COMP 3609 PROJECT REPORT

Annika Boodoosingh - 816035294

Elena Panchoo - 816034966

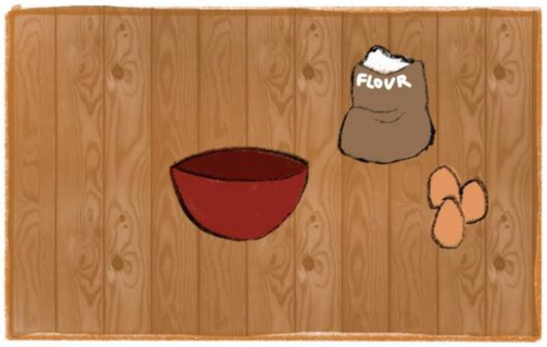
Table of Contents

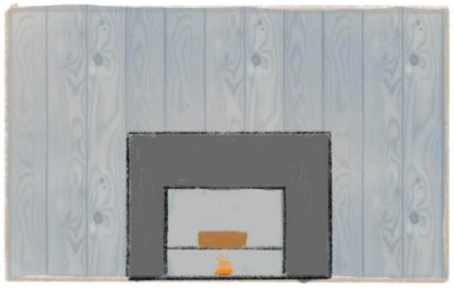
Game Concept

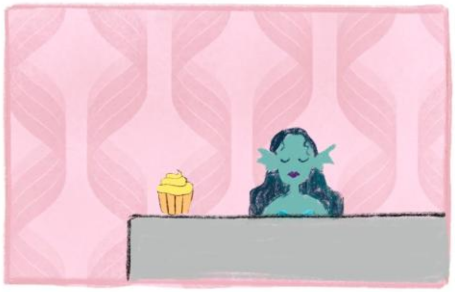
In Enchanted Bakery Dash, players step into the flour-dusted shoes of a young magical baker-in-training who has just inherited his grandmother’s mystical bakery, nestled deep within a dreamy, pastel-lit forest. Fairy customers arrive in waves, twinkling with anticipation but each has very limited patience. Wait too long and they will poof, vanishing. Players race between stations, catching falling enchanted ingredients with a floating mixing bowl. Be careful not to gather spoiled ingredients and waste time! Each dish must follow ancient, animated instructions, and timing is everything. Bake too early and your treat is raw. Wait too long and it’s burnt. Once the treat is perfectly baked, the real artistry begins. Players trace curves and swirls to decorate pastries into dazzling creations worthy of ice fairies or forest sprites. Each successful bake earns shimmering coins. Enchanted Bakery Dash is a cozy fantasy adventure full of whimsical charm and fast-paced magic. It is a world where baking is a spell, timing is your wand, and happiness is served warm and frosted.

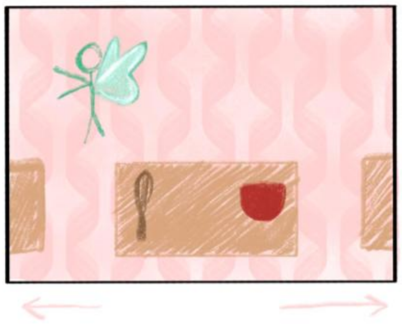
Concept Art

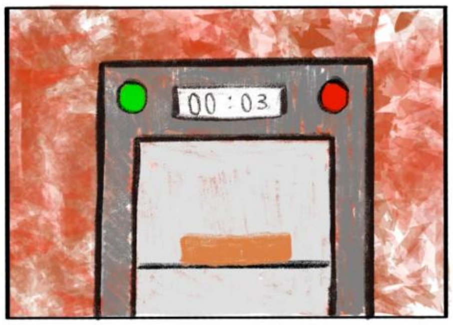
Enchanted Bakery Dash!
PLAY
EXIT

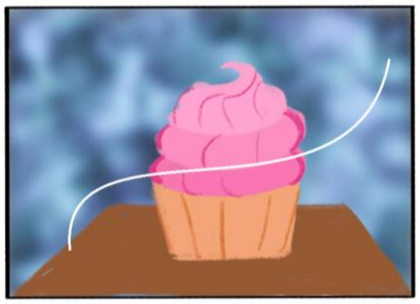


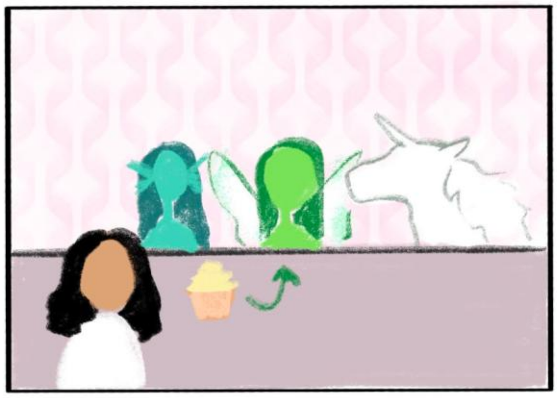












How to Play the Game

In Enchanted Bakery Dash, you control the baker using your keyboard's left and right arrow keys to move across the bakery. After selecting Start Game, you will be transported to the bakery, but you cannot move until 5 seconds have passed. Once the timer runs out, the first customer will appear, give their order, and you will see their patience level displayed as a bar, which decreases over time. At this point, you can begin moving.

To start, move left and right across the horizontal scrolling bakery. Head to the ingredient station and click it to start the mini-game. Instructions will be displayed, along with the order and recipe. Press Enter to begin and use the left and right arrows to catch the required ingredients. Be careful, catching bad ingredients will cost you points and money. If you fail to gather the correct ingredients, you will need to replay the ingredient mini-game. Once you have collected everything correctly, the oven station will unlock, and you can move on.

At the oven, click to open it. Instructions will appear, and you will press Enter to start. You will see the oven and the pastry inside, with a flame rotating around it. Press the Spacebar when the status says "Baked Perfectly". If you do this too early, the pastry will be raw; if you do it too late, it will be burnt. If you get the timing just right, move on to the decoration station. If not, you will need to replay until you get it right.

At the decoration station, which unlocks after winning the oven mini-game, click to open it. Instructions will appear, and you can press Enter to begin. You will see the pastry and a curve that you need to trace. Use your mouse to trace the given Bezier curve to decorate the pastry. If you trace it correctly, the decoration will be complete. If not, you will need to try again.

Once the decoration is done, serve the pastry to the customer. Depending on how well you have performed in the mini-games and how much patience the customer has left, you will earn money and points. If a customer's patience runs out while you are still working, they will leave with an angry message.

If you collect $100 in money in level 1, you will unlock level 2, where things get sped up and tricky! If not, the game ends. Good luck!

Scoring System

Ingredient Station

Successfully collecting all correct ingredients:

- Score increases by Time Left × 5

Oven Station

Perfect bake (press Spacebar at the right time):

- +75 points

Raw or burnt pastry (bad timing):

- 0 points

Decoration Station

Correct tracing of the curve:

- +75 points

Serving the Customer

Successfully serving the pastry:

- Earn $50 per customer who receives their order

Level Progression

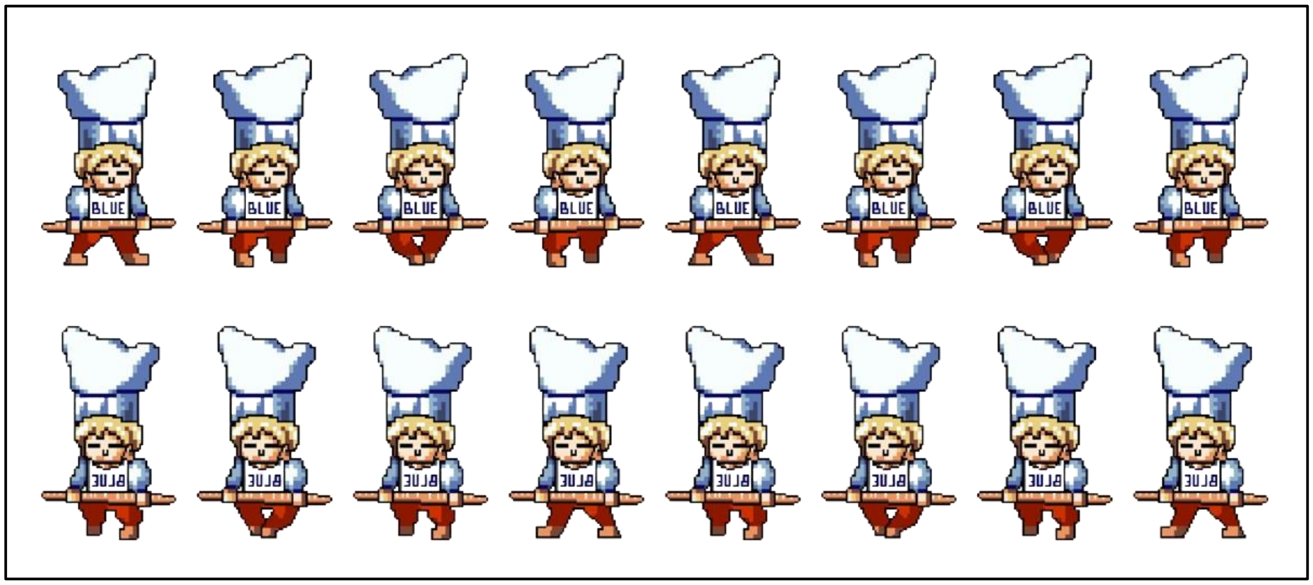
Unlock Level 2:

- If you earn $100 in Level 1, you move onto Level 2.

Game Entities

Player (Baker)

You control the baker using the left and right arrow keys to move across the horizontally scrolling bakery. The baker is responsible for completing each order step-by-step by interacting with different stations. From gathering ingredients to baking and decorating, the baker must work quickly and precisely to satisfy each magical customer.



Customers

These are whimsical NPCs including Fairies who appear one at a time per level. Each customer gives a specific pastry order and comes with a visible patience bar that slowly decreases over time. Their reactions, ranging from sparkly joy to dramatic disappointment, depend on how well their order is fulfilled and how long they waited.

Ingredient Station

This is where players collect ingredients through a mini-game. Clicking on the station opens it and pressing Enter begins the game. A list of required ingredients appears, and the player must move the bowl left and right to catch them while avoiding bad items like rotten eggs or spoiled sugar.

Ingredients

These include baking essentials like flour, sugar, eggs as well as occasional spoiled or cursed ones. Good ingredients must be caught during the mini-game to progress, while bad ones should be avoided, as they negatively impact your score.

Oven Station

This is the magical oven that players click to open and start the baking mini-game. The oven features a rotating flame, and the player must press the spacebar when the flame completes a full circle to bake the pastry perfectly. Early presses result in raw dough, late presses in burnt disasters.

Baked Pastry

This is the result of a successful baking session. If under- or overbaked, the player must repeat the baking mini-game.

Decorated Pastry

The final version of the pastry, after tracing the Bezier curve, now in vibrant color and ready to delight the customer. A well-decorated treat earns more money and better reactions.

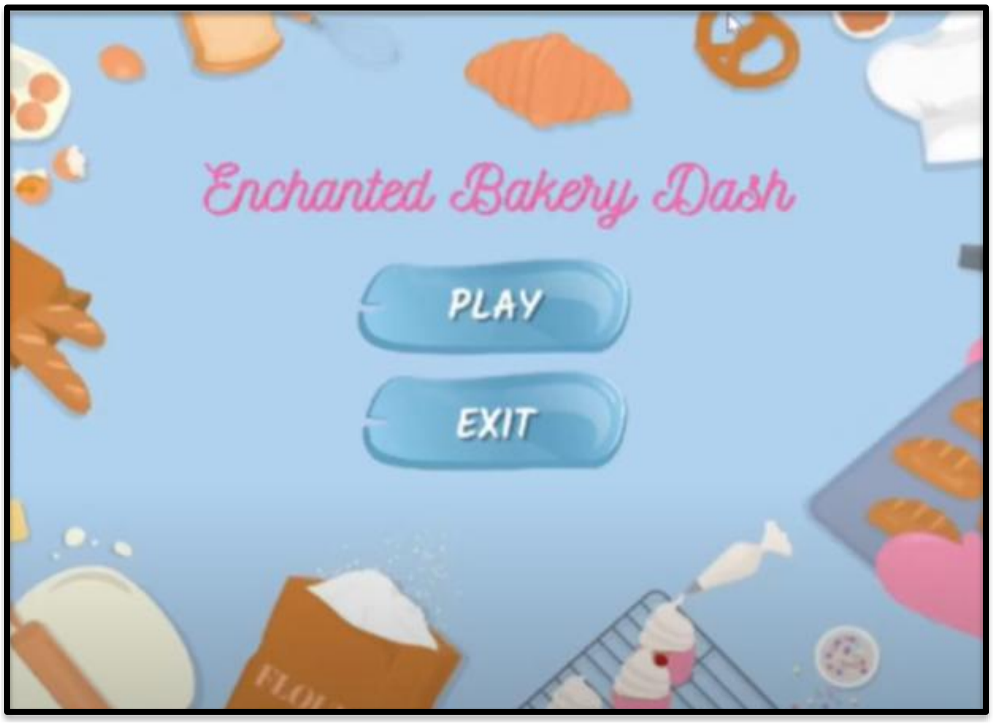
Decoration Station

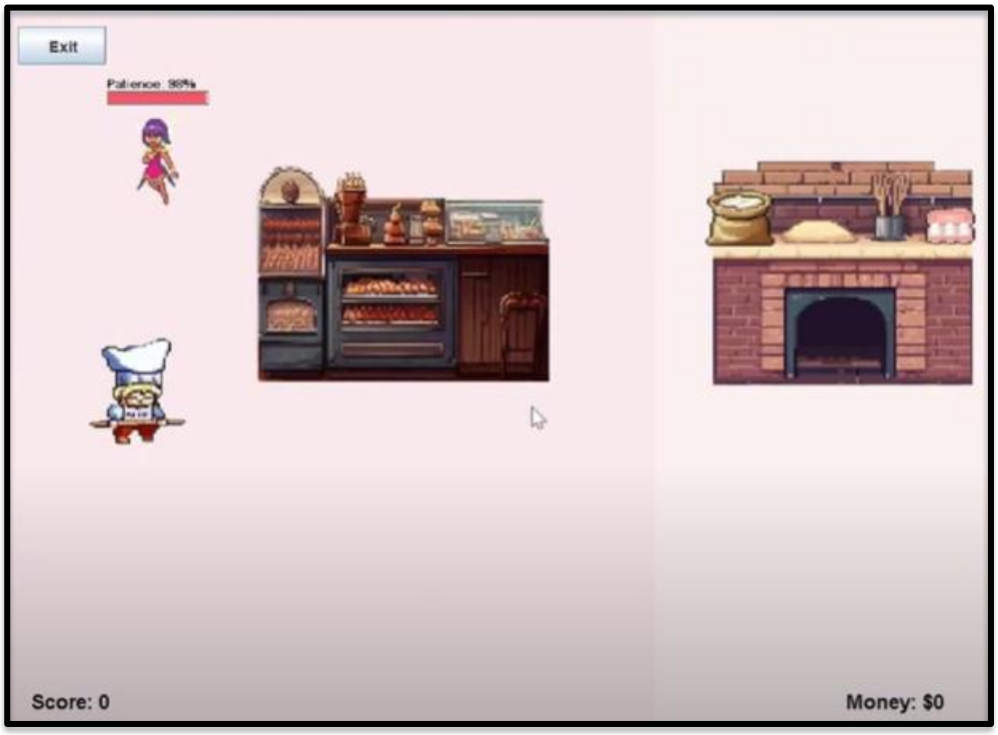
Once the pastry is baked, players click this station to begin decorating. The mini-game involves tracing glowing Bezier curves with the mouse. Only successful tracing results in a beautifully colored pastry. Failed attempts must be retried before delivering.

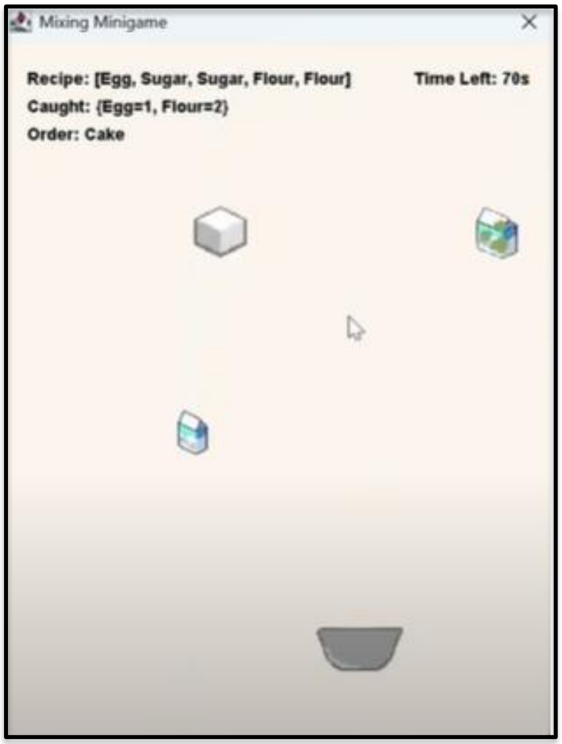
Order Counter

This is where customers place their orders and where finished pastries are delivered. After all mini-games are completed, players return to this counter to hand over the decorated pastry. Customer satisfaction is based on how well the steps were performed and how much patience they have left.

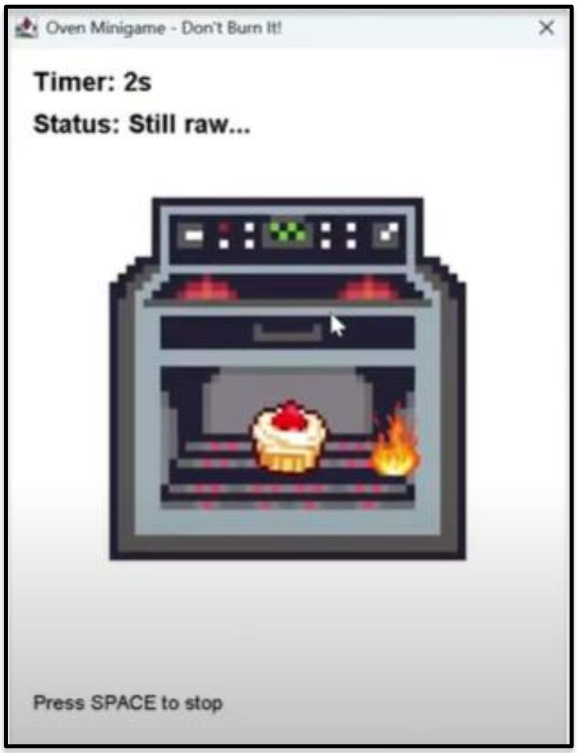
Game Screens





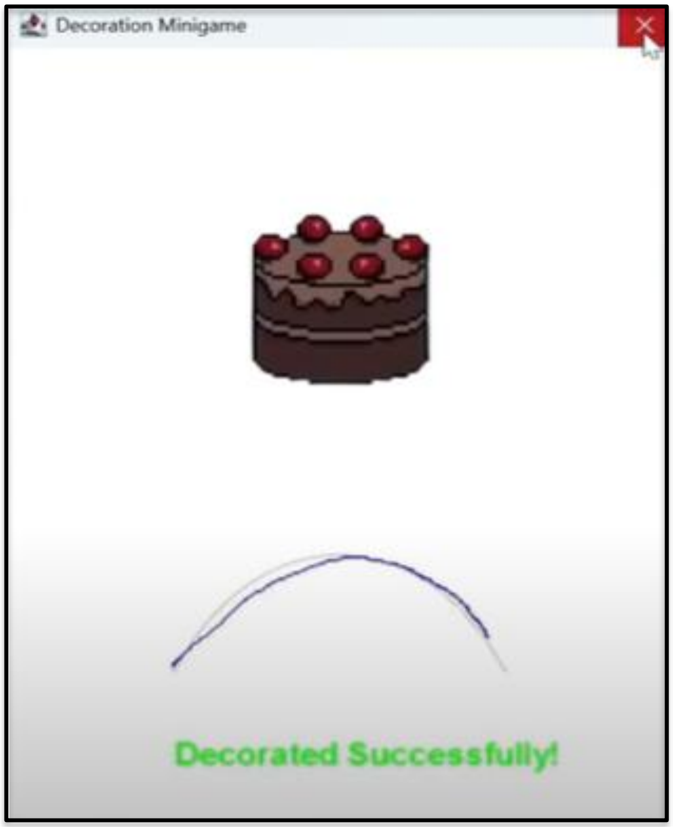


Recipe: [Egg, Sugar, Sugar, Flour, Flour] Time Left: 70s
Caught: {Egg=1, Flour=2}
Order: Cake

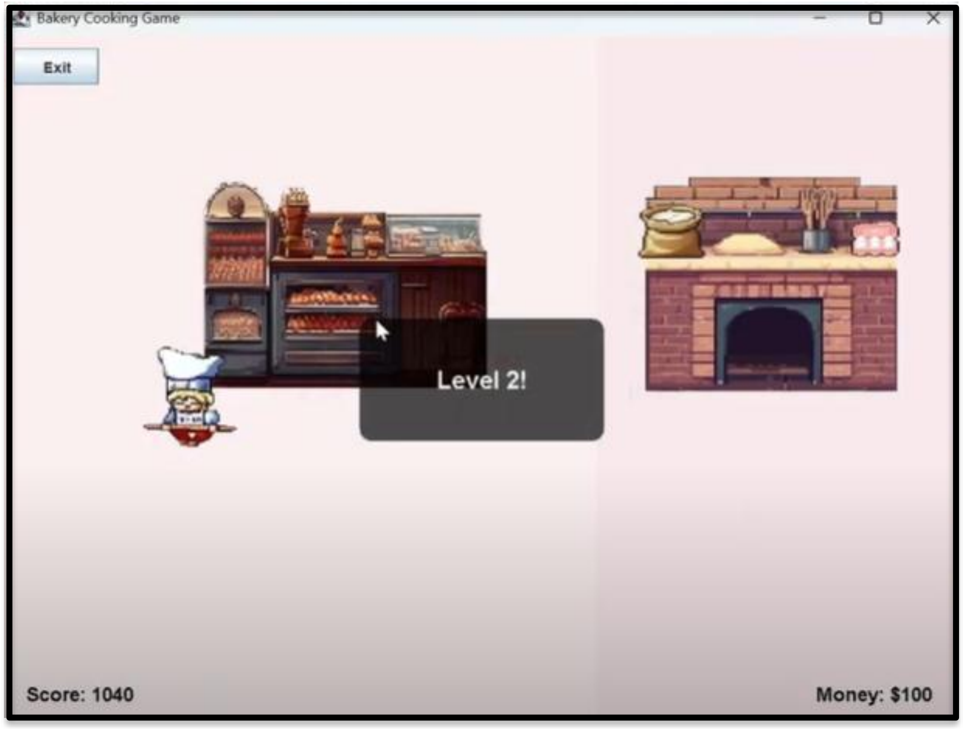


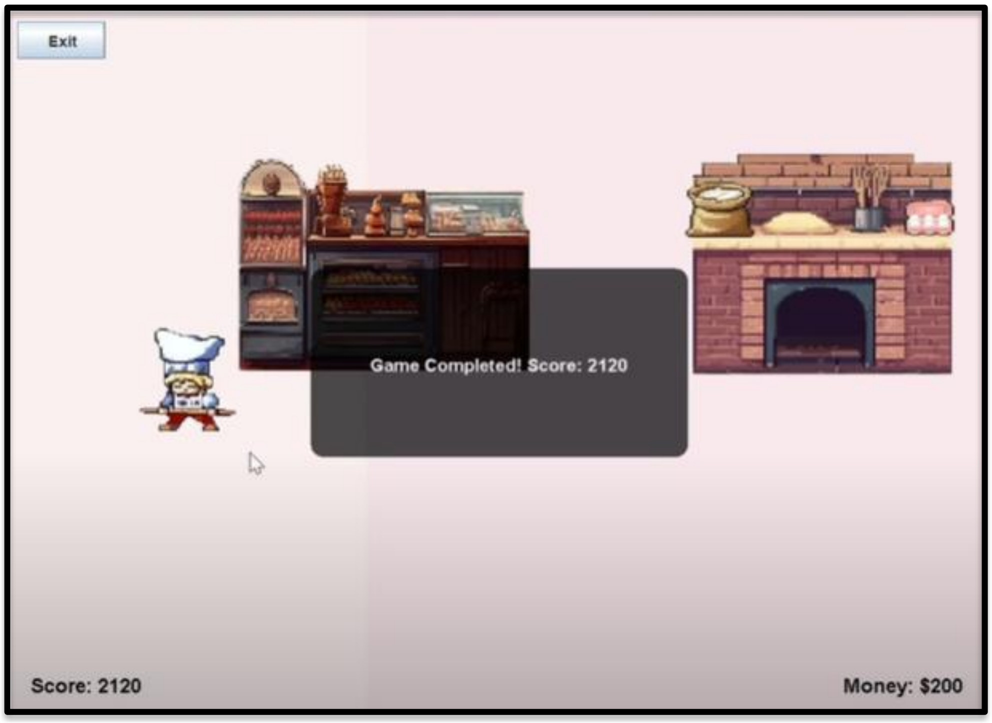
Timer: 2s
Status: Still raw...

Press SPACE to stop



Decorated Successfully!


Bakery Cooking Game
Exit
Level 2!
Score: 1040
Money: $100


Exit
Game Completed! Score: 2120
Score: 2120
Money: $200

Description of Implemented Classes

Game Application

This class is the main entry point of the game, responsible for launching the game.

Game Window

The GameWindow class manages the main game window using Java Swing. It sets up the frame, handles transitions between different screens (such as the start screen and the main gameplay panel), and manages basic window properties like size and close operations. It initializes both the StartScreen and GamePanel, displays the start screen by default, and switches to the gameplay panel when the player begins the game. It also ensures the gameplay panel receives keyboard focus when shown.

Game Panel

The GamePanel class is the main gameplay area where all visual elements like the baker, workstations, and customers are rendered. It handles the core game loop, user interactions through mouse and keyboard events, and manages animations, background movement, and minigame triggers. The panel dynamically updates based on player movement and customer actions, ensuring an engaging and interactive game experience.

ImageManager

The ImageManager handles the loading, storing, and providing of images required in the game. It helps optimize image usage by loading images once and reusing them when needed for rendering game objects.

SoundManager

The SoundManager class is responsible for managing sounds within the game. It handles loading, playing, stopping, and looping sounds and music. It also supports volume control, allowing for adjusting sound levels.

Baker

The Baker class represents the player character's visual and movement logic in the game. It handles loading and scaling a sprite sheet for animation, and manages the baker’s position on screen. The class includes functionality for horizontal movement (left and right), with boundary checks to keep the baker within the visible screen area. It also allows for dynamic updates to the right boundary based on the current panel width. Animation is handled by cycling through frames of the sprite sheet as the baker moves. The draw method is responsible for rendering the current animation frame at the baker’s position. Position, speed, and sprite dimensions are managed internally and can be accessed via getters.

Customer

The Customer class represents a static customer character in the game with a fixed position on screen. It handles loading and scaling a single customer image (not a sprite sheet), and provides methods for drawing the customer at the correct position relative to the camera. The customer's visual representation is managed through a BufferedImage, and their position and size are determined at instantiation. The draw method applies camera offsets to ensure proper placement in a scrolling environment. The class also provides access to the customer’s bounding box through the getBounds method, which can be used for collision detection or interaction.

WorkStation

The Workstation class represents a static interactive station in the game, defined by its position, dimensions, and an associated image. It is responsible for rendering its visual representation on the screen relative to a camera offset and for detecting mouse interactions through bounding box checks. The class includes methods for drawing the image, checking if the mouse is hovering over the workstation, and retrieving its position and dimensions. This class provides a generic framework for any in-game station where interactions or mini-games may be triggered.

Ingredient

The Ingredient class represents both good and bad ingredients that appear in the IngredientStation mini-game. Each ingredient has a name and coordinates, and it falls vertically from the top of the screen. The class provides functionality to generate random ingredients, update their vertical position to simulate falling, and detect collisions with a bowl using bounding logic. It includes predefined arrays of valid and bad ingredient names and ensures that ingredients are spawned within the bounds of the game screen. Getter methods allow external access to position data.

Bowl

The Bowl class represents the player's bowl in the IngredientStation mini-game, where it is used to collect falling ingredients. The bowl can be moved left or right within the bounds of the game screen. The class also handles the drawing of the bowl on the screen, either with a custom image or a default filled shape. Collision detection is performed using the getBounds() method to determine if the bowl collects ingredients as they fall. The position and dimensions of the bowl are adjustable based on the screen size.

IngredientGame

The IngredientGame class is a mini-game where players catch falling ingredients in a bowl to complete a recipe. The game starts with instructions displayed on the screen, explaining how to play, the recipe order, and the required ingredients. The recipe is chosen based on the player's selection, and ingredients are shown in a list, such as "Egg," "Sugar," and "Flour." The player must catch the correct ingredients while avoiding bad ones (e.g., bad eggs or milk). The game includes timers for updates and a countdown, and once all ingredients are collected correctly, a mixing animation is triggered, symbolizing the completion of the recipe. Players control the bowl's movement using the arrow keys to catch the ingredients and must do so within the time limit.

Pastry

The Pastry class represents a pastry image, such as a cupcake or cake, used in the baking game. It handles the loading, scaling, and display of the pastry image based on the selected recipe. The class adjusts the size of the pastry image depending on the recipe, with different scaling factors for various types (e.g., cookies, cakes, cupcakes). Once the pastry is baked, it is processed using the GrayScale class to simulate a baked (grayscale) appearance. After decorating, the pastry returns to its original color form. The Pastry class interacts with both the OvenStation and DecorationStation in the game, ultimately being served to the customer by the player once completed.

OvenGame

The OvenGame class simulates a baking mini-game where the player attempts to bake a pastry to perfection. The game features an animated oven with a timer that tracks the baking process. The player must stop the timer at the right moment to avoid burning the pastry. If the pastry is underbaked, it will remain raw, and if overbaked, it will burn. The game includes interactive elements like a flame animation and instructions for the player, allowing them to start and stop the baking process. The objective is to time the baking precisely to achieve the "Perfect Bake" status. The game is reset if the pastry is burnt, raw, or baked successfully, and the player can try again to achieve the best result.

DecorationGame

The DecorationGame class provides an interactive decoration mini-game where players decorate freshly baked pastries. After the pastry has been baked, it is sent to the decoration station. The station uses a Bézier curve to guide the player's decoration path. A trace of the curve is displayed on the screen, and the player must follow this path to successfully decorate the pastry. If the player traces the path accurately, the pastry transforms from grayscale to its full color, signifying a successful decoration. If the player deviates from the curve, the path is reset, and the player must try again, accurately following the curve before progressing. This adds a challenge to the decoration process, encouraging precision and concentration for completing the mini-game.

Background

The Background class is responsible for managing and rendering a scrolling background in the game. It stores an image of the background and handles its movement, either to the left or right, based on player input. The background scrolls by updating the bgX, bg1X, and bg2X positions. When the background images move off-screen, they are reset to create a continuous loop effect. The move method determines the direction of movement, and the draw method is used to render the background images using Graphics2D. This allows for smooth scrolling of the background as the player navigates the game environment.

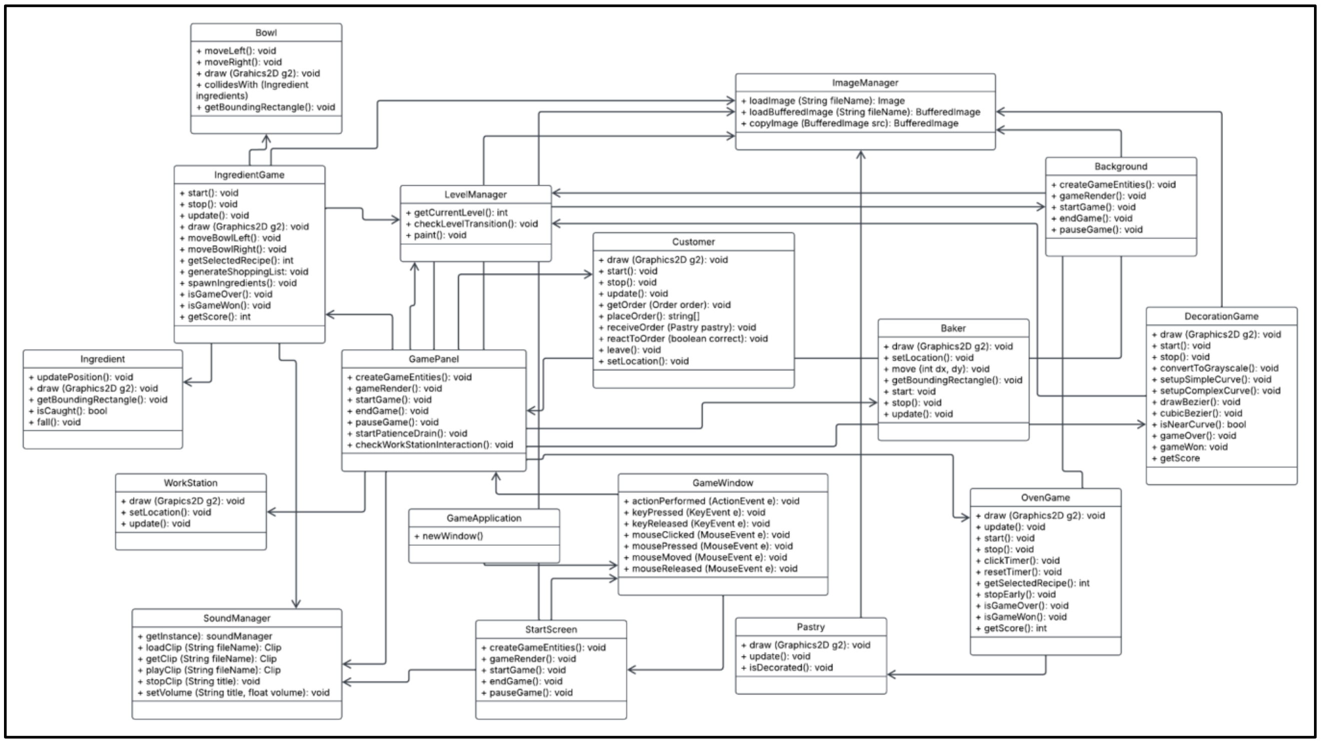
Start Screen

The StartScreen class represents the introductory screen of a game, where players can either start the game or exit. It extends JPanel to manage the layout and painting of UI components such as the background and buttons. The class uses BufferedImage and Graphics to handle double buffering for smooth rendering of the game elements. It contains references to images for the background and buttons (Start and Exit), and uses mouse listeners to detect user interactions with these buttons. When a player clicks the start button, the game switches to the gameplay screen. The class also includes a SoundManager to play background music, and it dynamically updates the positions of the buttons whenever the window is resized. The visual components are drawn using paintComponent, ensuring efficient rendering and proper scaling of the UI elements.

Level Manager

This class uses the method checkLevelTransition where it takes in the money earned, and if it is 100, it triggers level 2 to begin. It also contains the current level, which can then be passed to other classes in order to trigger level 2 behaviours, such as sped up ingredients falling, faster baking times and more complex Bezier curves. It allows for a cleaner integration of levels within the game, and allows the game to further expand into more levels if needed.

Object-Oriented Model of Classes


Bowl
+ moveLeft(): void
+ moveRight(): void
+ draw (Grahics2D g2): void
+ collidesWith (Ingredient ingredients)
+ getBoundingRectangle(): void
ImageManager
+ loadImage (String fileName): Image
+ loadBufferedImage (String fileName): BufferedImage
+ copyImage (BufferedImage src): BufferedImage
IngredientGame
+ start(): void
+ stop(): void
+ update(): void
+ draw (Graphics2D g2): void
+ moveBowlLeft(): void
+ moveBowlRight(): void
+ getSelectedRecipe(): int
+ generateShoppingList: void
+ spawnIngredients(): void
+ isGameOver(): void
+ isGameWon(): void
+ getScore(): int
LevelManager
+ getCurrentLevel(): int
+ checkLevelTransition(): void
+ paint(): void
Background
+ createGameEntities(): void
+ gameRender(): void
+ startGame(): void
+ endGame(): void
+ pauseGame(): void
Customer
+ draw (Graphics2D g2): void
+ start(): void
+ stop(): void
+ update(): void
+ getOrder (Order order): void
+ placeOrder(): string[]
+ receiveOrder (Pastry pastry): void
+ reactToOrder (boolean correct): void
+ leave(): void
+ setLocation(): void
DecorationGame
+ draw (Graphics2D g2): void
+ start(): void
+ stop(): void
+ convertToGrayscale(): void
+ setupSimpleCurve(): void
+ setupComplexCurve(): void
+ drawBezier(): void
+ cubicBezier(): void
+ isNearCurve(): bool
+ gameOver(): void
+ gameWon: void
+ getScore
Baker
+ draw (Graphics2D g2): void
+ setLocation(): void
+ move (int dx, dy): void
+ getBoundingRectangle(): void
+ start: void
+ stop(): void
+ update(): void
Ingredient
+ updatePosition(): void
+ draw (Graphics2D g2): void
+ getBoundingRectangle(): void
+ isCaught(): bool
+ fall(): void
GamePanel
+ createGameEntities(): void
+ gameRender(): void
+ startGame(): void
+ endGame(): void
+ pauseGame(): void
+ startPatienceDrain(): void
+ checkWorkStationInteraction(): void
WorkStation
+ draw (Grapics2D g2): void
+ setLocation(): void
+ update(): void
GameWindow
+ actionPerformed (ActionEvent e): void
+ keyPressed (KeyEvent e): void
+ keyReleased (KeyEvent e): void
+ mouseClicked (MouseEvent e): void
+ mousePressed (MouseEvent e): void
+ mouseMoved (MouseEvent e): void
+ mouseReleased (MouseEvent e): void
GameApplication
+ newWindow()
OvenGame
+ draw (Graphics2D g2): void
+ update(): void
+ start(): void
+ stop(): void
+ clickTimer(): void
+ resetTimer(): void
+ getSelectedRecipe(): int
+ stopEarly(): void
+ isGameOver(): void
+ isGameWon(): void
+ getScore(): int
SoundManager
+ getInstance(): soundManager
+ loadClip (String fileName): Clip
+ getClip (String fileName): Clip
+ playClip (String fileName): Clip
+ stopClip (String title): void
+ setVolume (String title, float volume): void
StartScreen
+ createGameEntities(): void
+ gameRender(): void
+ startGame(): void
+ endGame(): void
+ pauseGame(): void
Pastry
+ draw (Graphics2D g2): void
+ update(): void
+ isDecorated(): void

Sources

Images:

General

- https://www.pinterest.com/
- https://www.canva.com/

Pastries

- https://pentixel.itch.io/cake-assets
- https://captainskolot.itch.io/100-cakes-asset-pixelart-pixel-art-sprite-dessert-pack-rpg

Chef

- https://www.artstation.com/artwork/zONRwd

Customers

- https://www.pinterest.com/pin/757871443543351762/

Ingredients

- https://www.reddit.com/r/PixelArt/comments/we7k1s/learning_how_to_pixel_art_egg/?rdt=42301
- https://www.craiyon.com/image/A3o_ZOMyQSe1uQJn_QYU9g
- https://www.freepik.com/premium-vector/pixel-art-illustration-milk-pixelated-delicious-milk-fresh-milk-drink-icon-pixelated-pixel_80325076.htm
- https://www.shutterstock.com/image-vector/pieces-butter-stick-cut-isolated-on-2228443773

Station

- https://www.reddit.com/r/PixelArt/comments/uw15jb/my_first_little_piece_of_pixel_art_a_baby_lil_oven/

Sounds

Customers

- https://typecast.ai/voices/cartoon-character-voice-generator

Collect Correct Ingredients

- https://pixabay.com/sound-effects/coin-recieved-230517/

Collect Wrong Ingredient

- https://pixabay.com/sound-effects/wrong-47985/

Background Music

- https://www.boomplay.com/songs/188462586?from=artists

Requirements Fulfillment

Course Content	Met (Yes/No)	How Requirement Was Fulfilled
Creating and drawing shapes	✓	The patient bar of the customer is drawn using rectangles, where a gray border is drawn first, followed by a red fill representing the patient's current level. Uses calls such as drawRect() and fillRect()
Drawing text on the screen	✓	Text such as instructions, cash received, points and timers are drawn onto the screen using Graphics.drawString()
The basic game loop	✓	The game loop is implemented in the game to continuously update the game state and render frames. It consists of a cycle of handling input, updating game objects, and drawing graphics on the screen at a constant frame rate. The loop ensures smooth transitions and continuous gameplay.
Drawing and updating	✓	In the game, graphics are drawn and updated in the main game loop. This includes redrawing game entities, backgrounds, and UI elements in each iteration of the loop. The screen is cleared before drawing the updated scene to prevent previous frames from persisting.
Game entities	✓	Game entities such as the baker, customers, ingredients, pastries and more are represented as objects with properties like position, speed, and image. These entities are updated and drawn each frame, interacting with each other based on game logic.
Moving a simple 2D shape	✓	The game updates the position and appearance of a 2D shape (the patience bar) dynamically, reflecting changes in the patience level over time and adjusting the width of the shape accordingly.

Getting input from the user	✓	The game accepts user input in the form of the left and right arrow keys to move the player left and right. In the ingredient collection game, the bowl moves left and right to collect ingredients. It also takes mouse input to trace a curve to detect the oven's path, and the space bar is used to stop the baking process.
Bounding rectangles	✓	Bounding rectangles checked and used to ensure player, ingredients and bowl remain within screen bounds when moving.
Rectangle intersection	✓	Used in the game for collision detection, to handle the detection when the player's bowl collides with ingredients, ensuring proper ingredient collection.
Screen boundaries	✓	Screen boundaries are utilized several times, to ensure the player cannot move off screen, the bowl in the ingredient game does not move off screen and the falling ingredients disappear at screens end to not go beyond.
Loading images	✓	Using the ImageManager class, images are loaded into the game for the customers, ingredients, pastries and workstations.
Loading and manipulating sprite sheets	✓	For the player's walking animation, a sprite sheet is loaded. Each frame corresponds to a specific walking pose, and the game updates the player's animation by cycling through these frames at regular intervals.
Flickering, double buffering, tearing, and page flipping	✓	An ImageManager class is used and game entity images are drawn onto the buffered image ImageContext, to prevent flickering.
Image effects	✓	Grayscale is applied to the pastry image when baking and prior to decoration, its original color restored once the pastry is successfully decorated.
Loading and playing sound file in Java	✓	Sounds are loaded and played in the game, such as the jazzy background music, customers placing orders, collecting ingredients, baking, angry customers and more.

Simulating movement and distance in games	✓	The game horizontally scrolls as the player moves left and right, simulating movement around the bakery.
Game states	✓	The game has many states, including start, baker and workstation games states.
Managing game screens	✓	Many screens are present and managed, the start screen appears when the game runs, upon clicking start taken to the bakery, clicking on the workstation opens smaller screens with workstation games.
Side scroller concept	✓	The game horizontally scrolls as the player moves left and right, simulating movement around the bakery.
Tiled map editor	X	NA
Loading and displaying tile maps	X	NA
Map collision detection	X	NA
Smooth jump motion and gravity	X	NA
Projectile motion	X	NA
Circular motion	✓	Flame image moves in a circular motion around the pastry while it is baking in the oven minigame.
Movement along a slope	X	NA
Movement along a Bezier curve	✓	Bezier curve drawn for player's to trace correctly to decorate the pastry in the decoration minigame.
Levels	✓	The game includes 2 levels, and the player moves onto the 2nd level only if they make enough money in level 1. Level 2 introduces new customers such as mermaids.

22/28 = 79% of course content met

Demo Video

https://youtu.be/8zPSawUuCMA?si=UyWOVBY9MfD6N6UC

Contributions

Annika Boodoosingh	Elena Panchoo
- Sourced graphics & sound - Created base of gameplay - Created stations 2 & 3	- Sourced graphics - Created stations 1 & 4 - Implemented leveling system