Group Member Names
David Hatcher
Date
3/12/2020
Proposal Draft #1

Project Title:
The Disappearance of Emily Jane

Summary of Project:
The Disappearance of Emily Jane will be a mystery solving game, in the vain of the old Myst games, but entirely text based. In this game you will be a detective working the case of a young woman who has been reported missing. After weeks of no leads and very little information you decide to search Emily's house one last time in the hopes that you can figure out what happened to her. While there you will be forced to find clues and defeat a sinister entity in order to determine what happened to Ms. Jane.

There are details of the rooms, along with a layout of the game world, at the end of this proposal.

Planned Classes:
Game – This class will be the main engine of the entire game and will handle all the interactions with the game world
Room – This class will be the parent and abstract class for all the rooms of the game. These rooms will contain interactives and items.
Interactive – This class will provide things to interact with in the various rooms. Interactives will be things such as the mirror, drawers, the bookshelf, the front door and Emily's diary.
Player – This class will hold all the information regarding the player
Item – This class will be used create objects for the player to add to the inventory and complete various objectives of the game. These items will be needed to interact with certain parts of the game. Items are things like keys, flashlights, and hammers. Mirror/Jewelry Box Keys will allow access to some interactives, flashlights will allow you to see in a dark room, and a hammer will allow you to break the mirror.
Library – This will be a child of room and be used specifically for the library
Office – This will be a child of room and be used specifically for the Office
Bedroom – This will be a child of room and be used specifically for the Bedroom
Basement – This will be a child of room and be used specifically for the Basement
Kitchen – This will be a child of room and be used specifically for the Kitchen
Living Room – This will be a child of room and be used specifically for the Living Room
Entrance – This will be a child of room and be used specifically for the Entrance

Game:
Data Members:
1. title – string, this is a static string variable with the name of the game

2. rooms_names – vector<string>, this is a vector of all of the room names this will be used to create the rooms map
3. rooms – map<string,Room> this is used to store all of the rooms for the game and used to build the exits for each room
4. player – Player, this is the player object for the game
5. current_room – Room, this is for storing the current room
6. save_loc – string, this is where the player data will be saved

Member Functions:

1. Game(string) – constructor, this is initializing the game
2. print(string) – this is a static member function used to print to the console
3. read() – this is a static member function used to read from the console
4. buildItem(name) – this is a static member that will be used to build an item
5. printJournal() – this will access the friend function getJournal() from the play object
6. goToExit(exit) – this is used to change the current_room variable
7. interactWith(name) – this function is used to interact with various Interactives in the rooms
8. loadData(name) – this is used to load player data
9. saveData() – this is used to save player data
10. buildRooms() – this will build out all of the rooms
11. buildRoom(file_path) – this is used to load a single room from a file path

Player:

Data Members:

1. name – string, this is the player name
2. inventory – vector<item> this is the players inventory
3. journal – string, this is the players journal giving them an idea of what they've done and need to do next
4. flags – map<string,bool> this is a map for various game flags

Member Functions:

1. Player(file_name) – constructor, this initialized the player using the file_name passed
2. setName(name) – setter for name member
3. getName() – getter for name member
4. getJournal() – getter for journal member, this will be a friend function
5. checkFlags(flag) – this function will check the player flags to determine various things in the game.
6. setJournal(entry) – setter for journal member
7. checkInventory(item_name) – this check for an item present in the inventory vector
8. addToInventory(item) – this adds an Item object to the inventory vector
9. setFlag(flag_name,value) – this will set a flag in the flags map to whatever value is passed

Item:

Data Members:

1. name – string, name of item
2. description – string, description of item

Member Functions:

1. Item(name,description) – constructor, sets name and description
2. setName(name) – name member setter
3. getName() - name member getter
4. setDescription(desc) – description member setter
5. getDescription() – description member getter
6. operator==(item) – this will compare items to determine if two items are the same

Room:

Data Members:

1. description – string, this is the room description
2. name – string, this is the name of the room
3. exits – vector<Rooms>, this is a vector of the exits from this room
4. interactives – map<string,Interactive> this is a map containing all of the interactives
5. file_path – string, this is the file path for the room
6. items – vector<Item>, this is a vector of items in the room

Member Functions:

1. Room(file_path) – constructor for the room class
2. setDescription(desc) – setter for description member this is a pure virtual function
3. getDescription() – getter for room description
4. addInteractives(name,Interactive) – adds an Interactive object to the interactives map pure virtual
5. getInteractives() – returns the list of interactives
6. setExits() – sets the exits for a room
7. addExit(room) – adds a room to the exit
8. interactWith(action,name,player) – this is the function used to interact with an Interactive, this is pure virtual

Library:

Data Members:

1. door_close_desc – string, this is the description for when the secret door is closed
2. door_open_desc – string, this is the description for when the secret door is open

Member Functions:

1. Library(file_path, player) – constructor for the Library class
2. setDescription(desc : string) – overloads the virtual function from Room
3. addIteractives(name,interactive) – overloads the virtual function from Room
4. interactWith(action,name,player) – overloads the virtual function from Room
5. readDescriptions() – this pulls in the rooms descriptions from text files
6. checkBookshelf(player) – this function allows the player to interact with the bookshelf
7. updateDescription() – this updates the description for the room depending on player flags

Office:

Data Members:

1. mirror_broken_desc – string, this is the description for when the secret door is closed
2. mirror_not_broken_desc – string, this is the description for when the secret door is open

Member Functions:

1. Office(file_path, player) – constructor for the Office class
2. setDescription(desc : string) – overloads the virtual function from Room
3. addIteractives(name,interactive) – overloads the virtual function from Room
4. interactWith(action,name,player) – overloads the virtual function from Room
5. readDescriptions() – this pulls in the rooms descriptions from text files
6. breakMirror(player) – this function allows the player to interact with the mirror
7. touchMirror(player) – this function allows the player to interact with the mirror
8. lookMirror(player) – this function allows the player to interact with the mirror
9. updateDescription() – this updates the description for the room depending on player flags

Kitchen:

Data Members:

1. room_description – string, the basic description of the room

Member Functions:

1. Kitchen(file_path, player) – constructor for the Kitchen class
2. setDescription(desc : string) – overloads the virtual function from Room
3. addIteractives(name,interactive) – overloads the virtual function from Room
4. interactWith(action,name,player) – overloads the virtual function from Room
5. readDescriptions() – this pulls in the rooms descriptions from text files
6. getFlashLight(player) – this function adds the flash light from this room to the players inventory

Bedroom:

Data Members:

1. room_description – string, the basic description of the room

Member Functions:

1. Bedroom(file_path, player) – constructor for the Bedroom class
2. setDescription(desc : string) – overloads the virtual function from Room
3. addIteractives(name,interactive) – overloads the virtual function from Room
4. interactWith(action,name,player) – overloads the virtual function from Room
5. readDescriptions() – this pulls in the rooms descriptions from text files
6. checkDiary(player : player) – displays Emily's diary and adds information to the players journal

Basement:

Data Members:

1. lit_description – string, this is the description for when the player has the flashlight

2. door_open_desc – string, this is the description for when the player does NOT have the flashlight

Member Functions:

1. Basement(file_path, player) – constructor for the Basement class
2. setDescription(desc : string) – overloads the virtual function from Room
3. addIteractives(name,interactive) – overloads the virtual function from Room
4. interactWith(action,name,player) – overloads the virtual function from Room
5. readDescriptions() – this pulls in the rooms descriptions from text files
6. openLockbox(player) – this function allows the player to interact with the lockbox and add the mirror key to their inventory
7. updateDescription() – this updates the description for the room depending on player flags

Entrance:

Data Members:

1. no_beast_description – string, this is the description for when the mirror beast is present
2. door_open_desc – string, this is the description for when the mirror beast is NOT present

Member Functions:

1. Entrance(file_path, player) – constructor for the Entrance class
2. setDescription(desc : string) – overloads the virtual function from Room
3. addIteractives(name,interactive) – overloads the virtual function from Room
4. interactWith(action,name,player) – overloads the virtual function from Room
5. readDescriptions() – this pulls in the rooms descriptions from text files
6. openDoor(player) – this function allows the player to interact with the open the door and finish the game, assume they have found emily and broken the mirror
7. checkDrawer(player) – this function allows the player to interact with the drawer and find the jewerly box key
8. updateDescription() – this updates the description for the room depending on player flags

Living_Room:

Data Members:

1. room_description – string, the basic description of the room

Member Functions:

1. Living_Room(file_path, player) – constructor for the Living_Room class
2. setDescription(desc : string) – overloads the virtual function from Room
3. addIteractives(name,interactive) – overloads the virtual function from Room
4. interactWith(action,name,player) – overloads the virtual function from Room
5. readDescriptions() – this pulls in the rooms descriptions from text files
6. getHammer(player : player) – adds the hammer to the players inventory

Interactive

Data Members:

1. name – string, name of the interactive
2. description – string, description of the interactive

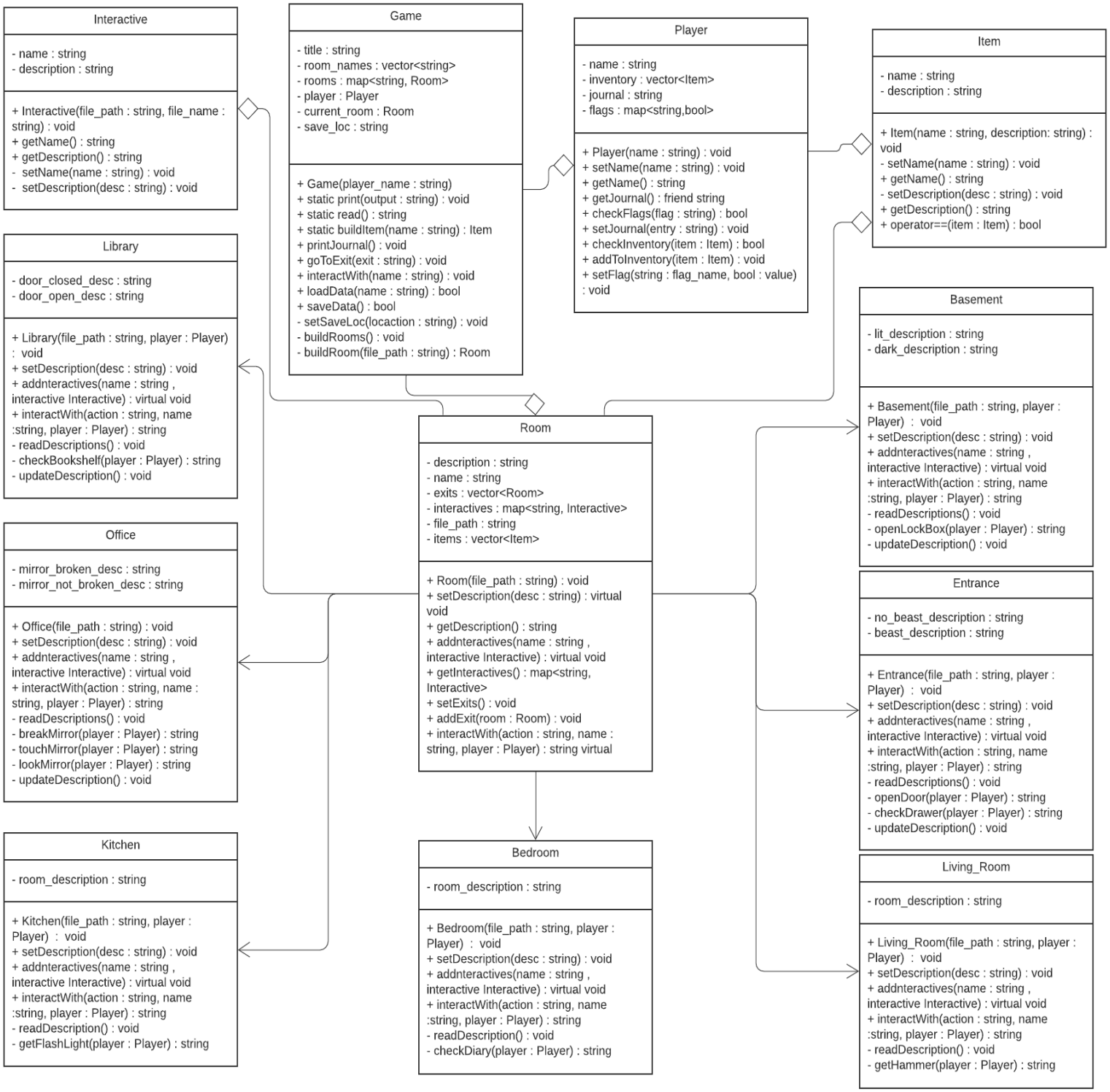
Member Functions:

1. Interactive(file_path,file_name) – constructor for interactive object
2. getName() – getting for name member
3. getDescription() – getter for description member
4. setName(name) – setter for name member, will be used by constructor
5. setDescription() – setter for description member, will be used by constructor

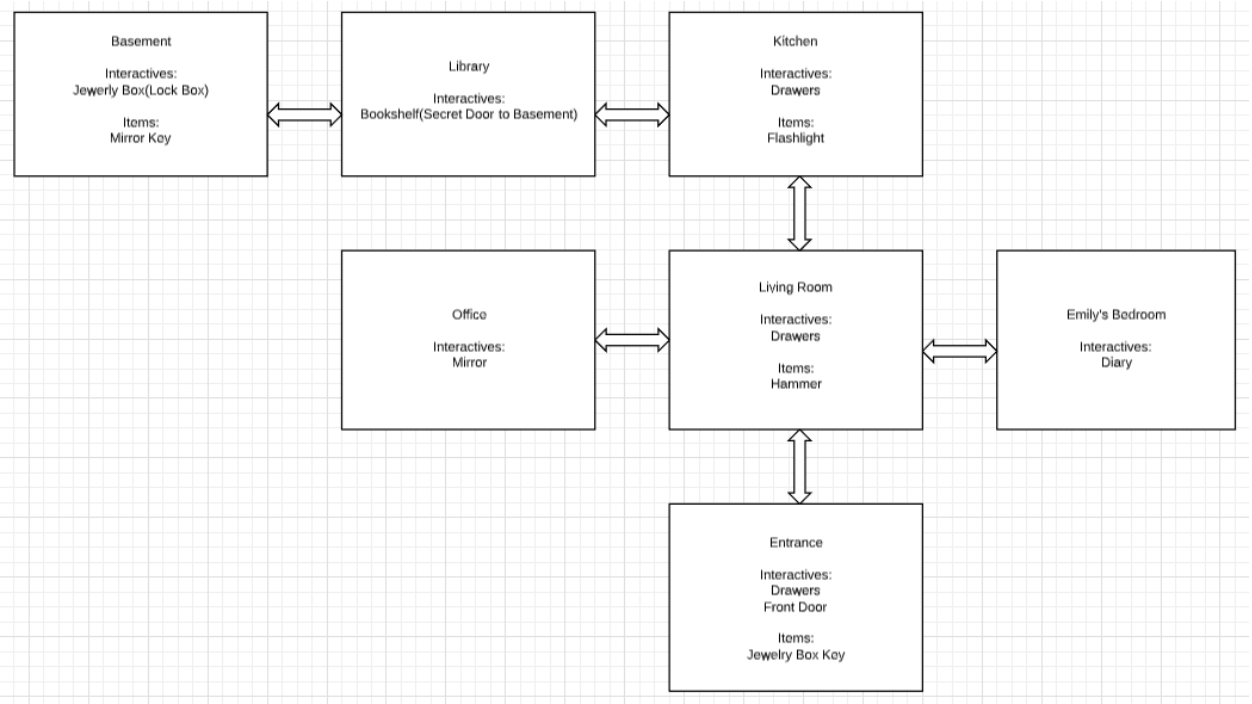
Planned Demonstration of OOP Concepts:

1. Encapsulation: To accomplish encapsulation I will be using private members in all classes and providing getters and setters for each of these. Specifically, the flags map from the player class will contain game state flags and will be accessed with the checkFlags member function on player.
2. Inheritance: Inheritance will be used with the Room abstract class and it's subclasses.
3. Polymorphism: Polymorphism is being used with the setDescription, addInteractive, and interactWith member functions of the Room class.
4. Static Members/Functions: The title member of the Game class is static as well as the buildItem, print, and read functions of the game class.
5. Friend functions: This is will be done using the printJournal function from game and getJournal function from player.
6. Overloaded operators: Item class == will be overloaded to compare Item objects
7. Text file(s): Most of the descriptions for rooms and items will be brought in from text files. The player data will also be loadable and savable.

UML Diagram:


Interactive
- name : string
- description : string
+ Interactive(file_path : string, file_name : string) : void
+ getName() : string
+ getDescription() : string
- setName(name : string) : void
- setDescription(desc : string) : void
Game
- title : string
- room_names : vector<string>
- rooms : map<string, Room>
- player : Player
- current_room : Room
- save_loc : string
+ Game(player_name : string)
+ static print(output : string) : void
+ static read() : string
+ static buildItem(name : string) : Item
+ printJournal() : void
+ goToExit(exit : string) : void
+ interactWith(name : string) : void
+ loadData(name : string) : bool
+ saveData() : bool
- setSaveLoc(locaction : string) : void
- buildRooms() : void
- buildRoom(file_path : string) : Room
Player
- name : string
- inventory : vector<Item>
- journal : string
- flags : map<string,bool>
+ Player(name : string) : void
+ setName(name : string) : void
+ getName() : string
+ getJournal() : friend string
+ checkFlags(flag : string) : bool
+ setJournal(entry : string) : void
+ checkInventory(item : Item) : bool
+ addToInventory(item : Item) : void
+ setFlag(string : flag_name, bool : value) : void
Item
- name : string
- description : string
+ Item(name : string, description: string) : void
- setName(name : string) : void
+ getName() : string
- setDescription(desc : string) : void
+ getDescription() : string
+ operator==(item : Item) : bool
Library
- door_closed_desc : string
- door_open_desc : string
+ Library(file_path : string, player : Player) : void
+ setDescription(desc : string) : void
+ addnteractives(name : string , interactive Interactive) : virtual void
+ interactWith(action : string, name :string, player : Player) : string
- readDescriptions() : void
- checkBookshelf(player : Player) : string
- updateDescription() : void
Basement
- lit_description : string
- dark_description : string
+ Basement(file_path : string, player : Player) : void
+ setDescription(desc : string) : void
+ addnteractives(name : string , interactive Interactive) : virtual void
+ interactWith(action : string, name :string, player : Player) : string
- readDescriptions() : void
- openLockBox(player : Player) : string
- updateDescription() : void
Room
- description : string
- name : string
- exits : vector<Room>
- interactives : map<string, Interactive>
- file_path : string
- items : vector<Item>
+ Room(file_path : string) : void
+ setDescription(desc : string) : virtual void
+ getDescription() : string
+ addnteractives(name : string , interactive Interactive) : virtual void
+ getInteractives() : map<string, Interactive>
+ setExits() : void
+ addExit(room : Room) : void
+ interactWith(action : string, name : string, player : Player) : string virtual
Office
- mirror_broken_desc : string
- mirror_not_broken_desc : string
+ Office(file_path : string) : void
+ setDescription(desc : string) : void
+ addnteractives(name : string , interactive Interactive) : virtual void
+ interactWith(action : string, name : string, player : Player) : string
- readDescriptions() : void
- breakMirror(player : Player) : string
- touchMirror(player : Player) : string
- lookMirror(player : Player) : string
- updateDescription() : void
Entrance
- no_beast_description : string
- beast_description : string
+ Entrance(file_path : string, player : Player) : void
+ setDescription(desc : string) : void
+ addnteractives(name : string , interactive Interactive) : virtual void
+ interactWith(action : string, name :string, player : Player) : string
- readDescriptions() : void
- openDoor(player : Player) : string
- checkDrawer(player : Player) : string
- updateDescription() : void
Kitchen
- room_description : string
+ Kitchen(file_path : string, player : Player) : void
+ setDescription(desc : string) : void
+ addnteractives(name : string , interactive Interactive) : virtual void
+ interactWith(action : string, name :string, player : Player) : string
- readDescription() : void
- getFlashLight(player : Player) : string
Bedroom
- room_description : string
+ Bedroom(file_path : string, player : Player) : void
+ setDescription(desc : string) : void
+ addnteractives(name : string , interactive Interactive) : virtual void
+ interactWith(action : string, name :string, player : Player) : string
- readDescription() : void
- checkDiary(player : Player) : string
Living_Room
- room_description : string
+ Living_Room(file_path : string, player : Player) : void
+ setDescription(desc : string) : void
+ addnteractives(name : string , interactive Interactive) : virtual void
+ interactWith(action : string, name :string, player : Player) : string
- readDescription() : void
- getHammer(player : Player) : string

Game Description:



Entrance:

This room will be the where the player starts the game. The interactives available here will be the Front Door and the Entrance Drawers. The items available here will be the Jewelry Box Key. The Jewelry Box Key will be accessed through interacting with the Drawers. The Front Door will be the exit to finish the game this will only be accessible once the player has rescued Emily from the Mirror in which she has been trapped and has defeated the Mirror Beast by smashing the Mirror.

Living Room:

This room will contain the interactive Living Room Drawers and the item Hammer. The Hammer can be obtained by interacting with the Drawers and is needed to smash the Mirror.

Emily's Bedroom:

This room will only contain the interactive Emily's Diary. This will give the user the first clues as to where Emily might be.

Office:

This room will contain the interactive Mirror. This is where Emily is trapped. The user will be able to LOOK at the Mirror, TOUCH the Mirror, and SMASH the Mirror. The look action will be available from the start, however, to access touch and smash they will need to find the Mirror Key and the Hammer, respectively. The look action will allow them to see Emily and her description of what happened to her will be displayed. The touch action will allow them to rescue Emily. The smash action will allow them to smash the mirror.

Kitchen:

This room will contain the interactive Kitchen Drawers and the item Flashlight. To obtain the Flashlight the player will need to interact with the Drawers.

Library:

This room will have the interactive Bookshelf. This is a secret door and will add an exit to the basement to the room when interacted with.

Basement:

This room will contain the interactive Jewelry Box and the item Mirror Key. When the player enters this room, it will be described as too dark to see anything. In order see the description of the room the player will need to find the Flashlight from the Kitchen and come back to the room. They will also need to find the Jewelry Box Key to be able to open the Jewelry Box and obtain the Mirror Key.