

# Process & Decision Documentation

## Project/Assignment Decisions

### Side Quests and A4 (Individual Work)

One key decision I made was that I wanted to initially be able to have several combinations of ingredients and a more interactive story that includes an NPC as the magic assistant, guiding the player. This required a file where all data and options were stored, as well as multiple graphics included on each screen had to be drawn, which was not able to do due to limited time. Instead, I simplified the code so that there were 6 different combinations. I also reduced the amount of endings I reduced the number of screens and states by mapping all outcomes to three shared ending screens (endSilly, endChaotic, endCure). I also used for loops and direct conditional logic (if/else), functions that I have some knowledge of to do the different endings and draw buttons.

### Role-Based Process Evidence

```
50 // start.js
51 if (currentScreen === "start") drawStart();
52 else if (currentScreen === "instr") drawInstr();
53 else if (currentScreen === "game") drawGame();
54 else if (currentScreen === "addon") drawAddon();
55 else if (currentScreen === "endCure") drawEndCure();
56 else if (currentScreen === "endChaotic") drawEndChaotic();
57 else if (currentScreen === "endSilly") drawEndSilly();
58
59 // (Optional teaching note)
60 // This "if/else chain" is a very common early approach.
61 // Later in the course you might replace it with:
62 // - a switch statement, or
63 // - an object/map of screens
64
65
66 // -----
67 // mousePressed() runs once each time the mouse is clicked
68 // -----
69 // This routes mouse input to the correct screen handler.
70 function mousePressed() {
71   // Each screen may define a mouse handler:
72   start.js      - startMousePressed()
73   instructions.js - instrMousePressed()
74   game.js       - gameMousePressed()
75   win.js        - winMousePressed()
76   lose.js       - loseMousePressed()
77
78   if (currentScreen === "start") startMousePressed();
79   else if (currentScreen === "instr") instrMousePressed();
80   else if (currentScreen === "game") gameMousePressed();
81   else if (currentScreen === "addon") addonMousePressed();
82   // The ?() means "call this function only if it exists"
83   // This prevents errors if a screen doesn't implement a handler.
84   else if (currentScreen === "endCure") endCureMousePressed();
85   else if (currentScreen === "endChaotic") endChaoticMousePressed();
86   else if (currentScreen === "endSilly") endSillyMousePressed();
87
88   { name: "Rainbow Flower", x: 250, y: 480, w: 200, h: 60 }
89   { name: "Honey Drops", x: 530, y: 480, w: 200, h: 60 }
90 }
91
92 function drawAddon() {
93   // Red-tinted background to communicate failure
94   background(220, 200, 255);
95
96   fill(0);
97   textAlign(CENTER, CENTER);
98
99   // Main message
100   textSize(32);
101   text("Choose an Add-on", width / 2, 160);
102   textSize(26);
103   text(`Base: ${chosenBase}`, width / 2, 220);
104
105   drawGameButtons(addonButtons);
106   cursor(addonButtons.some(isHover) ? HAND : ARROW);
107 }
108
109 // -----
110 // Mouse input for lose screen
111 // -----
112 // Any mouse click returns the player to the start screen
113 // (no buttons needed for this simple end state)
114 function addOnMousePressed() {
115   // (no buttons needed for this simple end state)
116   currentScreen = "start";
117   for (let b of addonButtons) {
118     if (isHover(b)) {
119       chosenAddon = b.name;
120
121       if (chosenBase === "Moonwater" && chosenAddon === "Rainbow Flower")
122         currentScreen = "endCure";
123       else if (chosenBase === "Goblin Hair")
124         currentScreen = "endSilly";
125       else
126         currentScreen = "endChaotic";
127     }
128   }
129   return;
130 }
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

## *GenAI Documentation*

GenAI was used for this project:

**Date Used:** Feb 1 2026

**Tool Disclosure:** ChatGPT 5.2

**Purpose of Use:** I used ChatGPT to debug and help guide me how to do specific functions/tasks

**Summary of Interaction:** I prompted ChatGPT to help fix my code, particularly debugging my for loop” as I forgot to assigned something to [i]. I suggested functions to use like the interpolation expression `${chosenBase}`.I made ChatGPT explain to me how these methods worked in detailed. Lastly, they helped me generate some pixel art for the ingredients.

**Human Decision Point(s):** When I needed some to help to debug my code so that the screening change would work from base ingredients to add-on ingredients, they made all the functions and draw happen on the same game.js file. I didn’t take this advice, and separated each screen into multiple files to easily transition the screens.

**Integrity & Verification Note:** I cross checked by going through our coding tutorial slides and checked if the information ChatGPT provided was accurate.

### **Scope of GenAI Use:**

- Gen ai was used as a debugging assistance, specifically to help identify why ingredient buttons were not rendering correctly. This included pointing out missing variable declarations and forgetting to declare buttons with const.) It was also used for general coding guidance, such as clarifying JavaScript syntax or logic when errors occurred.
- GenAI was not used for ☐ Designing the visual layout or UI structure choosing background colours, typography, or text sizing, creating or positioning buttons and images, structuring the project into multiple files/screens
- Writing the core draw() logic in main.js

**Limitations or Misfires:** The tool didn’t understand that there are some concepts we didn’t learn yet, and used overcomplicated codes.

## *Summary of Process (Human + Tool)*

Describe what you did, focusing on process rather than outcome. This may include:

The parts I did not use AI on was changing the background colours, text sizes/fonts, creating the buttons and images layout, as well as creating the different files/screens and coding them under function draw() in the main.js file. I loaded in the images and laid them out. I added in the buttons for base ingredients and add-on ingredients, as well as fixing errors for drawing the ingredient button. AI was interfered here because I forgot to include add const to the buttons.

### *Decision Points & Trade-offs*

Describe one or two key decisions you made:

- I used a for-loop to draw all three buttons because it was easier for me to understand than other methods like (for let) as I have used for loops before
- Initially, I had the images embedded with the const buttons and drawing them directly in the same loop, this causes lots of layering issues. I decided to preload and draw images independently in the screen's main draw function (drawGame() / drawAddon())

### *Verification & Judgement*

I verified the code by replaying the code many times, making sure that the text is readable and there are clear visual cues like keeping the hover, and on-screen instructions, or different colours good or bad endings. I also got peer feedback and explained the code myself so if we both understand it, then it's clear.

### *Limitations, Dead Ends, or Open Questions*

- The ending results were not that varied and used the same text for 3 different endings, but there were around 6 different combinations.
- I wanted to make an ending for each of them, but that most likely required a different approach, and more graphics to be made, which I did not have much time.

## Appendix

Me: I'm working on a potion game in p5.js. I stored data in an array of objectives so it's easier for me to call upon and draw 3 of the buttons at the sametime (drawGameButtons(basebuttons). But for some reason the buttons aren't responding.

```
baseButtons = [  
  { name: "Dragon Scale", x: 200, y: 420, w: 180, h: 60 },  
  { name: "Moonwater", x: 400, y: 420, w: 180, h: 60 },  
  { name: "Goblin Hair", x: 600, y: 420, w: 180, h: 60 }  
];
```

**ChatGPT:** Yes — you're missing a declaration keyword. You should define this with const or let. Without it, JavaScript won't properly scope the variable.

```
const baseButtons = [
  { name: "Dragon Scale", x: 200, y: 420, w: 180, h: 60 },
  { name: "Moonwater", x: 400, y: 420, w: 180, h: 60 },
  { name: "Goblin Hair", x: 600, y: 420, w: 180, h: 60 }
];
```

Me: I realize that the user needs to know what ingredient they picked before, and have it labeled underneath. is there a function that keeps track of what base ingredient the user picks previously every time?

```
function drawAddon() {
  // Red-tinted background to communicate failure
  background(220, 200, 255);

  fill(0);
  textAlign(CENTER, CENTER);

  // Main message
  textSize(32);
  text("Choose an Add-on", width / 2, 160);
```

**ChatGPT:** You can do that by storing the user's choice in a variable and then displaying it. Here's an example snippet:

```
Chatgpt: textSize(32);
text("Choose an Add-on", width / 2, 180);
textSize(18);
text(`Base: ${chosenBase}`, width / 2, 230);

drawButtons(addonButtons);
cursor(addonButtons.some(isHover) ? HAND : ARROW);
}
```

Me: I tried using a for loop to create the 3 buttons , but the buttons aren't showing up.

```
function drawGameButtons(buttons) {
  rectMode(CENTER);
  textAlign(CENTER, CENTER);

  for (let i = 0; i < buttons.length; i++) {

    fill(isHover(b) ? 200 : 255);
    stroke(0);
    rect(b.x, b.y, b.w, b.h, 12);
```

```

    fill(0);
    noStroke();
    textSize(18);
    text(b.name, b.x, b.y);
  }
}

```

**ChatGPT:** Ahh you're *very* close — this is a classic for-loop + variable mix-up 🙄  
 Your logic is right, but the buttons aren't showing because **you never define `b` inside the loop.**

Right now you have:

```

for (let i = 0; i < buttons.length; i++) {

  fill(isHover(b) ? 200 : 255);
  rect(b.x, b.y, b.w, b.h, 12);
  text(b.name, b.x, b.y);
}

```

aprr

## ✅ The fix

Inside the loop, you need to grab the current button from the array:

```

function drawGameButtons(buttons) {
  rectMode(CENTER);
  textAlign(CENTER, CENTER);

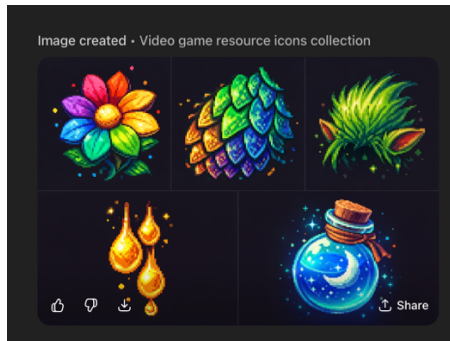
  for (let i = 0; i < buttons.length; i++) {
    let b = buttons[i]; // ← THIS LINE

    fill(isHover(b) ? 200 : 255);
    stroke(0);
    rect(b.x, b.y, b.w, b.h, 12);

    fill(0);
    noStroke();
    textSize(18);
    text(b.name, b.x, b.y);
  }
}

```

can you generate images game icons of a rainbow flower, dragon scale, goblin hair, honey drops, and moonwater? make them kind of video game style illustrative/ maybe a bit pixelated



me: I tried loading images inside basegame.js like this, but after I did that, my game stopped moving past the start screen.

```
function preloadBaseGameImages() {  
  dragonImg = loadImage("assets/ dragonscale.png");  
}
```

**ChatGPT:** That's because p5.js only supports **one preload()**, and it needs to be in main.js. Other files can *use* the images, but not preload them.

```
function preload() {  
  
  dragonImg = loadImage("assets/ravonscale.png");  
  
  moonwaterImg = loadImage("assets/moonwater.png");  
  
  goblinImg = loadImage("assets/goblin.png");  
  
}
```

**me:** That's is exactly what I did, but it still not loading the images, do I need to add images after that?

Chatgpt: Ahhh got it — you're asking about **folder structure**, not code 🙄

Short answer: **no, you don't have to** use assets/images, **but your path MUST match your folders exactly.**

---

## ✅ Valid setups (pick ONE)

Option 1: Images directly in `assets/`  
project-folder/

```
|
├─ index.html
├─ main.js
├─ start.js
├─ assets/
│   └─ dragonscale.png
│   └─ moonwater.png
│   └─ goblin.png
```

Then in `preload()`:

```
loadImage("assets/dragonscale.png");
```

---

Option 2: Images inside `assets/images/`  
project-folder/

```
|
├─ index.html
├─ main.js
├─ start.js
├─ assets/
│   └─ images/
│       └─ dragonscale.png
│       └─ moonwater.png
│       └─ goblin.png
```

Then you **must** do:

```
loadImage("assets/images/dragonscale.png");
```

**Me:** Nothing shows up still.

**ChatGPT:** Alright, let's debug this calmly and **surgically** — when images don't show in p5.js, it's almost always **one of 4 things**, and we can rule them out fast.

I'll walk you through this in the **exact order that catches the bug 99% of the time**.

---

## ✅ 1. First: confirm your folder + path (this is the #1 cause)

You said your images are in:

```
project-folder/
├─ index.html
```

```
| sketch.js
| start.js
| game.js
| assets/
|   | images/
|   |   | dragon.png
|   |   | moonwater.png
|   |   | goblin.png
```

👉 Then your paths **MUST** be:

"assets/images/dragon.png"

❌ NOT /assets/images/dragon.png

❌ NOT ../assets/images/dragon.png

---

## ✅ 2. Images MUST be loaded in `preload()` (one place only)

In **sketch.js** (or whatever file has `setup()` and `draw()`):

```
let dragonImg, moonwaterImg, goblinImg;

function preload() {
  dragonImg = loadImage("assets/dragon.png");
  moonwaterImg = loadImage("assets/moonwater.png");
  goblinImg = loadImage("assets/goblin.png");
}
```

⚠️ If this is not in the file that actually runs `setup()` → images will be undefined.

---

## ✅ 3. Buttons must store the IMAGE VARIABLE, not a string

This is a super common silent bug.

✅ Correct

```
const baseButtons = [
  { name: "Dragon Scale", x: 200, y: 420, w: 180, h: 60, img: dragonImg },
  { name: "Moonwater", x: 400, y: 420, w: 180, h: 60, img: moonwaterImg },
  { name: "Goblin Hair", x: 600, y: 420, w: 180, h: 60, img: goblinImg }
];
```

❌ Wrong (won't draw)

```
img: "assets/images/dragon.png"
```

---

## ✅ 4. Your draw function is almost right — add a safety check

Update your image loop slightly so we can *see* if the image exists:



```

for (let i = 0; i < buttons.length; i++) {
  let b = buttons[i];

  if (b.img !== undefined) {
    image(b.img, b.x, b.y - 90, 100, 100);
  }
}

```

If `b.img` is undefined, **nothing will draw and no error appears** — p5 fails silently.

**Human decision point(me):** I ended not using suggestion above because it was made everything extremely buggy, unnecessary, and nothing was loading. I just did it the easy way and loaded the game images individually, and loaded it under function `drawgame()`. I also found out the reason the images weren't loading because I didn't write down the entire folder structure when reloading images, I had to make it `assets/images`.

```

function drawGame() {
  // Set background colour for the game screen
  background(240, 230, 140);

  // ---- Title and instructions text ----
  fill(0); // black text
  textSize(32);
  textAlign(CENTER, CENTER);
  text("Choose a base ingredient.", width / 2, 160);

  imageMode(CENTER);
  image(dragonImg, 200, 320, 100, 100);
  image(moonwaterImg, 400, 320, 100, 100);
  image(goblinImg, 600, 320, 100, 100);
  // ---- Draw the button ----
  // We pass the button object to a helper function

  drawGameButtons(baseButtons);

  // ---- Cursor feedback ----
  // If the mouse is over the button, show a hand cursor
  // Otherwise, show the normal arrow cursor
  cursor(baseButtons.some(isHover) ? HAND : ARROW);
}

```

