

Implementation Report

I began coding by designing two important classes one for pizza and another for the conveyor belt, laying out the structure for the game. The pizza class will allow me to describe the properties and behaviour of the pizzas in the game, whilst the belt class will control the conveyor belt's operation. This first step establishes the basis for a dynamic and interactive pizza factory game, and as I continue to code, these classes will serve as the basis of my project, allowing me to add more features and interactivity.

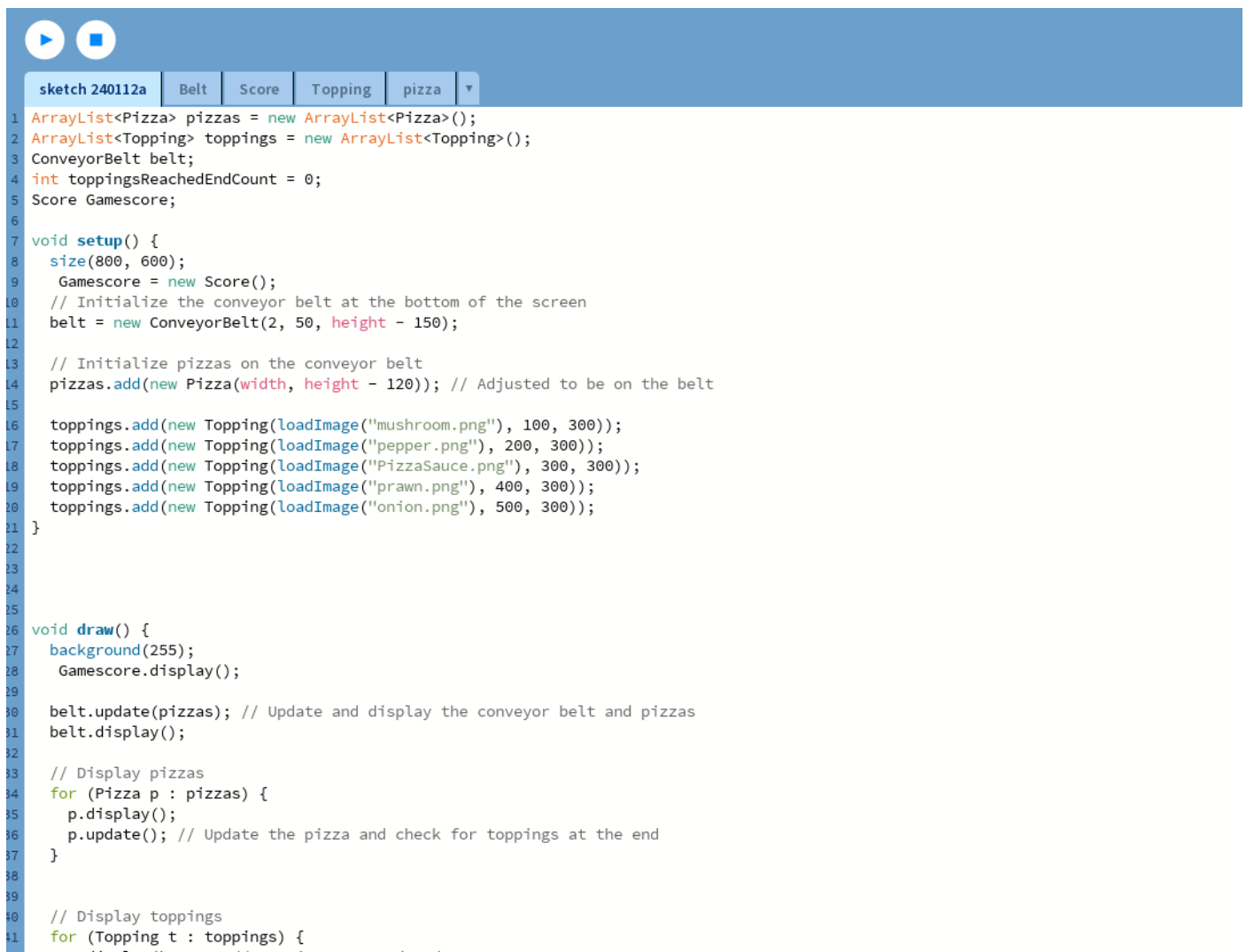
```
sketch 240112a | Belt | pizza ▾
1 ArrayList<Pizza> pizzas = new ArrayList<Pizza>();
2 ConveyorBelt belt;
3
4 void setup() {
5     size(800, 600);
6
7     // Initialize the conveyor belt at the bottom of the screen
8     belt = new ConveyorBelt(2, 50, height - 150);
9
10    // Initialize pizzas on the conveyor belt
11    pizzas.add(new Pizza(width, height - 120)); // Adjusted to be on the belt
12
13 }
14
15 void draw() {
16     background(255);
17
18     belt.update(pizzas); // Update and display the conveyor belt and pizzas
19     belt.display();
20
21     // Display pizzas
22     for (Pizza p : pizzas) {
23         p.display();
24         p.update(); // Update the pizza and check for toppings at the end
25     }
26 }
27
28
29
30
31
32
33
```

In addition to the pizza and conveyor belt classes, I've included a toppings class, which allows users to have different sorts of toppings in the game. Players can now drag and drop these toppings on their pizza while it moves on the conveyor belt.

```
sketch 240112a | Belt | Topping | pizza ▾
1 ArrayList<Pizza> pizzas = new ArrayList<Pizza>();
2 ArrayList<Topping> toppings = new ArrayList<Topping>();
3 ConveyorBelt belt;
4
5 void setup() {
6     size(800, 600);
7
8     // Initialize the conveyor belt at the bottom of the screen
9     belt = new ConveyorBelt(2, 50, height - 150);
10
11    // Initialize pizzas on the conveyor belt
12    pizzas.add(new Pizza(width, height - 120)); // Adjusted to be on the belt
13
14    // Load and initialize toppings
15    toppings.add(new Topping(loadImage("mushroom.png"), 100, 300));
16    toppings.add(new Topping(loadImage("pepper.png"), 200, 300));
17    toppings.add(new Topping(loadImage("PizzaSauce.png"), 300, 300));
18    toppings.add(new Topping(loadImage("prawn.png"), 400, 300));
19    toppings.add(new Topping(loadImage("onion.png"), 500, 300));
20 }
21
22
23 int toppingsReachedEndCount = 0;
24 void draw() {
25     background(255);
26
27     belt.update(pizzas); // Update and display the conveyor belt and pizzas
28     belt.display();
29
30     // Display pizzas
31     for (Pizza p : pizzas) {
32         p.display();
33         p.update(); // Update the pizza and check for toppings at the end
34     }
35
36     // Display the count of toppings that have reached the end
37
38     // Display toppings
39     for (Topping t : toppings) {
40         t.display(); // toppings are updated
41     }
42 }
43
```

Now the code is divided into different classes, such as Pizza, Topping, and Conveyor Belt. The Pizza class represents individual pizzas with multiple toppings. The Topping class illustrates the many toppings that may be added to pizzas. The Conveyor Belt class controls the movement of pizzas along the conveyor belt. The code allows the user to interact by dragging and releasing the mouse. When the player pulls a topping over a pizza, it sticks to the pizza. The code also tracks which toppings have reached the end of the conveyor belt.

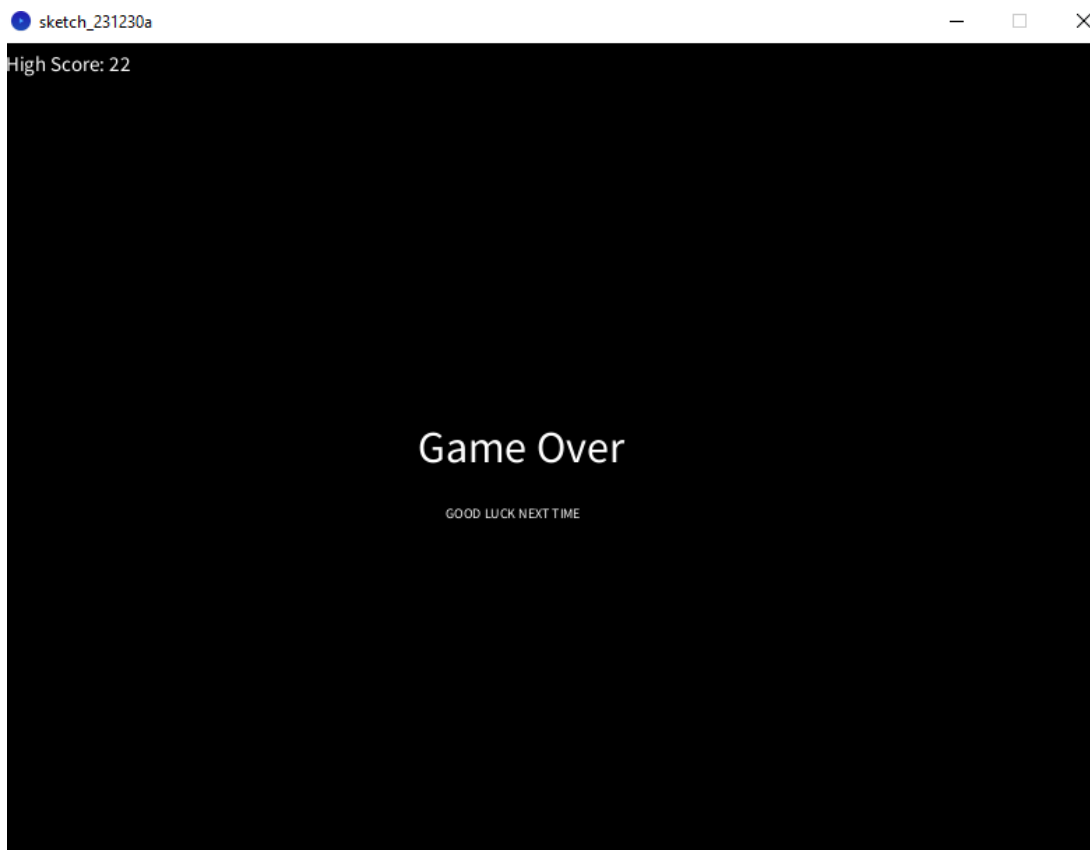
I have added score class that includes a scoring system that improves the game. The score class contains the score management which allows for simple score increments resets and display on the screen. It sets the score to zero and includes methods for incrementing and resetting it. The display method of the Score class ensures that the current score is visible on the game screen. This improvement improves overall gameplay by displaying a clear and dynamic sign of the player's progress.



```
1 ArrayList<Pizza> pizzas = new ArrayList<Pizza>();
2 ArrayList<Topping> toppings = new ArrayList<Topping>();
3 ConveyorBelt belt;
4 int toppingsReachedEndCount = 0;
5 Score Gamescore;
6
7 void setup() {
8     size(800, 600);
9     Gamescore = new Score();
10    // Initialize the conveyor belt at the bottom of the screen
11    belt = new ConveyorBelt(2, 50, height - 150);
12
13    // Initialize pizzas on the conveyor belt
14    pizzas.add(new Pizza(width, height - 120)); // Adjusted to be on the belt
15
16    toppings.add(new Topping(loadImage("mushroom.png"), 100, 300));
17    toppings.add(new Topping(loadImage("pepper.png"), 200, 300));
18    toppings.add(new Topping(loadImage("PizzaSauce.png"), 300, 300));
19    toppings.add(new Topping(loadImage("prawn.png"), 400, 300));
20    toppings.add(new Topping(loadImage("onion.png"), 500, 300));
21 }
22
23
24
25
26 void draw() {
27     background(255);
28     Gamescore.display();
29
30     belt.update(pizzas); // Update and display the conveyor belt and pizzas
31     belt.display();
32
33     // Display pizzas
34     for (Pizza p : pizzas) {
35         p.display();
36         p.update(); // Update the pizza and check for toppings at the end
37     }
38
39
40     // Display toppings
41     for (Topping t : toppings) {
```

The Level class adds a dynamic levelling system to the code which improves the overall gaming experience. The Level class is responsible for handling the current game level and its corresponding scoring rules. It sets the current level to 1 and the initial threshold to 2. The update Level method allows the player's level to be updated based on their current score. If the score is restored to zero, the level and threshold are reset to 1 and the player needs to complete two pizzas to move to the next level. The display method ensures that the player's current level is displayed on the screen. But I changed this so that I could meet the criterion, so instead of returning to level 1, I made the game stop and show a black screen with game over text in the middle.

```
sketch 231230a | ConveyorBelt | NewToppings | Pizza | Toppings | level | score | v
1 ArrayList<Pizza> pizzas = new ArrayList<Pizza>();
2 ArrayList<Topping> toppings = new ArrayList<Topping>();
3 NewToppings newToppings;
4 ConveyorBelt belt;
5 //int toppingsReachedEndCount = 0;
6 Score Gamescore;
7 Level gameLevel;
8 boolean gameOver = false;
9 void setup() {
10     size(800, 600);
11     Gamescore = new Score();
12     gameLevel = new Level();
13     belt = new ConveyorBelt(2, 50, height - 150);
14
15     pizzas.add(new Pizza(width, height - 120));
16
17     // Initialize only two toppings: PizzaSauce and Mushroom
18     toppings.add(new Topping(loadImage("PizzaSauce.png"), 300, 300));
19     toppings.add(new Topping(loadImage("mushroom.png"), 400, 300));
20
21     newToppings = new NewToppings(); // Initialize the new toppings class
22 }
23
24 void draw() {
25     if (!gameOver) {
26         background(255);
27         Gamescore.display();
28         gameLevel.updateLevel(Gamescore.score);
29         gameLevel.display();
30         belt.update(pizzas);
31         belt.display();
32
33         for (Pizza p : pizzas) {
34             p.display();
35             p.update();
36         }
37
38         if (Gamescore.score < 2) {
39             for (Topping t : toppings) {
40                 t.display();
41             }
42         }
43     } else {
44         // Game Over screen
45     }
46 }
```



Create a new class called New Toppings. When the player advances to level 2, this class will be in control of containing all the toppings. I modified the current Topping class. In this change, the Topping class will now only feature two specific toppings: Pizza Sauce and Mushroom, which will only appear in the first level. This indicates that the toppings from the New Toppings class should only appear on the screen when the game level reaches 2. This design ensures that the Topping and New Toppings classes have different functions, as well as the ability to render toppings depending on the game Level.

High Score: 22

Level: 1 Score: 0



My code resulted in the following Gameplay Mechanics: Players must add toppings to the pizzas before they reach the end of the conveyor belt. The game begins with two basic toppings: Pizza Sauce and Mushroom, and as the player progresses, more toppings become available, including pepper, onion, and Prawn. Each topping is represented by a separate class that handles its display, positioning, and interaction with the pizza. The pizzas themselves are objects capable of holding different toppings. Players use the mouse to drag toppings on the pizzas. This happens by detecting if the mouse moves over a topping and whether it is already attached to a pizza. When players select a topping, it moves with the mouse until player release it. If a topping is properly placed on a pizza, it becomes connected to it and change shape to fit the pizza shape. If a pizza exit the screen with no toppings, the game ends.

High Score: 22

Level: 2

Score: 2

