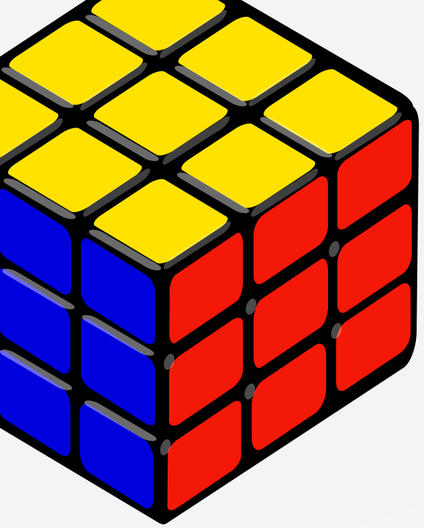
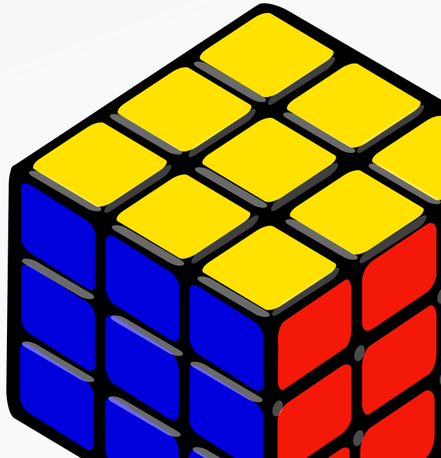
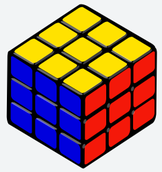


RUBICK'S CUBE
PROJECT

ANTONI, MARLON E RICARDO

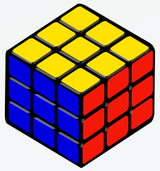


CÓDIGO



Estrutura do cubo

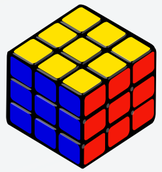
```
const faceU = ["red","red","red","red","red","red","red","red","red"];
const faceL = ["blue","blue","blue","blue","blue","blue","blue","blue","blue"];
const faceF = ["white","white","white","white","white","white","white","white","white"];
const faceR = ["green","green","green","green","green","green","green","green","green"];
const faceB = ["yellow","yellow","yellow","yellow","yellow","yellow","yellow","yellow","yellow"];
const faceD = ["orange","orange","orange","orange","orange","orange","orange","orange","orange"];
```



Método de chamada do movimento

```
function bMove(){
    movement(faceB);
    sideMovement("B", faceU, faceL, faceD, faceR);
    load();
}

function dMove(){
    movement(faceD);
    sideMovement("D", faceF, faceR, faceB, faceL);
    load();
}

function invertedFMove(){
    invertedMovement(faceF);
    invertedSideMovement("F", faceU, faceR, faceD, faceL);
    load();
}

function invertedUMove(){
    invertedMovement(faceU);
    invertedSideMovement("U", faceB, faceR, faceF, faceL);
    load();
}
```

CÓDIGO

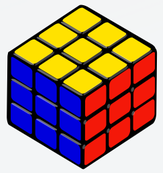


Movimento frontal e lateral

```
function movement(face){
    let save = face[1];
    face[1] = face[3];
    face[3] = face[7];
    face[7] = face[5];
    face[5] = save;

    save = face[0];
    face[0] = face[6];
    face[6] = face[8];
    face[8] = face[2];
    face[2] = save;
}
```

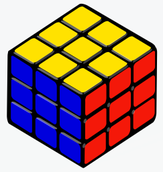
```
function sideMovement(movement, face1, face2, face3, face4){
    let save = face1.slice();
    if (movement === "F") {
        face1[6] = face4[8];
        face1[7] = face4[5];
        face1[8] = face4[2];

        face4[2] = face3[0];
        face4[5] = face3[1];
        face4[8] = face3[2];

        face3[0] = face2[6];
        face3[1] = face2[3];
        face3[2] = face2[0];

        face2[0] = save[6];
        face2[3] = save[7];
        face2[6] = save[8];
    } else if (movement === "R") {
        face1[8] = face4[8];
        face1[5] = face4[5];
        face1[2] = face4[2];

        face4[2] = face3[2];
        face4[5] = face3[5];
        face4[8] = face3[8];

        face3[2] = face2[6];
        face3[5] = face2[3];
        face3[8] = face2[0];

        face2[0] = save[8];
        face2[3] = save[5];
        face2[6] = save[2];
    } else if (movement === "B") {
        face1[2] = face4[8];
        face1[1] = face4[5];
        face1[0] = face4[2];
```

CÓDIGO



Shuffle

```
function random(){
    let move1 = allMoves[parseInt(Math.random() * 12)];
    let move2 = allMoves[parseInt(Math.random() * 12)];
    let move3 = allMoves[parseInt(Math.random() * 12)];
    let move4 = allMoves[parseInt(Math.random() * 12)];
    let move5 = allMoves[parseInt(Math.random() * 12)];
    let move6 = allMoves[parseInt(Math.random() * 12)];
    move(move1);
    move(move2);
    move(move3);
    move(move4);
    move(move5);
    move(move6);
}
```



Método para movimentar o cubo chamado pelo shuffle

```
function move(movement){
    switch (movement){
        case 'U':
            uMove();
            break;
        case 'L':
            lMove();
            break;
        case 'F':
            fMove();
            break;
        case 'R':
            rMove();
            break;
        case 'B':
            bMove();
            break;
        case 'D':
            dMove();
            break;
        case 'U\'':
            invertedUMove();
            break;
        case 'L\'':
            invertedLMove();
            break;
        case 'F\'':
            invertedFMove();
            break;
        case 'R\'':
            invertedRMove();
            break;
        case 'B\'':
            invertedBMove();
            break;
        case 'D\'':
            invertedDMove();
            break;
    }
}
```

OBJETIVOS

nº 1

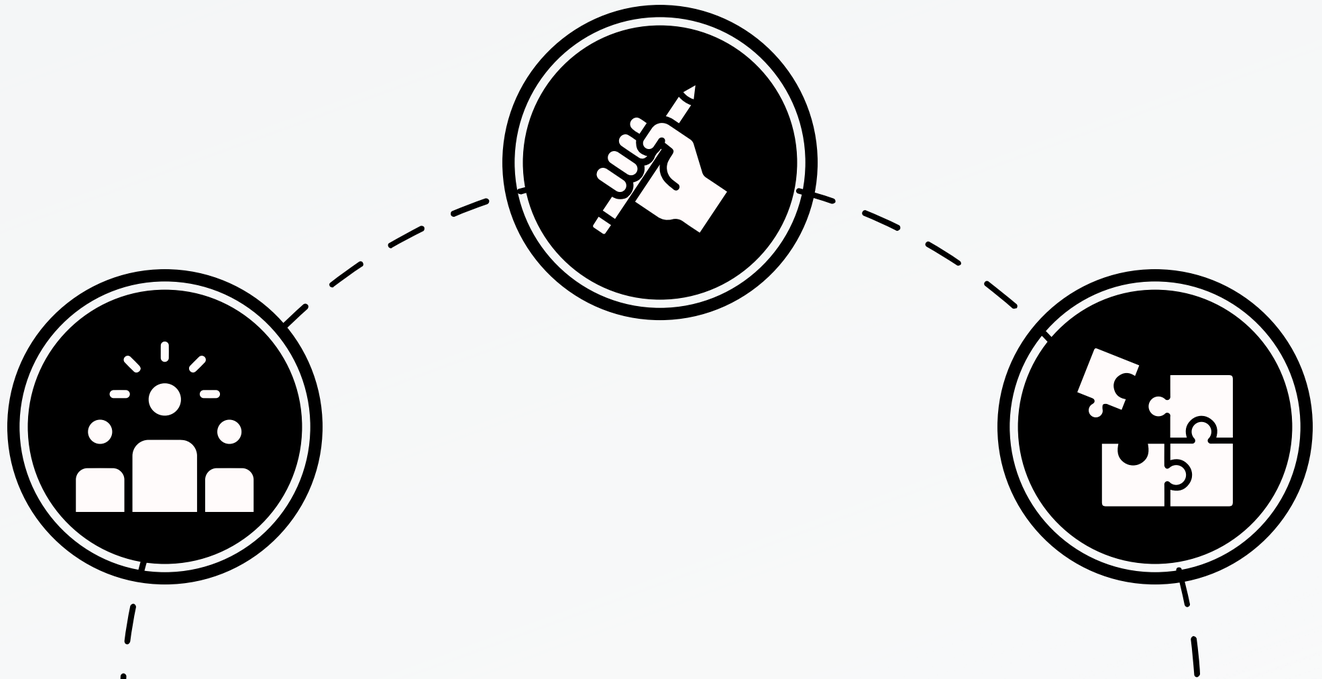
Entender o funcionamento do cubo e como resolve-lô

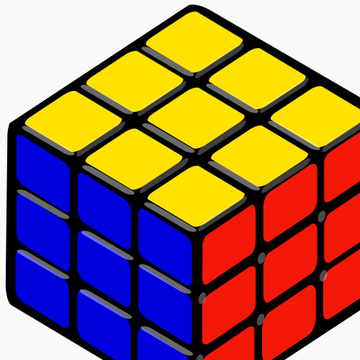
nº 2

Implementar em HTML e JS o funcionamento do cubo, tal como movimentos,cores,estrutura e randomização das posições

nº 3

Implementar o algoritmo Kociemba para a resolução do cubo, sem a utilização de bibliotecas e/ou frameworks





DIFICULDADES ENCONTRADAS

- Movimento anti-horário
- Dificuldade de implementar o algoritmo de resolução puro
- Exemplos encontrados na internet utilizam bibliotecas para resolver o cubo.

- Erros aleatórios no desenvolvimento, principalmente com node.js
- Erros persistindo mesmo mudando versões das aplicações e tentando corrigir o código

COMO SOLUCIONAMOS

- Estudando a estrutura do cubo
- Mudando de computador (3x)
- Utilizando biblioteca kociemba para resolução do cubo

OBRIGADO