18.9.2022

I am making Ludo Game! Finally! I wanted to do this project in quite some time for now, but didn`t really have much free time to actually start doing it (and feeling lazy sometimes ha!). So I can surely tell that I`m somewhat excited about this project. So, without further chatting, let`s start!

Yesterday I decided to jumpstart this. I was thinking about all techologies that I`ll use in this one. And I mapped everything out. More on this and initial plan will be written in the README.md file in the project repository.

I will first do the UI for this game (which shouldn`t really be that complicated) and then I will do the backend and DBMS and connect all the necessary dots between them.

So let`s start by actually doing something. In the meanwhile, project has been initialized localy on my PC and has the code repository on my GitHub. All further savings will be saved there.

1. Initialize the project

Project initialization starts with opening Visual Studio Code Editor (just VS Code in continuation). Creating one empty HTML, CSS and JS file.

HTML file is called app.html, because the script will contain all markup stuff that will be used in the game and it will be used as a central point of the game and it`s UI.

CSS file is called board.css, because it will contain styles relative to the game board.

JS file is called app.js, because it will be used for central gaming loop logic.

All files don`t have any special directory in which they should reside for now, they all have one relative path, which will change in the future as the project grows.

A screenshot of a computer screen

Description automatically generated with medium confidence

1. Game board

The game board I will use will have the standard Ludo Game board, without anything fancy added to it, really.

The board is having three parts that are repeating, which are: player start area (which contains 4 player rings), board squares (which some of them will have different features and styles) and central triangle thing (I cannot find any better name for this one, it`s just a design without any real funcionality, really).

Since those three parts are repeating through the board, I will code each one of them once and have them repeat in app.html, just with different styles for each player.

Shape, square

Description automatically generated

Icon, bubble chart

Description automatically generated A picture containing text, accessory

Description automatically generated A picture containing shoji

Description automatically generated

1. Game board analysis

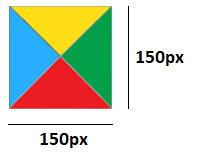
As shown on the image, there are exactly 3 main patterns (as said earlier) that are repeating.

We can see that all the board elements can be translated to the size of the one board square on which player moves. So, let`s consider that the one board square have a width of 50px and a height of the 50px (they are equal, because it`s a cube after all). We can see from the image that player start area has a width and the height of six of the board cubes, and the central triangle thing has a width and the height of three of the board cubes. So, player start area has the size of 300 x 300, central triangle thing has the size of the 150 x 150, and one group of the board squares has the size of 300 x 150 (or 150 x 300, depends on the fact are they rotated or not).

From that fact, we can conclude that our board size will be 750 x 750, after we add up all the sizes of these repeating board elements.

Shape, square

Description automatically generated

Chart, bar chart

Description automatically generated

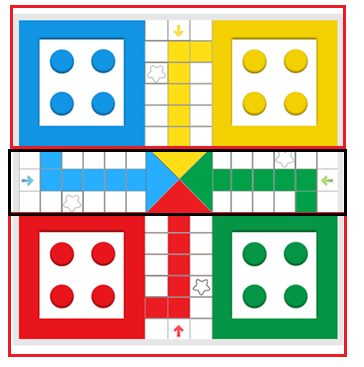
1. Creating game board

We have to open app.html and do all the board markup elements.

The boards will contain one major div container, which will serve as a wrapper. All of the HTML elements will be of type <div></div>.

The board will have 3 different rows, and the way the rows will be mapped is:

1. player start area – board squares – player start area
2. board squares (rotated on the Z axis) – center triangle thing – board squares (rotated on the Z axis)
3. player start area – board squares – player start area



Let`s set the player start area, and four player rings, in which our players will reside before they are called for the game:

Text

Description automatically generated

Player rings are put inside of the player start area, because we will style them relatively to that point.

Next, we are setting the a wrapper around around player start area:

Text

Description automatically generated

Next, let`s set the board squares, which will be just squares with the same size that are repeting in the pattern, and set their wrapper, which is called „board squares“ :

Graphical user interface, text

Description automatically generated

Next, we have to set the center triangle thing, and the following code will do the job:

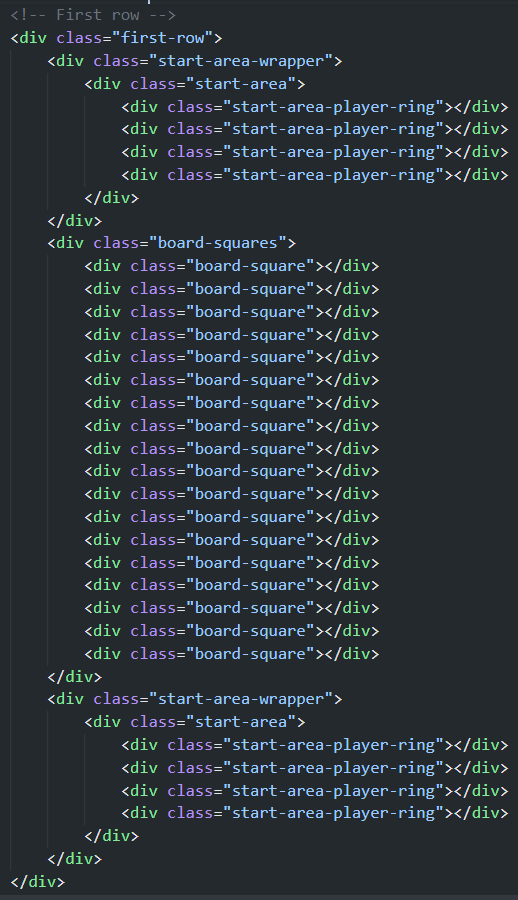


(that`s actually it for the markup of this one xD)

So now that we are having all of the three essential elements created, let`s create different rows

with them, as shown on the picture above:

First row:



Second row:

Graphical user interface, text

Description automatically generated

Third row:

Graphical user interface, text

Description automatically generated

Third row is essentially without any difference at all from the first row.

I like to add comments for better navigation in editor and better reading, but they are not essential.

After we completed markup, let`s style those elements next. We are going to open board.css script and write all the styles there.

When I create styling, first I like to see the repeating patterns. If there are any repeating patterns in the styles, I usually create standalone variable for that. For example, if I see that some color is repeating, I will extract that value in :root pseudoclass, and use it under that name in the file. With that practice, we are making sure that if we are changing the value of that variable in the future, we only have to change it`s value in one place, and not anywhere else in the code. That way the code is cleaner and more maintainable (as well as we`re saving some time to ourselves).

Going from that, I extracted all the color values used in this game in :root pseudoclass:

Text

Description automatically generated

In the future, of course this code will be changed, but for now it is what it is.

Next, I reseted all the values on the body element, so that I don`t have to deal with any sort of that later in the project:

Text

Description automatically generated

DISCLAIMER: all of the files in this entire project will reside in the SASS project files after I decide to create them in the future. For now, and always as I`m doing projects, I like to keep things simple at the beginning, and adding all the complexity later.

Using SASS is making the CSS so much more organized and clean. But more of that later in the process. For now, I am keeping things as simple as possible.

Next, let`s style the player start area:

Text

Description automatically generated

The start area is essentialy the white square on the board which holds four of the player rings, for each player of course. The width should be 200 x 200, because the colorized wrapper around is 100 x 100 of the size, so that both can add up to 300 x 300. The start area is displayed as a grid container, because it contains four rigidly and 2D-spreaded player rings. With grid-template-columns property, we can determine how we want player rings to be spreaded --> in this case we want to repeat 2 player rings in each row, and as we got 4 of them, this gird property will put two of the player rings in one row, and two of the player rings on the next row. Background color and border properties are self explanatory.

The start area wrapper has width and height of 300px, and it located under the start area. You could argue that I could have used padding with size of 100 x 100 on the start area, and that alone would cancel the start area wrapper, but I just prefer this way.

The start area player ring is 50 x 50 of the size, or the same as the board squares. They are essentialy board squares which are turned in the rings (or circles) with border-radius property set to 50%. Since they are the children of the start area, we need to give them some vertical centering (align-self: center) and some horizontal centering (justify-self: center), so that they can be dead-centered inside of their each grid container.

Next, let`s style the board squares:

Text

Description automatically generated

The board square is of size 47 x 47. Why not 50? Because it`s border and the border of board squares container are adding up 1px of the size to overall container size. The board squares container must have size of the 150 x 300 (or 300 x 150 if rotated). So with that in mind, we have to put size like that.

The board squares container is displayed as a grid, because it contains rigidly and 2D-spreaded like squares. It`s pattern is 6 columns, and each column contains 3 board squares. If rotated, the board container`s pattern is 3 columns, and each column contains 6 board squares. The gap property is inserted between each of the board square, to distinguish them one from another, and it`s value is 1px.

Background color is white and we can put that in the separate variable in :root pseudoclass.

Next, let`s style the center triangle thing:

Text

Description automatically generated

This one is pretty straitforward. This is because the borders of an HTML element form an angle where they join. We can make use of this to create a triangle by setting a transparent color for 3 of the borders and leaving only one visible. So, the bottom border will create a triangle pointing up, the left border will create a triangle pointing right and so on.

Now, let`s make some utility classes which will be used in some cases where some style properties are not the same, even though the HTML element is the same. And these cases for now are:

1. different start area wrapper
2. different start area player ring
3. different board square – color and special board square

Text

Description automatically generated Text

Description automatically generated

These utillity classes cover every color case of the board squares for each of the four players. But what about the four of the board squares with the arrow inside of it, which points to the board squares in which each of the player resides if they complete their game cycle? To attack this problem, I will use Font Awesome Icons UI library, which will provide me with already predefined classes, which I`ll put inside class property of each arrow. So let`s see this in action:

First, I have to put link to the external script from Font Awesome Icons livrary, so that I can use their icons in my project:

Graphical user interface, text, application

Description automatically generated

We paste this link in app.html file in the head section:

Text

Description automatically generated

When the library is imported, we can now use the their icons. So let`s type the „arrow“ in the search bar:

Graphical user interface, application

Description automatically generated

The icon choice will appear with the icons as the result, and we need to click on each icon that we want to show up the HTML code for them. We need top, bottom, right and left arrow. So let`s find them and copy their code:

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Next, we will put those elements inside four different board squares for each player and make utillity classes for each of the arrow and it`s board square which holds the arrow:

Text

Description automatically generated

Text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, text

Description automatically generated

I will not go full on details here, I just opened the browser and inspect with Grid Inspector in which board square I need to put the arrow. And here are the the utillity classes from board.css file:

Text

Description automatically generated Text

Description automatically generated

Text

Description automatically generated Text

Description automatically generated

And utillity clas for the square board which holds the actual arrow:

Text

Description automatically generated

And final step is to colorize the each of the players starting position and home board squares. So we`ll make a use of the <player>-board-square classes and put them in the corresponding board squares. After that, we get the full sized finished Ludo board:

Shape, square

Description automatically generated

Obviously, it`s not 100% perfect, and has some minor alignment issues, but that`s 99% to the final result. Those little gaps will be fixed in the later iterations of the project.