Escola Politècnica Superior d'Enginyeria de Vilanova i la Geltrú
UNIVERSITAT POLITÈCNICA DE CATALUNYA

IMPLEMENTACIÓ D'UN GENERADOR PROCEDURAL DE PLANETES

ORIOL FERNÁNDEZ BRIONES
UNIVERSITAT POLITÈCNICA DE CATALUNYA UPC, DEPARTAMENT DE FÍSICA, AV. DE VÍCTOR BALAGUER, 1, 08800 VILANOVA I LA GELTRÚ, BARCELONA

Resum

Aquest treball consisteix en el desenvolupament d'una aplicació web per visualitzar l'evolució temporal de planetes com la Terra mitjançant gràfics per ordinador. L'objectiu és recrear el desenvolupament que va experimentar des de la seva formació, explorant els diferents mecanismes que l'han portat a ser com el coneixem avui dia. (formació, l'origen de l'aigua, la creació i moviment de les plaques tectòniques, l'atmosfera i clima a escala global). L'objectiu final no és tractar de simular el planeta Terra, ja que és una tasca de gran complexitat. Es vol explorar la idea d'implementar de forma simplificada el que coneixem del nostre i altres planetes per a obtenir resultats aproximats que puguem extrapolar per a poder veure quin aspecte podrien tindre altres planetes amb característiques molt diverses.

Paraules clau: Simulador gràfic; Ombrejador (*Shaders*); Gràfics per ordinador ; Planetes terrestres; Evolució planetària; tectònica de plaques; Erosió hidràulica; Atmosfera; OpenGL; Clima.

1. Introducció

És difícil fer-se una idea de la quantitat de temps que va ser necessari perquè la Terra es converteixi en el que coneixem avui dia (milers de milions d'anys).

Per assimilar aquestes enormes escales de temps, podem comprimir tot aquest període en una simulació d'un o dos minuts, suficient per veure tots aquests fenòmens que han intervingut en el modelat del nostre planeta en funcionament. Això és exactament el que és capaç de fer aquest programa.

Perquè la simulació sigui fluida, s'han de realitzar càlculs molt complexos i costosos múltiples vegades cada segon. Aquí és on s'aprofita la capacitat computacional de les targetes gràfiques que incorporen els dispositius moderns. Els càlculs que es fan es poden distribuir en els milers de nuclis que tenen aquestes targetes, ja que estan dissenyades per aquest tipus de càrrega de treball, a diferència de les CPUs.

El programa ofereix la possibilitat d'interaccionar i mostrar de forma molt visual el que es coneix respecte a l'evolució dels planetes i els processos que intervenen. Només s'explora una part molt superficial, pel fet que molts són conceptes complexos amb molta informació, però s'ha tractat de focalitzar el contingut al màxim per obtenir els millors resultats amb una gran simplificació en el temps disponible.

Aquesta idea va sorgir del treball en el qual ha treballat en els últims anys David A. Roberts[31], desenvolupador de software i artista digital. Té una gran experiència en el món del *shaders* i ha anat desenvolupant simulacions físiques de tot tipus i també d'aquests fenòmens que finalment va unificar en una única simulació de la Terra molt visual.

A continuació es presenten els capítols on es parla dels esdeveniments a través del temps que van portar a cada un dels aspectes que volem simular. Per a cada un, es dona una introducció teòrica on s'expliquen les bases que es volen simular per després passar a la part pràctica on s'implementen els conceptes en forma de codi.

El primer capítol servirà per a conèixer l'entorn en el qual es treballa i com està estructurat el projecte. A continuació s'introdueix el primer gran element de la simulació, s'explica la primera etapa de formació del planeta i en acabar tenim una base la qual anem ampliant amb més característiques. Es presenta la formació dels oceans, com va arribar l'aigua a la superfície terrestre i també un element clau a la simulació, el pas del temps i com representar-lo en el programa. Lligat amb l'anterior, s'exploren els efectes que ocasiona l'aigua al planeta i com transforma el seu paisatge amb rius i l'erosió de l'aigua. Seguidament, es presenta l'origen i funcionament de les plaques tectòniques al planeta i com simular la interacció entre elles de forma realista. Continuant amb la formació de l'atmosfera, es detalla el seu funcionament per entendre els fenòmens que causen que el cel tingui el seu color característic i perquè es torna vermell quan el sol està baix en l'horitzó, així com implementar tots aquests càlculs necessaris tan costosos de forma eficient, entre altres coses. Per acabar, es presenta el funcionament del clima a escala global, s'exploren aspectes com la pressió atmosfèrica, els corrents del vent, temperatura, precipitacions i com implementar aquests fenòmens de forma que puguem obtenir resultats prou realistes.

Finalment, es presenten les conclusions a les quals s'ha arribat un cop finalitzat el treball i millores que es podrien implementar per a poder ampliar-lo.

2. Introducció a l'entorn 3D

El motor principal de l'aplicació és Three.js, una biblioteca JavaScript que s'utilitza per crear i mostrar gràfics 3D en un navegador web mitjançant WebGL una altra API de gràfics a baix nivell creada específicament per a web, que alhora deriva de OpenGL[1]. La renderització s'executa a la unitat de processament gràfic (GPU), el que significa que el rendiment en temps d'execució és molt bo. Un altre pilar molt important són els shaders. Es tracta d'un conjunt d'instruccions que s'executen alhora per a cada píxel de la pantalla. Això vol dir que el codi que s'escriu s'ha de comportar de manera diferent

segons la posició del píxel a la pantalla. El programa funciona com una funció que rep una posició i retorna un color, i quan es compila, s'executarà extraordinàriament ràpid. Això vol dir que en una pantalla antiga de 800 x 600, s'han de processar 480.000 píxels per fotograma, la qual cosa significa 14.400.000 càlculs per segon. Aquest és un problema prou gran per a sobrecarregar un microprocessador. Aquí és on entra el processament paral·lel. En lloc de tenir un parell de microprocessadors, és més intel·ligent tenir molts microprocessadors petits que funcionin en paral·lel al mateix temps. Això és el que és una unitat de processador gràfic (GPU). Aquestes també treballen molt be amb les funcions matemàtiques especials accelerades mitjançant maquinari, de manera que les operacions matemàtiques complicades es resolen directament amb el *hardware* en comptes de amb el programari. Això significa que les operacions trigonomètriques i matricials seran molt ràpides.

Ara que s'han introduït les tecnologies informàtiques que es faran servir, es pot veure com començar amb l'entorn de treball sobre el qual es treballa i amplia en cada apartat. Es parteix d'una escena simple amb una càmera, un objecte *renderer*, que s'encarrega de renderitzar els elements en pantalla amb *WebGL*, i una geometria esfèrica senzilla que serveix per simular el planeta. Per visualitzar aquesta escena es necessària una pàgina web on renderitzar-la. Per tant, cal un fitxer HTML que contingui, entre altres coses, un element on es carregui l'escena, que és el que es veurà per pantalla.

3. Període de formació inicial

Els cossos del sistema solar es van formar fa més de 4.600 milions d'anys a partir d'una barreja de gasos, principalment hidrogen, barrejat amb petites quantitats d'elements més pesats formats a les estrelles més antigues, i una mica de pols addicional. La contracció gravitatòria d'aquesta massa i la formació del disc protoplanetari a causa de la conservació del moment angular van ser els primers passos en la formació del sistema. Mentre que la major part de la massa es concentrava al centre de gravetat del sistema, formant el Sol, aproximadament un 1% va romandre al disc, convertint-se en la matèria primera per formar els planetes, satèl·lits i altres cossos menors.[2]. Els planetes terrestres com la Terra es va anar formant gràcies a aquestes col·lisions entre partícules de pols, asteroides i altres planetes en creixement, inclòs un darrer impacte gegant que va llançar prou roca, gas i pols a l'espai per formar la lluna. La Terra primerenca estava coberta per un oceà de magma: una capa de roca fosa de centenars de quilòmetres de profunditat.[3] Per a replicar com era el planeta al seu període inicial, la primera tasca és generar un mapa d'aquest terreny, això ho farem de forma *procedural*, sense cap textura pre-generada. Per a aquesta tasca farem ús de *shaders*, per tant, el primer que cal fer és crear i definir els fitxers. El primer fitxer es el vertex shader, que s'encarrega d'aplicar el codi a cada vèrtex d'una malla de la nostra geometria. L'altre fitxer es el fragment *shader*, que com el nom indica, s'aplica a cada fragment de la malla y conseqüentment per a cada píxel que cobreix aquesta malla. El que farem serà dibuixar sobre la nostra geometria amb aquest *shader*. Els mateixos cràters es generen mitjançant una variació d'un tipus de soroll anomenat soroll cel·lular (o worley). Aquest es basa en els diagrames de Voronoi [4], que per definició es tracta de dividir l'espai en regions cel·lulars on tots els punts de cada regió estan més a prop del seu punt definitiu que qualsevol altre punt[5]. El soroll cel·lular es basa en la distància al punt més proper d'un conjunt de punts característics. Amb aquesta idea com a base, creem una funció que ens permeti obtenir una distribució de cràters realista. Per evitar una regularitat visible, els centres del cràter reben un desplaçament pseudoaleatori dels punts de la quadrícula, utilitzant una funció *hash*. Per calcular la influència d'un cràter en un lloc donat, es fa una mitjana ponderada dels cràters que pertanyen als punts de la quadrícula propers, amb pesos que disminueixen exponencialment amb la distància des del centre. El shader itera sobre múltiples nivells de detall, superposant cràters de mida decreixent els uns sobre els altres per generar una varietat de mides.

En aplicar aquest material a una esfera, tal com estan posicionats els vèrtexs, en projectar aquest mapa en una projecció cilíndrica, estira la textura en els seus pols, resultant en una deformació. Per a solucionar-ho traduïm les coordenades 2D (vUv) del nostre *shader* a coordenades cartesianes 3D, que ens donaran una latitud i una longitud determinades a l'esfera:

```
float lat = 180. * vUv.y - 90.;
float lon = 360. * vUv.x;
return vec3( sin(lon*PI/180.) * cos(lat*PI/180.),
sin(lat*PI/180.), cos(lon*PI/180.) * cos(lat*PI/180.));
```

Perquè els cràters tinguin un aspecte accidentat realista, fem una barreja amb un soroll artificial FBM (*Fractional Brownian Motion*) i s'escala de manera que els cràters més grans tinguin el màxim impacte sobre el terreny. El resultat és un mapa en escala de grisos amb valors del 0 (negre) al 255 (blanc) deformat a les porcions superiors i inferiors per adaptar-se a una projecció esfèrica. Aquest mapa projectat una esfera no mostra cap imperfecció a la superfície.

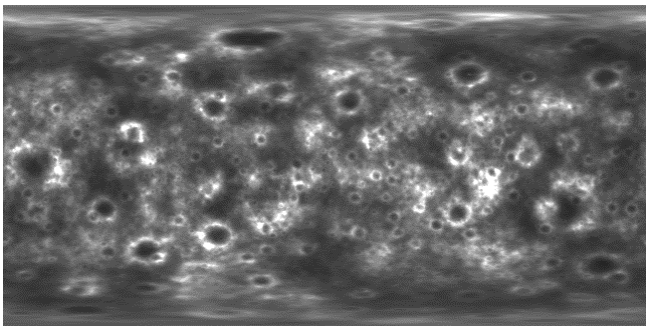

Fig. 1. *Mapa generat proceduralment a partir de shaders. Font: pròpia*

És una aproximació al resultat desitjat, però un planeta no és una esfera perfectament llisa. Three.js permet treballar amb una varietat de mapes que permeten aplicar una sèrie de propietats al material. Fins ara s'ha vist com crear el que seria un mapa de textura, però existeixen més, aquests són alguns a tindre en compte:

bumpMap: La textura per crear un mapa de desnivells. Els valors de blanc i negre es corresponen amb la profunditat percebuda en relació amb les llums. En realitat no afecta la geometria de l'objecte, només la il·luminació.

displacementMap: El mapa de desplaçament afecta la posició dels vèrtexs de la malla. A diferència d'altres mapes que només afecten la llum i l'ombra del material, els vèrtexs desplaçats poden projectar ombres, bloquejar altres objectes i actuar com a geometria real.

normalMap: Els valors RGB afecten la superfície normal de cada fragment de píxel i canvien la manera com s'il·lumina el color. Els mapes normals no canvien la forma real de la superfície, només la il·luminació.

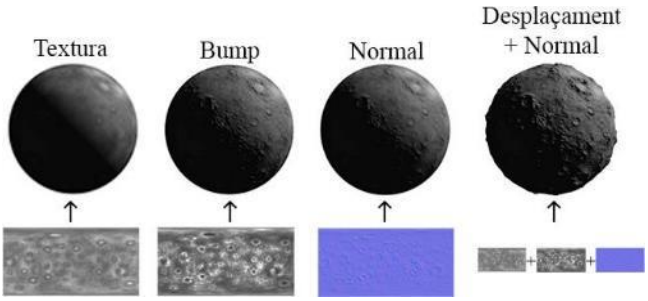

Fig. 2. *Comparació dels tipus de mapes aplicats al planeta. Font: pròpia*

Aquests mapes són una manera fantàstica d'afegir detalls i realisme a un objecte renderitzat. Normalment, per afegir petits detalls, s'han de modelar amb molts polígons petits. Això redueix el rendiment, en canvi, s'aconsegueix el mateix efecte fent una mica de trampes. Això permet que el planeta resultant sigui semblant a com es pensa que va ser la Terra en el seu període de formació inicial. Com que llavors les temperatures eren encara molt elevades, podem afegir una tonalitat vermella al resultat com a una primera aproximació per tractar de representar aquesta etapa.

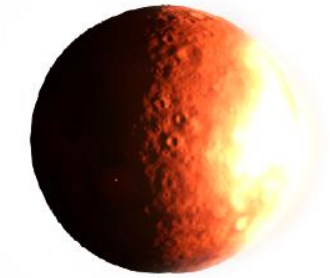

Fig. 3. *Resultat amb aparença semblant a la Terra jove. Font: pròpia*

4. Oceans i evolució temporal

Durant els pròxims centenars de milions d'anys (4500 a 3900)[6], l'oceà de magma es va refredar prou per formar una superfície sòlida. Diversos processos van ocórrer en aquest període, entre ells, l'aparició de l'aigua repartida pel planeta en forma d'oceans[3].

Es creu que part d'aquesta aigua va ser lliurada pels cometes i meteorits que s'estavellaven a la superfície[3] i la resta és aigua que va quedar atrapada durant la formació del planeta.[7] Gràcies a la baixada de temperatura de la Terra, es van començar a formar núvols que finalment dipositaven encara més aigua a la superfície[8].

Per introduir aquests oceans en el programa, caldrà modificar els fitxers de *shaders*, ja que es parteix del terreny que s'ha generat. S'aprofita el mateix mapa de textura per obtenir un altre mapa en escala de grisos que es fa servir per determinar l'elevació del terreny.

El primer que cal fer és crear un altre fragment *shader* que gestioni tots els altres *shaders* actuals, ja que el programa s'està complicant i caldrà treballar amb buffers.

Els buffers són objectes d'OpenGL que permeten guardar i recuperar dades. Això és útil per poder executar múltiples *shaders* diferents i permetre la comunicació entre ells.

Com que el que s'envien són resultats d'execucions dels *shaders*, és a dir, una textura, cal fer ús de la funció que ens proporciona GLSL anomenada *texture*. Això permet obtenir el valor de la textura lligada al buffer que s'indiqui en la crida a la funció. Recordi que aquests *shaders* s'executen per cada píxel de la pantalla i p representa el píxel actual, per tant, obtenim el píxel corresponent en el buffer indicat.

```
#define buf(p) texture(iChannel0,(p)/uResolution.xy)
```

Com que aquesta funció es farà servir més vegades, és bona idea referenciar-la sota un nom com buf per simplificar el codi. En fer la crida de la funció, es rep un vector de 4 dimensions. Per guardar el mapa d'altura, al *shader* de generació del terreny, es fa ús de la dimensió z del vector de retorn.

```
gl_FragColor.z = max(gl_FragColor.z, 0.);
```

Ara al *shader* principal es pot recuperar aquest valor amb:

```
float y = buf(p).z;
```

El color de l'aigua es pot definir amb dos colors, un que fa referència al color de l'aigua més profunda, per tant, amb un color més fosc, i un altre per a l'aigua més propera a la costa. El factor que decidirà si mostrar un color més proper a un o a l'altre el defineix el mapa d'altura guardat a la variable y, per tant, es pot fer ús de la funció mix per, com indica el nom, barrejar els colors en funció de l'altura.

```
deepWaterColor = vec4(0.01, 0.02, 0.08, 1);
shallowWaterColor = vec4(0.11, 0.28, 0.51, 1);
ocean = mix(deepWaterColor, shallowWaterColor, y);
```

Per a combinar el mapa de l'oceà i el del terreny, es pot fer tenint en compte el valor d'altura del píxel actual i decidint si dibuixar part del mapa de l'oceà o el mapa del terreny. Aquests valors es poden ajustar per veure el seu impacte al resultat final.

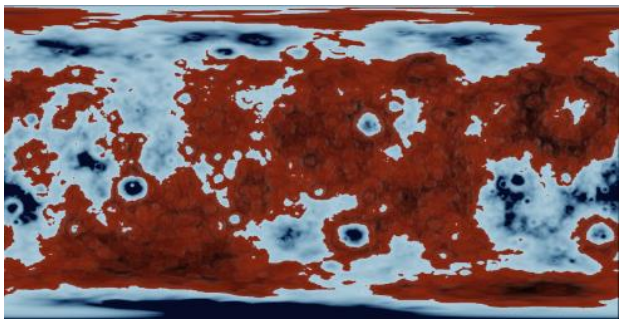

Fig. 4. *Mapa de terreny amb l'oceà. Font: pròpia*

Fins ara, s'ha vist com implementar elements individualment, però per mostrar una evolució en el temps, caldrà introduir en aquests càlculs una variable temporal i

executar en temps real, múltiples vegades per segon, aquests *shaders* per formar una espècie d'animació

Es fa servir una variable que s'anirà incrementant mentre el programa estigui en execució, aquesta s'envia als *shaders* mitjançant un uniform anomenat uTime.

```
if (y < OCEAN_DEPTH && uTime > OCEAN_START_TIME) {
  r = mix(land,ocean,smoothstep(0,2,uTime-OCEAN_START));
} else {
  r = land;
}
```

Per acomodar aquesta transició, es fa ús de la funció *smoothstep*, que el que fa és retornar un valor suavitzat segons la funció definida, en aquest cas en funció del temps. La gràfica que dibuixa és la següent:

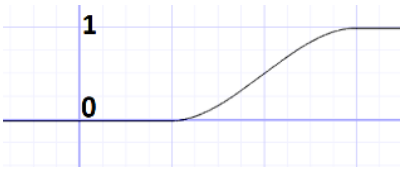

Fig. 5. *Funció smoothstep. Font: pròpia*

Amb això s'aconsegueix una transició suau i homogènia entre els diferents estats de la simulació.

En executar el programa, el planeta comença en el seu estat de formació inicial i a mesura que arriba al temps establert de 15 segons, comencen a aparèixer petits oceans d'aigua que van omplint el planeta. El resultat és el següent:

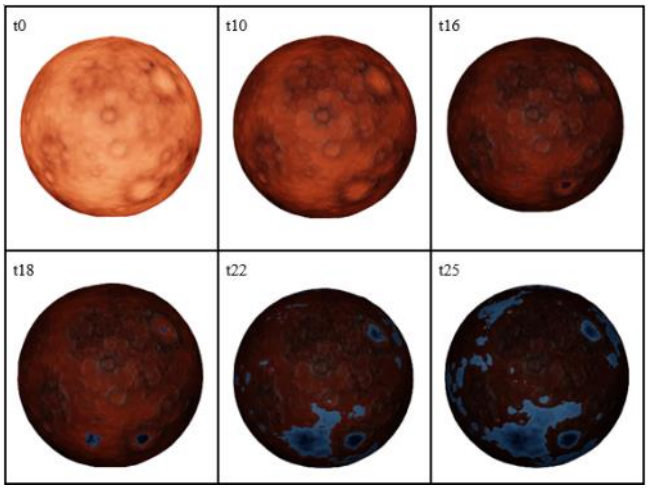

Fig. 6. *Transició del terreny amb oceans en executar el programa. Font: pròpia*

5. Rius i erosió hidràulica

L'erosió hidràulica és un procés geològic pel qual l'aigua transforma el terreny al llarg del temps.[9] Es produeix pel pas de l'aigua en moviment que va retirant material de la terra, és a dir que produeix l'erosió del sòl i el va desgastant a poc a poc, i després diposita aquest material aigües avall.

El material arrossegat pot comprendre des de fines partícules sorrenques a roques de grans dimensions i és un procés que contribueix de forma substancial al modelatge del terreny, tant a la zona de càrrega de material com a la zona de dipòsit. [10]. Hi ha diverses formes d'enfocar el problema quan es tracta de simular l'erosió. Tots els mètodes simulen el mateix fenomen: l'aigua es mou de llocs alts a llocs baixos, erosiona el terreny a mesura que flueix i diposita sediments a mesura que van més enllà dels seus camins.

Aquest procés dona lloc a una sèrie de característiques del terreny reconeixibles, com ara barrancs i valls on flueixen els rius, deltes on es troben amb el seu destí i ventalls al·luvials on els rierols més petits es combinen en rius més grans.

L'objectiu d'aquest mètode és crear entorns creïbles en lloc d'assolir un alt grau de realisme. La fidelitat es pot sacrificar per la velocitat, sempre que els resultats semblin naturals.

Per això cal iniciar el mapa en el qual guardem aquesta informació amb valors aleatoris calculats per una funció de *hash* per aconseguir resultats homogenis al terreny. Fem servir la dimensió w del vector de sortida per aquesta tasca.

```
gl_FragColor.w = hash12(p);
```

La resolució del mapa del terreny és bastant baixa per a un planeta sencer, per tant, el model haurà de ser capaç de simular rius que no tenen més d'un píxel d'amplada. Richard Barnes proposa un model senzill que aconsegueix això en una aportació anomenada acceleració d'un model d'incisió fluvial i d'evolució del paisatge amb paral·lelisme.[11]

Cada píxel examina els seus vuit veïns per determinar quina direcció té la major disminució d'elevació (ajustada pel fet que les diagonals veïnes estan més lluny). Aquesta direcció de major pendent és on viatjarà l'aigua que surt d'aquest píxel. L'aigua primer es distribueix per pluja, i després es transporta entre píxels veïns amb el pas de temps.

Si es troba que no hi ha cap pendent, es dipositen els sediments, i en cas que hi hagi un pendent, es calcula la quantitat d'erosió que ocorre.

Aquest valor el dona la llei de potència del corrent, que prediu la velocitat d'erosió d'un riu al seu llit.

$$E = KA^m S^n \qquad (1)$$

On E és l'erosió causada, A és la quantitat d'aigua en aquest punt i S és el pendent del canal.[12]

Els paràmetres K, m i n no són necessàriament constants, sinó que poden variar en funció de les lleis d'escala suposades, el procés d'erosió, l'erodibilitat de la roca, el clima, el flux de sediments i/o el llindar d'erosió. No obstant això, les observacions de l'escala hidràulica de rius reals indiquen que la relació m/n hauria d'estar al voltant de 0,5[12].

```
elevació -= 0.05 * pow(aigua, 0.8) * pow(pendent, 2.);
```

El resultat del càlcul serà la distancia que cal reduir en aquest punt. Com que aquesta es guarda en la dimensió z, serà finalment on s'apliqui la modificació.

D'aquesta forma tenim l'elevació i la quantitat d'aigua situada al punt actual, juntament amb el pendent en la direcció en què va l'aigua. La disminució d'elevació es limita de manera que no sigui inferior a la ubicació on flueix l'aigua. La interacció entre l'aigua i la pròpia erosió dona lloc a la formació natural de conques fluvials al terreny:

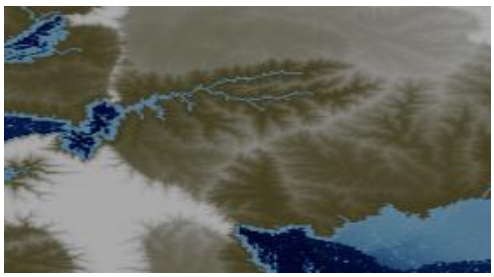

Fig. 7. *Erosió en funcionament. Font: pròpia*

Si s'acoloreixen els canals connectats de forma que el mateix color representi la ubicació de la desembocadura del riu, és possible produir visualitzacions que recorden els mapes de conques fluvials reals:

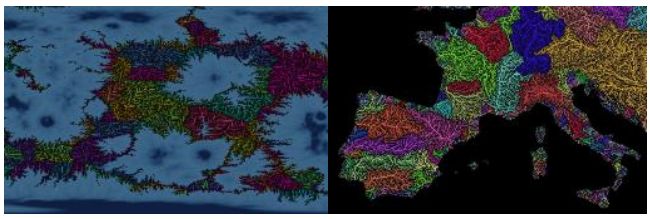

Fig. 8. *Comparació amb conques hidrogràfiques d'Europa. Font: [14]*

6. Model tectònic

Progressivament, l'oceà de magma de la Terra es va refredar prou per formar una superfície sòlida, l'atmosfera terrestre es va començar a formar gràcies a les erupcions volcàniques, així com l'aigua i altres gasos lliurats pels cometes i meteorits que s'estavellaven a la superfície. Aquest va ser el primer pas cap al desenvolupament de les plaques tectòniques del nostre planeta[3].

Aquestes es mouen lentament per la superfície del planeta. Es creu que van començar a moure's fa 3.400 milions d'anys[15]. El moviment relatiu d'aquestes plaques era molt més elevat que en l'actualitat[16], ara per ara oscil·la entre zero i 10 cm anualment[15].

Existeixen tres tipus de interaccions entre plaques:

Divergents: Als oceans, el magma de les profunditats del mantell terrestre s'eleva cap a la superfície i separa dues o més plaques. Muntanyes i volcans s'aixequen al llarg de l'àrea. A terra, es formen canals gegants on les plaques es separen. Si les plaques continuen divergent, poden arribar a separar continents per formar una nova massa terrestre.

Convergents: Allà on les plaques xoquen, l'escorça s'ensorra i es doblega en cadenes muntanyoses. També es produeixen quan una placa oceànica s'enfonsa, en un procés anomenat subducció, sota una massa terrestre. A mesura que la placa superior s'aixeca, també forma serralades.

Transformants: Tenen lloc a zones on dues plaques ni s'ajunten ni se separen sinó que es mouen lateralment, lliscant entre elles. Aquests límits no estan associats a l'aparició de volcans, però sí a terratrèmols.[17]

Per poder simular, la formació de muntanyes, trinxeres oceàniques i formes de terra continentals, cal implementar un model de moviment tectònic que pugui representar aquests fenòmens en el programa.

Primer es generen de forma aleatòria la ubicació d'aquestes plaques, que estan repartides sobre el terreny. Per això es fa servir una funció *hash*. Segons el valor de retorn, es decideix si es crearà una nova placa tectònica en aquest punt i també la quantitat i mida que tindran. En aquest cas es fa servir la dimensió x del vector de sortida per guardar aquest nou mapa. Tots els píxels d'una placa emmagatzemen la velocitat del moviment de la placa.

Aquestes plaques creixen de mida a mesura que passa el temps amb un model d'agregació molt senzill, que selecciona aleatòriament els punts veïns i els afegeix a una placa si encara no s'han assignat a una altra placa.

```
int dir = int(4.*hash13(vec3(p,uFrame)));
switch (dir) {
    case 0: gl_FragColor.x = s.x; break;
    case 1: gl_FragColor.x = w.x; break;
    case 2: gl_FragColor.x = n.x; break;
    case 3: gl_FragColor.x = e.x; break;
}
```

Simular el moviment continu de les plaques és una tasca difícil, ja que requeriria tenir en compte els moviments mesurats en fraccions de píxel. Per evitar-ho, les plaques es mouen en intervals de temps discrets, per un píxel sencer, ja sigui horitzontalment o verticalment. Aquests temps són aleatoris per a cada placa de manera que la velocitat mitjana es mantingui a la velocitat i la direcció establertes, i també de manera que és poc probable que les plaques veïnes es moguin simultàniament.

En executar el programa, es veurà com es formen aleatòriament i es comencen a expandir les plaques. En mostrar la dimensió x per pantalla, es veurà el codi de colors de cada grup de píxels que formen les plaques.

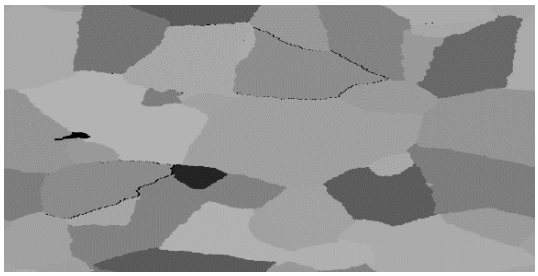

Fig. 9. *Mapa de plaques generades. Font: pròpia*

Les col·lisions de plaques es produeixen quan els píxels de límit d'una placa es mouen cap a una ubicació ocupada anteriorment per píxels pertanyents a una altra placa. Això provoca la subducció, que es modela simplement augmentant lleugerament l'elevació del terreny als llocs de la col·lisió. Tot i que això només es produeix als píxels al llarg del límit d'una placa, l'impacte s'estén gradualment als píxels veïns mitjançant un model d'erosió tèrmica senzill, que empeny l'elevació d'un píxel en la direcció de la mitjana dels seus veïns.

```
if (subduct) {
    gl_FragColor.y = 1.;
} else if (uTime < TECTONICS_END_TIME) {
    gl_FragColor.y = max(gl_FragColor.y - 0.0001, 0.);
}
```

En conjunt, podem veure que totes aquestes petites aportacions, proporcionen una aproximació decent de la

formació de continents amb serralades i en general un terreny accidentat molt interessant.

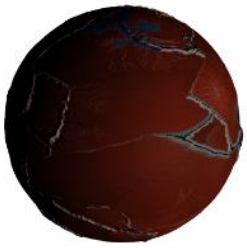

Fig. 10. *Model tectònic en funcionament. Font: pròpia*

Es pot visualitzar de forma més clara el funcionament d'aquest model tectònic si es representen cada una d'aquestes plaques amb un color distintiu. Per això fem ús de tres mapes diferents per acabar mostrant una combinació d'aquests. Es fa ús del mapa de plaques de la dimensió x , el mapa de subducció de la dimensió y, i també el propi mapa del terreny amb la roca i oceans. En executar la simulació, comencen a aparèixer les plaques tectòniques que van omplint el planeta. El resultat és el següent:

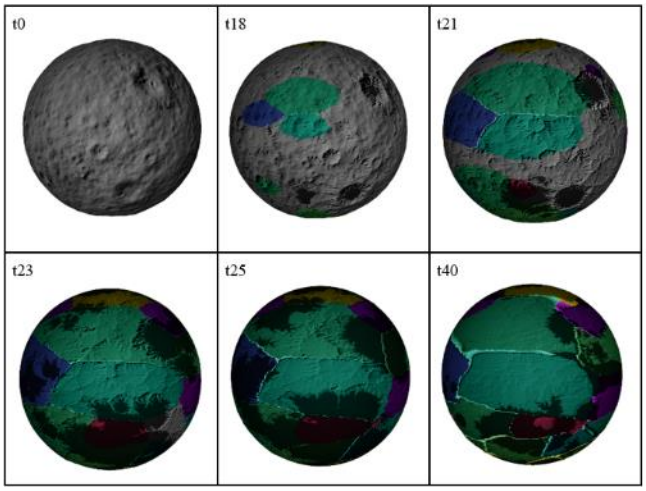

Fig. 11. *Formació i interacció del model tectònic en el programa. Font: pròpia*

7. Atmosfera

L'atmosfera és una barreja de gasos que envolta un planeta. A la Terra, l'atmosfera ajuda a fer possible la vida, ens protegeix dels efectes nocius de la llum solar, la radiació ultraviolada, el vent solar i els raigs còsmics per protegir els organismes del dany genètic. escalfa la superfície del nostre planeta a través de l'efecte hivernacle evita diferències extremes entre les temperatures diürnes i nocturnes[18].

A més de la Terra, molts dels altres objectes astronòmics del Sistema Solar tenen atmosferes. Aquests inclouen tots els gegants gasosos, així com Mart, Venus, Tità i Plutó. Diverses llunes i altres cossos també tenen atmosferes, com els cometes, el Sol i planetes extrasolars[19].

Després de refredar-se, la Terra jove tenia una atmosfera amb el diòxid de carboni com a constituent principal, així com nitrogen i una mica d'aigua. La pressió superficial també era molt més alta, gairebé cent vegades la d'avui i l'atmosfera era molt més alta, a causa de la superfície calenta. Aquestes característiques la feien més semblant a l'atmosfera de Venus actual que a la de la Terra actual.[20]

El cel sol ser blau, però vermell al capvespre principalment a causa d'un fenomen òptic que s'anomena dispersió de Rayleigh. Històricament, el descobriment de la dispersió atmosfèrica s'atribueix a lord Rayleigh (John William Strutt), un físic anglès premi Nobel de Física (1842-1919).[21]

Generar una dispersió atmosfèrica realista en ordinador sempre ha estat un problema difícil. Les equacions que descriuen la dispersió atmosfèrica són tan complexes que s'han dedicat llibres sencers al tema. Els models de gràfics per ordinador generalment utilitzen equacions simplificades, i molt pocs funcionen a velocitats de fotogrames interactives.[22]

El primer pas per replicar els efectes òptics de la dispersió atmosfèrica és entendre com la llum viatja a través d'un medi com l'aire. Només podem veure alguna cosa quan la llum arriba als nostres ulls. En el context dels gràfics en 3D, el nostre ull és la càmera que s'utilitza per renderitzar l'escena. Les molècules que formen l'aire que ens envolta poden desviar els raigs de llum que les travessen. Per tant, tenen el poder d'alterar la manera com percebem els objectes. Vist de forma molt simplificada, hi ha dues maneres en què les molècules de l'aire poden afectar la nostra visió[23].

La dispersió exterior (*out-Scattering*) fa referencia a la dispersió que ocorre quan les molècules d'aire interactuen amb la llum és desviant-la, canviant-ne la direcció. Quan la llum és desviada per una partícula, també pot passar que es redirigeixi cap a la càmera. Això és efectivament el contrari de la dispersió exterior (*out-scattering*) i, s'anomena dispersió interna (*in-scattering*).

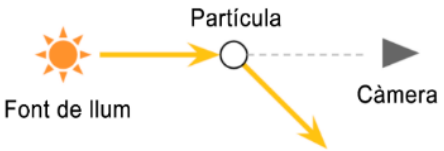

Fig. 12. *Dispersió out-scattering. Font: [23]*

El primer pas serà escriure un *shader* de post processament que s'encarregarà de dibuixar l'atmosfera al voltant del planeta. Aquest programa s'executarà per a cada píxel de la pantalla i li donarà un color segons els nostres càlculs.

El primer que cal esbrinar es on es creua la vista de la càmera amb una esfera al voltant. D'aquesta forma es poden obtenir els punts d'origen i direcció del raig, així com el punt on interseca amb l'atmosfera, tant el de entrada com el de sortida. En cas de no arribar a sortir per l'atmosfera i impactar amb el planeta també es tindrà en compte.

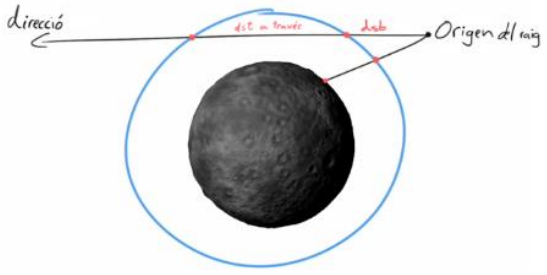

Fig. 13. *Raig de llum que creua l'atmosfera del planeta. Font: pròpia*

Per calcular la quantitat de llum transmesa a la càmera, és útil fer el mateix viatge que experimenten els raigs de llum del sol. Mirant el diagrama següent, és fàcil veure que els raigs de llum que arriben a C viatgen per l'espai buit. Com que no hi ha res al seu pas, tota la llum arriba a C sense estar sotmesa a cap efecte de dispersió. Indiquem amb I_C la quantitat de llum no dispersa que rep el punt C del sol. Durant el seu viatge cap a P, la llum entra a l'atmosfera del planeta. Alguns raigs xocaran amb les molècules suspeses a l'aire, rebotant en diferents direccions. El resultat és que part de la llum s'allunya del camí. La quantitat de llum que arriba a P (indicada amb I_P) serà inferior a I_C candeles (cd).[21][22][23]

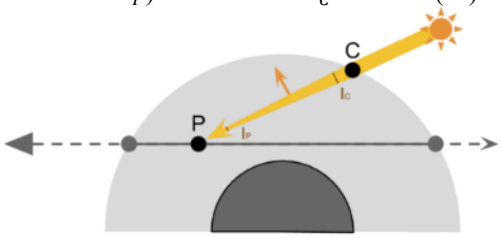

Fig. 14. *Diagrama de la funció de transmissió. Font: [23]*

La relació entre I_C i I_P s'anomena transmitància:

$$T(\underline{CP}) = \frac{I_P}{I_C} \qquad (2)$$

I el podem utilitzar per indicar el percentatge de llum que no es dispersa durant el trajecte de C a P.
El punt P rep la llum directament del sol. Tanmateix, no tota la llum que passa per P es torna a transferir a la càmera. Per calcular quanta llum arriba realment a la càmera, cal d'introduir un altre concepte: la funció de dispersió S. La seva finalitat és indicar quanta llum es desvia en una determinada direcció. Si mirem el diagrama següent, ens adonem que només aquells raigs de llum que es desvien per un angle de θ es dirigeixen cap a la càmera. [23]

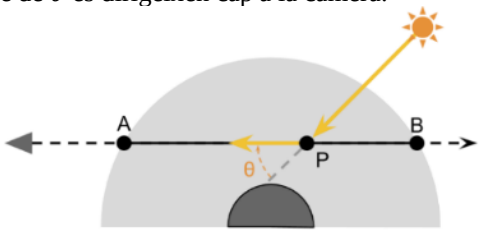

Fig. 15. *Diagrama de la funció de transmissió Font: [23]*

El valor de $S(\lambda, \theta, h)$ indica la proporció de llum desviada per θ radians. Aquesta funció és el centre del nostre problema. De moment, l'únic que cal saber és que depèn del color de la llum entrant (indicat per la seva longitud d'ona λ), de l'angle de dispersió θ i de l'altitud h de P. El motiu pel qual l'altitud és important és perquè la densitat atmosfèrica canvia en funció de l'altitud. I la densitat és, en definitiva, un dels factors per determinar quanta llum es dispersa. [23] La equació general que indica quanta llum es transfereix de P a A es:

$$I_{PA} = I_P \, \mathrm{S}(\lambda, \theta, h) \, \mathrm{T}\,(\overline{\mathrm{PA}}) \qquad (3)$$

La llum viatja de la font de llum a C, sense dispersar-se en el buit de l'espai; La llum entra a l'atmosfera i viatja de C a P. En fer-ho, només la fracció $T(\overline{\mathrm{CP}})$ arriba al seu destí a causa de la **dispersió externa** (*out-scattering*). Part de la llum que ha arribat a P des del sol es desvia cap a la càmera. La proporció de llum sotmesa a la **dispersió interna** (*in-scattering*) és $S(\lambda, \theta, h)$. La llum restant viatja de P a A, i una vegada més només es transmet la fracció $T(\overline{\mathrm{PA}})$.

Imaginem un raig de llum viatjant per l'espai buit, xocant de sobte amb una partícula. El resultat d'aquesta col·lisió varia dràsticament depenent de la mida de la partícula i del color de la llum. Si la partícula és prou petita (com àtoms i molècules), el comportament de la llum es prediu millor mitjançant la dispersió de Rayleigh

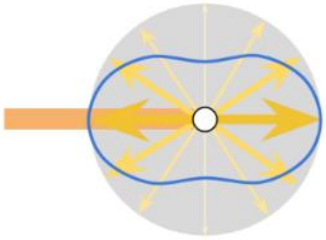

Fig. 16. *Possibles direccions de desviació d'un fotó en interactuar amb una partícula. Font: [23]*

El que passa és que una part de la llum continua el seu viatge sense ser afectada. Tanmateix, un petit percentatge d'aquesta llum original interactua amb la partícula i es dispersa en totes direccions. No totes les direccions, però, reben la mateixa quantitat de llum. És més probable que els fotons passin directament a través de la partícula o que rebotin. Per contra, el resultat menys probable per a un fotó és desviar-se 90 graus. Aquest comportament es pot veure al diagrama següent. La línia blava mostra les direccions preferides per a la llum dispersa.[23]

Aquest fenomen òptic es descriu matemàticament per l'equació de dispersió de Rayleigh $S(\lambda, \theta, h)$, que indica la proporció de la llum original I_0 que es dispersa cap a la direcció θ:

$$I = I_0 \, S(\lambda, \theta, h)$$

$$S(\lambda, \theta, h) = \frac{\pi^2 \left(n^2 - 1\right)^2}{2} \underbrace{\frac{\rho(h)}{N}}_{\text{density}} \overbrace{\frac{1}{\lambda^4}}^{\text{wavelength}} \underbrace{\left(1 + \cos^2\theta\right)}_{\text{geometry}} \qquad (4)$$

On:

- λ: Longitud d'ona de la llum entrant (m);
- θ: Angle de dispersió (graus);
- h: Altitud del punt (m);
- n=1,00029: l'índex de refracció de l'aire;
- $N=2{,}504 \cdot 10^{25}$: Densitat del nombre molecular de l'atmosfera estàndard. Aquest és el nombre de molècules per metre cúbic;
- $\rho(h)$: la relació de densitat. Aquest nombre és igual a 1 al nivell del mar, i disminueix exponencialment amb h.

El primer que observem sobre la dispersió de Rayleigh és que determinades direccions reben més llum que altres. El segon aspecte important és que la quantitat de llum dispersa depèn fortament de la longitud d'ona λ de la llum entrant.
El següent diagrama polar visualitza $S(\lambda, \theta, h)$ per a tres longituds d'ona diferents. Avaluar S amb h=0 sovint es coneix com a dispersió al nivell del mar. [23]

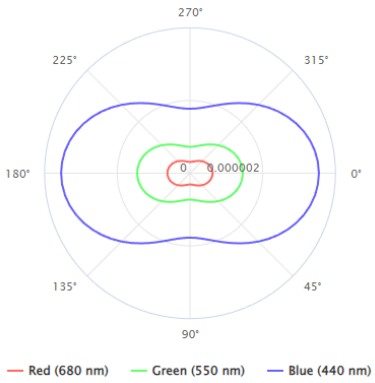

Fig. 17. *Diagrama polar de dispersió per a tres longituds d'ona (vermell, verd i blau). Font: [22]*

Malauradament, calcular el coeficient d'extinció utilitzat en la definició de la transmitància $T(\overline{AB})$ sobre un segment $\overline{AB}$ és molt car computacionalment. Al *shader* que anem a escriure, intentarem estalviar el màxim de càlcul possible. Per aquest motiu, és útil "extreure" del coeficient de dispersió tots els factors que són constants. Això ens deixa amb $\beta(\lambda)$, que es coneix com el coeficient de dispersió de Rayleigh com a nivell del mar (h=0):

$$\beta(\lambda) = \frac{8\pi^3 \left(n^2 - 1\right)^2}{3} \frac{1}{N} \frac{1}{\lambda^4} \qquad (5)$$

Aquesta nova equació dona una altra manera d'entendre com es dispersen els diferents colors de la llum. Aquest gràfic mostra la quantitat de llum dispersa a la qual està sotmesa, en funció de la seva longitud d'ona:

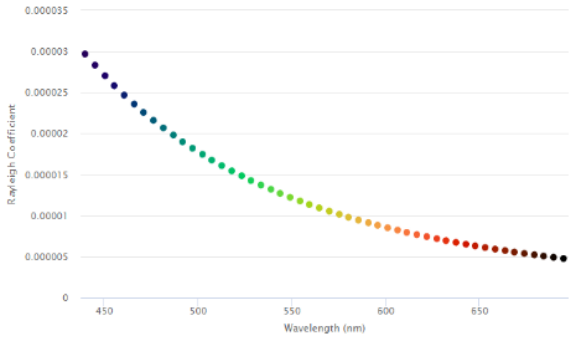

Fig. 18. *Dispersió de la llum en funció de la longitud d'ona λ Font: [22]*

És la forta relació entre el coeficient de dispersió β i λ que torna vermells els cels de la posta de sol. Els fotons que rebem del sol es distribueixen en una àmplia gamma de longituds d'ona. El que ens diu la dispersió de Rayleigh és que els àtoms i les molècules de l'atmosfera terrestre dispersen la llum blava amb més força que la verda o la vermella. Si es permet que la llum viatgi durant prou temps, tota la seva llum blava es perdrà per dispersió. Això és exactament el que passa al capvespre, quan la llum viatja gairebé paral·lela a la superfície[23].

Amb el mateix raonament, podem entendre per què el cel apareix blau. La llum del sol arriba amb una direcció específica. Tanmateix, el seu component blau està dispers en totes direccions. Quan mirem el cel, la llum blava prové de totes direccions.

Finalment podem definir un *shader* amb codi que pugui descriure tots aquests fenòmens presentats.

El primer que cal fer és definir la funció amb la que es calcularà. Cal declarar el primer punt (*inScatterPoint*) on es calcula la dispersió i s'executa un bucle amb el nombre de punts desitjats (*inScatterPoints*).

Dins del bucle, es mou el punt a traves del raig de visió sumant-li la distancia entre punts (stepSize). Que es poden calcular fàcilment.

```
calculateLight(vec3 rayOrigin, vec3 rayDir, float rayLength)
{
  vec3 inScatterPoint = rayOrigin;
  float stepSize = rayLength / (inScatterPoints - 1.0);

  for (float i = 0.; i < inScatterPoints; i++) {
    inScatterPoint += rayDir * stepSize;
  }
}
```

Per fer un seguiment de la quantitat total de llum dispersa en tots aquests punts, es defineix la variable *inScatteredLight* inicialitzada a 0. Com es poden calcular la quantitat de llum de l'estrella arriba a cada un d'aquests punts? Ja que una part es dispersarà abans de poder arribar i la quantitat que es dispersa depèn de la densitat de l'atmosfera a través d'aquest raig per on la llum viatja, és a dir, la profunditat òptica. En el codi caldrà conèixer la llargada d'aquest raig que travessa l'atmosfera. La proporció de llum que arriba fins al punt de dispersió s'anomena transmitància. Quan la profunditat òptica és zero, la transmitància serà 1 perquè tota la llum arribarà al seu destí. Però a mesura que la profunditat òptica augmenta, més i més llum es dispersarà i així veurem un caiguda exponencial de la quantitat de llum transmesa.

Un cop es coneix la quantitat de llum que arribarà al punt, la següent pregunta és quina quantitat es dispersarà cap a la càmera. El factor més important és la densitat al punt de dispersió, de manera que al codi es fa ús d'una nova funció per calcular la densitat en un sol punt. Com més gran sigui aquesta densitat, més llum es dispersarà, de manera que podem sumar aquest valor a la llum dispersada, multiplicar-la per la transmitància i la mida del pas.

Una cosa més que cal tenir en compte és que a mesura que la llum dispersada internament viatja cap a la càmera, una part es dispersarà de nou i pot acabar finalment en la càmera. Així que al codi es calcula la profunditat òptica des del punt de dispersió fins a la càmera i després, s'actualitza la transmitància per tenir en compte el càlcul. Aquest càlcul es pot simplificar per utilitzar només un exponent únic.
L'atmosfera és més densa a prop de la superfície i es torna exponencialment menys densa amb l'altitud[24], així que al codi es calcula l'alçada sobre la superfície i es normalitza el resultat perquè sigui 0 a la superfície i 1 a la capa exterior de l'atmosfera per facilitar-ne el treball.

```
float densityAtPoint(vec3 densitySamplePoint) {
  float heightAboveSurface = length(densitySamplePoint -
planetPosition) - planetRadius;
  float height01 = heightAboveSurface / (atmosphereRadius -
planetRadius);
  float localDensity = exp(-height01 * densityFalloff) *
(1.0 - height01);
  return localDensity;
}
```

Arribats a aquest punt, es pot executar el programa i l'únic que es veurà és una petita boira al voltant del planeta, el següent pas serà ampliar el codi per implementar la dispersió dels colors. Cal formar la imatge amb els colors vermell, verd i blau, així que cal fer nos una idea aproximada d'aquestes longituds d'ona. El vermell s'apropa als 700 nanòmetres, el verds a uns 530 i el blau 440.

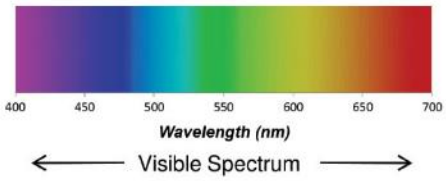

Fig. 19. *Longitud d'ona λ (nm) de la llum visible de l'espectre electromagnètic. Font: [25]*

Cal recordar que la quantitat de dispersió és inversament proporcional a la quarta potència de la longitud d'ona (Fórmula 4), llavors podem dir que la força de dispersió de la llum vermella és igual a 1 entre la longitud d'ona multiplicat per 4 al quadrat multiplicada per la força de dispersió. El mateix per al verd i el blau.

Aquestes dades s'envien al *shader* i cal adaptar-lo per fer ús d'aquestes noves dades. Caldrà multiplicar la variable de profunditat òptica dins de la funció exponencial per el coeficient de dispersió. A un gràfic de transmitància amb la una força de dispersió proper a 0, pràcticament es transmetrà tota la llum. A mesura que augmenta la força, es pot veure que les diferents longituds d'ona es separen en funció del coeficient de dispersió calculat i es pot observar que la llum vermella és la que més es transmet i la llum blava la que menys, així que les matemàtiques semblen funcionar.

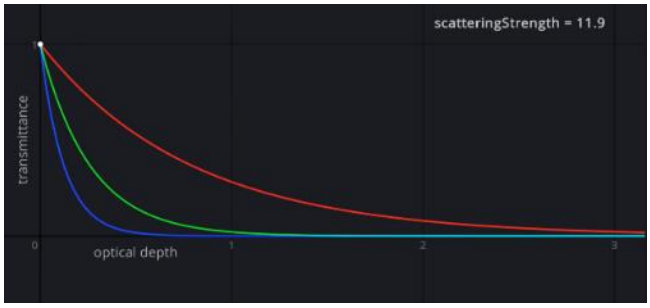

Fig. 20. *Gràfic de la funció de transmitància on la força de dispersió és 11.9. Font: [26]*

El coeficient de dispersió també afecta a la quantitat de llum que es dispersa cap a la càmera, així que cal comptabilitzar-la a la variable *inScatteredLight*. Això implica que la variable *inScatteredLight* haurà de ser un vec3 i, per tant, la funció *calculateLight* també haurà de tornar un vec3.

En executar el codi, es pot observar l'atmosfera sobre el planeta, aquesta vegada amb uns colors molt propers a la realitat gràcies als càlculs realitzats. Es poden modificar els paràmetres de l'atmosfera per veure l'impacte que té en temps real (Mida, corba de densitat, quantitat de punts als raigs de llum o fins i tot modificar els paràmetres de les longituds d'ona per modificar el color de l'atmosfera).

Fig. 21. *Resultat final del model atmosfèric en execució. Font: pròpia*

8. Clima global

Simular el sistema climàtic d'un planeta sencer és notòriament complicat, però per sort resulta que es pot aproximar amb relativa facilitat. Pràcticament tot el que és important sobre els climes es pot deduir dels següents principis físics:

1. Tot el calor prové del sol.
2. L'aigua s'escalfa i es refreda molt més lentament que la terra.
3. L'aire calent puja, l'aire fred s'enfonsa; això és perquè l'aire s'expandeix a mesura que s'escalfa i, per tant, es torna menys dens.
4. L'aire fred dona lloc a zones d'alta pressió, i l'aire calent dona lloc a zones de baixa pressió.
5. El vent es mou de zones d'alta pressió a zones de baixa pressió.
6. A causa de l'efecte Coriolis, l'efecte de la rotació de la terra sobre el flux d'aire, els vents es desvien cap a la dreta a l'hemisferi nord i cap a l'esquerra al sud.
7. L'aire que puja és propici a la causar precipitacions, l'aire que s'enfonsa no ho és. A mesura que l'aire puja, es refreda i les gotes d'aigua es poden condensar en núvols
8. L'aire calent transporta més humitat que l'aire fred. [27]

Si la Terra no girés i es mantingués estacionària, l'atmosfera circularia entre els pols (zones d'alta pressió) i l'equador (una zona de baixa pressió) en un patró senzill d'anada i tornada. Però com que la Terra gira, l'aire que circula es desvia. En lloc de circular en un patró recte, l'aire es desvia cap a la dreta a l'hemisferi nord i cap a l'esquerra a l'hemisferi sud, donant lloc a camins corbes. Aquesta desviació s'anomena efecte Coriolis i porta el nom del matemàtic francès **Gaspard Gustave de Coriolis** (1792-1843)[28].

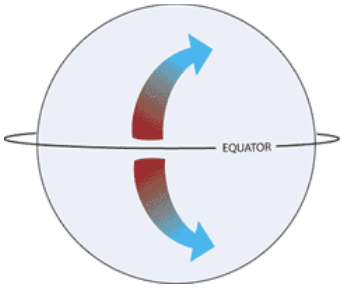

Fig. 22. *L'efecte Coriolis en acció. Font: [28]*

La primera etapa consisteix a localitzar les zones a gran escala d'alta i baixa pressió. El més important a tenir en compte és el cinturó de baixa pressió anomenat zona de convergència intertropical, o **ITCZ**, sobre el qual les característiques de temperatura i pressió són teòricament simètriques; aquesta zona és causada per l'ascens d'aire tropical calent. [28]

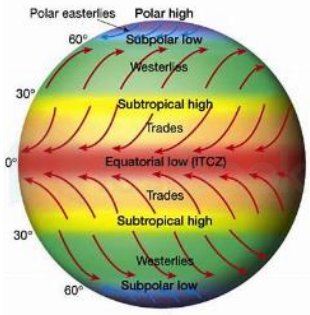

Fig. 23. *Distribució dels tròpics i la ITCZ al planeta Terra. Font: [29]*

Hi ha un altre cinturó d'alta pressió entre el **ITCZ** i els pols, conegut com a zona d'alta pressió subtropical, o **STHZ**, que és causat pel refredament de l'aire del **ITCZ** i que s'enfonsa cap a terra. Entre la **STHZ** i els pols hi ha el front polar o **PF**, una banda de baixa pressió on l'aire fred dels pols es troba amb l'aire càlid de la **STHZ**.

Si la superfície del planeta fos completament aigua uniforme, la distribució d'aquests cinturons de pressió i els vents dominants seria com es pot veure a la figura 24. [27]

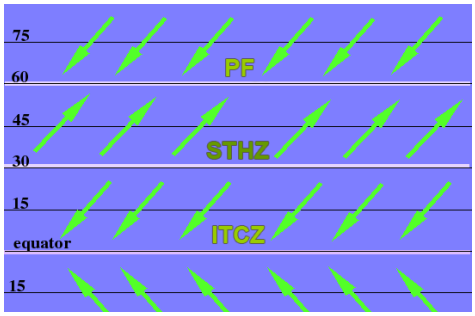

Fig. 24. *Distribució dels cinturons de pressió. Font: [27]*

La presència de terra té dos efectes en la distribució de la pressió, ambdós resultats dels principis 2,3 i 4 que s'han presentat a l'inici del capítol. Els cinturons de pressió frontal es dobleguen cap al nord sobre la terra al juliol i cap al sud al gener, i es trenquen per les àrees de pressió estacional sobre el terreny. En general, com més gran sigui la superfície de terra, més notable serà l'efecte. A l'hivern, la baixada de temperatura de la terra crea una zona d'alta pressió a l'interior, que es barreja amb la zona d'alta pressió al voltant de la **STHZ** i deixa sistemes de baixa pressió sobre els oceans:

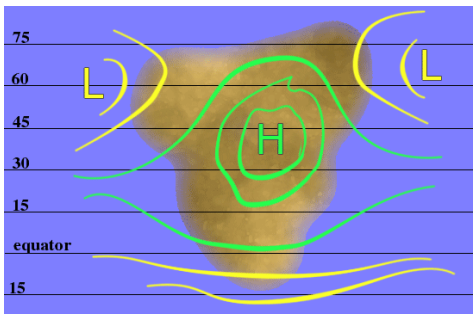

Fig. 25. *Zones d'alta i baixa pressió a l'hivern. Font: [27]*

Mentre que a l'estiu la terra s'escalfa per crear una zona de baixes pressions, que s'uneix amb la **ITCZ** i la **PF**, deixant zones d'alta pressió sobre els oceans:

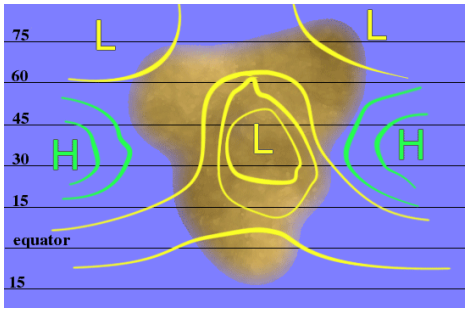

Fig. 26. *Zones d'alta i baixa pressió a l'estiu. Font: [27]*

En general, aquestes zones de pressió es troben a l'est de la meitat longitudinal (est-oest) del continent, i són més intenses quan la massa terrestre circumdant és més gran. Això es nota especialment amb Àsia. En conseqüència, el gradient de pressió és més gran a les costes est que a les costes oest; la diferència precisa depèn de la forma del continent. [27] Per a crear un mapa de pressió mitjana a nivell del mar (**MSLP** - *Mean sea level pressure*) cal saber on es troben les formes de terra que hi ha entre els oceans i l'impacte que té la latitud en cada punt. Per definició, la pressió mitjana a nivell del mar del nostre planeta és d'una atmosfera, o 1013.25 hPa.[30] Si agafem dades d'un mapa MSLP real de la Terra, el separem segons si són zones terrestres o oceàniques i tracem el MSLP en funció de la latitud, acabem amb dues corbes sinusoidals per a la terra i l'oceà amb lleugerament diferents formes. [31]

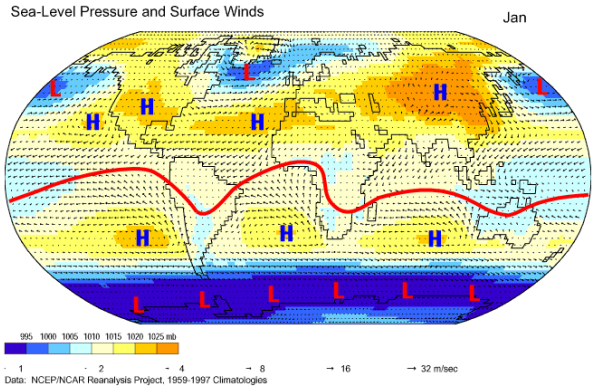

Fig. 27. *Mapa de vents i pressió atmosfèrica de gener, 1959-1997. La línia vermella representa la ITCZ. . Font: [32]*

En ajustar els paràmetres adequadament, podem obtenir un model brut de la pressió mitjana anual.

```
if (land) {
    mslp = 1012.5 - 6. * cos(lat*PI/45.);
} else { // ocean
    mslp = 1014.5 - 20. * cos(lat*PI/30.);
}
```

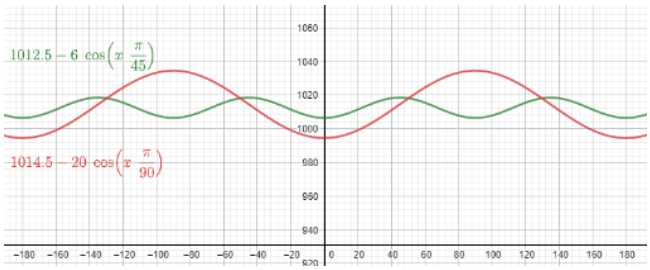

Fig. 28. *Corbes sinusoidals per terra i oceà en funció de la latitud Font: pròpia*

Això no és suficient per generar un mapa MSLP realista, ja que generar valors per a la terra i l'oceà per separat provoca discontinuïtats molt brusques als límits entre ells. En realitat, MSLP varia suaument al llarg de la transició de l'oceà a la terra. Aquest procés es pot aproximar bastant bé aplicant simplement un desenfocament al mapa MSLP.

L'eix de rotació de la Terra està inclinat respecte al seu pla orbital uns 23.5 graus. Això és el que provoca les estacions. Per permetre que el clima canviï amb les estacions, també cal modelar la diferència de MSLP entre gener i juliol. Una vegada més, les dades terrestres suggereixen que això segueix un patró sinusoïdal.

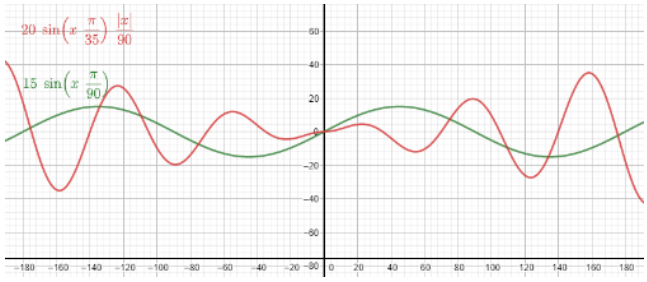

Fig. 29. *Corbes sinusoidals per representar la diferència de MSLP entre les estacions en funció de la latitud. Font: pròpia*

Després d'ajustar els paràmetres i aplicar un desenfocament gaussià, es pot combinar amb el mapa MSLP anual per generar patrons climàtics dinàmics que varien al llarg de l'any.

```
if (land) {
   delta = 15. * sin(lat*PI/90.);
} else { // ocean
   delta = 20. * sin(lat*PI/35.) * abs(lat)/90.;
}
```

El desenfocament gaussià que es fa servir és un efecte per suavitzar imatges. En essència, l'efecte barreja lleugerament els colors dels píxels que estiguin veïns l'un a l'altre, cosa que provoca que la imatge perdi alguns detalls minúsculs i, així, fa que la imatge es vegi més suau (encara que menys nítida o clara). Utilitza una funció gaussiana (que també expressa la distribució normal en estadístiques) per calcular la transformació que cal aplicar a cada píxel de la imatge. [33]

El codi per a definir aquesta distribució normal és el següent:

```
#define SIGMA vec4(6,4,1,0)
vec4 normpdf(float x) {
        return 0.39894 * exp(-0.5 * x*x / (SIGMA*SIGMA)) /
SIGMA;
}
```

Després d'aplicar colors càlids a les àrees de baixa pressió (entre 1000 i 1012 hPa) i colors freds a les àrees d'alta pressió (entre 1012 i 1024 hPa), podem veure el resultat:

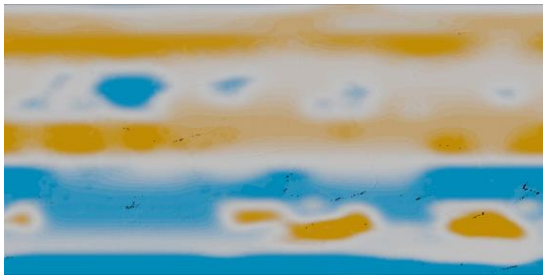

Fig. 30. *Visualització de la pressió atmosfèrica. Font: pròpia*

Amb aquest model de pressió atmosfèrica simple en funcionament, és possible generar corrents de vent i temperatures. El vent, en termes meteorològics, és un flux d'aire d'una zona d'alta pressió a una zona de baixa pressió. La força (velocitat) del vent augmenta amb la diferència de pressió. Els vents tenen dos efectes importants sobre el clima: transporten humitat i són la principal causa dels corrents oceànics. Recullen humitat mentre bufen sobre els oceans i la dipositen com a pluja o neu sobre la terra. Òbviament, un vent només pot portar una quantitat finita d'humitat, de manera que s'assecarà després de bufar per una gran àrea de terra.

A causa de l'efecte Coriolis, els vents no bufen directament d'alta pressió a baixa pressió, sinó que es desvien, a l'hemisferi nord, en sentit horari al voltant de zones d'alta pressió i en sentit antihorari al voltant de zones de baixa pressió. A l'hemisferi sud, la desviació és en sentit contrari.[27]

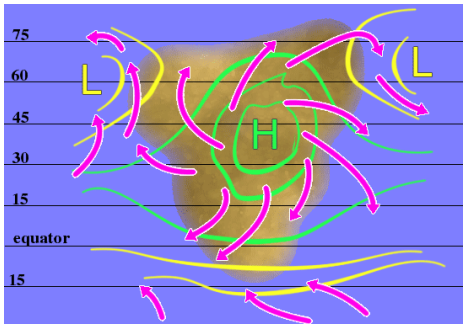

Fig. 31. *Direcció dels vents dominants a l'hivern. Font: [27]*

Per a implementar-lo, es fa ús del mapa MSLP per obtenir els vectors de direcció del vent, gràcies a les diferencies de pressió i temperatura entre diferents punts del mapa, aquest vector es **grad** al codi. La velocitat d'aquest vent anirà directament determinada per la diferència de pressió.

```
vec2 coriolis = 15 * sin(lat*PI/180)*vec2(-grad.y, grad.x);
vec2 velocity = coriolis - grad;
```

Tot i que és una aproximació molt superficial, aquesta és capaç de generar patrons de circulació del vent bastant realistes, ja que és possible observar fenòmens naturals que es reprodueixen amb bastant exactitud, almenys en l'escala en la qual treballem.

La variació anual de la temperatura és l'altre tret característic d'un clima. Com a primera aproximació, la temperatura és més alta a l'equador i disminueix constantment cap als pols. Les variacions de temperatura són més baixes al llarg de les costes i més altes a les zones allunyades de la influència marítima. La variació augmenta amb la distància dels oceans, i menys amb la distància de la costa oest; les regions orientals dels interiors continentals experimenten així les majors variacions de temperatura.[27] Es pot aproximar la temperatura en un punt concret basant-nos en la formula:

```
float temp = 40. * tanh(2.2 * exp(-0.5 * pow((lat + 5. *
season)/30., 2.))) - 15. - (mslp - 1012.) / 1.8 + 1.5 * land -
4. * elevation;
```

Fig. 32. *Mapa de la temperatura global en la simulació. Els valors més freds en blau, i els més calents en carabassa. Font: pròpia*

La precipitació es pot simular a partir del vapor d'aigua de l'oceà, que es transporta a través del vent cap a la terra. La distribució anual de precipitacions en forma de pluja i neu és un dels factors que caracteritzen un clima determinat. Per a simular l'increment de vapor d'aigua en un punt determinat mitjançant l'evaporació, primer tenim en compte que sigui un punt a l'oceà, llavors s'aplica soroll FBM (*Fractional Brownian Motion*) perquè el resultat no sigui massa homogeni. La quantitat de vapor d'aigua dependrà de la temperatura, pressió atmosfèrica i velocitat del vent en aquest punt.

```
if (ocean) wVapour += noise * clamp(temp + 2., 0.,
100.)/32. * (0.075 - 3. * div - 0.0045 * (mbar - 1012.));
```

En el cas de la precipitació i els núvols orogràfics, simplement es pot reduir aquesta quantitat de vapor d'aigua progressivament en funció del temps per al primer cas, i en funció de l'altura del terreny per al segon cas.

```
w -= 0.005 * w;
w -= 0.3 * length(hgrad);
```

Com a últim detall, es pot l'impacte que té la temperatura i humitat a l'hora de formar els paisatges terrestres i la vegetació. Basant-nos en dades de creixement de diferents cultius sota unes condicions concretes, podem definir una funció per aproximar la velocitat de creixement de la vegetació en un punt determinat segons la temperatura i humitat.

```
float plant_growth(float moisture, float temp) {
  float growth = clamp(moisture / 3., 0., 1.);
  growth *= smoothstep(0, 25., temp) - smoothstep(25., 35.,
temp);
  return growth;
}
```

Per a representar amb certa precisió el paisatge del terreny també cal tenir en compte principalment aspectes com l'elevació, quantitat de precipitacions anuals i latitud. Això té un gran impacte en el tipus de bioma que tindrà una àrea determinada. Es poden aproximar aquestes dades d'una forma molt senzilla en funció de la temperatura i la funció de creixement de vegetació que s'ha definit prèviament. D'aquesta forma es poden obtenir uns resultats bastant propers a la realitat, almenys al nivell de precisió al que treballem.

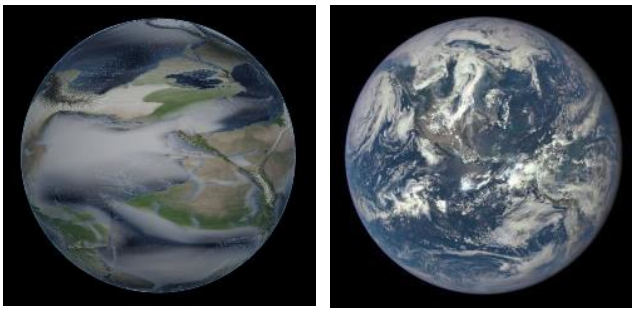

Fig. 33. *Comparació del resultat final en executar el programa (esquerra) i una imatge real de la Terra (dreta). Font:[34]*

9. Conclusions

Simular la formació de planetes i els processos que intervenen és una tasca molt complexa, encara que es simplifiqui i es duguin a terme aproximacions. Amb tot i això, s'han pogut complir els objectius proposats en aquest projecte. S'ha demostrat que es possible implementar els principals propòsits del treball en les diferents tecnologies que hi ha disponibles. Aconseguir que el programa s'executi de forma fluida en *hardware* més contingut continua sent un repte, sobretot a l'hora d'actualitzar les textures generades amb els càlculs fets als diversos *shaders* que hi ha. Lamentablement, es tracta d'una àrea en la qual només s'ha pogut veure la punta de l'iceberg, a causa de la gran complexitat del mateix i la gran càrrega de treball que requereix cobrir tots els aspectes possibles. Pel que fa a Three.JS, ha demostrat ser un *framework* molt potent i ha complert amb les expectatives i les necessitats del projecte.

Com a treball futur, existeixen moltes millores que es podrien realitzar al llarg del treball; la primera a mencionar podria ser el rendiment. Encara que s'ha tractat de simplificar els càlculs, el programa en general continua sent molt costós d'executar i es necessita un equip relativament modern per a poder reproduir les simulacions amb fluïdesa. Per una altra part, es podria ser més rigorós a l'hora de simular cada un de les característiques que hem vist, així com ampliar-les i introduir més variables per poder generar resultats més precisos, realistes i de millor qualitat. Tampoc podem oblidar que, amb aquesta base que hem definit, és relativament fàcil extrapolar el simulador gràfic per a considerar incloure molts

més tipus d'objectes astronòmics i variacions d'aquests. Planetes gasosos, amb anells, llunes, asteroides o estrelles, per mencionar alguns.

Referències

[1] Ricardo Cabello - Three.js Manual: Fundamentals https://threejs.org/manual/#en/fundamentals [Consultada 28/02/2022]

[2] Sanchez-Lavega, Agustin (2010). An Introduction to Planetary Atmospheres Taylor & Francis. ISBN 978-1-4200-6732-3.

[3] Sasha Warren, University of Chicago - How the Earth and moon formed, explained (2021) https://news.uchicago.edu/explainer/formation-earth-and-moon-explained [Consultada 01/04/2022]

[4] Patricio Gonzalez Vivo & Jen Lowe - The book of shaders (2015) https://thebookofshaders.com/ [Consultada 13/03/2022]

[5] Steven Worley - A Cellular Texture Basis Function, pg. 291 (1996) http://www.rhythmiccanvas.com/research/papers/worley.pdf [Consultada 03/04/2022]

[6] National Geographic - The History of Earth: How Our Planet Formed (2011) https://www.imdb.com/title/tt1985159/

[7] Tobias C. Owen of the Institute for Astronomy in Honolulu, Hawaii - What do we know about the origin of the earth's oceans? (1999) https://www.scientificamerican.com/article/what-do-we-know-about-the/ [Consultada 12/06/2022]

[8] Sebastian Lague - Coding Adventure: Atmosphere (2020) https://www.youtube.com/c/SebastianLague [Consultada 07/06/2022]

[9] Job Talle - Simulating hydraulic erosion (2020) https://jobtalle.com/simulating_hydraulic_erosion.html [Consultada 13/06/2022]

[10] Cristina Novillo - Erosión hídrica: definición, tipos, causas y consecuencias (2019) https://www.ecologiaverde.com/erosion-hidrica-definicion-tipos-causas-y-consecuencias-2172.html [Consultada 13/06/2022]

[11] Richard Barnes - Accelerating a fluvial incision and landscape evolution model with parallelism (2018) https://arxiv.org/abs/1803.0297 [Consultada 13/06/2022]

[12] Wikipedia - Stream power law (2010) https://en.wikipedia.org/wiki/Stream_power_law [Consultada 14/06/2022]

[13] ck12 - Landforms from Stream Erosion and Deposition (2012) https://www.ck12.org/earth-science/landforms-from-stream-erosion-and-deposition/lesson/landforms-from-stream-erosion-and-deposition-hs-es/ [Consultada 14/06/2022]

[14] Grasshopper Geography - River basins of Europe https://www.grasshoppergeography.com/ [Consultada 14/06/2022]

[15] Wikipedia - Plate tectonics (2021) https://en.wikipedia.org/wiki/Plate_tectonics [Consultada 17/06/2022]

[16] Joseph A. DiPietro, in Geology and Landscape Evolution (Second Edition) - Movement of Tectonic Plates (2018) https://www.sciencedirect.com/science/article/pii/B9780128111918000051 [Consultada 17/06/2022]

[17] Irene Martínez Pérez - Les plaques tectòniques http://www.xtec.cat/~imartin6/1/hotpot/tectonica/tectonica3.htm [Consultada 17/06/2022]

[18] Sanchez-Lavega, Agustin (2010). An Introduction to Planetary Atmospheres Taylor & Francis. ISBN 978-1-4200-6732-3. (2010)

[19] Seki, K.; Elphic, R. C.; Hirahara, M.; Terasawa, T.; Mukai, T. (2001). "On Atmospheric Loss of Oxygen Ions from Earth Through Magnetospheric Processes". (2001) doi:10.1126/science.1058913. PMID 11239148. S2CID 17644371.

[20] Redox state of Earth's magma ocean and its Venus-like early atmosphere, Sossi, P.A., Burnham, A. D., Badro, J., Lanzirotti, A., Newville, M. & O'Neill, H.St.C., Science Advances. doi: 10.1126/sciadv.abd1387 (2020) https://nccr-planets.ch/blog/2020/11/26/new-insights-into-earths-primeval-atmosphere/ [Consultada 20/06/2022]

[21] Scratchapixel - Simulating the Colors of the Sky (2015) https://www.scratchapixel.com/lessons/procedural-generation-virtual-worlds/simulating-sky [Consultada 18/04/2022]

[22] Sean O'Neil - GPU gems 2 : programming techniques for high-performance graphics and general-purpose computation - Accurate Atmospheric Scattering (ISBN 0-321-33559-7) (2005) https://developer.nvidia.com/gpugems/gpugems2/part-ii-shading-lighting-and-shadows/chapter-16-accurate-atmospheric-scattering [Consultada 18/04/2022]

[23] Alan Zucconi - Volumetric Atmospheric Scattering (2017) https://www.alanzucconi.com/2017/10/10/atmospheric-scattering-1/ [Consultada 18/04/2022]

[24] Tomoyuki Nishita - Display of The Earth Taking into Account Atmospheric Scattering (1996) https://www.researchgate.net/publication/2933032_Display_of_The_Earth_Taking_into_Account_Atmospheric_Scattering [Consultada 18/06/2022]

[25] The electromagnetic spectrum By Marcus Griffiths https://theelectromagneticspectrumgriffithsm.weebly.com/3-which-colour-in-the-visible-light-spectrum-has-the-least-energy.html [Consultada 07/06/2022]

[26] Sebastian Lague - Coding Adventure: Atmosphere (2020) https://www.youtube.com/c/SebastianLague [Consultada 07/06/2022]

[27] Geoff's Climate Cookbook (2008) https://web.archive.org/web/20130619132254/http://jc.tech-galaxy.com/bricka/climate_cookbook.html [Consultada 21/06/2022]

[28] National Ocean Service - The Coriolis Effect (2013) https://oceanservice.noaa.gov/education/tutorial_currents/04currents1.html [Consultada 21/06/2022]

[29] Testbook - Intertropical Convergence Zone (ITCZ), a low-pressure zone, is located (2013) https://www.globe.gov/explore-science/scientists-blog/archived-posts/sciblog/index.html_p=1661.html [Consultada 21/06/2022]

[30] Wikipedia – Atmospheric pressure https://en.wikipedia.org/wiki/Atmospheric_pressure [Consultada 22/06/2022]

[31] David A Roberts - Simulating worlds on the GPU (2018) https://davidar.io/post/sim-glsl [Consultada 30/02/2022]

[32] PhysicalGeography | FUNDAMENTALS eBOOK - CHAPTER 7: Introduction to the Atmosphere - Global Scale Circulation of the Atmosphere (2010) http://www.physicalgeography.net/fundamentals/7p.html [Consultada 22/06/2022]

[33] Wikipedia – Gaussian blur https://en.wikipedia.org/wiki/Gaussian_blur [Consultada 23/06/2022]

[34] NASA - NASA Captures "EPIC" Earth Image (2015) https://www.nasa.gov/image-feature/nasa-captures-epic-earth-image [Consultada 19/06/2022]