**ALGORITMES PROCEDIMENTALS PER AL DESENVOLUPAMENT D'ENTORNS ALEATORIS: CREANT UNA ILLA**

Marc Anglès

Directora: Anna Puig

# 1. Introducció

Des de ja fa uns anys que la creació de mons virtuals ha avançat cap a la generació aleatòria dels mateixos. Des de Minecraft fins a The Binding of Isaac són molts els títols que apunten per aquest sistema per a produir experiències úniques per a cada usuari sense haver d'invertir grans quantitats en disseny.

Tots aquests sistemes es troben dins del paradigma de la programació procedimental. L’objectiu d’aquest treball és el desenvolupament d’un mètode procedimental per a la generació d'una illa de forma aleatòria però coherent, podent alterar un conjunt de paràmetres per a aconseguir certes característiques en aquesta àrea de treball, tals com l'altitud sobre el nivell del mar, superfície, rugositat del perfil, etc. Aquest treball serà portat a terme en C# fent servir Unity per a analitzar els resultats del mètode proposat i per a modelar la interfície d'usuari.

## 1.1. Àmbit del projecte

Aquest projecte ens situa entre dos aigües. Si be la temàtica es centra en el camp de l’algorísmica el fet de que el resultat final sigui aplicat amb una representació tridimensional ens apropa també a la visualització de dades per computador.

En el transcurs d’aquest grau ens hem trobat amb múltiples assignatures que ajuden a la comprensió d’aquesta tasca. La primera i mes obvia es, potser, algorísmica i algorísmica avançada ja que gran part del projecte es basa únicament en la capacitat del programador per a resoldre certs problemes amb mètodes eficients.

També la matèria de Gràfics per Computador ajuden en el moment que es comencen a tractar els models 3D. El coneixement de l’estructura d’un d’aquests models a nivell teòric ens ha permès manipular-los lliurement via codi tot i que l’aproximació de Unity es molt mes senzilla de tractar que un sistema complet de motor gràfic com pot ser OpenGL o DirectX.

Finalment, alguns dels problemes presentats haurien pogut ser solucionats via paral·lelització i concurrència però per a no desviar massa el projecte i no entrar en camps que no son propis de la idea original hem preferit deixar-ho com a possibles ampliacions. A mes a mes, Unity ens proveeix d’un compilador propi que no assegura el funcionament de totes les característiques del llenguatge de programació que es fa servir.

## 1.2. Motivació

Originalment aquest la idea d’aquest projecte té origen a un dubte personal: ¿com s’ho fa Minecraft per a generar aquests paratges? Minecraft es un videojoc en el que els usuaris es troben amb mons voxelics generats de forma completament procedimental de tal manera que cada un dels jugadors viu una experiència única i totalment nova amb pràcticament infinites variacions. Però aquests mons no son aleatoris, ja que tot te una relativa coherència.

I no nomes Minecraft, son molts els títols que opten per a la generació procedimental com a marca insígnia per a oferir molt mes contingut del que podrien amb el pressupost del que disposen. Però en molts casos aquests algorismes son pobres i limitats i poden no oferir una experiència equilibrada de joc.

El mes important de tots es, possiblement, *No Man’s Sky*. Producte encara en desenvolupament en el que l’usuari es troba viatjant de planeta en planeta d’una galàxia en mida real en la que cada astre i planeta esta creat d’aquesta manera.

## 1.3. Objetius generals

A aquest projecte de desenvolupament ens plantejem la creació d’un algorisme de creació d’entorns aleatoris. Donat que aquest camp es molt ampli ens centrarem en la generació de terrenys i, mes concretament, en illes. Aquesta decisió ens permetrà simplificar ja que no es te en compte el tipus de frontera.

## 1.4. Objectius especifics

Desglossar els objectius generals en subobjectius que es recullen en els diferents apartats de la memòria

## 1.5. Organitzacio de la memòria

Breu descripció del contingut de cadascun dels capítols de la memòria.

# 2. Antecedents

**2.1 Resum del treballs més recents**

De "A survey on Porcedural Modeling fo Virtual Worlds" podem veure que la necessitat de generar estructures de tot tipus de forma procedural neix fa mes de trenta anys. Una gran varietat de models poden ser creats amb aquests métodes: textures, plantes, terreny, edificis, arees urbanes, xarxes viaries, conques fluvials, etc.

La principal raó d'aquest creixement es troba en la Amplificació de Dades el qual vol dir que a partir d'un petit set de dades o un conjunt de parametres podem obtenir una font enorme de contingut.una altra de les caracteristiques importants es la Compressió de Dades que implica que mitjançant un conjunt petit de parametres podem generar un nou model, dades molt mes senzilles de desar que el format del model mateix. Però aquesta generació sovint es estocàstica raó pero la qual aquest conjunt de dades nomes definira unes caracteristiques que crearan un set de models amb les mateixes qualitats però tots diferents entre sí.

En el context de la generació de mons virtuals trobem molts sistemes i algorismes que de formes diferents ens permeten crear el que necessitem

* Terreny

La forma mes habitual de representació d'un terreny son els mapes d'alçades (Height Map), una matriu de dues dimensions on cada cel·la representa l'elevacio al terrreny original. Son facils de comprimir i manipular per la GPU però es veuen limitats al no poder representar estructures sortints ni coves. Per a solucionar aquest problema es van crear les Estructures per Capes de Dades(Layered Data Structures), Voxels o Mesh 3D. Finalment Peytavie ha desenvolupat un model que permet la representació de sortints, diferent materials i roques soltes.

Els primers algoritmes de generacio de de heightmaps inroduien una porció controlada d'aleatorietat per a generar una elevació detallada. El primer algoritme de subdivisio fet servir es conegut com a Midpoint Displacement Method en el que l'altura d'un punt es calculada amb la mitjana dels seus veins en forma de triangle o diamant.

Hi ha altres metodes estocàstics com per exemple els de generació de soroll com el soroll de Perlin i variacions d'aquest anomenats Moviment Brownià que van be per a generar crestes o petits turons.

Tot i així aquests metodes es veuem limitats per el fet de que els parametres a fer servir no son intuitius i pot fer que generar el contingut desitjat sigui una mica mes dificil de que podria ser. Però, en contraposicio, el rendiment del soroll del Moviment Brownià es extramadament alt i fàcil d'optimitzar i paralelitzar ja que el valor de cada punt no depen dels seus veins.

Zhou et al. va desenvolupar un algorisme que genera terrenys a partir de línies dibuixades per l'usuari que poden definir crestes o d'altres elevacions contínues. L'input es intuitiu peró el métode no esta recomanat si volem control sobre petites unitats de terreny. A mes a mes el sistema, com cada métode basat en exemples, esta limitat per els conjunts d'exemples desats.

També trobem altres sistemes mes especifics que fan servir dades introduides per l'usuari, com regions delimitades amb un dibuix, o que serveixen per a generar un sol tipus de terreny senzills com pot ser una muntanya pero amb molt de detall i facilitat de parametrització.

**2.2. Treballs relacionats amb illes**

La universitat de Stamford, un dels sus grups d’alumnes, ha creat un projecte de generació aleatòria d’illes. [link]

Aquest projecte fa una aproximació matemàtica, molt mes complexa que la desenvolupada aquí, i en dues dimensions que genera un mapa d’alçades molt complex però realista.

L’algoritme comença generant un nombre concret de centroides i els fa servir per generar polígons de Voronoi. Per a evitar que sigui tot una mica massa aleatori aquests centres aleatoris els genera mitjançant l’algoritme de Lloyd.

A partir d’aquests centres i arestes es generen dos grafs. Un d’arestes i l’altre de centres, sent aquest últim el que es fa servir com a perfil i cel•les de l’illa

Un cop seleccionada la forma de l’illa, quines cel•les seran aigua i quines terreny, es generen tres mapes diferents:

1. Mapa de terreny: que es terra, aigua interna i mar. El desenvolupador ho fa amb sinus, blobs, soroll de Perlin o siluetes entrades manualment.
2. Mapa d’alçades: Generat per terreny mes lluny de costa com el mes alt i mes proper com el mes baix. A partir d’aqui i seguint lògiques d’alçada(arestes que equivalent a baixants cap al mar) es pode pendre algunes arestes per a generar un riu.
3. Mapa d’humitats: finalment aquest sistema es genera amb la proximitat a aigua. Com mes distant menys humitat i es fa servir per a poder assignar biomes.

Arribat a aquest punt ja tenim una illa completament generada, a la qual nomes hi podem afegir millores com una representació mes realista del mapa mitjançant soroll, generació de carreteres i camins, creació de centres de població entre d’altres. Al projecte de Stamford totes aquestes millores ja es duen a terme tot i que per al que necessitem a aquest projecte ja en tenim prou.

**2.3. Conclusió**

Com veiem, cada vegada i desde fa molt de temps, hi ha sistemes de generació procedimental mes complexos, complets i que generen resultats mes realistes. Es veritat, també, que els mes realistes venen definits per infinitat de paràmetres i que els mes senzills d’utilitzar son massa homogenis però tot i així ens anem apropant a punts ideals on objectes, plantes o mapes poden ser generats per ordinador amb una intervenció humana mínima alleugerint els costos de disseny gràfic.

# 3. Anàlisi

La illa que generarem es veurà definida per les següents característiques partint de la idea que genera un cert nombre de blobs els quals donen alçada a un pla:

* + **Costes**

Podem escollir la forma bàsica de les costes entre cercles i quadrats.

* + **Distinció entre mar i terreny**

Partim d’un pla d’alçada zero i tot allò que sigui mes alt que zero serà considerat terreny

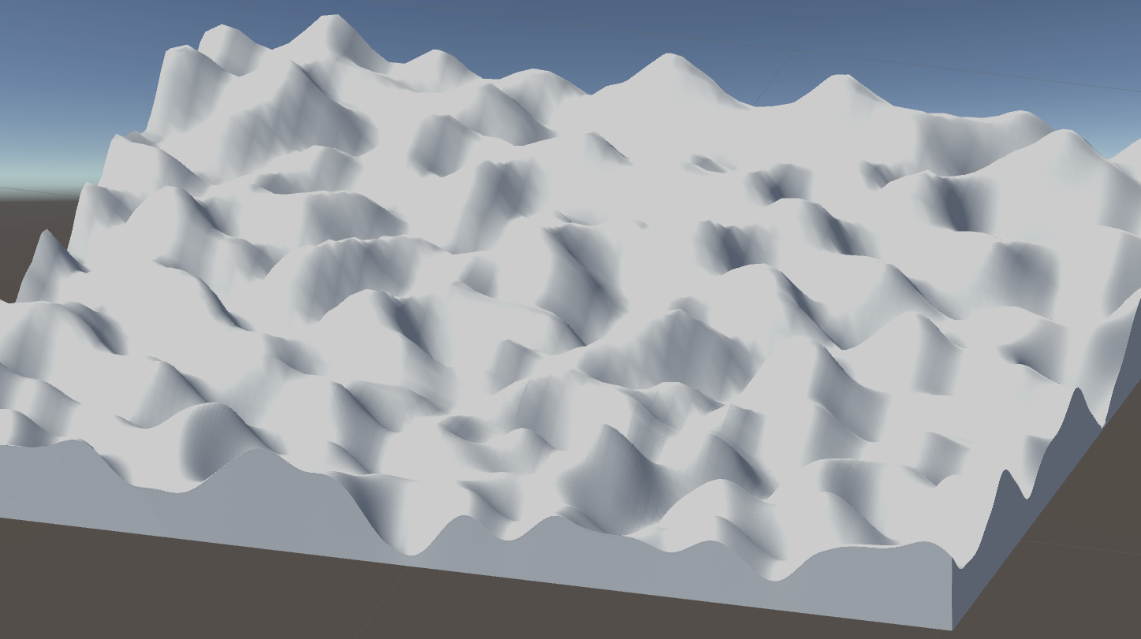
* + **Distribucio de biomes**

Distribuits per alçada

Els biomes es generaran per alçades amb cert soroll. Començant per sorra/platja i acabant per cims glaçats als punts mes alts.

**4. Disseny**

A l’inici del projecte es comença plantejant la generació de mapes d’alçades, el sistema mes senzill de generació de terrenys però alhora limitat, i els sistemes de modelatge de Unity.

Comencem acceptant que la millor idea es fer servir el sistema Terrain de Unity. Aquest sistema genera un model tridimensional a partir d’un mapa d’alçades directament. Aquest es un model totalment suau i de mida pràcticament il•limitada, a diferencia dels Mesh que estan limitats a 64000 vertexs, aproximadament.

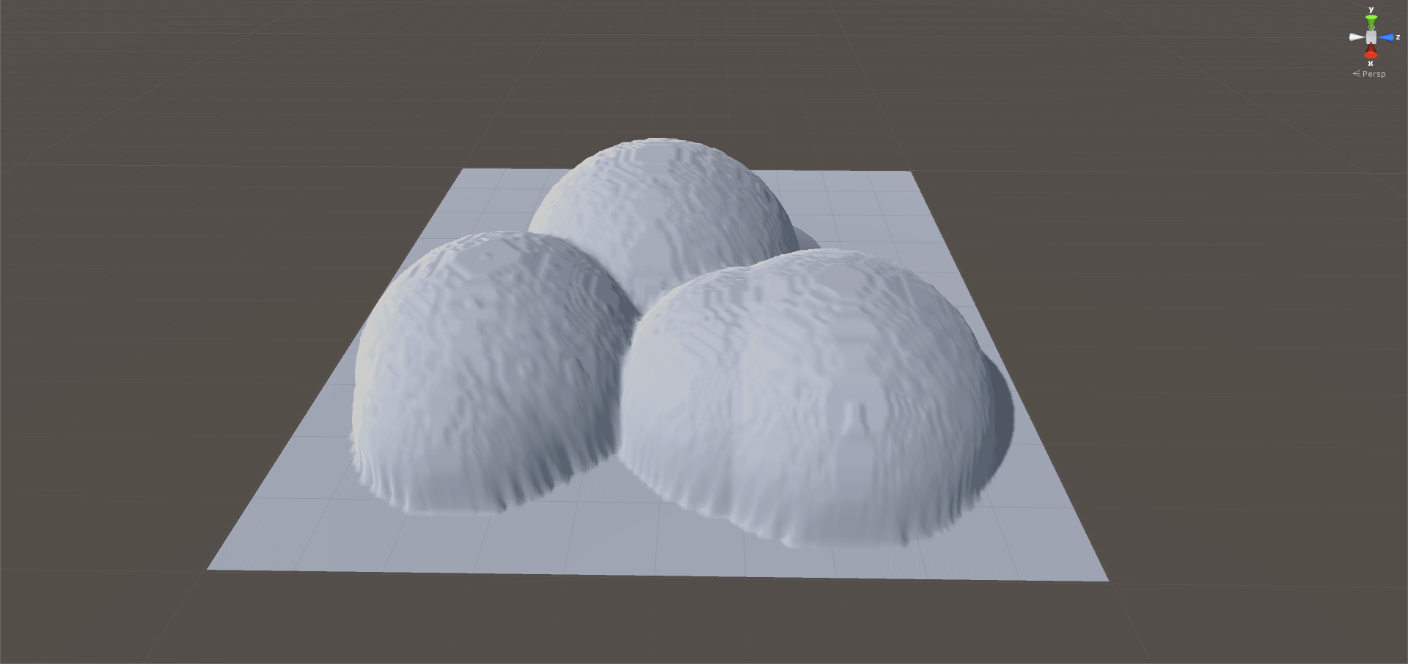
Per a generar aquest mapa optem per a fer servir el sistema de blobs que consisteix en crear taques de a unes coordenades i d’una mida aleatòries. Per a começar aquests blobs son semiesferics i l'alçada al punt objectiu es sumada per a cada un dels blobs trobats, per tant es dificil que trobem una semiesfera perfecta a no ser que generem un sol blob o que aquests molt separats.

El primer problema amb el que em vaig trobar en treballar amb HeightMaps i Terrains es que aquests nomes accepten dades percentuals, compreses entre 0 i 1, i aquest mapa d'alçades nomes pot ser quadrat i amb costats potencia de 2. La solució que s'em va presentar va ser tractar-ho com una imatge en escala de grisos i fer un reescalat, ja sigui a major o menor. D'aquesta forma el primer que generariem seria un mapa d'alçades amb valors generals, de mon, i acte seguit normalitzariem aquestes dades. Aquest metode ens obliga a tenir sempre visibles els valors d'alçada, amplada i profunditat amb els que estem treballant així com els valors de resolució objectiu.

Unity, tot i treballar amb C#, no te del tot integrades les llibreries standard i algunes les ignora o les subtitueix per les própies. Es recomanat fer sempre ús de les que ens proporciona el propi Unity ja que no s'assegura la compatibilitat completa del programa i poden sorgir alguns problemes a l'hora d'executar i compilar. Dit aixó, per a tractar aquesta imatge que es el mapa d'alçades fem servir la classe Texture2D, que ens proporciona un métode de canvi de mida anomenada Resize molt prometedora peró que no ens serveix ja que la única cosa que fa es modificar la mida del objecte i, si conté dades, les reinicialitza.

Amb una petita cerca a internet ens trobem amb altres usuaris que es troben amb el mateix problema i trobem la solució la comunitat de Unity, un script anomenat TextureScale que ens permet reescalar una imatge amb o sense interpolació bilinear.

Un cop tenim el mapa d'alçades ja el podem aplicar al Terrain. El resultat es el seguent:



Ara, passem a generar els biomes. He decidit, de forma arbitraria que les variacions climatologiques vinguin donades per l'alçada. Com es habitual en molt tipus d'illes ens trobem, d'abaix a d'alt:

|  |  |  |
| --- | --- | --- |
| Aigua | 0% |  |
| Platja | <= 15% |  |
| Vegetació espessa | <= 50% |  |
| Vegetació Lleugera | <= 75% |  |
| Roca | <= 95% |  |
| Neu | < 100% |  |

Peró, com representem aquests biomes? Amb textures, el principal problema es que Terrain fa servir un sistema propi de textures, que requereix crear una textura per a cada forma creada. La solució per a evitar aquests canvis ve amb el canvi de Terrain a Mesh. Unity no disposa, oficialment, de cap métode per a convertir terreny a mesh. Peró cercant per les comunitats de Unity en trobem un per exportar a \*.obj qualsevol terrain desenvolupat per Miker Hergaarden.

Aquest algorisme ja fa la feina dura, generar vertexs, triangles i UVs del mesh, per tant es facil de modificar per a que faci el que necessitem. Elminem totes les referencies al UnityEditor, que integraria el sistema a la interficie d'usuari de l'editor i assignem aquestes dades a un Mesh que retornem al sistema original.

D'aquí, nomes ens queda cercar els triangles que considerem que estan cada bioma i assignar-los a un Submesh. Els Submesh son llistes de triangles del Mesh de Unity a cada un dels quals li pertoca un Material diferent, també llistats a l'objecte Mesh. D'aquesta manera generem els amterials amb les textures que necessitem i assignem cada un al submesh corresponent, es a dir, al bioma que li pertoca.

Al seguent apartat el que hem fet es cercar métodes de fer-ho mes realista. La generació de semiesferes ens crea uns terrenys molt poc creibles, molt similars als mons de fantasia de Bola de Drac. La solució es troba en generar altres tipus de perfils. Optem per la silueta de la Campana de Gauss ja que el que trobem a faltar es costa i el faldó allargat d'aquesta forma ens dona la longitut de platja que necessitem. Com que el valor de cada coordenada es la suma de les alçades de tots els blobs que es troben allà tampoc trobem progressio de forma constant i aixó ja aplica un cert soroll que resulta en uns paissatges mitjanament interessants.

Tot i aixó, tornem enrrere al moment en que el mapa d'alçades es generat, just abans de ser aplicat i pensem, ¿que en podem fer per a que sigui una mica mes interessant? La resposta es troba al soroll. Mes concretament al Soroll de Perlin. Aquest metode ens dona un valor entre 0 i 1 per cada conjunt de coordenades x/y que li donem. Recorrem tot el mapa d'alçades i multipliquem el valor de cada coordenada d'aquest per el seu equivalent al Soroll de Perlin, d'aquesta manera aconseguim 3 coses: suavitzar les alçades, fer-ho tot mes irregular i no passar mai de 1, que es el valor màxim que podem tenir al mapa d'altures.

5. Resultats i Simulacions

Aquí mai no va el codi.

Simulacions: descripció del funcionament de l'aplicació amb captures del interface, visualització de resultats, taules de temps i càlcul de memòria requerida. Taules comparatives de temps

# 6. Valoració econòmica (no és un capítol)

## 6.1.Anàlisi del temps de realització del projecte

### - Hores de formació

### - Hores d'anàlisi

### - Hores implementació

### - Hores documentació

## 6.2 Valoració del cost econòmic del projecte

S'han exclòs del cost econòmic del projecte les hores de formació i, de les hores de documentació, les de la memòria, computant únicament les de documentació del manual tècnic i d'usuari.

Cost anàlisi = total hores anàlisi \* preu hora anàlisi ( entre 80 i 120 euros)

Cost programació = total hores programació \* preu hora programació ( entre 60 i 70 euros)

Cost documentació = total hores documentació manuals \* preu hora programació ( entre 50 i 70 euros)

# 6. Conclusió

- Recordar l'objectiu del projecte

- Enumerar les contribucions

- Concloure que s'han assolit els objectius

- Enumerar línies de continuació del projecte

# 7. Referències bibliogràfiques

# Apèndix A: Manual tècnic

## A.1 Instal·lació: Requeriments mínims i passos a seguir

## A.2. Manual del desenvolupador

## Apèndix B: Manual d'usuari de l'aplicació