

## Discovering / Understanding

This assignment will require a more advanced understanding of how to link varying rooms in more creative ways than we have seen in the past. We will need to determine how the rooms are similar and different and how to develop a structure that effectively reflects their attributes. I will need to understand how to better design the game so that the development process does not become too cumbersome to complete in a relatively short amount of time.

I would like to come up with a creative way for the user to interact with the environment since we have a bit more freedom with this assignment. Included in the structure I will need to include how the user will play the game, as we will need the user to be able to collect different items and interact with a more complex room structure.

## Design

Because this design process includes the structure of the “game board”, i.e. the linked rooms, how the user interacts with the game, and how the user accomplishes the goal of the game, portions of my design include many notes that show the plan that I have for the game. The design is more abstract than previous designs I have made, as it is more based on the structure and purpose of the game, with an assumption that I will be able to implement the game in code more effectively than I have in the past.

Thus far, the game I have designed involves the following:

The user is a parasitic brain worm; the rooms are the body parts of the infected host.

3 types of rooms:

extremity (foot/hand)

limb (leg/arm)

torso (stomach, chest, etc.)

This shows the hierarchical structure of the classes that I will eventually implement as room objects:

Body part

Extremity - Collect tokens

Foot

Right

Left

Hand

Right

Left

- Limb - contains clues to solve the game puzzle
  - Leg
    - Right
    - Left
  - Arm
    - Right
    - Left
- Torso - must infect all three in order to advance to blood
  - Stomach
    - enter 3 times to get symptom - vomiting
  - Chest
    - enter 3 times to get symptom - coughing
  - Neck
    - enter 3 times to get symptom - headache
- Mouth
  - has\_token?
- Blood - must enter brain to win, can only be done from blood
  - i.e. the user parasite must "cross the blood-brain barrier"
- Brain

This shows a slightly more advanced/detailed level of the design process of the game. It includes some of the shared attributes and functions of the varying classes:

- Body part
  - Four links/pointers to other body parts
  - print\_infected
  - display\_symptoms
- Extremity
  - has\_token
- Limb
  - is\_infected
- Torso
  - symptom\_active ("collected" by parasite)
- mouth
- blood
  - cannot be entered until the all torso symptoms are active
- brain

Directions for the parasite are considered as follows, but will be relatively arbitrary since it is a host body that we're talking about, rather than rooms in normal cardinal space:

NESW

Start in right foot – this is where the parasite (the user) enters the body (the room

structure)

right foot

N: right leg

E: (right hand) (locked / invisible?)

left foot

(token located here)

N: left leg

W: (left hand) (locked / invisible?)

right leg

S: (right foot)

N: stomach

E: left leg

left leg

S: (left foot)

N: stomach

W: right leg

(token required to enter stomach)

stomach

W: (right leg)

E: (left leg)

N: chest

chest

S: (stomach)

N: neck

W: right arm

E: left arm

right arm

S: (chest)

N: right hand

left arm

S: (chest)

N: left hand

right hand

S: (right arm)

N: mouth

E: blood (always locked)

W: right foot

left hand

S: (left arm)

N: mouth

W: blood (always locked)

E: left foot

(You must visit both hands in order to enter the mouth [must pay a token to enter the mouth, get a token on every other hand visit {This function is turned off once the mouth is reached}])

mouth

W: (left hand)

E: (right hand)

N: neck

S: blood

neck

(find a token here)

S: (mouth)

N: brain

E: blood

blood

W: (right hand) (not locked)

E: (left hand) (not locked)

S: (neck)

N: brain

(Pay a token)

brain

win!

(If you try for the brain, you must spend a token in order to access the secret password entry. Another token appears in the left foot.)

In order to access the brain, you must enter the determinant of the displayed matrix, which is generated randomly

Elements to include:

fluffy the parasitic brain worm!

immune system too strong! You can't enter the brain!

age of the host as difficulty level?

In order to implement the design, the bulk of the design will take the form of a class hierarchy, which will need to include a player class that will store elements of current game progress. I will need to add functions that will allow the user to traverse room to room, check different game statuses to ensure proper actions occur at each stage of the game, and correctly store information as the user discovers more within the game.

## Test Plan

- Ensure the player is created and initialized correctly
- Ensure each room link is accurately followed by a user's traversal
- Ensure tokens appear correctly
- Ensure tokens are collected correctly
- Ensure tokens are spent correctly
- Ensure rooms are correctly locked
- Ensure rooms are correctly unlocked
- Ensure symptoms are caused correctly
- Ensure symptoms are "collected" correctly
- Ensure the clues are collected correctly
- Ensure the clues are displayed correctly
- Ensure current room is tracked correctly
- Ensure room messages are displayed correctly
- Ensure the matrix puzzle is created correctly
- Ensure the determinant is determined correctly
- Ensure the user's input is validated correctly:
  - Choosing to display user information
  - Choosing room to travel to
  - Choosing to use a token to do an action
- Ensure the user's answer allows for a win once in the brain

## Implementation

Below I have attached maps of the implemented room relations. The first map shows how the rooms link together and what the underlying structure is. The second map reflects the game's implementation over the map, showing how rooms are accessed and more detailed information about the rooms. In implementing the design, as far as gameplay is concerned, I changed a few of the room relations to make the gameplay a bit more straightforward. My decisions in implementing singly-linked and doubly-linked rooms are best reflected in the maps below as well.

I originally thought that locked rooms would first have a null pointer that would be replaced by a pointer to the correct room once unlocked. While implementing the design, I decided to have all room connections visible to the user, as updating the links would be more confusing for both the player and for development. Once the user tries to enter a locked room, if the room's "is locked" attribute is true, the requirement to enter the room is displayed to the screen. The varying directions and varying locked rooms required quite a bit of "ad hoc" code to ensure correct behavior. Given more time, I would hope that I could come up with a more straightforward structure for the code, perhaps using functions to test different lock scenarios.

The clues became a much more necessary element to the game as I got further in the development process. The game became more challenging in the implementation than I had originally designed for, which I was happy about. I originally intended the clues to be relatively unnecessary hints, more telling of the gameplay itself, hinting towards the final solution. In the end, using the clues becomes *almost* necessary in order for a player to win the game. I liked that while developing the game, the straightforward play gave way to a truer puzzle.

I decided to remain rather enigmatic about what the final solution should be, which requires a bit of math knowledge. I decided to keep the puzzle harder to solve, as the parasite's final goal to infiltrate the entire host requires "crossing the blood-brain barrier," so I thought it appropriate to challenge the parasite's brain.

In addition to the designed class hierarchy for the body, I used a player object to store information that was relevant for gameplay regardless of the user's current room. I used additional functions to print the current body part's linking body parts, to travel from body part to body part, to display the menu for room traversal and menu for player information display, to process the user's menu selection, and to display the map of the broad game board structure as the user advances through the body.

## Testing

Ensure the player is created and initialized correctly	<input checked="" type="checkbox"/>
Tokens – start as 0 tokens	<input checked="" type="checkbox"/>
Clues – start as empty vector	<input checked="" type="checkbox"/>
Symptoms – start as empty vector	<input checked="" type="checkbox"/>
Known map – start with feet only	<input checked="" type="checkbox"/>
Ensure each room link is accurately followed by a user's traversal	<input checked="" type="checkbox"/>
Right foot     -> Right leg	<input checked="" type="checkbox"/>
Right leg     -> Right foot	<input checked="" type="checkbox"/>
-> Left leg	<input checked="" type="checkbox"/>
-> Stomach (initially locked/pay token prompt)	<input checked="" type="checkbox"/>
Left leg     -> Left foot	<input checked="" type="checkbox"/>
-> Right leg	<input checked="" type="checkbox"/>
-> Stomach (initially locked/pay token prompt)	<input checked="" type="checkbox"/>
Stomach     -> Right leg	<input checked="" type="checkbox"/>
-> Left leg	<input checked="" type="checkbox"/>
-> Chest	<input checked="" type="checkbox"/>
Chest       -> Stomach	<input checked="" type="checkbox"/>
-> Right arm	<input checked="" type="checkbox"/>
-> Left arm	<input checked="" type="checkbox"/>
-> Neck	<input checked="" type="checkbox"/>
Right arm   -> Right hand	<input checked="" type="checkbox"/>
-> Chest	<input checked="" type="checkbox"/>
Left arm     -> Left hand	<input checked="" type="checkbox"/>
-> Chest	<input checked="" type="checkbox"/>
Right hand   -> Right arm	<input checked="" type="checkbox"/>
-> Mouth (initially locked/pay token prompt)	<input checked="" type="checkbox"/>
-> Blood (initially locked/weaken host prompt)	<input checked="" type="checkbox"/>
-> Right foot	<input checked="" type="checkbox"/>
Left hand    -> Left arm	<input checked="" type="checkbox"/>
-> Mouth (initially locked/pay token prompt)	<input checked="" type="checkbox"/>
-> Blood (initially locked/weaken host prompt)	<input checked="" type="checkbox"/>
-> Left foot	<input checked="" type="checkbox"/>

Neck	-> Mouth (initially locked/pay token prompt)	<input checked="" type="checkbox"/>
	-> Blood (initially locked/weaken host prompt)	<input checked="" type="checkbox"/>
	-> Chest	<input checked="" type="checkbox"/>
Mouth	-> Neck	<input checked="" type="checkbox"/>
	-> Right hand	<input checked="" type="checkbox"/>
	-> Left hand	<input checked="" type="checkbox"/>
Blood	-> Neck	<input checked="" type="checkbox"/>
	-> Right hand	<input checked="" type="checkbox"/>
	-> Left hand	<input checked="" type="checkbox"/>
	-> Brain (initially locked/pay token prompt)	<input checked="" type="checkbox"/>
Brain	-> Blood	<input checked="" type="checkbox"/>
	(prompt for question display payment)	<input checked="" type="checkbox"/>
Ensure tokens appear correctly		<input checked="" type="checkbox"/>
One token in left foot (does not reappear)		<input checked="" type="checkbox"/>
One token generated every time both hands are visited		<input checked="" type="checkbox"/>
Ensure tokens are collected correctly		<input checked="" type="checkbox"/>
Token removed from room		<input checked="" type="checkbox"/>
Token added to player token count		<input checked="" type="checkbox"/>
Ensure tokens are spent correctly		<input checked="" type="checkbox"/>
Token removed from player token count		<input checked="" type="checkbox"/>
Room unlocked/Question displayed		<input checked="" type="checkbox"/>
Ensure rooms are correctly locked		<input checked="" type="checkbox"/>
Stomach		<input checked="" type="checkbox"/>
Mouth		<input checked="" type="checkbox"/>
Blood		<input checked="" type="checkbox"/>
Brain		<input checked="" type="checkbox"/>
Ensure rooms are correctly unlocked		<input checked="" type="checkbox"/>
Stomach (one token payment)		<input checked="" type="checkbox"/>
Mouth (one token payment)		<input checked="" type="checkbox"/>
Blood (all symptoms collected/caused)		<input checked="" type="checkbox"/>
Brain (one token payment)		<input checked="" type="checkbox"/>
Ensure symptoms are caused correctly (game board correct)		<input checked="" type="checkbox"/>
Stomach (once visited 3 times, Vomiting collected/caused)		<input checked="" type="checkbox"/>
Chest (once visited 3 times, Coughing collected/caused)		<input checked="" type="checkbox"/>
Neck (once visited 3 times, Headache collected/caused)		<input checked="" type="checkbox"/>
Ensure symptoms are "collected" correctly (stored in player object)		<input checked="" type="checkbox"/>
Stomach (once visited 3 times, Vomiting collected/caused)		<input checked="" type="checkbox"/>
Chest (once visited 3 times, Coughing collected/caused)		<input checked="" type="checkbox"/>
Neck (once visited 3 times, Headache collected/caused)		<input checked="" type="checkbox"/>
Ensure the clues are collected correctly		<input checked="" type="checkbox"/>
Each limb contains a clue:		<input checked="" type="checkbox"/>
Right arm ("...")		<input checked="" type="checkbox"/>

Right leg ("...")	<input checked="" type="checkbox"/>
Left arm ("...")	<input checked="" type="checkbox"/>
Left leg ("...")	<input checked="" type="checkbox"/>
Ensure the clues are displayed correctly (stored in player object)	<input checked="" type="checkbox"/>
Collected only when corresponding limb is entered	<input checked="" type="checkbox"/>
Right arm ("...")	<input checked="" type="checkbox"/>
Right leg ("...")	<input checked="" type="checkbox"/>
Left arm ("...")	<input checked="" type="checkbox"/>
Left leg ("...")	<input checked="" type="checkbox"/>
Ensure current room is tracked correctly	<input checked="" type="checkbox"/>
Ensure room messages are displayed correctly	<input checked="" type="checkbox"/>
Ensure the matrix puzzle is created correctly	<input checked="" type="checkbox"/>
Randomly generated matrix elements	<input checked="" type="checkbox"/>
Displayed to user correctly	<input checked="" type="checkbox"/>
Ensure the determinant is calculated correctly	<input checked="" type="checkbox"/>
$\begin{vmatrix} a & b \\ c & d \end{vmatrix} \quad \text{determinant} = a*d - b*c$	
Ensure the user's input is validated correctly:	<input checked="" type="checkbox"/>
Choosing to display user information	<input checked="" type="checkbox"/>
Choosing room to travel to	<input checked="" type="checkbox"/>
Choosing to use a token to do an action	<input checked="" type="checkbox"/>
Ensure the user's answer allows for a win once in the brain	<input checked="" type="checkbox"/>
Display congratulations	<input checked="" type="checkbox"/>
Exit program	<input checked="" type="checkbox"/>



This is a basic map of the structure of the rooms, i.e. body parts. Once every room has been unlocked by varying means, this map is an accurate depiction of the links between the rooms.

The map below is a more accurate depiction of the game board, including what things are located where and how the user accesses different rooms.

An inventory of the game elements:

The user can “collect”:

- Tokens
- Clues
- Symptoms

During a turn, the user can view:

- Token count
- Clues
- Symptoms caused
- Known map (ASCII representation of where they have visited, symmetry is assumed)

Some rooms are locked. Rooms are unlocked by:

- Paying tokens
- Causing symptoms

In order to win (take over the whole host body) the user must cross the blood-brain barrier. This is done by solving a puzzle.

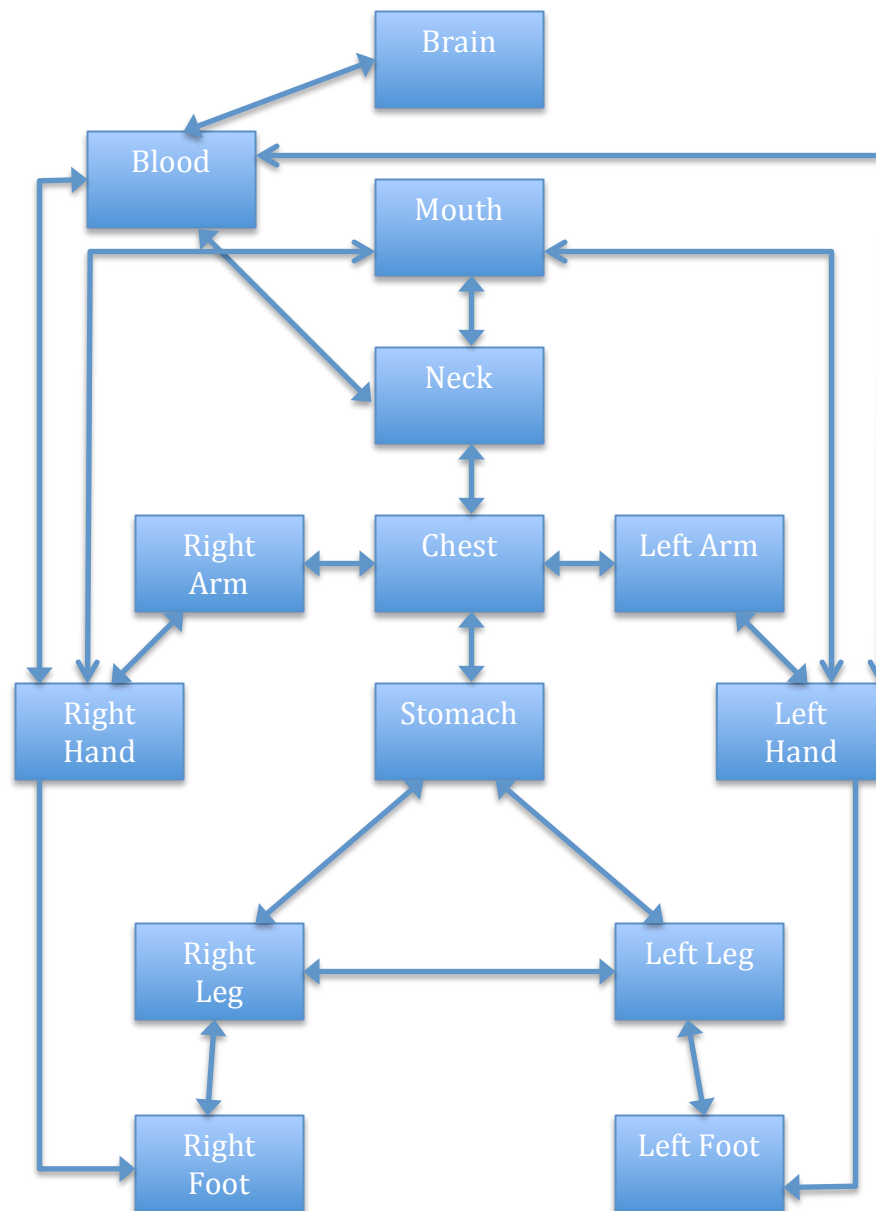
The clues reference:

- Where to look for tokens
- How to generate tokens
- How to cause symptoms
- How to get the final integer
- Answer




Gameplay strategy:







- Pay a token to unlock the mouth in order to make token generation easier

- Pay close attention to the capitalized words in the clues

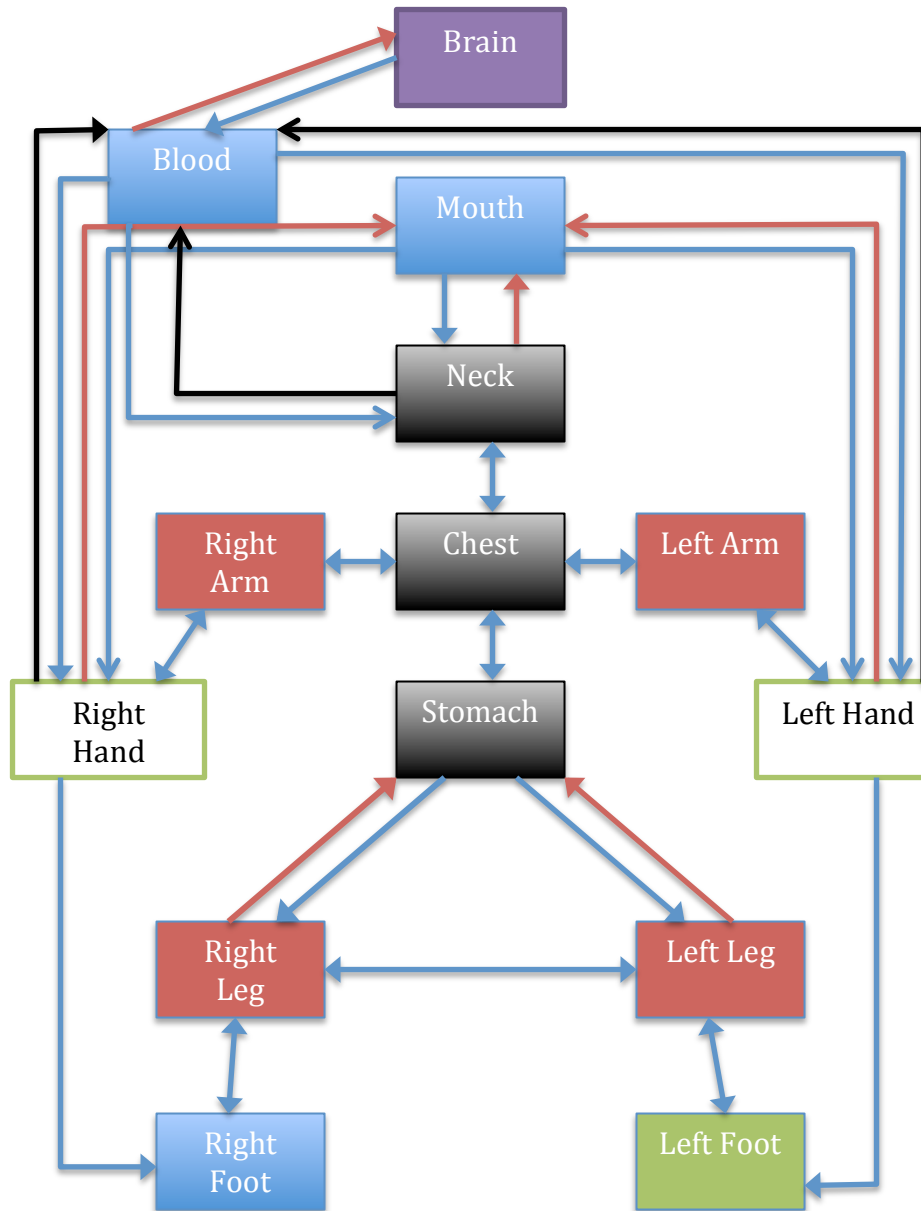


## MAP KEY

-  Normal travel between rooms
-  Must use a token for initial entry
-  Must activate all symptoms for initial entry

-  Part Normal room
-  Part Contains one token
-  Part Every time *both* hands have been visited, a token will appear
-  Part Contains one clue
-  Part The user can cause a symptom in this body part by entering the room 3 separate times
-  Part Pay a token to enter an integer answer to the final puzzle.

The user is presented with a 2 x 2 matrix. If they enter the determinant of the matrix, the user wins



## Reflection

Designing and implementing this program was one of the most rewarding programs that I have developed thus far. This program required the majority of the skills that I have developed over the previous 20 weeks of doing introductory work in C++. This program also provided the feel of somewhat of a capstone project for the introduction to C++, as we had a maximum level of freedom in creating a program from scratch. I reused a small amount of code from my program in Lab 8 of this course, which allowed me to develop the structure of the game board and quickly start traversing room to room, making the core game development less daunting.

It's interesting to see the minimal amount of coding required to create such a game structure. A program can provide rather complex play after developing for a relatively small amount of time with a relatively small amount of C++ knowledge. I am very excited to see what developing a graphical user interface and creating a platform on which to create other games entails. I know that my code must be relatively messy, involving roundabout ways of accomplishing certain tasks that I am sure are easier to code using fewer lines for a more advanced programmer. I am excited to continue growing in my programming development so that I can see my work with this program and better understand how to most effectively implement it.

This program's use of more variables of greater complexity allowed me to see how to use simple tools to create functioning complex structures in C++. For large portions of this program I found myself creating ad hoc solutions to problems that I would have preferred more transferable solutions. In testing for different types of rooms' locks from different directions, I ended up using many analogous if-else structures that tested for analogous, but not exact, conditions of the room before letting the user traverse. I think with greater time I could implement a better solution; I assume, in fact, that I already know a much better solution, but as I was already at a certain stage of development, I could not estimate whether the time cost of implementing it would benefit me. In the end, I think in this instance it would have.

This project offered a better perspective of our abilities than past assignments due to the degree of freedom we had in its design and implementation. As I get farther in my programming education, I am more and more eager to work on projects that have even greater degrees of freedom. It seems like creating more and more complex projects of your own design is the most effective (read: only) way of truly understanding any programming language.