Crawlspace design (HW 2a)

Aaron Borjas

**Requirement specification**

This project is a text-based adventure game. The overall project will be held in a map, or crawlspace, where there are multiple different locations for the user to explore. The explorer is a person who is located at one of the locations inside the crawlspace, and has a backpack with items in it, or their inventory. The explorer begins at a certain location in the crawlspace and is able to examine or interact with their environment, as well as move to a different location. The different locations have different properties. Each location has a description that is unique to itself as well as one or more exits. Locations might also contain treasure somewhere inside. The two different objects the explorer can interact with are treasures and exits. Treasure is an item with a point value, and also has a description and name. Treasure can be picked up and the number of points it has contribute to the player's score. Exits also have a name and description, but when interacted with, can send the explorer to a different location. The explorer's goal is to traverse the crawlspace and locate as many treasures and secrets they can.

**Use cases**

Use case for beginning the game

|  |  |
| --- | --- |
| User Action | System Action |
|  | System asks user if they would like to start their adventure |
| user enters no | The game ends |
| user enters yes | the game begins, places explorer in the starting location. |

Use case for each turn

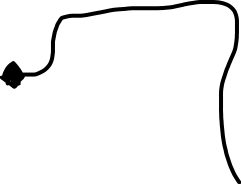
|  |  |
| --- | --- |
| User Action | System Action |
|  | System presents user their options: examine, take, or move |
| user enters examine | asks user if they want to examine the location or get a description of a selected object |
| user enters location | description of the location is printed |
| user enters object | description of the selected object is printed |
| user enters take | asks user what they want to take (presents a list of treasures in the location) |
| user enters invalid object | prints "That is not here!" |
| user enters valid treasure | explorer adds the item to their inventory and gains points for finding it. |
| user enters move | displays a list of exits |
| user enters invalid location | prints "You cannot move that way!" |
| user enters valid location | moves the explorer to the selected location and updates their current location |

Use case for ending the game

|  |  |
| --- | --- |
| User Action | System Action |
| If a user reaches a certain number of points | tells the user that they won |
|  | The game ends |

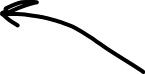
**UML diagram**

|  |
| --- |
| Location |
| -string name  -string descr  -list<Exit> exits  -list<Treasure> inventory  -void dig(Exit e)  -void drop(Treasure t) |
| +Location(string name, string desc)  +int exitLocation(string name)  +Object\* find(string name)  +void printname()  +void listExits()  +void listTreasures()  +void describe() |



|  |
| --- |
| Crawlspace |
| -vector<Location> maze  -unsigned short start  -unsigned short size |
| +Crawlspace(int start, int size)  +void addLocation(Location& l)  +void updateLoc(Exit& e, int loc)  +void setStart(unsigned short startLoc) |

|  |
| --- |
| Explorer |
| -vector<Treasure> inventory  -Location currLocation |
| +Explorer()  +void getInventory()  +Location getLoc()  +void examine(string type)  +void take(Treasure &t)  +void move(Location& l) |



|  |
| --- |
| Object |
| -string name  -string descr  -int value |
| +Object(string name, string desc, int value)  +void describeme()  +string getName()  +int getValue()  +virtual Object\* takeme() |

|  |
| --- |
| Treasure |
| +Treasure(string name, string desc, int value)  +Object\* takeme() |



|  |
| --- |
| Exit |
| +Exit(string name, string desc, int value)  +Object\* takeme() |

**Pseudocode**

***Crawlspace:***

void addLocation(Location& l) {

add a location to the maze vector

}

void updateLoc(Exit& e, int loc) {

add an exit to the exit vector for loc

}

void updateLoc(Treasure& t, int loc) {

add a treasure to the treasure vector for loc

}

***Location:***

void dig(Exit e) {

push a new exit to the exits vector

}

void drop(Treasure t) {

push a new treasure to the treasures vector

}

int exitLocation(string name) {

for(int i = 0; i < exits.length; i++) {

if name matches the name of the current exit {

return the locations number

}

}

}

Object\* find(string name) {

Object pointer item = nullptr

make an iterator called cur of a list of Treasures, set it equal to an iterator for the starting position of the list

while cur is not equal to the ending iterator of the list {

if the current item in the list is equal to name {

set item = cur->takeme

erase cur from inventory

break the loop

}

}

if item is equal to nullptr {

throw an exception stating, "that item isn't here!"

}

return item

}

void listExits() {

make an iterator of a list of exits called it, it = beginning of exits

while it is not equal to the end of exits {

describe it

increment it

}

}

void listTreasures {

make an iterator of a list of treasures called it, it = beginning of inventory

while it is not equal to the end of inventory {

describe it

increment it

}

}

void describe() {

print the name of the location

print the description of the location

if the list of exits is not empty

list the exits

if the inventory of treasures is not empty

list the treasures

}

***Explorer:***

void getInventory() {

for(int i =0; i < inventory.size; i++) {

print the current treasure's name and description

}

}

Location getLoc() {

return the Explorer's currLocation

}

void examine(string type) {

if type == "location" {

print the description of the current location

}

else {

print the description of a selected object

}

}

void take(Treasure& t) {

add the treasure "t" to the explorer's inventory of treasures

}

void move(Location& l) {

currLocation = l

}

***Object:***

void describeme() {

print the name of the object

print the description of the object

}

string getName() {

return the name of the object

}

int getValue() {

returns the value of the object

}

***Exit:***

Object\* takeme() {

print "What a concept!"

return nullptr

}

***Treasure:***

Object\* takeme() {

print "You got <x> points!"

return the treasure object this function is being acted on

}