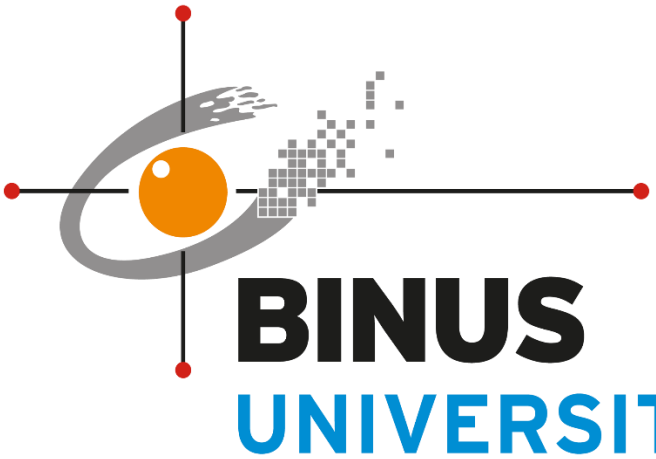
Final Project

Algorithm & Programming



Project name: “Multi Player Battleship”

Student name: William Ekapanna Rusmana

Student ID: 2802523111

Class L1CC

TABLE OF CONTENTS:

A. Specification

I. Game Overview

- A two-player naval combat game
- Players take turns attacking each other's grid to sink ships
- Turn-based gameplay with time limits

II. Core Features

SETUP PHASE

- Each player can place 5 different ships consisting of:
  - Carrier (5 cells)
  - Battleship (4 cells)
  - Cruiser (3 cells)
  - Submarine (3 cells)
  - Destroyer (2 cells)
- Ships can be rotated positioned either horizontally or vertically
- Ships can be reset to readjust ship position
- Ships cannot overlap

BATTLE PHASE:

- Turn-based combat with 15-second time limit per turn if exceeded move on to opponent
- Red player goes first
- Players can make single attack using the “Nuke”:
  - Becomes available after specific rounds
  - Destroys a 4x4 area at once
  - Can only be used once per player
  - Red’s activate on the fourth turn, one turn here refers to the transition from red to blue and vice versa
  - Blue’s activate on the third turn

III. Variables and constants

Screen settings:

```
SCREEN_WIDTH = 1400    # Window width in pixels
SCREEN_HEIGHT = 850    # Window height in pixels
```

Grid settings

```
GRID_SIZE = 10      # Number of cells in grid (10x10)
CELL_SIZE = 50      # Size of each cell in pixels
GRID_LEFT_TOP_RED = (50, 70)    # Starting position of red player's grid
GRID_LEFT_TOP_BLUE = (850, 70)  # Starting position of blue player's grid
```

Time Constans and Variables

```
TIME_PER_TURN = 15    # Time limit per turn in seconds
turn_start_time       # Start time of current turn
red_timer           # Red player's remaining time
blue_timer          # Blue player's remaining time
minutes = 0         # Game time in minutes
```

Ship Specifications

```
# Ship Identifier dictionary
SHIPS = {
        "Carrier": 5,
        "Battleship": 4,
        "Cruiser": 31,    # Note: 31 is identifier for Cruiser
        "Submarine": 32,  # Note: 32 is identifier for Submarine
        "Destroyer": 2
}

# Size of ships in grid cells
```

```
SHIPS_SIZES = {
            "Carrier": 5,
            "Battleship": 4,
            "Cruiser": 3,
            "Submarine": 3,
           "Destroyer": 2
}
```

Colors

```
WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
BLUE = (33, 19, 235)
RED = (225, 34, 34)
GRAY = (200, 200, 200)
GREEN = (0, 255, 0)
OCEAN_BLUE = (21, 127, 233)
WHITE_GRAY = (240, 240, 240)
YELLOW = (255, 255, 0)
DARK_YELLOW = (214, 186, 24)
```

Game State Variables

```
current_player = "red"     # Tracks whose turn it is
ship_orientation = "horizontal"   # Current ship placement orientation
round = 0              # Game round counter
start_game = False       # Controls game start state
fp_setup = True         # First player setup phase
sp_setup = True         # Second player setup phase
round_start = False      # Battle phase state
display_winner = True    # Winner display state
```

Score Variable: Greater score wins

```
red_score = 0          # Red player's score
blue_score = 0         # Blue player's score
```

Ship Deployment Variables

```
selected_ship = None     # Currently selected ship for placement
deployed_ships = []      # Ships placed on grid
red_deployed = []       # Red player's deployed ships
blue_deployed = []      # Blue player's deployed ships
```

Nuke Variables

```
nuke = "deactive"       # Nuke ability state
red_nuke = True        # Red player's nuke availability
blue_nuke = True       # Blue player's nuke availability
```

Class Variables

```
self.grid           # 2D array representing game grid
self.ship_log = []     # Records ship positions
self.eliminated_squares = []  # Tracks attacked positions
self.ship_health = {    # Tracks remaining health of each ship
            "Carrier": 5,
            "Battleship": 4,
            "Cruiser": 3,
            "Submarine": 3,
            "Destroyer": 2
}

self.ship_type       # Type of ship
self.identifier      # Unique ship identifier
self.position        # Current position
self.orientation     # Current orientation
self.size          # Ship size in cells
self.rect          # Ship's rectangle for rendering
```

IV. UI features

- Two 10x10 grids (one for each player)
- Ship visualization
- Turn indicator
- Timer for each player
- Game time tracking
- Visual feedback for hits/misses
- Sound effects for various actions
- Technical Specifications:
- Screen dimensions: 1400x850 pixels
- Cell size of 50 pixels
- Uses custom graphics for ships
- Includes sound effects for:
- Explosions
- Victory/draw sounds
- Intro music
- Explosion sound effect
- Ocean ambience

- Winning sound effect
- Draw sound effect

V. Game flow

1. Home screen with play button
2. Red player ship placement
3. Blue player ship placement
4. Battle phase
5. Winner/Draw display
6. Option to play again

VI. Win and draw conditions

- A player wins when all enemy ships are destroyed
- Scores are tracked across rounds
- Draw when battle is abruptly ended

B. Solution Design

I. Program Requirements

1. Python:

The program requires Python to be installed on the system, ideally python 3 and above.

2. Pygame:

The Pygame library is essential for handling graphics, sound, and user interaction.

II. Class Diagrams

Rectangle class

```
class Rectangle(object):
        def __init__(self, x, y, width, height, color):
            self.rect = pygame.Rect(x, y, width, height)
            self.color = color
```

This class serves as a parent for the Button and Timer displays to inherit from.

Timer_display

```
class Timer_display(Rectangle):
```

```
        def __init__(self, x, y, width, height, color, text_color=GREEN, font_size=60):
            super().__init__(x, y, width, height, color)
            self.font = pygame.font.Font(None, font_size)
            self.text_color = text_color

    def draw(self, screen, current_time) -> None:
        current_time_str = str(current_time)
        if current_time <= 3:
            text_surf = self.font.render(current_time_str, True, RED)
        else:
            text_surf = self.font.render(current_time_str, True, self.text_color)
        text_rect = text_surf.get_rect(center=self.rect.center)
        screen.blit(text_surf, text_rect)
```

Timer display is a child of rectangle with a custom-made draw() method to display a current time object. The draw method here displays the time in red color when time is less than or equal to 3 seconds, and otherwise it will display it inn green color.

Button

```
class Button(Rectangle):    # Child of Rectangle that serves as a button with text and hover color change
  def __init__(self, x, y, width, height, text, text_color, color, hover_color, text_hover_color, action = None):
    super().__init__(x, y, width, height, color)
    self.text = text
    self.font = pygame.font.Font(None, 36)
    self.text_color = text_color
    self.hover_color = hover_color
    self.action = action
    self.text_hover_color = text_hover_color

  def draw(self, screen) -> None:    # Draws button and check for collision for hover effect
    mouse_pos = pygame.mouse.get_pos()  # get mouse position
    text_surf = None
    if self.rect.collidepoint(mouse_pos):  # checks for collison with mouse pos
      pygame.draw.rect(screen, self.hover_color, self.rect)
      text_surf = self.font.render(self.text, True, self.text_hover_color)
    else:
```

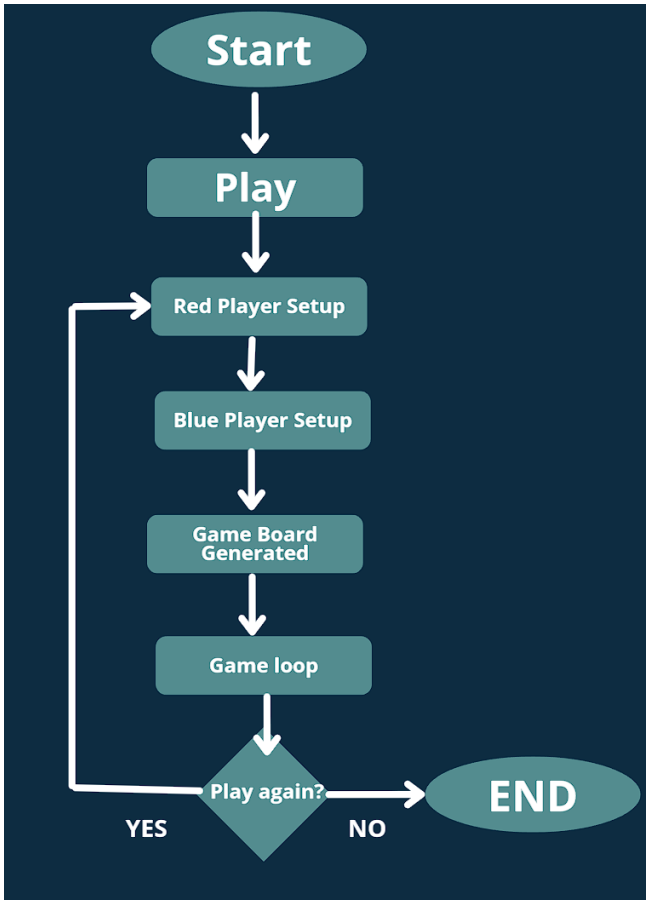
```
        pygame.draw.rect(screen, self.color, self.rect)
        text_surf = self.font.render(self.text, True, self.text_color)

    text_rect = text_surf.get_rect(center=self.rect.center)
    screen.blit(text_surf, text_rect)

def is_clicked(self, event, screen) -> bool:   # Check if button is clicked
    if self.rect.collidepoint(event.pos):   # check if button down is on button
        if self.action is not None:
            self.action(screen)
        return True
    return False
```

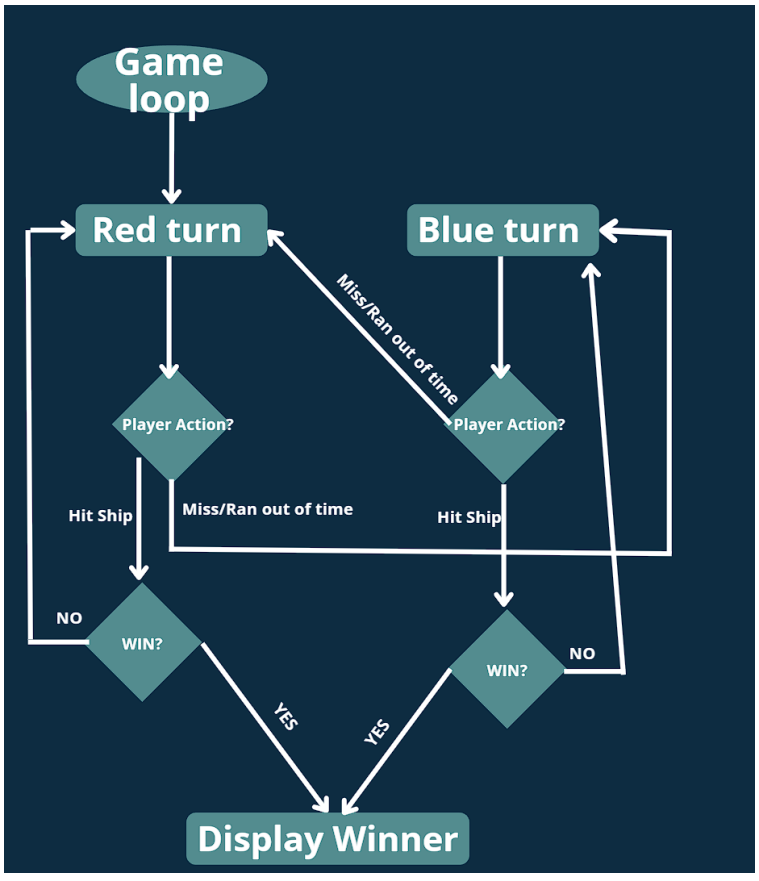
The Button class is also a child of Rectange, used for buttons that interactively change colors when hovered and also execute a function when it's pressed on.

It has methods draw() for drawing the button when it's not hovered and when it's hovered; the last method is_clicked() used to initiate an action in the main program by returning a Boolean value by checking wheter or not the button is pressed on.

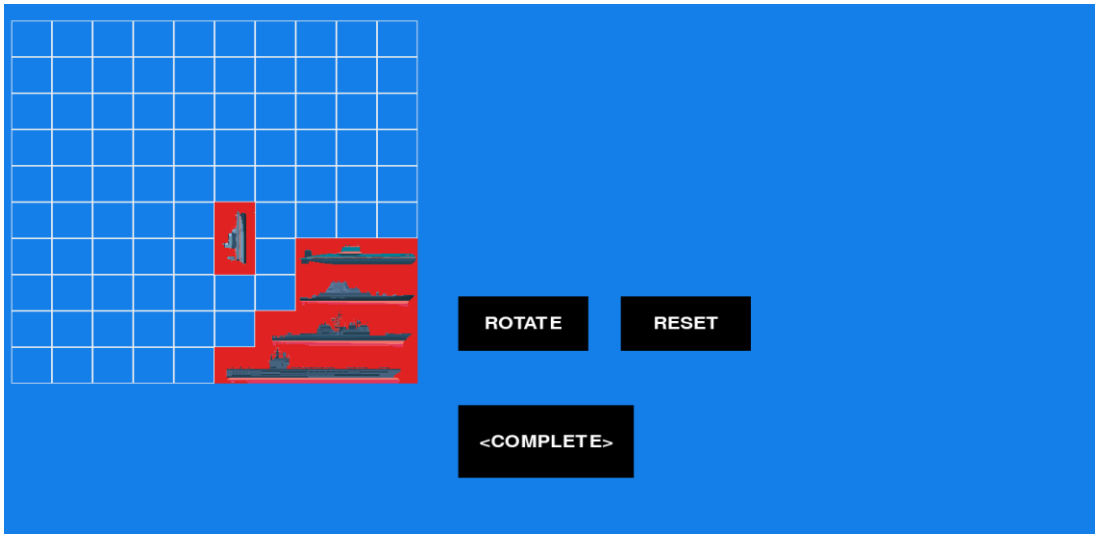
The rest of the Classes, helper functions, and main program have been commented for in the code

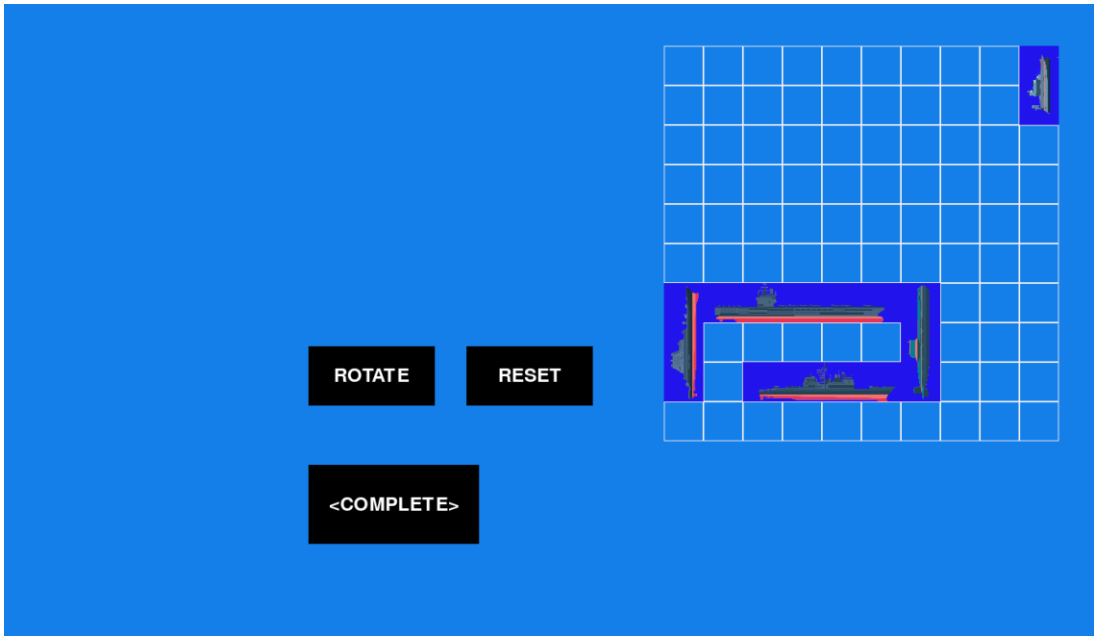
III. Use Case Diagram

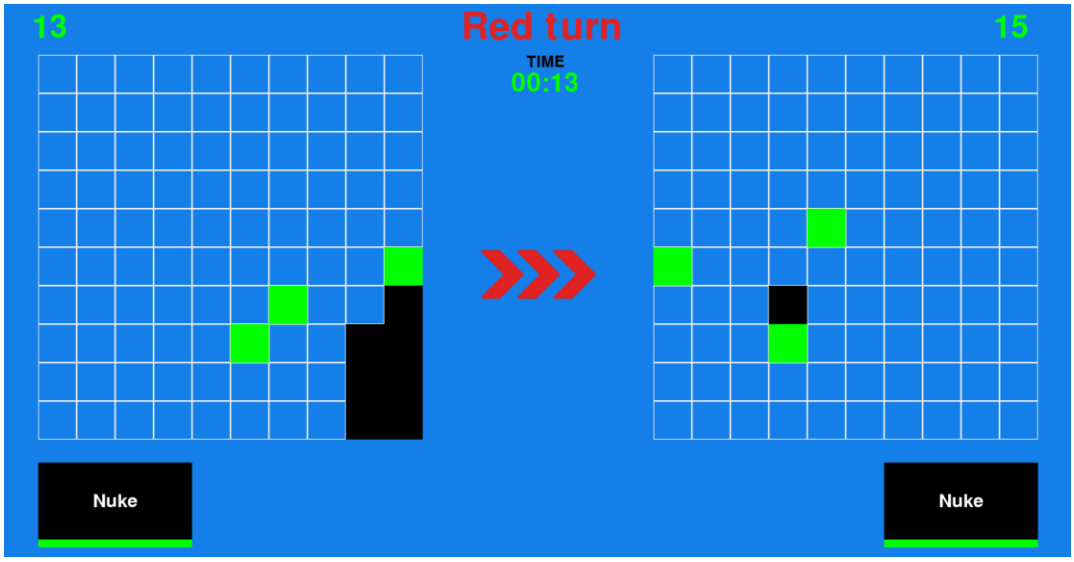

Start
Play
Red Player Setup
Blue Player Setup
Game Board Generated
Game loop
Play again?
YES
NO
END

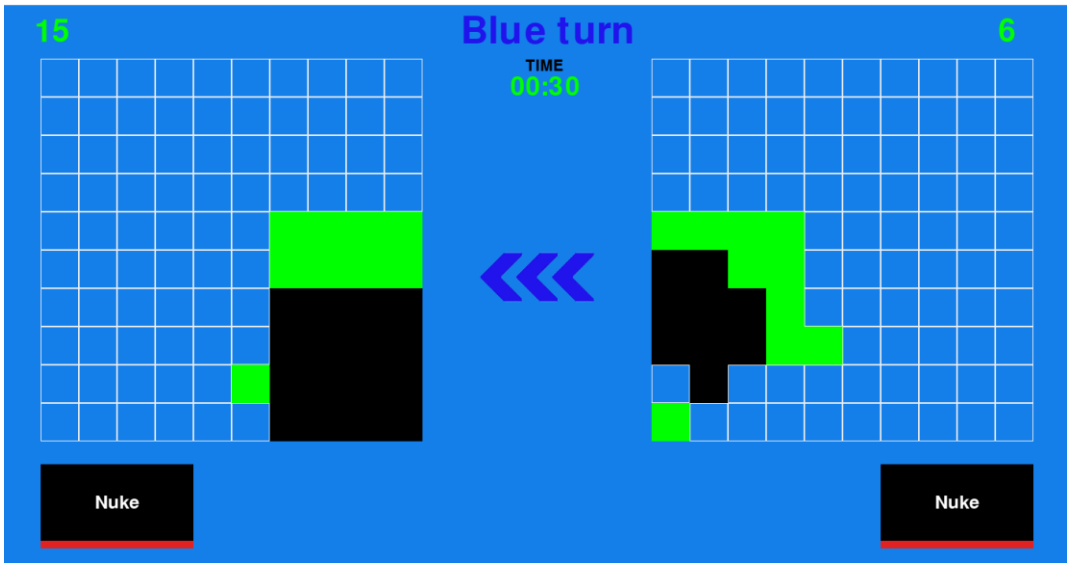

Game loop
Red turn
Blue turn
Miss/Ran out of time
Player Action?
Player Action?
Hit Ship
Miss/Ran out of time
Hit Ship
NO
WIN?
WIN?
NO
YES
YES
Display Winner

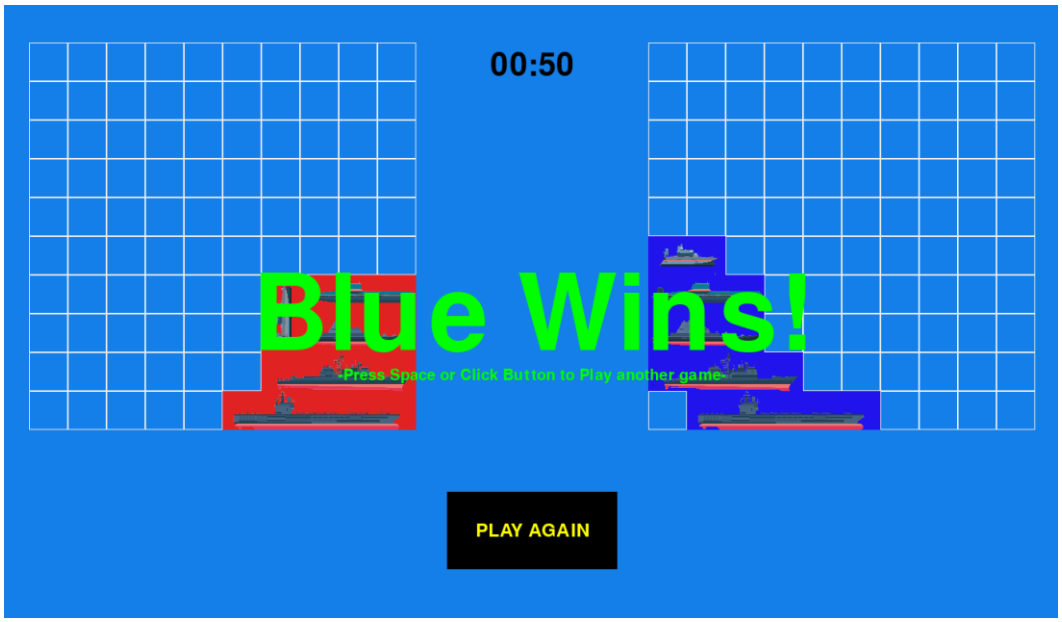
C. Game in Action






13
Red turn
15
TIME
00:13
Nuke
Nuke


15
Blue turn
6
TIME
00:30
Nuke
Nuke


00:50
Blue Wins!
Press Space or Click Button to Play another game
PLAY AGAIN

D. References

Button press beep. (n.d.). *Pixabay*. Retrieved December 3, 2024, from https://www.pixabay.com/sound-effects/button-press-beep-269718/

Cannon sound effect (HD for videos and games). (n.d.). *YouTube*. Retrieved November 29, 2024, from https://www.youtube.com/watch?v=WdFxHb9wpW8

Ew brother ew sound effect. (n.d.). *YouTube*. Retrieved January 5 December, 2024, from https://www.youtube.com/watch?v=zyH33HMYJbo

Freepik. (n.d.). Militaristic ships set: Navy ammunition warship, submarine, nuclear battleship, float cruiser, trawler, gunboat, frigate, ferry. Retrieved December 6, 2024, from https://www.freepik.com/free-vector/militaristic-ships-set-navy-ammunition-warship-submarine-nuclear-battleship-float-cruiser-trawler-gunboat-frigate-ferry_10704121.htm

Good day to die [Epic]. (n.d.). Miguel Johnson. *YouTube*. Retrieved December 6, 2024, from https://www.youtube.com/watch?v=M_k-RQA7lBE&list=PLfP6i5T0-DkL7c2fuDGmpchMT6ie5XlA3

Home ship logo image. (n.d.). *PNGTree*. Retrieved November 6, 2024, from https://pngtree.com/element/down?id=NTY3NjIxOA==&type=1&time=1731405470&token=NDdkZmVlNjEwNDNhY2YwZTBjM2MyMjY5NWQ4ZTVkODQ=&t=0

Ocean sound. (n.d.). *YouTube*. Retrieved December 24, 2024, from https://www.youtube.com/watch?v=B3ZbsOhNTas

Pew pew. (n.d.). *MyInstants*. Retrieved November 21, 2024, from https://www.myinstants.com/en/instant/pew_pew/

We are the champions. (n.d.). Queen. Retrieved November 29, 2024, from https://www.youtube.com/watch?v=d5GkgVhFeZY