KOÇ ÜNİVERSİTESİ
COMPUTER ENGINEERING

COMP132: Advanced Programming

Programming Project Report

Food Chain Through Time: Simulation Design and Development

Ahmet Salih Çiçek,89195

Fall 2025

1. General Demo Information

In this section, the usage scenarios and main functionalities of the Food Chain simulation are demonstrated. The application allows users to simulate a biological food chain across different time eras with customizable parameters.

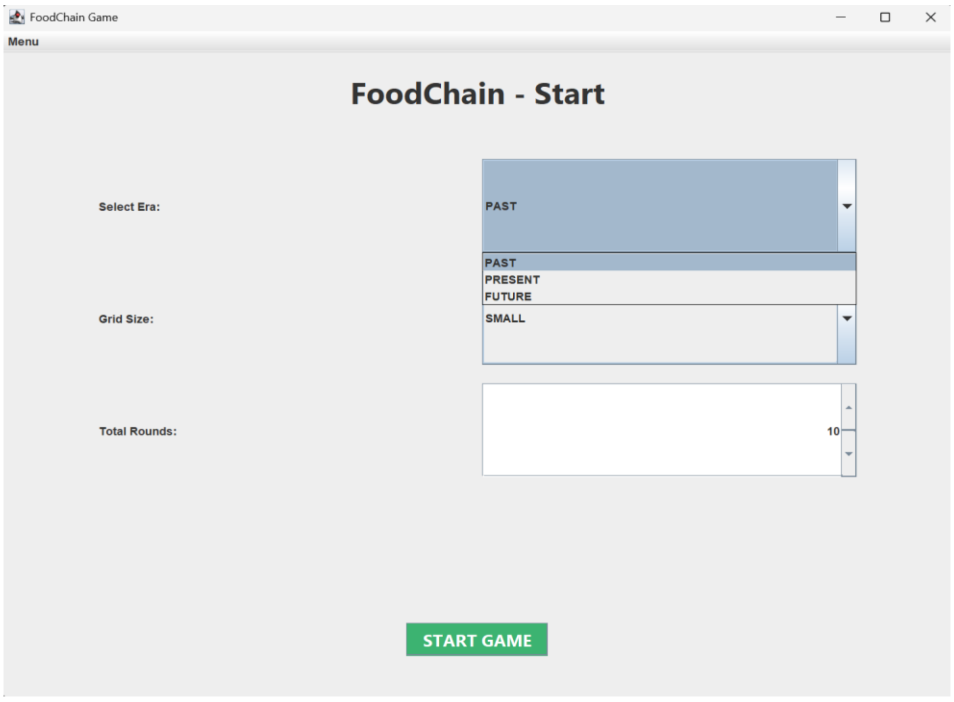
1.1. Starting a New Game & Configuration

The application launches with a configuration screen. Before starting the simulation, the user can customize the game environment:

-Era Selection: The user can choose between Past, Present, and Future eras.

-Grid Size: The board dimensions can be selected as Small (10x10), Medium (15x15), or Large (20x20).

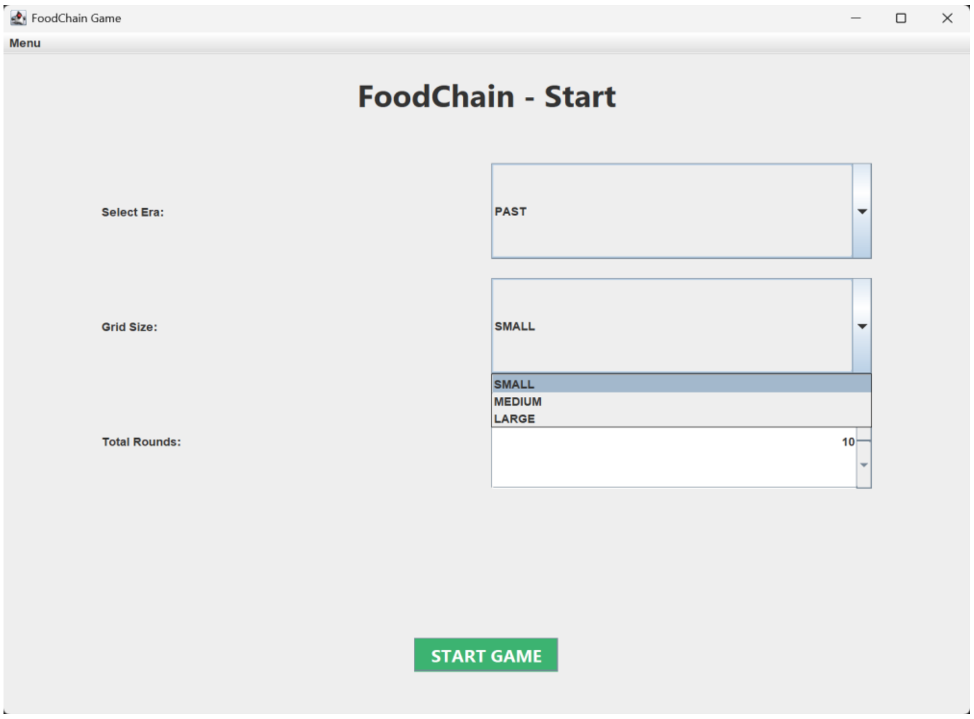
- Rounds: The total duration of the game is set via the round counter.



Select Era: PAST

Grid Size: SMALL

Total Rounds: 10



Select Era: PAST

Grid Size: SMALL

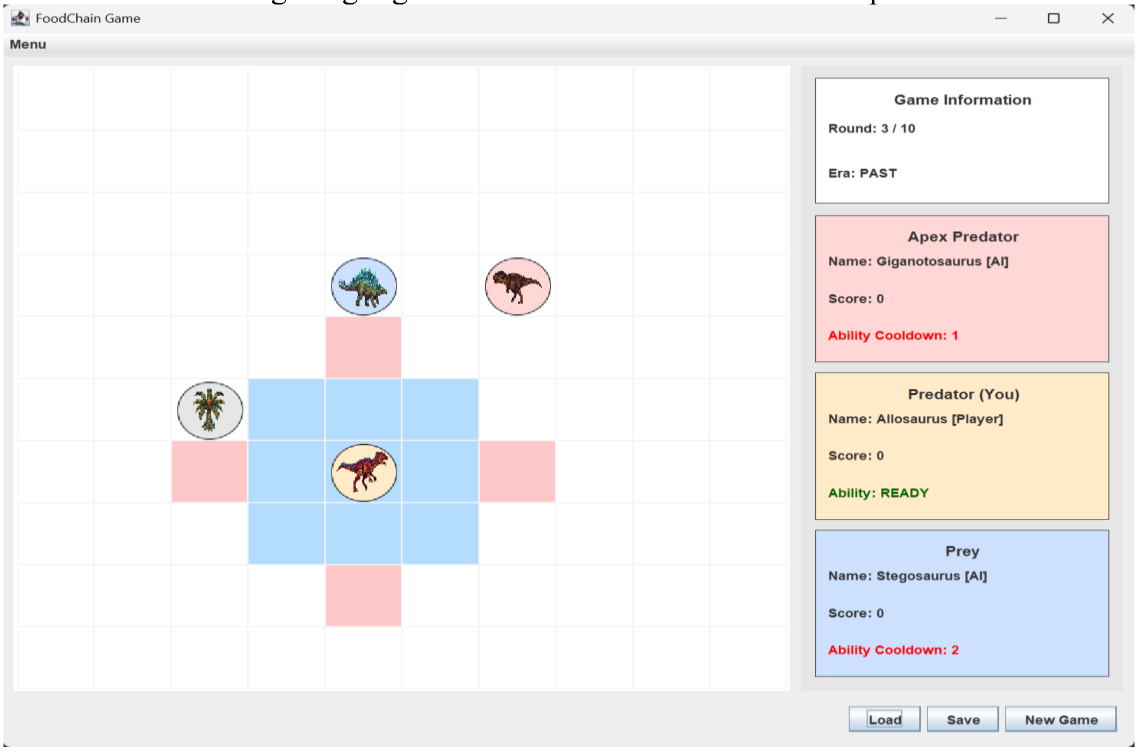
Total Rounds: 10

Figure 1&2: The Start Panel allowing Era, Grid Size, and Round configuration.

1.2. Gameplay and Movement

The game operates on a turn-based system. The user controls the Predator role.

-Movement: When it is the player's turn, valid moves are highlighted on the grid. Blue cells indicate standard walking moves.
- Mouse Interaction: Clicking a highlighted cell moves the Predator to that position.



Game Information
Round: 3 / 10
Era: PAST

Apex Predator
Name: Giganotosaurus [AI]
Score: 0
Ability Cooldown: 1

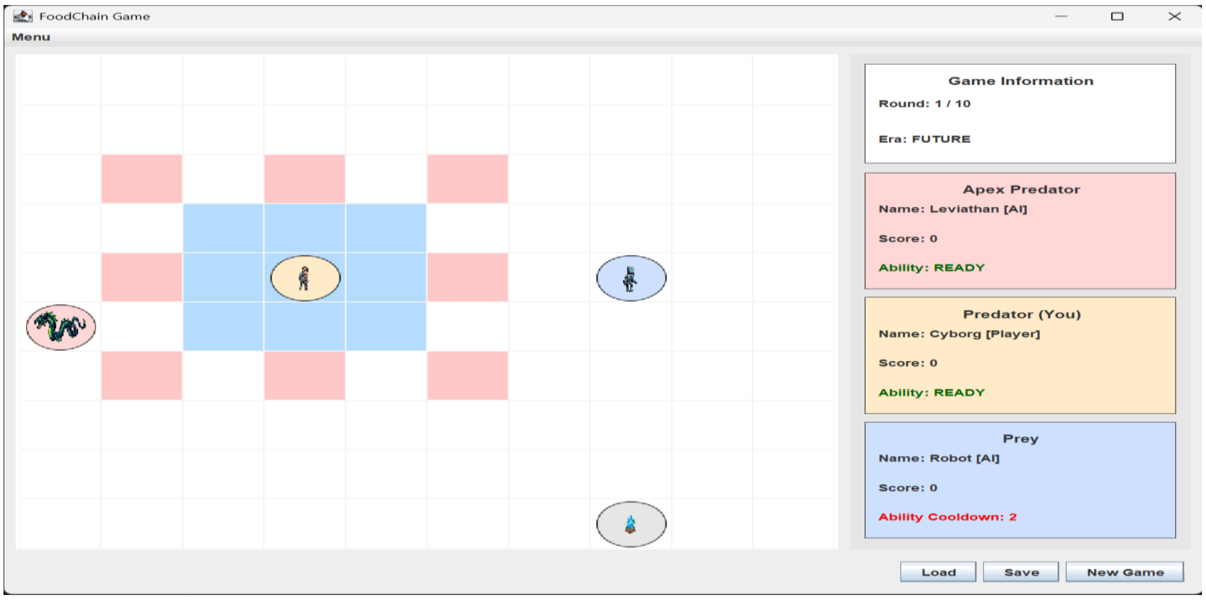
Predator (You)
Name: Allosaurus [Player]
Score: 0
Ability: READY

Prey
Name: Stegosaurus [AI]
Score: 0
Ability Cooldown: 2

Figure 3: Valid move highlights for the Predator player during gameplay.

1.3. Special Moves and Mechanics

Depending on the selected Era and the relative positions of other animals (Apex, Prey), special moves may become available. For example, in the Present Era, the Predator can perform a "Dash" move if specific alignment conditions with the Apex are met. These special moves are highlighted distinctively (e.g., in Red) to alert the user.



Game Information
Round: 1 / 10
Era: FUTURE

Apex Predator
Name: Leviathan [AI]
Score: 0
Ability: READY

Predator (You)
Name: Cyborg [Player]
Score: 0
Ability: READY

Prey
Name: Robot [AI]
Score: 0
Ability Cooldown: 2

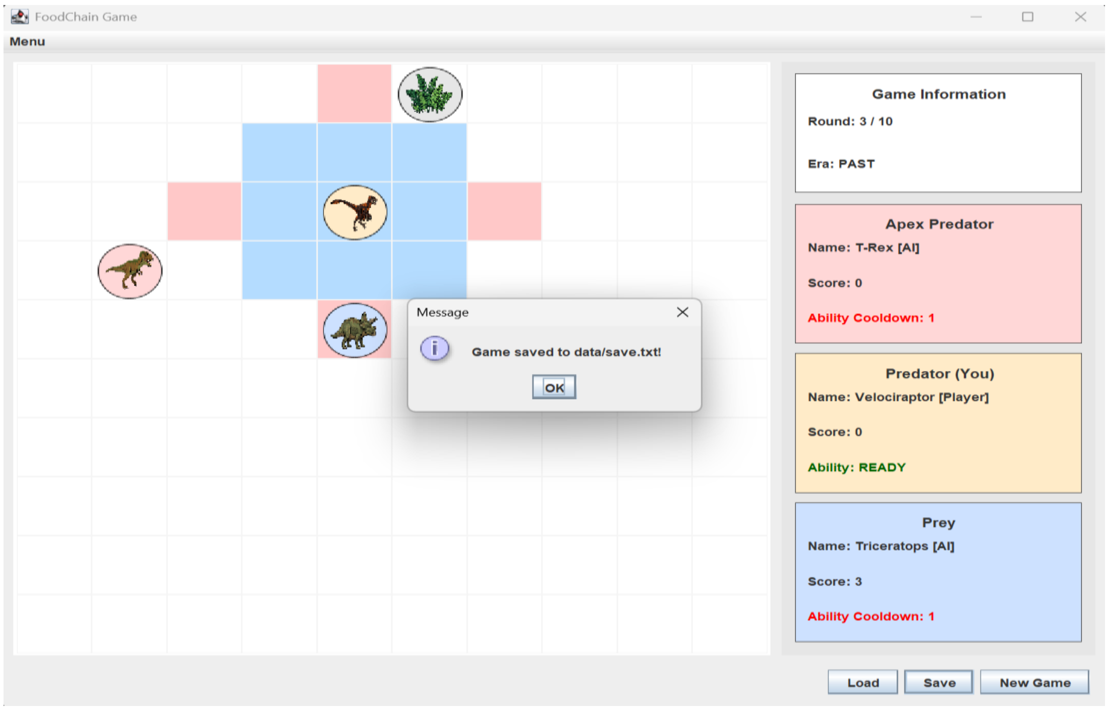
Figure 4: In-game scoring events and the corresponding log file records.

1.4. Save and Load Functionality

The application supports data persistence. The user can save the current state of the game and reload it later.

-Saving: The game state is serialized into a readable text format in data/save.txt.

- Loading: Loading restores the exact positions, scores, and turn order from the file.



Message

Game saved to data/save.txt!

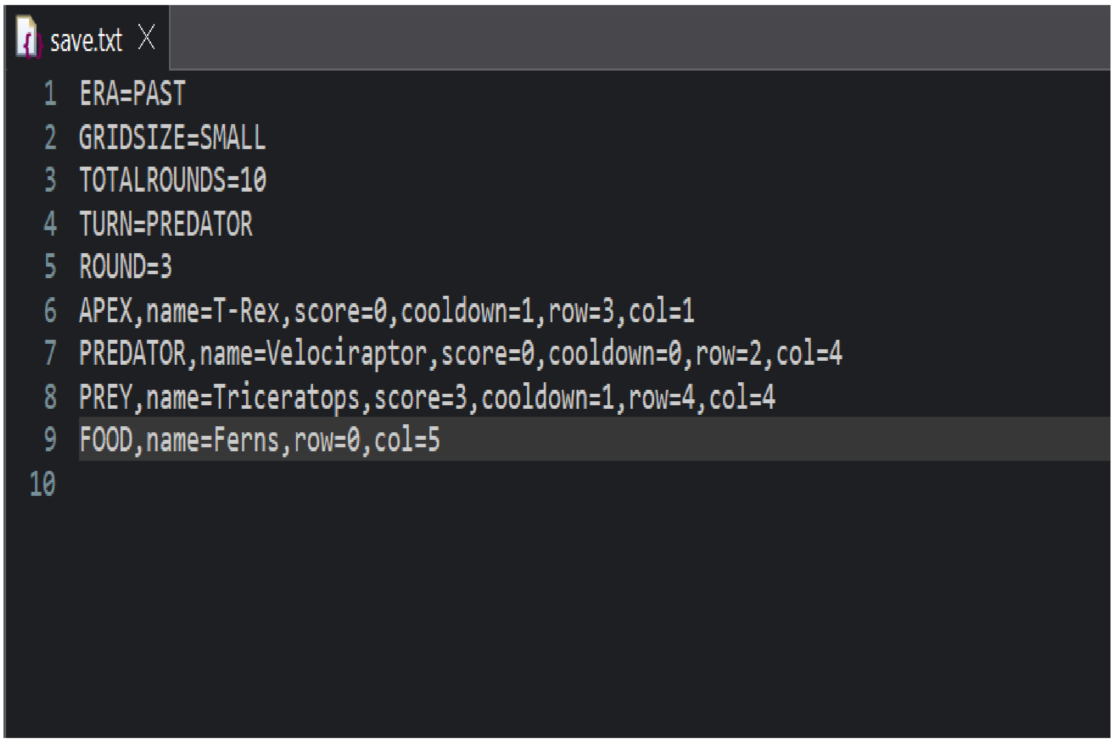
Game Information
Round: 3 / 10
Era: PAST

Apex Predator
Name: T-Rex [AI]
Score: 0
Ability Cooldown: 1

Predator (You)
Name: Velociraptor [Player]
Score: 0
Ability: READY

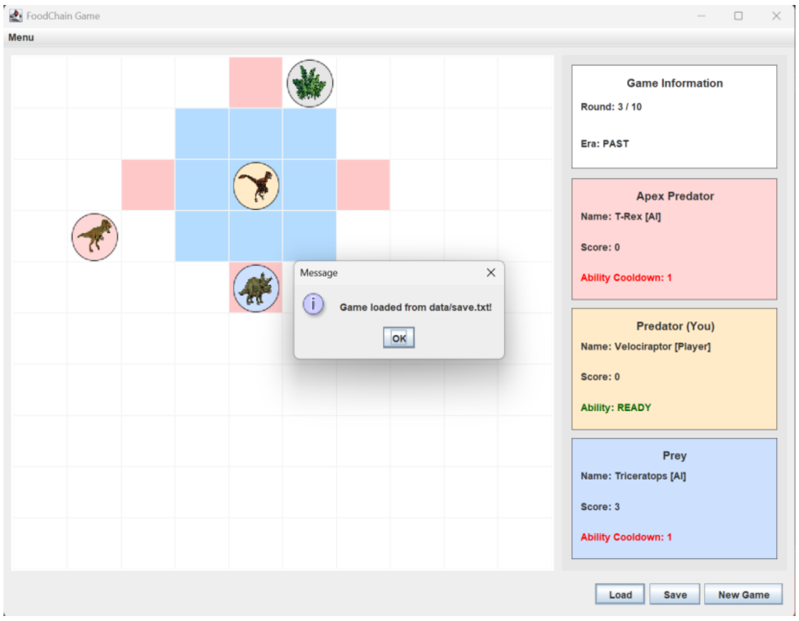
Prey
Name: Triceratops [AI]
Score: 3
Ability Cooldown: 1

Figure 5: Successfully saved game simulation.



```
ERA=PAST
GRIDSIZE=SMALL
TOTALROUNDS=10
TURN=PREDATOR
ROUND=3
APEX,name=T-Rex,score=0,cooldown=1,row=3,col=1
PREDATOR,name=Velociraptor,score=0,cooldown=0,row=2,col=4
PREY,name=Triceratops,score=3,cooldown=1,row=4,col=4
FOOD,name=Ferns,row=0,col=5
```

Figure 6: The serialized game state saved in the text file.



Game loaded from data/save.txt!

Game Information
Round: 3 / 10
Era: PAST

Apex Predator
Name: T-Rex [AI]
Score: 0
Ability Cooldown: 1

Predator (You)
Name: Velociraptor [Player]
Score: 0
Ability: READY

Prey
Name: Triceratops [AI]
Score: 3
Ability Cooldown: 1

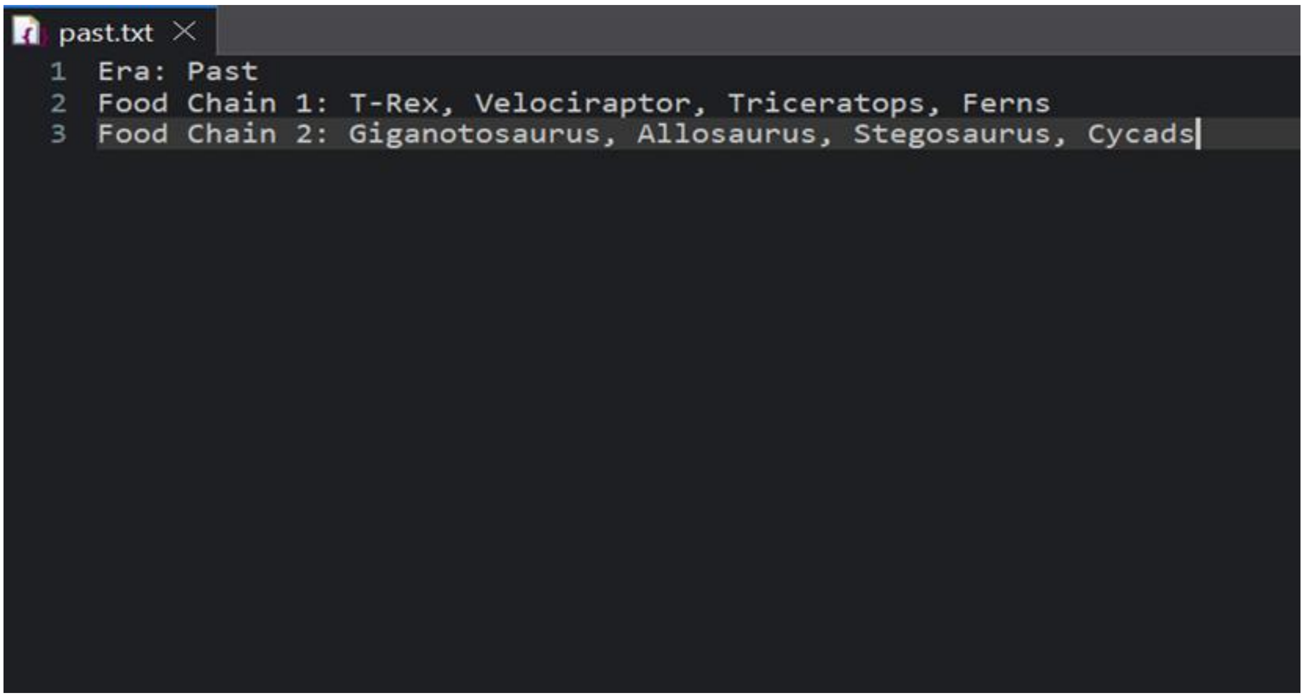
Figure 7: The game simulation continuing from the loaded state.

1.5. Asset & Logic Variations

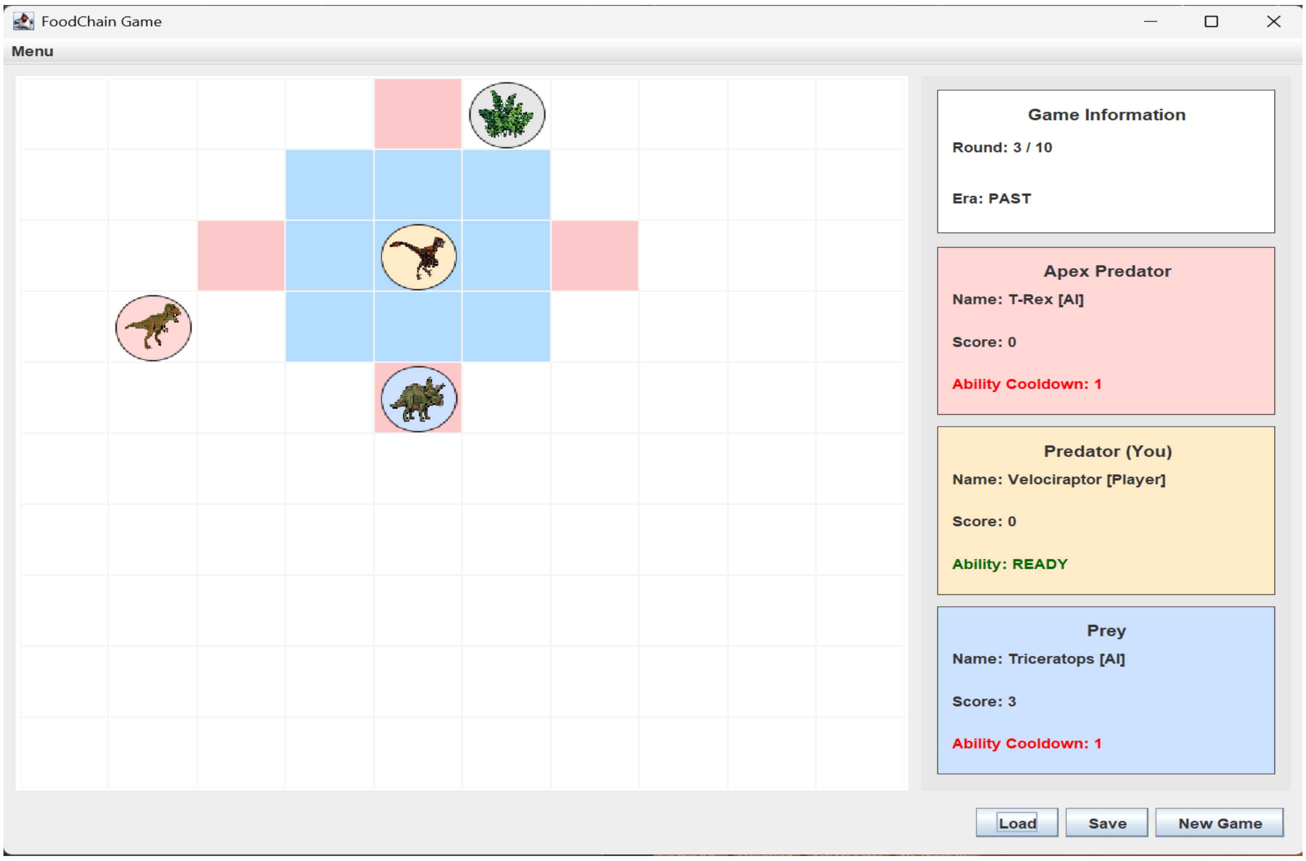
The game content is strictly data-driven to satisfy project requirements. The specific animals and their corresponding icons are loaded directly from foodchain.txt files.

-Text File Dependency: As requested, modifying the foodchain.txt (e.g., past.txt) file external to the source code immediately updates the in-game entity names and icons upon the next launch.

-Icon Differences: Different Eras display distinct visual assets (e.g., Dinosaurs for Past vs. Robots for Future) based on these file definitions.



```
Era: Past
Food Chain 1: T-Rex, Velociraptor, Triceratops, Ferns
Food Chain 2: Giganotosaurus, Allosaurus, Stegosaurus, Cycads
```



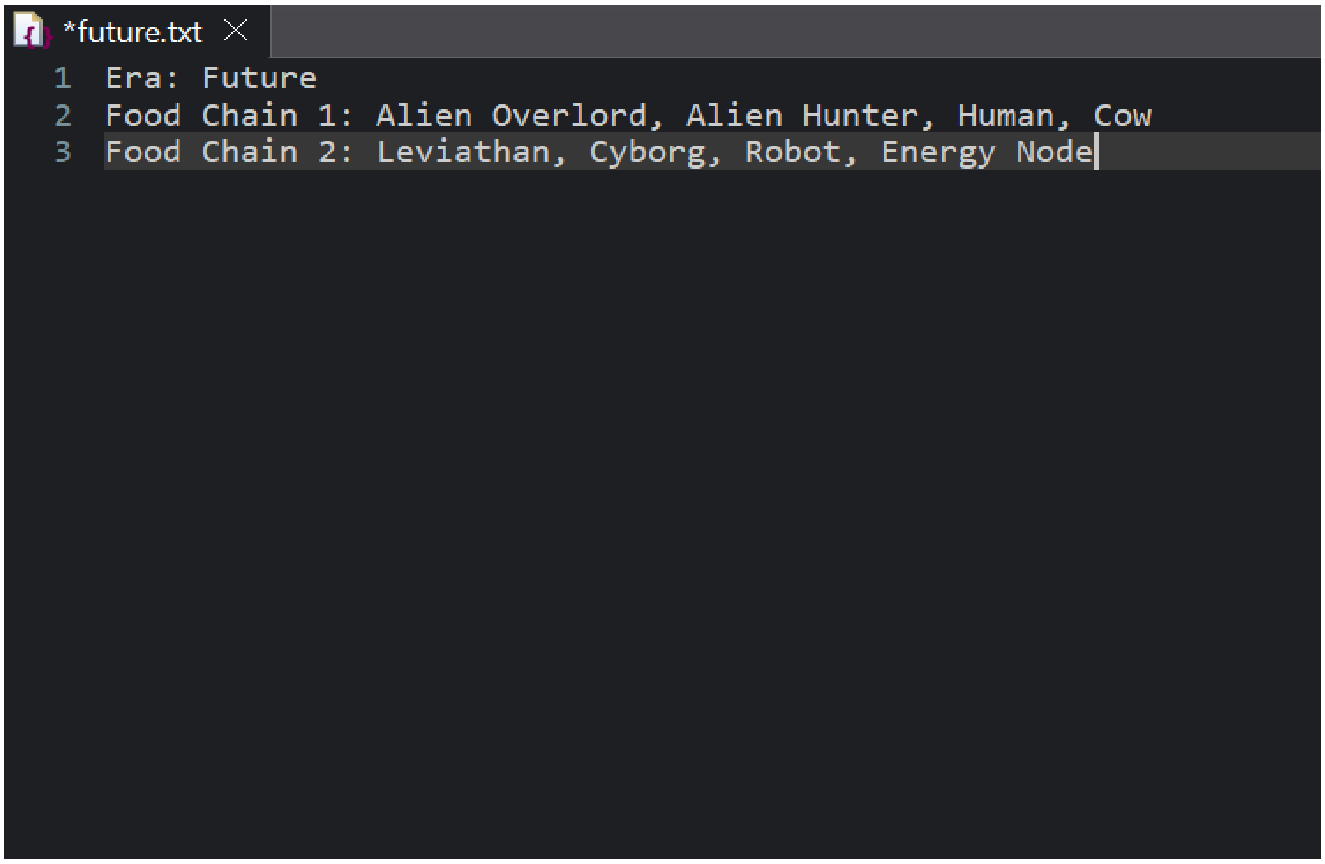
Game Information
Round: 3 / 10
Era: PAST

Apex Predator
Name: T-Rex [AI]
Score: 0
Ability Cooldown: 1

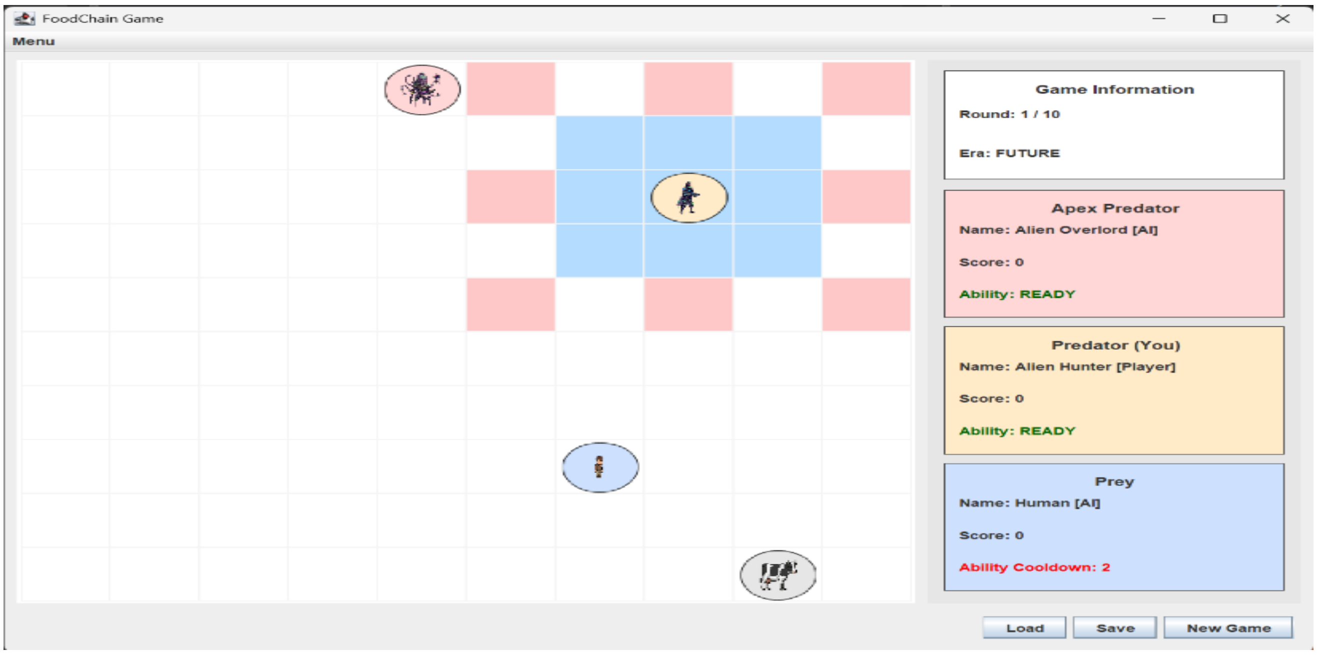
Predator (You)
Name: Velociraptor [Player]
Score: 0
Ability: READY

Prey
Name: Triceratops [AI]
Score: 3
Ability Cooldown: 1

Figure 8&9: Demonstration of how the game changes based on foodchain.txt (Past)



```
Era: Future
Food Chain 1: Alien Overlord, Alien Hunter, Human, Cow
Food Chain 2: Leviathan, Cyborg, Robot, Energy Node
```



Game Information
Round: 1 / 10
Era: FUTURE

Apex Predator
Name: Alien Overlord [AI]
Score: 0
Ability: READY

Predator (You)
Name: Alien Hunter [Player]
Score: 0
Ability: READY

Prey
Name: Human [AI]
Score: 0
Ability Cooldown: 2

Figure 10&11: Demonstration of how the game changes based on foodchain.txt (Future)

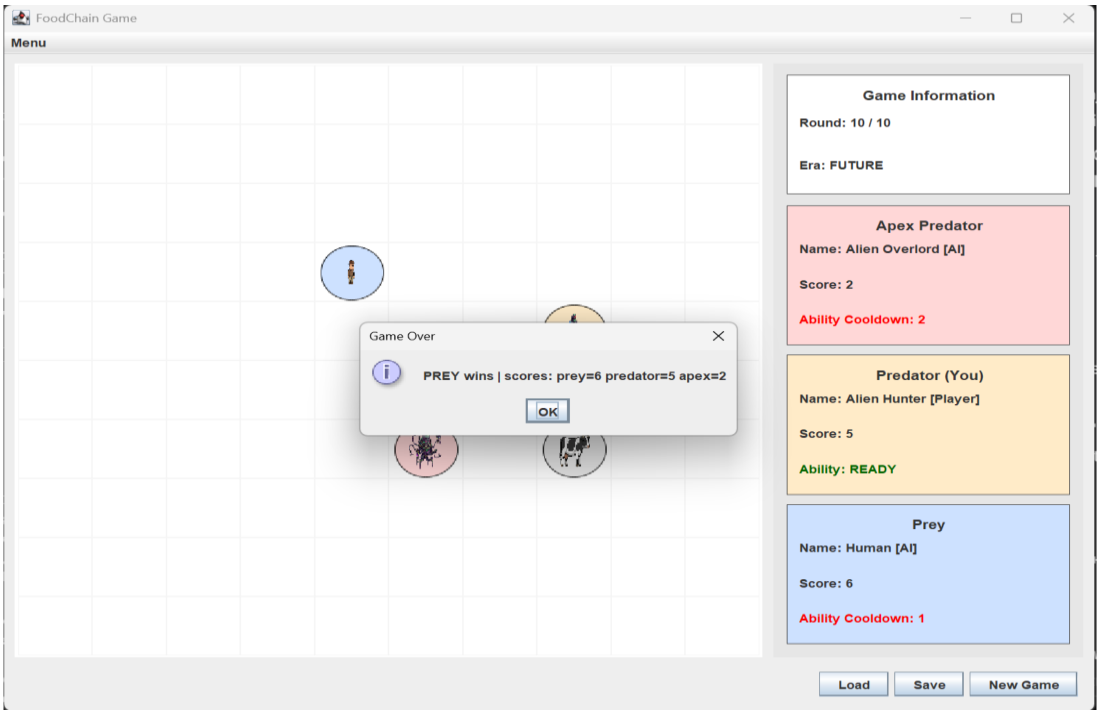
1.6. Scoring, Respawn & Game Completion

The simulation implements dynamic game mechanics tracked via the log system:

-Point Changes: When the Predator consumes a Prey, the score increases by +3 points.

-Respawn Behavior: Upon being consumed, the Prey is removed and immediately respawns at a random empty location on the grid.

-Game Completion: When the total rounds are completed, the game ends, and a message box displays the winner.



PREY wins | scores: prey=6 predator=5 apex=2

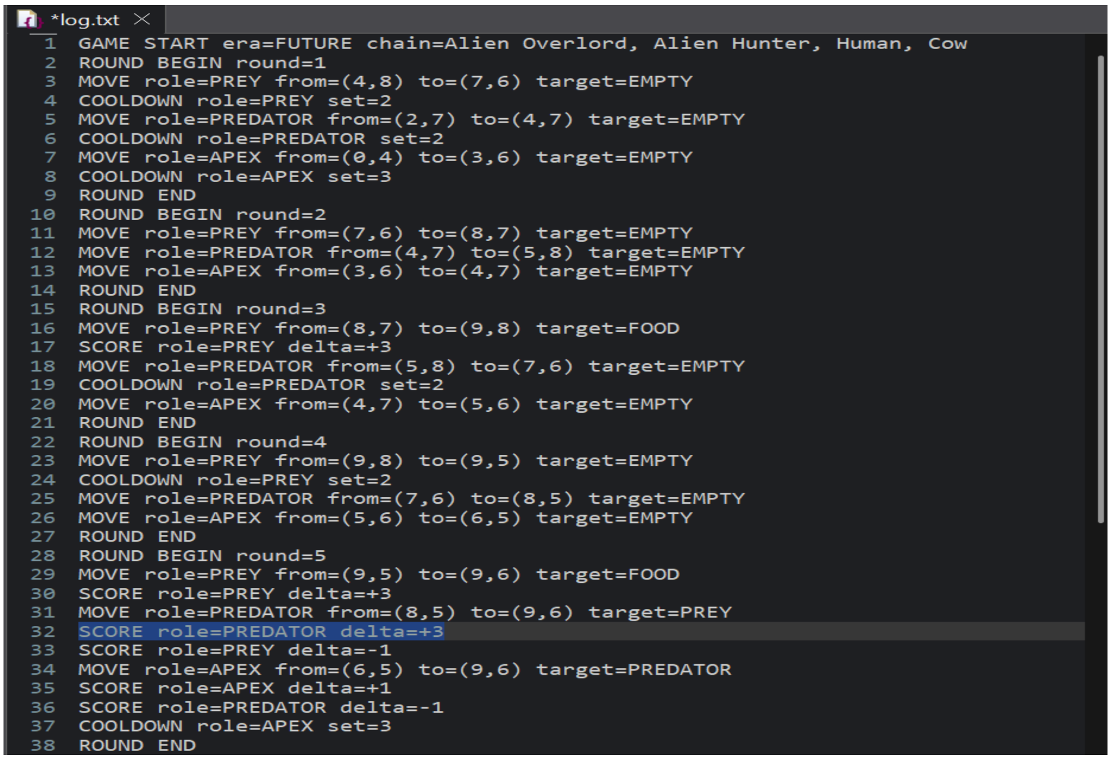
Game Information
Round: 10 / 10
Era: FUTURE

Apex Predator
Name: Alien Overlord [AI]
Score: 2
Ability Cooldown: 2

Predator (You)
Name: Alien Hunter [Player]
Score: 5
Ability: READY

Prey
Name: Human [AI]
Score: 6
Ability Cooldown: 1

Figure 12: The game completion dialog displaying the winner, with the final score visible on the status panel.



```
GAME START era=FUTURE chain=Alien Overlord, Alien Hunter, Human, Cow
ROUND BEGIN round=1
MOVE role=PREY from=(4,8) to=(7,6) target=EMPTY
COOLDOWN role=PREY set=2
MOVE role=PREDATOR from=(2,7) to=(4,7) target=EMPTY
COOLDOWN role=PREDATOR set=2
MOVE role=APEX from=(0,4) to=(3,6) target=EMPTY
COOLDOWN role=APEX set=3
ROUND END
ROUND BEGIN round=2
MOVE role=PREY from=(7,6) to=(8,7) target=EMPTY
MOVE role=PREDATOR from=(4,7) to=(5,8) target=EMPTY
MOVE role=APEX from=(3,6) to=(4,7) target=EMPTY
ROUND END
ROUND BEGIN round=3
MOVE role=PREY from=(8,7) to=(9,8) target=FOOD
SCORE role=PREY delta=+3
MOVE role=PREDATOR from=(5,8) to=(7,6) target=EMPTY
COOLDOWN role=PREDATOR set=2
MOVE role=APEX from=(4,7) to=(5,6) target=EMPTY
ROUND END
ROUND BEGIN round=4
MOVE role=PREY from=(9,8) to=(9,5) target=EMPTY
COOLDOWN role=PREY set=2
MOVE role=PREDATOR from=(7,6) to=(8,5) target=EMPTY
MOVE role=APEX from=(5,6) to=(6,5) target=EMPTY
ROUND END
ROUND BEGIN round=5
MOVE role=PREY from=(9,5) to=(9,6) target=FOOD
SCORE role=PREY delta=+3
MOVE role=PREDATOR from=(8,5) to=(9,6) target=PREY
SCORE role=PREDATOR delta=+3
SCORE role=PREY delta=-1
MOVE role=APEX from=(6,5) to=(9,6) target=PREDATOR
SCORE role=APEX delta=+1
SCORE role=PREDATOR delta=-1
COOLDOWN role=APEX set=3
ROUND END
```

Figure 13: Log file records capturing the dynamic +3 point increase upon eating prey and subsequent entity respawn events.

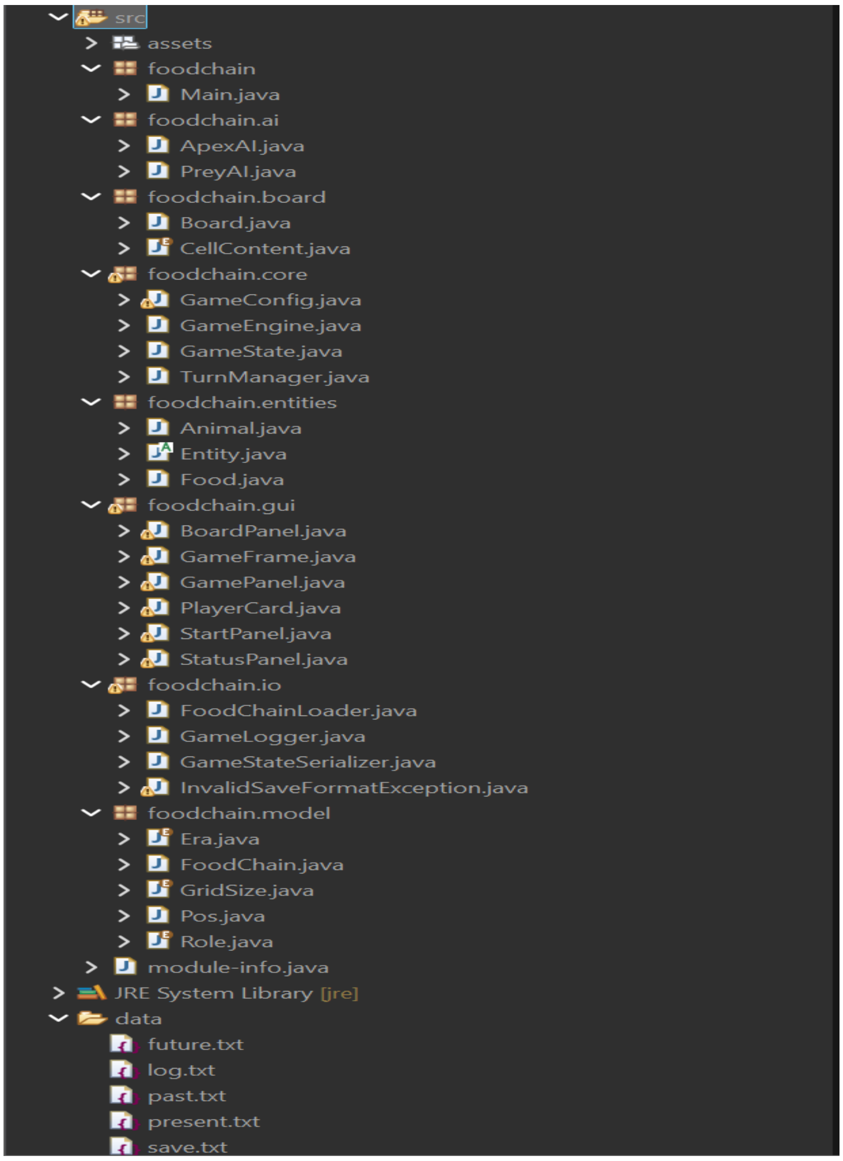
2. Project Design Description

In this section, the technical design and architectural decisions of the application are presented. The project follows Object-Oriented Programming (OOP) principles, utilizing inheritance, polymorphism, and encapsulation to create a modular and maintainable codebase based on MVC principles [1, 4].

2.1. Class Relations

The game entities are modeled using a hierarchical structure to share common behavior and reduce code duplication.

- Entity (Abstract Class): This is the base class for all objects on the board. It defines fundamental properties such as the entity's name and position (Pos). Being abstract, it prevents the instantiation of generic entities.
- Animal (extends Entity): Represents dynamic agents in the game. It introduces gameplay-specific attributes including Role (Enum: APEX, PREDATOR, PREY), score, and abilityCooldown.
- Food (extends Entity): Represents static targets for the Prey. It inherits position and name properties but does not contain complex behavior.
- Role (Enum): Defines the distinct types of animals to strictly manage turn order and interaction rules.



```
src
  assets
  foodchain
    Main.java
  foodchain.ai
    ApexAI.java
    PreyAI.java
  foodchain.board
    Board.java
    CellContent.java
  foodchain.core
    GameConfig.java
    GameEngine.java
    GameState.java
    TurnManager.java
  foodchain.entities
    Animal.java
    Entity.java
    Food.java
  foodchain.gui
    BoardPanel.java
    GameFrame.java
    GamePanel.java
    PlayerCard.java
    StartPanel.java
    StatusPanel.java
  foodchain.io
    FoodChainLoader.java
    GameLogger.java
    GameStateSerializer.java
    InvalidSaveFormatException.java
  foodchain.model
    Era.java
    FoodChain.java
    GridSize.java
    Pos.java
    Role.java
  module-info.java
JRE System Library [jre]
data
  future.txt
  log.txt
  past.txt
  present.txt
  save.txt
```

Figure 14: Class hierarchy and project structure.

2.2. GUI Components

The user interface is constructed using the Java Swing library, separating the visualization from the game logic [1].

- GameFrame: The main container that manages application states (Start Menu vs. Game Screen) using a CardLayout.

-BoardPanel: Responsible for rendering the 10x10, 15x15, or 20x20 grid. It uses a 2D array of JButton components to capture user input via ActionListeners [6]. The panel dynamically updates cell icons based on the current state of the GameEngine [2].

-StatusPanel: Located on the side of the game window, this panel displays real-time statistics, including the current round, player scores, and ability cooldown status.

2.3. .txt File Processing Details

The application relies on external text files to configure the simulation environment, ensuring flexibility using Java NIO **[5]**.

-Configuration Loading: The FoodChainLoader class reads era-specific files (e.g., past.txt). It parses each line to extract the names of the Apex, Predator, Prey, and Food entities.

-Save/Load System: The GameStateSerializer class handles data persistence. It saves the game by writing the current state (Era, Turn, GridSize, Entity Positions) into a structured text format (key=value). When loading, it parses this file to reconstruct the GameState object exactly as it was [5].

2.4. Implementation Details

2.4.1. Turn Management: A TurnManager class acts as a state machine, cycling through the order of PREY -> PREDATOR -> APEX. It ensures that the human player (Predator) moves only when it is their turn.

2.4.2. AI Logic: The application implements simple heuristic AI for non-player characters.

-PreyAI: Calculates a safety score for adjacent cells, prioritizing moves that increase distance from predators while seeking food.

-ApexAI: Uses a greedy algorithm to minimize the Chebyshev distance to the nearest Prey or Predator [3].

2.5. Exception Handling

Robust error handling is implemented to prevent application crashes

2.5.1. IOException: Managed during file operations (reading assets or save files).

2.5.2.InvalidSaveFormatException: A custom exception created to handle corrupted or malformed save files. If a save file is invalid, the user is notified via a generic dialog box instead of a crash.

2.5.3.IllegalArgumentException: Used to validate game inputs, such as ensuring grid sizes and round numbers are positive integers.

3. References

1. Oracle. "A Swing Architecture Overview (MVC)". The Java™ Tutorials. Available: https://www.oracle.com/java/technologies/a-swing-architecture.html

2. Oracle. "Lesson: Performing Custom Painting". The Java™ Tutorials. Available: https://docs.oracle.com/javase/tutorial/uiswing/painting/index.html

3. GeeksforGeeks."ChebyshevDistance".Available: https://www.geeksforgeeks.org/machine-learning/chebyshev-distance/

4. GeeksforGeeks."MVCDesignPattern".Available: https://www.geeksforgeeks.org/mvc-design-pattern/

5. TutorialsPoint."JavaNIO-Files".Available: https://www.tutorialspoint.com/java_nio/java_nio_file.htm

6. Oracle. "How to Write an Action Listener". The Java™ Tutorials. Available: https://docs.oracle.com/javase/tutorial/uiswing/events/actionlistener.html