Game Proposal: Feral Freedom

CPSC 427 – Video Game Programming

Team: Stray Souls

Selin Uz – 55505440

Vincent Lee – 84258847

Shuang Qiu – 96241351

Nicolás Serrano de la Paz – 30253751

Cengiz Kagan Kuscu – 93079598

Siri Phanichkrivalkosil – 52080348

Story:

Briefly describe the overall game structure with a possible background story or motivation. Focus on the gameplay elements of the game over the background story.

On July 29th, 2024, the Turkish Parliament voted in favour of a bill that effectively required stray animals to be immediately culled unless they were adopted and registered to a human. This law has resulted in indiscriminate cullings and persecution of stray animals, with mass animal graves being found earlier this year.[1] It is from the compassion and solidarity we felt towards and with these stray animals that our game was born.

The player (henceforth referred to as ‘you’) will play as a stray animal in a city where the ‘Animal Patrol’ is out to get any and all strays that they find. Your objective is to find a path to freedom; get adopted by a friendly human or escape this hostile human civilization and live freely in the wilderness. If your run goes well and you reach a successful outcome, you’ll unlock a new character, with new base stats that invite you to try a different playstyle.

Your default character is a cat with 9 lives and base stats. You will traverse a 2D top down open world, moving through a City, from Street to Street, where there are Rooms at each corner that you can enter and exit. There are other beings that exist in this dystopian (from a stray perspective) world alongside you.

There is the ‘Animal Patrol’, who roam the Streets and are on the lookout for strays like you; they will catch you if they get the chance. When they start chasing you, be careful to not be caught! You can try to outsmart them or hide in bushes or trash cans to evade them, but if the

[1] Turkey’s Stray Animal Crisis: Mass Graves Discovered After Controversial Law Was Passed - World Animal News. August 23, 2024

animal patrol catches you, you'll lose all of your items, one of your lives, and have to start your path to freedom again (though you'll keep some of your skills and experience).

In Rooms and on the Streets, you will find items as well as humans and/or other strays. You can interact with them as you wish. They may have items (such as food and new collars that will buff your stats) which they may give you freely. Some may also have wisdom to share with you. But in most cases, you will find yourself in a turn-based encounter in which you'll have to intimidate, charm, or defeat the human or stray in order to progress or get the item they are safekeeping. If the encounter goes on and you feel success slipping away, you have the option to try and run away to live another day. What approach you take with each encounter is up to you and your probability of succeeding will depend on your stats and strategy.

Regardless of how you progress through an encounter, the actions you take (charm, intimidate, fight) during the encounter will provide you with experience. This will improve your stats and prepare you to face future encounters with a higher chance of succeeding. Each action, however, will drain your energy. Keep your eye on your energy meter, if it runs out you will pass out and the 'Animal Patrol' will deploy more agents to your location, increasing their presence on the Streets of the City you are in. To avoid running out of energy, you can sleep in specific safe locations or consume energy-replenishing items.

As a warning, be wary of engaging others in combat; you may end up in a situation where you are fighting for your life and if you lose all of your HP, you will (like when you're caught by the 'Animal Patrol') lose your items, one of your lives, and start your path from zero.

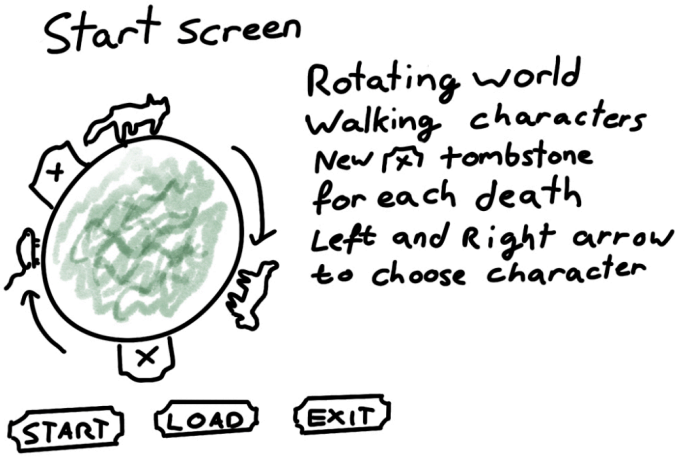
You may have noticed that if you still have lives remaining, though you start again from the beginning, you get to keep your stats. But if you lose all of your lives, it's a different story. As the default character, if you lose all 9 lives, you'll have to start over with the default stats.

Similarly, if you're playing as an unlocked character and lose all your lives, that character will be reverted to its base stats. You will notice a tombstone on the loading screen that represents this character. Survival in this world isn't easy for a stray, and there are multiple strays to discover and potentially lead to freedom. Explore the world's various paths and storylines and strategically manage your character's lives as you chase the version of freedom that best suits your current character.

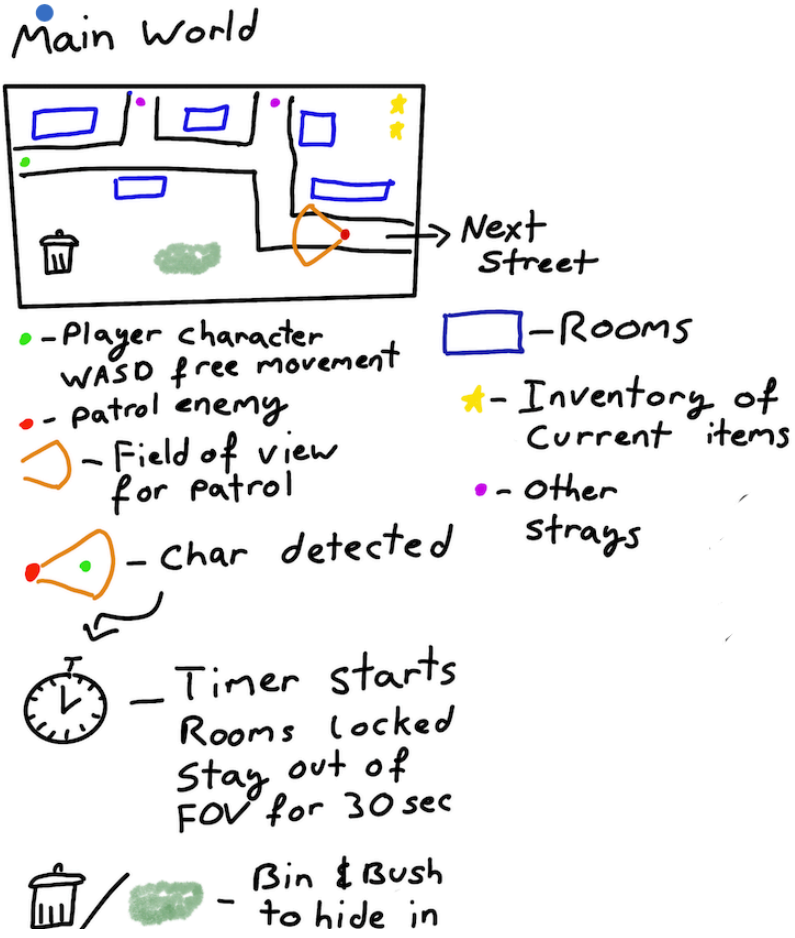
Scenes:

Produce basic, yet descriptive, sketches of the major game states (screens or scenes). These should be consistent with the game design elements, and help you assess the amount of work to be done. These should clearly show how players will interact with the game and what the outcomes of their interactions will be. For example, jumping onto platforms, shooting projectiles, enemy pathfinding or 'seeing' the player. This section is meant to demonstrate how the game will play and feeds into the technical and advanced technical element sections below. If taking inspiration from other games, you can include annotated screenshots that capture the gameplay elements you are planning to copy.

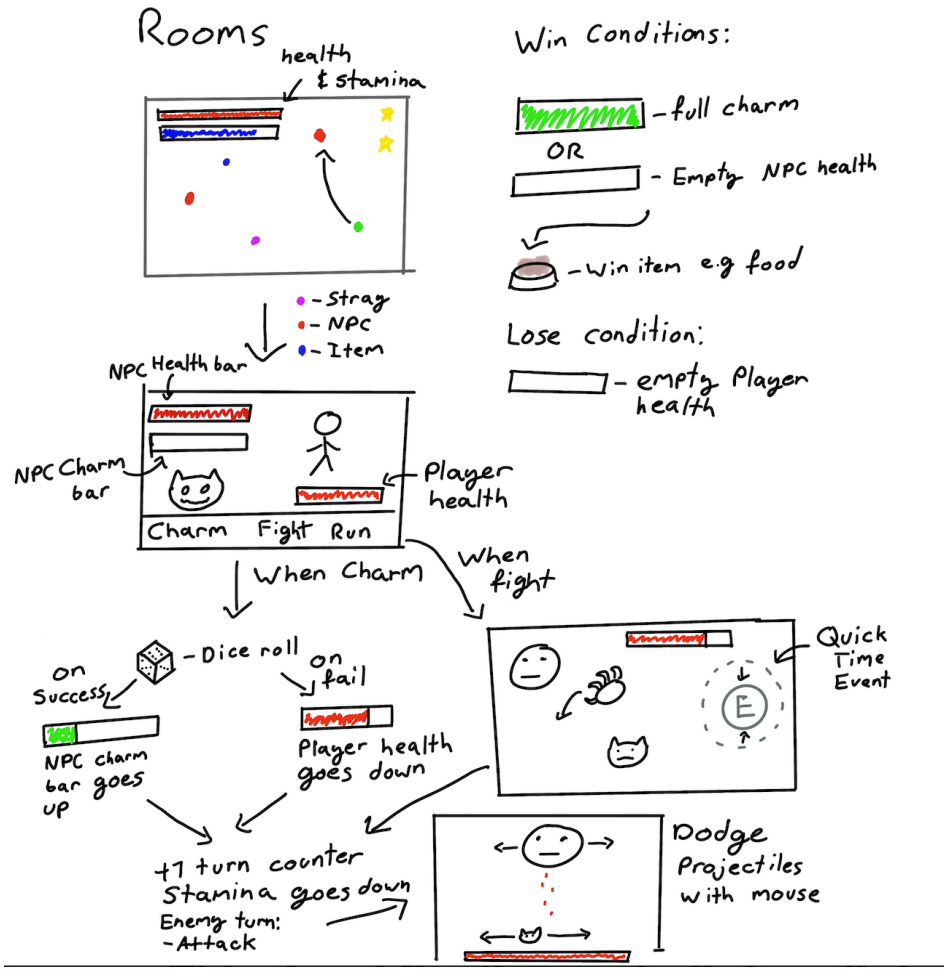
1. Start Screen



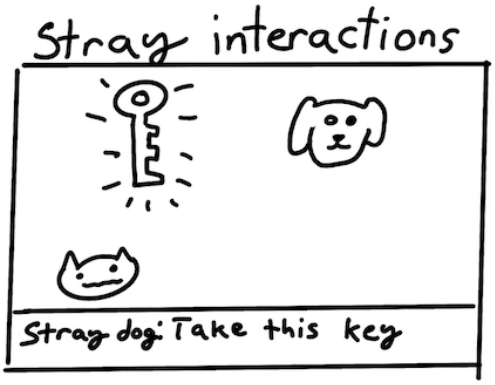
2. Main world



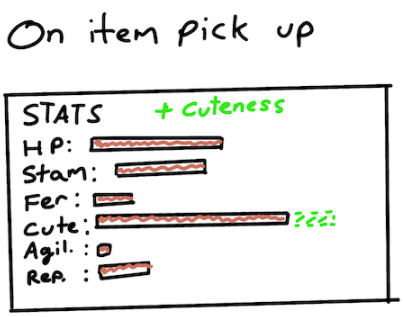
3. Human encounter & turn based combat



4. Stray encounter



5. Item pickup



Technical Elements:

Identify how the game satisfies the core technical requirements: rendering; geometric/sprite/other assets; 2D geometry manipulation (transformation, collisions, etc.); gameplay logic/AI, physics.

Rendering & Animation:

- Rendering player and assets on screen in overworld
- Rendering sprite as well as user actions during encounter phase
- Darken screen and maybe slowly limit vision during chase sequence
- Creating animation using sprite sheets for movement, interaction, taking damage, etc.
    - If time allows, do animated skinned mesh.

2D geometry manipulation:

- Player and Enemy sprite movement, horizontal and vertical translate,
- Player collision against assets
- Use matrix transformations for visual effects (shake, shear, rotation, etc)
- Use linear interpolation to create smooth transformations

AI:

- A* pathfinding algorithm for Enemy Patrol pathfinding
- NPC AI during encounters will be pre-determined in code such as dialogue, special methods of defeating, responses to certain actions, etc.

Gameplay Logic:

- **Player Controls**
    - Event Queue system to decouple input detection and input handling
    - Using WASD key for movement
    - Using E key for interaction with NPC and items
    - Using mouse for bullet hell minigame during encounters
    - Using mouse to select action during encounters
    - Using Enter to select action during encounters
    - Using Arrow Keys to switch between actions
    - Use pathfinding algorithm to implement mouse point and click movement
- **World/Player Interactions:**

- Player will be able to interact with doorways and paths in overworld to enter smaller rooms
- Player will be able to interact with NPCs if they choose to which will start the encounter system
    - Players progressing through encounter will give stats based on the action they chose
    - During an encounter players will take turns against the NPC where the player can choose from a selection of actions
        - Attack (QTE that does damage based on your Strength Stat)
        - Charm (dice roll mechanic that is affected by your Cuteness Stat)
        - Intimidate (dice roll mechanic that is affected by your Ferocity Stat)
        - Run Away (costs x amount of energy depending on Agility Stat and NPC)
    - NPC will have a health bar, a Charm bar, and an Intimidate bar
        - Players will be able to decide how they want to progress through the interactions
- Player will be able to pick up items which can affect their stats or give certain advantages/disadvantages
- Players will need to get out of line of site of the enemy patrol if they are seen and actively being chased(i.e. staying in flashlight/a certain distance for a set amount of frames)
- Players who don't escape for patrol and are caught will lose all their items during their run and be transported back to the start of the game with a rerandomized map and a life lost (Player will keep a portion of stats gained through encounters)
- Once all player lives are lost then the character reverts back to base stats
- Number of enemy patrols will change based on certain factors such as how many turns a character has spent inside a room
- Performing actions will cost the players energy
    - If a players energy runs out, they will pass out on the spot
    - Passing out in an open area will greatly increase the chances of enemy patrols increasing throughout the overworld
    - Players can choose to sleep in certain areas which will refill the energy level and slightly increase the chances of enemy patrols increasing
    - Energy levels can also be replenished through items found in the world
- The world will consist of 3 different tiers of traversable locations:
    - **City:** the biggest level of location. It consists of different streets and will be procedurally generated along with its streets and the room entrances within the streets when the player decides to enter the city. A city only consists of streets and can have specific attributes. There will be locations where the player can choose to travel between cities, and choosing to do so will delete the previous city, making it inaccessible to the player.
    - **Street:** the middle level of location. It is generated when the encapsulating city is generated. It can contain entrances to multiple rooms, items, interactable characters, and enemy patrol. The transition

between streets can be done by traveling to the edge of the screen where an open path is available. The transition when changing streets is smooth without any loading time.
    - **Room:** the smallest level of location. The room is procedurally generated when the player decides to enter it from a street. It does not contain any transition to other locations other than the street the player entered from. It can contain interactable characters and items.
- **Physics:**
    - Particle system for player movement(smoke and dust while moving)
    - Collision system with assets in the world
    - Potien
- **Sound:**
    - Background music based on the current level/room
    - Encounter music
    - Sound effects for dialogue
    - Sound effects for movement in the overworld
    - Sound effects when in the chase sequence such as change in music and heartbeat audio

Advanced Technical Elements:

List the more advanced and additional technical elements you intend to include in the game prioritized on likelihood of inclusion. Describe the impact on the gameplay in the event of skipping each of the features and propose an alternative.

1. **Bullet hell combat mini-game:** replaces the pokemon style turn-based combat system with an under-tale inspired bullet hell mini game, where the player has to dodge projectiles using their mouse.
2. **Quick-time-events:** For player Attack option, implementing a dynamic QTE rather than doing something like an arrow key press sequence with a timer
3. **Line of Sight Chase Sequence:** Implementing a line of sight mechanic during the chase sequence instead of using the simpler pixel distance method
4. **Complex Dice Manipulation:** Implementing more dice mechanics such as dices with multiples of the same number, special dice that have effects when rolling even/odd numbers, etc. rather than the basic dice mechanic where the number of dice/values of dice increase base on stats
5. **Interactable Character Information Display:** A small pop-up dialogue when approaching an interactable character that displays some character characteristics based on the player's intelligence stat.

Devices:

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

- **Keyboard**

    - Movement will be mapped to the WASD keys and the E key will be used to interact with items, NPCs, dialogue, and events.
    - Arrow keys and Enter key can also be used to select the action that the user wishes to take
- **Mouse**
    - Used to select the action that the user wishes to take during an encounter
    - Mouse will be used for the dodging sequence during the enemy turn of an encounter (advanced technical element)
    - Alternative movement option, point and click on the tiles to automatically move there

Tools:

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

GLFW - keyboard and mouse input handling

Box2D - 2d collision engine if we're allowed to use it

SDL2 - Window creation & audio handling

GLM - OpenGL mathematics library

SFML - Simple and Fast Multimedia Library

Team management:

Identify how you will assign and track tasks and describe the internal deadlines and policies you will use to meet the goals of each milestone.

We are using Notion for project management. We will User Stories for each big feature, and create tickets for them using the sub-task feature in Notion. We will be assigning story points to each task to better estimate the work completed by each team member.

We will meet each week in scrum style and do sprint planning every Monday after class. The tickets will be assigned to members based on priority and story points, making sure everyone has a similar workload. In each scrum meeting we will give each other updates on blockers or progress. We also created different channels on Discord for technical assistance and design questions to help each other throughout the week. Any tickets not completed during a sprint will be added to the backlog and revisited in the next sprint.

Details on the internal deadlines can be found in the development plan below. There are tasks planned for each week under certain categories. We will take on tasks based on our building interests during the development process. To meet the goals of each milestone, we turned the requirements into user stories. Each user story has a certain number of tasks and DoDs created based on the rubric to ensure we capture all the elements needed.

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).

[Created Elements for each milestone are highlighted purple in the development plan]

Milestone 1: Skeletal Game

User Stories:

1. As a player, I want to see the game world rendered with basic textures and sprites, so that I can start playing without performance issues.
    - ☐ Implement a basic fragment shader for sprite rendering of textured geometry.
    - ☐ Implement 3-layer parallax scrolling backgrounds for the encounter scenes like traffic in the background.
    - ☐ Implement drawing order logic.
    - ☐ Implement correct blending for motion using SFML.

    DoD: Simple rectangular sprites are loaded and rendered on the screen with textures.

    DoD: Backgrounds move with camera motion (parallax effect).

    DoD: Drawing order is consistent and predictable when objects overlap, or when objects are hidden.

    DoD: Sprite animation render without flickering.
2. As a player, I want to see smooth interpolation of movements, so that transitions feel realistic.
    - ☐ Implement interpolation for smoother position transitions.

    DoD: Entity smoothly moves between two points when commanded.
3. As a player, I want to control the default cat character with my keyboard and mouse, so that I can interact with the game world.
    - ☐ Implement keyboard controls (WASD) for player movement using GLFW.
    - ☐ Implement mouse input for entity interaction (e.g., clicking on items).
    - ☐ Implement translation, rotation, and scaling on entities.

    DoD: The player can move the entity using the WASD keys.

    DoD: The player can interact with objects via clicking the mouse.

    DoD: Entities can be translated, rotated, and scaled via hard-coded logic or keyboard controls, and appear as expected.

4. As a player, I want to see randomized entity behaviors, so that the game feels dynamic.
    - ☐ Implement random movement patterns for non-player entities. (can be hardcoded)
    - ☐ Start working on basic AI pathfinding if time permits.

      DoD: Entities exhibit random movements through either pre-determined logic or procedurally create

5. As a player, I want to have game boundaries, so that I can't go outside the intended play area.
    - ☐ Define game boundaries and implement boundary enforcement.

      DoD: Player's movement is restricted to the boundaries of the level.

6. As a player, I want to have collisions between entities, so that objects interact realistically.
    - ☐ Implement basic collision detection between entities (AABB).
    - ☐ Implement collision detection for random event interactions (e.g., a random patrol gets triggered when the player crosses a boundary).

      DoD: Collisions between entities are detected and logged.

      DoD: Entities respond correctly upon collision, either stopping or changing direction.

7. As a player, I want simple visual effects in the game, so that the game has basic, noticeable graphical feedback when interacting with the environment.
    - ☐ Implement fragment shaders to handle sprite color changes.
    - ☐ Implement basic heads-up display (HUD) UI with stamina and health for now
    - ☐ Implement the stamina depletion system and show it on UI.

      DoD: The UI should reflect the stamina decreasing by turning gradually red as the player does actions.

8. As a developer, I want to create a spreadsheet in Excel to track my bugs, so that I can work systematically to fix them.
    - ☐ Create a bug list in Microsoft Excel.

9. As a developer, I want to create a test plan, so that I know the player or game actions and their expected outcomes.
    - ☐ Test for rendering artifacts, overlapping sprite issues, and collision accuracy.

Development Plan for M1 (due Oct 6)

Week 1 (Sept 23 - 29):

Rendering & Animation:

- [ ] Implement basic fragment shader for sprite rendering of textured geometry.
- [ ] Implement 3-layer parallax scrolling backgrounds for the encounter scenes like traffic in the background.
- [ ] Implement drawing order logic.
- [ ] Implement correct blending for motion using SFML.

Player Movement:

- [ ] Implement keyboard controls (WASD) for player movement using GLFW.
- [ ] Implement translation, rotation, and scaling on entities.

Interpolation & Collision:

- [ ] Implement interpolation for smoother position transitions.
- [ ] Implement basic collision detection between entities (AABB).

Testing:

- [ ] Test rendering performance with parallax layers.
- [ ] Test movement interpolation for any stutters or input lag.

Week 2 (Sept 30 - Oct 6):

Player Interactions:

- [ ] Implement mouse input for entity interaction (e.g., clicking on items).
- [ ] Define game boundaries and restrict player movement.

AI & Randomized Behavior:

- [ ] Implement random movement patterns for non-player entities.
- [ ] Start working on basic AI pathfinding if time permits.

Collision:

- [ ] Implement collision detection for random event interactions (e.g., a random patrol gets triggered when the player crosses a boundary).

Visual Feedback:

- [ ] Implement fragment shaders to handle sprite color changes.
- [ ] Implement basic heads-up display (HUD) UI with stamina and health for now
- [ ] Implement the stamina depletion system and show it on UI.

Testing & Reporting:

- [ ] Test for rendering artifacts, overlapping sprite issues, and collision accuracy.

- [ ] Create a bug list in Microsoft Excel.
- [ ] Report and fix bugs, prepare the submission video showcasing basic rendering, movement, and interactions.

Milestone 2: Minimal Playability

User Stories:

1. As a player, I want to see animations for entities so that the game feels dynamic and lively.
    - [ ] Implement sprite animations using sprite sheets for walking, idle and interaction states.
    - [ ] Implement the animation using SFML for smooth transition.

      DoD: Entities animate using a series of frames from a sprite sheet, creating fluid animations.

      DoD: Entities transition smoothly between animations, there is no flickering.
2. (if not completed in M1) As a developer, I want to implement AI behavior so that entities can act autonomously based on the game state.
    - [ ] Refine random event system based on player interactions (trigger based on player proximity, location, or movement).
    - [ ] Introduce more complex AI responses to random events (e.g., patrols speeding up or leaving due to a time-based event like a shift change).
    - [ ] If not completed in M1, implement A* pathfinding for NPCs.

      DoD: AI can make decisions based on specific conditions (e.g., move towards the player if within range).

      DoD: AI entities can navigate the game world and avoid obstacles when moving toward a target.
3. As a player, I want more advanced collision detection so that interactions between objects feel natural.
    - [ ] Implement mesh-based collision detection for non-rectangular objects.
    - [ ] Add collision resolution (e.g., bouncing off, stopping).

      DoD: Entities with complex shapes can detect collisions with each other and respond accordingly.

      DoD: Complex collision events are handled without noticeable slowdown or visual artifacts.
4. As a developer, I want to see the game performance (FPS) on the screen so that I can track how smoothly it runs.
    - [ ] Implement real-time FPS counter to track game performance.

DoD: The FPS counter updates in real-time and can be toggled on/off by pressing a key (e.g., F).

5. As a player, I want to see a help/tutorial system so that I can understand how to play the game.
    - ☐ Implement a help/tutorial system that provides instructions (e.g., controls).

      DoD: The help menu displays game instructions, accessed via a key press (e.g., H).

6. As a player, I want audio feedback during interactions, so that I get immediate sound cues during gameplay.
    - ☐ Implement background music that adjusts based on the player's location or situation (e.g., heartbeat sound when danger is nearby).
    - ☐ Add audio feedback for key actions (e.g., jumps, scratches, item collection).

      DoD: Background music changes dynamically between safe and dangerous zones.

      DoD: Sound effects trigger for jumping, landing, and item interactions.

7. As a player, I want to be able to save my progress in the game, so that I can go back and continue building on it.
    - ☐ Implement the ability to save and reload the game from the most recent save the user did by serializing and deserializing from a human-readable text file.

      DoD: Serialize all the entities and ensure they can be reloaded as closely as possible to its pre-save state.

Development Plan for M2 (due Oct 27)

Week 3 (Oct 7 - 13):

Entity Animations:

- ☐ Implement sprite animations using sprite sheets for walking, idle and interaction states.
- ☐ Implement the animation using SFML for smooth transition.

Basic AI Pathfinding:

- ☐ If not completed in M1, implement A* pathfinding for NPCs.

Reloadability:

- ☐ Implement the ability to save and reload the game from the most recent save the user did by serializing and deserializing from a human-readable text file.

Testing:

- [ ] Test animation transitions for frame drops or flickers.
- [ ] Check AI navigation for performance bottlenecks or pathfinding issues.

Week 4 (Oct 14 - 20):

Advanced Collision Detection:

- [ ] Implement mesh-based collision detection for non-rectangular objects. (variable hitboxes, different shapes for obstacles).
- [ ] Add collision resolution (e.g., bouncing off, stopping).

Sound Implementation:

- [ ] Implement background music that adjusts based on the player's location or situation (e.g., heartbeat sound when danger is nearby).
- [ ] Add audio feedback for key actions (e.g., jumps, scratches, item collection).

UX:

- [ ] Refine random event system based on player interactions (trigger based on player proximity, location, or movement).

Testing:

- [ ] Test collision detection for consistency and smooth response.
- [ ] Test audio feedback to ensure timely sound triggers without lag.

Week 5 (Oct 21 - 27):

Help System:

- [ ] Implement a help/tutorial system that provides instructions (e.g., controls).

FPS Counter:

- [ ] Implement real-time FPS counter to track game performance.

AI:

- [ ] Introduce more complex AI responses to random events (e.g., patrols speeding up or leaving due to a time-based event like a shift change).

Testing & Reporting:

- [ ] Full playtest of all M2 features (animations, AI, collisions, and audio).
- [ ] Report bugs and fix identified issues.

- [ ] Prepare submission video for M2, highlighting new animations, AI, and collisions.

Milestone 3: Playability

User Stories:

1. As a developer, I want to improve memory management so that the game runs efficiently.
    - [ ] Implement efficient memory allocation and deallocation for game assets.

        DoD: The game uses memory efficiently, with no memory leaks or crashes after extended play sessions.

        DoD: Memory usage remains stable under stress testing with multiple entities and complex behaviors.

2. As a developer, I want to test for unexpected user input so that the game remains stable.
    - [ ] Implement error handling for invalid or unexpected user input.
    - [ ] Implement proper window resizing and loss of focus handling.

        DoD: The game does not crash or behave unexpectedly when invalid inputs are received (e.g., out-of-bounds clicks).

        DoD: The game adjusts correctly when resized or minimized and maintains state when the player returns.

3. As a player, I want to see fluid animations of my cat character, so that my movements feel more realistic.
    - [ ] Implement skinned mesh animations for the cat.
    - [ ] Bind the cat model to a skeleton and animate based on walking, jumping, and climbing actions.

        DoD: Cat animations smoothly transition between walking, jumping, and climbing.

        DoD: Animations do not break or freeze during gameplay.

4. As a player, I want to see all of my stats in my play screen, so that I can make calculated decisions for my next move.
    - [ ] Implement the full HUD UI elements to show all the player stats (stamina, health, ferocity, etc.) and items(key, collar, etc.)
5. As a player, I want enemies to coordinate their actions, so that combat and avoidance strategies feel more challenging.
    - [ ] Implement AI group behavior for patrolling police or stray animals.
    - [ ] Add cooperative planning where enemies can surround the player.

- [ ] AI should interact with player animations (e.g., police reacting to player actions).

  DoD: Police patrols work together to block escape routes.

  DoD: Stray animals gang up on the player in coordinated attacks.

6. As a player, I want to see better visual effects like smoke and dust while my character moves, so that the game feels closer to real life.
   - [ ] Implement a particle system to add particle effects such as smoke and dust to the character's move.

     DoD: Use instanced rendering to render many instances of the same object more efficiently.

Development Plan for M3 (due Nov 10)

Week 6 (Oct 28 - Nov 3):

Memory Management:

- [ ] Implement efficient memory allocation and deallocation for game assets.

Input Error Handling:

- [ ] Implement error handling for invalid or unexpected user input.
- [ ] Implement proper window resizing and loss of focus handling.

UI:

- [ ] Implement the full HUD UI elements to show all the player stats (stamina, health, ferocity, etc.) and items(key, collar, etc.)
- [ ] Implement a particle system to add particle effects such as smoke and dust to the character's move.

Testing:

- [ ] Stress test the game to check memory usage and error-handling stability.
- [ ] Test user input errors like resizing, loss of focus, or minimizing the window.

Week 7 (Nov 4 - Nov 10):

Advanced AI Group Behavior:

- [ ] Implement AI group behavior for patrolling police or stray animals.
- [ ] Add cooperative planning where enemies can surround the player.,
- [ ] AI should interact with player animations (e.g., police reacting to player actions).

Cat Animation (Skinned Mesh):

- ☐ Implement skinned mesh animations for the cat.
- ☐ Bind the cat model to a skeleton and animate based on walking, jumping, and climbing actions.

UX:

- ☐ Implement UI elements to show player stats (stamina, health, ferocity, etc.).

Testing & Reporting:

- ☐ Final test of all M3 features, with focus on performance and group AI.
- ☐ Report bugs, fix remaining issues, and record the submission video for M3.

Milestone 4: Final Game

User Stories:

1. As a player, I want to interact with a branching narrative, so that my decisions impact the game's story.
    - ☐ Implement story-driven mechanics where decisions affect the storyline (e.g., if you charm a human enough, they adopt you).
    - ☐ Begin work on more complex gameplay mechanics, such as forming alliances with strays and using equippable items.
    - ☐ Implement dialogue trees with different paths based on player input.

      DoD: Dialogue trees reflect the player's choices and change the outcome of interactions.

      DoD: Different endings are possible based on player decisions.

2. As a player, I want the game world to feel dynamic, with realistic lighting and shadows, so that the environments feel immersive.
    - ☐ Implement 2.5D lighting with dynamic shadows that react to player movement.
    - ☐ Add day/night cycles that impact gameplay (e.g., night makes it easier to avoid police patrols).

      DoD: Lights cast realistic shadows based on the player's position.

      DoD: Day/night cycles visibly affect lighting and gameplay.

3. As a player, I want to face enemies that use advanced decision-making, so that combat feels intelligent and strategic.

- [ ] Implement advanced AI decision-making using a goal-based system (e.g., chase or flee based on health).

  DoD: Enemies can assess whether to chase, block, or flee from the player.

  DoD: Pathfinding leads enemies efficiently toward the player while avoiding obstacles.

Plan B: If advanced story progression or alliances take too long, focus on final polish and ensuring the game is stable and bug-free for release.

Development Plan for M4 (due Dec 1)

Week 8 (Nov 11 - Nov 17):

Branching Narrative & Story-driven Elements:

- [ ] Implement story-driven mechanics where decisions affect the storyline (e.g., if you charm a human enough, they adopt you).
- [ ] Begin work on more complex gameplay mechanics, such as forming alliances with strays and using equippable items.

Dialogue System:

- [ ] Implement dialogue trees with different paths based on player input.

Testing:

- [ ] Test multiple storylines and dialogue options for bugs or incorrect outcomes.

Week 9 (Nov 18 - Nov 24):

Dynamic Lighting & Day/Night Cycles:

- [ ] Implement 2.5D lighting with dynamic shadows that react to player movement.
- [ ] Add day/night cycles that impact gameplay (e.g., night makes it easier to avoid police patrols).

Advanced Enemy Decision Making:

- [ ] Implement advanced AI decision-making using a goal-based system (e.g., chase or flee based on health).

Testing:

- [ ] Test lighting and shadow effects under various gameplay conditions.

- [ ] Test AI decision-making under combat and chase sequences.

Week 10 (Nov 25 - Dec 1):

Final Testing & Polish:

- [ ] Full test of all game mechanics, from story elements to combat and AI behavior.
- [ ] Fix any final bugs and optimize performance for smooth gameplay.

Submission Video:

- [ ] Record final gameplay video showcasing all core features (story, AI, lighting).

Final Submission:

- [ ] Ensure documentation is up to date.
- [ ] Submit final game build and polished submission video.