**Software Engineering**

**Year 11 , 2025**

**Assessment Task 2**

**Object-Oriented Programming Assignment:**

**“Hunting Wumpus”**

**By: An Tran**

Contents

[Software Requirement Specification 3](#_kgw5ohvm64x6)

[Gantt Chart 4](#_co78xohu4nx8)

[Budget 5](#_lwcuzcmk73qs)

[Justification of Technology 6](#_8yorg1azlo4x)

[Design 7](#_bcmy72q4uopk)

[UML Notation 8](#_9mu59y944cqj)

[Class Diagram 9](#_fs29qnnxjtag)

[Flowchart 10](#_hdsr3syal5rh)

[Pseudocode 11](#_jef34zp9qv9n)

[Sequence Diagram 12](#_h02xxpaew1ui)

[Graphical User Interface GUI 13](#_qpr8bjujhlwq)

[Artificial Intelligence Conversion Code 14](#_36ah9omxpozp)

[Explanation of why Java is safer than Python or vice versa 15](#_vp2tztc35xvi)

[Security 15](#_96tq1tdzhvmq)

[Compiling and Execution 15](#_vz7hj6k7qptr)

[Storing data 16](#_9es95hc0ksyg)

[Encryption. 17](#_xo8g61x21czs)

[Why prototyping might be done in Python rather than Java. 18](#_3uv7bl1ehyji)

[What Tools were used in the development of this Project and their justification 19](#_3w9deko4qm3)

[Visual Code IDE 20](#_4d8czq9d4h4m)

[Python Compiler 21](#_cuo328dulok2)

[Artificial Intelligence Converter 22](#_heagytnllc9l)

[Java 23](#_8sdvna8kp6k6)

[Code is commented and following industry standard practices 24](#_2fc14vwu4p29)

[Justification of Git and GitHub and their difference 25](#_q7n3hyxr66y1)

[Frequency of committing Code 26](#_tk8xe4pbrash)

[Appendix 1 Python Code 27](#_4y9it8b56l4w)

[Readme File for Python 28](#_94daymcpbzuo)

[Appendix 2 Java Code 29](#_eh0ag43evd7y)

[Readme File for Java 30](#_6s3wjcf5c0n2)

[Reflection 31](#_dkuwziabjg2q)

# Software Requirement Specification

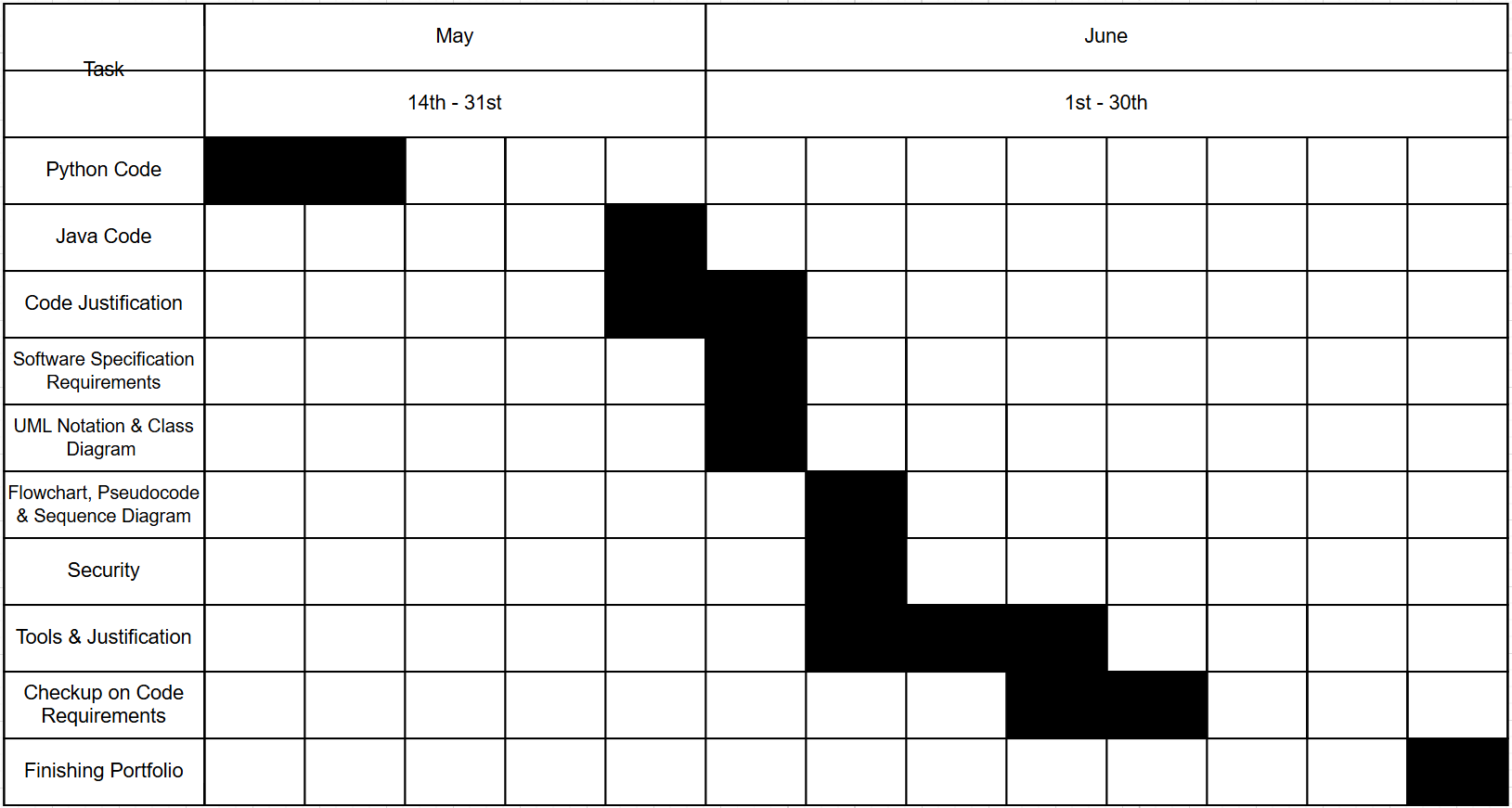
**Explain the Game. This can be found in the Programming Journal Attached to the Assignment**

Hunt the Wumpus is a 2D adventure game made in 1973. The objective is to hunt and kill the Wumpus, a dangerous creature hiding in a cave of 20 interconnected rooms, arranged like a dodecahedron. Each room connects to 3 others, and you have to navigate through pits, bats, and the Wumpus itself. You get one arrow to shoot the Wumpus with, and you can deduct where it is through game clues such as "I smell a Wumpus," "I feel a draft," or "Bats nearby".

## Gantt Chart

An accurate track record of what you did for the project.

This can be record from Git or GitHub.



## Budget

**You are a software engineer charging $60 per hour.**

**Get the time spent from GitHub and multiply by $60 per hour.**

6 hours in class  
10 hours at home

16 x $60 = $960

## Justification of Technology

**Why is Python chosen over other language such as Java. What IDE are you using and Why? What are the advantages over other IDE.**

Python is often chosen over other languages like Java because it is simpler to read and write, making it ideal for beginners like me and rapid development. It uses less code to accomplish the same tasks and has a vast standard library that supports everything from web development to data analysis. For Python, I use Visual Studio Code (VS Code) as my IDE. VS Code is lightweight, highly customizable, and supports debugging, and extensions for Python-specific tools. Compared to other IDEs like PyCharm or IDLE, VS Code is faster to load, supports multiple languages, and integrates well with Git and terminal commands, making it a versatile choice for many coding projects.

# Design

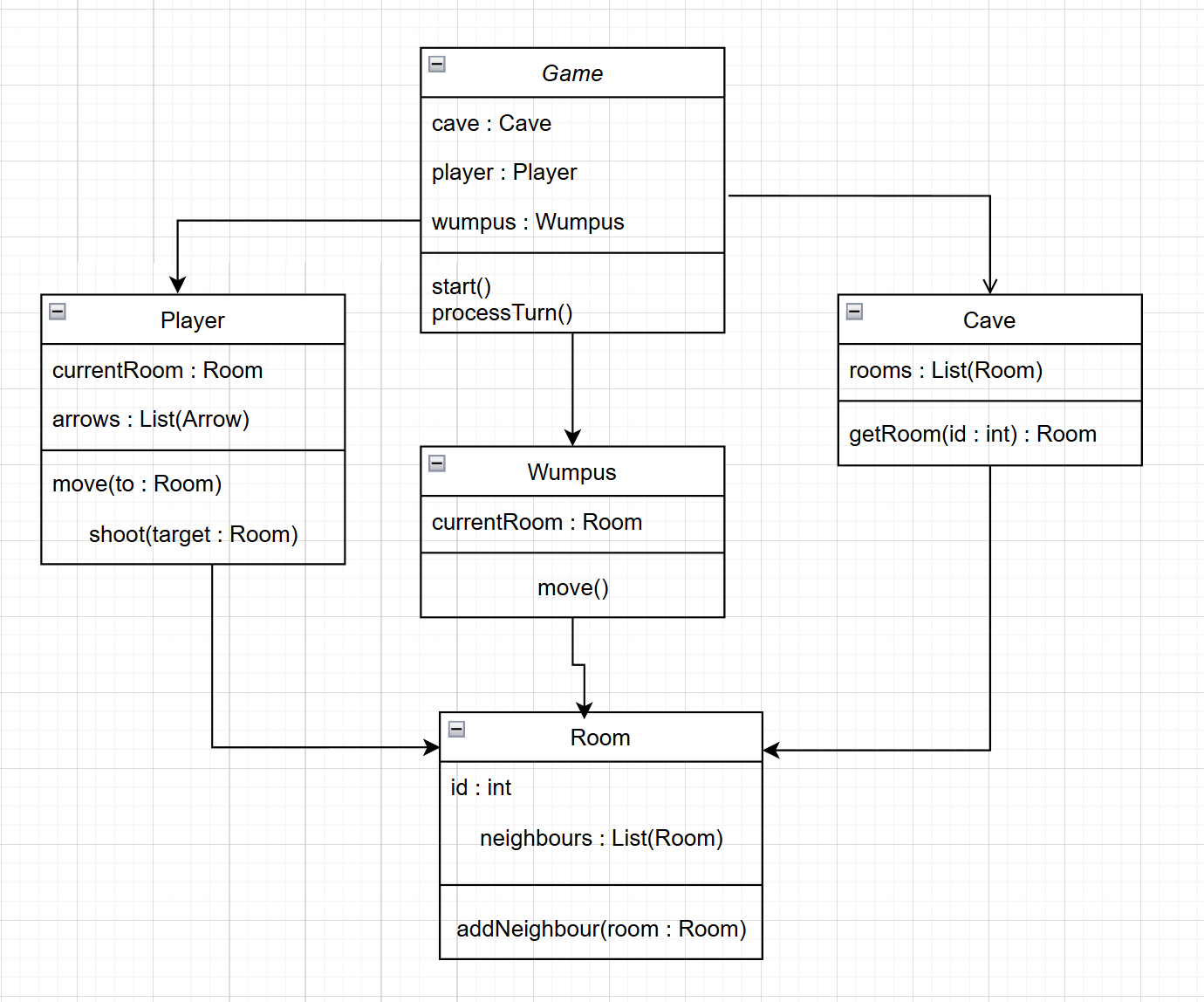
Here you need to insert such design elements as: UML Notation, i.e. Class Diagram, Sequence Diagrams, Flowchart and Pseudocode, Context Diagram and Explain, Graphical User interface so that a person who never played Wampus can understand how to play it. You can model this from draw.io website.

## UML Notation

**Why do we use UML Notation?**

People use UML notation because it’s a simple and standard way to draw pictures of how software works. It helps everyone involved understand the design clearly, even if they come from different backgrounds. UML diagrams make it easier to plan the software before writing code, which helps avoid mistakes. They also serve as helpful documents that teams can use later to fix or improve the program.

## Class Diagram



## Flowchart



## Pseudocode

START

INITIALIZE game settings and variables

LOAD images (bat, player, wumpus, arrow)

SETUP screen and font

DISPLAY instructions

WAIT for player to press Enter

FUNCTION populate\_cave():

RANDOMLY place player in cave

CALL place\_wumpus()

PLACE bats, pits, and arrows randomly avoiding player and other hazards

FUNCTION place\_wumpus():

REPEAT

PICK random room for Wumpus

UNTIL wumpus\_pos != player\_pos

FUNCTION place\_bat():

REPEAT

PICK random room for bat

UNTIL room not occupied by player, Wumpus, pits, or other bats

ADD bat position to bats\_list

FUNCTION place\_pit():

REPEAT

PICK random room for pit

UNTIL room not occupied by player, Wumpus, bats, or other pits

ADD pit position to pits\_list

FUNCTION place\_arrow():

REPEAT

PICK random room for arrow

UNTIL room not occupied by player, Wumpus, bats, or pits

ADD arrow position to arrows\_list

FUNCTION check\_neighbor\_rooms(position, item\_list):

FOR each adjacent room of position:

IF item in that room is in item\_list:

RETURN True

RETURN False

FUNCTION draw\_room(position):

CLEAR screen

DRAW brown circle for room background

DRAW exits (left, right, up, down) if available

IF Wumpus nearby:

DRAW red blood circle

IF pit in current room:

DRAW black circle (pit)

DRAW player image

IF bats in room:

DRAW bat image

IF Wumpus in room:

DRAW Wumpus image

DISPLAY current position and number of arrows

DISPLAY warnings if bats, pits, or Wumpus nearby

IF bats in room:

PAUSE briefly

FUNCTION check\_room(position):

IF player in Wumpus room:

CALL game\_over("Eaten by Wumpus")

ELSE IF player in pit room:

CALL game\_over("Fell into pit")

ELSE IF player in bats room:

DISPLAY message about bats carrying player

MOVE bats to new random position

MOVE player to new random position

ELSE IF player in arrow room:

DISPLAY message about finding arrow

INCREASE player arrows by 1

REMOVE arrow from list

FUNCTION move\_wumpus():

IF Wumpus can move AND random chance succeeds:

FOR each exit from Wumpus room:

IF exit not occupied by player, bats, or pits:

MOVE Wumpus to that exit

BREAK

FUNCTION shoot\_arrow(direction):

IF no arrows left:

RETURN False

DECREASE arrow count by 1

IF Wumpus is in adjacent room in direction:

CALL game\_over("Wumpus killed")

ELSE:

DISPLAY message about missed arrow

RELOCATE Wumpus randomly

IF no arrows left:

CALL game\_over("Out of arrows")

FUNCTION check\_pygame\_events():

IF quit event or ESC pressed:

EXIT game

IF arrow key pressed:

IF shift held:

CALL shoot\_arrow(direction)

ELSE IF exit exists in direction:

MOVE player to that exit

CALL move\_wumpus()

FUNCTION game\_over(message):

DISPLAY message on screen

WAIT briefly

EXIT game

MAIN LOOP:

WHILE True:

CALL check\_pygame\_events()

CALL draw\_room(player\_pos)

UPDATE display

CALL check\_room(player\_pos)

END

## Sequence Diagram

## Graphical User Interface GUI

The game starts on a black screen, with a brown circle on top. Your character is in the middle of the screen, and at the bottom of the screen it says the amount of arrows you have left.

When you approach nearby objects, the game will warn you.

**Wumpus:** A red circle will appear below your character, signifying that the wumpus is one move away in a direction.

**Bats:** A message will appear, saying “You hear the squeaking of bats nearby”. This signifies that bats are one move away in a direction.

**Pits:** A message will appear, saying “You feel a draft nearby”. This signifies that a bottomless pit is one move away in a direction.

**Controls:** Arrow keys move in respect to their direction.

To fire an arrow, hold shift and then arrow key in the direction you want to fire it in.

## Artificial Intelligence Conversion Code

**Explain what artificial intelligence engine you used to convert the Python Code into Java and were you successful and explain how software and hardware can be used in rapid software development.**

I used ChatGpt to convert my code from Python to Java and it was successful after I asked it to refine the code. Software and hardware can both play important roles in rapid software development. On the software side, modern development tools like integrated development environments , version control systems , and frameworks help speed up the coding, testing, and deployment processes. Code libraries and APIs also allow developers to reuse existing functions instead of building everything from scratch. On the hardware side, powerful computers with fast processors and more RAM allows for quicker code compiling, smoother multitasking, and faster testing.

# Explanation of why Java is safer than Python or vice versa

Address in terms of

# Security

### Compiling and Execution

Java is generally considered safer than Python in terms of security features, as Java runs inside the Java Virtual Machine (JVM), which adds an extra layer of protection by isolating programs from the system. It has strong type checking, bytecode verification, and automatic memory management, reducing the risk of buffer overflows or memory corruption. Java also provides a Security Manager that can limit what code is allowed to do (like reading files or accessing the network), which is useful in environments.

Python, while secure in many ways, is more flexible and dynamically typed, which can introduce vulnerabilities if developers are not careful. For example, Python's dynamic typing and reflection features can be misused, potentially allowing injection attacks.

## Storing data

Java is more structured and better suited for large or complex projects. It has built in tools like JDBC that help it connect with databases such as MySQL or Oracle, and it checks for errors before the program even runs. This makes it easier to spot mistakes early, like saving the wrong type of data. Java also has features like file reading, writing, and object storage, plus frameworks like Hibernate that make handling data even more reliable.

Python is more relaxed and easier to learn, which makes it great for smaller or quicker projects. It can store data in different ways, like saving to text files, CSVs, or databases using tools like sqlite3 or SQLAlchemy. Since Python is flexible and uses fewer lines of code, it’s faster to set up and test. But because it figures out data types while running (not before), it can sometimes let mistakes slip through, like saving the wrong kind of data without warning.

In terms of safety, Java is better at catching errors early and avoiding bugs that could cause security issues, like injection attacks. It also uses “prepared statements” to safely send data to a database.

## Encryption.

Java is often used in big systems and has strong built-in support for encryption through its Java Cryptography Architecture (JCA). This lets developers use secure features like hashing, digital signatures, and key generation. It’s designed to be strict and secure, which is helpful in situations where data protection is really important, like in banking or enterprise apps. Java also makes sure you follow proper steps when dealing with encryption, which helps prevent mistakes that could make your system vulnerable.

Python is more flexible and easier to work with when learning or building smaller projects. It has libraries like cryptography and PyCrypto that make it simple to add encryption to your code. While Python makes things quicker and easier to set up, it doesn’t guide you as strictly as Java does, which means there’s more room for errors if you don’t fully understand what you're doing.

Overall, Java is better for encryption in serious, secure environments because it’s stricter and more reliable. Python is great for learning and quick use, but you need to be extra careful to make sure you're using the right tools and following best practices.

## Why prototyping might be done in Python rather than Java.

Prototyping is often done in Python instead of Java because Python lets you write code much faster and with less effort. Its simple and clear syntax means you can turn your ideas into working programs quickly without worrying too much about complex details like data types or strict rules. This makes Python perfect for testing out new concepts or creating a basic version of an app to see if it works before spending time building the full thing.

Java, on the other hand, requires more setup and stricter code because of its static typing and detailed structure. While this makes Java better for building solid, large-scale applications, it slows down the early stages of development. It allows developers to experiment, get feedback, and make changes quickly before moving on to a more polished version in Java or another language if needed.

# What Tools were used in the development of this Project and their justification

## Visual Code IDE

For writing and compiling the code, Microsoft Visual Basic was chosen as the compiler and development environment. Visual Basic provides a user friendly interface with drag and drop features, making it simple to design graphical user interfaces and quickly test programs. It is especially useful for building Windows based applications and allows for rapid development with minimal coding effort. Using Visual Basic helped speed up the coding process and made debugging easier.

## Python Compiler

For writing and compiling the code, Python was the main programming language. Python’s interpreter acts as the compiler and executor, allowing for rapid testing and iteration without lengthy compile times. Its simplicity and readability speed up development, especially during prototyping. Additionally, Java was considered due to its strong type checking and security features, but Python was preferred here for its flexibility.

## Artificial Intelligence Converter

For converting the code to Java, I used ChatGpt as it is a very efficient AI capable of writing code, and I did several prompts so the converted code could work.

## Java

Java was considered due to its strong type checking and security features, but Python was preferred during the early stages for its flexibility and speed in prototyping. Once the main logic and gameplay were successfully tested and refined using Python, the project was later translated into Java to take advantage of its stricter structure, stronger error checking, and better performance for larger applications.

## Code is commented and following industry standard practices

The code in the project is well commented and follows industry standard practices such as consistent indentation, meaningful variable names, modular functions, and clear documentation. This makes the code easier to understand, debug, and maintain by others or future developers.

## Justification of Git and GitHub and their difference

The project’s code was hosted on GitHub, a web based platform that builds on Git by providing an easy interface for sharing code, managing issues, and reviewing contributions. The main difference is that Git is the underlying version control system, while GitHub is a hosting service that adds collaboration tools on top of Git.

## Frequency of committing Code

Unfortunately, I didn’t end up tracking the code through Github, so committing code was not frequent.

# Appendix 1 Python Code

Paste your Python Code in Here (White Theme and coloured Syntax)

import pygame

import random

import time

import sys

#===============================================================================

# Functions Area =

#===============================================================================

def check\_neighbor\_rooms(pos, item\_list):

""" Checks each orthogonal cell next to pos for the requested item

returns True as soon as the item is found.

"""

exits = cave[pos]

return any(item in cave[pos] for item in item\_list)

def draw\_room( pos, screen):

""" Draws the room in the back buffer

"""

x=0

y=1

exits = cave[player\_pos]

screen.fill( (0,0,0) ) #paint the background in black

#draw the room circle in brown

circle\_radius = int ((SCREEN\_WIDTH//2)\*.75)

pygame.draw.circle(screen, BROWN, (SCREEN\_WIDTH//2, SCREEN\_HEIGHT//2), circle\_radius, 0)

#next draw all exits from the room

if exits[LEFT] > 0:

left = 0

top = SCREEN\_HEIGHT//2-40

pygame.draw.rect(screen, BROWN, ( (left,top), (SCREEN\_WIDTH//4,80)), 0)

if exits[RIGHT] > 0:

#draw right exit

left = SCREEN\_WIDTH-(SCREEN\_WIDTH//4)

top = SCREEN\_HEIGHT//2-40

pygame.draw.rect(screen, BROWN, ((left,top), (SCREEN\_WIDTH//4,80)), 0)

if exits[UP] > 0:

#draw top exit

left = SCREEN\_WIDTH//2-40

top = 0

pygame.draw.rect(screen, BROWN, ((left,top), (80,SCREEN\_HEIGHT//4)), 0)

if exits[DOWN] > 0 :

#draw bottom exit

left = SCREEN\_WIDTH//2-40

top = SCREEN\_HEIGHT-(SCREEN\_WIDTH//4)

pygame.draw.rect(screen, BROWN, ((left,top), (80,SCREEN\_HEIGHT//4)), 0)

#find out if bats, pits or a wumpus is near

bats\_near = check\_neighbor\_rooms(player\_pos, bats\_list)

pit\_near = check\_neighbor\_rooms(player\_pos, pits\_list)

wumpus\_near = check\_neighbor\_rooms(player\_pos, [wumpus\_pos, [-1,-1]])

#draw a blood circle if the Wumpus is nearby

if wumpus\_near == True:

circle\_radius = int ((SCREEN\_WIDTH//2)\*.5)

pygame.draw.circle(screen, RED, (SCREEN\_WIDTH//2, SCREEN\_HEIGHT//2), circle\_radius, 0)

#draw the pit in black if it is present

if player\_pos in pits\_list:

circle\_radius = int ((SCREEN\_WIDTH//2)\*.5)

pygame.draw.circle(screen, BLACK, (SCREEN\_WIDTH//2, SCREEN\_HEIGHT//2), circle\_radius, 0)

#draw the player

screen.blit(player\_img,(SCREEN\_WIDTH//2-player\_img.get\_width()//2,SCREEN\_HEIGHT//2-player\_img.get\_height()//2))

#draw the bat imag

if player\_pos in bats\_list:

screen.blit(bat\_img,(SCREEN\_WIDTH//2-bat\_img.get\_width()//2,SCREEN\_HEIGHT//2-bat\_img.get\_height()//2))

#draw the wumpus

if player\_pos == wumpus\_pos:

screen.blit(wumpus\_img,(SCREEN\_WIDTH//2-wumpus\_img.get\_width()//2,SCREEN\_HEIGHT//2-wumpus\_img.get\_height()//2))

#draw text

y\_text\_pos = 0 #keeps track of the next y position on screen to draw text

pos\_text = font.render("POS:"+str(player\_pos), 1, (0, 255, 64))

screen.blit(pos\_text,(0, 0))

arrow\_text = font.render("Arrows: "+str(num\_arrows), 1, (0, 255, 64))

y\_text\_pos = y\_text\_pos+pos\_text.get\_height()+10

screen.blit(arrow\_text,(0, y\_text\_pos))

if bats\_near == True:

bat\_text = font.render("You hear the squeaking of bats nearby", 1, (0, 255, 64))

y\_text\_pos = y\_text\_pos+bat\_text.get\_height()+10

screen.blit(bat\_text,(0, y\_text\_pos))

if pit\_near == True:

pit\_text = font.render("You feel a draft nearby", 1, (0, 255, 64))

y\_text\_pos = y\_text\_pos+pit\_text.get\_height()+10

screen.blit(pit\_text,(0, y\_text\_pos))

if player\_pos in bats\_list: #if bats are here, go ahead and flip the display and wait a bit

pygame.display.flip()

time.sleep(2.0)

def populate\_cave():

global player\_pos, wumpus\_pos

#place the player

player\_pos = random.randint(1, 20)

# place the wumpus

place\_wumpus()

#place the bats

for bat in range(0,NUM\_BATS):

place\_bat()

#place the pits

for pit in range (0,NUM\_PITS):

place\_pit()

#place the arrows

for arrow in range (0,NUM\_ARROWS):

place\_arrow()

print ("Player at: "+str(player\_pos))

print ("Wumpus at: "+str(wumpus\_pos))

print ("Bats at:" + str(bats\_list) )

print ("Pits at:" + str(pits\_list))

print ("Arrows at:" +str(arrows\_list))

def place\_wumpus():

global player\_pos, wumpus\_pos

wumpus\_pos = player\_pos

while (wumpus\_pos == player\_pos):

wumpus\_pos = random.randint(0,20)

def place\_bat():

#place the bats

bat\_pos = player\_pos

while bat\_pos == player\_pos or (bat\_pos in bats\_list) or (bat\_pos == wumpus\_pos) or (bat\_pos in pits\_list):

bat\_pos = random.randint(1,20)

bats\_list.append(bat\_pos)

def place\_pit():

pit\_pos = player\_pos

while (pit\_pos == player\_pos) or (pit\_pos in bats\_list) or (pit\_pos == wumpus\_pos) or (pit\_pos in pits\_list):

pit\_pos = random.randint(1,20)

pits\_list.append(pit\_pos)

def place\_arrow():

arrow\_pos = player\_pos

while (arrow\_pos == player\_pos) or (arrow\_pos in bats\_list) or (arrow\_pos == wumpus\_pos) or (arrow\_pos in pits\_list):

arrow\_pos = random.randint(1,20)

arrows\_list.append(arrow\_pos)

def check\_room(pos):

global player\_pos, screen, num\_arrows

#is there a Wumpus in the room?

if player\_pos == wumpus\_pos:

game\_over("You were eaten by a WUMPUS!!!")

#is there a pit?

if player\_pos in pits\_list:

game\_over("You fell into a bottomless pit!!")

#Are there bats in the room? If so move the player and the bats

if player\_pos in bats\_list:

print("Bats pick you up and place you elsewhere in the cave!")

screen.fill(BLACK)

bat\_text = font.render("Bats pick you up and place you elsewhere in the cave!", 1, (0, 255, 64))

textrect = bat\_text.get\_rect()

textrect.centerx = screen.get\_rect().centerx

textrect.centery = screen.get\_rect().centery

screen.blit(bat\_text,textrect)

pygame.display.flip()

time.sleep(2.5)

#move the bats

new\_pos = player\_pos

while (new\_pos == player\_pos) or (new\_pos in bats\_list) or (new\_pos == wumpus\_pos) or (new\_pos in pits\_list):

new\_pos = random.randint(1,20)

bats\_list.remove(player\_pos)

bats\_list.append(new\_pos)

print ("bat at: "+str(new\_pos))

#now move the player

new\_pos = player\_pos # set new\_pos equal to the old os so the first test fails

# Now place the player in a random location

while (new\_pos == player\_pos) or (new\_pos in bats\_list) or (new\_pos == wumpus\_pos) or (new\_pos in pits\_list):

new\_pos = random.randint(1,20)

player\_pos = new\_pos

print ("player at:"+str(player\_pos))

#Is there an arrow in the room?

if player\_pos in arrows\_list:

screen.fill(BLACK)

text = font.render("You have found an arrow!", 1, (0, 255, 64))

textrect = text.get\_rect()

textrect.centerx = screen.get\_rect().centerx

textrect.centery = screen.get\_rect().centery

screen.blit(text,textrect)

pygame.display.flip()

time.sleep(2.5)

num\_arrows +=1

arrows\_list.remove(player\_pos)

def reset\_game():

global num\_arrows

populate\_cave()

num\_arrows = 1

def game\_over(message):

global screen

time.sleep(1.0)

screen.fill(RED)

text=font.render(message, 1, (0, 255, 64))

textrect = text.get\_rect()

textrect.centerx = screen.get\_rect().centerx

textrect.centery = screen.get\_rect().centery

screen.blit(text,textrect)

pygame.display.flip()

time.sleep(2.5)

print (message)

pygame.quit()

sys.exit()

def move\_wumpus():

global wumpus\_pos

if mobile\_wumpus == False or random.randint(1,100) > wumpus\_move\_chance:

return

exits = cave[wumpus\_pos]

for new\_room in exits:

if new\_room == 0:

continue

elif new\_room == player\_pos:

continue

elif new\_room in bats\_list:

continue

elif new\_room in pits\_list:

continue

else:

wumpus\_pos = new\_room

break

print ("Wumpus moved to:"+str(wumpus\_pos))

def shoot\_arrow(direction):

global num\_arrows, player\_pos

hit = False

if num\_arrows == 0:

return False

num\_arrows -= 1

if wumpus\_pos == cave[player\_pos][direction]:

hit = True

if hit == True:

game\_over("Your aim was true and you have killed the Wumpus!")

pygame.quit()

sys.exit()

else:

print ("Your arrow sails into the darkness, never to be seen again....")

place\_wumpus()

if num\_arrows == 0:

game\_over("You are out of arrows. You have died!")

pygame.quit()

sys.exit()

def check\_pygame\_events():

global player\_pos

event = pygame.event.poll()

if event.type == pygame.QUIT:

pygame.quit()

sys.exit()

elif event.type == pygame.KEYDOWN:

if event.key == pygame.K\_ESCAPE:

pygame.quit()

sys.exit()

elif event.key ==pygame.K\_LEFT:

if pygame.key.get\_mods() & pygame.KMOD\_SHIFT:

shoot\_arrow(LEFT)

elif cave[player\_pos][LEFT] > 0:

player\_pos=cave[player\_pos][LEFT]

move\_wumpus()

elif event.key == pygame.K\_RIGHT:

if pygame.key.get\_mods() & pygame.KMOD\_SHIFT:

shoot\_arrow(RIGHT)

elif cave[player\_pos][RIGHT] >0:

player\_pos = cave[player\_pos][RIGHT]

move\_wumpus()

elif event.key == pygame.K\_UP:

if pygame.key.get\_mods() & pygame.KMOD\_SHIFT:

shoot\_arrow(UP)

elif cave[player\_pos][UP] > 0:

player\_pos = cave[player\_pos][UP]

move\_wumpus()

elif event.key ==pygame.K\_DOWN:

if pygame.key.get\_mods() & pygame.KMOD\_SHIFT:

shoot\_arrow(DOWN)

elif cave[player\_pos][DOWN] > 0:

player\_pos = cave[player\_pos][DOWN]

move\_wumpus()

def print\_instructions():

print(

'''

Hunt The Wumpus!

This is the game of "Hunt the Wumpus". You have been cast into a

dark 20 room cave with a fearsome Wumpus. The cave is shaped like a

dodecahedron and the only way out is to kill the Wumpus. To that end

you have a bow with one arrow. You might find more arrows from unlucky

past Wumpus victims in the cave. There are other dangers in the cave,

specifically bats and bottomless pits.

\* If you run out of arrows you die.

\* If you end up in the same room with the Wumpus you die.

\* If you fall into a bottomless pit you die.

\* If you end up in a room with bats they will pick you up

and deposit you in a random location.

If you are near the Wumpus you will see the bloodstains on the walls.

If you are near bats you will hear them and if you are near a bottomless

pit you will feel the air flowing down it.

Use the arrow keys to move. Press the <SHIFT> key and an arrow key to

fire your arrow.

'''

)

#===============================================================================

# Globals and Constants area =

#===============================================================================

#Our screen width and height

SCREEN\_WIDTH = SCREEN\_HEIGHT= 1000

#number of bats, pits and arrows in the cave#load our three images

bat\_img = pygame.image.load('images/bat.png')

player\_img = pygame.image.load('images/player.png')

wumpus\_img = pygame.image.load('images/wumpus.png')

arrow\_img = pygame.image.load('images/arrow.png')

#increase the number of bats and pits to make it harder

#increase the number of arrows to make it easier

NUM\_BATS = 3

NUM\_PITS = 3

NUM\_ARROWS = 0

player\_pos = 0 #tracks where we are in the cave

wumpus\_pos = 0 #tracks where the Wumpus is

num\_arrows = 1 # Starting arrows

mobile\_wumpus = False #Set this to true to allow the wumpus to move

wumpus\_move\_chance = 50

#constants for directions

UP = 0

DOWN = 1

LEFT = 2

RIGHT = 3

#color definitions

BROWN = 193,154,107

BLACK = 0,0,0

RED = 138,7,7

cave = {1: [0,8,2,5], 2: [0,10,3,1], 3: [0,12,4,2], 4: [0,14,5,3],

5:[0,6,1,4], 6: [5,0,7,15], 7: [0,17,8,6], 8: [1,0,9,7],

9: [0,18,10,8], 10: [2,0,11,9], 11: [0,19,12,10], 12: [3,0,13,11],

13: [0,20,14,12], 14: [4,0,15,13], 15: [0,16,6,14], 16: [15,0,17,20],

17: [7,0,18,16], 18: [9,0,19,17], 19: [11,0,20,18], 20: [13,0,16,19] }

bats\_list = []

pits\_list = []

arrows\_list = []

#===============================================================================

# Initialization area =

#===============================================================================

print\_instructions()

input("Press <ENTER> to begin.")

pygame.init()

screen = pygame.display.set\_mode( (SCREEN\_WIDTH, SCREEN\_HEIGHT), pygame.DOUBLEBUF | pygame.HWSURFACE )

pygame.display.set\_caption("Hunt the Wumpus")

#load our three images

bat\_img = pygame.image.load('images/bat.png')

player\_img = pygame.image.load('images/player.png')

wumpus\_img = pygame.image.load('images/wumpus.png')

arrow\_img = pygame.image.load('images/arrow.png')

#setup our font

font = pygame.font.Font(None, 36)

#Get initial game settings

reset\_game()

#===============================================================================

# Main Game Loop =

#===============================================================================

while True:

check\_pygame\_events()

draw\_room(player\_pos, screen)

pygame.display.flip()

check\_room(player\_pos)

## Readme File for Python

How do I start the Project

# Hunt the Wumpus - Python Pygame Edition

A classic cave exploration and survival game written in Python using Pygame. Inspired by the original Hunt the Wumpus game, the player must navigate a 20-room cave, avoid hazards, collect arrows, and ultimately kill the Wumpus to win.

## Features

- 20 interconnected cave rooms

- Hazards: bottomless pits and bats

- Moving or stationary Wumpus (configurable)

- Arrow based combat with limited ammo

- Visual representation using Pygame

- Environmental hints based on proximity to dangers

## Controls

- Arrow Keys: Move the player through available exits

- Shift + Arrow Keys: Fire an arrow in that direction

- ESC: Quit the game

## Requirements

- Python 3.x

- Pygame (`pip install pygame`)

# Appendix 2 Java Code

Paste you Java Code in here. (White Theme and coloured Syntax).

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

import java.util.\*;

import java.awt.image.\*;

import javax.imageio.ImageIO;

import javax.sound.sampled.\*;

import java.io.\*;

public class HuntTheWumpus extends JPanel implements KeyListener {

private static final int SCREEN\_WIDTH = 1000;

private static final int SCREEN\_HEIGHT = 1000;

private static final int NUM\_BATS = 3;

private static final int NUM\_PITS = 3;

private static final int NUM\_ARROWS = 0;

private static final int UP = 0;

private static final int DOWN = 1;

private static final int LEFT = 2;

private static final int RIGHT = 3;

private static final Color BROWN = new Color(193, 154, 107);

private static final Color RED = new Color(138, 7, 7);

private int playerPos = 0;

private int wumpusPos = 0;

private int numArrows = 1;

private boolean mobileWumpus = true;

private int wumpusMoveChance = 50;

private boolean gameOver = false;

private String gameMessage = "";

private BufferedImage playerImg, wumpusImg, batImg, arrowImg, pitImg;

private HashMap<Integer, int[]> cave = new HashMap<>();

private java.util.List<Integer> batsList = new ArrayList<>();

private java.util.List<Integer> pitsList = new ArrayList<>();

private java.util.List<Integer> arrowsList = new ArrayList<>();

public HuntTheWumpus() {

setPreferredSize(new Dimension(SCREEN\_WIDTH, SCREEN\_HEIGHT));

setFocusable(true);

addKeyListener(this);

initCave();

loadImages();

resetGame();

}

private void initCave() {

int[][] data = {

{0,8,2,5},{0,10,3,1},{0,12,4,2},{0,14,5,3},{0,6,1,4},

{5,0,7,15},{0,17,8,6},{1,0,9,7},{0,18,10,8},{2,0,11,9},

{0,19,12,10},{3,0,13,11},{0,20,14,12},{4,0,15,13},

{0,16,6,14},{15,0,17,20},{7,0,18,16},{9,0,19,17},

{11,0,20,18},{13,0,16,19}

};

for (int i = 0; i < data.length; i++) cave.put(i + 1, data[i]);

}

private BufferedImage loadImage(String path) {

try {

return ImageIO.read(new File(path));

} catch (IOException e) {

System.err.println("Could not load image: " + path);

return null;

}

}

private void loadImages() {

playerImg = loadImage("images/player.png");

wumpusImg = loadImage("images/wumpus.png");

batImg = loadImage("images/bat.png");

arrowImg = loadImage("images/arrow.png");

pitImg = loadImage("images/pit.png");

}

private void playSound(String soundFile) {

try {

File file = new File("sounds/" + soundFile);

if (!file.exists()) {

System.err.println("Missing sound: " + soundFile);

return;

}

Clip clip = AudioSystem.getClip();

AudioInputStream inputStream = AudioSystem.getAudioInputStream(file);

clip.open(inputStream);

clip.start();

} catch (Exception e) {

System.err.println("Error playing sound: " + soundFile + " - " + e.getMessage());

}

}

private void resetGame() {

gameOver = false;

gameMessage = "";

batsList.clear();

pitsList.clear();

arrowsList.clear();

Random rand = new Random();

playerPos = rand.nextInt(20) + 1;

do { wumpusPos = rand.nextInt(20) + 1; } while (wumpusPos == playerPos);

for (int i = 0; i < NUM\_BATS; i++) placeEntity(rand, batsList);

for (int i = 0; i < NUM\_PITS; i++) placeEntity(rand, pitsList);

for (int i = 0; i < NUM\_ARROWS; i++) placeEntity(rand, arrowsList);

numArrows = 1;

repaint();

}

private void placeEntity(Random rand, java.util.List<Integer> list) {

int pos;

do {

pos = rand.nextInt(20) + 1;

} while (pos == playerPos || list.contains(pos) || pos == wumpusPos);

list.add(pos);

}

private void checkRoom() {

if (playerPos == wumpusPos) {

playSound("wumpus.wav");

endGame("You were eaten by the WUMPUS!");

} else if (pitsList.contains(playerPos)) {

playSound("pit.wav");

endGame("You fell into a bottomless pit! Press 'R' to restart.");

} else {

if (batsList.contains(playerPos)) {

playSound("bats.wav");

Random rand = new Random();

batsList.remove((Integer) playerPos);

int newBatPos;

do {

newBatPos = rand.nextInt(20) + 1;

} while (batsList.contains(newBatPos) || newBatPos == wumpusPos || pitsList.contains(newBatPos));

batsList.add(newBatPos);

int newPlayerPos;

do {

newPlayerPos = rand.nextInt(20) + 1;

} while (newPlayerPos == playerPos || newPlayerPos == wumpusPos || pitsList.contains(newPlayerPos));

playerPos = newPlayerPos;

gameMessage = "Bats picked you up and dropped you elsewhere!";

}

if (arrowsList.contains(playerPos)) {

playSound("arrow.wav");

numArrows++;

arrowsList.remove((Integer) playerPos);

gameMessage = "You found an arrow!";

}

}

}

private void endGame(String message) {

gameOver = true;

gameMessage = message;

repaint();

}

private void shootArrow(int direction) {

if (numArrows == 0) return;

numArrows--;

int targetRoom = cave.get(playerPos)[direction];

if (targetRoom == wumpusPos) {

playSound("victory.wav");

endGame("Your aim was true! You killed the Wumpus!");

} else {

playSound("miss.wav");

Random rand = new Random();

do { wumpusPos = rand.nextInt(20) + 1; } while (wumpusPos == playerPos);

if (numArrows == 0) endGame("Out of arrows. You have died! Press 'R' to restart.");

else gameMessage = "You missed. The Wumpus may have moved...";

}

}

private boolean isNear(java.util.List<Integer> list) {

int[] exits = cave.get(playerPos);

for (int room : exits) if (list.contains(room)) return true;

return false;

}

private boolean isWumpusNear() {

int[] exits = cave.get(playerPos);

for (int room : exits) if (room == wumpusPos) return true;

return false;

}

@Override

protected void paintComponent(Graphics g) {

super.paintComponent(g);

g.setColor(Color.BLACK);

g.fillRect(0, 0, SCREEN\_WIDTH, SCREEN\_HEIGHT);

g.setColor(BROWN);

g.fillOval(SCREEN\_WIDTH / 4, SCREEN\_HEIGHT / 4, SCREEN\_WIDTH / 2, SCREEN\_HEIGHT / 2);

int[] exits = cave.get(playerPos);

if (exits[LEFT] > 0) g.fillRect(0, SCREEN\_HEIGHT / 2 - 40, SCREEN\_WIDTH / 4, 80);

if (exits[RIGHT] > 0) g.fillRect(SCREEN\_WIDTH - SCREEN\_WIDTH / 4, SCREEN\_HEIGHT / 2 - 40, SCREEN\_WIDTH / 4, 80);

if (exits[UP] > 0) g.fillRect(SCREEN\_WIDTH / 2 - 40, 0, 80, SCREEN\_HEIGHT / 4);

if (exits[DOWN] > 0) g.fillRect(SCREEN\_WIDTH / 2 - 40, SCREEN\_HEIGHT - SCREEN\_HEIGHT / 4, 80, SCREEN\_HEIGHT / 4);

if (pitImg != null && pitsList.contains(playerPos)) {

g.drawImage(pitImg, SCREEN\_WIDTH / 2 - pitImg.getWidth() / 2, SCREEN\_HEIGHT / 2 - pitImg.getHeight() / 2, null);

} else if (batImg != null && batsList.contains(playerPos)) {

g.drawImage(batImg, SCREEN\_WIDTH / 2 - batImg.getWidth() / 2, SCREEN\_HEIGHT / 2 - batImg.getHeight() / 2, null);

} else if (wumpusImg != null && playerPos == wumpusPos) {

g.drawImage(wumpusImg, SCREEN\_WIDTH / 2 - wumpusImg.getWidth() / 2, SCREEN\_HEIGHT / 2 - wumpusImg.getHeight() / 2, null);

}

if (!gameOver && playerImg != null) {

g.drawImage(playerImg, SCREEN\_WIDTH / 2 - playerImg.getWidth() / 2, SCREEN\_HEIGHT / 2 - playerImg.getHeight() / 2, null);

}

g.setColor(Color.GREEN);

g.drawString("Position: " + playerPos + " Arrows: " + numArrows, 10, 20);

int y = 50;

if (!gameOver) {

if (isWumpusNear()) { g.drawString("You see bloodstains on the walls.", 10, y); y += 20; }

if (isNear(batsList)) { g.drawString("You hear the squeaking of bats.", 10, y); y += 20; }

if (isNear(pitsList)) { g.drawString("You feel a draft.", 10, y); y += 20; }

}

if (!gameMessage.isEmpty()) {

g.setColor(RED);

g.drawString(gameMessage, 10, y);

}

if (gameOver) {

g.setFont(new Font("Arial", Font.BOLD, 36));

g.setColor(Color.RED);

g.drawString("GAME OVER", SCREEN\_WIDTH / 2 - 120, SCREEN\_HEIGHT / 2 + 200);

g.setFont(new Font("Arial", Font.PLAIN, 18));

g.drawString("Press 'R' to restart or use Game > Restart menu", SCREEN\_WIDTH / 2 - 200, SCREEN\_HEIGHT / 2 + 240);

}

}

@Override public void keyPressed(KeyEvent e) {

int key = e.getKeyCode();

if (gameOver && key == KeyEvent.VK\_R) {

resetGame();

return;

}

if (gameOver) return;

boolean shift = (e.getModifiersEx() & KeyEvent.SHIFT\_DOWN\_MASK) != 0;

int[] exits = cave.get(playerPos);

if (key == KeyEvent.VK\_LEFT) {

if (shift) shootArrow(LEFT);

else if (exits[LEFT] > 0) playerPos = exits[LEFT];

} else if (key == KeyEvent.VK\_RIGHT) {

if (shift) shootArrow(RIGHT);

else if (exits[RIGHT] > 0) playerPos = exits[RIGHT];

} else if (key == KeyEvent.VK\_UP) {

if (shift) shootArrow(UP);

else if (exits[UP] > 0) playerPos = exits[UP];

} else if (key == KeyEvent.VK\_DOWN) {

if (shift) shootArrow(DOWN);

else if (exits[DOWN] > 0) playerPos = exits[DOWN];

}

checkRoom();

repaint();

}

@Override public void keyReleased(KeyEvent e) {}

@Override public void keyTyped(KeyEvent e) {}

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> {

JFrame frame = new JFrame("Hunt the Wumpus");

HuntTheWumpus gamePanel = new HuntTheWumpus();

JMenuBar menuBar = new JMenuBar();

JMenu gameMenu = new JMenu("Game");

JMenuItem restartItem = new JMenuItem("Restart");

restartItem.addActionListener(e -> gamePanel.resetGame());

gameMenu.add(restartItem);

menuBar.add(gameMenu);

frame.setJMenuBar(menuBar);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setContentPane(gamePanel);

frame.pack();

frame.setLocationRelativeTo(null);

frame.setVisible(true);

});

}

}

## Readme File for Java

How do I start the Project

# Hunt the Wumpus - Java Swing Edition

A classic adventure reimagined in Java! This version of Hunt the Wumpus uses Java Swing and AWT for GUI rendering, and includes basic audio support, hazard detection, and a turn based room navigation system.

## Game Overview

You're trapped in a 20 room cave with:

- A deadly Wumpus monster

- Bottomless pits

- Giant bats

- A bow with limited arrows

Your goal is to survive and kill the Wumpus before it gets you!

---

## Features

- Java Swing based GUI

- Room based cave map (dodecahedron layout)

- Randomized hazard placement (bats, pits, Wumpus)

- Audio support for immersive feedback

- Restart option from in-game menu or keypress

##  Controls

**Action Keys**

Move = Arrow keys

Shoot arrow = Shift + Arrow key

Restart (if dead) = Press `R`

Quit = Close window

# Reflection

What did you learn from this project?

From this project, I learned a lot about programming in both Python and Java. Writing the game first in Python helped me understand how to quickly build and test ideas because Python’s syntax is simple and easy to use. It showed me how prototyping can speed up development by letting me focus on the game’s logic without worrying about strict rules. Then, converting the code into Java taught me about the importance of strong typing, structure, and error checking to make the program more reliable and secure. I also gained experience working with different programming styles and learned how tools like ChatGPT can help improve and translate code. Overall, this project improved my coding skills, problem solving, and understanding of how different languages are used in real world development.