Functional Specification

Year: 2024 Semester: Spring Team: 1 Project: Dungeon Crawler Board
Creation Date: 01/19/2024 Last Modified: 01/19/2024

Member 1: Landon Carre Email: lcarre@purdue.edu
Member 2: Neil Brown Email: brow1950@purdue.edu
Member 3: Jackson Luna-McCrocklin Email: jlunamcc@purdue.edu
Member 4: Grace Whitaker Email: gwhitake@purdue.edu

Assignment Evaluation: See Rubric on Brightspace Assignment

1.0 Functional Description

The Dungeon Crawler Board is a physical game board that helps to visualize and simulate the dungeon crawler experience. The gameplay is based on a simplified version of Dungeons and Dragons, simulating combat, healing, and looting. It will be a turn-based game, using various pathing algorithms for the movement of both players and enemies, with the Dungeon Master controlling the enemies. Magnets in the board and on the character tokens will keep track of changes to position. The map will be displayed on the 16 by 16 hexagonal board with different lights underneath the hexes representing floor, walls, character positions, and chests. The map and character information will come from a separate laptop using a custom-made application. The board will be interactive, with prompts displayed on an LCD screen that will facilitate the gameplay, and a keypad will be used to respond to the prompts. The game board will do a lot of math involved with dungeon crawlers, but dice-rolling will still be done by the DM and players.

2.0 Theory of Operation

The Dungeon Crawler Board uses a hexagonal-based map, which offers many challenges to standard pathing algorithms, since most are quadrilateral-based. Also, because hexagonal maps do not exactly line up with any standard data structure, either a modified data structure with substantial amounts of empty space must be used, or the map must use a coordinate system that can be translated into a standard array [1]. The project will use offset coordinates since it allows for the easiest translation to a standard 2D vector data structure, similar to an array.

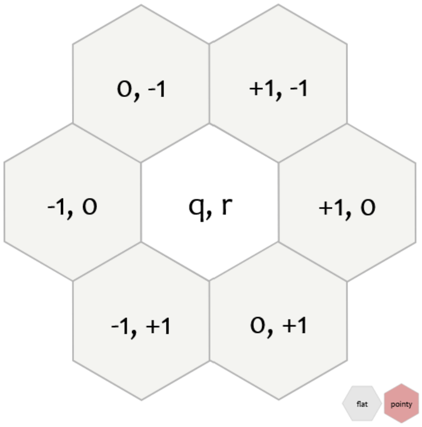
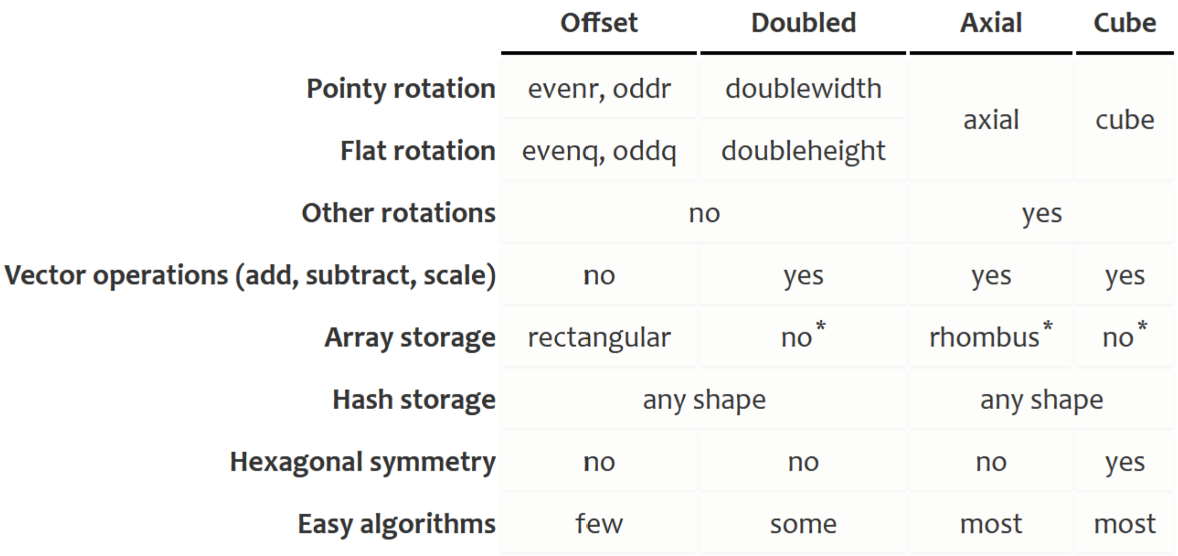
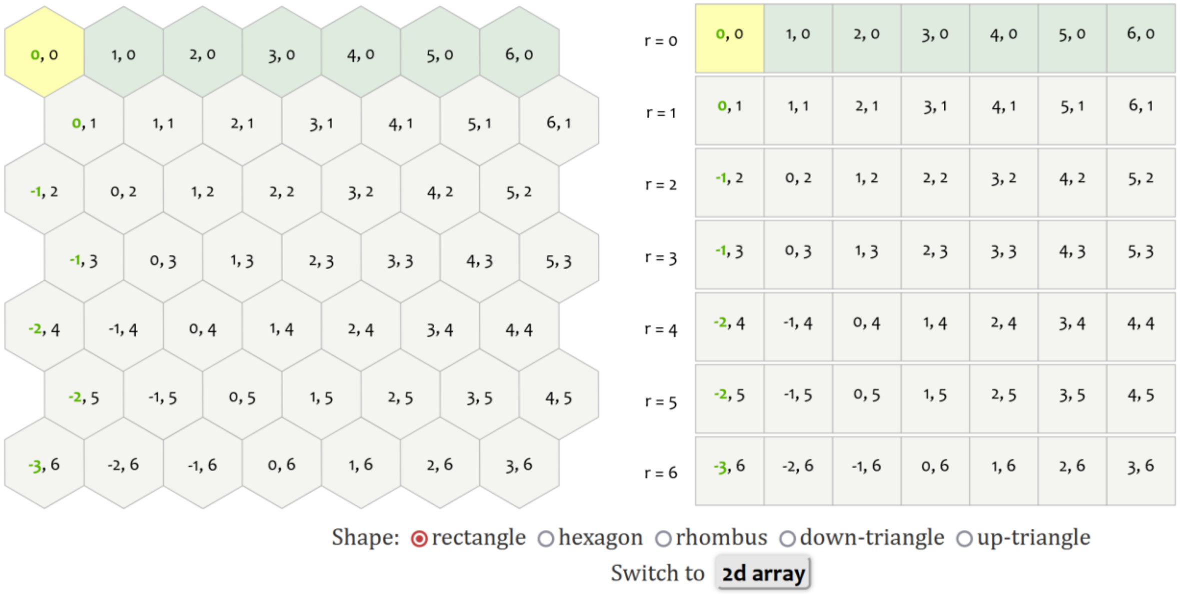


Figure 1. Offset coordinates defined with q and r.
Source: Adapted from [2].



	Offset	Doubled	Axial	Cube
Pointy rotation	evenr, oddr	doublewidth	axial	cube
Flat rotation	evenq, oddq	doubleheight		
Other rotations	no		yes	
Vector operations (add, subtract, scale)	no	yes	yes	yes
Array storage	rectangular	no*	rhombus*	no*
Hash storage	any shape		any shape	
Hexagonal symmetry	no	no	no	yes
Easy algorithms	few	some	most	most

Figure 2. Comparison of hexagonal coordinates with data structure storage.
Source: Adapted from [2].

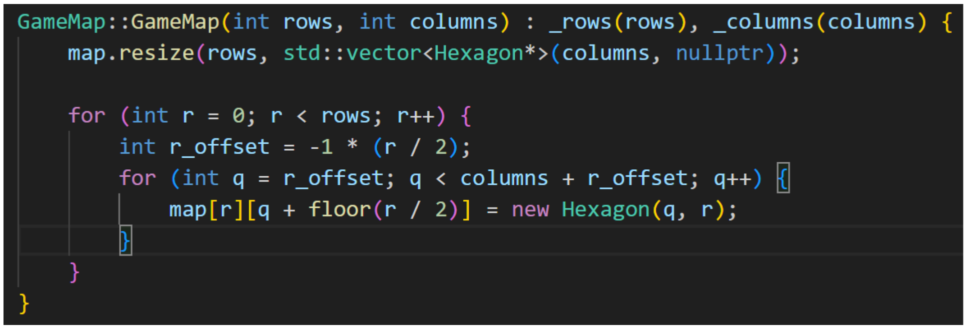


r = 0	0, 0	1, 0	2, 0	3, 0	4, 0	5, 0	6, 0
r = 1	0, 1	1, 1	2, 1	3, 1	4, 1	5, 1	6, 1
r = 2	-1, 2	0, 2	1, 2	2, 2	3, 2	4, 2	5, 2
r = 3	-1, 3	0, 3	1, 3	2, 3	3, 3	4, 3	5, 3
r = 4	-2, 4	-1, 4	0, 4	1, 4	2, 4	3, 4	4, 4
r = 5	-2, 5	-1, 5	0, 5	1, 5	2, 5	3, 5	4, 5
r = 6	-3, 6	-2, 6	-1, 6	0, 6	1, 6	2, 6	3, 6

Shape: ● rectangle ○ hexagon ○ rhombus ○ down-triangle ○ up-triangle

Figure 3. Offset coordinates translated into an array-of-array storage space.
Source: Adapted from [2].

As seen in Figure 2 above, few simple algorithms exist for offset coordinates, which comes at the tradeoff of simple data storage seen in Figure 3 above. The time complexity of pathing algorithms does not change in comparison to a standard square grid, but the complexity of writing such algorithms does increase, due to the need of converting between offset coordinates, vector coordinates, and row and column coordinates. Row = r and column = q + floor(r / 2), where q and r can be seen in Figure 1.



```
GameMap::GameMap(int rows, int columns) : _rows(rows), _columns(columns) {
    map.resize(rows, std::vector<Hexagon*>(columns, nullptr));

    for (int r = 0; r < rows; r++) {
        int r_offset = -1 * (r / 2);
        for (int q = r_offset; q < columns + r_offset; q++) {
            map[r][q + floor(r / 2)] = new Hexagon(q, r);
        }
    }
}
```

Figure 4. Project code snippet converting offset coordinates into row and column coordinates, and storing the Hexagon into map, a custom vector-of-vectors data structure.

Instead of checking up to four locations a character can move in a square-based grid, six locations will have to be checked. This means defining differential arrays for offset coordinates to check bordering hexes, as well as defining different boundaries of locations (i.e., the top left corner q-coordinate is 0, but the bottom left corner q-coordinate is –3 as seen in Figure 3, meaning only scanning for positive coordinates as a boundary is not possible).

3.0 Expected Usage Case

The Dungeon Crawler Board is expected to be used amongst all fans of dungeon crawlers and board games of all ages. Since it is based on Dungeon Crawlers, it is expected to be played by two to six players in a standard board game environment, either in homes or game shops. It will have a similar level of portability to a gaming console, with the ability to move it from location to location, but with the expectation it should be handled with care and caution. It will have to be plugged into a wall for power. Given the average D&D player has an above-average technical experience, using the off-board application to generate the map and port character information should be easy for users. It is important to note that the Dungeon Master is expected to have complete control of the application, while regular players will only be in contact with the board.

4.0 Design Constraints

4.1 Computational Constraints

The primary computational functions of the Dungeon Crawler include:

- Read in and store imported data from a computer via a USB cable
- Run the breadth first search algorithm to check for character movement options
- App for users to build maps and characters to send to the board
- Run field of view calculations and turn based combat during gameplay
- DMA and PWM timing control over a matrix of WS2812B LEDs

The Dungeon Crawler board needs to be able to store data for maps and character details from the app. In addition, it will have some maps pre-loaded into the system to allow for easy setup without the app. The microcontrollers that we are considering hold 512 KB of flash memory. We need to limit the size and number of files if we wish to avoid adding external memory storage.

For controlling the LEDs, a DMA and PWM will be used to vary the duty cycle to pass a 24-bit signal to each LED in each row. It would be ideal to minimize the frequency as much as possible to decrease power consumption. However, too low of a frequency could cause dim lights and flickering. The WS2812B LEDs typical frequency range is around 800 KHz, so we will need to keep our frequency around this range for optimal display [3].

4.2 Electronics Constraints

The main electronic components of the project include LEDs, Hall Effect sensors, an LCD, Keypad, and a USB port. The LEDs will use GPIO pins and utilize the DMA with PWM. Hall Effect sensors will be digital, and data will be read using the I2C IO expander. The LCD system will be operated using SPI and the keypad via a GPIO matrix. The loading of both map and character data will be done via a USB and stored in SD card via SPI.

The LED matrix system will be implemented using 16 GPIO pins instead of 32 given that the board is designed in a 16 by 16 grid. WS2812B needs to be supplied with 3.3V to 5V to be operational. There are several systems requiring GPIO pins. Thus, the I2C IO expander is used for the Hall Effect sensors to free more GPIO pins for use by the LEDs and keypad. The keypad and the Hall Effect sensors implementation will also need to be designed using pull down resistors.

4.3 Thermal/Power Constraints

Due to the constant power requirements required by the LEDs on the game board, the product must be plugged into wall power. The highest voltage requirement is from the LED strips which require a voltage of 5V. Each WS2812B LED requires 60 mA at full power [3]. Considering this as a maximum constraint the maximum current required by our game board would be 15.36 A. The board will reduce the usage of colors that require lots of amps, such as white, and reduce the strength of LEDs when displaying certain colors. The MCU should require a maximum of 240 mA [4]. To be safe, the system should expect a current consumption of 15-16 A at maximum power while operating.

Since this is not a handheld device, or something that will be in constant contact with the user, the temperature constraints have more to do with the specifications of the electronics. The microcontroller’s maximum operating temperature is 85 degrees centigrade [4]. Since the box will be made of wood, this temperature would not affect the packaging. As a result, an upper limit of 85 degrees centigrade is sufficient.

4.4 Mechanical Constraints

The Dungeon Crawler Board will be used as a standard game board, but also will hold several electronic components. Therefore, it needs to be small and light enough to fit into packaging for easy portability between locations, but large enough to comfortably hold the microcontroller and other components. The board and packaging should weigh no more than 10 pounds. The overall size of the board and packaging should mimic the size of regular tabletop game boards. We also need to consider the size of our gameplay area. The top surface should be large enough to hold a 16x16 hex grid that is adequately spaced for our hall effect sensors, as well as area for an LCD and a keypad. The material used on the top of the board will have to be opaque enough to allow light to shine through while also not showing the electronics underneath.

The board is intended for indoor use only, so the board itself will not be waterproof. In addition, there will be small character tokens included for gameplay. The packaging material should be durable and seal well to prevent any pieces from falling out and to protect the board from weather during transport. The packaging will also prevent dust from accumulating on the game board. We can model our packaging based on standard packaging for most tabletop games.

4.5 Economic Constraints

The economic constraint for our game board is based on the digital game board Teburu [5]. This system is priced at $200 to get the hardware for the game system as well as a game with it. The dungeon crawler board should be a lower price alternative to something like Teburu. To remain competitive in this market, the goal for this game board should be less than $200 dollars. Our most current cost estimates put the cost of production of our own board close to $250. However, it is important to note that at scale this could be decreased and that our board has functional differences than Teburu and is a unique product from it. Less sophisticated game boards such as an electronic chess board [6] cost as little as $60, so our board would be expected to cost more than this.

5.0 Sources Cited:

[1] A. Patel, "Implementation of Hex Grids," *www.redblobgames.com*, May 2015. https://www.redblobgames.com/grids/hexagons/implementation.html (accessed Jan. 19, 2024).

[2] A. Patel, "Hexagonal Grids," *www.redblobgames.com*, Oct. 2021. https://www.redblobgames.com/grids/hexagons/#coordinates (accessed Jan. 19, 2024).

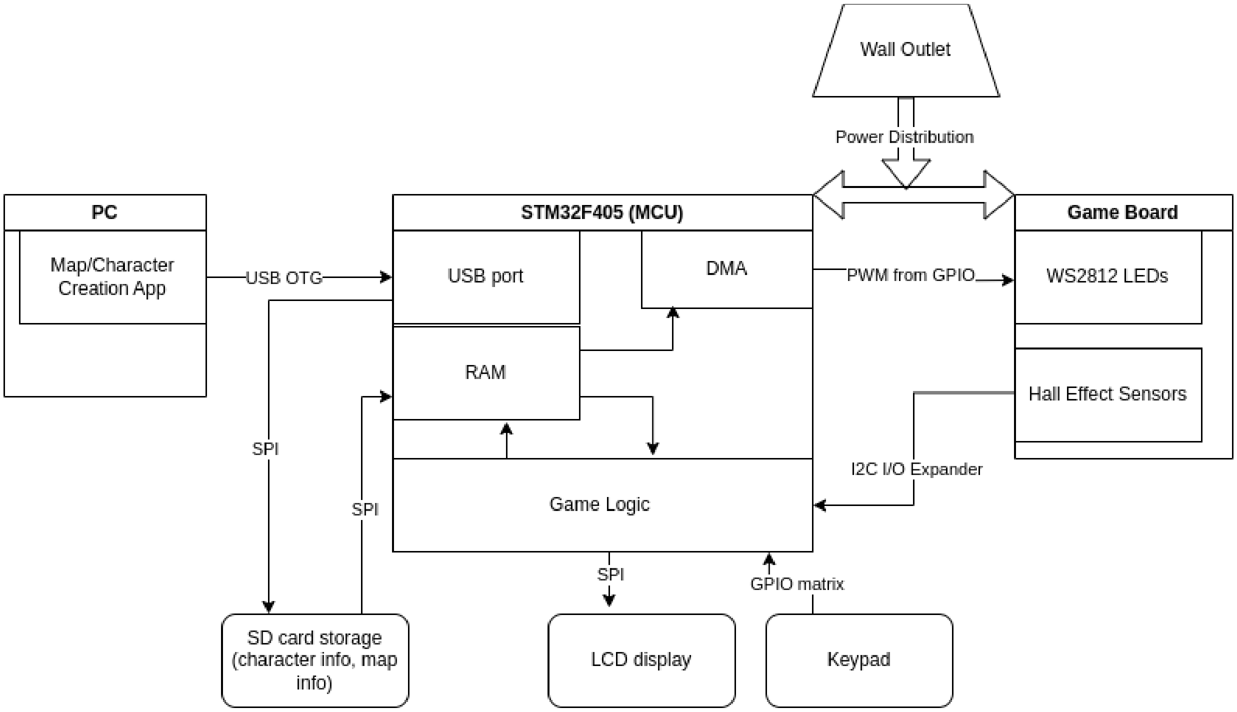
[3]WS2812B Datasheet by Adafruit Industries LLC - Digi-Key Electronics. (n.d.). https://www.digikey.com/htmldatasheets/production/3889527/0/0/1/ws2812b.html

[4] "STM32F405OE - STMicroelectronics," STMicroelectronics, 2024. https://www.st.com/en/microcontrollers-microprocessors/stm32f405oe.html#documentation (accessed Jan. 19, 2024).

[5] "Vampire: The Masquerade - Milan Uprising by Teburu," *gamefound.com*. https://gamefound.com/en/projects/xplored/vampire-the-masquerade-milan-uprising?refcode=-TjW8-PwaE-j_yiSunlEBQ#/section/rewards/reward-product-43462 (accessed Jan. 19, 2024).

[6] "Lexibook ChessMan® Elite Interactive electronic chess game +, 64 levels of difficulty, LEDs, family child board game, black / white, CG1300US," Walmart.com, 2024. https://www.walmart.com/ip/Lexibook-ChessMan-Elite-Interactive-electronic-chess-game-64-levels-of-difficulty-LEDs-family-child-board-game-black-white-CG1300US/1358595749?wmlspartner=wlpa&selectedSellerId=101035386 (accessed Jan. 19, 2024).

Appendix 1: Functional Block Diagram


Wall Outlet
Power Distribution
PC
Map/Character Creation App
USB OTG
STM32F405 (MCU)
USB port
DMA
RAM
Game Logic
PWM from GPIO
Game Board
WS2812 LEDs
Hall Effect Sensors
I2C I/O Expander
SPI
SPI
SPI
GPIO matrix
SD card storage (character info, map info)
LCD display
Keypad