UNIVERSITY OF LONDON

BSc Computer Science

Module: CM2030 – Graphics Programming

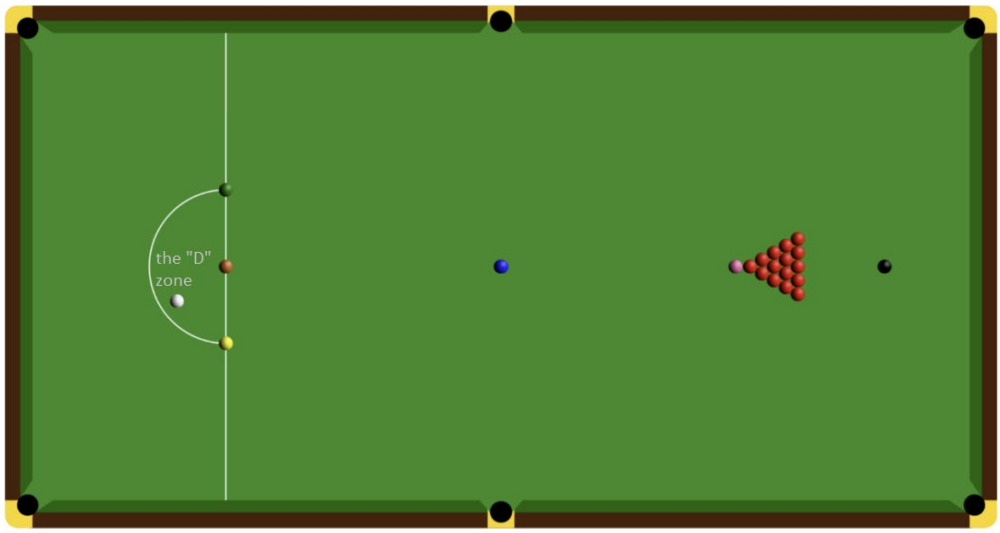
Introduction

During the course so far, we have developed a number of graphics applications using P5js. For this assignment you are tasked with developing a snooker app. You have to draw the snooker table with pockets, lines etc., add the balls and the cue. A standard full-size snooker table measures 12 ft × 6 ft. Here you choose the size of table in pixels, but you should maintain the ratio i.e., *table width = table length / 2.*

The ball size is approximately 2 inches in diameter so you could use the formula below

ball diameter = table width / 36.

The pocket size should be 1.5 times the size of the ball's diameter. The cue ball is only inserted in the "D" zone. Use the Snooker wiki (https://en.wikipedia.org/wiki/Snooker) for more detailed information so you can complete the table as seen below.



Task/steps

1. Define your variables for the table, balls and the cue. Store the balls in appropriate arrays
2. Draw the snooker table in the middle of the canvas using the correct colours, pockets and lines as described above
3. Draw the balls in three modes a) starting positions using keystroke “1”, b) random positions (reds only) using keystroke “2”, c) random positions (reds and coloured balls) using keystroke “3”. This excludes the cue ball
4. Give all balls the necessary restitution (for bouncing) and friction (for slowing down). You are expected to use matter.js
5. Draw the cue. Here you have a few options: use the mouse, keyboard or a combination of them to draw and manipulate the cue so you can hit the cue ball. Top marks will be awarded when using both mouse **and** key interaction. Adjust the speed of the cue i.e. you don’t want your cue to hit very hard and through the balls outside the table. Zero points if cue is acting as an elastic band.
6. The cue ball is inserted (in the “D” zone) using human interaction as above i.e. use the right combination of key/mouse etc. Zero points if cue ball already in place at start of the game without user interaction or outside the “D”.
7. Cushions should also have the necessary physics properties for bouncing (i.e. restitution) the balls (different from the ones the balls have). You are expected to use matter.js
8. When a red ball is in the pocket it should be removed from the array
9. Define a function for collision detection to prompt the user of the type of impact e.g. cue-red, cue-colour, cue-cushion. The function should work only for collisions of the cue ball

Gaming aspects

1. When the cue ball is in the pocket it should be given back to the player (run step 6 again). A potted cue ball can only be placed back inside the "D" zone
2. If a coloured ball is potted, it should be re-spotted on its designated spot on the table
3. If two consecutive coloured balls fall in the pocket then notify the user of this mistake

Coding style

1. Code presentation: use appropriate syntax, comments, consistent indentation and redundant code
2. Code competency: use of object orientation, code reusability, use of functions, variables global vs local

Extension

Since this is a creative module, we would like to give more marks for implementing further ideas. Also, you should write some words about it in the commentary (see next paragraph). Please note that we will award marks for the uniqueness of your extension and how technically challenging it appears to have been. Please note that we will not award any marks if you decide to develop the snooker app with scoring as this is not a unique extension. Try something novel or innovative that has not been seen in snooker gaming before. The extension is worth 20% of your mark.

Commentary

Explain the app design: e.g. why you used a mouse-based only cue function - how does it work? The report quality (i.e. language) will also be assessed. Also discuss your extension and why it is a unique idea. Be precise and deliver the information within 500 words. Include this in your main .js file.

Video demo

Demonstrate your work using a video. Verbally go through all functionalities and your development decisions. More specifically your demo should include:

- the 3 modes as discussed in step 3 using 1,2,3 keystrokes
- scenario: if cue ball is potted is then returned to user
- scenario: if a coloured ball is potted it should be returned to its designated spot
- scenario: error prompts when two coloured balls are potted
- scenario: collision detection of cue ball: cue-red, cue-colour, cue-cushion

Also talk about your extension. Always have the console window open so any behaviour is recorded. Use OBS Studio or similar software and not your mobile phone and make sure you talk through your app during the demo. We will give zero points to this question if console is not shown or you are using your mobile phone to record the demo. The video should be up to 5 minutes long.

General information

You should complete this work using the libs shown in the module. Do not use external code for this assignment. All your code and commentary will be checked for plagiarism/AI generation.

Submission requirements

1. Compress all your code in .ZIP format and upload it in the first prompt
2. Upload the video demo in .mp4 format in the second prompt
3. Use alternative video submission link to upload the video demo to YouTube or similar, then submit the URL only in the third prompt

4. Merge all your .js code in a single file, then upload it in the last prompt. Use the JavaScript Bundler Tool (see previous learning item) to merge all your .js files into a single .txt file. Exclude any libraries you used such as p5.js, matter.js etc. **This is a submission requirement as we need this to run your code.**

Rubric

[3 points]: Program runs and without errors? (check console)

[3 points]: Program is usable e.g. easy to use without confusing behaviour

[3 points]: Pool table drawn (1 point for a simple one, 2 for an average, 3 for a detailed one)

[2 points]: Bouncing implemented for the cushions (1 point for matter.js, 1 point for realistic collision)

[2 points]: Bouncing implemented for the balls (1 point for matter.js, 1 point for realistic collision)

[2 points]: Physics implemented for the cue (1 point for matter.js, 1 point for realistic collision)

[1 point]: Balls have the necessary friction

[1 point]: Red balls are removed from the array when in pockets

[2 points]: Cue drawn on screen using mouse and/or key interaction (discuss your choice in report)

[2 points]: Cue manipulated using mouse/key interaction (discuss your choice in report)

[1 point]: Cue has the necessary speed limit

[2 points]: Table starts in three modes. Use key interaction (i.e. keystrokes 1, 2, 3) to load each mode

[2 points]: 1st mode all balls in place as in starting position. Cue ball (white) excluded

[3 points]: 2nd mode use a random algorithm of your choice to get all balls excluding the cue ball (white) on the table. Discuss the choice of the random algorithm in your report

[2 points]: 3rd mode. Adjust the previous step to only randomly allocate the red balls leaving the coloured balls intact. Cue ball is again excluded

[2 points]: Cue ball (white) to be inserted using the necessary constrains and by using mouse and/or key interaction (not random init)

[1 point]: If cue ball falls in the pocket it should be given back to the player and the previous requirement should be executed again

[1 point]: If a coloured balls gets in the pocket it should be returned to its original location

[1 point]: An error prompt is shown when two coloured balls are inserted into the pocket

[3 points]: Collision detection of cue ball

[4 points]: Code presentation: indentation, white space, comments, variable naming

[4 points]: Code competency: code reusability (some functionality is repeated – did student use functions/OO to organise the code better?)

[5 points]: Commentary included?

[4 points]: Video included?

[14 points]: Has learner implemented any unique ideas for further development?