**Software Engineering**

**Year 11 , 2025**

**Assessment Task 2**

**Object-Oriented Programming Assignment:**

**“Hunting Wumpus”**

**By: Ayden 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)

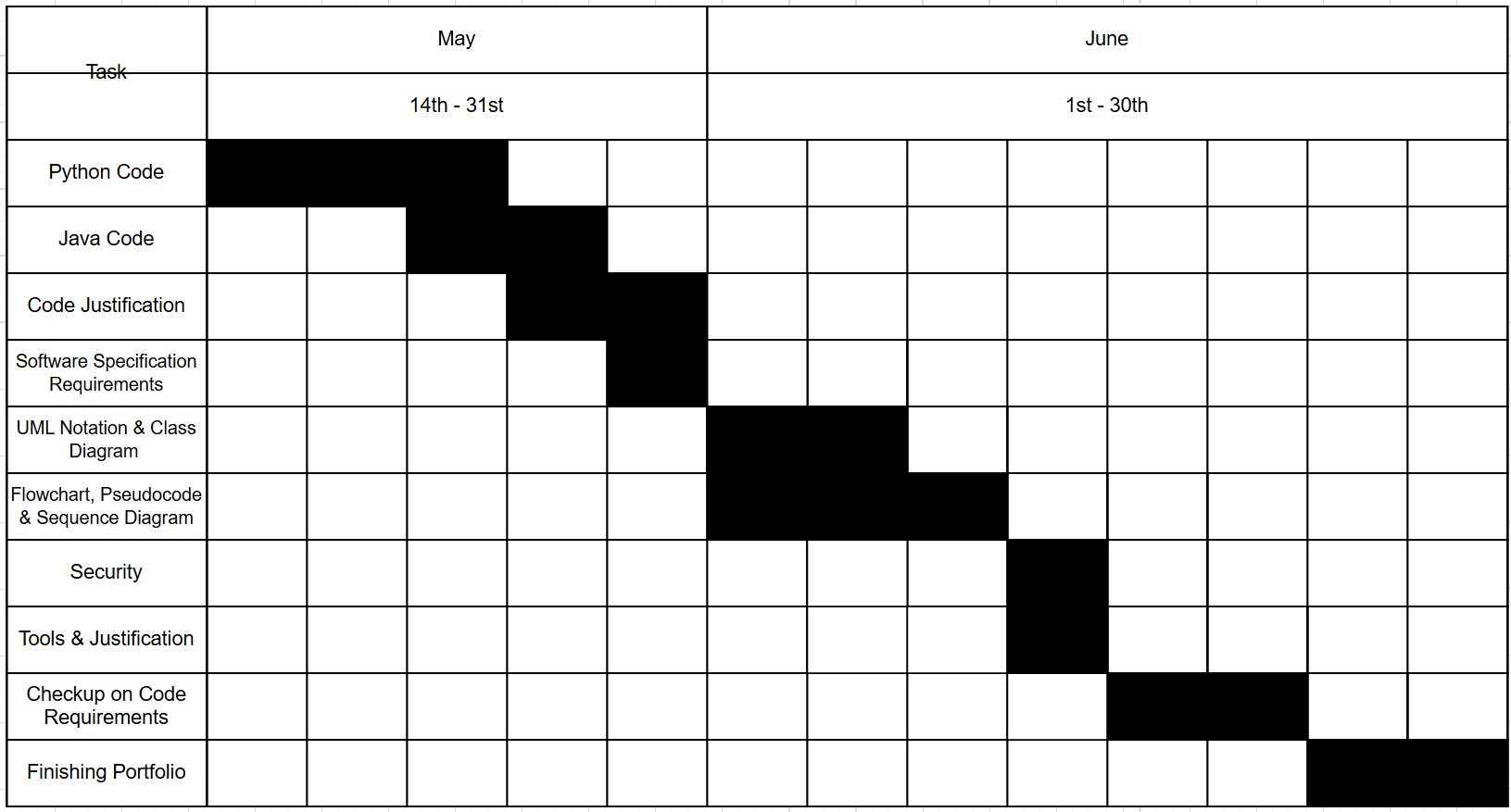
# Software Requirement Specification

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

Hunt the Wumpus is a 2D text-based adventure game created in 1973 by Gregory Yob. The goal is to track down and kill the Wumpus, a deadly creature lurking somewhere in a network of 20 interconnected rooms, arranged in the shape of a dodecahedron. Armed with a single arrow, you rely on clues to find the creature’s location and survive the hunt. Each room connects to three others, and as you explore, you must avoid hazards like bottomless pits, giant bats, and the Wumpus itself.

## 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.

$480

## 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 beginner friendly:** Python has a clean and simple syntax that closely resembles everyday English, which makes it easier for new programmers like me to learn and understand.
* **Less code, faster results:** Python allows developers to write fewer lines of code compared to languages like Java, which helps in building programs more quickly and efficiently.
* **Versatile standard library:** Python includes a rich set of built in modules for tasks like file handling, web development, data analysis, and automation, reducing the need to install extra tools.
* **Great for prototyping:** Python is ideal for quickly testing ideas and building basic versions of programs before fully developing them in more complex languages like Java.
* **Chosen IDE: Visual Studio Code (VS Code):** I use VS Code to write and test my Python code because it's lightweight, starts quickly, and doesn't slow down my computer.
* **Powerful and customizable:** VS Code supports extensions for Python, including linters, formatters, and auto complete features that improve coding accuracy and speed.
* **Built in Git integration:** With VS Code, I can commit, push, and pull code directly from the editor, making it easy to manage version control using Git and GitHub.
* **Multi language support:** Unlike some IDEs that are focused on one language (like IDLE for Python or Eclipse for Java), VS Code can be used for Python, JavaScript, HTML.
* **Better alternative to other IDEs:** Compared to PyCharm, which is heavier and slower on some computers, VS Code is more responsive and easier to set up, especially for beginners.

# 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?

**Visual clarity:** It helps illustrate complex systems clearly using diagrams, making it easier to grasp how components interact.

**Planning and design:** UML diagrams assist in planning software architecture before coding begins, reducing errors and misunderstandings.

**Documentation:** UML provides useful documentation that can be referenced throughout a project’s life cycle, aiding maintenance and updates.

**Supports multiple views:** UML covers different perspectives, such as structure (class diagrams) and behavior (sequence diagrams), giving a complete understanding of the system.

## 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

## 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.

* **Code conversion with ChatGPT:** I used ChatGPT to help convert my Python code into Java. It was successful after I asked for improvements and refinements to match Java's structure and style.
* **Helpful for learning:** ChatGPT made the conversion process easier to understand by explaining how certain Python features translate into Java code, which helped me learn more about both languages.
* **Software boosts development speed:** Tools like IDEs (e.g. Visual Studio Code, IntelliJ), version control systems like Git, and frameworks help developers write, test, and update code faster.
* **Reusable code with libraries and APIs:** Instead of writing everything from scratch, I used existing code libraries and APIs that provided ready made functions, which saved time and reduced errors.
* **Powerful hardware improves performance:** A good computer with a fast processor and more RAM helps run software more smoothly, compile code faster, and manage multiple programs at once.
* **Better testing and multitasking:** Strong hardware also speeds up the process of testing and debugging, especially when switching between tools like browsers, compilers, or terminal windows.
* **Software and hardware work together:** Fast and efficient development depends on both. The right software tools to support smart coding, and the hardware to run everything quickly without slowing down.

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

Address in terms of

# Security

### Compiling and Execution

**Java is generally more secure:** Java runs inside the **Java Virtual Machine (JVM)**, which helps protect the computer by keeping the program separate from the system itself.

**Security Manager in Java:** Java has a special tool called the Security Manager that can block unsafe actions, like reading files or using the internet, which is useful for running code in a controlled or limited environment.

**Python is more flexible but riskier:** Python allows more freedom when coding, which is great for fast development, but it also means there's a higher chance of mistakes if the developer isn’t careful.

**Dynamic typing can cause issues:** Python decides variable types at runtime, which can lead to unexpected errors or vulnerabilities if the code isn’t written and tested properly.

**Summary:** Java is usually safer because it focuses on strict rules and protecting the system, while Python is easier and faster to use but requires extra care to keep it secure.

## Storing data

**Java is better for big projects:** Java is a more structured language, making it ideal for large or complex applications that need to manage a lot of data or users.

**Built in database tools:** Java has tools like JDBC that make it easy to connect to databases such as MySQL or Oracle, and frameworks like Hibernate help manage data more efficiently.

**Catches errors before running:** Java checks for mistakes (like the wrong data type) before the program starts, so problems are caught early, which helps prevent crashes or bugs.

**Python is simpler and faster for small tasks:** Python is easier to learn and allows quick setup using fewer lines of code, which is great for small or fast projects.

**Flexible data storage:** Python can store data in many formats like text files, CSV, or even databases using tools like sqlite3 or SQLAlchemy.

**Java is safer for storing data:** Java uses features like prepared statements, which help prevent security problems like injection attacks, making it more secure for handling sensitive data.

## Encryption.

**Java is built for secure systems:** Java is commonly used in large scale or sensitive systems like banking apps because it has strong built in support for encryption.

**Strict encryption process:** Java follows a more controlled and structured approach to encryption, which helps developers avoid common security mistakes.

**Ideal for high-security projects:** Java is a good choice when data protection is critical, especially in business or enterprise level applications where reliability matters most.

**Python is easier for beginners:** Python is simpler to learn and works well for smaller projects or when you're just starting to use encryption.

**Has helpful libraries:** Python provides libraries like cryptography and PyCrypto that make adding encryption to programs quick and easy.

**More flexibility, but more risk:** Python doesn’t force strict rules like Java, so there’s a higher chance of making mistakes if you don’t fully understand how encryption works.

**Best use cases:** Use Java when building secure systems where reliability and safety are priorities. Use Python for learning, testing, or smaller projects, with regards to safety.

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

**Python is great for prototyping:** Python is often used for early development because it lets you write code quickly and easily without needing to set up too much.

**Simple and readable syntax:** Python’s code looks clean and is easy to understand, so you can turn ideas into working programs without dealing with complex rules or strict formatting.

**Less setup needed:** You don’t have to worry about declaring variable types or creating detailed class structures right away, which saves time during the planning phase.

**Java is slower to start:** Java needs more setup, like defining data types and creating structured classes, which can slow down the early stages of a project.

**Best workflow:** Start with Python to test and develop ideas quickly, then move to Java if you need a more secure, structured, or scalable final version.

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

## Visual Code IDE

**Microsoft Visual Basic was used:** This was chosen as the main compiler and development environment for building the program.

**Easy to use interface:** Visual Basic has a user friendly layout with drag and drop tools, which makes it simple to design windows, buttons, and other parts of the user interface.

**Quick development process:** It allows you to build programs quickly with minimal code, which is helpful for beginners or when time is limited.

**Ideal for Windows apps:** Visual Basic is designed for making Windows based applications, so it’s great for creating desktop software with a clean, familiar look.

**Speeds up testing and debugging:** The built in tools in Visual Basic make it easier to find and fix errors, test how the program runs, and make changes on the go.

## Python Compiler

**Python was the main language used:** It was chosen for writing and running the code throughout the project.

**Fast for prototyping:** Python’s simple and readable syntax helps speed up development, especially when building and testing early versions of a program.

**Great for quick changes:** Because Python doesn’t need strict setup like other languages, it’s easier to make changes and try out new ideas quickly.

**Python was preferred for flexibility:** Even though Java is more structured, Python’s flexibility and shorter code made it a better fit for this project, especially in the early stages.

## Artificial Intelligence Converter

**Used ChatGPT to convert code:** I used ChatGPT to help translate my Python code into Java.

**AI-powered support:** ChatGPT is a powerful AI tool that can generate and explain code in different programming languages.

**Helpful for learning:** The process also helped me understand how certain Python features translate into Java, which improved my coding knowledge in both languages.

**Time saving:** Using ChatGPT sped up the conversion process and helped avoid common mistakes during manual translation.

## Java

**Java chosen for its strengths:** Java offers advanced security features, making it reliable for complex or sensitive projects.

**Testing and refining in Python:** The main gameplay and core functions were built and improved in Python because changes could be made quickly and easily.

**Translation to Java later:** After the Python version was stable, the project was converted into Java to benefit from its stricter structure, better error detection, and improved performance for bigger, more demanding applications.

**Balanced approach:** Using both languages allowed rapid development at first, followed by a more secure and efficient final product.

## Code is commented and following industry standard practices

**Code is well commented:** Comments are added throughout the code to explain what different parts do, making it easier to follow.

**Follows industry standards:** The project uses best practices like consistent indentation and clear formatting to keep the code neat.

**Clear documentation:** Important parts of the code and its overall design are documented, providing helpful information for anyone reading or updating it.

**Easier to maintain and debug:** These practices make it simpler for others or future developers to find issues, add features, or fix bugs.

## Justification of Git and GitHub and their difference

**Code hosted on GitHub:** The project’s source code was stored on GitHub, a popular web platform for managing and sharing code.

**Git is the foundation:** Git is the core version control system that tracks changes to files and helps manage different versions of the code.

**GitHub adds collaboration:** GitHub builds on Git by providing an easy to use interface with features like issue tracking, code reviews, and team collaboration tools.

**Main difference:** Git is the software managing code history locally, while GitHub hosts repositories online and enables multiple people to work together smoothly.

**Benefits:** Using GitHub makes it easier to share progress, manage bugs, and review updates from other contributors.

## Frequency of committing Code

**Frequency:** Completed Python code over the course of one week, and Java code over 4-5 days.

# 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!!")

#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\_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 = []

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

# Initilizations 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 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)

## Readme File for Python

How do I start the Project

# 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

# Reflection

What did you learn from this project?

**Learned to code in both Python and Java**: I gained experience writing the same game in two different programming languages.

**Python helped with fast development:** Python’s simple syntax made it easy to build and test the game quickly during the early stages.

**Java taught me structure and safety:** Converting the game to Java helped me understand the importance of strong typing, error checking, and organized code.

**Improved problem solving:** Working in two languages challenged me to think differently and solve coding problems in new ways.

**Used AI tools:** I learned how ChatGPT can assist in translating and refining code between languages.