, increase x array value by one
* score/Score: display current score
* turns/Turns: Show turns remaining
* inventory/Inventory: Display current inventory, output items with a true bool value
* location/Location: Output current x/y location + location properties.
* get/Get: Add an item to your inventory
* exit/Exit: exit the game

**Constructors, accessors, and mutators**

In total I expect to use at least the following functions:

Character functions:

* Character constructor
* Mutator functions for inventory bool variables
* Accessor functions for inventory bool variables
* Mutator functions for the string name variable
* Mutator functions for the string name variables
* Accessor functions for the quit bool and victory bool
* Mutator functions for quit bool and victory bool variables
* Accessor functions for the int turns remaining variable
* Mutator functions for the turns remaining variable
* Int function to return location and location property
* Function for the help file

Location functions:

* Location constructor
* Accessor location for x, y and x&y variables
* Mutator location for x, y and x&y variables
* Accessor location property for 2d array
* Mutator location property for 2d array
* Mutator entire array function to set entire 2d array values at once