Generarea modelului terenului

Terenul in mod particular fata de alte obiecte va avea urmatoarele proprietati in plus (care se vor regasi si in fisierul de configurare):

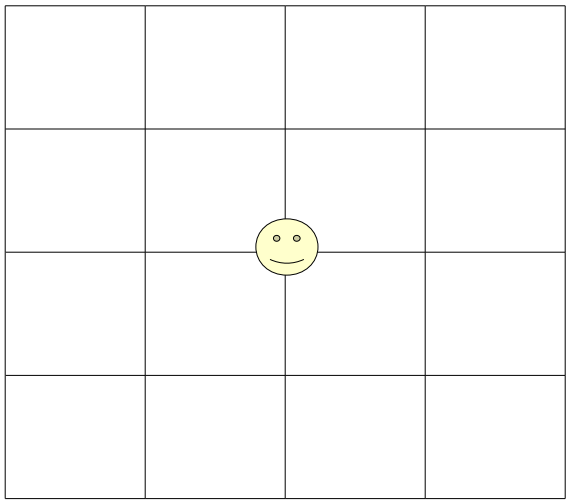
-numar celule pe orizontala

-numar celule pe verticala (de obicei egal cu numarul de celule pe orizontala, deci cele doua proprietati pot fi inlocuite de una singura: **nr_celule**)

-**dimensiunea unei celule**

-un **offsetY** (explicat mai jos)

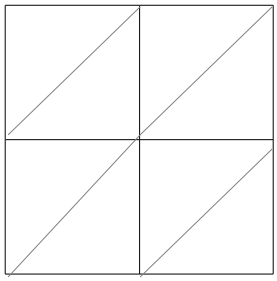
-La crearea terenului, acesta se va genera centrat fata de camera, asa cum se vede in imagine (in exemplu fiind un teren de 4*4 celule):



Terenul va fi paralel cu planul xOz si va avea un y prestabilit (**offsetY**).

In generarea vertecsilor terenului trebuie sa avem in vedere urmatoarele:

- va fi o grila dreptunghiulara de vertecsi
- in OpenGL ES nu avem quad-uri ca primitive, deci va trebui sa impartim patratelele din grid in triunghiuri, de exemplu:



-pozitiile initiale ale vertecsilor depind de pozitia camerei, deci putem face in doua moduri:

- Fie consideram centrul terenului ca fiind originea (coordonatele 0,0,0) si generam vertecsii fata de aceasta origine si translatam terenul cu centrul in pozitia camerei (doar pentru coordonatele x si z), iar y-ul sa fie egal cu **offsetY**.
- Fie (si solutia asta mi se pare mai usor de implementat si administrat) consideram de la inceput centrul terenului egal cu pozitia camerei (pentru coordonatele x si z) si y-ul egal cu **offsetY**, si generam pozitiile vertecsilor fata de acest centru asa cum am fi facut si mai sus.

Apoi generam si indicii corespunzatori, observand ca pentru fiecare quad avem cate 2 triunghiuri, deci 6 indici. O implementare simpla ar fi sa face doua for-uri care parcurg quad-urile (de ce 2 for-uri? Pentru ca putem considera ca avem o matrice de quaduri, deci parcurgem liniile si coloanele acestei matrici).

- pe teren se vor aplica 3 texturi de relief (Rock.tga, Dirt.tga, Grass.tga), fiecare din aceste trei texturi aplicandu-se in acelasi timp pe fiecare dintre celulele terenului => GL_REPEAT

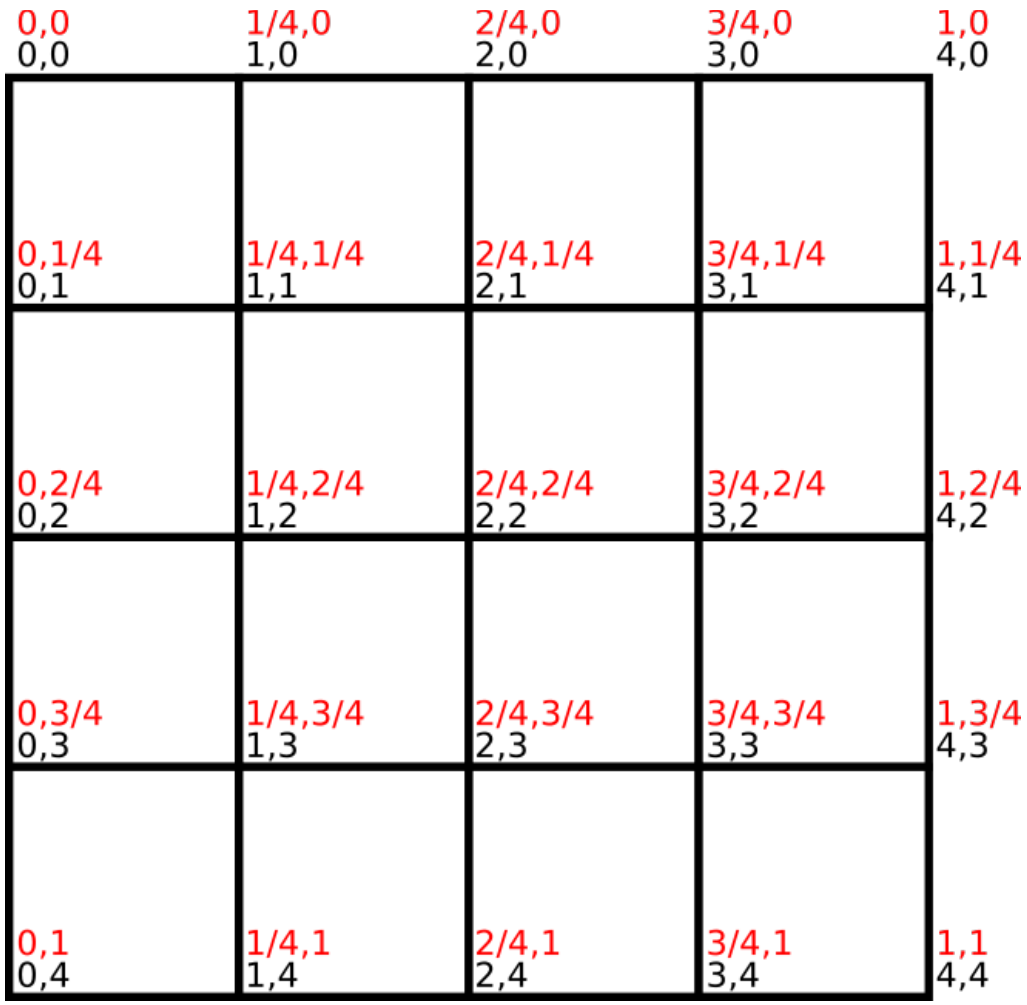
- pe teren se va mai aplica o textura de amestec (Terrain_blend_map.tga). Aceasta va determina in fiecare fragment al terenului care textura se aplica dintre cele 3 mentionate mai sus. Pe textura de amestec se observa zonele rosii, verzi si albastre. Vom asocia, cu ajutorul fisierului de configurare cate o culoare fiecarei texturi. De exemplu pentru rosu avem pietris (rock), pentru albastru avem pamant (dirt) si pentru verde avem iarba (grass). Pentru a evita if-urile si pentru a putea trata usor si zonele de tranzitie intre culorile din textura de blending, vom folosi formula urmatoare:

```
c_final= c_blend.r*c_rock+c_blend.g*c_grass+c_blend.b*c_dirt;
c_final.a=1.0;
```

Unde, c_rock e culoarea fragmentului, preluat din textura rock, c_grass, culoarea din textura grass si c_dirt, culoarea din textura dirt.

Texturile rock, grass si dirt se aplica pe patratelele gridului (cu repeat), pe cand textura de blend se aplica pe tot gridul. Prin urmare o sa avem doua uv-uri: uv-ul pentru texturile mici, care e la fel pentru toate in cadrul unei patratele de grid, deoarece toate cele 3 texturi se aplica la fel pe un patratel; si uv-ul blend_map-ului, care se aplica pe intreg gridul.

De exemplu, pentru un grid 4X4 uv-urile vor arata asa (cu rosu uv-ul blend-map-ului si cu negru uv-urile texturilor mici):



La deplasarea camerei, trebuie sa avem senzatia de teren infinit. Astfel daca ne deplasam pe x sau pe z cu mai mult decat dimensiunea unei patratele trebuie “sa se genereze” un rand sau coloana noua in grid in sensul deplasarii (ca sa avem in continuare teren in fata pe care sa ne deplasam) si sa “se stearga” ultimul rand sau coloana in sensul opus deplasarii (ca sa nu avem un teren din ce in ce mai mare pe masura ce ne deplasam). Adaugarea si stergerea de randuri/coloane e de fapt fictiva pentru ca in final trebuie sa ramanem cu un grid cu acelasi numar de vertecsi si aceeasi dimensiune. Asa ca, practiv vom simula adaugarea si stergerea prin deplasarea terenului cu *d*=dimensiune_celula in sensul miscarii si shiftarea in sens opus a texturii de blend(de amestec) cu 1/nr_celule.

De exemplu, sa presupunem ca ne-am deplasat suficient de mult in dreapta (pe OX sensul pozitiv) astfel incat am depasit fata de centrul curent al terenului pe orizonatala distanta *d*. In acest moment deplasam tot gridul la dreapta, cu acel *d*, dar, ca sa nu observe utilizatorul ca s-a deplasat terenul, mutam in stanga textura de blend cu 1/nr_celule (scazand practic pentru toti vertecsii u cu 1/nr_celule). Daca punem GL_REPEAT si pentru textura de blend, nu vom aveam probleme cu u si v care ies din intervalul [0,1].

Generarea reliefului

Vom folosi blend_map-ul si pe post de height map. Astfel fiecare culoare (rosu, verde, albastru) va reprezenta o anumita inaltime (de exemplu 3,2,-1). Aceste inaltimi vor fi preluate din fisierul de configurare. De exemplu, partea corespunzatoare din xml poate arata asa:

```
<inaltimi>
      <r>3</r>
      <g>2</g>
      <b>-1</b>
</inaltimi>
```

Evident, aplicarea inaltimilor se va face in vertex shader. Se va prelua culoarea din textura pentru fiecare vertex si se va aplica o formula asemanatoare cu cea pentru aplicarea mai multor texturi.

Presupunand ca a_pos este atributul pentru pozitie, formula ar fi:

```
pos_nou = a_pos
pos_nou.y += c_blend.r*height.r+c_blend.g*height.g+c_blend.b*height.b
```

Unde height e un vec3 cu cele 3 inaltimi preluate din xml.