Programming a
Virtual Pet

Project Assignment 2014-2015

Introduction

In this project we will perform an experiment in programming an **integrated system** in Scheme. The hardware to do so will be delivered to you by us. This consists of a **micro-controller** that has been equipped with a number of **sensors.** This document only specifies the bare essentials of what we expect from you. However, individual creativity and functional extensions are heavily encouraged and will be rewarded as such.

The concrete goal of the assignment is to program a **virtual pet** and run the pet on the micro-controller in order to really connect it to the **physical world** by means of the aforementioned sensors. For example: the pet might go to sleep as soon as an accelerometer indicates that there is no more movement.

Minimal Functionality

Actuators en sensors

The pet must foresee the following **sensors** in order to get input from its owner: three buttons and an accelerometer. These sensors will affect the mental state of the pet.
The pet must foresee the following **actuators** in order to communicate its mental state to its owner: three LEDs and a graphical LCD screen.

Vitale Characteristics

The behaviour of the virtual pet largely depends on its **vital** characteristics which can gradually vary in time between two extremes. As such, a rebellious pet will display unwanted behaviour more frequently than a submissive one. The actual value of the vital characteristic will be determined through the interactions with the owner. E.g., the rebellious pet can become more submissive by punishing it or by playing games with it.

You minimally have to model the following vital characteristics:

state of mind: varies between happy and unhappy

compliance: between submissive and rebellious

hunger: between satisfied and hungry

health: between healthy and ill

exhaustion: between awake and exhausted

These vital characteristics have to be displayed on the **LCD Screen** at any moment in time. E.g. you might display a row of characteristics in which every characteristic is indicated by a colour that corresponds to the current value of the characteristic. You are free to come up with more appealing solutions.

Interactions with the environment

The virtual pet should minimally support the following interactions with its environment and/or its owner:

asking for attention: whenever the value of one of the vital characteristics becomes critical, this has to be indicated on the screen. A rebellious pet can ask for more attention than really needed (i.e. even when the value of the vital characteristic is not necessary critical).

playing a game: the owner of the pet can play a simple game with the pet as soon as it feels unhappy. The game consists of guessing which LED will be lighted by pressing the corresponding button. When guessed correctly, the state of mind of the pet improves.

feeding: If you don't feed the pet sufficiently frequently, the pet dies! You can feed the pet with two different amount of food: meal and snack. Meal are healthier than snacks (i.e. eating too many snacks will make the pet ill).

sleeping: after a while, the pet gets exhausted and needs some sleep. You can make the pet go to sleep by turning the pet upside-down for a short period. If the pet does not receive sleep sufficiently frequently, it dies of exhaustion.

cleaning/toilet: The pet will leave droppings around the screen from time to time, and can become sick if they are not cleaned up. Before the pet needs to use the bathroom, it will advice the player. If the player activates the toilet action during this advise, but before the pet has gone to the bathroom, the pet will use a toilet instead. When done repeatedly, the pet can be toilet trained.

ill/drugs: The pet can "die" due to poor care and sickness. The pet can become sick for a number of reasons such as overfeeding of snacks or failing to clean up droppings. The pet can die if sickness is left unchecked. The pet can be cured by pressing the "Medicine" option.

Autonomous Operation

The virtual pet should work **autonomously** and should implement its own "live loop" such that the REPL of Scheme is not used to change its state. I.e. **it has to be a real application and not a bunch of procedures that have to invoked by the owner.**

Creativity will be rewarded: The more realistic the virtual pet, the higher your mark will be!

Architecture

The pet will be conceived as a **sense-process-act** loop which perpetually reads the sensor data and processes that data by means of an appropriate action. However, the action not only depends on the sensor values but also on the current state (i.e. the values of the vital characteristics) of the pet. Hence, you will model the processing part of the sense-process-act loop as a **finite state automaton**. The automaton contains states that have two procedures that will be called upon **entering and leaving** that state.

We will give you skeleton code that stipulates how your automaton should be constructed and used by the pet application code. It is up to you to program the automaton ADT and to write the main loop of the pet, given a concrete automaton. The main loop has to call the sensoring as well as the actuating code. We will provide you with the libraries to access the sensors and to manipulate the buttons and the screen.

Guidelines for the submission of the project

1. Specifications for the submission of the report:
    - **Date** : January 6, 2015
    - **Location** : Students Secretariat of the Computer Science Department, local 2N8.104.
    - **Time** : before 15h (The secretariat closes at 16h.).
    - The project submitted after 16h will be considered not submitted on time. This will imply the loss of half of the points on your final score (plus one point per day late). Delayed projects must be deposited in the box provided for this purpose near the secretariat.
2. Specifications for the delivery of the code:
    - One unique ZIP file or tar.gz must be submitted.
    - The archive must be named «SCIP.studentName.zip».
    - The archive must be sent to lporrett@ulb.ac.be.
    - The subject line must be exactly «SICP-Projet2015».
    - If you do not meet any of these criteria, your code will be considered not submitted.

A demo of your project, through the hardware that has been delivered, will be required during the oral examination.