IGME 106 Group Game Project

Spring 2235

Team: **Don’t Put Me On The Spot**

Game: **Topping Tumble**

Table of Contents

Team and Production

Student Information

Team Name: Don’t Put Me On The Spot

Game Repo and Presentation Slides

GitHub repo link: IGME 106 Group 3 | GitHub

Task board link: Topping Tumble | Trello

Links to Google Slides:

Milestone	*Google Slides*
1 - Team Formation & Conceptualization	https://docs.google.com/presentation/d/1uQPaF3aOaA3mGZI3ozn6hkd4iTDAFDJMM0izEISvGKU/edit?usp=sharing
2 - Game Loop	https://docs.google.com/presentation/d/1curBGSqxGf_XQY68hddKROX2Ox6OPgPtsyYbkzLdyOU/edit?usp=sharing
3 - Polish	https://docs.google.com/presentation/d/1uRZUog3vF5fqm6JDdR5_nubhVY6oWMFS3xND-grgCLc/edit?usp=sharing

Team Goals and Mission Statement

Discuss each of your individual and team goals for this project (beyond “get a good grade and pass the class”).

Individual Goals:

Team member name:	Their individual goals:
Nathan McComber	Gain experience with working on a complex project in a group, be able to be more flexible with my schedule, and better understand the processes that are in a professional workflow (documentation and presentations).
Gage Magar	I hope to build on my teamwork skills and learn how to communicate effectively on a team programming project. I’ve worked very few times with other people on the programming/building side of a game, so I hope for this to be a vital learning experience.
Jeremy Kotz	Learn how to work collaboratively on a game and improve my communication skills. Specifically, effectively communicating when I’m working on our project, what changes I’m making, and ensuring significant changes are approved by the team first.
Zach Jordan	I hope to gain more experience with gameplay and possibly backend coding. I also want to gain experience in working with a team on a major project and practicing effective communication. I also hope to learn more on game architecture and overall design.

Team Goals:

Goal #1: Create a product that is both playable and presentable

Goal #2: Improve our teamwork and communication skills

Goal #3: Gain a better understanding of programming architecture and project management

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.

- Our mission statement is to use teamwork as a tool for developing and completing presentable projects.

Group Norms

List your group's norms here:

- We will meet on Saturdays and Thursdays when available.
- If someone begins working on something (particularly code), they should notify the rest of the team and update that on Trello.
- If a group member won't be able to meet or work on something, they should notify the team ahead of time.
- Our programming norms will match that of the course. We will additionally name private fields prefixed with underscores.

Milestone 1: Design

https://docs.google.com/presentation/d/1uQPaF3aOaA3mGZI3ozn6hkd4iTDAFDJMMOizEISvGKU/edit?usp=sharing

Design

Give a brief description of your game:

- Our game is a puzzle-strategy game where you play as a chef guiding his ingredients to an oven. The ingredients move on their own, so you must manipulate the level around them to guide them to the end.

Answer the following questions:
How does it fit within this year's chosen theme?

- This fits within this year's theme "you can't make me" because the ingredients move on their own with no intent to reach the oven. You must guide them to the end without their cooperation.

What genre is your game?

- Our game can be classified as puzzle-strategy.

What are the objectives?

- The objective is to bring every ingredient in a level to the level end, which is an oven, to cook a pizza.

How many players?

- One player which guides a set number of ingredients each level.

What does the player control (weapon, puzzle blocks, etc.)? What is their goal and motivation? Why do they want to keep playing?

- The player controls the placing and erasing of tiles and certain objects, as well as "starting" the level which causes the ingredients to start moving. Their goal is to arrange the level in a way to guide the ingredients to the oven and progress through increasingly challenging levels.

How does the player progress through the game? What is the win condition?

- The player progresses through the game by guiding all of the ingredient characters to the end of the level. Once all of the ingredients have reached the oven, the player will progress to the next level.

Are there NPC's in the game? If so, how does the player interact with them? What is the result of those interactions?

- The ingredients the player guides would be NPCs. The player does not directly interact with them in any way. Their AI would be simple, moving in a given direction and turning around if they reach a wall. They can also walk off platforms and fall. As a stretch goal, we may have a "Live Chef Reaction" which interacts with the player and the game by responding to the state of the game.

What is the "control scheme" - what controls will the player utilize?

- The only controls the player will use are the mouse buttons. Left click will be used for most actions, such as placing tiles, starting the level, selecting tiles from the menu, and interacting with any UI or menu elements. Right click will be used to erase tiles from the level.

Did you draw inspiration from any games? If so, how is your game different?

- We are drawing inspiration from the games Lemmings, Mario vs. Donkey Kong, and Bridge Constructor Portal. All of these games involve guiding characters to a goal by changing the level. Although our game is similar in that sense, the manner in which you guide the characters will be different. In our game, the player will interact with the level by placing tiles and objects on a set grid to guide the characters.

Why is your game fun?

- Our game is fun because solving puzzles and successfully guiding all ingredients to the goal is rewarding, which is amplified by the increasing difficulty of the puzzles as you progress through them.

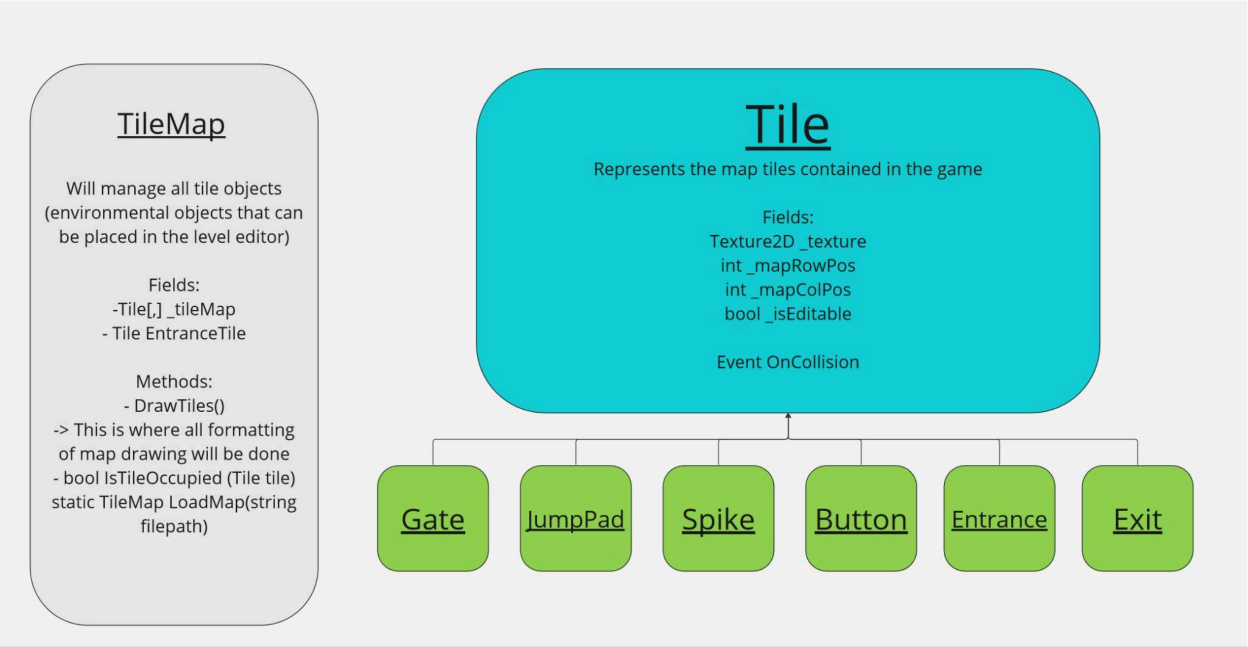
Scope

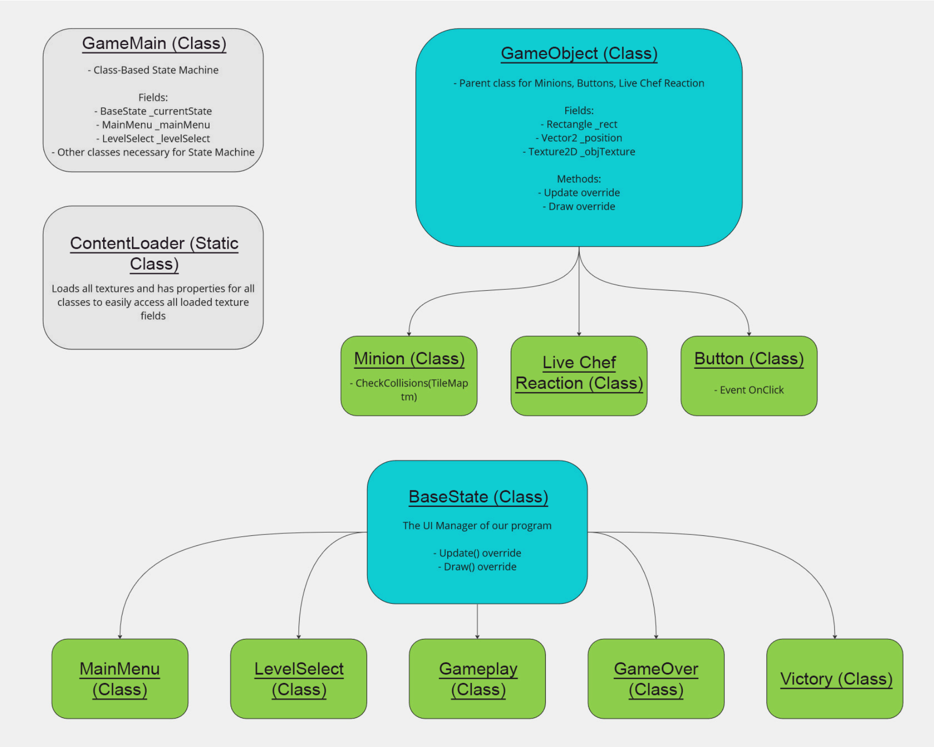
Describe your group's overall priorities and rationale that will guide your development priorities. *For example, for your game, which is critical: prototyping a variety of NPC's or showcasing one NPC that's done really well? Are many levels a key feature of your game, or is one superb level ideal?*

- At absolute minimum, we plan to have menus/states, the ability to place tiles, at least one type of ingredient functionality, the level-editor game state, the developer level-editor tool, and at least one level. However, ideally we should also have a few types of contraptions such as spikes, launchers, coins, and buttons. We also hope to have more than one level (potentially around 3 to 4).

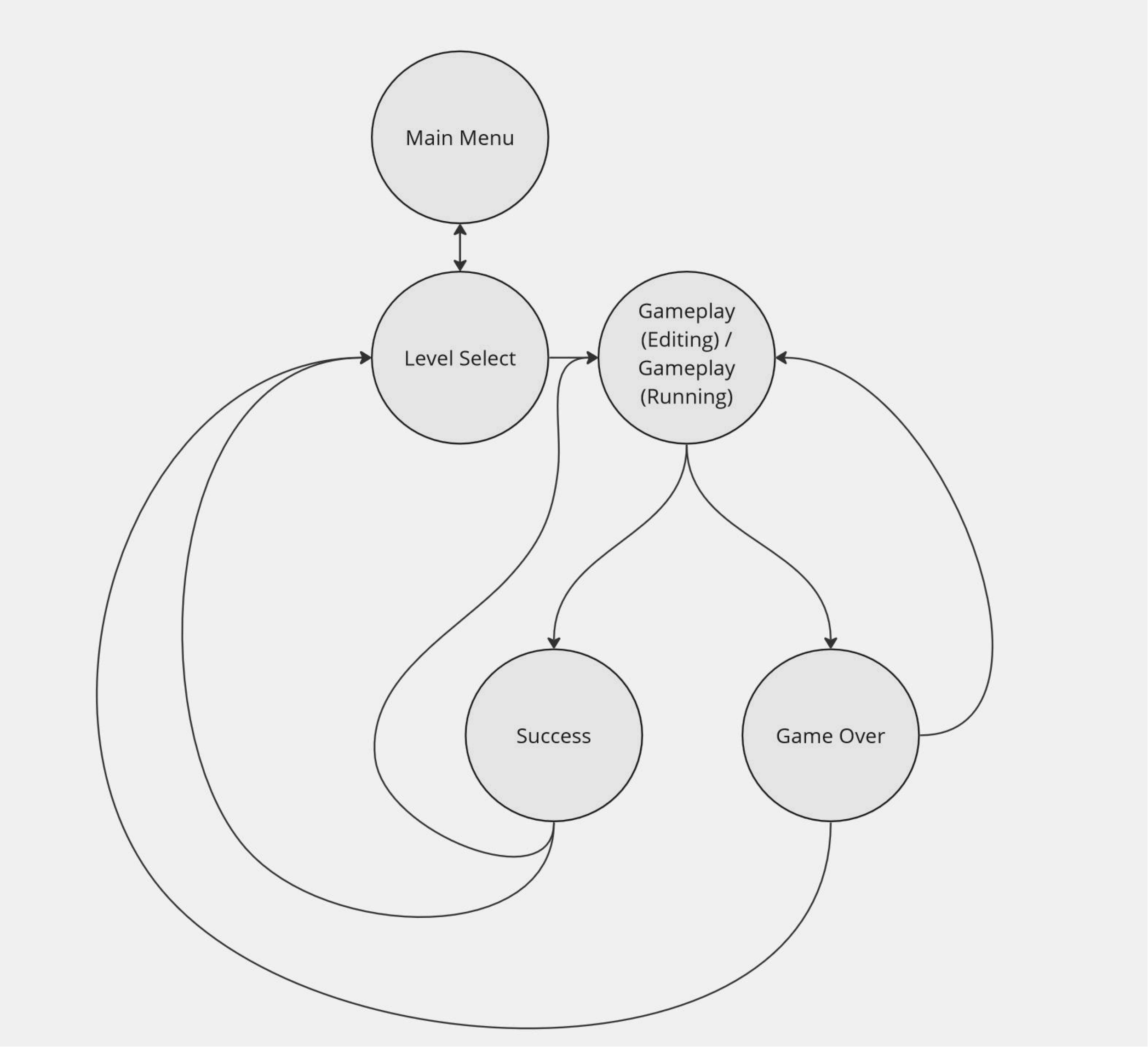
Architecture

Include your architecture diagram(s) here. If any classes are abstract or implement an interface, notate that on your diagram.


TileMap
Will manage all tile objects (environmental objects that can be placed in the level editor)
Fields:
-Tile[,] _tileMap
- Tile EntranceTile
Methods:
- DrawTiles()
-> This is where all formatting of map drawing will be done
- bool IsTileOccupied (Tile tile)
static TileMap LoadMap(string filepath)
Tile
Represents the map tiles contained in the game
Fields:
Texture2D _texture
int _mapRowPos
int _mapColPos
bool _isEditable
Event OnCollision
Gate
JumpPad
Spike
Button
Entrance
Exit


GameMain (Class)
- Class-Based State Machine
Fields:
- BaseState _currentState
- MainMenu _mainMenu
- LevelSelect _levelSelect
- Other classes necessary for State Machine
ContentLoader (Static Class)
Loads all textures and has properties for all classes to easily access all loaded texture fields
GameObject (Class)
- Parent class for Minions, Buttons, Live Chef Reaction
Fields:
- Rectangle _rect
- Vector2 _position
- Texture2D _objTexture
Methods:
- Update override
- Draw override
Minion (Class)
- CheckCollisions(TileMap tm)
Live Chef Reaction (Class)
Button (Class)
- Event OnClick
BaseState (Class)
The UI Manager of our program
- Update() override
- Draw() override
MainMenu (Class)
LevelSelect (Class)
Gameplay (Class)
GameOver (Class)
Victory (Class)

Include a finite state machine diagram of your game's states:



Describe which classes and features are data-driven. What will the file format be for those classes and features? *(File format means the type of data and structure of the data found in a file. For instance, a file describing enemy information could be serialized to include the enemy's name, level, starting health, preferred weapon, damage, list of objects in its inventory system, etc.)*

- The game's levels will be data driven and will be stored in external ".lvl" files. These files will be stored in a binary format that specifies the width and height in tiles of the level at the beginning. After the width and height, the amount of currency the player is given will be stored. The rest of the file will be a long sequence of integers for every tile, each integer represents a different type of tile which will allow for easy loading in the TileMap class.

What is the team's plan for an external tool?

- The external tool will be a level editor that can be used to set up initial tiles, the start/end position of the ingredients, and level-specific values. Level specific values could include how much currency the player is given to start and how many ingredients spawn.

Which classes handle multiple game entities (characters, enemies, NPC's, etc.)?

- The individual state classes manage any created game objects. The TileMap class manages all tiles currently in a level. The TileMap object will be managed by the gameplay state.

Where and how is player input handled?

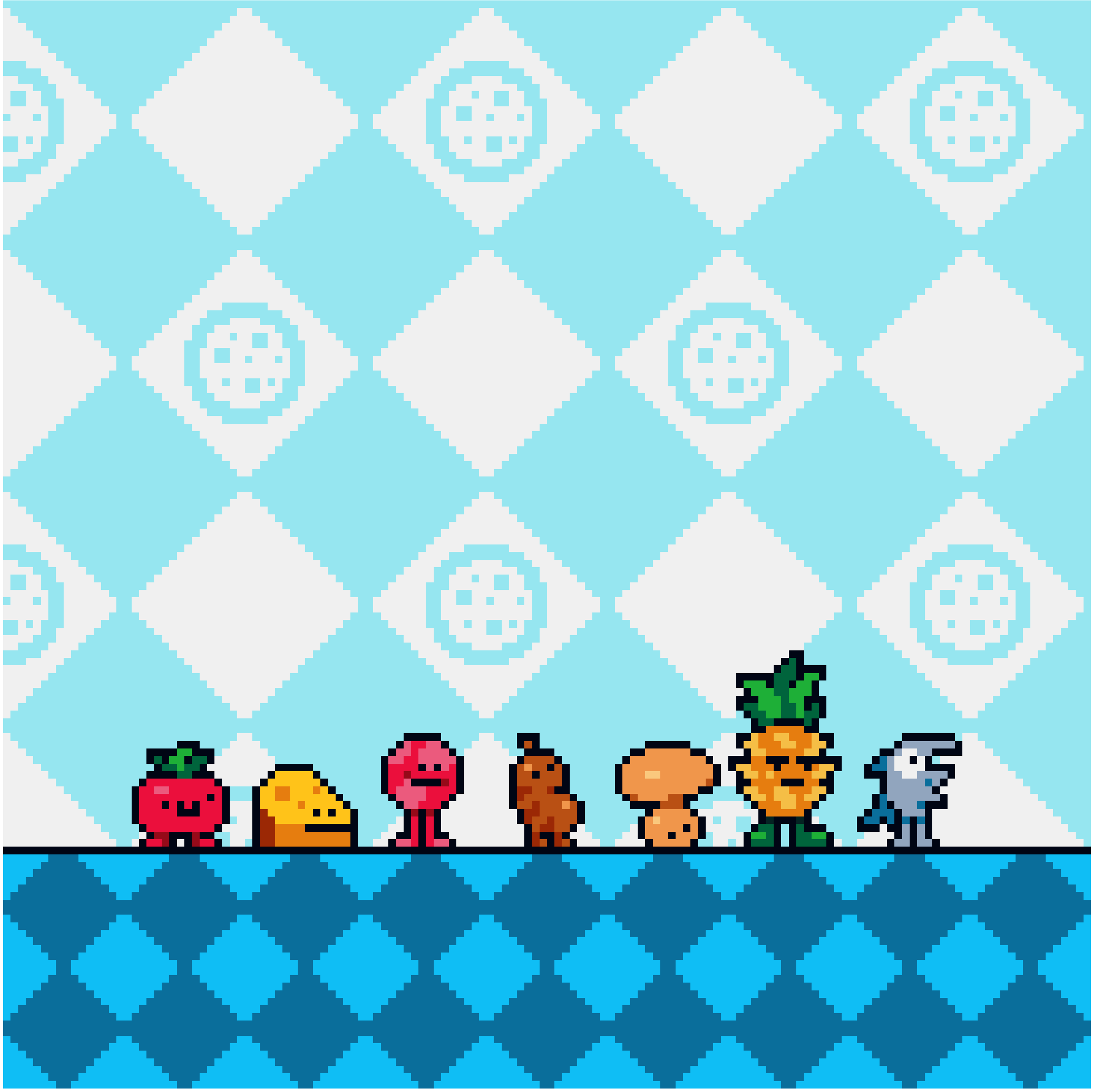
- Player input would be handled within the state classes since there is no character that the player directly controls.

Art

What is the game's visual style? Describe what the player will see.

- The game's visuals follow a cartoon pixel art style with bold outlines and saturated colors, inspired by the games Garbanzo Quest and Pizza Tower. The color palette we are using is WLK44 V2 by WildLeoKnight which compliments the visual style we are aiming for.

Include any concept art for reference. This could be screenshots of current games you've played and are taking inspiration from, a mood board, or custom drawings from team members. Include those references here/ If your team already has the assets you will use, include those here instead of concept art.

- 

Where will you gather art? Include links to potential 2D assets here:
(You don't have to have your full set of art yet, but having a set of images that are potential contenders for your game is good planning. Find them now, not later.)

- All art in the final game will be original. We may still use temporary free art during development from websites such as itch.io and opengameart.org.

Interface

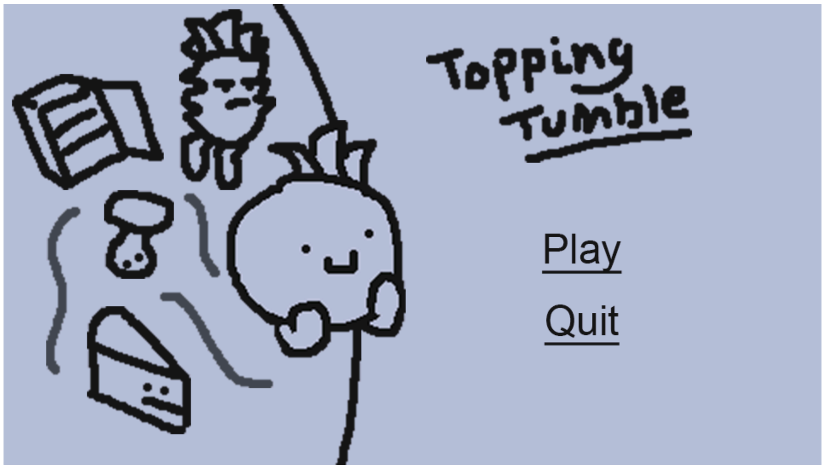
How does the user interact with your game? *(Are you using both keyboard and mouse, or just one of those?)*

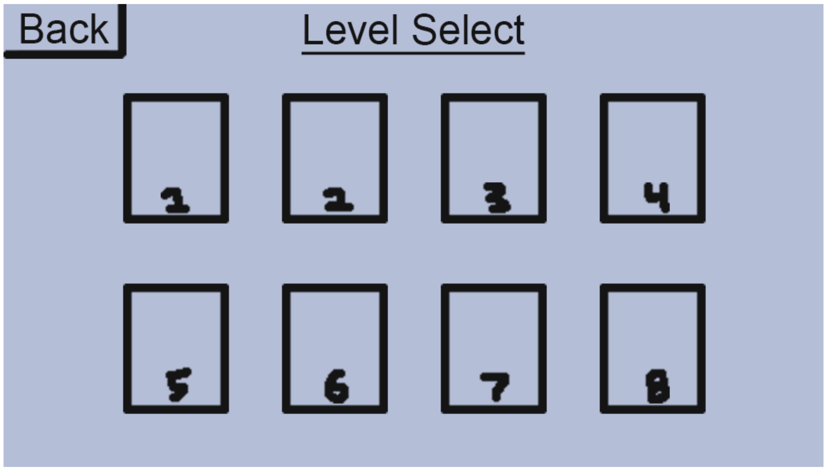
- The user will only need to interact using mouse buttons to modify the current level and interact with buttons on the UI.

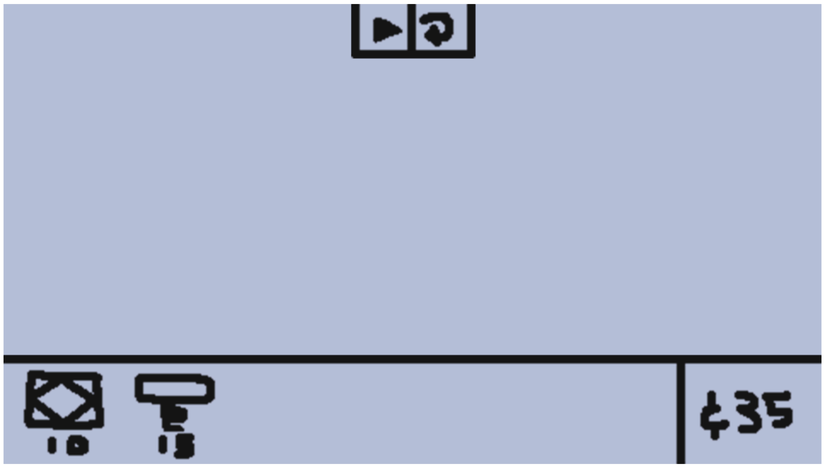
What are the user interface requirements? *(What buttons, bars, UI elements does your game need?)*

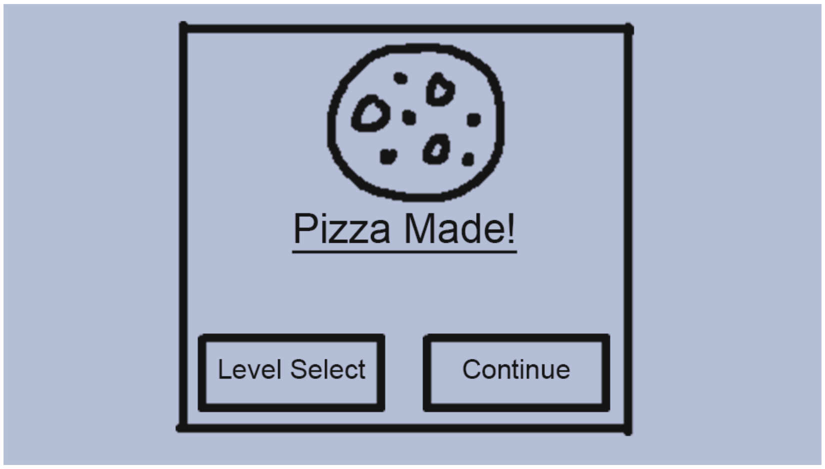
- The title screen will require a start button and title graphic. The level select will have buttons for each level that the user can play. During gameplay, there will need to be a UI for the types of tiles the user can place and text indicating the player's remaining currency to spend on tiles. There will need to be a button to start sending out the ingredients and a button to reset them. There will be a game over screen with buttons to return to the level select or try again. There will also be a victory screen which displays the user's score and buttons to proceed to the next level or return to the level select.

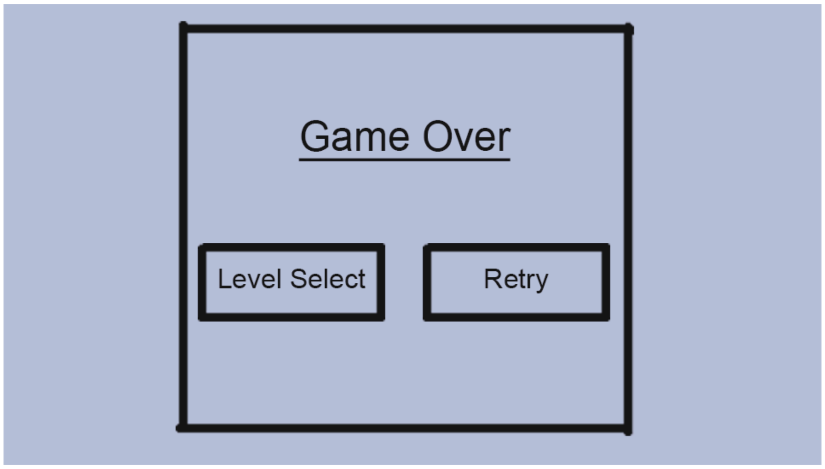
Include mockups of your game screens, menus and HUD here:
(These can be hand-drawn or digitally created.)











Production

When and where does your group meet? Who is the “record-keeper” for meetings?
(Does the record keeper rotate, or is it always the same person?)

- The group’s main meeting times are Thursdays at 1PM and Saturdays at 10AM. We will rotate who the record keeper is each meeting.

What is your group’s chosen method of communication?

- The group’s main method of communication is Discord. We also have a Google Drive that can be used to share documents with each other, but actual planning/updating usually happens on Discord.

How often are group members expected to communicate with each other?

- Group members are expected to communicate with the group when they are actively working on the project, have finished working on a feature, will not be able to finish a feature, or when they cannot attend a meeting. Additionally, group members can share any information that they think the group will find useful.

How will your group utilize Trello (or another Kanban board) to handle task management?

- The group uses Trello to divide tasks among different members. We create several tasks depending on the goal, and then assign each group member to a task. Once the task is completed they should move it to the “DONE” list and notify the group on Discord.

How will your group assign/divide programming responsibilities throughout the project?

- We will assign programming responsibilities based on how familiar the person is to that part of the program. For example, if somebody has an idea for a particular class, then they will most likely be the one to implement it because they know the most about it.

Game Requirements

What are the minimal specifications you need to have a working version of the game by the end of the semester? Include a brief description of those requirements.
(For example, “1 level with 3 enemy types and 2 power-ups. Background tiling system for the single level. Art assets for the player and each type of enemy,” et cetera)

These are the BARE MINIMUM requirements for a working game:

- A main menu that features a button to move to a level select screen
- A level select screen that only allows players to choose levels that they have unlocked (levels are unlocked by completing the level previous to it)
- A victory screen that allows the player to return to the level select screen, or to move on to the next level

- A game over screen that allows the player to return to the level select screen, or to retry the level that they failed on
- System to add and remove placeable blocks to a grid by using the mouse
- A “START” button that spawns ingredients into the level
- A “RESET” button that deletes all user-placed objects and moves all of the ingredients back to the “start” point
- Ingredients that are affected by gravity and walk forward until they hit a wall and turn around
- Level objects that denote a “start” and “end” point for the ingredients (victory is achieved when all ingredients reach the “end” point)
- An external editor to create levels
- Very basic art that represents pizza toppings (ingredients), a fridge (“start” point), and an oven (“end” point)

These are the features that the game SHOULD have when it is finished:

- A currency type system to restrict the number of placeable objects
- Placeable launchers/springs that send ingredients up into the air
- Spike traps that kill ingredients on contact
- Optional coins that can allow the player to go for high scores
- Buttons that can unlock a door/gate in another part of the level
- Very basic art that represents a launcher, a spike, a coin, a button, and a door

What are your stretch goals for the game?

If time allows, these features would be good to implement:

- Sound/background music
- A “Live Chef Reaction” that features a chef making different expressions depending on what is going on during gameplay

Task Timeline for Milestone 2

Include a bulleted list of tasks to complete during milestone 2:

- Programming for BaseState and GameMain classes to handle state switching
- Basic code for the main menu, gameplay, game over, and victory states
- Basic code for buttons
- A content loader class for easy access to assets across the codebase
- A GameObject class
- Basic code and at least basic art for ingredients
- Basic code for the tiles and tilemap system
- Basic art for the editable and non-editable tiles

- Basic art for the start and end of a level
- Basic code for the user level editor

Lessons Learned

Write a 2 - 3 paragraph summary of lessons learned during this milestone.

One major lesson we learned is that the scope of a game heavily affects how complicated the architecture for it is. While brainstorming ideas for Topping Tumble, our scope was too large at first and we realized that it was unrealistic to implement certain things, such as moving platforms. Our architecture was too messy so we had to cut back on some of our ideas in order to have our game be finishable. After focusing the scope of our game onto the core gameplay loop of placing and removing blacks from a level, we were able to make our architecture more realistic.

Another experience we gained was that of collaborating with a team for a relatively large game project. We all had different ideas for the game and how we would break it down, and it took a lot of collaboration, compromise, and throwing ideas around to finally get an idea of what we actually wanted to do for this project. There was a lot to be learned about what it's like working with people with ideas and skills different from your own, and we came up with something that we all agreed would be fun and also possible to do.

Describe how this milestone's experience will influence your plans (workflow, communication, task management, etc.) for the next milestone.

From this milestone, we've learned that our current workflow and team communication methods are quite effective. Two of us were unavailable during the weekend, but the group was still able to work around that due to notifying the team ahead of time. Therefore, we plan to continue with our current communication methods in the future. This milestone also gave us a better understanding of the scope of our game, which will be useful as we develop it in the coming milestones.

Milestone 2: Bare-Bones Game

Link to your group's presentation slides:

https://docs.google.com/presentation/d/1curBGSqxGf_XQY68hddKROX2Ox6OPgPtsyYbkzLdyOU/edit?usp=sharing

Scope

Given the priorities, what is your target MVP? Why? This is not a list of features. Describe the target player experience of your prototype.
"MVP" means "minimum viable product" -- the bare minimum functionality that needs to be complete in order for the game to ship. For this class, "ship" means it demonstrates your overall priorities and meets the course requirements.

- Our target MVP for Topping Tumble is a game that requires the player to help a group of ingredients navigate a level to reach an oven. Specifically, the player will have to place down solid blocks that can be used as a platform or can change the direction of the ingredients. For our MVP, the player should be able to place down blocks and then decide when to spawn the ingredients into the level. Additionally, the player should be able to reset the ingredients at any time they want.

Team Responsibilities

List each team member's responsibilities during this milestone.

Team Member:	Responsibilities:
Nathan	Working on the TileMap class, Level Editor, ingredient spawning/resetting, victory conditions, and AnimationController class
Gage	Working on the GameMain state machine, GameObject class, tile placement/shop framework, level select, and various tile functionality
Jeremy	Working on new art assets, in-engine auto-tiling, and UI programming, including menu buttons and custom text rendering with custom fonts.
Zach	Working primarily with core gameplay mechanics and the ingredient class, such as collision detection, movement, gravity, and interactions between ingredients and level objects

Production

How has your group utilized the Kanban task management strategy(ies) that you chose in the first milestone?

- The group has used Trello to “check-out” certain classes in order to avoid merge conflicts. If someone is working on a class they move it to a separate category so that nobody else tries to make changes. Then, they move the class back to the “open” category once they are done so that other people can work on the class.

Challenges

What challenges has your team faced with the project? How has your team overcome those challenges?

- So far, the only major challenges we’ve faced are with bugs and figuring out how to solve them. For example, the ingredient collision wasn’t working and it took a long time to figure out the issue. We also struggled with an issue where springs sometimes wouldn’t launch ingredients properly.

Bugs

Are there bugs that require fixing? If so:

A) **What are they?**
B) **What is your team’s plan for tracking and then fixing the bugs?**

- Currently, we are not aware of any major bugs that need fixed. For tracking and fixing any future bugs, we plan to add them to our Trello board as items to complete, in order of priority.

Design/Architecture/Style/UI/Production Changes

If your design has changed, describe the changes here:

- So far, our gameplay/design has not changed. We are following along with the plan we created in milestone 1.

If your architecture has changed, include an updated architecture diagram here:

- Our overall architecture has not changed, we have just organized classes into folders for ease of readability.

If your visual style has changed, describe the changes here and include new reference images:

- Our visual style has not changed.

If the user interface has changed, include new mockups here:

- The UI design has not changed.

If your method of task management has changed, include information about it here:

- Our method of task management has not changed, we still use Trello as we planned to and everyone is working on the features that they planned to.

Task Timeline for Milestone 2a *(the in-class checkpoint)*

Include a bulleted list of tasks to complete during the next part of this project:

- Fix collisions with placeable walls.
- Complete and add more art assets, such as those for the menu states and unique tile types.
- Implement jump pad tiles
- Add death functionality for ingredients
- Add editing number of ingredients and currency in the level editor

Lessons Learned

Write a 2 - 3 paragraph summary of lessons learned during this milestone.

Describe how this milestone's experience will influence your plans for the next milestone.

We learned that it is important to make code that is reusable and extendable. For example, animation was made specifically for ingredients at first. However, we found that many different types of objects needed animation too. As a result, we made an AnimationController class that can be used by anything to add animation. Additionally, it was easy to add more objects into the level editor because of the way it was set up. In terms of bug fixing, we found that taking time away from the computer and talking through it with people or getting some alone time allows your brain time to think through the problem and come at it with a fresh approach.

Milestone 2a: Almost there!

There is no required documentation for this milestone. However, students are required to bring a list of tasks for game completion to class on check-in day. To keep all group members on the same page, it's a good idea to list those tasks here.

Task Timeline for Milestone 3

Include a bulleted list of tasks to complete during the final milestone:

- Design levels
- Finish necessary art assets & animations
- More placeable tiles (IE: a gravity flip)
- Tutorial/Instructions

Which elements of your game are currently stretch goals?

- Audio
- "Live Chef Reaction"
- Level unlocking and progress saving

Milestone 3: Finished Game

Link to your group's presentation slides:
https://docs.google.com/presentation/d/1uRZUog3vF5fqm6JDdR5_nubhVY6oWMFS3xND-grgCLc/edit?usp=sharing

Team Responsibilities

List each team member's responsibilities during this milestone.

Team Member:	Responsibilities:
Jeremy Kotz	UI Programming (Options sliders, better texture loading for buttons, “gray-ing” out of shop buttons), art and UI assets, level design.
Nathan McComber	Helped implement spring placing upside down, clear button, victory screen UI, added Gravity Flip to level editor, added fullscreen option, bug fixing
Gage Magar	Reworked level loading to add more info to level select, progress and options data saving, Gravity Flip tile/functionality, credits screen, bug fixes
Zach Jordan	Added audio support, as well as music and sound effects implementation. Started working on a moving pizza cutter obstacle, bug fixes

Post Mortem

What went right?

Communication:

Our communication throughout the course of the project was thorough and there was never much confusion about where we were or what had to be done. By using Discord, we were able to let people know when we pushed changes, and then people could pull those changes and test them. This allowed for quick feedback on any changes that were made, and kept us all in the loop about what was happening. Additionally, we would let each other know when we were working on certain classes, which helped us to avoid major merge conflicts.

Time Management/Scope:

Throughout the course of the project we made swift progress by focusing on what our core ideas were. For example, we began by focusing mainly on making ingredients that could walk and making blocks and springs that could be placed. Once we finished those core concepts, we slowly expanded our scope to include coins, spikes, speed boosts, gravity flips, and salt shakers. By expanding our scope feature by feature, we ensured that we had a minimum viable product for the entire time of the project. By managing our time this way, we “finished” very early on and then got to expand our game without feeling rushed or overwhelmed.

Game/Programming Structure:

In terms of underlying systems, our program architecture changed very little compared to our initial plans and diagrams. The use of a class-based state machine was incredibly advantageous for organization and code segmentation, as it allowed us to chunk the project in a way that was easy to understand. For instance, anything to do with playing a level was contained in the GameplayState class. Additionally, splitting code between several files helped us avoid major merge conflicts. We also retained classes such as GameObject for general objects, the TileMap class for rendering and updating tiles, and so-on. We did, however, end up needing substantially more additional functions, properties, and classes as we expanded our scope.

Style Consistency:

Our game’s visual identity and style were established early on in the project, and stayed consistent throughout. This is largely due to a couple of key decisions we made, which were the size of an individual tile, the size of the screen, the color palette, and the primary art references. Because we stuck to these decisions, the game could be programmed around these parameters without worrying about issues regarding implementing the art into the game. The final project feels very cohesive, which can partially be accredited to the visuals.

What was not ideal?

Gravity/Collision Programming:

We initially thought that gravity and collision detection would be simple for our game, as there weren’t many complex interactions between objects. However, this task turned into something we had to write and rewrite several times throughout the project as we added and changed more features. Initial implementation proved to be the most difficult to implement, as we had to come up with a far more sophisticated collision system than the one we wrote in our collision detection practice exercise. Then, later in the project we had to change most of the original implementation to include smaller hitboxes to prevent glitches with small collisions. We also created our own “TF2 Coconut” situation, where every collision relied on the existence of our first texture, Testing Terry. We had to fix this as well, and our current implementation will hopefully be the final one, as we spent much more time on this than originally planned.

Division of Responsibilities:

While we were all working on the project at a similar pace, there were still issues with how we divided work. Having one person solely responsible for graphics was the biggest issue, as it meant they had a large workload in addition to helping with programming. While it did allow for the graphics to be very consistent and visually pleasing, it would have helped lighten the workload if other members took on art as well.

Changing when players can place tiles:

For our initial game concept, we imagined that the player would be able to interact with the level by placing tiles and objects as the ingredient characters walked around on screen. However, while designing levels, we found that the ability to change the level as ingredients were walking largely took away from the gameplay rather than adding to it. The core gameplay revolves around ingredients moving freely, so being able to place tiles during the level allowed for players to trap ingredients in place, essentially giving the player full freedom of their movement. We made the decision to only allow players to place tiles before spawning the ingredients, essentially splitting the gameplay into two phases. This change both helped the gameplay better fit our original vision, and made level designing easier and more streamlined.

What did your group learn from this process?

Cooperating on a Game Project:

A major thing that we learned was how to properly cooperate on making a single game with multiple people. We had to develop habits like pulling code from our repository before making changes and notifying group members when we added anything. Additionally, the importance of commenting became very clear during this project. By properly commenting our code, we were able to decipher how people made a certain feature work and then make changes to it. For example, the comments for only allowing springs to be placed on walkable tiles were very thorough, which was helpful when changing code to allow springs to be placed upside down. Now that we have completed our game, we have skills that we can take into future game and software development projects.

Presenting our Work in a Concise/Approachable Manner:

A very useful skill we all developed in the production of our game was the ability to condense all that we had done into a concise, uncomplicated slide format. The ability to explain exactly what you have been working on to people who have never seen your game before is very important in the industry, as you present your work often to people who may not understand it. It also increased our effective communication skills, another soft skill essential to efficient game development, as multiple people from different technical backgrounds need to communicate ideas between each other.

What will your group do differently next time?

Making better use of our organization tools:

At the start of our project we created a Trello board to keep track of all our tasks. However, we slowly stopped using Trello as we continued development of our game. Instead of using Trello, we only communicated in-person and on Discord. Those methods of communication worked, however keeping track of what we had to do got more difficult. Furthermore, features that we finished were still marked as WIP on our Trello board. For our next project, we should be more diligent about keeping our Trello board updated so that it accurately reflects the state of our game.

Future-Proofing:

There were a few times where we needed to rewrite or extend parts of the code that we didn't design to be extended in such a way. For instance, the shop system and tile placement wasn't initially written to support showing or hiding tiles depending on the type of level. The system had to be rewritten such that a new ShopData structure gets recreated every time the gameplay is loaded. When writing the level select screen, we needed to be able to pull information about a level from a file without building the entire tilemap. However, we assumed we wouldn't need to do this at first, so levels were only able to be loaded while a TileMap was active. In future projects, it would be ideal to take more time considering how our classes and features might need to be expanded upon. Thus, we could write code to be more easily extended and modified.