Rochester Institute of Technology
SWEN 563/CMPE 663/EEEE 663

Project 5 - Game

Overview:
Design and implement an embedded, real-time stand-alone system to play a primitive "mirror" game using the terminal for display, and the gyroscopic sensor for input.

Details:
The game: The goal of the game is to have the player "chase" the position of the computer. The game makes use of the terminal for display; The computer moves to 2D position at a random location on the game field. The position of the player by changing pitch and yaw of the discovery board.

Game play is as follows: The computer generates a random position and displays it to the user; The player then has a fixed amount of time to orient the discovery board to the same position as the computer. The terminal is used as live, real time, feedback interface to the player displaying both the computers position and the users' position, while the game is running. If the user makes it to and holds the proper position when the time elapses, then it is a hit; if not it is a miss. Either way – the computer then moves its position to a new random position and starts the game all over again.

This repeats for 5 times; then a score is computed for that level; Then the time is shortened, and the next level is played.

The setup: Using the STM discovery board with putty or terra term, or other serial interface – the program should display the game board and state. The demo board already as a gyroscope that is connected via a bus interface.

(more)….

Design Constraints:

- The gyroscopic sensor is used to determine the rotational position of the board
- You should also use the red and green LEDs to show additional player feedback…
- Positions: can be anywhere within the range of the display. The game field should be at least 40x20 cells; a cell can be thought of as a single character location;
- You will want to drive the serial port at a rate fast enough to be useable (9600 will not be fast enough) – adjust serial port to so that your game updates at least 15 times per second
- You need to send a "clear display" ascii sequence to home and reset the terminal between updates;
- Be sure to seed the rand function before starting the game.
- Successive computer random moves may overlap (be too close ex: within 10 cells distance) – make your game correct for this.
- The fixed amount of time to reach a destination is 10 seconds for the first level.
- Each successive level allows is 2/3 of the time of the previous level: 10; 6.66; 4.44; 2.96, 1.97, 1.31 seconds….
- Continue to level up until the score for that level is 0; or you're at level 6
- Final display the max level achieved and score for each level;
- There are libraries already available for interfacing to the gyroscope; however, you will need to add some signal processing to the raw returned information to get a final value. This is a rate sensor – not an accelerometer; you will need to process the output of this sensor in real time to get an angle.
- Player position counts as a hit if it is within 1 cell of the computers position. (3x3 matrix);
- During play the "player" position is continuously updated and shows the position of the demo board at all times; the player marker directly mirrors the boards position.
- You are free to use an RTOS or not
- You should be using cubeIDE on its own or with Keil;

Grad Student Extension:

Display a "tail" on movements of the user; the tail should be at least 3 cells long. The last 3 positions of the user are always displayed. You can use "O", "o", "." with "O" the previous position, "o" for two positions in the past, and "." for three positions in the past.

Approach:

I recommend you start with a clear design document, and share we me to discuss.
You should have a good idea of ALL functions; threads, queues, interrupts, etc.

Consider initialization and calibration of the gyro
Consider timing constraints.
Consider interfaces.
Consider memory use;

WARNING:
This project does involve moving the demo board while it is attached; Be carrefull with the USB connection - You need to maintain signal integrity at all times.

Report:
In addition to the demonstration of your project, a brief report with the required sections is required. List the trade-offs and assumptions of your implementation. Inclusion of proof of operation is not required for this project – the demo is sufficient. Your source code must be included in your electronic submission.

Grading Criteria:

- Program Operation and Demo – 50%
  - Hardware setup is orderly and well organized – 10%
  - Demo sheet functions all completed – 30%
  - Demo operates without faults or restarts – 10%
- Program Design --- 15%
  - Proper initialization
  - Correct use of functions (no copy/paste/edit slightly)
  - Separation of hardware related code from pure software (e.g. the results reporting code)
- Source Code Structure and Readability – 10%
  - Appropriate use of white space – 2%
  - Consistent and good indentation – 2%
  - Appropriate comments at the function and paragraph levels (such as a for loop) – 2%
  - Following C style guide (good names, etc.)
- Report Content – 25%
  - Report is at least 2 pages (not counting pictures, cover page, diagrams) – 5%
  - Demonstrates team understands the problem, solution, and technology (hardware and software) – 5%
  - For this project it must include a complete description of the communication system design – 5%
  - Report contains all required sections (except as noted above) per the report guidelines – 10%
- Bonus Opportunity – up to 20% at instructor discretion
  - Use the “recipe” concept from the other projects to define several progressively more difficult “levels” through the game;
  - Change the time intervals for each level to increase difficulty