Byron Laferriere

Dr. Ostrowski

IT-209

**Final Project**

**PROJECT PROPOSAL**

The robotic application that I will be designing for the Final Project this semester is an entertainment toy for cats. The robot will emulate a rodent being in the house and supply noises, quick bursts of speed, and changing lights to pique the interest of the animal. By using contact sensors on the robot, the robot will rely on its closed loop control system to creatively adapt to its ever-changing environments. These sensors will communicate information on full-contact or close-contact points to the robot’s intelligence, in order to be relayed to the actuators for a physical reaction to occur.

The robot will also come with a home that is only accessible by the robot. It will use UV to communicate to sensors on the beak of the robot, to help it ‘home in’ and align for entrance to the robots dwelling. The robot will run on a timer and then return to this home, so that the animal does not become bored by too much association/exposure. The closed loop controller in this robot will be the major difference in what is offered by competitors. Other automated solutions to this point are still offering a small sample space of what the entertainment toys are capable of doing and thus, are losing the interest of the animals they are designed for. Combining this fact with the home that the rodent returns to, which is inaccessible to anything else, will keep interest piqued longer than any other toys.

This solution will frequently change its combinations to make new noises, velocity burst distances, and lights so that the animal will not become easily disinterested in the toy. There is a very popular trend in animal care and different ways to spoil the creatures we care about. This robot will be a hit with the community that is looking to entertain their furry friends while away at work all day.

**ANALYSIS**

The Finch Cat Toy would not be one of the first designs in the cat toy market. There already are different options for cat toys with limited interaction and in a much different price range. There is a cat toy with robotic integration called the AMBUSH by PetFusion. This toy is similar to a Roomba in design but has much less functionality and only spins and drives while the feather attached to the side attracts the cat. The second application that is similar to the Finch Cat Toy is an interactive laser by Friends Forever. This stationary application has a laser on a constantly spinning tower with three ‘chase’ speeds. Another robotic application in this cat toy category is a stationary design with a swinging arm possessing attachment capabilities by YUETOYS. These toys each differ in drastically in the price different from my Finch Cat Toy design because they have one functionality. The Finch Cat Toy instead has many different actuators being used, which is why it is more expensive. However, the combined effect of using lights, noises, and motor functions will provide longer entertainment value and prove its own value over time.

**PROTOTYPE**

The two critical behaviors that will be addressed for this prototyping assignment will be escape left and escape right. This assignment was tricky at the beginning, so I had to visit the BirdBrains website to get some ideas of how to begin incorporating these behaviors. When the robot detects an object with its sensors, it will begin the escape sequence that will combine the three available features within the robot. When an object is detected within the 2-4 inches range, the Finch will plan a short escape and change its light according to the direction it is escaping. Once it has escaped, it will emit a squeak to tease the cat playing with it and then change back to the original driving color. I wrote out some pseudocode to lay out the design that I wanted to instantiate for the escape behaviors. This helped me visualize the blocks that would be needed before I began actually laying the blocks.

**Escape**

**WHILE ( Finch != Home)**

**IF ( SENS\_L == TRIGGERED)**

**LEFT WHEEL SPEED == 100;**

**RIGHT WHEEL SPEED == 50;**

**BEAK COLOR == GREEN;**

**NOISE == ON;**

**ELIF (SENS\_R == TRIGGERED)**

**LEFT WHEEL SPEED == 50;**

**RIGHT WHEEL SPEED == 100;**

**BEAK COLOR == RED;**

**NOISE == ON;**

**ELSE (**

**BEHAVIOR == DRIVE MODE;**

**)**

**ENDWHILE**

The next critical behaviors that will need to be addressed to move this project closer to completion will be the drive mode and then home. This exercise has given me a good idea of how to design these behaviors and was a massively helpful foundational piece for the final project. By using a ‘forever’ loop in combination with ‘if’ loops, I was able to combine two competing behaviors within the robot to cooperate. These nested ‘if’ loops repeat until the left or right obstacle is identified, respectively. By pressing key number one, the right obstacle escape is initiated. The left obstacle avoidance behavior was assigned the number two. I also began to work on the ‘drive’ mode within this zipped project but will need to come back to it to perfect it. This behavior was initialized under the number three for now. I look forward to spending more time with this project and coming one step closer to the final project completion.

**SOFTWARE DESIGN**



**DEV AND TESTING**

package **CatToyProject**;

import com.birdbraintechnologies.Finch;

/\*\*

\* Created by: Byron Laferriere

\* Date: 06/20/2021

\* Cat Toy Program

\*/

/\*

\* This code was posted by the CMU team as an update for testing the Finch bot

to ensure everything is working properly when in the 'home' state. This program

aligns with my previously defined home behavior in the UML submission for ‘Home’. The finch

will check each sensor with Booleans when turned on and then also terminate when off is selected.

\*/

import CatToyProject.FinchCatToy;

import java.awt.Color;

import edu.cmu.ri.createlab.terk.services.accelerometer.AccelerometerGs;

import edu.cmu.ri.createlab.terk.services.accelerometer.AccelerometerState;

public class **FinchHome** {

// Returns the {@link FinchProperties} for this finch. \*/

FinchProperties getFinchProperties();

//Returns the state of the accelerometer; returns <code>null</code> if an error occurred while trying to read the state.

AccelerometerState getAccelerometerState();

// Returns the state of the accelerometer in g's; returns <code>null</code> if an error occurred while trying to read the state.

AccelerometerGs getAccelerometerGs();

// Returns the state of the obstacle detector specified by the given <code>id</code> where 0 denotes the left obstacle detector and 1 denotes the right obstacle detector. Returns <code>null</code> if an error occurred while trying to read the value.

Boolean isObstacleDetected(final int id);

//Returns the state of the obstacle detectors as an array of <code>boolean</code>s where element 0 denotes the left obstacle detector and element 1 denotes the right obstacle detector. Returns <code>null</code> if an error occurred while trying to read the values.

boolean[] areObstaclesDetected();

// Returns the current values of the photoresistors as an array of <code>int</code>s where element 0 denotes the left photoresistor and element 1 denotes the right photoresistor. Returns <code>null</code> if an error occurred while trying to read the values.

int[] getPhotoresistors();

// Returns the current value of the analog input specified by the given <code>id</code>. Invalid analog input ids cause this method to return <code>null</code>. Note that, for finches without analog inputs, this method will always return <code>null</code>. This method also returns <code>null</code> if an error occurred while trying to read the value.

\* @see FinchProperties#getAnalogInputDeviceCount()

Integer getAnalogInput(final int id);

// Sets the full-color LED to the given red, green, and blue intensities. Returns <code>true</code> if the command succeeded, <code>false</code> otherwise.

\* @param red the intensity of the LED's red component [0 to 255]

\* @param green the intensity of the LED's green component [0 to 255]

\* @param blue the intensity of the LED's blue component [0 to 255]

boolean setFullColorLED(final int red, final int green, final int blue);

// Sets the full-color LED to the given {@link Color color}. Returns <code>true</code> if the command succeeded, <code>false</code> otherwise.

boolean setFullColorLED(final Color color);

// Sets the motors to the given velocities. Returns <code>true</code> if the command succeeded, <code>false</code> otherwise.

\* @param leftVelocity velocity of the left motor [-255 to 255]

\* @param rightVelocity velocity of the left motor [-255 to 255]

boolean setMotorVelocities(final int leftVelocity, final int rightVelocity);

// Sets the buzzer to the given <code>frequency</code> for the given <code>durationInMilliseconds</code>. Returns <code>true</code> if the command succeeded, <code>false</code> otherwise.

\* @param frequency the frequency of the tone [0 to 32767]

\* @param durationInMilliseconds the duration of the tone in milliseconds [0 to 32767]

boolean playBuzzerTone(final int frequency, final int durationInMilliseconds);

// Plays a tone having the given <code>frequency</code>, <code>amplitude</code>, and <code>duration</code>. \*/

void playTone(final int frequency, final int amplitude, final int duration);

// Plays the sound clip contained in the given <code>byte</code> array.

void playClip(final byte[] data);

// Converts the given text into speech, and returns the resulting WAV sound clip as a byte array.

byte[] getSpeech(final String whatToSay);

// Converts the given text into audio and plays it.

void speak(final String whatToSay);

// Turns off both motors and the full-color LED. Returns <code>true</code> if the command succeeded,

\* <code>false</code> otherwise.

boolean emergencyStop();

void disconnect();

//Returns <code>true</code> if {@link #disconnect()} has been called; <code>false</code> otherwise.

boolean isDisconnected();

}

public class **FinchCatToy** extends **FinchHome** {

public static void main(final String[] args)

{

// Instantiating the Finch object

Finch myFinch = new Finch();

// Loop while beak != isBeakDown()

while(!myFinch.isBeakDown()) {

// If there's an obstacle on the left, turn LED red, turn right for

//1500 ms then drive forward for 1500 ms. Buzz and then speak

if(myFinch.isObstacleLeftSide()) {

myFinch.setLED(255,0,0);

myFinch.setWheelVelocities(255,30, 1500);

myFinch.setWheelVelocities(250,250,1500);

myFinch.buzz(440, 500);

myFinch.saySomething("You can't catch me");

myFinch.sleep(2000);

}

// If there's an obstacle on the right, turn LED blue, turn left for

//1500 ms then drive forward for 1500 ms. Buzz and then speak

else if(myFinch.isObstacleRightSide()) {

myFinch.setLED(0,0,255);

myFinch.setWheelVelocities(30, 255, 1500);

myFinch.setWheelVelocities(250, 250,1500);

myFinch.buzz(880, 500);

myFinch.saySomething("You can't catch me");

myFinch.sleep(2000);

}

// If there's an obstacle on the right and left, turn LED blue then red,

//back up for 1500 ms. Buzz and then speak

else if(myFinch.isObstacleRightSide() && myFinch.isObstacleLeftSide()) {

myFinch.setLED(0,0,255);

myFinch.setLED(255, 0, 0);

myFinch.setWheelVelocities(-255, -255, 1500);

myFinch.buzz(880, 500);

myFinch.saySomething("Stay Back!");

myFinch.sleep(2000);

}

// Else, robot goes straight, LED is green

else {

myFinch.setWheelVelocities(255,255);

myFinch.setLED(0,255,0);

}

**EXPLANATION OF DESIGN**

After completion of the project, I find it is appropriate to address the successes and shortcomings in the design. The robot was able to successfully begin with a ‘home’ behavior that checked sensors that were relied on for input that would allow the intelligence to communicate with actuators accordingly. Once instantiated, the robot would end ‘home’ and begin the ‘cruise’ method. This method was combined with ‘escape’ as the else statement, instead of creating a separate class for it. I was able to successfully create an escape left and escape right but my code did fall short in this area as well. I tried to use an AND statement to combine both sensors and have them communicate when both were tripped at once. However, I could not assign the proper arbitration for the behaviors to recognize this. The program would begin the escape behavior according to which side was registered first.

To fix these mentioned shortcomings in the project, I would need to study more about arbitration techniques that should be used to read to sensory inputs simultaneously. This also highlighted the need for subsumption to be used in this robotic solution design. Studying subsumption in Java further could help me with understanding how to control the robot better by having arranged priorities. The code looked as if things would work initially but the simulation proved otherwise. When in the physical world, the robot struggled to recognize darker objects in its way, which needs to be addressed as most shadows are usually obstacles. The lighter objects could be recognized from 2-4” regularly and helped the program run easier. Simulation was important in this design because it showcased specific shortcomings that should be used for further investigation into perfecting this robotic solution.