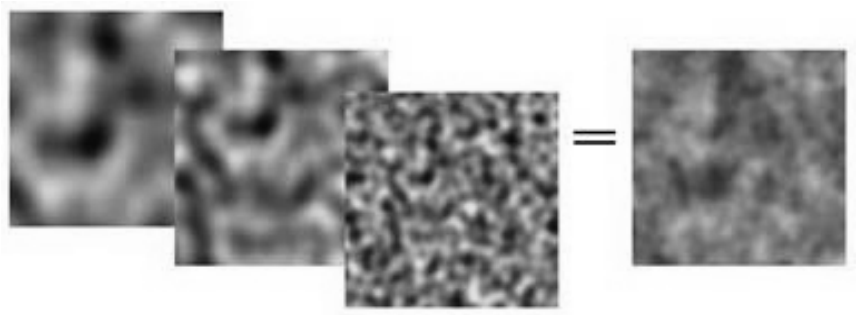
Programozói dokumentáció

Pár alapfogalom:

- seed: A random generálás azonosítója: ebből a számból generálja a véletlen dolgokat. Ez azért hasznos, mert ugyanaz a szám ugyanolyan generálást eredményez.
- biome: A biom alapján lesz a bolygó kiszínezve. Minden biomhoz tartozik egy magassági szint (0-tól 1-ig valós szám), és egy szín. A program az adott pontot a magassága alapján színezi ki: az adott magassághoz az azt körbezáró két biom színének keverékét rendeli. Egy példa.: ha az első biom 0 magasságú és szürke, a második 0.5 magasságú és fehér, akkor a 0.25-ös magasságú pontokhoz világos szürkét rendel, mert a 0.25 félúton van az első kettő biom között. Ügyelni kell arra, hogy a biomok szintje növekvő sorrendben legyen, különben a program nem megfelelően működik.
- szín: a színeket hexadecimális RGB kódjukkal kell megadni a konzolban
- terrLayer: a domborzat érdességéhez több rétegnyi zajt használunk. Ezeknek az skáláját, (mennyire “sűrű”) és súlyait(a végleges zajba milyen szorzóval kerülnek) tudjuk állítani



.

- bolygófájl: egy .plnt kiterjesztésű fájl, ami tartalmazza a bolygó domborzatát, és kiszínezési szabályait.
- bolygóparaméter-fájl: egy szöveges fájl, ami olvasható módon tartalmazza a bolygó paramétereit. A fájl **param** előtagű parancsokból áll, azonban nincs kiírva az előtag.

A program főbb struktúrái

struct PlanetData

A program fő struktúrája a PlanetData, ez felel az adatok tárolásáért:
A mezők feladata a következő:

- **int dataW,dataH**
  Eltárolják a magasságtömb dimenzióit.
- **ushort** heightData**
  Kétdimenziós ushort (16 bites unsigned integer) tömbre mutat, itt van eltárolva a bolygó magassága egy adott ponton.

- **PlanetParam* planetParam**
  Egy PlanetParam struktúrára mutató pointer. Ebben a struktúrában vannak eltárolva a bolygó színei.
- **ColorList *terrainColors**
  Ez a dinamikus lista tárolja el a bolygó biomait.
- **Vec3** sphereMap**
  Ez egy futásidőben generált kétdimenziós tömb. Eltárolja a bolygó szélesség-magasság koordinátához tartozó térbeli pontokat. Ez segíti a programot a gyorsabb futásban.

struct GenSettings

A feladata, hogy eltárolja a generáláshoz szükséges adatokat:
A mezők feladata a következő:

- **int32 seed**
  egy 32 bites integer, ez a véletlen generálás seed-je
- **int continentCount**
  kontinensek száma
- **int minContinentNodes**
  kontinens minimum részletessége
- **int maxContinentNodes**
  kontinens minimum részletessége
- **int edgeLayerCount ,**
  float *edgeWgh
  a kontinensek határának érdességét befolyásolják
- **int terrLayerCount,**
  float *terrScl,
  float *terrWgh
  a domborzat érdességét határozzák meg

struct CommandList

Ez a láncolt lista felelős azért, hogy a parancsokat eltárolja. Sok fgv. nem tartozik hozzá, mivel statikus, így nincs értelme elemeket kivenni és módosítani futásidőben.

- **int id**
  A parancs azonosítója. Mindegyiknek egyedi kell legyen. Ez alapáj lehet switch-blokkba rakni a parancsokat.
- **char *command**
  A parancs neve
- **struct CommandList* nextElement**
  Köv. elem.

A program fontosabb részei

commands.c

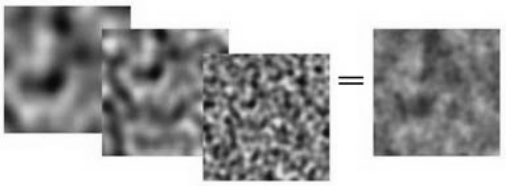
Ez a fájl kezeli a felhasználó által beírt parancsokat.
Fontosabb függvényei:

- **int executeCommand(CommandCache *commands, char *command, SceneSettings *sceneSettings, GenSettings *genSettings);**
  Ide érkezik be a parancs a *main*-ből. Ennek a fgv.-nek a feladat szétosztani előtagonként.
- **int executeParamCmd(...); int executeGenCmd(...);**
  int executeImportCmd(...); int executeExportCmd(...);
  Ezek a fgv-ek hajtják végre előtagonként külön-külön a parancsokat.

perlin.c

Ez a fájl felelős a véletlenszerű generálásért.
Függvényei:

- **float perlin(float x, float y, float z, int32 seed)**
  Ez a függvény egy -1 és 1 közötti véletlen értéket generál a Perlin-féle algoritmussal. Tulajdonsága, hogy ugyanazon bemeneteire ugyanaz az eredmény, ezért ideális generáláshoz. Az eredeti algoritmust Ken Perlin írta. (https://hu.wikipedia.org/wiki/Perlin-zaj)
  Paraméterek:
  - **x,y,z:**
    Térbeli koordináták
  - **seed**
    A generálás seed-je.
- **float layeredPerlin(float x, float y, float z, int32 seed, float layerWeights[], float layerScales[], int layerCount)**
  Ez a fgv. többrétegű perlin-zajt generál. Ez eredményezi, hogy valósághű lesz a bolygó felszíne.



  Paraméterek:
  - **x,y,z:**
    Térbeli koordináták
  - **seed**
    A generálás seed-je.
  - **layerWeights**
    A rétegek súlya, azaz hogy mennyit számítanak a zajba.
  - **layerScale**
    A zaj rétegenkénti nagyságát adja meg (mennyire “nagyítunk bele”)
  - **layerCount**
    A rétegek száma

heightgeneration.c

Ez a fájl felel a domborzat generálásáért.
Legfontosabb függvénye:
void generateHeightData(PlanetData* planetData, GenSettings* settings)
Ez a fgv. generálja le a domborzatot a megadott beállítások alapján (settings).

rendering.c

Ez a fájl felelős a kép rendereléséért.
Fontosabb függvényei:

- **int renderFrame(RenderSettings *renderSet, const SceneSettings *sceneSet)**
  Ez a fgv. rendereli ki a képet. Képpontonként megy végig a buffer-en, és hívja meg a *getColor(...)* fgv-t.
- **RGB getColor(int x, int y, const RenderSettings *renderSet, const SceneSettings *sceneSet)**
  Ez a függvény mondja meg, hogy milyen színű legyen egy adott pixel.
  Úgy működik, hogy megkeresi, hogy a bolygó melyik pontjára mutat a pixel. Ezután a *planetData*-ból kikeresi a hozzá tartozó magasságot, majd az alapján kiszínezi.

dataio.c

Ez a fájl felelős az adatok mentéséről és olvasásáról.
Függvényei:

- **int exportTexture(char *filename, PlanetData *planetData)**
  Ez a fgv menti el a képet textúraként, azaz hogy melyik ponton a bolygón milyen szín tartozik.
- **int exportHeightmap(char *filename, PlanetData *planetData)**
  Ez a fgv menti el heightmap, azaz magasságtérképként a bolygó felszínét. Ez egy fekete-fehér kép, ahol a 0 a legkisebb, a 255 a legmagasabb pontot, az ezek közötti értékeket meg 0-255 közötti értékek jelképezi
- **int exportSattelite(char *filename, SceneSettings *sceneSettings)**
  Ez a fgv menti el a bolygó felszínét úgy, ahogy az megjelenik. Látszik rajta a bolygó sötét oldala is, és az ott világító városok.
- **int exportNormalmap(char *filename, PlanetData *planetData, bool flatOcean)**
  Ez a fgv normalmap-ként, azaz normálvektor-térképként menti el a bolygót. Itt minden szín egy normálvektort jelképez.
- **int exportShadedMap(char *filename, PlanetData *planetData, float maxShade, float maxAngle, bool flatOcean)**
  Ez a fgv egy térképként meni el a bolygófelszínt, ahol be vannak rajzolva a hegyek által vetett árnyékok.

Könyvtárak

A programhoz íródott kettő könyvtár:

- **rgb.c**
  Ez a könyvtár tartalmazza a program által használt szín-struktúrát, és a vele kapcsolatos fgv-eket.
- **vector.c**
  Ez a könyvtár tartalmazza a program által használt Vec3-struktúrát, és a vele kapcsolatos fgv-eket. Erre a struktúrára a renderelés miatt volt szükség.

Végezhető vele összeadás, kivonás, skalár és vektoriális szorzás, meg lehet mérni távot, vektor-hosszt, és két vektor szögét is meg lehet mérni.