Game Proposal: Harmonic Hustle

CPSC 427 – Video Game Programming

Team 2

Sherry Shi 35520220

Arianna Joe 85283554

Tracy Chow 47273370

Keelan Hui 40161151

Ivena Du 98916729

Wendy Han 51108264

Story

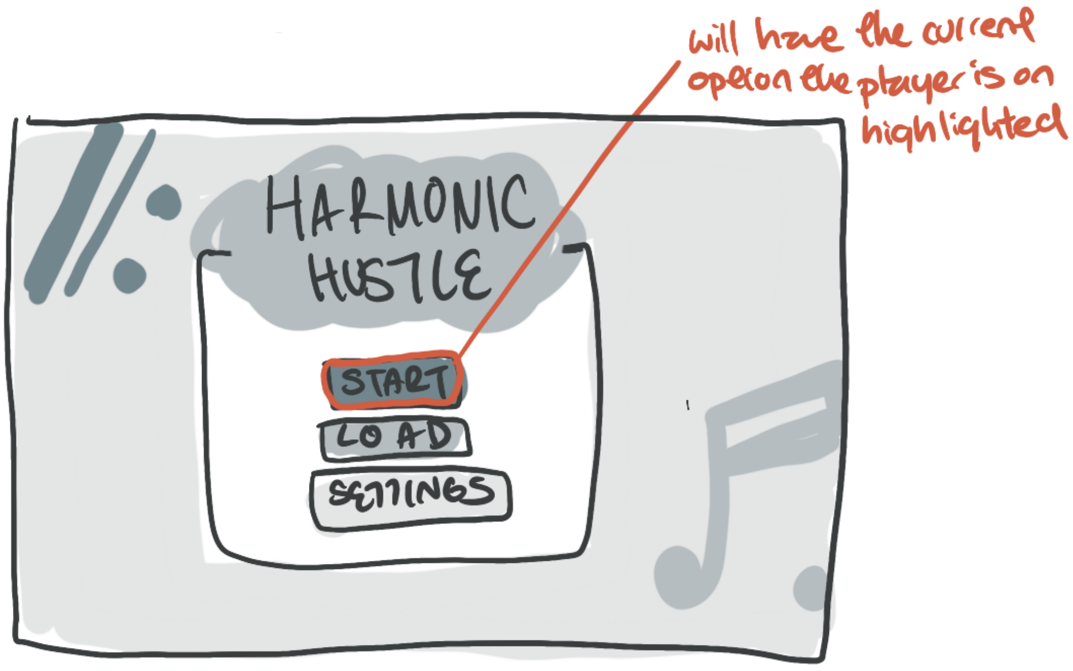
Background/Motivation:

- ❖ After being forced to listen to the world's terrible music, the player decides to take action against its musical offenders. The player then goes on a rampage about the world to change its music to be exclusively in the genre that they prefer.

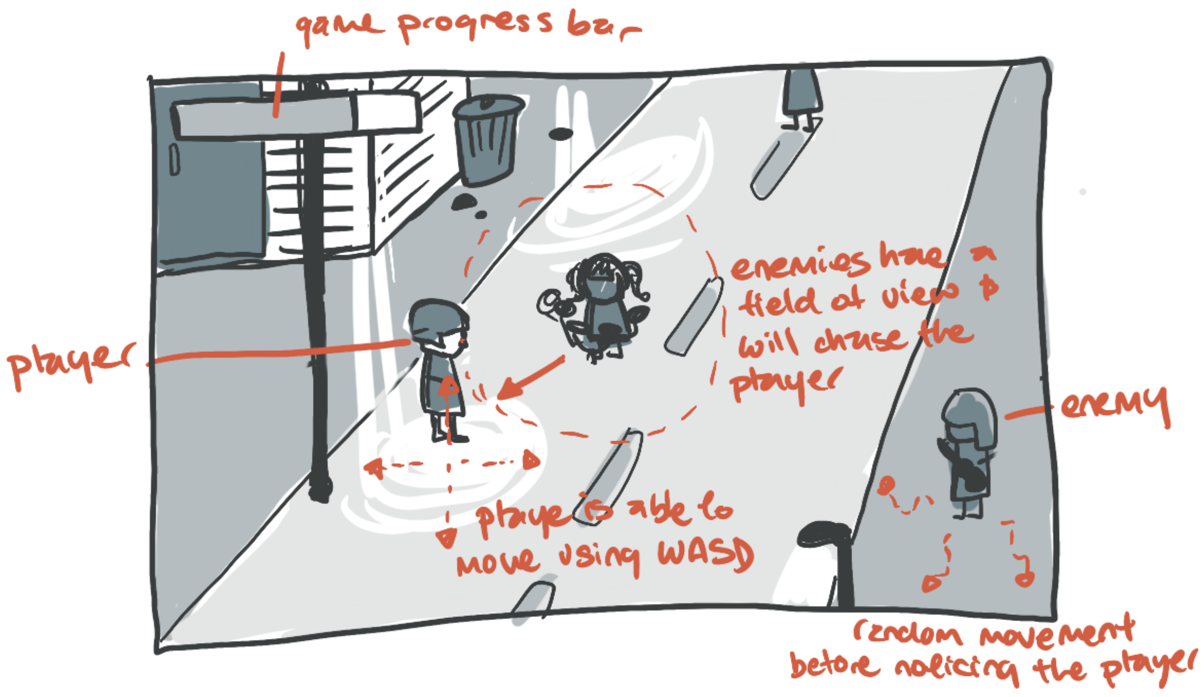
Gameplay:

- ❖ Our game will have one primary zone resembling a city, with up to three different enemies within functioning as mini-levels. Each enemy encounter will be a 1-2 minute battle sequence where the player must time their keyboard inputs against the enemy's actions. To win a battle, the player must time their inputs precisely and correctly enough to score above a certain threshold. After enough enemies have been defeated, including one final boss, the player wins the game!

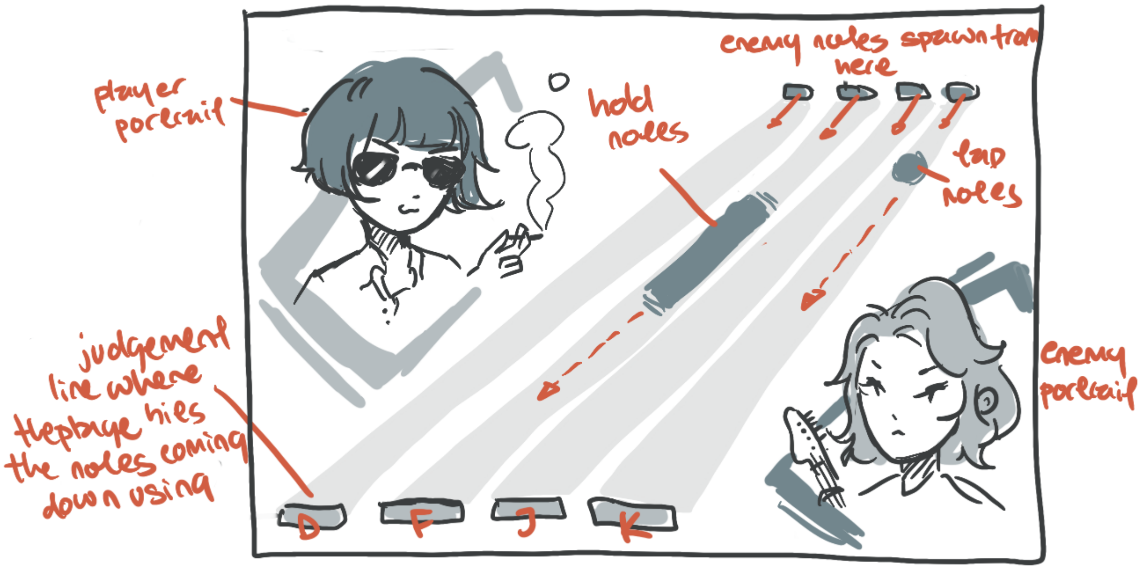
Scenes



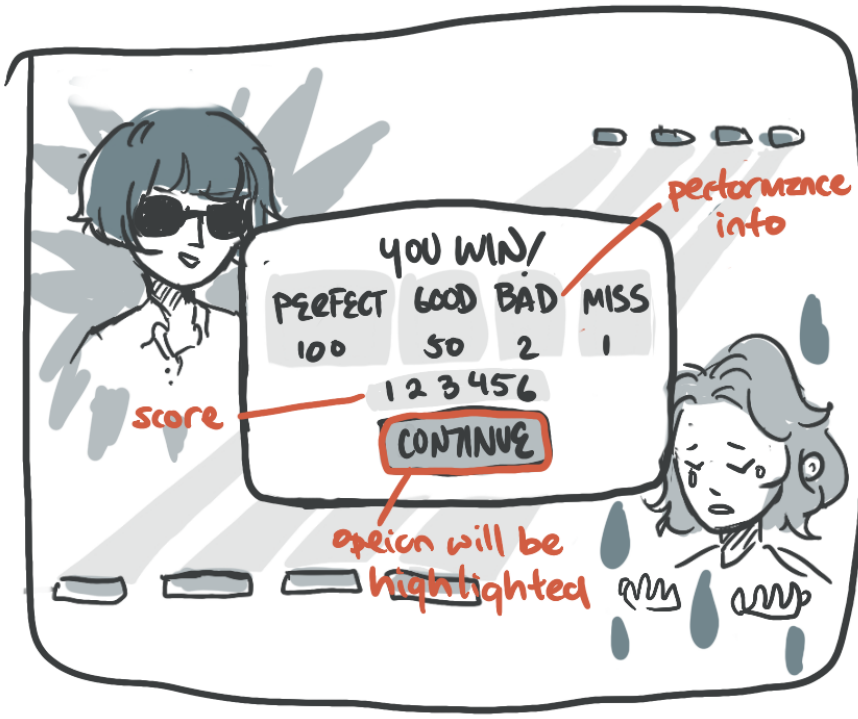
The main menu. This screen will be displayed when the player first launches the game.


game progress bar
player
enemies have a field of view & will chase the player
enemy
player is able to move using WASD
random movement before noticing the player

The overworld. The player and enemies can move freely within the game's boundaries. Enemies will spawn, and will chase the player when within a certain radius.


player portrait
hold notes
enemy notes spawn from here
tap notes
enemy portrait
judgement line where the player hits the notes coming down using
D
F
J
K

The battle scene. Upcoming sets of keyboard inputs will be telegraphed on the screen.



The battle-over scene. The outcome of the battle will be displayed, as well as a more detailed breakdown of the player's score.

Technical Elements

Technical Requirements	Our implementation
Rendering (visual effects, parallaxing, etc.)	● 2D graphics ● Static background ● Dynamic display of input commands
Assets (geometry, sprites, audio, etc.)	● Player and enemy sprites ● Music and SFX ● Backgrounds ● Input command sprites
2D geometry manipulation (transformation, collisions, etc.)	● Transformations for input commands moving across the screen ● Collisions with scoring regions for input commands and a judgment window
Gameplay logic (world interaction, character interaction, player controls, etc.)	● WASD for menu and world navigation ● Four keys for rhythmic inputs (JK and DF) ● Enter as confirmation key ● ESC to enter menu/settings and as back
AI (pathfinding, entity behavior, etc.)	● Pathfinding for enemies towards the player
Physics(entity interactions, particle effects, kinematics, etc.)	● Note projectiles will have acceleration ● Player interacts with enemies ● Collide with map boundaries
Sound (effects, music, etc.)	● Enemy-specific background music ● SFX on player hits and misses

Advanced Technical Elements

- Various difficulty settings for the player to pick before the start of the game
    - Impact in the event of skipping: The game will only have one difficulty which may result in a much easier/non-challenging experience for the player.
    - Alternative: Set some random enemy rhythms to be faster (in speed) than others to introduce another element of difficulty into the game.
- Saving the state of the game (reloadability, serialization)
    - Impact in the event of skipping: player is unable to return to previous state of game after exiting. Will have to restart each time
    - Alternative: Only being allowed to save and load one checkpoint at a time (the most recent) instead of multiple.
- Story introduction (quality/UX)
    - Impact in the event of skipping: the game doesn't provide a more immersive description of the background story of the game, and jumps right into the gameplay
    - Alternative: The background story is displayed as a single-page narrative at the beginning to the player.
- Dialogue between characters
    - Impact in the event of skipping: the game's characters have fewer opportunities to show personality.
    - Alternative: Have a single, brief story introduction at the beginning of the game.
- Enemy group AI - following the player out of respect after they have been beaten
    - Impact in the event of skipping: The in-game visual indicator of progress is less exciting (a progress bar).
    - Alternative: Display the defeated enemy sprites statically in the UI instead.

Devices:

- We plan on supporting keyboard inputs. WASD will be mapped to the movement of the player and menu controls. Enter will be mapped as a confirmation key and ESC will be to enter the menu/settings from the game and also act as a back key. DFJK will be mapped for the rhythm sections of the game.

Tools:

- SoLoud library for playing audio in C++.
- Reaper and Audacity for sound/music production.

Team management

- We will use Jira to create all the tasks and sub-tasks and assign team members to each one based on their role and which feature the tasks correlate to. They will also all be grouped into expected deadlines.

- ❖ We will also schedule weekly static meeting days to go over our progress and make sure we are on track.
- ❖ Week 1s will mainly focus on implementing the core features for that milestone. Week 2s will be skewed towards polishing, bug-fixing and testing. We may try to work on some stretch goals or work ahead at the end of week 2 if we end up doing well for time.

Development Plan

Milestone 1: Skeletal Game

Week 1

- Implement ECS classes/structure
- Import OpenGL boilerplate for game window
- Map and detect all player inputs
- Basic rendering of simple overworld scene
- Button to test transition to predetermined battle scene
- Basic collision detection in battle scene
- Assets: Player sprite, battle scene portraits
- Play audio files statically in C++

Week 2

- Track and stabilize FPS
- Render player sprite in overworld scene
- Basic transformations of sprites - from top to bottom of screen
- Player movement & boundary collisions in overworld scene
- Render enemy sprites on the overworld screen
- Assets: One enemy sprite and music theme, basic battle scene UI elements
- Finalize bug list
- Record demo video
- Write up formal test plan
- Report on Milestone 1

Milestone 2: Minimal Playability

Week 1

- Simple transition from overworld to battle scene on enemy collision
- Enemy pathfinding
- Deterministic input commands in battle scene
- Standing of nodes (perfect / good / bad / miss) and score
- Play SFX audio files dynamically on input hits/misses
- Assets: Overworld scene music

Week 2

- Fine-tune timing of rhythmic events; visuals should line up with player expectation
- Assets: Static background for overworld scene
- Score computations and win/lose threshold for battle
- Battle-over overlay displaying win or lose
- Testing the current game flow

Milestone 3: Playability

Week 1

- Display progress bar in overworld scene
- Implement back & forth beats system with the enemy
    - Enemy demonstrates/shows input commands, then player copies
- Link battle win/lose outcome with overworld progress bar
- Assets: Boss enemy sprite

Week 2

- Game pauses on ‘ESC’ key press, and pause screen is displayed
- Implement reloadability and serialization
    - Add checkpoints to game progress
- Polish UI for overworld & battle scene with more final designs
- Significant testing and polish for game feel and flow
- Display battle-over overlay, showing the outcome of the battle and final scores

Milestone 4: Final Game

Week 1

- Main menu displays on game startup.
- Main menu buttons are functional
- Add a third enemy scene (regular non-boss enemy) and music
- Add lighting effects
- Finalize and touchup all game features
- Potentially map more complex player vs enemy input system

Week 2

- In-depth testing of game flow
- Work on final bugs