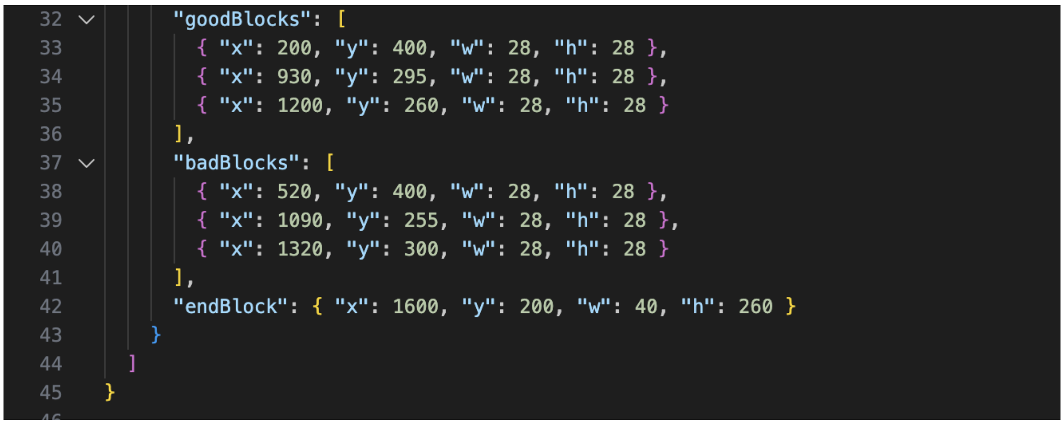
Process & Decision Documentation

Significant decision or change:
Add “good” and “bad” blocks in green and red colors as the game mechanism that leads to the end result, win or lose.

Why did you make it:
To add visualized feedback and results to make the player more engaged in the game.

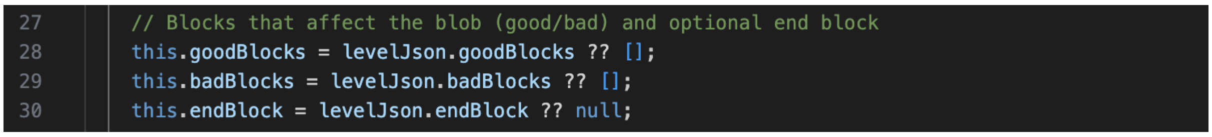
What effect did it have on the work:
When the blob hit different-colored blocks, it changed the same color as the block. And the blob color when it passes the end block determines the ending in win or lose.

Screenshots:



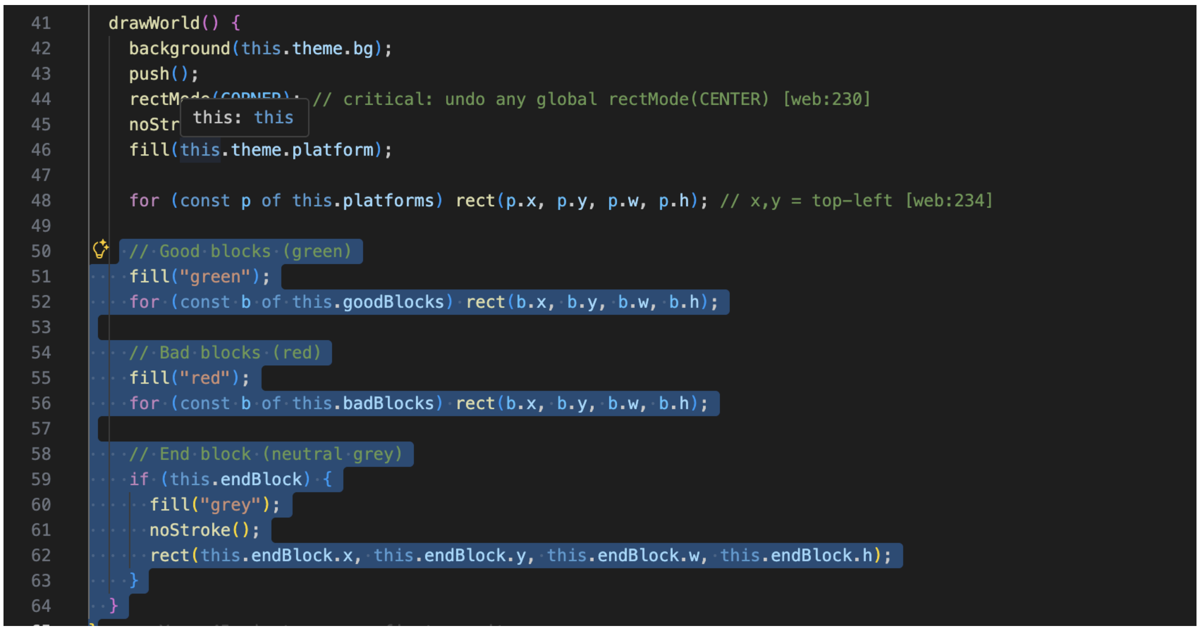
```
      "goodBlocks": [
        { "x": 200, "y": 400, "w": 28, "h": 28 },
        { "x": 930, "y": 295, "w": 28, "h": 28 },
        { "x": 1200, "y": 260, "w": 28, "h": 28 }
      ],
      "badBlocks": [
        { "x": 520, "y": 400, "w": 28, "h": 28 },
        { "x": 1090, "y": 255, "w": 28, "h": 28 },
        { "x": 1320, "y": 300, "w": 28, "h": 28 }
      ],
      "endBlock": { "x": 1600, "y": 200, "w": 40, "h": 260 }
    }
  ]
}
```

Add the “good” blocks, “bad” blocks, and the end block(as a final determine block) in different positions on the page in the levels.json



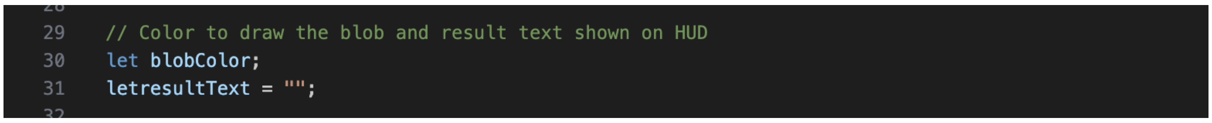
```
    // Blocks that affect the blob (good/bad) and optional end block
    this.goodBlocks = levelJson.goodBlocks ?? [];
    this.badBlocks = levelJson.badBlocks ?? [];
    this.endBlock = levelJson.endBlock ?? null;
```

These screenshot shows the change in WorldLevel.js, are to ensure the blocks are added to the level object. ?? [ ] and ?? null ensure the game doesn’t crash if a level does not include these fields.



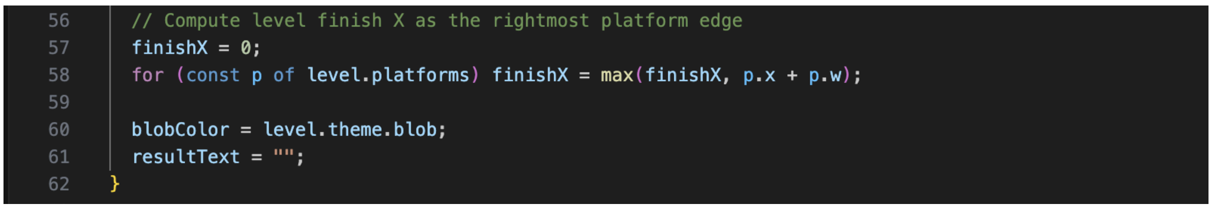
```
drawWorld() {
  background(this.theme.bg);
  push();
  rectM          // critical: undo any global rectMode(CENTER) [web:230]
  noStr
  fill(this.theme.platform);

  for (const p of this.platforms) rect(p.x, p.y, p.w, p.h); // x,y = top-left [web:234]

  // Good blocks (green)
  fill("green");
  for (const b of this.goodBlocks) rect(b.x, b.y, b.w, b.h);

  // Bad blocks (red)
  fill("red");
  for (const b of this.badBlocks) rect(b.x, b.y, b.w, b.h);

  // End block (neutral grey)
  if (this.endBlock) {
    fill("grey");
    noStroke();
    rect(this.endBlock.x, this.endBlock.y, this.endBlock.w, this.endBlock.h);
  }
}
```

This screenshot shows the change in WorldLevel.js. Each block is drawn using the rectangle data from JSON. This step is for the visual of the blocks, it does not affect gameplay logic yet.



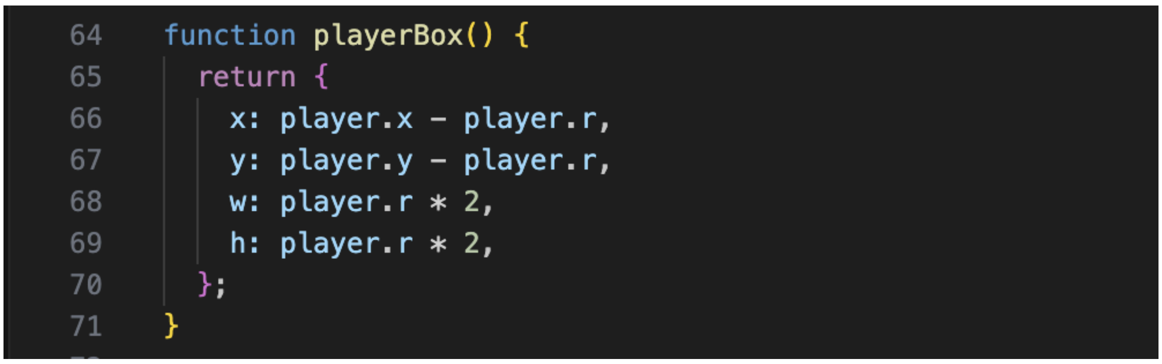
```
// Color to draw the blob and result text shown on HUD
let blobColor;
letresultText = "";
```

This screenshot shows the change in sketch.js. blobColor is used to store the blob's current color state. resultText stores the final result, either "win" or "lose".



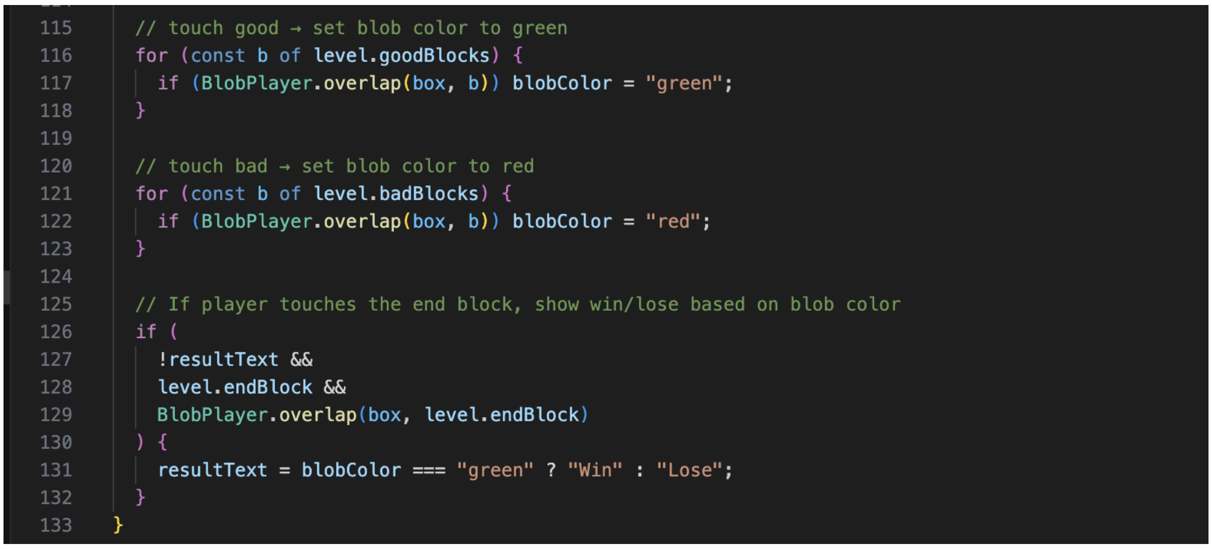
```
  // Compute level finish X as the rightmost platform edge
  finishX = 0;
  for (const p of level.platforms) finishX = max(finishX, p.x + p.w);

  blobColor = level.theme.blob;
  resultText = "";
}
```

This screenshot shows the change in sketch.js. The blob starts with the default color (blue) defined by the level theme. The result text is cleared, so previous runs don't affect new attempts. This ensures each playthrough starts.



```
function playerBox() {
  return {
    x: player.x - player.r,
    y: player.y - player.r,
    w: player.r * 2,
    h: player.r * 2,
  };
}
```

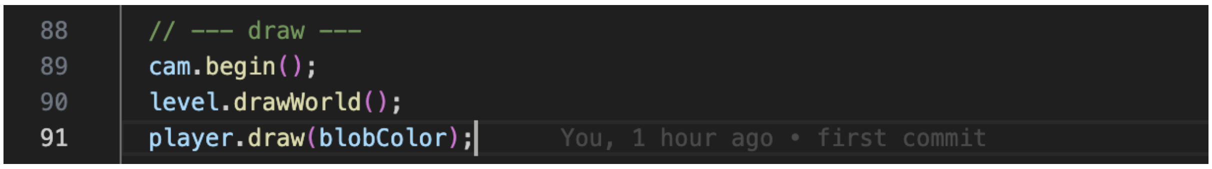
This screenshot shows the change in sketch.js. The function is to estimate if the blob overlaps with the rectangle blocks.



```
  // touch good → set blob color to green
  for (const b of level.goodBlocks) {
    if (BlobPlayer.overlap(box, b)) blobColor = "green";
  }

  // touch bad → set blob color to red
  for (const b of level.badBlocks) {
    if (BlobPlayer.overlap(box, b)) blobColor = "red";
  }

  // If player touches the end block, show win/lose based on blob color
  if (
    !resultText &&
    level.endBlock &&
    BlobPlayer.overlap(box, level.endBlock)
  ) {
    resultText = blobColor === "green" ? "Win" : "Lose";
  }
}
```

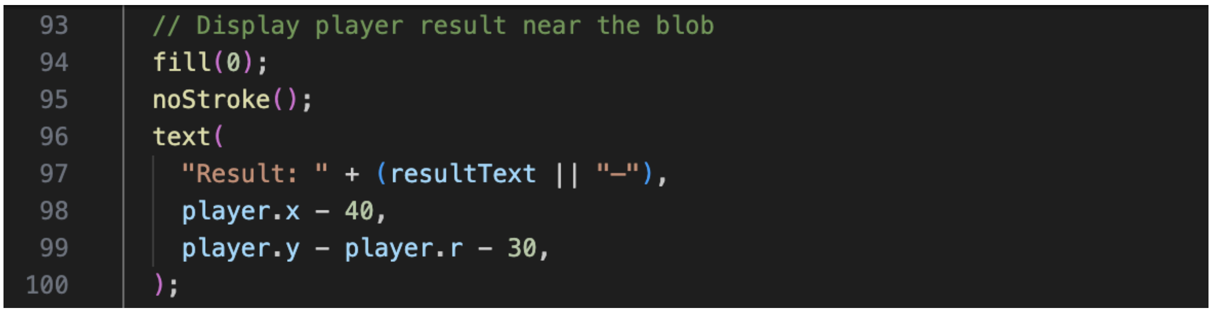
This screenshot shows the change in sketch.js. This code shows the main mechanism of the game, to determine the color change on blob. If the blob hits green blocks, the blob's color changes to green; if the blob hits the red blocks, the color changes to red.

The if function determines whether the blob enters the end block. The blob’s color determines the outcome: Green → win, Red(or blue) → lose



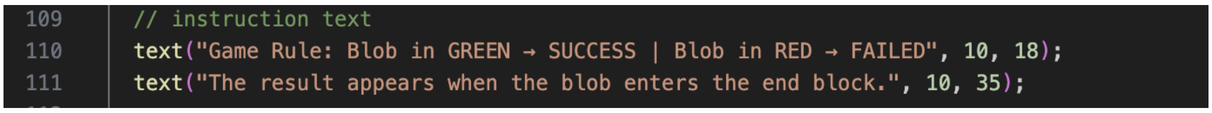
```
  // --- draw ---
  cam.begin();
  level.drawWorld();
  player.draw(blobColor);        You, 1 hour ago • first commit
```

This screenshot shows the change in sketch.js. It ensures the visual state of the blob matches the game logic(changing colors)



```
  // Display player result near the blob
  fill(0);
  noStroke();
  text(
    "Result: " + (resultText || "—"),
    player.x - 40,
    player.y - player.r - 30,
  );
```

This screenshot shows the change in sketch.js. It set up a text message beside the blob and followed the blob when it is moving. The result is linked to the resultText, which is win or lose.



```
// instruction text
text("Game Rule: Blob in GREEN → SUCCESS | Blob in RED → FAILED", 10, 18);
text("The result appears when the blob enters the end block.", 10, 35);
```

This screenshot shows the change in sketch.js. It's the change in the text information at the top of the page to introduce the game rules.

GenAI

No GenAI use in this Side Quest