Introduction to Robotics: Matrix Project (8-10p)

Mini 8x8 LED Matrix game

Andrei Dumitriu andrei.dumitriu@fmi.unibuc.ro

Last Updated: November 28nd, 2023, 23:02

1 Description

Dear students, the moment has come when you have to do the matrix project. This year, we'll be doing a bit differently. Depending on the maximum grade you want, you can chose from 3 requirements:

1. For a maximum grade of 10 (+2 bonus): current requirements. A game on a 16x16 logical matrix (it's OK to have only 1 physical matrix) that implements either multiple rooms, visibility / fog of war and/or multiple physical matrix. **Forbidden games:** snake, atari breakout, pong.

2. For a maximum grade of 8 (+1.6 bonus): Same requirements and restrictions, just on a 8x8 logical and physical matrix. Same forbidden games. Basically the grading will be the same but at the end the total value will be multiplied by 0.8.

3. For a maximum grade of 5 (+1 bonus): Snake, with fewer requirements. See separate document.

Remember that you are creating a game! It must be fun and intuitive, and also remember that it is **your game**! Pick something that you like and have fun creating it. You get to keep it for the holidays and show it to your family and friends. Make it count. This is a chance to show what you can build when you are tasked with a complete, final project. Bring it on.

You can use more components than the basic ones: extra led matrix, buttons, joysticks etc. From the 2nd led matrix on, you can use a matrix shield (basically, it connects the matrix with just 5 wires).

Keep in mind that you are creating a menu for **a game**. It should be intuitive and straight down obvious how to use it. A good test is if you give it to someone that has no idea what it is - are they able to use it easily? Think about when you pick up a game - the menu feels intuitive.

Have fun!

See each page for specific requirements regarding the menu, the game and the github repo. We'll keep some of the best games as examples for future students.

2 Menu Requirements

- **Menu Task:** Create a menu for your game, emphasis on 'the game. You should scroll on the LCD with the joystick. Remember you have quite a lot of flexibility here, but do not confuse that with a free ticket to slack off. The menu should include the following functionality:

  1. **Intro Message** - When powering up a game, a greeting message should be shown for a few moments.
  2. Should contain roughly the following categories:
     (a) **Start game**, starts the initial level of your game
     (b) **Highscore:**
     - Initially, we have 0.
     - Update it when the game is done. Highest possible score should be achieved by starting at a higher level.
     - Save the top 3+ values in EEPROM with name and score.

     (c) **Settings:**
     - Enter name. The name should be shown in highscore. Maybe somewhere else, as well? You decide.
     - LCD brightness control (mandatory, must change LED wire that's directly connected to 5v). Save it to eeprom.
     - Matrix brightness control (see function setIntesnity from the ledControl library). Make sure to display something on the matrix when selecting it. Save it to eeprom.
     - Sounds on or off. Save it to eeprom.
     - Extra stuff can include items specific to the game mechanics, or other settings such as chosen theme song etc. Again, save it to eeprom. You can even split the settings in 2: game settings and system settings.

     (d) **About:** should include details about the creator(s) of the game. At least game name, author and github link or user (use scrolling text?)
     (e) **How to play:** short and informative description
  3. **While playing the game:** display all relevant info
     - Lives
     - Level
     - Score
     - Time?
     - Player name?
     - etc
  4. **Upon game ending:**

(a) Screen 1: a message such as "Congratulations on reaching level/score X". "You did better than y people." etc. Switches to screen 2 upon interaction (button press) or after a few moments.

(b) Screen 2: display relevant game info: score, time, lives left etc. **Must inform player if he/she beat the highscore.** This menu should only be closed by the player, pressing a button.

3 Game requirements

- **Game requirements:**
  - **Minimal components:** an LCD, a joystick, a buzzer and the led matrix.
  - You must add basic sounds to the game (when "eating" food, when dying, when finishing the level etc). **Extra:** add theme songs.
  - Each level / instance should work on 16x16 matrix. You can apply the concept of visibility / fog of war (aka you only see 8x8 of the total 16x16 matrix, and you discover more as you move around) or you can use the concept of "rooms". Basically you will have 4 rooms that you need to go through on each level.
  - It must be intuitive and **fun to play**.
  - It must make sense in the current setup.
  - You should have a feeling of progression in difficulty. Depending on the dynamic of the game, this is done in the same level or with multiple levels. You can make them progress dynamically or have a number of fixed levels with an endgame. Try to introduce some randomness, though.
- **Useful resources:**
  - LedControl library: http://wayoda.github.io/LedControl/pages/software
  - 9999 games in one: https://www.youtube.com/watch?v=RPTanMNGmek
  - Cool example of a simple build: https://www.youtube.com/watch?v=gluRdsmNAwU
  - LiquidCrystal library: https://docs.arduino.cc/learn/electronics/lcd-displays
- **Example of from previous years, including menu:** (they do not meet the requirements, use only for inspiration!)
  1. Santa's workshop: `https://www.youtube.com/watch?v=GFQkmgMiz-4` (nice touch with the songs)
  2. Space invaders: `https://www.youtube.com/watch?v=QNpVnlfoA0Q` (good, but should've used the extra button as a trigger)

3. Rush hour `https://www.youtube.com/watch?v=x1dpq7ZVwLE` (great menu, but care too big)
4. Hungry πxl: `https://www.youtube.com/watch?v=-xkd1i6417I` (should've used a blink for player)
5. Racing car: `https://drive.google.com/file/d/11A9FUwp7jgpe2jkebaNmO6tHpTHz0t3x/view` (nice touch with difficulty in settings)
6. Tetris: `https://www.youtube.com/watch?v=y1NTdkCfhUc`
7. Other type of racing game: https://www.youtube.com/watch?v=P7bDNxj5xQ8
8. Pong: `https://www.youtube.com/watch?v=yNq-2f87Ewc` (too simple now)
9. Tower of Power: `https://www.youtube.com/watch?v=k4eKOsfRSm8` (too simple and must input name with joystick, not from PC)
10. T-REX run: `https://www.youtube.com/watch?v=FNnYLSPdGB8` (too simple mechanics)
11. Matrix heroes: `https://www.youtube.com/watch?v=rVG6xQdty2I`
12. Reverse it: `https://www.youtube.com/watch?v=ipwmN_Qqrns`
13. Perilous path: `https://drive.google.com/file/d/13Wp0T1PxEz2l59WMqxqNn7WIKvp2JqQh/view`
14. Snake: `https://www.youtube.com/watch?v=5pgvKoDdCVE`
15. Flappy birds: `https://drive.google.com/file/d/1VHPbNwdJ3gTokfjorNThMMhMGoxdmGS_/view`

4 Publishing task: Github readme

For the Matrix project, **you must create a new github repository**, separate from the homework ones. This repository will contain all the details of the project, in similar fashion to the homework. You must add the code to the Github repo and continue updating the readme with at least the following details (but feel free to be more creative):

1. Task Requirements
2. Picture of the setup
3. Link to video showcasing functionality (I recommend youtube, but anything publicly accesible is fine)
4. Used components
5. Remember to publish the video in the correct orientation. Don't do this: `https://youtu.be/Y8HOPlUtcto`
6. **Hand in the homework on MS teams when done - aka when github is up to date**

5 Development guidelines

1. Create the game in steps. Start with a proof of concept and work up from there. Validate your initial ideas, assumptions and vision.
2. Remember that it's better to have a simple, yet working game, rather than a more complex and buggy one.
3. Make it feel like a complete project as much as possible. Small details and finishing touches separate a prototype from a product.
4. Remember to follow the coding guidelines.
5. Remember that while details are important, the bulk of your grade will come from the fact if it works.
6. Ask the staff for any questions and details. Do not assume instead of asking.
7. Add bonus stuff and make it personal: animations, theme songs, hardware and build robustness etc.
8. Remember that we have way more components in the laboratory that you have not used yet: mp3 players, sensors etc.

6 Detailed grading, requirements and pitfalls

6.1 Detailed Requirements

- **16x16 functionality: if you don't have it, total points will be multiplied by "0.8"** (visibility, fog of war, more matrices etc):
- **Menu: 2p**
  - Intro message
  - **Start Game:**
    - Shown details while playing
    - Screen(s) upon game ending with input in order to move on
    - Informs you when highscore is achieved
  - **Highscore:** at least top 3 in eeprom with name and score
  - **Settings:**
    - Enter name (can be somewhere else, depending on how and where you use it)
    - LCD brightness control (eeprom)
    - Matrix brightness control (eeprom). Light up the entire matrix when changing it, so you can see the actual difference.
    - Sound control on/off (eeprom)
    - Reset high scores button in settings
  - **About:** github link/user (specifcy if user!), developer name and game name
  - **How to play:** short and informative description
  - Navigation style: Usability, intiuitive, beauty. Different "picture" on the matrix for each menu category (hammer for settings, challice for high score etc). Should do a small sound when changing selected menu category etc.
- **Game: 2p**
  - Sounds upon interaction (collision, firing, level up, increase score, losing a life etc)
  - Difficulty progresses so that it's neither boring, nor impossible too fast.
  - Reasonable game length
- **Documentation: 2p**
  - Backstory about the game / your situation (why you chose it, from what game you it is inspired, how you built it etc). Basically an introduction

  - Game description
  - Instruction on how to play
  - Used Components
  - Picture
  - Detailed video of the entire functionality of both the game and the menu (link, not uploaded to github!). I recommend explaining while filming, as well

- **Code: 2p**
  - Overall architecture, correct spacing, correct naming, magic numbers etc. Check out coding styles on the internet and the coding style document and github.
  - Good luck, you'll need it

- **Feeling: 2p**
  - Usability, intuition, fun, creativity, flow and how it all goes together as a product
  - Rewards an entire product - even if rough on the edges - and discourages just checking different requirements without fitting them in

6.2 Possible bonus points:

- **Easy:**
  - Different animation / image on matrix for each sub menu item, as well
  - Theme songs when starting the game

- **Medium:**
  - Multiple settings for sound: theme songs, menu sounds and game sounds with separate settings, saved in eeprom
  - Sound volume control (using a potentiometer instead of resistor)
  - Easter eggs
  - Add sliding bars or an interesting animation for settings
  - Creative ways of inserting name
  - Not having to reset the game due to the wiring

- **Hard:**
  - Building a case around it
  - Add pause option
  - Add save game option
  - Add automatic save in case of power outage
  - Multiple games

6.3 Possible subtracted points: 2p

- Asking for name in settings but only showing it in high score (if so, it makes sense to only ask it if the player defeats the high score)
- Not adding sound effects when scrolling through menu
- Having contrast settings outside visible bounds
- Having a too low or too high step when setting contrast and/or LCD and matrix brightness
- Not showing arrows or something similar on the LCD menu in order to easily understand current position and the fact that you can go down or up in the menu
- Having the joystick position different in menu and game, basically forcing you to switch joystick direction when you start the game
- If I find bugs that you did not specify beforehand (either you didn't know or you try to lie), the penalty is greater than when presenting them upfront
- Having to reset the game too many times
- So many more things...