Minerva’s Marvelous Mixturette

by WitchWorks

By

**List all team members here**

**Giovanni DeFiore**

**Audrey Ruiz**

**Bee Morse**

**Gina Piccirilli**

**Diego Martinez**

**IGME-106, Game Development and Algorithmic Problem Solving II**

**Prof. Mesh  
Prof.Snyder**

**Spring 2024**

**Rochester Institute of Technology**

**School of Interactive Games & Media**

**B. Thomas Golisano College of Computing and Information Sciences**

**Instructions**

**Your project documentation should be professionally written, proofread and spellchecked, and organized for consistency. It should look like a single document from a unified team, and everyone on the team is responsible for it in its entirety.**

Throughout all of your documentation, we expect you to write in well-formed, complete sentences and provide your rationale for all decisions. In other words, **don’t just tell us WHAT you are doing, tell us WHY**. This document should tell the story of what you made and how you made it. It should be interesting to read. To this end, everyone on the team should read and help edit all sections!

**Timeline**

It is VERY tempting to focus on making the game and write everything up later. “It’ll be fine. I just have to write it up.” is a common sentiment. Don’t succumb to this! Yes, the writing isn’t the fun part - but it’s a required element of this course.

**Instead of a final hurdle to complete, think of this document as an evolving view into the internals of your team’s objective, progress, and lessons learned. Use it as a communication tool internally and externally.** If you work on it slowly, writing when things are fresh in your head, you’ll find that, before you know it, it’s done and fairly painless. (You’ll also get MUCH better feedback from stakeholders if they can read in small chunks instead of all at once.)

For more about letting your writing evolve and preventing perfect from being the enemy of good/done, see: <https://wrd.as.uky.edu/sites/default/files/1-Shitty%20First%20Drafts.pdf>

*Replace this page with a flyer/promotional image of some kind once you have it.* 🙂

**Quick Reference**

**Contact Info**

|  |  |  |
| --- | --- | --- |
| **Name** | **RIT Email** | **GitHub Username** |
| Gio DeFiore | gld1976@g.rit.edu | GioDefiore |
| Diego Martinez | dm7450@g.rit.edu | DiegoMartinez66 |
| Gina Piccirilli | gnp3602@rit.edu | GineBean |
| Bee Morse | bcm6816@rit.edu | superwhack |
| Audrey Ruiz | ar9388@g.rit.edu | Audrey\_Ruiz |

**Team Availability & Weekly Meeting Time(s)**

I suggest creating a <https://www.when2meet.com/> survey using “Days of the Week” to capture the overall availability of the team and paste a link to that survey here. Then decide on a few times per week when the team will meet (if you sometimes agree to skip a meeting, that’s fine, but make sure you have times set aside anyway!)

**Other Links**

Add more links, info, etc. as needed here as the project progresses. Some suggestions are below. These aren’t things that we'll grade, but it can be helpful for me to be able to check various artifacts directly so you can keep the [Sprint Reports](#_f5lnz425m4hq) focused on reflections vs. details.

|  |  |
| --- | --- |
| ***GitHub Repo*** | <https://github.com/DiegoMartinez66/gdaps2_2235_team_G> |
| ***Final Project Presentation Slides*** | <https://docs.google.com/presentation/d/1y8cbs0ZUi_CA2Qc36_eQD9ZnaGxVrRUSkDFUIB-sYHk/edit?usp=sharing> |

**Acknowledgments**

Thank people here. Your faculty, external team members, peers, etc. No, this isn’t a published paper, graduate thesis, etc., but all work we do is supported by others. wet’s good to learn how to acknowledge them officially. This doesn’t need to be long, sappy, etc., but it should also be more than a list of names. <https://www.discoverphds.com/advice/doing/acknowledgements-for-thesis-and-dissertations> is in the context of Ph.D. acknowledgments but has nice examples of appropriate tone and scope.

Thank you to the professors for giving us the opportunity as a team to create a game that we all can be proud of. Guiding us through difficult and troubling times and showing us to be effective professionals in our studies.

Thank you to all the playtesters who donated their time to test our game for any further steps and their words of encouragement about our project.

Thank you to the graduate students who gave their time to playtest and guide us through any further steps, giving their great wisdom in the design. Hoping for them a great graduation and they shall not be blessed with a gray hair.

Thank you to the TAs with advice about our pre-production, production, and final production of our project. Guiding us through their knowledge and their time to playtest our project as well.

Thank you to our peers who listened and participated with our project either asking questions during presentation or playtesting our project as we did with theirs.

Thank you to Gio who was with us during the initial introduction of our team and pre-production for our project. You will not be forgotten and we acknowledge your initial efforts for the team and project.

Finally, a thanks to each team member who in difficult and stressful circumstances was able to contribute to the project and are now able to say that they’ve created a game. Additionally, we acknowledge that as a team we were able to work through the semester and not call quits on the project, and we were able to run through and give our contributions.

**Table of Contents**

This should update correctly whenever refreshed if you use the formats already defined consistently.

[1 Introduction 10](#_6amjflbtw2ng)

[1.1 Game Synopsis 10](#_omb3lmvqdu0i)

[1.2 Mission Statement 10](#_1r3bteb2w5y9)

[2 Production 11](#_mzz2096otcd7)

[2.1 Organization 11](#_y7gwaor2lvvb)

[2.2 Core Values & Responsibilities 11](#_xc11gyt3l82k)

[2.3 Risk Analysis 12](#_217bahsucrtf)

[2.4 Planning & Scope 13](#_lnbhdos2pgnc)

[2.5 Task Management 14](#_psr9w1hggxqc)

[2.6 Asset Pipeline 15](#_qvdvsoajgrp9)

[3 Game Design 15](#_bi27a7s433ob)

[3.1 Game Overview 15](#_goqsel7dpent)

[3.2 Audience & Genre 16](#_ttp2cpfmmv1i)

[3.3 Inspirations 16](#_730j49pf6v4z)

[3.4 Narrative Design 17](#_r0ds3n8ntpr3)

[3.5 Gameplay 18](#_prc5dtezrs65)

[3.6 Aesthetics 19](#_io1gxy7b1bt0)

[3.7 User Interface 20](#_xbsr0sy25iih)

[3.8 Onboarding 20](#_n1fv0n9j3b81)

[4 Technical Architecture 21](#_6ylq2kgjcebz)

[4.1 Overview 21](#_gnfn8ig21602)

[4.2 Player Interactions 22](#_ruyb3lv1317h)

[4.3 Levels & Environment 22](#_7e2wnis6uqfi)

[4.4 Game Modes & Menus 23](#_t1da2tdz92rt)

[4.5 Data-Driven Elements 23](#_zaclelj3u2ng)

[4.6 Gameplay Sub-States 24](#_peuwrh8a13jy)

[4.7 <Sub-System> 25](#_tvtjtgo3rh6y)

[5 Sprint Reports 26](#_f5lnz425m4hq)

[5.1 Sprint #1 26](#_te8a3azf51wj)

[5.2 Sprint #2 27](#_lvxq6rl2pvq4)

[5.3 Sprint #3 28](#_kui4i2dv8se3)

[5.4 Sprint #4 28](#_bag4y04x06vp)

[5.5 Wrap-Up 29](#_8juz4ggtx1ic)

[6 Postmortem 31](#_fy8um6syhezh)

[6.1 Overview 31](#_w6nss2fdqsn)

[6.2 Reflections 31](#_dqu5fflcyjcy)

[6.3 Conclusions 32](#_q463wnwkmjsa)

[6.4 Future Work 32](#_pba5wk8w5jam)

# Introduction

This is NOT your entire game design, but there should be enough here to provide context for Production, etc. even without having all of the game design details. Potentially interested stakeholders should also be able to read ONLY this section in order to decide if they want to read more.

## Game Synopsis

Introduce the game itself and your goals. This is effectively your elevator pitch. What is the intended overall player experience? Ensure that you give a tagline, summarize the gameplay game mechanics, and introduce what will make this game unique.

<https://davidmullich.com/2018/06/25/an-actionable-game-design-document-template/> suggests describing an imaginary play session. “Create and name an imaginary player and then put yourself in their shoes, playing the game for five minutes, beginning with him or her starting the game for the first time. Focus on the player’s experience: what they’re seeing, what they’re hearing, what they’re doing, and what they’re thinking about. Any thoughts or feelings they have should be reflections of what is going on in the game.

Minerva’s Marvelous Mixturette is a cartoonish arcade game where you prepare, mix, and decorate potions from a horse-drawn food cart! Use rapid sequences of keystrokes to satisfy your customers’ orders before they run out of patience across five different levels.

## Mission Statement

Discuss each of your individual and team goals for this project (beyond “get a good grade and pass the class”) and decide on an overall team mission statement. For example, do you want a strong portfolio piece? Learn new technical skills? Improve project management skills? What are your priorities? Having (and documenting!) this conversation early will help keep everyone moving in the same direction when the project gets busier later in the semester.

We want to gain experience working on a game as a group by communicating with each other and an opportunity to experiment with this project.

Diego: I want to continue to look for experiences, work with a group of people and create something new and proud of.

Audrey: I want to gain insight on group work so I can find ways to apply myself within group settings using the skills I already have as a programmer.

Gina: I want to get practice working with others on a coding project, learning how to read other’s work and make my own work readable for others to be able to work smoothly as a team and progress in the project.

Bee: I want to better learn how to apply my programming skills in an undirected setting, using concepts we learn in a more natural environment than PEs or Homework. I also want to learn how to work on a codebase with multiple others.

# Production

**This chapter drills down on your current process. As it changes, update this!! You should have the 1st pass of everything here complete by the end of sprint 1 and then continue to update it throughout the project.**

Note: your analysis of how well the production process worked and changes you made along the way goes into the post-mortem later in this document.

## Organization

How did you structure your team? What skills did you have available as a combined group and how did that influence your objectives and process?

We structured ourselves by picking aspects of the project we each felt we were best suited for. For instance, while we all did technical work on the project, some of us worked on the UML diagram at the same time others were designing assets for the game. Creating this game will require us to focus on creating a solid technical basis before we can handle more complex tasks, so organizing ourselves in this way will prepare us for those tasks as they come.

## Core Values & Responsibilities

Combined notes on team expectations regarding communications and creating a productive, inclusive environment. How will you all communicate? What is a reasonable response time expectation? What are the team’s overall priorities? How will you handle disagreements? How, as a TEAM, will you make sure everyone has value-added work to do and has a “voice” on the project?

Ask yourselves & each other the following questions to help start this conversation:

* What styles/forms of communication are most effective for you?
* How do you prefer to handle conflict and how does this vary based on the topic/type of conflict?
* What would it take for you to feel included in the team? What will you do to help others feel included?
* Is there anything else you want the team to know?

As a team, we will primarily communicate through Discord chat, but meet in person when the opportunity arises. Preferably, all messages will be responded to within 30 minutes to an hour so we can all always be on the same page about our current progress. Our team’s priorities are to create a good technical foundation for the game with little to no bugs or glitches. Should any disagreements arise, we will resolve them through dialogue to find a healthy middle ground. If the argument escalates, we will turn to the professors for further discussion. Moreover, as a team, we will listen to and value each other's thoughts and opinions to get the most from this experience and make a game where everyone's input is valued and used.

## Risk Analysis

What could go wrong, and how can you reduce the impact if it happens?

Throughout the project, I’d like you to discuss & track potential risks. Use the table below to help (add rows as needed). I’ve entered a few that we suspect will impact all teams to help you get started. You should discuss and add at least 2-3 more.

|  |  |  |  |
| --- | --- | --- | --- |
| **Risk** | **Likelihood**  *What are the chances of this happening? (Usually low, med, high)* | **Severity**  *What would the impact be? (Usually low, moderate, high, catastrophic)* | **Mitigation**  *How will you respond/minimize the impact of this?* |
| Time to meet synchronously will be limited | Med | Low: | We will communicate about the tasks each member will take |
| Someone might get sick or need to be offline unexpectedly | Med | Moderate: | We will let the member rest and will divide the work among the others |
| We are all students with other courses & commitments to consider | High | High: | We understand to dedicate our studies first(the professor stated 3-4 hours per week) |

## Planning & Scope

This section WILL change over time as you refine the game design. That’s fine. This is a living document!

### Priorities

Describe the overall priorities and rationale that guided your decisions about scope (this should add depth and context to the goals you introduced in [Expected Outcomes](#_60p2tpvt9ky5)). For example, for YOUR game, what is more critical: prototyping a variety of NPCs or doing one really well? Many levels or one representative level with a variety of mechanics? WHY are these things important?

At a minimum, your game this semester must include:

* Four main game states: A title screen/main menu, active gameplay, a pause screen, and a final win/loss screen
* A full game loop with a starting state where the player can leverage at least one core mechanic to progress towards an objective and a clear win/loss/game over condition based on that objective.
* Some element of the gameplay beyond asset files and basic player stats that is loaded from a file. (Teams most commonly make the level design and/or dialogue data-driven.)

Summarize, in terms of your concept, how your minimum scope this semester will include these features. (E.g. what will happen in the minimum gameplay mode, what is the one core mechanic, etc.)

Priorities

1) Make a functional game.

2) Make 5 different levels

3) Make appealing and clear to understand visuals

4) Make those levels have a balanced ramp-up in difficulty

5) Make an endless mode

Main States:  
- A main menu that can access the level select and score screens

- A level select screen that has a visual component to selecting a level and loads in a level for gameplay

- A gameplay screen that can pause, switch between sub-screens, and allow you to go back to the main menu from the pause screen or level select when you win or lose.

- In the order sub-screen, take orders from available customers and add them to the customer list.

- A score screen that updates and displays the high scores for each level, with a visual component for selecting levels.

### Target Scope

Going beyond the bare minimum required for the class, what is your target scope for the semester? Why are these things important? (You won’t be held to these, and they can change, but it’s important to identify them anyway.)

1) Make a functional game.

2) Make 3 different levels

3) Debug any errors

4) Leave the project in a good state for further development.

### Stretch Goals

List any features that you want to accomplish but weren’t deemed essential. (Effectively future work section). In the postmortem, close the loop by discussing more how your concept evolved and priorities changed (& why!)

1) Higher visual polish, like particle effects

2) An Endless Mode

3) Vague orders and recipes to introduce different types of challenges

4) More levels to play

## Task Management

Having a game concept so you know what you want to implement is crucial, but it won't be enough unless you also break it down into tasks that individuals can claim and work on. Tasks will eventually need to be prioritized based on your desired scope (i.e., the goals you defined above) and the dependencies between them.

Given the priorities and target scope + your team organization and communication strategies, how are you tracking what needs to be done? How do you know who is doing what and when? How are dependencies managed and progress assessed?

**I STRONGLY recommend using a task board such as Trello to support your task management process. See** [**https://trello.com/b/FYbymkCg/template-gdd-project-board**](https://trello.com/b/FYbymkCg/template-gdd-project-board) **for a template board.**

We are tracking what needs to be done through the Trello board, which we update based on the tasks needed for the week or the sprint. Furthermore, in Trello, we have a different section where we can track when a task has been completed, or is in progress, and what hasn’t been started. Here, we can pick up tasks and begin working on them throughout the week or sprint to ensure everyone has an opportunity to contribute to the project. Moreover, we will know who is working on which tasts by assigning ourselves to the tasks with our names and current progress. Our progress will be assessed on the progress made through the task card and the code pushed onto Github.

## Asset Pipeline

**Do NOT waste time on excessive amounts of custom assets! These will NOT help your grade! Free assets & basic geometrics are fine!**

Summarize how you store, manage, and incorporate assets into your game. Include any strategies for finding or creating assets here as well.

Currently, all assets have been made in Krita, a free art software. Assets are stored in the Content folder because it is finicky to change which folder MonoGame is reading the content from.

# Game Design

The final game design document (GDD) is a written explanation of how the proposed game actually works. The GDD is “living” specification of your game—another team could make the same game using your GDD. This should be the complete design document. At the beginning of this section, you should give a short summary of the entire design for those that don’t want to read the gory details of the game design.

Add concept mockups and eventually game screenshots throughout this section.

This is the big picture - not just what you implemented. Include WHY you made these decisions, not just a report on what the game is. This is not a user manual! Technical and functional will not grab our heartstrings. What’s the hook?!

Explain your game to potential players, investors, etc. - NOT to developers. Yes, describe your mechanics, but do it in the context of the game concept, theme, etc. All of this should be specific to YOUR game. For example, for a 2d platformer, this section shouldn’t be applicable to any platform, just YOURS.

Details about how mechanics are implemented, etc. go in technical design. This is about objectives of mechanics, level design, etc.

**THIS is where you provide the details about how your game will achieve the goals laid out in the introduction.**

## Game Overview

Early in the project, you should define the core concept that will drive your design activities for the rest of the semester. Make sure to cover what will make this game fun!

What does the player control (an avatar, puzzle blocks, …), and what is their goal and motivation? What is the “win” condition/level? What NPCs are present in the game? How does the player interact with them, and what is the result of these interactions?

What is the context for the game? Main story/characters? Setting?

Much of this should have already been covered lightly in the Introduction (so that the reader had context when reading the Production and Analysis chapters). Here is where you expand on that and paint a **full** picture of what it will like to play your game. The rest of this chapter then details how the design will help you achieve those goals.

Minerva’s Marvelous Mixturette is a game where you play as a witch, Minerva, selling potions out of a horse-drawn food cart. The game is intended to feel silly, leaning on classic spooky (not scary) tropes. When else will you be able to serve a potion made of babies and nightmares to The Creature From the Black Lagoon? Over the course of three levels of increasing difficulty, serve customers quickly and accurately to get a high score. To beat a level you have to beat a threshold score. In the cart, there are four stations. In the Order station, you take orders from customers. In the Mix station, you add ingredients to an order. In the Brew station, you add orders to a cauldron to cook them. Lastly, in the Finish station, serve orders to the customers. You can pass orders between these stations, modifying them one at a time.

NPCS:



## Audience & Genre

Why did you target the audience you did? How does the game design meet their needs? -- From the perspective of attributes of the audience and your game. Define the audience itself and how to address their interests/needs using relevant references. How did you choose this genre, what evidence/reference supports this decision, etc.?

We chose to target people who enjoy fast paced and skill based games, especially those who played Papa’s pizzeria and Cook, Serve, Delicious since our games follow similar concepts and gameplay on these games but with our own twist. The game design meets their needs because in this project we aspire to create a game where people can have the opportunity to create various potions for various witches and wizards in the game’s world. Our game is similar to the games listed above, as you make various items such as food either pizza or ice cream. We chose this genre because Bee suggested this idea and as a team, we agreed on this idea with the various references from other similar games like Papa’s Pizzeria and Cook, Serve, Delicious.

## Inspirations

What other media did you reference? (games, films, etc.) This is meant to help draw out design knowledge and technique that is apparent and relevant to your work, as well as ensuring your game is not simply retreading the work of other game developers.

### 3.3.1 Aesthetically Similar Games

These are games that strongly facilitate your design goals, such as player experience, visual aesthetic, and other design traits that may be applicable to your game. Major elements, such as overall genre can be omitted, instead try and focus on the specific elements of the experience provided that would make these inspirations relevant.

|  |  |  |
| --- | --- | --- |
| **Title** | **Aesthetic** | **Relevance** |
| Fancy Pants Adventure | Hand-drawn, cartoony, colorful. | The wacky, rough art in this game is fitting with what we had imagined. |
| Pizza Tower | Hand-drawn, absurd | The almost slapstick nature of animation in this game and their use of particles fit with the strange setting and humor of the game. |

### 3.3.2 Mechanically Similar Games

This section is for games that use mechanics and systems that might be similar to your game. Games that you can reference as proof-of-concepts and measuring sticks when building your own systems. Make sure to note why each mechanic referenced was relevant, and why the specific game was chosen over another game that might have a similar mechanic.

|  |  |  |
| --- | --- | --- |
| **Title** | **Mechanic** | **Relevance** |
| *Cook, Serve, Delicious!* | Repetitive keyboard inputs as a way to cook food | You use keyboard input to create potions in our game |
| *Papa’s Pizzeria* | Creating food by adding ingredients and performing operations across multiple different screens | This is the same way that you create potions in our game, but with less of a focus on precision and more of a focus on speed. |

### 3.3.3 Other Media

Inspiration for games can come from anywhere, this section is for any additional references you may come across, whether it be stories, movies, images, etc.

|  |  |
| --- | --- |
| **Title** | **Relevance** |
|  |  |

### 3.3.x Additional Criteria

If the team decides to narrow the scope for references further, include additional tables with a brief explanation to contextualize them.

## Narrative Design

If applicable, provide an overview of the grander narrative content of the game: who, what, where, when, why, and how? This is NOT just a story or written dialogue. The goal here is to describe how narrative supports your design goals. Story details can go into an appendix if needed.

* **How does the narrative provide context and support the player’s experience & progression toward the objective?**
* In what ways does the player interact with this narrative? Is that separate from how the player’s character experiences the narrative?
* Is the narrative static, or dynamic? Additionally, explain the reasoning behind your narrative decisions.

There is no “real” narrative to the game, other than the implied narrative of a witch just trying to get by and running her own small business. The narrative would detract from the main focus, which is the moment-to-moment gameplay.

## Gameplay

### Player(s) & Player Objectives

What does the player control (an avatar, puzzle blocks, …), and what is their goal? What is the “win” condition/level?

The player controls the ingredients, adding or removing, mixing, and cooking the potions, trying to get the customer’s order perfect. The player has to make each customer of that level’s order as quickly and accurately as possible to gain points and get a certain number of points to beat a level. Points are based on the number of correct ingredients, the accuracy of the brewing time, and the time remaining when the order is completed. There are five levels of increasing difficulty, meaning there are more complex orders and less time to complete each order.

### Player Interactions

Describe the player controls in detail. How does the player progress through the game? What is the core mechanic?

All game controls are by the keyboard. To move across screens during gameplay, the A and D keys are used to move left or right. Tab is used to view different customer’s orders during gameplay on any screen. Each ingredient has a letter key attached to put that ingredient into the current potion, and these keys are displayed beneath their respective ingredient on the shelf. Space is used to move the orders between the various stations. We assigned keys to certain cauldrons which are displayed on the cauldrons themselves for brewing the potions. Various keys move between main menu options with instructions on the screen, and number keys are used to select a level on both the level select and score screens.

### Levels & Environment

What is the environment in which the player interacts? Are there distinct levels? What can the player interact with & how?

There are five distinct levels of increasing difficulty. The player interacts with the game menus and in-game interacts with the elements via key presses. The player can use keys to select ingredients to add or remove to a potion, move the order and themselves across screens, brew the potions, and give the potion to the customer who ordered it. The four screens during gameplay that the player can interact with and toggle between are ordering, mixing, cooking, and finishing. The actions for each screen include receiving a customer’s order (ordering), adding or removing ingredients from the potion (mixing), cooking the potion in a cauldron (brewing), and giving it to the customer (finishing).

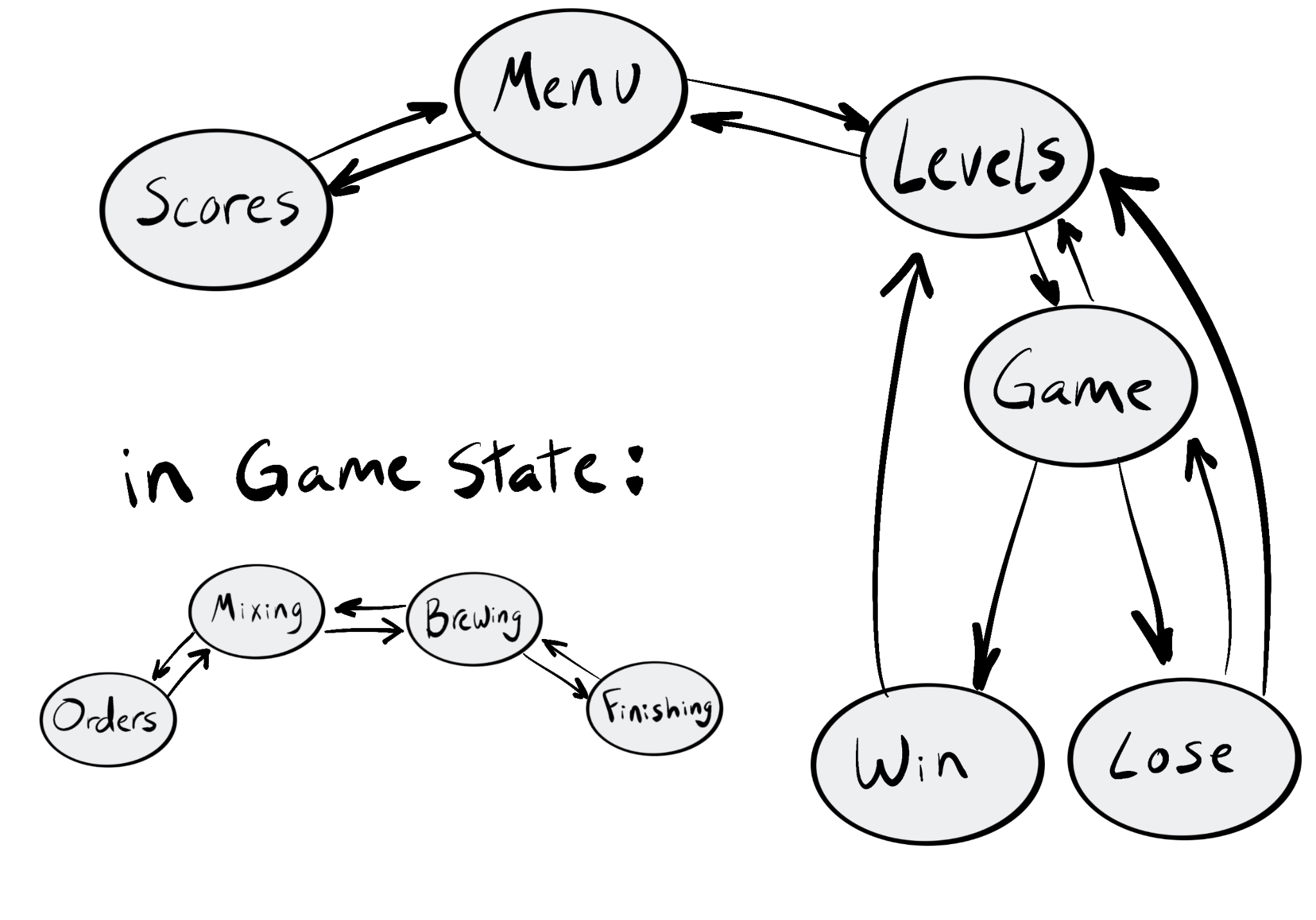
### Non-Playable Characters *(if any)*

What NPCs are present in the game? How does the player interact with them and what is the result of these interactions?

The customers are all non-playable characters who walk up to the shop at various times throughout a level and give their order to Minerva (when she, the player, is on the ordering screen). Each customer has an order with specific ingredients, cooking times, and a patience level (time that the player has to complete their order in). Customers will remain outside the wagon (on the finishing screen) until they receive their order or run out of patience. If the player can complete their order before their patience runs out, they are rewarded with points. If they don’t complete the order on time, they lose the level.

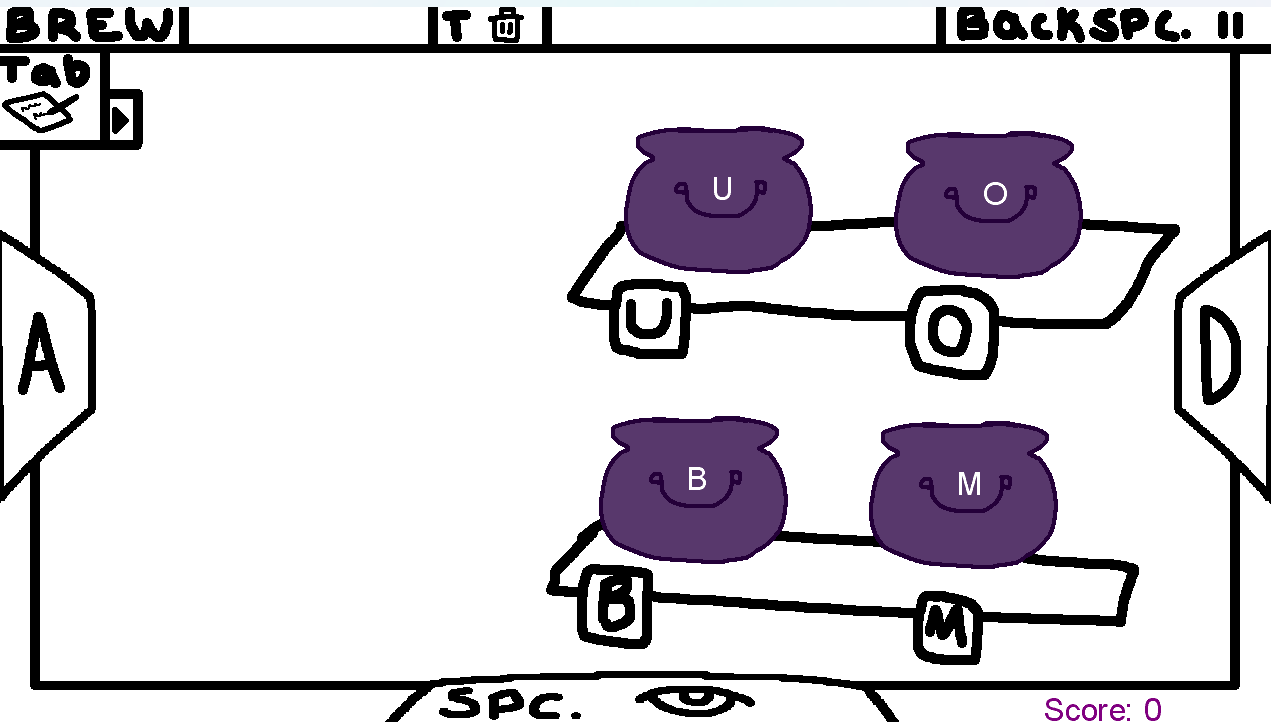
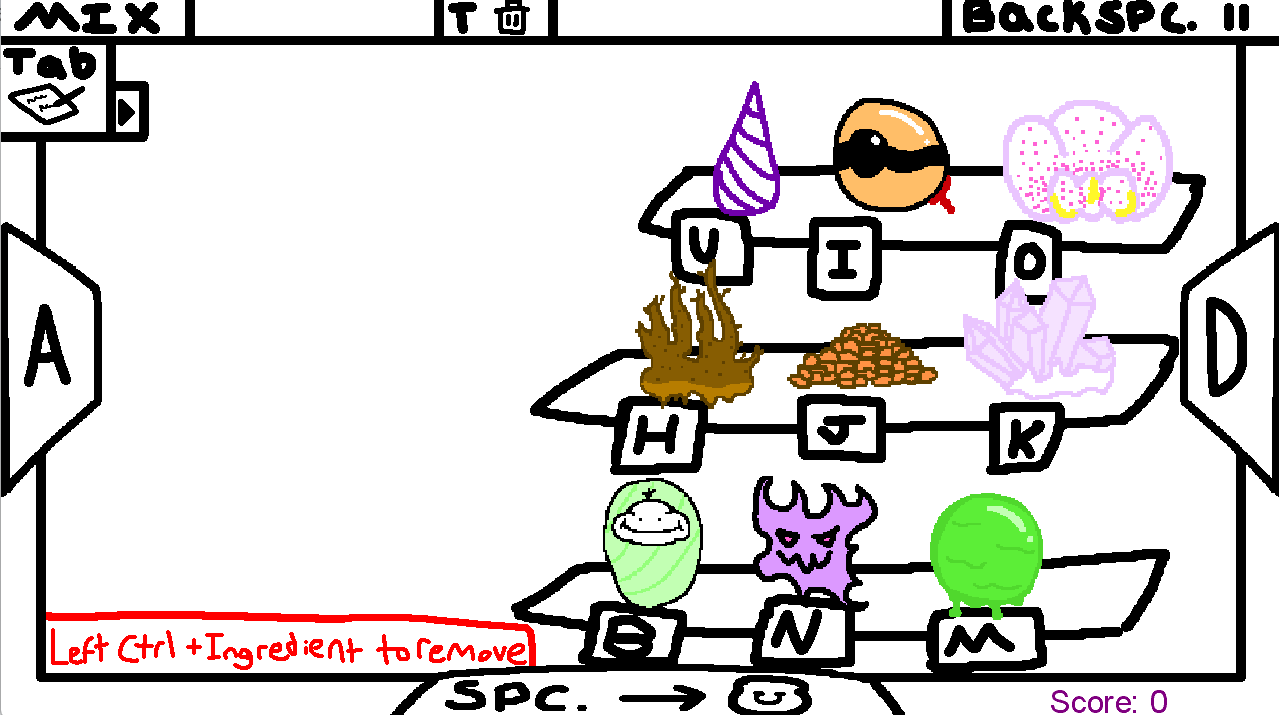
### Game Modes

Describe AND provide a state machine diagram for the key game modes.

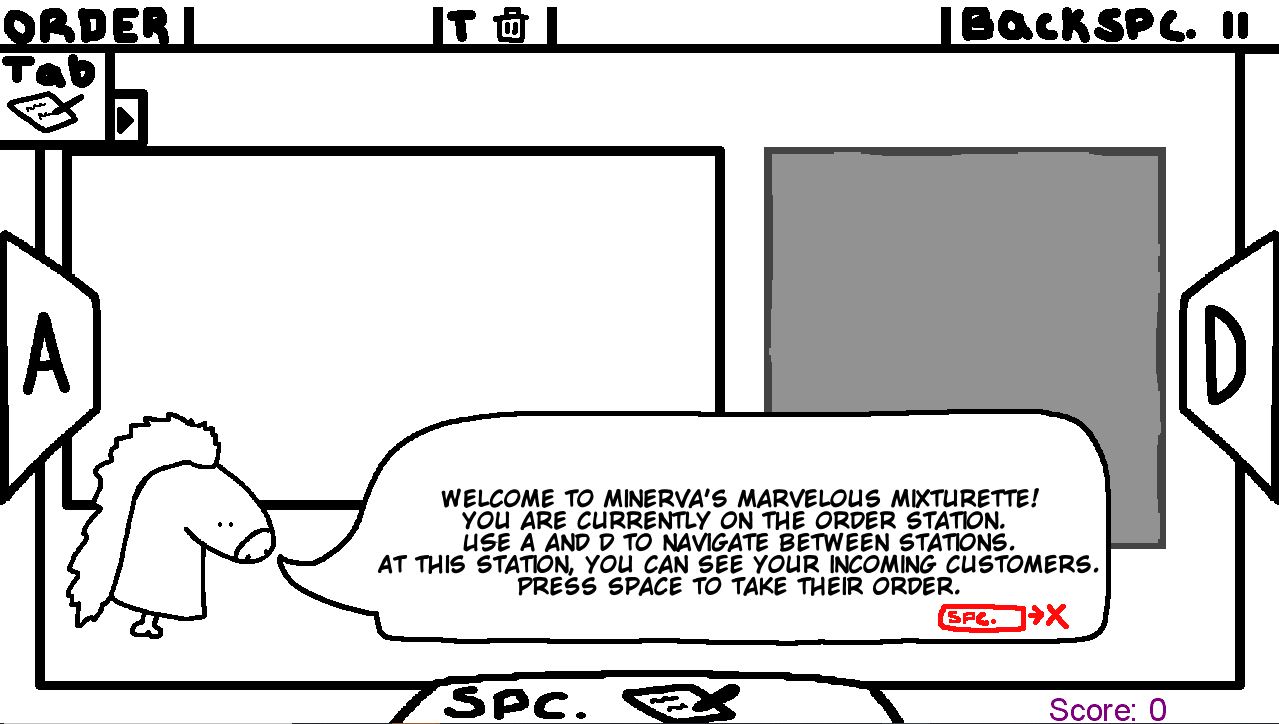
Here in the diagram, the key game modes would be menu, score, game, and level since these will be seen commonly by the player throughout their gameplay experience and formatted the various end game states, either win or lose. In addition, the smaller modes of orders, mixing, brewing, and finishing within the game mode.

## Aesthetics

Mood, feel, inspirations, how it fits with the game, etc. PLEASE use visuals, etc. to SHOW the aesthetics. Let us see what you’re making - screenshots and a few sentences about how this shows how these aesthetics support the game design.







The game is designed to be silly and absurd, so the art style was made to convey those aspects through simple yet comical designs. We also wanted players to be able to know what the controls are at all times, so button prompts are placed near areas of interest, such as the different ingredients and the screen selection to aid with visual recognition.

## User Interface

What are the major game states? How does the player get information about the game status? Visual, audio, controls (not details, but what can the user control? Timing? Precision? ...) -- things that matter in terms of how the player interacts with the game. Feedback to the player as well - interfaces go two ways!

The player learns how to navigate the main game states (i.e. the main menu, scores screen, level screen, and gameplay) through on-screen directions instructing them on what keys take them to what screens (of the ones they can access from their current screen). Within the gameplay state, the player is instructed on how to switch between the states within the gameplay (ordering, mixing, brewing, and finishing) and how to return to the main FSM (via the pause menu). Since the game is played using a keyboard only, less precision is required than using a mouse, but errors (missing a key/pressing the wrong one) are still possible, especially when moving quickly to complete orders on time.

## Onboarding

How is the player introduced to the objectives, mechanics, etc. IN GAME?

The game has indicators for every action available, and gives an in depth tutorial on your first time through.

# Technical Architecture

This is your technical design document. No code should be in the document unless there is something incredibly clever/core to the implementation of the game.

Even in technical design, the language here should be in the context of your game! Avoid jargon, etc. As said by a past IGME-601 student, **frame everything in the context of "Architecture in Service of Gameplay":**

I want to see how each team considered the architecture of the game and how it could be best designed to deliver gameplay that lent itself to the larger vision of the game.

* Discuss the evolution of the code and how the design decisions influenced gameplay.
* When the reasoning for decisions is discussed, explain how this architecture is best suited to make the game fun for players and show off their skillsets vs solely referencing ease of development and implementation (which is still very important!)

...

Without this, how the design choices should best translate to technical design aren’t obvious, so knowing how best to implement any given mechanic seems like it might be a personal judgment call rather than something that would be derived from the game design.

**Focusing on how the technical aspects are in service of the game itself:**

**1. Motivates why this technical work is important.**

**2. Keeps the non-technical members of the audience engaged (and even the technical ones - because tech alone is really dry.**

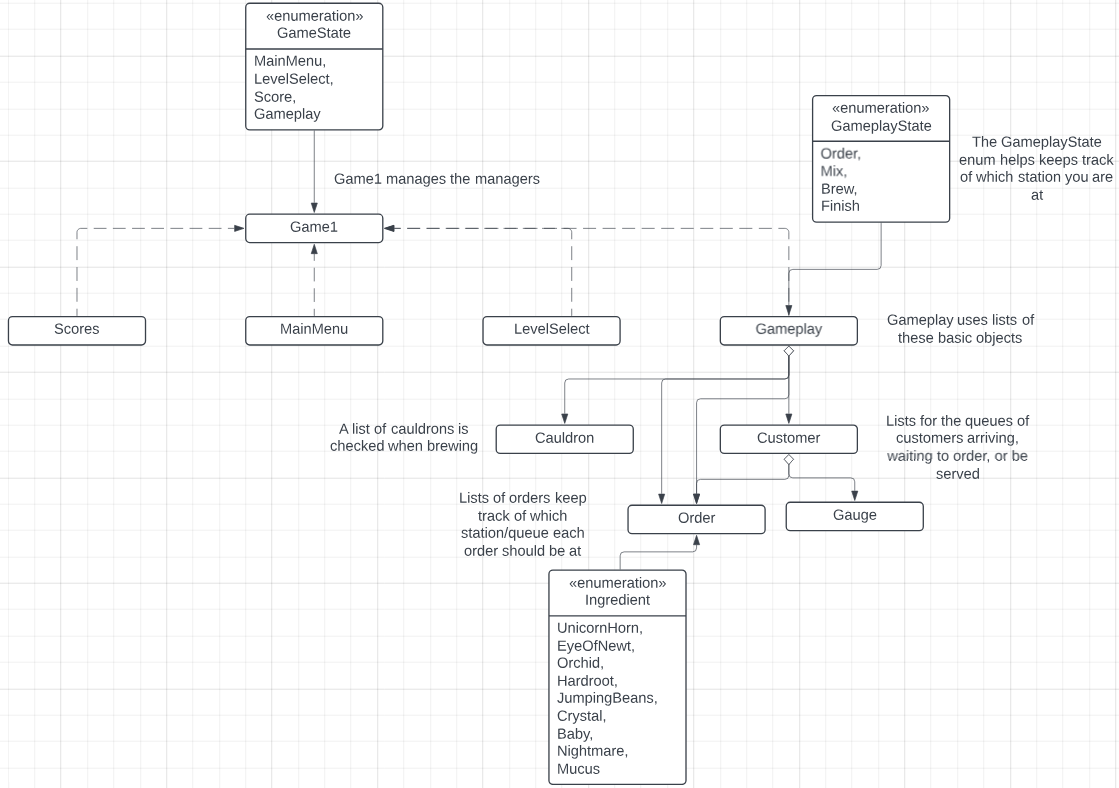
**Do NOT include code screenshots or long lists of classes/attributes**. We can see all that in the code if needed. The key here is to provide the context that would be needed to understand those details. Things that are appropriate at this level include high-level UML class diagrams, state machines, custom file formats, etc., and a discussion of WHY/HOW you are using them.

## Overview

Discuss, in plain terms, your overall approach to technical design for the game. Think of this as a summary of the rest of this section. This is NOT the entire OO design. The focus should be on important information needed to understand the structure of your technical design. Make sure to cover the fundamental choices you made in the development process of the game. Start with a big-picture view here and then organize/describe your technical approaches and decisions for each subsystem below. Major questions to answer in a general sense here include:

* What major data structure, patterns, and algorithms did you leverage? Why? (Mostly what and why here. Save the details about “how” for the subsystems discussions.
* Will there be some common base classes that are extended by different entities in your game?
* Will you need any abstract classes and/or interfaces?
* How will input be handled?
* How will the gameplay be data-driven?

Use class diagrams as needed to show the major OO relationships, but there’s no need for details about fields and methods within classes at this level unless those details are truly critical for understanding how things work together!



Game1 acts as a manager for all of the game’s major systems, drawing and updating using methods from classes representing the main screens. Many of the game's elements are separated into bespoke parts, making it easier to edit the code and find where errors are happening while debugging. Gameplay handles the actual gameplay loop, using regions to separate its major methods and bundle code into understandable chunks. The different states within gameplay are handled by an Enum and a switch statement based on that Enum.

Input and changing states are both managed by delegates, calling functions from Game1. Originally, this was done to reduce how many times the KeyboardState was called and to make input simpler, but it was ultimately unnecessary and would have been easier to just write a SingleKeyPress and HeldKey function in each class. The ChangeToState delegates were useful to switch between the different GameStates easily, but this could have been handled by just passing Game1 into each of the manager classes.

Originally, a GameObject class was inherited by Customer and Order, but they did not end up having enough shared functionality to justify this in the end. Eventually, GameObject was un-implemented.

The body of the game revolves around moving and modifying Orders, and tracking Customers. The stations, represented as GameplayStates, handled deciding what happens to Orders or Customers on a given screen. Queues between stations handle where orders should be and in what order they are in. Three lists keep track of customers, one for before they enter, one for when they are ready to have their order taken, and one for when they are ready to receive their order.

Levels are loaded from a text file, with one for each level. Each level consists of a list of customers and information about those customers, including their orders, the asset used, their patience level, and when they arrive.

Customers have an order and “enter” at a certain time after the level starts. After a certain amount of time, they leave. When a customer leaves, you lose the level. You can score points by delivering their order to them before they leave, and you get more points for being accurate and timely.

Orders are a list of ingredients with a brew time, with a few properties and methods to ensure that they fit the “rules” (i.e. having only 3 of the same ingredients).

## Player Interactions

How, specifically, did your team implement the primary player interactions & mechanics described in [Player(s) & Player Objectives](#_rag39wh4fvck) and [Player Interactions](#_mya5ns9e81bn)? There will likely be connections between the OO design described here and other sections. Don’t repeat yourselves. Reference other discussions/diagrams as needed. Include class diagrams of the major classes that support these features.

Include some details about key public properties and methods via class diagrams, but there’s no need for details about private fields & helper methods. That will just clutter up the diagram!

The keyboard buttons allow the player to select different ingredients and toggle/move between different menus. We decided to emphasize quick and accurate button presses, leading to a higher score and acting as a visual guide for the player to figure out what to do to proceed in the game quickly.

Classes for the individual game states determine which controls can be used based on the current game state.

While all user inputs are processed through their respective classes' Update() method, the Gameplay class has a unique method (private void ModifyIngredientsFromInput). This is referred to during the Update() loop to lay out all the different ingredients that can be added or removed with the keys.

## Levels & Environment

How, specifically, did your team implement the creation and management of the levels/environment as described in [Levels & Environment](#_7e2wnis6uqfi)? There will likely be connections between the OO design described here and other sections. Don’t repeat yourselves. Reference other discussions/diagrams as needed.

Include some details about key public properties and methods via class diagrams, but there’s no need for details about private fields & helper methods. That will just clutter up the diagram!

Levels are loaded from a text file, as discussed in 4.5. Each level’s text file has the same template (outlined in a template text file), that includes the level number and the number of customers in that level. For each customer, each line of the file lists their number in order (i.e. the first customer says # 1), the asset used for their image, the ingredients in their order (separated by commas), the amount of time the order needs to brew, at what time they appear, and their patience level (how long will they wait for their order before you lose). This makes it simple to adjust levels and add new levels by creating and filling a new text file. The information for the current level is stored in Gameplay. The environment is the same across the whole game, being the series of backgrounds you see as you switch between the game’s stations.

## Game Modes & Menus

How, specifically, did your team implement the game state FSM defined in [Game Modes](#_b4e9rbjpjxt4)?

Include some details about key public properties and methods via class diagrams, but there’s no need for details about private fields & helper methods. That will just clutter up the diagram! There’s also no need to repeat the FSM diagram unless you have a different version that adds more technical details compared to the one in the design section.

In Game1, there is a GameState enum that controls what happens during its Draw() and Update() methods. Depending on the game’s state, it calls the Draw() and Update() from the relevant classes.

## Data-Driven Elements

Your game must allow the team to quickly and easily alter some core data that the game uses. This data will be stored in an external file and read in when the game starts.

### Overview

What features in the game will be data-driven?

The loading in of levels (the details of what orders appear for each level, with each level having its own text file), and the loading/saving of each level’s highest scores (in a text file for each level).

### File Format

Describe the file format in detail. What data does it contain? How is it organized? … There should be enough information here that we could use this section as specifications to write our own data file that works with your game. Include format, what characters are allowed, and what each character means. Also include an example small file with a matching screenshot.

Levels and scores are both text files. Levels are written from a template, shown below. The toppings were never implemented, but room is left for them in the future.

Scores are saved as a text file named for the relevant level and are simply the scores in descending order, separated by line breaks.

 <- The level template file

<- Level 1’s file and its customers.

### Game Integration

How does the OO design of the game itself support the loading of these files? What aspects of the gameplay are impacted? What classes are involved? Again, use diagrams and descriptive text.

Include some details about key public properties and methods via class diagrams, but there’s no need for details about private fields & helper methods. That will just clutter up the diagram!

The text from the levels is loaded by the LevelSelect class, which then loads information into the gameplay class using its LoadLevel method.

## Gameplay Sub-States

**Copy/paste this section as needed** to describe any other key subsystems \*and\* how they interact with each other + how they helped you meet the objectives of the game.

### Overview

What elements of the game design are supported by this subsystem? What were the major motivations and decisions made in support of those goals?

Within gameplay, there are 4 “sub-states” representing the stations inside the food carriage.

- In the order station, orders can be taken from customers.

- In the mixing station, ingredients are added/removed to an order, and then the order can be sent to the next station.

- In the brew screen, the player can put an order into a cauldron, where it brews until the player takes it out.

- In the finishing screen the player serves orders to customers and receives a score based on how quickly they completed the order and how close the order was to being correct.

### Approach

How, specifically, did your team approach solving the technical challenges for this subsystem? What data structure, patterns, and algorithms did you leverage? Include some details about key public properties and methods via class diagrams, but there’s no need for details about private fields & helper methods. That will just clutter up the diagram!

The gameplay states are managed with an enum within Gameplay. It functions similarly to how Game1 manages the main game states, except without classes representing the smaller substates. In Draw() and Update() a switch statement determines what happens based on the sub-state. To reduce clutter, the functions of each station are broken down into methods.

## Order Passing

**Copy/paste this section as needed** to describe any other key subsystems \*and\* how they interact with each other + how they helped you meet the objectives of the game.

### Overview

What elements of the game design are supported by this subsystem? What were the major motivations and decisions made in support of those goals?

The passing of orders between stations is necessary for them to get modified on each station, and allows the player to have more control over what happens to an order. It also supports the ability to manage multiple orders at once.

### Approach

How, specifically, did your team approach solving the technical challenges for this subsystem? What data structure, patterns, and algorithms did you leverage? Include some details about key public properties and methods via class diagrams, but there’s no need for details about private fields & helper methods. That will just clutter up the diagram!

The team approach on this subsystem was implementing a queue system for passing the orders between the stations. Furthermore, the mix, brew, and final stations have individual queues responsible for holding the order and are able to move between the stations. In the mix station, the user can place the ingredients for the order and can pass it (remove from queue) onto the next station (add to new queue) to continue to add upon the order. After placing all the items into the order, they can be placed on the final queue where the user can give it to the customer.

## Cauldrons

**Copy/paste this section as needed** to describe any other key subsystems \*and\* how they interact with each other + how they helped you meet the objectives of the game.

### Overview

What elements of the game design are supported by this subsystem? What were the major motivations and decisions made in support of those goals?

The brew station needs the cauldrons to function, allowing the player to brew orders. There are 4 cauldrons that function identically, so a class was made.

### Approach

How, specifically, did your team approach solving the technical challenges for this subsystem? What data structure, patterns, and algorithms did you leverage? Include some details about key public properties and methods via class diagrams, but there’s no need for details about private fields & helper methods. That will just clutter up the diagram!

The cauldrons are simple, holding an order and having an associated key. When the key is pressed, it will either add the current order to the cauldron or remove the order inside and pass it to the next station.

## Winning and Losing

**Copy/paste this section as needed** to describe any other key subsystems \*and\* how they interact with each other + how they helped you meet the objectives of the game.

### Overview

What elements of the game design are supported by this subsystem? What were the major motivations and decisions made in support of those goals?

The game needs a win and loss state, rather than just kicking you to the Level Select whenever an “end” state is reached, to communicate to the player if they won and what their score was.

### Approach

How, specifically, did your team approach solving the technical challenges for this subsystem? What data structure, patterns, and algorithms did you leverage? Include some details about key public properties and methods via class diagrams, but there’s no need for details about private fields & helper methods. That will just clutter up the diagram!

All customers are checked to see if they are out of patience, and if one is then the player is taken to the loss screen using a boolean. The game stops updating and you can return to the level select screen. When all of the customers are cleared, the game takes you to the win screen using a boolean. On the win screen, you can see your score and return to the level select screen.

## Score system

**Copy/paste this section as needed** to describe any other key subsystems \*and\* how they interact with each other + how they helped you meet the objectives of the game.

### Overview

What elements of the game design are supported by this subsystem? What were the major motivations and decisions made in support of those goals?

The score system is based on how accurately the user completed the order. The overall implementation behind this system was to give progression for playing the level and act as a state change to proceed into the next level.

### Approach

How, specifically, did your team approach solving the technical challenges for this subsystem? What data structure, patterns, and algorithms did you leverage? Include some details about key public properties and methods via class diagrams, but there’s no need for details about private fields & helper methods. That will just clutter up the diagram!

The design behind this system is the use of the order class and customer class to compare the correctness of the order within the final queue. This system is contained within a method that is called once the player gives an order to the customer. The method compares the ingredients, brew time, and the remaining patience based on the current customer in the queue. Each element of the order has a different amount of points and an additional boost based on the remaining patience left on the customer. There are some places where the scoring is forgiving. At the end of the method, returns the final score to the total score.

# Sprint Reports

*The general sprint goals that apply to all teams, as described in the project details in myCourses, are also pasted in the next steps sections below for your convenience. Be sure to consider these when planning since these are what will be considered when grading sprint updates & assessing if the project is on track overall.*

## Sprint #1

**General sprint 1 goals for ALL teams**: Form your team and establish how you will communicate and set up the infrastructure you'll need to stay organized .Start conceptualizing and setup the MonoGame project. The game should launch with at least a basic title screen by the middle of sprint 1. Establish your game concept, gameplay, and a prioritized feature list. Once you have this, you’ll make initial art/UI mockups and start your technical design (state machines, classes, etc.). Implement the core FSM to support all game modes and start implementing major systems.

Team Progress

Describe the progress of the game in an overall sense including rationale for any changes in directions, etc. We have the your task boards & GitHub repos for status reports. This should be a reflection on the progress as a whole, NOT a bulleted list of completed tasks! If you didn’t quite meet the goals defined in the last sprint’s “Next Steps”, that’s okay. Explain why and what progress was made instead (e.g. had to pivot, unexpected challenges, etc.).

Individual Progress

In addition to overall progress, give a quick summary of what each team member worked on this sprint AND how it tied to the overall sprint goals. This should NOT be a detailed report or task breakdown! A sentence or two is enough. We have the overall progress already + the task board. This is just to give a general sense of what parts of that progress each person contributed towards.

|  |  |
| --- | --- |
| **Team Member** | **Sprint Progress** |
| Gina | Drew FSM diagram. Created score class and outlined functionality, including reading in each file, sorting scores from highest to lowest with only top five displaying, and created the text files for each level’s scores. Cleaned up some code in all classes to be cleaner and fit course coding standards. |
| Audrey | Worked on documentation. |
| Bee | Implemented major FSM states, level loading, Customer and Order classes. Created visuals for ingredients and customers. |
| Diego | Helping with the initial implementation of the project, creating the gameobject class and editing the level select. |

#### 

Lessons Learned

With each sprint, give a 2-3 paragraph summary of any lessons learned and how your experience in that sprint influenced your plans for the next. Be sure to cover these things from multiple perspectives: task management, communications, game design, technical design, implementation, etc.

We learned we needed to document as we go and to communicate better about tasks everyone was doing.

## Sprint #2

**General sprint 2 goals for ALL teams**: Complete a "bare-bones” version of your game. This means the game should start up, and there is some core functionality present, such as character movement, game states, a menu system, etc. It is a good idea to have code stubs for most components in addition to working implementations of core functionality. Start designing and working on the data-driven features.

Team Progress

Describe the progress of the game in an overall sense including rationale for any changes in directions, etc. We have the your task boards & GitHub repos for status reports. This should be a reflection on the progress as a whole, NOT a bulleted list of completed tasks! If you didn’t quite meet the goals defined in the last sprint’s “Next Steps”, that’s okay. Explain why and what progress was made instead (e.g. had to pivot, unexpected challenges, etc.).

We created a system to help keep track of the current order in which they are working, which can be added to the many stations and select an order to take and be seen through the stations. We made the feature toggleable to see the current order from the customer to reduce the number of UI elements on the screen.

Individual Progress

In addition to overall progress, give a quick summary of what each team member worked on this sprint AND how it tied to the overall sprint goals. This should NOT be a detailed report or task breakdown! A sentence or two is enough. We have the overall progress already + the task board. This is just to give a general sense of what parts of that progress each person contributed towards.

|  |  |
| --- | --- |
| **Team Member** | **Sprint Progress** |
| *Diego Martinez* | We have created a system for orders to travel between “stations” or game states. Orders taken by the player can be seen and toggleable to reduce UI elements in the game. Additionally, prototyping a score system. |
| *Bee Morse* | Implemented basic functionality for all gameplay states, Customers, and Cauldrons. Updated visuals. |
| *Gina Piccirilli* | Added/implemented new fonts and added numbers/visuals for the level select and score screens. |
| *Audrey Ruiz* | Created logic for storing orders within cauldrons objects on the brew screen. |

#### 

Lessons Learned

With each sprint, give a 2-3 paragraph summary of any lessons learned and how your experience in that sprint influenced your plans for the next. Be sure to cover these things from multiple perspectives: task management, communications, game design, technical design, implementation, etc.

## Sprint #3

**General sprint 3 goals for ALL teams**: Completion of the bare minimum for all features.

Team Progress

Describe the progress of the game in an overall sense including rationale for any changes in directions, etc. We have the your task boards & GitHub repos for status reports. This should be a reflection on the progress as a whole, NOT a bulleted list of completed tasks! If you didn’t quite meet the goals defined in the last sprint’s “Next Steps”, that’s okay. Explain why and what progress was made instead (e.g. had to pivot, unexpected challenges, etc.).

We created a comparison method that compares the current order being made to the customer’s order, and will be rewarded points based on accuracy. In the meantime, it's the bare minimum as we will alter the points entirely based on other factors and balance from the team. Added new assets and began implementing them into the game. We added a feature where the player can take and view the order through the different stations. Fixing bugs pointed out in the latest playtest included fixing the end state based on loss of patience and proper end state logic. Refactoring the gameplay code into their respective sections based on the station and implementing a tutorial level for the player to learn the game basis. Altering the visuals and its code for the game to see for a more effective and readable program. Overall, the current condition of the game is meeting up with the current goal we have with the sprint but we are looking a bit into polishing and unnecessary code.

Individual Progress

In addition to overall progress, give a quick summary of what each team member worked on this sprint AND how it tied to the overall sprint goals. This should NOT be a detailed report or task breakdown! A sentence or two is enough. We have the overall progress already + the task board. This is just to give a general sense of what parts of that progress each person contributed towards.

|  |  |
| --- | --- |
| **Team Member** | **Sprint Progress** |
| *Diego Martinez* | I was responsible for the logic behind moving the orders between the stations and comparing the order with the customer’s order. |
| *Bee Morse* | Checked and refactored Gameplay to be easier to edit and understand using regions and splitting functions. Added winning/losing and a display for orders. Removed Overhauled visuals for all stations. |
| *Audrey Ruiz* | Provided some ideas for customer designs, mainly a potato-sack wearing customer and a tall customer wearing pantaloons. |
| *Gina Piccirilli* | Updated class diagram. Fixed level visuals/made only 5 instead of 10, tested loading in of scores with and without data in the text files, debugged. |

#### 

Lessons Learned

With each sprint, give a 2-3 paragraph summary of any lessons learned and how your experience in that sprint influenced your plans for the next. Be sure to cover these things from multiple perspectives: task management, communications, game design, technical design, implementation, etc.

As a team, we currently may not have a lot of time to dedicate to the project with the approaching tests and study for final exams, and will see a production decline as a result. But, we will coordinate with each other to leave this project on a good note and have time for ourselves and studies for final exams. As well as be better prepared for the final presentation with information based on the requirements and the feedback we got from them from the last presentation.

## Sprint #4

**General sprint 4 goals for ALL teams**: Polish the game via bug fixes and minor improvements. Time permitting, add extra levels or implement any of your "extra" features.

Team Progress

Describe the progress of the game in an overall sense including rationale for any changes in directions, etc. We have the your task boards & GitHub repos for status reports. This should be a reflection on the progress as a whole, NOT a bulleted list of completed tasks! If you didn’t quite meet the goals defined in the last sprint’s “Next Steps”, that’s okay. Explain why and what progress was made instead (e.g. had to pivot, unexpected challenges, etc.).

We made sure that the project met all requirements, completing the basic few features that we needed like pausing and god mode. We ended up scrapping the topping system because it was not in a finished enough state to complete by the end of the sprint.

Individual Progress

In addition to overall progress, give a quick summary of what each team member worked on this sprint AND how it tied to the overall sprint goals. This should NOT be a detailed report or task breakdown! A sentence or two is enough. We have the overall progress already + the task board. This is just to give a general sense of what parts of that progress each person contributed towards.

|  |  |
| --- | --- |
| **Team Member** | **Sprint Progress** |
| *Bee Morse* | Implemented tutorial, trashing, god mode, pausing, real win/loss screens, seeing orders inside cauldrons, and score saving. Overhauled the UI and did minor QoL changes. |
| *Gina Piccirilli* | Fixed level select/score draw method so selected numbers are larger and remove the smaller ones. Made the scores and text display in the correct position (implementing helper method), and fixed exceptions in class/debugged. Consolidated instructions in top left of each screen and changed from basic arial 12 font. Wrote and adjusted/tested all level text files to create and balance levels and make sure they are possible. Added text to win loss screens in the font used on other screens. |
| [Diego Martinez](mailto:dm7450@rit.edu) | Implemented the score system to display the score based on order performance, resolve bugs with selecting unreachable levels and scores. |
| *Audrey Ruiz* | Made a toggle for the brew screen so players could see what orders were in which cauldrons without needing to remember. |

#### 

Lessons Learned

With each sprint, give a 2-3 paragraph summary of any lessons learned and how your experience in that sprint influenced your plans for the next. Be sure to cover these things from multiple perspectives: task management, communications, game design, technical design, implementation, etc.

## Wrap-Up

**Wrap-up goals for ALL teams**: Present your final prototype & technical approach + Do a final proof of all project deliverables and work as a team to write a postmortem.

Team Progress

Describe the progress of the game in an overall sense including rationale for any changes in directions, etc. We have the your task boards & GitHub repos for status reports. This should be a reflection on the progress as a whole, NOT a bulleted list of completed tasks! If you didn’t quite meet the goals defined in the last sprint’s “Next Steps”, that’s okay. Explain why and what progress was made instead (e.g. had to pivot, unexpected challenges, etc.).

The game was mostly complete by the final playtest and especially by the final presentation, so we were focused on making sure our documentation was as good as possible and putting the finishing touches on our game.

Individual Progress

In addition to overall progress, give a quick summary of what each team member worked on this sprint AND how it tied to the overall sprint goals. This should NOT be a detailed report or task breakdown! A sentence or two is enough. We have the overall progress already + the task board. This is just to give a general sense of what parts of that progress each person contributed towards.

|  |  |
| --- | --- |
| **Team Member** | **Sprint Progress** |
| *Bee Morse* | Documentation, new art, code/game polishing. |
| *Gina Piccirilli* | Made visuals clearer and nicer so screens were easier/smoother for players to navigate. Created levels and balanced them for smooth gameplay of increasing difficulty. |
| *Audrey Ruiz* | Made a toggle for the brew screen so players could see what orders were in which cauldrons without needing to remember. |
| [Diego Martinez](mailto:dm7450@rit.edu) | Fixed and resolved bugs through the final production and documentation. |

#### 

Lessons Learned

With each sprint, give a 2-3 paragraph summary of any lessons learned and how your experience in that sprint influenced your plans for the next. Be sure to cover these things from multiple perspectives: task management, communications, game design, technical design, implementation, etc.

Communication is beyond key. It has been the biggest problem for our group, but in the last week / two weeks we tried to communicate as frequently as possible and it was very productive. Not much was done in terms of design or implementation, aside from smaller features, but testing and polishing really brought our game to the most finished state it could be.

# Postmortem

Every research report has two final sections: Conclusions and Recommendations/Future Work. This document merges these sections into the post-mortem, which is a common industry practice. In the last section, you discussed how the playtest results confirmed or denied what did or did not work about your game. This section extends that discussion by reflecting on why those responses happened for your audience and your team.

As part of this discussion, include what you learned (e.g., how to improve the team dynamics, the game, and the processes). Everything in this section should consider your ENTIRE project, both semesters, from multiple perspectives: game design, production process, technical design, implementation, ….

For each, cover: What went well? What didn’t? What impact did this have on the game and project as a whole? What did you learn as a result?

This is NOT about criticizing your entire project. It’s a candid discussion of strengths, weaknesses, and lessons learned -- Have you reflected on the entire experience? Do you understand what went well and what didn’t, and, more importantly, do you understand WHY that happened. This is the story of your journey with some hindsight, but without beating yourselves up.

**Write to future students! What would you have wanted to know from past teams so you could learn from their experiences?**

## Overview

Give an “executive summary” of the entire postmortem chapter.

Our communication and planning really needed work. Without setting frequent definite goals and consistent meetings, we were rarely able to get everything done that we wanted to for each sprint. Even though we would meet after classes, it was not the right environment for the project to really gain and keep momentum. If we met outside of class, we would likely have been more able to complete everything we had planned. Descoping slightly made our game a bit better by the end, but we still missed out on a lot of elements that would have made the game better, like the topping system or more thematic visuals. Aside from short term planning, longer term planning was also a bit missing. We had a lot of small ideas that were never recorded and we likely should have written out if they were ever going to be implemented.

Documentation was a big problem for us too, likely due to our inexperience with it and how it can be tedious to do and redo consistently. If we had kept it up to date better, perhaps we could have completed more features because it would have been easier to understand each of the systems.

Sprints frequently came down to the wire, often putting together presentations and final commits days or hours before they were due. This likely expounded on our problems with planning and communication, because these big rushes worked well enough despite being stressful and difficult.

## Reflections

Look back to what you wrote for each section above - you should probably be touching on all of the major topics/decisions you made and reflect on the effectiveness of those. Add/break apart subsections as needed!

### Production

Including team dynamics, etc. -- there IS a way to discuss issues here professionally. Talk to us about how! Remember, this will be ready by others! Content here needs to be from a team/process perspective, not about individuals!

As mentioned frequently across this document, planning and communication were problems for us. We did not meet in person outside of classes, and asynchronous communication was infrequent at many points. This led to a lot of sprints falling short of what we envisioned, even if they met requirements. The Trello board was used a lot earlier in the semester, to great effect, but as things started moving quicker it became a task list that was added to frequently but rarely cleared or properly maintained.

### Game Design & Aesthetics

Be sure to discuss more how your concept evolved, and priorities changed (& why!)

The design and aesthetics actually stayed very true to how we envisioned, aside from a small amount of descoping. We originally wanted a bit of a crude cartoonish look, and we stuck with it until the end. Even though we didn’t get to do the stretch goals we planned and had to scrap toppings, we almost exactly made the “basic” version of the game that we discussed at the very beginning.

### Technical Design

Not just what you did, but how these technical decisions impacted the game and project overall!

Dividing up the code made it a lot easier to work on different bespoke elements without worrying about affecting the entire codebase. Perhaps it was a little overboard because it resulted in manager files that didn’t do much in the end, like MainMenu, but it leaves room for improvement in the future. Refactoring the code was also huge for productivity because it made it much clearer what was happening and when. Easier navigation and understandability made it easy to test and implement changes.

### Playtesting

Reflect on how you conducted playtesting, not only the results from playtesting.

Playtesting really helped to determine our priorities when designing. We noted down everything everyone said, paying special attention to repeated complaints. Due to playtest feedback, we added new indicators, revamped the tutorial, and rearranged controls because confusion was relatively frequent before. Now, the game is much clearer at a glance. Playtests also helped us figure out a lot of the bugs that we didn’t find in testing. It’s really difficult to figure out what’s wrong with a game when you’re only doing what you have coded in as “right”.

## Conclusions

Discuss, overall, how the project went. State what you (or someone else) would (or should) do next to improve the game AND your process if you were to continue this project.

The project went relatively well but really required more effort on everyone’s part to bring it to the state we originally envisioned. If we were to continue, we would need more rigorous planning to make sure everything got done in a timely manner. Regardless, we managed to put in the work and create a playable game, which is not something that many people can say. Every part of the project was enjoyable in some fashion, and it is very cool to have pieces of a game that you can point to and say “I made that.”

## Future Work

Finally, assess the project’s readiness to move into production and document what would be required to do this. This section should, in effect, act as a mini-business plan. You can organize this section however you see fit (I want to make it engaging and structured appropriately for the current status of your game). However, you must cover:

* How are you going to evaluate and measure your future progress?
* What are the risks associated with your project, and what are your plans to mitigate these risks?
* What resources do you need to continue? (Consider this from a people, skill, hardware & software perspective.)
* What major milestones are you planning, and what is a rough timeline in which you think these can be accomplished (assuming access to the required resources)?
* What are your long-term plans for the project?
* What is your plan for distribution and marketing?

If you really intend never to touch this again, don’t just bail on this section. Explain why.

We had a few stretch goals that we had never gotten to implement, as well as planned features that hadn’t been implemented. Additional polish to make the game feel complete, like animations and sound, would make it a better portfolio piece. It would be difficult to actually continue as a group on this project unless we dedicated ourselves to resolving our problems with communication and planning, as there would likely be even less motivation to work on the project when we didn’t have deadlines assigned by our professors. Perhaps over time we could chip away at the stretch goals, but we are likely going to leave the game as it presently is.

If development were to continue, we would add each of the stretch goals to make our game go from demo to an actual product. We would implement the toppings and customer selection, add more colorful art as well as animations, create an endless mode, and put sound into our game. This would bring it in line with what we had originally envisioned, and would likely only take 2-3 sprints worth of work to complete because the game was programmed with some of these features already in mind.

If we were to continue, we would likely be able to keep going with everything we currently have. The game’s scope is pretty small and only uses Krita and Visual Studio, which are both free pieces of software. The codebase is simple enough that any one of us could decide to continue working on it, and our stretch goals have enough discrete features that it could be picked up and put down relatively easily.

We could advertise the game through a video development log and uploading clips to social media. If we really took the time to polish and expand the game, we could approach our professors about refining our pitch and taking it to MAGIC. If things were going unpredictably well, we could reach out to publishers to gauge interest, but only after refining our concepts and pitch to a near crystallized form.

Overall, while we are unlikely to fully realize this game, it has the potential to be a much more polished product without an extraordinary amount of work. With a plan and a little bit of motivation, it could be a sweet little flash-era-esque game.