Game Proposal: Just Parry

CPSC 427 – Video Game Programming

Team: A++ 2
- Story: 2
- Scenes: 2
- Technical Elements: 14
  - Combat System 14
    - Basic Movement 14
    - Inputs 14
    - Parry 16
    - Posture 16
    - Stage 17
  - Rendering 17
  - 2D Geometry Manipulation: 17
  - AI (Arcade Mode) 17
  - Physics 18
- Advanced Technical Elements: 19
- Devices 20
- Tools 20
- Team management 22
- Development Plan: 23
- Milestone 1: Skeletal Game 23
  - Week 1 (Until Oct 2 midpoint) 23
  - Week 2 (Oct 3 - Oct 6) 24
- Milestone 2: Minimal Playability 25
  - Week 1 (Oct 7 – Oct 13) 25
  - Week 2 (Oct 14 - Oct 20) 26
  - Week 3 (Oct 21 - Oct 27) 26
- Milestone 3: Playability 27
  - Week 1 (Oct 28 -Nov 3) 27
  - Week 2 (Nov 4 - Nov 10) 28
  - Week 3 (Nov 11 - Nov 17) 29
- Milestone 4: Final Game 30
  - Week 1 (Nov 18 - Nov 24) 30
  - Week 2 (Nov 25 - Dec 1) 31

Team: A++

Shaun Gao	74620345
Siddh Patel	24495285
Armaan Sawhney	44616670
Roy Tao	17267535
Nic Ung	67524991
Jenny Zeng	48191399

Story:

You are but a bird who happened to be in the room when EVO moment #37 happened. Shocked by what you saw with your tiny bird eyes, you fly out of the tournament ready to take on the world with a newfound concept in your mind: Just. Parry.

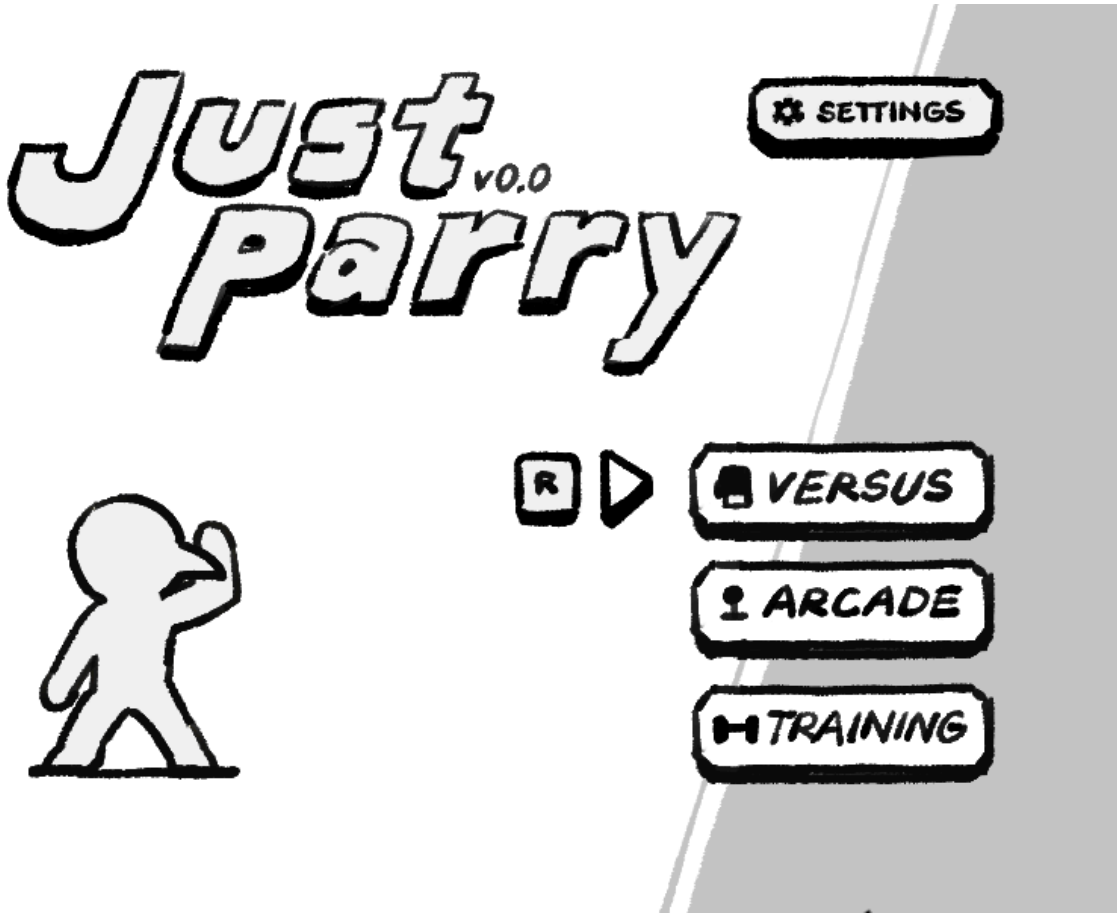
Just Parry is a simple 2D fighting game that focuses on parrying. Time your parry perfectly to counter your opponents, or punish their attempts to parry you.

Scenes:

The following mockups depict various interactions that are possible in Just Parry. Actual game assets will be simple pixel art with a limited palette, made in Aseprite.

MENU

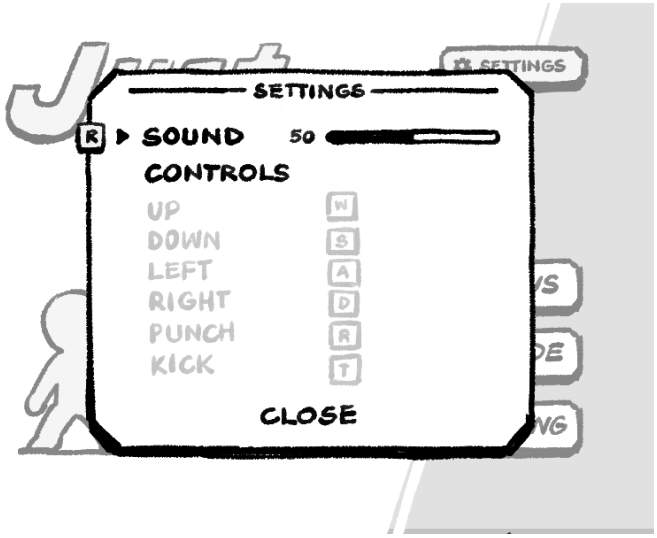
The menu is navigated using WASD and R/T if on a keyboard, D-PAD and A/B if on a controller. There is no cursor support, and input prompts are needed for players to know what buttons to use in the menu.



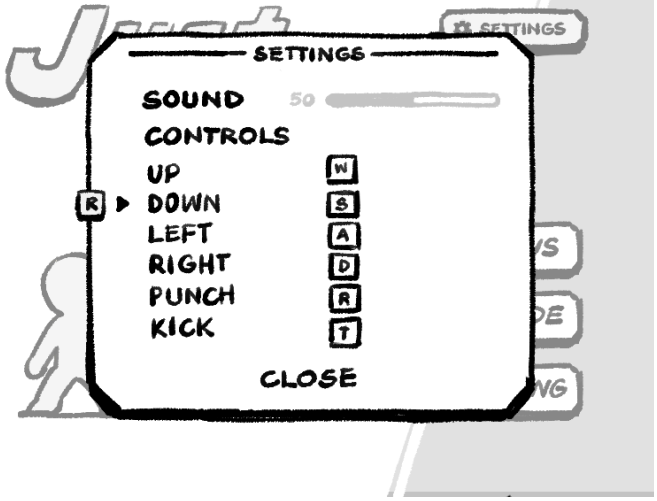
Fun potential feature (not prioritized): when pressing buttons to navigate the menu, the character moves in place according to the inputs.

Versus mode (local multiplayer) will be our main gameplay mode.

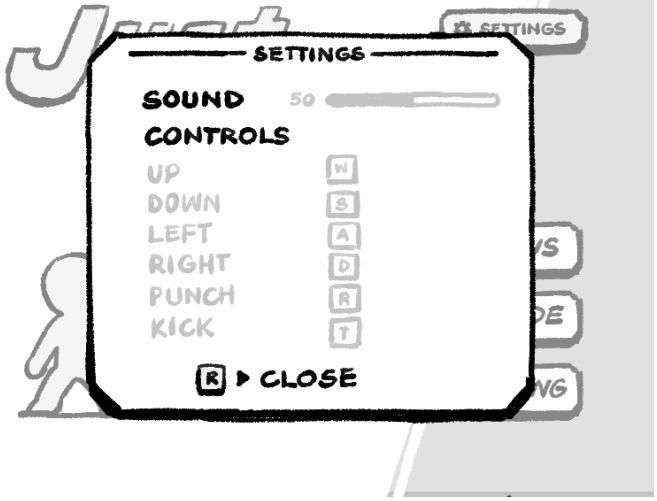
Settings:



Volume slider:
Controlled by left and right directional inputs



Button controls (P1):
Press confirm (A on XBOX, X on PS, R on keyboard) to select an action to rebind,
then press the button to rebind the action to

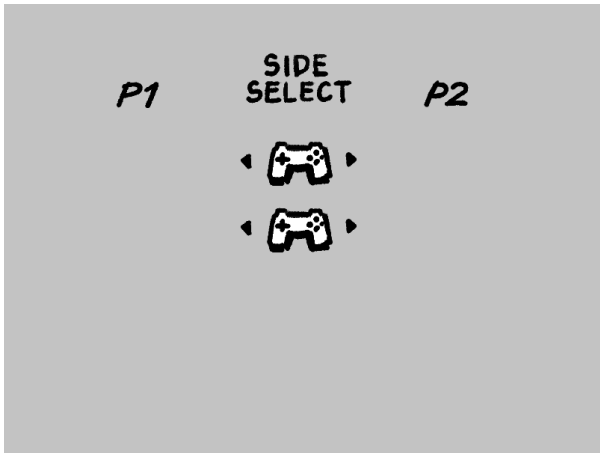


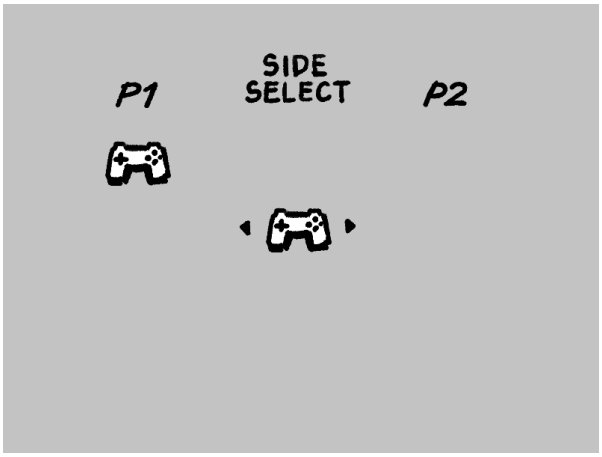
You can close the settings pop up by either:

1. pressing back (B on XBOX, O on PS, T on keyboard)
2. navigating to the close option and pressing confirm (A on XBOX, X on PS, R on keyboard)

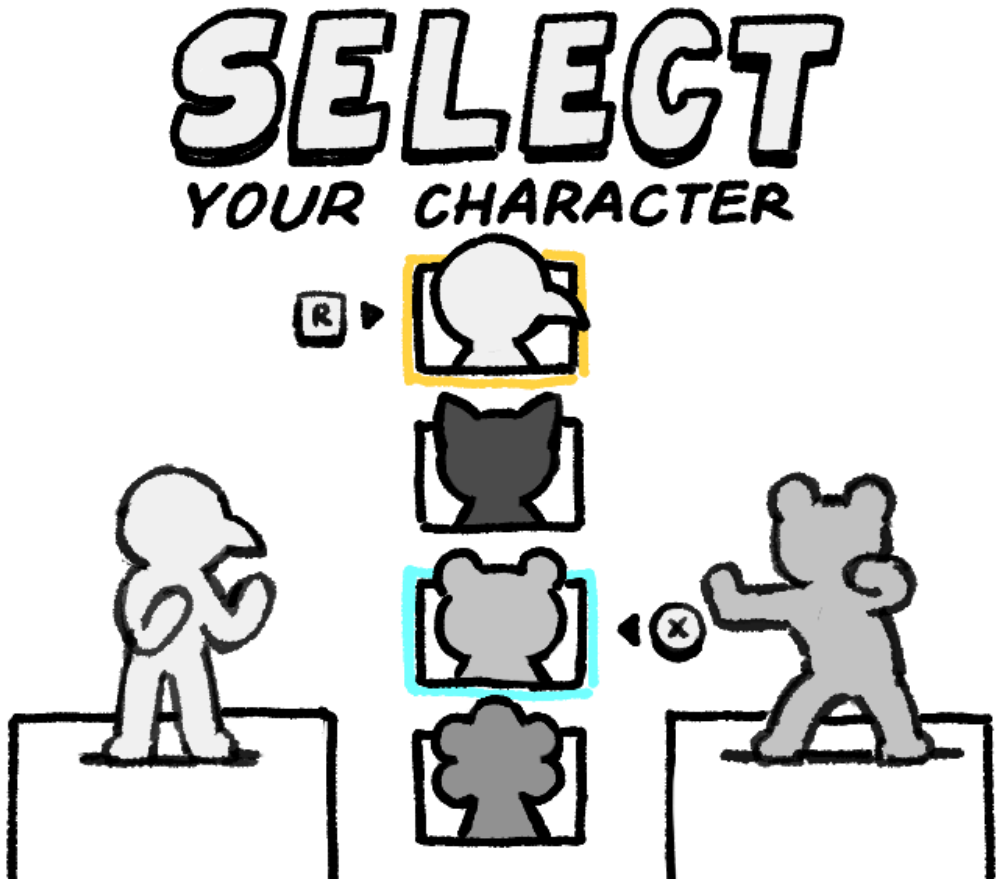
VERSUS MODE

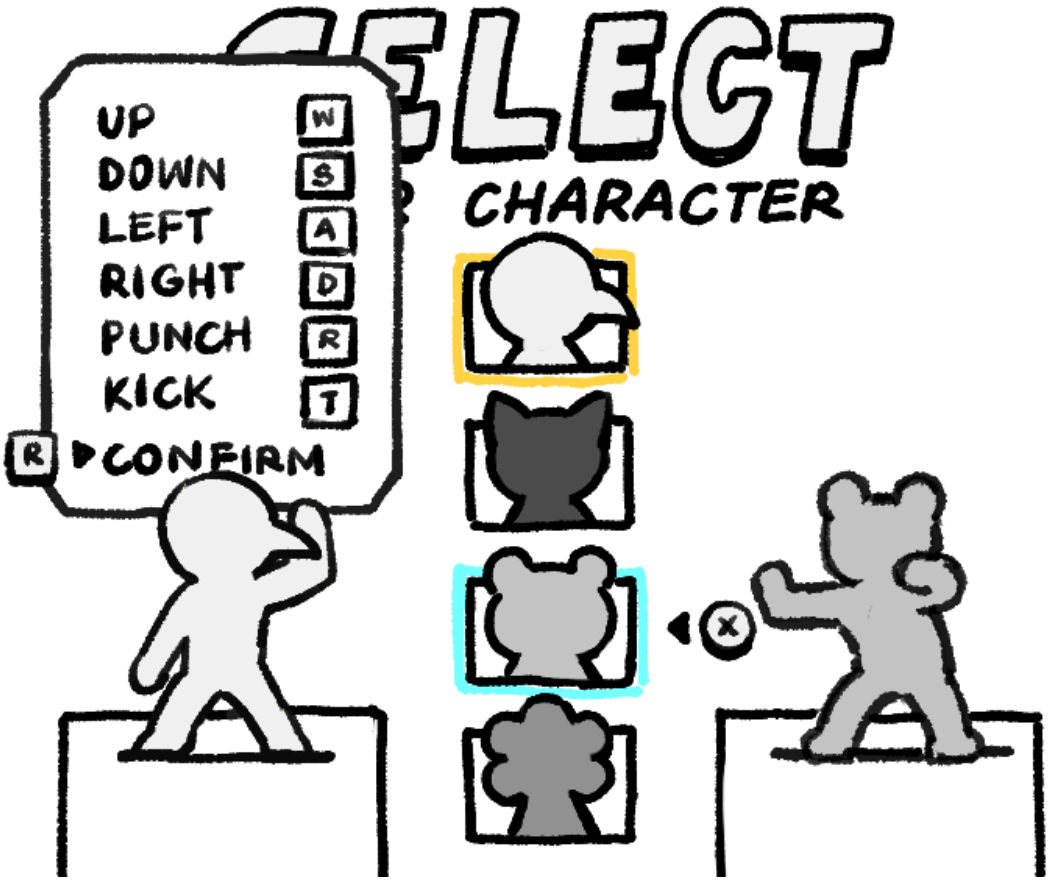
Side Select: All active input devices, including the keyboard, are displayed on the screen. Using directional inputs, players can select the side they want to be on.





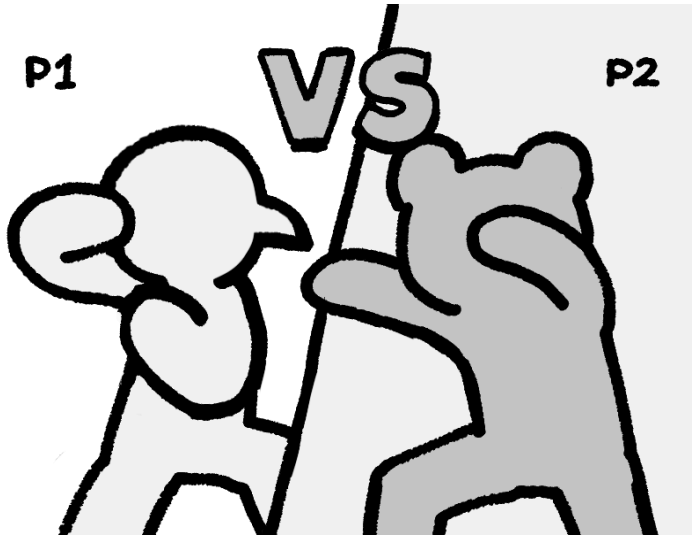
Character Select:



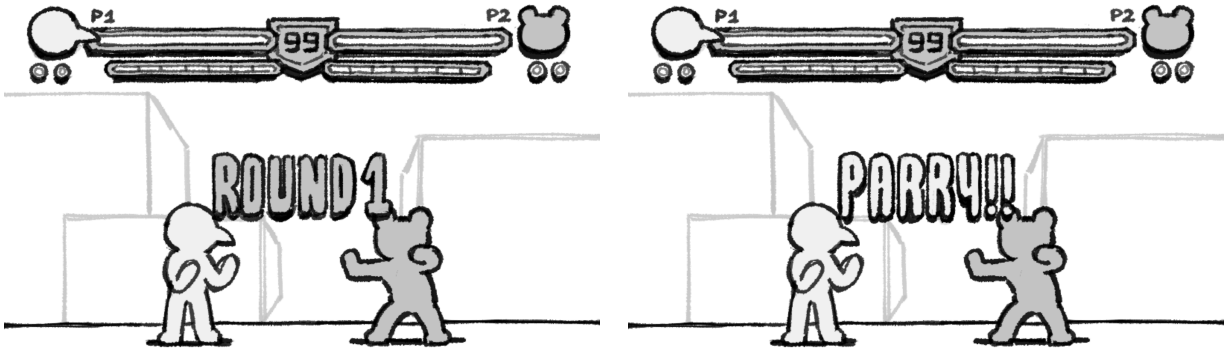


Once a character is selected, the player is prompted to confirm or change their controls. The character will also perform their parry animation on selection.

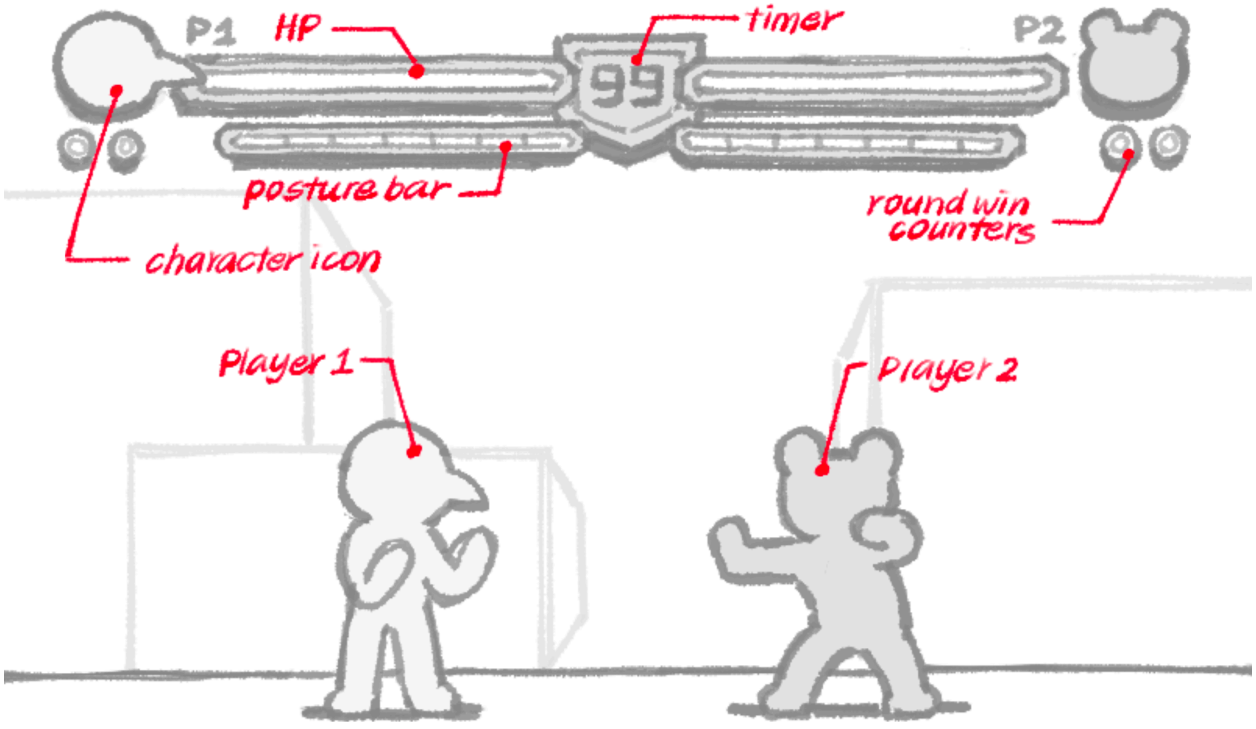
Loading: A simple splash screen generated from the characters selected on each side.



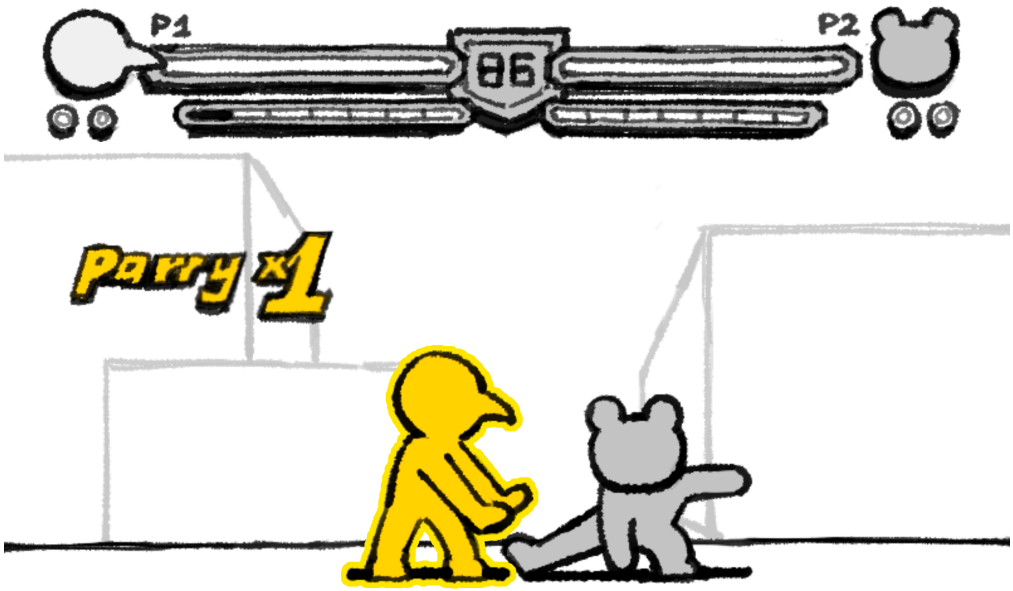
Round start: ROUND 1...PARRY!! appear on the screen with sound effects. Players have control over their characters once the text disappears.

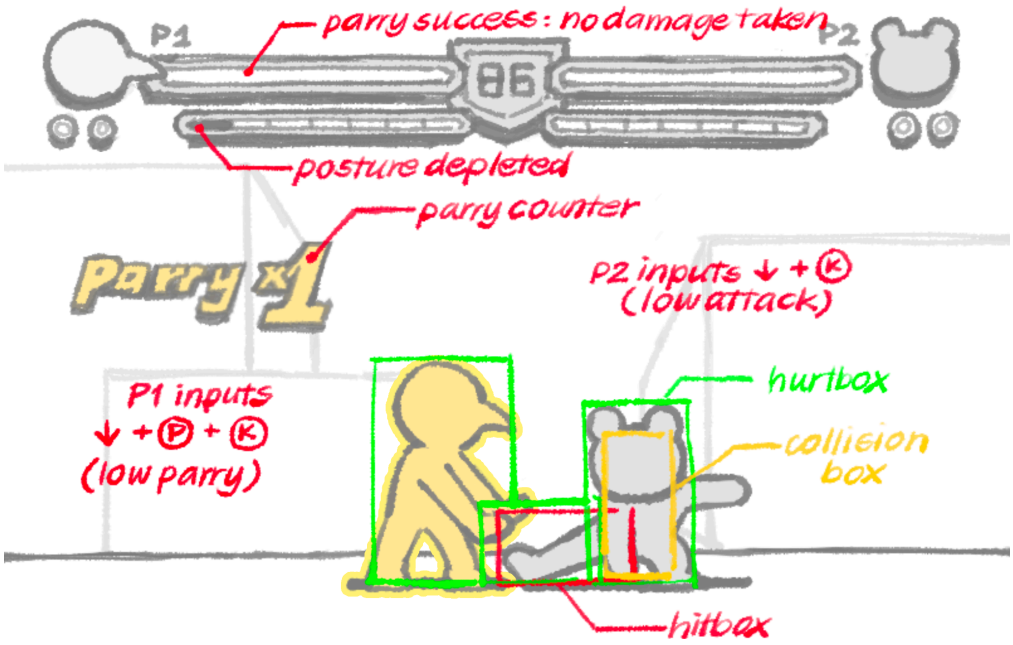


Initial Game State: Both players start with full HP (100) and posture (7). The timer begins at 99 seconds. Round win counters are reset. Players begin in neutral: a position where neither player can reach each other. Each parry consumes a portion of the posture bar.

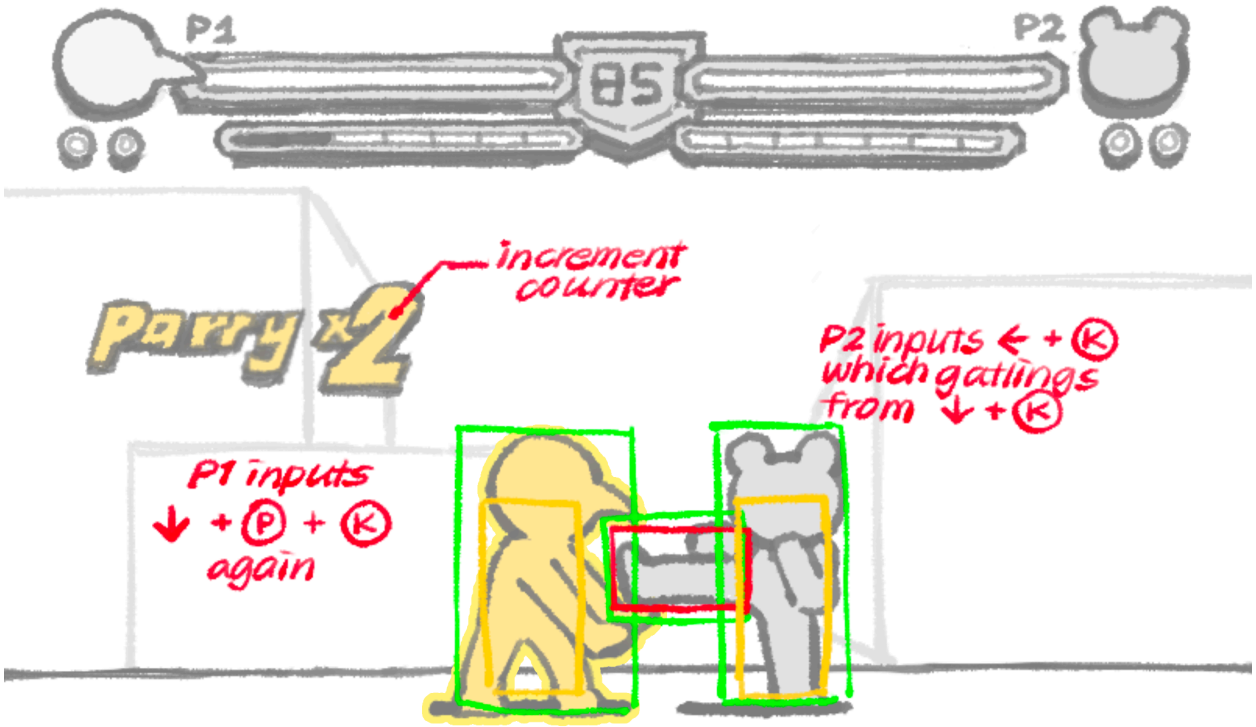


Situation 1: Successful Parry

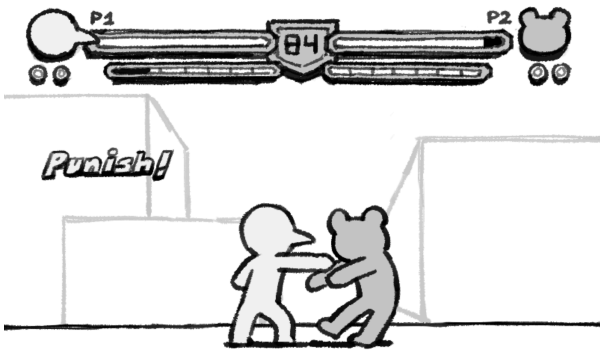


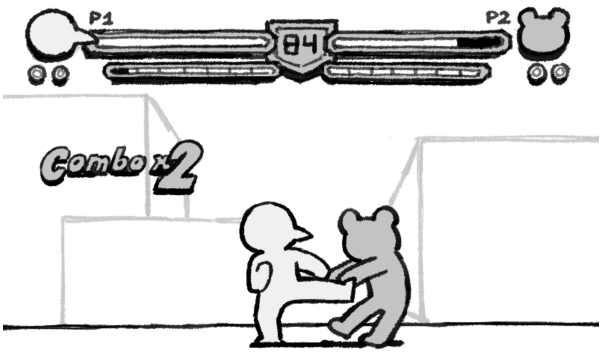


Situation 1 Continued:
P2 continues a gatling from down + kick, which is parried again by P1.



Forward + kick will have a lot of recovery, so P1 can then punish with a combo such as punch > kick.

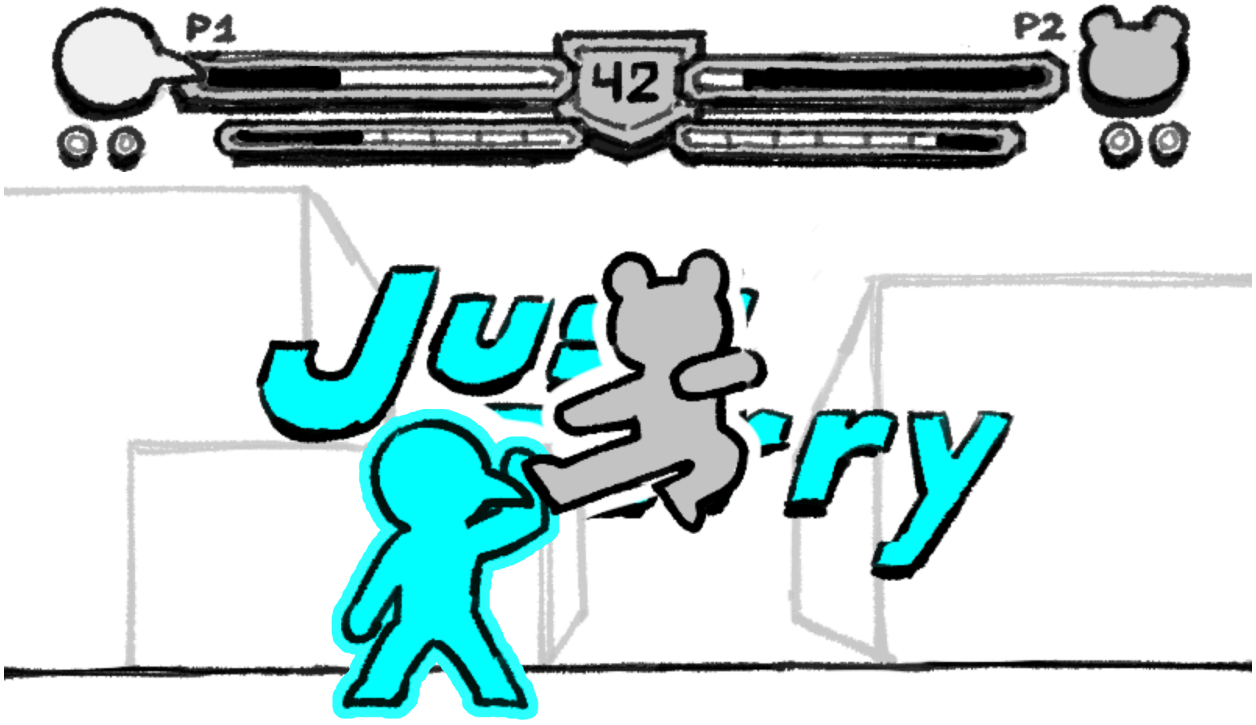


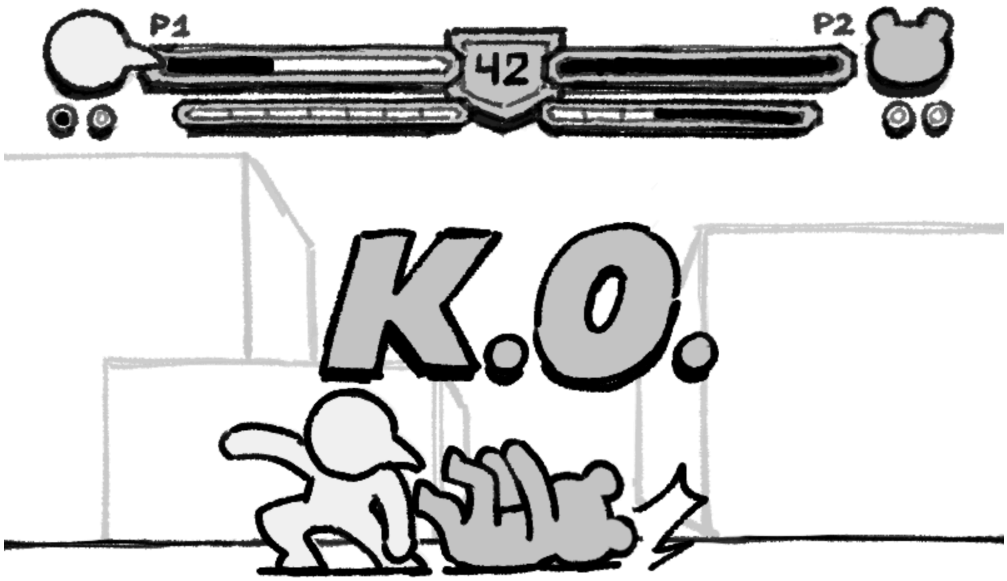


Note: there will be hit effects such as flash, screen shake, and particles to ensure that players are clear whether a hit landed or not.

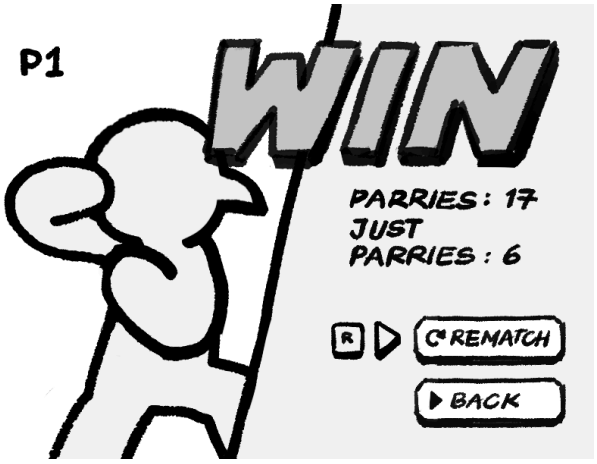
Situation 2: Successful Just Parry

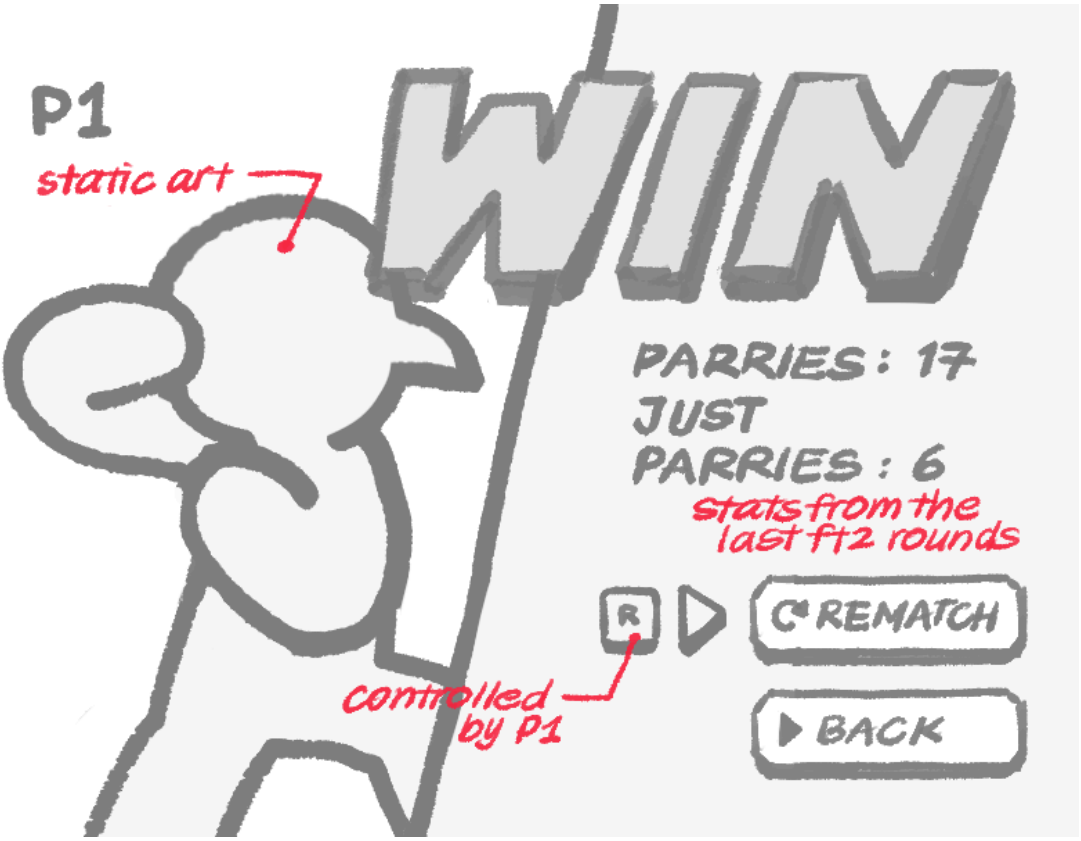
P1 inputs punch + kick (parry) 2 ± 1 frames before the P2's jump-in attack lands, resulting in a Just Parry. The screen freezes for a moment before the counter attack animation.





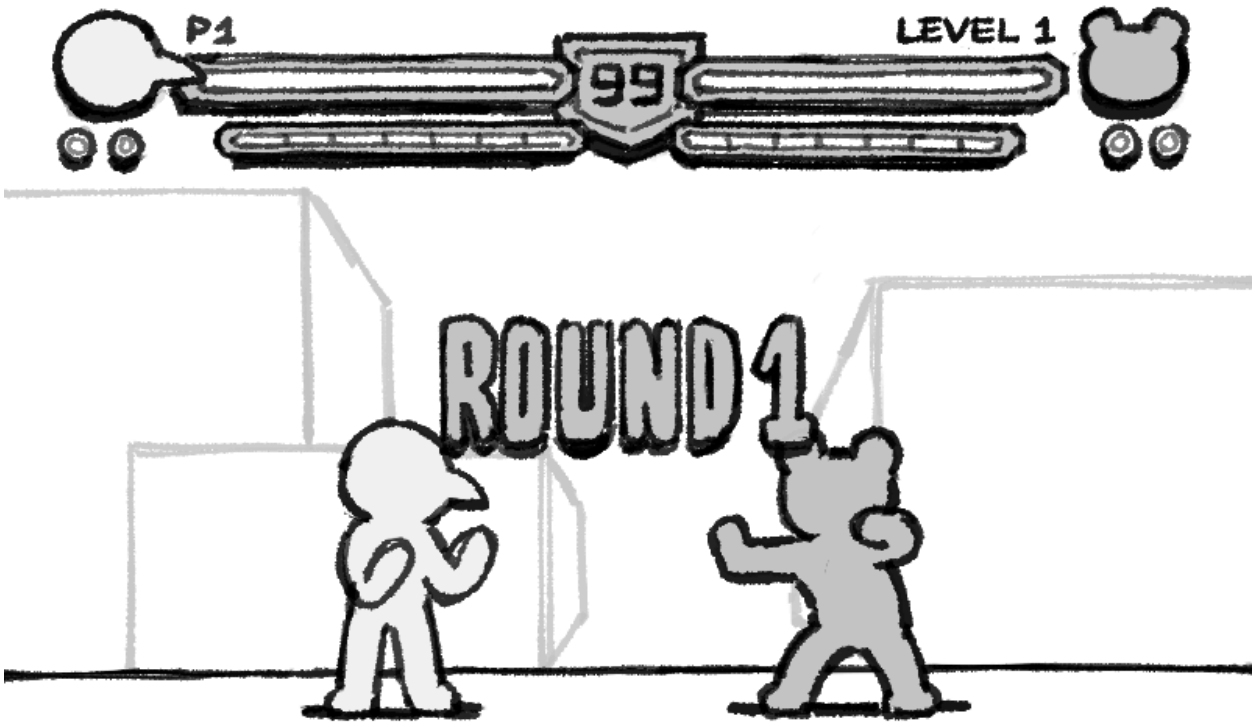
Win Screen: Occurs after a player wins the first to 2 (ft2) rounds. Here you have the option to rematch, which restarts the game with the same characters, or go back to character select.



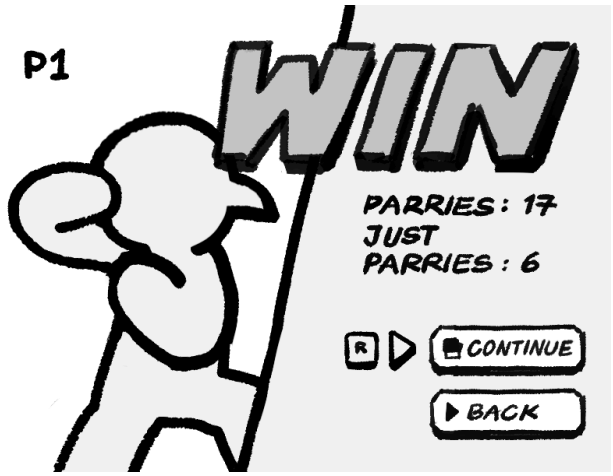


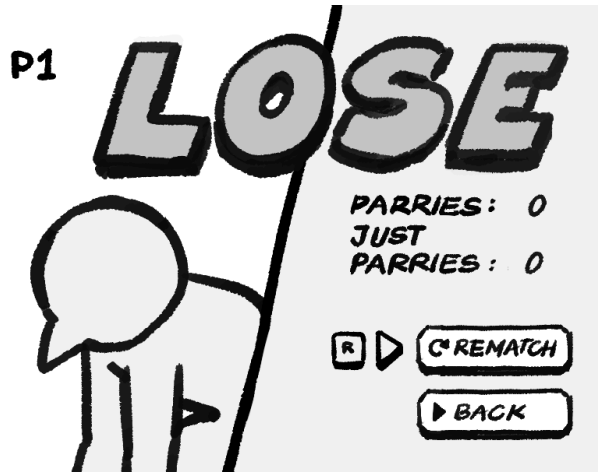
ARCADE MODE

Begin: Arcade mode puts you against an AI opponent that gradually increases in difficulty, for up to 10 levels. Each level is ft2, just like in versus mode.



Level Win: After winning an arcade mode level, you can choose to progress to the next level, or go back to the main menu. Losing will allow you to rematch against the same level.

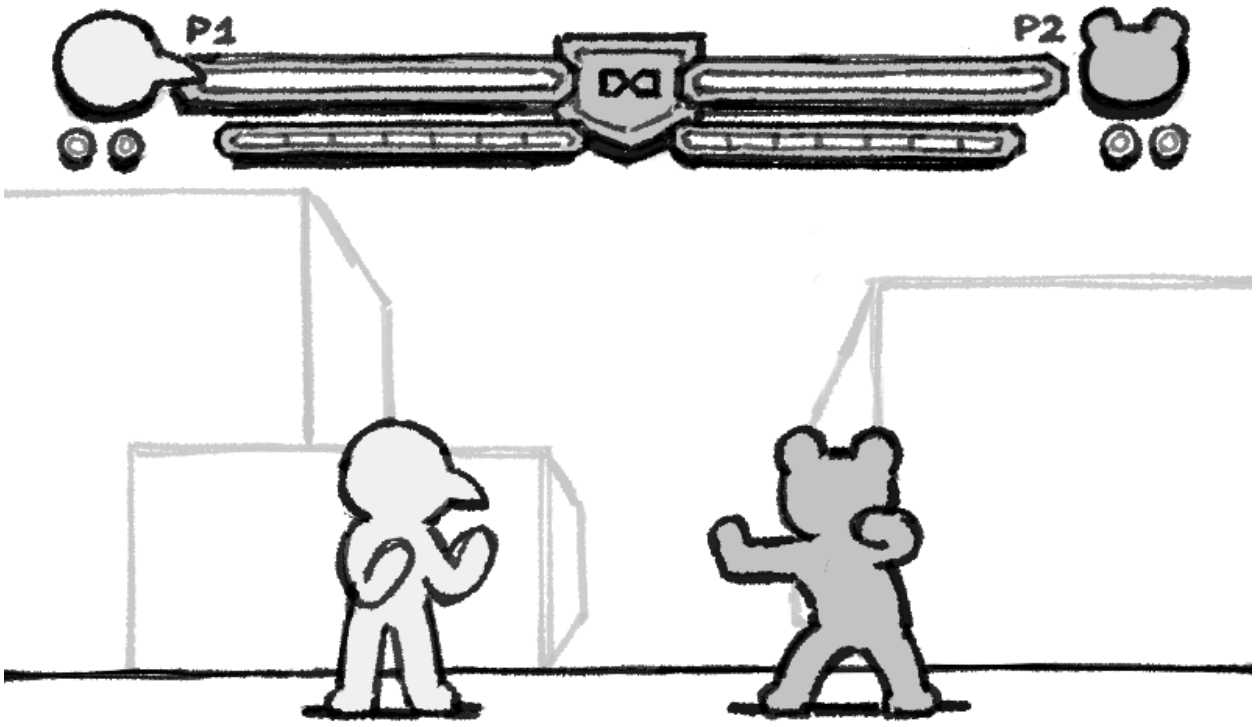




Level 10 Win: Winning against the level 10 AI will play an ending for the character (a short cutscene), before returning to the main menu.

TRAINING MODE

A simple mode where your opponent's health does not deplete no matter how much you strike them. For now, the opponent does not do anything and is merely a sandbag.

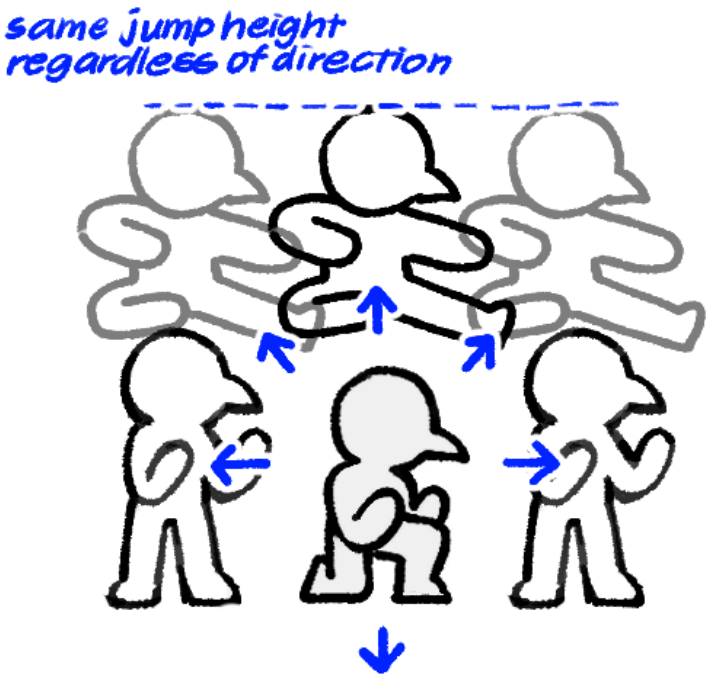


Technical Elements:

Our core systems are related to player vs. player combat. It is important that the game runs at exactly 60fps, as the combat system is directly tied to the frame rate.

Combat System

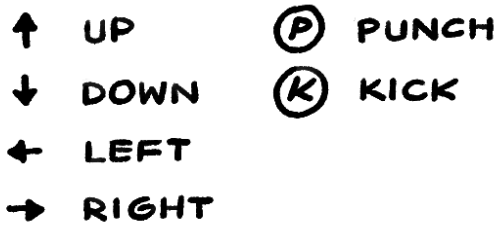
Basic Movement



- can move left and right
- can crouch
- can jump forward, jump back, and jump up
    - jump arcs are fixed

Inputs

Our input system will parse device input into the following inputs.



UP, DOWN, LEFT, and RIGHT are categorized as **directional inputs**.
PUNCH and KICK are **action inputs**.

Simultaneous inputs will need to be handled for cases such as:

- UP + RIGHT = jump forward
- DOWN + KICK = low kick
- PUNCH + KICK = parry

- DOWN + PUNCH + KICK = low parry

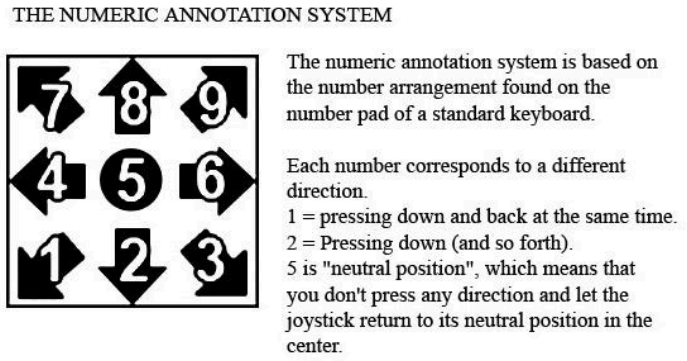
Characters

We aim to release at least 2 and up to 4 playable characters. To differentiate the characters, they will have different sprites, and they can have slightly different movelists, damage output, speed, parry reward, etc.

To start, we will focus on developing only our first character: BIRD. For ease of development, future characters will be based off of this character.

Movelist

A character has a set of moves they can use in battle. To simplify the inputs in this document, we will be using the numeric annotation system.



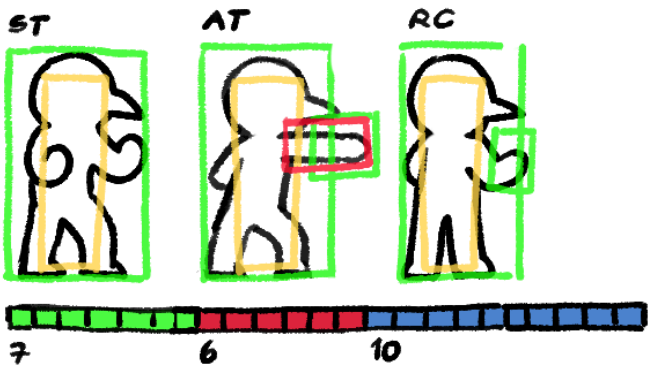
For example:

- KICK = **5P**
- DOWN + PUNCH = **2P**
- Parry (PUNCH + KICK) = **PK**
- Low Parry (DOWN + PUNCH + PARRY) = **2PK**

Here is a list of moves that are available to BIRD. All of the frame counts are subject to change once we are able to test and balance each move. The startup of each move takes into account that the human reaction speed is around 16 frames.

Input	*Description*	*Startup*	*Active*	*Recovery*	*Notes*
5P	Punch	7	6	10	
2P	Crouching Punch	7	5	10	
5K	Kick	17	9	19	
2K	Crouching Sweep	10	5	19	Hits low
j.P	Jumping Punch	7	-	-	Jumping attacks are active until landing, and recover instantly on land
j.K	Jumping Kick	7	-	-	
PK	Parry	1	[3] 12	26	The first three active frames are JUST PARRY frames.
2PK	Low Parry	1	[3] 12	26	

Frame data explanation:



Startup: how long it takes for a move to become active

Active: when the move has an active hitbox

Recovery: the move no longer has a hitbox and the player cannot do anything until the end of the recovery frames.

Each move (other than parry) will also do a set amount of damage on hit, and blockstun if it is parried. Blockstun is how many frames the opponent is in the parried animation before they can move again.

Parry
Parry is the primary defensive mechanic. Avoid taking damage by inputting parry within 12 frames of your opponent's attack, and counter attack by inputting a JUST parry within 3 frames.

Whether a parry lands is based on if the opponent's **hitbox** intersects with the player's **hurtbox** during its active frames.

Posture
Each parry consumes a portion of the player's posture bar, and once depleted, the player becomes unable to parry, leaving them vulnerable to further attacks.

The posture bar slowly regenerates as long as the player is neither attacking or parrying for more than 30 frames. The player also gains posture for landing a JUST Parry.

1. We will experiment with the number of segments of the posture bar, posture recovery rate and delay before the regeneration begins, as to not punish overly aggressive or defensive playstyles.
2. We adjust the posture damage inflicted by a normal parry (to yourself) and a perfect parry (to the opponent).

Stage
When either character is backed to the corner of the stage, they can no longer move backwards. The character backed against the corner can push the opponent back either by landing a JUST Parry or by landing combos.

Rendering

For the rendering of the game, we will focus on player sprites, parallax backgrounds, HUD elements, and various animations. The player sprites will be drawn using Aseprite, where multiple characters will have unique animations for movement, attacks, and parries. These animations will be exported as sprite sheets, with each action mapped to specific frames.

To manage the rendering, an OpenGL system will handle transformations using vertex shaders for smooth transitions between different states like Idle, Parry, or Attack. Also, the parallax background will be composed of two layers to create depth, where the foreground layer moves faster than the background. We will use translation matrices to move these layers based on the player's position.

The layers are as follows:

1. HUD elements such as health bars, posture bars, and round win counters are rendered above everything else and unaffected by game-world transformations.
2. Character sprites
3. Feedback such as the combo counter, parry counter, punish indicator, etc.
4. Background

2D Geometry Manipulation:

The game will rely on transformation matrices to handle the character movement and interactions, like translating, rotating, and scaling sprites during gameplay. Collision detection will use axis-aligned bounding boxes in the beginning Milestone Phases, given the 2D nature of the game, and we will move towards using Polygon or Pixel Collision. More precise pixel-perfect detection will need to be applied for attack and parry detection. We will need to implement a hitbox hurtbox system, where each frame of an attack will correspond to a unique hitbox.

AI (Arcade Mode)

For AI opponents in Arcade Mode, we will use a Finite State Machine to control their behavior, like attacking, moving, or parrying, based on player movements.

We plan for the AI to have random variations in their response to avoid predictable patterns, as well as making the AI more challenging as the Arcade Mode progresses.

- The AI will gain access to more user inputs the further the player gets
- Have AI perfect parry more often
- Have AI perform more combos
- Have AI always confirm into a combo

Physics

For physics, we will need to implement simple kinematic-based movement for the player characters. Involving velocity and acceleration to control the smoothness of the movements, as well as handling gravity for characters when jumping or being knocked back. Knockback physics will be important during combat, since successful attacks will push opponents back.

Advanced Technical Elements:

Input Buffer

- Allows us to process more inputs than there are opportunities to show for (each frame)
- It allows us to determine how inputs should be handled for example
    - If 2 players input at the same time what should the result be
    - How large is the time window it is to execute a motion input (ie: a sequence of inputs for a single attack)
    - Allows us to parse an input while the players are in a "non-actionable-state" (ie in some kind of stun) to execute when the player is actionable
    - If an input is sent between frames when should it be shown on screen
- If this were to not be included there would be a reduction in the precision of the multiplayer mode as well as the inability for motion inputs to be implemented

Local Multiplayer

- This will allow for this game to be replayable and enjoyable for multiple people to compete with one another in a competitive or casual way to increase replayability
- The game will be played on the same machine with 2 people on the keyboard or multiple input devices
- If this were not to be implemented we would need to focus in on our AI and "arcade mode" implementation

Custom Key Bindings

- It allows for greater accessibility and lets people choose a comfortable control scheme for them. This is most important for the case of 2 players on a keyboard
- If it were not to be implemented we would have to hard code in a control scheme that will be comfortable both for single player and multiplayer

Gamepad Support

- Allows for accessibility especially for those with motor issues
- As a stretch goal implement the input parsing of the analog stick to be mapped to a directional. Otherwise just button inputs
- Also allows for greater comfort for multiplayer

Training Mode

- A gamemode where there is an opponent who's health does not deplete so you can test how the game feels and learn the inputs of a character

- As a stretch goal, we can implement different options for the training mode
    - the training dummy attack at a set interval to allow player to practice parrying
    - the training dummy does certain combos to practice parrying against combos
- if this is not implemented there would not be a great consequence as it is largely a quality of life change

Multiple Stages (just art and maybe how long/short it maybe)

- Different art for the background of the main gameplay, Also potentially different lengths of the stage

Multiple Characters

- This would increase the amount of player expression and choices in what people might want to play with ie: slower and longer range, faster and shorter range
- BIRD - Balanced range and speed
- BEAR - Longer range but slower
- CAT - Shorter range but faster
- PLANT - Regains posture at a faster rate, but takes half a second longer to start regenerating

Devices

Keyboard:

- Player 1:
    - WASD : Up / Left / Down / Right
    - R / T: Punch / Kick
- Player 2:
    - UP LEFT DOWN RIGHT : Up / Left / Down / Right
    - < / >: Punch / Kick

Gamepad:

- DPAD (UP LEFT DOWN RIGHT): Up / Left / Down / Right
- A / B: Punch / Kick

Tools

Gainput:

- Our team will use the Gainput library to process inputs from various sources, to allow users to use a keyboard or controller to play our game. We will also be using Gainput to implement input buffering, using the build in input history function to track recent inputs and handle features such as combo detection

GLFW :

- Furthermore, our team will be using the GLFW library to assist in window management. We will also be using GLFW to handle input events, as GLFW allows for simultaneous input handling, which will be crucial as we will mainly be focusing on the multiplayer element of our game

GLEW:

- Library that helps to manage and load OpenGL functions. This simplifies accessing openGL by automatically loading the required function pointers based on the user's hardware allowing for compatibility over different OpenGL versions and different operating systems

Team management

We will assign tasks to team members based on strengths. Task points are assigned according to the estimated effort of the task, using the Fibonacci scale (1, 2, 3, 5, 8…) to better approximate the nature of the exponential growth in the complexity of tasks. Each member will provide an estimated delivery time for their assigned tasks.

Task status and other attributes will be updated and tracked using Trello.

If a task cannot be delivered on time due to unexpected complexity, the member responsible for that task must let the team know as early as possible. A 1-day grace day will be given for unexpected events. After the grace day, other team members will jump in to ensure our project is on track.

Member	*Role*	*Descriptions*
Shaun Gao	Core Mechanics	Implement core game mechanics, such as designing and implementing ECS, handle gameplay logics, ensure parry mechanics and other game features are present, and help with other things.
Siddh Patel	Core Mechanics	Work closely with Shaun on core game mechanics, lead the development in enemy AI
Armaan Sawhney	Sound, UI Design	Sound effects, background music, and UI Design, along with helping with other things (implementation, testing, and debugging…).
Roy Tao	HUD, UI Design	Design and implement HUD (health bar, timer…), and work closely with Armaan in designing the UI, and help with other things.
Nic Ung	Build Master	Environment setups, managing GitHub repo, potentially setup CI, ensuring the game compiles and each component integrates smoothly
Jenny Zeng	Rendering, Combat Design	Visual rendering, combat level design, character balancing and setting difficulty progression

Development Plan:

Provide a list of tasks that your team will work on for each of the weekly deadlines. Account for some testing time and potential delays, as well as describing alternative options (plan B). Include all the major features you plan on implementing (no code).

Milestone 1: Skeletal Game

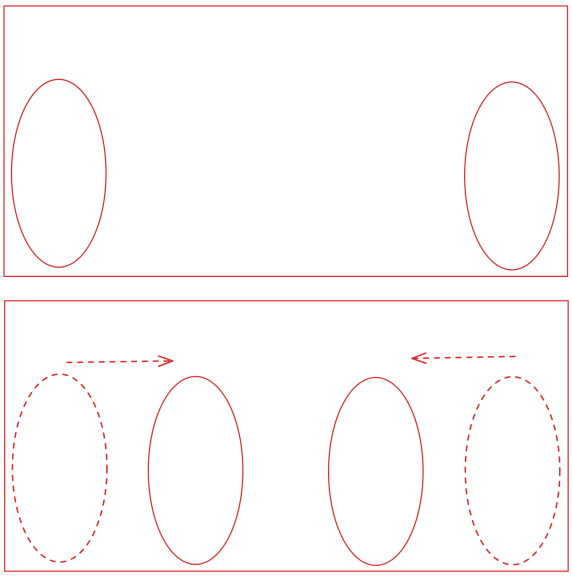
Sep 27 (Friday) - Oct 6 (Sunday) at 11:59pm

Week 1 (Until Oct 2 midpoint)

1. **Task 1: Set Up Development Environment**
    a. Install dependencies (OpenGL, other libraries...) and set up the OpenGL pipeline. Set up project folder/file structure.
    b. Reference: https://blog.bitbebop.com/gamedev-file-structure/
    c. Assigned to: Nic, Entire team
    d. Expected: Sep 29
2. **Task 2: Rendering - Basic Static Geometry and Background (8 points)**
    a. Add placeholder character models (rectangles or simple shapes for now). Add a simple floor and a temporary black background, making sure they are in the correct drawing order specified before.
    Stretch goal: timer, health bar, basic HUD
    b. Assigned to: Armaan, Roy
    c. Expected: Oct 2
3. **Task 3: Rendering - 2D Transformations (8 points)**
    a. Implement translation of rectangular players, without input at this point. Can be as simple as two models moving towards the center.
    Stretch goal: 180 degree rotation to "turn player around"
    b. Assigned to: Jenny
    c. Expected: Oct 2
4. **Task 4: Define Game-Space Boundaries (8 points)**
    a. Implement basic collision detection for the ground and screen edge walls.
    b. Sprites cannot walk into each other, and cannot walk outside of screen boundaries...
    c. Plan B: hardcode it
    d. Assigned to: Shaun, Siddh
    e. Expected: Oct 2

Week 2 (Oct 3 - Oct 6)

1. **Task 6: Basic Keyboard Handling (8 points)**
    a. Implement moving Player 1 rectangle by pressing WASD.
    b. Assigned to: Shaun, Siddh
    c. Expected: Oct 4
2. **Task 7: Key-Frame/State Interpolation (8 points)**
    a. Idea: Add this before starting the game, 2 players walk from 2 sides into the middle.
    b. Use linear interpolation to smoothly animate from one coordinate to another for both players.
    c. Reference: https://www.gamedev.net/tutorials/programming/general-and-gameplay-programming/a-brief-introduction-to-lerp-r4954/



    d. Assigned to: Armaan, Roy
    e. Expected: Oct 4
3. **Task 8: Random/Coded Action (8 points)**
    a. Implement basic randomized movement for Player 2 rectangle.
    b. Plan B: Hard code the action Player 2 performs instead of it being random.
    c. Assigned to: Siddh
    d. Expected: Oct 4
4. **Task 10: Test Plan (4 points)**
    a. Create a test plan that outlines how to user-test our game - a list of player or game actions and their expected outcomes. (/doc/test-plan.docx)
    b. Assigned to: Entire Group
    c. Expected: Oct 5

5. **Task 11: Bug List Report (4 points)**
    a. Compile a known bug-list report, if we are aware of bugs in the Skeletal Game that we cannot fix on time. Google Sheets or Microsoft Excel spreadsheet. (/doc/bug-report.xlsx)
    b. Assigned to: Entire Group
    c. Expected: Oct 5
6. **Task 12: Demonstration video (8 points)**
    a. As a group, create a max 3 min demonstration video that showcases our Skeletal Game features. (YouTube link or .mp4)
    b. Assigned to: Entire Group
    c. Expected: Oct 6

Milestone 2: Minimal Playability

Due Oct 27 (Sunday) at 11:59pm

Week 1 (Oct 7 – Oct 13)

1. **Task 1: Game logic response to user input. (15 points)**
    a. Implement state and decision tree driven (possibly randomized) responses to user input and game state (create a simple decision tree data structure and reuse it for multiple entities).
    b. Reference: https://www.gamedev.net/articles/programming/artificial-intelligence/the-totalbeginners-guide-to-game-ai-r4942/
    c. Assigned to: TODO
    d. Expected: Oct 10
2. **Task 2: Implement Sprite Animations (15 points)**
    a. Sprite animations using individual images or sprite sheets.
    b. Reference : https://www.piskelapp.com/
    c. Assigned to: TODO
    d. Expected: Oct 12
3. **Task 3: Implement Sprite and Background Assets (10 points)**
    a. Sprite and background assets as well as corresponding actions to enable interesting gameplay.
    b. Reference: background asset libraries
    c. Assigned to: TODO
    d. Expected: Oct 12
4. **Task 4: Implement Mesh - Based Collision Detection & Resolution (10 points)**

a. The mesh must be of non-trivial shape (e.g. rectangles and circles are trivial). The mesh collision can be with a simple object (e.g. mesh-line or mesh-box collisions, mesh-mesh is not required), and some assumptions about the meshes can be made (e.g. only working with convex meshes). Check with TA for what would be acceptable.
b. Assigned to: TODO
c. Expected: Oct 13

Week 2 (Oct 14 - Oct 20)

1. **Task 5: Implement Basic User Tutorial / Help (10 points)**
    a. Simple tutorial pointing to basic buttons, and examples of how to play
    b. Assigned to: TODO
    c. Expected: Oct 17
2. **Task 6: Implement FPS counter (toggle on/off with "F") (5 points)**
    a. Simple frame counter at top left/right of the screen, toggle on/off with F key
    b. Reference : https://www.opengl-tutorial.org/miscellaneous/an-fps-counter/
    c. Assigned to: TODO
    d. Expected: Oct 19
3. **Task 7: Ensure non-repetitive gameplay using all features for 2 mins (5 points)**
    a. Sustain progressive, non-repetitive gameplay using all required features for 2 min or more (for now you can assume that you can provide users with oral instructions). During these 2 minutes, the player should be able to interact with the game and see new content for most of the time.
    b. Assigned to: TODO
    c. Expected: Oct 19
4. **Task 8: Ensure minimal lag and improve performance (2.5 points)**
    a. The game does not crash or stop responding at any stage.
    b. Assigned to: TODO
    c. Expected: Oct 20
5. **Task 9: Ensure no crashes, bugs, or unexpected behavior (2.5 points)**
    a. The game maintains a consistent frame rate (as shown by the FPS counter above)
    b. Assigned to: TODO
    c. Expected: Oct 20

Week 3 (Oct 21 - Oct 27)

6. **Task 10: Creative Elements (20% of the milestone)**

    a. To receive full creative credits, the game should have either two basic features or one advanced feature outside the mandatory requirements. Examples of tasks you should be able to complete at this stage include adding more advanced rendering effects (e.g. basic physics and parallax scrolling backgrounds), complex gameplay logic, or a significant number of manually created sophisticated (i.e., not purchased) assets.
    b. Reference: https://canvas.ubc.ca/courses/147789/pages/suggested-game-features
    c. Assigned to: TODO
    d. Expected: Oct 25
7. **Task 11: Updated test plan (2.5 points)**
    a. Update list of player or game actions and their expected outcomes. (/doc/test-plan.docx)
    b. Assigned to: ALL
    c. Expected: Oct 26
8. **Task 12: Updated bug list - includes open and closed bugs. (2.5 points)**
    a. Update Google Sheets or Microsoft Excel spreadsheet. (/doc/bug-report.xlsx)
    b. Assigned to: ALL
    c. Expected: Oct 26
9. **Task 13: Demonstration Video (5 points)**
    a. 4 mins. max, showcasing all required and creative features
    b. Assigned to: ALL
    c. Expected: Oct 26

Milestone 3: Playability

Due Nov 17 (Sunday) at 11:59pm

Week 1 (Oct 28 -Nov 3)

1. **Task 1: Design - 5 Minutes of Non-Repetitive Gameplay**
    a. Propose an implementable design for progressive levels of AI opponents (3 levels, easy, medium, hard).
    b. Difficulty increases by increasing how many perfect parries the AI lands, based on how many wins the current player has.
    c. More wins = harder levels.
    d. Assigned to: TODO
    e. Expected: Oct 29
2. **Task 2: Implement - 5 Minutes of Non-Repetitive Gameplay (15 points)**

    a. Implement AI components following the design
    b. Plan B: Use random AI attack patterns to add unpredictability to the game.
    c. Assigned to: TODO
    d. Expected: Nov 1
3. **Task 3: Handle All User Input (5 points)**
    a. Ensure the game handles all user input without crashing, including pressing invalid keys. Ensure sprite behaviors are expected when keys are pressed. Identify and test edge cases.
    b. Do the keyboard inputs first, and then finish off with joystick controls.
    c. Plan B: Use a whitelist for keyboard input, or disable unaccepted input. Hard code to resolve complex scenarios.
    d. Assigned to: TODO
    e. Expected: Nov 3
4. **Task 4: Real-Time Gameplay (5 points)**
    a. Ensure the game is smooth. If it's laggy, identify the bottleneck and optimize the performance.
    b. Plan B: Suppress CPU/GPU intensive features to ensure the game is smooth, and try refactoring for improved performance.
    c. Assigned to: TODO
    d. Expected: Nov 6

Week 2 (Nov 4 - Nov 10)

TODO: We need to select a maximum of 40% from the optional features.

1. **Task 5: Parallax Scrolling Backgrounds (10%) (Optional)**
    a. Implement Multiple background layers (at least 3) that create a parallax effect upon camera motion.
    b. Plan B: Use less layers (1-2).
    c. Assigned to: TODO
    d. Expected: Nov 10
2. **Task 6: Implement Particle Systems (Optional)**
    a. Simulate dust/blood/parry effect using a particle system. Use instanced rendering to render many instances of the same object more efficiently. See also: https://en.wikipedia.org/wiki/Particle_system
    b. Plan B: Implement very basic effect.
    c. Assigned to: TODO
    d. Expected: Nov 10
3. **Task 7: Implement Physics-Based Animation (20%) (Optional)**

    a. The sprite will have an initial speed burst (0.2-0.5s) after it's been standstill, then slow down to normal constant speed. Simulating the initial acceleration.
    b. Reference: Euler's Method - https://www.youtube.com/watch?v=Vz6l-tLmlAo
    c. Plan B: Dismiss this one. Choose another feature to implement.
    d. Assigned to: TODO
    e. Expected: Nov 10
4. **Task 8: Implement Precise Collisions (20%) (Optional)**
    a. Implement precise collision detection to handle overlaps between hitboxes and hurtboxes. Precision collision detection is essential to frame-perfect detection.
    b. Plan B: Get frame detection to as perfect as possible.
    c. Reference: https://developer.mozilla.org/en-US/docs/Games/Techniques/2D_collision_detection
    d. Assigned to: TODO
    e. Expected: Nov 11
5. **Task 9: Develop Advanced Decision-Making AI Opponent (20%) (Optional)**
    a. Design and implement Finite State Machine for the AI.
    b. Plan B: Use a different AI algorithm.
    c. Assigned to: TODO
    d. Expected: Nov 11

Week 3 (Nov 11 - Nov 17)

1. **Task 10: Memory Management (5 points)**
    a. Ensure memory is managed properly. Use tools like Valgrind to find and resolve leaks. If there are issues with memory usage, assess the risks and prioritize issues on the bug list.
    b. Plan B: If time is short, only fix significant issues.
    c. Assigned to: ALL
    d. Expected: Nov 5
2. **Task 11: Oversees Stability (20 points)**
    a. Fix any prior missed milestone features & bug fixes. Ensure consistent game resolution, No crashes, glitches, unpredictable behavior.
    b. Assigned to: ALL
    c. Expected: Nov 14
3. **Task 12: Updated test plan (2.5 points)**

    a. Update list of player or game actions and their expected outcomes. (/doc/test-plan.docx)
    b. Assigned to: ALL
    c. Expected: Nov 16
4. **Task 13: Updated bug list - includes open and closed bugs. (2.5 points)**
    a. Update Google Sheets or Microsoft Excel spreadsheet. (/doc/bug-report.xlsx)
    b. Assigned to: ALL
    c. Expected: Nov 16
5. **Task 14: Demonstration Video (5 points)**
    a. 5 mins. max, showcasing all required and creative features
    b. Assigned to: ALL
    c. Expected: Nov 16

Milestone 4: Final Game

Due Dec 1 (Sunday) at 11:59pm

Week 1 (Nov 18 - Nov 24)

Mandatory Requirements (70% grade)

1. **Task 1: Playability (15%): Ensure playability for 10 minutes**
    a. During this time, players should be able to interact with the game and see new content most of the time.
    b. Assigned to: TODO
2. **Task 2: Scalability (15%):**
    a. Fix all prior milestone bugs
    b. Ensure all previous milestone requirements are completed
    c. Assigned to: TODO
    d. Expected: Nov 20
3. **Task 3: Scalability (15%):**
    a. Ensure game resolution & aspect ratio are consistent across different machines/ screens.
    b. Assigned to: TODO
    c. Expected: Nov 21
4. **Task 4: Scalability (15%):**
    a. Ensure proper loading, and exiting of the game.
    b. Assigned to: TODO
    c. Expected: Nov 21
5. **Task 5: User Experience (10%):**

a. Include a tutorial introducing the player to the game mechanics. The game should be self-explanatory with no verbal explanation required at any point during the gameplay.
b. Evaluate and optimize user-game interactions (choice of user gestures, ease of navigation, etc.).
c. Assigned to: TODO
d. Expected: 23

6. **Task 6: Robustness: Memory Management (5%)**
    a. Include proper memory management (no memory hoarding or leaks, the game should not hog memory even after extended play time).
    b. Assigned to: TODO
    c. Expected: Nov 23
7. **Task 7: Robustness: User Input (5%)**
    a. The game should robustly handle any user input. Unexpected inputs or environment settings should be correctly handled and reported, and not crash the game. For example, tabbing out of the game, minimizing the window, hitting invalid keys, etc.
    b. Assigned to: TODO
    c. Expected: Nov 23
8. **Task 8: Robustness: Real-time performance (5%):**
    a. The gameplay should be real-time, i.e., there should be no input lag or stuttering. Use a profiler to locate runtime bottlenecks and resolve them. Report on your performance testing in your video, especially if you find and correct anything significant.
    b. Assigned to: TODO
    c. Expected: Nov 24

Week 2 (Nov 25 - Dec 1)

Creative Component (30% grade) : Note that you must verify any features not listed below with your TA before submission to ensure that they agree with your “advanced” or “basic” classification; failure to do so will result in the TA deciding unilaterally at their discretion.

Reference : https://canvas.ubc.ca/courses/147789/pages/suggested-game-features

9. **Task 1: External Integration (up to 10%)**
    a. Include integration of one or more external tools or libraries: physical simulation (PhysX, Bullet, ODE, etc.), or alternatives.
    b. Assigned to: TODO

c. Expected: Nov 26

10. **Task 2: Advanced Graphics (up to 20%)**
    a. Implement an advanced graphics feature such as visual effects (Particle Systems, 2.5D (3D) lighting, 2D dynamic shadows), or advanced 2D geometric modifications (2D deformations, rigged/skinned motion).
    b. Assigned to: TODO
    c. Expected: Nov 26
11. **Task 3: Advanced Gameplay (up to 20%)**
    a. Implement an advanced gameplay feature such as advanced decision-making mechanisms based on goals (path planning, A*, or similar), advanced group behavior (e.g. coordination between enemies), or more complex physical interactions with the environment (e.g. gravity, bouncing, complex dynamics).
    b. The physical effects implemented should be correctly integrated in time and should not be locked to the machine's speed by correctly handling the simulation time step and integration.
    c. Assigned to: TODO
    d. Expected: Nov 28
12. **Task 4: Audio (up to 10%)**
    a. Add audio feedback for all meaningful interactions in the game as well as background music with tones reflecting the journey of the game.
    b. Assigned to: TODO
    c. Expected: Nov 28
13. **OPTIONAL: OTHER (up to 10% / 20%)**
    a. As an alternative to the above you can implement a selection of basic (10%) or advanced (20%) features listed in the reference which were not part of prior milestones.
    b. Reference: https://canvas.ubc.ca/courses/147789/pages/suggested-game-features
    c. Assigned to: TODO
    d. Expected: Nov 30
14. **Task 5: Reporting (5%): Update & reprioritize Bug List/Doc**
    a. Anything below P3 (P0-P2) should be resolved before submission or reclassified P4 or P5 if not critical.
    b. Anything above P3 should be justifiably not required for this release.
    c. For P3, you will either resolve the bugs or you reprioritize to critical (fix before ship) or non-critical (fix after ship, next version, or won't fix).
    d. Assigned to: TODO
    e. Expected: Nov 30
15. **Task 6: Reporting (5%): Create Video Report**

a. Video report detailing the game features, highlighting new creative features, any significant bug fixes, and the evolution of your game from concept to final release.
b. Video should be no more than 6 minutes in length.
c. Assigned to: TODO
d. Expected: Nov 30

16. **Task 7: Reporting (5%): Create User Testing Report**
    a. Report on the user testing you performed, including user feedback, and describe the changes you implemented in response to the feedback in your video report.
    b. Assigned to: TODO
    c. Expected: Nov 30