Computação Gráfica – Projeto 2

Objetivo:

Construir um cenário 3D a partir de modelos/malhas pré-existentes e com aplicação de textura. O programa deve permitir explorar o cenário por meio de manipulação da câmera (Matrizes Model x View x Projection).

Requisitos:

1. O cenário deve conter um ambiente interno (casa, prédio, submarino, etc.) e um externo (quintal/jardim, rua, oceano, etc). Ele deve ter um objetivo bem definido, ou seja, os modelos e transformações devem fazer sentido para a cena.
2. Devem ser importados e exibidos seis ou mais modelos com textura, todos 3D. Repetições de um mesmo modelo são permitidas (por ex, várias árvores vindas de um mesmo modelo), mas contarão como um só modelo. **Não será permitido o uso dos modelos já vistos nas aulas.**
3. Pelo menos três modelos devem estar no ambiente interno e pelo menos três modelos devem estar no ambiente externo. Por exemplo, se o ambiente interno for uma casa, então a casa pode conter móveis, tv, pessoas, animais de estimação, etc. O ambiente externo pode ter, por exemplo, árvores, carros, pessoas, animais, etc.
4. Não serão aceitos modelos que contenham múltiplos objetos. Por exemplo, o modelo desse link oferece uma sala de estar já com móveis e objetos de decoração. Esse é um exemplo de algo não permitido. Cada item da cena deve vir de um arquivo .obj diferente.

5. Os modelos devem ser ajustados convenientemente e de um jeito coerente com o mundo real. Exemplos: sofás e pessoas não flutuam; um óculos não é maior que uma mesa, etc.
6. Ambos os ambientes devem ter piso/chão, e o piso do ambiente interno deve ser diferente do piso do ambiente externo.
7. As transformações escala, rotação e translação devem ser aplicadas, cada uma em um modelo diferente e com acionamento via teclado de forma independente.
8. O ambiente externo deve possuir um céu, com sua respectiva textura. Dica: pesquise por SkyBox.
9. O seu programa deve restringir a exploração do cenário dentro dos limites da "borda" do céu e terreno. Não é necessário restringir o acesso aos modelos, ou seja, a câmera pode passar livremente "por dentro" dos modelos (com exceção do céu e terreno).
10. O usuário deve poder visualizar a malha poligonal quando quiser. Caso ele aperte 'p', as malhas dos objetos devem ser exibidas (caso estejam ocultas) ou ocultadas (caso estejam sendo exibidas).
11. Os modelos devem ser importados de arquivos Wavefront (.obj). É possível utilizar o Blender para importar modelos em outros formatos e exportar para o formato .obj. Também é possível fazer download diretamente dos arquivos .obj. Alguns repositórios são: https://free3d.com/3d-models/, https://www.blendswap.com/ e https://www.cgtrader.com/free-3d-models. Existem outros repositórios que podem ser utilizados livremente. **Importante: antes de usar um modelo, confira se ele vem com alguma imagem (.jpg, .png, etc.) de textura.**
12. Neste trabalho, NÃO devem ser utilizados efeitos de iluminação. Esse será um tema coberto pelo próximo trabalho.

Critérios de Avaliação:

1. Atendimento aos requisitos.
2. Complexidade da cena e o quanto o resultado final faz sentido / é coerente.
3. Qualidade do código (estruturação e comentários), **que deverá ser submetido via e-disciplinas até o dia 29 de outubro.**
4. **Apresentação em sala de aula no dia 30 de outubro. Note que a entrega deve ser feita no dia anterior à apresentação.**

Outras informações importantes:

1. Pode ser feito individualmente ou em dupla.
2. **Caso você tenha feito o primeiro projeto sozinho e agora fará em dupla, ou caso a dupla tenha mudado, enviar email para jeanponciano@icmc.usp.br comunicando os integrantes da dupla. Coloque o código da disciplina (SCC0250 ou SCC0650) no assunto.**
3. Pode utilizar qualquer código-base fornecido.
4. Pode-se utilizar, inclusive, outras linguagens de programação, desde que utilize apenas bibliotecas do OpenGL e do sistema de janelas. O uso de outras bibliotecas gráficas não será aceito.
5. Devem ser utilizadas apenas funções do pipeline moderno. No OpenGL, isso significa que as seguintes funções são obsoletas (*deprecated*) e não podem ser utilizadas: glRotate, glTranslate, glScale, glVertex, glColor, glLight, glMaterial, glBegin, glEnd, glMatrix, glMatrixMode, glLoadIdentity, glPushMatrix, glPopMatrix, glRect, glBitmap, glAphaFunc, glNewList, glDisplayList, glPushAttrib, glPopAttrib, glVertexPointer, glColorPointer, glTexCoordPointer, glNormalPointer, glMatrixMode, glCal.