Tamagotchi Pet Simulator Game

Adrian Armenta

2054931

Project final report

My experience:

Project 2 has been a lot of fun to work on, the amount of creativity and freedom we had in this assignment was amazing, it really made me get into project-making, and it made me learn the proper way to organize my ideas before creating a project. At first, the project seemed a little difficult, but once I started coding, I really enjoyed every part of it, or at least most of it, bugs and glitches were not fun to address, especially those that took hours to fix. I want to continue improving this project and add more features during the summer. Now I want to explain a few changes the project had that differ from the first report as well as how some things were designed.

1. Text

   Description automatically generatedThe classes:

Figure Private members of the parent class

The classes were straightforward to implement, I created some variables to control the status of the pet, and I also made them private because we don’t want people to mess with them.

Text

Description automatically generatedAfter that, I created a couple of getters and setters so we can modify the private values in the derived classes, a constructor, and the classes with the activities you can perform with the pet, as well as functions that work with the game loop. I also added a destructor to free the memory each time you create a new pet, I forgot to add it in the initial report, but it was implemented later. 3 Classes were inherited from this class: Cat, Dog, and Duck, each has different attributes that make them unique.

Figure Public members of the parent class

Text

Description automatically generatedThe derived classes were a bit more tedious to do since I had to override most of the functions to have different dialogs, sprites, interactions, modifiers, etc. But the number of functions used was reduced by a lot.

On the next page, I will explain one example of a function that is different from the other classes.

Figure Public members of the Dog class

Text

Description automatically generatedThis is an example of one of the functions inside the Dog class, it can also be found inside the parent class and the other pet classes, however, each of them has different modifiers depending on the personality of each pet, for example, happiness can increase more or less depending on the type of pet and the action performed, also, a new message is displayed, as well as a new sprite.

1. The functions:

Graphical user interface, text

Description automatically generatedA lot of functions had to be overridden to have new interactions for each type of pet. One of the most important functions is the display status function that updates the status of the pet, and it also displays a new image every time the mood of the pet changes.

A picture containing text

Description automatically generatedText

Description automatically generatedI accomplish this by using fstream, at first I tried doing “couts” but it was very difficult to pull off, so I created several files containing the sprites of each pet. Each pet has 3 statuses, happy, sad, and sleepy.

Text

Description automatically generatedGraphical user interface

Description automatically generatedGraphical user interface

Description automatically generatedGraphical user interface

Description automatically generatedAlso, every time you perform an action with your pet, it will show a little sprite with a phrase and the program will stop for a few seconds, I accomplished this by using the library “unistd.h”

I think this addition gives it a more professional look.

Load Function:

This was arguably the most difficult function to implement, it takes a save file, reads the type of pet, and then calls the setters to restore the progress of the pet, however, I found myself trying to make it work for hours since it kept either missing the correct information, the correct type of pet, etc. I had to rewrite the function many times, but I finally made it work. It takes a pet pointer as an argument and then reads the file and sets the status to the ones on the last time you played.

1. The menus:

Text

Description automatically generatedThe menus were easy to implement, they’re just loops with a couple of switches. For example, the main menu is a loop that has 3 options, each option does something different, option 1, creates a new pet pointer, it shows the available pets and asks the user to select one, then it asks the user to name their pet, the program calls the set name and set type methods and then it calls the game loop function to start the game.

Text

Description automatically generatedFor option 2, the program deletes any already created pet to free the memory, then it reads the save file to get the type of pet, and then it creates a new pet depending on the type, then it calls the load function and the game loop function to start the game. And option 3 simply will exit the program.

Text

Description automatically generated

Finally, the game loop function is a menu that constantly calls the display status method, each of the options will call a different method, and after a certain amount of action will call the next hour method in order to represent the passage of time.

Conclusion:   
There are many other functions and actions that I will not be able to explain in much detail, but this is a general idea of how the program was designed, it had a lot of changes with respect to the initial plan like the destructors and some implementations of the main loops such as the load option which I had to rewrite several times, and some parts were stressful to develop but in general, it was a fun experience, I will definitely come back to the project and more things in the future, I also learned how to use more libraries and workarounds for some of the problems I encountered through the development. I think the functions are easy to read and by looking at them it will be easy to understand how I implemented my code. All the artwork was designed with the same fstream principle and I think that was the most enjoyable part of this project, I think I could probably create a function to handle those in the future, but other than that, I’m very happy with the results.

Video Demonstration: <https://drive.google.com/file/d/1xw9lER9MIMyd2jaeRRYAkhb5hbqJFjoV/view?usp=sharing>