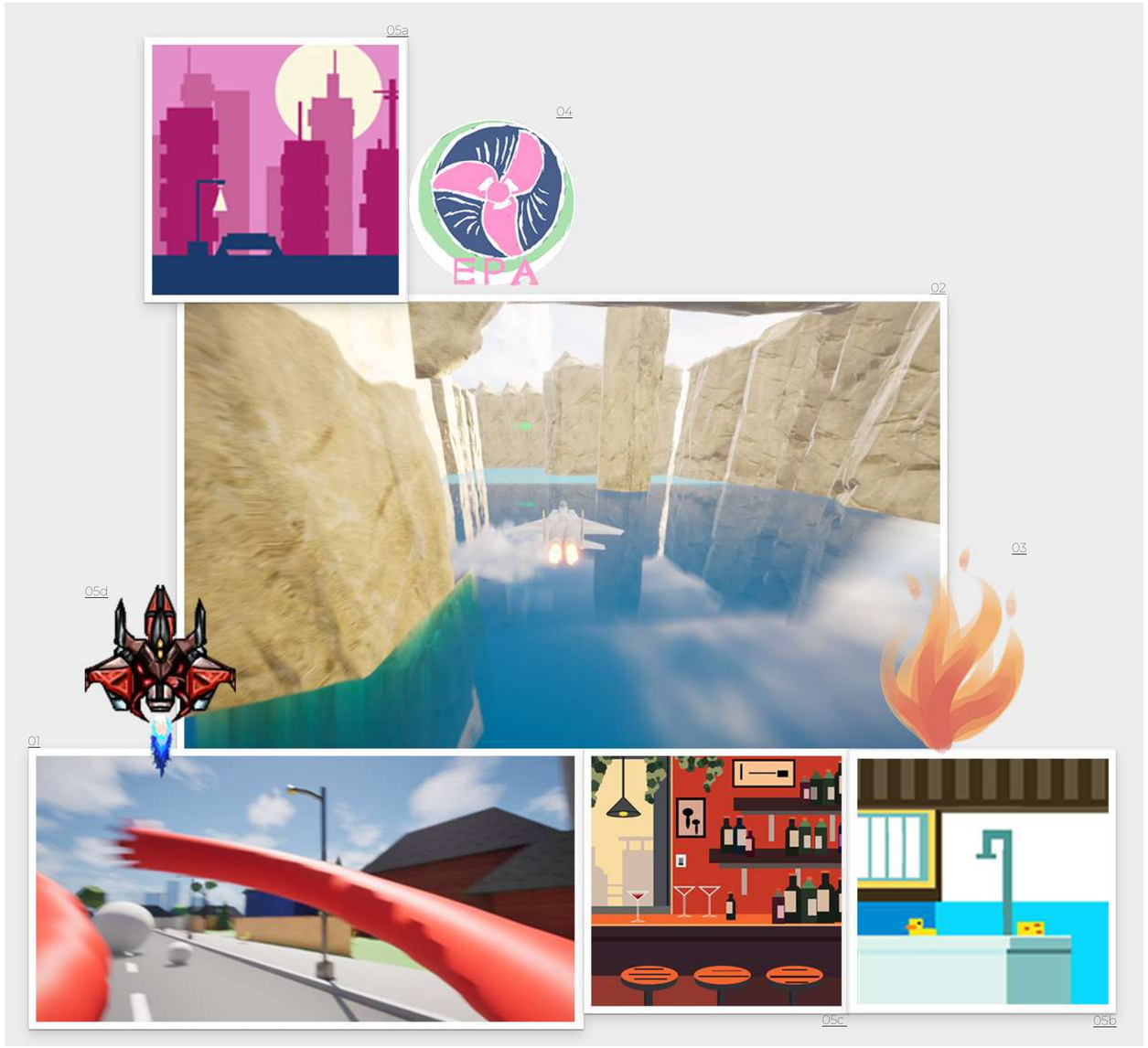
VOL. 01

FIRST YEAR PORTFOLIO

Jinxuan Lu

Honours Bachelor of Game Design

Sheridan College / 2025–2026



FLAIL OUT IS A SANDBOX GAME PROJECT WHERE YOU TAKE ON THE ROLE OF A WACKY WAVY INFLATABLE ARM FLAILING TUBE MAN!

THIS PROJECT WAS MADE AS A SCHOOL PROJECT BY 5 STUDENTS WITH THE CHALLENGE TO CREATE A TOY BASED ON AN UNDER-USED ACTION; IN THIS CASE: FLAIL.

GAME LOOP

EXPLORATION

PLAYER MOVES FREELY AROUND THE SANDBOX LEVEL.EXPERIMENTING WITH NEW OBJECTS AND MOVEMENTS.

CONTROL ARMS

PLAYER USES THE MOUSE TO SWING THE INFLATABLE ARMS.

CHAOS & COMEDY

UNEXPECTED REACTIONS CREATE FUNNY GAMEPLAY MOMENTS.

SMACK OBJECTS

THE ARMS COLLIDE WITH PROPS AND ENVIRONMENTAL OBJECTS.

PHYSICS REACTION

OBJECTS BOUNCE, FALL, FLY AWAY, OR BREAK.

CORE MECHANICS

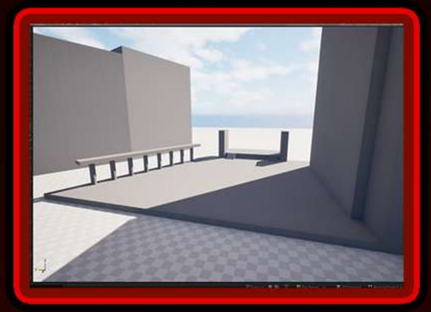
FLAILING ARM INTERACTION

THE CORE MECHANIC OF FLAIL OUT IS USING THE CHARACTER'S LONG

INFLATABLE ARMS TO SMACK, PUSH, KNOCK OVER, AND LAUNCH OBJECTS AROUND THE SANDBOX LEVEL.

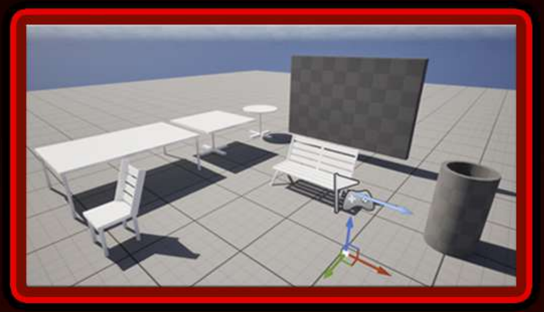
THE ARMS ARE INTENTIONALLY LOOSE AND UNPREDICTABLE, SO THE FUN COMES FROM EXPERIMENTING WITH MOVEMENT RATHER THAN CONTROLLING THEM PERFECTLY. EACH HIT CAN CREATE DIFFERENT PHYSICS REACTIONS, TURNING SIMPLE ACTIONS INTO CHAOTIC AND FUNNY GAMEPLAY MOMENTS.

LEVEL DESIGN DEVELOPMENT PROCESS



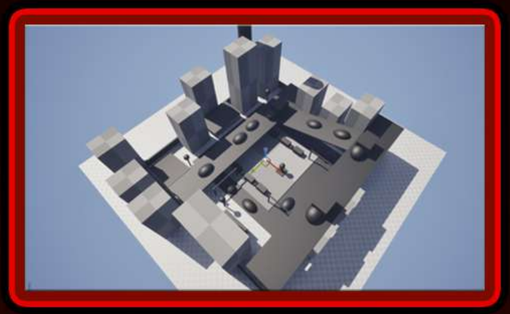
1.EARLY BLOCKOUT

TESTING SCALE AND MOVEMENT SPACE.



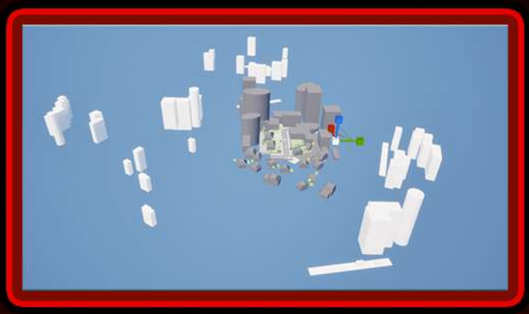
2.PROP TEST

EXPLORING OBJECT PLACEMENT AND INTERACTION POINTS.



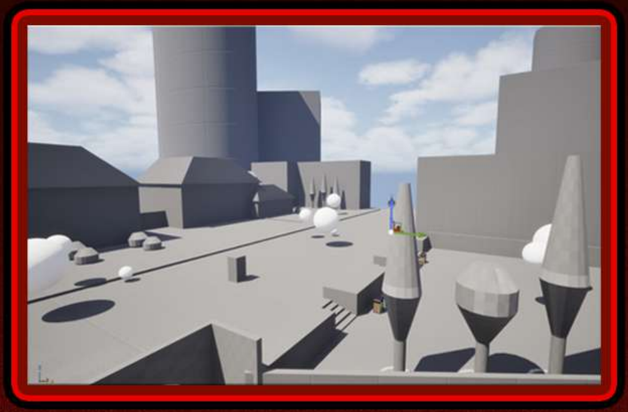
3.LAYOUT OVERVIEW

CHECKING LEVEL STRUCTURE FROM ABOVE.



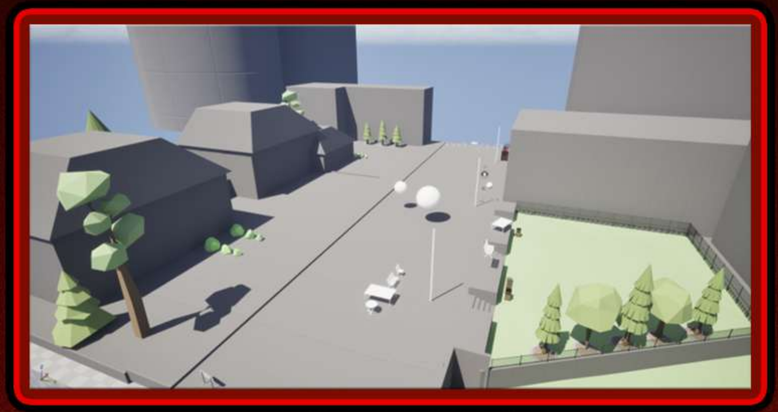
4.SCENE ASSEMBLY

BUILDING THE LARGER SANDBOX ENVIRONMENT.



5.SCALE TEST

REVIEWING STREETS AND PROPS FROM PLAYER VIEW.



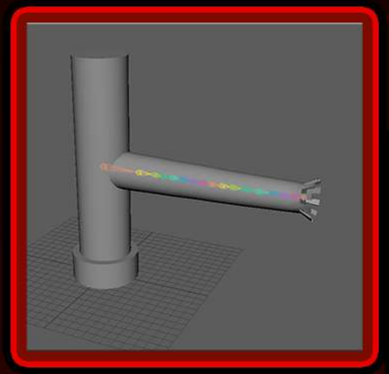
6.INTERACTION SPACE

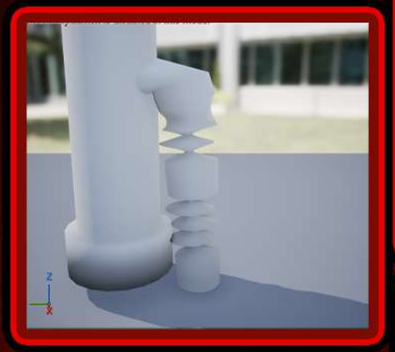
PLACING OBJECTS FOR PHYSICS-BASED PLAY.

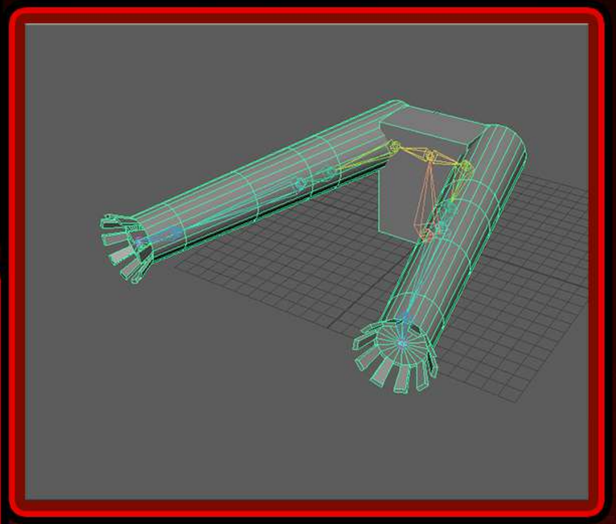
7.FINAL GAMEPLAY VIEW

SHOWING THE COMPLETED PLAYABLE LEVEL.

CORE MECHANIC IMPLEMENTATION



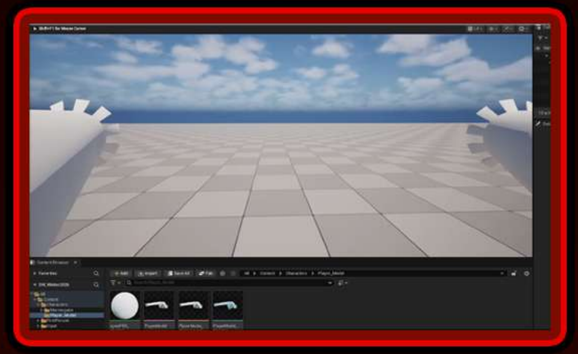




THIS PAGE PRESENTS THE DEVELOPMENT PROCESS BEHIND THE PLAYER'S CORE MECHANIC. WE EXPLORED ARM RIGGING, PLAYER CONTROL, AND IN-ENGINE TESTING TO TURN THE INFLATABLE ARMS INTO THE MAIN TOOL FOR INTERACTING WITH THE SANDBOX ENVIRONMENT.

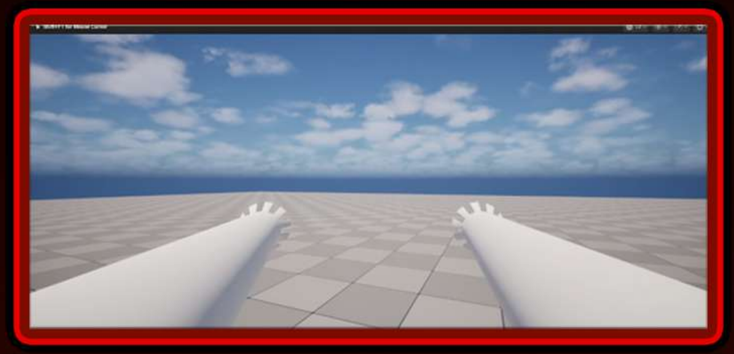
I CONTRIBUTED TO THE VISUAL AND LEVEL DESIGN SIDE WHILE WORKING WITH THE TEAM TO TEST HOW THE INFLATABLE ARMS INTERACTED WITH THE ENVIRONMENT AND SUPPORTED THE SANDBOX GAMEPLAY.

ARM MECHANIC ITERATION



1. STATIC ARM PROTOTYPE

THE FIRST VERSION USED SIMPLE WHITE ARMS TO TEST THE BASIC FIRST-PERSON VIEW, BUT THE ARMS WERE NOT MOVABLE YET.

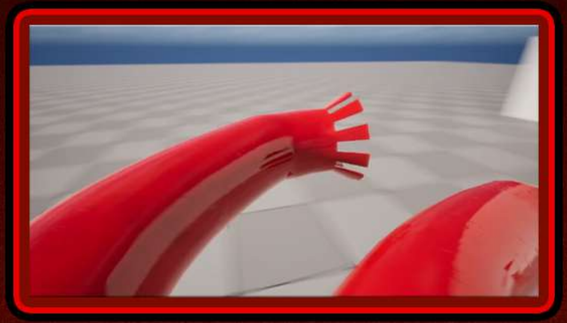


2. EXTENDED ARM TEST

THE ARMS WERE EXTENDED TO INCREASE THE SENSE OF FLAILING MOVEMENT AND MAKE THE INTERACTION FEEL MORE EXAGGERATED.

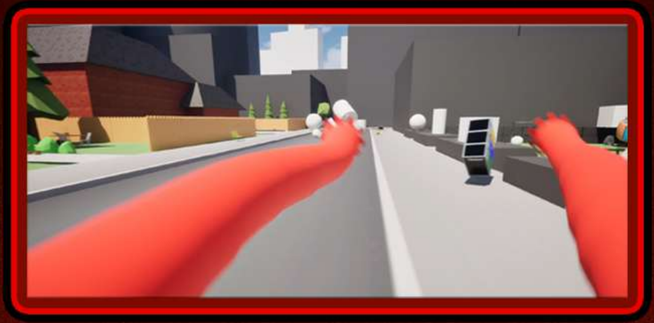
3. PHYSICS MOVEMENT TEST

PHYSICS MOVEMENT WAS ADDED TO THE ARMS, BUT THE SWINGING EFFECT BECAME TOO INTENSE AND DIFFICULT TO CONTROL.



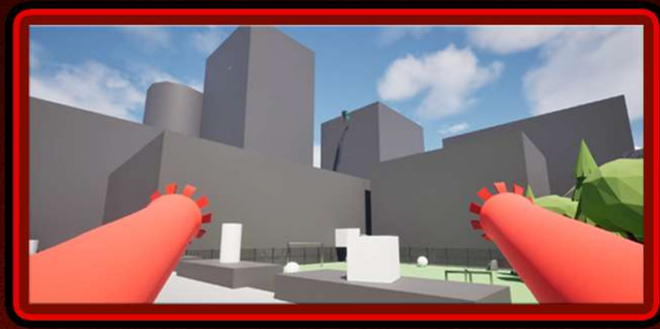
4. IMPROVED CONTROL SYSTEM

THE PHYSICS MOVEMENT WAS ADJUSTED TO REDUCE EXCESSIVE SWINGING AND MAKE THE ARMS FEEL MORE CONTROLLABLE.



5. VIEW-BASED ARM MOVEMENT

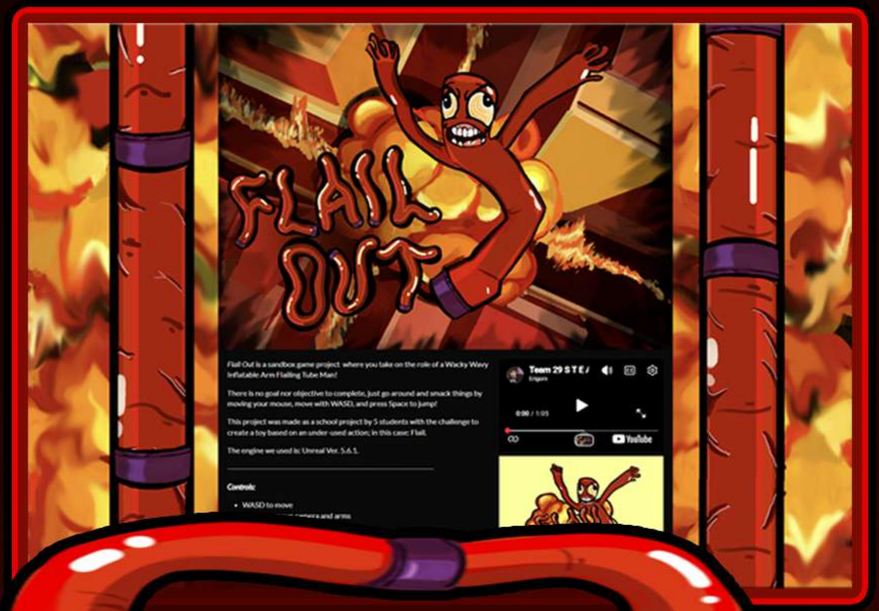
THE ARMS WERE UPDATED TO FOLLOW THE PLAYER'S CAMERA DIRECTION, MAKING THE INTERACTION FEEL MORE RESPONSIVE AND CONNECTED TO PLAYER INPUT.

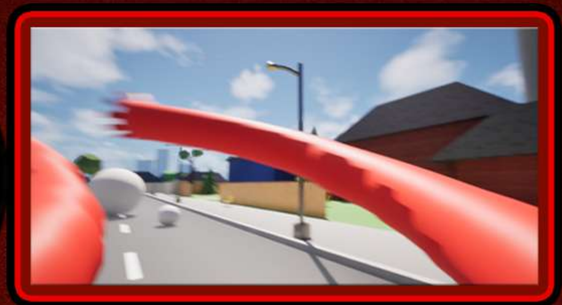


6. FINAL ARM DESIGN

THE FINAL VERSION INCREASED THE ARM LENGTH AND APPLIED THE RED INFLATABLE MATERIAL TO MATCH THE GAME'S VISUAL STYLE.

FINAL GAMEPLAY SHOWCASE



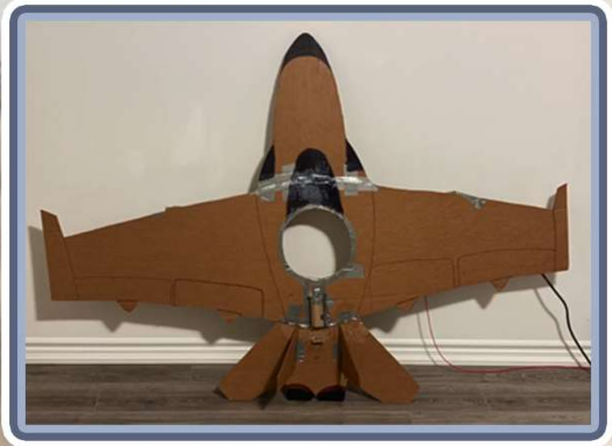


https://sinsord.itch.io/flail-out

THE FINAL BUILD OF FLAIL OUT FOCUSES ON PLAYFUL SANDBOX INTERACTION. PLAYERS CAN MOVE FREELY AROUND THE MAP, SWING THEIR INFLATABLE ARMS, AND INTERACT WITH ENVIRONMENTAL OBJECTS IN UNPREDICTABLE WAYS. BY BOLDLY FLAILING THE ARMS, PLAYERS CAN PUSH, HIT, AND LAUNCH OBJECTS, CREATING CHAOTIC AND FUNNY PHYSICS-BASED RESULTS.

THE PROJECT WAS ALSO UPLOADED TO ITCH.IO AS A PLAYABLE RELEASE PAGE, INCLUDING THE TRAILER, SCREENSHOTS, CONTROLS, CREDITS, AND FINAL GAME BUILD.

Janky Cardboard Top Gun Simulator



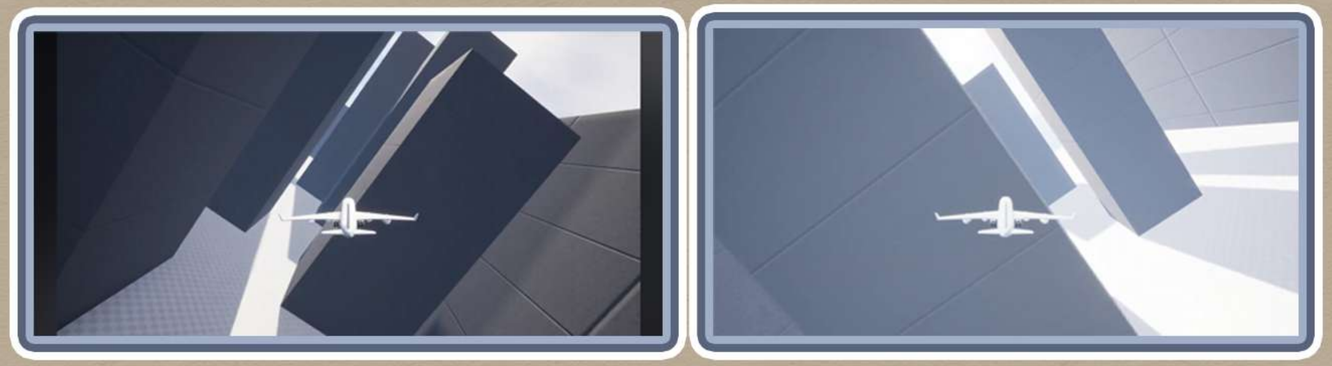
Attach

Reset

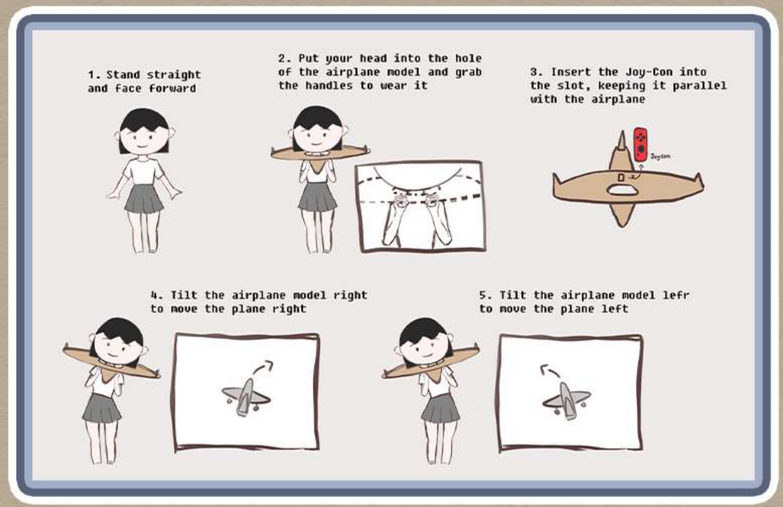
Start game

Janky Cardboard Top Gun Simulator is a five-day unconventional controller project. Players control a fighter jet by moving their body with a handmade cardboard airplane controller. Instead of relying on traditional inputs, the game turns physical movement into the main control method, creating a playful and humorous flight experience.

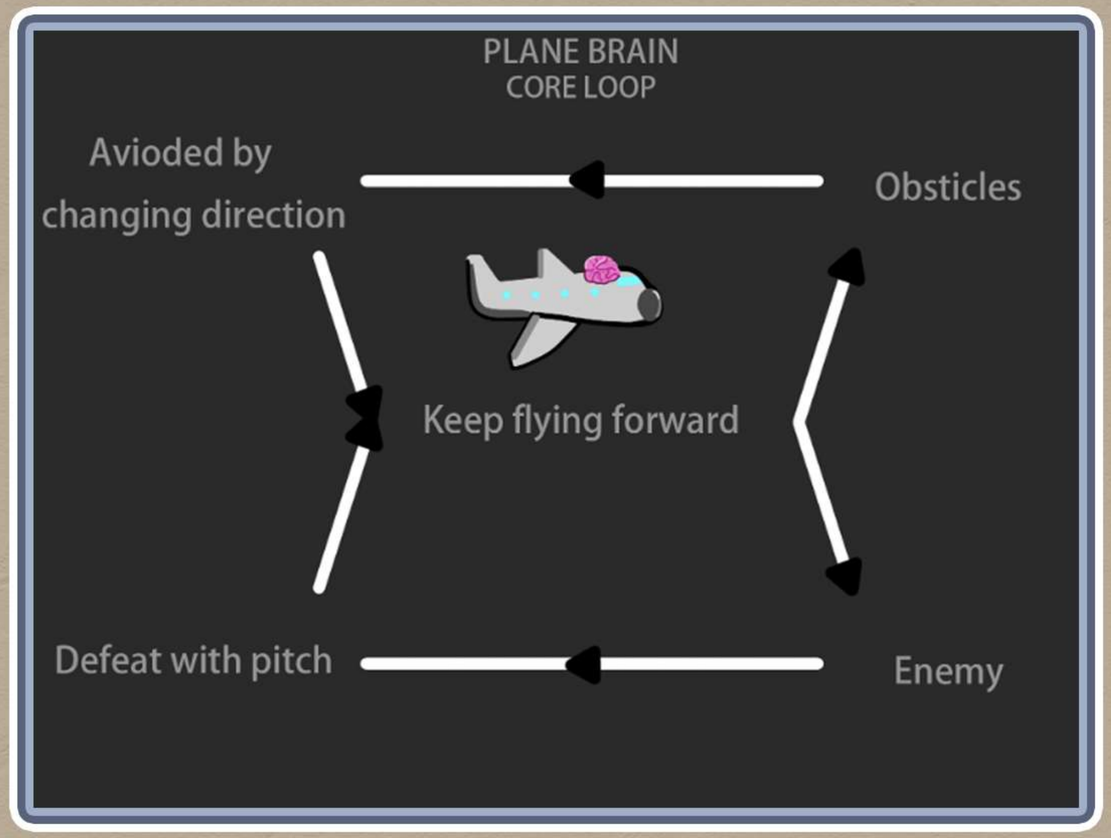
Core Mechanic



The core mechanic is body-based flight control. Players lean and shift their body while holding the cardboard airplane controller, and their movement changes the aircraft's direction in-game. This transforms a simple flight simulator into a physical performance, making the control experience intentionally janky and entertaining.



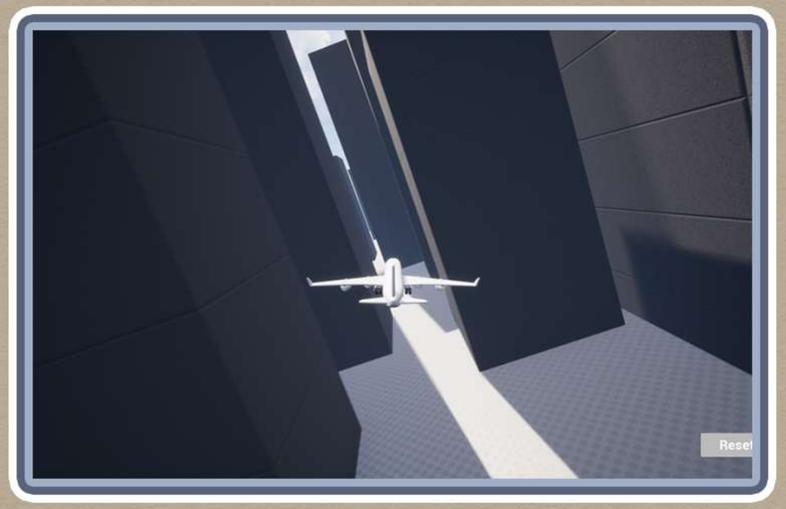
Game Loop



Gameplay Implementation

1. Joy-Con Motion Input Test

Tested the Joy-Con gyroscope as the main input method for the cardboard controller.

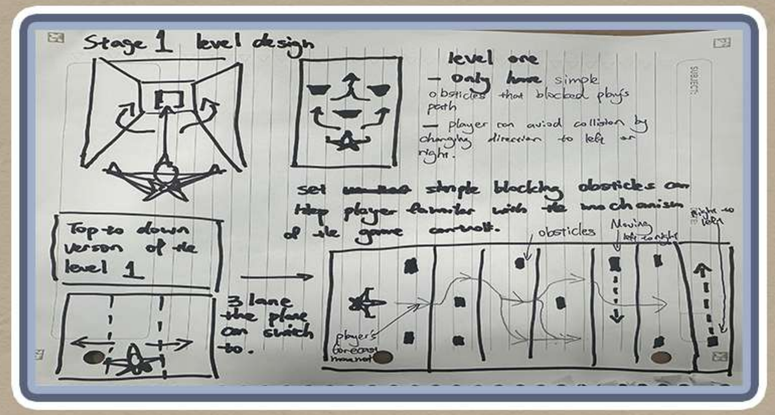


2. Early Flight Prototype

Converted X and Y motion data into basic aircraft movement inside the engine.

3. Final Flight Environment

Applied the motion control system to the final flying level with larger scenery and obstacles.



4. Level Design Sketch

Planned the flight route, obstacle placement, and challenge flow before building the playable level.

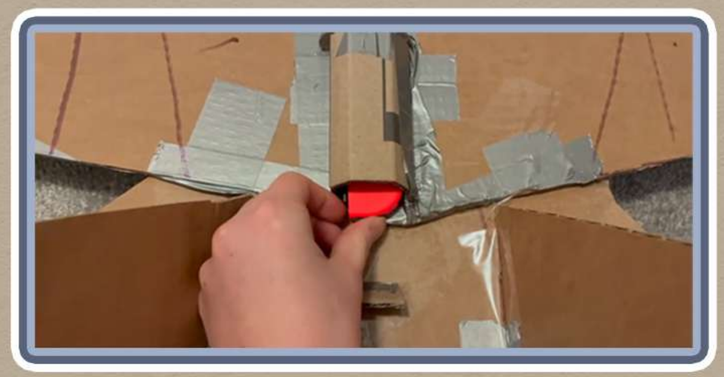
Motion-Based Flight Control

The player controls the aircraft by physically moving the cardboard controller. The Joy-Con's gyroscope detects changes in X and Y movement, which are mapped to the plane's direction in the game.This design turns a simple flight simulator into a physical interaction experience. Instead of pressing buttons to steer , the player must move their body like they are piloting the aircraft, making the gameplay more playful and performative.

Controller Construction Process

1. Joy-Con Placement Test

Tested where the Joy-Con should be placed inside the cardboard controller.



2. Internal Mounting

Built a simple cardboard slot to hold the Joy-Con securely during movement.

3. Controller Interior

Checked the internal structure to make sure the Joy-Con stayed aligned with the aircraft body.

4. Cardboard Shell Assembly

Constructed the main aircraft body using cardboard, tape, and lightweight materials.

5. Wing Structure

Reinforced the wings and body shape to make the controller easier to hold and move.

6. Final Physical Controller

Completed the cardboard airplane controller for motion-based flight gameplay.

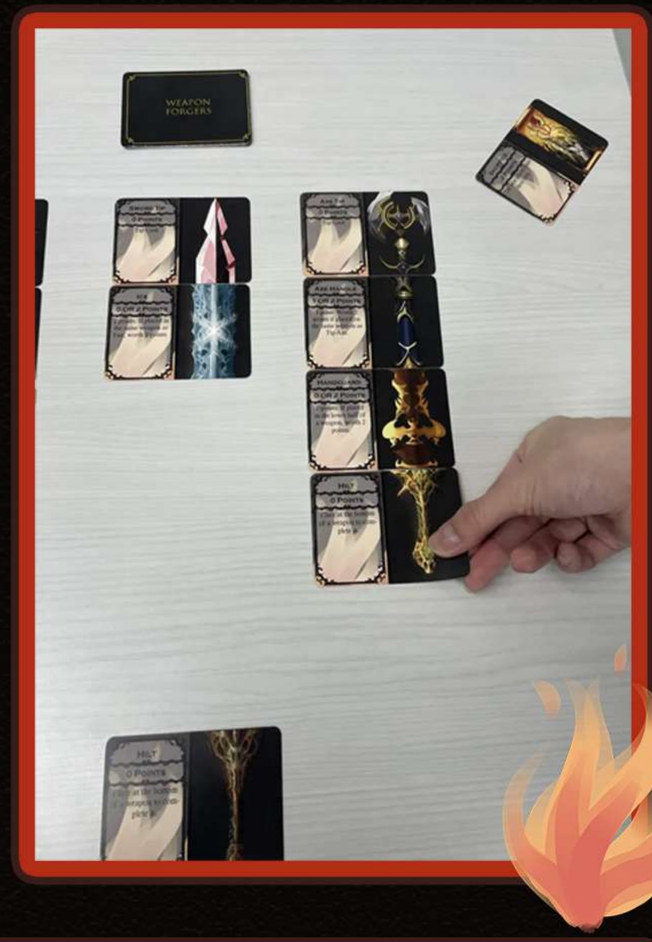
Final Game Showcase

The final version of Janky Cardboard Top Gun Simulator combines a digital flight game with a handmade cardboard aircraft controller. Players physically move their body while holding the controller, and the Joy-Con gyroscope translates their motion into the plane's flight direction in-game.

The gameplay challenges players to fly through a canyon environment, avoid obstacles, and survive as long as possible. By turning body movement into the main control method, the project creates a playful and performative flying experience that feels intentionally awkward, physical, and funny.

https://youtu.be/RE_wP4WB79w?si=96zhUi3LHoMXiLWq

WEAPON FORGERS



Weapon Forgers is a 2–3 player competitive card game about building, modifying, and claiming powerful weapons. Players draw material cards, forge them onto different weapons, and use card effects to influence the final score. The key strategy comes from deciding when to hold cards, when to forge them, and when to claim a weapon before another player takes it. Game develop base on the mechanic of kill steal.

GAME RULES

Weapon Forgers is a 2-3 players game, where you collaborate with your friends to build the most incredible weapons, but only one of you will get the credit for its creation.

OBJECTIVE

Your objective is to be the player with the greatest number of points at the end of the game. Build weapons with different parts that award different points and claim the weapons using a Hilt.

COMPONENTS

- 3 Tip Cards
- 3 Hilt Cards
- 12 Material Cards

SETUP

1. Locate the 3 Tip cards of the weapons: Sword, Axe, and Flail. Lay them out next to each other in front of the players.

2. Locate the 3 Hilt cards and give 1 to each player. Discard any remaining Hilts. Hilt cards DON'T count towards your held cards and CANNOT be stolen.

3. Shuffle the rest of the deck and put it in a reachable location for all the players. This is the Materials deck.

TURN STRUCTURE

The players decide who has the first turn, and the turn order moves clockwise.

FORGING PHASE

1. Draw a card from the Materials deck

2. Read the effects of the card. They can become active during Forging or Scoring phase.

3. Choose your action for the turn:

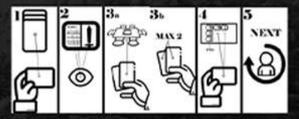
a. Forge: Place a card on one of the available weapons and activate their effects if they apply.

b. Hold: Choose to hold up a card instead of forging it. You can only hold up to 2 cards at a time. If you are only holding one card or no cards when you do this, your action ends. If you are already holding 2 cards, you are forced to forge the one of the previous cards to hold the new one.

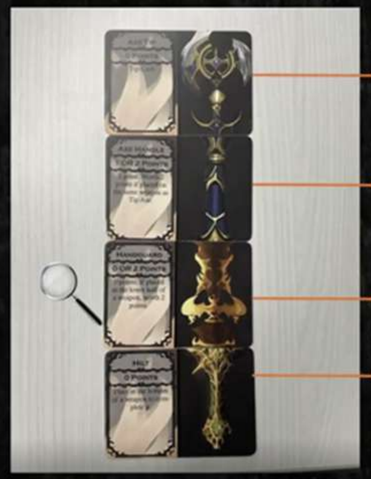
4. Decide if you will or will not place your Hilt card. If you place a Hilt, you finish the forging of that weapon. Leave the weapon on the table. It now counts towards your score, and no new cards can be forged into it. You continue to Forge and Hold cards in subsequent turns after placing your Hilt.

5. Next player's turn.

Forging Phase ends when all the Hilts have been placed.



Scoring



Axe Tip (0 Points)

Axe Handle (2 Points)

Handguard (2 Points)

Hilt (0 Points)

Total: 4 Points

Turn Structure

1.

Draw Material Card

2.

Option #1: Hold

Option #2: Forge

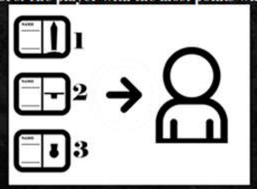
SCORING PHASE

All players participate.

1. Add the point value of each card in the weapon, starting from the card nearest to the Tip.

2. Read the special effects and adjust the point value as it applies

3. Add all the points to determine your final score. The player with the most points wins.



CREDITS

The following people made this game possible.

Game Designers: Aritmeos Sanchez Miranda, Charlie Theodoulou, Daniel Yeung, Jinxuan Lu

Artwork: Charlie Theodoulou, Daniel Yeung, Jinxuan Lu

Card Layout: Charlie Theodoulou

Rulebook: Aritmeos Sanchez Miranda

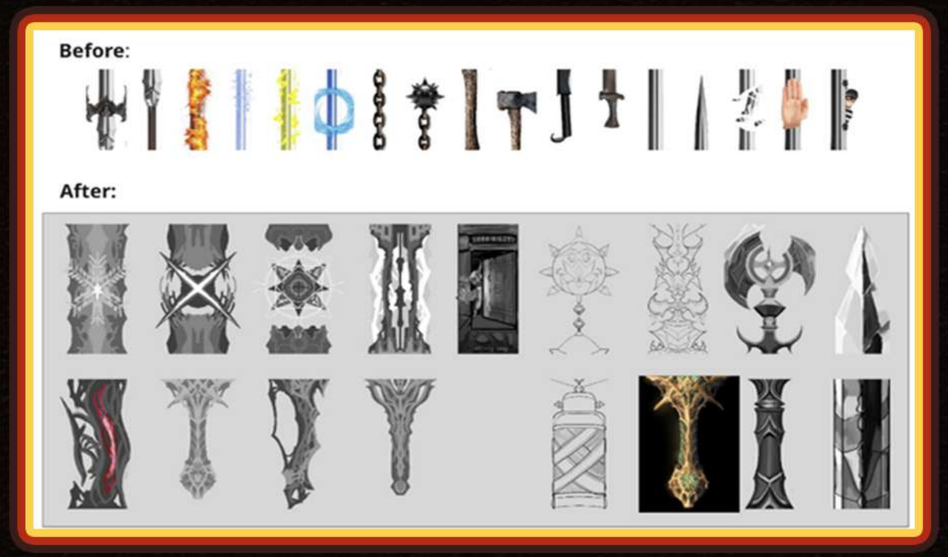
LEGAL

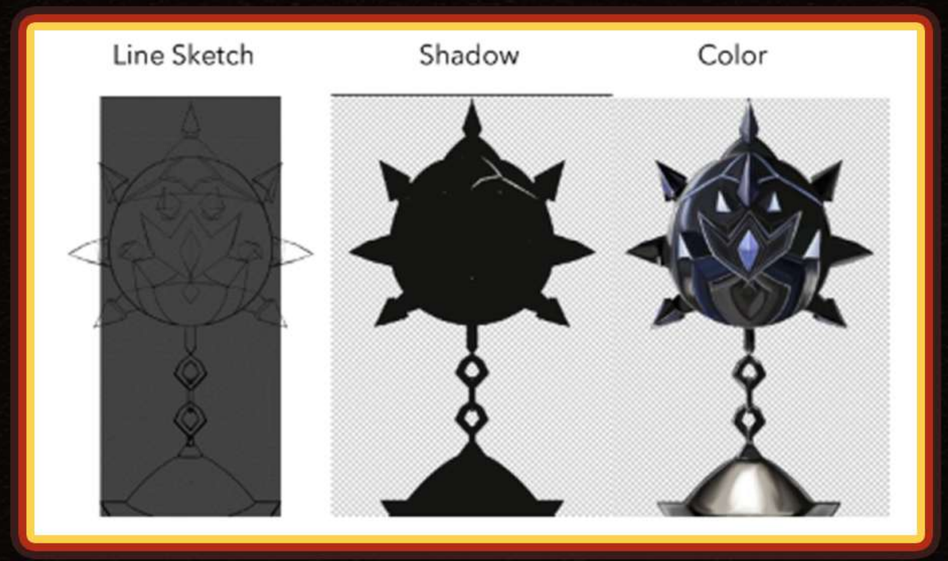
CONTACT US

For more information about this game or our other great games, email us at:

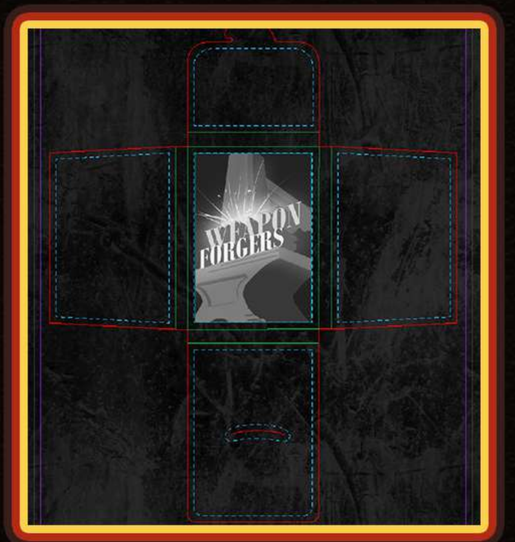
CADS@gmail.com

Card Production Process



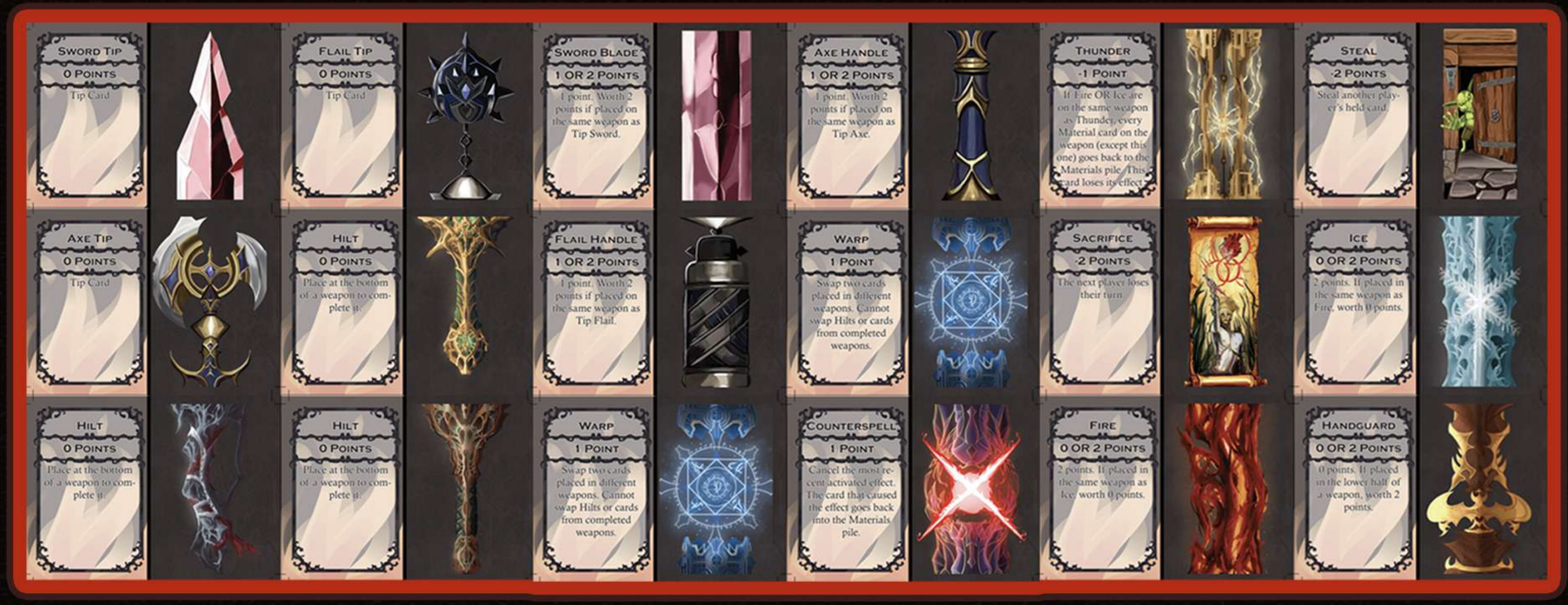


This page presents the production process of Weapon Forgers, from card artwork development to physical prototype assembly. We refined the weapon visuals, created card illustrations through sketching, shading, and coloring, then printed and cut the final cards for playtesting. The process also included rulebook icons and a custom card box design to make the game feel like a complete tabletop product.



Final Game Components

https://youtu.be/7_cOYFlyPEY



This is the final output of Weapon Forgers, including the completed card designs, card box, and physical prototype. The final card set supports the full gameplay loop of building weapons, applying card effects, and claiming the highest-scoring weapon. The linked video demonstrates the game setup, player turns, and scoring process.

ARG project
Theme: Climate Action
Name: Jinxuan Lu

Game description:

Story background:

In this ARG game, the player is very dissatisfied with how the government is currently managing global climate and weather changes. However, because the government is hiding key information and the real situation, civilians are unable to respond effectively or take action. The protagonist's goal is to expose the dark side behind the organization, reveal the hidden data, and identify the mastermind behind it.

(Organization name：EPA)

Game procedure:

The game takes place on a website. At the start, the player receives a clue from someone who previously tried to expose the organization's truth but failed， an old poster. The player must find a useful clue (a keyword) on the poster and enter it into the website's search engine to reach the next puzzle. Throughout the game, the player continually gathers information and solves puzzles to obtain new keywords, eventually discovering the mastermind's full name (first name and last name) and completing the main objective. Along the way, the player decodes files and uncovers the organization's hidden dark secrets.

Key and unique elements:

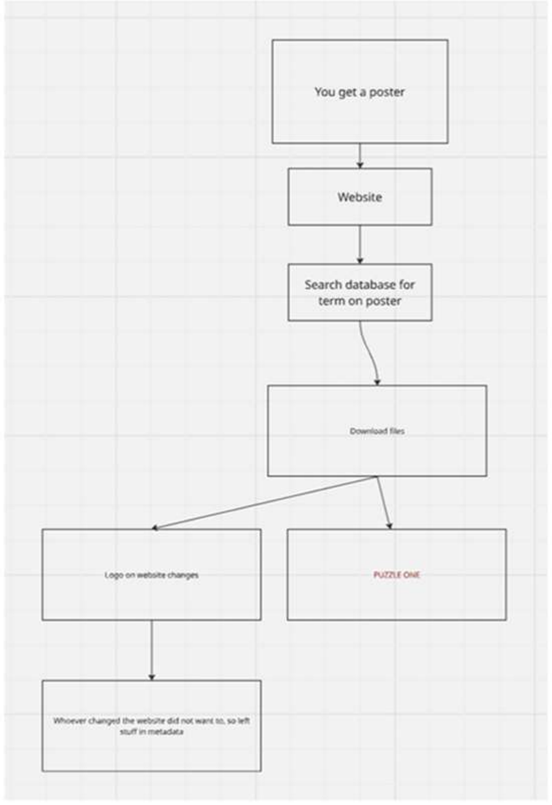
- The game creates a database-like experience where players use a website search engine to look up information. This helps players step into the protagonist's role and strengthens the role-playing immersion.

- The game uses a hybrid real-world + digital design. It begins with a physical-style poster, guiding the player's attention from a real object (the poster) and gradually shifting them into the online website world. This smooth transition helps maintain immersion and avoids feeling abrupt.

- The game heavily uses real online tools (e.g., Audacity, decoding tools) as interactive mechanics. Players get immediate feedback, can choose different decoding methods, and the tools support the "hacker/investigator" identity.

- The game follows a linear narrative structure, which keeps the story pacing stable, makes playtesting easier, and improves accessibility for first-time ARG players.

- The mechanics and theme are tightly connected. Each puzzle and decoding method matches the investigation theme and simulates real-world or film-style investigative processes. The story explains why data is hidden, why documents are redacted, why the player searches the "EPA" website, and why investigation tools are needed—so narrative and mechanics support each other.



(Game flow)

Team members and main role:

Grandin, Ryan----Transposition cipher puzzle; story background design; game flow planning
Lu, Jinxuan----------Audio puzzle; story outline design; game flow planning
Macdonald, Neil----Redacted documents puzzle; detailed script writing; game flow planning
O'bonsawin, Skyler-----Image puzzle; website building; logo illustration; game flow planning

Self-assessment contribution:

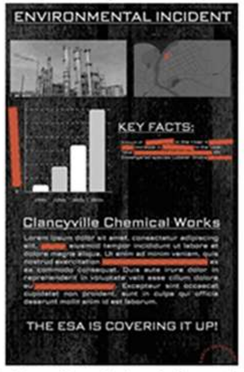
Overall, I am fairly satisfied with my contribution. I completed the audio section that I was responsible for and applied ARG techniques we learned in class, such as using software to create and read spectrograms, creating and decoding Morse code, and adding metadata notes to files to support the audio puzzle design. From a teamwork perspective, I attended every group meeting, both in class and outside of class. I also actively brainstormed the story background, including the idea of introducing the game through a "poster," and the overall direction of the narrative, such as making the main goal to identify the person behind the organization. Although not all of my ideas were adopted, they helped diversify the design options. Overall, I believe my contribution was fairly balanced within the team: I finished my own puzzle while also supporting the group with design input. This collaboration also allowed me to learn new skills and experience common problems in game design, along with ways to solve them.

Design approach and process（ARG structure）:

Our team's design process started by choosing the core ARG mechanics and the overall game structure. For example, I first decided to create an audio puzzle using a spectrogram and Morse code, and then wrote the story around those mechanics. The main structure of the game was built around a search engine. This approach ensured that each team member could design a puzzle within their own skill level, instead of forcing overly difficult puzzles just to fit the story, which could have slowed down production. After we confirmed what type of puzzle each person would create, we began designing the full game flow and narrative.

I designed the story outline. Based on the search-engine mechanic and the themes of "the organization" and "climate action," I developed a flow where the player acts as a hacker investigating and stealing hidden information—the player is the hacker. In my first draft of the script and overall flow, I introduced the idea of using a "poster" as the starting point, because I wanted the player to switch between the real world and the digital world to strengthen immersion. This helps the player feel like a real investigator, gathering clues from a physical object and then decoding information online.

"A poster about the person in charge, and climate change indicator data However, due to corrupted code, soome information is missing from the poster, for example, the main person-in-charge's name and appearance, and several pieces of data have been erased. The player's main task is to restore this poster by solving different puzzles."



(Poster with missing information)

The rest of the team built on this outline by adding details and expanding the world settings. The exact direction of the story was not the most important part， as long as it connected the gameplay and clearly supported the theme, different story setups could work. Because of this, we first wrote out multiple possible story ideas and then selected the version we were most satisfied with.

Specific Story Event Ideas:

- Maybe the government is hiding the fact that a species has gone extinct by means of non-environementally friendly actions. Instead, they are saying it was as a result of an invasive species possibly brought over by plane or boat
- Maybe the government has ignored petitions to shut down certain pollutenous businesses, and an employee who tried to speak up about it was fired and had their online presence wiped.
- oil spill cover-up

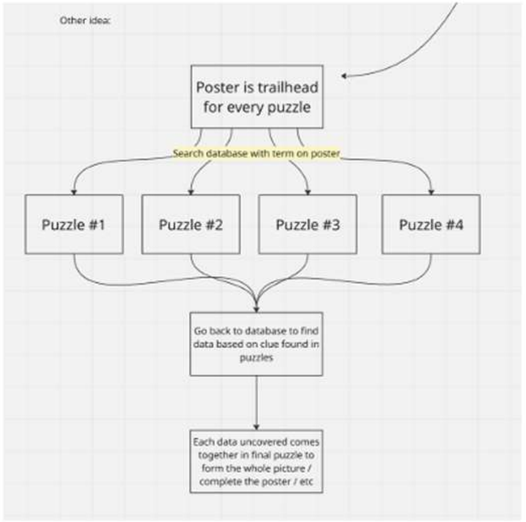
Characters:

- player: uninformed government "agent" who has recently become suspicious about the federal government's dubious relationship with environmental protection after the firing of the good guy
- good guy: another government agent who has recently been fired and "silenced" after making multiple accusations against the federal government. before his firing, he left secret documents and clues on the EPA's website
- big baddy (Naomi): person responsible for the various environmental atrocities descibed in the secret documents. also responsible for the hiding of the EPA's reports. high ranking government official. the player's poster is outing this person

The federal goverenment has been removing information from the EPA website, that puts them in a bad light. But the good guy has kept these documents/articles, etc hidden within there. The MC, a co-worker of the good guy, was given special access to the hidden documents and it is their job to find them and gather them into a propoganda poster.

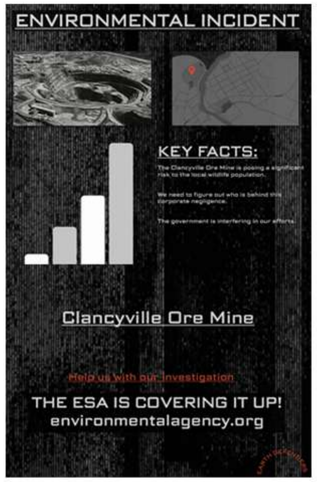
The diagram shows our initial game flow. Because a search engine is usually open-ended and people can search for anything they are interested in, our first version of the ARG was designed as a branching (multi-path) narrative. Players could collect any clues they found

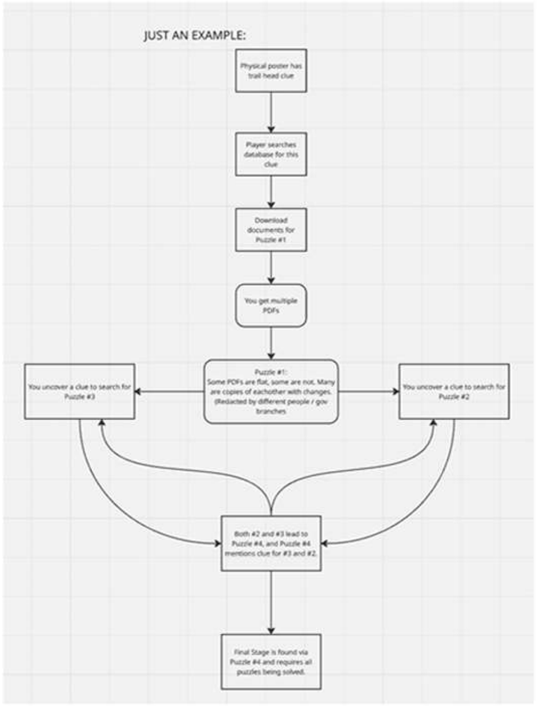
interesting and use them as search terms to explore different routes.



(First version)

However, because investigation-based stories usually require information to be revealed in a more linear order, showing too much at once can confuse players and cause them to lose track of the main objective. To avoid this, we revised the structure and changed it to the following flow.





(Second version)

Later, our playtesting showed that this revision was essentially still linear, the only difference was the order in which players solved the puzzles. However, the extra production time and workload did not create a proportional improvement in player experience. As a result, we decided to commit to a fully linear structure. The benefit of this choice is that the story becomes easier to control, and players are less likely to lose the main goal and quite due to confusion.

Revisions after Round 1 Playtest:

After the first round of testing, we found problems with the game's overall guidance and reward/feedback system. For example, the poster design included too many unnecessary elements that distracted player attention, and we also avoided repeating key words too much. In addition, when players found the correct keyword, there was no clear positive feedback to confirm they were on the right track. After discussion, I proposed revising the poster by removing distracting design elements (such as overly highlighted text, heavily censored lines, and meaningless photos). Instead, we added early hints, through small images or short prompt text, showing the decoding tools players would need later.

This change helps players recall possible solutions when they encounter later puzzles, instead of feeling completely stuck with no direction. It also allows one object (the poster) to be reused across multiple stages, which reduces the number of new items introduced and prevents the game from becoming overloaded with new hints every level. Reusing the poster also improves immersion and narrative consistency: if the poster becomes useless after the first keyword, players may see it as only a "door key." By keeping it relevant, it functions more like a real evidence board or clue source, strengthening the ARG's realism. Once players understand the poster's importance, they can return to it to re-check clues and confirm their decoding approach, reducing the chance of losing direction.

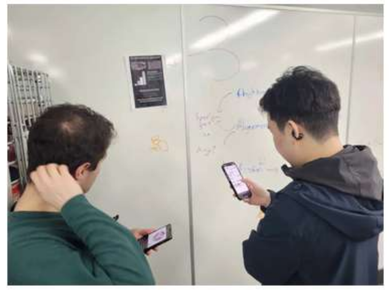
Individual contribution:

My contributions to the ARG project mainly included creating the audio puzzle, designing the overall story/flow outline, and providing revision suggestions.

Audio design process:

I chose to make an audio puzzle because I noticed it is a common way for game designers to hide secrets or "easter eggs." For example, in Minecraft, there is a record that reveals a creeper face when viewed in a spectrogram. This idea strongly interested me, so I used a spectrogram-reading technique in the first audio puzzle to let players decode hidden information. I chose Morse code for the second part because it is easier to produce and also takes time to decode, which helps prevent players from finishing the spectrogram too quickly and completing the game too fast. Overall, I wanted to keep the total playtime around five minutes.

In the first version of the audio design, the player's goal was to identify two specific audio files required to progress. One audio file was clearly marked as the starting point, and after decoding it, the player would obtain the second audio file, which led into the next puzzle. All other audio files did not directly advance the main flow; they were mostly dialogue recordings designed to mislead the player and reduce the chance of randomly selecting the correct second file without solving the puzzle. However, if the player listened carefully, these dialogues could reveal more about the overall story.

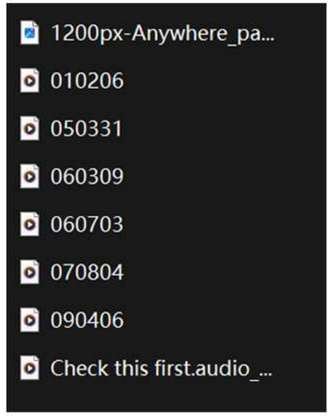


Audio puzzle procedure

Jinxuan Lu

Goal: Find the full name of the organization manager (First name + Last name).

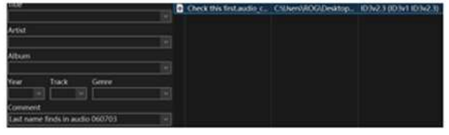
After the player downloaded the file they will see a lot of audios, but only two of them can used to find the right answer.



Step1: Open the file with the title (check this file)

Use the Audacity tools to check the sound Spectrum.

Step2 to check the comment inside the attribute of the MP3 to see find out the audio that contain the last name of the person. (clue are shown in the name of the file "Check this first.audio_comment")



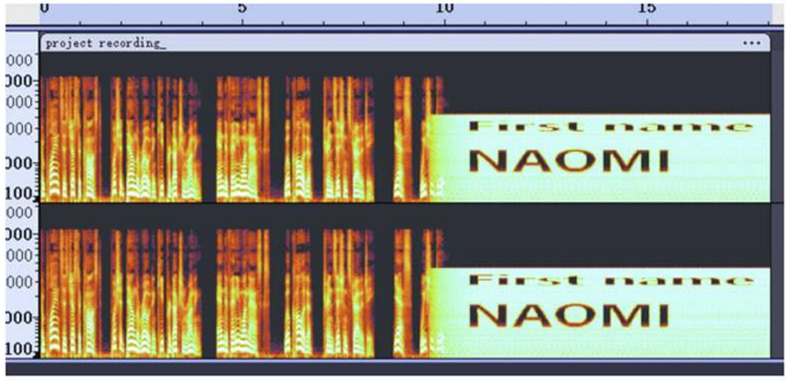
Step 3, find out the MP3 file name 060703, to solve the morse code.

Final answer: Naomi Carter

(First version)

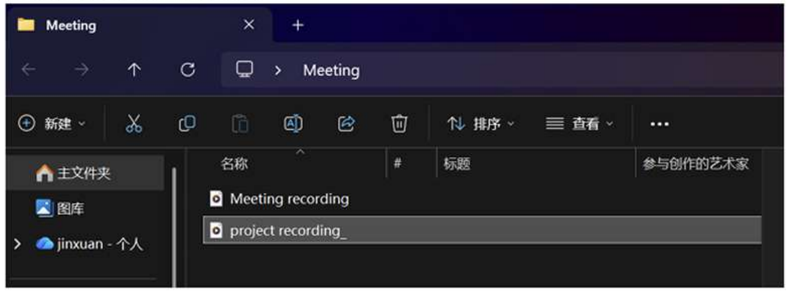
However, later playtesting showed that there were not enough clear hints to help players realize they needed to check file metadata, and reading metadata also requires some technical computer knowledge. Adding this step created an unnecessary barrier, but my goal was to make a game that anyone could play. Therefore, in the final version, I removed the metadata-checking step. To better control the playtime, I also removed the extra "misleading" audio files that were only used for confusion.

In the final version, I changed the player's objective to finding the mastermind's full name (first name + last name). I also added human dialogue into the spectrogram audio, and within the conversation I embedded the sound pattern that reveals the first name in the spectrogram. This improves role-play immersion while ensuring the puzzle stays connected to the game's theme and narrative.



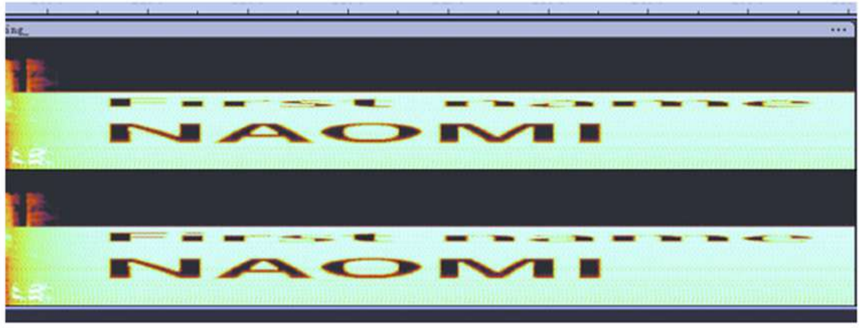
Solution:

When the player enters the correct keyword into the website's search engine, the site downloads the two audio files.



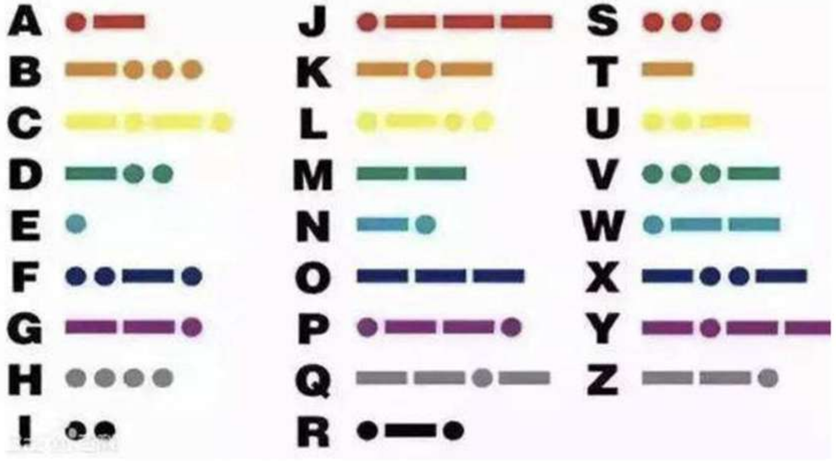
Step1:

The order does not affect the final answer. When the player imports the project recording into Audacity and views the spectrogram, they can discover the first half of the answer.



Step2:

The player opens the meeting recording and decodes it using a Morse code chart, which reveals the last name, the second half of the answer.



Answer: First name: Naomi

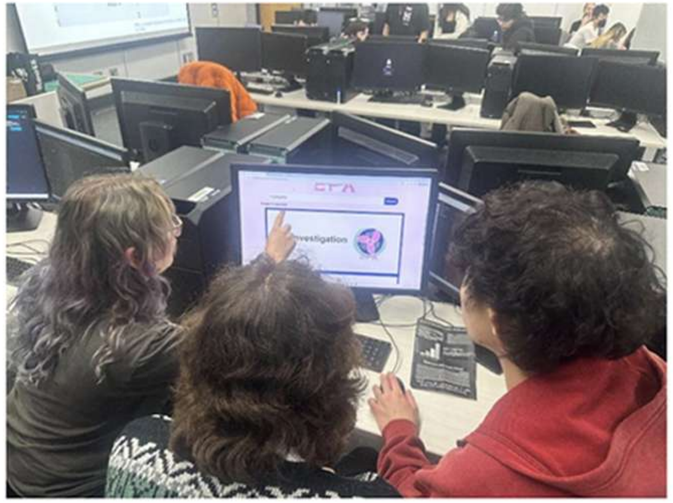
Last name: Carter

Connection：

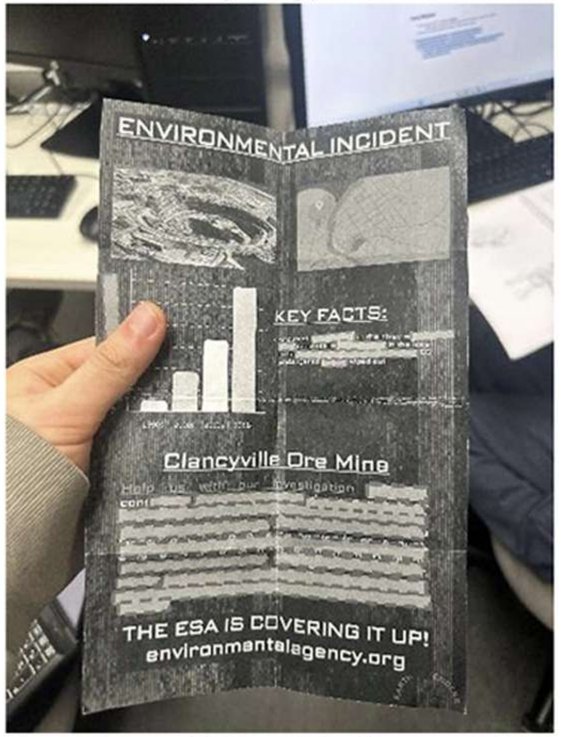
My audio puzzle was designed to fit into the team's overall ARG structure as the final puzzle. Players progress by entering keywords into the EPA website's search engine and continuously collecting evidence. After they discover the name of the person responsible, they enter it into the search engine one more time to complete the game.

The answer to the audio puzzle"Naomi Carter"directly supports our team's main objective: identifying the mastermind behind the cover up. As the closing puzzle, it gives the simplest but most important piece of information: the name the player has been working toward throughout the entire experience. It also connects to the other puzzles because the player cannot fully understand the story without solving this final step. In addition, the keyword needed to access the audio puzzle appears in the previous puzzle as its answer, which foreshadows the next puzzle type and guides the player forward.

Although the game ends for the player, the story treats this moment not as an ending, but as a beginning. Once the protagonist knows the responsible person's name, they can begin real resistance and speak up for climate action. The decoding methods are also echoed from the start through the poster: the poster's images and short lines hint at the tools and techniques needed later. Even if players do not notice these hints immediately, they can recall them when they reach the audio puzzle, helping them find the correct approach instead of getting stuck.

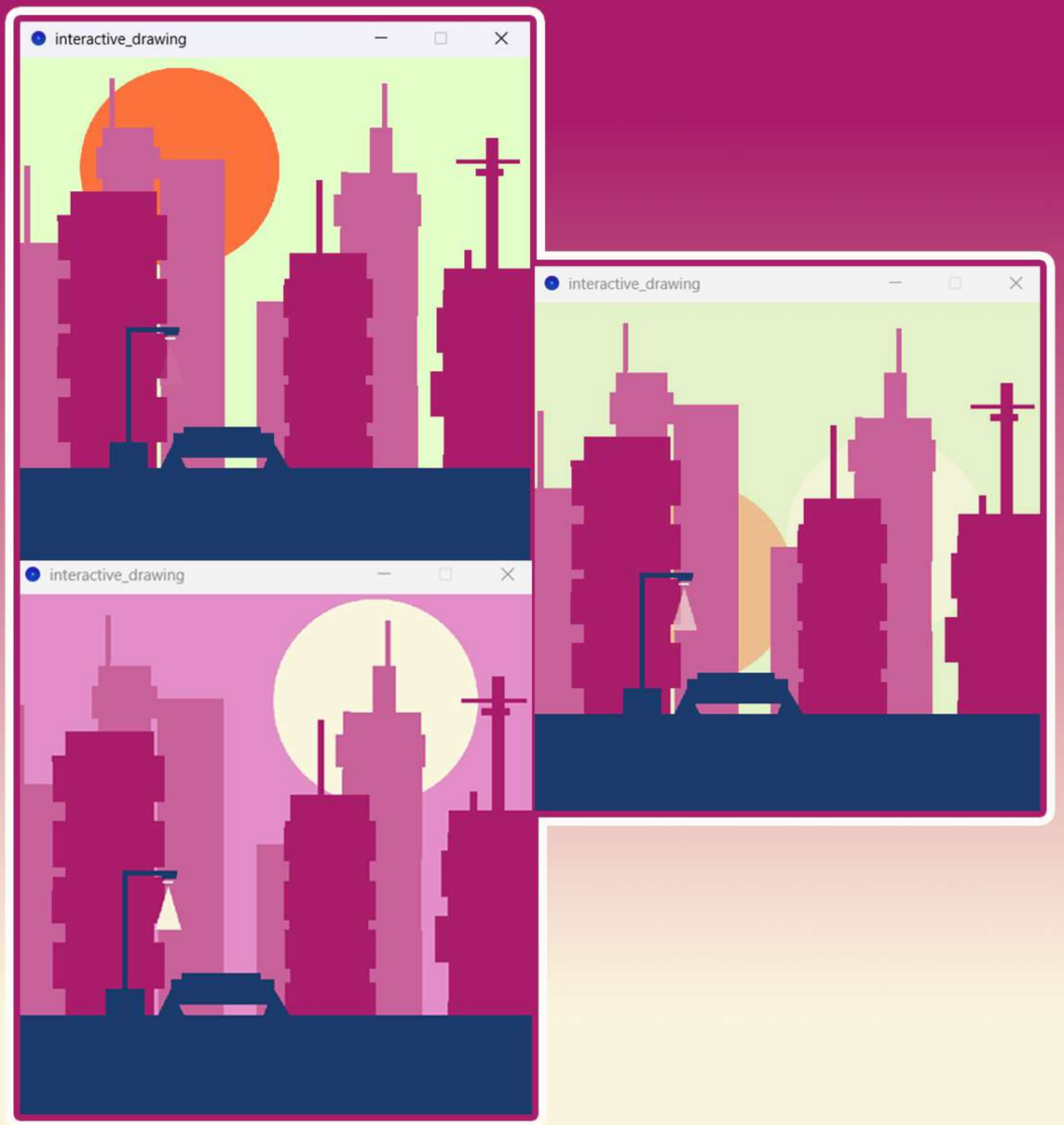


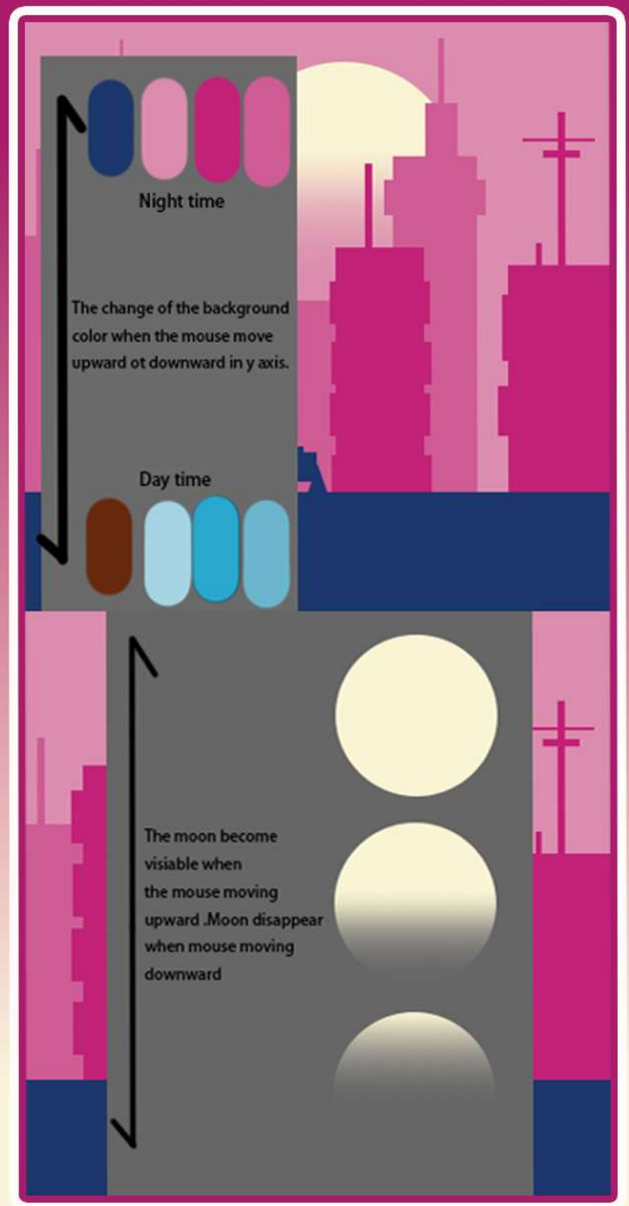
(Play testing photo)

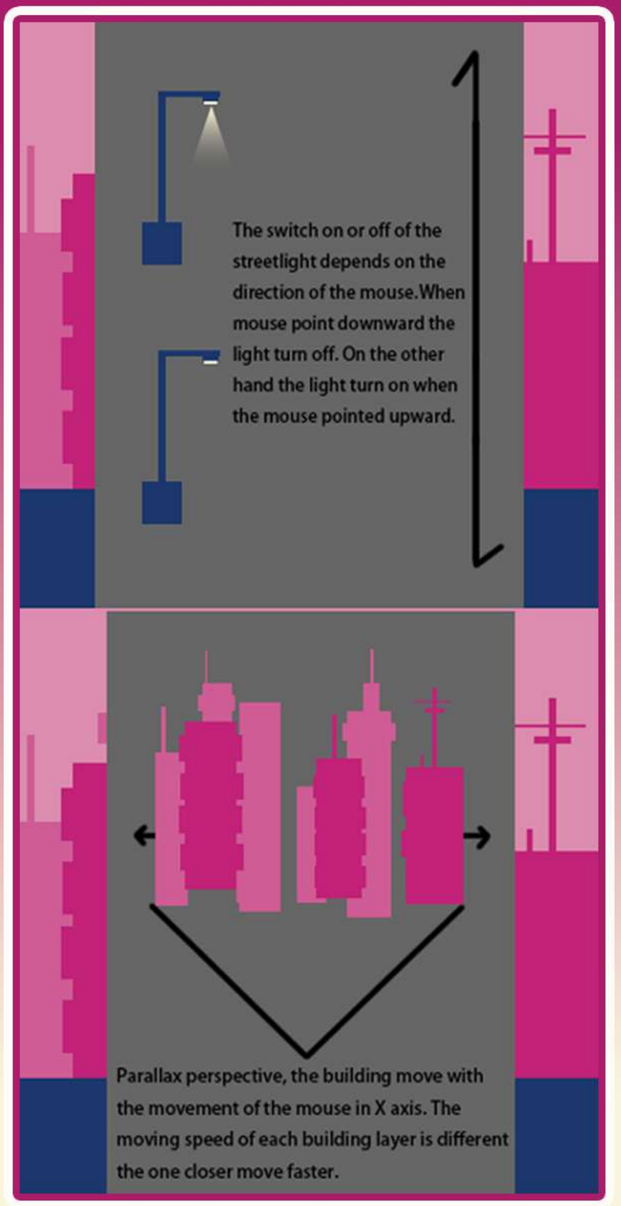


(Poster as tile head)

Neon City


interactive_drawing
interactive_drawing
interactive_drawing


Night time
The change of the background color when the mouse move upward ot downward in y axis.
Day time
The moon become visiable when the mouse moving upward .Moon disappear when mouse moving downward


The switch on or off of the streetlight depends on the direction of the mouse.When mouse point downward the light turn off. On the other hand the light turn on when the mouse pointed upward.
Parallax perspective, the building move with the movement of the mouse in X axis. The moving speed of each building layer is different the one closer move faster.

Code

interactive drawing

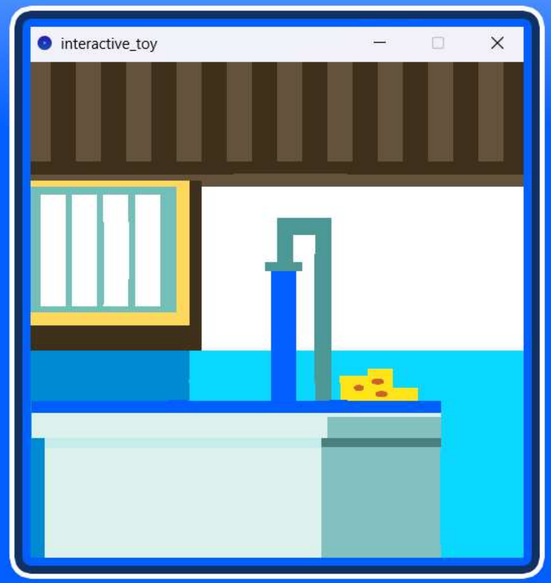
```
void setup(){
  size(400,400);
  noStroke();
}
void draw(){
background(227,mouseY+60,200);

  //Moon
 fill(250,246,222,340-mouseY);
ellipse(275,mouseY,155,155);
 //Sun
 fill(255,95,41,mouseY-90);
 ellipse(125,height-mouseY,155,155);

  //Building in the backward
fill(201,97,154);
rect(0,145,40,255);
rect(4,85,5,315);
rect(65, 55, 40,345);
rect(60,70,50,25);
rect(70,17,4,383);
rect(110, 80, 50,320);
rect(185, 190, 25,210);
rect(250, 90, 60,310);
rect(273,55, 17,345);
rect(282,21,4,379);
rect(245,107,68,25);


//Building in the frontward
fill(170,29,107);
rect(40,105,67,295);
rect(30,123,85,35);
rect(30,173,85,35);
rect(30,215,85,35);
rect(30,265,85,35);

rect(210,153,60,250);
rect(206,173,69,30);
rect(206,218,69,30);
rect(206,255,69,30);
rect(206,295,69,52);
rect(231,96,5,304);
rect(330,165,70,235);
rect(325,195,75,35);
rect(320,245,80,35);
rect(363,63,10,337);
rect(340,80,50,3);
rect(355,87,22,6);
rect(346,150,6,50);
```

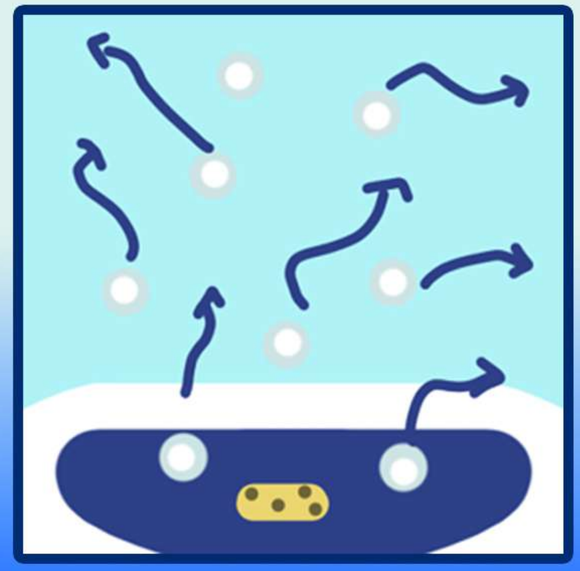
```
//Streetlight
fill(27,59,108);
rect(70,300,30,20);
rect(83,210,87-83,320-210);
rect(83,210,125-83,214-210);
rect(113,213,124-113,217-213);
//Branch
rect(120,293,198-120,312-293);
rect(128,288,188-128,312-293);
triangle(120,300,110,320,130,320);
triangle(198,300,190,320,210,320);

//light
fill(250,246,222);
rect(114,218,122-114,220-218);
fill(250,246,222,330-mouseY);
triangle(118,222,108,255,128,255);

//Ground
fill(27,59,108);
rect(0,320,400,80);

}
```

Interactive Day & Night Scene

An interactive environment study where mouse movement controls the transition between daytime and nighttime. Moving the mouse vertically changes the sky color, controls the moon's position, and adjusts the visibility of the streetlights. When the mouse moves upward, the scene becomes brighter and the moon disappears; when the mouse moves downward, the environment shifts into night with stronger moonlight and streetlight effects. The project also explores parallax movement, using different mouse X values to create depth in the city skyline. This interaction was built using mouse input, RGB color changes, transparency values, and position-based animation.

Bath Room Toy





Core Interactions

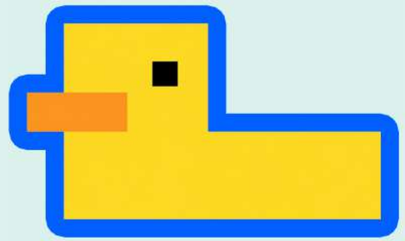
.Random bubble movement with wall bouncing

.Bathtub and water define the playable area

.Sponge follows horizontal mouse control

.Clicking bubbles triggers explosion feedback and removal

Code



interactive toy

```
//The initial value of the R(red in RGB)
int R = 255;
//The initial value to control the transpancy of the water
int T = 0;
int duckxposition = 60;
//Position of duck y use to change the duck's position after turn on the water
int duckyposition = 255;
//Position of sponge y use to change the duck's position after turn on the water
int spongeyposition = 260;

//Condition to test the  turn on/off of the tap
boolean a = false;
//Condition to test the creation of the bubble
boolean bubblealive = false;
//Value of the movement of bubble
float bubbleX, bubbleY;
float bubblenewX, bubblenewY;
float bubbleSize = 18;

void setup(){
size(400,400);
noStroke();

}

void draw(){
  background(255);
  //To constrain the movement of the duck only following the mousex with in the wide of the bathtub
  int duckX = constrain(mouseX, 15, 330 - 40);

  //Bath room
  //Brown wood textrue ceiling
fill(100,83,60);
rect(0,0,400,100);
fill(62,48,27);
rect(0,80,400,10);
rect(20,0,20,80);
rect(60,0,20,80);
rect(100,0,20,80);
rect(140,0,20,80);
rect(180,0,20,80);
rect(220,0,20,80);
rect(260,0,20,80);
rect(300,0,20,80);
rect(340,0,20,80);
rect(380,0,20,80);
```

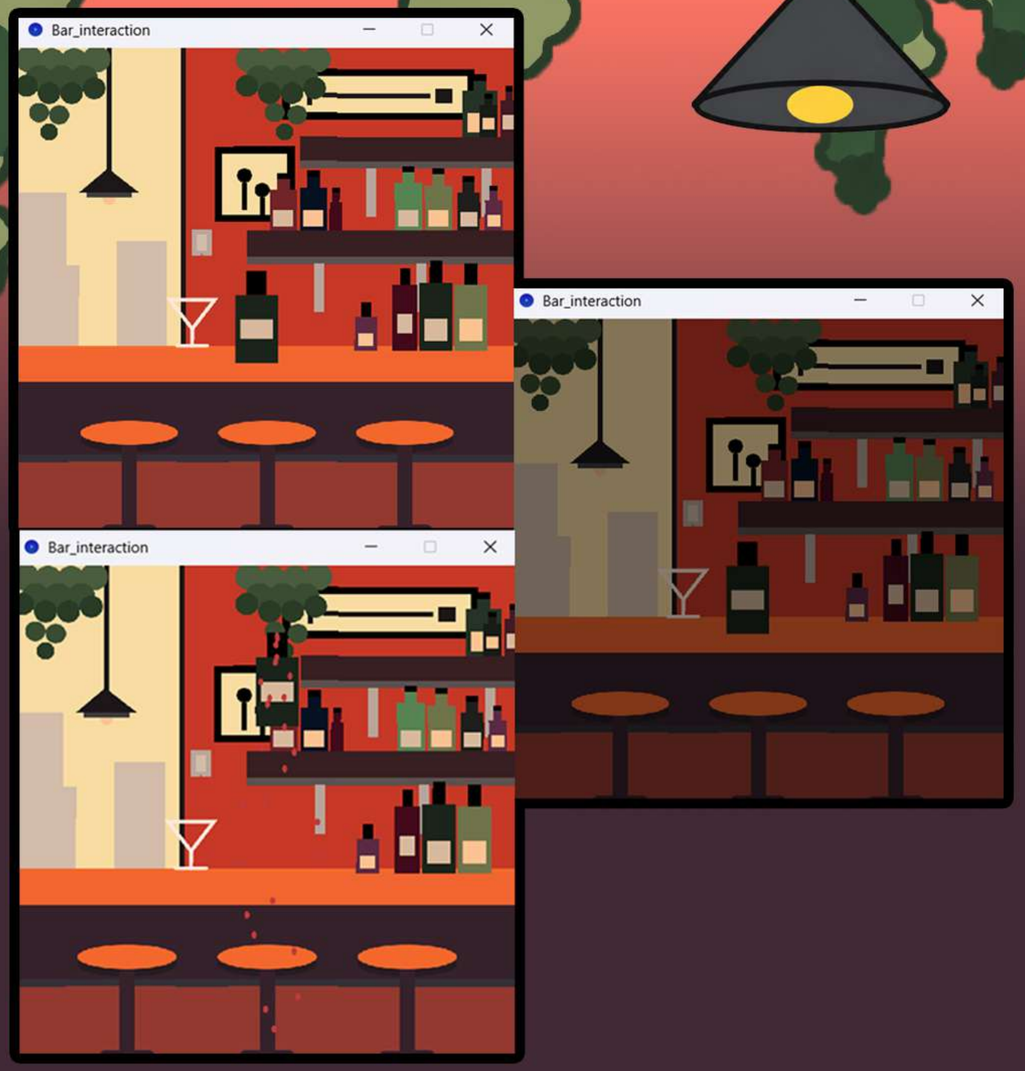
interactive toy

```
//The initial value of the R(red in RGB)
int R = 255;
//The initial value to control the transpancy of the water
int T = 0;
int duckxposition = 60;
//Position of duck y use to change the duck's position after turn on the water
int duckyposition = 255;
//Position of sponge y use to change the duck's position after turn on the water
int spongeyposition = 260;

//Condition to test the  turn on/off of the tap
boolean a = false;
//Condition to test the creation of the bubble
boolean bubblealive = false;
//Value of the movement of bubble
float bubbleX, bubbleY;
float bubblenewX, bubblenewY;
float bubbleSize = 18;

void setup(){
size(400,400);
noStroke();

}

void draw(){
  background(255);
  //To constrain the movement of the duck only following the mousex with in the wide of the bathtub
  int duckX = constrain(mouseX, 15, 330 - 40);

  //Bath room
  //Brown wood textrue ceiling
fill(100,83,60);
rect(0,0,400,100);
fill(62,48,27);
rect(0,80,400,10);
rect(20,0,20,80);
rect(60,0,20,80);
rect(100,0,20,80);
rect(140,0,20,80);
rect(180,0,20,80);
rect(220,0,20,80);
rect(260,0,20,80);
rect(300,0,20,80);
rect(340,0,20,80);
rect(380,0,20,80);
```

```
//Blue green layer of the window
fill(116,191,185);
rect(0,100,120,100);
//Outside view of the window
//In night color represent night time
fill(0,0,0);
rect(12,106,20,195-106);
rect(37,106,20,195-106);
rect(62,106,20,195-106);
rect(87,106,20,195-106);
//Use mouseX to control the transpancy value. When mouseY = 255, color become white and represent day time
fill(255,255,255,mouseY);
rect(12,106,20,195-106);
rect(37,106,20,195-106);
rect(62,106,20,195-106);
rect(87,106,20,195-106);

//Yellow layer of the window
fill(255,217,93);
rect(0,95,135,5);
rect(120,100,10,115);
rect(0,200,130,10);

//Brown lay of the window
fill(62,48,27);
rect(0,210,140,20);
rect(130,95,10,220-95);

//Blue background of the bath room
fill(0,139,211);
rect(0,230,130,400-230);
fill(10,216,255);
rect(130,230,400-130,400-230);

//Bath tub
fill(220,242,236);
rect(5,280,325,20);
rect(15,300,315,100);

//Shadow
fill(196,237,236);
rect(15,300,315,8);
fill(131,193,192);
rect(240,283,330-240,300-283);
rect(235,303,330-235,400-303);
fill(74,129,126);
rect(235,300,330-235,7);
```

```
if (bubblealive) {
  // Movement of the bubble
  bubbleX += bubblenewX;
  bubbleY += bubblenewY;

  // loop effect
  //Use loop to create the textrue of the bubble, with three layer.
  //loop in 3 rounds
  for (int i = 0; i < 3; i++) {
    fill(122,198,247,120 - i * 30);
  //Outer the layer, higher transpancy
    ellipse(bubbleX, bubbleY, bubbleSize + i * 10, bubbleSize + i * 10);
  //Outer the layer, larger the size of the layer
  }
}
}
void keyPressed(){
  //To build the function of the switch.
  //The close and open of the tap
if (!a){
  a = true;
}else{
  a = false;
}
}
void mousePressed() {
  // The interaction happen in the sponge's range(x,y)
  if (mouseX >= 250 && mouseX <= 290 && mouseY >= 260 && mouseY <= 280) {

  //Creation of the bubble inside the sponge's range
    bubblealive = true;
  //Random position of the bubble created in side the range of sponge
    bubbleX = 270 + random(-5, 5);
    bubbleY = 270 + random(-5, 5);

  //The upper movement of the bubble, in a random moving speed
    bubblenewX = random(-5, 2);
    bubblenewY = random(-5, -0.5);

   //The random size of the bubble, each bubble created is different in size
    bubbleSize = random(25, 60);
  }
}
```

Bar Game



A simple bartending game where players mix drinks in a bar scene built with basic rectangles and circles. Players can pick up bottles with the mouse, move them with a wobble effect, and press Space to pour liquid into a glass.

Additional interactions include turning a desk lamp on and off, random lamp flickering, and smashing a wall bottle. The project uses conditionals, classes, and random functions to create playful interactive feedback.

Code

Bar interaction | Bottle dropoff | Bottle pickup | Reference

```
//Use of boolean expression to test the conditon of the interaction of the toy
boolean holdBottle = false;
//The condition to test if bottle is being held. use to determind wheather the bottle is following the mouse
boolean pourDrink = false;
//The condition to test if space is held (pouring the liquid). when the condition is true the liquid stop droping
boolean lampOn = true;
//The condition to test lamp on/off. when is the condition is false the light turned off. image turn dark. when is true the light turn on
Bottle bottle;
//An object from Bottle class that the player can interact with.to carry or pour liquid.

ArrayList<Drop> drops = new ArrayList<Drop>();
//The funtion of the arrayList stores the values of many drops. use to deal with the effect that the liquid is pouring down and pass certain area it disappeared.
int lampBtnX = 140;
int lampBtnY = 150;
int lampBtnW = 16;
int lampBtnH = 22;
//The hti box of the lamp button in a rectangle area. when the mouse click inside the area the light switch on or off

void setup() {
  size(400, 400);
  noStroke();
  bottle = new Bottle(200, 230);
  //Create bottle at start position on the table. it move back to the place everytime it restart
}

void draw() {
//The drawing of the bar
  background(199, 56, 39);
//view outside window
  fill(249, 220, 162);
  rect(0, 0, 131, 250);
//building outside the window
  fill(211, 188, 169);
  rect(0, 120, 40, 130);
  fill(211, 188, 169);
  rect(0, 180, 50, 120);
  fill(211, 188, 169);
  rect(80, 160, 40, 150);
//line between window and thw wall
  fill(28, 23, 24);
  rect(131, 0, 4, 400);
//table of the bar
  fill(244, 104, 45);
  rect(0, 246, 400, 6);
  fill(242, 101, 48);
  rect(0, 250, 400, 26);
  fill(54, 34, 43);
  rect(0, 276, 400, 66);
//floor
  fill(53, 48, 55);
  rect(0, 316, 400, 6);
```

Bar interaction | Bottle dropoff | Bottle pickup | Reference

```
class Drop {
  // Drop class used for the value,position and display of every drop

  PVector pos;
  //Position of the drop

  PVector vel;
  //Velocity of the drop when moving

  PVector acc;
  //Acceleration of the drop when falling

  Drop(float x, float y)
  //Creation of a new drop
  {

    //Start position of the drop
    pos = new PVector(x, y);

    //Start speed of the drop with random to make the falling feel more close to reality
    vel = new PVector(random(-1, 1), random(1, 2));

    //Gravity of the drop, the increase in velocity when falling down
    acc = new PVector(0, 0.2);
  }

  //Update the movement of every drop
  void update() {

    //Speed adds acceleration make the falling speed faster
    vel.add(acc);

    //Position adds velocity,make it moving
    pos.add(vel);

  }

  void display()
   //Display and drawout the drop, in earch creation
  {
    fill(190, 60, 60);
    ellipse(pos.x, pos.y, 4, 6);
    //gernated in the position of x and y
  }
}
```

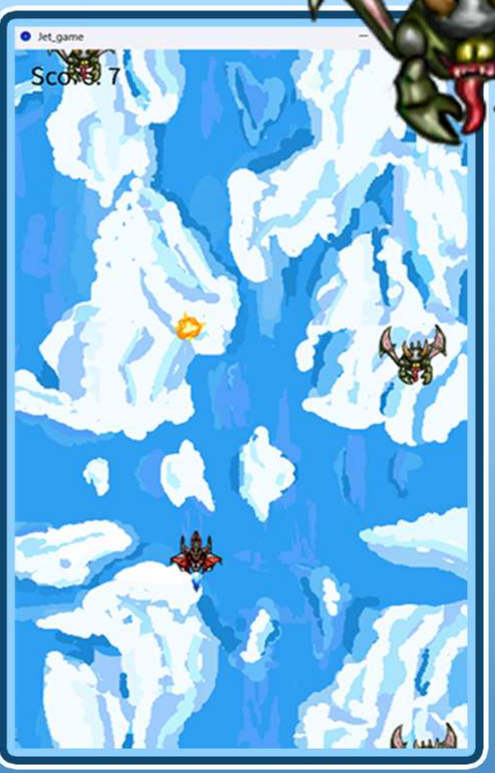
Bar interaction | Bottle dropoff | Bottle pickup | Reference

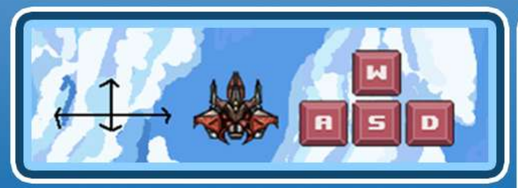
```
class Bottle {
  // Bottle class used for the value inside the bottle itself

  PVector pos;
  //position of the bottle

  PVector vel;
  //velocity of the bottle when moving

  PVector acc;
  //acceleration of the bottle when following with the mouse

  boolean held = false;
  //condition to test holding bottle

  Bottle(float x, float y)
  //the x and y value of the bottle
  {

   //Set the starting value for PVector
    pos = new PVector(x, y);
    vel = new PVector(0, 0);
    acc = new PVector(0, 0);
  }

  //Update movement
  void update() {

    //Every frame need to reset the acc or it may accumulate and keep increasing become faster and faster
    acc.x = 0;
    acc.y = 0;

  //If held, acc points to mouse
    if (held) {
      acc.x = (mouseX - pos.x) * 0.1;
      acc.y = (mouseY + 10 - pos.y) * 0.1;
    }

  //The update of the vel and pos
    vel.x = (vel.x + acc.x) * 0.5;
    vel.y = (vel.y + acc.y) * 0.5;

    pos.x = pos.x + vel.x;
    pos.y = pos.y + vel.y;
//when *decimal number make the bootle shake in a small range
  }

  //Display bottle
  void display() {

    //Bottle body
```

Jet Game

2D Shooting Game

A 2D arcade shooting game where the player controls a fighter jet, dodges incoming enemies, and shoots bullets to destroy targets. Random enemy spawning, collision detection, explosion effects, scoring, and a scrolling cloud background create a fast-paced flying experience. The project uses variables, conditionals, arrays, and classes to manage player movement, bullets, enemies, and game feedback.





Code

Jet game | Jet | bullet | monster | reference

```
//I give each game paper one values to represent it's status. by using the 'if'conditional code,to switch the game status value can change the whole game page.
// Insert all the the image that used in the game
PImage[] jetImage;//image for jet. using arraylist is because to include the animation of the flame of the jet
PImage monsterImage;//image for monster
PImage mapImage;//image for map
PImage[] boomImage;//image for the boom animation

Jet jet;//Create object for jet

Bullet[] bullets = new Bullet[25];//create object for bullets. using arraylist to make 25 bullet exist at the same time

Monster[] monsters = new Monster[8];//create object for monster. using arraylist therefore can have 8 monsters at the same time

float background1 = 400;// y axis of two background. using two same background image to make loop. create the feeling that the jet is flying forward
float background2 = -400;//the different of the two value add up to  800.therefore can cover all the blank space as the map size = 800

float backgroundSpeed = 5;//The speed movement of the background

int score = 0;//Variables for the score system in the game. the score values change base on how many monster player killed

int jetFrame = 0;//the number of frame in the animation

int gameState = 0;//value set for creating game status. when number is 0 = main page. 1=game page.2= died page.
//default the game value is 0. which mean enter the game in the main page.

void setup() {
  size(500, 800);//size for paper

  frameRate(60);//Set the frame to 60. make the animation and the game image looks smooth.

  imageMode(CENTER);//all the image draw from the center

  jetImage = new PImage[6];//create array to store 6 images for the jet animation frames
  for (int i = 0; i < jetImage.length; i++) {//a loop to load every jet image one by one
    jetImage[i] = loadImage("jet_smile " + (i + 1) + ".png");//load the image in order from jet_smile 1. to 6
    jetImage[i].resize(180, 180);//resize all the image to 180*180.make the jet in suitable size. and make sure all the animation image is in the same size
  }

  boomImage = new PImage[6];//create array to store 6 image for boom animation
  for (int i = 0; i < boomImage.length; i++) {//loop to load all the 6 frame animation
    boomImage[i] = loadImage("boom " + (i + 1) + ".png");//load the image in the order from 1 to 6
    boomImage[i].resize(180, 180);//resize the image to 180*180
  }

  monsterImage = loadImage("monster.png");//load the monster image to use as the appearance of the monster

  mapImage = loadImage("map.png");//load the map image use as background

  monsterImage.resize(250, 250);//size change of monster image
  mapImage.resize(500, 800);//size change of map size
```

Jet game | Jet | bullet | monster | reference

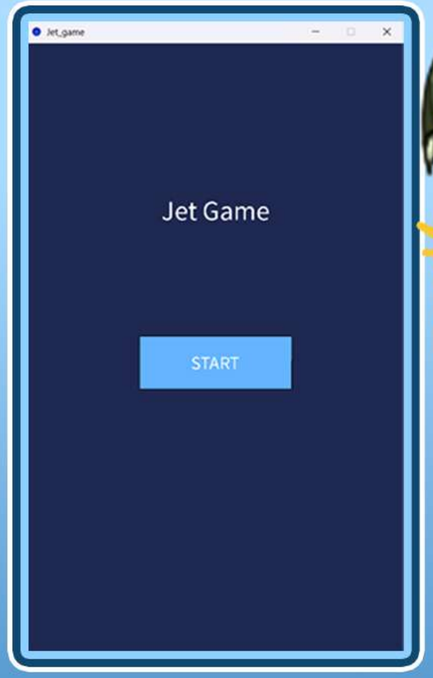
```
//bullet interaction using for loop. condition code. array for animation and boolean to fire the bullet.
class Bullet {// Define the Bullet class for the projectile shot by the jet.update's value and status

  PVector pos;// bullet's current position on the screen

  PVector vel;// bullet's movement speed. moving forward

  PVector acc;

  boolean active;// Track whether the bullet is currently being fired in the game

  Bullet(float tempX, float tempY) {// Constructor that creates a bullet at a starting position

    pos = new PVector(tempX, tempY);// Set the bullet's starting position using x and y values

    vel = new PVector(0, -15);// Give the bullet an upward speed so it moves toward the top of the screen. no changgin in x

    acc = new PVector(0, 0);// Start with no acceleration changing the bullet's speed

    active = false;// make the bullet inactive until the player shoots it.keep status to false
  }

  void move() {// Update the bullet's movement every frame

    vel.add(acc);// Add acceleration

    pos.add(vel);// Add velocity

    if (pos.y < 0) {// Check if the bullet has moved past the top edge of the screen
      active = false;// Deactivate the bullet so it disappears and can be reused. reduce stress for computer
    }
  }

  void display() {// Draw the bullet on the screen

    fill(255, 0, 0);// Set the bullet color to red

    noStroke();

    ellipse(pos.x, pos.y, 12, 20);// Draw the bullet as an oval at its current position
  }
}
```

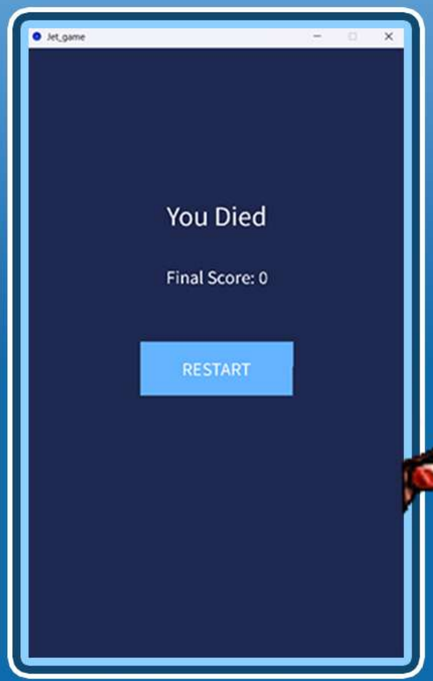
Jet game | Jet | bullet | monster | reference

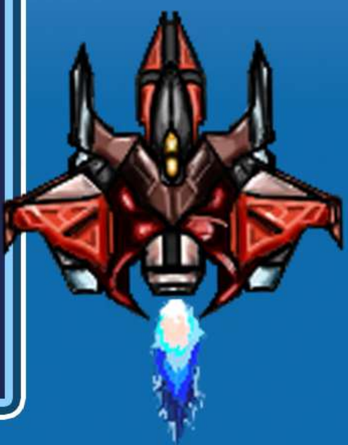
```
class Jet {// Define the Jet class, put all the jet's related value and function

  PVector pos;// jet's position on the screen

  PVector vel;// The jet's movement speed in x and y directions

  PVector acc;// the jet's acceleration

  boolean alive;// check the alive condition of the jet

  Jet(float tempX, float tempY) {// Constructor that creates a jet at a starting position. using tempx and y is because jet's xy is changing all the time.

    pos = new PVector(tempX, tempY);// Create the position vector using the given x and y values

    vel = new PVector(0, 0);// Start with no movement speed in any direction

    acc = new PVector(0, 0);// no acceleration affecting the jet when game start

    alive = true;// Set the jet as alive when it is first created
  }

  void move() {// Update the jet's movement every frame

    vel.add(acc); // add acceleration to velocity

    pos.add(vel);// add velocity to position so the jet moves on the screen

    pos.x = constrain(pos.x, 45, 450);// Keep the jet fly inside this screen boundaries

    pos.y = constrain(pos.y, 40, 750);
  }

  void display() {// Draw the jet image at its current position

    image(jetImage[jetFrame], pos.x, pos.y); // show the current animation frame of the jet on screen
  }
}
```

Jet game | Jet | bullet | monster | reference

```
//monster interaction include randomness movement, move in random x and y at the same time keep moving toward player
//when monster touch the jet game over
//monster have explode animation using for loop and array.
class Monster {// Create monster class as enemy in the game for updating status

  PVector pos;// monster's current position on the screen

  PVector vel;// monster's movement speed

  PVector acc;//monster's acceleration

  boolean exploding;// Track whether the monster is currently playing its explosion animation

  int explodeFrame;// Store which explosion image frame is being shown now

  int explodeDelay;// Time used to show the boom animation after the monster been killed

  Monster(float monsterX, float monsterY) {// Constructor that creates a monster with a starting position

    pos = new PVector(monsterX, monsterY);// monster's starting position in x and y

    vel = new PVector(random(-2, 2), random(3, 8));// Give the monster a random sideways and downward speed.want the monster move random in x but still keep forward consitently in y

    acc = new PVector(0, 0);//no acceleration affecting the monster

    exploding = false;// Set the monster to its normal state at the beginning

    explodeFrame = 0;// frame start at zero

    explodeDelay = 0;// Start the explosion timing counter at zero
  }

  void move() {// Update the monster's movement every frame

    if (exploding == false) {// Only move the monster when it is not exploding

      vel.add(acc);// Add acceleration to velocity

      pos.add(vel);// Add velocity to position

      if (pos.y > 900) {// if the monster has moved below the bottom of the screen
        reset();// Reset the monster to a new starting position above the screen
      }

      pos.x = constrain(pos.x, 45, 450);// Keep the monster inside the left and right screen boundaries
    }
  }

  void display() {// Draw the monster or its explosion animation on the screen
```







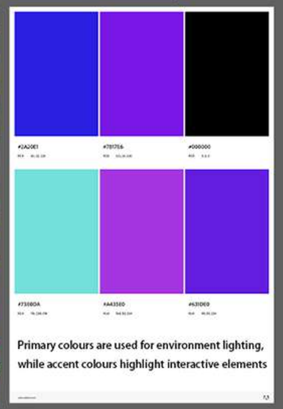
2D Assert

Environmental Atmosphere

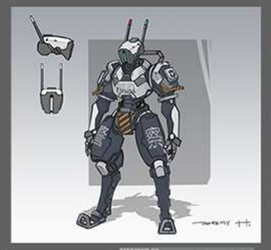





Colour Style



Player Character








MOOD BOARD
CYBERPUNK NEON FUTURE CITY


Game UI/Logo


Visual Scene







Enemy Character