Universidade Federal da Paraíba
Centro de Informática

Introdução à Computação Gráfica
Atividade Prática 0 - JavaScript e Three.js

Francisco Siqueira Carneiro da Cunha Neto[1]
20190029015
[1]francisconeto@eng.ci.ufpb.br

20 de agosto de 2021

QUESTÕES RESPONDIDAS

Todas as questões propostas foram respondidas. Isso inclui os problemas 1 – 5 da seção 3 (Problemas JavaScript) e os problemas 1 – 3 da seção 4 (Problemas Three.js).

O código dos problemas da seção 3 se encontra na pasta `/JavaScript/js/`, em que cada problema possui um arquivo .js diferente, enquanto `/JavaScript/index.html` é um documento HTML que os exemplifica. O código referente a seção 4 se encontra no arquivo único `/ThreeJS/js/cubes.js`, similarmente junto com o documento HTML `/ThreeJS/index.html`.

O código também pode ser encontrado em seu repositório no GitHub.

ABORDAGEM

Problemas JavaScript

Todo o código para os problemas da seção 3 foi estruturado na forma de classes e funções. Algumas dessas resolvem os problemas como descritos na atividade e outras fazem uma interface entre o código JavaScript e o usuário que interage com ele através do documento HTML. Esse relatório foca apenas naquelas de resolvem os problemas.

Os problemas 1 e 2 da seção 3 necessitaram apenas do conhecimento prévio de JavaScript do autor. No problema 1 foi usado apenas o comando `alert("Hello world!")`. No problema 2 foram usadas duas funções, a primeira delas, `make_random_int_array(size)`, recebe um tamanho *n* como parâmetro, adiciona a um *array n* números inteiros aleatórios entre 0 e 100, e retorna esse *array*. A segunda função, `num_even_in_array(arr)`, recebe um *array* como parâmetro e conta quantos elementos contidos nele tem resto da divisão por 2 igual a 0.

O algoritmo Quicksort usado para resolver o problema 3 foi baseado no artigo [1]. Foram criadas três funções, `swap(array, a, b)`, `partition(array, low, high)` e `quicksort(array, low, high)`. A função auxiliar `swap(array, a, b)` inverte o conteúdo de `array[a]` e `array[b]`, já `partition(array, low, high)` faz parte do algoritmo em si. Ela organiza um *subarray* delimitado pelos parâmetros `high` e `low`, definindo uma variável `pivot` (o valor mais a direita do *subarray*) e duas variáveis `left_greater` e `left_unknown`, que marcam o índice inicial respectivamente do grupo de valores maiores que `pivot` e do grupo de valores cuja relação com `pivot` é desconhecida, ambos inicialmente iguais a `low`. A função então testa cada valor do grupo delimitado por `left_unknown`, se o valor for maior que `pivot`, apenas se incrementa `left_unknown`, se não, o valor é trocado com aquele que se encontra em `left_greater`, e ambos `left_greater` e `left_unknown` são incrementados. Finalmente, `pivot` é trocado com o valor que se encontra em `left_greater`, e o índice de `pivot` é retornado. A função `quicksort(array, low, high)` faz uma chamada da função `partition(array, low, high)` e então faz duas chamadas a si mesma, uma delimitando o subarray após o *pivot* retornado por `partition()` e outra o anterior ao *pivot*.

A teoria matemática usada para o problema 4 foi consultada do livro [2], e a criação das classes de JavaScript do artigo [3]. A classe `Vector3` possui as propriedades `x`, `y` e `z`, que representam suas coordenadas num espaço cartesiano, o *getter* `module()` que retorna sua norma, e as funções estáticas `cross(a, b)` e `dot(a, b)`, que recebem como parâmetro dois objetos `Vector3` e retornam, respectivamente, o resultado do produto vetorial entre eles como um novo objeto `Vector3` e o resultado do produto escalar entre eles. A classe `Matrix3x3` possui uma única propriedade `el`, um *array* bidimensional. Seu construtor recebe três objetos `Vector3` como parâmetro, e cada um é definido como uma coluna da matriz. Ela também possui dois *getters*, `determinant()` que retorna o determinante da matriz e `transpose()` que retorna um novo objeto `Matrix3x3` com a sua transposta, e duas funções estáticas, `mult_matrix_vec(mat, vec)` que retorna o produto entre um objeto `Matrix3x3` e um objeto `Vector3` e `mult_matrix(a, b)` que retorna o produto entre dois objetos `Matrix3x3`.

Embora o enunciado solicite a criação de "uma pequena biblioteca", o uso da *keyword* `export` gerou um erro no documento HTML de *Cross-Origin Resource Sharing* (CORS), descrito no artigo [4] e mostrando na figura 1. Esse erro não foi

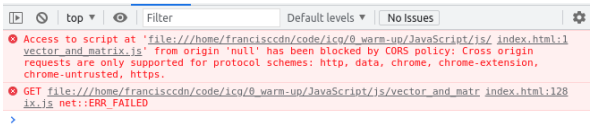

Figura 1: *Erro de CORS.*

resolvido ao definir *type="module"* na *tag script* que importa o código no documento e, segundo [4], para resolvê-lo é necessário usar uma URL HTTPS, o que foge do escopo da atividade. Portanto, as classes foram escritas, mas sem a criação de uma biblioteca.

O uso do elemento HTML canvas para a resolução do problema 5 foi norteado pelo artigo [5]. Foram criadas várias funções para desenhar diferentes figuras, cada uma delas declara uma variável `ctx` que armazena o contexto do canvas e então utiliza as funções `ctx.beginPath()`, `ctx.moveTo()`, `ctx.lineTo()`, `ctx.arc()`, `ctx.stroke()`, `ctx.fill()` e `ctx.fillRect()` para desenhar alguma forma.

Problemas Three.js

O código que resolveu os problemas da seção 4 foi estruturado da segunite forma: Foram declaradas variáveis globais `camera`, `scene` e `animation` para armazenar respectivamente uma referência à câmera, à cena e a alguma função anônima, e foi declarada uma função `animate()`, chamada a cada *frame*, que executa a função referenciada por `animation` e renderiza a cena referenciada por `scene` usando a câmera referenciada por `camera`. Para cada problema foram, então, criadas funções que modificavam a posição e rotação da câmera referenciada por `camera` e subsituiam tanto a cena referenciada por `scene` quanto a função referenciada por `animation`. Assim, foi possível manter todos os *renderings* em uma mesma exibição, escolhendo qual exibir de acordo com a função chamada (controlada por botões no documento HTML). Todos os problemas da seção 4 foram resolvidos com base na documentação do Three.js [6].

Para o problema 1 e 2, a cena criada contém um único cubo branco usando o material *Mesh Basic Material* [7]. No problema 1 `animation` é mantido como uma função vazia, já no problema 2 `animation` referencia uma função que incrementa as rotações ao longo do eixo x e y do cubo por 0.01. A cena criada para o problema 3 contém três cubos espaçados por 5 unidades ao longo do eixo x, usando os materiais *Mesh Physical Material* [8], *Mesh Toon Material* [9] e *Mesh Normal Material* [10], além de uma luz ambiente com intensidade fraca e uma luz direcional forte posicionada acima dos cubos. A variável `animation` também é mantida como uma função vazia no problema 3.

Um problema extra também foi proposto e resolvido, em que três cubos com materiais distintos e sendo rotacionados em tempo real são desenhados na tela do *browser*. Para tal, usou-se a mesma cena do problema 3, apenas substituindo a função de `animation` por uma que incrementa as rotações ao longo do eixo x e y dos três cubos por 0.01.

RESULTADOS

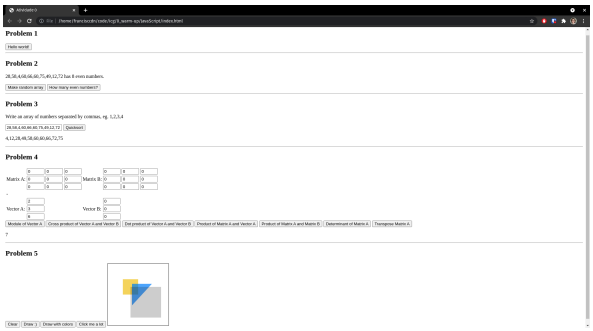

Figura 2: *Problemas da seção 3.*

A interface desenvolvida para os problemas da seção 3, bem como o resultado de algumas das funções criadas para estes, pode ser vista na figura 2. Os desenhos feitos no canvas para o problema 5 da seção 3 podem ser vistos nas figuras 3 e 4.

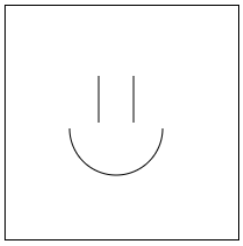

Figura 3: *Smiley desenhado no canvas.*

Figura 4: *Formas desenhado no canvas.*

A figura 5 mostra o *rendering* do problema 1 da seção 4, enquanto a figura 6 mostra o *rendering* do problema 3 da seção 4. Para observar os cubos em rotação, o leitor é convidado a baixar o código fonte e abrir o documento HTML `/ThreeJS/index.html` em seu navegador.

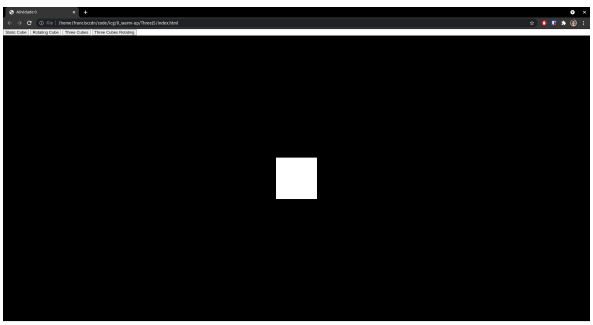

Figura 5: *Cubo estático branco.*

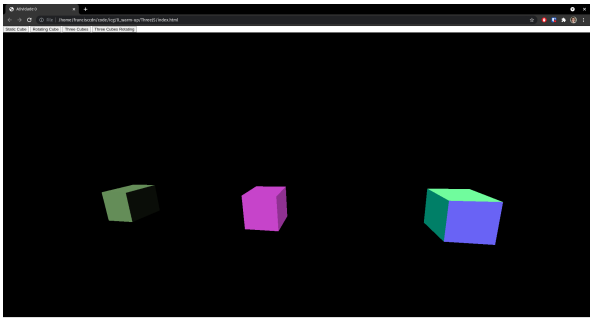

Figura 6: *Cubos de diferentes materiais.*

REFERÊNCIAS

[1] T. Cormen e D. Balkcom. (2014). "Khan Academy - Algorithms: Quick Sort," endereço: https://www.khanacademy.org/computing/computer-science/algorithms#quick-sort (acesso em 16/08/2021).

[2] W. K. Nicholson, *Álgebra Linear*. AMGH Editora, 2014.

[3] MDN. (2021). "Classes - JavaScript," endereço: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Classes (acesso em 17/08/2021).

[4] ——, (2021). "Reason: CORS request not HTTP," endereço: https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS/Errors/CORSRequestNotHttp (acesso em 17/08/2021).

[5] ——, (2021). "<canvas>: The Graphics Canvas element," endereço: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/canvas (acesso em 17/08/2021).

[6] Three.js. (2015). "three.js docs," endereço: https://threejs.org/docs/ (acesso em 18/08/2021).

[7] ——, (2015). "MeshBasicMaterial," endereço: https://threejs.org/docs/?q=mesh#api/en/materials/MeshBasicMaterial (acesso em 18/08/2021).

[8] ——, (2015). "MeshPhysicalMaterial," endereço: https://threejs.org/docs/?q=mesh#api/en/materials/MeshPhysicalMaterial (acesso em 18/08/2021).

[9] ——, (2015). "MeshToonMaterial," endereço: https://threejs.org/docs/?q=mesh#api/en/materials/MeshToonMaterial (acesso em 18/08/2021).

[10] ——, (2015). "MeshNormalMaterial," endereço: https://threejs.org/docs/?q=mesh#api/en/materials/MeshNormalMaterial (acesso em 18/08/2021).