Project Tightrope

Team Tightrope

HCID 521: Prototyping Studio

Varun, Samantha, Annie, Husna

What is Tightrope?

Overview

Tightrope is a **narrative-driven, multiplayer interactive experience** about the **physical and mental experiences of anxiety**, and how we can **support those around us** who may be going through a tough time.

Player 1 takes on the role of the protagonist going through a tough time and trying to cross a precarious tightrope. Their goal is to **collect golden orbs**, representing positivity and joy, and **avoid dark orbs**, representing negativity and self-doubt.

Players 2 and 3 take on the role of Player 1's **guardians** (represented as owls). Their goal is to **move golden orbs within Player 1's reach** and **move dark orbs out of Player 1's way**.

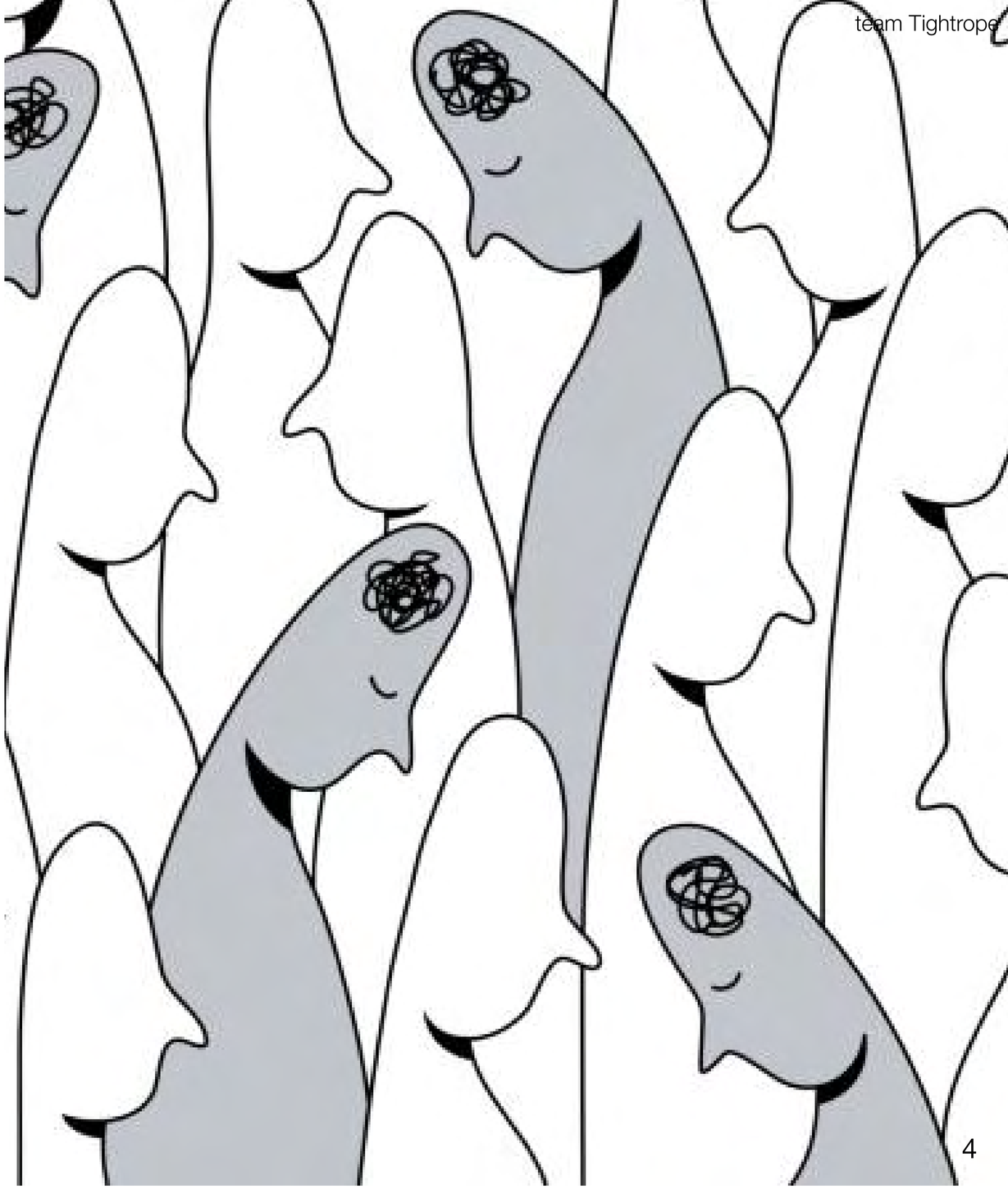
As more golden orbs are collected, the **clouds in the background change from dark to light**, representing **healing** and a change in mindset for Player 1.

Inspiration

Our idea draws inspiration from a team member's **personal experience with anxiety**. She previously used the **metaphor of being on a tightrope** to explain what she was going through to others.

Beyond personal experience, **anxiety disorders affects a great number of people**. Globally, an estimated 310 million are affected by anxiety disorders and 280 million individuals are affected by depression (World Health Organization).

Within the US, an estimated 31% and 8% of adults have experienced some form of anxiety disorder and depressive episode at some point in their lives, respectively (NIMH).



Sources:
https://www.who.int/news-room/fact-sheets/detail/depression#:~:text=Approximately%20280%20million%20people%20in,due%20to%20suicide%20every%20year.
https://www.who.int/news-room/fact-sheets/detail/anxiety-disorders#:~:text=An%20estimated%204%25%20of%20the,all%20mental%20disorders%20(1).
https://www.nimh.nih.gov/health/statistics/major-depression
https://www.nimh.nih.gov/health/statistics/any-anxiety-disorder
https://www.istockphoto.com/vector/crowd-of-people-with-depression-hand-drawn-outline-vector-gm1770096283-545736025?phrase=drawing+of+lonely+in+a+crowd

Player 1 Gameplay

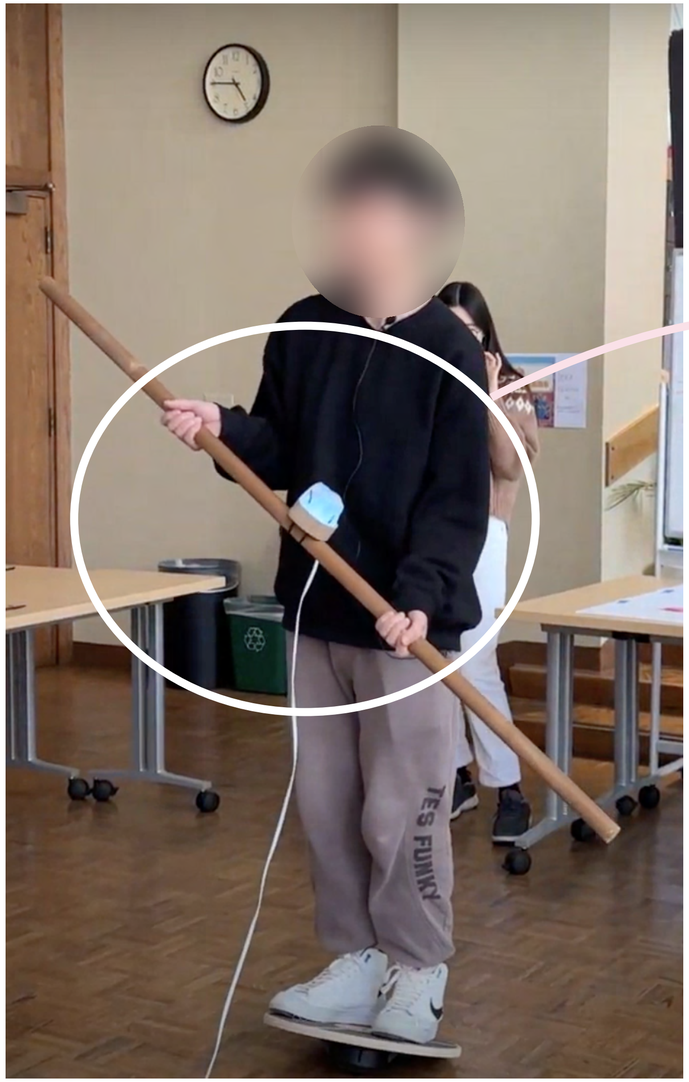
Player 1 (P1)

Input: (1) CPX-mounted rod that can be tilted to the left and the right; (2) balance board to increase difficulty

Output: Avatar tilting left or right on screen

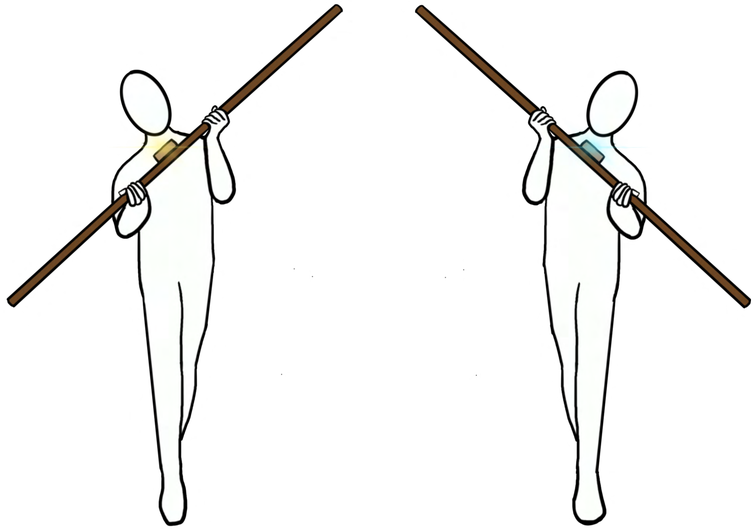
Key takeaways:

1. Physical sensation of instability and unease
2. The mentality shift through darkness to lightness may not always be linear due to things affecting us in life (trying to avoid dark orbs and collect light orbs).



Participant play testing as Player 1.

On-screen tutorial with arrows to indicate movement.



Tutorial sketches at the start of the experience.

Player 2 Gameplay

Players 2 / 3 (P2/3)

Input: CPX-mounted wings

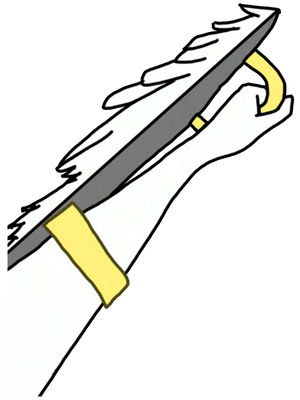
Output: On-screen owls flapping their wings to change the positions of dark and light orbs away and towards Player 1, respectively.

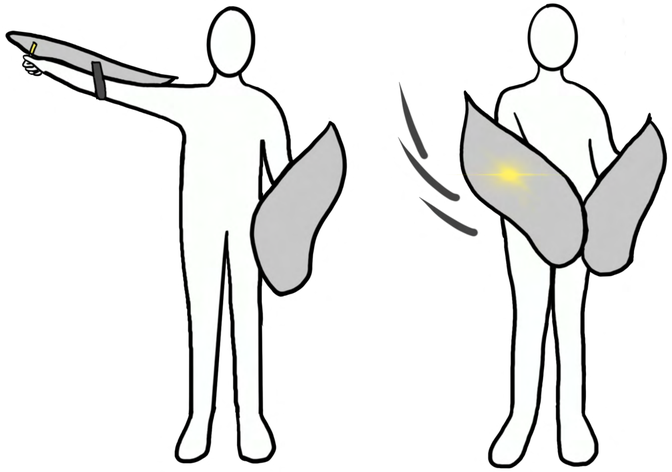
Key takeaway:

1. We can support those around us by bringing them moments of joy and helping them to navigate negative emotions.
2. However, we may not be able to do everything and they would still need to do some navigating on their own.

Participant play testing as Player 2.

On-screen tutorial with arrows to indicate movement.





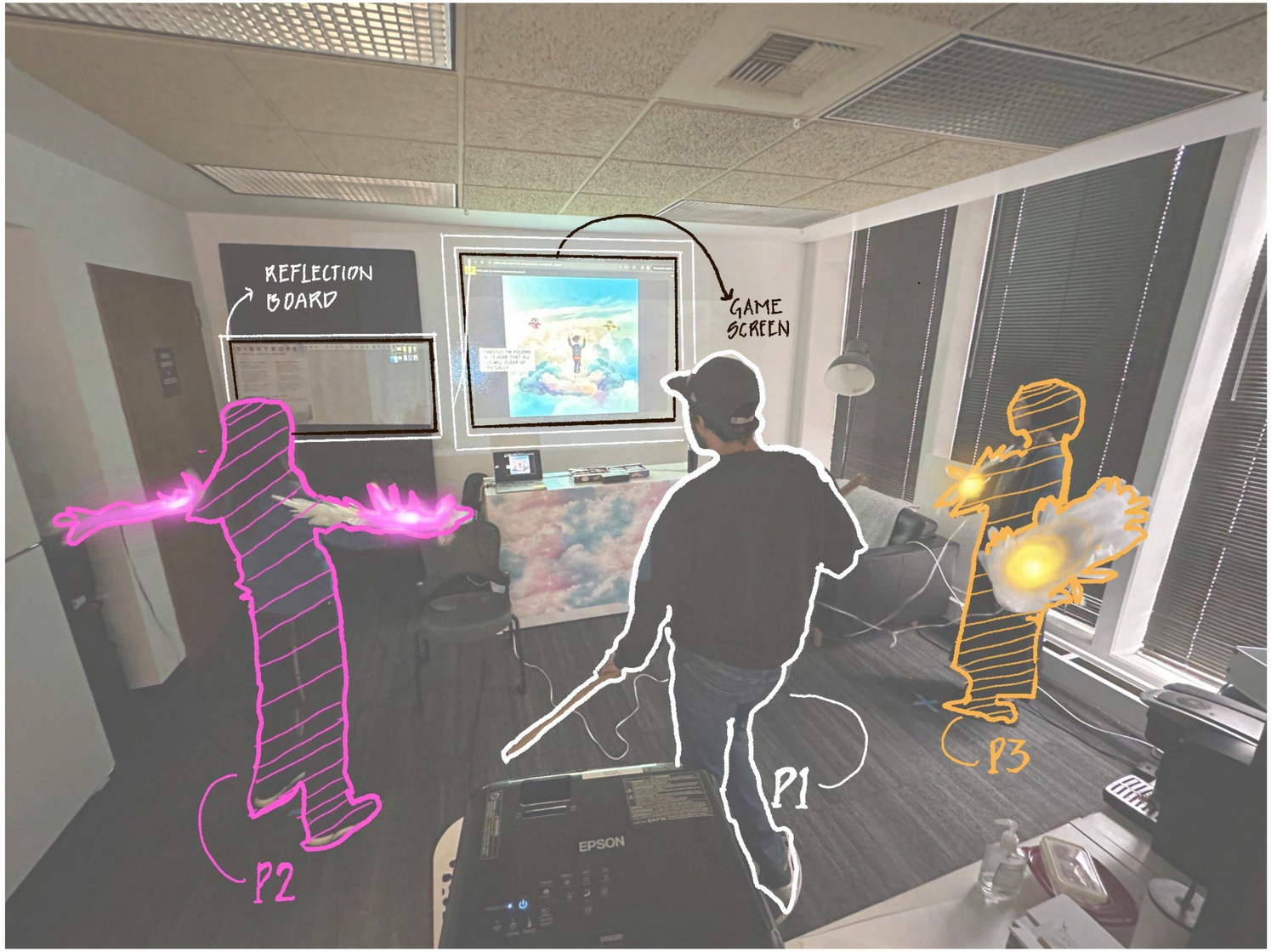
Tutorial sketches at the start of the experience.

Play Area Set-up

The game can be set up in any open space that has a large screen or projector.

The 3 players stand in a line or a slight curve facing the screen, with P1 in the middle, and P2/3 on either side.

A board for jotting down reflections is placed within the play area for players to reflect on their experiences.



Code

Visual elements - P5.js

- The game is built around a game state array that determines if the player is currently in narrative, tutorial or gameplay.
- The game waits for keyboard key entries and does different functionalities determined by the key pressed (e.g. moving a wing controller, tilting the main character)

```
//game states
let game_state_counter = 0;

//Game state options
let GSO = {
  "BEFORE_PLAYING": "before_playing",
  "STORY_1": "story_1",
  "STORY_2": "story_2",
  "STORY_3": "story_3",
  "STORY_4": "story_4",
  "STORY_5": "story_5",
  "EMOTION_GENERATION": "emotion_generation",
  "TUTORIAL_1_IMG_1":  "tutorial_1_img_1",
  "TUTORIAL_1_IMG_2":  "tutorial_1_img_2",
  "TUTORIAL_1_IMG_3":  "tutorial_1_img_3",
  "TUTORIAL_1_IMG_4":  "tutorial_1_img_4",
  "TUTORIAL_1_1_L": "tutorial_1_1_r",
  "TUTORIAL_1_1_R": "tutorial_1_1_l",
  "TUTORIAL_1_2": "tutorial_1_2",
  "TUTORIAL_2_IMG_1":  "tutorial_2_img_1",
  "TUTORIAL_2_IMG_2":  "tutorial_2_img_2",
  "TUTORIAL_2_IMG_3":  "tutorial_2_img_3",
  "TUTORIAL_2_IMG_4":  "tutorial_2_img_4",
  "TUTORIAL_2_1": "tutorial_2_1",
  "TUTORIAL_2_2": "tutorial_2_2",
  "READY": "ready",
  "GAME": "game",
  "ENDING_1": "ending_1",
  "ENDING_2": "ending_2",
  "RESOURCE_LIST": "ending_resources"
}

let game_states_array = Object.values(GSO);
let current_state = game_states_array[game_state_counter];

//hit variables
let good_object_hit = 0;
let bad_object_hit = 0;
let last_hit_time;
```

Game state code in p5.js including story, emotion generation, tutorial, gameplay and more

```
//Functionality to detect movemement from the owls and switch orb locations
function keyPressed() {
  if (game_states_array[game_state_counter] == GSO.GAME || isTutorialPlayer2()) {
    wingFlapped();
  } if (game_states_array[game_state_counter] == GSO.GAME || isTutorialPlayer1()) {
   if (keyCode >= 48 && keyCode <= 57) {
       rotation_val = keyCode - 57;
       rotation_angle = rotation_val * 5;
   }
   if (keyCode >= 73 && keyCode <= 81){
     rotation_val = keyCode - 72;
     rotation_angle = rotation_val * 5;
   }
  }
}

//move wing beats and check if any orbs are nearby
function wingFlapped() {
  for (let i = 0; i< emotion_array.length; i++) {
      emotion = emotion_array[i];
     //Player 2 code
     if (keyCode == 70) {
       owl.startLeftWingBeat();
       if (dist(owl.get_x_position(), owl.get_y_position(), emotion.get_x_position(),
emotion.get_y_position()) < 80) {
         emotion.set_horizontal_move(width/2-70);
         updateWingTutorial(owl, true);
       }
     } else if (keyCode == 72) {
       owl.startRightWingBeat();
       if (dist(owl.get_x_position(), owl.get_y_position(), emotion.get_x_position(),
emotion.get_y_position()) < 80) {
         emotion.set_horizontal_move(width/2-210);
         updateWingTutorial(owl, false);
       }
     }
     //Player 3 code
     else if (keyCode == 65) {
       owl2.startLeftWingBeat();
```

Key pressed code in p5.js with functionality depending on keycode

```
let run_code = false;
let x_value = 0;
let letter_vals = ["i", "j", "k", "l", "m", "n", "o", "p", "q"];
input.buttonA.onEvent(ButtonEvent.Click, function () {
    light.setAll(0x000000)
    run_code = !(run_code)
    pause(1000)
})
input.onSwitchMoved(SwitchDirection.Left, function () {
    run_code = true
})
forever(function () {
    pause(100)
    if (run_code == true) {
        x_value = Math.map(input.acceleration(Dimension.X), -700, 700, -45, 45)
        let state = Math.round(x_value / 5);
        if (state > 9) {
            state = 9;
        } else if (state < - 9) {
            state = -9;
        }
        let output_state = state + 9;
        if (output_state > 9) {
            light.setAll(0X00FF00);
            keyboard.type(letter_vals[output_state - 10]);
        } else {
            light.setAll(0X0000FF);
            keyboard.type(output_state.toString());
        }
    }
})
```

Player 1 code using tilt functionality X direction

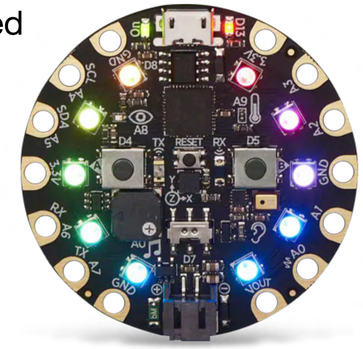
```
input.onGesture(Gesture.TwoG, function () {
    if (cpx_num == 1) {
        light.setAll(0xFF00FF);
        if (left == true) {
            keyboard.type("A");
            light.setBrightness(255);
        } else {
            keyboard.type("D");
            light.setBrightness(255);
        }
        pause(100);
        light.setAll(0x000000);
    } else {
        light.setAll(0xFFFF00);
        if (left == true) {
            keyboard.type("F");
            light.setBrightness(255);
        } else {
            keyboard.type("H");
            light.setBrightness(255);
        }
        pause(100);
        light.setAll(0x000000);
    }
})
input.buttonB.onEvent(ButtonEvent.Click, function () {
    left = false;
    light.setAll(0x00ff00);
    // keyboard.type('A');
    pause(500);
    light.setAll(0x000000);
})
input.buttonA.onEvent(ButtonEvent.Click, function () {
    left = true;
    light.setAll(0xff0000);
    // keyboard.type('A');
    pause(500);
    light.setAll(0x000000);
})
let left = false;
let cpx_num = 0;
```

Player 2/3 code using 2G motion code dependent on player number and wing direction

Controller - CPX / MakeCode JavaScript

- **Player 1:**
  Utilizes tilt acceleration in the X direction, as recorded by the accelerometer. Based on the acquired value, the position is then mapped to one of 19 distinct states. Each state corresponds to a specific key press within the P5 game code.

- **Players 2 / 3:**
  Leverages the 2G shake functionality. Each CPX is assigned to a distinct keyboard button based on whether the player is designated as P2 or P3 and whether the wing is the left or the right.



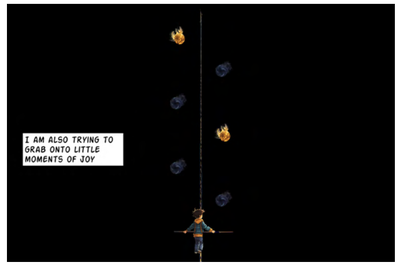
Experience Flow

1. Opening narration

Sets the context for the interactive experience through a skit about two friends catching up over the phone.

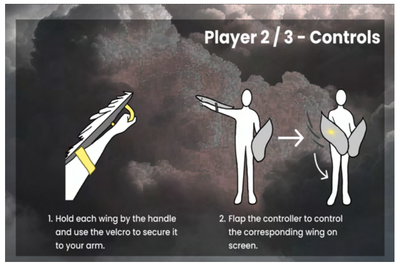
2. Screen orientation

Orients players to the elements on screen, and offers an option for players to record custom audio to personalize the gameplay



3. Gameplay tutorial

Teaches players how to use the controllers.



4. Main game

Players are free to interact with the game however they wish.

5. Closing narration

Concludes the interactive gameplay and completes context setting.

6. Resources & Reflection

Invites all players to reflect on the experience and their personal takeaways through a post-it wall, and provide mental health resources for follow-up.

Key Links

Game (p5.js) [link]

CPX Player 1 [link]
CPX Player 2/3 [link]

Visual Assets [link]

Project Video [link]

Design Process

Our design process mirrored our design prompt

Process

1. Desk research & Ideation
2. Hardware prototyping & code development
3. Narrative & input mechanism iterations
4. Implementation

Prompt

1. Game that addresses a societal challenge
2. Has a custom hardware input controller
3. Is multiplayer
4. Is walk-up-and-playable

Desk Research

Our initial desk research explored a wide range of video games and interactive installations linked to societal issues.

We delved into captivating multiplayer interactions, intriguing input controllers, and experiences that conveyed explicit social messages or employed compelling narrative techniques.

Findings from Desk Research

The importance of *physicality*

Interactive installations sometimes employed techniques that create intense physical sensations to better convey a message.

The **Feeling Pressured** activity at Exploratorium SF conveys the embodied sensation of feeling pressured due to social anxiety, mental health challenges, health challenges, and more. It creates this sensation by allowing people to be squeezed into an extra large garbage bag with the help of a vacuum cleaner.

The use of *metaphors and abstraction*

Games often employed metaphors and abstraction techniques to create moments of interaction within a narrative, especially when touching on sensitive topics such as death.

The game **Spiritfarer** uses anthropomorphized animal characters to tell the stories of the everyday person dealing with loss, fractured relationships, regrets, and acceptance of death. It also employs the commonly used metaphor of a guide ferrying spirits into the afterlife to create interactions with different characters. The art style and abstracted stories make the heavy topic of death and grieving more lighthearted and approachable, while retaining a reflective nature.

The role of *haptic feedback*

Large scale interactive installations use multi-sensory feedback to make full use of its scale and capture the attention of audiences.

The installation **Lightbattle III**, which is a tribute to the cycling culture in the Netherlands, uses 6 static bicycles as input devices to generate light and sound outputs. The large installation features 3 bikes on each side, connected by a canopy of lights. As cyclists on each end pedal, the colored lights from each end “battle” to fill up canopy, creating friendly competition between the cyclists and a connection between the cyclists and the installation.

Ideation Process

Taking inspiration from our desk research findings, we generated a broad range of ideas across societal challenge areas such as prejudice and bias, climate change, industrial water consumption and pollution, and mental health.

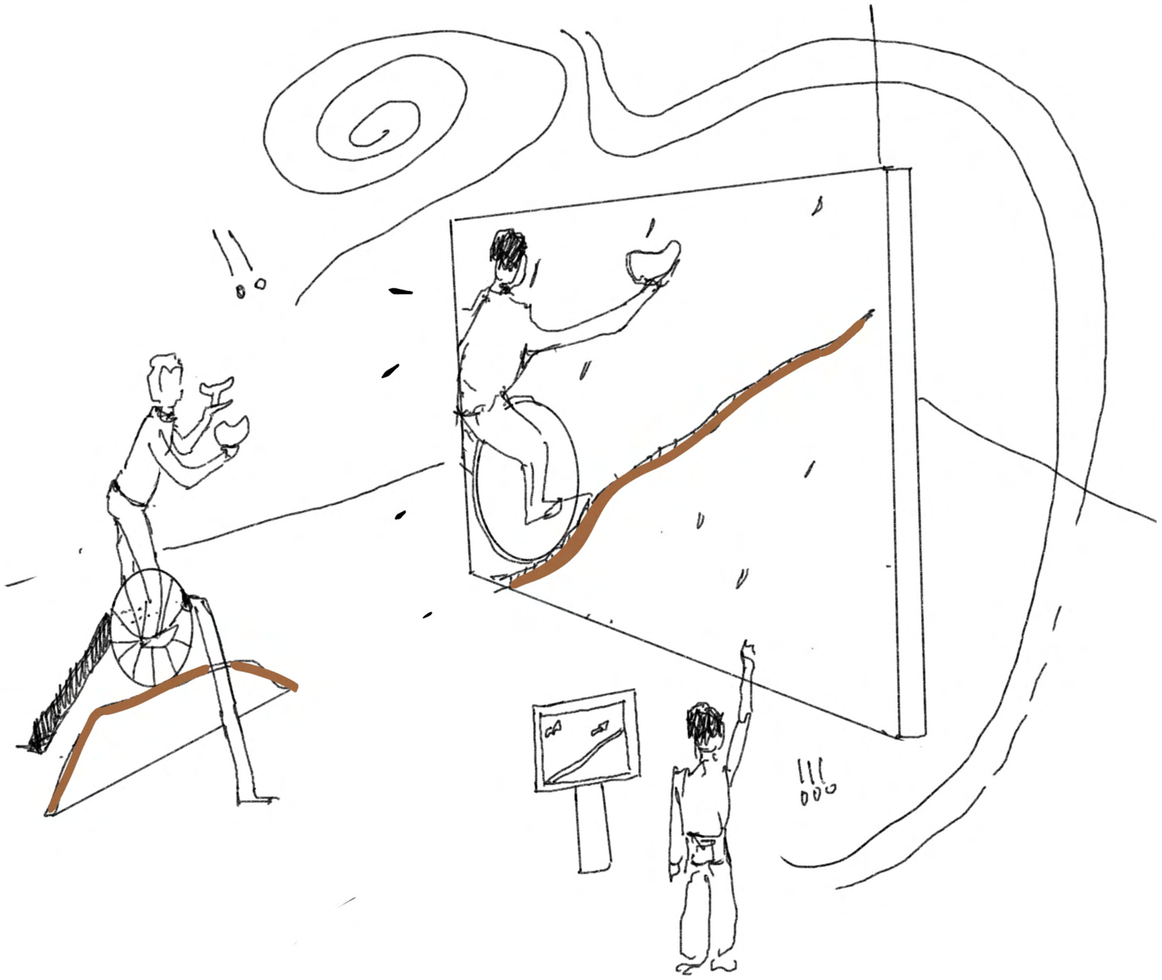
We employed **AI tools** such as **ChatGPT and DALL-E** to support our ideation process - asking ChatGPT for one game suggestion each, and using DALL-E to generate concept art to communicate our vision.

Downselecting to Tightrope

As we were downselecting, we asked ourselves:

What stories can *we* tell?

We decided to proceed with **Tightrope** as it was inspired by a team member's personal experiences with mental health challenges. We imagined Player to be balancing on a tightrope somehow, and another player supporting Player 1.



Initial concept sketch of Tightrope, which imagined a unicycle as part of the input mechanism.

Rapid Prototyping for Custom Controllers

The Tightrope metaphor presented an intriguing input mechanism of using balance to control the game.

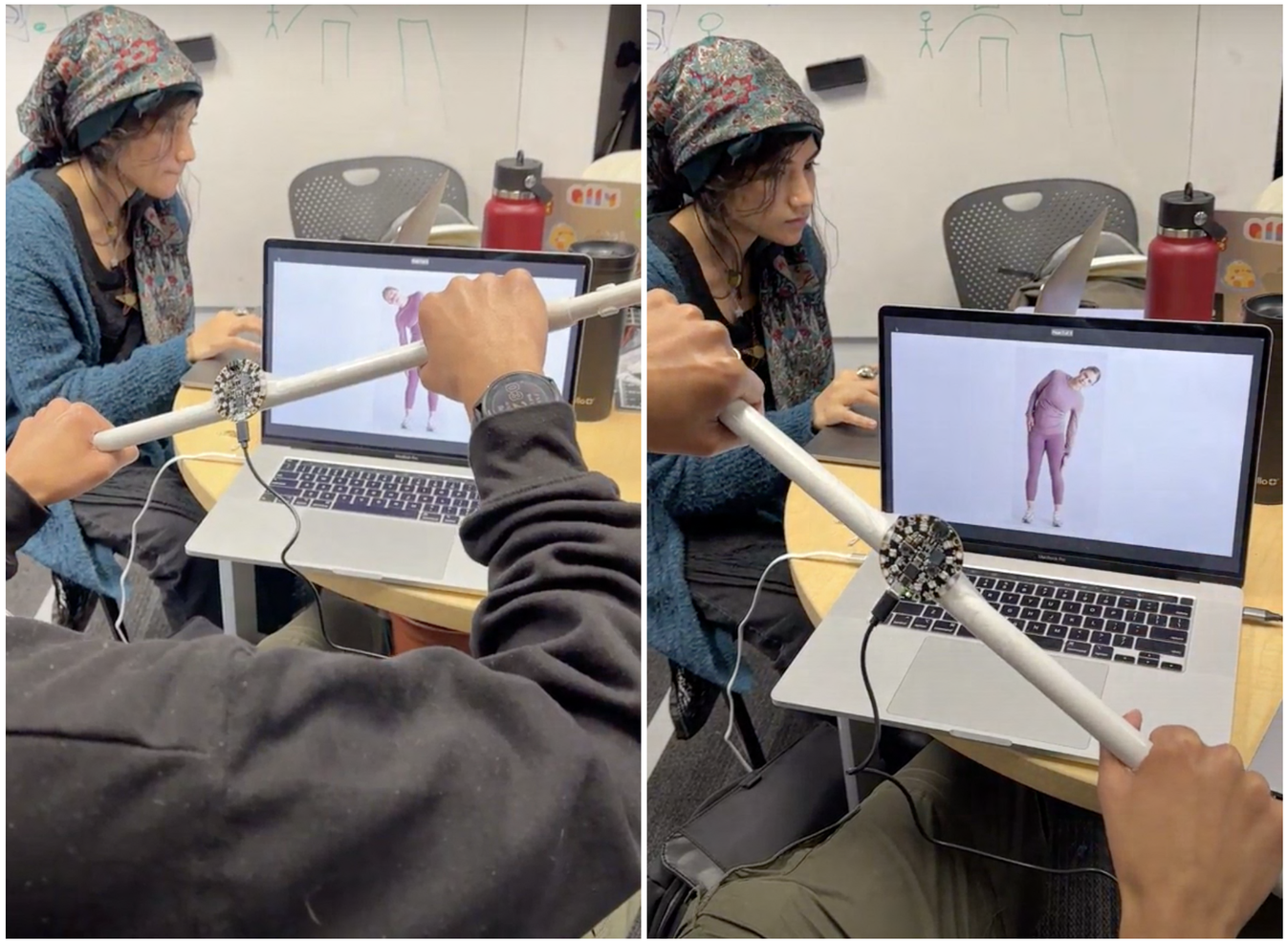
Taking inspiration from the **balance pole used by tightrope walkers**, we rapidly prototyped a **pole controller** with a CPX device at its center.

Rapid prototype of the balance pole controller, with a broom handle, tape, and CPX.

Controller Code Explorations

This prototype facilitated our initial exploration of the project code. As the user tilted the pole left or right, the **X-direction acceleration** values logged by the CPX activated distinct code blocks, triggering arrow key clicks.

We integrated this functionality with a set of slides featuring a woman leaning left or right. **Tilting the rod shifted between slides**, making the woman appear to mirror our movements. This process **confirmed the feasibility and responsiveness of our prototype**, providing the confidence to proceed with development.



Lo-fi balance pole controller proof of concept - tilting the pole left and right activated corresponding key presses that moved through a slide deck showing a woman leaning left or right.

Making it Multiplayer

Given the focus on mental health, our game naturally leaned towards a personal experience - **introducing additional players required careful consideration**.

We decided on having 2 players play as abstracted, friendly, supporting roles for P1 as it best suit the mental health narrative. We used a **Wizard-of-Oz** (WoZ) prototype to evaluate

1. 2 potential mechanisms for P2/3
2. Resonance of Player 1's controls
3. Comprehension of the gameplay narrative and animated intro/outro sequences

Version 1 of P2/P3 input: Coordinated wing flapping between P2 and P3 to propel P1 towards the goal

Version 2 of P2/P3 input: Isolated swatting motion on each hand to fend off obstacles to protect P1

Narrative ↔ Mechanism Interdependence

Our WoZ testing revealed just how **interconnected input mechanisms and narrative are** - participants interpreted the mechanisms and gameplay in different ways, sometimes contradicting the takeaways we were trying to convey. This was a **pivotal moment in our process.** For every iteration after this, we held rigorous discussions to ensure any changes to the mechanisms delivered the narrative impact we wanted them to.

1

Dodging responsibilities?

Participants understood the connection between the P1 controls and the tightrope metaphor for poor mental health. However, the gameplay goal of purely dodging obstacles (represented as bats) was interpreted as avoiding responsibilities instead of avoiding triggers.

Iteration: Changed game goal to collecting joy and dodging fears, and used more abstract visuals (orbs)

2

Challenge vs collaboration:

There was no strong preference between the flapping and swatting mechanisms, but participants liked them for different reasons. They enjoyed the challenge of the swatting mechanism and the collaborative aspect of playing with other participants for the flapping mechanism.

Iteration: Changed P2/3's role to support P1 through a fast-paced flapping mechanism

3

Strengthening narrative

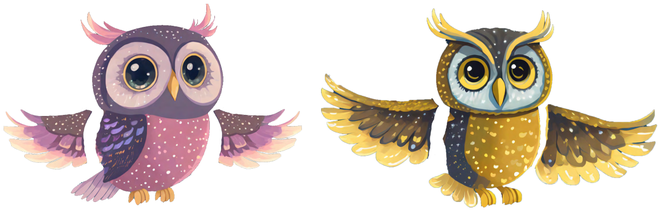
Participants advised bolstering the narrative connection with more dialogue, milestones, and audiovisual feedback for navigating obstacles to clarify outcomes in the game. The gameplay at this fidelity did not convey the message clearly on its own without additional narrative elements.

Iteration: Added audiovisual feedback in-game for obstacles, and voiceovers for intro, outro and in-game dialogue

Graphic Asset Generation using AI

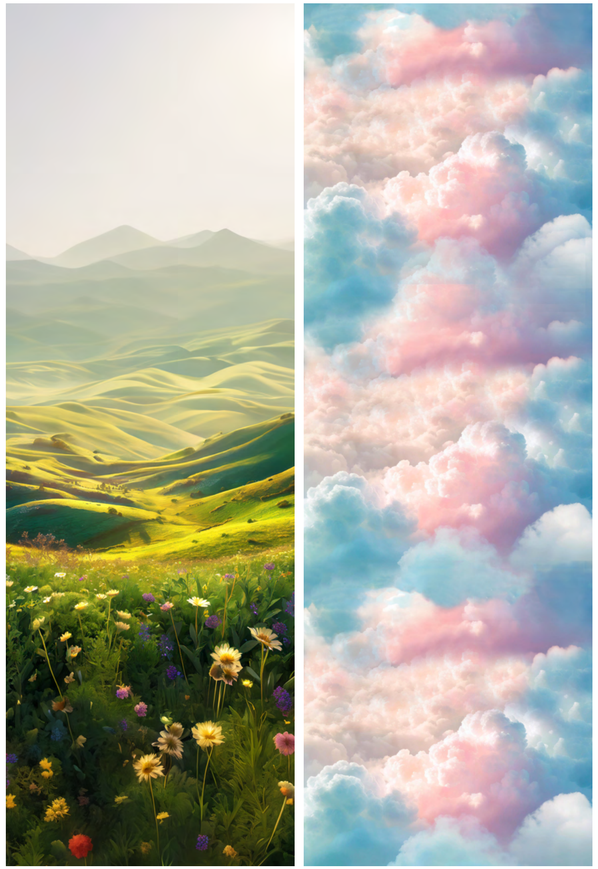
We used generative AI extensively in visual asset generation for our project (e.g. DALL-E, generative fill)

While it helped us to experiment with different tones, art styles, and story settings, we opted to do animation in Figma and p5.js instead as the functionality of generative AI for video was much more limited.



Owl graphics were entirely generated in DALL-E. Wings were cropped out in photoshop and made into individual assets for animation.

Orbs and tightrope walker graphics were composite images made in Photoshop using a combination of several DALL-E outputs.

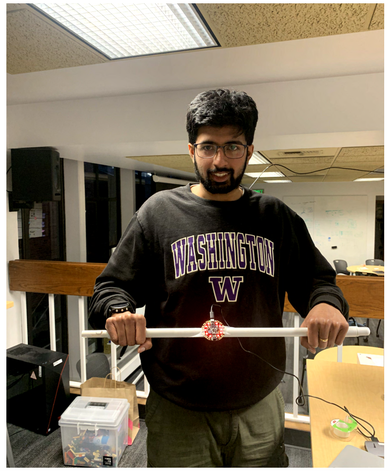


For background images, we used DALL-E outputs as a base, and used generative fill to create a scrollable, long background image.

Evolution of P1 Input Controller

With each round of testing we refined the **fidelity, scale, and complexity** of the P1 rod controller.

Rapid Prototype



Our initial testing used a small broom handle with the CPX board taped directly on. It helped us to demonstrate feasibility, but was fragile, flimsy, and lacked the scale we wanted.

WoZ Prototype

Our WoZ iteration used a solid wooden rod, improving upon the scale. We also added a balance board that P1 stands on to better convey the unsteadiness of balance. However, the CPX enclosure was low fidelity and unstable.

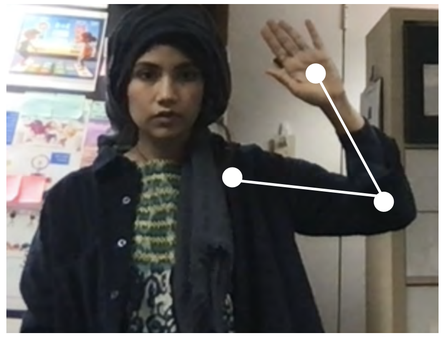
Final Iteration

Our final version utilized the same rod form and balance board, but also included a 3D printed CPX enclosure, painted white to match the styling of our other controllers.

Evolution of P2/3 Input Controller

We tested more variations of the P2/P3 input controllers in comparison, strengthening the tie to the game's narrative and message with each round of prototyping and testing.

Initial Prototyping

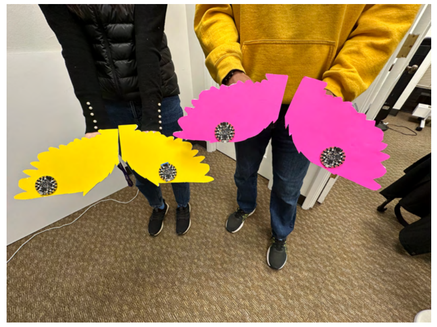


Our first prototype explored computer vision. However, the input was inconsistent and did not function well with multiple players on screen. This was a problem, due to multiplayer requirement in the prompt.

WOZ prototyping

In our WoZ prototype, we used cardboard to create lo-fi physical wing controllers. These wings were attached to users' forearms using tied rope. However, players found the wings to be unstable.

MVP Prototype



In our MVP game, we made color-coded wings using foam core, with Velcro attachments. However, these were still prone to slipping and were were still at a much smaller scale compared to the P1 controller.

Final Iteration

Our final controller designs featured larger wings crafted from a mix of foam core, paper maché, and sewn feathers and cords. They were equipped with a handle for better control and grip, and Velcro attachments for added security.

Evolution of code

Building on the established tilt mechanism functionality, we made further refinements to the game code.

p5.js

To enhance the game code, we integrated a hit box functionality for P1, incorporated the wing flap mechanism to move orbs for P2/P3, adapted the code for both positive and negative obstacles, defined game states and introduced narrative elements like conversation audio and text boxes.

```
if (game_state_counter >= 2 && game_state_counter < 6) {
  push();
  imageMode(CENTER);
  image(rope, width/2, height/2, 15, height+200)
  pop();

  textSize(20);
  fill(255);
  text('Good orbs hit: ' + good_object_hit, 10, 30);
  text('Bad orbs hit: ' + bad_object_hit, 10, 60);
  if (game_state_counter >=4) {
  owl.display();
  owl2.display();
  owl.moveLeftWing();
  owl.moveRightWing();
  owl2.moveLeftWing();
  owl2.moveRightWing();

  }

  for (let i = 0; i< emotion_array.length; i++) {
    if (emotion_array[i].y_position > height) {
      emotion_array.splice(i, 1);
    } else {
      let emotion = emotion_array[i];
      emotion.move();
      emotion.display();
      boyCollision(emotion, i);
      //owlMoveOrbs(emotion);
    }
  }
```

p5.js new code for game state functionality, showing the owl characters and more in the draw function

```
//Check if the boy collides with an orb
function boyCollision(emotion, index) {
  //Get the boys current position
  let boy_position = getboyPosition();
  //Check if the boy colides with any of the orbs
  if (dist(boy_position.x, boy_position.y, emotion.get_x_position(), emotion.get_y_position()) < 100) {

    //If the orb is a positive emotion
    if (emotion.get_good_value() == true) {

      //Increase hit marker for positive and decrease opacity of background
      last_hit_negative = false;
      last_hit_time = millis();
      good_object_hit += 1;
      opacity_value -= 20;
      blur_counter -= 1;
      if (blur_counter < 0) {
        blur_counter = 0;
      }
    }
    //If the orb is a negative emtion
    else {

      // add hit marker
      last_hit_negative = true;
      last_hit_time = millis();
      opacity_value += 20;
      blur_counter += 1;
      bad_object_hit += 1;
      if (opacity_value > 255) {
        opacity_value = 255;
      }
```

New p5.js code built to show collision code, utilizing the distance functionality determined by the positions of the characters and the orbs.

MakeCode

We refined the code from our rapid pole prototype to better match P1 movements. For P2/3, we increased the sensitivity to 2G shake motion. We focused on ensuring seamless communication with p5 visuals using arrow keys.

```
forever(function () {
    if (run_code == true) {
        x_value = Math.map(input.acceleration(Dimension.X), -700, 700, -45, 45)
        console.log(x_value)
        if (x_value > 0) {
            console.log("higher");
            light.setAll(0x00FFFF);
            keyboard.functionKey(KeyboardFunctionKey.RightArrow, KeyboardKeyEvent.Up);
            keyboard.functionKey(KeyboardFunctionKey.LeftArrow, KeyboardKeyEvent.Down);
        } else if (x_value < 0) {
            console.log("lower");
            light.setAll(0xFFFF00);
            keyboard.functionKey(KeyboardFunctionKey.LeftArrow, KeyboardKeyEvent.Up);
            keyboard.functionKey(KeyboardFunctionKey.RightArrow, KeyboardKeyEvent.Down);
        } else {
            console.log("other");
            light.setAll(0xFF00FF);
        }
    } else {
        light.setAll(0x000000);
        keyboard.functionKey(KeyboardFunctionKey.LeftArrow, KeyboardKeyEvent.Up);
        keyboard.functionKey(KeyboardFunctionKey.RightArrow, KeyboardKeyEvent.Up);
        pause(500);
```

Player 1 CPX code, utilizing the x acceleration and holding down the left or right arrow keys determined by the value of the acceleration.

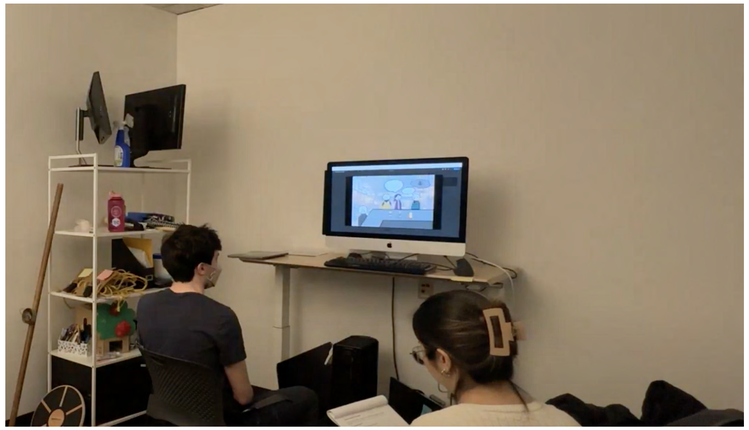
```
nput.onGesture(Gesture.ThreeG, function () {
    if (cpx_num == 1) {
        light.setAll(0xFF00FF);
        if (left == true) {
            keyboard.type("A");
        } else {
            keyboard.type("D");
        }
        pause(100);
        light.setAll(0x000000);
    } else {
        light.setAll(0xFFFF00);
        if (left == true) {
            keyboard.type("F");
        } else {
            keyboard.type("H");
        }
        pause(100);
        light.setAll(0x000000);
    }
)

nput.buttonA.onEvent(ButtonEvent.Click, function () {
    left = true;
    light.setAll(0xff0000);
```

Player 2 CPX code utilizes the 3G gesture code, and clicks different arrow key determined by the cpx_num (player number) and the left or right boolean value.

Making it walk-up-and-playable

With our concept, the role of storytelling was vital to connecting the gameplay to our extended metaphor. We tested different ways of onboarding the player throughout each iteration of the project.



We first created a slideshow video depicting a character navigating life and reflecting on their mental health. This format fell short in connecting the narrative with the gameplay due to the different visual styes and heavy use of text.

Next, we integrated narrative elements into gameplay tutorial to explicitly link narrative to gameplay elements. However, participants had difficulty focusing on the narrative while learning to play the game.

Finally, we broke down onboarding into a condensed phone call skit that orients players to the game screen and context, before jumping into controller tutorials. We also added an option record custom audio to further personalize the experience.

Clarifying takeaways and providing closure

We likewise refined the game conclusion to make the intended takeaway as clear as possible, and made sure to provide closure for players after the vulnerable gameplay.

In the first version of the offboarding sequence, we extended the pre-game narrative, with the main character sharing their feelings with friends. Similar to our findings from the first opening sequence iteration, the disjointed art style and use of text compromised the final narrative's effectiveness.

In our full game iteration, we concluded the experience with the phone call skit ending. While this version improved comprehension of the narrative, players felt that they were not able to give the emotional narrative the attention it deserved due to the competing game tutorial and narration at the start of the experience.

In the final prototype, the offboarding retained the subtitles and voice narration over a dark screen. It was followed by a mental health resources page, and a discussion board for players to share and explore their thoughts.

Team Reflections

On using AI strategically

In asset generation

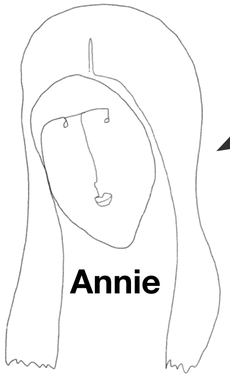
While DALL-E was useful and saved us time in generating static assets, we quickly recognized how limited Runway and other generative AI tools for video were. We relied on other tools (Figma, p5.js) for animating our assets, and also employed generative fill on Photoshop and Firefly.

In coding

ChatGPT was useful in establishing a skeleton code that non-coders in our team could quickly learn to customize. However, as our game got more complex, its value diminished and we leaned heavily on our programmer for development. The more coding context the AI needed, the less valuable the output became.

The need to timebox

It was easy to get trapped in a constant loop of trying to refine prompts, hoping the next output would be “the one”. We ended up needing to timebox ourselves to 15min or 20min of toying with AI tools, before seeking alternative, sometimes manual, solutions.



Annie

AI is such an essential part of our lives now and actively using AI in this class helped me to practice this skill set.

Sam

I went through such a roller coaster of thinking that AI was all-powerful, then thinking that it was so limited, to realizing its usefulness when applied strategically.

Varun

For me with code, AI’s involvement was inconsistent, ranging from fine to completely unusable.

Husna

I am a purist and was repulsed by the idea of using AI for the project. As the project progressed, I figured there was a way to use AI to guide and aid in design and retain my agency at the same time.

On the topic of mental health

Approaching it sensitively

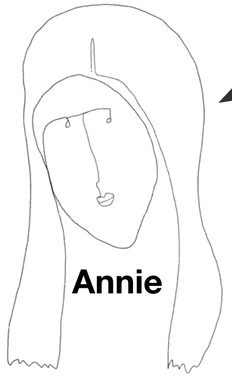
How should we make a game about mental health fun without making fun of the experience? We ended up spending a lot of time talking about game concept and narrative, and made a tradeoff between fun, fast-paced gameplay and storytelling.

The importance of feedback

Testing with people outside of our team proved critical. Because the game had to be abstracted metaphorically, testing helped us to understand what players took away from the game, and pushed us to ensure that our narrative and gameplay were tightly intertwined.

Personal vs relatable

Because the inspiration for the game was so deeply personal, it was challenging to find a middle ground between making it relatable to the general public and making it a specific immersive experience.



Mental health is an interesting topic; however, incorporating it with care and sensitivity into a game proved challenging.

Annie

Working on something so personal was challenging - how can I tell this story authentically, without taking over it completely and holding space for others' experiences?

Sam

Balancing personalizing the experience and providing a detailed experience was hard!

Varun

With discussions, guidance and input through feedback sessions and testing, we could develop a smartly placed game, making sure the narrative and gameplay remained related and intertwined.

Husna

On prototype development

Complexity of code

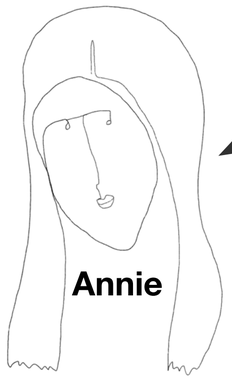
The limited functionality of p5.js meant we had to be creative with how we wanted to execute certain functions. However, the pace of the project and the complexity of our gameplay made it hard for non-coders on the team to get more involved.

Scope creep

As testing and development continued, we found ourselves adding new elements and increasing the scope of our game for more nuanced storytelling. It's a constant work in progress, but we are happy to present what we have achieved so far.

Sharp knife of feasability

We were starkly reminded of the gulf between what we can imagine and what is feasible, working within the bounds of computational power of our laptops, p5.js functionality, available hardware and time.



Annie

Would have been good to incorporate more diverse visuals to fill up our game screen, for example using Figma to create game screens.

Sam

Realized I'm more scrappy - had fun with low-fi prototyping, but I didn't have a strong eye for hi-fi fabrication!

Varun

From a coding standpoint, the complexity of the game was really interesting. Perhaps with other coding platforms it would be easier, but the logic that p5 utilizes can sometimes cause additional complexity, which was annoying but fun!

Husna

It was a smooth transition from the lo-fi to hi-fi prototypes as we had strong ideas from our initial tests.

On what we would do differently

More complex hardware and software

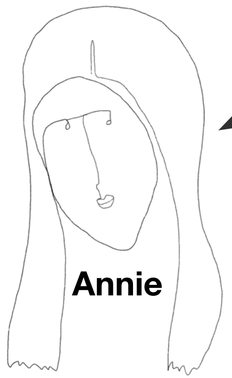
We would have liked to experimented more with Arduino and Unity for more complex input and gameplay, and increased the fidelity of our balance rod controller.

More time spent researching game design and play testing

We would have liked to done more research into game design to improve the immersion of our game, especially for the Player 2 ad 3 roles, and done more testing with real players to understand their takeaways.

More research into mental health

We would have liked to work with a mental health professional to better understand the strategies used by them and their clients to navigate anxiety.



It would have been fun to incoporate the balace board as part of our input mechanism!

Annie

Would've love to do more research into game design - we were focused on development and prototyping but didn't really have time to learn how to build a good **game**

Sam

I would like to have more time for testing, especially given how human centered our project was.

Varun

I would have liked to use Arduino just to get a better hang of it.

Husna