horizontal line

**Department of Electronic & Telecommunication Engineering (EN)**

**University of Moratuwa**

Proposal – Hybrid Farm

EN3350 - Software Design Competition

# Team Introduction

|  |  |
| --- | --- |
| **Name** | **Index Number** |
| L. G. S. Clarance | 200094R |
| V. Kurrshanth | 200331T |
| C. M. C. C. Miranda | 200396U |
| R. Sairisan | 200552V |

# Game Concept Overview

## **Concept Summary [Hybrid Farm]**

Our game is a 2.5D point-and-click strategic gameplay that enables the player to manage the production processes on a farm and makes decisions to increase profit and score using the available resources. The resources the player will be managing are different livestock [farm animals]. There can be chicken, pig, and cow as livestock [not finalized]. As the profit increases the player will have the ability to build two types of small factories to produce byproducts from the base products. The player will be using transport vehicle to his products to market and receive money. The player purchases grass to feed the livestock. The player has to take strategic decisions on which livestock to raise using his initial money, which factories to be built, which transport vehicle to be used as he continues the gameplay while making profit. There will be a time limit for the player to achieve certain objectives like achieving a profit target, producing a certain number of eggs or meat [base products] or producing certain number of egg powder packets or egg powder cake. At random times, the farm will come under attack from certain predators. The player must capture the predator [by continuously clicking] before it kills the livestock. The player then can sell the captured predator or just keep it as it is captured. But the things he can sell in one go will depend on his inventory space. There will be time delay after each purchase of grass for the next purchase. Also, there will be a time delay to produce byproducts from factories and to receive profit from the transported goods. A player must utilize his resources in an optimized way to achieve his objectives within the time limit. According to his finishing time, he will get gold, silver, or bronze trophies.

The energy saving actions of the player which are retrieved through the API every 10 seconds will affect how often the farm comes under predator attack, how many predators will attack in one go and the time delays mentioned above for profit generation, grass purchase and byproduct production. When building factories, for each type of factory the player gets to choose from solar-powered and fuel-powered factories. When choosing the solar-powered factories, the player gets the advantage of extra time for completing his objectives but at the cost of more money for building. Similarly, when upgrading his transport vehicle, he gets to choose electric vehicle or fuel powered vehicle. When choosing the electric vehicle, he can get the goods delivered faster in turn make profit faster, but the upgrade will cost more. In order to achieve trophies, the player must strategically take his decisions on upgrades.

## **Educational Objectives**

The game inspires the player to make decisions on energy efficient upgrades in the game play to receive added benefits while instilling the importance and benefits of those decisions. Throughout the gameplay we motivate the player to go for energy efficient upgrades by providing them with the advantages they can get. For example, even though the initial cost is high they can make more profit as the gameplay goes on with these energy efficient upgrades.

# Development Plan

## **Phase 0 - Player Authentication/Registration**

We implement a component for handling the API calls for the whole game. This component will be the controller for getting and sending API responses and requests. There will be a component for loading the main menu which will talk with this API controller to get necessary values to authenticate the player, do the authentication and load the main menu.

## **Phase 1 - Player Profile**

We will create another component for the player profile, and we retrieve already available information through our API controller and save it in our player profile component. In the main menu component, we implement an option for the player to create profile. When he clicks that we access the player profile component and retrieve already available values in a form like interface where there will be empty spaces for the users to fill in the missing details. After filling in the details and pressing submit button the missing values will be updated in the profile component locally, then the API controller will be called to save the profile details through API. If any information is not filled in there will be a prompt to fill in the missing information and after completing the profile the game environment will be visible.

## **Phase 2 – Questionnaire**

For the first time players after the profile completion, the main menu component will prompt the user to click a button for the questionnaire asking the player to attempt. After clicking the button, the main menu component will redirect the player to the questionnaire web application in a separate browser tab and waits for the response with the boost score from the web app. Once the player enters the web application, the backend will retrieve the questions from the database one-by-one with the answer choices and send them to the front end, there it will format it for the user to select the answer. Once the answer is selected it will be sent to the backend, and the backend evaluates and stores the results in database. After completing every question, the backend retrieves the evaluation results from the database and the correct answers with the questions and sends them to the frontend to show the results to the user. While this happens, the boost score of the player also will be calculated in the backend. After the player reviews the questions and clicking the button the webapp will redirect the player to the main menu component with the boost score. To make this communication happen between these two applications, we will use the API controller to handle local HTTP communication.

## **Phase 3 - Game Environment**

As mentioned in the game description the farm will come under attack from predators at certain random times. The frequency of these attacks and number of attacking predators will depend on the player’s energy consumptions retrieved by the API calls every 10 seconds. We use the “View daily power consumption by current month” API provided to get the daily power consumption of the user for the current month, and we implement our own algorithm to calculate the nature of the above attacks. Also, the time delays for profit generations [Time to transport the good to market receive money], for consecutive grass purchases and for by-products production completion will change according to the player’s energy consumption. This also will be done by the algorithm. So a player will experience difficulty to play the game when his energy consumption increases.

## **Phase 4 - Leaderboard**

The player will receive trophies [Gold, silver, and bronze] according to the time he takes to complete the objectives and pass the levels. There will be a defined score for each trophy that the player achieves. Also, the player will get extra score to his choices in energy efficient upgrades and the profit he makes. In the game environment there will be a button to view the leaderboard. To make the leaderboard we use the “View Player List” API to get the list of each player. To calculate the leaderboard, we use the current player’s score as the base value for which we use the retrieved other players’ information to calculate a variation using our own algorithm and calculate the other players’ score. This leader will be calculated at a certain time interval using the current score of the player and will be saved in the database and retrieved for the view when clicking then ‘Leaderboard’ button.

# Expected Challenges and Solutions

## **Challenges**

* Difficulties in making the game art. [Models, Sprites]
* Remote team-members.
* Difficulties in finding the audio and sound effects for the game.
* Learning new language [C#, Java], software stack [Unity, Photoshop, Illustrator, React, Aseprite].
* Time constraint due to internship workload.

## **Solutions**

* Learning graphic design for making simple and unavailable game arts.
* Using royalty-free game arts, sound effects and software.
* Using GIT, GitHub, and MS Teams to collaborate.
* Learning through online courses and other online platforms.
* Proper time management and workload management.