Adaugarea unui foc animat

Pentru foc vom folosi modelul: **fire.nfg**
De asemenea vom folosi texturile:

- **fire_mask.tga** (atentie, trebuie inversata)
- **fire3.tga**
- **DisplacementMap.tga**

Modelul focului are o forma trapezoidala. Pentru a da o forma rotunjita si usor transparenta la marginea flacarilor folosim fire_mask care va reprezenta componenta alpha a culorii focului.

Textura DisplacementMap e folosita pentru a realiza animatia flacarilor. Aceasta va fi deplasata pe verticala cu un anume pas.

Se va transmite catre fragment shader un float de tip uniform, sa ii spunem *u_Time*, care va creste pe masura ce trece timpul de la inceperea aplicatiei (puteti folosi functia *clock()*). Atentie, pasul de crestere ar trebui sa fie mic (sa zicem la nivelul sutimilor, miimilor), pentru ca *u_Time* trebuie sa se afle in intervalul [0,1), iar cand ajunge mai mare sau egal cu 1 sa fie redus la partea fractionara (aceste calcule se fac in programul C++ nu in shader). Variabila *u_Time* va fi folosita pentru a calcula coordonatele de displacement din textura *DisplacementMap*:

disp = texture2D(*nume_textura_displacement* , vec2(v_uv.x, v_uv.y + u_Time)).rg;

Atentie, pentru a calcula u_time se poate folosi si deltaTime insa e un pic problematic (datorita limitarii de timp, pe care ati adaugat-o la inceput, aplicate update-ului din framework – adica sa nu se realizeze operatiile la intervale mai mici de timp decat limita data).

Componentele r si g (echivalene cu x,y) vor reprezenta displacement-ul pe orizontala si respectiv verticala, a texturii.

Se va transmite si un uniform (tot float) *u_DispMax*, care va da capetele inervalului de displacement: [-u_DispMax, u_DispMax]. Practic, *disp* trebuie raportat de la intervalul [0,1] la acest interval. Valoarea obtinuta va fi notata: *offset* si se va aduna la v_uv, obtinand v_uv_displaced.

Se va prelua apoi culoarea din textura focului, insa cu coordonatele v_uv_displaced, si se va obtine *c_fire*.

Pentru a pastra forma focului, in cazul lui fire_mask se vor folosi coordonatele v_uv

initiale, obtinandu-se *c_alpha*.

Apoi alpha-ul lui c_fire se va inmulti cu oricare din componentele r,g sau b ale lui *c_alpha* (fiind o nuanta de gri, componentele r,g,b sunt egale).

Culoarea obtinuta la final cu alpha-ul modificat va fi transmisa lui gl_FragColor.

Nu uitati in fisierul cpp sa activati blendul:

glEnable (GL_BLEND);
glBlendFunc (GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);// pentru o aplicare corecta peste background (celelalte obiecte desenate)

Obsevatie: In fisierul de configurare, pentru foc, se va adauga un element cu valoarea pe care trebuie sa o ia *u_DispMax*.

Pasi de urmat:

Clasa Fire si modificari in xml-uri

1) Se creeaza o clasa numita Fire derivata din SceneObject.
2) In fisierul resourceManager.xml se vor adauga cele 3 texturi (cea de displacement , cea cu focul propriu-zis, si masca de transparenta). Textura de displacement va avea wrap-ul repeat, pentru a se repeta in continuu miscarea focului. In sceneManager.xml se defineste un obiect nou de tip fire. Se va modifica parsarea XML-ului astfel incat sa creeze un obicet din clasa Fire (care se va adauga in map-ul de obiecte). In XML se va adauga proprietatea dispMax a focului.
3) In clasa se vor adauga proprietatile u_DispMax (float) si u_Time.
4) In Update-ul focului se va prelua timpul curent si se va pune in u_Time (se poate folosi si functia clock() pentru asta – chiar daca reprezinta numarul de clock ticks ale procesorului si nu un timp in (mili)secunde). Se va imparti la un factor de micsorare (astfel incat diferenta intre valorile succesive din doua update-uri sa fie un numar subunitar, eventual chiar mai mic de 0.1 altfel textura se va misca prea repede si chiar un pic sacadat). In functia Draw a focului, se vor transmite timpul curent catre shader si u_DispMax (valorile sunt aceleasi pentru toate fragmentele, deci vor fi de tip uniform).
5) Deoarece focul are zone semi-transparente, vom seta un blending function in init-ul framework-ului:
   glEnable (GL_BLEND);
   glBlendFunc (GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);

Lucrul in shadere

1) Se porneste de la shaderul pentru obiectele obisnuite (se copiaza cele doua shadere si se denumesc, de exemplu: fire.vs si fire.fs).
2) In fragment shader se preia culoarea din textura displacement map folosind coorodonatele (v_uv.x, v_uv.y+u_Time), astfel “mutand” textura pe verticala si obtinand la fiecare frame alt dispalcement. Rezultatul e un vector cu 4 componente (o culoare)

dar noi avem nevoie doar de doua componente pe care sa le adunam la coordonatele de textura initiale. Astfel ca vom lua doar primele doua (.rg sau .xy, cele doua fiind echivalente).
vec2 disp = texture2D(nume_textura_displacement, vec2(v_uv.x, v_uv.y + u_Time)).rg;

3) Cele doua componente ale lui disp, fiind componente de culoare sunt in intervalul [0,1], dar noi dorim sa le aducem in [-u_DispMax, u_DispMax]. Astfel, inmultim disp cu 2 sa il aducem in intervalul [0,2], scadem 1 sa ajunga in [-1,1] si inmultim cu u_DispMax ca sa ajunga in [-u_DispMax, u_DispMax]. Punem rezultatul in *vec2 offset*.
4) Obtinem coordonatele deplasate adunand *offset* la v_uv-ul initial:
vec2 v_uv_displaced=v_uv+offset;
5) In variabila, de tip *vec4*, c_fire, vom pune culoarea obtinuta din textura cu focul de la coorodonatele v_uv_displaced (folosind functia texture2D)
6) De la coodonatele v_uv initiale, din textura alpha, luam culoarea pentru fragmentul curent si o punem in c_alpha
7) Schimbam alpha-ul lui c_fire (componenta .a) inmultind-o cu oricare dintre componentele r,g,b ale lui c_alpha (cele 3 componente sunt egale fiind vorba de o nuanta de gri).
8) Setam gl_FragColor sa fie egala cu c_fire (cu alpha-ul schimbat)