

# Escape Room

Three.js – 2º Projeto

Introdução à Computação Gráfica - 2020/2021

Joaquim Madeira ([jmadeira@ua.pt](mailto:jmadeira@ua.pt))

93310 - Gonçalo Pereira

# Índice

<b>Introdução</b>	<b>3</b>
<b>Correr o Programa</b>	<b>3</b>
<b>Modelos e Texturas</b>	<b>4</b>
<b>Movimentação</b>	<b>4</b>
<b>Câmara</b>	<b>5</b>
<b>Interação</b>	<b>5</b>
<b>Ambientação</b>	<b>7</b>
<b>Informações relevantes</b>	<b>9</b>
<b>Conclusão</b>	<b>11</b>
<b>Repositório Git</b>	<b>11</b>

# Introdução

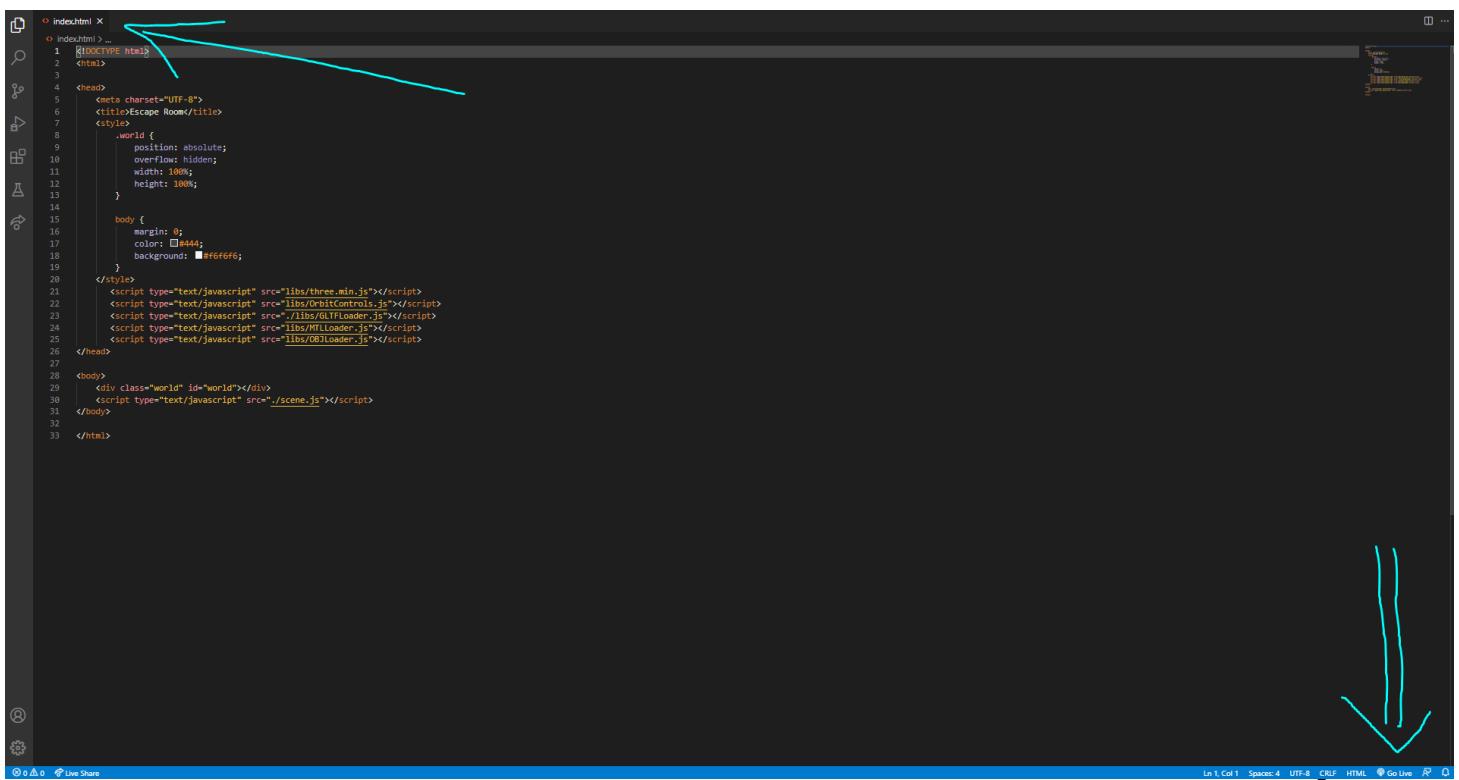
Para o segundo projeto de Introdução à Computação Gráfica decidi criar uma escape room, usando a Showroom que apresentei no primeiro projeto como base. Tal como numa escape room normal, o utilizador terá que resolver um mistério para poder escapar da mesma e ganhar o jogo.

Para resolver o mistério são dadas pistas ao jogador, que tem de as decifrar para conseguir a chave que abre a porta e finaliza o jogo.

Neste relatório vou falar dos aspetos principais da aplicação e das opções que escolhi implementar, assim como os desafios que enfrentei.

# Correr o Programa

Para executar o programa é necessário abrir a pasta num editor de código (ex: Visual Studio Code) para as pastas necessárias carregarem, depois abrir o projeto como um todo, e não apenas um ficheiro.



```
index.html
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Escape Room</title>
    <style>
        .world {
            position: absolute;
            overflow: hidden;
            width: 1000px;
            height: 1000px;
        }
        body {
            margin: 0;
            color: #444;
            background: #f6f6f6;
        }
    </style>
    <script type="text/javascript" src="libs/three.min.js"></script>
    <script type="text/javascript" src="libs/OrbitControls.js"></script>
    <script type="text/javascript" src="./libs/GLTFLoader.js"></script>
    <script type="text/javascript" src="./libs/MTLLoader.js"></script>
    <script type="text/javascript" src="libs/MTLLoader.js"></script>
</head>
<body>
    <div class="world" id="world"></div>
    <script type="text/javascript" src="./scene.js"></script>
</body>
</html>
```

# Modelos e Texturas

Como qualquer escape room, decidi conceder à sala um ambiente tenebroso, com pouca iluminação (apenas o suficiente para ver bem o jogo) e materiais antigos, como a pintura das paredes, objetos na sala e as próprias luzes.

Os modelos encontram-se na pasta ObjectImport, as texturas estão na pasta Texturas.

Importei os modelos do site Sketchfab e as texturas são imagens retiradas da internet.

# Movimentação

O movimento da personagem é controlado com as teclas 'WASD' e o movimento da câmara é efetuado com as setas do teclado.

A showroom é feita em primeira pessoa, e os objetos da sala têm tamanho proporcional ao personagem para criar imersão e realismo.

Existem também hitboxes que não permitem ao personagem atravessar os modelos ou as paredes.

As hitboxes são aplicadas através de uma caixa, cameraBox, que acompanha a câmera do jogador e verifica as colisões com os restantes modelos e paredes. Resumidamente, quando a cameraBox atinge um objeto a posição do jogador volta à anterior e aparece uma mensagem na consola.

Implementei também controlos que permitem ao jogador agachar-se (Espaço) e correr (Shift).

# Câmara

Como já tinha dito, a câmera é controlada com as setas do teclado.

O utilizador pode olhar à sua volta, para cima e para baixo, havendo limite na rotação vertical, de modo a não dar a volta.

Implementei um controlo na câmara, que permite ao jogador fazer zoom (E), que é importante para ler as dicas dadas pelo jogo.

# Interação

Sendo um jogo de resolução de enigmas, o utilizador interage com a aplicação de várias maneiras.

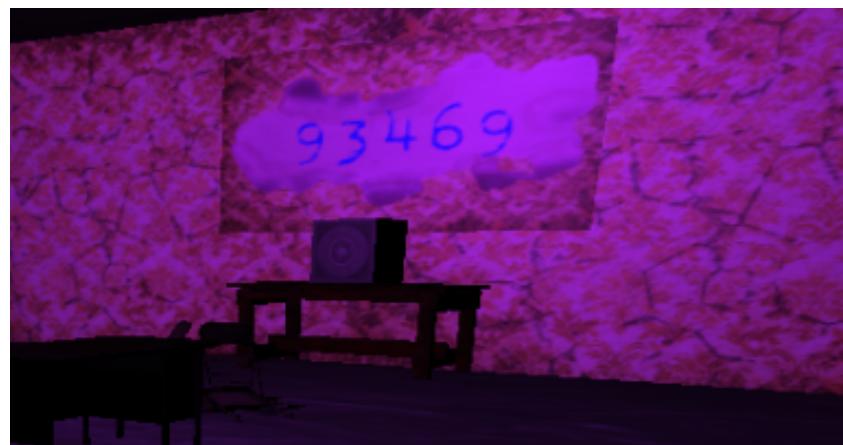
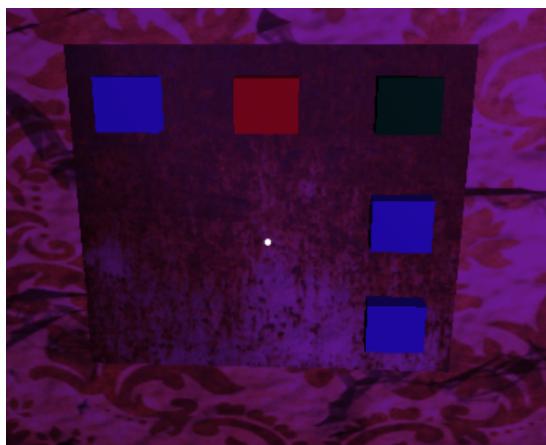


A pinta branca à frente do utilizador mostra para onde este está a olhar.

A primeira interação direta que o utilizador deve ter com o jogo acontece nesta placa com botões que mudam de cor ao serem clicados (R para interagir) pelo jogador.

Cada botão tem três cores possíveis, vermelho, verde e azul (RGB), e o utilizador deve encontrar uma mensagem na sala que diz a sequência correta das cores.

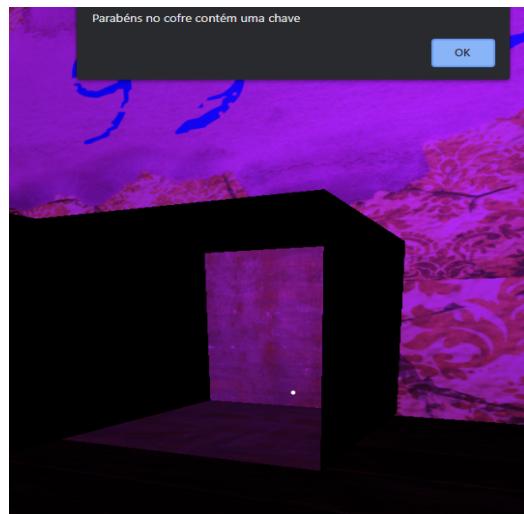
Depois de descobrir a ordem certa e clicar nos botões até estes a representarem, a luz da sala muda, revelando uma mensagem escondida na parede oposta.



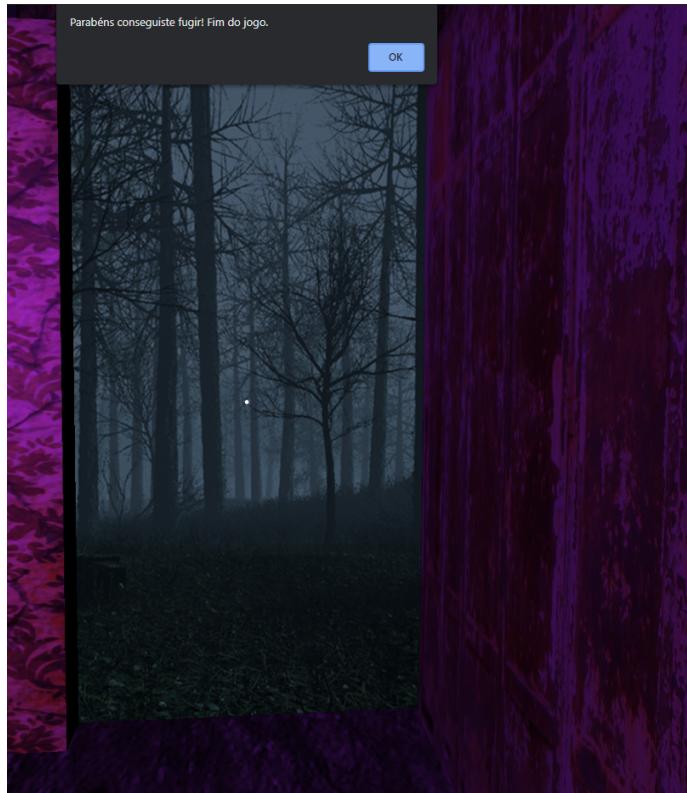
Estes números são a sequência do cofre, então quando o utilizador se aproxima e tenta interagir com o cofre, é-lhe apresentado uma caixa de texto onde deve inserir o código certo.



Se o jogador colocar o código certo, abre o cofre e é revelada a chave da porta principal.



Se já tiver a chave, o utilizador tem que chegar perto da porta e interagir com ela. A porta abre e aparece uma nova mensagem a congratular o jogador por ter conseguido fugir.



## Ambientação

Para criar a sensação de imersão durante o jogo existe um som de fundo para criar suspense, simulando uma floresta, que é onde a escape room está localizada.

Além do som, visualmente é tão ou mais importante o utilizador sentir que está no jogo. Optei por controlos em primeira pessoa, pois é o mais realista possível e, como tinha mencionado, os modelos são proporcionais à altura do personagem. A luz da sala cria um ambiente misterioso, especialmente quando muda para néon, com o objetivo de surpreender o utilizador.

As texturas da sala, papéis de jornal antigos e modelos também foram escolhidos com o objetivo de criar esse ambiente realista.

Escape room vista de dois cantos da sala.



# Informações relevantes

Os controlos da aplicação são apresentados no início do jogo e sempre que o jogador premir a tecla zero “0”. Estes consistem em:

- 'WASD' para controlar o movimento do personagem;
- Setas para direcionar com a câmara;
- 'E' para dar zoom;
- 'Space' para agachar;
- 'SHIFT' para correr;
- 'R' para interagir com objetos.

Para implementar as interações utilizei a classe Raycaster (<https://threejs.org/docs/#api/en/core/Raycaster>).

```
if (keyboard[82]) { // R interagir
    console.log("interagir")
    pos.x = 0
    pos.y = 0

    const intersects = raycaster.intersectObjects(scene.children);
```

Funções que controlam as colisões da personagem com os modelos e paredes.

```
function checkCollisions() {
    let checkBox = new THREE.Box3().setFromObject(cameraBox)
    for (let i = 0; i < wallsArray.length; i++) {
        let obstBox = new THREE.Box3().setFromObject(wallsArray[i])
        let colision = checkBox.intersectsBox(obstBox)
        if (colision) {
            console.log("HIT")
            return true
        }
    }
    return false
}

function checkCollisionDoorWall() {
    let checkBox = new THREE.Box3().setFromObject(cameraBox)
    for (let i = 0; i < doorWallsArray.length; i++) {
        let obstBox = new THREE.Box3().setFromObject(doorWallsArray[i])
        let colision = checkBox.intersectsBox(obstBox)
        if (colision) {
            console.log("Hit")
            return true
        }
    }
    return false
}

function checkCollisionsObjects() {
    let checkBox = new THREE.Box3().setFromObject(cameraBox)
    for (let i = 0; i < objectsArray.length; i++) {
        let obstBox = new THREE.Box3().setFromObject(objectsArray[i])
        let colision = checkBox.intersectsBox(obstBox)
        if (colision) {
            console.log("Hit")
            return true
        }
    }
    return false
}
```

Exemplo de aplicação do movimento, neste caso para a frente (W).

```
if (keyboard[87]) { // W key
    camera.position.x -= Math.sin(camera.rotation.y) * player.speed;
    camera.position.z += -Math.cos(camera.rotation.y) * player.speed;

    cameraBox.position.x -= Math.sin(camera.rotation.y) * player.speed;
    cameraBox.position.z += -Math.cos(camera.rotation.y) * player.speed;

    if (checkCollisions()) {
        cameraBox.position.x = oldPos.x
        cameraBox.position.z = oldPos.z

        camera.position.x = oldPos.x
        camera.position.z = oldPos.z
    }

    if (checkCollisionDoorWall()) {
        cameraBox.position.x = oldPos.x
        cameraBox.position.z = oldPos.z

        camera.position.x = oldPos.x
        camera.position.z = oldPos.z
    }

    if (checkCollisionsObjects()) {
        cameraBox.position.x = oldPos.x
        cameraBox.position.z = oldPos.z

        camera.position.x = oldPos.x
        camera.position.z = oldPos.z
    }
}
```

Para a realização deste projeto aproveitei partes do meu primeiro projeto da cadeira, a Showroom, o que me permitiu desenvolver algumas implementações complexas que não teria tempo sem a base da sala.

A feature onde senti mais dificuldades em implementá-la foi sem dúvida desenvolver as lógicas de resolução dos enigmas de forma eficiente, para não tornar o jogo lento. Ainda assim, devido à sua complexidade, o programa demora a carregar as texturas, logo é melhor esperar uns segundos depois de abrir o jogo para começar a mexer a personagem.

# Conclusão

Com este trabalho decidi criar um jogo interessante e, de certo modo arriscado, pois não sabia se teria tempo para implementar tudo o que eu queria. Apesar desses desafios sinto que obtive um resultado bom, apesar de apenas ter criado um enigma (mudar as cores dos botões para mostrar o código do cofre ao jogador). No início do projeto o meu objetivo era criar talvez mais um sala com novos desafios mas infelizmente devido ao tempo não foi possível.

Ao desenvolver este projeto pude aplicar os conhecimentos aprendidos nas aulas práticas de Introdução à Computação Gráfica, mas acima de tudo aprendi a aplicar muitas ferramentas novas que tive que usar.

# Repositório Git

O repositório Git onde está o jogo é o seguinte:

[https://github.com/pereira-goncalo/93310\\_ICG](https://github.com/pereira-goncalo/93310_ICG)