LBYCPEI Group Project Final Proposal

Team CYBORG

“Atmos-Fear”

Section - EQ3

SALVADOR, Bernard

SIMBAJON, Gabrielle

TALICOL, Matthew

I. Introduction

The project, Atmos-Fear, is a game similar to “Plague INC” , a game where you must evolve a virus to infect the entire world; however, Atmos-Fear is a game where the objective is to control certain aspects of the environment to make the world a better place. The game provides an engaging and interactive experience where players are tasked with maintaining the stability of the environment. The game displays different kinds of gauges that measure all kinds of elements such as “Tree Count,” “Air Quality,” and “Garbage Quantity.” Through interdisciplinary skills, players can learn about the problems associated with climate change and explore ways to reduce its effects.

Climate change is one of the most extreme global problems we, as humans, face today. Its effects such as, rising temperatures, air pollution, waste accumulation, etc., are becoming more prevalent. By creating a project that focuses on the chosen Sustainable Development Goal, “Atmos-Fear” aims to educate, raise awareness, and engage people in understanding the important role we play in protecting our planet.

The main goal of this project is to increase awareness of climate change and motivate people to take action to reduce its effects and adapt to it. The game helps people to see the effects of their decisions and appreciate the value of good environmental stewardship by replicating a world that is similar to our own. The project attempts to contribute to SDG 13 by encouraging people to explore creative solutions in the game and by promoting the understanding of climate-related issues.

Environmental Management. Players will be able to manage and control different environmental aspects, such as tree count, air quality, and garbage quantity, to promote sustainability.

Difficulty Customization. Players are able to adjust the difficulty level to allow them to adapt to the challenge according to their skill level and preference.

Saving Feature. A save system will be implemented for the players to save their progress and continue playing where they left off.

Simplistic GUI. The user interface will be simplisting, ensuring that the players can intuitively understand the metrics and gauges.

Easter Eggs. Hidden surprises and discoveries will also be implemented, adding excitement and to encourage players to explore more.

However, the game will be developed using only the Java programming language, which can limit some creative aspects that can be implemented into the game. The

game will also rely on a simplistic interface, which means there is no additional information that can be relayed to the player besides the provided.

II. Methodology

1. Technical Design and Architecture: **(MAJOR PHASE)**

- Identify the required software and hardware resources to develop the game, considering the target platform (Java) and any additional dependencies.

- Design the software architecture, considering factors like modularity, scalability, and maintainability.

- Define the data structures and algorithms necessary for managing game elements, such as the tree count, air quality, and garbage quantity.

2. Prototyping and Iterative Development: **(MAJOR PHASE)**

- Develop a minimum viable product (MVP) that includes the core gameplay mechanics and basic functionalities.

- Implement a simplistic GUI that provides users with an intuitive interface for interacting with the game.

- Iterate on the prototype based on user feedback and playtesting, adding or refining features as necessary.

- Incorporate customizable difficulty levels, allowing users to adjust the game's challenge based on their preferences.

3. Content Creation and Integration: **(MAJOR PHASE)**

- Create visually appealing graphics, animations, and sound effects to enhance the user experience.

- Design and implement additional game elements, such as easter eggs, power-ups, or challenges, to increase engagement and replayability.

- Ensure seamless integration of the content into the game, maintaining consistency with the overall theme and user interface.

4. Testing and Quality Assurance:

- Conduct rigorous testing to identify and fix any bugs, glitches, or gameplay imbalances.

- Perform compatibility testing on different Java platforms and devices to ensure optimal performance and user experience.

- Implement user feedback mechanisms within the game to gather suggestions and bug reports from players.

6. Continuous Updates and Enhancement: **(OPTIONAL)**

- Monitor user feedback and analytics to identify areas for improvement and new features.

- Regularly release updates, bug fixes, and new content to keep the game engaging and address any emerging issues.

- Stay up-to-date with technological advancements and player preferences to maintain the game's relevance over time.

Encapsulation involves bundling data and methods that operate on that data within a single unit (class). In the "Atmos-Fear" project, various classes, such as AtmosFearGame, GUI, Player, and Earth, will encapsulate their specific functionalities and data. For example, the AtmosFearGame class will encapsulate the game's state, including tree count, air quality, garbage quantity, and time remaining. It will provide public methods to update these attributes while ensuring that the internal state remains consistent and valid.

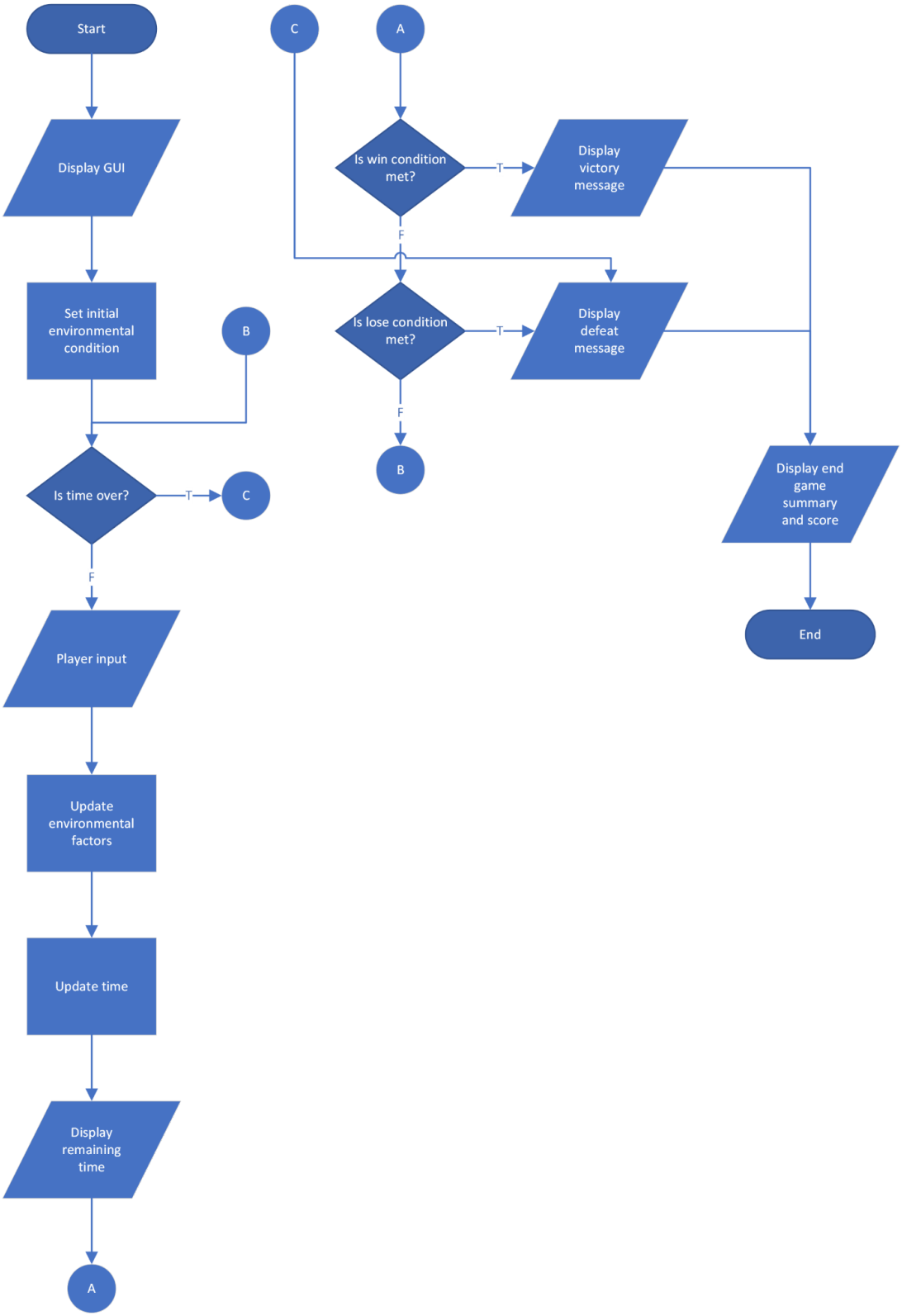
Inheritance allows the creation of new classes based on existing ones, inheriting their attributes and behaviors. In the project, inheritance can be used to create specialized classes that share common properties with the base classes. For instance, the GameFeature class can serve as a base class for various additional features in the game, such as power-ups or challenges. Subclasses can inherit from GameFeature and provide specific implementations for each type of feature.

Polymorphism enables objects of different classes to be treated as objects of a common superclass, allowing more flexible and dynamic behavior. This principle can be applied in the project to create a consistent interface for different game elements. For instance, the game can have a Drawable interface that defines a method like draw(). Both the AtmosFearGame and Earth classes can implement this interface,

enabling the GUI to call draw() on any drawable object without knowing its specific class.

Abstraction involves defining the essential characteristics of an object while hiding the unnecessary details. In the project, abstraction can be used to create clear and concise interfaces for the game elements. For example, the Player class can abstract away complex score management logic, exposing only the necessary methods like increaseCurrentScore(). This allows other parts of the code to interact with the Player object without worrying about its internal implementation.

III. Project Description

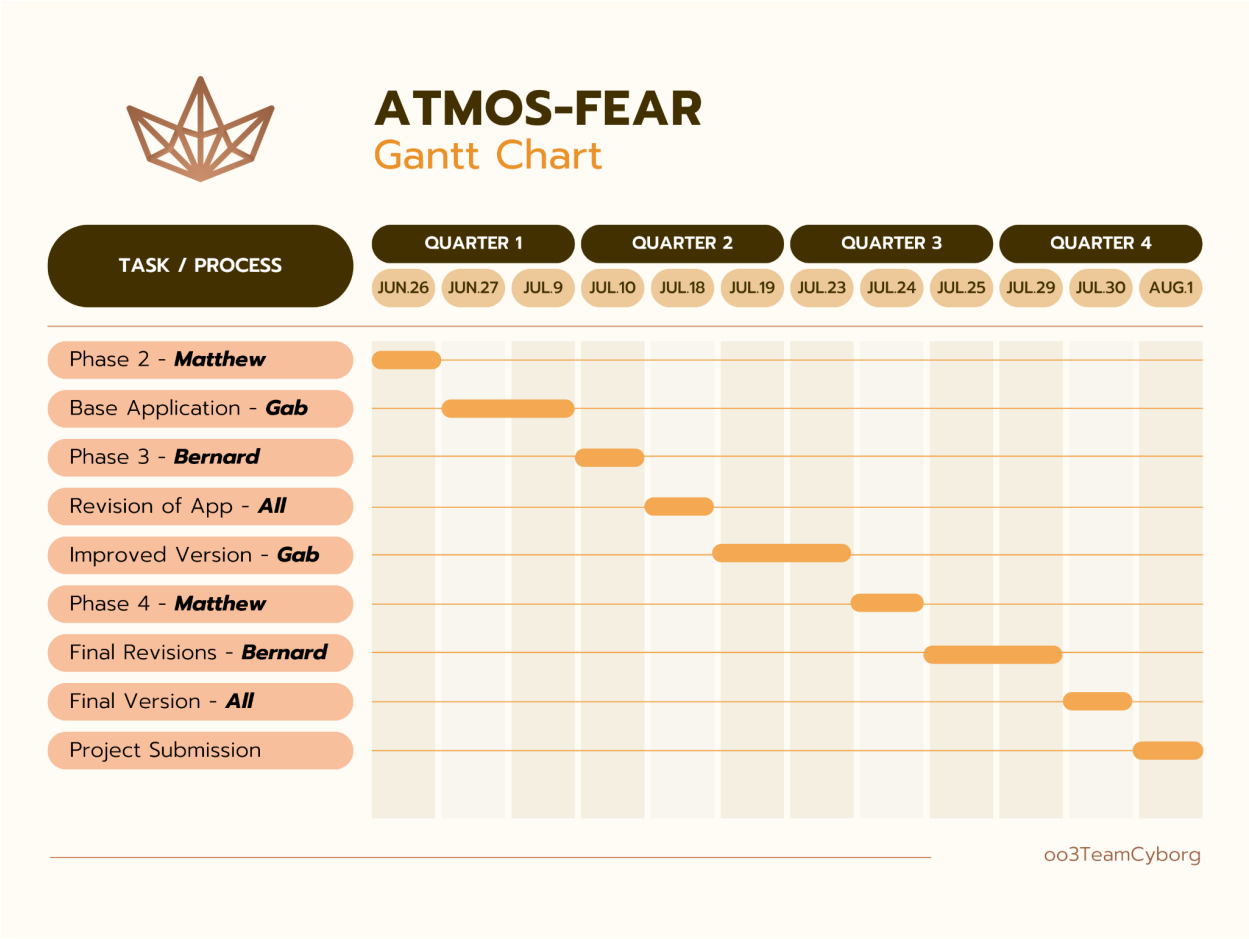


INPUT	PROCESS	OUTPUT
• Player Decisions: The game takes input from the player in the form of decisions regarding environmental management. These decisions could include actions such as planting trees, implementing pollution control measures, or managing waste disposal. • Difficulty Level: The player can input their preferred difficulty level at the start of the game, which determines the complexity and challenge they will face. • Time: The game incorporates time as an input, tracking the progress and	• Environmental Simulation: The game processes the input received from the player and applies it to simulate the effects on various environmental factors. For example, if the player decides to plant more trees, the game will calculate the impact on tree count and air quality. • Decision Evaluation: The game evaluates the player's decisions based on their impact on environmental factors. It determines the consequences of these decisions and adjusts the environmental	• Visual Feedback: The game provides visual feedback to the player through the user interface, displaying gauges and metrics representing tree count, air quality, garbage quantity, and remaining time. This feedback allows players to monitor the current state of the environment and the impact of their decisions. • Victory/Defeat Messages: If the player successfully meets the win conditions, the game displays a victory message, acknowledging their achievements. Conversely, if

allowing players to make decisions within a specific timeframe.	conditions accordingly. • Win/Lose Conditions: The game continuously evaluates whether the player has met the win or lose conditions, based on predefined criteria related to tree count, air quality, and garbage quantity. This evaluation takes into account the player's progress and the current state of the environment.	the player fails to meet the win conditions or triggers the lose conditions, the game displays a defeat message. • End Game Summary: At the end of the game, the system presents an overview of the player's performance, including their score, the time taken, and the state of the environment. This summary provides closure to the gameplay experience.


AtmosFearGame
treeCount: int
airQuality: int
garbageQuantity: int
timeRemaining: int
difficultyLevel: int
isGameOver: boolean
AtmosFearGame()
startNewGame()
updateTreeCount(amount: int)
updateAirQuality(amount: int)
updateGarbageQuantity(amount: int)
updateTimeRemaining()
checkGameOver()
saveGame()
loadGame()
GUI
drawGameUI(game: AtmosFearGame)
displayGameOver()
displaySettings()
displayDifficultySelection()
displaySaveConfirmation()
displayLoadConfirmation()
GameFeature
activateFeature(game: AtmosFearGame)
Player
playerName: string
highScore: int
currentScore: int
Player(name: string)
updateHighScore()
increaseCurrentScore(amount: int)
Earth
stability: int
Earth()
updateStability(treeCount: int, airQuality: int, garbageQuantity: int)
getStability()

IV. Deliverables



V. References

"SimCity titles are real-time management and construction simulators. Across most titles, the player (acting as mayor) is given a blank map to begin and must expand the city with the budget provided. As the city matures, the player may be able to add government and other special buildings (such as a mayor's house or courthouse), depending on how large the city is. Proper management of the city requires citizens to be provided with basic utilities (electricity, water and sometimes waste management) along with public services such as health, education, safety, parks and leisure facilities. These are provided by building relevant buildings or infrastructure, with each building covering a circular "range" in its vicinity. Inadequate funding of these services can lead to strikes or even urban decline."

Wikipedia contributors. (2023a). SimCity. Wikipedia. https://en.wikipedia.org/wiki/SimCity

“Plague Inc. is a strategy-simulation game in which the player indirectly controls a plague, which has infected patient zero. The player can choose between game modes and pathogens and complete the objective set by the game mode by evolving the plague and adapting to various environments. The goals include, but are not limited to, infecting and killing the world's population with a pathogen, enslaving the world's population with the 'Neurax Worm' or converting the world's population into zombies with the 'Necroa Virus'. However, there is a time pressure to complete the game before humans (the opponent) develop a cure for the plague.”

Wikipedia contributors. (2023b). Plague Inc. Wikipedia. https://en.wikipedia.org/wiki/Plague_Inc.