**Software Engineering**

**Year 11 , 2025**

**Assessment Task 2**

**Object-Oriented Programming Assignment:**

**“Hunting Wumpus”**

**By: Joshua Morrison**

Contents

[Software Requirement Specification 3](#_Toc198125157)

[Gantt Chart 4](#_Toc198125158)

[Budget 5](#_Toc198125159)

[Justification of Technology 6](#_Toc198125160)

[Design 7](#_Toc198125161)

[UML Notation 8](#_Toc198125162)

[Class Diagram 9](#_Toc198125163)

[Flowchart 10](#_Toc198125164)

[Pseudocode 11](#_Toc198125165)

[Sequence Diagram 12](#_Toc198125166)

[Graphical User Interface GUI 13](#_Toc198125167)

[Artificial Intelligence Conversion Code 14](#_Toc198125168)

[Explanation of why Java is safer than Python or vice versa 15](#_Toc198125169)

[Security 15](#_Toc198125170)

[Compiling and Execution 15](#_Toc198125171)

[Storing data 16](#_Toc198125172)

[Encryption. 17](#_Toc198125173)

[Why prototyping might be done in Python rather than Java. 18](#_Toc198125174)

[What Tools were used in the development of this Project and their justification 19](#_Toc198125175)

[Visual Code IDE 20](#_Toc198125176)

[Python Compiler 21](#_Toc198125177)

[Artificial Intelligence Converter 22](#_Toc198125178)

[Java 23](#_Toc198125179)

[Code is commented and following industry standard practices 24](#_Toc198125180)

[Justification of Git and GitHub and their difference 25](#_Toc198125181)

[Frequency of committing Code 26](#_Toc198125182)

[Appendix 1 Python Code 27](#_Toc198125183)

[Readme File for Python 28](#_Toc198125184)

[Appendix 2 Java Code 29](#_Toc198125185)

[Readme File for Java 30](#_Toc198125186)

[Reflection 31](#_Toc198125187)

# Software Requirement Specification

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

Hunt the Wumpus is a classic text-based adventure game where players navigate a network of 20 interconnected rooms to hunt a creature called the Wumpus while avoiding hazards like bottomless pits and giant bats. Players receive subtle clues such as smells, drafts, and sounds to detect nearby dangers and must use logic to decide whether to move or shoot arrows into adjacent rooms. The game is often used in programming assignments to teach concepts like graph structures, control flow, and user interaction.

## Gantt Chart

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

This can be record from Git or GitHub.

Week 1- planning code and gathering resources

Week 2- designing pseudocode and UML diagrams

Week 3- python development and initial testing

Week 4- translation to java and debugging

Week 5- final testing and reflection

## Budget

You are a software engineer charging $60 per hour.

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

18 x 60= $1080

## 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.

* Simplicity & Readability: Python has a cleaner, more readable syntax which helps you focus on the logic of the game rather than boilerplate code. This is ideal for beginners and for educational projects like this one.
* Rapid Development: Python enables faster prototyping and iteration, which means you can build, test, and refine your game without as much overhead.
* Built-In Libraries: Python comes with great built-in libraries (like random for room hazards or sys for input/output control) that make game logic more manageable.
* Community & Resources: There are tons of tutorials, examples, and community projects that make it easier to find help when you're stuck.

What IDE Is Being Used and Why?

You might be using VS Code (Visual Studio Code)—and if not, it’s a strong contender. Here's why:

* Lightweight but Powerful: Unlike more heavy-duty IDEs (like Eclipse for Java), VS Code is quick to load and easy on resources.
* Python-Friendly: With extensions like the Python plugin, linting, debugging, and autocomplete all work seamlessly.
* Git Integration: You can see your commit history, make branches, and push updates right from the editor.
* Terminal Access Built-in: This makes it easy to run your game and commit code—all from one window.

Advantages Over Other IDEs for *Hunt the Wumpus*

* Eclipse or IntelliJ are Java-focused and can feel bloated for a small Python project.
* IDLE is Python’s default editor, but it lacks advanced features like Git integration or extension support.
* PyCharm is excellent but can be overkill (and slower to launch) for compact scripts like this one.

So if you’re working solo or in a student setting, VS Code hits the sweet spot between functionality and simplicity.

# 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?

Unified Modeling Language (UML) notation is used to visually represent the design and structure of software systems. It helps developers, analysts, and stakeholders understand how different parts of a program interact—*before* a single line of code is written. By mapping out things like class hierarchies, use cases, object interactions, and system workflows, UML lets teams plan more effectively, identify design flaws early, and ensure everyone’s on the same page. It’s especially useful in collaborative or complex projects where clarity is key. Think of it as the architectural blueprint for your software castle—it keeps you from bricking up the staircase halfway through the build.

## Class Diagram

A screenshot of a computer program

AI-generated content may be incorrect.

## Flowchart

A diagram of a computer program

AI-generated content may be incorrect.

## Pseudocode

## START GAME

## Initialize rooms, player, and hazards

## WHILE game is running:

## DISPLAY current room and nearby warnings

## PROMPT player to move or shoot

## IF move:

## GET chosen room

## IF valid:

## player moves

## CHECK if Wumpus, Pit, or Bat

## ELSE:

## DISPLAY invalid move

## IF shoot:

## IF player has arrows:

## DECREASE arrow count

## CHECK if Wumpus is in chosen room

## IF hit:

## DISPLAY win

## END game

## ELSE:

## Wumpus may move

## ELSE:

## DISPLAY "no arrows"

## DISPLAY game over

## END

## Sequence Diagram

A diagram of a game

AI-generated content may be incorrect.

## Graphical User Interface GUI

import tkinter as tk

import random

from tkinter import messagebox

class Room:

    def \_\_init\_\_(self, id, connections):

        self.id = id

        self.connections = connections

class WumpusGameGUI:

    def \_\_init\_\_(self, root):

        self.root = root

        self.root.title("Hunt the Wumpus")

        self.rooms = self.create\_rooms()

        self.player\_room = random.choice(self.rooms)

        self.wumpus\_room = random.choice([r for r in self.rooms if r != self.player\_room])

        self.create\_widgets()

        self.update\_ui()

    def create\_rooms(self):

        # Create 5 simple interconnected rooms

        return [

            Room(1, [2, 3]),

            Room(2, [1, 4]),

            Room(3, [1, 5]),

            Room(4, [2]),

            Room(5, [3])

        ]

    def create\_widgets(self):

        self.label = tk.Label(self.root, text="", font=("Helvetica", 14))

        self.label.pack(pady=10)

        self.buttons\_frame = tk.Frame(self.root)

        self.buttons\_frame.pack(pady=10)

    def update\_ui(self):

        self.label.config(text=f"You are in Room {self.player\_room.id}")

        # Clear old buttons

        for widget in self.buttons\_frame.winfo\_children():

            widget.destroy()

        # Create new move buttons

        for room\_id in self.player\_room.connections:

            btn = tk.Button(

                self.buttons\_frame,

                text=f"Move to Room {room\_id}",

                command=lambda rid=room\_id: self.move\_to\_room(rid),

                width=20

            )

            btn.pack(pady=2)

        # Warning if near Wumpus

        if self.wumpus\_room.id in self.player\_room.connections:

            messagebox.showinfo("Warning!", "You smell something terrible nearby...")

    def move\_to\_room(self, room\_id):

        new\_room = next(r for r in self.rooms if r.id == room\_id)

        self.player\_room = new\_room

        if self.player\_room == self.wumpus\_room:

            messagebox.showerror("Game Over", "The Wumpus ate you! Game Over.")

            self.root.destroy()

        else:

            self.update\_ui()

# Main loop

if \_\_name\_\_ == "\_\_main\_\_":

    root = tk.Tk()

    game = WumpusGameGUI(root)

    root.mainloop()

## Artificial Intelligence Conversion Code

Explain what artificial intelligence engine your 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.

To assist in converting the Hunt the Wumpus Python code into Java, I used an AI code conversion tool powered by OpenAI's ChatGPT. This engine is capable of understanding programming logic and generating equivalent code in another language while maintaining structure and functionality.

The AI engine correctly translated many core components (such as classes, method definitions, and control structures), but I had to manually adjust or rewrite parts of the code due to key differences between Python and Java, such as: Strict typing in Java (e.g., specifying int, String, etc.), Java’s lack of built-in dynamic data structures like Python’s dictionaries, Need for getter/setter methods in Java for encapsulation, Manual handling of input/output in Java. The AI made the initial translation faster, but full success required understanding both languages and making logical adjustments.

AI tools like ChatGPT accelerate development by helping with code translation, debugging, and documentation, but developer knowledge is still essential. Combined with powerful software and hardware, they enable rapid, efficient, and collaborative software development.

Explanation of why Java is safer than Python or vice versa

## **Security**

## Java is generally considered more secure by default for enterprise or large systems. Python is safe for many applications, but requires careful coding practices due to its flexibility.

## **Compiling and Execution**

## Java catches more errors before you run the program. Python allows you to write and test faster, but runtime errors are more common if not carefully tested.

## **Storing data**

## Both can store data well, but Java enforces stricter controls and is better suited for highly secure data storage. Python’s ease of use can sometimes sacrifice safety if developers are careless.

## **Encryption**

## Java has stronger built-in support for encryption, certificates, and regulatory compliance

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

Python is often chosen for prototyping over Java because of its simplicity, speed, and flexibility. Python’s syntax is much more concise and readable, allowing developers to write and modify code quickly without the need for strict type declarations or complex setup. Its dynamic typing and interactive interpreter make it easy to test small pieces of code on the fly, which is ideal during the early stages of development when requirements may still be changing. Additionally, Python offers a vast ecosystem of third-party libraries for data analysis, machine learning, GUI design, and web development, enabling developers to rapidly add features without building everything from scratch. In contrast, Java’s strict object-oriented structure, compile-time type checking, and more complex setup processes make it more suitable for production-level software but slower for initial prototyping. As a result, Python is often the preferred language for quickly building and testing early versions of applications, especially when working with AI, data science, or experimental projects.

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

**1. Python Programming Language**

Python was chosen for its simplicity, readability, and rapid development capabilities. Its syntax is clean and easy to learn, which is ideal for educational projects and quick prototyping. Python also has strong community support and extensive libraries, making game development more accessible, especially with frameworks like Pygame.

**2. Pygame Library**

Pygame is a set of Python modules designed for writing video games. It simplifies handling graphics, sounds, and user input. It’s lightweight and well-documented, perfect for building a 2D text-based adventure game with basic graphics, like “Hunt the Wumpus.”

**3. OpenAI ChatGPT**

Used to assist in converting Python code to Java. This AI tool accelerates the process of understanding code logic and generating equivalent code in another programming language. It helped in initial code translation and problem-solving during development.

**4. Git & GitHub**

Git was used for version control to track code changes, manage different versions, and enable collaboration. GitHub provided a remote repository platform to back up the code, facilitate teamwork, and share the project with teachers or peers. It also supports issue tracking and project management.

**5. Integrated Development Environment: Visual Studio Code**

VS Code was used because it supports Python and Java development, has a vast ecosystem of extensions, and offers integrated terminal and Git support. It improves productivity through features like code autocomplete, syntax highlighting, debugging, and error detection. For Java, IntelliJ IDEA is often preferred due to its strong refactoring tools and deep Java support.

**6. Microsoft Word**

Microsoft Word was used to create documentation, including the journal, Gantt Chart, flowcharts, UML diagrams, and other project artifacts. It is widely used, supports rich formatting, and is a standard for professional documentation submissions.

**7. Diagramming Tools**

These tools were used to create UML diagrams, flowcharts, data flow diagrams, and sequence models. They provide intuitive drag-and-drop interfaces and templates that help visualize system architecture and design, which improves understanding and communication.

**8. Time Tracking Tools**

Tools like Toggl or GitHub time tracking features helped estimate development time for budgeting and reporting the cost of the project.

**Summary:**

Each tool was selected to optimise productivity, collaboration, clarity, and code quality throughout the development lifecycle. Together, they created an environment that supported both efficient coding and comprehensive project documentationAppendix 1 Python Code

import pygame

import random

import time

import sys

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

#                       Functions Area                                         =

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

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

    """ Checks each orthagonal 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 positiojn 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!!")

    #is 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\_instructoions():

    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

dodachedron 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,

specifcally 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 = []

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

#                       Initializations area                                   =

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

print\_instructoions()

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 iniital game settings

reset\_game()

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

#                       Main Game Loop                                         =

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

while True:

    check\_pygame\_events()

    draw\_room(player\_pos, screen)

    pygame.display.flip()

    check\_room(player\_pos)

**Project Description**

This project is a Python implementation of the classic "Hunt The Wumpus" text-based adventure game. The player explores a dangerous cave full of hazards such as bats and bottomless pits while hunting the Wumpus monster. The game features a simple graphical interface built using the Pygame library.

**How to Start the Project:**

1. Install Python

* Download and install \*\*Python 3.x\*\* from the official website: <https://www.python.org/downloads/>

1. Install Pygame

* Open a terminal or command prompt and install the Pygame library by running: pip install pygame

1. Download or Clone the Repository

* Clone this project from GitHub: git clone <https://github.com/your-username/your-repository-name.git>
* Or download the ZIP file and extract it.

1. Verify Project Files

* Ensure the following files and directories are present: /project-folder, main.py, /images, bat.png, player.png, wumpus.png, arrow.png

1. Run the Game

* In your terminal or command prompt, navigate to the project folder: cd project-folder
* Run the game using: python main.py
* The game window should launch, and you're ready to play!

# Appendix 2 Java Code

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

**Project Description**

This is a Java-based version of the classic adventure game \*\*Hunt The Wumpus\*\*. The player explores a dangerous cave while trying to avoid pits and bats, and ultimately kill the Wumpus. The Java version was converted from Python to demonstrate object-oriented principles and improve code safety and structure.

**How to Start the Project**

1. Install Java Development Kit

- Download and install the “Java JDK (version 8 or higher)”:

<https://www.oracle.com/java/technologies/javase-jdk-downloads.html>

1. Install an IDE

- IntelliJ IDEA

<https://www.jetbrains.com/idea/download/>

- Eclipse

<https://www.eclipse.org/downloads/>

- NetBeans

<https://netbeans.apache.org/download/index.html>

1. Download or Clone the Repository

- Clone this project from GitHub:

<https://github.com/your-username/your-repository-name.git>

* Or download the ZIP file and extract it

1. Import the Project into Your IDE

* Open your IDE
* Select "Open Project" or "Import Project" and choose the downloaded project folder.
* If you use an IDE, it will automatically detect and configure the Java project.

1. Compile and Run

* If using terminal: javac Main.java,  java Main
* If using an IDE: Simply click the Run button after importing.

# Reflection

Working on the *Hunt the Wumpus* project taught you valuable lessons in both programming and project planning. You strengthened your Python skills, especially in object-oriented programming, and learned how to use classes to structure your code effectively. Tools like GitHub, UML diagrams, Gantt charts, and flowcharts helped you manage your workflow and visualize the project clearly. You also gained insight into code translation challenges when converting Python to Java with AI, reinforcing the importance of hands-on problem-solving. Overall, the experience enhanced your confidence as a programmer and equipped you with skills that will benefit future development work.