INDIVIDUAL PORTFOLIO

A personal reflection on the Group 3's Game Project

Dizon Joselito Villanueva
joel.dizon@live.com

Introduction

This document provides my personal reflection as well as provide a technical report on my participation for this project. The technical report provides a high-level view on the Game Setup – a subset of my overall participation. The flowchart only covers the Game Setup component as per requirement for this documentation.

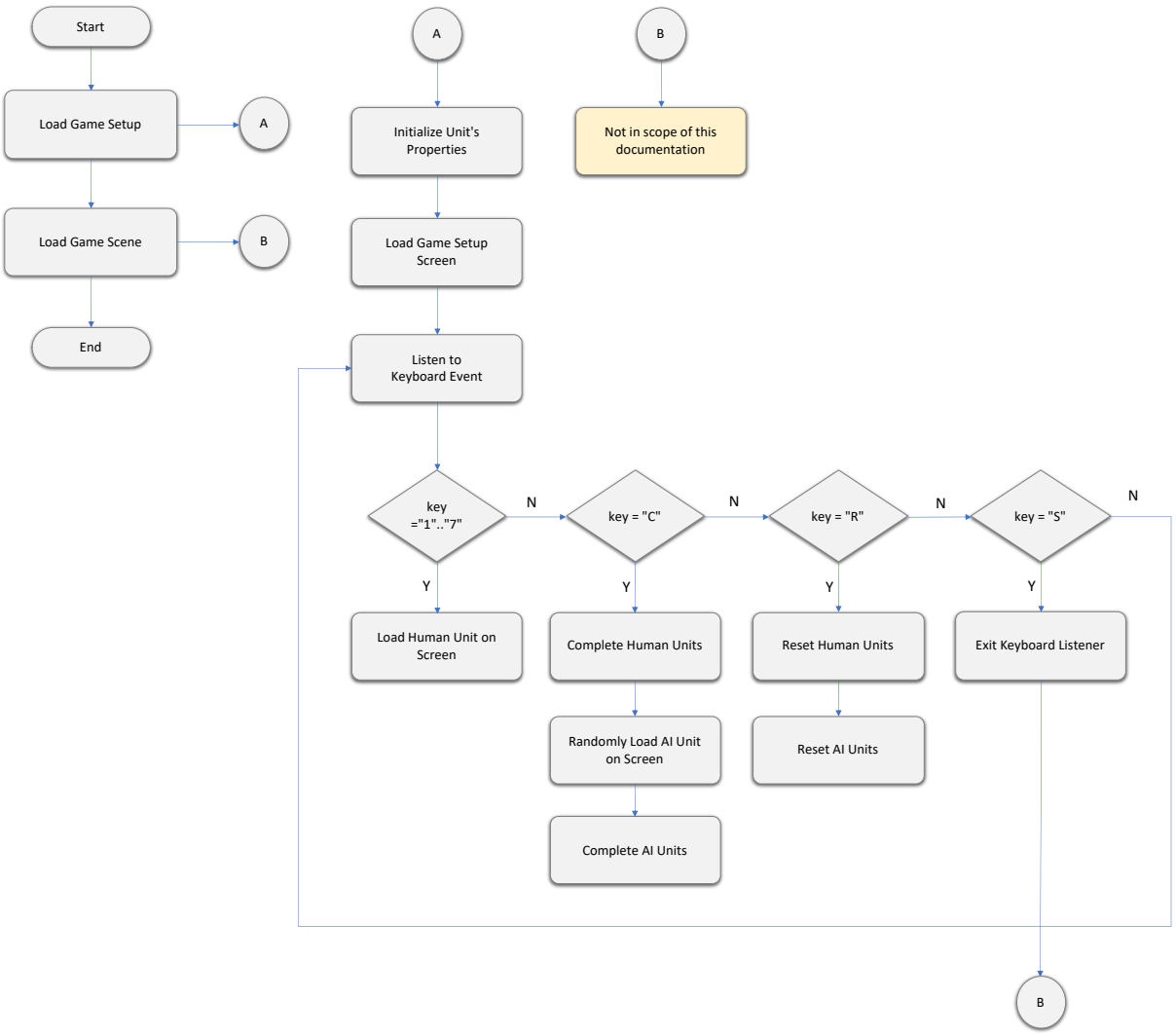
The source code is also provided as evidence of the implementation of the Game Setup. Since this module is tightly knitted to the Game Scene module, I have provided the whole source code for the sake of completeness. The test report is also included which details the test results of each part of the Game Setup. Evidence of the test results is also provided.

The personal reflection provides my personal walkthrough on the project, the thought process, challenges, solutions/workarounds, and most of all the learnings in this experience.

Technical Report

1. Flowchart

The diagram below captures the high-level flow of the Game Setup.

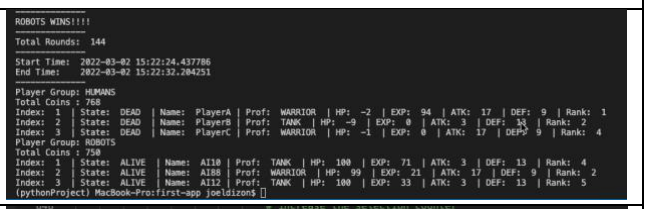
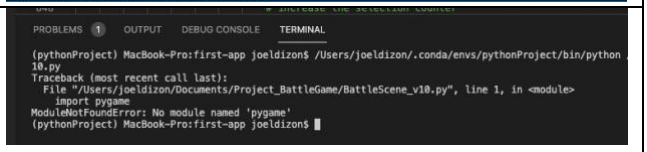
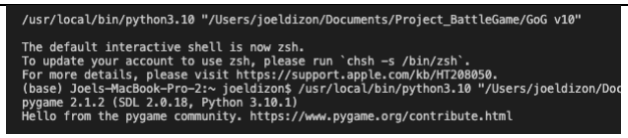


2. Source Code

As most of the programming work was done by me. I am attaching the whole program.

GoG_v10_SourceCodes.docx

3. Test Report

#	Test Case	Results	Evidence
1	Business Logic Test	Passed	
2	PyGame Test on Macbook Pro M1	Failed	
3	PyGame Test on Macbook Pro Intel-based	Passed	
4	Load Game Setup	Passed	
5	Selection of Human Units – Selection 1 - 7	Passed	

6	Selection of Human Units of more than 7 (should not be allowed)	Passed	
7	Reset Units selection using key “R”	Passed	
8	Complete selection using key “C”. Game auto select units for AI	Passed	
9	Start the game using key “S”	Passed	

Personal Reflection Report

Finally, we have done it. In hindsight, the experience was not a walk in the park. While it was fun, it was equally tough knowing four guys who have no programming backgrounds are relying on you to deliver most of the programming works. The pressure is real. It was a sad undertaking but, in the end, it was one of the most fulfilling experiences I had in my life. I wouldn't say I am new to programming. I have several projects in the past ranging from languages Pascal, C++, Java, and C#. However, this is the 1st time to develop program in Python and Pygame in this magnitude.

Planning

The first challenge I had after the team had decided of what the solution will look like is how to distribute the works within the team. Knowing that most of the members have no programming experience, the course is still in early stage and most of the key Python topics have yet to be discussed. To address the capability problem, we need to take advanced studies on Python object-oriented programming as well as development in Pygame. Me and the team have undergone two weeks of training watching YouTube videos to understand Object Oriented Programming and how to develop game using Pygame. I have identified 5 aspirations for the game (listed in table below), however, due to time and capability constraints only 3 have successfully delivered.

#	Component	Requirement Type	Delivered
1	Introduction / Title and Post Game Credits	Nice To Have	No
2	Game Setup	Must Have	Yes
3	Game Scene	Must Have	Yes
4	Logging	Good To Have	Yes
5	Persistence and Resumption	Good To Have	No

Design

The overall architecture has followed the classic layered architecture. For this implementation, I used combination of classes and common functions (modules) to implement the business logic of the game. It is not an easy task. There is always a debate in my mind whether a function can be a method or just remain a generic or common function. In the end, the finished product ended up the best of both worlds.

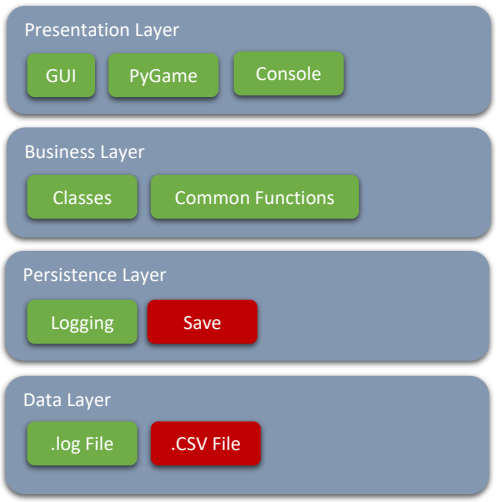


Figure 1 - Layered Architecture for Game of Champions

The next challenge is how to store information for the Player Units. The basic principle must be:

1. Information must persist throughout the game;
2. Should be self-explanatory or can be easily identified (e.g. unit.healthpoints, unit.name, etc)
3. Must be iterateable or referenceable (e.g. unit[0].healthpoint, unit[0].name, etc);
4. Easy to implement

The options are to use either a plain list, a class, a list within a class, or a class within a list. Using a plain list is super easy, and especially using a loop. However, it is not self-explanatory. I'm sure it will be difficult for me to refer to unit[1] as name, compare to unit.name. The next option is to use a class. It's the next best thing it complies to principle 1 and 2, but to my knowledge, and its basic structure you can't iterate. Hence, efficient referencing using a loop is not possible. The next option is to use a list within a class. While it complies with Princples 1 and 3, it has problems in the association of class properties such as names, healthpoints, etc. The final option is to use a class within a list. This is my preferred option for example, human[0] = Player("HM01","Warrior",…). Hence, human[0].name = "HM01", human[0].type = "Warrior". Same goes for the AI instance.

#	Options	Example	Principle 1	Principle 2	Principle 3	Principle 4
1	List	HumanUnit[0] 0 = Unit Number	Y	N	Y	Y
2	Class	Human.Unit.Name	Y	Y	N	Y
3	Class + List	Human.Unit[0] 0 = Unit Number	Y	N	Y	N

4	List + Class	Human[0].name 0 = Unit Number	Y	Y	Y	Y

Development

I started by experimenting on Pygame to understand the complexities of the engine. It is complicated and hard. While there are paid software where you can "draw" the controls on a screen, in Pygame a freeware, you must code everything. So a lot of trial and error hours are expended on the User Interface.

After the UI and core structure such as main loop and events tracking are set, I moved on to develop the business logic. This is where most of the effort are focused. Then COVID hit me and my family. During this period, I have to stop developing and focus on myself and my family's well-being. For the next 2.5 weeks no development for me. I knew we will not meet the deadline if a didn't go back to the project. Hence, for the next 3 weeks I worked hard to catch up. There were times that I must work 72 hours programming, with 8 hours of sleep to finish the business logic. Shifted to 4-5 hours nightly to refine the business logic. The development process is not a smooth sailing. Throughout development I have to refactor several functions to merge to a class. For example attack() and heal() are originally a common function. However, it would be more efficient and more self-explanatory if these are in class object. Rather than using attack(human, ai), it's better if human[0].attack (ai[0]). While it's always ideal to plan and design ahead to avoid refactoring (as it is time consuming), in most cases design considerations are happened by necessities discovered during the development process. I always believe change during development are inevitable on most scenarios. However, this does not negate the planning and design benefits, planning and design must have provisions for future change.

Testing

Manual testing is done throughout the development process. However, testing the logic during my development has been very slow and time consuming. It may not be able to capture most of the logical errors. Hence, I have devised an automated process to test the business logic. Testing a 300-round fight will probably finish in 5 hours using manually guided play. For the automated test, it is completed in just 5-10 minutes. The manual test is done after the UI and Business Logic are integrated. The team have done a great job capturing problems during the actual turn-based fight.

Integration

There are 2 programming works assigned to members. The game setup and logging. Unfortunately, final version of game setup and logging have integration problems due to requirements and technical

incompatibilities. So, I must revise 70% of game setup and 50% of the logging module to be able to fully integrate. Looking back, I think better communication must be in place between team members to understand the progress, code review, changes in methods and requirements. Also, better team development tools like Git should have been adopted.

Conclusion

This project has never been completed without the challenges that me and my team faced throughout the development process. While the team is handicapped in terms of programming experience, and lack of communication amongst members exacerbated by COVID restriction, the overall experience is very fulfilling knowing you have overcome those challenges. On the personal level, I have learned a lot from this subject, it thrust me to learn Python which I can immediately use in my line of work. I always believe challenges are there to test our mental fortitude, and every time you succeeded with the help from your teammates, guidance from your teacher, understanding and support from your family you gain inspiration and motivate you to move forward.