Kenneth Shortrede

Computer Science Major – Class of 2021

Software Engineering Projects

This document includes a list of four projects that I have worked on or am currently working on.

- Capstone RPG Project: developed a turn-based, fighting RPG game in Unity3D, with a randomized maze generation, as a final project for my Computer Science Major. Worked with a team of 4, and the main tasks I was in charge of were:
  - Mission System
  - Dialogue System & Interactions with NPCs
  - Save and Load System
  - Connection between scenes and persistence of data
  - Menus and GUI
  - Main Scene design, lighting, and camera
- Fitness Mobile Application for a client. Project done in different sprints, covering the various parts including Database Integration, Authentication, User Input, Recommended Routines, etc. I am using a form of project management based on the Agile Methodology '*Scrum*'.

- Personal Game Development project done in Unity and utilizing C# and Visual Studio for coding. It consists of two teams, and allows the user to purchase units, change gameplay (3rd person to 1st person), and track his units as well as select them and fight with their abilities.

- Statistical Project: done for a professor at Rollins College in conjunction with a group of 3 other students. It is an application on Statistical Equations, that produces a txt at the end with all the due results. The project was completed using Java and Eclipse, and was packed in order to be used in both MAC and Windows operating systems.

Capstone RPG Project

As a final project for our major, my team and I are developing an RPG with the classic elements of a turn-based fighting game, and the added value of a randomized maze to enhance the experience of the player.

GitHub for project in its most updated status: Kenneth's Repository

- Video Sample
- Take me straight to the Pictures!

Application Functionalities include:

- Dialogue System: ability to interact with the different NPCs in the environment.
- Mission System: closely connected to the Dialogue system and with the possibility to accept and complete 3 types of missions:
  - Delivery: Interact with NPC #1, deliver certain item to NPC#2, and either finish the mission there or go back to finalize and receive reward.
  - Execute: Accept mission and eliminate a certain number of enemies of a defined monster type to finalize.
  - Escort: transport a certain NPC to a new location without them getting harmed
- Save & Load Systems: maintain the information in a Binary Formatted file to make it harder for the user to change its values and affect the gameplay. The system will save all the information needed to return to the game at the exact place where it was left. This includes:
  - Player and NPCs positions
  - Player's current mission list
  - Current states of all the missions (Not started, Started, Completed, or Finalized)
  - Player's statistics
- Fighting System: will be triggered when interacting with an enemy and generate a new scene to fight the opponent on a turn-based way, with the added opportunity to make multiple attacks in a row depending on the speed of the 2 fighters.
- Maze Generation: randomized generation of a maze where the player will encounter different enemies and/or NPCs to obtain missions. Every time the player wants to get into the maze, it will get generated randomly using an algorithm.
- Menus: this includes the Main Menu, a Loading Scene to show the player while asynchronously generating the scenes, a Pause Menu during game, and a secondary Pause Menu during the Maze.
- Movement System: which includes movement and animations for the player, the NPCs, and the enemies.

Capstone Project Pictures:

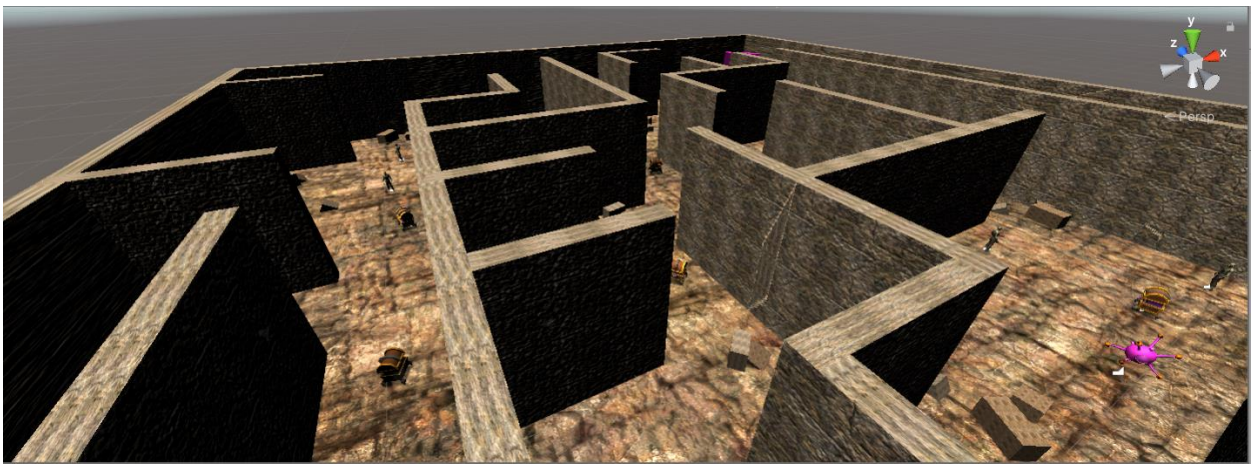
As the different parts of the project get integrated, more screenshots and videos will be added. For now, the following shows screenshots and diagrams of the following finished parts: Main Menu, Loading Screen, Save & Load System, Mission System. For more information and scripts, look at the github link → [Kenneth's Repository]

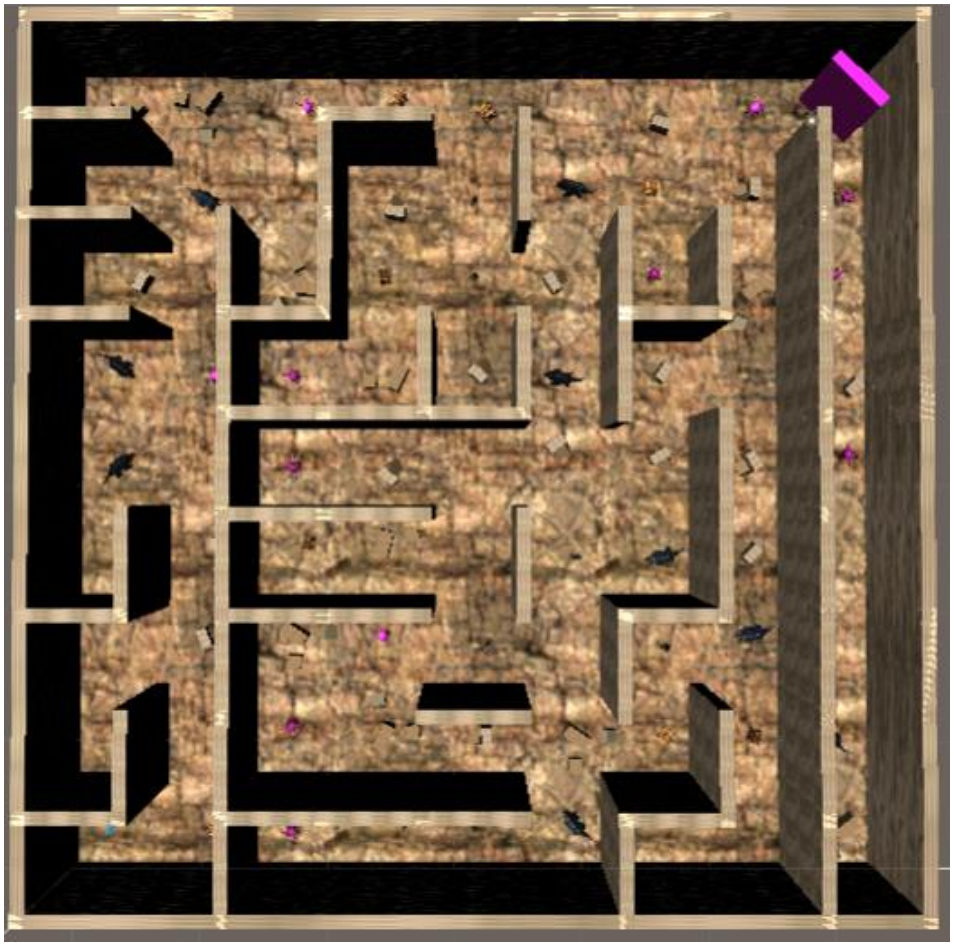
Dialogue System Screenshots

The dialogue system will be in charge of managing the correct interactions with the NPCs for a good user experience showing different dialogues depending on the states of the NPC and the player (Player already has too many missions, NPC has no missions available, NPC's mission hasn't been started yet, NPC's mission started, NPC's mission completed). It will also be connected with the mission system in order to add new missions to the player whenever needed.

Randomly generated Maze

The maze is generated randomly using an algorithm whenever the player gets inside. It will keep its state until the user exits it. Below, a side and top view of 2 randomly generated mazes populated with monsters and objects (also randomized)



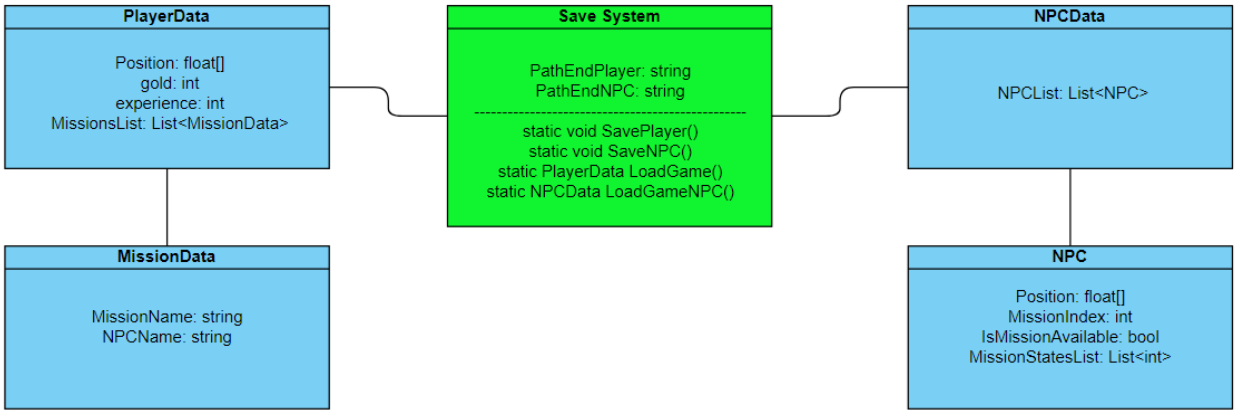


Fighting Scene

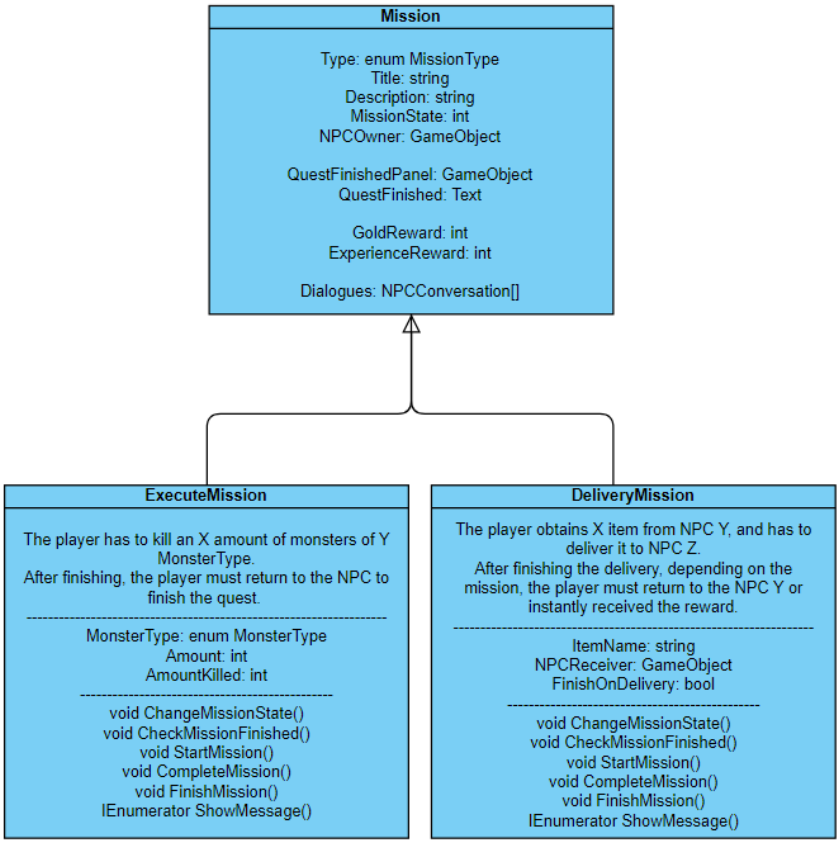
The fighting scene is triggered whenever the player collides with an enemy in the maze, and takes you to a turn-based fighting system against that same enemy.

On win, the player receives a certain amount of experience, and if he levels up, his stats are updated and his HP and Mana reset.

Save & Load Diagrams: The saving system works by abstracting the data of the game into a PlayerData and an NPCData objects, and passing that to the SaveSystem class to save them to a binary formatted file. The save system methods will be called by the Main Menu or Pause Menu when the player decided to save or load the game.



Mission System Diagrams:



Fasting Shape

Fitness Application

- Developing an application for a company client utilizing flutter SDK and Android Studio.
- Take me straight to the pictures!

Application Functionalities include:

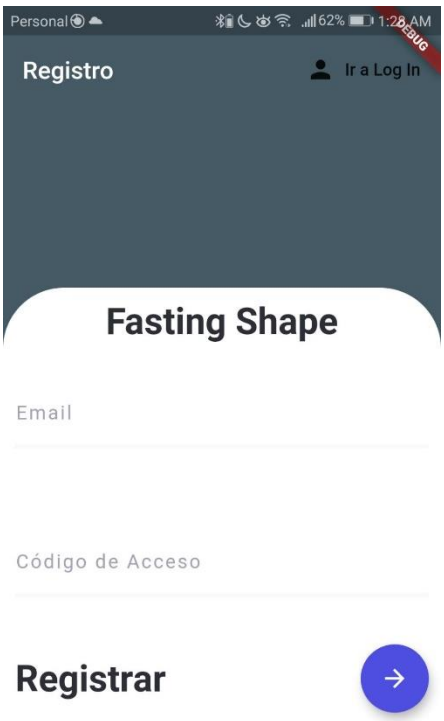
- Main activity with options to enable/disable motivational notifications and reminders, go to the different services in the app.
- List of training exercises with explanations and videos. Exercises and videos will be downloaded from a database in Firebase at the first start of the application.
- Screen for specific exercise with videos/longer description/images and the possibility to add the exercise to the daily routine.
- Daily Routine for each day of the week that the user can see, insert new exercises to, delete exercises, or modify exercises. It also contains a button to set the routine as completed.
- Recommended Routine based on the user selected preferences and current workout experience. It will show on the main menu. It is downloaded from the Database, but customizable by the user.
- List of foods with the caloric intake for a specific amount taken. This will be downloaded utilizing an API from Open Food Facts (https://world.openfoodfacts.org/)
- Daily food intake that the user can change by selecting eaten foods with the amount consumed. This class will show the user how much above or below his set limitation is and will provide the user with advice.
- Intermittent Fasting screen that will allow the user to set a timer for the desired fasting time and will allow the user to enable/disable notifications for the hours remaining until next meal is "allowed."

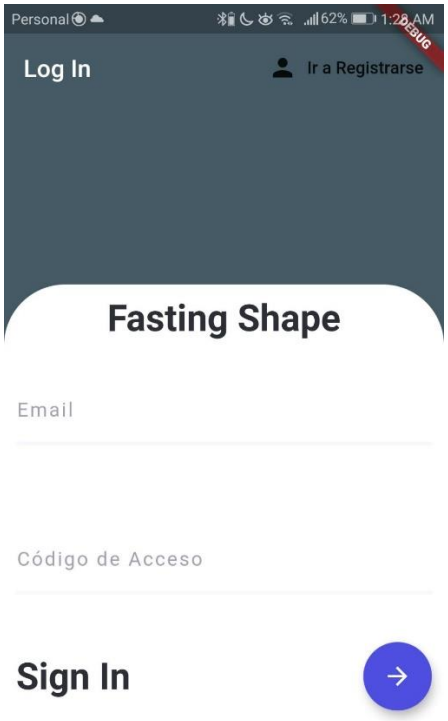
There will be an approximate of 7 different activities (different screens) that will be updated based on information from database and/or previous use by the user.

Fitness App Screenshots

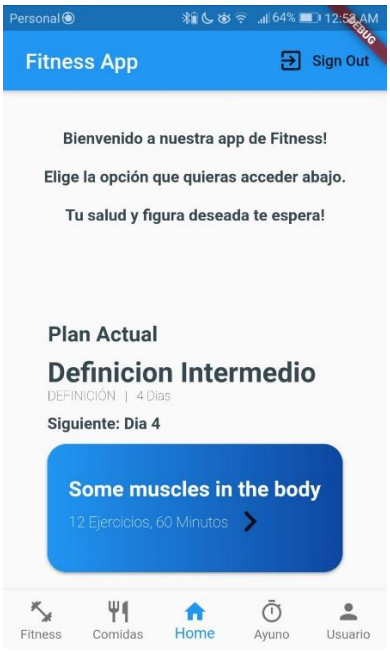
 # Important #: Some of the project screens are in Spanish, as the project will be released for a client in Argentina, and a company which focuses on Spanish-speaking users.

Register and Login Screens that will generate an account on Firebase for the user, and will maintain its login status. If a new account is created, the User will then be prompted to add some extra information such as gender, workout experience, goals, etc.



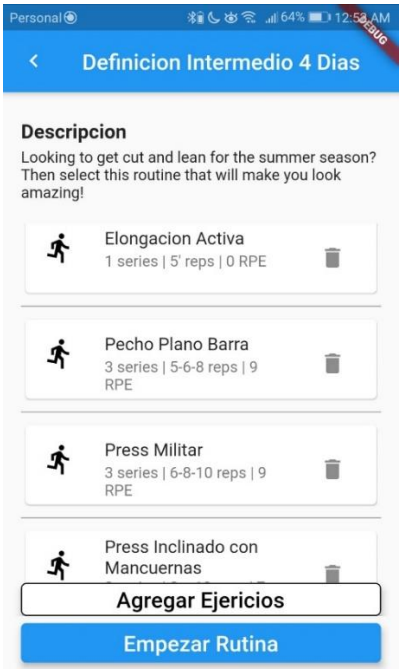


On the left, the profile's information with the data set for the generated user and it's selected email. The user can change the information, which will generate changes in the app and what is shown to him/her. On the right, the home screen with the automatic recommended routine.



On the left, the recommended routine, that was chosen based on the user's preferences, has been opened, and the different exercises are shown. The user can add or remove exercise before starting the routine for the day.

On the right, we can see one of the exercises shown once the routine has been started. All the information was downloaded from the database, and will be shown to the user. The user can also select the buttons on the top to go back, change the exercise, or view the past logs (information created every time the user does a certain exercise)



Personal Project - Unity Game

- Programming of a game with two teams, and AI controlled units, that decide on their paths based on the other objects in their view. User is able to create new units based on the gold he has, and units fight each other in order to reach the other player's nexus.
- Take me to the pictures!

Requirements:

- Two teams, with multiple possibilities of units to create
- Units move and travel on their own, following semi-random patterns
- Units also interact with the environments and other enemy units around them
- When a unit meets another enemy unit, it interrupts it's moving pattern, to attack the enemy at once
- Enemy units are created randomly, with specific chances of creating each different unit to balance out their power levels.
- The use can select units in order to transform the game into a 1st person gameplay
- User has a certain amount of gold to purchase units
- Game has a main menu
- Game offers a tutorial with easy explanations of the game before starting
- User can save the game, which saves the position and state of the units, the amount of money, etc.
- The player earns points whenever an enemy unit is killed

Personal Unity Game – Screenshots

Units move through the map until they find an enemy, or the enemy base to attack

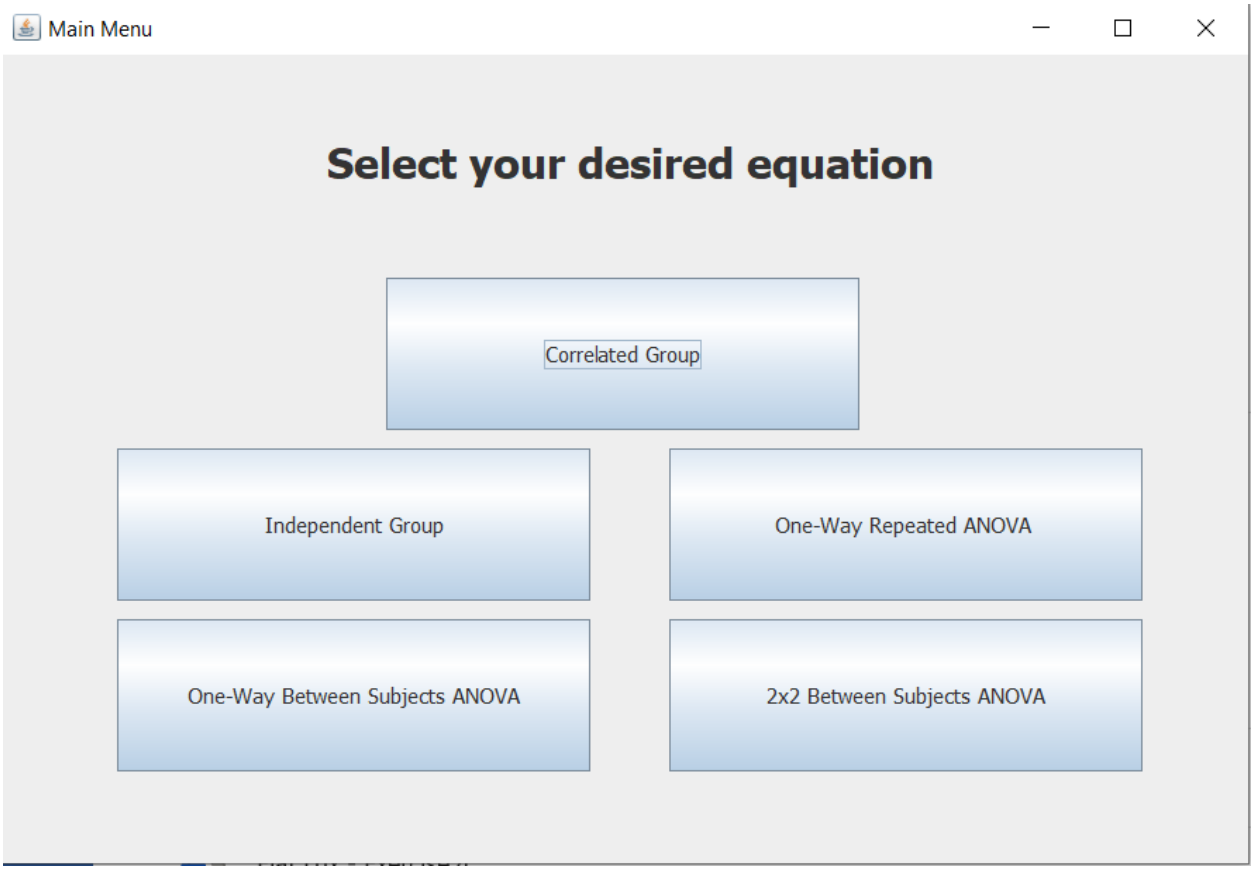
Purchasing a new archer – As gold is too low, fighters or Hero can't be bought.

Statistical Project

This project was created in an Object-Oriented Class in collaboration with a professor from the Psychology department which utilizes a wide variety of statistics in his classes. Since he had to grade so many different exercises a ton of times, we developer an app for him to have an easier time solving the exercises. The app is mostly back-end with ways to solve the different equations, and a simple GUI that the user sees.

Application Requirements:

- Solve 5 different equations for the User
- Allow to input different data sets
- Allow to input the desired tests
- Provide a user-friendly interface
- Provide correct results
- Print the results to a .txt generated in the user's folder
- Provide an executable that is packaged with Java, so the user doesn't need to have it installed in his/her computer
- Provide a working program for both Windows and Mac OS



Example of one of the tests: