INSTITUTO TECNOLÓGICO DE AERONÁUTICA

Fundamentos de Computação Gráfica

CCI-36

# Laboratório 3


**COMP 20**

Heládio Sampaio Lopes

Sebastião Beethoven Brandão Filho

**Professor:**

Carlos Henrique Q. Forster



ITA-SJC-2019

# Descrição do código implementado

O Laboratório proposto possui como objetivo a construção de um ambiente virtual interativo, como um minijogo ou um editor, por exemplo. Para o projeto confeccionado, realizou-se a simulação de um tabuleiro de xadrez, com as devidas movimentações específicas para cada peça, além do movimento de captura.

Para a realização do Laboratório foram seguidos determinados pré-requisitos estabelecidos. Para alguns tópicos são ilustrados determinados trechos de código implementando, objetivando a validação do pré-requisito associado.

1. **Responder a cliques do mouse em objetos específicos**

   Para implementar esse requisito foi necessário dividir esse momento em duas ocasiões. Primeiro, quando não temos nenhuma peça selecionada, é possível selecionar apenas as peças da vez, brancas ou pretas. Ao segurar na peça, a mesma se desloca no eixo z ascendente, e se torna possível movimentá-la no tabuleiro para as posições permitidas pela regra do xadrex.

   Em um segundo momento, depois de selecionado a posição final da peça, ao soltar o botão do mouse a peça passará a ocupar aquela posição caso permitido ou retornará ao local inicial caso seja uma posição inválida.

   O código a seguir realiza a ilustração da resposta a cliques do mouse em objetivos específicos. No trecho de código abaixo é possível ver a seleção, movimentação e clique do mouse.

```
1    if (event.type === 'mousedown' && state === 'preselect') {
2        intersects = raycaster.intersectObjects(pieces);
3        if (intersects.length) {
4            if (intersects[0].object.is_white === whites_turn) {
5                state = 'grab';
6                grab = intersects[0].object;
7                col = grab.is_white;
8                grab.position.z = 1.0;
9                grab_x = grab.position.x;
10               grab_y = grab.position.y;
11               ax = grab_x;
```

```
                ay = grab_y;
                type = grab.type;
                return; // sempre retornar se mudou de estado
            }
        }
    }

    if (event.type === 'mousemove' && state === 'grab') {
        intersects = raycaster.intersectObjects(slots);
        if (intersects.length) {
            target_slot = intersects[0].object;
            if (get_possibilites(ax, ay)) {
                target_slot.material = mat_highlight;
                grab_x = target_slot.position.x;
                grab_y = target_slot.position.y;
            }
        }
    }
    if (event.type === 'mouseup' && state === 'grab') {
        console.log(event.type);
        intersects = raycaster.intersectObjects(slots);
        if (intersects.length) {
            target_slot = intersects[0].object;
            if (get_possibilites(ax, ay)) {

                if (kill) {
                    scene.remove(target_slot.piece);
                }

                origin_slot = grab.slot;
                origin_slot.piece = null;
                target_slot.piece = grab;
                grab.slot = target_slot;
                grab.position.x = target_slot.position.x;
                grab.position.y = target_slot.position.y;
                grab.position.z = 0.5;
                state = 'turning';
```

```
49                  anim_time = 0;
50                  whites_turn = !whites_turn;
51
52                  update_board();
53
54                  grab = null;
55                  return; // sempre retornar se mudou de estado
56              }
57          }
58          state = 'preselect';
59          origin_slot = grab.slot;
60          grab.position.x = origin_slot.position.x;
61          grab.position.y = origin_slot.position.y;
62          grab.position.z = 0.4;
63          grab = null;
64          return; // sempre retornar se mudou de estado
65      }
```

2. **Gerar uma sequência complexa de ações ao longo do tempo**

   Esse requisito é cumprido quando um jogador seleciona sua peça, a mesma se desloca para cima, ao arrastar para posições válidas a casa selecionada muda de cor, permitindo que o jogador saiba quais movimentos são possíveis, caso a posição em questão possuir uma peça adversária é permitido ao jogador capturá-la. Por fim, ao finalizar a jogada o tabuleiro gira dando vez ao outro jogador.

   Alé do trecho anterior, que exemplifica parte desse requisito, o código a seguir ilustra a implementação associada à geração de sequências complexas de ações ao longo do tempo.

```
1       if (event.type === 'paint' && state === 'turning') {
2           anim_time = anim_time + 0.05;
3           if (whites_turn) t = t_blacks + anim_time;
4           else t = t_whites + anim_time;
5           if (anim_time > Math.PI) state = 'preselect';
6       }
```

Listagem 1: Movimento de girar tela

3. **Permitir arrastar e soltar objetos específicos em alvos específicos**

Para cumprir esse requisito é permitido ao jogador selecionar apenas as peças da sua vez e, ao clicar e arrastar, cada peça só pode ter uma movimentação expecífica de acordo com o seu tipo.

O código a seguir representa a implementação associada a arrastar e soltar objetos específicos em alvos específicos.

```
1         intersects = raycaster.intersectObjects(pieces);
2         if (intersects.length) {
3             if (intersects[0].object.is_white === whites_turn) {
```

Listagem 2: Selecionar peças da vez.

```
1         intersects = raycaster.intersectObjects(slots);
2         if (intersects.length) {
3             target_slot = intersects[0].object;
4             if (get_possibilites(ax, ay)) {
```

Listagem 3: Soltar peças em slots válidos (get_possibilites(x, y)).

4. **Descrever o modelo de interação (conjunto de regras)**

O projeto realizado corresponde a uma simulação de um jogo de xadrez entre dois jogadores. Foram simplificadas algumas regras do jogo de xadrez completo, devido a sua grande complexidade. Em sua vez o jogador pode selecionar uma peça específica. Cada peça (peão, torre, cavalo, bispo, rei e rainha) possui movimentos específicos e, após a seleção, é indicado ao jogador os movimentos que podem ser realizados, bem como a captura. Após concluída sua rodada, o tabuleiro gira, possibilitando que o outro jogador faça suas escolhas e que o jogo continue.

Exemplificando o conjunto de regras, a função `get_possibilites(x, y)` do arquivo ***functions.js*** retorna se é possível que a peça selecionada avance para a posição (x, y) desejada.

# Validação do código implementado

As Figuras 1 e 2 ilustram os resultados obtidos após a implementação do código criado. Apresentam o tabuleiro de xadrez no início do jogo e após a realização de algumas jogadas.

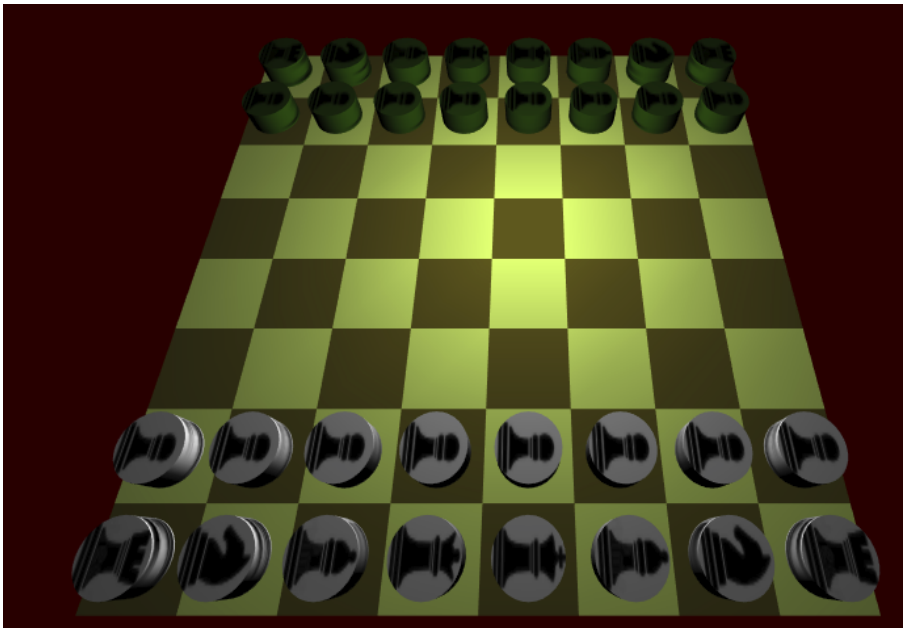

Figura 1: Representação do tabuleiro no início do jogo.

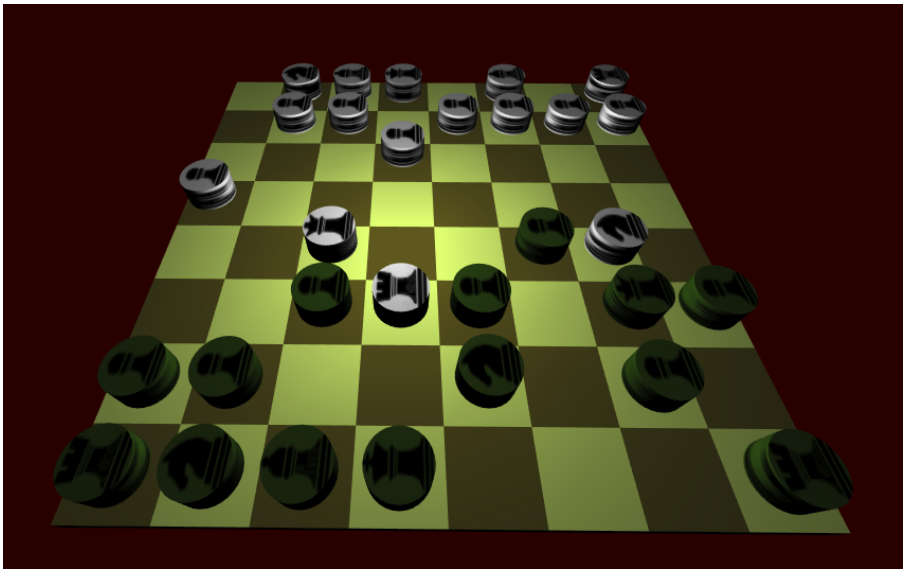

Figura 2: Representação do tabuleiro após a realização de algumas jogadas.