



---

INTRODUCTION TO INTERACTIVE MEDIA

# STARSHIP GO

## DOCUMENTATION

---

FINAL MIDTERM

---



SYED FAHAD RIZWAN



# BACKSTORY



Princess Lyva's 8-year old daughter, the last descendant of royal blood, has been abducted by aliens from planet Zolo and she must maneuver past Zolo's fatal landscapes in her spaceship **to secure humanity's future.**

I was **artistically inclined** to illustrate the backstory through images instead and sound instead of text & therefore paired **9 photoshop images** with minimal text and a dramatic background music to form a **personal connection** and **increase the stakes**, giving the simple premise of a maze game a **greater purpose**.

Illustrations are attached **below**.

DOCUMENTATION



# CANVAS 1200 X 1200

---



---

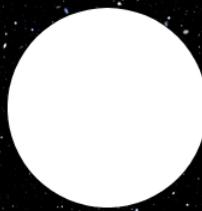
DOCUMENTATION



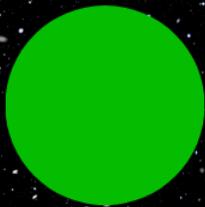
# INSTRUCTIONS

---

## PRINCESS LYVA'S SPACE POD



Under constant pull of planet **ZOLO's** gravity, it will diminish over time so get to the **end point** as soon as possible



Use this **healing pod** to regenerate

Click anywhere to move forward



# INSTRUCTIONS

---

## ZOLO'S TRAPS



These **rotating yellow NO GOs**  
will kill you immediately

The **BLOOD SPIKES** will  
instantly tear your pod apart

The **GREY ZONE** will diminish  
your pod extremely fast

Click anywhere to move forward



# INSTRUCTIONS

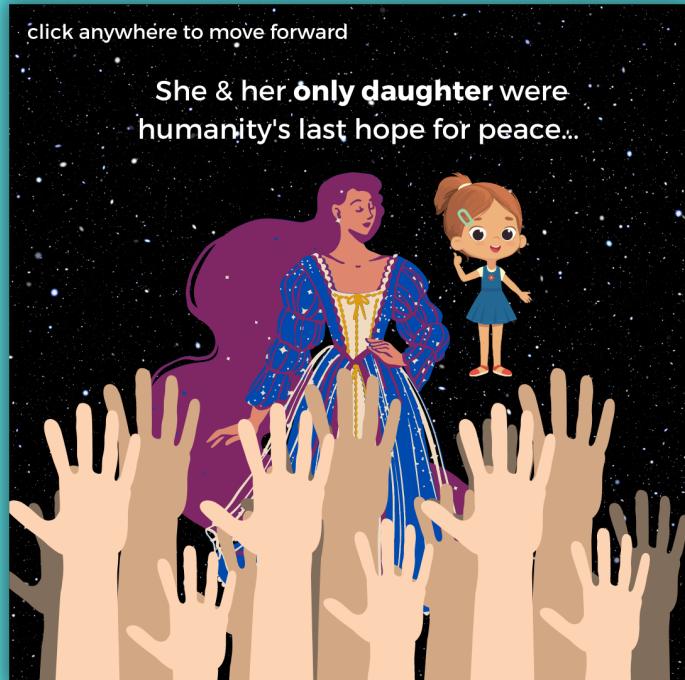
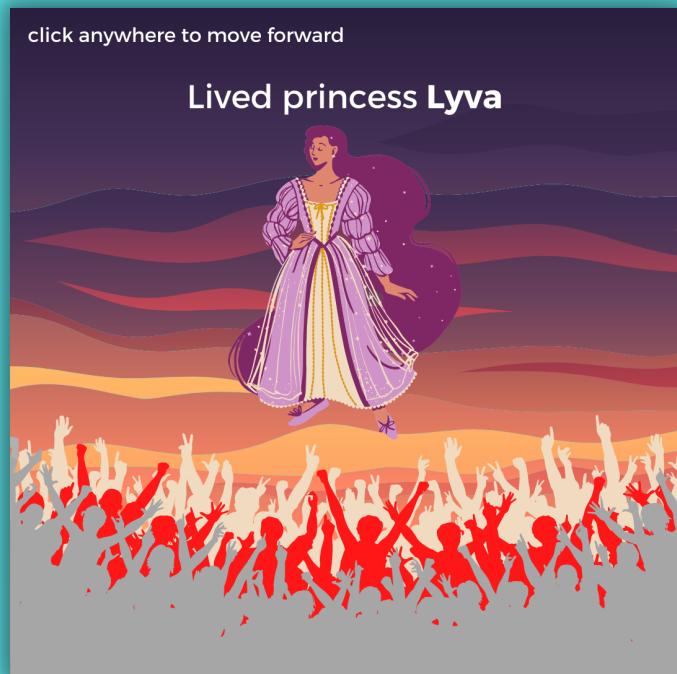
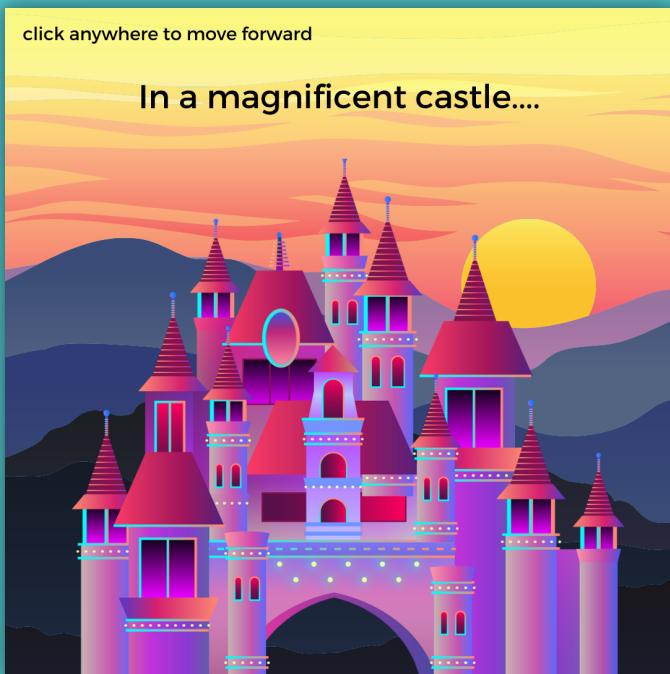
---





# BACKSTORY

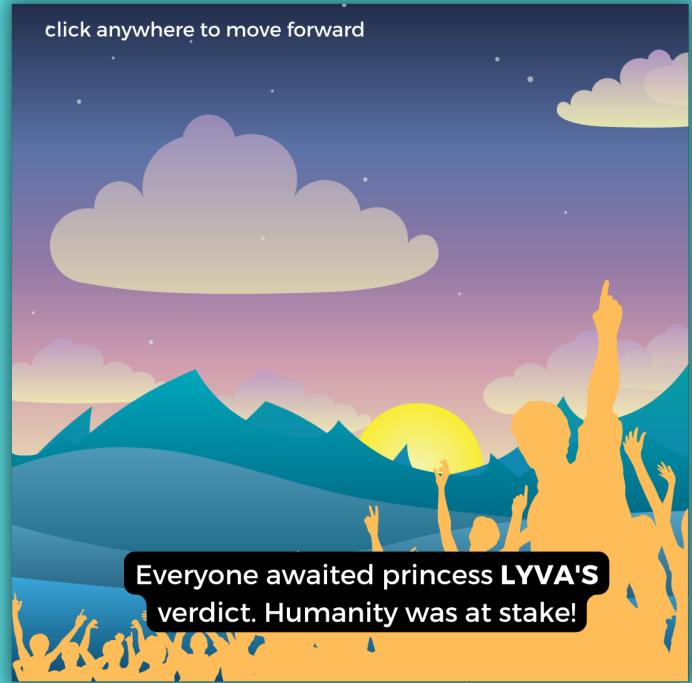
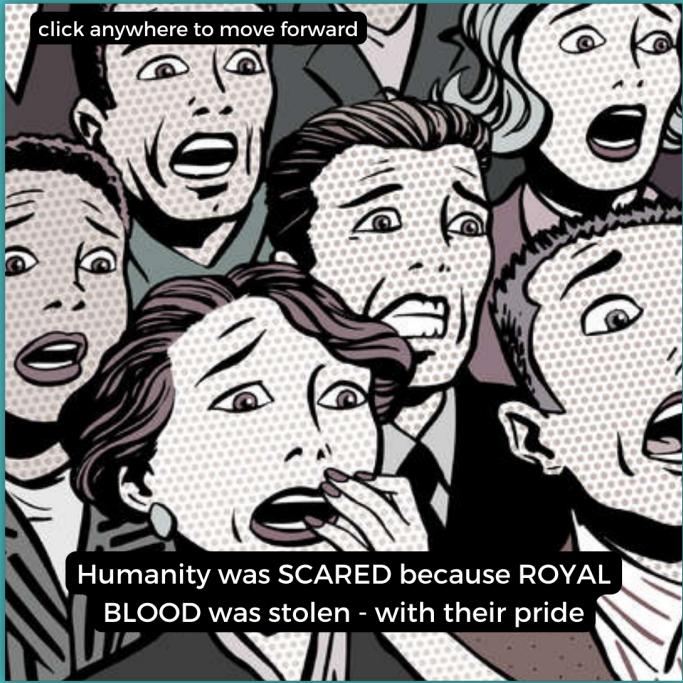
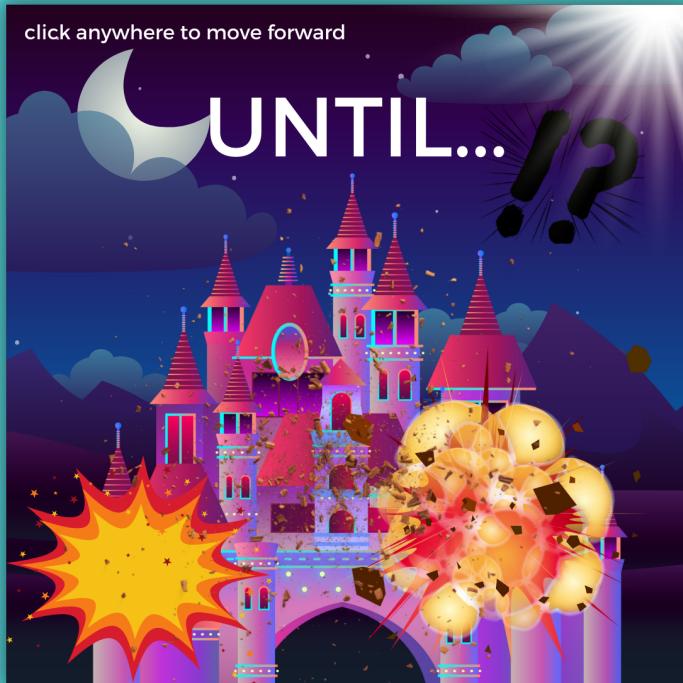
---





# BACKSTORY

---

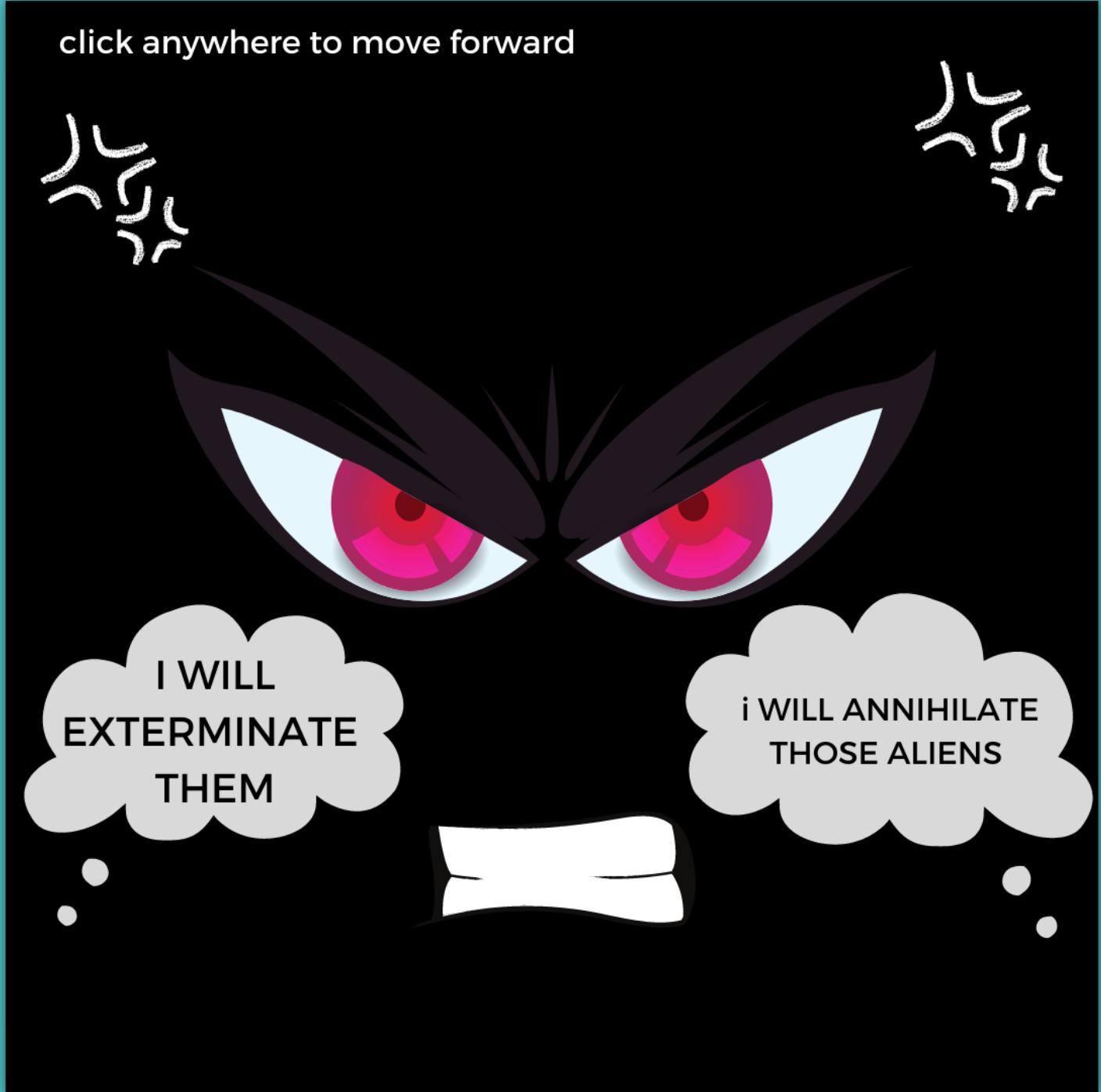




# BACKSTORY

---

click anywhere to move forward





# PROCESS

---

01

I designed this simple maze via photoshop, and **did not use** grids as they limited my artistic boundaries. While harder, it was undoubtedly worthwhile.

02

I created **monochromatic** spikes, rotating obstacles, and walls to make detection easy via **image.get**, comparing the pixel value of my ship with that of the background.

03

I used a **curved rectangle** as princess Lyva's spacepod, making it easier to define pixel value ranges compared to something non-linear.

04

The background image is in the **draw function** to avoid trails once the white sphere moves on screen to move past in the maze.

05

The mousepressed() function is extremely useful, incrementing my **mode++** upon each click and moving the game and story forward.



# PRELOAD

---

```
// loading all assets
function preload() {
    maze = loadImage("Assets/maze_final.png");
    font = loadFont("Assets/Starjout.ttf");
    font2 = loadFont("Assets/Roboto-Medium.ttf");
    score = loadSound("Assets/score.mp3");
    instruction1 = loadImage("Assets/Instruction_1.png");
    instruction2 = loadImage("Assets/Instruction_2.png");
    instruction3 = loadImage("Assets/Final_Instructions.png");
    story1 = loadImage("Assets/Story 1.0.png");
    story2 = loadImage("Assets/Story 2.0.png");
    story3 = loadImage("Assets/Story 3.0.png");
    story4 = loadImage("Assets/Story 4.0.png");
    story5 = loadImage("Assets/Story 5.0.png");
    story6 = loadImage("Assets/Story 6.0 (1).png");
    story7 = loadImage("Assets/Story 7.0.png");
    story8 = loadImage("Assets/Story 8.0.png");
    story9 = loadImage("Assets/Story 9.0 (1).png");
    humanWin = loadImage("Assets/humanWinX.png");
    humanLost = loadImage("Assets/Humanity_Destroyed.png");
    backstory = loadSound("Assets/backstory.mp3");
}
```

Storing **backstory images, soundtracks, and instructions** in the assets folder, I loaded them under the `preload()` function as I initialized my game so they are available wherever called



# CLASS ELEMENTS

---

```
// spacepod formation class
class spaceship {
    constructor(xPos, yPos, color, width) {
        this.x = xPos;
        this.y = yPos;
        this.color = color;
        this.width = width;
    }

    // draws ship and initiates gravity
    drawShip() {
        noStroke();
        fill(255);
        this.color = rect(this.x, this.y, this.width, this.width, 10);
        this.y = this.y + 1;
    }

    // controls the ship
    moveShip() {
        if (keyCode == UP_ARROW) {
            this.y = this.y - 2.5;
        } else if (keyCode == DOWN_ARROW) {
            this.y = this.y + 2;
        } else if (keyCode == LEFT_ARROW) {
            this.x = this.x - 2;
        } else if (keyCode == RIGHT_ARROW) {
            this.x = this.x + 2;
        }
    }
}
```

Class formation, downward pull initiation, spaceship creation, and movement



# CENTRAL SPINE

---

```
// detects healing pod using pixel value and increases health
increaseHealth() {
    let pixValShipCENTER = this.color.get(this.x, this.y);
    let pixValShipRT = this.color.get(
        this.x + this.width / 2,
        this.y - this.width / 2
    );
    let pixValShipRB = this.color.get(
        this.x + this.width / 2,
        this.y + this.width / 2
    );
    let pixValShipLT = this.color.get(
        this.x - this.width / 2,
        this.y - this.width / 2
    );
    let pixValShipLB = this.color.get(
        this.x - this.width / 2,
        this.y + this.width / 2
    );
    let healthPixValue = health_ellipse.get(360 * 2, 560 * 2);

    if (pixValShipRT[0] === healthPixValue[0]) {
        this.width = this.width + 1;
    } else if (pixValShipRB[0] === healthPixValue[0]) {
        this.width = this.width + 1;
    } else if (pixValShipLT[0] === healthPixValue[0]) {
        this.width = this.width + 1;
    } else if (pixValShipLB[0] === healthPixValue[0]) {
        this.width = this.width + 1;
    } else if (pixValShipCENTER[0] === healthPixValue[0]) {
        this.width = this.width + 1;
    }
}
```

This piece is the **core** of my game. I used the same concept for **wall detection, obstacle detection, and win and loss detection**. I essentially used pixel values at specific coordinates to make decisions. I didn't find any such examples online. Maybe we pioneered something unique here: innovation is brewing!



# HOW TO WIN?

---



Maneuver the spaceship past the spiked walls and rotating monsters till you get to the end line

## THINGS I LEARNT

I learnt that **keyIsDown()** function allows you to detect two keys at once, allowing greater control yet I chose not to use it as you constantly had to press the keys for the desired impact, seeming **counterintuitive** to me.

I learnt that **sounds** should never be put in the draw function as it repeats them, hence butchering them. I, therefore, used variables in the **mousepressed()** function to begin sounds **only once** before stopping them where ever I like.

I learnt that **pixelValue** via `image.get` is extremely useful. I also learnt that some concepts as a standalone topic might seem worthless but integrating them with other concepts can produce **beautiful results**.



# CHALLENGES

Resetting **but** collating variables and resetting them to original value was quite helpful. I could not **reset pressed keys** but I used **keypressed()** to trigger reset upon pressing enter.

Adding sound. **Neither** in the classes or the draw function did it properly work. However, it's clear now. Repetitively calling them distorts and slows them.

Creating backstory art and finding the right songs - **not difficult just lengthy work** but worth it. My game will force the player to make decisive choices. It **is not easy**. On the contrary, it is **quite hard**. Therefore, it will force the player beyond their comfort zone, helping them grow and develop. This intent drove me past this daunting yet rewarding journey.

Determining how exactly to use **pixelvalue** and comparing it to a predetermined digit was tricky to figure out. However, it carried the whole project once I did.

**Thank you so much** for all the help! Looking forward to a more creative and impactful rest of the semester.