INFO1111: Computing 1A Professionalism

2024 Semester 1

Self-Learning Report
Task 2: Advanced: Tool/Technology

Student Name: XingChen Li
Student ID: 530376956

Submission number: 2

Github link: ??

Instructions

***Important:** This section should be removed prior to submission.*

You should use this Word template to generate your self-learning report. Keep in mind the following key points:

- **Submissions**: There will be three opportunities during the semester to submit a self-learning. For each submission you can attempt one task (the Foundation Report is Task 1) and aim for a rating of 'OK' or 'STRONG'. Each submission should use the same report template but amended to include new information. You can only attempt the Advanced Task (the Advanced Report is Task 2) after you have achieved a 'STRONG' on the Foundation Task.
- **Minimum requirement:** There is no minimum requirement
- **Using this template:** When completing each section you should remove the explanation text and replace it with your material.
- **Referencing:** You should also ensure that any resources you use are suitably referenced, and references are included into the reference list at the end of this document. You should use the IEEE reference style [1] (the reference included here shows you how this can be easily achieved)

Overview of the tool/technology to be self-learnt

For Task 2: Advanced you need to select a topic from the 'Self-Learning: List of Topics' list on Canvas.

Steps ratings for self-learning your selected technology

The following is a list of steps you need to carry out to meet the goals of this report to achieve specific ratings. For each step you must provide evidence that you have successfully carried out that step, as described in Section 4 above.

OK Rating

1. Demonstrate your understanding of the tool/technology.
   1.1. Identify three things you will do to demonstrate your understanding of the tool/technology you have chosen and your ability to use it. One of these must be creating a practical application that might also be useful.
   1.2. Application artefacts
      - Include here a description of what you actually created to demonstrate your understanding of the artefact (what does it do? How does it work? How did you create it, what could it be used for?) (50-100 words)
      - Include any code or other related artefacts that you created (these should also be included in your GitHub repository).

- If you do include screengrabs to show what you have done then these should be annotated to explain what it is showing and what the application does.

2. Show that you have actually understood the tool or technology at a relatively advanced level. You will need to analyse it by e.g. comparing it to alternatives, identifying key strengths and weaknesses, and the areas where this tool is most effective.
   2.1. Strengths
   What are the key strengths of the item you have learnt? (50-100 words)
   2.2. Weaknesses
   What are the key weaknesses of the item you have learnt? (50-100 words)
   2.3. Usefulness
   Describe one scenario under which you believe the topic you have learnt could be useful. (50-100 words)
   2.4. Key Question 1
   Note: This question is in the table in the `Self Learning: List of Topics' page on Canvas. (50-100 words)
   2.5. Key Question 2
   Note: This question is in the table in the `Self Learning: List of Topics' page on Canvas. (50-100 words)
3. Demo what you have created to your tutor in the tutorial

STRONG Rating

To achieve a STRONG rating, in addition to the above, you will need to do the following.

1. Alternative tools/technologies - analysis and comparative evaluation
   1.1. Identify 2 alternative tools/technologies that can be used instead of the one you chose for your topic. (e.g. if your topic was Python, then you might identify Java and Golang)
2. Comparative Analysis
   2.1. Describe situations in which both your tool/technology and each of the identified alternatives would be preferred over the others (100-200 words).

1. Tool/Technology Selection

1.1. What is the tool/technology you have selected?

I selected Pygame.

1.2. Why have you selected this tool/technology?

Pygame uses the Simple DirectMedia Layer (SDL) library, with the intention of allowing real-time computer game development without the low-level mechanics of the C programming language and its derivatives. This is based on the assumption that the most expensive functions inside games can be abstracted from the game logic, making it possible to use a high-level programming language, such as Python, to structure the game. [2]

1.3. What benefits do you think learning this tool/technology will give you in the IT industry?

The high-level nature of Python and the abstraction provided by Pygame allow for rapid prototyping of game ideas without sacrificing the power and flexibility needed for creating sophisticated games, and easly gain a solid understanding of core game development concepts such as event handling, frame rates, collision detection, and multimedia management. This makes it an excellent tool to make a game for the beginner like me.

2. Journal of self-learning activities and reflections. *Use the template **Journal of self-learning activities an reflections**) to record each of your activities as you do them, along with your thoughts about the activity. Submit this journal weekly to show and discuss in each week's tutorial. Use the journal as a source of information for sections 5 and 6 of this template.*

3. Self-learning Plan

3.1. **Goals**:

Your goals for this report are to:

1. Demonstrate your knowledge (understanding) of the tool/technology you have selected by carrying out the steps specified in the instructions.
2. Demonstrate your skill in applying that knowledge to create artefacts using your tool/technology by carrying out the steps specified in the instructions.
3. Analyse, compare and evaluate your tool/technology in a broader context by carrying out the steps specified in the instructions.
4. Evaluate your self-learning processes.

3.2. Resources

3.2.1. Information resources required

Identify the resources you plan to use (add more rows if required).

TITLE	AUTHOR	PURPOSE	REFERENCE & LINK
What is the title of the resource?	*Who is the author of the resource?*	*What will you use this resource for (e.g. learning how to create a project).*	*Include the reference for the source and the link (shorten the link if it is long).*
Pygame	*Wikipedia*	To learn the basics and functionalities of Pygame.	Pygame -Wiki- Pygame - Wikipedia
Pygame tutorial	Tech With Tim	For practical, step-by-step guidance on using Pygame.	Pygame Tutorial #1 - Basic Movement and Key Presses (youtube.com)
Unity (Game Engine),	Wikipedia	To understand Unity's features, history, and capabilities.	Unity (game engine) - Wikipedia
Godot (Game Engine)	Wikipedia	To learn about Godot's design, features, and advantages.	Godot (game engine) - Wikipedia

3.3. Schedule

3.3.1. *List and describe the steps you will take to execute your plan and when you will complete them.*

STEPS NO	DESCRIPTION	DATE
1	I choose the tool and decide the application that I want to create.	11/4

2	I will draw the key items on the screen, sprites	12/4
3	I will achieve responding to user input to control the sprite	12/4
4	I will make the sprite responding to events	12/4
5	Add Sound/music to the application	13/4
6	Change the sprite to some fun image	13/4
7	I will analyse the tool at strengths and weaknesses, and the areas where this tool is most effective.	14/4
8	I will Identify 2 alternative tools and explain	15/4

4. Results: evidence of the steps you have attempted from the instructions

4.1 I will Identify three things to demonstrate my understanding of all the Key concepts of **Pygame** from ***Self-Learning: List of Topics (Advanced Task).***

1, Install the Pygame library.

2, Write an application that called "Dodge the Shito", In this game, the red block controlled by the player needs to avoid the blue blocks falling from the top of the screen. If the player's block is touched by a blue block, the game ends. The player should try to survive as long as they can.

It is a simple game using the following aspects:

- Drawing key items on the screen, sprites
- Responding to user input
- Responding to events

3, Demonstrate the Sound/music by adding music to the game and I add image to the sprite.

• Drawing key items on the screen, sprites • Responding to user input • Sound/music • Responding to events	• Is PyGame useful for 3D First Person games? Or only 2D games? • How good a Python programmer do you need to be before you can use PyGame effectively?

4.1.0 Install the Pygame library.

First, Open the command prompt or terminal on your computer.
Second, enter the following command in the command line to install Pygame.

```
pip install pygame
```

Third, to verify the installation after the installation is complete, you can run one of Pygame's included example games with the following command:

```
python -m pygame.examples.aliens
```

4.1.1 Drawing key items on the screen, sprites.

i) First, we need to initialize Pygame and create a window where the sprite will be displayed.

`Display` is a submodule of Pygame that contains various functions for working with the game display or screen. It lets you create, manage, and manipulate the game window.

When you call `pygame.display.set_mode(size),` it returns a Surface object representing the visible part of the window.[3]

```
import pygame
import random

# initialize
pygame.init()

# set the dimensions of the window
size = (700, 500)
screen = pygame.display.set_mode(size)

# set the title of the window
pygame.display.set_caption("Dodge the Blocks")
```

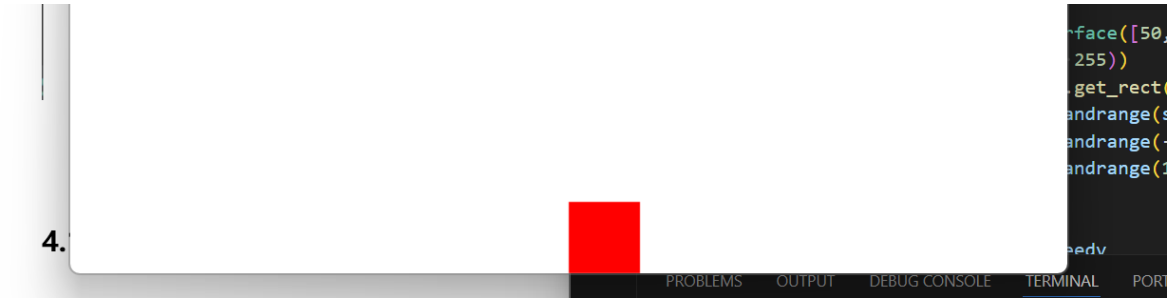
ii) Second, Sprite is an abstract concept in Pygame that represents a movable object in the game. We can create the Sprite object by inheriting the `pygame.sprite.Sprite` class.

We use `self.image = pygame.Surface([50,50])` to initialize a 50x50 pixel square and we use `self.image.fill((255, 0, 0))` to fill the square with red colour.

At last, we use can make the square positioned at the bottom center of the window.

```
        self.rect = self.image.get_rect()
        self.rect.x = 350
        self.rect.y = 450
```

```
# Creating the Player Sprite
class Player(pygame.sprite.Sprite):
    def __init__(self):
        super().__init__()
        self.image = pygame.Surface([50, 50])
        self.image.fill((255, 0, 0))
        self.rect = self.image.get_rect()
        self.rect.x = 350
        self.rect.y = 450
```



iii) Third, We create the enemy Sprites. This code defines an Enemy class, also a subclass of `pygame.sprite.Sprite`, representing the game's enemies. We create the enemy Sprites in the same way we create the player Sprites.

```
# create enemy sprites
class Enemy(pygame.sprite.Sprite):
    def __init__(self):
        super().__init__()
        self.image = pygame.Surface([50, 50])
        self.image.fill((0, 0, 255))
        self.rect = self.image.get_rect()
        self.rect.x = random.randrange(size[0] - self.rect.width)
        self.rect.y = random.randrange(-150, -100)
        self.speedy = random.randrange(1, 4)

# update the position of the enemy sprites
    def update(self):
        self.rect.y += self.speedy
        if self.rect.top > size[1]:
            self.rect.x = random.randrange(size[0] - self.rect.width)
            self.rect.y = random.randrange(-100, -40)
            self.speedy = random.randrange(1, 4)
```

Each enemy is a 50x50 pixel blue square that starts at a random position above the screen and moves downwards at a random speed.

```
        self.rect.x = random.randrange(size[0] - self.rect.width)
        self.rect.y = random.randrange(-150, -100)
        self.speedy = random.randrange(1, 4)
```

This line `self.rect.x = random.randrange(size[0] - self.rect.width)` randomly determines the horizontal position (**x** coordinate) of the enemy sprite within the game window.

If an enemy moves off the bottom of the screen, it reappears at the top with a new random speed and horizontal position.

```
# update the position of the enemy sprites
    def update(self):
        self.rect.y += self.speedy
        if self.rect.top > size[1]:
            self.rect.x = random.randrange(size[0] - self.rect.width)
            self.rect.y = random.randrange(-100, -40)
            self.speedy = random.randrange(1, 4)
```

4.1.2 Responding to events.

i) In Pygame, all operations are implemented through events. We can use the `event.get()` method provided by Pygame to obtain all events, and judge the user's operation by the event type. For example, `event.type == pygame.QUIT` shows the event is typically triggered by actions clicking the close button on the window's title bar:

```
# game loop
done = False
while not done:
    # Handle Event
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done = True
```

ii) This code checks for collisions between the player sprite and any of the enemy sprites using `hits = pygame.sprite.spritecollide(player, enemies, False).`

If there is a collision `(if hits:)`, it sets `done = True` , effectively ending the game.

```
# checks for collisions
hits = pygame.sprite.spritecollide(player, enemies, False)
if hits:
    done = True  # If there is a collision effectively ending the game.
```

The `pygame.sprite.spritecollide()` function is used in Pygame to detect collisions between a single sprite and a group of sprites.

```
pygame.sprite.spritecollide(player, enemies, False)
```

We can draw background and all sprites using:

```
    # display background and all the sprites
    screen.fill((255, 255, 255))
    all_sprites.draw(screen)
```

4.1.3 Responding to user input.

i) In Pygame, to response user input we use the key module[5] and mouse module provided by Pygame to detect keyboard and mouse input. For example the Sprite moves right and left :

```
    # move the player
    keys = pygame.key.get_pressed()
    if keys[pygame.K_LEFT]:
        player.rect.x -= 5
    if keys[pygame.K_RIGHT]:
        player.rect.x += 5
    if keys[pygame.K_UP]:
        player.rect.y -= 5
    if keys[pygame.K_DOWN]:
        player.rect.y += 5
```

This code below checks if the key is pressed. Each key on the keyboard is represented in this list, and the value is True if the key is pressed down at the moment and False if it's not.

```
    keys = pygame.key.get_pressed()
```

This code checks if the left arrow key is pressed using `if keys[pygame.K_LEFT]:` .If so, it moves the player sprite to the left by 5 pixels with `player.rect.x -= 5`. The `player.rect.x` refers to the sprite's horizontal position in the game window, and

decreasing this value moves the sprite left.

```
    if keys[pygame.K_LEFT]:
        player.rect.x -= 5
```

4.1.4 Sound/music

i) Sound and music are also very important elements in games. Sounds and music can be loaded and played using the mixer module provided by Pygame. I will play the file `Decisive Battle.mp3` and the `-1` argument indicates that the music should loop indefinitely.

`pygame.mixer.music` is a submodule of Pygame that provides functionality for handling sound effects and music in games. It allows developers to load, play, pause, resume, stop, and control the volume of audio files.

For example :

```
# Load Music
pygame.mixer.music.load("Decisive Battle.mp3")
# Play music
pygame.mixer.music.play(-1)
```

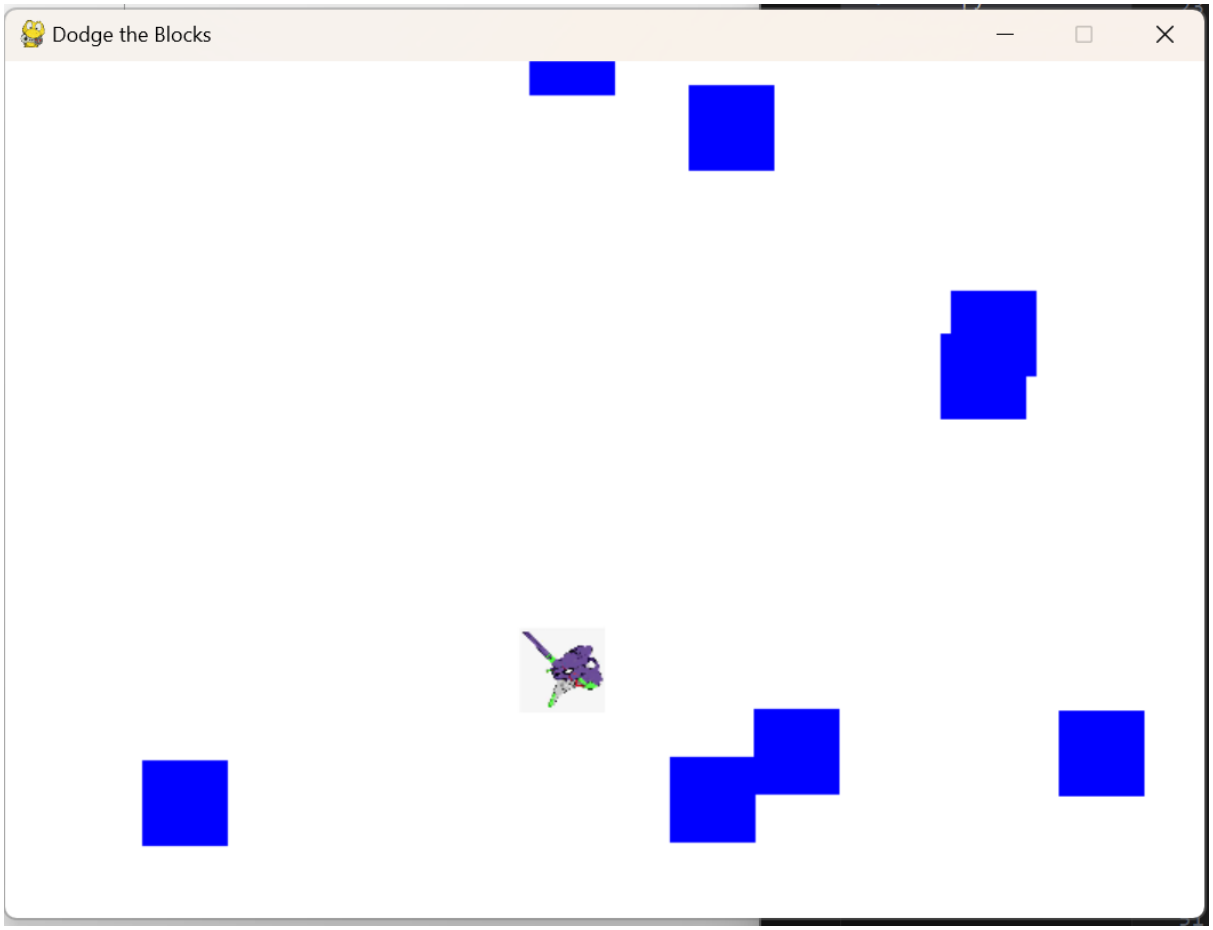
4.1.5 Demonstration of game and an explanation of its gameplay

In this game, the red block controlled by the player needs to avoid the blue blocks falling from the top of the screen. If the player's block is touched by a blue block, the game ends. The player should try to survive as long as they can.

I changed the red sprite to the head of Evangelion Mark.01[7] , it is fictional biomechanical humanoid mechas introduced in the Japanese anime television series Neon Genesis Evangelion [8].

I load the image with code below:

```
pygame.image.load('eva01.png')
```



4.2 Strengths,Weaknesses and Usefulnesses and Key questions of Pygame

I will analyse the Pygame to show that I have actually understood the tool at a relatively advanced level. I will identifying key strengths and weaknesses, usefulnesses and two Key Questions.

4.2.1 Strengths of Pygame.[6]

1,Versatile Development Tools :

Pygame is a comprehensive library for game development, offering tools for event handling, sprite operations, frame pacing, and rendering text, ensuring broad compatibility across platforms.

2,Media Processing and Game Physics:

It simplifies creating 2D and pseudo-3d games with built-in audio and image processing functionalities that facilitate the loading and playing of media files.

3,Cross-Platform Efficiency :

Additionally, Pygame comes equipped with essential mathematical functions for vector calculations and collision detection, crucial for realizing game physics. Its ease of use and efficient performance, coupled with strong cross-platform capabilities, make it suitable for running on Windows, MacOS, and Linux, streamlining the game development process.

4.2.2 Weaknesses of Pygame.

1, Limited to 2D Graphics:

Pygame is great for making simple 2D games,[4] but if you're dreaming of creating complex 3D games, it might not cut it. You can use Pygame to make pseudo-3d games

2, Performance Issues:

Since it's built on Python, it can be slower than games made with lower-level languages like C++. For big, complex game projects, Pygame might feel a bit basic.

3, Limited Mobile Support:

Plus, if you're aiming to develop games for mobile, Pygame isn't the easiest path since its mobile support is kinda lacking.

4.2.3 Usefulnesses of Pygame.

Pygame can be used in game development to quickly test feedback on gameplay.

Developers can quickly build prototypes to implement the basic functions and gameplay of the game, and interact in real time through the event processing and graphics rendering functions provided by Pygame.

This rapid iterative development process allows developers to quickly obtain feedback on gameplay and adjust and optimize the game experience in a timely manner.

Because Pygame is easy to learn and use, developers can quickly create simple prototypes and test different game mechanics and gameplay, thus accelerating the game development process.

4.2.4 Key Question 1: Is PyGame useful for 3D First Person games? Or only 2D games?

No, Pygame is a wrapper for SDL, which is a 2D api. Pygame doesn't provide any 3D capability and probably never will.

3D libraries for Python include Panda3D and DirectPython, although they are probably quite complex to use, especially the latter.

Pygame is primarily designed for 2D game development, and while it does have some limited support for 3D graphics, it's not well-suited for complex 3D games, especially first-person games. Pygame's 3D capabilities are minimal compared to dedicated 3D game development frameworks like Unity or Unreal Engine.

4.2.5 Key Question 2: How good a Python programmer does you need to be before you can use PyGame effectively?

To use Pygame effectively, we should have a basic understanding of Python programming concepts such as variables, data types, loops, conditionals, functions, and classes.

Although Pygame is suitable for beginners, as we use Pygame we may encounter more advanced topics such as collision detection, sprite management, and game physics, which may require a deeper understanding of Python programming concepts.

Overall, we don’ t need to be expert Python programmers to start using Pygame, but having a solid foundation in Python will definitely speed up our learning process and enable us to utilize Pygame more effectively.

4.3 Alternative tools/technologies - analysis and comparative evaluation

4.3.1 Unity [9] and Godot [10]

Unity and Godot can be used instead of the Pygame.

Unity is a cross-platform game engine that supports both 2D and 3D game development.

Godot is an open-source game engine that supports both 2D and 3D game development.

4.3.2 Pygame vs Unity vs Godot

Pygame: Suitable for beginners and educational purposes, especially for those already familiar with the Python language. Pygame is great for developing simple 2D games and prototypes, or when you want to focus on programming fundamentals rather than the complexities of game development. For small projects or as a tool to learn programming and game logic, Pygame is a great choice.

Unity: Ideal for developing commercial-grade 2D and 3D games, especially when you need to publish cross-platform or take advantage of Unity's powerful physics engine and advanced rendering capabilities. With its large asset store and community support, Unity is ideal for developers who want to pursue game development as a career, or for projects that require the implementation of high-quality visuals and complex game mechanics.

Godot: Godot is an excellent choice for developers looking for an open source solution, especially those independent developers or small teams focused on 2D game development. Its flexibility and ease of use make it a top choice for those who want complete control over their projects, while its support for 3D games is constantly improving, becoming a comprehensive game development tool.

5. Evaluation

5.1. Knowledge and skills

QUESTIONS	YOUR ANSWER
1. To what extent did you reach the goals for this report?	I can fully understand the Pygame as beginner and how to use Pygame basic functions to implenment the simple applications.
2. What barriers did you face in reaching the goals?	Pygame needs python experience but I have forgot the python, so I have to review the

	python code and learn the pygame code together.
3. What worked well for you in doing the report?	There are many resouse about Pygame online including tutorial video and report. So that I can easly find informations like : how to add music to the game and so on...
4. What was frustrating?	I cannot use Pygame to make a real 3D game and It was hard to conceive a game
5. Other?.....	Pygame is really fun!

5.2. Self-learning learning processes

QUESTIONS	YOUR ANSWER
1. What worked?	developing simple games quickly and learning basic game mechanics worked.
2. What didn't?	Complex 3D game development didn't work due to Pygame's limitations.
3. What would you do differently?	Would use Pygame for prototyping and switch to more advanced engines for full development.
4. What did you learn about yourself?	I enjoy game development and am creative in gameplay.
5. What recommendation would you make to your future self?	explore and learn more advanced game development tools early.
6. What would you recommend to someone else?	Start with Pygame to grasp fundamentals, then progress to Unity or Godot.
7. Other?.....	Try experimenting with different types of games to broaden your skills.

6. Learning sources

Learning Source - What source did you use? (Note: Include source details such as links to websites, videos etc.). Contribution to Learning - How did the source contribute to your learning (i.e. what did you use the source for?). You may use information from your Journal for this.

Learning Source - What source did you use? (Note: Include source details such as links to websites, videos etc.).	Contribution to Learning - How did the source contribute to your learning (i.e. what did you use the source for)?
Pygame -Wiki- Pygame - Wikipedia	Learn about Pygame history
python Pygame basic game development -CSDN	Learn how to use Pygame functions and introduction of Pygame
python - Does PyGame do 3d? - Stack Overflow	Finding out why pygame is not working on 3D game
Pygame Tutorial #1 - Basic Movement and Key Presses (youtube.com)	Deeper learning about pygame movement and ket presses event
The Pros And Cons Of Pygame For Game Development ⚖ - Code With C	Find the Pros and Cons of Pygame for Game Development
Angels in Neon Genesis Evangelion - Wikipedia	Improve the gameplay and game theme to attract the players

Bibliography

[1] The University of Sydney, _Referencing and citation styles: IEEE,_ 2022, see https: //libguides.library.usyd.edu.au/c.php?g=508212.

[2] "Pygame," *Wikipedia*, 22-Feb-2024. [Online]. Available: Pygame - Wikipedia. [Accessed: 13-Apr-2024]

[3] *CSDN*, 2020. [Online]. Available: python Pygame basic game development -CSDN. [Accessed: 13-Apr-2024]

[4] Matthew HaywoodMatthew Haywood 3633 silver badges1313 bronze badges, "Does PyGame DO 3D?," *Stack Overflow*, 01-Nov-1956. [Online]. Available: python - Does PyGame do 3d? - Stack Overflow. [Accessed: 13-Apr-2024]

[5] Pygame tutorial #1 - basic movement and key presses," *YouTube*, 07-Nov-2017. [Online]. Available: Pygame Tutorial #1 - Basic Movement and Key Presses (youtube.com). [Accessed:13-Apr-2024]

[6] CodeLikeAGirl, CodeLikeAGirl, and Name, “The Pros and cons of pygame for game development ⚖️,” *Code with C*, 14-Oct-2023. [Online]. Available: The Pros And Cons Of Pygame For Game Development ⚖️ - Code With C. [Accessed: 13-Apr-2024]

[7] “Evangelion (mecha),” *Wikipedia*, 29-Feb-2024. [Online]. Available: Evangelion (mecha) - Wikipedia[Accessed: 13-Apr-2024]

[8] “Neon genesis evangelion,” *Wikipedia*, 06-Apr-2024. [Online]. Available: Neon Genesis Evangelion - Wikipedia. [Accessed: 13-Apr-2024]

[9] “Unity (Game Engine),” *Wikipedia*, 07-Apr-2024. [Online]. Available: Unity (game engine) - Wikipedia[Accessed: 13-Apr-2024]

[10] “Godot (Game Engine),” *Wikipedia*, 11-Apr-2024. [Online]. Available: Godot (game engine) - Wikipedia. [Accessed: 13-Apr-2024]