

P1: Outline key considerations that support games design

In the gaming industry there are a wide variety of key and unique features that make a game different from others. The first thing you would need to do for developing a game is understanding the target audience. Game designers will need to consider multiple factors such as the age group, the experience, and preferences of the players. This will help shape the overall design, mechanics, and difficulty level to ensure an engaging experience for the player. Game mechanics include the rules, interactions, and systems that determine the gameplay players will get. Game designers must create different mechanics that are balanced and aligned with the game's objectives. Additionally, clear goals and objectives are essential for providing players with a purpose and progression in the game to help keep them motivated and engaged throughout the game.

Key Features in Games:

Games incorporate various key features that serve different purposes. One common feature is the choice between first-person and third-person. **First-person** provides a more immersive experience allowing players to directly see through the eyes of the character as if they are the character experiencing the world around them, while **third-person** offers a broader view of the game environment and character actions. **Player characters** are another important feature, acting as the primary avatars through which players interact with the game world. These characters can be customised, possess unique abilities or attributes, and allow players to project themselves into the game. Non-player characters otherwise known as **NPC'S** are controlled by artificial intelligence and interact with players or other NPCs to give the game more life and story. They can provide quests, assistance, engage in combat, or contribute to the game's story and atmosphere.

Sprites are 2D images or animations used to represent characters, objects, or effects within a game. They play a significant role in creating a visual representation of the game's elements and contribute to its overall aesthetic. Achievement attainment is another key feature that adds an extra layer of engagement. Achievements provide players with specific goals or milestones to strive for, rewarding them for completing challenges, reaching milestones. This feature enhances replay value and gives players a sense of accomplishment. Maintaining player interest throughout the game is essential. This can be achieved through advanced gameplay, amazing storylines, challenging puzzles, dynamic environments, and other content that can factor into this. Player interaction and controls also play a crucial role in game design, as intuitive and responsive controls are necessary for player engagement and satisfaction as if the controls for the game are bad it can cause the player to not want to play it anymore.



Launching a game is the easier part when it comes to game development however maintaining the players interest in the game is a much more difficult task as this would have to be done through updates to make things such as new content, bug fixes, new area and new gear. The updates will increase the life expectancy of the game from dying (player count dropping) and this can be done in 2 ways: free updates every now and again or paid DLC (downloadable content). **DLC's** are paid expansions for the game, this could be things such as a new quest line that is added and the paid version still allows the game studio to make a profit rather than releasing free updates and not making anymore money. If you are the game studio who are making the game you could incorporate some sort of **achievement** in the game that the players can work towards this can be something like new gear, skin or character. The more things like this that are in the game the more things that people are likely to do and not lose interest in the game. An example of this would be in a game such as 'Call Of Duty' that has mastery camos that the player has to play for around 100 hours to unlock. This is achieved by levelling up every gun and doing all the challenges for them. This makes the user have to play the game for so long and this increases the chance of players making purchases on the game as they can save time by doing this by purchasing the gun from the store if they don't have it unlocked already this is done using microtransactions.



Games like call of duty are classed as a **first person** arcade shooter as there is no ending for the game you just play against other players fighting. This is a popular game and probably the most well known in the shooter category however this game can get repetitive as you are always doing the same thing over and over again to try and maintain people's interests. They implement challenges for guns and character skins to help keep the player count high however after so long the count starts to drop. The method this game uses is every 80 days there is a new season update with new content to keep the player interested in the game with new content through a paid battle pass that includes skins and to level up the battle pass you would need to play loads to level it up.

Some more basic games focus on the **competition** aspect. This would be features such as highscore, fastest time to complete a level and number of levels or rounds completed. An example of this would be an arcade game such as space invaders works based on the amount of the enemies killed by increasing your score and a time it takes you to complete the level this would be displayed on a public scoreboard and the aim of the game is to be as

high on there as possible. Some racing games such as Forza Motorsport have races called time trials where the aim is to complete the race the fastest time possible. In the game there is a global leaderboard and a leader board with friends and this collects the time for the race and the whole world can see your time. This gets the player to play and do the race more time and get the quickest time to maintain the player's interest. They also update the game with new races to try and keep the player interested.



Some story games based games such as Red Dead Redemption 2 have **NPC's** (Non-player character) that help create the environment of the game as these characters add aspects to the story and the environment as in RDR2 it is set in western times and without the NPC the western vibe wouldn't be as visible and engaging as it is. These NPCs create dialogue and movement for the town's story; this could be in the method of cutscenes when the main character speaks to other characters within the story. The main focus of the game is to progress through the story of the game which is around 60 hours of playtime and the game also contains side missions to help increase the players focus on the game.

P2 - Explain the benefits of developing game prototypes

What are prototypes

Prototypes are the first version of something that can be anything from physical products to games. Prototypes are used to find if something will work or how well it could sell. For example, in the gaming industry they use prototypes of games to see if it is possible to do and as well as what changes need to be made to that version to make the next version better. This can be done in many different ways: one could be a visual representation of the game and what it could look like could be something such as a 3d rendering or drawing for the game; this is known as non-working prototyping. Another version of prototyping would be having a basic working concept of the game; this would be simple mechanics already made and some stuff unfinished just to show how the game would work; this is known as a working prototype.

Some prototypes of games get released to specific people or are open to the public to try as the game company is trying to see the reaction people will take to the game this would be things such as trying to see how many people may be intrigued by the game or how many people the server can handle on the game at once. This is a good method for testing a game

as the people play testing it might be able to find bugs or issues with the game that was not visible to the developers before.

Types of prototypes

There a wide range of methods that can be used to create the prototype of the game these methods are paper, wireframe and grey box. Paper prototyping is creating a very simple version of the game or the idea you have for the game using paper that could be done by drawings. These do not need to be detailed and can be rough sketches as it only has to show the concept of the game and now the actual functionality. This method is very basic and not that good for getting an accurate design however this is good for helping you get a better understanding of the game. Wireframe prototyping is where you make a very basic version of your game that doesn't need to be good graphically and this can be done using colourful diagrams, shapes and moving aspects. This is probably the quickest way to make a prototype for the game and it helps you get a better idea of what is going to be created later down the road compared to paper prototyping. Grey box prototyping is where you are creating a more detailed version of the game using grey shapes and some coded aspects. In this method of prototyping you have to have the main gameplay elements in it to make it a proper prototype. This is beneficial for the developer of the game as it helps with testing for bugs and also helps to expand upon further ideas during the testing phase. This version helps get a detailed understanding of the game's physics, code and some basic graphical elements.

Testing concepts

Testing is paramount to ensuring a high-quality final product. Functionality testing involves verifying movement, menus, controls, graphics, audio, and gameplay against specifications provided by the game studio. Combinatorial testing explores various input combinations to uncover bugs and glitches from specific scenarios. Exploratory testing allows playtesters to freely explore the game, uncovering issues allowing them to not be caught by scripted scenarios. Compatibility testing assesses the game's performance across different platforms, browsers, and hardware configurations for example testing a release on xbox and pc to see what it performs better on. Cleanroom testing rigorously checks for bugs pre-release, utilising both manual and automated methods. Regression testing rechecks the entire game to catch any new bugs introduced by updates. Playtesting, conducted by a selected group, identifies bugs during gameplay, ensuring the game is polished for release. Some examples of play testing include alpha and beta testing as well as early access versions of games. Alpha is where a select group of people get to play the prototype version of the game. A game company that does this a lot is Call of Duty. To get access to the closed alpha of their new COD game you have to preorder the game to get access. The beta is a more refined version of the alpha that includes some of the bug fixes that was needed in alpha most beta of games are open to anyone to try. This is also a good time to see how many people can get onto the server at this time.

Gauging player interest

By having the playtest you could use this chance to help find out the overall time the players invested into the game during their chance to play the game this will be a figure visible to the game company and this will be able to demonstrate to the developers how difficult the game is and how intriguing to players it is. After the playtest people might have made reviews for the game and these reviews could be extremely helpful to the company as it gives more

information that some figures that are available to them as by having people playtest their game they will be able to see what the majority of people think about their game. Some companies do not do playtest for their game and this can cause issues for the full release of the game an example of this would be a game called split gate that was released back in 2020 and this game was made by a small team of developers around 10 and they didn't have the budget to do a playtest and from not having a playtest the game servers were not able to handle the demand of people trying to play it at once due to this people started not liking the game as they were never able to play it causing the player just to give up on the game now the game's player count was only able to hold up to 67,000 players when it came out and the servers would be full after that now days the count is at 292 players.

Skill level

Every game in the gaming industry requires a skill level to make the game demanding for its players as if the game was at one skill level and all the players played the same it would get boring and repetitive. By having skill gaps in the game it helps entice the players to play more this could be done through making the game have skill based matchmaking or having different levels of difficulty for the game for example normal, advanced, hard, extreme. These levels of difficulty allow players to replay a game on different levels and achieve a different experience from it due to it being harder than the mode they originally played. An example of a game that has different levels of difficulty will be Tom Clancy's Ghost Recon Wildlands this game changes the way you play depending on the difficulty you choose to play the game on these are arcade (easiest), regular, advanced, extreme and ghost mode. The ghost mode in the game makes the whole style of the game different, only allowing the player to play the game in full stealth and only being allowed to die once. This encourages the player to replay the game on the hardest difficulty to encourage replaying and keeping the player count alive.

Benefits

Game prototypes offer different benefits throughout the development process. They serve as a tool for refining concepts, allowing developers to visualise and experiment with different ideas before developing them further. By identifying issues early on, it helps in identifying the potential risks and ensuring a better development. Player feedback from prototypes helps in making the final product, this allows for devs to enhance gameplay mechanics, graphics, and the user experience. This approach is more cost-effective, saving both time and resources while building anticipation for the full release.

P3: Create a design for an identified game concept

The concept

I'm going to be working on a space game concept that will work on both online browsers and mobile devices. In order to survive as long as possible, the player of this game must avoid aliens and other obstacles while piloting a spaceship across space. The goal of the game is to see who can survive in space the longest. Results from this challenge will be posted on a worldwide leaderboard to determine who has had the best time. The player is given a gun to shoot about two enemies after surviving 20 rounds but this weapon has a 4-second cooldown. The enemy's speed increases after each wave as well as every five rounds, the number of aliens doubles, making them more difficult to avoid.

Protagonist

The player plays as a spaceship that is going through its journey in space and this is designed as a pixel art spaceship that has a red colour scheme. The controls for the ship on Pc will be WASD. W for forward, A for backwards, S for going left, D for going right to shoot the player has a choice of using space bar to shoot or left mouse click. On mobile the game is played by swiping the ship in the direction wanted and holding down on the screen shoots where your finger is.



Controls

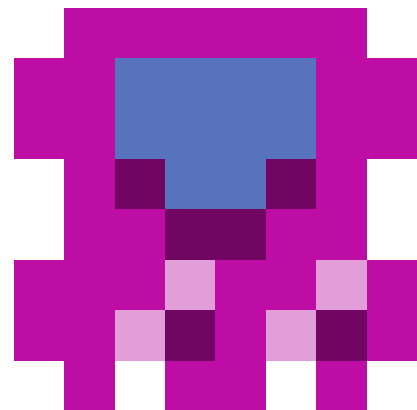
In this game, players control a red spaceship using easy controls designed for both PC and mobile. On PC, use the WASD keys to move and spacebar or left mouse click to shoot. On mobile, swipe to move and tap to shoot. These controls make gameplay intuitive and enjoyable on any device.

Scoring

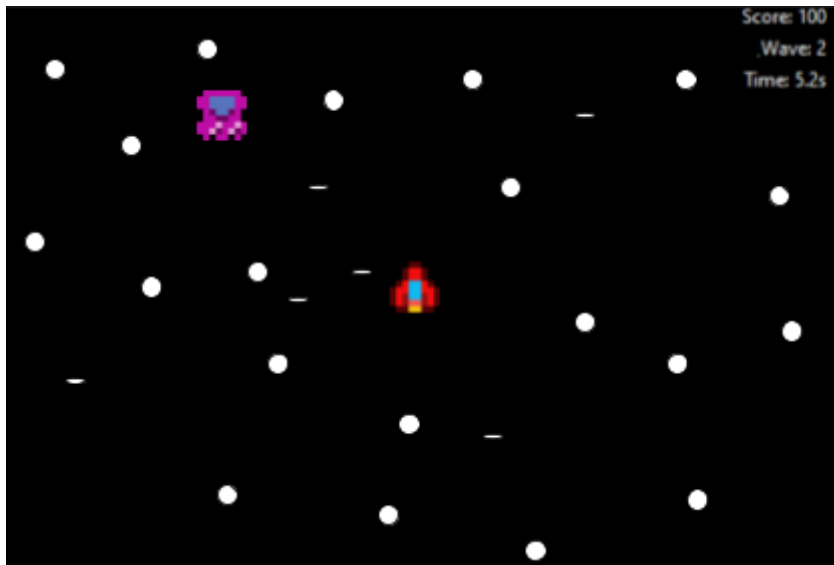
Score is generated for the player when they pass each wave of enemies and a bonus 50 points is given for each alien the ship has killed. There is no score limit in the game or wave limit just however long the player can go on for. This makes the game able to be replayed and each wave is different every time you play it as all waves are randomly generated so nothing is repetitive. If the player touches an alien or an alien touches the ship the game ends and the user is presented with the game over screen. After the game is over they can press try again to try and play again and get a better score.

Enemies

The ship that the player controls has to dodge all of the enemy aliens that are the same size as the ship. The enemies move only one direction making them unidirectional which is going down to the bottom of the player screen. The enemy is controlled by the computer and these are randomly dropped within the screen so no wave is the same. If the player has reached enough waves to get a gun they can shoot up to 2 enemies per wave when doing so it generates them 50 points to their score. They respawn every new wave with one and after 5 waves the number of enemies doubles and increases the speed they spawn at.



Game screen



Game over screen



System requirements

Platform: Works on both online browsers and mobile devices.

Minimum Screen Resolution: 480x270 pixels.

Input Devices:

- PC: Keyboard and Mouse.
- Mobile: Touchscreen.

Operating System: Compatible with Windows, macOS, Linux, iOS, and Android.

Browser Support: Compatible with major browsers such as Chrome, Firefox, Safari, and Edge.

Hardware:

- PC: Basic hardware with a modern web browser.
- Mobile: Compatible mobile device with a modern web browser.

Software:

Required libraries: tkinter, random, time.

Language: Python (version 3.7 or higher).

Internet Connection: Required for online play and leaderboard functionality.

Colour palette

Palette



Project plan

1. Idea:

- Generate the core concept of the game.
- Brainstorm key gameplay mechanics.
- Explore potential visual styles.

2. Design Phase:

- Develop game mechanics.
- Create wireframes and sketches for game screens.
- Design pixel art characters.

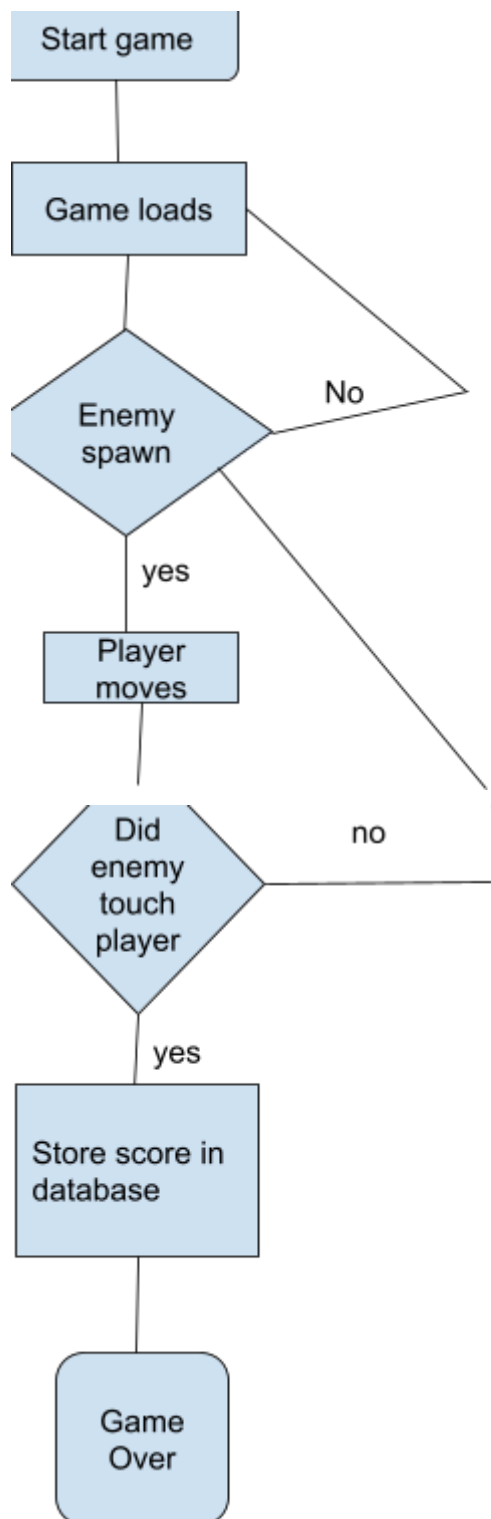
3. Development Phase:

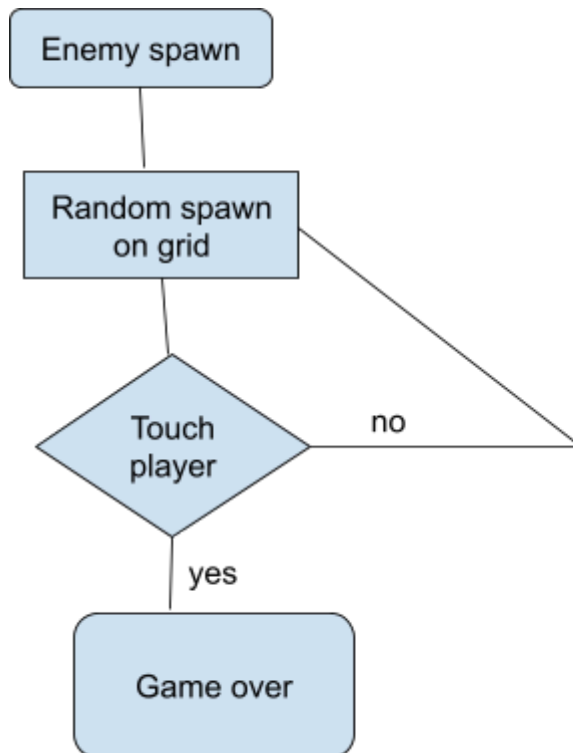
- Code the basics.
- Implement core gameplay mechanics.
- Design and integrate game screens.
- Test gameplay mechanics.

4. Testing Phase:

- Conduct playtesting on both PC and mobile devices.
- Address bugs or glitches.
- Optimise performance.

P4 Produce a logic structure for the identified game concept:





The flow charts show the game starts with player controls allowing movement by using WASD keys on PC or swipe gestures on mobile devices, with shooting controlled by spacebar/left mouse click or tapping, respectively. The player navigates a red spaceship, dodging obstacles and enemy aliens spawned randomly within the game area, initially at one enemy per wave, doubling every five waves. These enemies move downward, increasing speed over time. Players earn points by surviving waves and shooting enemies, with a bonus for each kill. The way that player progresses is through advancing waves, unlocking a weapon with cooldown after certain waves. The game ends if the player's spaceship collides with an enemy, showing a restart option. The players' scores are visible on a leaderboard for everyone to see.

P5: Build a prototype using core programming techniques and test for functionality

Here is the link to the video of the first working concept of the game

https://docs.google.com/presentation/d/1kFZL1vgHZ7GP8Im_eX2QD5xWL3z_dWIU_bzRpBkwZRuc/edit?usp=sharing

Test plan

Objective	Steps
Ensure player controls function correctly.	<ul style="list-style-type: none">• Press 'W', 'A', 'S', 'D' keys individually.• Verify that the player's ship moves accordingly in all directions.• Hold down each key to check continuous movement.• Release each key and ensure the ship stops moving.• Press the spacebar to shoot and verify bullets are fired.
Ensure enemies spawn and move correctly.	<ul style="list-style-type: none">• Observe enemy movement from top to bottom of the screen.• Check if enemies respawn after reaching the bottom.• Ensure enemies increase in size and speed with each wave.• Verify additional enemies are spawned after every 5th wave
Ensure collisions between the player's ship and enemies are detected.	<ul style="list-style-type: none">• Move the player's ship to collide with an enemy.• Verify the game ends upon collision.• Repeat the collision test.
Ensure the game over screen appears and the game can be restarted.	<ul style="list-style-type: none">• Intentionally collide with an enemy to trigger a game over.• Verify the game over screen is displayed with the option to restart.• Click on the "Try Again" button and ensure the game restarts from the initial state.
Check game performance	<ul style="list-style-type: none">• Play the game for an extended period.• Ensure the game remains responsive without lag or crashing.• Verify the elapsed time counter updates accurately.
Test edge cases and boundary conditions.	<ul style="list-style-type: none">• Move the player's ship to the edges of the screen.• Ensure the ship does not move beyond the screen boundaries.

- Check behaviour when enemies spawn near the screen edges.

```
from tkinter import *
import random
import time

# Constants
SCREEN_WIDTH = 480
SCREEN_HEIGHT = 270
SHIP_SPEED = 6
ENEMY_BASE_SPEED = 2
ENEMY_BASE_SIZE = 20
SCORE_PER_WAVE = 100
WAVE_INCREMENT = 50
ENEMY_SPAWN_INCREMENT = 1 # Increment for spawning one more enemy each
wave
ENEMY_SPEED_INCREMENT = 0.5 # Increment for increasing enemy speed

class Ship:
    def __init__(self):
        self.x = SCREEN_WIDTH // 2
        self.y = SCREEN_HEIGHT // 2
        self.size = 32
        self.image = PhotoImage(file="spaceship.gif")
        self.has_gun = False # Initially the ship doesn't have a gun
        self.shots_remaining = 2 # Number of shots remaining before
cooldown
        self.cooldown_timer = 0 # Cooldown timer for the gun
        self.last_shot_time = 0 # Time when the last shot was fired

    def shoot(self):
        if self.has_gun and self.shots_remaining > 0 and time.time() -
self.last_shot_time >= 5:
            self.shots_remaining -= 1
            self.last_shot_time = time.time()
            for enemy in enemies:
                if (self.x < enemy.x + enemy.size and self.x +
self.size > enemy.x
                    and self.y < enemy.y + enemy.size and self.y +
self.size > enemy.y):
                    enemies.remove(enemy)
```

```

        self.shots_remaining += 1 # Resetting shots if an
enemy is hit

    def update_cooldown(self):
        if time.time() - self.last_shot_time >= 5:
            self.shots_remaining = 2

class Enemy:
    def __init__(self, size, speed):
        self.x = random.randint(0, SCREEN_WIDTH)
        self.y = random.randint(-100, -50)
        self.size = size
        self.speed = speed
        self.image = PhotoImage(file="alien.gif")

def update():
    global ship, total_score, wave_number, enemies, start_time,
elapsed_time

    # Calculate elapsed time
    elapsed_time = time.time() - start_time

    # Move the player's ship based on key presses
    if keys['d']:
        ship.x += SHIP_SPEED
    if keys['a']:
        ship.x -= SHIP_SPEED
    if keys['s']:
        ship.y += SHIP_SPEED
    if keys['w']:
        ship.y -= SHIP_SPEED

    # Clamp ship position within the screen
    ship.x = max(0, min(SCREEN_WIDTH - ship.size, ship.x))
    ship.y = max(0, min(SCREEN_HEIGHT - ship.size, ship.y))

    # Update enemies
    for enemy in enemies:
        enemy.y += enemy.speed

    # Check for collisions with the player's ship

```

```

        if (ship.x < enemy.x + enemy.size and ship.x + ship.size >
enemy.x
            and ship.y < enemy.y + enemy.size
            and ship.y + ship.size > enemy.y):
            game_over()
            return

    # Respawn enemies if they reach the bottom of the screen
    if enemy.y > SCREEN_HEIGHT:
        enemy.x = random.randint(0, SCREEN_WIDTH)
        enemy.y = -enemy.size
        enemy.size += WAVE_INCREMENT // 10
        enemy.speed += ENEMY_SPEED_INCREMENT # Increase enemy
speed

        total_score += SCORE_PER_WAVE
        wave_number += 1

    # Spawn more enemies on every 5th wave
    if wave_number % 5 == 0 and len(enemies) < wave_number // 5:
        enemies.append(Enemy(ENEMY_BASE_SIZE, ENEMY_BASE_SPEED))

    # Update ship cooldown
    ship.update_cooldown()

    # Schedule the next update
    window.after(16, update)

def draw():
    canvas.delete("all")

    # Draw stars
    for star in stars:
        sx, sy = star
        canvas.create_oval(sx, sy, sx + 2, sy + 2, fill="white")

    # Draw player's ship
    canvas.create_image((ship.x, ship.y), image=ship.image)

    # Draw enemies
    for enemy in enemies:
        canvas.create_image((enemy.x, enemy.y), image=enemy.image)

    # Display total score

```

```

        canvas.create_text(SCREEN_WIDTH - 50,
                             20,
                             text="Score: " + str(total_score),
                             fill="white",
                             anchor="e")

        canvas.create_text(SCREEN_WIDTH - 50,
                             40,
                             text="Wave: " + str(wave_number),
                             fill="white",
                             anchor="e")

    # Display elapsed time
    canvas.create_text(SCREEN_WIDTH - 50,
                       60,
                       text="Time: {:.1f}s".format(elapsed_time),
                       fill="white",
                       anchor="e")

    # Schedule the next draw
    window.after(16, draw)

def handle_key(event):
    if event.keysym in keys:
        keys[event.keysym] = True
    if event.keysym == 'space':
        ship.shoot() # Shoot when spacebar is pressed

def handle_key_release(event):
    if event.keysym in keys:
        keys[event.keysym] = False

def game_over():
    global game_over_frame
    canvas.create_rectangle(0, 0, SCREEN_WIDTH, SCREEN_HEIGHT,
                             fill="red")
    game_over_frame = Frame(window)
    game_over_frame.place(relx=0.5, rely=0.5, anchor="center")
    game_over_label = Label(game_over_frame,
                             text="GAME OVER",
                             font=("Helvetica", 36),

```

```

        fg="white",
        bg="red")

    game_over_label.pack(pady=20)
    try_again_button = Button(game_over_frame,
                              text="Try Again",
                              command=restart_game)
    try_again_button.pack()

def restart_game():
    global game_over_frame, total_score, wave_number, ship, enemies,
    start_time
    game_over_frame.destroy()
    total_score = 0
    wave_number = 1
    ship = Ship()
    enemies = [Enemy(ENEMY_BASE_SIZE, ENEMY_BASE_SPEED)]
    start_time = time.time() # Reset start time
    update()
    draw()

# Main Program
window = Tk()
window.title("Space Game")
canvas = Canvas(window, width=SCREEN_WIDTH, height=SCREEN_HEIGHT,
bg="black")
canvas.pack()

keys = {"w": False, "a": False, "s": False, "d": False}

window.bind("<KeyPress>", handle_key)
window.bind("<KeyRelease>", handle_key_release)

ship = Ship()
enemies = [Enemy(ENEMY_BASE_SIZE, ENEMY_BASE_SPEED)]
start_time = time.time() # Start time for elapsed time calculation

total_score = 0
wave_number = 1
elapsed_time = 0

# Generate stars

```



```

stars = []
for _ in range(100):
    sx = random.randint(0, SCREEN_WIDTH)
    sy = random.randint(0, SCREEN_HEIGHT)
    stars.append([sx, sy])

update()
draw()

window.mainloop()

```

Test results

Objective	Steps
Player Controls	<ul style="list-style-type: none"> Pressed 'W', 'A', 'S', 'D' keys individually: The player's ship moved correctly in all directions. Held down each key: The ship moved continuously in the respective direction. Released each key: The ship stopped moving as expected.
Enemy Behaviour: <ul style="list-style-type: none"> 	<ul style="list-style-type: none"> Enemies spawned and moved correctly from top to bottom of the screen. Enemies respawn after reaching the bottom of the screen. Enemies increased in size and speed with each wave. Additional enemies were spawned after every 5th wave.
Collisions	<ul style="list-style-type: none"> Collision between the player's ship and enemies was detected accurately. Game ended immediately upon collision. Game over screen appeared with the option to restart.
Game Over Screen:	<ul style="list-style-type: none"> The game over screen displayed correctly with the final score and wave number. Clicking on the "Try Again" button restarted the game successfully.
Performance Testing:	<ul style="list-style-type: none"> Played the game for an extended period without experiencing lag or crashing.

	<ul style="list-style-type: none"> Elapsed time counter updated accurately throughout the gameplay.
Boundary Testing:	<ul style="list-style-type: none"> The player's ship and enemies behaved correctly at the edges of the screen, unable to move beyond the boundaries. Enemies spawned near the screen edges without any issues.

P6: Present the prototype to stakeholders to obtain feedback on the games concept

Feedback on the presentation

Hi there,

I am sending you a presentation outlining the game concept I propose for our company's development. Please provide any feedback you have on it.

https://docs.google.com/presentation/d/1kFZL1vgHZ7GP8Im_eX2QD5xWL3z_dWIUbzRpBkwZRuc/edit?usp=sharing

Hi Jamie,

I've reviewed the presentation you sent regarding the proposed game design, and I wanted to provide some feedback. We're impressed by the idea and prototype you've developed; it effectively outlines the game's key aspects and its intended purpose. At this stage, we're particularly eager to see the implementation of the shooting mechanics, as this is the final component needed and also crucial for the leaderboard feature.

Looking forward to seeing the progress

M1: Compare and contrast the features of games for different audiences

When designing games for different audiences and genres, it's essential to consider how these features are adapted to suit varying player preferences and different demographics.

First and third person

The choice between first-person and third-person perspectives impacts the player's experience and immersion within a game. First-person perspective offers an intimate and intense viewpoint, allowing players to directly perceive the game world through the eyes of their character. This creates a heightened sense of immersion, particularly suitable for genres that prioritise fast-paced action and immersive storytelling, such as first-person shooters or narrative-driven role-playing games examples of this would be games such as 'Call of duty and the Far Cry series'. On the other hand, third-person perspective provides a more detailed view of the game environment and the actions of the character. This broader perspective allows for more strategic gameplay and enhances the player's awareness of their surroundings, which is beneficial for genres that mainly depend on exploration,

character interaction, and tactical decision-making, such as action-adventure or open-world games and example of this would be 'ghost recon wildlands' which has you go round exploring the world in a third person perspective as a special military group. The choice between first and third person should meet with the intended gameplay experience and genre of the game. The difference between first and third-person games can be seen in "Call of Duty" and "Far Cry", where players have intense first-person action, whereas with "Ghost Recon Wildlands," they navigate the vast open world and engage in tactical missions from a third-person perspective. Call of duty focuses on the fast paced combat and arcade style shooter where as games such as farcry and Ghost recon wildlands focuses on the open world and exploring. These tailor to 2 different types of audiences.

NPC

Non-Player Characters (NPCs) are the supporting actors in a game's story, and they're super important for making the gaming experience richer. They don't just add to the background of the game world they also help drive the story forward and keep players engaged in the story as it helps make the environment. They are typically the ones who give you quests, offer advice, and make the game world feel more realistic with their interactions. Especially in games where the story is a big deal, like adventure or RPGs, NPCs play a huge role. They bring different personalities and motivations to the game, making the game world feel more real and interesting. Whether you're exploring a forest or fighting enemies, NPCs are there to help make the gameplay feel more immersive. In "The Witcher 3: Wild Hunt," NPCs present complex choices that affect the game. Meanwhile, in "Red Dead Redemption 2," NPCs focus more on showing the people who lived life in the wild West. players who like story based games and like the different environment that are in them will prefer games such as red dead redemption compared to the fast paced shooter call of duty.

Achievement

Achievement systems are like bonus challenges in games that keep players coming back for more even if they are done playing the game. They help expand the game's life expectancy before it dies because they give players extra goals to aim for, like completing a level in a certain time or finding hidden treasures or unlocking a skin or camo. This not only adds more fun and excitement to the game but also makes it last longer. Players who love completing every task or reaching every milestone in a game really like an achievement system. It gives them a sense of satisfaction and accomplishment. But, not everyone is into them. Some players prefer to just explore the game's environment. Some people prefer to just replay the game than going on further to get every achievement in the game as it can become very time consuming. In "The Legend of Zelda: Breath of the Wild," achievements are earned by exploring the open world and completing side quests like finding Shrines and Korok Seeds. Meanwhile, in "Halo: The Master Chief Collection," achievements are tied to specific in-game challenges, motivating players to play in both modes. These are things such as playing all the missions on the hardest difficulty. Players who like the difficulty and challenge of unlocking all achievements and challenges will enjoy games based around that such as Halo and zelda these both offer different sets of challenges that are completely different to each other.

DLC

Regular updates and DLC expansions are a very good method to keep players playing a game. They're super important for keeping players hooked, especially in games where you can play with others online or where the game keeps evolving over time. These updates bring in new stuff to explore, new challenges, and overall make the game experience even better. They're a get more players who might have stopped playing the game as they got bored to come back and keep playing. However, whether players are willing to pay for DLC can vary. Some might be fine with spending money for extra content, while others might prefer free updates. Offering free updates is a great way to keep a big crowd of players happy, especially if the game has fans of all ages and interests. By updating the game regularly it allows for the player count to stay up and players to keep playing as they want to see more changes made in the game as it may bring new cool features and items. In "The Elder Scrolls V: Skyrim," updates and DLCs like "Dawnguard" and "Dragonborn" added new quests and items to help keep players interested long after the game's release. Similarly "Fortnite" regularly updates with new skins and events, encouraging players to keep coming back for more fun and fresh content. People who like large open world games such as skyrim that offer paid dlcs and the game fortnite offers a shooter with free dlcs.

Competition

Competition and leaderboards make gaming more exciting for players who like challenges and enjoy beating their own scores or competing with others. These features are most popular in fast-paced games like arcade or racing games such as forza, where skill is crucial. They make a virtual stage where all players across the world can show their abilities and see how they rank against other players. However, not everyone enjoys competitive play. Some players prefer to explore game worlds at their own pace, focus on stories, or relax without pressure to beat someone or get a better score. Cooperative or solo adventures are more appealing. In "Mario Kart 8 Deluxe," players compete online or locally to achieve the best lap times and rankings, with global leaderboards promoting competition worldwide. However, in "The Legend of Zelda: Breath of the Wild," the focus is on exploration and adventure, with competition and leaderboards playing a minor role compared to individual discovery and immersion. The players who like competition will enjoy mario kart where it is you versus multiple players trying to get the best time.

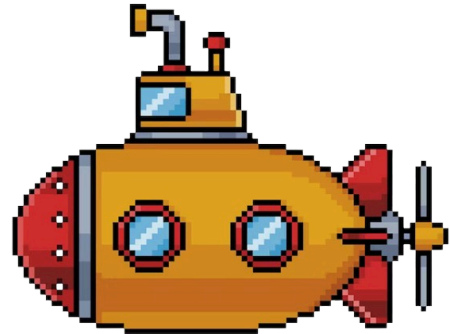
M2: Prepare alternative interface designs for the identified game concept

The Concept

The concept for this alternative game design is an underwater adventure where players control a submarine navigating through the ocean. The goal of the game is similar to the space game concept: to survive as long as possible while avoiding obstacles and enemies. However this game scrolls to the side instead, similar to games such as flappy bird.

Protagonist:

The player takes control of a submarine exploring the ocean. The submarine's pixel art design adds a nostalgic charm to the game while maintaining a modern aesthetic with a simple concept of gameplay that all ages of players should enjoy.



Controls:

On PC, players navigate the submarine using the WASD keys, similar to the space game concept. Pressing 'W' moves the submarine forward, 'A' moves it backwards, 'S' steers left, and 'D' steers right. To shoot, torpedoes players have the option to use either the spacebar or the left mouse click.

On mobile devices players can swipe in the direction they want the submarine to move. Holding down on the screen activates the submarine's weapons, allowing players to shoot at enemies.

Score and Gameplay:

As players navigate through the ocean, they encounter various enemies' obstacles. Score is generated for the player as they progress through the game, with each wave of enemies passed earning them points. Additionally, players receive a bonus of 50 points for each enemy creature destroyed by their submarine.

Similar to the space game concept, there is no score limit or wave limit in the underwater game. Players can continue exploring the ocean for as long as they can avoid obstacles and enemies. Each playthrough offers a unique experience, as enemy spawns are randomly generated, ensuring that dives are ever the same.

Game Over and Replay:

If the player's submarine collides with an enemy creature or hazard, the game ends, and they are presented with a game over screen. From here, players have the option to try again, challenging themselves to improve their score.

Enemies - Squid:

In the underwater game concept, players will encounter a variety of hostile creatures lurking in the depths, with one of the main ones being the squid.

Behaviour:

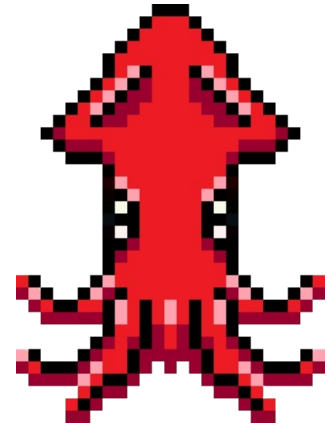
Squid enemies in the game are programmed to go towards the sub and try to ignore it; however if one of the tentacles touch the sub it will cause the game to end.

Challenges:

The squid enemies adds an element of danger to the underwater gameplay, requiring players to navigate carefully and strategically to avoid being overwhelmed.

Game Over:

If the player's submarine comes into contact with a squid enemy, the game will end, signalling the death of the player's submarine. The game over screen will be displayed, prompting players to try again.



Game Screen



Colour palette

Palette



System requirements

Platform: Works on both online browsers and mobile devices.

Minimum Screen Resolution: 480x270 pixels.

Input Devices:

- PC: Keyboard and Mouse.
- Mobile: Touchscreen.

Operating System: Compatible with Windows, macOS, Linux, iOS, and Android.

Browser Support: Compatible with major browsers such as Chrome, Firefox, Safari, and Edge.

Hardware:

- PC: Basic hardware with a modern web browser.
- Mobile: Compatible mobile device with a modern web browser.

Software:

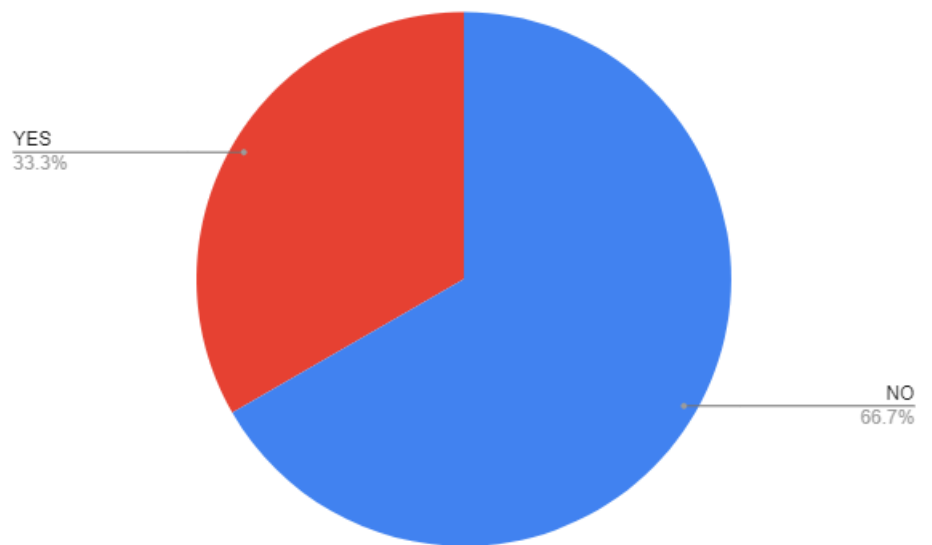
Required libraries: tkinter, random, time.

Language: Python (version 3.7 or higher).

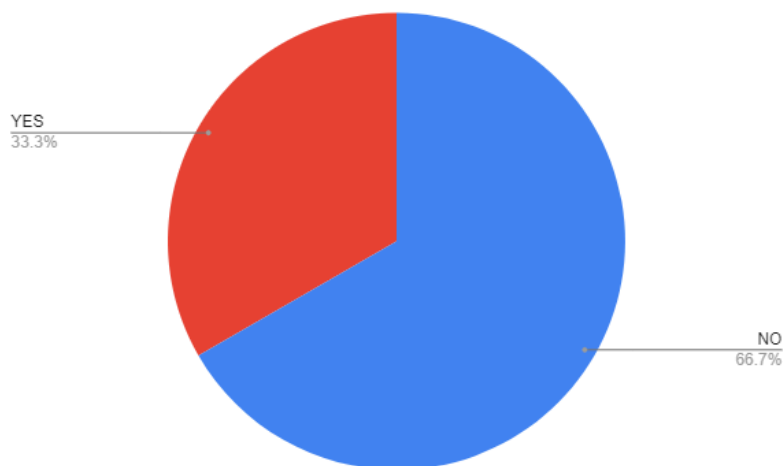
Internet Connection: Required for online play and leaderboard functionality.

M3 - Make changes to the game design and prototype based on stakeholder feedback

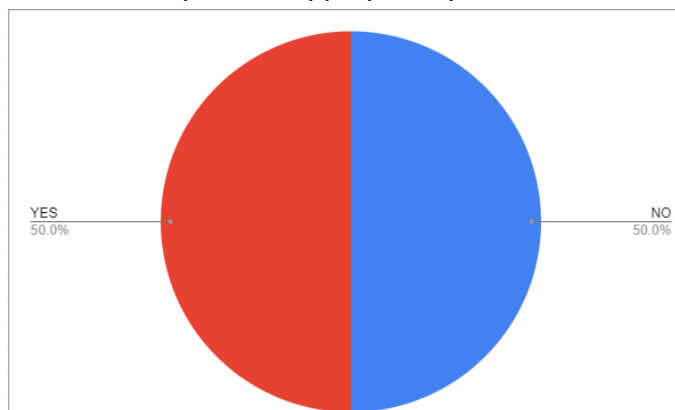
1. Is the timer for the game accurate with the speed it's meant to be ?



2. Is the enemies speed appropriate for when the game progresses or does it get too fast too early on?



3. Do enemies spawn in appropriate places



Feedback from stakeholders

- The timer appears to be faster compared to the actual timing
- The enemies get a bit too fast later on in the game
- If you are able to get far enough some of the enemies spawn on top of each other

The new code addresses the issues stated from the stakeholders with timing, enemy speed, and overlapping. It still uses the system's timing function, which might cause timing inconsistencies however this is as accurate as i can get it to be this might be a couple milliseconds off however this shouldnt be too noticeable. To manage enemy speed, setting maximum limits on their speed has helped as they are moving a bit more slower. The code stops enemies from spawning on top of one another this could be later improved by setting spawn locations however this might make the game become too predictable .

```
def update():
    global ship, total_score, wave_number, enemies, start_time,
    elapsed_time

    # Calculate elapsed time
    elapsed_time = time.time() - start_time

    # Move the player's ship based on key presses
```



```

if keys['d']:
    ship.x += SHIP_SPEED
if keys['a']:
    ship.x -= SHIP_SPEED
if keys['s']:
    ship.y += SHIP_SPEED
if keys['w']:
    ship.y -= SHIP_SPEED

# Clamp ship position within the screen
ship.x = max(0, min(SCREEN_WIDTH - ship.size, ship.x))
ship.y = max(0, min(SCREEN_HEIGHT - ship.size, ship.y))

# Update enemies
for enemy in enemies:
    enemy.y += enemy.speed

# Check for collisions with the player's ship
if (ship.x < enemy.x + enemy.size and ship.x + ship.size >
enemy.x
        and ship.y < enemy.y + enemy.size
        and ship.y + ship.size > enemy.y):
    game_over()
    return

# Respawn enemies if they reach the bottom of the screen
if enemy.y > SCREEN_HEIGHT:
    # Find a non-overlapping position for the new enemy
    while True:
        new_enemy_x = random.randint(0, SCREEN_WIDTH)
        new_enemy_y = random.randint(-100, -50)
        overlapping = False
        for existing_enemy in enemies:
            if (new_enemy_x < existing_enemy.x +
existing_enemy.size and
                    new_enemy_x + ENEMY_BASE_SIZE >
existing_enemy.x and
                    new_enemy_y < existing_enemy.y +
existing_enemy.size and
                    new_enemy_y + ENEMY_BASE_SIZE >
existing_enemy.y):
                overlapping = True
                break

```

```

        if not overlapping:
            break

        enemies.remove(enemy) # Remove the old enemy
        enemies.append(Enemy(ENEMY_BASE_SIZE, ENEMY_BASE_SPEED)) #
Add a new enemy
        total_score += SCORE_PER_WAVE
        wave_number += 1

# Spawn more enemies on every 5th wave
if wave_number % 5 == 0 and len(enemies) < wave_number // 5:
    enemies.append(Enemy(ENEMY_BASE_SIZE, ENEMY_BASE_SPEED))

# Update ship cooldown
ship.update_cooldown()

# Schedule the next update
window.after(16, update)

```

D1: Justify the design rationale for the identified game concept

Overall Game:

Audience Suitability:

The game's mechanics make it accessible and challenging to all ages that have access to a mobile device, making it appealing to a diverse audience. The controls and the pixel art design is suitable for all ages being simple and minimalist

Purpose Alignment: The game aims to provide replayability and some form of competitiveness. By incorporating random enemy spawns and making the difficulty levels increase as you go up in the number of waves, motivating players to continuously strive for improvement and higher scores.

Each Character (Player Spaceship and Enemies):

Audience Suitability: The nostalgic pixel art design of the player spaceship and enemies appeals to players of all generations due to simplicity and age friendly design. Its simplicity allows for easy gameplay when first starting to play due to the controls being very basic and not using the whole keyboard, this ensures that gameplay remains fun and competitive.

Purpose Alignment: Each character's design enhances gameplay by facilitating quick recognition and reaction. The vivid red colour scheme of the spaceship ensures its visible when playing in space. The enemy design is a simple purple alien that creates some sort of hard visibility when going through space that requires the player to have some form of focus while playing the game.

Each Screen (Game Screen and Game Over Screen):

Audience Suitability: the game screens feature a user-friendly layout and clear colours, catering to players of all ages by making it bright and vivid making it easy to

read and understand what is going on. By focusing on the simplicity and functionality, it ensures that players can easily navigate through the game's interface without any difficulty.

Purpose Alignment: The game screen provides players with only necessary information that stands out to the player. The seamless transition to the game over screen keeps players engaged and makes the player want to keep trying due to the screen flashing red.

Colour/Font Choices:

Audience Suitability: The choice of colours and fonts is designed to make sure readability and visual appeal across all demographics is available. Vibrant colours and simple fonts create a unique atmosphere, ensuring that players feel engaged throughout their attempt.

Purpose Alignment: Colours and fonts are strategically designed to provide crucial information and guidance for the player interactions. Every design element serves to make a competitive gameplay experience.

D2: Evaluate the game design and prototype against the identified game concept

Evaluation of Game Design Against Identified Game Concept:

Meeting Criteria:

The game is designed to create a replayable and challenging experience where players control a spaceship, avoid enemies and survive as long as possible in space. I've planned to include very simple controls but to counterbalance the level of difficulty with the ship manoeuvring through space I have made it an increasing difficulty by adding scoring mechanics and increasing the enemies speed the further you go to achieve this.

Planning:

The primary focus was to make the core gameplay mechanics simple but still making it difficult to get further this was done by including simple player controls, changing the enemy behaviour, adding a scoring system. The art style of the game is a pixel art style and colour scheme to match the space theme this idea came from space invaders game due to its arcade style graphics and simplicity of the game, this made my space game aim for a nostalgic and modern design.

Success:

The prototype successfully incorporates the planned game mechanics. Players can control the spaceship using WASD keys or touch/swipe controls, dodge enemy obstacles, and are able shoot enemies with a cooldown feature. The game's difficulty increases gradually, with faster enemy spawns and doubling enemy count every five waves. The scoring system rewards players for surviving each wave and destroying enemies, providing motivation to keep playing and improve their score. One part from the plan that did not make it to this version of the game is the global leaderboard that will be added at a later stage due to this requiring the game to be server sided where at the moment this is client scripted.

Matching Game Design with Finished Game:

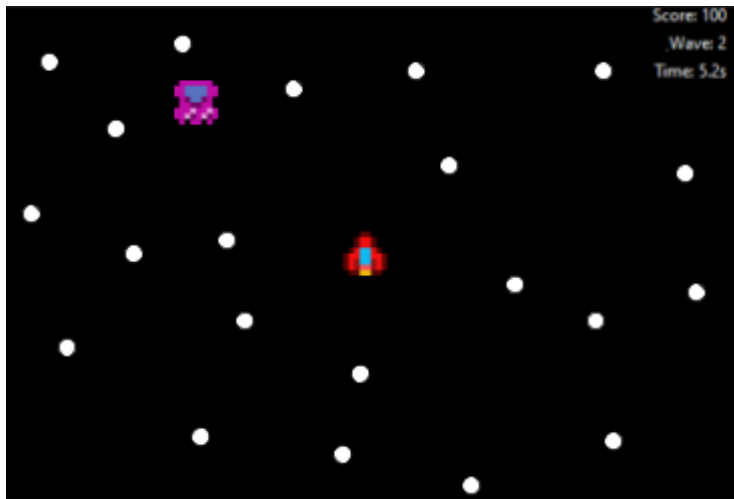
Matching:

The finished game closely matches with the initial design concept. The core gameplay mechanics, visual style, and overall experience remain consistent with the original plan thus making this iteration of the game successful and this allows for a good idea for the stakeholders on how successful the game can be.

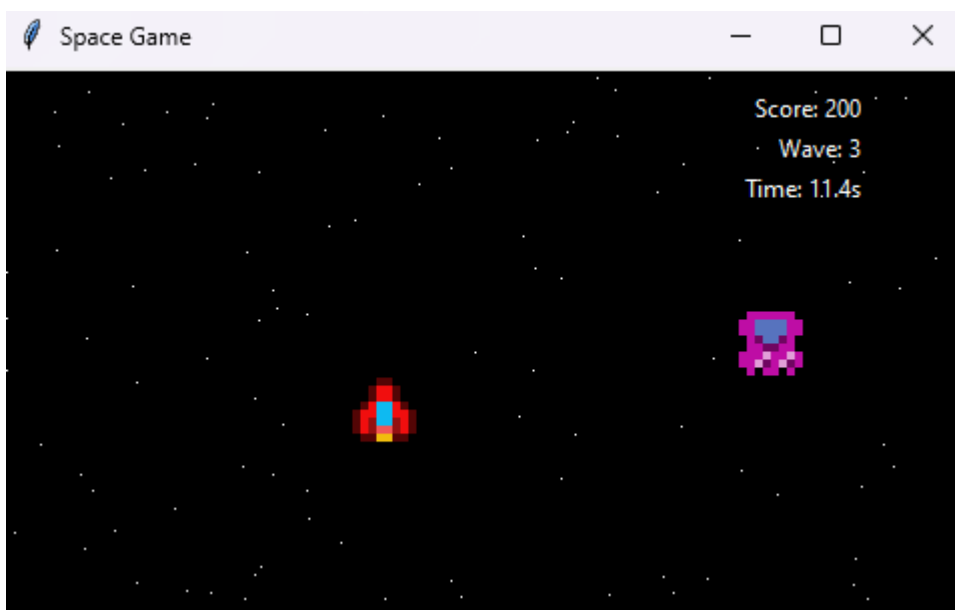
Evidence:

The prototype reflects the planned game mechanics, including player controls, enemy behaviour and scoring. The pixel art style and colour scheme are clear within the design, maintaining the nostalgic space theme throughout the player's time playing the game. Some minor adjustments and optimisations during development of this iteration of the game were needed, such as refining enemy spawn rates and adjusting scoring mechanics for balance, the main aspects of the game design remain clear apart from the leaderboard being there.

Design



Actual version



Evaluation Against Game Concepts:

Meeting Criteria:

The game concept was mainly aimed to provide an engaging and replayable experience for players of all ages by offering simple controls and a simple UI, increasing difficulty, and replayability and competition through scoring and progression.

Actions Taken:

To meet this plan, I designed and implemented simplistic controls, and to maintain the difficulty aspect, the increase of the difficulty of levels, and a scoring system that rewards players for their skills. Additionally, the random generation of enemy waves adds variety and replayability to each playthrough making it different from each playthrough and making it unpredictable where an enemy can spawn from.

Success:

The game prototype successfully meets the success criteria designed for the game. Players can easily understand the controls and gameplay mechanics, while the increasing difficulty and scoring system keep them challenged and motivated to

improve their score. The random generation of enemy waves ensures that each gameplay session feels unique, enhancing the replay value.

Works Cited

Wikipedia, <https://images.app.goo.gl/dHMvhChELxCKLepx6>. Accessed 6 March 2024.

Gordon, Whitson. "Forza Horizon 5 on the ROG Ally: performance guide & best settings."

ROG, 9 June 2023,

<https://rog.asus.com/articles/rog-ally/forza-horizon-5-on-the-rog-ally-performance-guide--best-settings/>. Accessed 6 March 2024.

Roeder, Mark, and Julian Wolanski. "Space Invaders." *Wikipedia*,

https://en.wikipedia.org/wiki/Space_Invaders. Accessed 6 March 2024.

"Splitgate." *Steam Charts*, <https://steamcharts.com/app/677620>. Accessed 21 February 2024.