Personal Information

- **Name:** Chaebean Yang
- **ID:** 20230412
- **Email:** kazed0102@kaist.ac.kr

Notice

Due to file size limitations, only the source code and README.md are included in the submitted zip file. Please pull the full repository from the link below to access all project files.

To install dependencies, navigate to the project root directory and run:

http://git.prototyping.id/20230412/individual_project.git

```
cd svelteP5play
npm install
npm run dev
```

If you do not have Markdown Preview Enhanced installed, the Mermaid diagrams may not be visible in the preview. In that case, please refer to the `README.pdf` file below.

If you do not have Markdown Preview Enhanced installed, the Mermaid diagrams may not be visible in the preview. In that case, please refer to the `README.pdf` file below.

Git Information

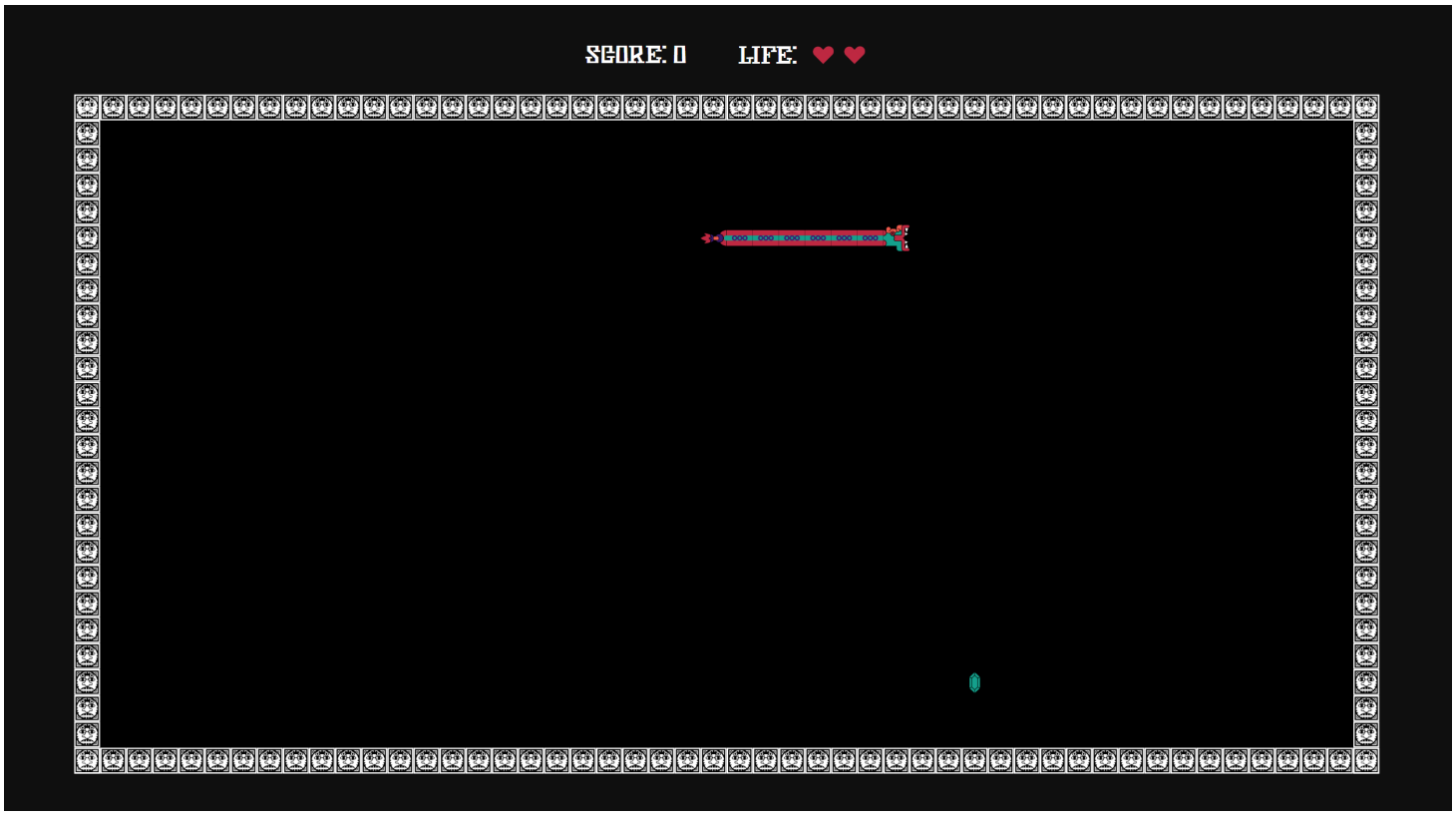
- **Repository URL:** http://git.prototyping.id/20230412/individual_project.git

Demo Video

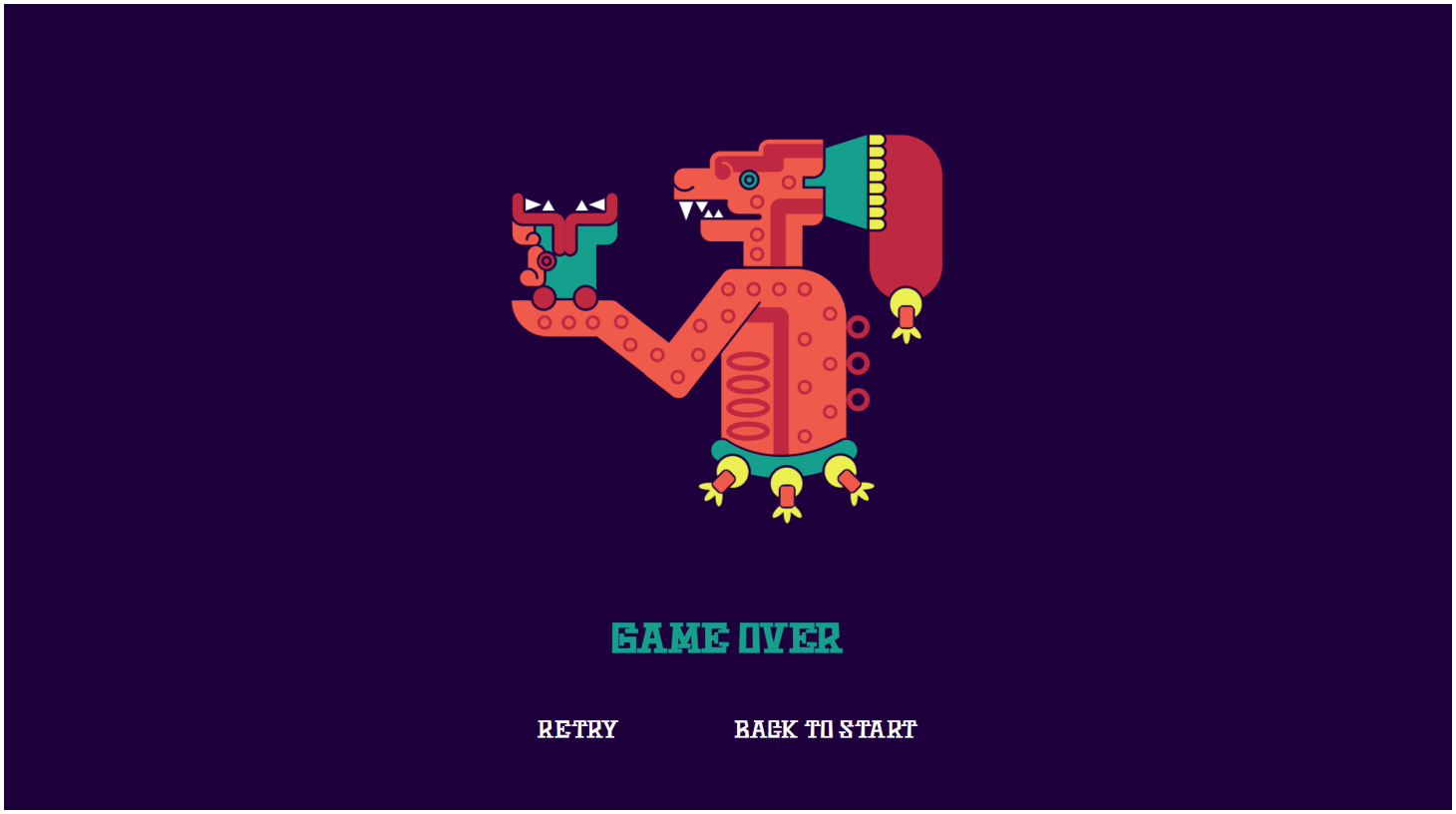
- **Video URL:** Game Demo Video

Game Screens

Start Screen



Game Play Screen



Gameover Screen

Game Clear Screen

Game Description

Basic Information

This game is based on the classic Snake game.
To make the gameplay more engaging, we incorporated elements of Aztec mythology—specifically the story of Quetzalcoatl, the serpent-shaped god who becomes the fifth sun—through both graphics and narrative.
Additionally, various items and more complex mechanics were introduced to increase difficulty and enhance player experience.

How It Works

- The user can control both the head and the tail.
- Body length increases differently depending on the item collected.
- Items provide different scores.
- Speed increases as the score rises.
- Obstacles appear randomly in the current direction of movement.

Head Control

The head is controlled using the arrow keys.

Key	Effect
↑	The head faces upward
↓	The head faces downward
←	The head faces left
→	The head faces right

Tail Control

The tail is controlled using the W, A, S, and D keys.

Key	Effect
W	The tail faces upward
S	The tail faces downward
A	The tail faces left
D	The tail faces right

Items

Item	Score	Body Length Increase
Sun	+15	+3
Gemstone	+10	+2
Heart	+5	+1

Speed

Score Range	Speed Condition
< 70	`p.frameCount % 7 === 0`
< 90	`p.frameCount % 5 === 0`
≥ 90	`p.frameCount % 3 === 0`

Obstacles

Obstacles appear in the direction the snake is currently moving.
If the snake's head or tail collides with an obstacle, the player loses one life.

Game Clear Condition

- Score reaches 100 or more.

Game Over Conditions

- Life reaches 0.
- Collision with walls.

Code Structure

The code is organized into two main parts: the Svelte frontend for UI/state management and the `sketch.js` file for game logic and rendering using p5.js.

Components

1. `Game.svelte`

- Manages the game's state (`start` , `play` , `clear` , `over`)
- Contains the main UI layout, life/score display, and `<div id="game-canvas">` for the p5 sketch

- Handles `onMount()` logic to attach and remove the p5 instance dynamically

```
<script>
  onMount(() => {
      const preventArrowScroll = (e) => {
        const keys = ["ArrowUp", "ArrowDown", "ArrowLeft", "ArrowRight",
        "w", "a", "s", "d", "W", "A", "S", "D"];
        if (keys.includes(e.key)) {
          e.preventDefault();
        }
      };
      window.addEventListener("keydown", preventArrowScroll, {passive: false});

      return () => {
        window.removeEventListener("keydown", preventArrowScroll);
      };
    });
</script>
```

- Receives game data via `receiveGameData()` function and updates HUD

```
function receiveGameData(data) {
    life = data.life;
    score = data.score;
    if (data.state && data.state !== gameState && data.state !== "play") {
      gameState = data.state;
    }
  }
```

2. `sketch.js`

- Contains the `createSketch(p, onGameData)` function that defines the p5 sketch
- Handles all rendering logic: drawing the snake, items, obstacles, and walls
- Manages key logic: movement, collision, growth, scoring, speed change
- Sends current score/life/game state to Svelte using the `onGameData()` callback
- The snake uses an array to manage the head, body, tail, and corner images according to the situation, and to allow the body to grow in length.

Key Features in `sketch.js`

1. Grid System & Basic Setup

- The canvas is divided into cells of size 20 using `cellSize`.
- All coordinates are based on `{x, y}` positions in this grid system.

```
let cellSize = 20;
p.createCanvas(1000, 520);
p.imageMode(p.CENTER);
```

2. Snake as an Array

- The snake is stored as an array of coordinates.
- Movement is implemented using `unshift()` to add a new head and `pop()` to remove the tail.
- When direction changes, a corner segment is drawn automatically.

```
let snake = [
  {x: 10, y: 5},
  {x: 9, y: 5},
  ...
];

snake.unshift(newHead);
if (!growNext) snake.pop();
```

```
for (let i = 0; i < snake.length; i++) {
  const seg = snake[i];
  const px = seg.x * cellSize + cellSize/2;
  const py = seg.y * cellSize + cellSize/2;

  p.push();
  p.translate(px, py);

  if (i == 0) {
   // head dorection setting
  }
  else if (i == snake.length - 1) {
    // tail direction setting
  }
  else {
    //body parts and corner case
    const prev = snake[i - 1];
    const next = snake[i + 1];
    const dx1 = seg.x - prev.x;
    const dy1 = seg.y - prev.y;
    const dx2 = next.x - seg.x;
    const dy2 = next.y - seg.y;
    const isCorner = dx1 !== dx2 || dy1 !== dy2;

    if (isCorner) {
      let angle = ...; // rotation angle logic
      p.rotate(angle);
      p.image(cornerImg, 0, 0, cellSize, cellSize);
    } else {
      const angle = dx1 == 0 ? p.HALF_PI : 0;
      p.rotate(angle);
      p.image(bodyImg, 0, 0, cellSize, cellSize);
    }
  }

  p.pop();
}
```

- Segment rendering (head/body/corner/tail) is dynamically determined:

```
const isCorner = dx1 !== dx2 || dy1 !== dy2;
if (isCorner) p.image(cornerImg, 0, 0, cellSize, cellSize);
```

3. Obstacle Collision Logic

- Obstacles are randomly placed ahead of the snake's direction with spacing and jitter.
- Snake colliding with an obstacle (head or tail) reduces life.

```
if ((head.x == o.x && head.y == o.y) || tail.x == o.x && tail.y == o.y) {
  if (o.hit == null) {
    life -= 1;
    o.hit = p.frameCount;
  }
}
```

```
function makeObstacle() {
    // moving head or tail? which direction?
    let base;
    let dir;

    if (dirHead.x !==0 || dirHead.y !== 0) {
      base = snake[0];
      dir = dirHead;
    }
    else if (dirTail.x !== 0 || dirTail.y !== 0){
      base = snake[snake.length - 1];
      dir = dirTail;
    }
    else {
      return;
    }
    ...;
    //gap between the snake and an obstacle
    const gap = p.floor(p.random(3, 7));
    let newX = base.x + dir.x * gap + p.floor(p.random(-1, 2));
    let newY = base.y + dir.y * gap + p.floor(p.random(-1, 2));
    ...;
  }
```

- Obstacles flash and disappear after impact.

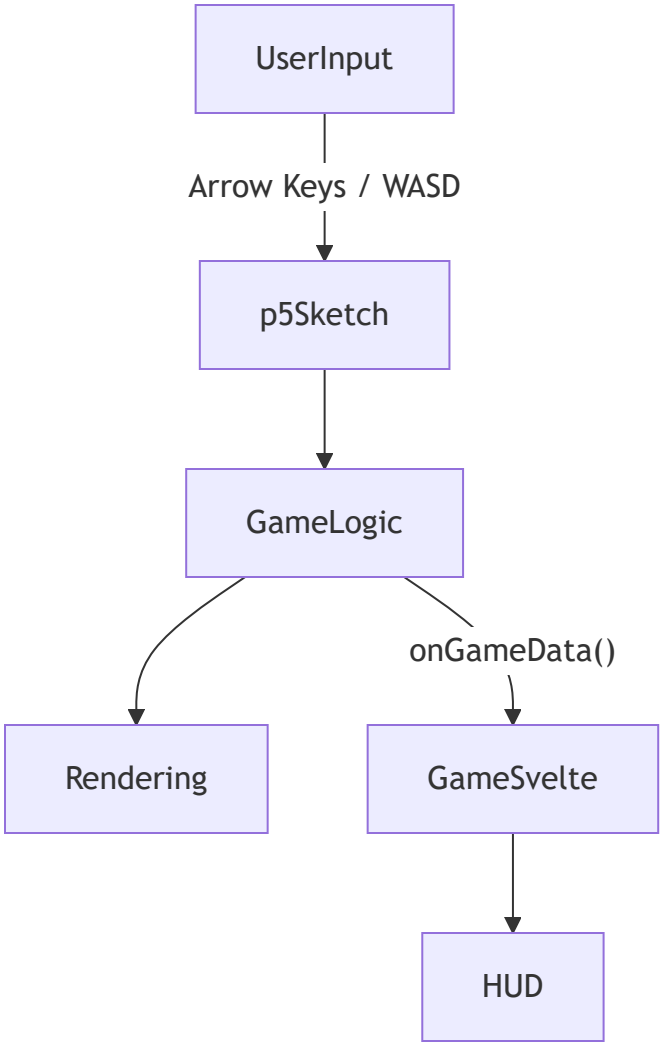
4. Item Collection System

- Items (`heart` , `gemstone` , `sun`) are generated at random, non-overlapping grid locations.
- Each item provides different score and body growth effects.

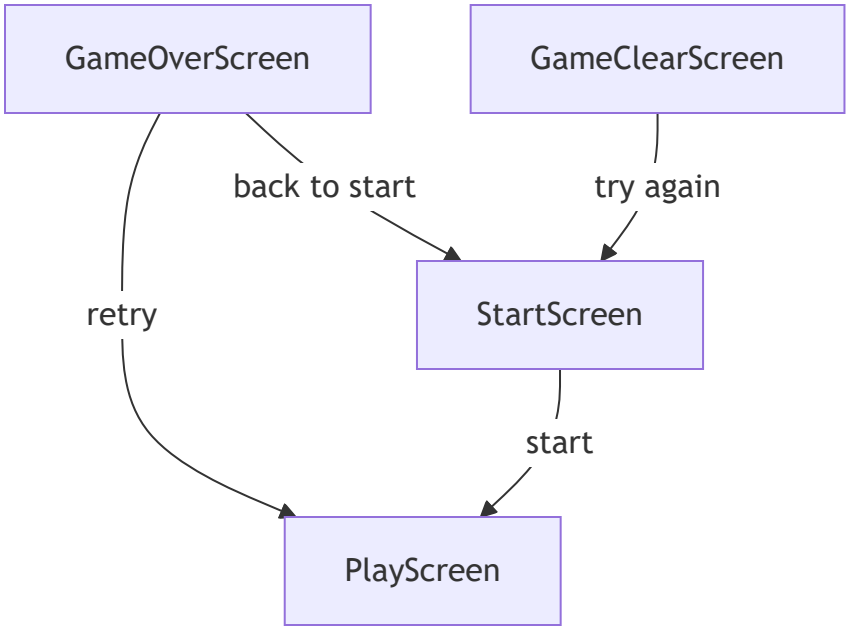
```
if (item.type == "gemstone") {
  score += 10;
  growNext += 2;
}
```

- Items are rendered with different images and refreshed automatically.

Interaction Diagram



Flow Chart



Challenges

- I attempted to include background music based on the game state, but it could not be implemented due to browser restrictions.
- When rendering the corner segments in the snake array, the rotation did not work as expected using theoretically calculated angles; I had to adjust the values empirically for it to display correctly.

References

- Mermaid.js documentation
- p5.js reference
- Svelte official tutorial
- MDN: Styling the content
- Svelte: Reactive statements
- Svelte: If-else statement
- MDN: `Array.unshift()` method
- StackOverflow: Prevent arrow keys from scrolling in Svelte
- Parts of this README were grammar-checked using ChatGPT and Google translator.

Future Imporvements

If I had more time, I would like to explore the following features:

- Adding background music to enhance the atmosphere of the game.
- Changing the appearance of the snake dynamically based on the score.
- Making the canvas layout responsive, so that it automatically adjusts to different screen sizes. *(Responsive design)*