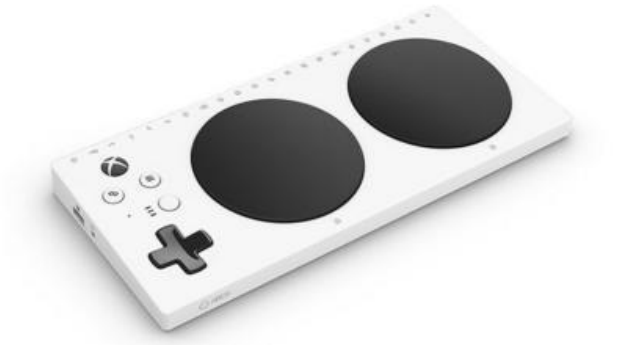
Text-to-Speech Adventure

Senior Project Proposal



Mujeeb Adelekan
madelekan@mail.stmarytx.edu
Spring 2022

TABLE OF CONTENTS

OVERVIEW

The following is an overview of my senior project, beginning with a discussion of the problem, my motivation, some background solutions, my solution, and the benefits.

Problem/Issue

Human-computer interaction needs to offer options to as many users as possible. Unfortunately, hardware and software manufacturers are prone to ignoring the needs of the visually impaired. This extends to gaming, a medium where interactivity is paramount to user experience. Although the video game industry has made great strides in terms of accessibility in recent years, including adaptive controllers and extensive accessibility options, many game developers still overlook the needs of players with different sensory and motor abilities than themselves.

Motivation

I am fascinated by alternative methods of controlling video games. Methods of input, through controllers or otherwise, and feedback to user input, are the most important aspects of game design. I admire Nintendo for their innovations in the control methods of their consoles, from the directional pad in their Game-and-Watch line, to the analog control stick in the Nintendo 64 controller, to the motion controls implanted in the Wii Remote. However, these advancements come at a price.

Several years ago, I read an IGN article by a Nintendo fan who had to come up with alternative methods of holding and interacting with the controller due to his motor impairments. Although the author conceded that the technique was not ideal, it was enough to allow him to enjoy the games to the best of his ability. However, a new challenge presented itself when the Wii was revealed. While the rest of the gaming community, and the media at large, was enamored by the novelty of swinging a controller like a tennis racket to do perform the action in-game, the author felt worried and displaced. The Wii was a revolution in game control, but to him, it was the opening of an already significant rift being him and his favorite medium.

I also read a story of a Japanese fifth-grader who mailed a letter to the the development team of Nintendo's game series Rhythm Heaven. This title is a rhythm-action game in which the player must respond to audio cues to succeed. The player wrote in their letter that they were grateful to be able to enjoy the game even though they were blind.

Stories like these made me realize that there is still a barrier between gaming and players who require alternative means of input. I want to study the domain of accessibility in games and develop a product that helps bridge the accessibility gap.

Background

Since gaming is a medium driven by input and feedback, game companies have experimented with alternative methods of control since the early days of the industry. Japanese video game company Nintendo developed a neck harness accessory for their Famicom console (known in the west as the Nintendo Entertainment System) that allowed the player to control a game with their chin or lower lop. However, these early attempts at alternative control, or adaptive technology, were little more than a novelty. Many of these initiatives held a lower

priority than the goal of establishing and maintaining a strong platform and brand presence.

In the last decade, adaptive technology became increasingly prevalent in the gaming industry, despite the perceived ambivalence of government regulators. In 2010, the United States Congress passed the Video Communications Act. Although the act regulates accessibility in TV, radio, and movies, among other media, it overlooks video gaming. This prompted independent organizations to advocate for accessibility in gaming and develop innovations that expand the medium to those who were left out.

One such organization is AbleGamers. Steven Spohn, the Chief Operating Officer of AbleGamers, said in an interview with gaming news website NintendoLife that AbleGamers “is a humanitarian charity”, one that is focused on the “social output” of gaming. Spohn, who suffers from spinal muscular atrophy (SMA), a disease that causes muscles used for movement to weaken and deteriorate, stated that for the first few years of AbleGamers’ existence, the biggest hurdle was convincing both those within and those outside the gaming industry that accessibility in gaming was a legitimate issue to begin with. Throughout the past decade, however, AbleGamers have “moved from advocacy to action,” said Spohn.

In 2011 AbleGamers collaborated with Evil Controllers to develop the Adroit Controller. Designed for the Xbox 260, the Adroit Controller was the first accessibility switch-based controller to be produced. AbleGamers’ work on the Adroit Controller would set the groundwork for their collaboration with Microsoft that would result in the creation of the Xbox Adaptive Controller, released in 2018 (shown on the cover of this proposal).

Microsoft has continually worked to make their Xbox gaming platform more accessible to those with disabilities. Anita Mortaloni, the Director of Accessibility at Xbox, said the number one question she sees form consumers is “What game can I play?” In 2016, Xbox added the option to

give assistive tech to Xbox Avatars, which are virtual representations of the players. As of the writing of this proposal, Xbox is rolling out a Game Accessibility Tag system that displays what features are present in a game (e.g., “Narrative Game Menus”, “Input Remapping”, “Single Stick Gameplay”, etc.)

However, there is still work to be done. Although the Xbox Adaptive Controller offers assistance for those with motor disabilities, those with visual or hearing disabilities still seek more options. Furthermore, the XAC only works with Xbox and Windows devices, thus making those devices the only ones with a dedicated controller for accessibility. Tara Voelker, the Accessibility Lead of Xbox Game Studios, stated that the gaming industry needs to form inter-corporate partnerships to combat feature exclusivity issues like the XAC’s limited device compatibility.

In addition to user input, accessibility must be considered in the design of the game itself. One example of an often-overlooked accessibility design principle is to put accessibility settings at the *start* of a first-time game launce, rather than hiding it in the settings. One particularly egregious violation of this principle occurs in the 2019 Nintendo Switch game Pokémon Sword and Shield, which has its closed captioning options locked behind “hearing buds” the player collects in an optional side quest.

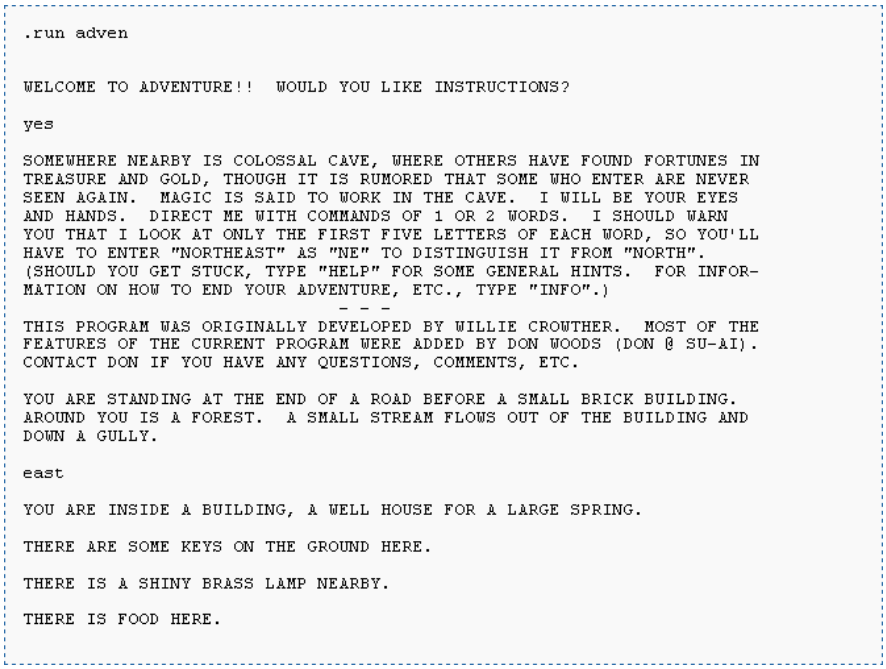
To advance accessible game design, AbleGamers runs the Player Panel Program, a body of hundreds of players with disabilities who advise the video game industry. AbleGamers also hosts *accessible.games,* a website dedicated to accessible player experiences, or APX. This website offers a Certified APX Practitioner course that trains developers to adopt a new “APX language” for game design. For example, the issue of a player clicking a button may lead to the wrong button being pressed. To remedy this, the designer can implement a way to undo a mistake. APX would call this “undo-redo”, Spohn said.

Great milestones in accessibility have been made in mainstream gaming. Controller remapping, for example, is considered a standard practice in games as closed captioning is to movies. In fact, the 2021 game Dungeons & Dragons: Dark Alliance received controversy for its lack of control remapping. Other advancements include the 2012 game Borderlands 2 adding a colorblind mode at the requests of players and the 2021 game Forza Horizon 5 adding a myriad of

accessibility features like game speed modification, high contrast mode, a screen reader, and an option to disable moving backgrounds.

When it comes to relaying environmental information to a blind player, one of the most effective means is through audio descriptions. Reading a script to a blind player with hearing ability is a great way to help the player form an image of the scenario. Richard Bartle, a developer who creates games of the text-based genre interactive fiction (IF), stated that although our senses filter information and graphics create a “model”, text speaks directly to the imagination. Bartle reinforced that despite the advancements of graphics, in terms of player experience, “the farthest you can go is text.”

The documentary *Get Lamp* stated that IF explores the power of words in an interactive context. That is, IF can be considered virtual reality that exists in words. What is considered the first IF game, 1975’s Colossal Cave Adventure, was inspired by creator Will Crowther’s spelunking activities. Writing an IF game, like spelunking, involves both atmospheric description and objective, scientific bearing. Zork, created in 1977 by MIT students, was the first major IF game to allow full sentences. The popularity of Zork (later referred to as Zork I to distinguish the title from its sequels) led to the finding of IF developer Infocom, whose titles found great success in the early to mid-80s. Even though the ear of commercial IF companies is long dead, over 2,000 new works have been created since and an active homegrown IF community persists to this day.



```
.run adven


WELCOME TO ADVENTURE!!  WOULD YOU LIKE INSTRUCTIONS?

yes

SOMEWHERE NEARBY IS COLOSSAL CAVE, WHERE OTHERS HAVE FOUND FORTUNES IN
TREASURE AND GOLD, THOUGH IT IS RUMORED THAT SOME WHO ENTER ARE NEVER
SEEN AGAIN.  MAGIC IS SAID TO WORK IN THE CAVE.  I WILL BE YOUR EYES
AND HANDS.  DIRECT ME WITH COMMANDS OF 1 OR 2 WORDS.  I SHOULD WARN
YOU THAT I LOOK AT ONLY THE FIRST FIVE LETTERS OF EACH WORD, SO YOU'LL
HAVE TO ENTER "NORTHEAST" AS "NE" TO DISTINGUISH IT FROM "NORTH".
(SHOULD YOU GET STUCK, TYPE "HELP" FOR SOME GENERAL HINTS.  FOR INFOR-
MATION ON HOW TO END YOUR ADVENTURE, ETC., TYPE "INFO".)
                              - - -
THIS PROGRAM WAS ORIGINALLY DEVELOPED BY WILLIE CROWTHER.  MOST OF THE
FEATURES OF THE CURRENT PROGRAM WERE ADDED BY DON WOODS (DON @ SU-AI).
CONTACT DON IF YOU HAVE ANY QUESTIONS, COMMENTS, ETC.

YOU ARE STANDING AT THE END OF A ROAD BEFORE A SMALL BRICK BUILDING.
AROUND YOU IS A FOREST.  A SMALL STREAM FLOWS OUT OF THE BUILDING AND
DOWN A GULLY.

east

YOU ARE INSIDE A BUILDING, A WELL HOUSE FOR A LARGE SPRING.

THERE ARE SOME KEYS ON THE GROUND HERE.

THERE IS A SHINY BRASS LAMP NEARBY.

THERE IS FOOD HERE.
```

IF can be and has been enjoyed by blind players. *Get Lamp* displayed a blind user playing Zork with a speech synthesizer. The documentary interviewed Debee Norling, another IF fan who is blind, and she claimed that playing IF is “playing at being sighted." The blind players in Get Lamp emphasized the importance of creating a sense of objects in addition to providing a sense of place and movement. IF allows the player to explore a world with sight, described as “liberating” by player Austin Seraphin.

Despite the entertainment IF can bring to blind players, according to IF developer Aaron Reed, expanding accessibility of IF is not prioritized due to the medium’s niche modern audience. This is an issue I intend to help remedy.

Solution

My project will be a text adventure in the style of *Zork* (1977), but the text will be read using a text-to-speech program to assist blind players. I intend to create a room-based adventure in which the player will navigate through exits and solve simple puzzles to escape the dungeon. Tailoring the text descriptions to blind players, each room will be described with care put into the spatial relationships of objects. At first, I assumed that object descriptions outside of texture would be of less importance, but after doing my research I know that equal consideration must be put into creating a sense of object. In order to eliminate the chance of the player entering in an action that is not valid, the game will offer the player four options at any given movement and will have the player “scroll” through them using the F and J keys and select/confirm them with the Space bar. Room and exit data will be stored in a relational database and queried by the main program module. This game will be a Windows console application written in C# in Visual Studio Community 2019.

Benefits

The project will allow both sighted and visually impaired players to enjoy the same game. The screen reader will allow a blind player to process the options given to the player as well as the results of the actions the player has taken. The

text descriptions will have blind players build mental maps of the environment and their bearing in relationship to the exits and objects in each room. Since IF is an on-rails genre, inactivity can be controlled to what the designer wants the player to do. This can be used to reduce confusion, as is the case with this game limiting the player to four options. Having the user use the F and J keys and Space bar to scroll through the options simplifies controls by locking them to three keys that are part of the “neutral” typing position.

Consider the case of a new player. Once the player launches the game, the game will return the description of the starting room along with the actions the user will take. The screen reader will read the description, list the actions, then give the user directions on how to scroll with the F and J keys and select/confirm an action with the Space bar. Choosing an action will then have the game log that action for convenience and display the results of that action.

ANALYSIS

For my analysis of the text-to-speech adventure, I had to think about how information should be relayed to a blind person and how blind people should interface with the program. I chose spoken text as a medium to describe the game's world to a blind person, yet that led to another issue. How can the world be described to a blind person? Certain characteristics of objects that sighted people take for granted (i.e., the color of objects) will be ineffective. Furthermore, over-describing details of objects may lead to a loss of comprehension by a player who has no visual experience of said descriptions. That said, being too vague will create the same problem. A happy medium of detail must be considered when creating the text descriptions. As stated in the overview, a sense of object is ultimately as important to a blind person as a sense of place and movement.

Even though it is not wise to assume that *all* blind people are able to reason visually, we are able to take advantage of the fact that blind people more than often perceive the world in "spatial" means. That is, the positional relationships between objects will play a major factor in how the text descriptions are written.

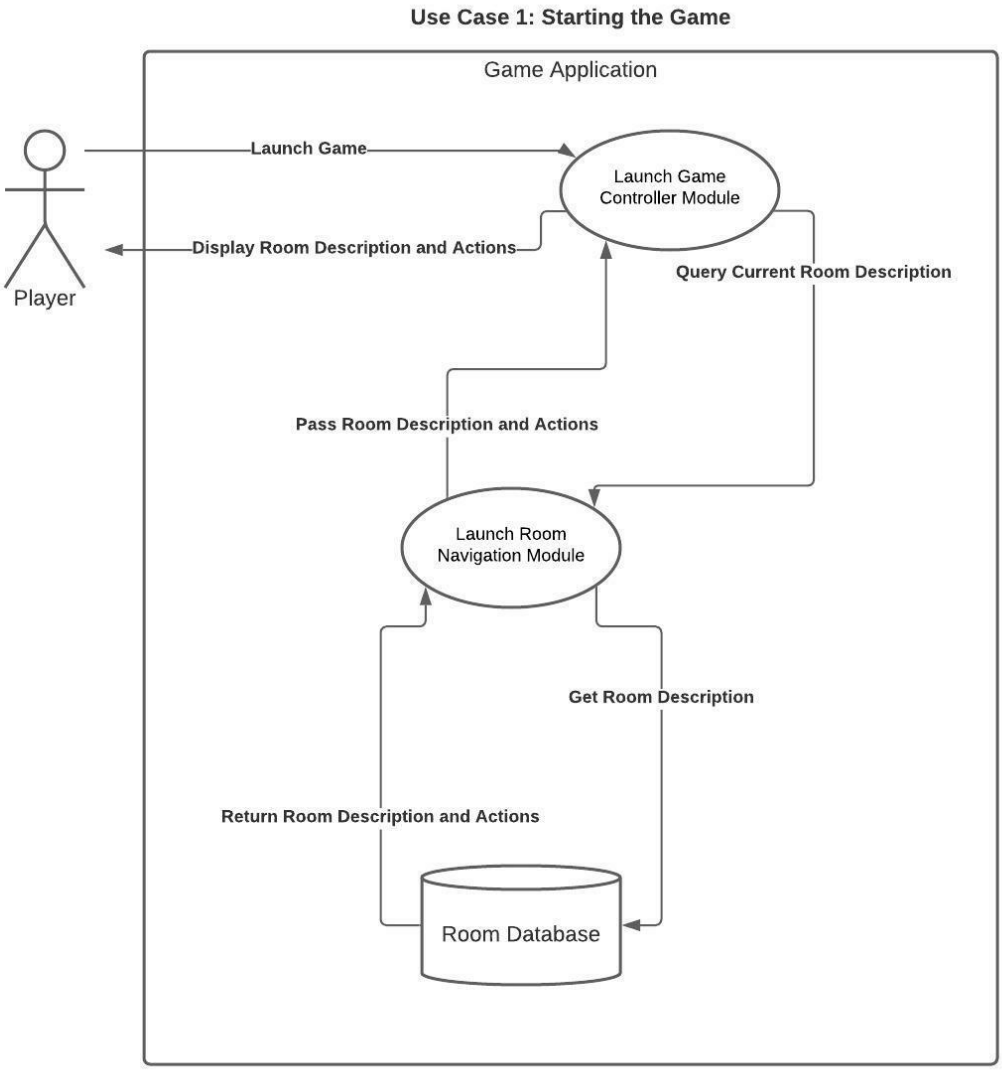
The input problem is interesting since there are various input options for blind people, including the standard keyboard. What is fascinating about the keyboard is that to be a developed typist one must solidify the spatial relationships of the keys. Very rarely do developed typists look down at the keyboard. We are trained to think about all keys in relation to the ASDF row on the keyboard. The space bar is also prominent enough so the typist can "code-switch" between their thumbs and other fingers, making it easy to remember which key is the space key.

This input analysis led to the idea of used the F,J and Space keys as input for the program.

Use Cases

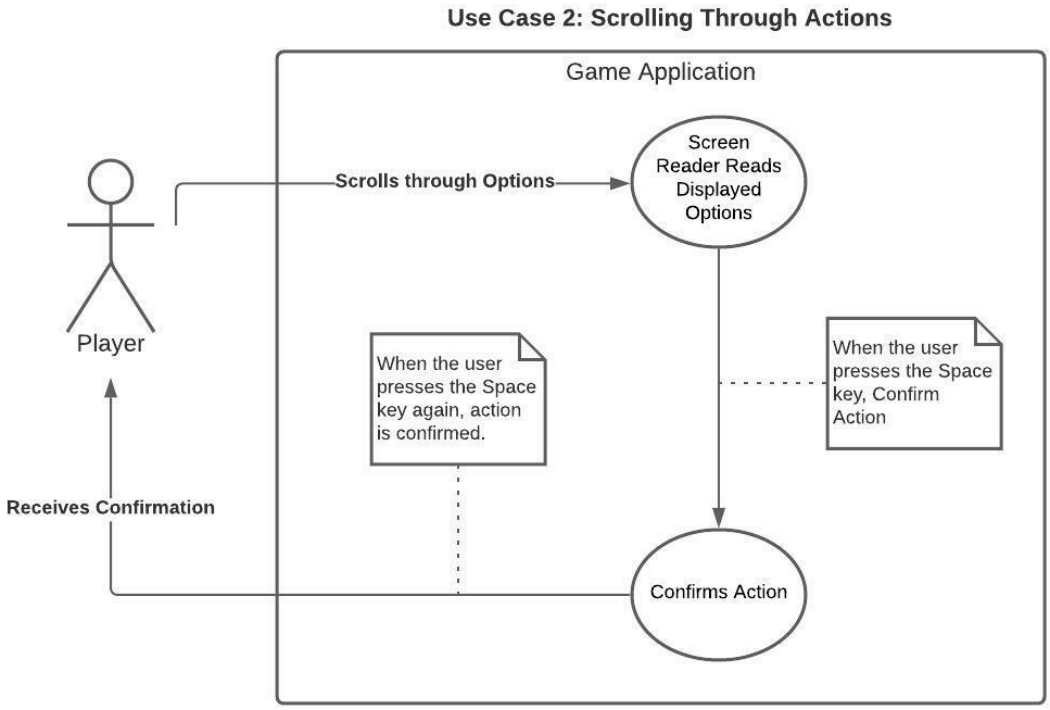
A few use cases for Text-to-Speech Adventure are discussed in detail. Each use case diagram consists of a Player actor, a large rectangle labeled "Game Application" representing the system, circles representing use cases and arrows representing external and internal interactions. The diagrams refer to system

modules further discussed in the Design section. However, to briefly explain them, the Game Controller is the central module controlling all processes of the game, the Room Navigation module keeps track of the properties of the current room the player is in as well the results of moving between rooms, and the Room Database stores information on each room as well as its exits.

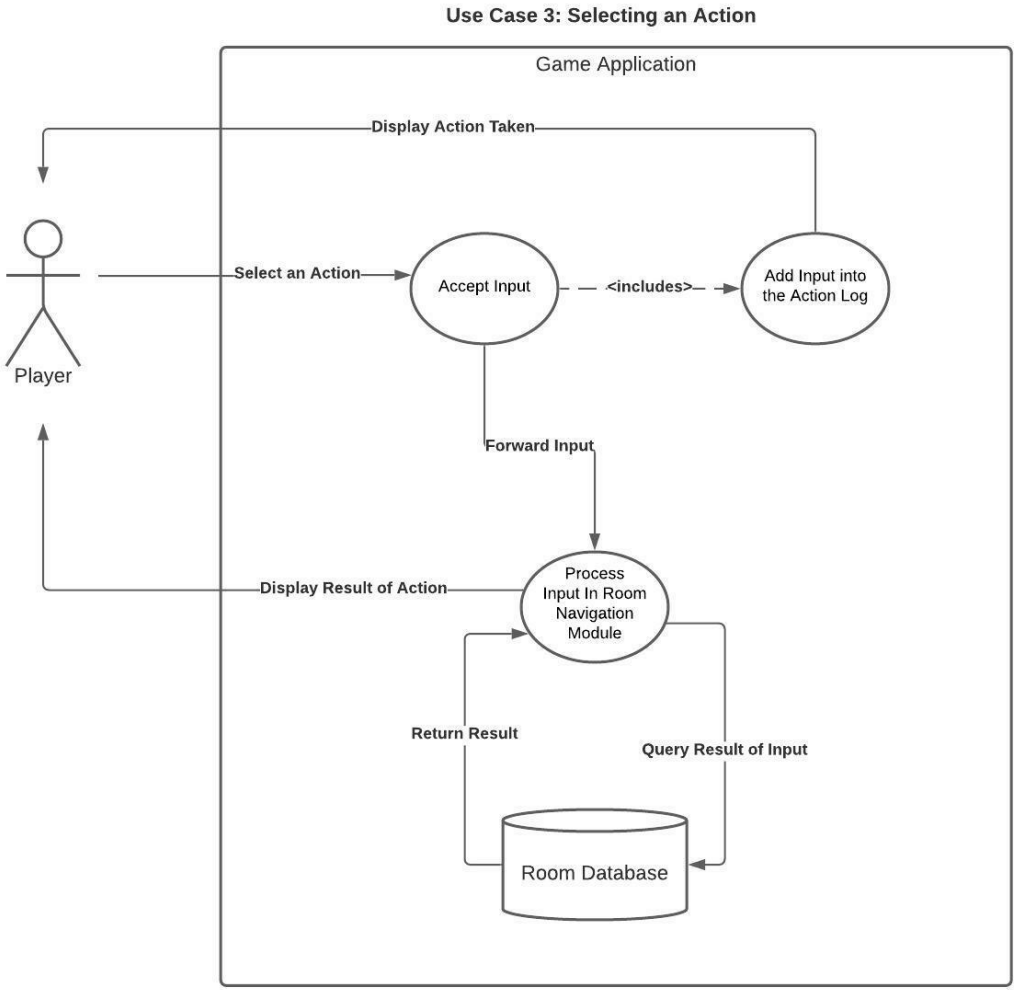


Use Case 1 shows the scenario where the user starts the game. Upon launching the game, the Game Controller module will query the description of the starting room. This query will be received by the Room Navigation module, which will get the room description and the room's four action options from the Room Database. The Room Navigation module will then pass the description and

actions back to the Game Controller module, which will then display them on screen to the player.



Use Case 2 shows the scenario where the user is scrolling through the actions. Notice the notes explaining the conditions of certain interactions in the diagram. The current action of consideration is read aloud, and that action is chosen by hitting the Space key. The player will be asked to confirm their choice by hitting the Space key a second time; doing so makes the game verbally confirm that action to the user. This is done to make sure the user does not select an action by mistake.



Use Case 3 shows how the system will display the results of a selected action. The action entered by the player is given to the Room Navigation module, which will query the result of the input from the Room Database. At the same time. the player's chosen action will be accepted and stored in the action log (**<includes>** means the "Add Input into the Action Log" use case automatically activates when the "Accept Input" use case activates). The action log will be used to display the action taken by the user. The Room Navigation module will then display the returned result of the player's action.

Functionalities

The following are the user and system functionalities of Text-to-Speech Adventure:

The system will:

- Display text on screen
- Read text aloud for visually impaired or blind players
- Read all actions the player can take
- Allow the user to “scroll” through the options
- Repeat a selected option so the user can confirm their choice
- Take keyboard input (F, J, and Space Bar)
- Share conditions and text descriptions in a database
- Store an action log that keeps a record of the users actions and the results of those actions
- Store unique data for each room
    - Room description
    - Room name
    - Array of exits
- Link each current exit with the room the player will exit into
- Use a decision network

The user will:

- Listen to an audio description of the scenario
- Press Space to relisten to the description
- Press F and J to scroll through the options
- Press Space to select an option
- Press Space a second time to confirm a selected option

DESIGN

The design of this project will be based on a decision network contain each specific description of the project. A database containing each description with the four outcomes from the four options the user is going to choose will be queried to.

Environments

Here the development environments of Text-to-Speech Adventure will be discussed. Text-to-Speech Adventure will be written C# in Visual Studio Community 2019 on a Windows 10 machine. I chose C# since it is an object-oriented language, allowing me to build the game system in a modular fashion. I am more familiar with Visual Studio than any other IDE, and the same goes for Windows among operating systems. The game will be developed as a Windows Console application, ensuring that it will run on all Windows 10 machines.

The relational database will be created using MS SQL Server Management Studio 2019. Database creation and queries will be designed using SQL since I am familiar with it and C# allows for SQL integration.

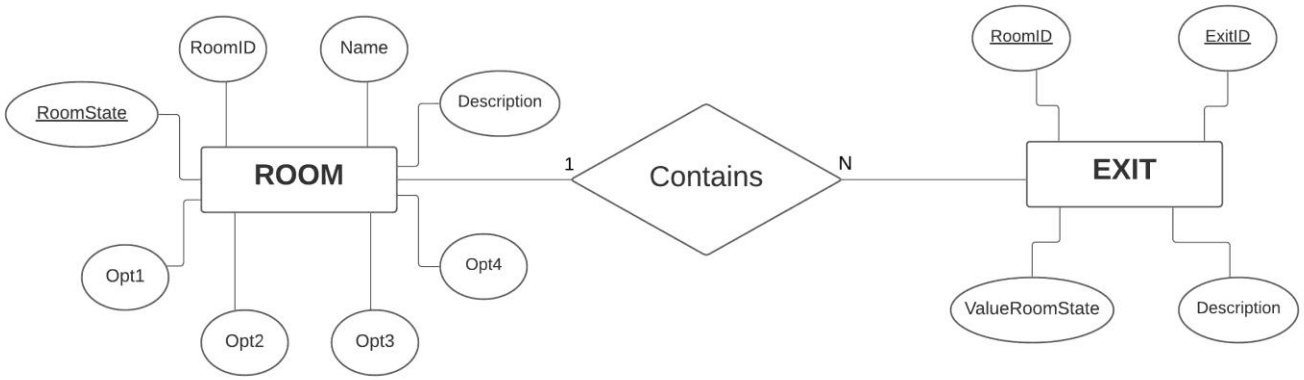
Data Structures

Text-to-Speech Adventure will work with a database that stores Room and Exit data. The development and design of the database will be discussed. First, an overview of the entity relationship diagram that describes the data will be given. Next, the schema will be discussed with a visual showing the foreign keys of each table. Finally, a discussion regarding how the database functions with example data will be given.

Entity Relationship Diagram (ERD)

Below is an entity relationship diagram, or ERD, of the Room Database. Rectangles represent entities, which correspond to database tables. Circles extending from an entity represent attributes, or columns of a table. A set of underlined attributes extending from a single entity form that entity’s primary key,

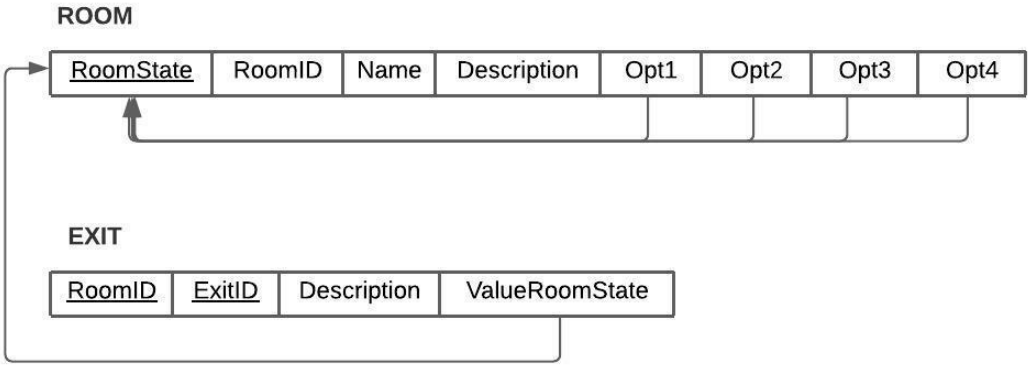
a unique value (or set of values) that identifies each instance of that entity. The diamond in the center represents a logical relationship. In this case, one ROOM contains many (or N) EXITs.



Due to the possibility of different scenarios taking place within a single room, the attribute RoomState was created as the primary key for the ROOM entity to distinguish between scenarios regardless of the room the player is in. For example, if a player picks up an orb but does not change rooms, the RoomState will change to a new value that corresponds to an instance of ROOM that has the same RoomID and Name but a different Description, one that is affected by the player currently holding the orb.

Room Database Schema

Below is the schema for the Room Database. The schema consists of the names and column headers of the database along with arrows representing foreign keys (the arrowhead pointing to the primary key that is referenced by the foreign key). The foreign keys of ROOM are Opt1, Opt2, Opt3, and Opt4 (where “Opt” is short for option.) That is, the options correspond to new room states normally; if an option is chosen, the game updates to the room state value associated with that option. **If the option corresponds to an exit, however, the “value” will be null.** This is to allow processes in the C# program to distinguish exits from other interactable objects and to determine the rooms that are the other side of each exit (Note: Using null as a “value” is permitted for foreign keys, since a foreign key value only needs to match a primary key value if the foreign key is given an actual value to begin with). The foreign key of EXIT is ValueRoomState, which references the ROOM primary key RoomState.



Note that this choice of primary and foreign keys makes ROOM a self-referential entity. Also note that EXIT.RoomID is not explicitly related to ROOM.RoomID. Since EXIT.RoomID may see repeated values (due to it being a *part* of EXIT's primary key (RoomID, ExitID)), and ROOM.RoomID may also see repeated values (due to different scenarios taking place in the same room), this design does not permit a relational link between the two. However, matching EXIT.RoomID and ROOM.RoomID will be done at the C# level by querying the values of one and using it to query information from the other table.

Room Database with Example Data

Below is a figure showing the Room Database with example data. The ExitID of each exit corresponds to the option number. For instance, if the player is in RoomState 000, picking option 3 will have the C# code check for an exit with RoomID = 0 (since that is the RoomID of RoomState 000) and ExitID = 3 (since that is the option number). If the player picked option 4 instead, the C# code will check for an exit with RoomID = 0 and ExitID = 4. The game will then go to the scenario that corresponds with the ValueRoomState of the exit found.

ROOM

RoomState	RoomID	Name	Description	Opt1	Opt2	Opt3	Opt4
000	0	Dark Room	You are in a dark room..	001	002	null	null
001	0	Dark Room	You have the orb in your hand...	005	006	null	null
002	0	Dark Room	You have the rock in your hand...	007	008	null	null
003	1	North Room	The walls are...	...	...	...	...
004	2	East Room	There is a...	...	...	...	...

EXIT

RoomID	ExitID	Description	ValueRoomState
0	3	North exit.	003
0	4	East exit	004

The hierarchal “flow” of this example data is illustrated in Sample Flowchart shown in the System Structure subsection.

Software Modules

Since C# is an object-oriented programming language, the software modules of the Text-to-Speech Adventure system will represent classes. There are five major classes in the system of this game:

1. **GameController:** The central module of Text-to-Speech Adventure. GameController coordinates the game system by being referenced in the two other major modules, Input and RoomNavigation. It is responsible for maintaining the action log and for controlling what gets displayed/read to the user.

2. **Input:** Runs an event listener for the user's keystrokes and takes in the action chosen by the user. Also makes sure the right options are being displayed for the right rooms.

3. **RoomNavigation:** Keeps track of the properties of the current room the player is in. Also determines the results of the player moving between rooms.

4. **Room:** A category of objects that contain room information queried from the database.

5. **Exit:** A category of objects that is tightly coupled with Room objects and contain exit information queried from the database.

Each class contains a particular set of attributes (characteristics) and/or methods (procedures). Each attribute and method will have its C# data type or class written after the colon (:). The attributes and methods of each class are as follows:

1) **GameController**

a. Attributes

i. **action : int** – an integer representation of the option chosen by the user.

ii. **RoomNavigation : RoomNavigation** – a reference to the RoomNavigation class.

iii. **actionLog : List<string>** -- list of actions the user has taken

b. Methods

i. **Main() : void** – entry point of the GameController class

ii. **DisplayLoggedText() : void** – display the current contents of the action log.

iii. **DisplayRoomText() : void** – display the current room's description

iv. **UnpackRoom() : void** – will unpack the characteristics of the current room

v. **ClearCollectionsForNewRoom() : void** – clear object and description data from the previous room

vi. **AddToActionLog(actionDescription : string) : void** – add the description of the chosen action to the action log

2) **Input**

a. Attributes

i. **controller : GameController** – a reference to the GameController class

b. Methods

i. **Main() : void** – entry point of the Input class
ii. **AcceptInput(action : int) : void** – takes in the integer representation of the option chosen by the user
iii. **InputComplete() : void** -- prepares the screen for a new set of options

3) **RoomNavigation**

a. Attributes

i. **currentRoom : Room** – holds the room the player is currently in
ii. **controller : GameController** – a reference to the GameController class
iii. **exitDictionary : Dictionary<int, Room>** -- a directionary consisting of int key and Room value pairs

b. Methods

i. **Main() : void** – entry point of the Input class
ii. **UnpackExitsInRoom() : void** – will send descriptions of each of currentRoom's exits to the GameController to be displayed on screen. Also will populate exitDictionary with (keyInt, valueRoom) pairs.
iii. **RespondToInput(action : int) : void** – will determine the result of the action chosen
iv. **AttemptToChangeRooms(exitAction : int) : void** – updates current if the player chose to go through an exit
v. **ClearExits() : void** – clear exit data from the previous room

4) **Room**
   a. Attributes
      i. **description : string** – holds room's description
      ii. **roomName : string** – hold room's name
      iii. **exits : Exits[ ]** – hold the room's exits

5) **Exit**
   a. Attributes
      i. **keyInt : int** – integer representing an action associated with the exit
      ii. **exitDescription : string** – the exit's description that will be displayed
      iii. **valueRoom : Room** – the room the exit goes into

System Structure

In this subsection, a class diagram of Text-to-Speech Adventure will be presented, followed by a discussion regarding a sample flowchart based on the example data discussed in the Data Structures section.

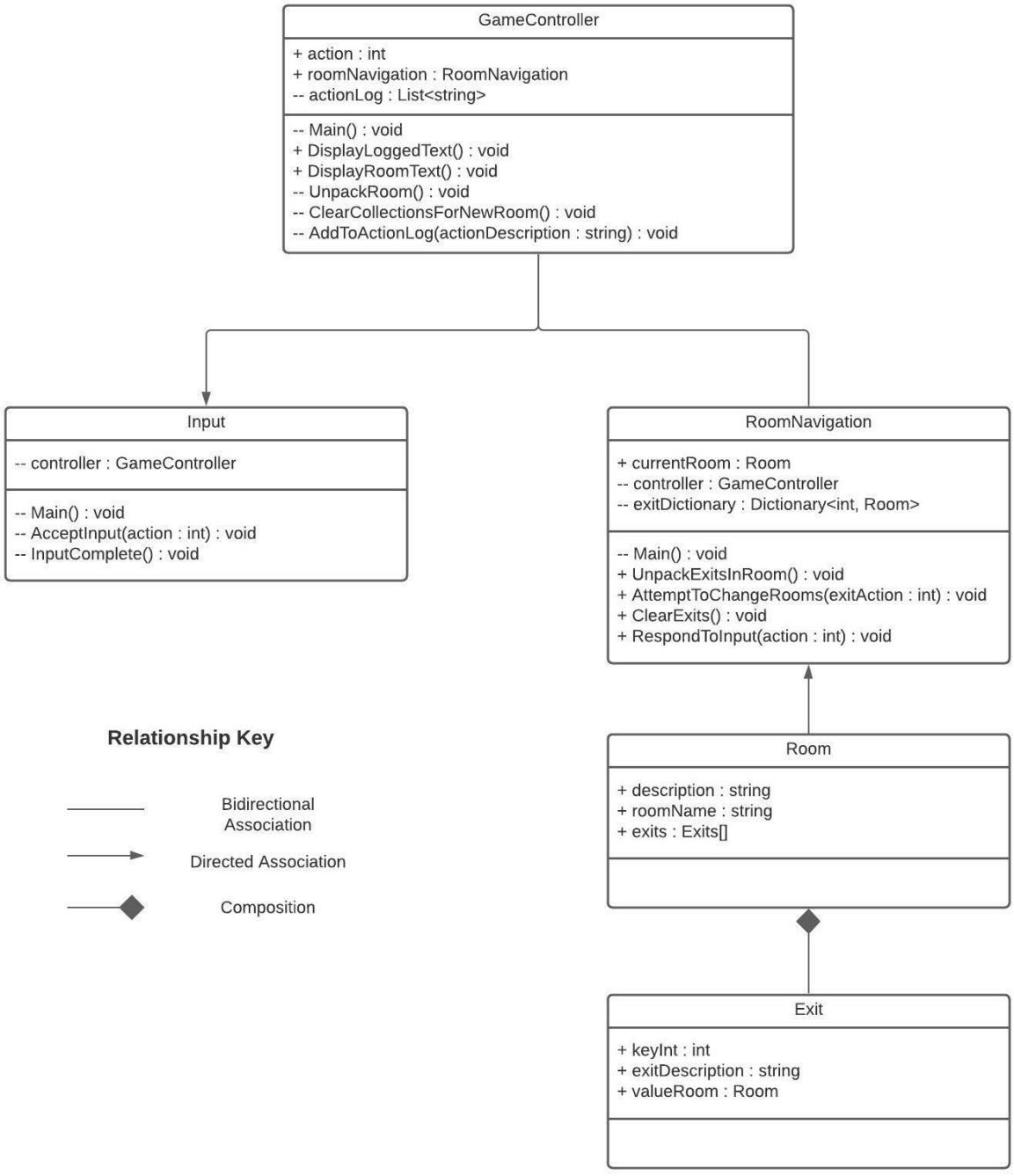
Class Diagram

Below is the class diagram of Text-to-Speech Adventure, which organizes the class software modules, attributes, and methods discussed in the Software Modules section. This diagram shows the class names, attributes, methods, and relationships between each class. A plus sign (+) signifies that an attribute or method is public, or visible by other classes. A minus sign (--) signifies that an attribute or method is private, or only visible by the class in which that attribute or method is contained.

The relationships present in the class diagram stand for the following:

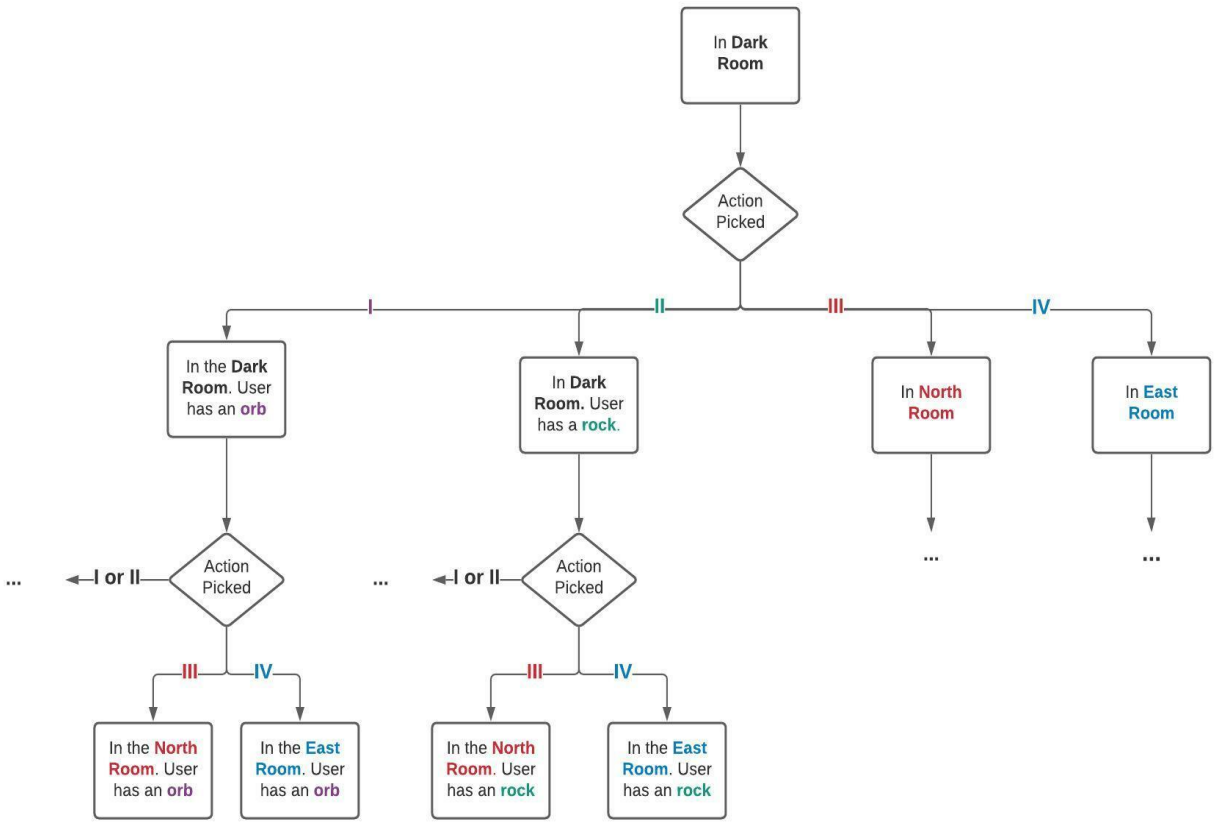
- **Bidirectional Association:** Two classes both contain an instance or reference of the other.

- **Directional Association:** One class contains an instance or reference of the second class. The arrowhead points to the class containing the instance or reference.
- **Composition:** One class is part of another. That is, the former cannot exist on its own; it logically must be part of the latter class (the container). The filled diamond is attached to the container class.



Sample Flowchart

The following flowchart offers a higher-level view of the flow of scenarios outlined in the example data discussed in the Data Structures subsection.

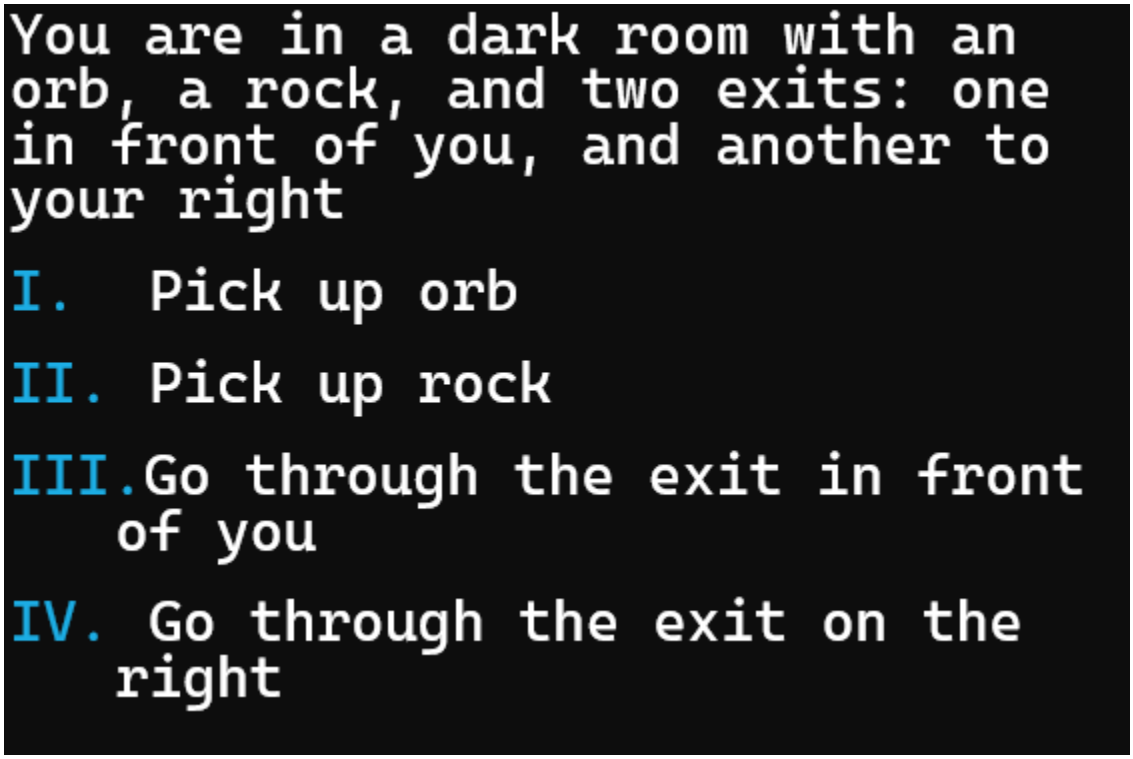


Based on the example data, the user will start in the Dark Room. The player will be able to choose one of four actions, each with a unique result. For example, if the player chooses Option I, they will obtain an orb. If the player picks Option IV afterward, they will be in the East Room *and* still have the orb. From the start of the game, the player has the chance to pick up an orb, pick up a rock, or ignore both and head straight into the exits to their north or east.

Interfaces/Scenes/Screens

The following are two example draft screens of the program. The screens hold descriptions based on the example data as details of all the rooms have not been finalized:

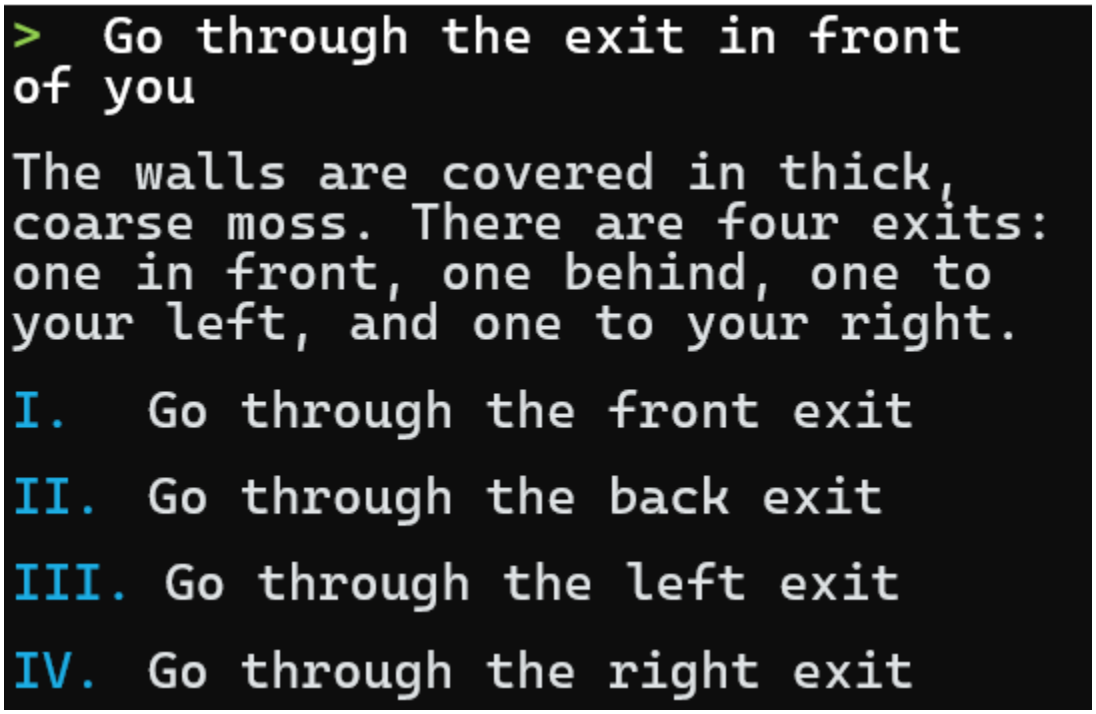
(Draft Screen 1) Start of Game: the Dark Room --- Room State 000



The program will do the following

1. *Read the description aloud*
2. *Read all the options (e.g., "Your options are 1) pick up orb, 2) pick up rock, 3) Go through the exit in front of you, 4) Go through the exit on the right")*
3. *Ask the user to hit F or J key to start scrolling through the options, and if the user wants to choose an option, they are told to press the Space key*
    a. *Each option will be read aloud when "highlighted"*
4. *When the Space Key is pressed, tell the user to either hit Space to confirm or hit F or J to return to scrolling.*

(Draft Screen 2) Option III Chosen: Player is in the North Room --- RoomState 003



The program will do the following

1. ***Update the screen*** *with the action the user has taken, the new room description, and the new actions the user can take.*
2. *Read the description aloud*
3. *Read all the options (e.g., "Your options are 1) go through the front exit, 2) go through the back exit, 3) Go through the left exit, 4) Go through the right exit")*
4. *Ask the user to hit F or J key to start scrolling through the options, and if the user wants to choose an option, they are told to press the Space key*
    a. *Each option will be read aloud when "highlighted"*
5. *When the Space Key is pressed, tell the user to either hit Space to confirm or hit F or J to return to scrolling.*

PLAN

The following is a plan for the development of the Text-To-Speech Adventure. There are 21 development tasks that span from December 1st to April 28th. The two longest tasks is testing the whole product and fixing any bugs. The second longest task is researching game accessibility.

Tasks

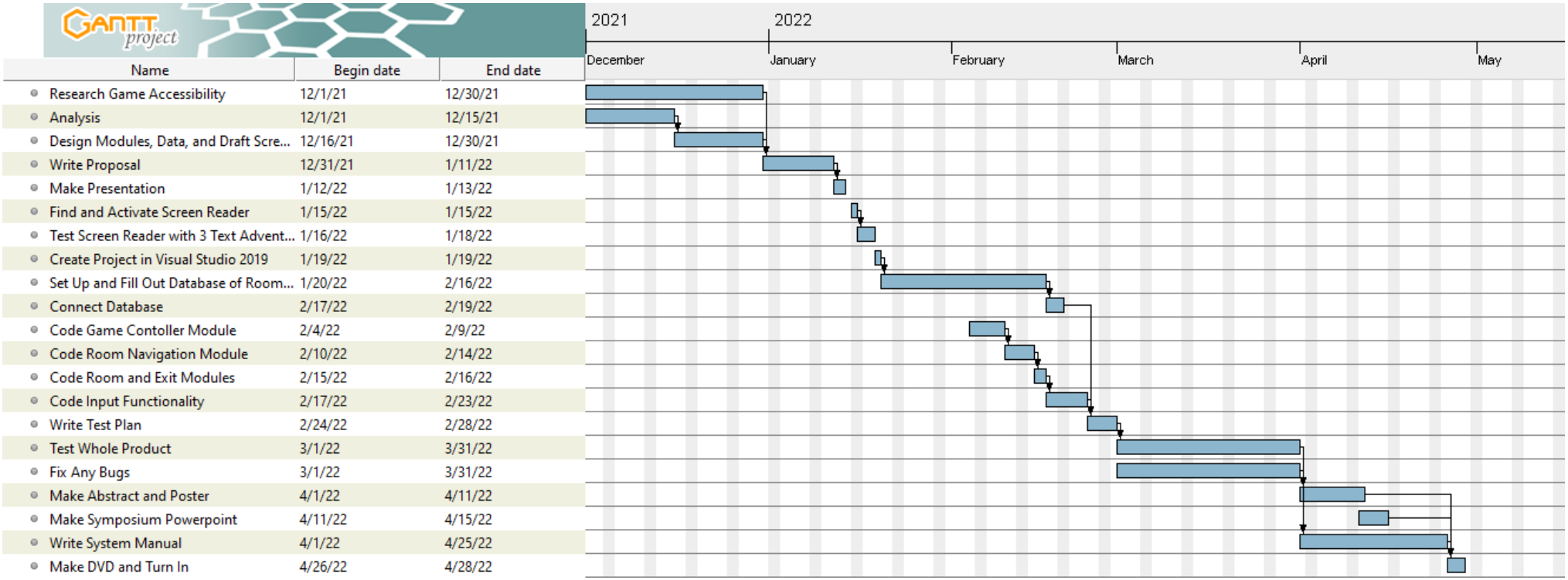
Below is a list of tasks that were determined to be important for the development of Text-To-Speech Adventure. Each task has a corresponding description. The following subsection Schedule organizes each task into a Gantt Chart.

Task Name	Description
Research game accessibility	Read articles regarding accessibility, write findings in the proposal
Analysis	Think about how to relay information to and take input from blind players
Design- modules, data, charts, screens	Determine the functionalities and draft the screens of the game
Write proposal	Write the proposal
Make presentation for Senior Project	Use proposal to write PP for Senior Project with just the slides stated by the instructor
Find and Activate Screen Reader	Find a suitable screen scraper to use on the project.
Test Screen Reader with 3 text adventure games.	Test the screen reader with the text adventure titles Zork I, Seastalker, and Hitch-Hiker's Guide to the Galaxy.
Create Project in Visual Studio 2019	Create the project using the Visual Studio 2019 IDE. The project will be a Windows Console Application.
Set up and Fill Out Database of Rooms and Exits	Use MySQL Server to create the room database. Fill the tables with the values of the current room, the number of exits, the room description, and the number of actions the user can take. Also store Exits with their directional descriptions and room values will be stored for each room.
Connect Database	Connect database with the Visual 2019 Studio Project
Code Game Controller module	Program the Game Controller class module as the central module that displays the text, keeps an

	action log, and refreshes interactable options in each room.
Code Room Navigation module	Program the Room Navigation class module which will keep track of the current room, the room's information, and processes the actions the user takes to change the room.
Code Room and Exit modules	Program the Room and Exit class modules. The Room class module will hold a room description, name, and exits. The Exit class module will hold the exit description and room values. The information will be queried from the game database.
Code Input Functionality	Code the "scrolling" mechanism of the game options. Have the program state the options in order and give directions to the player on how to select an option using the F and J keys and hitting Space Bar twice to confirm.
Write test plan	Test modules independently and when integrated
Test whole product	Black box test the product.
Fix any bugs	Debug to ensure system runs perfectly
Make abstract and poster	Develop and abstract and poster describing the product
Make symposium PP	Create the symposium power point
Demonstrate product	Show completed project to judges
Write system manual	Complete a document that describes the development and use of a system manual
Make DVD and turn in	Copy all documents, presentations, and product files with exe to a DVD (maybe at CS office) and give to instructor

Schedule

Using GanttProject, the 21 tasks were organized into a Gantt chart shown on the following page. The time span of each task is both written and represented in a visual manner, with the chart spanning from the start of December to the start of May.



Name	Begin date	End date
Research Game Accessibility	12/1/21	12/30/21
Analysis	12/1/21	12/15/21
Design Modules, Data, and Draft Scre...	12/16/21	12/30/21
Write Proposal	12/31/21	1/11/22
Make Presentation	1/12/22	1/13/22
Find and Activate Screen Reader	1/15/22	1/15/22
Test Screen Reader with 3 Text Advent...	1/16/22	1/18/22
Create Project in Visual Studio 2019	1/19/22	1/19/22
Set Up and Fill Out Database of Room...	1/20/22	2/16/22
Connect Database	2/17/22	2/19/22
Code Game Contoller Module	2/4/22	2/9/22
Code Room Navigation Module	2/10/22	2/14/22
Code Room and Exit Modules	2/15/22	2/16/22
Code Input Functionality	2/17/22	2/23/22
Write Test Plan	2/24/22	2/28/22
Test Whole Product	3/1/22	3/31/22
Fix Any Bugs	3/1/22	3/31/22
Make Abstract and Poster	4/1/22	4/11/22
Make Symposium Powerpoint	4/11/22	4/15/22
Write System Manual	4/1/22	4/25/22
Make DVD and Turn In	4/26/22	4/28/22

COST ESTIMATE

The following is a hypothetical cost estimate given the scenario that the materials for Text-to-Speech Adventure need to be acquired monetarily. First, the cost of the materials will be discussed. Then, the sum of the material cost and labor cost will be calculated.

Materials

The hardware and software costs of Text-to-Speech Adventure are shown below. Each cost includes a web link to the source of the pricing.

Hardware							
Tool	Brand	Screen Size	OS	Processor	RAM	Price	Link
HP Pavilion Laptop 15z-cw100	HP Inc.	15.6 in.	Windows 10	AMD Ryzen 7	16 GB	$647.07	https://www.walmart.com/ip/HP-Pavilion-15Z-CW100-15-6-16GB-256GB-AMD-Ryzen-7-3700U-Mineral-Silver-Certified-Refurbished/220323746

Software		
Tool	**Price**	**Link**
GitHub	Free	https://github.com/pricing
MS SQL Server Management Studio 2019	Free	https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver15
GanttProject	Free	https://www.ganttproject.biz/download/upgrade
Visual Studio 2019 Community	Free	https://docs.microsoft.com/en-us/visualstudio/releases/2019/release-notes
Microsoft Narrator (Screen Reader)	Free	https://support.microsoft.com/en-us/windows/hear-text-read-aloud-with-narrator-040f16c1-4632-b64e-110a-da4a0ac56917

Additional Costs		
Name	**Price**	**Link**
Microsoft Developer Account Fee	$19	https://developer.microsoft.com/en-us/microsoft-store/register/

Note that all software costs are free. Also, not that a Microsoft Developer Account is required to distribute the application on Windows storefronts. The total cost of these materials comes out to **$666.07**.

Cost Calculation

Now comes the calculated cost of development.

I will assume that 2 hours of work will be put into this project each day (10 hr./week, or 12 hr./week if weekends are included). There are 21 development tasks, each with varying durations that span from December 1st to April 28th. This comes up to 149 days. Assuming hours are put into the project on weekends, this comes out to:

Number of Hours = 2 hr./day x 149 days = **298 hours**

According to the U.S. Bureau of Labor Statistics, the average U.S. labor cost as of September 2021 is $39.55/hr. Assuming this average is being paid to employees, the cost of labor would be:

Cost of Labor = $39.55/hr. x **298 hours** = **$11,785.90**

Adding the cost of materials:

Total Cost = **$11,785.90** + **$666.07** = **$12,451.97**

Although my intention is for Text-to-Speech Adventure to be free, for this hypothetical the project will be sold for $2. Selling Text-to-Speech Adventure at this price would make it so that 6,226 downloads are needed to make up the total production cost of $12,451.97

REFERENCES

"Accessible Player Experiences (APX)." *Accessible.Games - Home of APX*, The AbleGamers Foundation Inc., 2021, accessible.games/accessible-player-experiences/. Accessed 10 Jan. 2022.

Brown, Mike. "Gaming for Everyone: The Accessibility Features of Forza Horizon 5." *Xbox Wire*, Microsoft, 4 Nov. 2021, news.xbox.com/en-us/2021/11/04/gaming-for-everyone-the-accessibility-features-of-forza-horizon-5/. Accessed 10 Jan. 2022.

"C# Docs - Get Started, Tutorials, Reference." *Microsoft.com*, 2019, docs.microsoft.com/en-us/dotnet/csharp/. Accessed 12 Jan. 2022.

"Download SQL Server Management Studio (SSMS) - SQL Server Management Studio (SSMS)." *Docs.microsoft.com*, Microsoft, 8 Dec. 2021, docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver15. Accessed 11 Jan. 2022.

"Employer Costs for Employee Compensation News Release Text." *Www.bls.gov*, U.S. Bureau of Labor Statistics, 17 Dec. 2021, www.bls.gov/news.release/ecec.nr0.htm. Accessed 11 Jan. 2022.

Frank, Allegra. "Nintendo Thanks Blind Fan for His Support in a Touching Note." *Polygon*, Vox Media, 22 May 2017, www.polygon.com/2017/5/22/15675630/nintendo-blind-fan-letter-rhythm-heaven. Accessed 10 Jan. 2022.

"GanttProject: Free Project Management Tool for Windows, MacOS and Linux." *GanttProject*, GanttProject, www.ganttproject.biz/download/upgrade. Accessed 11 Jan. 2022.

"GET LAMP: The Text Adventure Documentary." *Www.youtube.com*, Google TechTalks, 10 Mar. 2011, www.youtube.com/watch?v=LRhbcDzbGSU. Accessed 22 Dec. 2021.

"Hear Text Read Aloud with Narrator." *Support.microsoft.com*, Microsoft, 2022, support.microsoft.com/en-us/windows/hear-text-read-aloud-with-narrator-040f16c1-4632-b64e-110a-da4a0ac56917. Accessed 10 Jan. 2022.

"HP Pavilion 15Z-CW100 15.6" 16GB 256GB AMD Ryzen 7 3700U, Mineral Silver (Certified Refurbished)." *Walmart.com*, Walmart, 2022, www.walmart.com/ip/HP-Pavilion-15Z-CW100-15-6-16GB-256GB-AMD-Ryzen-7-3700U-Mineral-Silver-Certified-Refurbished/220323746. Accessed 11 Jan. 2022.

"Microsoft SQL Documentation - SQL Server." *Docs.microsoft.com*, 2022, docs.microsoft.com/en-us/sql/?view=sql-server-ver15. Accessed 12 Jan. 2022.

Pressman, Roger. *Software Engineering*. 9th ed., Mcgraw-Hill, New York NY, 2020.

"Pricing · Plans for Every Developer." *GitHub*, GitHub, Inc., 2022, github.com/pricing. Accessed 10 Jan. 2022.

"Register as an App Developer – Windows App Development." *Developer.microsoft.com*, Microsoft, 2022, developer.microsoft.com/en-us/microsoft-store/register/. Accessed 10 Jan. 2022.

Spohn, Steven. “Steve Spohn: My Last Good Day.” *Abilities.com*, Abilities.com, 2022, www.abilities.com/community/last-good-day.html. Accessed 10 Jan. 2022.

Stoner, Grant. “How Xbox Is Advancing Gaming Accessibility on Its Platforms, And, Hopefully, throughout the Industry.” *IGN*, IGN Entertainment Inc., 12 Nov. 2021, www.ign.com/articles/xbox-gaming-accessibility-advancements-controllers-ui. Accessed 10 Jan. 2022.

Tilley, Scott. *System Analysis and Design*. 12th ed., Cengage Learning, Boston, MA. 2019.

“The AbleGamers Charity | the Worlds Largest Charity for Gamers with Disabilities.” *The AbleGamers Charity*, The AbleGamers Foundation Inc., 2021, ablegamers.org/. Accessed 10 Jan. 2022.

“Visual Studio 2019 Release Notes.” *Docs.microsoft.com*, Microsoft, 11 Jan. 2022, docs.microsoft.com/en-us/visualstudio/releases/2019/release-notes. Accessed 11 Jan. 2022.

“Visual Studio IDE Documentation.” *Docs.microsoft.com*, 2022, docs.microsoft.com/en-us/visualstudio/ide/?view=vs-2022. Accessed 12 Jan. 2022.

Whitehead, Thomas. “Steven Spohn on Strides in Accessibility for Gaming, and Nintendo’s Room for Improvement.” *Nintendo Life*, Hookshot Media, 30 Dec. 2021, www.nintendolife.com/features/steven-spohn-on-strides-in-accessibility-for-gaming-and-nintendos-room-for-improvement. Accessed 10 Jan. 2022.

“Xbox Adaptive Controller | Xbox.” *Xbox.com*, Microsoft, 2022, www.xbox.com/en-US/accessories/controllers/xbox-adaptive-controller. Accessed 12 Dec. 2021.