Schema World

Universidade de Aveiro

Isac Cruz (90513)

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

1.1 Proposta

Gerador de mundo com base em ficheiros JSON, de modo a converter um ficheiro em um mundo tridimensional, criando uma ilha por cada objeto JSON, conectando cada ilha ao seu respetivos objetos relacionados por encapsulamento. O programa deve permitir a personalização de cada ilha através de campos extra que não sejam objetos.

ficheiro exemplo :

```
    {
    "main": {
        "branch1": {
            "node1": {},
            "node2": {
                "material.h": "wood"
            }
        },
        "branch2": {
            "node3": {}
        }
    }
}
```

1.2 Nomenclatura

- Raiz : A primeira ilha a ser gerada.
- Distância : numero de connecções mínimo entre uma ilha e a raiz
- Geração : O conjunto de ilhas que partilham a mesma distância da raiz.
- Ilha pai : A ilha imediatamente conectada com menor distância da raiz.
- Ilha filha : Qualquer ilha imediatamente conectada que não seja pai.
- Ilha descendente : Qualquer ilha que seja conectada a um filho independentemente da geração.
- Ilha irmã : Ilha que partilha o mesmo pai.
- Ilha Tia : Qualquer ilha descendente de um dos irmãos do pai ou qualquer irmão do pai.
- Herança : Qualquer atributo recebido de um pai.
- Ângulos Limitantes : Os ângulos entre os quais podem ser geradas ilhas filho, consiste num ângulo inicial e num ângulo final
- Amplitude : a diferença entre os ângulos limitantes herdados atribuída a uma raiz para a propagação dos filhos.

Capítulo 2

Base do Código

Este projeto usa como base um ambiente tridimensional de FPS(First Person Shooter) onde o utilizador é liver para explorar todo o cenário disponível sem qualquer tipo de restrições. estão disponíveis as ações de mover a câmera no estilo FPS, mover o peão (Seção 2.1) para as 4 direções(esquerda, direita, frente e trás) e saltar.

2.1 Peão

O peão é o corpo virtual do utilizador e está descrito no ficheiro player.js. Para além de descrever as ações de movimento do utilizador, também calcula o efeito da gravidade e gere algumas colisões para garantir que o peão pode subir rampas e caminhar sobre objetos planos.

2.2 Criação de terreno

Para facilitar a gerração de terreno o sistema foi divido em três etapas : funções de construção (Subseção 2.2.2), vetores de dispersão (Subseção 2.2.3), renderização de terreno (Subseção 2.2.4).

2.2.1 Fluxo de processamento

O ficheiro é carregado como objeto e iterado pelo gerador de mundo que calcular o tamanho de cada ilha com base no numero de conecções que esta possui, guardam o raio de cada ilha. São também calculados os as localizações de cada ilha num processo recursivo (Subseção 2.2.3) onde são adicionados os valores de personalização de cada ilha. Após esta analise e gerado um ficheiro com a descrição do mundo que é renderizado por um segundo iterador recursivo. (Ver Figura 2.1)

2.2.2 Funções de construção

Existem duas funções de construção de mundo principais:

- o gerador de cilindros que permite gerar ilhas com localização, raio, espessura e texturas específicos;
- o gerador de pontes onde dados dois cilindros, uma expessura, uma largura e um material, gera uma ponte entre os dois;

Desta forma limitamos todo o restante do código à pura geração de texturas, coordenadas e conecções entre pontes.

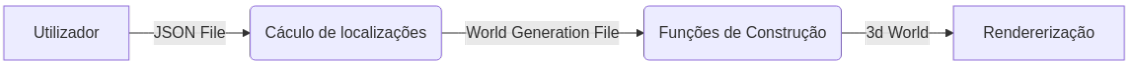

Figura 2.1: Fluxo de processamento

2.2.3 Calculo dos vetores de dispersão

Cálculo do raio de cada Ilha

É necessário garantir que cada ilha tem espaço para que seja explorável, possua perímetro suficientemente grande para fazer todas as conecções e tenha um mínimo de espaço entre connecções para evitar a sobrecarga visual. Abaixo apresenta-se a formula para o cálculo do raio dadas as restrições. O $espaamento_{EntrePontes}$ é um valor em percentagem referente ao tamanho das pontes. Caso seja 1 entre cada ponte vai existir o espaço mínimo de uma ponte.

$$raio = \max(\frac{bridge_{width} * number_{connections} * (1 + espacamento_{EntrePontes})}{2\pi}, raio_{minimo})$$

Restrição de Amplitudes

Para gerar um mundo explorável é necessário dispersar as ilhas de forma a evitar colisões e criar um mundo interessante de se explorar. Para facilitar o cálculo, foram utilizadas coordenadas polares (ρ, θ, φ).O mundo foi divido em quatro sectores de vinte graus de amplitude em φ :[40^{o},20^{o}] ; [20^{o},0^{o}] ; [0^{o},-20^{o}] ; [-20^{o},-40^{o}].

Cada secção será populada por uma parte das ilhas. Após uma dada ilha ser atribuída a uma secção, a inclinação dada é propagada para todos os filhos da mesma.

$$\varphi = \frac{\pi}{2} + \frac{\pi}{6} \vee \varphi = \frac{\pi}{2} + \frac{\pi}{18} \vee \varphi = \frac{\pi}{2} - \frac{\pi}{18} \vee \varphi = \frac{\pi}{2} - \frac{\pi}{6}$$

A divisão em θ é feita através divisão da amplitude disponível pelas conecções restantes. Cada ilha recebe uma amplitude que divide entre os filhos, amplitude recebida da ilha pai é propagada para as próximas gerações a fim de ser redividida posteriormente.

$$\theta = \theta_{InicialHerdado} + \frac{Amplitude_{Herdada}}{connections} * iteration$$

$$\varphi = \varphi_{Herdado} \vee \varphi = \varphi_{Atribuido}$$

Cálculo de comprimento de ponte

O comprimento de cada ponte é definido pela distância necessária a percorrer para que haja altura maior que 3 metros ($1.5 * AlturaDoPeao$), ou distancia suficiente para não colidir com a ilha mais próxima. É levado em consideração o maior destes dois valores. α é o ângulo em φ entre duas secções, neste caso é 20^{o}. β é a amplitude em θ entre duas ilhas irmãs(Seção 1.2).

$$\rho_{AlturaMinima} = \frac{3}{\sin(\alpha)}$$

$$\rho_{SemColisao} = \frac{raioIlha1 + raioIlha2 + miminaldistance_{entrePontes}}{2 * \sin(\beta/2)}$$

$$\rho = \max(\rho_{AlturaMinima} \quad , \quad \rho_{SemColisao})$$

O ρ é o comprimento da ponte

Calculo de Vectores

Após a definição ((ρ, θ, φ)), é necessário calcular a localização de cada ilha, através do calculo do vector de deslocação final(VDF). Este ultimo vector resulta da soma do vector ponte(VP) com o vector da ilha pai(VIP) e o vector da ilha filho(VIF). O VIP e o VIF servem para ajustar os cálculos às variações produzidas pelos raios variáveis de cada ilha.

O VP é o calculado com os valores ρ da Seção 2.2.3 e (θ, φ) da Seção 2.2.3:

$$VP = (\rho, \theta, \varphi)$$

De seguida temos o VIP e o VIF que dependem de θ e do raio de cada ilha:

$$VIP = (raioPai, \theta, \frac{\pi}{2})$$

$$VIF = (raioFilho, \theta, \frac{\pi}{2})$$

O VDF é dado pela expressão:

$$VDF = VIP + VP + VIF$$

Converter coordenadas polares para coordenadas cartesianas

$$z = sin(\varphi) * c \wedge hxy = \rho^2 - z^2$$

$$x = cos(\theta)hxy \wedge y = sin(\theta)hxy$$

$$Vector = (x, y, z)$$

Resultado

Para obter a localização da ilha é necessário apenas somar o VDF á ilha pai :

$$Loc = LocPai + VDF$$

Este processo precisa ser repetido para cada objecto JSON do ficheiro, a fim de gerar o ficheiro de criação de mundo.

2.2.4 Renderização

No final do processo recursivo de calculo das localizações, é gerado um objecto JSON com as variaveis para a criação de cada ilha.

```
{
 "location": {
     "x": 0,
     "y": 0,
     "z": 0
 },
 "radius": 0,
 "connections": [
     {
         "location": {
             "x": 47.00850714413811,
             "y": 0,
             "z": 15.000000000000004
         },
         "radius": 23.873241463784304,
         "connections": [
             {
                 "location": {
                     "x": 122.89025575206053,
                     "y": 0,
                     "z": 30.000000000000007
                 },
                 "radius": 28.873241463784304,
                 "connections": [
                     {
                         "location": {
                             "x": 203.77200435998293,
                             "y": 0,
                             "z": 45.000000000000014
                         },
                         "radius": 28.873241463784304,
                         "connections": [],
                         "name": "node1"
                     },
                     {
                         "location": {
                             "x": 163.33113005602175,
                             "y": 70.04564899696746,
                             "z": 45.000000000000014
                         },
                         "radius": 28.873241463784304,
                         "connections": [],
                         "name": "node2"
                     }
```

```
                    ],
                    "name": "branch1"
                },
                {
                    "location": {
                        "x": -28.87324146378431,
                        "y": 9.29283405463963e-15,
                        "z": 30.000000000000007
                    },
                    "radius": 28.873241463784304,
                    "connections": [
                        {
                            "location": {
                                "x": -109.75499007170673,
                                "y": 1.919799150885294e-14,
                                "z": 45.000000000000014
                            },
                            "radius": 28.873241463784304,
                            "connections": [],
                            "name": "node3"
                        }
                    ],
                    "name": "branch2"
                }
            ],
            "name": "main"
        }
    ],
    "name": null
}
```


Este ficheiro é iterado recursivamente utilizando as funções de construção(Subseção 2.2.2) para criar o mapa.

Capítulo 3

Futuras melhorias

3.1 Inclusão de personalização de Ilhas

É possível incluir personalização da ilha modificando o parser do ficheiro de forma a que este reconheça materiais instruções de iluminação e população da ilha. Para este efeito são também necessárias novas funções de construção.

3.2 Aproveitamento de espaço entre irmãos

Atualmente todos os irmãos se propagam de forma paralela, para garantir que as ilhas não se colidem em nenhum ponto, o que resulta em grandes corredores não populados. É possível criar amplitudes alargadas até preencher esse espaço, e retomar as amplitudes de propagação paralelas a partir daí. O algoritmo proposto consiste no aumento da amplitude herdada(representada abaixo por α) em 50% na direção de ambas as fronteiras, isto cria um ponto de colisão virtual que pode ser calculado pela formula :

$$Vector_{Colisao} = (\frac{distncia_{entreIrmaosVizinhos}}{2 * \cos(\frac{\pi-\alpha}{2})}, \frac{\alpha}{2})$$

$$Colisao_{virtual} = Localizacao_{IlhaIrmaoVizinho} + Vector_{Colisao}$$

As formulas deduzem a colisão nas coordenadas (x,y), o que permite calcular a partir de que ponto a amplitude alargada passa a gerar uma colisão. O algoritmo volta a utilizar fronteira herdada assim que a ilha gerada se apresenta mais longe do centro do mapa que o ponto de colisão virtual.

Isto permite utilizar a amplitude alargada

3.3 Aproveitamento de espaço não utilizado pelos tios

O algoritmo calculado divide a amplitude dada a cada ilha de forma igual independentemente do numero de gerações que as sucedam, ou seja, quando uma ilha recebe 90^{o} de amplitude, estes 90^{o} serão divididos entre os filhos e estes dividirão a sua fração entres os seus descendestes. Caso a Irmã da ilha a quem foi dado 90^{o}(a qual também recebeu 90^{o}) morrer na primeira geração, a sua amplitude não é aproveitada para as irmãs. Isto resulta em grandes espaços de vaco. Se for passado a cada família um array com as amplitudes dos seus seus tios para que estes saibam quando aproveitar a sua amplitude de herança.