Game Proposal: Barista Beat Battle

CPSC 427 – Video Game Programming

Team: Fluid Games

Jubelle Paa - 52420270
Raphael Deonova - 15392822
Ameya Goel - 74681834
Hans Lam - 80010721
Trevor Glennon - 27719343
Justin Rahardjo - 92665561

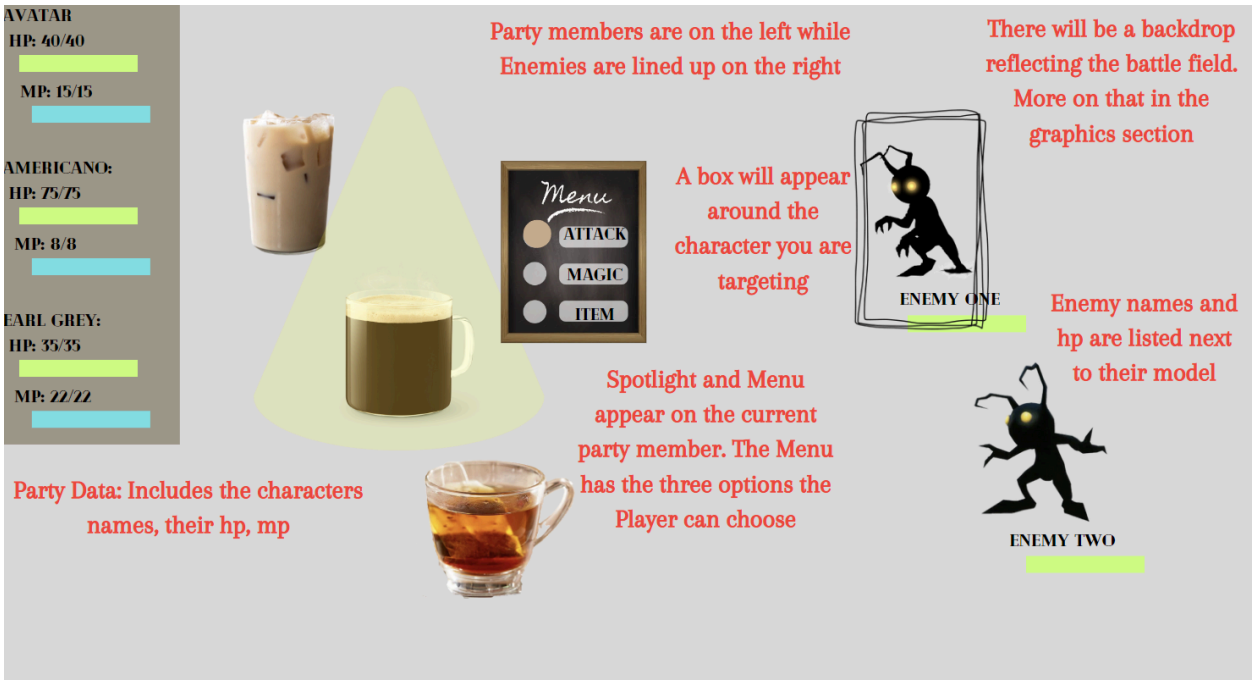
Story:

A war brewing between Tea and Coffee in the small cafe of *Barista Beats*. It's a battle between the two to make the best possible drink and become the stars of the cafe. You play as Dirty Chai, a drink who recently entered the cafe and chooses not to take a side but to bring peace to the two groups and uncover secrets while trying to avoid becoming roasted. You will start by entering the cafe and finding out about the war and meeting members of the two sides. Choosing to make your own path, you will engage in a turn-based battle against both sides to create the best drink. Going from level to level from the cafe menu, you will show the enemy your drink-making skills and defeat them in battle through minigames like pouring latte art and steeping tea. Completing these minigames will help you defeat the enemies, while failing them could give the enemy an advantage. Inspired by your skills and your strength in battle, characters from each side will join your team. They will help you fight tougher battles against each side and aid you in your quest to make the best drink possible and put an end to the Tea vs Coffee war.

Scenes:

Turn-Based Combat:

The turn-based combat is going to be the standard Final Fantasy 1 combat.



Turn-Based Mechanics:

1. All characters will appear on the screen, your party on the left and enemies on the right
2. The party will have their state listed in the top left while each enemy will have their state below their character
3. Each character (Ally and Enemy) will have a stat sheet:
4. HP; MP; Strength; Magic; Armour; Magic Resist; Speed

Speed system:

When a new encounter starts each character's turn order is based on their speed value. It will be a running tally turn system (this will all be in the backend, if there is time adding a visual element will happen but that is something we consider extra). Each character will go once their turn value reaches 100; during every ‘tick’ a character's speed will be added to their turn value. If a character’s speed is over 100 they will act, otherwise tick again. This system is so that fast characters will act more than slow characters, as opposed to other turn-based systems where everyone always acts the same amount.

Ally Turn:

An ally will have 3 options:

1. Attack: Target an enemy based on Strength
2. Magic: Rhythm time game based on Magic, consumes MP
3. Item: Select either a Potion or Ether to restore HP or MP of a party member

When selecting a Target a Box will appear around the unit you are selecting

Enemy Turn:

An enemy will attack a Party member, there will be both strength and magic enemies whose basic attack is based on their type

In each encounter, you will take turns taking actions to either defeat all enemies (reduce their HP to 0) or until all your party members are knocked out (our HP goes to 0).

If you win the encounter, you will unlock the next level on the main menu, otherwise, you lose and will have to start the fight over to progress

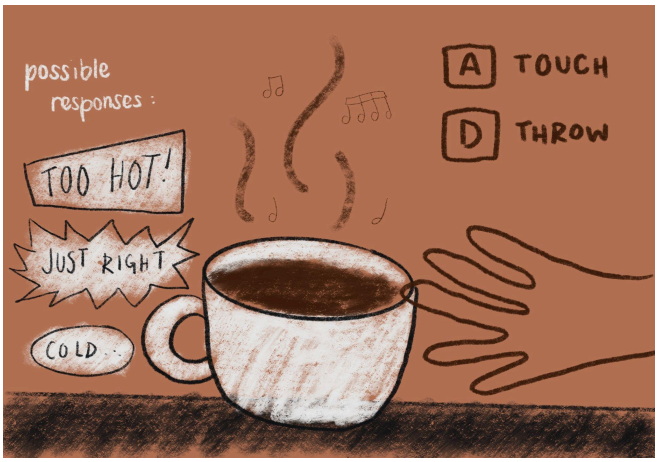
Rhythm minigame mechanic:

When a Magic attack is selected the Menu panel will expand to show the mini-game. After you play the game, then it will shrink again and the result/effect of the minigame will be applied to the targeted enemy.

Rhythm Game Scenes:

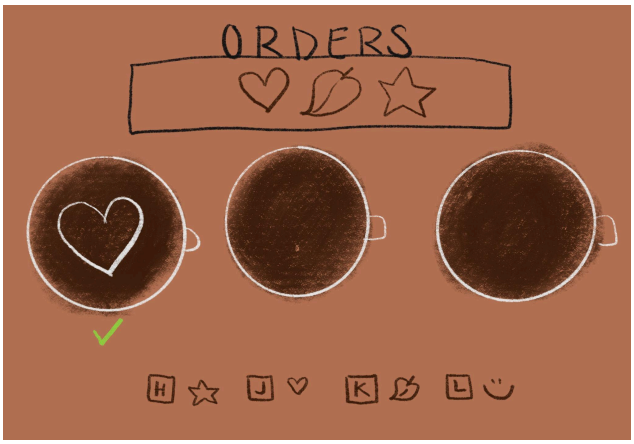
Note: We will stick to implementing Cool It! first, with all battles using it, then implementing the other two minigames and adding those to the later levels if time allows.

1. **Cool It!** Music (4/4 timed) will be playing in the background. On the 4th beat, players can hit A to touch the cup to check its temperature, and one of the possible responses will pop up. If, upon checking the temp, the response is *Too Hot!,* the player should not do anything. If the response is *Just Right,* the player should throw the cup at the enemy by hitting D. If the response is *Cold…,* the player should still throw the cup (but the damage amount of the attack will be half as much since it's not as hot anymore). The game ends either once the song ends, or the cup is thrown (whichever comes first).



2. **Pour It!** The player will press and hold the W key to pour tea. They must successfully stop pouring at the right time, which will be signaled by the music playing. A new empty cup will always appear after finishing pouring into the last one. The attack amount will be calculated by how many cups were successfully filled perfectly. The game ends once the song ends.

3. **Milk it!** Players will be given 3 lattes, along with the next 3 latte art orders. Each art is mapped to a keyboard button, as shown at the bottom of the scene. The player must select the right buttons in the correct order, but they can only have the game successfully place the latte art when they hit the keys on the beat of the music. The game ends once the song ends.



Technical Elements:

This game includes several technical elements that meet the criteria for our project. These elements are as follows:

Rendering: The game renders several sprites during turn-based combat, which includes the player, the opponent, attack buttons, and the HUD which has the health and mana display. When a player attacks, the game changes scenes to a mini-game, and more sprites will be rendered throughout the mini-game.

2D geometry manipulation: During the turn-based combat, sprites will move after a mini-game is completed to do an attack animation. Another animation will also happen when a character dies. Apart from animations, sprites in the mini-game will also collide.

Gameplay logic/AI: The game loop determines the gameplay: The player jumps right into a turn-based combat, and chooses an action for the currently active character, which takes the player into a mini-game. We will only develop 1 mini-game for the MVP (minimum viable product) and have the player do this mini-game over and over again for each attack. During the mini-game, the player is then asked to follow a specific rhythm by clicking on a button from their keyboard. The attack becomes more powerful as the player is more in sync with the rhythm of the mini-game.

Physics: simplified physics is used during the mini-game aspect. As sprites move around in a rhythm, simple physics (like gravity) will be used to determine their position.

Advanced Technical Elements:

Syncing Mechanisms: Precise algorithms that can precisely synchronize between user input, visuals, mechanics, and sound will be a top priority for this game after the MVP is done. This enables us to differentiate between Perfect, Great, Good, and Miss input from the player. This is to ensure honest gameplay.

Dynamic difficulty adjustments: After we can precisely measure a player's capability to complete a rhythm game, the next step is to include difficulty adjustments for subsequent mini-games: better players have to do harder mini-games with a higher reward, and worse players have to do easier mini-games for the same reward.

Devices:

Explain which input devices you plan on supporting and how they map to in-game controls.

For the level selection, we plan on taking in mouse inputs, where left-clicking will map to opening the settings, party, or help tab, selecting the next stage, and starting the next stage via the Play Level button.

For the turn-based combat, we plan on taking in mouse inputs again, where left-clicking will map to the selection of certain options available to the user (left-click to select attack, magic, or item).

For the Cool It! minigame, we plan on taking in only keyboard inputs, specifically for the ‘A’ and ‘D’ buttons. The ‘A’ button will map to the act of touching the cup on the correct beat, and the ‘D’ button will map to the act of throwing the cup on the enemy depending on the response provided.

For the Pour It! minigame, we plan on taking only keyboard inputs, specifically the ‘W’ button. The ‘W’ button will map to the pouring of the tea, where the tea will pour into the cup so long as the ‘W’ button is held down. As soon as the ‘W’ button is released, the tea will stop pouring.

For the Milk It! minigame, we plan on taking in only keyboard inputs, specifically the ‘H’, ‘J’, ‘K’, and ‘L’ buttons. Each of the keyboard inputs will be mapped to a specific symbol (eg. ‘H’ is mapped to the star symbol), where once a specific button is pressed at the right time then the mapped symbol will be placed on the first empty cup (starting from the left side).

Tools:

Specify and motivate the libraries and tools that you plan on using except for C/C++ and OpenGL.

For the level selection, turn-based combat, and Pour It! minigame, those components will be done natively in C++ and OpenGL.

For the rhythm mini-games (Cool It! and Milk It!), we will need additional libraries and tools to help us with the music playback, beat detection, audio feedback (upon hitting something on the correct beat), and song/beat mapping.

The music playback and audio feedback require us to use some sort of audio library, and we plan on using either SoLoud or PortAudio. These libraries will allow us to play certain sounds on top of the music, and play those sounds when certain events are triggered.

The beat detection will require an audio/music analysis library, and we plan on using either Essentia or aubio. These libraries will help us to analyze the music, which is important for song mapping as we will need to map/sync what is being displayed on the screen to certain beats, as well as determine whether or not the user pressed the inputs at the correct timing.

Team management:

- The development plan below is a good reference to use throughout the project, we are all aware that it is subject to change.
- Each member will have a different workload each week so that we will prioritize a flexible schedule per week. This means that every week after class on Monday and Wednesday we will check in on priorities in the week.
- Making a hard decision on the division of work on those days will be required, as well as setting a time to check in with each other on the work. The expectation is that the work will be done by that time.
- For small deliverables, the internal deadline can be a day before the submission (ex: the game proposal is due on Friday, and we will check in on Thursday expecting the work to be done)
- For large deliverables, the internal deadline can be up to a week before the deadline.
- The reason for an overly optimistic internal deadline is in case one or more team members are unable to get the work done. This provides the team with time to consider if the work can be redistributed or for the team members to confirm that they will be able to complete the work.
- Main communication will be done through Discord and options such as Trello will be explored throughout the process.

Development Plan:

Each week will have testing time baked into the schedule. A more detailed list of tasks will be laid out for every milestone before the two weeks start like the one for Milestone 1:

Milestone 1: Skeletal Game

Week 1: Set up the repo, copy over simple ECS, and make sure the ECS can create entities with components for the main combat scenario (entities like playable character(s), enemies, components like health, moves, etc) the plan is to start from the most interesting part but also simplest part of the game, combat, specifically the turn-based functionality. This milestone will prepare us to set up that functionality.

Week 2: Render a background for combat and/or start screen. Given that it's reasonable to expect to complete all this in week 1, we can aim for a very simple start screen with some preliminary artwork or background artwork. It's also a good idea to start thinking about how the music will work but these are all "nice to haves"

1. Figure out which files are required in the repo and put those resources (basically getting the src folder sorted out)
2. Start design:

    a. Brainstorm what to add/subtract to the ECS, doesn't have to be final, we can add as we go and consult instructors as needed
    b. Put comments on what we are adding that wasn't from the A0 ECS.
    c. The task is complete when we can say that the next step can be transitioned into easily and there won't be much need for discussion when the next milestone starts
3. Based on A1's requirements, we should be able to get either a background for the combat scene or a start menu screen.
4. If there is time, we will also need to decide on where elements of the game will appear on screen. Refer to the first image in the Scenes section, start a discussion on questions like "Is the attack/magic/item menu an entity?, What components will it have?, etc"

Milestone 2: Minimal Playability

Week 1: Implement combat system without minigames, begin design on minigames, and decide specific aspects of it:

- Will it be closer to a conventional rhythm game like OSU
- Or more just about beats, and imitating rhythms
- Or is it more like a timing minigame like Undertale or paper mario
- Or a combination?

Week 2: Potentially modify ECS to accommodate entities for the rhythm minigame. Make sure it interacts with the required entities, damage output, etc. Planning for the rhythm system should start now too, questions like: what will be used to track when the game knows that the player input is on beat? What is an appropriate range for getting an accurate hit? Will there be degrees of "on beat"?

Milestone 3: Playability

Week 1: Implement minigames. This week should focus mostly on getting elements related to the rhythm game to appear on the screen. Work for audio syncing, etc should begin as well but should not be the main focus of this week. For the sake of testing, the minigames should be developed separately so we don't have to go through an entire combat sequence just to get to the minigames. Testing whether the minigame works and whether the minigame can be baked into the combat can be separate things.

Week 2: Continue minigame implementation. This week will focus more on the rhythm, making sure that the audio and player input interact in the way we want it to (when the player presses a key, it gets registered with an appropriate latency).

Milestone 4: Final Game

Week 1: Insert story elements, dialogue, cutscenes, etc. These are the “press to continue” parts of the game. We can also use these final weeks to finish up any technical aspects of the game if need be, and if there is nothing left to do regarding game logic, we can focus on writing, art, and overall presentation. Attack animations are the priority at this point since it is a feature that can just be pasted on top of existing mechanics and it is unlikely that this will be done earlier since it can slow down the development of the actual game logic. If any additional **development** work will need to be done, this is where it will be done.

Week 2: We will aim to have everything done and finalized by the start of this week. If any additional **testing** needs to be done, this is where it will be done.