Tema 3 - Cappadocian Balloons

- **Responsabili:** Andrei Lăpușteanu, Vlad-Matei Drăghici, Mihnea-Petruţ-Ilie Mitrache
- **Lansare:** 15 decembrie 2024
- **Termen de predare:** 14 ianuarie 2025, ora 23:59
- **Regulament:** Regulament general
- **Notă:** Această temă este considerată **temă suplimentară**
- **Orice informație ce nu a fost acoperită în acest document este la latitudinea voastră!**

În cadrul acestei teme veți implementa o scenă virtuală ce face referire la faimoasele baloane cu aer cald cappadociene (puteți vizualiza aici o imagine de referință) - se vor studia astfel conceptele de iluminare și texturare. Elementele vizuale principale ce compun scena vor fi un teren texturat și modificat în înălțime printr-un shader, baloanele cu aer cald, precum și un soare la asfințit.

Puteți studia în următorul videoclip o posibilă implementare a cerințelor.

Scena virtuală

Scena va fi compusă dintr-o serie de modele 3D, anume:

- **Terenul**: Format dintr-un plan (construcția acestuia este specificată în capitolele următoare)
- **Baloane**: Formate din mai multe primitive (construcția acestora este specificată în capitolele următoare)
- **Soarele**: Format dintr-o sferă

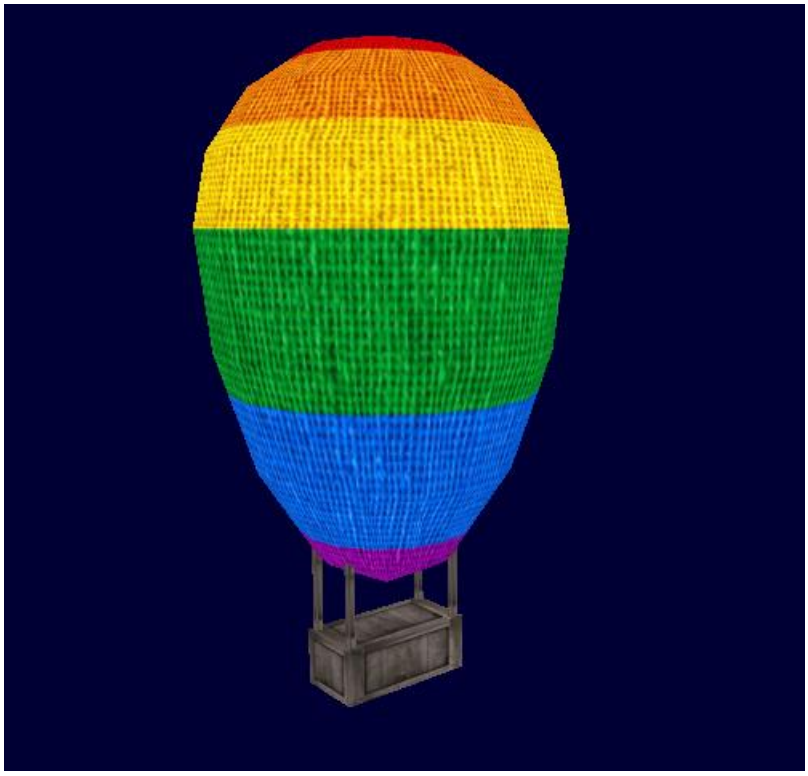
Poziționarea camerei

Camera va fi de tip perspectivă și trebuie plasată astfel încât să includă în cadru toate elementele de interes din scenă (terenul, baloanele și soarele). Nu este necesar să permiteți controlul poziției sau rotației camerei în timpul execuției aplicației, aceasta putând rămâne fixă.

Construcția baloanelor

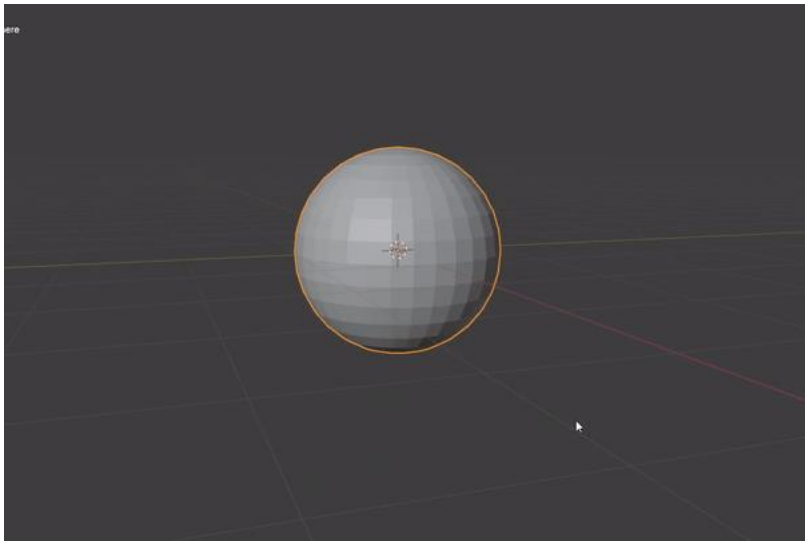
Baloanele din scenă sunt construite folosind o serie de primitive. Varianta recomandată, care este utilizată și în demo, are următoarele componente:

1. Nacela – paralelipiped
2. Corp balon – sferă (cu deformarea propusă mai jos)
3. Conectoare – 4 paralelipipede



Deformare în Vertex Shader

Pentru a obține un rezultat apropiat de aspectul real al corpului unui balon cu aer cald, este necesar să aplicăm o deformare simplă de alungire unei sfere. Procesul presupune deplasamentul programatic, în Vertex Shader, al vârfurilor din jumătatea inferioară a sferei pe verticală. O reprezentare vizuală a rezultatului pe care îl veți obține este evidențiată mai jos:



Veți face această deformare raportându-vă la coordonatele obiect ale sferei. O simplă verificare a modelului disponibil în framework (sphere.obj) vă va arăta că acesta este definit în origine și are rază 1. Vârfurile care sunt mutate au coordonata

y intre [0, -0.5] în coordonate obiect. Observați că referința realizează un deplasament diferit în funcție de această valoare 😊.

Texturare

Fiecare element constitutiv al unui balon se va textura. Pentru nacelă și conectoare puteți să folosiți aceeași textură pentru toate baloanele. Corpul trebuie să fie colorat prin alegere aleatorie dintr-un set de texturi. Este obligatoriu ca acesta să fie texturat diferit față de restul componentelor și ca alegerea texturii să se facă dintr-un set de minim 5. Un exemplu este următorul:

Comportamentul baloanelor

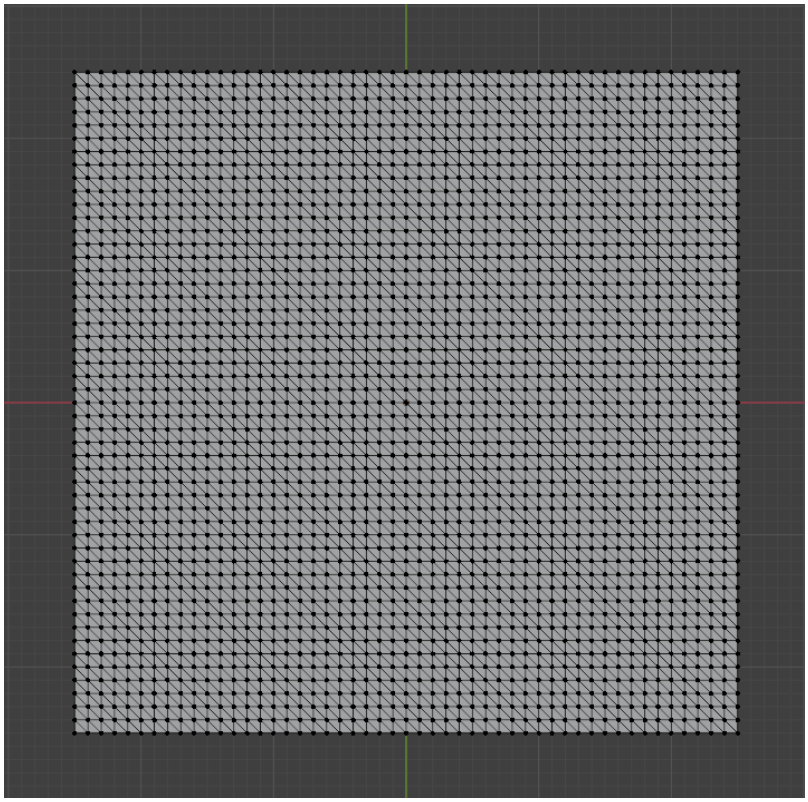
Baloanele din scenă trebuie să se rotească pe traiectorii concentrice. Traiectoriile sunt sub formă de cercuri paralele cu planul XOZXOZ și cu centrul comun. Pentru acest lucru trebuie să alegeți un punct din planul XOZXOZ, CC (de exemplu (0,0)(0,0)), pentru a desemna centrul cercurilor.

Pentru fiecare balon trebuie să se aleagă (de preferat aleator) o înălțime HH la care să se rotească (astfel centrul traiectoriei va fi (C.x,H,C.y)(C.x,H,C.y)), o rază RR a cercului care descrie traiectoria și o viteză de rotație ωω. Rezultatul va fi o serie de traiectorii concentrice paralele cu XOZXOZ și paralele între ele, la înălțimi diferite față de pământ.

În plus, pentru un efect mai realistic, fiecare balon va avea o oscilație pe axa OYOY de forma Δy=A·sin(ω·Δt)Δy=A·sin(ω·Δt) (unde AA și ωω se vor alege astfel încât efectul vizual să pară realistic).

Construcția terenului

Terenul este inițial plat pentru ca mai apoi înălțimea vârfurilor sale să fie modificată pe GPU, cum este explicat în secțiunea de mai jos. Pentru ca această modificare să fie posibilă, este necesar ca acesta să fie subdivizat.



Un astfel de model, ce are 256×256 de quad-uri, se găsește aici: plane256.zip

Deformare în Vertex Shader folosind o hartă de înălțime

Veți realiza deformarea în Vertex Shader luând informația de înălțime din textură, la care veți aplica un factor de intensificare constant în funcție de rezultatul dorit.

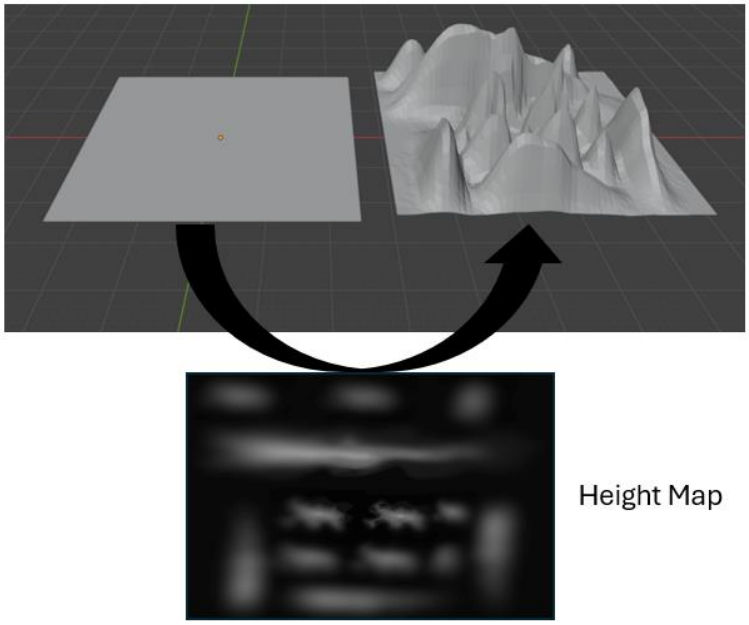
// GLSL code

const float Y_OFFSET = 0.5;

// Get vertex height from the height map

new_position.y = Y_OFFSET * texture(texture_1, v_texture_coord).r;

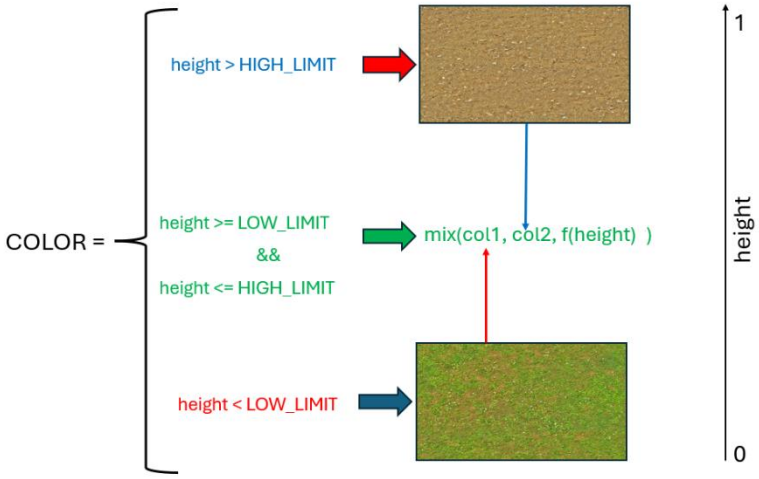
Rezultatul pe care îl veți obține este următorul:



Puteți prelua harta de înălțimi folosită în referință, de rezoluție 256×256, de aici: heightmap256.zip

Texturare

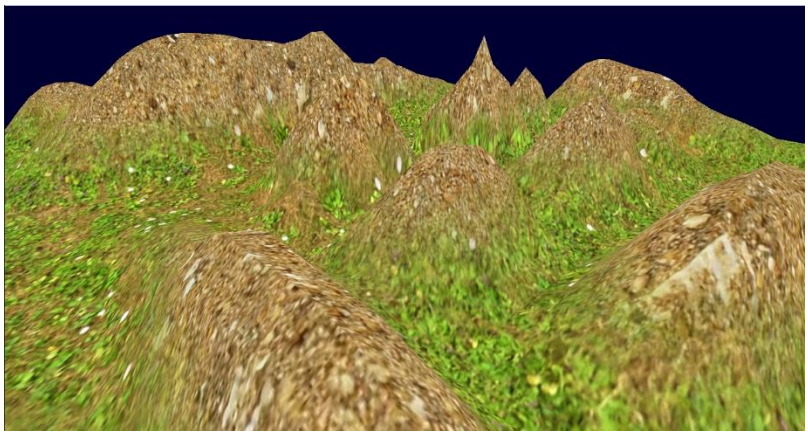
Terenul, reprezentat de planul subdivizat, se va textura folosind coordonatele sale de textură. Pentru a obține un efect interesant vă veți folosi de două texturi de culoare, împreună cu textura de adâncime. Fiecare fragment se colorează în funcție de înălțime. Pentru o trecere uniformă este necesară interpolarea liniară în zonele de înălțime medie. Ideea este evidențiată de diagrama de mai jos:



Funcția care indică gradul de interpolare trebuie să normalizeze intervalul de mijloc [LOW_LIMIT,HIGH_LIMIT][LOW_LIMIT,HIGH_LIMIT] în [0,1][0,1]. Aceasta poate să arate așa:

f(height)=height−LOW_LIMITHIGH_LIMIT−LOW_LIMITf(height)=height−LOW_LIMIT HIGH_LIMIT−LOW_LIMIT

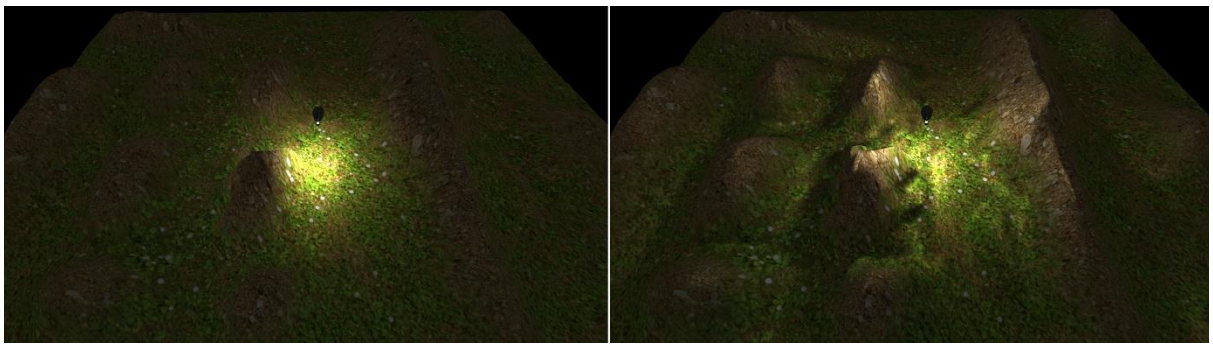
Rezultatul pe care trebuie să îl obțineți este următorul:



Recalculare normale

Deformarea aplicată în vertex shader se aplică asupra poziției fiecărui vertex, urmărind metodolgia prezentată mai sus. Totuși, amintiți-vă de formulele de la calculele de iluminare - acestea se bazează pe vectorul normală la suprafață ($\vec{N}$ N→), care a rămas neschimbat în urma deplasării vertecșilor, ceea ce conduce la un rezultat eronat al iluminării.

În imaginea următoare este desenat terenul deformat și o lumină punctiformă - observați diferențele între imagini, in partea stângă normalele planului sunt toate orientate către (0,1,0)(0,1,0), iar imaginea din dreapta ilustrează rezultatul după recalcularea normalelor.



La nivel conceptual, ne vom folosi de metoda diferențelor finite pentru a aproxima aceste normale. Pentru coordonatele de texturare ale fiecărui vertex vom eșantiona harta de înălțimi, unde ne vom folosi de texelii vecini pentru a determina aceste normale.

Pentru a calcula texelSizetexelSize, presupunem că harta de înălțimi este o imagine pătratică, iar dimensiunea sa (rezoluția) este dimdim. Texelul reprezintă o unitate discretă a texturii, iar dimensiunea sa relativă în coordonate UV se poate calcula astfel:

texelSize=1dimtexelSize=1dim

Se eșantionează textura suport (harta de înălțimi) pentru a determina “înălțimea” texelilor (în esență, se identifică valoarea de luminozitate a texelului, pe care o notăm cu (hh)). Se folosesc coordonatele de textură (vtexCoordvtexCoord) pentru a extrage valoarea înălțimii corespunzătoare (hh), dar și valorile de înălțime ale vecinilor acestuia, de-a lungul axelor X și Z.

h=texture(heightMap,vtexCoord).rhright=texture(heightMap,vtexCoord+2$^{\rightarrow}$ (texelSize, 0)).rhup=texture(heightMap,vtexCoord+2$^{\rightarrow}$ (0,texelSize)).rh=texture(heightMap,vtexCoord).rhright=texture(heightMap,vtexCoord+2→(texelSize,0)).rhup=texture(heightMap,vtexCoord+2→(0,texelSize)).r

Următorul pas este calcularea gradientelor pe direcțiile X și Z. Factorul de scalare pe verticală (yoffset)(yoffset) reprezintă valoarea utilizată anterior pentru ajustarea înălțimii terenului pe baza hărții de înălțimi:

Δx=(hright−h)yoffsetΔz=(hup−h)yoffsetΔx=(hright−h)yoffsetΔz=(hup−h)yoffset

Pe baza acestor variații, construim doi vectori, tangenta și bitangenta în **spațiul obiectului**:

T$^{\rightarrow}$ OS=3$^{\rightarrow}$ (texelSize,Δx,0)B$^{\rightarrow}$ OS=3$^{\rightarrow}$ (0,Δz,texelSize)T→OS=3→(texelSize,Δx,0)B→ OS=3→(0,Δz,texelSize)

Introducem texelSizetexelSize în acești vectori pentru a ne asigura că distanțele orizontale corespund exact dimensiunii unui texel în spațiul obiectului, astfel rezultatul nu depinde de modul în care scalăm terenul.

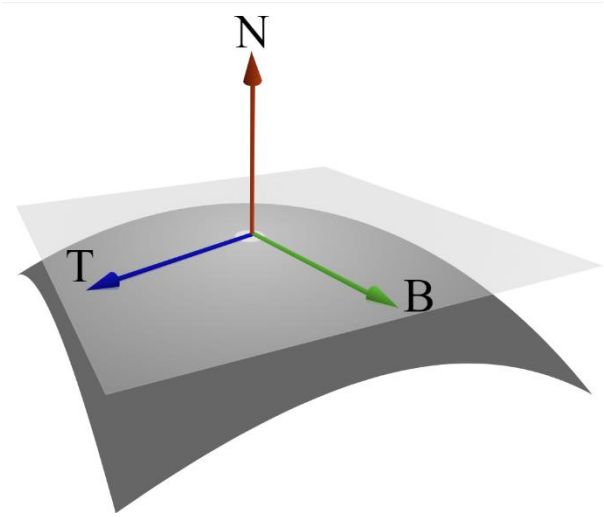
Aplicăm matricea de modelare pentru a obține vectorii în **spațiul lumii**:

T$^{\rightarrow}$ WS=(Model·4$^{\rightarrow}$ (T$^{\rightarrow}$ OS,0))xyzB$^{\rightarrow}$ WS=(Model·4$^{\rightarrow}$ (B$^{\rightarrow}$ OS,0))xyzT→WS=(Model·4→(T→OS,0))xyzB→WS=(Model·4→(B→OS,0))xyz

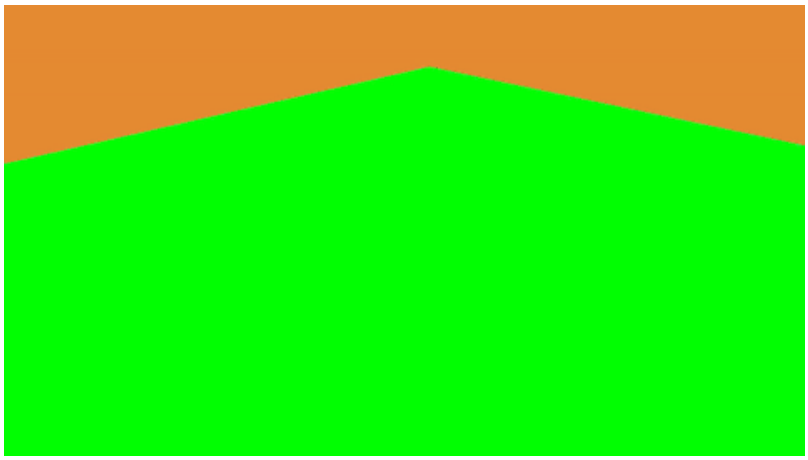
Prin trecerea de
la (T$^{\rightarrow}$ OS,B$^{\rightarrow}$ OS)(T→OS,B→OS) la (T$^{\rightarrow}$ WS,B$^{\rightarrow}$ WS)(T→WS,B→WS), garantăm că direcțiile rezultate țin cont de toate transformările de modelare aplicate terenului, asigurându-ne astfel că normala finală (N$^{\rightarrow}$)(N→), obținută prin produsul vectorial dintre B$^{\rightarrow}$ WSB→WS și T$^{\rightarrow}$ WST→WS, reflectă corect forma terenului în spațiul lumii.

N$^{\rightarrow}$ =normalize(cross(B$^{\rightarrow}$ WS,T$^{\rightarrow}$ WS))N→=normalize(cross(B→WS,T→WS))

Imaginea următoare oferă suport vizual pentru vectori și rezultatul produsului vectorial.



În final, animația de mai jos ilustrează cum sunt normalele recalculate în funcție de înălțimea terenului, precum și rezultatul obținut în urma adăugării unei lumini direcționale.



Animația continuă a înălțimii terenului din GIF-ul anterior este pur ilustrativă, nu este o cerință impusă.

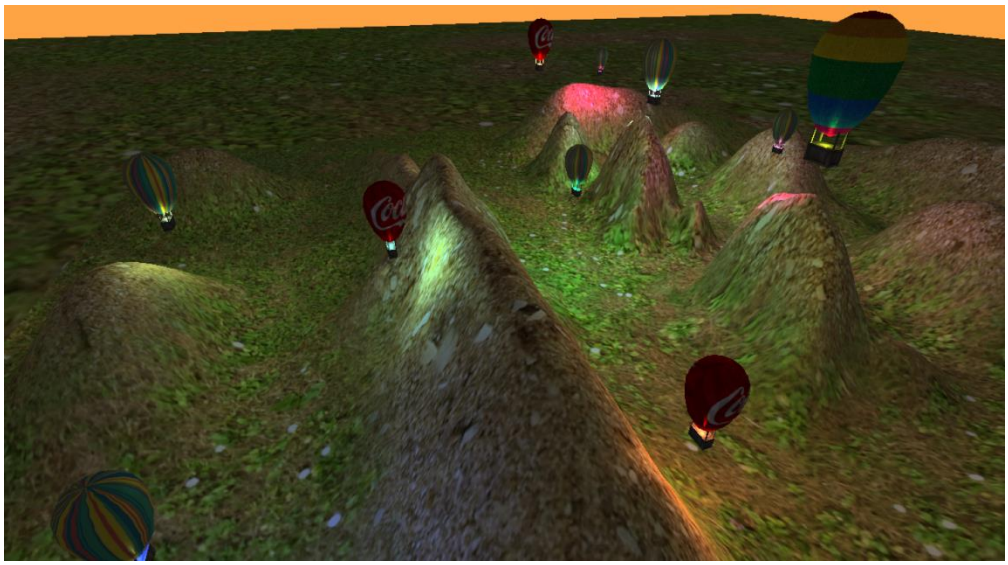
Iluminare

Iluminarea scenei se va realiza folosind cel puțin 2 tipuri de surse de lumină: **punctiformă** și **direcțională**. Fiecare sursă de lumină (indiferent de tipul acesteia) o să aibă o culoare specifică și trebuie să se țină cont de această culoare în calculele de iluminare.

Iluminarea trebuie să fie implementată folosind **modelul de shading Phong**, precum și **modelul de calcul al reflexiei luminii Phong**.

Lumina punctiformă: Aceasta este reprezentată de sursele care emit lumina cu aceeași intensitate in toate direcțiile. Acest tip de lumină trebuie implementat pentru **fiecare balon cu aer cald**: Lumina trebuiă să ramână la o poziție fixă relativ la balon și să aibă o culoare și intensitate luminoasă aleatorie.

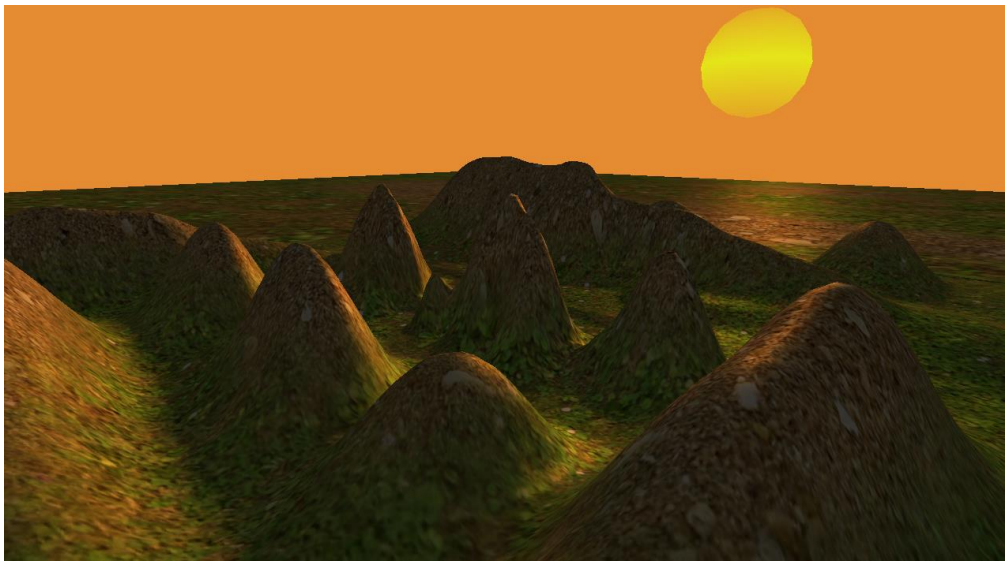
În următoarea imagine au fost activate numai luminile punctiforme:



Implementarea voastră trebuie să suporte **randarea mai multor surse de lumină în același frame!**

Lumina direcțională: aceasta va ilumina toate obiectele din scenă cu aceeași intensitate. Specific luminii de tip direcțional este faptul că vectorul luminii incidente LL nu depinde de poziția luminii sau a fragmentului care trebuie iluminat (precum în cazul luminilor de tip point și spot). Așadar, pentru fiecare fragment, iluminarea va fi calculată folosind același vector LL (corespunzător direcției luminii). Astfel, pentru o sursă de lumină de tip direcțional este nevoie să se definească direcția acesteia și culoarea luminii emise. În cadrul acestei teme vom considera soarele ca `sursa` acestei lumini direcționale, așadar, va trebui să setați direcția acestei lumini în mod corespunzător.

În următoarea imagine a fost activată numai lumina direcțională:



Barem [150p]

- **Construcția baloanelor [30p]**
  - Asamblarea componentelor [7.5p]
  - Deformarea balonului in Vertex Shader [15p]
  - Texturarea [7.5p]
- **Dinamica baloanelor [20p]**
  - Randarea a minim 5 baloane [5p]
  - Deplasarea baloanelor (fiecare cu rază diferită) în jurul unui punct [15p]
- **Construcția terenului [60p]**
  - Desenarea terenului (plan) [5p]
  - Deformarea în Vertex Shader folosind o hartă de înălțime [10p]
  - Texturarea [15p]
    - Folosirea a minim 2 texturi [10p]
    - Interpolarea între texturi pe baza înălțimii [5p]
  - Recalcularea normalelor [30p]
- **Iluminarea [40p]**
  - Lumină direcțională [15p]
    - Desenarea modelului de soare (cu o culoare uniformă) [5p]
    - Calcule pentru iluminarea direcțională [10p]
  - Lumini punctiforme [25p]
    - Câte o lumină de culoare și intensitate aleatorie pentru fiecare balon [25p]

Exemple de Funcționalități Bonus

- Unele baloane să aibă o traiectorie definită de un set de puncte generate aleatoriu și să se interpoleze poziția acestora între ele.
- Pe un balon să se simuleze efectul gravitației și efectul unui vânt cu o forță aleatorie care se schimbă în timp. În plus, la apăsarea unei taste balonul să primească o accelerație pe axa OY (se intensifică focul din balon).
- Să se genereze la fiecare rulare o hartă de înălțime diferită care să se adune la cea de bază pentru a avea variații ale terenului la fiecare rulare.

Întrebări și răspunsuri

Pentru întrebări vom folosi forumurile de pe Moodle. Orice nu este menționat în temă este la latitudinea fiecărui student!

Notare

Baremul este orientativ. Fiecare asistent are o anumită libertate în evaluarea temelor (de exemplu, să dea punctaj parțial pentru implementarea incompletă a unei funcționalități sau să scadă pentru hard coding). Același lucru este valabil atât pentru funcționalitățile obligatorii, cât și pentru bonusuri.

Tema trebuie încărcată pe Moodle. Pentru a fi punctată, tema trebuie prezentată la laborator. Vor exista laboratoare speciale de prezentare a temelor (care vor fi anunțate).

Indicații suplimentare

Tema va fi implementată în OpenGL și C++. Este indicat să folosiți framework-ul și Visual Studio.

Pentru implementarea temei, în folderul **src/lab_m1** puteți crea un nou folder, de exemplu **Tema3**, cu fișierele **Tema3.cpp** și **Tema3.h** (pentru implementare POO, este indicat să aveți și alte fișiere). Pentru a vedea fișierele nou create în Visual Studio în Solution Explorer, apăsați click dreapta pe filtrul **lab_m1** și selectați **Add→New Filter**. După ce creați un nou filtru, de exemplu **Tema3**, dați click dreapta și selectați **Add→Existing Item**. Astfel adăugați toate fișierele din folderul nou creat. În fișierul **lab_list.h** trebuie adăugată și calea către header-ul temei. De exemplu: **#include "lab_m1/Tema3/Tema3.h"**

Arhivarea Proiectului

- În mod normal arhiva trebuie să conțină toate resursele necesare compilării și rulării
- Înainte de a face arhiva asigurați-vă că ați curățat proiectul Visual Studio:
    - Click dreapta pe proiect în **Solution Explorer → Clean Solution**
    - Ștergeți folderul **/build/.vs** (dacă nu îl vedeți, **este posibil să fie ascuns**)
- În cazul în care arhiva tot depășește limita de 50MB (nu ar trebui), puteți să ștergeți și folderul **/deps** sau **/assets** întrucât se pot adăuga la testare. Nu este recomandat să faceți acest lucru întrucât îngreunează mult testarea în cazul în care versiunea curentă a bibliotecilor/resurselor diferă de versiunea utilizată la momentul scrierii temei.