Tartalom:

Normal mapping technikai háttere...

Mi is az a normal map? :

Ehhez elsőnek meg kell néznünk pár alapvető fogalmat.

A legalapvetőbb a **face normal**. A 3d-s megjelenítés alapja a háromszögek kirajzolása, az úgynvezett face-é. Minden face-hez hozzárendelünk egy vektort ami megmutatja a háromszög felülete milyen irányba néz.

Ennek egy továbfejlesztett változata a **vertex normal**, ahol magukhoz a vertexekhez (a face-k három pontjai) rendelünk hozzá egy-egy vektort, mégpedig úgy hogy kiszámítjuk a face normalokból, és egységnyi hosszúságúra állítjuk őket. Ezeket a vertex normalokat a vertex lightning technikához.

A normal map nem más, minthogy (akárcsak egy textúrát) egy **normal mapot** húzunk rá az objektum egész felületére, és ezzel **egy face-n belül is változhat a normalok iránya**. Ezzel

elérhetjük, hogy egy egyszerű egyenes felület is úgy nézzen ki, mintha komoly geometriájú lenne, holott csak a virtuális geometria árnyékait látjuk, maga a geometria nem jelenik meg.

Ezzel gyorsíthatjuk a valós idejű megjelenítést, mivel sokkalta kevesebb face-t kell megjeleníteni, viszont hátránya, hogy teljesen oldalról figyelve a felületet nem látszódik semmi sem.

Magukat a normal mapok adatait, a normalok irányait RGB színként tárolhatjuk, ahogy egy-egy szín jelenti egy koordinátarendszer egy-egy irányát, a 0 érték a -1-et, a 128 a 0-át, míg a 255 az 1-et. Itt természetesen használhatunk nem 8 bites, hanem akár nagyobb skálájú színeket, ezzel javíthatjuk a pontosságot.

A normal mapoknál fontos, hogy minden színértéknek megfelelő vektor pontosan egy hosszúságú legyen, különben hibás lesz a megjelenítés. De szerencsére a különböző normal map generálásra alkalmas programok (mind 3D, mind 2D módszerrel generáltak esetében), normalizálva adják ki ezeket az értékeket.

A normal mapoknak két típusa van.:

1. Object Space Normal: Az objektumhoz képest adjuk meg a normalok irányát (jellegzetesen sokfajta színű), és emiatt, ha az objektum deformálódik, maga a normal nem változik meg. Így magát a fényt is forgatnunk kell a deformációk helyes megjelenítéséhez. De ezt egyszerűen számolhatjuk a fényhez és a kamerához képest.
2. Tangent Space Normal (az elterjedtebb): A lokális koodinátarendszerhez képest kell megadnunk a normal irányát, amit a vertexet segítségével adunk meg. Emiatt ahogy deformáljuk a modellt, maga a normal map is vele változik De ilyenkor a helyes megjelenítéshez konverziókat kell végeznünk a tangent space és az object space normal között.

A tangent space normalnál a (0,0,1), rgb színben (128,128,255), ezért jellegzetesen a tangent space normalban dominál a világos-kék szín. Kék felfelé, a piros balra, a zöld pedig lefelé néz a koordinátarendszerben.

A normal map létrehozására két módszer adott.:

1. Létrehozunk egy alacsony felbontású modellt, majd pedig egy nagyobb felbontásút, majd egy program segítségével (melody, vagy az ati blenderbe beépülő pluginje és még sok más program), különböző raycasting módszerekkel a nagy felbontású geometriát rávetítjük az alacsonyabbfelbontású geometriára. Ebben az esetben egy 5000 polygonos alacsony felbontású modellhez, akár egy 1-2 millió polygonos nagy felbontású modell is tartozhat.
2. Különböző 2d-s rajzoló programokkal elkészítünk egy fekete-fehét úgynevezett height mapot (a fekete szín a felület alatt, a fehér szín a felület fölött van), és ebből pluginek segítségével átalakíthatjuk tangent space normal mappá.

Adott esetekben a magas felbontású modell létrehozása a célszerű, amikor nehéz „látni” a magasságokat, viszont amikor lehetséges érdemes a festés módszert használni, mert sokkalta gyorsabb, csak kevésbé látható előre a végeredmény. (legszebb eredményt a két modszer keverésével lehet elérni)

Tutorialok:

1-es módszer (modellekből):

- **NVIDIA:**
- **ATI:**

Szükségünk lesz a segédprogramokra, amiket az ATI honlapjáról tölthetünk le.: http://ati.com/developer/tools.html

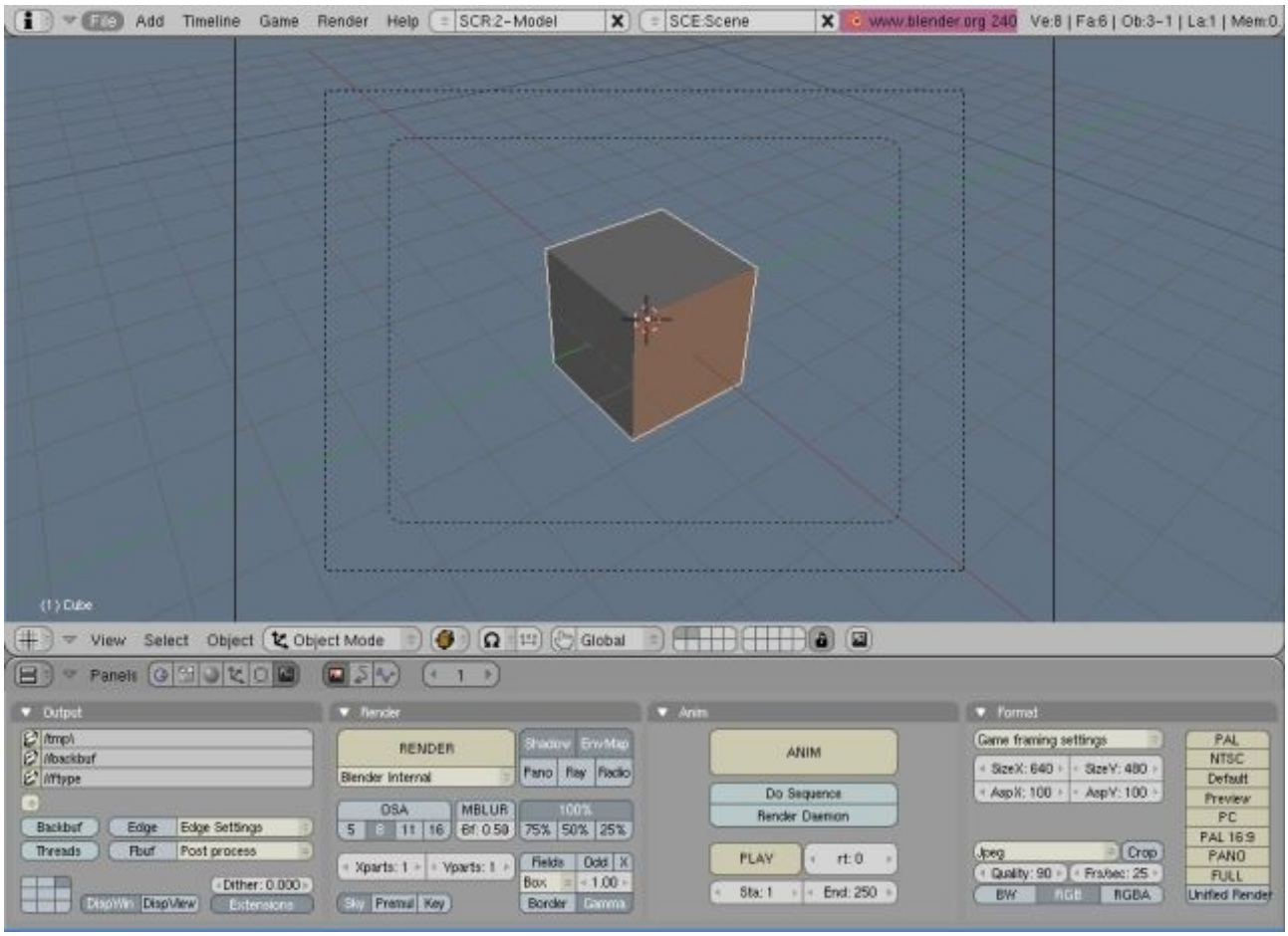
Innen a mi számunkra most a : **NormalMapper** és a **NormalMapperUI - Stand alone** a fontos.

A Blenderhez még egy python script-re is szükségünk lesz ami a modellekből a program számára legenerálja a szükséges nmf fájl-okat.

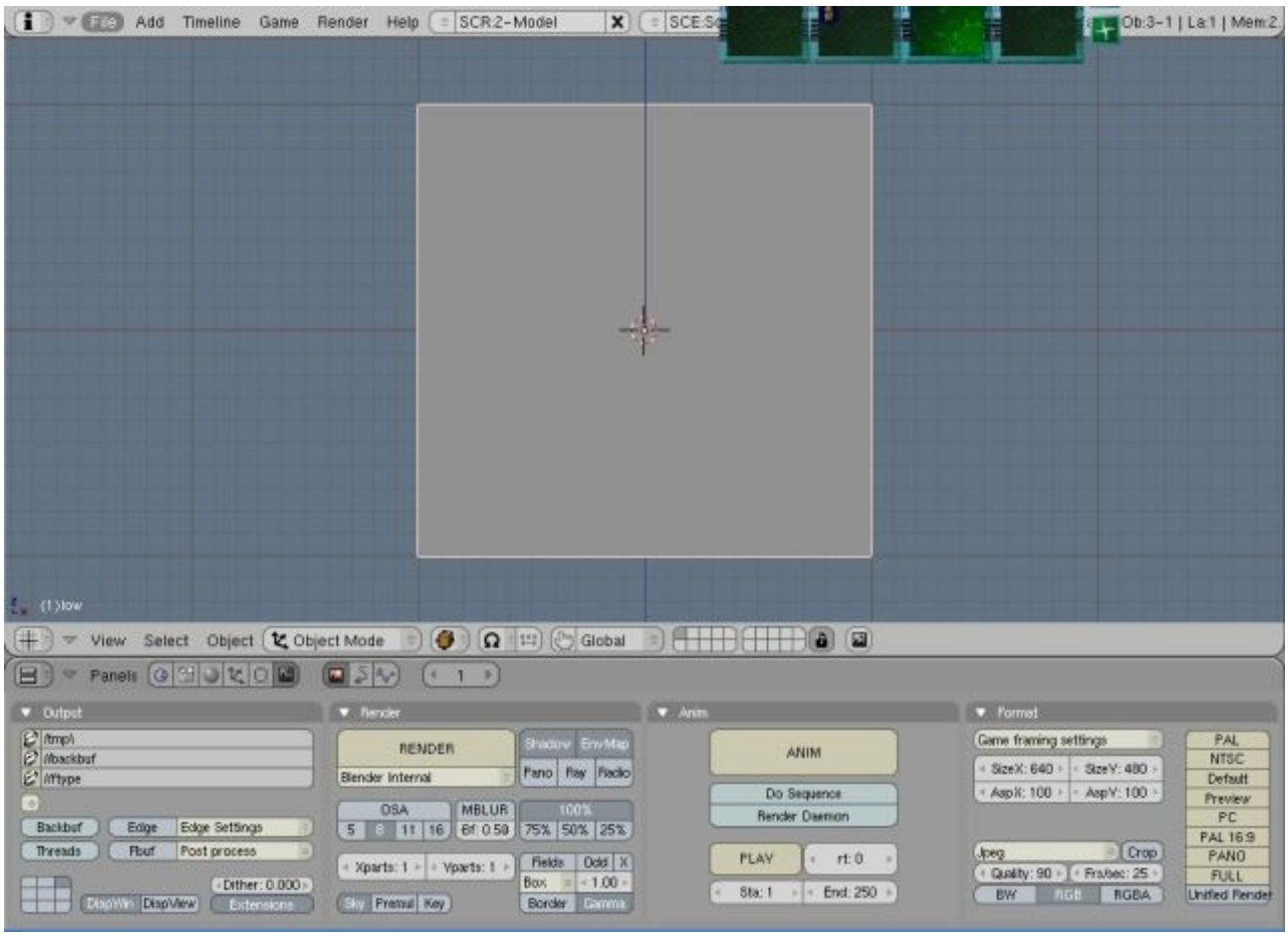
Amit innen tölthetünk le: http://neuralfuzz.com/opengl/NormalMapper.zip (ez a program alkalmas önmagában is a normalmap létrehozására)

Akkor kezdjünk is neki :

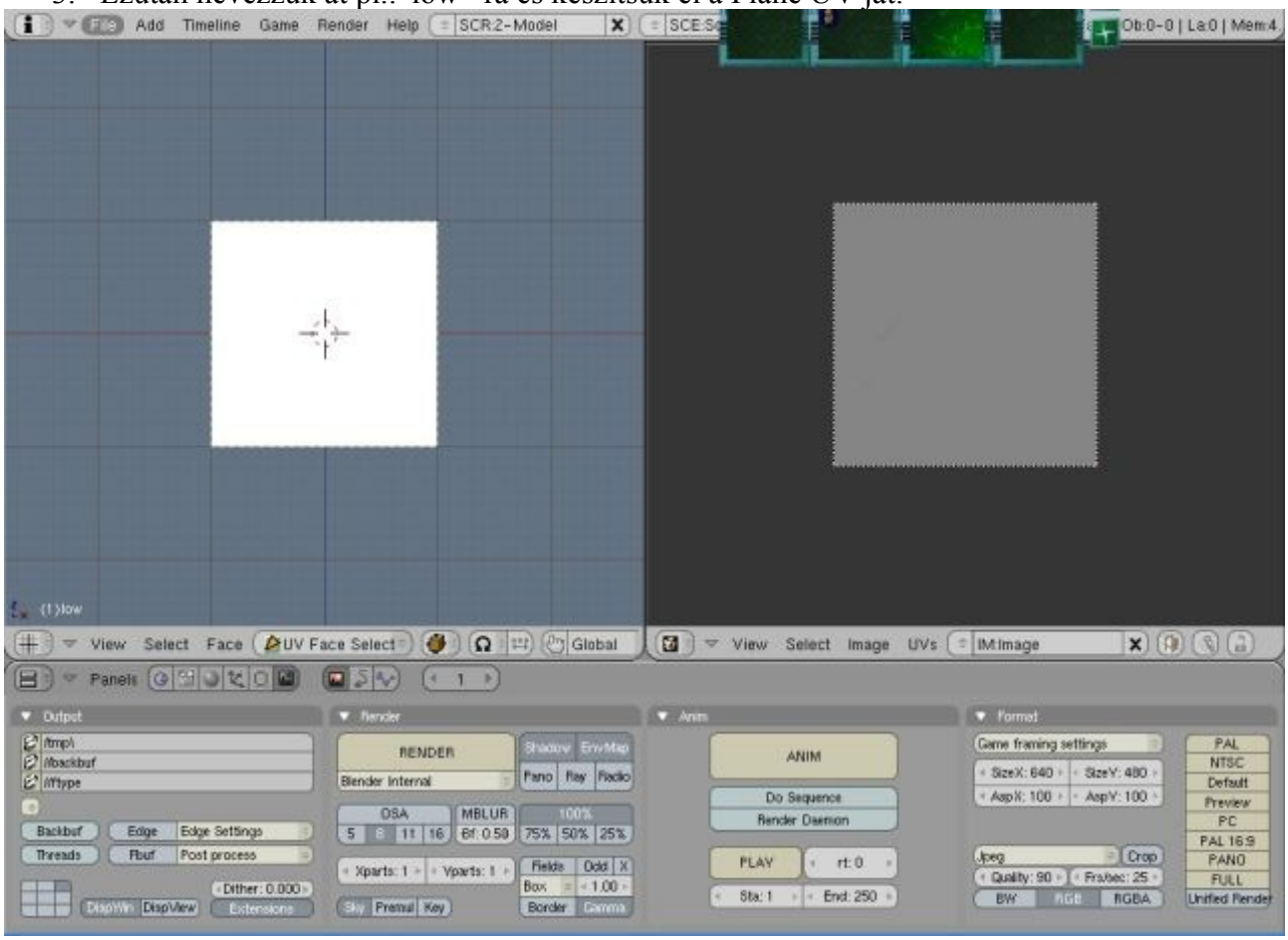
1. Indítsuk el a Blendert:



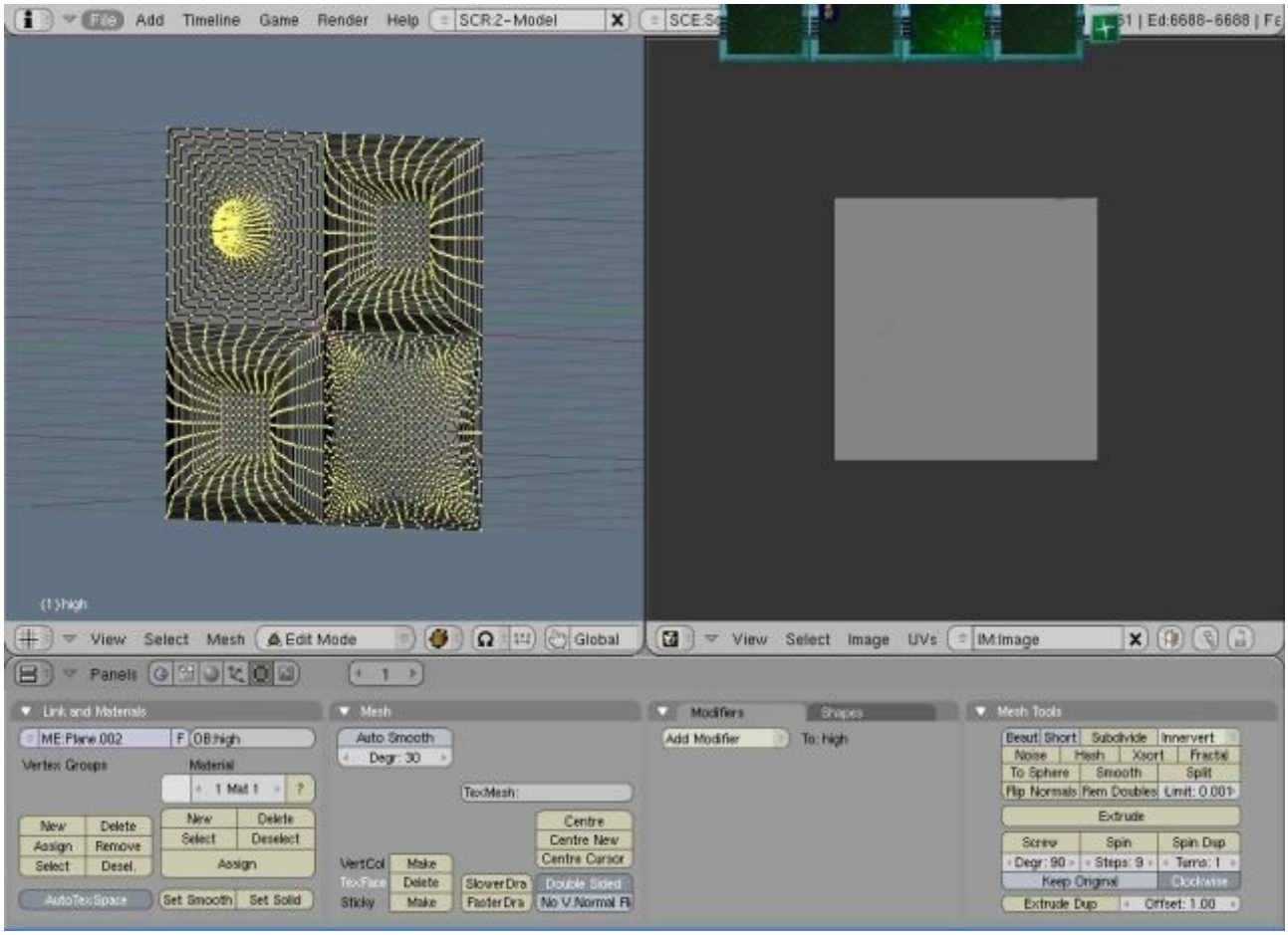
2. Töröljük le az alap kockát, és hozzunk létre egy „Plane”-t.



3. Ezután nevezzük át pl.:”low”-ra és készítsük el a Plane UV-ját.

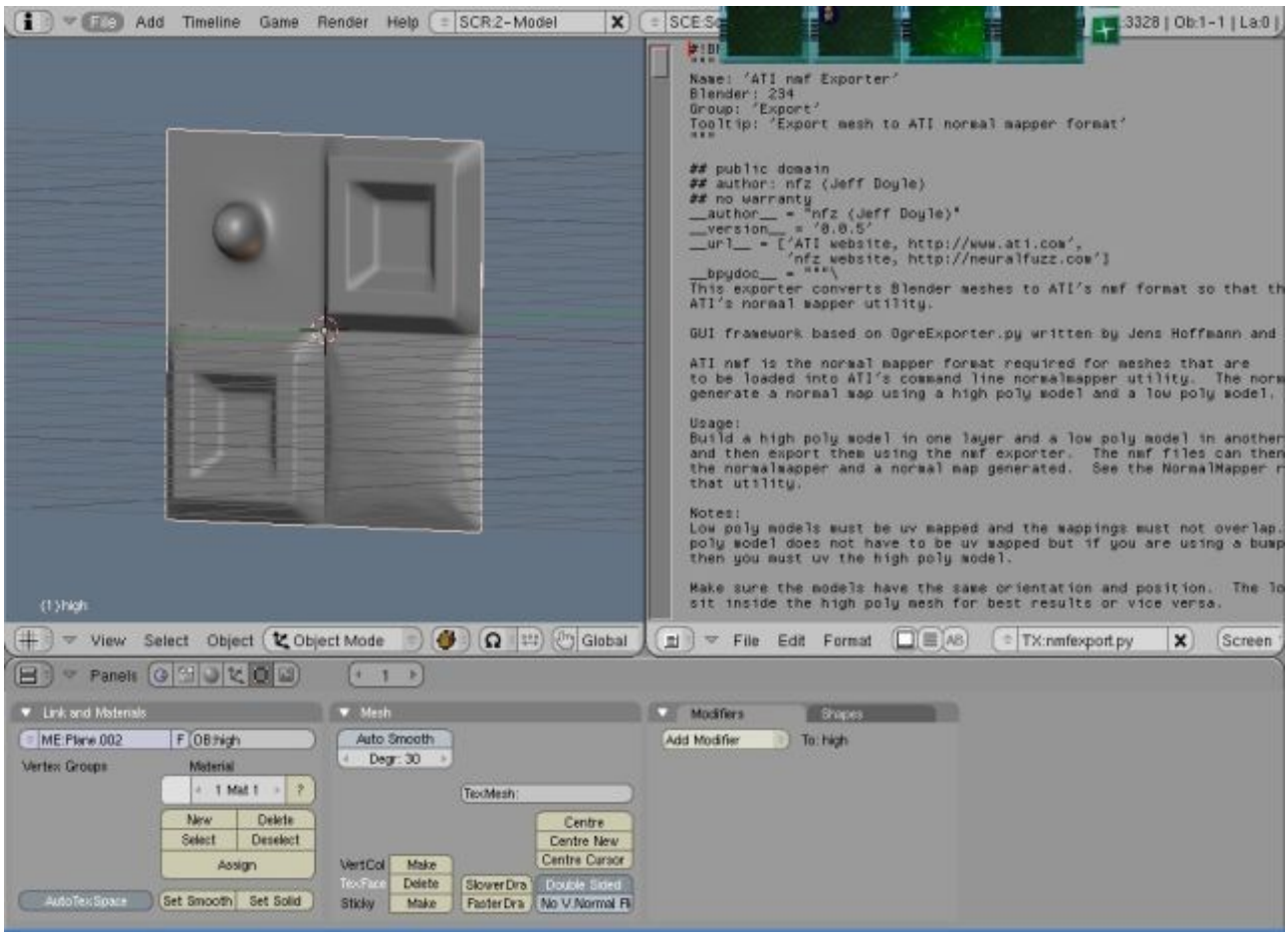


4. Következő lépésben csináljunk egy másolatot a Plane-ről (Shift+D)és nevezzük el pl.:”high”-nek. Helyezzük el egy masik layer-en, majd csináljunk belőle egy részletesebb modellt.

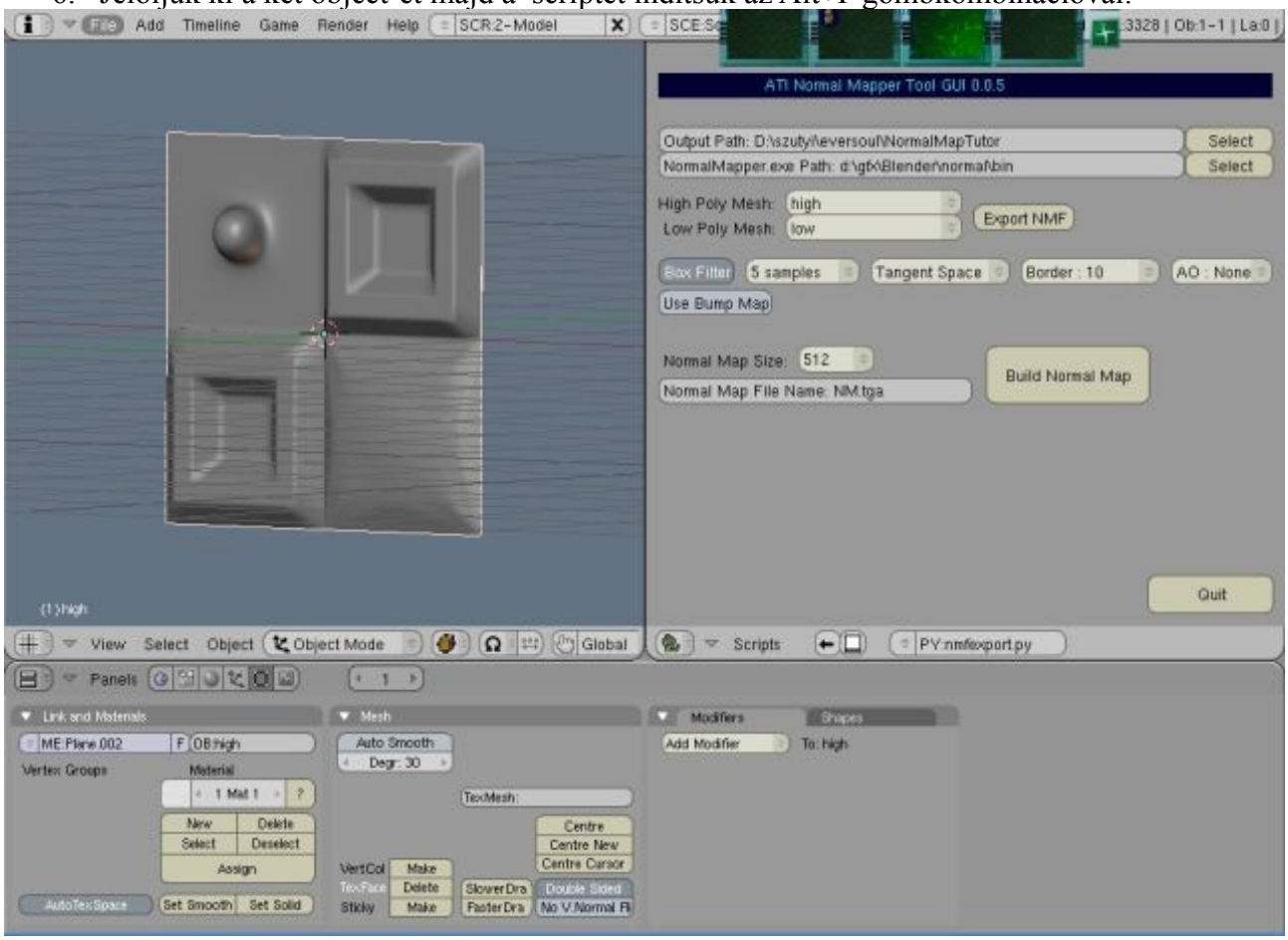


Minél részletesebb annál jobb.

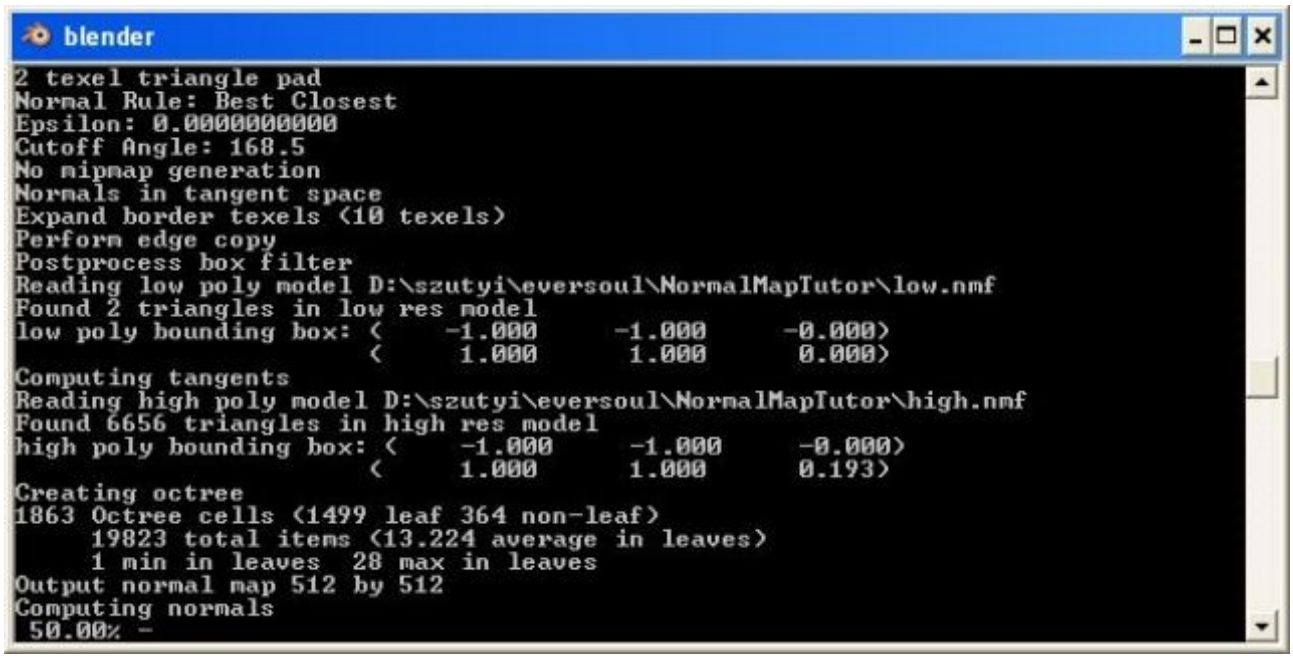
5. Most jön az a rész amikor szükségünk lesz a Python scriptre. Töltsük hát be!



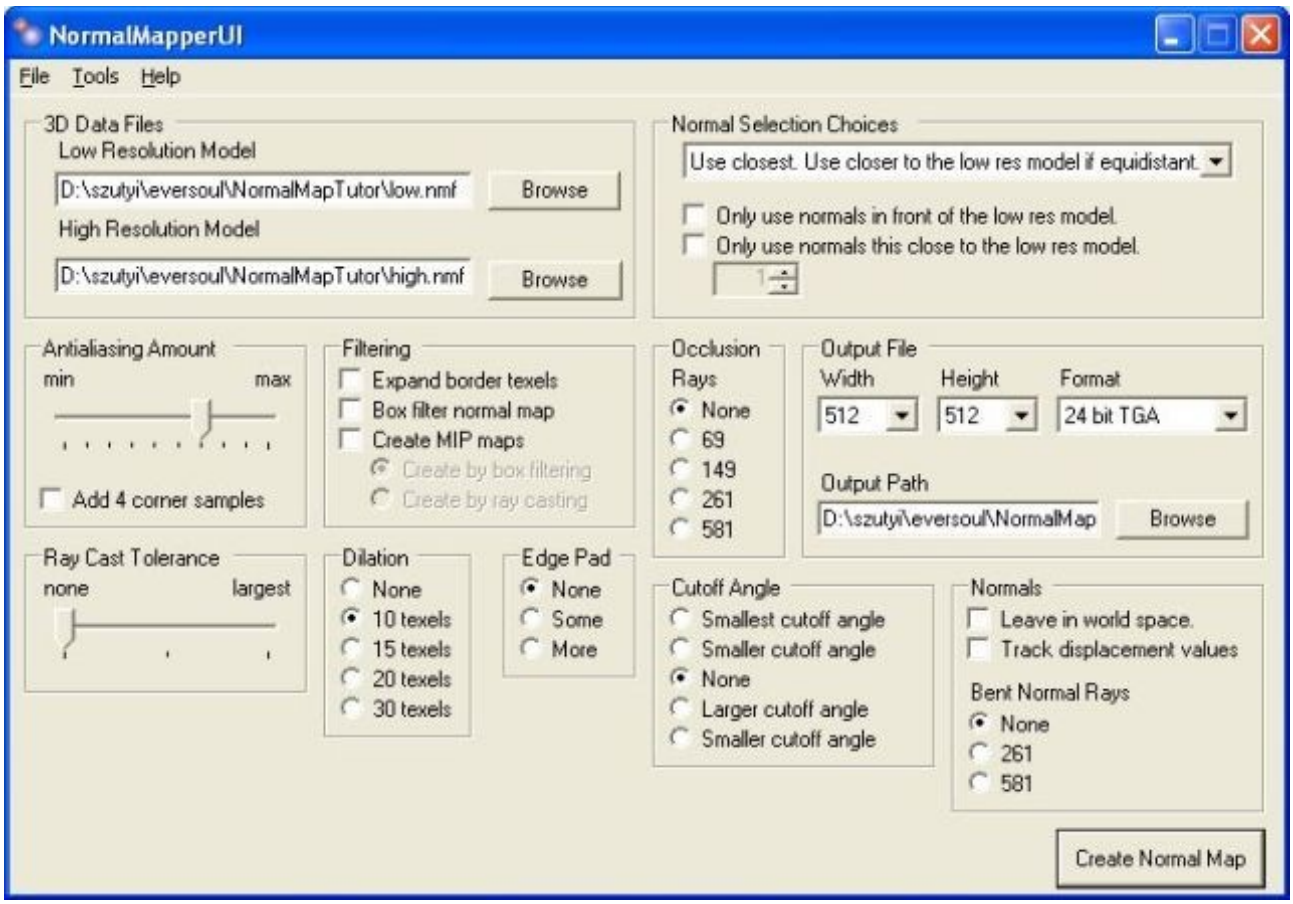
6. Jelöljük ki a két object-et majd a scriptet indítsuk az Alt+P gombkombinációval.



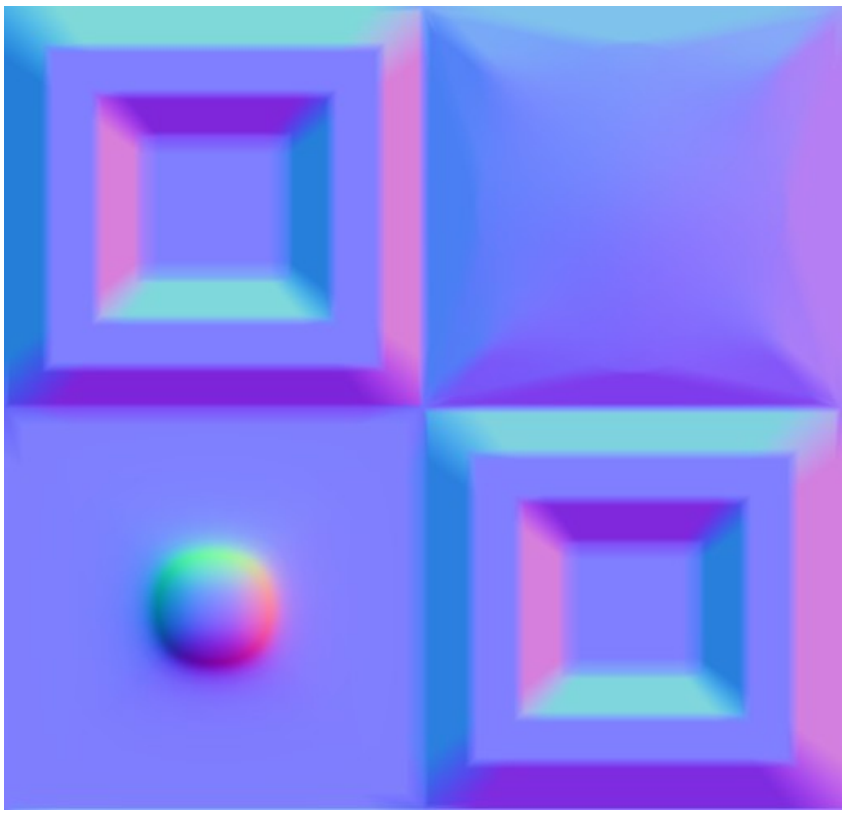
7. Töltsük ki a hiányzó adatokkal: High Poly Mesh:”high” Low Poly Mesh:”low”
   (ha másképp nevezted el akkor értelemszerűen azokat a neveket írd be)
   A NormalMapper.exe Path: Ahová ezt kicsomagoltad.
   (ez csak akkor kell, ha NormalMapperUI nélkül szeretnéd létrehozni a normalmap-ot)
   NormalMap fájl: Ide azt a fájl nevet írd amilyen néven el akarod menteni a normalmapod.
   (ez szintén csak abban az esetben kell ha NormalMapperUI nélkül szeretnéd megcsinálni)
   Első lépésben nyomjuk meg az „Export NMF”-gombot.
   Két NMF kiterjesztésű fájl jön létre, az objectek neveivel ebben az esetben: low.nmf és high.nmf
   Ezek kelleni fognak az UI-hoz. Ha itt akarjuk elkészíteni a mapot akkor nyomjuk meg a „Build Normal Map” -gombot. (A Blender konzolján figyelemmel kísérhetjük a map elkészülését.)



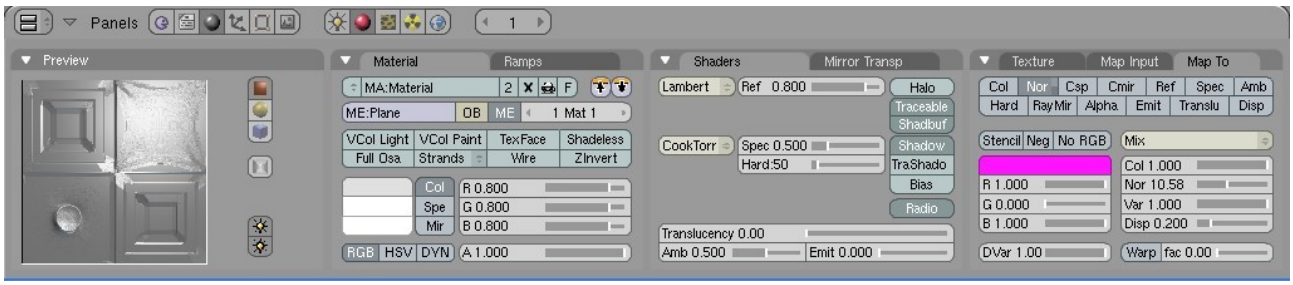
   Abban az esetben, ha az UI-val szeretnénk létrehozni a Normal Map-ot. Indítsuk el a programot.



Ezt is az előzőhöz hasonlóan töltsük ki, majd hozzuk létre a map-ot a „Create Normal Map” gomb lenyomásával. A kimenetet ugyancsak egy parancssori konzolon kísérhetjük figyelemmel.
A végeredmény kb. a következő lesz:



Végezetül ellenőrizhetjük a a low poly modellünkön ha betöltjük textúrának az uvzást használva és alkalmazzuk „nor”-textúraként.


Panels
1
Preview
Material
Ramps
MA:Material
2
F
ME:Plane
OB
ME
1 Mat 1
VCol Light
VCol Paint
TexFace
Shadeless
Full Osa
Strands
Wire
ZInvert
Col
Spe
Mir
R 0.800
G 0.800
B 0.800
RGB
HSV
DYN
A 1.000
Shaders
Mirror Transp
Lambert
Ref 0.800
Halo
Traceable
Shadbuf
CookTorr
Spec 0.500
Hard:50
Shadow
TraShado
Bias
Radio
Translucency 0.00
Amb 0.500
Emit 0.000
Texture
Map Input
Map To
Col
Nor
Csp
Cmir
Ref
Spec
Amb
Hard
RayMir
Alpha
Emit
Translu
Disp
Stencil
Neg
No RGB
Mix
R 1.000
G 0.000
B 1.000
Col 1.000
Nor 10.58
Var 1.000
Disp 0.200
DVar 1.00
Warp
fac 0.00

2-es módszer (height map festéssel)

(Ezt a modszert inkább height map létrehozására szokás használni, a felületi egyenetlenségek lemodellezésére, de lehet vele normal map-ot is létrehozni)

- **PhotoShop**

Photoshop-ba beépülő normal map és dds export plugin használata.:

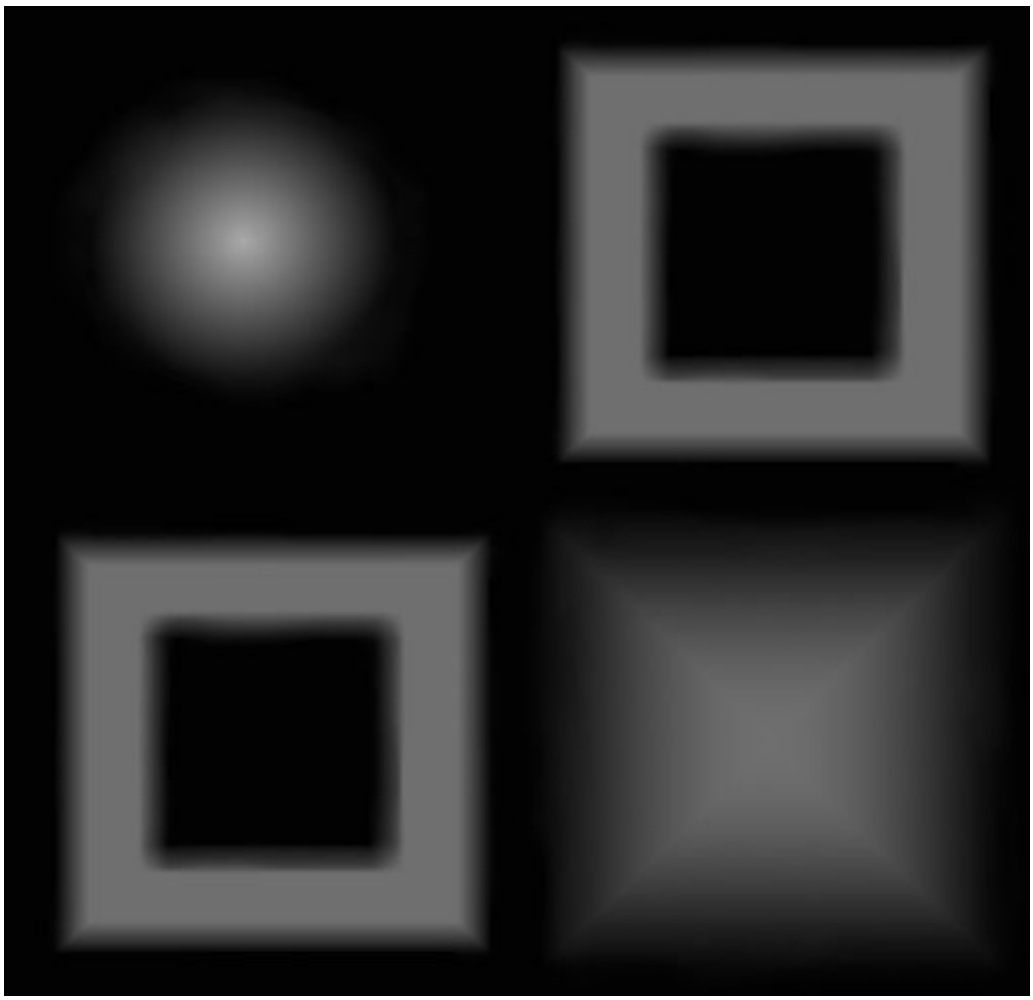
A plugin ingyenes és letölthető a http://developer.nvidia.com/object/photoshop_dds_plugins.html oldalról, mindig a legfrissebb változat.

A plugin tartalmaz egyrészt egy normal map generátor scriptet, ami egy ARGB képből képes egy normal mapot generálni, másrészt pedig tartalmaz egy dds export és import plugint.

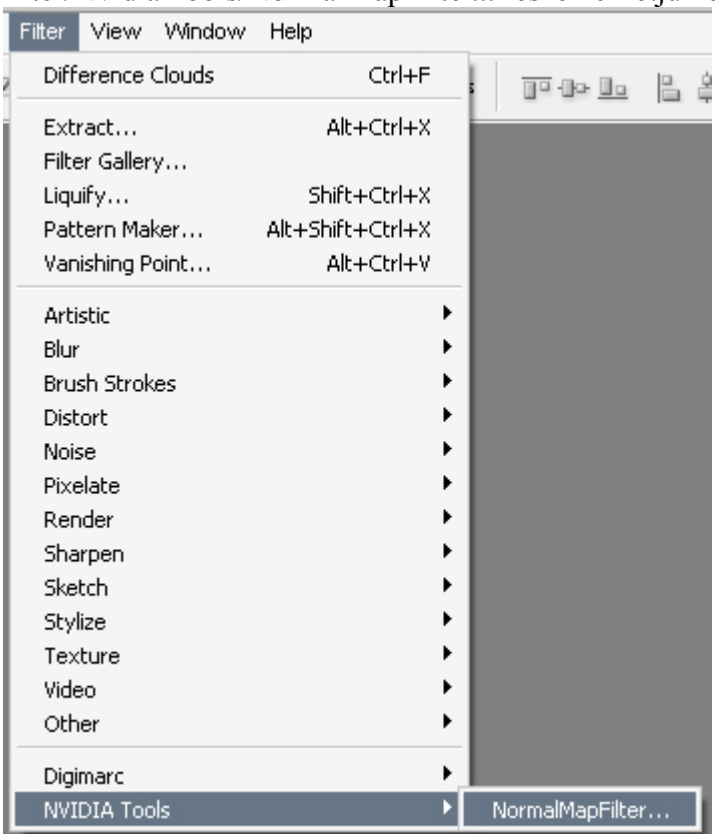
A plugint egyszerűen telepítenünk kell és szükségeltetik hozzá egy minimum 7.0-ás verziójú photoshop. Az exe automatikusan megtalálja a photoshopot és bemásolja magát a megfelelő helyre.

1. Normal map generátor script.

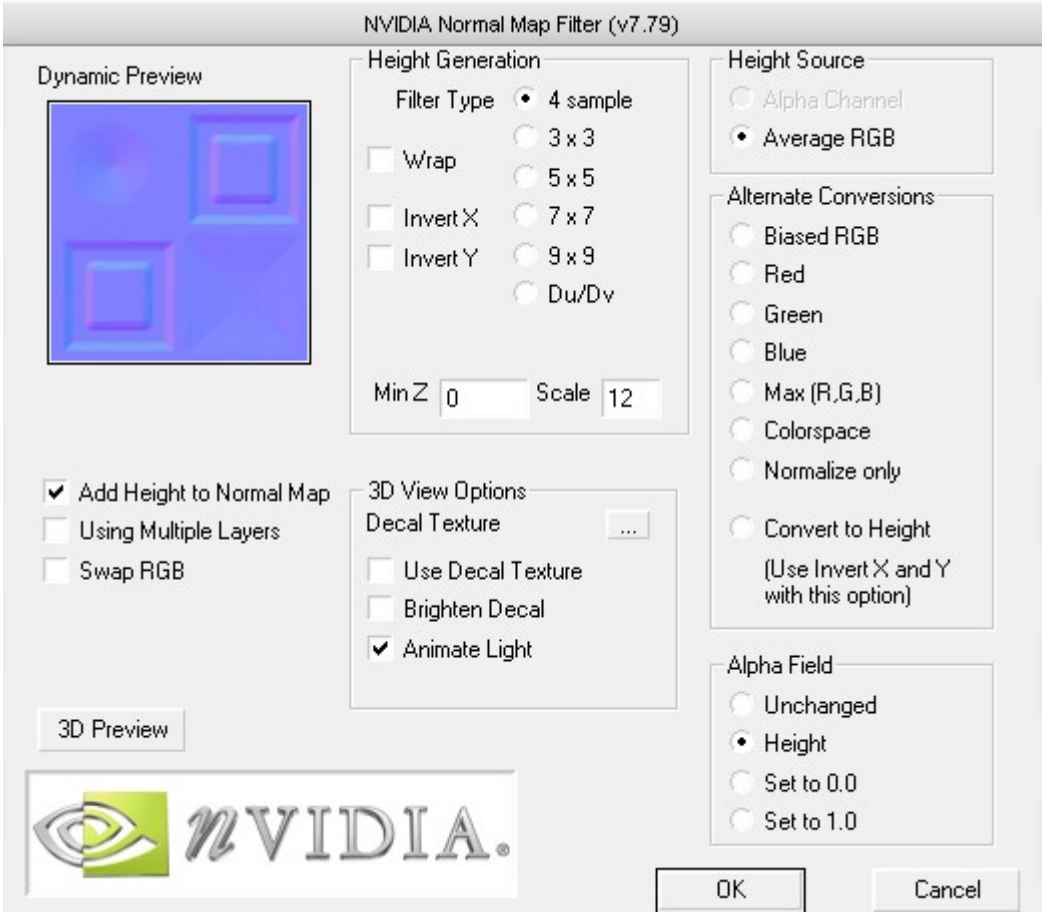
A fekete fehér kép (lásd lenti ábra), alapján lehet egyszerűen kiszámolni a normal mapot. Az egymás körüli pontokból lehet kiszámolni az adott kis rész dőlésszögét, és abból a felületi normálist. A világosabb pixelek jelentik a magasabb részeket, a feketék az alacsonyabb részeket. Alap beállításokkal a fekete jelenti a nulla szintet és a fehér az egyes szinet.



Miután betöltünk egy megfelelő képet (ha nem fekete fehér, az rgb intenzitásából számol, az RGB átlagából), a sciptet a Filter/Nvidia Tools/NormalMapFilter... részen érhetjük el.



Erre kattintva azonnal elérhetjük a konverziós ablakot. Itt az opciók egyszerűen és áttekinthetően jelenítődnek meg.



A height sourcénál beállíthatjuk, miből számoljon a program.
Alpha Channel: Az alfa csatornából (ami jelen példában nincs)
Avarage RGB: Az RGB színcsatorna átlaga (a legtöbb esetben ez a célravezető)
Biased RGB: Az átlagos RGB értéket kivonja a generáláshoz hasnzált RGB értékből, minden egyes texelhez.
Red: A vörös színcsatornából.
Green: A zöld színcsatornából.
Blue: A kék színcsatornából.
Max (R,G,B): Az vörös, zöld és kék színcsatorna közül a legnagyobb.
Colorspace: Magasság = 1.0 – [(1.0-vörös)*(1.0-zöld)*(1.0-kék)] (itt értelemszerűen a csatornák értékei egy 0 és 1 között mozgó lebegőpontos számként értelmezendők)
Normalize only: Ez már egy normal map, csak normalizálja az értékeket (hogy az összes vektor 1 hosszúságú legyen)
Convert to Height: Ez a normal mapot egy height map-á konvertálja (magassági térkép).

Alpha Field – Megadja, hogy mit csináljon az alfa csatornával a program
Unchanged : Békén hagyja, nem változtatja meg.
Height : Megegyezik a magasság értékkel.
Set to 0.0 : 0.0 értékre állítja mindenhol.
Set to 1.0 : 1.0 értékre állítja mindenhol.

Height Generation – A magasság generálásnák lehet beállítani extra opciókat.
Filter Type: Megadja, hogy hány pixelből számolja ki az adott helyre tartozó vektor értékeket. A 4 esetén a négy pixelből (jobb-bal, alsó felső), a szorzós értékek esetén, pedig az akkora terület átlagából. Minél nagyobbra vesszük ezt az értéket annél pontosabb lesz a generált normal map, és annál simább. A nagyterületű filter a nagy felbontású mapok esetében érdemes, míg a kicsi (4-es,

vagy 3x3) a gyorsan változó felületek esetében.

Du/Dv – EMBM mapok generálásához használható.

Invert X : A normal vektoroknál megfordítja az x értéket.
Invert Y : A normal vektoroknál megfordítja az y értéket.

MinZ : A felfelé iránynak ad egy minimális értéket.
Scale : A magasságérték szorzója, mennyivel változik meg. Alapesetben nagyon „sima” normal mapot kapunk, ezt érdemes növelni, hogy a 3D-s megjelenítés esetén látványos legyen.

Swap RGB: Az Rgb értékeket megfordítja a normal map értékeinél (érdekes eredményeket produkál;).

A többi opció a 3D-s valósidejű előnézetre vonatkozik, amit sajnos nem tudtam rendesen tesztelni, mert valószínűleg csak Nvidiás kárnyákon működik rendesen, de amúgysem annyira használható. Érdemes az éppen használt motorban előre megnézni, mert a motorban mindig más, mint ebben a preview nézetben. És egyes motornál pár dologgal rendesen fel is tudják dobni, így sokkalta látványosabb.

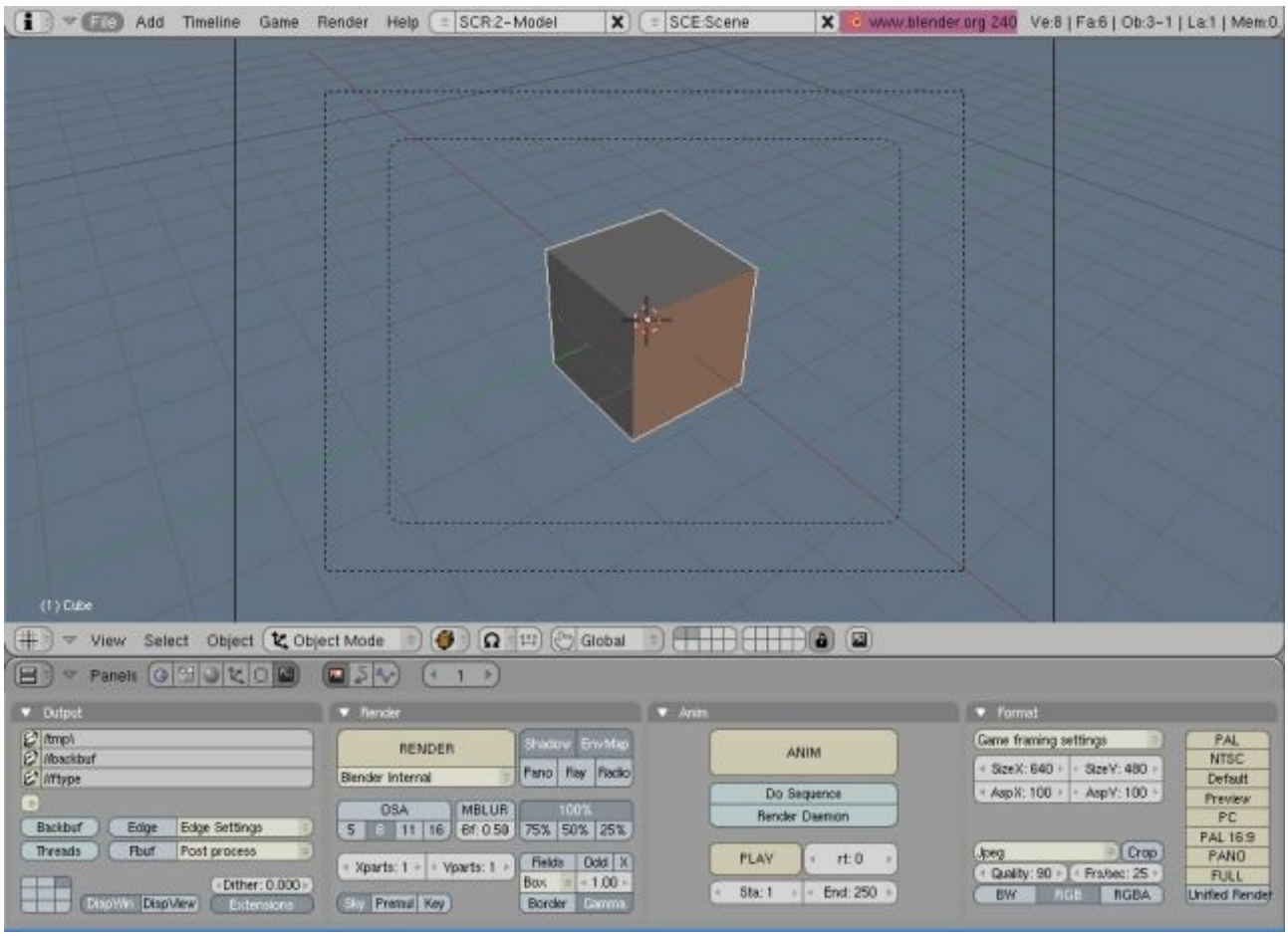
- ## Gimp

Szükségünk lesz a **Gimp NormalMap plugin**-jára amit le kell töltnünk a következő helyről: http://nifelheim.dyndns.org/~cocidius/normalmap/ .

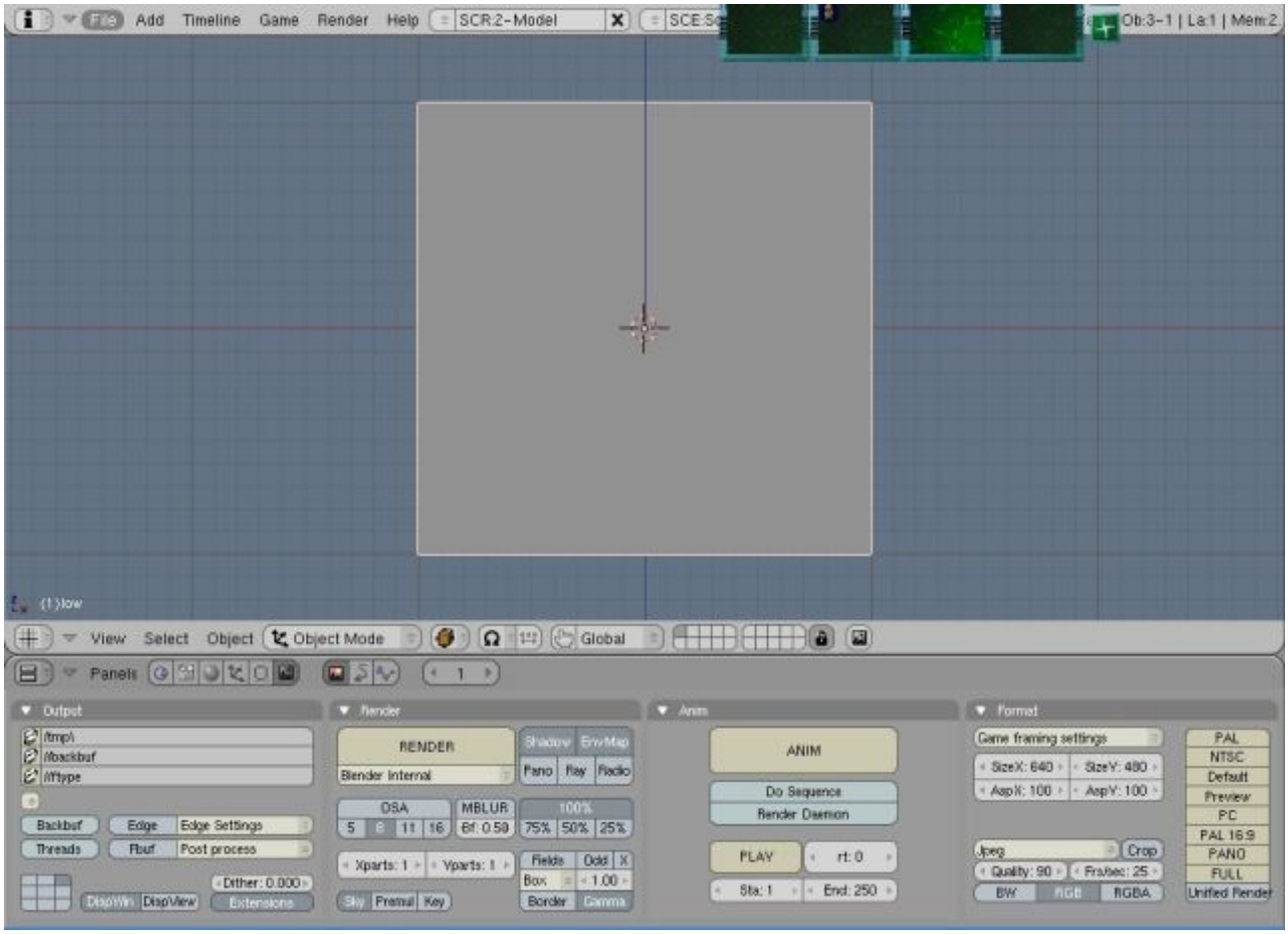
Akkor lássunk is hozzá!

Első lépésben létrehozzuk a modellt a Blenderben.

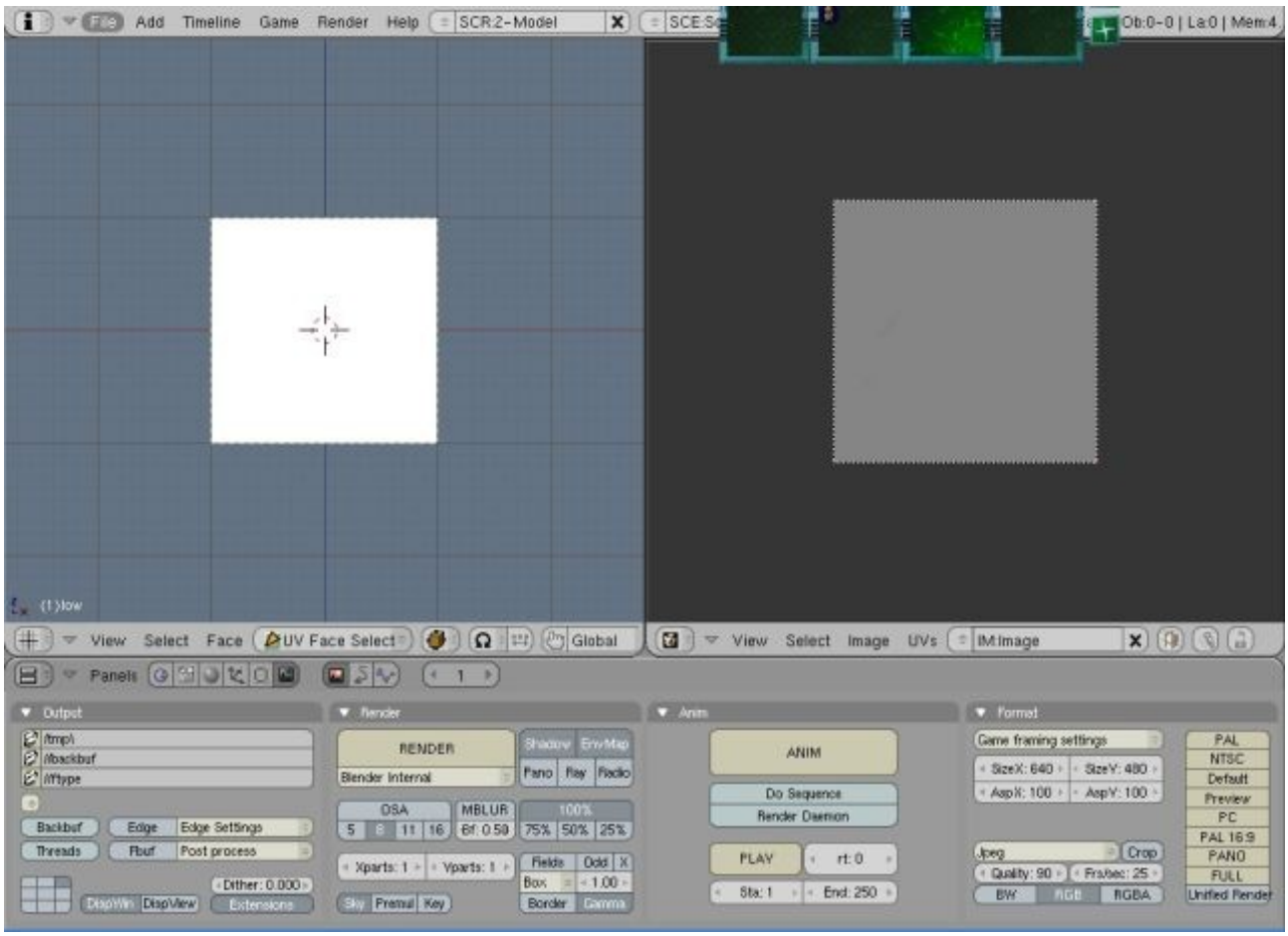
1. Indítsuk a Blendert.



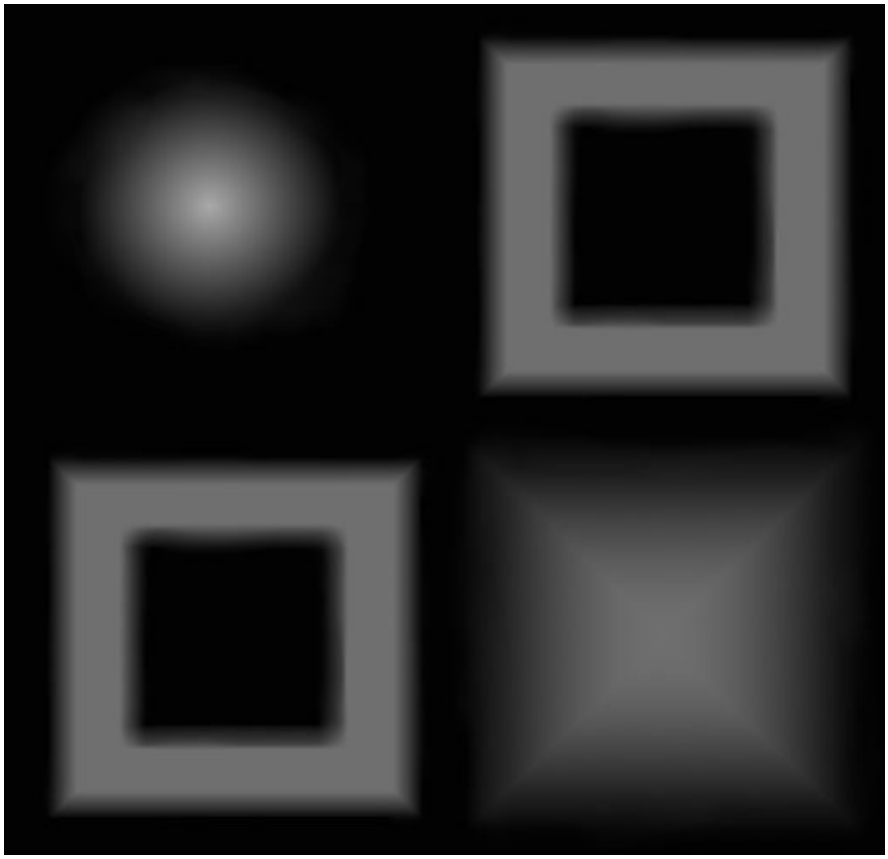
2. Készítsünk egy „Plane”-t.



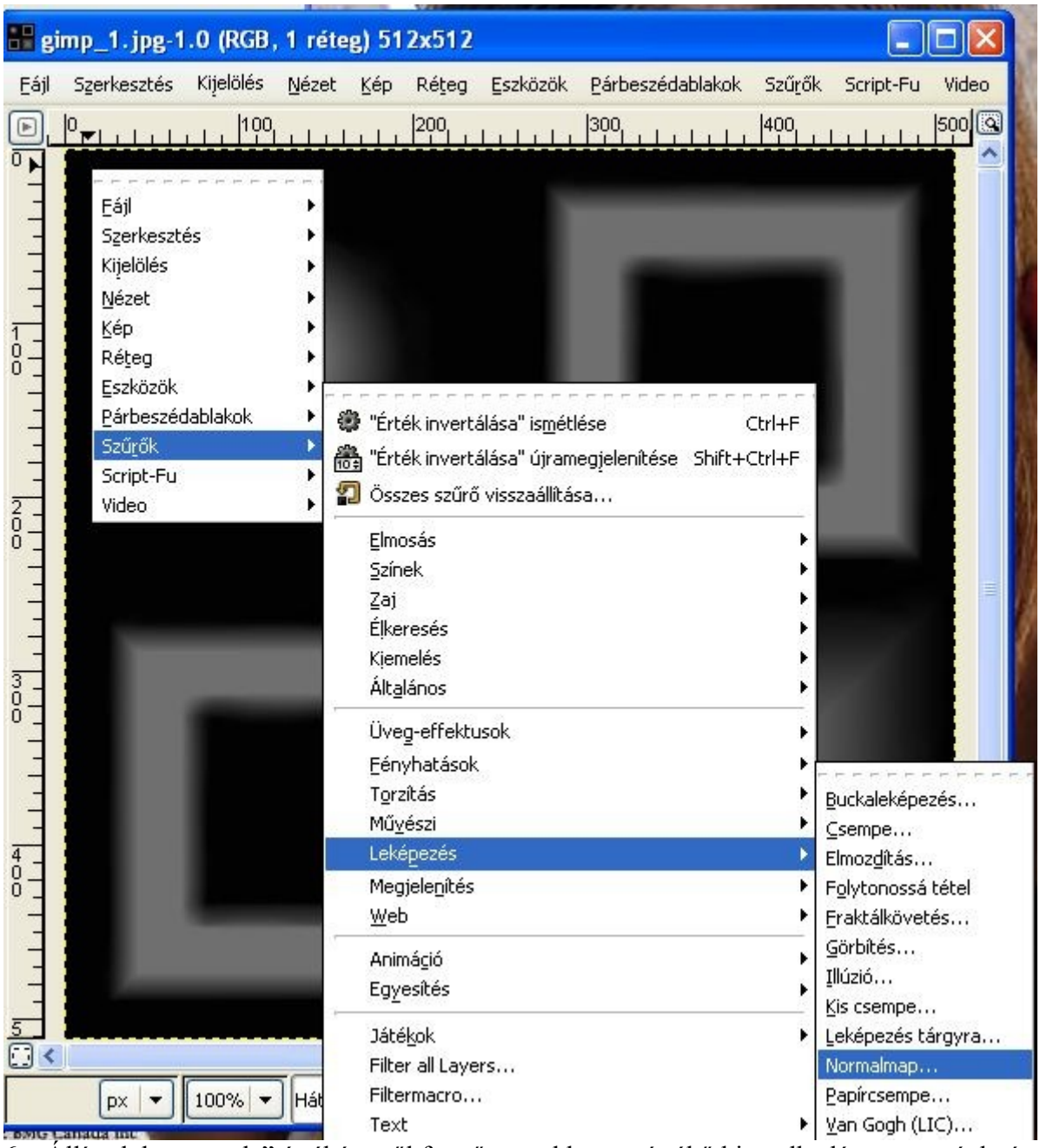
3. Mivel általában bonyolultabb modelleket készítünk, mint egy sima „Plane” és általában UV-mapokat használunk, ezért UV-zzuk le most is, és mentsük is el.



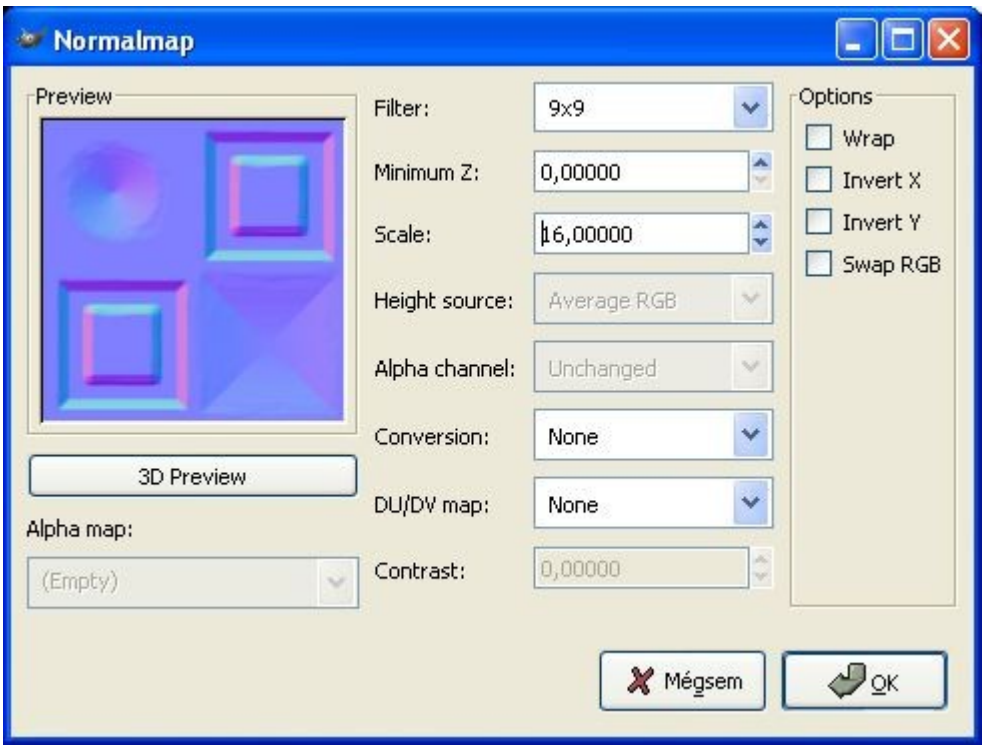
4. Ideje elindítani a Gimp-et. Hozzunk létre egy tetszőleges height mapot, ami abból áll, hogy a világos szín magasabb, a sötét szín alacsonyabb kiemelkedést eredményez.(Próbáljunk meg minél tisztább színátmeneteket létrehozni.)



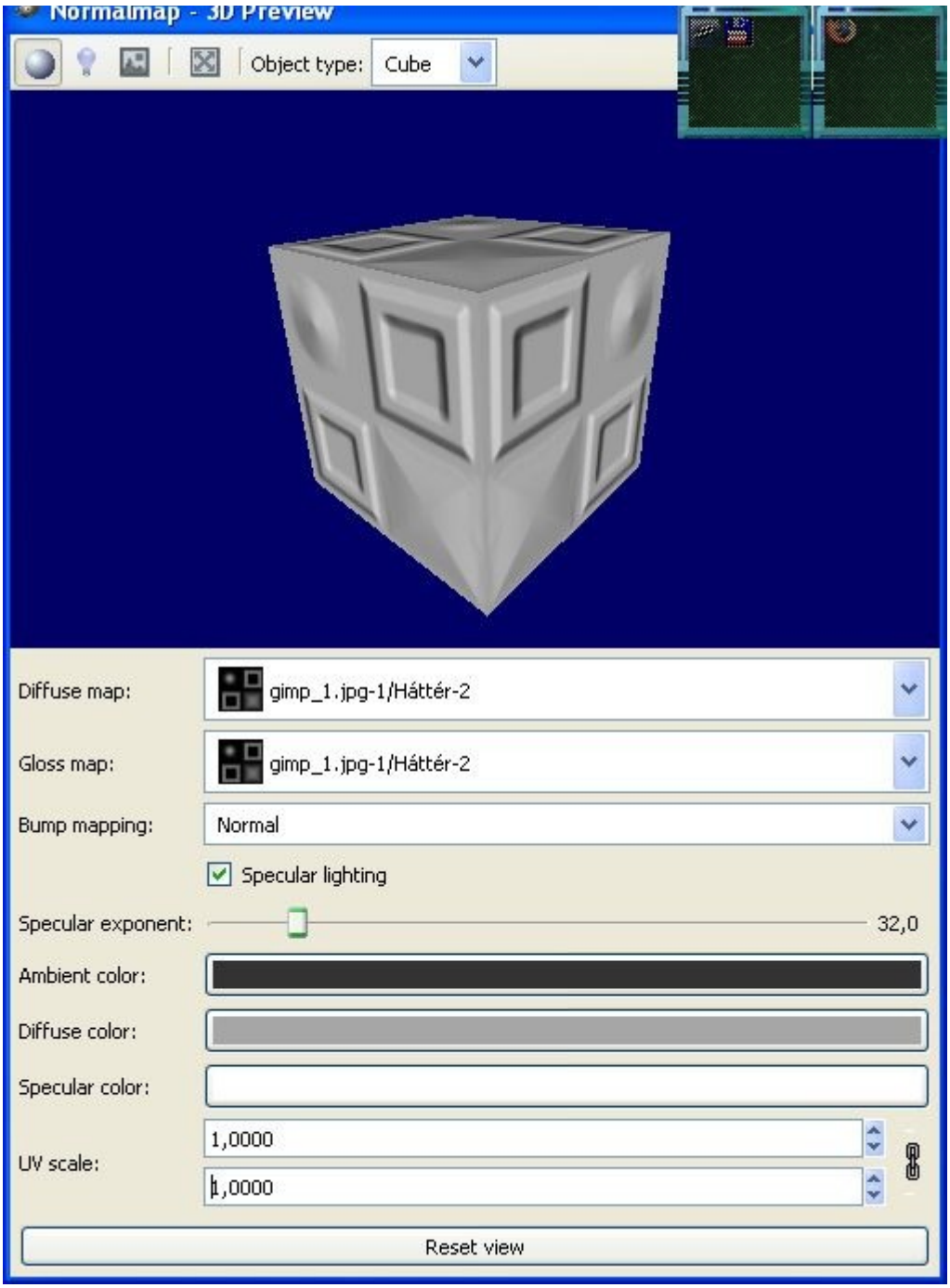
5. A föltelepített NormalMap Plugint a szűrők között találjuk meg.



6. Állítsuk be a „scale” értékét attól függően mekkora mértékű kiemelkedést szeretnénk, és a filtert attól függően mennyire részletes mapot akarunk létrehozni.

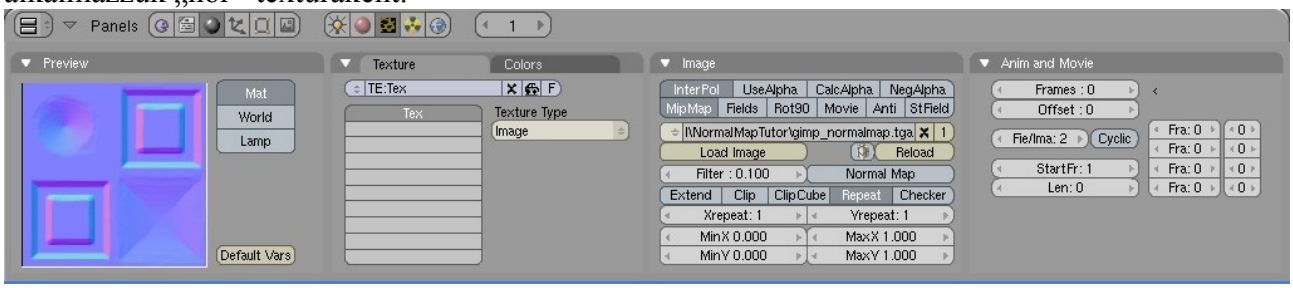


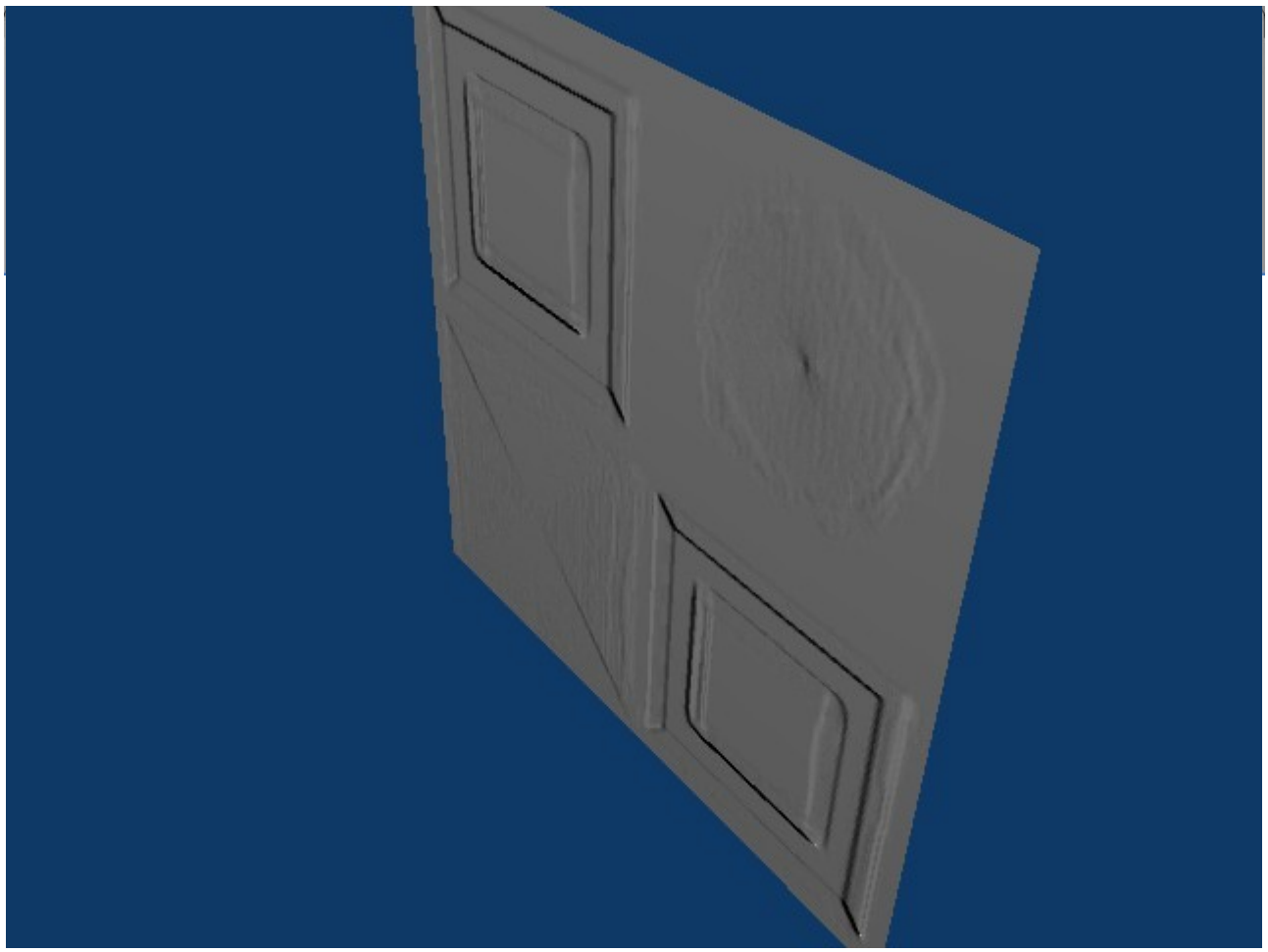
7. A „3D Preview” Gomb-al le is ellenőrizhetjük a normalmap-unkat.



8. Ha meg vagyunk elégedve az eredménnyel, akkor zárjuk be a Preview ablakot, és nyomjuk meg az „OK” -gombot és a normalmap generálása végrehajtódik. A végeredmény kb. a következő lesz:

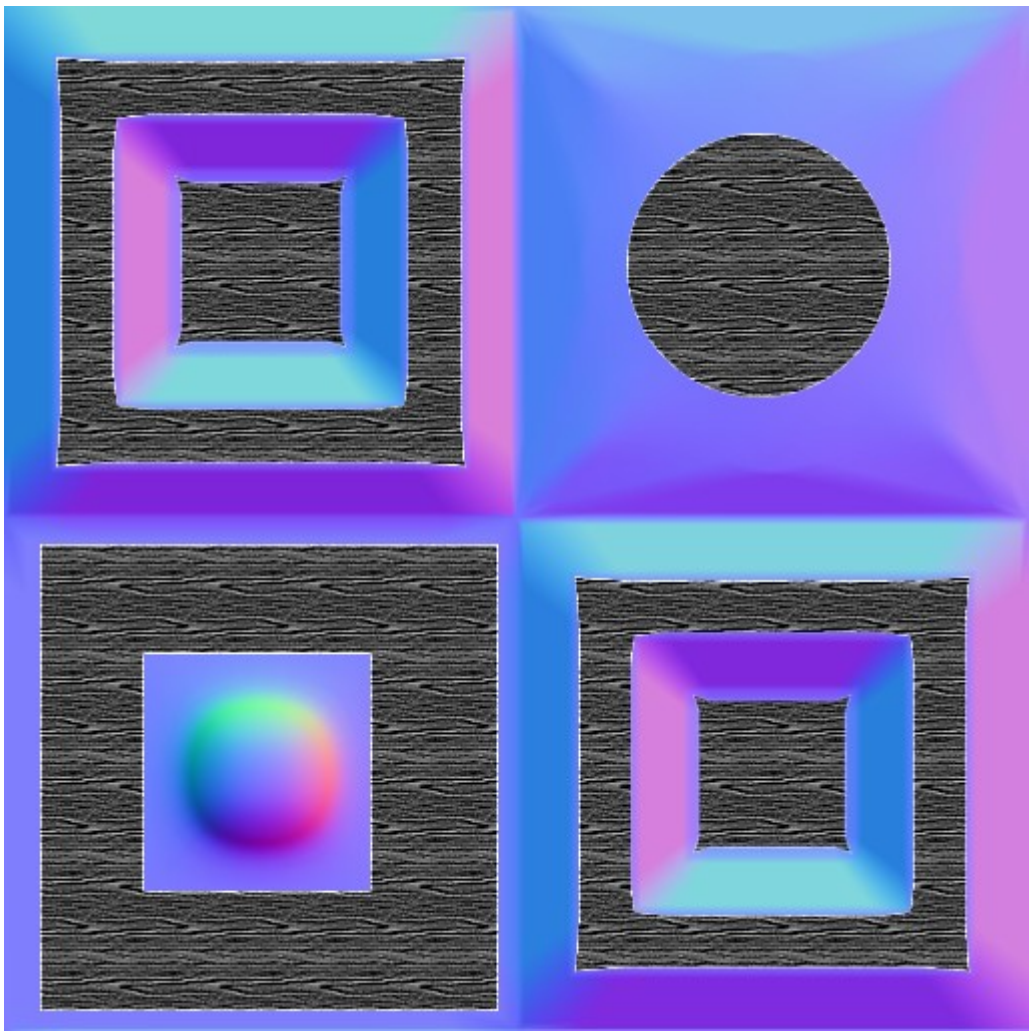
Végezetül ellenőrizhetjük a modellünkön ha betöltjük textúrának az uvzást használva és alkalmazzuk „nor”-textúraként.

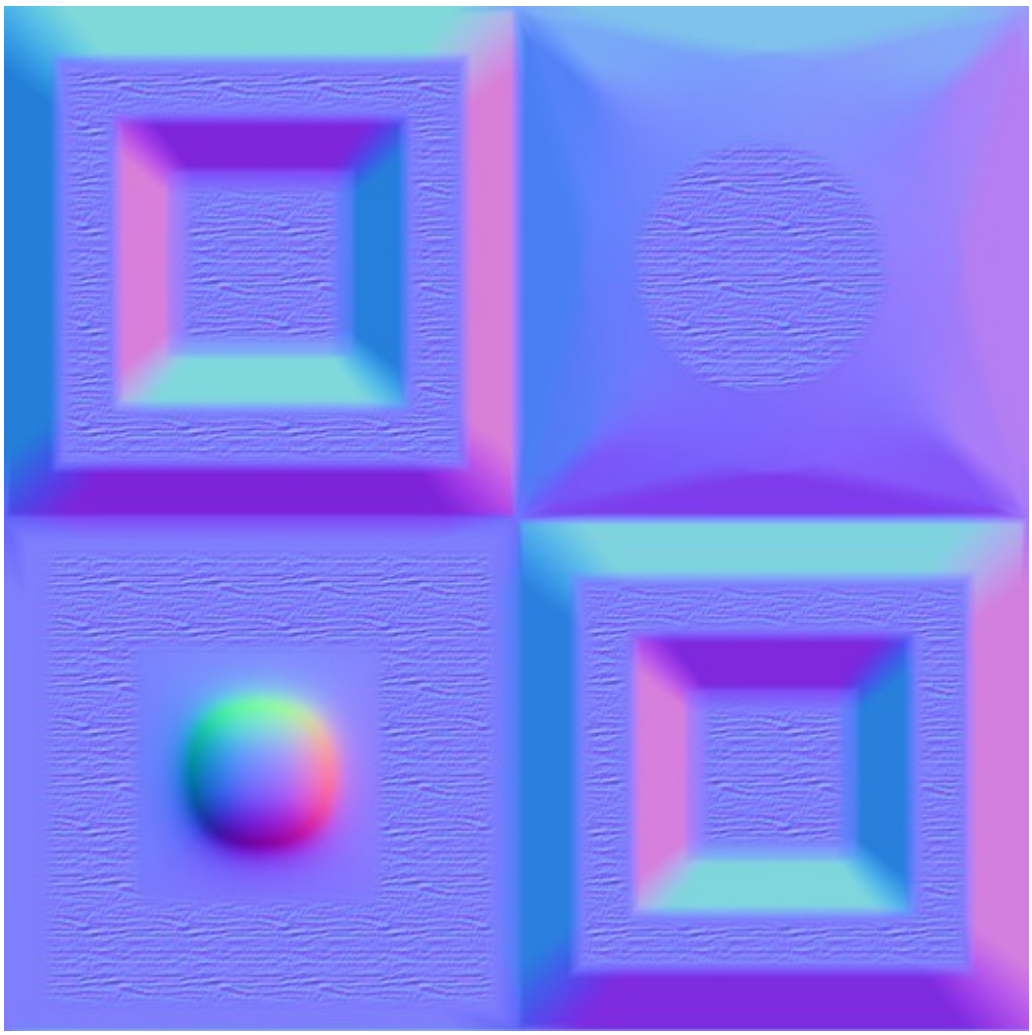


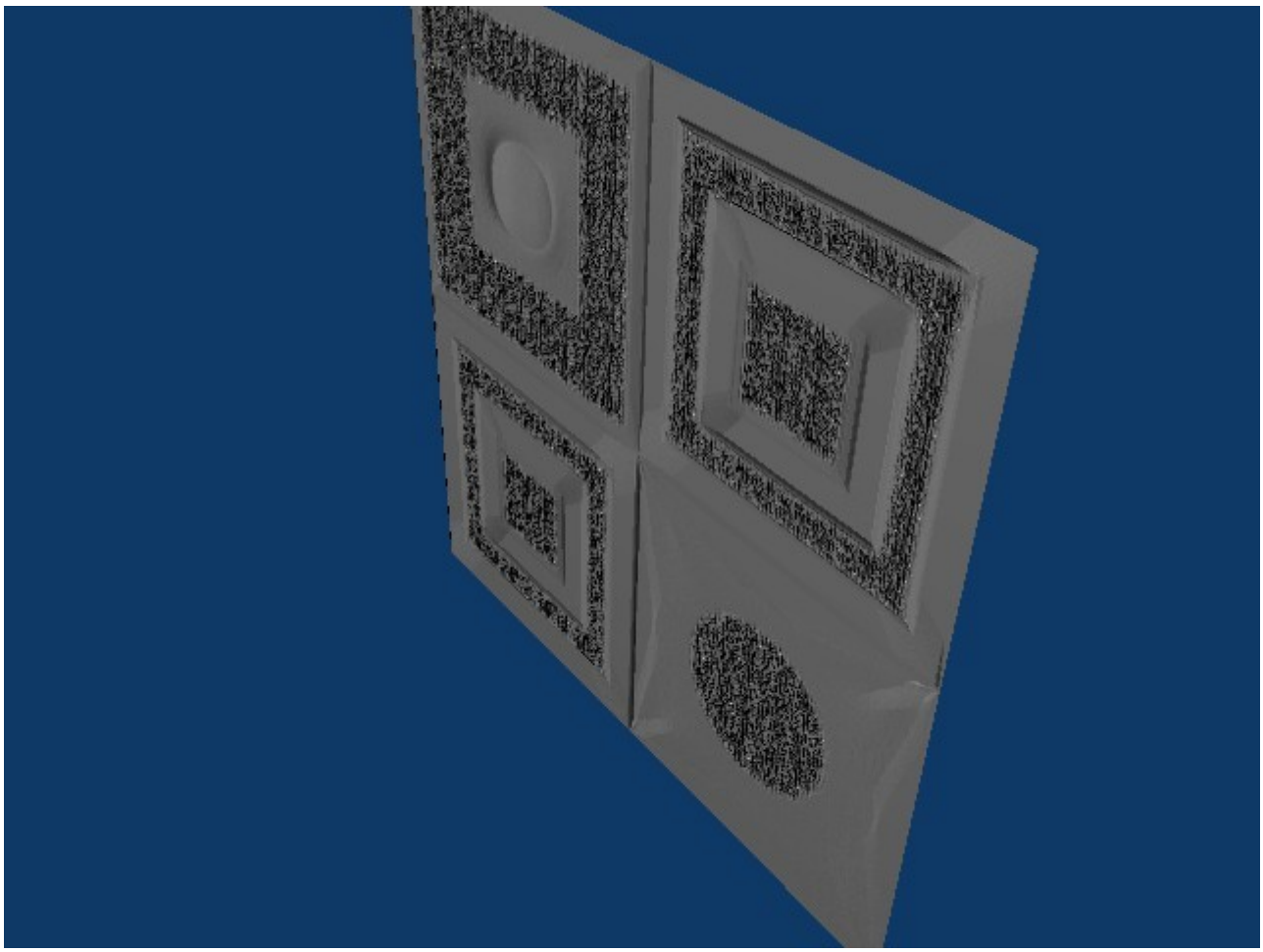


Összegzés

Látható, hogy pontosabb mormalmap készítéshez jobb modszer a modellekből előállított map, mert pontos mapot festeni nem könnyű feledat. De ha igazán részletes normalmap-ot akarunk előállítani, akkor célszerű a két modszert kombinálni.







Jo ezt lehet, hogy erősen eltúloztam, de csak azért, hogy jól látható legyen.