# Voxel-based environments - 30 pontos

Este trabalho consiste na criação de um protótipo de modelagem baseado em *voxels* (*voxel-based modelling*). Esse tipo de modelagem é muito utilizada em jogos baseados em voxels, sendo Minecraft o mais famoso exemplo. Nesta primeira etapa o foco será no desenvolvimento dos elementos básicos deste protótipo, incluindo a possibilidade de adicionar e remover voxels, carregar e salvar um grupo de voxels em arquivo etc.

---

### Ambiente de modelagem

*Tipos de voxels*

Neste trabalho teremos um total de 5 tipos de voxels, cada um com uma cor diferente. As duas cores básicas podem ser vistas neste mapa (voxels que compõem a camada `N1` e `N2` no mapa). As outras três cores serão utilizadas nas árvores que comporão o ambiente (exemplos aqui) onde forma e cores serão definidas pelos grupos.

*Ambiente de modelagem*

Os objetos que serão inseridos no ambiente de execução serão criados em um ambiente à parte chamado "Ambiente de modelagem" ou `Builder`. Esse ambiente será composto por um plano base 10 x 10, onde cada célula do plano será visível, e os *voxels* deverão ser inseridos através das instruções da tabela ao lado.

Voxels adicionados podem ser removidos. Pesquise a função `scene.remove(object)` para ver como isso é feito. A sugestão é remover o voxel através da sua posição, mas outras alternativas podem ser exploradas.

O local onde o objeto será adicionado no ambiente deve ser indicado com um cubo em modo wireframe, e deve-se criar uma forma de visualizar em qual altura aquele cubo está em relação ao plano base. O grupo deve definir como será essa visualização.

Para o ambiente de modelagem crie os arquivos `builder.js` e `builder.html`.

| Ação | Teclas |
|---|---|
| Movimentação no plano XZ | Setas direcionais do teclado |
| Movimentação em Y | PgUp e PgDown |
| Inserir voxel | 'Q' |
| Remover voxel | 'E' |
| Próximo tipo de voxel | '.' |
| Tipo anterior de voxel | ',' |

Ao menos três tipos de árvores baseadas em voxels devem ser criadas. Veja alguns exemplos nesta imagem.

*Gravação/carregamento de objetos modelados*

Ainda no ambiente de modelagem deve ser possível salvar os objetos criados em arquivo e posteriormente carregá-los. Ambas as opções devem ser feitas através da interface GUI (tem vários exemplos de uso dessa interface em nosso repositório e vocês já utilizaram em alguns exercícios). Além dos botões de salvar e carregar, deve-se passar como parâmetro o nome do arquivo a ser salvo ou o nome do arquivo a ser carregado dependendo do caso (esse link pode ser útil).

A forma de salvar/carregar arquivos deve ser simples, sem bibliotecas e/ou frameworks adicionais (*node* por exemplo). Pode-se usar, por exemplo, BLOB para salvar arquivos e FETCH para carregar.

---

### Ambiente de execução

No ambiente de execução teremos um mapa baseado em voxels (exemplo meramente ilustrado na Figura 1 ao lado). O ambiente a ser criado no T1 deve ser semelhante ao ilustrado neste mapa. No mapa a área `N0` tem altura 0 (é o plano base), `N1` tem altura 1 e `N2` tem altura 2. A forma como os voxels que comporão os níveis `N1` e `N2` serão inseridos no ambiente ficarão a critério do grupo.

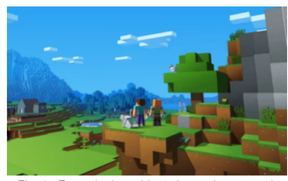


Fig. 1 - Exemplo de ambiente baseado em voxel.

Nas indicações marcadas pela letra "T" serão inseridas as árvores criadas no ambiente de modelagem. Como três modelos de árvores devem ser criados, dois objetos de cada modelo devem ser inseridos no ambiente de execução.

Aprimoramentos visuais serão incluídos nos próximos trabalhos.

*Controle de câmera*

Utilizaremos neste trabalho dois tipos de câmera: uma câmera de inspeção (`orbitControls`) e uma câmera em primeira pessoa (`first person camera`). Em nosso repositório, utilize o projeto `exampleFirstPerson` como base para a implementação desta segunda câmera. As câmeras serão alternadas pressionando a tecla 'C' do teclado. Deve-se armazenar a posição anterior de cada câmera ao alternar entre os modos para que, ao voltar para a câmera anterior, a posição seja a mesma.

---

### Outros

Para esta versão, utilize como material dos blocos o comando *setDefaultMaterial*, passando as cores como parâmetro. Para iluminar o ambiente, utilize o comando *initDefaultBasicLight(scene)*. Essas duas funções foram utilizadas em nosso primeiro exemplo (*basicScene.js*). O sistema definitivo de iluminação do projeto será definido em detalhes nos próximos trabalhos.

---

Considere para efeito de avaliação os seguintes critérios de pontuação geral (30 pontos no total):

| Grupo | Item | Pontos |
|---|---|---|
| **Ambiente de modelagem** (16 pontos) | Criação do 5 tipos de voxels<br>Voxels cúbicos serão diferenciados com 5 cores diferentes | **1** |
| | Criação do plano base do ambiente de modelagem<br>Plano 10 x 10 com visualização do grid para melhor posicionar os objetos | **1** |
| | Funcionalidade do ambiente de modelagem<br>Deve ser possível adicionar e remover objetos neste ambiente e local de inserção/remoção será indicado por um cubo em modo wireframe | **5** |
| | Mapeamento de teclado neste ambiente<br>Correto mapeamento das teclas para manipular os voxels neste ambiente | **2** |
| | Gravação/carregamento dos objetos modelados - criação da interface<br>Deve ser possível na interface incluir o nome do arquivo a ser gravado e/ou carregado além dos botões para salvar e carregar | **3** |
| | Criação das árvores<br>Criação de três tipos de árvores. Cada árvores será armazenada em um arquivo independente | **4** |
| **Ambiente de execução** (12 pontos) | Criação do ambiente de execução baseado no mapa disponibilizado como exemplo<br>O grupo deve tentar criar um ambiente o mais próximo possível do ilustrado como base | **5** |
| | Inclusão das árvores modeladas no ambiente<br>Inclusão das árvores modeladas nas posições indicadas no mapa | **3** |
| | Criação da câmera de inspeção (Orbit)<br>Correta inclusão da câmera e armazenamento de sua posição ao alternar entre os modos de câmera | **1** |
| | Criação da câmera de navegação (First person)<br>Correta inclusão da câmera e armazenamento de sua posição ao alternar entre os modos de câmera | **3** |
| **Outros** (2 pontos) | Material dos voxels<br>Correta utilização da função indicada para a criação dos materiais dos 5 tipos de voxels | **1** |
| | Iluminação<br>Correta inclusão da função básica de iluminação do ambiente | **1** |

**Nota 1:** Informações adicionais e/ou correções a este enunciado podem ser adicionadas na forma de comentários no Google Meet.

**Nota 2:** O trabalho pode ter uma penalização de até **30%** do total se forem encontrados problemas de usabilidade não mapeados na tabela acima.

**Nota 3:** Se forem identificadas cópias parciais ou totais de código, a nota será **dividida** pelos grupos (exemplo: se dois grupos envolvidos tirarem 24 pontos, cada grupo ficará com 12 pontos. Se forem três grupos envolvidos, serão 8 pontos para cada etc).

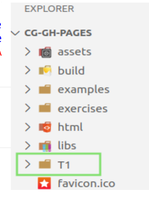
**Foco na apresentação**

Um dos aspectos mais importantes da implementação é a questão da clareza do código. O projeto deve ser minimamente **modelado** antes de ser implementado. O grupo será questionado a respeito de detalhes do código e a avaliação será individual.

Qualquer componente do grupo poderá ser questionado por qualquer parte da implementação. Estejam preparados de acordo.

---

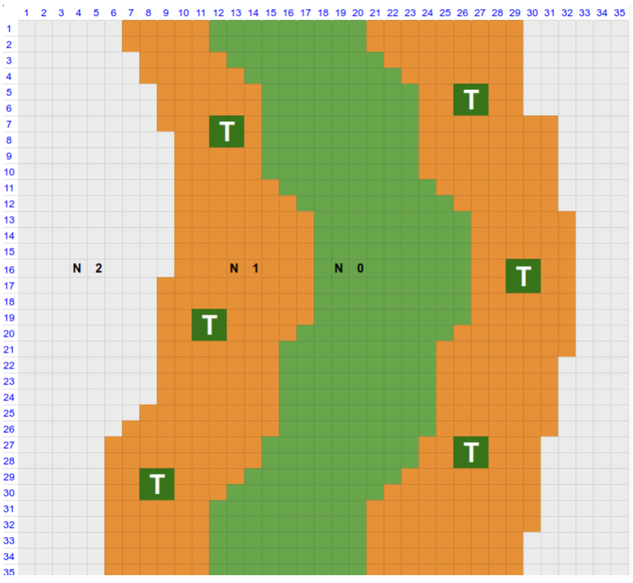
**Nota importante:** no mesmo nível da pasta *examples* do nosso repositório, crie uma pasta T1 e desenvolva seus códigos nesta pasta. Para enviar seu trabalho, compacte esta pasta (*zip*, *rar* etc) e envie via Google Classroom. TESTE SEU SISTEMA NO LABORATÓRIO ou EM ALGUMA MÁQUINA LINUX **ANTES** de enviá-lo.




| | |
|---|---|
| Prazo para envio do trabalho: | **07/12*** (sábado) |
| Datas de apresentação do trabalho: | **09/12** (segunda)<br>**12/12** (quinta) |

** Será aplicado um desconto de 10% na nota final para cada hora de atraso na entrega.*

**AMBIENTE DO T1** (voltar)




**Exemplos de árvores** (voltar)