Universidade Federal da Bahia - UFBA
Instituto de Matemática - IM
Departamento de Ciência da Computação - DCC
Curso de Bacharelado em Ciência da Computação

MATA65 - Computação Gráfica Período: 2018.1 Data: 08/05/2018.
Prof. Antonio L. Apolinário Junior
Estagiário Docente: Rafaela Souza

Roteiro do Laboratório 4 – Sistemas de Visualização e Projeções

Objetivos:

- Compreender de forma prática o conceito de **Sistemas de Visualização** e de **Transformações Projetivas**
- Com uso da linguagem *Three.js*, utilizar diferentes sistemas de visualização de objetos 3D de acordo com os conceitos apresentados em sala de aula.
- Compreender os aspectos práticos/visuais dos tipos de transformações projetivas;

Conceitos básicos (Sistemas de Visualização):

No modelo de câmera virtual utilizado em computação gráfica, é permitido o **posicionamento** da câmera no espaço 3D, o controle da **direção de observação** da câmera a partir da definição de um ponto focal, bem como a **orientação** da câmera em relação a esse ponto focal. Parâmetros como **razão de aspecto** (proporção entre altura e largura da imagem a ser gerada) e **campo de visão** também são utilizados para definir o volume de visão da câmera.

No *Three.js,* a posição da câmera pode ser definida a partir do atributo **`position`** (instância do tipo **`THREE.Vector3`**), a direção de observação pode ser definida a partir do atributo **`lookat`** (instância do tipo **`Three.Vector3`**), e a orientação da câmera pode ser definida a partir da manipulação dos atributos e métodos referentes à transformação geométrica do tipo **`THREE.Object3D`** (uma vez que o tipo abstrato **`THREE.Camera`** herda todos os métodos e atributos do tipo **`THREE.Object3D`**). Razão de aspecto (atributo **`aspect`**) e campo de visão (atributo **`fov`**) são atributos ponto-flutuante de um tipo específico de câmera (**`THREE.PerspectiveCamera`**). O volume de visão de uma câmera também pode ser definido a partir dos atributos ponto-flutuante **`left`**, **`right`**, **`top`**, **`bottom`**, **`near`** e **`far`**.

O *Three.js* também provê uma série de objetos, métodos e atributos que facilitam o controle da câmera virtual pelo usuário. Cada tipo de objeto possui um conjunto particular de parâmetros que fazem o ajuste automático dos parâmetros básicos da câmera. Exemplos de objetos para controle da câmera virtual são: **`THREE.TrackballControls`**, **`THREE.FirstPersonControls`**,

THREE.FlyControls, **THREE.OrbitControls**. Todos eles recebem como parâmetro uma câmera (do tipo **THREE.PerspectiveCamera**, por exemplo).

Conceitos básicos (Transformações Projetivas e Visibilidade):

As transformações projetivas planares podem ser divididas em dois grandes grupos: **paralelas** e **perspectivas**. A projeção **paralela** produz resultados pouco realistas em que o tamanho da imagem não varia de acordo com a distância do objeto em relação ao plano de projeção. A projeção **perspectiva** gera imagens cujo tamanho é inversamente proporcional a distância do objeto ao plano de projeção.

No *Three.js,* as matrizes de projeção são representadas a partir do atributo **projectionMatrix** do objeto **THREE.Camera**. Projeção paralela é obtida com o uso de uma câmera do tipo **THREE.OrthographicCamera**, enquanto a projeção perspectiva é obtida com o uso de uma câmera do tipo **THREE.PerspectiveCamera.**

Roteiro:

1. Baixe do repositório da disciplina no Moodle, os códigos fonte e as dependências base para esse Laboratório. Descompacte no diretório que será visível pelo servidor *web*.
2. Configure o servidor *web*[1] e execute cada um dos exemplos desse Laboratório.
3. Abra os códigos e analise cada um dos exemplos que compõe esse Laboratório. Entenda os parâmetros de controle da câmera virtual utilizados em cada exemplo. Note que é possível modificar os parâmetros do objeto diretamente mudando seus atributos, ou controlá-los através dos objetos controle de camera.
4. Modifique o código do exemplo **camera.js** para que o modelo da esfera possa ser visto em sua totalidade e no centro da tela.
5. Faça uma modificação semelhante para o código do exemplo **cameraGUI.js**. Utilize os controles para identificar mais facilmente a melhor localização da câmera.
6. Crie um novo código, baseado no exemplo **câmera.html/camera.js** substituindo o modelo da esfera por um modelo OBJ pronto: **../../Assets/Models/city.obj**. Faça os ajustes necessários para que a câmera consiga ver todo o modelo de uma perspectiva "aérea". Execute o seu novo código no servidor *web* e verifique se a visualização foi feita com sucesso.
7. Você deve ter notado que para um ajuste da câmera para um novo modelo você teve de “intuir" os parâmetros da câmera relativos a localização e direção de observação. Seria possível definir esses parâmetros de forma automática para cada objeto carregado? Quais seriam os parâmetros do modelo necessários para tal definição?
8. A partir da sua resposta a questão do item 7), procure no manual do ***Three.JS*** pela referência ao objeto **BoundingBoxHelper**, e seu atributo **Box**. Tente utilizar esse objeto como parte da resposta do item 7). Implemente uma mudança no seu novo código de tal

[1] necessário para o ambiente Windows. Utilize o programa USBWebServer (http://www.usbwebserver.net/en/) disponível no repositório da disciplina: http://homes.dcc.ufba.br/~apolinario/Disciplinas/2016.1/MATA65/USBWebserver%20v8.6.zip

sorte que a câmera tenha um posicionamento inicial “razoável" qualquer que seja o modelo carregado.

9. Agora que você pode carregar qualquer modelo, podemos tornar a sua visualização mais interativa com os controladores de camera do ***Three.JS.*** Procure no diretório **`../libs/examples/js/controls`** pelo controle de câmera **`TrackballControls`**. Inclua esse controle de câmera na sua aplicação de forma que o objeto OrbitContol controle o movimento da câmera. Execute o *script* modificado no servidor *web* e verifique se o controle de funciona de forma adequado. Experimente combinações de seus parâmetros para entender seus efeitos no movimento.

10. Crie novas aplicações para testar os controles de câmera **`TrackballControls`**, **`FirstPersonControls`**, **`FlyControls`**. Da mesma forma, teste seus principais parâmetros e veja os resultados obtidos pelas suas variações.

11. Modifique o exemplo do código **`projection+visibility.html/projection+visibility.js`** para que o modelo da cidade seja visto por uma câmera ortográfica que utiliza a projeção paralela. Modifique a posição da câmera para que você tenha uma visão “aérea” de toda a cidade.

12. Crie um novo código que permita que o usuário visualize simultaneamente o resultado de captura das duas câmeras, ortográfica e perspectiva, com controle orbital.

13. **Desafio:** Retorne ao laboratório 2 e tente alterar o modelo de sistema planetário que voce criou. Tente conseguir visualizações onde:

    1. A câmera esteja posicionada fixa no centro da Terra olhando em direção ao Sol.

    2. A câmera localizada sobre a superficie da Terra, de preferência no Equador, acompanhando sua rotação, e olhando para o horizonte.