Software Engineering II - Group 6 - Homework 1

Bauhaus-Universität Weimar, 15.10.2022

1. What is our software system?

This project has the working title of “Tamagotchi Pet Simulator”. Our software project aims at creating a digital pet-like species that users have to take care of and can play with. The aim is to be able to create and customize this pet according to the wishes of the user, whose task will be to keep their pet alive and happy.

First, when creating the pet, the user will be able to create his first one by setting a name and sex. The age of this pet will be automatically set to 0 while the attributes for hunger, hygiene, and attention to 10 (the highest, happiest value). The age of the pet will increase with time, whose concept will be further explored in the next steps of the project.

The score of hygiene, hunger, and attention will decrease after a while. However, the values of those attributes can be improved by taking certain actions: To improve hygiene, the bath can be cleaned, the toilet can be scooped, and the pet can be groomed. To improve the attention score, the user can choose whether the pet shall play with a ball, stick, or yarn. Finally, to improve the hunger score, the user may feed the pet an apple, bread, or steak. All of those actions have a different type of impact and will shape the preferences of the pet.

As soon as the pet becomes an adult, meaning reaches a certain age, the baby pet transforms into one of three pet species. The species depends on what nutrition the baby consumed when growing up, with what it played, and how it was taken care of overall. Combining those factors, the game chooses what species your pet has grown into.

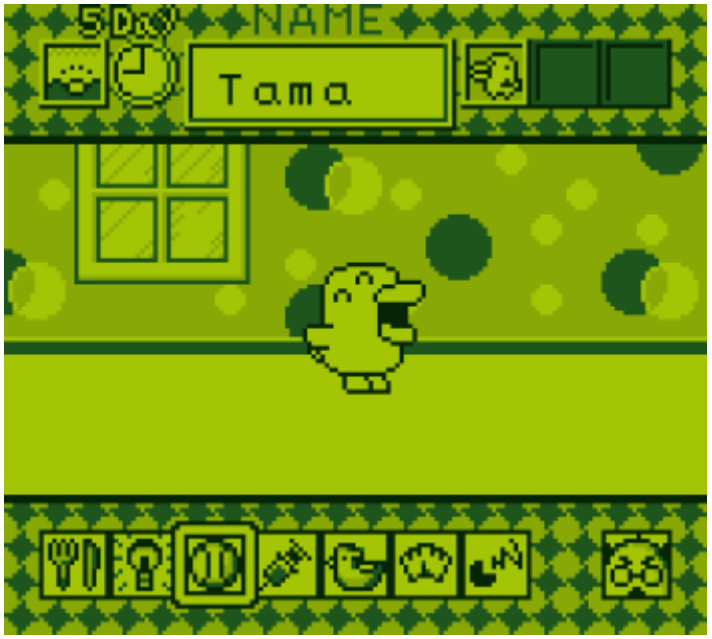
A possible extension would also be to create multiple pets after the first pet has acquired a certain age. Thus, actions can be implemented simultaneously according to the needs of the corresponding pet. When those pets are adults and if they are female and male, they can also mate, creating one to two new baby pets.

This project is easily extendable. The goal of the game is to enjoy raising and playing with virtual pets.

2. What are its main requirements?

Before clarifying the main requirements, we have to ask and evaluate what the users want or “need”. In a user centered design process, we have to conduct an analysis of requirements with data gathering methods like interviews or questionnaires. Since this project is only for this course, we can simplify this step, but the main requirements may be faulty or change in the following weeks. Some of the requirements are already defined in 1.) .

Functional requirements - What should the software do?
The software should let users choose what name and sex the pet should have. The pet has different age stages and different main stats (hunger, hygiene, attention). These stats deplete over time and can be raised through selecting hunger, hygiene or attention.



Therefore, the user can influence the health and happiness of the pet. Further elements like mini-games can be added.
The system displays relevant information through text and has only visual feedback. Although, sound and pictures could be added as a new function. The game can be saved with a saving function and resumed later.
The goal is to keep the pet alive and if its needs are not met, the pet may die. The life stages of the pet are: baby, child, teenager, adult and senior.

picture source: https://www.deadpark.com/articles/tamagotchi-gameboy/ (27.10.2022)

Non-functional requirements - constraints on system

The software should work as an executable file on any personal computer system and must be programmed with Java.

Data requirements

Progress of the game should be saved in a specific save file for each pet. The data amount should be small.

Context of use requirements

This game should be playable in home or school environments. The output text needs to be readable in any light. It is a single player game, but a multiplayer function is a good extension.

User requirements

The users need to be able to read English. Other than that, they can have any characteristics and capabilities. They have to download the files and manage to open the .exe file & close it.

Usability requirements

A great user experience is key. It needs to be easy to learn and come back to. The output text has to be clear but also entertaining.

3. What should we implement first?

A stable basic structure needs to be implemented. It should contain this framework:

1. Choose name and sex of pet

2. Random presets: favourite food and toy
3. Major Stats: Health - divided into: Hunger, Hygiene and Attention
4. Save file of some sort + time system, that makes stats go down over time
5. Activities to refresh stats (Feeding, cleaning, playing)
   5a. different types for each
      → Feeding: Apple, Bread, Steak;
        Cleaning: Bath, toilet, grooming;
        Playing: Ball, Stick, Yarn
6. Feedback-Text!

a. What features might be easy?

The first, easy features that we will implement will be the setting of the pet, meaning the user will be able to create it with its pre-set condition. The main idea here is to create a class for this pet with those three attributes mentioned above. Furthermore, we will be able to create functions that all entail the action that the user may perform, falling in the category of Hunger, Hygiene, and Attention. Those actions will thus improve the score and render the pet happy.

Another easy implementation would be the action of 'mating' of two pets. As soon as they have achieved a certain age, they are able to reproduce themselves by creating one to two babies. The user will be able to name the new pets immediately, while the sex is determined randomly. After that, we aim to be able to assign different actions to different pets, thus, keeping all the pets happy and alive.

Finally, the last easy feature to implement would be the death of the pets. By implementing another function, we will be able to determine whether a pet has to die due to a high hunger, low attention or low hygiene over a long period of time.

b. What features might be difficult?

Seeing as that the primary mechanic behind a classical tamagotchi pet is the passing of time and its effect on the pet, we will have to come up with a method of keeping track of time that has passed, whether the user is currently using the program or not. The progression of the pet's emotional and physical health is directly linked to the time that has elapsed, so we will require formulas that can calculate the rate of deterioration of each individual stat based on the time difference.

In order for our players to have a goal to work towards (other than keeping the pet alive), we want there to be a variety of "final forms" that the pet can assume, once it has reached maturity. The final forms will be dependent on the general level of care, and activities as well as consumables the pet was given. An exact method of implementation to achieve this, may prove difficult, due to the number of factors we would have to keep track of.

To be able to return to a pet, or to begin caring for another, we will without a doubt require a robust save system that can store all relevant information between accessing the program at different points in time. In order to get this up and running we will have to code a file loader and file writer which may take some time.