A choose your own adventure game

Task requirements:

Integrity rules:

- Your work must be your work. **Our work is our own work.**
- You must credit other people's code in the comments / group presentation. **The comments / cue cards for the presentation state the sources for the code.**
- The artwork must not be offensive. **The artwork is not offensive.**

Assessment rules:

- Members of the group might be individually graded.
- Your own work is all that is marked. Using external sources and properly crediting these sources without doing much will not get you any marks. **Should be fine for marks.**
- Your code must work in the processing IDE in the labs. **The code works in the labs.**

The documentation:

- **Your references:**
  - Included in the modules section of the following document.
- **Your plan:**
  - Included in the modules section of the following document.
- **Your progress:**
  - Included in the modules section of the following document.
- **Your testing:**
  - Included in the testing section of this document.

A choose your own adventure game

The interactive story program:

Our plan:

During the first session of the group project, Sara and I brainstormed what we were going to make – Sara and I both thought that an interactive choose your own adventure game with a horror theme would be a good, original idea. After deciding what we were going to make, we started to think about what the program would encompass which we referred to as "modules". After a week, we had created a list of a large number of modules into an online Google Docs document, which we quickly moved into an online Microsoft Word document. Throughout the brainstorming process, we did not allocate modules to anyone in particular as Corey decided that a "complete what you want to complete ASAP" plan would be far more efficient.

Next, our plan was to work together to individually complete the modules, try to get the smaller modules to work together with the larger modules, and to get the larger modules to work together with the core program. Throughout this process, we initially used GitHub as a way to keep copies of the modules and program – in case any serious problems came up. However, GitHub lacks a means to delete folders / directories so we switched to Microsoft OneDrive.

In short, Sara and I collaborated by responding to e-mails, attending workshops, discussing the direction of the program, working on the documentation (which is on an online Microsoft Word document), coming up with modules, and completing the modules, etc.

Throughout the coding process, a key focus was having tidy code. Corey achieved tidy code by incorporating lots of commenting, good indentation, keeping track of references, separating the code into 3 tabs (the core program, functions, and classes), and separating aspects of the program into functions / classes. On this note, Corey followed these Java conventions:

	1st words starts with	**Next words start with**	**Next middle words are**
Variables	Lower case letter	Upper case letter	Normal case
Constants	Upper case letter	Upper case letter	Upper case - separate words using the "_" symbol
Functions (Methods)	Lower case letter	Upper case letter	Normal case
Classes	Upper case letter	Upper case letter	Normal case

The only module that would have been impossible to complete ourselves was the "text based navigation system" by Chrisir who is a member of the Processing forums. The coded solution for the navigation system was difficult to make, as it required an in-depth understanding of the hashmap function, handling objects, separating a text file into components, and navigating through the text file using a pointer, which jumps across lines. However, we now understand how this works in greater depth and are glad that we received this insight, as our original solution for this was not that good.

A choose your own adventure game

The interactive story program - continued:

Our progress:

Throughout the development process, our progress was somewhat all over the place. The modules were initially completed on an "I want to get this done so I will get this done today basis". The modules were later made on an "I have to get this done otherwise we will not finish this program basis". Each of the modules have an author attributed to them and a date that states when those modules were finished. Once we fully integrated the modules into the program, we deleted them.

- **Interactive Story Program – V1.0 (Corey. 5/06/2017).**

- **Large Module – The mutual screens (Corey. 5/06/2017):**
    - Completed Module – Code for consistency (**Corey**. 13/05/2017)
        - The custom font file – the source (asset by **DaFont**).
    - Completed Module – The bloody hand cursor (**Corey**. 11/05/2017)
        - The cursor - the source (asset by **Zeds-Stock**).
    - Completed Module – States (**Corey**. 11/05/2017)
    - Completed Module – Haunted House icon (**Corey**. 21/05/2017)
        - The icon - the source (asset by **ClipartFest**).
    - Completed Module – Trapezium creator – not used (**Corey**. 14/05/2017)
    - Completed Module – File opener (**Corey**. 15/05/2017)
    - Completed Module – Menu bar (**Corey**. 21/05/2017)
    - Completed Module – Sound player and muter (**Corey**. 21/05/2017)
    - Completed Module – Canvas clearer (**Corey**. 21/05/2017)
    - Completed Module – Jump Scarer (**Corey**. 26/05/2017)
        - Pink Monster Graphic - the source (asset by **TrevorTheJedi**).
        - Jump Scare SFX – the source (asset by **Soundbible**).

A choose your own adventure game

The interactive story program - continued:

- **Large Module – The starting screen (Corey. 5/06/2017):**
    - Completed Module – The haunted house (**Sara**. 12/05/2017)
    - Completed Module – Starting Sound (**Sara**. 13/05/2017)
        - Background music - the source (asset by **IncompeTech**).
    - Completed Module – The timed lightning bolt animation (**Corey**. 10/05/2017)
        - Lightning SFX - the source (asset by **Free Sound Effects**).
    - Completed Module – Simplified clouds (**Corey**. 13/05/2017)
    - Completed Module – Name maker (**Corey**. 14/05/2017)
    - Completed Module – The trippy sky (**Corey**. 13/05/2017)
    - Completed Module – The rain animation (**Corey**. 9/05/2017)
        - Rain SFX - the source (asset by **Soundbible**).

A choose your own adventure game

The interactive story program - continued:

- **Large Module – The playing screen (Corey. 5/06/2017):**
    - Completed Module – Import text from an external file (**Corey**. 13/05/2017)
    - Completed Module – Manipulate a table (**Corey**. 13/05/2017)
    - ~~Failed Module – The core text navigation system (**Corey**. 21/05/2017)~~
    - Completed Module – The core text navigation system (**Chrisir**. 21/05/2017)
        - Navigation system (coded solution by **Chrisir**) – the source.
    - Completed Module – Background music (**Corey**. 1/06/2017)
        - Background music - the source (asset by **Archive.org**).
    - Completed Module – Write and plan the text (**Sara**. 31/05/2017)
    - Completed Module – Make the room backgrounds (**Sara**. 5/06/2017)
        - Attic – the source (asset by **Red Watermelon**)
        - Basement – the source (asset by **Wallpaper Safari**)
        - Bedroom one – the source (asset found on **Wallpaper Safari**)
        - Bedroom two – the source (asset found on **Wallpaper Gallery**)
        - Bedroom three – the source (asset by **Shadow House Creations**)
        - Bedroom three witch hat – the source (asset found on **Clipart All**)
        - Foyer – the source (asset by **photodash**)
        - Pirate map – the source (asset by **Find Your Duck**)
        - Hallways – the source (asset found on **Echomon**)
        - Hallway stripes – the source (asset by **Orig08**)
        - Kitchen – the source (asset found on **texturelib**)
        - Library - the source (asset by **Alaine Lookabaugh**)

A choose your own adventure game

The interactive story program – continued:

- **Large Module – The losing screen (Corey. 5/06/2017):**
    - Completed Module – Scary looking things (**Corey.** 17/05/2017)
        - Bat – the source (asset found on **ClipartFest**).
        - Cat – the source (asset found on **ClipartFest**).
        - Demon – the source (asset found on **ClipartPal**).
        - Devil – the source (asset found on **WPClipart**).
        - Ghost – the source (asset found on **ClipartFest**).
        - GIll Man – the source (asset found on **Google Images**).
        - Pink Monster Graphic - the source (asset by **TrevorTheJedi**).
        - Serial Killer – the source (asset found on **Google Images**).
        - Skeleton – the source (asset found on **ClipartFest**).
        - Slime monster – the source (asset found on **WPClipart**).
        - Spider – the source (asset found on **ClipartFest**).
        - Zombie – the source (asset found on **ClipartFest**).
    - Completed Module – Sounds of scary looking things (**Corey.** 17/05/2017)
        - Bat – the source (asset by **Soundbible**).
        - Cat – the source (asset by **Soundbible**).
        - Demon – the source (asset by **Soundbible**).
        - Devil – the source (asset by **Soundbible**).
        - Ghost – the source (asset by **Free Sound Effects**).
        - GIll Man – the source (asset by **Soundbible**).
        - Hydralisk – the source (asset by **Soundbible**).
        - Serial Killer – the source (asset by **Soundbible**).
        - Skeleton – the source (asset by **Soundbible**).
        - Slime monster – the source (asset by **Soundbible**).
        - Spider – the source (asset by **Soundbible**).
        - Zombie – the source (asset by **Soundbible**).
    - Completed Module – Backgrounds (**Corey.** 20/05/2017)
        - Wallpaper one – the source (asset by **Vahsi000**).
        - Wallpaper two – the source (asset by **Vahsi000**).
        - Wallpaper three – the source (asset by **Vahsi000**).
        - Wallpaper four – the source (asset by **Vahsi000**).
        - Wallpaper five – the source (asset by **Vahsi000**).
        - Wallpaper six – the source (asset by **Vahsi000**).
    - Completed Module – Background music (**Corey.** 17/05/2017)
        - Background music – the source (asset by **Archive.org**).
    - Completed Module – Make the screen flash (**Corey.** 11/05/2017)

A choose your own adventure game

The interactive story program – continued:

- **Large Module – The victory screen (Corey. 1/06/2017) :**
    - Completed Module – Animated Car (**Corey.** 14/05/2017)
    - Completed Module – Simplified clouds (**Corey.** 13/05/2017)
    - Completed Module – Ending monster (**Corey.** 20/05/2017)
        - Pink monster - the source (asset by **TrevorTheJedi**).
        - SFX - the source (asset by **FreeSound**).
    - Completed Module – Background music (**Corey.** 14/05/2017)
        - Background music - the source (asset by **Free Music Archive**).
    - Completed Module – Background for the Victory Screen (**Corey.** 14/05/2017)
        - Bush - the source (asset by **Singo.org**).
        - Sun - the source (asset by **ClipartFest**).
        - Tree - the source (asset by **OpenClipart**).

Game plot outline:

Constraints;

- Up to four possible choices
- Text must be short

Room list:

ROOM 01 - Foyer

ROOM 01 NOTHING - Foyer

ROOM 02 - Library

ROOM 02 NOTHING - Library

ROOM 03 - Bedroom 1

ROOM 04 - Kitchen

ROOM 05 - Bedroom 2

ROOM 06 - Attic

ROOM 07 - Hallway

ROOM 07 pt. 2 - Hallway with symbol

ROOM 01 pt. 2 - Foyer

ROOM 01 pt. 3 - Foyer

ROOM 08 - Basement

ROOM 09 - Bedroom 3

DEATH - Death screen

VICTORY - Victory screen

A choose your own adventure game

The interactive story program – continued:

Room text:

ROOM 1;

“Damn! Great aunt Agatha sure was odd… this place looks like it’s out of a horror movie!” *door slams*

“What the heck? I have no key! Let’s look around…”

- Open blue door => ROOM 2
- look in drawer => “great… nothing” text (ROOM 1 NOTHING)
- Open red door => die

ROOM 1 NOTHING;

“Great…”

- Open blue door => ROOM 2
- Open red door => die

ROOM 1 pt. 2;

“Finally! I found it! Let’s get outta here!”

- Try to open front door => ROOM 1 pt 3
- Open blue door => ROOM 2
- Open red door => die

ROOM 1 pt. 3;

“Damn it! Wrong key!”

- Open blue door => ROOM 2
- Open red door => die

ROOM 2;

“At least I’ve got something to do now.” “A map!” “Interesting…”

- Follow map => ROOM 7
- Open green door => ROOM 3
- Stay and brush up on the history of Imperial Russia => ROOM 2

A choose your own adventure game

The interactive story program – continued:

ROOM 2 NOTHING;

"Huh, the Tsar really did have a great beard. Man, I'm hungry."

- Continue reading => die
- Open green door => ROOM 3
- Follow map => ROOM 7

ROOM 3;

"A little girl's room, typical."

- Open purple door => ROOM 4
- Open wardrobe => die
- Follow map => ROOM 7

ROOM 4;

"Nothing! Not even a peanut!"

- Open fridge => die
- Open yellow door => ROOM 5
- Follow map => ROOM 7

ROOM 5;

"Okay I really need to find that key now…"

- Check in dresser => ROOM 1 pt 2
- Open red door => you die
- Follow map => ROOM 7

ROOM 6;

"An attic. It's like she wanted me to die!"

- Get out of there quick => ROOM 7
- Look around => die
- Open window => die

A choose your own adventure game

The interactive story program – continued:

ROOM 7;

"Will this madness ever stop?"

- Open attic hatch => ROOM 6
- Open red door => die
- Open orange door => ROOM 9

ROOM 9;

"My aunt was a witch! Well, this explains a lot."

- Open red door => die
- Open pink door => die
- Bury this harrowing family secret and run back to the hallway => ROOM 7 pt 2

ROOM 7 pt. 2;

"What the heck is that!?"

- Investigate the strange symbol on the wall => ROOM 8
- Open attic hatch => ROOM 6
- Open red door => die

ROOM 8;

"I'm trapped here forever. This is the end, I know it."

- Open blue door 1 => die
- Open blue door 2 => die
- Open blue door 3 => VICTORY
- Open blue door 4 => die

A choose your own adventure game

The interactive story program – continued:

Bibliography:

- https://processing.org/reference/ (used for modules that lack a reference)

Common errors removed by:

- Testing the program using a range of inputs.
- Fixing names to solve syntax errors
- Changing problematic code to solve execution errors
- Rewriting entire sections of the code to solve logical errors

Uncommon errors and solutions found:

- **Processing.app.SketchException: Badly formed character constant (expecting quote, got):**
  - Error caused by a mistake in commenting.
  - The error occurs on older versions of Processing.
  - Error fixed by making sure that all multi-line comments are closed.
- **Processing.app.SketchException: Badly formed character constant (expecting quote, got ,):**
  - Error caused by having a "," placed after a semi colon
  - Error fixed by removing the "," – it is odd that it did not count as a syntax error.
- **Node out of range / node not found:**
  - Error caused when you exit the sketch and the sound library is still running.
  - The error only occurs for the sound library by the Processing Foundation.
  - Error fixed by switching to the superior minim library.
- **Don't know the ID3 code TSS / Error parsing ID3v2: String index out of range:**
  - The minim library cannot read the tags in audio files.
  - Error fixed by getting a tag editor called Mp3Tag and removing all tags.
- **Java.lang.NullPointerException:**
  - Error caused by an invalid reference to something.
  - Error fixed by not referencing things that are not alive yet. So I basically added more conditions to how things are run.
- **Font config error**
  - Error caused by an issue with Red Hat Linux and Google fonts.
  - The error has nothing to do with the Processing program and is hence unfixable.
- **Java.lang.StackOverflowError:**
  - Issue with code to mute sounds. The code kept muting the same sound.
  - Error fixed by not looping the mute sounds function indefinitely.

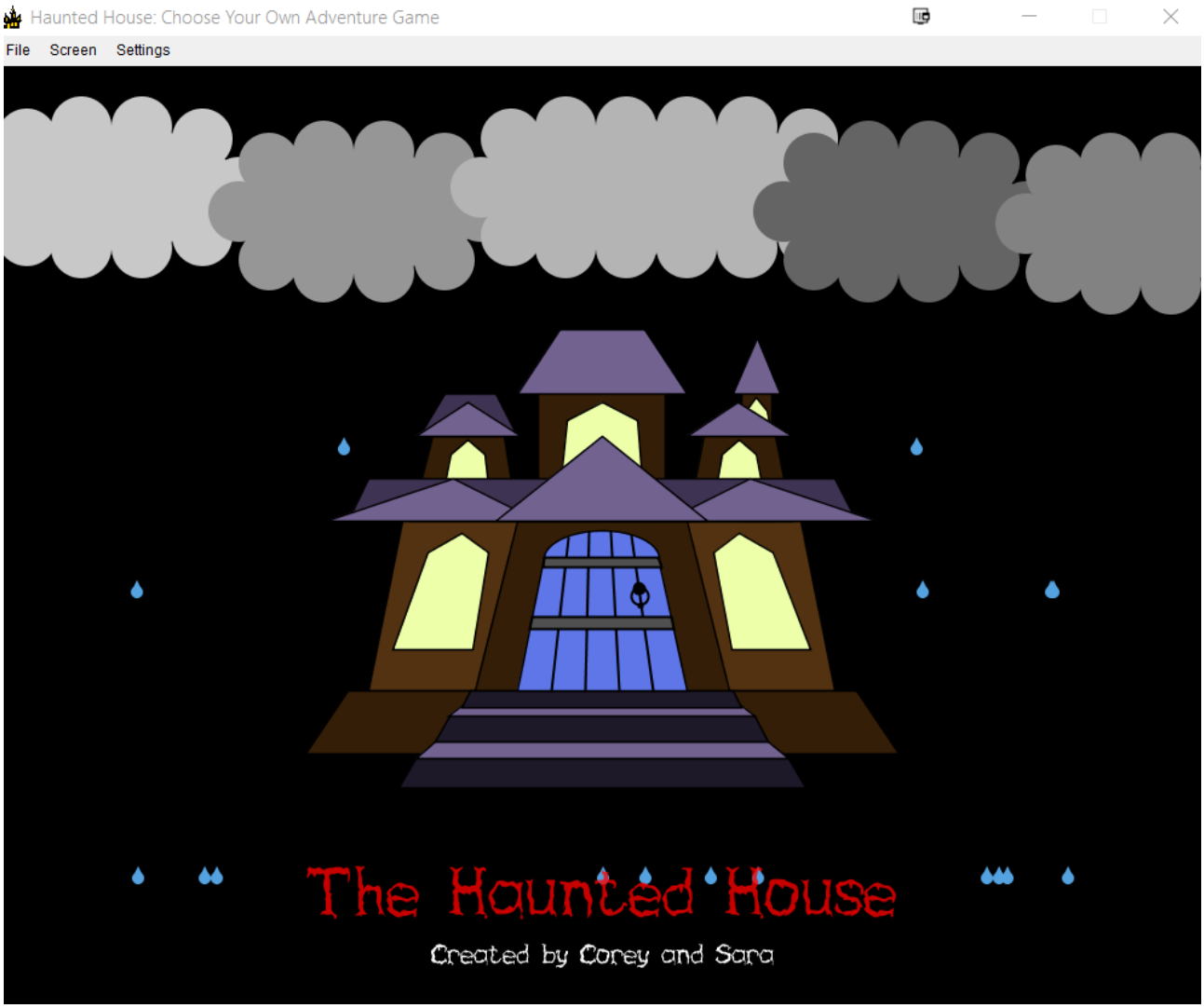
A choose your own adventure game

The interactive story program – continued:

Lag removed by:

- Limiting the number of elements running on the screen at the same time.
- Minimizing the number of loops that run every time the draw() function is run
- Fixing issues with the font loader - not loading font files every time the draw function is run
- Fixing issues with the sound library – limiting the number of sounds that play
- Improving the code in general, particularly the classes for the animated elements

An image of the final program:



A choose your own adventure game

The interactive story program – continued:

Evidence of test cases:

Global conditionals:

- There must be no execution errors.
- The data structures for the canvas size must equal the canvas size (error checking function).
- The game text file must not have any errors (error checking function).

Test Case 01 – "The jump scarer"

Inputs:

- Mouse click input.
- **The data structures:**
    - The jump scare counter (which increments for each mouse click)
    - The jump scare probability
    - The jump scare setting for displaying the scare.
    - The jump scare duration.
    - The jump scare click away setting.
- The current program time.

Conditionals:

- **Specific to the test case:**
    - The user must be clicking on the screen for the counter to increment.
    - The jump scare must be currently inactive for it to display.
    - The mouse click counter must equal the jump scare probability.
    - A timer should run for the current program time to the current program time + the duration of the scare before changing a setting, that allows you to skip the jump scare.

Expected outputs:

- **The screen should display:**
    - The jump scare monster
    - The jump scare text
- **The program should run:**
    - The function to play the jump scare SFX

Outputs:

- The expected output.

A choose your own adventure game

The interactive story program – continued:

Test Case 02 – "The menu bar"

Inputs:

- Mouse click input.
- **Libraries:**
    - The menu bar library.
- **Data structures:**
    - The menu bar
    - The menu
    - The menu item
    - The menu listener objects
    - The screen the user is currently on
    - Various settings that are not directly relevant to this test case.

Conditionals:

- **Specific to the test case:**
    - Make sure the user input is associated to a particular menu item / label.
    - If the user clicks a particular menu item / label, run the code that refers to this.
    - A conditional may be used to run particular commands if they are on a screen.

Expected outputs:

- **The screen should display:**
    - A menu bar that resizes with the canvas size automatically.
    - Menu bar file menus that expand on click.
    - Menu bar items within menu bar file menus that run particular lines of codes.
    - Particular menu bar items should run the desired lines of code.

Outputs:

- The expected outputs.

A choose your own adventure game

The interactive story program – continued:

Evidence of test cases - continued:

Test Case 03 – "the name maker animation"

Inputs:

- The text settings for the name.
- **Data structures:**
    - A timer so that I can generate names after a specified amount of time has elapsed
    - An amount to increment the timer by
    - A placeholder to store the names
    - The width of the canvas
    - A controller for whether or not names are currently being drawn
    - A placeholder to store the number of the name.
    - A placeholder to store the randomly generated width.
    - A placeholder to store the randomly generated height.
- The current program time.

Conditionals:

- **Specific to the test case:**
    - Make sure that the program time is equal to the time to start displaying names.
    - Only display a name if a name is not drawn.
    - The names must not display where the house is.

Expected outputs:

- **The screen should display:**
    - The program should generate different names and display those names.
    - The names should display at varying X and Y positions.
    - The names should display at either the left or right side, away from the house.
    - The names should be red and of a sufficiently large size.
    - Each time the program displays a name it should clear previous names.
    - The program should display one name at a time without moving set positions.

Outputs:

- The expected outputs.

A choose your own adventure game

The interactive story program – continued:

Test Case 04 – "the text based buttons for the navigation system"

Inputs:

- Mouse click, key press, mouse wheel input.
- **Data structures:**
    - The text file to import.
    - A text and number placeholder to store the current line.
    - The various constants / indexes that ensure that a text button refers to a line.
    - Information about what screen the user is currently on.
    - Button information – how many buttons, what colour are they, and where are they
    - Current text vertical position input so that you can scroll.

Conditionals:

- **Specific to the test case:**
    - The user must be on the playing screen before the navigation system runs.
    - Hitting the escape key will result in the key not terminating the program.
    - Scrolling conditions ensure that the text moves in the direction of the scroll.
    - The buttons must not exist for them to be drawn.
    - Hitting any key will take you to an overview of the text file screen.
    - The starting position must restart if you lose, or otherwise return to the start.
    - A conditional prevents the text from scrolling if the text file is short.
    - The command entered must run the associated command.
    - Conditionals to ensure that functions run with the desired pointer.

Expected outputs:

- **The screen should display:**
    - **The information in the text file:**
        - The entered question should appear on the screen.
        - The entered buttons (if there are any) should appear
        - The entered button-to-label text should lead to a label.
        - The entered links for the commands should also work on click.
    - The pointer should display the current line number.
    - Hitting a key should give you an overview of the text file. Clicking on a line should change the pointer to that particular line and give you a normal view of the text.
    - All of the set text settings and colour settings should be correct.

Outputs:

- The expected outputs.

A choose your own adventure game

The interactive story program – continued:

Evidence of test cases – continued:

Test Case 05 – "the sound muting functions"

Inputs:

- Mouse click
- **Data structures:**
    - The mute sound setting
    - The filename for the blank sound file.

Conditionals:

- **Specific to the test case:**
    - The user must click the menu bar item to mute the sound.
    - If sound is playing – play a silent sound instead and set future sounds to not play.
    - If sound is not playing set the future sounds to not play.

Expected outputs:

- **The program should run:**
    - Whenever the user clicks the mute sounds button all future sounds should not play.
    - Whenever the user clicks the unmute sounds button all-future sound should play.

Outputs:

- The expected outputs.

A choose your own adventure game

The group presentation – loose transcript:

Introduction:

Hello everyone here, feel free to interrupt and ask questions at any time. We have a lot of information that we may not cover – but that does not matter, as we will still cover enough to give you plenty of information about our program and approach to the task.

Description of the artwork:

Our artwork is an interactive text based game with a horror theme.

Description of the design:

We designed the program to challenge the player's memory. In order to complete the game, the player must choose from up to four possible pathways in each room. If the player dies, the game resets, and they probably forget what they clicked last. However, to win the legit way you must remember your choices. From the beginning, we developed the game for an audience of young children from the age range of about 8 to 15. We highlighted the intended age range for the game by using less scary images. In order to give the game a haunted atmosphere we used creative commons or public domain horror-themed music, sound effects, and graphics. We kept the text short and succinct and tried to make the game as interactive as possible. Our program generally refers to a number of clichés from horror films, books, and games, and has all of the mechanics that an interactive story needs.

Description of the code:

I will try to give you the general gist of how the program works while discussing key parts to the code.

The graphics: To create the graphics, we used a combination of draw functions – like **rectangle()**, **ellipse()**, **beginShape()**, **quad()**, **line()**, **triangle()**, and **arc()**. Note that the **image()** function saved a lot of time as we could not draw everything. The use of the **translate()**, **rotate()**, and **scale()** functions also made working with graphics easier as we could change groups of elements easier. We also used functions to manipulate what we drew – like **fill()**, **stroke()**, and the shape and image align functions. Note that Adobe Photoshop was used to edit almost all of the graphics – darkening backgrounds, cropping borders, making the monsters look better, adding elements to the background, etc.

The text: The text for the game simply uses text functions like the **createFont()**, **textFont()**, **textSize()**, and **textAlign()** functions. We used one custom font file for consistency but kept the text interesting by using white and red text, along with varying text sizes.

The sound: We originally used the sound library by the Processing Foundation but I realised that it handled sound files poorly and had many limitations / errors so I imported the **minim library** instead. However, the minim library's documentation was outdated and many useful features did not work on the latest version of Processing like the ability to loop sounds, mute sounds, etc. To solve the issue of the default mute function not working, I had to separate the sounds to two audio players – a sound effects and backgrounds one, which each have a separate function to play audio files. The function only plays the sound file if the setting to mute sounds is off. To solve the issue of not being able to loop sounds, I used a timer to check if the song has finished playing and then I would play it again. To solve the issue of sounds still playing when you switched screens, I had to load a blank sound file.

The navigation system: I separated the code for various parts of the program using **states**. I.e. The program only displays a particular screen or type of element on a particular screen if the settings state that it should. In order to make the navigation system, navigable, I imported a few java libraries that allowed for a **menu bar**. The **menu bar** checks if the user has clicked a particular label and then runs the associated code for this label. Note that changing states or otherwise changing what is drawn required clearing of the canvas and audio, along with a reset of the settings so that the program resets.

The settings: I set various settings for the program using a range of data structures – like **Boolean**, **integer**, **String**, and **float**. The menu bar can change a few of these settings but some of them are constant. The settings do different things like changing which animation is drawn, loading a screen again, muting sounds, increasing the probability of a jump scare, etc.

The text based navigation system: The text based navigation system works by breaking up a text file into components – such as the label of the line, the question to be displayed, the names of the buttons, what the buttons lead to, and the name of the function to be run. A set of buttons is also created according to the information in a given line of the text file. A pointer is used to navigate between lines and switch between the backgrounds. Mouse click input is then checked to see if the mouse position is on the button. Note that changing the pointer position leads to different backgrounds being loaded.

Important thing to say: Note that I had some help from Chrisir, a user on the Processing forums. I posted my attempted solution to the text navigation system and asked if anybody could suggest some improvements to what I tried. Chrisir realised that my solution was awful and was kind enough to provide a coded solution for myself and other people who visit the forums. Note that Sara wrote the text for the system and made the backgrounds. Note that I made some edits to the system to adapt it for the purposes of the program like allowing the system to switch backgrounds.

Custom classes / functions: Many aspects to the code were separated into custom classes and functions. A few classes include the lightning by Corey, trippy sky by Corey, rain by Corey, haunted house by Sara, and so forth. Generally, the animated classes use the program time and integer data structures to time the animated events. Different functions are called depending on if the animation is being displayed or updated, or otherwise manipulated. A few of the custom functions include a font loader, audio player, canvas clearer, etc. There is a function to load each screen and draw each screen. Using custom classes and functions made the program easier to debug and read.

Aesthetic touches: I added in a frame title, frame icon, and mouse cursor.

The jump scarer: Mouse click input is counted and has the probability to trigger a jump scare.

Testing:

Testing involved interacting with the latest feature added in the intended way (and if possible the wrong way too as this would often throw up the most errors). Note that many errors were prevented by developing aspects to the program individually and then merging them one-by-one into larger parts. However, numerous errors were still encountered. The straightforward syntax errors were the easiest to fix. Nonetheless, several of the execution errors failed to tell the line that needed to be fixed which made the testing process significantly more difficult. The most dodgiest error was an error that told me that there was an issue with a single, unidentified line in the code as it was missing a "quotation mark". However, the actual issue was the comments turning into blue code at random intervals – so it was hard to notice this particular problem. In the end, the issue was fixed by removing the problematic section of commenting.

Conclusion:

Thanks for listening!