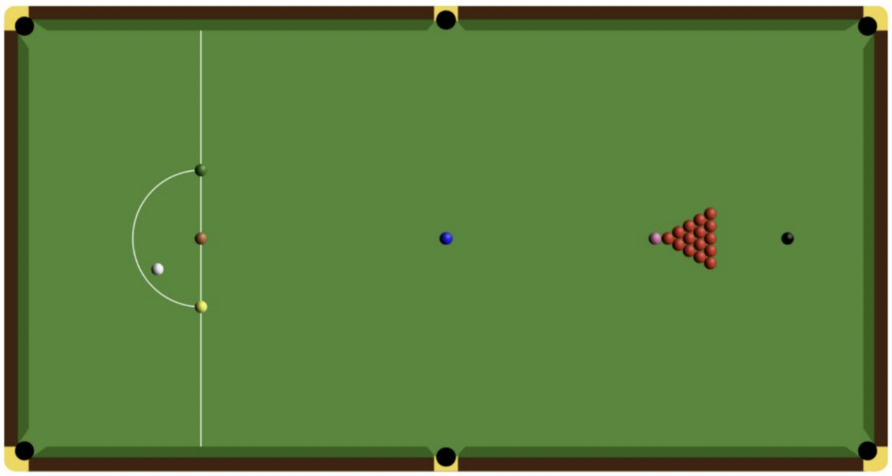
CM2030 week 10 - Mid-Term Assignment

Mid-Term Submission 🎲



Discussion of the Rubric 🎲

- ⭐ Make sure to fulfill the basic rubric requirements — we'll go through this shortly together ⭐
- Use `matter.js` for the physics (your physics model of the snooker table, cue, cushions, balls, etc)
- `matter.js` can also render your models to screen, but we **recommend** drawing the models on screen using `p5`.

Rubric Points

You can see these in the mid-term PDF downloadable in Week 10 on Coursera, and explains exactly how to achieve the marks.

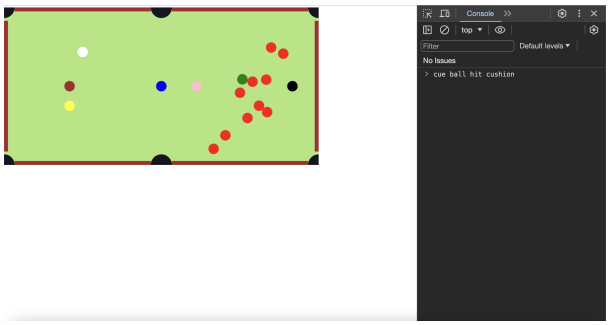
- ⭐ Error free - make sure your console is open in your video
    - **[3 points]:** Program runs and without errors? (check console)

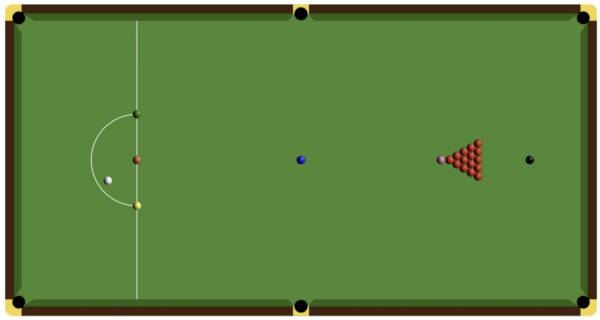
- ⭐ Make sure it's easy to use, not complicated, confusing, or cumbersome.
    - **[3 points]:** Program is usable e.g. easy to use without confusing behaviour

---

- ⭐ More marks available for more detailed snooker tables

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





- ⭐ Physics (matter.js)
  - **[2 points]:** Bouncing implemented for the cushions
  - **[2 points]:** Bouncing implemented for the balls
  - **[2 points]:** Physics implemented for the cue
  - **[1 point]:** Cue has the necessary speed limit
  - **[1 point]:** Balls have the necessary friction

  - For example, using `{ friction: 0.1, restitution: 0.1 }` — play around with the values so that it feels right.

- ⭐ Cue visuals
  - **[2 points]:** Cue drawn using mouse and/or key interaction (discuss your choice in report)
  - For example, the cue is drawn on screen and visually renders depending on mouse position and/or key presses

---

- ⭐ Cue behaviour
  - **[2 points]:** Cue manipulated using mouse/key interaction (discuss your choice in report)
  - For example, the cue properties might change depending on mouse position and/or key presses.

- ⭐ Remove red balls from the array
  - **[1 points]:** Red balls are removed from the array when in pockets
  - For example, splice your array or perhaps use ES6 features where appropriate if you want.

- ⭐ Three starting modes
  - **[2 points]:** Table starts in three modes. Use key interaction to load each mode, e.g. press 1️⃣, 2️⃣ or 3️⃣ or similar.
  - **[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

- For example
  - **none of the above involve the white** (cue) ball.
  - use some Perlin noise, not random noise.

- ⭐ Cue Ball ⚪ Behaviour
  - **[2 points]:** Cue ball (white) to be inserted using the necessary constrains [inside the D] and by using mouse and/or key interaction (not random initialisation)

- ⭐ Cue Ball Pocketed ⚪
  - **[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
  - For example, place inside D only.

- ⭐ Colored Ball Pocketed 🟡🟤🟢🔵🟣⚫
  - **[1 point]:** If a coloured ball gets in the pocket it should be returned to its original location

---

- ⭐ Two Colored Balls Pocketed, e.g. 🟢🟢 pocketed one after the other
  - **[1 point]:** An error prompt is shown when two coloured balls are inserted into the pocket — show this in your video!
  - For example, show a prompt on screen.

- ⭐ Cue Ball Collisions
  - **[3 points]:** Collision detection of cue ball
  - For example, ⚪ cue ball collides with cushions, 🔴 red balls, 🟡🟤🟢🔵🟣⚫ colored balls, etc.

- ⭐ Code related marks:
  - **[4 points]:** Code presentation: indentation, horizontal & vertical white space, comments, variable naming
  - **[4 points]:** Code competency: code reusability (if code is repeated then make a helper function, use OOP if appropriate, etc)
  - **[5 points]:** Commentary included? — put your `/* */` comment report at the top of your `sketch.js` file.

- For example,
  - indentiation
  - horizontal spacing
  - vertical spacing
  - variable naming & use, e.g. `camelCase` for variables (counters, positions, etc), `SNAKE_CASE` for constants, `PascalCase` for classes if you use classes — often descriptive variable naming can help avoid the need for commenting.
  - avoid magic numbers — use `A_CONSTANT_INSTEAD`
  - commenting - where appropriate, don't include obvious comments like `// counter is incremented` — this should be obvious from the code
  - style is not the main thing, consistency is the key
  - re-use code

```
function setup ()
{
  createCanvas( 400,    400 );
  background(220);
}



function draw() {


  noStroke();
      let spot_x_pos = random(width);
     let color = 255;

  fill (color,random(255),   random(255));

  ellipse(spot_x_pos, random(height), 10, 10);
}




// my report in 5 words
```

```
/*
        my report here... (check the word cou
*/

const CANVAS_WIDTH = 400;
const CANVAS_HEIGHT = 400;
const BACKGROUND_COLOR = 220;
const MAX_COLOR_VALUE = 255;
const CIRCLE_RADIUS = 10;

function setup() {
  createCanvas(CANVAS_WIDTH, CANVAS_HEIGHT);
  background(BACKGROUND_COLOR);
}

function draw() {
  noStroke();

  let x = random(width);
  let y = random(height);

  fill(random(MAX_COLOR_VALUE),
           random(MAX_COLOR_VALUE),
             random(MAX_COLOR_VALUE));

  ellipse(x, y, CIRCLE_RADIUS, CIRCLE_RADIUS);
}
```

- ⭐ Video:
  - **[4 points]:** Video included? 📽️
  - For example,
    - make sure the console is visible (Dev Tools / Developer Console
    - Discuss verbally — zero marks for AI or no voice discussion
    - Explain what you have done and why

    - Talk about your extension
    - Please don't use a mobile phone to record, use a desktop screen recorder.

- ⭐ Further development
  - **[14 points]:** Has learner implemented any unique ideas for further development?
  - This is the most difficult for us to explain because it has to be original and we can't give you direction on what to implement.
  - This must be your own original idea.
  - Some tips:
    - Zero or limited marks for standard things such as:
      - Scores
      - Sound effects
      - Gameplay
    - Add Something Surprising / Unique 💥
      - We are looking for something original from you, so we can't give ideas.
      - Think outside of the box
    - Take Inspiration From...
      - Material learnt so far, think back on previous topics, such as generative textures, noise, fractals
    - Advanced Code Features
      - Use ES6 functionality
      - Use Objct Oriented Programming (see Pedram's webinar link)

Marks Distribution

- 80% of marks available for core rubric
- 20% for further development

Plagiarism ⚠

- Do not copy other people's work, from fellow students, online work, or online repositories.
- Work submitted must be your own original work.
- Libraries may be used, but these must not be to replace the rubric itself.
- We use plagiarism checking software as part of the marking.

Resources 📚

- Check out other mid-term webinars in the ATLG (Pedrams — https://bit.ly/4gEGJPd, Erhan to post details shortly)
- You might find Perdram's Perlin noise tutorial helpful, possibly for randomising balls etc
  https://editor.p5js.org/pedbad/collections/3TMYEctTM
  https://docs.google.com/presentation/d/1SYtEAc5gXZnZak1faxW3H2AtYUeYbCMQfqrWtgGVvr4/edit#slide=id.gc6f9199
- Pedram's `matter.js` tutorials https://editor.p5js.org/pedbad/collections/JQL_O63kt
- `matter.js` https://brm.io/matter-js/

- You may wish to use some p5 libraries https://p5js.org/libraries/ (not a requirement)
- JavaScript style guides https://blog.codacy.com/4-popular-javascript-style-guides

Thanks, any Questions ❓

Extra: Textures Surfaces (Preview)

Load in `preload()`

- Load your images in the `preload()` function
- So that all images are loaded before `setup()` and `draw()` are called.

```
var img;

function preload(){
    img = loadImage('assets/rocks.jpg');
}
```

```
12 function draw() {
13      background(0);
14      texture(img);
15      rotateY(frameCount);
16      box(300);
17  }
18
```

Use `noStroke()` to remove any stroke lines, such as the one showing along the edges on the textured cube.