Tipovi shadera u Blender Cycles rendereru

Džebić, Ines

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Preddiplomski studij elektrotehnike

TIPOVI SHADERA U BLENDER CYCLES RENDERERU

Završni rad

Ines Džebić

Osijek, 2017.

SADRŽAJ

1. UVOD

Blender je profesionalni i besplatni 3D računalni grafički softver koji se koristi za izradu 3D modela, animiranih filmova, video igrica, interaktivnih 3D aplikacija te vizualnih efekata. Navedene su glavne primjene, a mogućnosti Blendera su brojne: 3D modeliranje, teksturiranje, simuliranje, oblikovanje, animiranje, renderiranje, video uređivanje. Blender Cycles Engine je objektivno, realno i fizički temeljen program napravljen za animiranje i animacije, što znači da stvara slike na temelju praćenja puta zraka svjetlosti kroz scenu (engl. *Ray tracing)*.

U prvom dijelu ovoga rada bit će opisani osnovni koncepti 3D računalne grafike potrebni za konstrukciju, manipulaciju i prikazivanje 3D objekta na zaslon računala. Također će biti opisani najpoznatiji alati za 3D modeliranje i renderiranje. U nastavku će biti rečeno nešto više o samom programu i njegovim mogućnostima, Cycles rendereru te će biti opisani *shader*-i s primjerima za svaki *shader* pojedinačno. Nakon proučene teorije znanje će biti preneseno na praktični rad pa će tako ovaj završni rad biti potkrijepljen primjerima, tj slikama koje će biti izrađene uz pomoć Blendera koristeći različite *shader*-e, uz detaljne opise izrade tih slika. Na kraju će se nalaziti zaključak u kojem će biti ukratko opisan put stvaranja ovog završnog rada uz sve pozitivne i negativne prepreke te krajnji rezultat rada.

1.1. Zadatak završnog rada

Zadatak ovog završnog rada jeste objasniti *Ray tracing* metodu na osnovu koje radi Cycles Engine, opisati i prikazati koji tipovi *shader*-a postoje u Cycles Engineu i kako se mogu kombinirati kako bi se postigli fotorealistični materijali. Za praktični dio potrebno je izraditi fotorealistične digitalne slike na kojima će se demonstrirati rad opisanih *shader*-a.

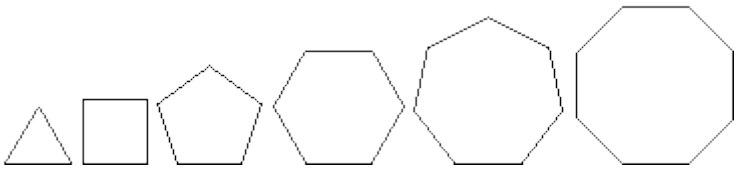
2. OSNOVNI KONCEPTI U 3D GRAFICI

Svi objekti koje vidimo oko sebe imaju tri dimenzije koje nazivamo visinom, širinom i dužinom. Dimenzija je karakteristika prostora koja identificira objekte i njihov položaj u prostoru. Kada te objekte želimo prikazati na zaslonu nekog uređaja primjećujemo da su oni prikazani u dvije dimenzije. Te dvije dimenzije su visina (od donjeg do gornjeg kraja zaslona) te širina (od lijevog do desnog kraja zaslona).

Grafika se definira kao vizualna animacija koja može biti na različitoj podlozi (digitalnoj ili realnoj opipljivoj površini), a cilj joj je informiranje, ilustriranje i zabava [1]. Računalna grafika je grafika stvorena pomoću računala, a 3D grafika je grafika koja koristi trodimenzionalno predočenje podataka u računalu u svrhu izvođenja izračuna i renderiranja 2D slika. Renderiranje je cjelokupni proces izračunavanja svih matematičkih parametara te prikazivanje 3D objekta u 2D okruženju (npr. zaslonu računala). Da bi objekt izgledao realan na zaslonu, potrebno je dodati određene vizualne karakteristike poput sjenčanja, sjena i tekstura, kako bi dobili simuliranu treću dimenziju – dubinu, a 3D model je upravo ta trodimenzionalna simulacija stvarnog objekta.

2.1. 3D model

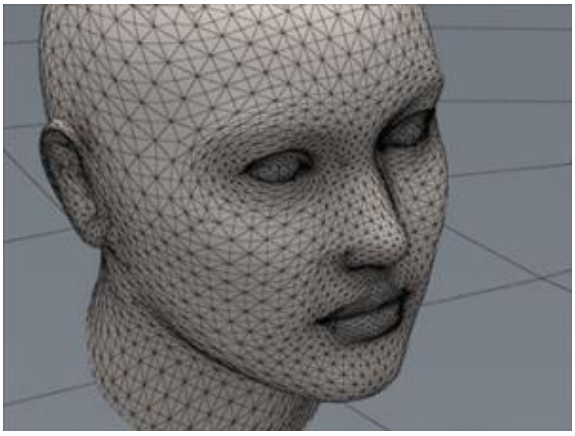
Krajnji cilj 3D modeliranja je 3D model koji je, kako je već opisano u prethodnom poglavlju, trodimenzionalna simulacija stvarnog objekta. Prepoznatljiv je po svojem obliku koji je definiran vrhovima (engl. *vertex*). Vrhovi se povezuju bridovima (engl. *edge*), a tri ili više povezana ruba čine poligon ili mreža (engl. *mesh*).



Sl. 2. 1. Poligoni[5]

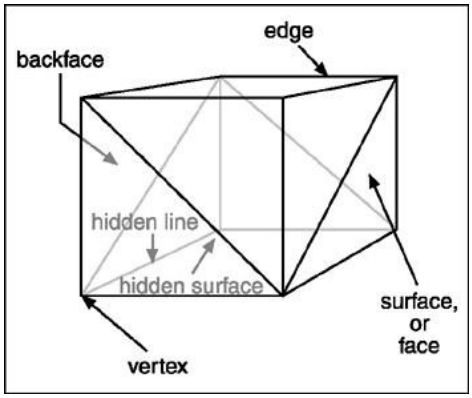
Iz slike 2.1. je jasno vidljivo da je najjednostavniji poligon trokut i upravo on je u najčešćoj upotrebi u 3D grafici gdje su hardveri stvoreni tako da mogu prikazati milijune trokuta u samo jednoj sekundi. Svaki od poligona koji nije trokut je rastavljiv na dva ili više trokuta. To uvelike doprinosi 3D modeliranju jer svaki složeniji 3D model se može rastaviti na veliki broj trokuta. Taj skup trokuta koji čine 3D model se također naziva mreža.

Mreža je zapravo skup svih vrhova i bridova modela, a oni čine trokute (najčešće su to trokuti, a mogu biti i ostali poligoni). Razlika između mreže i modela je to što je mreža dio modela i definira vrhove modela, a model može imati i druge elemente kao što su materijal, tekstura i animacija. Model može biti sastavljen od jedne ili više mreža. Mreža je zapravo geometrija koja se renderira.



Sl. 2. 2. Mreža 3D modela [6]

Dijelovi 3D modela su površina ili ravnina (engl. *surface*) i lica te površine ili ravnine (engl. *faces*). Lice je poligonalna površina. Ivice površine i lica se nazivaju rubovi (engl. *edges*). Modeli mogu imati površinu koja nam je nevidljiva zato što se može vidjeti samo s jedne strane. Tu "nevidljivu" stranu nazivamo skrivena površina (engl. *hidden surface*). Rubovi takvih površina se nazivaju skrivene linije (engl. *hidden lines*). Lice koje se vidi s obje strane se naziva obostrano lice (engl. *double-sided face*). Većina modela ima lica i na drugoj strani modela, a takva lica se nazivaju stražnja lica (engl. *backface*).



Sl. 2. 3. Dijelovi 3D modela [1]

Prema [3], 3D model možemo prikazati u sljedeća tri oblika:

[1] Žičani (engl. *Wireframe*)

[2] Plošni (engl. *Surface*)

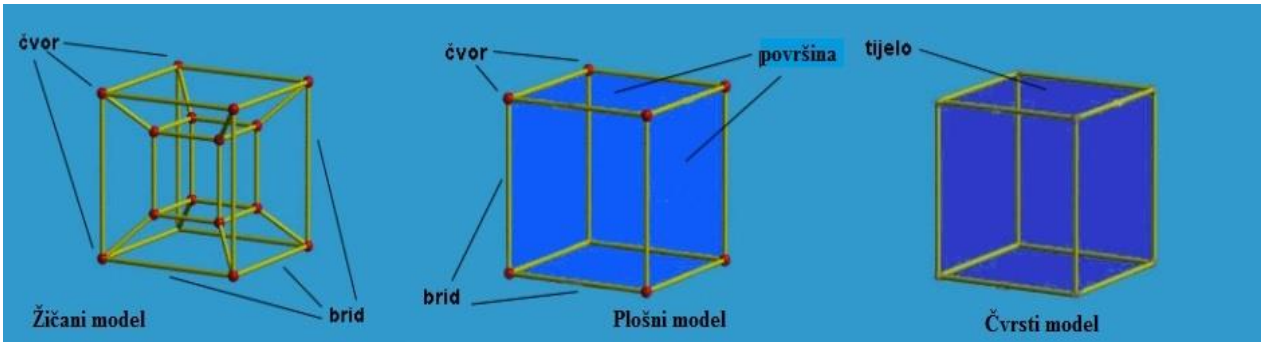
[3] Čvrsti (engl. *Solid*)

Ova podjela je zasnovana na temelju tehnika kreiranja i obrađivanja modela.

Žičani model je najjednostavniji trodimenzionalni model. Ograničen je uglavnom na bridove objekta, između bridova ne postoji veza, a nisu definirani ni odnosi površina. Geometrijski elementi koji čine žičani model su čvorovi i bridovi. Kod žičanog modela postoji problem dvosmislenosti jer često više različitih objekata mogu imati iste bridove i elemente, pa nije jasno definirano o kojem se točno objektu radi.

Plošni model je trodimenzionalni model koji za osnovnu informaciju o objektu koristi plohe. Plohe nije jednostavno opisati u analitičkom smislu, a i njihov vektorski prikaz nije moguć. Za matematički izraz plohe koriste se metode aproksimacije gdje svaka pojedina ploha predstavlja parametarsku funkciju zadanih točaka. Problem kod ovog modela je prikazivanje presjeka ili slično. Tada ovi modeli izgledaju „prazno“ kao da nemaju volumen.

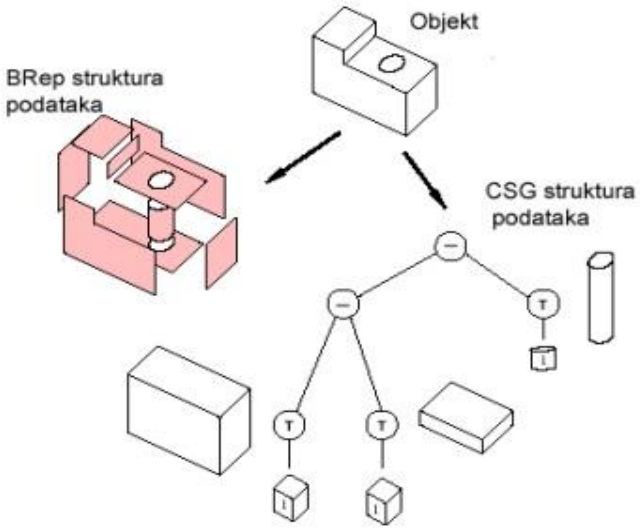
Tada koristimo čvrste modele koji posjeduju volumen 3D fizičkog objekta iz stvarno svijeta. Čvrsti modeli su pravi, čvrsti prikaz stvarnog objekta i kao takvi su potpuni i nedvosmisleni.



Sl. 2. 4. Žičani, plošni i čvrsti 3D model [7]

Koriste se dvije vrste čvrstih modela, a to su *boundary representation* model (BRep) i *constructive solid geometry* model (CSG). Kod BRep modela je jasno određena hijerarhija u strukturi podataka jer je objekt definiran osnovnim elementima: plohama, plohe bridovima, bridovi čvorovima, a čvorovi koordinatama. BRep je organiziran u topološkom smislu (povezanost pojedinih geometrijskih dijelova) i u geometrijskom smislu (oblik objekta, veličina i položaj). Kod CSG metode se koriste osnovni postupci konstrukcije čvrstih modela:

parametarski, osnovni 3D geometrijski oblici automatski generirani u 3D programima. To su npr. kugla, kvadar, piramida, cilindar itd.. Cilj je stvoriti kompleksne objekte koji će zapravo biti sastavljeni od većeg broja jednostavnih objekata, odnosno kao skup dodanog, oduzetog ili međusobno presječenog većeg broja jednostavnih objekata. S obzirom da svaka nosi i dobre i loše strane, najproduktivnije je koristiti obje metode u isto vrijeme kako bi se stvorio što realniji 3D model.



Sl. 2. 5. BPrep i CSG struktura istog objekta [7]

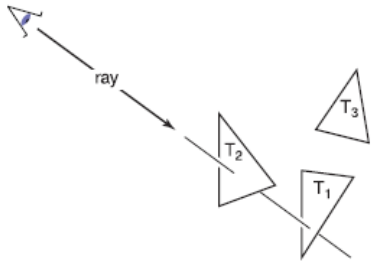
2.2. Metode fotorealističnog renderiranja

Jedan od glavnih zadataka računalne grafike je renderiranje 3D objekata. Renderirati znači pronaći scenu ili model koji se rastavlja na niz geometrijskih oblika u 3D prostoru i to pretvoriti u 2D sliku koja prikazuje odabranu scenu ili model s točno jednog gledišta. Renderiranje je proces u kojem objekt sa svim svojim dijelovima pretvaramo u niz piksela. Piksel je najmanji dio slike, osnovna jedinica od koje je slika sastavljena. Ono što je atom za molekulu, to je piksel za sliku.

Fotorealistično renderiranje je stvaranje računalno generirane slike koja izgleda kao fotografija. Cilj je stvoriti, kako i samo ime kaže, prizor ili objekt koji će dostojno i uvjerljivo izgledati kao da je fotografiran ili realan, a ne stvoren računalnim programom. Postići takav efekt znači točno simulirati sva svojstva uključujući osobine, materijal i geometriju prizora ili objekta, osvjetljenje i efekte kamere/fotoaparata.

Renderiranje se temelji na R*ay tracing*-u i *Ray casting*-u.

Kako je opisano u [2], *Ray tracing* (engl. *ray* – zraka, engl. *trace* – trag) je algoritam praćenja zrake za pravljenje renderiranih trodimenzionalnih scena. Računa jedan po jedan piksel i za svaki je piksel osnovni zadatak pronaći objekt koji vidljiv iz pozicije tog piksela na slici. Svaki piksel je okrenut u drugom smjer i svaki objekt koji piksel vidi mora presjeći takozvanu *viewing ray*, zraku koja se širi iz smjera gledišta prema smjeru u kojem piksel „gleda“, koja će u daljnjem tekstu biti nazvana „zrakom pogleda“. Željeni objekt je onaj koji presijeca zraku pogleda najbliže kameri jer blokira pogled na druge objekte iza njega. Kada je objekt pronađen, kalkulacija za određivanje boje piksela koristi točku presijecanja, normalu površine i neke druge informacije koje ovise o vrsti renderiranja. Za primjer je pokazana zraka koja presijeca dva trokuta no samo prvi trokut T_2 je pogođen zrakom i osjenčan (sl. 2.6.).



Sl. 2.6. Prikaz zrake pogleda [2]

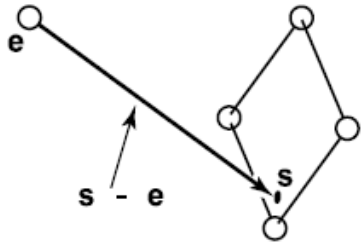
Osnovni *ray tracer* sadrži 3 dijela:

1. Stvaranje zrake (engl. *Ray generation*) – bazirajući se na geometriji kamere kalkulira izvor i smjer zrake pogleda svakog piksela
2. Presijecanje zraka (engl. *Ray intersection*) – pronalazi najbliži objekt koji presijeca zraku pogleda
3. Sjenčanje (engl. *Shading*) – bazirajući se na rezultatu presijecanja zraka kalkulira boju piksela

2.2.1. Stvaranje zrake

Osnovni alati korišteni za stvaranje zrake su točka promatranja i slika ravnine. Zraku možemo predstaviti točkom ishodišta *e* i smjerom širenja *(s - e):*

$$p(t) = e + t(s - e) \tag{2-1}$$



Sl. 2.7. Zraka e koja ide od oka to točke s na površini [2]

Od točke *e* preko vektora *(s - e)* se prijeđe frakcijska udaljenost *t* za pronalazak točke *p*.

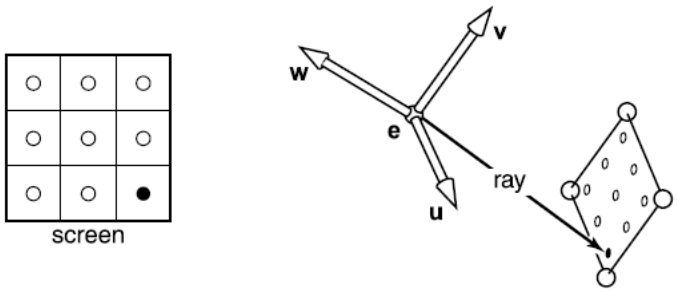
$$p(0) = e \tag{2-2}$$

$$p(1) = s \tag{2-3}$$

Za $0 < t_1 < t_2$, $p(t_1)$ je bliže oku nego $p(t_2)$.

Za $t < 1$, $p(t)$ je „iza“ oka.

Za proračun zrake pogleda nam je potreban *e* (koji je zadan) i *s*. Pronaći varijablu *s* se možda čini teško, no ukoliko se problemu priđe u ispravnom koordinatnom sustavu, rješenje je jednostavno.



Sl. 2.8. Preslikavanje varijabli iz trodimenzionalnog u dvodimenziolani prostor[2]

Metoda stvaranja zraka započinje od ortonormalnog koordinatnog okvira koji je poznat pod nazivom okvir kamere (engl. *camera frame*). On je označen sa *e* (*eye point*), i označava točku oka ili točku stajališta. Tri vektora baze su *u, v* i *w;* i oni su postavljeni tako da u odnosu na pogled kamere *u* ima pravac u desno, *v* je usmjeren ka gore, a *w* je usmjeren prema natrag, tako da $\{u, v, w\}$ formiraju desni koordinatni sustav. Najčešći način konstrukcije okvira kamere jeste iz točke stajališta (engl. *viewpoint*) koja postaje *e*, smjera pogleda (engl. *view direction*) koji postaje *w*, i vektora prema gore (engl. *up vector*) koji se koristi za konstrukciju baze koja ima u svojoj ravnini vektore *v* i *w*. Baza je definirana smjerom pogleda i smjerom vektora ka gore (sl. 2.8.).

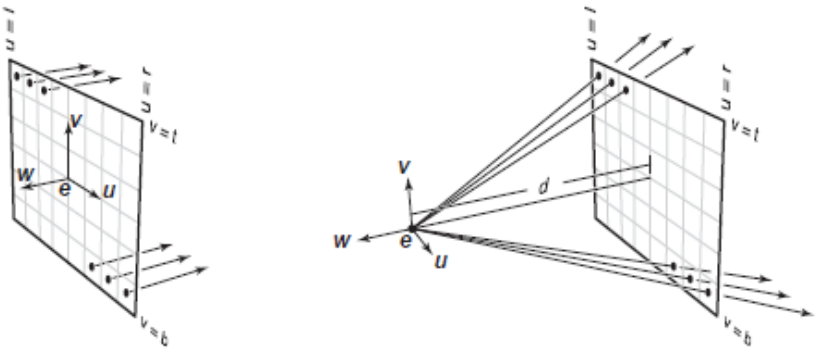
Za ortografski pogled, sve zrake će imati smjer vektora *w*. Zraka pogleda počinje od ravnine koja je definirana sa *e* i vektorima *u* i *v*. Jedina informacija koja preostaje za saznati jeste gdje na ravnini se slika treba nalaziti. Dimenzija slike se definira sa četiri broja za četiri stranice slike i to na sljedeći način: *l* i *r* za lijevi i desni rub slike (engl. *left* - lijevo, engl. *right* – desno) mjereno od *e* u smjeru *u*; *b* i *t* za donji i gornji rub slike (engl. *bottom* – dno, engl. *top* – vrh) mjereno od *e* u smjeru *v*. Obično je $l < 0 < r$ i $b < 0 < t$.

Kao što je već poznato, svaka slika se sastoji od velikog broja piksela. Da bi se slika sa $n_x \times n_y$ piksela uklopila u pravokutnik veličine $(r-l) \times (t-b)$, pikseli su razmaknu za udaljenost $(r-l)/n_x$ horizontalno i za $(t-b)/n_y$ vertikalno, sa prostorom od pola piksela oko ruba kako bi se mreža piksela uklopila sa pravokutnikom slike. To znači da piksel na poziciji (*i, j)* u raster slici (slika koja se sastoji od bezbroj malih pravokutnika – piksela) ima poziciju

$$u = l + (r-l)(i+0.5)/n_x, \tag{2-4}$$

$$v = b + (t-b)(j+0.5)/n_y, \tag{2-5}$$

gdje su (*u, v*) koordinate pozicije piksela na ravnini slike, izmjerene s obzirom na podrijetlo *e* i bazu $\{u, v\}$.



Sl. 2. 9. Primjer generiranja zrake uz korištenje okvira kamera (lijevo za ortografski pogled, desno za perspektivni prikaz) [2]

U ortografskom pogledu se koristi položaj ravnine slike piksela kao početna točka zrake, gdje se već unaprijed zna da je smjer zrake isto što i smjer pogleda. U perspektivnom pogledu sve zrake imaju isto podrijetlo (u točki stajališta), a ono što je različito za svaki piksel jeste smjer. Ravnina slike više nije na mjestu *e*, nego se definira kao neka udaljenost *d* ispred *e*. Ova udaljenost se naziva udaljenost slike ravnine ali češće se naziva žarišnom duljinom (engl. *focal lenght*), jer biranje udaljenosti *d* funkcionira na isti način kao biranje žarišne udaljenosti kod prave kamere. Smjer svake zrake u perspektivnom pogledu je definiran preko točke stajališta i pozicije piksela na ravnini slike.

2.2.2. Presijecanje zraka

Sada kada je prvi korak napravljen, odnosno zraka $e + td$ generirana, potrebno je pronaći prvi presjek ili sjecište sa bilo kojim objektom kada je $t > 0$. U praksi se pokazalo da je od velike koristi riješiti općenitiji slučaj gdje se traži prvo sjecište zrake i površine koja se pojavljuje u trenutku *t* u intervalu $[t_0, t_1]$. Osnovni slučaj presijecanja zrake je slučaj gdje je $t_0 = 0$ i $t_1 = +\infty$.

Za presjek zrake i sfere prema [2] postoji sljedeće rješenje:
Zadana je zraka $p(t) = e + td$ i implicitna površina $f(p) = 0$. Točke presjeka se pojavljuju kada točke na zraci zadovolje implicitnu jednadžbu tako da su vrijednosti *t* koje tražimo one koje rješavaju jednadžbu

$$f\big(p(t)\big) = 0 \tag{2-6}$$

ili

$$f(e + td) = 0. \tag{2-7}$$

Sfera sa središtem $c = (x_c, y_c, z_c)$ i promjerom *R* se može prikazati preko implicitne jednadžbe

$$(x - x_c)^2 + (y - y_c)^2 + (z - z_c)^2 - R^2 = 0 \tag{2-8}$$

ili u vektorskom obliku:

$$(p - c) \cdot (p - c) - R^2 = 0 \tag{2-9}$$

Svaka točka p koja zadovoljava jednadžbu (2-9) se nalazi na sferi. Ukoliko se uključe točke na zraci $p(t) = e + td$ u ovu jednadžbu, dobije se jednadžba sa *t* :

$$(e + td - c) \cdot (e + td - c) - R^2 = 0. \tag{2-10}$$

Preuređivanjem se dobije:

$$(d \cdot d)t^2 + 2d \cdot (e - c)t + (e - c) \cdot (e - c) - R^2 = 0. \tag{2-11}$$

U jednadžbi (2-11) sve je poznato osim parametra *t*, pa je tako ovo klasična kvadratna jednadžba koja ima oblik:

$$At^2 + Bt + C = 0. \tag{2-12}$$

Rješavanjem kvadratne jednadžbe, točnije diskriminante $B^2 - 4AC$ koja se nalazi pod korijenom, dobije se broj realnih rješenja. Ukoliko je diskriminanta negativna, rezultat korjenovanja će biti imaginarni broj i to znači da se zraka i sfera ne presijecaju. Ukoliko je diskriminanta pozitivna, postoje dva rješenja gdje jedno rješenje predstavlja točku gdje zraka ulazi u sferu, a drugo rješenje predstavlja točku izlaza zrake iz sfere. Ukoliko je pak rješenje

diskriminante nula, zraka samo dodiruje sferu u točno jednoj točki koja je zadana rješenjem kvadratne jednadžbe.

Za presjek zrake i trokuta, prema [2], rješenje je sljedeće:

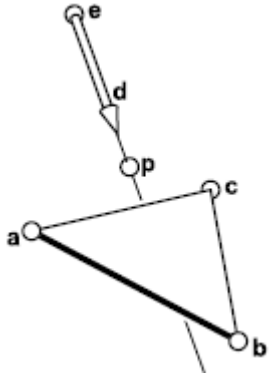
Za presijecanje zrake sa parametričnom površinom, postavi se sustav jednadžbi gdje sve Kartezijeve koordinate odgovaraju:

$$\begin{cases} x_e + tx_d = f(u,v) \\ y_e + ty_d = g(u,v) \\ z_e + tz_d = h(u,v) \end{cases} \quad \text{ili} \quad e + td = f(u,v). \tag{2-13}$$

S obzirom da su ovo tri jednadžbe sa tri nepoznanice (*t, u,* i *v*), rješenje se dobije numeričkim putem. Ako su stranice trokuta zadane sa *a, b,* i *c*, tada se presijecanje događa kada je

$$e + td = a + \beta(b - a) + \gamma(c - a)\ . \tag{2-14}$$

Sjecište *p* će biti u $e + td$ kako je prikazano na slici 2.10..



Sl. 2.10. Prikaz zrake u točki p pada na ravninu u kojoj se nalazi trokut [2]

Sjecište će biti unutar trokuta ako i samo ako je $\beta > 0,\ \gamma > 0\ i\ \beta + \gamma < 1$. U suprotnome, zraka nije pogodila trokut. Ukoliko ne postoje rješenja, ili trokut nije pravilnog oblika ili je zraka paralelna sa ravninom u kojoj se nalazi trokut. Za dobivanje rješenja jednadžbe (2-14) ona se proširuje iz vektorskog oblika u sljedeći oblik tri jednadžbe za tri koordinate:

$$x_e + tx_d = x_a + \beta(x_b - x_a) + \gamma(x_c - x_a), \tag{2-15}$$

$$y_e + ty_d = y_a + \beta(y_b - y_a) + \gamma(y_c - y_a), \tag{2-16}$$

$$z_e + tz_d = z_a + \beta(z_b - z_a) + \gamma(z_c - z_a). \tag{2-17}$$

Također se može zapisati kao standardni linearni sustav:

$$\begin{bmatrix} x_a - x_b & x_a - x_c & x_d \\ y_a - y_b & y_a - y_c & y_d \\ z_a - z_b & z_a - z_c & z_d \end{bmatrix} \begin{bmatrix} \beta \\ \gamma \\ t \end{bmatrix} = \begin{bmatrix} x_a - x_e \\ y_a - y_e \\ z_a - z_e \end{bmatrix}. \tag{2-18}$$

Najjednostavnije rješenje za rješavanje ovakvog linearnog sustava je Kramerovo pravilo koje daje sljedeća rješenja:

$$\beta = \frac{\begin{vmatrix} x_a - x_e & x_a - x_c & x_d \\ y_a - y_e & y_a - y_c & y_d \\ z_a - z_e & z_a - z_c & z_d \end{vmatrix}}{|A|}, \tag{2-19}$$

$$\gamma = \frac{\begin{vmatrix} x_a - x_b & x_a - x_e & x_d \\ y_a - y_b & y_a - y_e & y_d \\ z_a - z_b & z_a - z_e & z_d \end{vmatrix}}{|A|}, \tag{2-20}$$

$$t = \frac{\begin{vmatrix} x_a - x_b & x_a - x_c & x_a - x_e \\ y_a - y_b & y_a - y_c & y_a - y_e \\ z_a - z_b & z_a - z_c & z_a - z_e \end{vmatrix}}{|A|}, \tag{2-21}$$

gdje je A matrica:

$$A = \begin{bmatrix} x_a - x_b & x_a - x_c & x_d \\ y_a - y_b & y_a - y_c & y_d \\ z_a - z_b & z_a - z_c & z_d \end{bmatrix}, \tag{2-22}$$

i $|A|$ označava determinantu *A*.

Ukoliko se linearni sustav napiše pojednostavljeno:

$$\begin{bmatrix} a & d & g \\ b & e & h \\ c & f & i \end{bmatrix} \begin{bmatrix} \beta \\ \gamma \\ t \end{bmatrix} = \begin{bmatrix} j \\ k \\ l \end{bmatrix}, \tag{2-23}$$

tada su po Kramerovom pravilu rješenja sljedeća:

$$\beta = \frac{j(ei-hf)+k(gf-di)+l(dh-eg)}{M}, \tag{2-24}$$

$$\gamma = \frac{i(ak-jb)+h(jc-al)+g(bl-kc)}{M}, \tag{2-25}$$

$$t = \frac{f(ak-jb)+e(jc-al)+d(bl-kc)}{M}, \tag{2-26}$$

gdje je

$$M = a(ei - hf) + b(gf - di) + c(dh - eg). \tag{2-27}$$

Za presjek zrake i poligona, također prema [2], rješenje je sljedeće:

Za proračun sjecišta između zrake $e + td$ i ravnine u kojoj se nalazi poligon, zadan sa *m* vrhova p_1 do p_m i površinskom normalom *n*, koristi se implicitno zadana jednadžba:

$$(p - p_1) \cdot n = 0. \tag{2-28}$$

Ukoliko je $p = e + td$, *t* će biti:

$$t = \frac{(p_1 - e) \cdot n}{d \cdot n}. \tag{2-29}$$

Na ovaj način se računa točka sjecišta *p*, koju zraka pogodi ukoliko se nalazi u poligonu, u suprotnom slučaju zraka ne dolazi do poligona. Odgovor na pitanje da li je točka *p* unutar poligona može se dobiti projekcijom vrhova točke i poligona na *xy* ravninu. Najlakši način je iz točke *p* poslati 2D zraku i zbrojiti broj sjecišta te zrake i granice poligona. Točka će se nalaziti unutar poligona ukoliko je broj koji se dobije neparan. Ukoliko se dobije paran broj, točka se nalazi izvan poligona. Ovo rješenje je jednostavno i logično jer svaka zraka koja ulazi u poligon mora i iz njega izaći, kreirajući tako uvijek parni broj točaka presjeka.

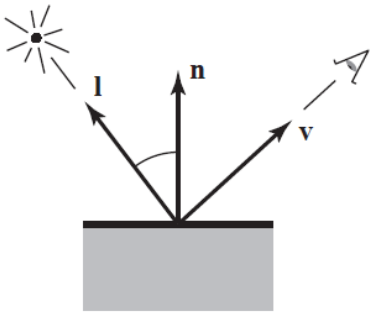
2.2.3. Sjenčanje

Po završetku drugog koraka slijedi treći korak zvan sjenčanje. Na temelju sada poznatih informacija koje se dobiju iz prethodnih koraka, odnosno kada je poznata vidljiva površina piksela, vrši se računanje vrijednosti piksela procjenom *shading model*-a (u nastavku teksta model sjenčanja). Većina modela sjenčanja su napravljeni kako bi uhvatili proces refleksije svjetlosti pri čemu su površine osvjetljene izvorom svjetlosti i reflektiraju dio svjetlosti prema kameri. Jednostavni modeli sjenčanja su definirani s obzirom na osvjetljenost koja dolazi točkastog izvora svjetlosti. Varijable koje su važne za refleksiju svjetlosti su smjer svjetlosti *l*, smjer pogleda *v*, normala površine *n* i karakteristika površine. Smjer svjetlosti je jedinični vektor u smjeru izvora svjetlosti, a smjer pogleda je jedinični vektor u smjeru oka ili kamere. Normala površine u ovom slučaju je jedinični vektor koji je okomit na površinu u točki u kojoj se događa refleksija. Karakteristika površine može biti boja, sjajnost ili neka druga svojstva.

Dva najpoznatija modela sjenčanja su Lambertianov i Blinn-Phongov model. Lambertianov model je jednostavniji i on se temelji na promatranju koje je Lambert napravio u 18. stoljeću; količina energije koja dolazi iz izvora svjetlosti i koja pada na određeno područje neke površine ovisi o kutu te površine u odnosu na izvor svjetlosti. Površina koja je okrenuta direktno prema izvoru primiti će najveće osvjetljenje, dok osvjetljenje neće primiti ona površina koja je okrenuta tangentno u odnosu na izvor svjetlosti. Sve površine koje se nalaze između ova dva krajnja slučaja svjetlost primaju proporcionalno kosinusu kuta Θ između normale na površinu i izvora svjetlosti (sl. 2.11.). Sljedeći izraz predstavlja Lambertianov model sjenčanja:

$$L = k_d \, I \max(0, n \cdot l) \tag{2-30}$$

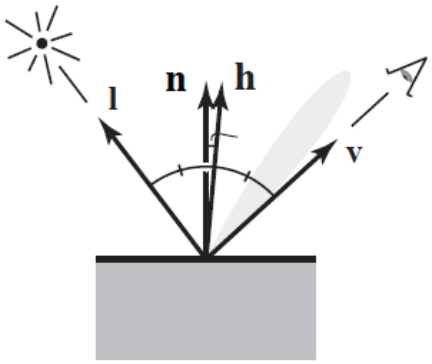
gdje je *L* boja piksela, k_d koeficijent difuzije (boja površine), *I* intenzitet izvora svjetlosti.



Sl. 2.11. Geometrijski prikaz Lambertian-ovog modela sjenčanja [2]

Ovaj model je neovisan o pogledu u smislu da boja površine ne ovisi o smjeru iz kojeg se na tu površinu gleda. Upravo iz ovog razloga on nije najidealniji model za realistične prikaze objekata jer ne proizvodi nikakvu istaknutost pojedinih dijelova površine, a stvara neprecizan i mat izgled objekta. Suprotno od Lambertiana, danas postoji model koji je jednostavan i često korišten u današnjici razvijen od strane Phonga, a kasnije nadograđen od strane J.F. Blinna te se upravo tako i naziva – Blinn-Phongov model sjenčanja.

Ovaj model se temelji na ideji postojanja refleksije koja je najsvjetlija kada su smjer pogleda *v* i smjer svjetlosti *l* simetrično pozicionirani preko normalne površine, a tada bi se pojavila i zrcalna refleksija. Kako se vektori pomiču iz zrcalne konfiguracije tako se i refleksija nakon toga proporcionalno smanjuje. Koliko blizu se refleksija nalazi zrcalnoj refleksiji govori usporedba vektora *h* (koji je simetrala kuta između *v* i *l*) sa normalom površine što je vidljivo iz slike 2.12.



Sl. 2.12. Gemetrijski prikaz Blinn-Phong-ovog modela sjenčanja [2]

Ukoliko je simetrala blizu normale površine, reflektirajuća komponenta bi trebala biti svijetla, a ukoliko nije blizu, tada bi reflektirajuća komponenta trebala biti prigušenog osvjetljenja. Ovaj

rezultat je postignut računanjem točkastog produkta između h i n (pri čemu treba obratiti pozornost da su to jedinični vektori i da $n \cdot h$ dostiže maksimum koji iznosi 1 kada su vektori jednaki), a nakon toga se taj rezultat prenosi na snagu $p > l$, kako bi se brže smanjila. Ta snaga, nazvana Phongovim eksponentom (engl. *Phong exponent*), kontrolira prividnu sjajnost površine. Simetralu je lako izračunati, ukoliko su v i l iste duljine, njihov zbroj je vektor koji dijeli kut između njih. Taj kut je potrebno normalizirati kako bi se dobila simetrala h. Sljedeći izraz predstavlja Blinn-Phongov model sjenčanja:

$$h = \frac{v+l}{\|v+l\|}, \tag{2-31}$$

$$L = k_d\, I \max(0, n \cdot l) + k_s\, I \max(0, n \cdot h)^p, \tag{2-32}$$

gdje je k_s koeficijent refleksije ili boja refleksije površine.

Površine koje uopće ne primaju osvjetljenje biti će renderirane kao potpuno crne što nije poželjno. Kako bi se te crne sjene izbjegle, modelu sjenčanja se dodaje jedna stalna komponenta čiji učinak na boju piksela ovisi samo o pogotku objekta, a ne ovisi o geometriji površine. Ova metoda je poznatija pod nazivom ambijentalno sjenčanje (engl. *Ambient shading*) i čini da površine izgledaju kao da su osvjetljenje „ambijentalnim“ svjetlom koje dolazi jednako sa svih strana. Ova metoda je obično objašnjena kao produkt boje površine sa bojom svjetla ambijenta, tako da se ambijentalno sjenčanje može podesiti ili za površinu pojedinačno ili za sve površine zajedno. Zajedno s ostatkom Blinn-Phong modela ova metoda predstavlja jedan konačni korisni model sjenčanja koji se može predstaviti sljedećim izrazom:

$$L = k_a\, I_a + k_d\, I \max(0, n \cdot l) + k_s\, I \max(0, n \cdot h)^n, \tag{2-33}$$

gdje je k_a ambijentalni koeficijent površine, ili „boja ambijenta“, a I_a je intenzitet ambijentalnog osvjetljenja.

Ukoliko postoji više od jedan izvor svjetlosti, pojavljuje se efekt superpozicije svjetlosti pa je tako konačni efekt zbroj svih efekata pojedinih izvora svjetlosti:

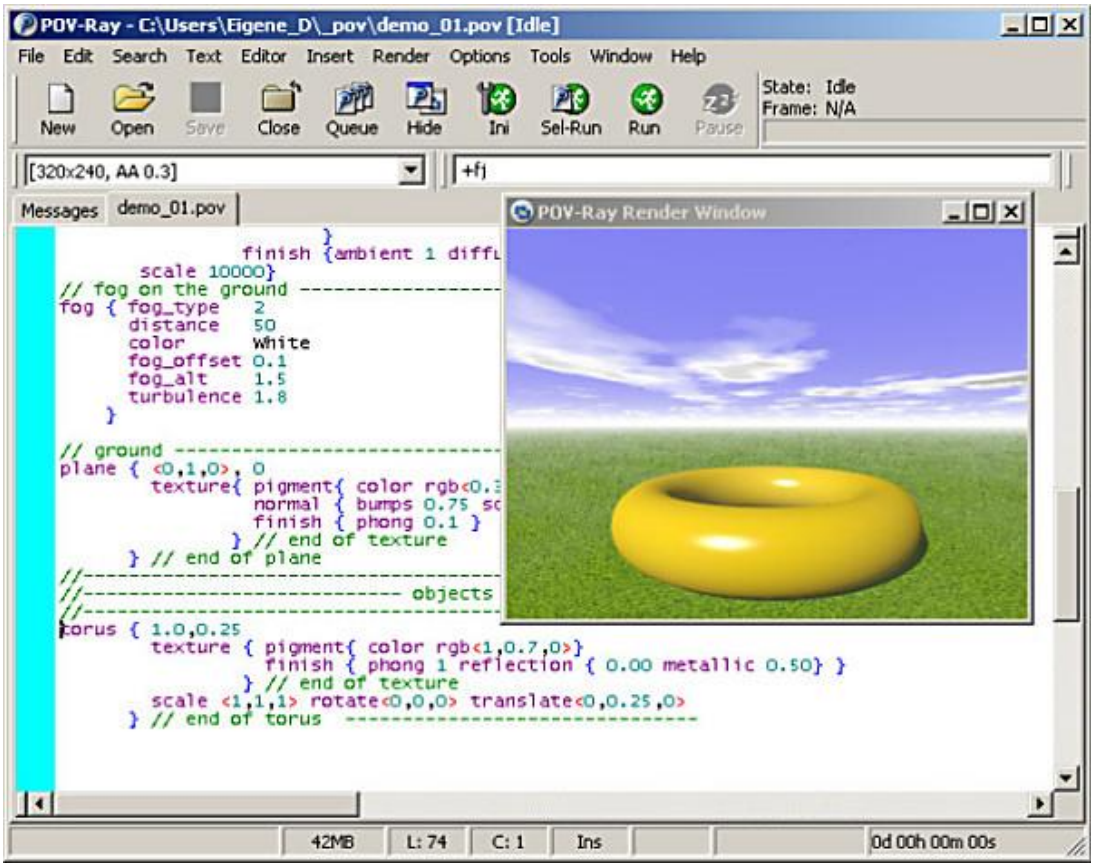
$$L = k_a\, I_a + \sum_{i=1}^{N} [k_d I_i \max(0, n \cdot l_i) + k_s\, I_i \max(0, n \cdot h_i)^p], \tag{2-34}$$

gdje su I_i, l_i i h_i intenzitet, smjer i simetrala *i*-tog izvora svjetlosti.

2.3. Alati za 3d modeliranje i renderiranje

2.3.1. POV-Ray

The Persistence of Vision Raytracer (POV-Ray) je otvoreni besplatni program za izradu trodimenzionalne grafike i za izradu animacija koji implementira *Ray tracing* algoritam [8]. Također se koristi za izradu fotorealističnih slika koji prikazuju objekte iz stvarnog svijeta ili vizualizaciju virtualnih objekata koji fizički ne postoje. Koristi se profesionalno u raznim znanstvenim poljima, primjerice biokemiji (istraživanje proteina), medicini, arhitekturi, inženjerstvu, matematici. Kod ovog programa svi se elementi definiraju preko teksta u integrirani *text editor*. Odličan je za početnike u 3D grafici jer je jednostavan unatoč prividnoj kompleksnosti. Nije nužno profesionalno poznavati određeni programski jezik, dovoljno je znati osnove programskog jezika C ili C++ jer je sintaksa vrlo slična, te je tako dostupan i u amaterske svrhe. Od velike važnosti je i bogati *help file.*



Sl. 2. 9. POV-Ray sučelje [9]

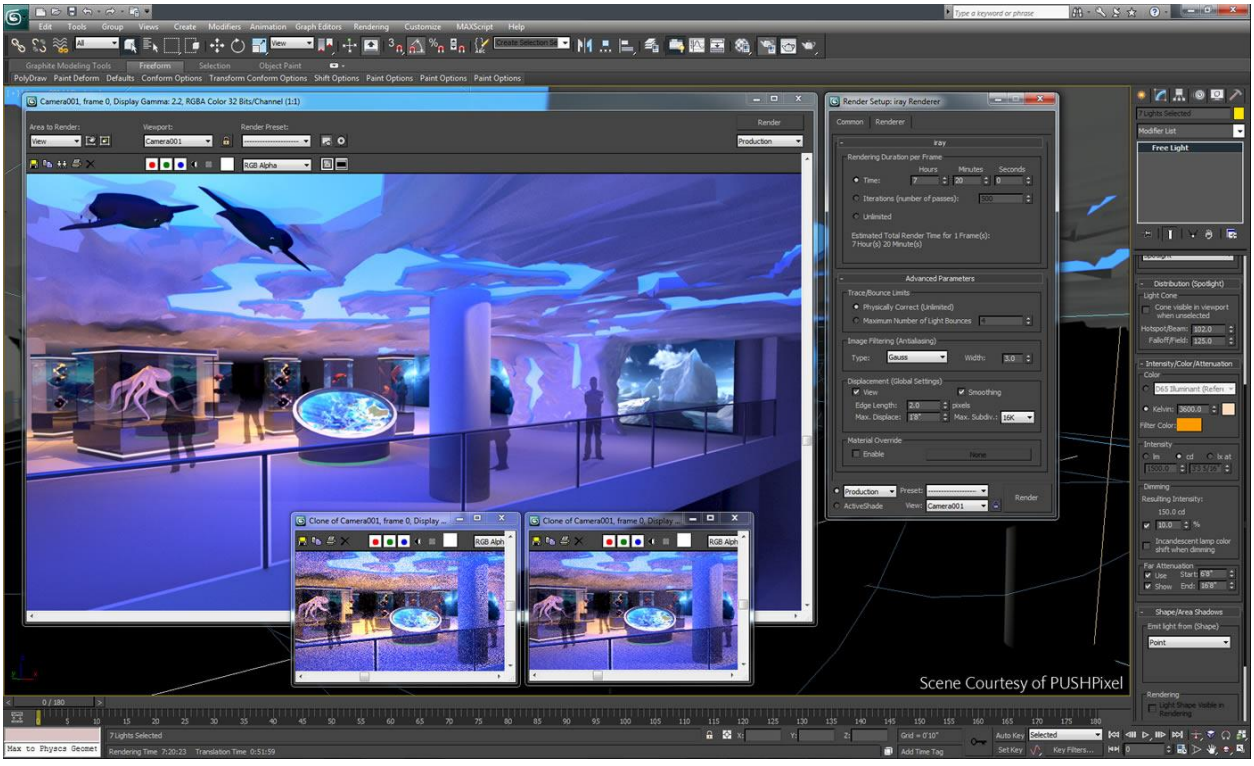
2.3.2. Autodesk Maya

Autodesk Maya, često skraćeno Maya, je još jedan program za izradu trodimenzionalne računalne grafike, animacija i efekata [10]. Dostupan je za Windows, OS X i Linux platforme. Koristi se za izradu interaktivnih 3D aplikacija, video igrica, animiranih filmova, tv serija, vizualnih efekata i digitalnih slika. Maya program uključuje prirodne zakone fizike kako bi kontrolirao ponašanje virtualnih objekata u računalnoj animaciji. Osim što simulira pokrete objekata i čestica (npr. kretanje oblaka, vode, dima; vrtnja prašine u tornadu, kretanje odjeće prilikom pokreta i slično), Maya omogućuje prikaz emocija animiranih likova pojačavanjem izraza lica i govora tijela. Potrebno je poznavati programski jezik *Maya Embedded Language* (MEL). Kako samo ime piše to je ugrađeni jezik koji služi za pojednostavljenje zadataka u program Maya. Ovo je skriptni jezik što znači da je moguće izvršavanje MEL naredbi u istom trenutku kada se pritisne tipka „*ENTER*“, bez konvertiranja koda u jezik koji razumije računalo. To ovaj jezik čini dosta brzim pa tako MEL nudi ubrzano rješavanje kompliciranih ili ponavljajućih zadataka. Postoji mogućnost postavljanja određenih vlastitih komandi drugim korisnicima Maya programa ukoliko oni te komande nalaze korisnim. Program je korišten u više poznatih filmova gdje svakako treba izdvojiti svima poznate „Stuart Little“, „The Matrix“, „Spider-Man“, „The Girl with the Dragon Tattoo“, „Frozen“, „Game of Thrones“, „The Walking Dead“, „Futurama“, „South Park“.

Sl. 2. 10. Autodesk Maya sučelje [11]

2.3.3. Autodesk 3ds Max

Autodesk 3ds Max, prije zvan 3D Studio i 3D Studio Max, profesionalni je 3D računalni grafički program za izradu 3D animacija, video igara i slika. Dostupan je za Windows platformu. Ovo je dosta poznat program, posebno u svijetu video igrica. Također se koristi i u filmskoj industriji za filmske efekte i TV reklame. 3ds Max radi s većinom glavnih renderera kao što su VRay ili Iray stvarajući nevjerojatne scene i upečatljive vizualne efekte. Koriste ga prestižne tvrtke koje se bave dizajnom i animacijom za stvaranje maštovitih likova i scena u igrama i arhitekturi. Alatima za modeliranje i animaciju su u zadnjoj verziji programa dodani i *shader*-i, dinamička simulacija, renderiranje, sučelje koje je prilagodljivo korisnicima i vlastiti programski jezik. Programski jezik koji se koristi je MAXScript. To je ugrađeni skriptni jezik koji automatizira ponavljajuće zadatke, kombinira postojeće funkcije u nove, stvara nove alate i korisnička sučelja. I ovaj program je zadužen za pomoć u izradi mnogih poznatih filmova: „Avatar“, „2012.“, „Alice in Wonderland“, „Black Hawk Down“, „Lara Croft: Tomb Raider“, „Iron Man“, „Harry Potter and the Deathly Hallows“, „Shutter Island“, „X-Man“, „Transformers“, „The Curious Case of Benjamin Button“.



Sl. 2.11. 3ds Max sučelje

2.3.4. SketchUp

SketchUp je još jedan u nizu 3D računalnih programa. Vrlo je jednostavan za upotrebu i lako ga je svladati. Namijenjen je primjeni u arhitekturi, građevini i mehaničkom inžinjerstvu, u proizvodnji namještaja, dekoriranju te u svrhe kreiranja filmova, igara te srodnih grana te profesije. Softver je vrlo jednostavan za korištenje te kao takav dopušta implementaciju izrađenih modela na Google Earth. Svi izrađeni modeli postaju javno dostupni na Google-ovu servisu 3D *Warehouse* kako bi svatko mogao doraditi postojeće modele. Ovaj program se razlikuje od prethodnih jer ne koristi nikakav programski jezik te je zbog toga jednostavan za korištenje bez potrebnog predznanja o CAD aplikacijama. SketchUp nam nudi mnogo mogućnosti, neke od njih jesu: kreativno oblikovanje i mijenjanje 3D objekata, kreiranje zahtjevne 3D geometrije uz nekoliko klikova mišem, kreiranje organskih oblika i terena, proizvoljno konstruiranje, dodjeljivanje boja i materijala iz baze podataka, mijenjanje postojećih materijala ili kreiranje vlastitih, sjenčanje s točnim položajem sunca (ovisno o položaju, vremenu i godišnjem dobu), animacija toka svjetlosti odnosno sjena, odabir optimalnog grafičkog prikaza (npr. "prikaz kao skica"), unos fotografija (teksture, osobe) i proizvoljno 3D obrađivanje, kreiranje 3D modela na osnovu ručnih skica, kreiranje/importiranje 3D simbola. Svi modeli mogu se eksportirati kao 3D modeli u 3DS, AutoCAD DWG, AutoCAD DXF I WRML formatu, a u 2D obliku mogu se eksportirati u formatima datoteka: PDF, BMP, JPG, TGA, PNG, AutoCAD DWG i AutoCAD DXF.

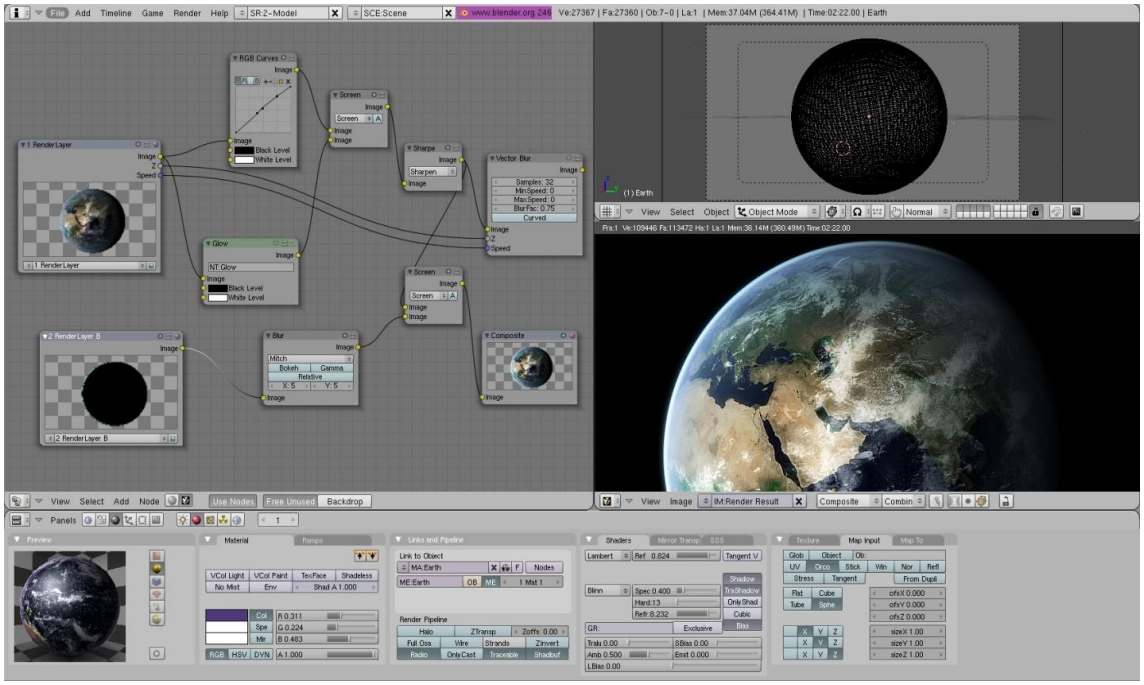
Ovaj program je osobno korišten prilikom jednog grupnog zadatka iz kolegija “Modeliranje i simulacija”. Projekt je glasio: Staklena cijev – poveznica dva kraja “lega grada”. Zadatak je bio prikazati 3D model na zadanu temu (sl. 2.12).

Sl. 2. 12. „Staklena cijev – poveznica dva kraja „lega grada““ - projekt izrađen u Sketchup-u

3. FOTOREALISTIČNO RENDERIRANJE U BLENDERU

Blender je besplatan i svima dostupan 3D programski paket koji nudi široki spektar alata, uključujući modeliranje, renderiranje, animiranje, video uređivanje, teksturiranje, mnoge vrste simulacija i stvaranja igara. Visoko kvalitetna 3D arhitektura omogućuje brzi i učinkovit radni tok. Male je izvršne veličine te je prenosiv. Podržan je na Windows, Linux i OS X platformama koje se redovito testiraju od strane razvojnog tima, kako bi se potvrdila određena kompatibilnost. S obzirom da je program otvorenog tipa, postoji mogućnost male i velike promjene koda baze. Program zbog toga neprestano nudi nove mogućnosti. Iako je besplatan, svatko može investirati u njega, sudjelovati i pomoći u unapređenju samog programa. Njegova upotreba je široko rasprostranjena, koristiti ga može svatko - pojedinci, male i velike grupe, velike firme i kompanije. Ima izvrsnu društvenu podršku na forumima. Postoji nekoliko neovisnih web stranica posvećenih Blenderu (forumi, blogovi, tutorijali). Jedan od najvećih foruma je Blender Artists [12] na kojem korisnici mogu zatražiti pomoć i dobiti ili pružiti savjete, raspravljati o Blenderu te pokazati svoje gotove radove.

Program je licenciran pod općom javnom licencom *(*engl. *GNU General Public License - The GPL)* [13]. To znači da on daje određena prava ali i zahtjeva određene obaveze. Što se prava tiče, program se može koristiti u bilo koju svrhu, mogu se vršiti izmjene programa, poboljšati program, ustupiti vlastita verzija i polaže se pravo na pristup izvornim kodovima. Program se može kopirati i distribuirati. U drugu ruku, što se obaveza tiče, svatko mora dostaviti kopiju GPL s programom, tako da se uvidi da su korisnici svjesni svojih prava pod licencom. Ukoliko dođe do izmjene koda, modificirana verzija se mora licencirati pod GPL licencom.



Sl. 3.1. Korisničko sučelje Blendera

Blender ima širok spektar mogućnosti upotrebe, sadrži veliki broj alata što se može učiniti zastrašujuće na samom početku korištenja. Vrlo je važno imati na umu da je Blender samo alat, a koliko dobro ćemo ga iskoristit je na nama. Motiviranost i volja za učenjem su ključni. Poznavanje osnovnih fizikalnih principa je nužno, a isto tako i pojmova kao što su materijali, teksturiranje, *shader*-i, mreže, mapiranje i slično. Razumijevanje ovih termina je potrebno kako bi se Blender iskoristio na najbolji mogući način.

Zanimljive mogućnosti Blendera su npr. klasično renderiranje u Cycles Engineu gdje se stvaraju fotorealistične slike, zatim simuliranje fluida gdje se može simulirati sve od prskajućih fluida do tople čokolade. Moguće je napraviti vlastiti ocean koji se nalazi nigdje drugo no u našoj mašti. Također se može prikazati kretanje dima ili vjetra. Što se fizike nadalje tiče, Blender nudi mogućnost simulacije "pucanja" raznih sustava. Tako se može vršiti ispitivanje elastičnosti i čvrstoće predmeta u prisustvu raznih parametara koji utječu na njihove promjene. Video uređivanje koje je prethodno već spomenuto je također opcija, a Blender nudi uobičajene alate za uređivanje videa, dodavanje isječaka ili slika, dodavanje prijelaza, rezova i drugih promjena, kao i uređivanje tempa i audio razina. Još jedna zanimljivost koju pruža je kombiniranje dvije ili više fotografija u jednu (engl. *compositing*). Na taj način mogu se dodati dubinske oštrine ili zamućivanja, kontrolirati boje, kontrast i količina svjetlosti slike. Blender je otvorio umjetnicima

nove puteve kroz takozvano digitalno kiparstvo. Program pruža veliki broj različitih kiparskih "četki" te velike količine alata za detalje koji model čine i više no realnim.

Iako u dosta slaboj upotrebi, Blender omogućuje stvaranje igara. Planira se razvijanje programa u ovom smjeru u većoj mjeri kako bi se i taj dio upotrebe Blendera povećao. Smatra se da ni tada ova pogodnost Blendera neće biti među prioritetima.

3.1. Cycles renderer

Cycles je Blenderov alat za renderiranje koji koristi *Ray-tracing*. Da bi bio u korisnoj upotrebi on se treba nalaziti kao aktivni renderer u gornjem zaglavlju programa. Kada je to učinjeno, interaktivno renderiranje može se započeti postavljajući 3D *View editor* na način crtanja *Rendered* koristeći prečac na tipkovnici SHIFT-Z.

Dok Blender renderer radi na način da računa koji objekt je vidljiv kameri, Cycles renderer koristi simulaciju ponašanja svjetla. Točnije rečeno Cycles renderer stvara sliku tako što prati put zraka svjetlosti kroz scenu te prati svjetlosne zrake tako što ih šalje od kamere, umjesto da ih šalje od izvora svjetlosti. Cycles pri tome u smislu fotorealizma daje bolje rezultate jer prikazuje potpuno osvjetljenje i daje fizički točne izračune.

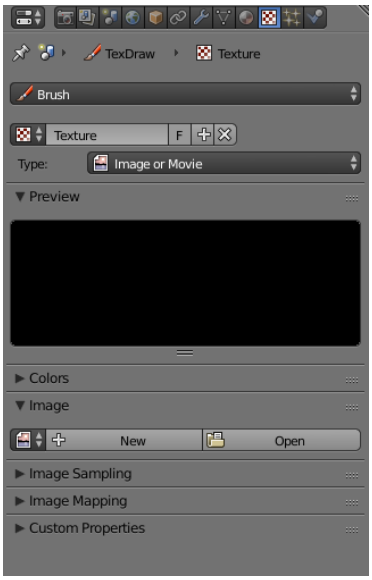
Iako se razvojem Cyclesa postižu sve veće i veće brzine renderiranja, i dalje je negativna strana ovog renderera upravo brzina renderiranja. Međutim, uzevši u obzir količinu operacija i informacija s kojim raspolaže te činjenicu da „imitira" kompleksni sustav stvarne svjetlosti, ova negativna strana se ne bi trebala uzimati za zlo ovako kompleksnom i korisnom rendereru. Otkako je Cycles Engine postao dijelom Blendera, sam program je postao mnogo korisniji te je i broj njegovih korisnika porastao.

3.1.1. Teksturiranje u Cycles-u

Izrada bilo koje scene u Blenderu bi bila gotovo nemoguća bez korištena teksturiranja (engl. t*extures)*. Teksturiranje u Blenderu podrazumijeva dodavanje dvodimenzionalne slike na 3D objekt s ciljem postizanja što realnijeg prikaza te je neizostavan dio računalnog modeliranja. Dodavanjem materijala i tekstura činimo upravo da modelirani sivi objekt poprimi željeni izgled te da on postane „stvaran". Primjenu u 3D grafici je prvi započeo Edwin Catmull 1974. [14].

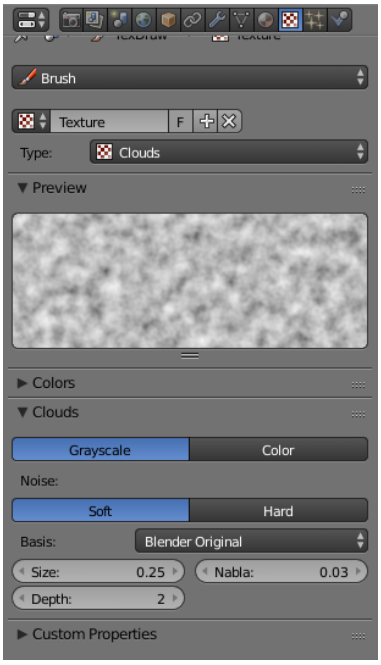
U početku se teksturiranje odnosilo na metodu koja je jednostavno zamotavala i mapirala piksele iz tekstura na trodimenzionalnu površinu. Napretkom i razvojem renderiranja i kompleksnog mapiranja stvorila se podloga za kreiranje fotorealističnih scena u stvarnom

vremenu uz korištenje smanjenog broja kalkulacija i proračuna (pri tome se ovo najviše odnosi na proračune svjetlosti).



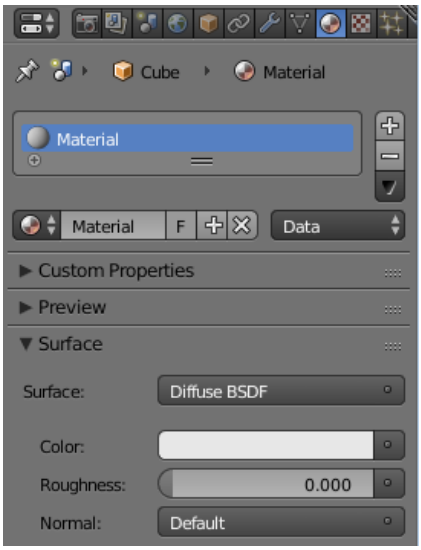
Sl. 3.2. Glavni prozor za kreiranje tekstura

Kreirati teksturu možemo uz pomoć određene fotografije ili videa ili koristiti već pripremljene teksture koje se naknadno mogu uređivati po vlastitoj volji.



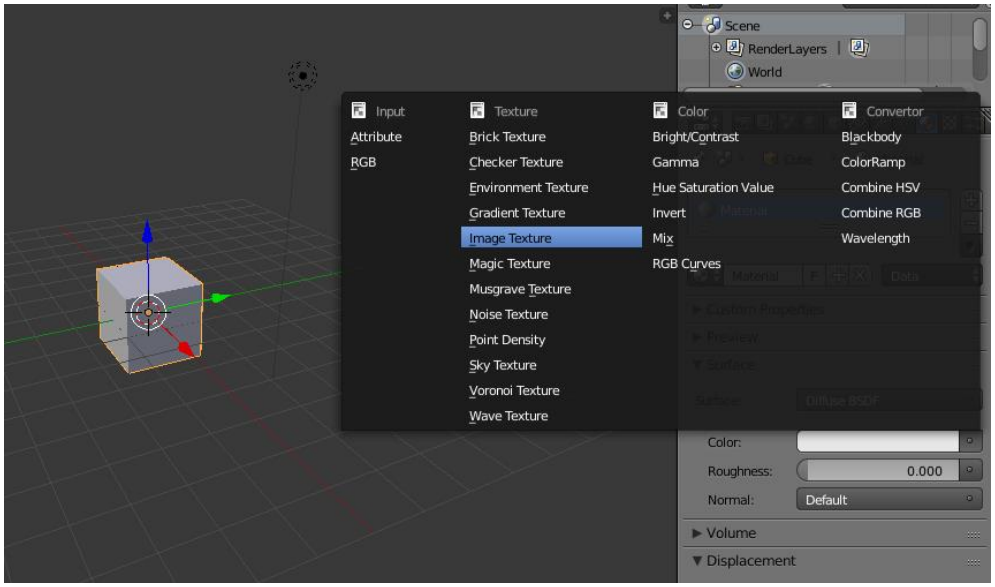
Sl. 3.3. Tekstura Clouds s opcijama podešavanja

Proces teksturiranja započinje dodavanjem slike na neki materijal. Odabere se objekt na koji se želi postaviti slika. Materijali sami po sebi ne mogu postići efekt realizma, međutim oni daju obilježja objektu, kao što su na primjer prozirnost (engl. *transparency*), efekt stakla (engl. *glass*), efekt emitiranja svjetlosti (engl. *emission*), itd.



Sl. 3.2. Glavni prozor sa naredbama za dodavanje materijala

Nakon što se objektu doda materijal, potrebno je kliknuti na sivu točkicu pored alata *Color*. Otvara se prozor sa opcijama prikazan na slici 3.3. .

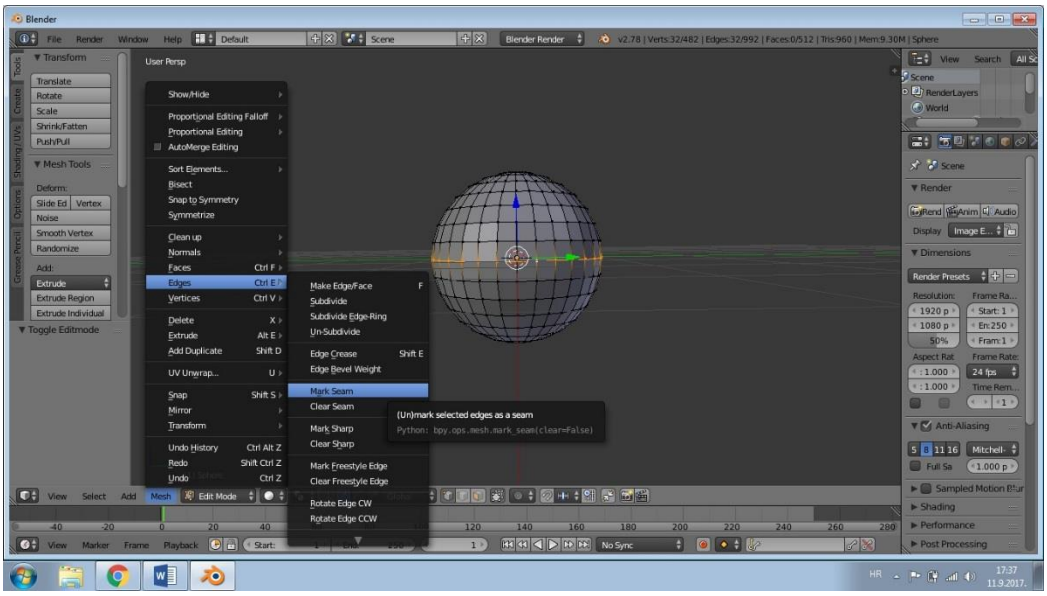


Sl. 3.3. Prozor za izbor teksture

Postoje i neke već gotove ponuđene opcije tekstura kao što su cigla, šah ploča ili nebo. Svakako je najzanimljivija opcija *Image Texture* gdje sami možemo odabrati izgled našeg objekta tako što mu pridodajemo sliku po vlastitoj želji. Nakon odabira slike slijedi UV *Unwraping*.

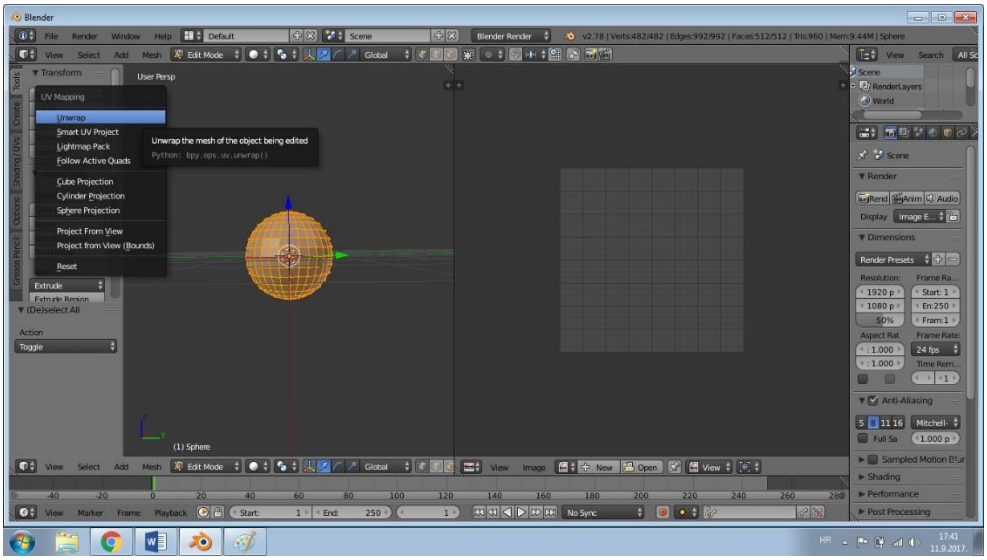
UV *Unwraping* je dio UV *mapping*-a koji označava tehniku „zamotavanja“ dvodimenzionalne teksture na trodimenzionalnu mrežu. Da bi se moglo odrediti kako će biti aplicirane na objekt, teksturama su potrebne koordinate mapiranja. U i V su oznake za osi ravnine jer su X, Y i Z već iskorišteni za koordinate u trodimenzionalnom prostoru.

Primjer za UV mapiranje:



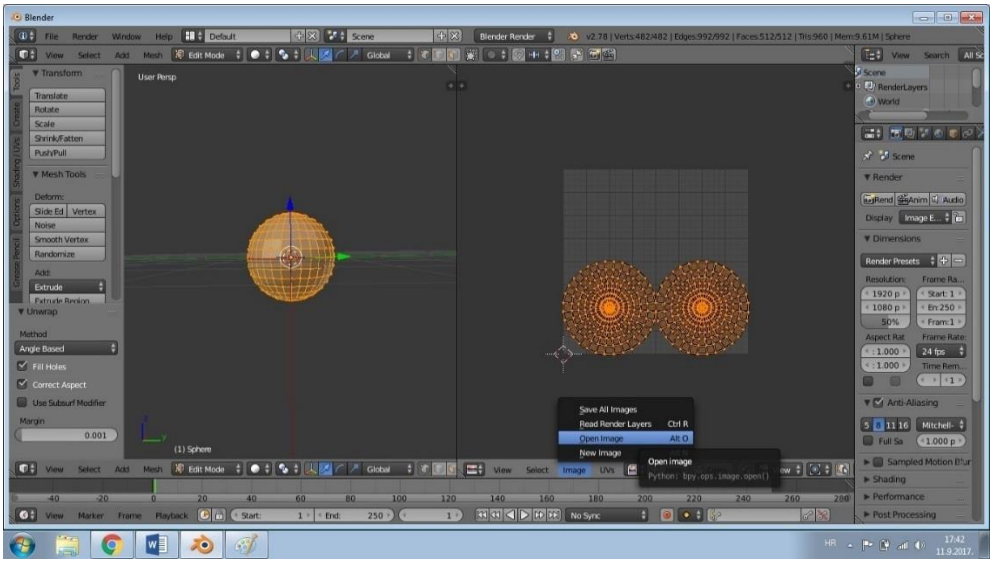
Sl. 3.4. Kreiranje UV Sphere

Prvi korak je kreiranje UV kugle (engl. UV *Sphere*). Slijedi odabir „ekvatora“ kugle koji uz pomoć naredbi *Mesh – Edges – Mark Seam* postaje crta razdjelnica dvije polukugle koje će se koristiti u nastavku.



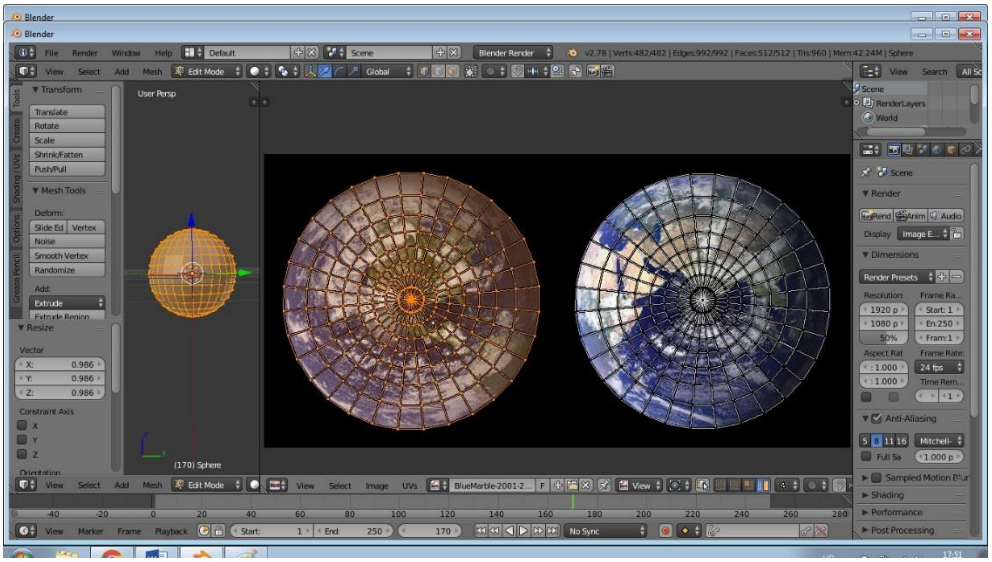
Sl. 3.5. Označena kugla (lijevo) i prozor s UV/Image načinom rada (desno)

Nakon toga je otvoren pomoćni prozor sa *UV/Image Editor* načinom uređivanja modela. Model kugle u *Object mode* (na slici 3.6. s lijeve strane) je cjelokupno označen, uz pomoć prečaca slovom „U“ otvoren je prozor sa naredbama među kojima je izabrana naredba *Unwrapp.*



Sl. 3.6. Prikaz kugle nakon Unwrapping-a

Kao što se vidi na slici 3.6. u desnom prozoru sada su dvije „kugle“, što zapravo znači da je kugla mapirana u dvodimenzionalnom prostoru. Sljedeći korak je klik na *Image – Open image* te odabir fotografije po želji koja je prethodno spremljena na računalu. U ovom slučaju će to biti dvostrani prikaz Zemljine površine.



Sl. 3.7. Dodijeljivanje položaja poluglama u odnosu na pozadinsku sliku

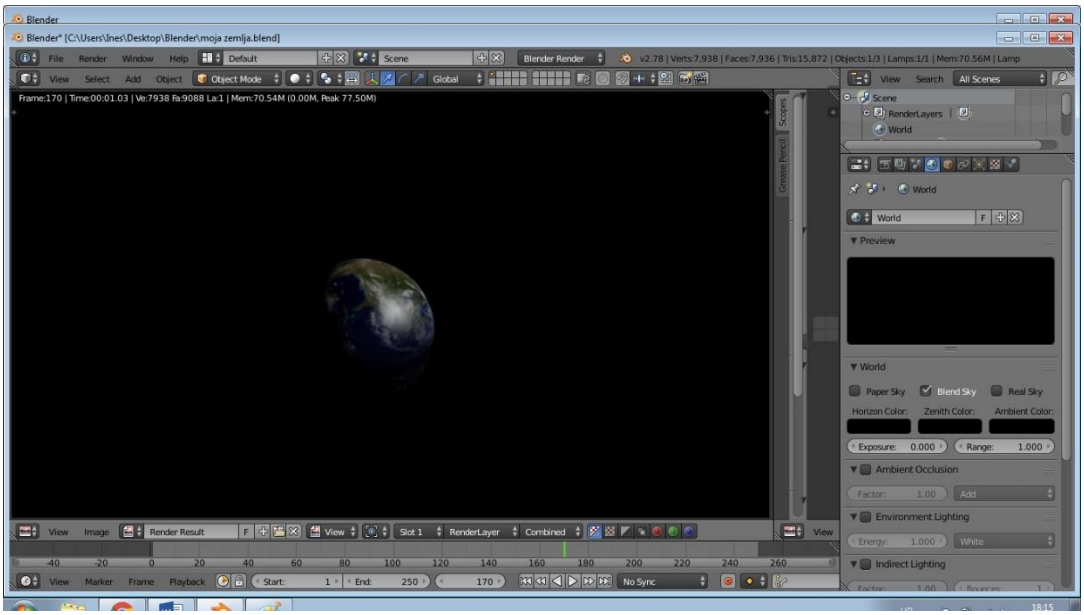
Uz pomoć naredbi S – *scale*, G – *grab*, R – *rotate*, B – *border selection* i A – *select/deselect all* osigura se podudaranje potrebnih dijelova slike sa rubovima modela. (sl. 3.7.)

UV mapiranje je u ovom koraku završeno, no primjer će biti razvijen do kraja zato što se u sljedećim koracima vidi i proces teksturiranja te krajnji rezultat.

Sl. 3.8. Viewport Shading izbornik

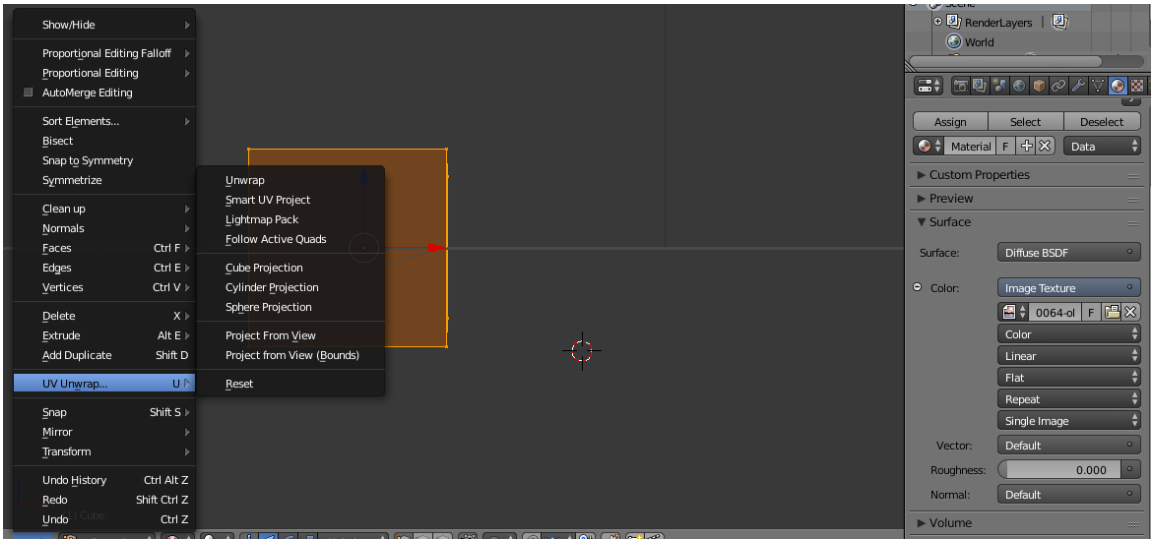
Ponovno se prebaci u *Object mode* te se iz *Viewport Shading* izbornika, vidljivog na slici 3.8. s desne strane, odabere *Texture – Type – Image or movie*. Upotrijebi se ista slika Zemlje koja je već upotrijebljena kod UV mapiranja. Također je potrebno dodati kugli teksturu u obliku

materijala: *Material – New*. Nakon povratka u UV mapu koordinate postavimo na „UV“, *Map* na *UVMap*, a *Projection* na *Flat.* Rezultat je vidljiv na slici 3.9..

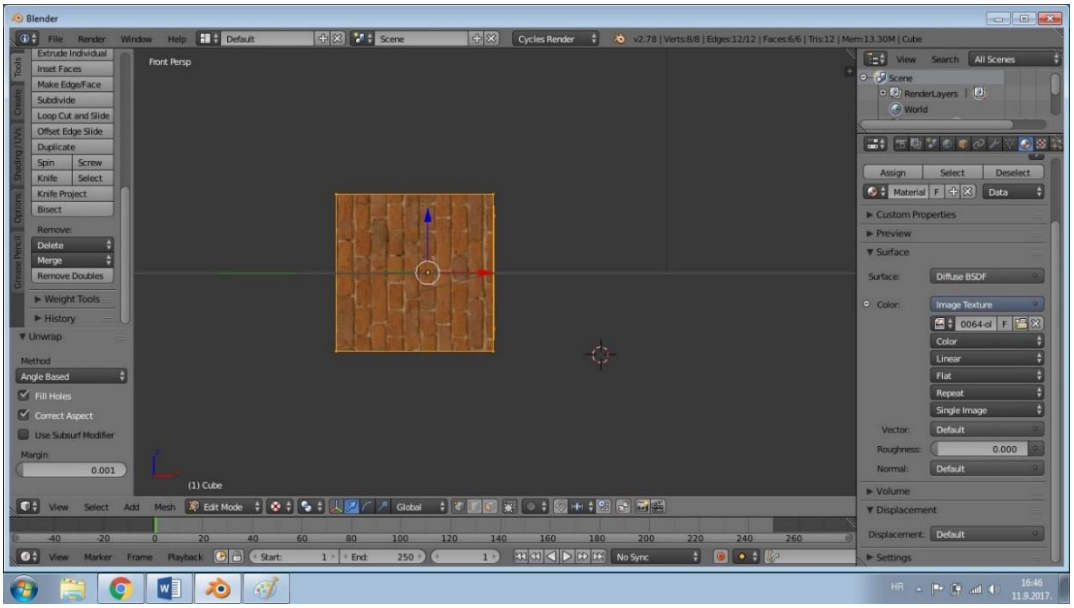


Sl. 3.9. Rezultat UV mapiranja

Sada kada je UV mapiranje objašnjeno može se prijeći na sljedeći korak procesa teksturiranja, a to je da se u glavnom prozoru izabere naredba *Edit mode*, zatim klikne U slovo na tipkovnici kao prečica do UV *Unwrapping* i od ponuđenih opcija se izabere *Unwrap* što je prikazano na slici ispod (sl. 3.10.).



Sl. 3.10. Unwrapping



Sl. 3.11. Rezultat teksturiranja stranica kocke

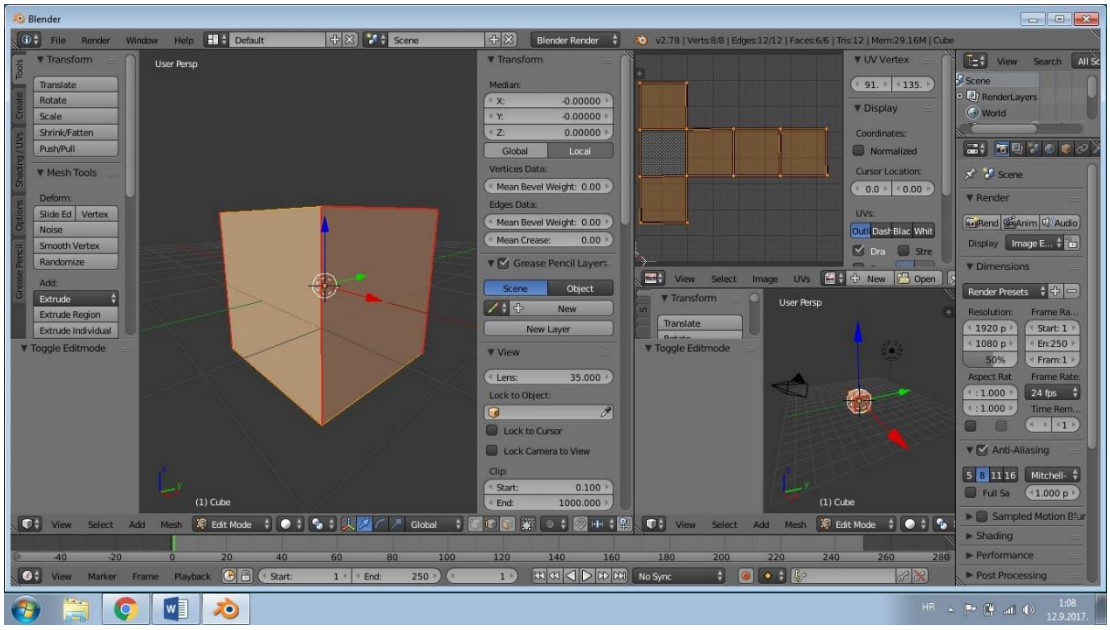
Rezultat je prikazan na slici 3.11. .Ovo nije krajnji rezultat teksturiranja koje je podložno daljim izmjenama, no, iako vrlo prost i jednostavan, ovo je vrlo dobar primjer osnovnog principa teksturiranja. Nešto drugačiji postupak teksturiranja je također vidljiv i na slici 3.8., unutar objašnjenja za UV *mapping*.

3.1.2. Bump mapping u Cycles-u

Bump mapping je tehnika kojoj se simuliraju male nepravilnosti na površini objekta, primjerice ispupčenja, udubine ili bore. U svijet modeliranja ju je uveo James Blinn 1978. [15]. Tehnika realizaciju postiže perturbacijom (uznemiravanjem) normala na površini objekta i korištenjem tih perturbiranih normala prilikom vršenja proračuna svjetlosti. Kao rezultat se dobije neravna površina koja se čini realnijom od glatke površine. Za jednostavnije objašnjenje se uzima sljedeće: *Bump* mape su crno bijele slike (skala sivih tonova), sa 8bitnim informacijama o boji, što znači da mogu imati 256 različitih nijansi bijele, crne ili sive boje. Vrijednosti sivih tonova govore Blenderu da li se radi o „gore“ ili „dolje“, odnosno o ispupčenjima ili udubljenima. Kada su vrijednosti blizu 50% sive boje, tada se radi o jako malo ili nijednom detalju na površini objekta. Kada se vrijednost približi vrijednosti bijele boje, efekt se pojavljuje u vidu izbočina, dok za vrijednosti bliže crnoj boji efekt se pojavljuje u vidu udubina. *Bump* mapiranje je jako korisno za kreiranje sitnih detalja na modelu. Najbolji primjer za to su pore na lišću, pore na koži, bore itd. Isto tako je brže i efikasnije od nekih drugih mapiranja te troši

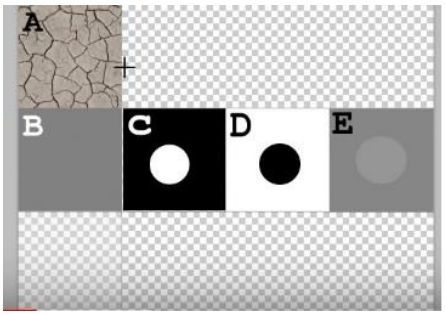
manje resursa za istu količinu detalja jer geometrija ostaje nepromijenjena. Loša strana ovog mapiranja jest ograničenje da perturbira samo normale površine objekta, ne mijenjajući pritom normale ispod površine. Zbog toga sjene i siluete ostaju netaknute što je posebno vidljivo kod većih simuliranih nepravilnosti.

Za primjer *Bump mapping*-a može se uzeti model obične kocke koja se rastavlja u dvodimenzionalni oblik *UV Unwrapping*-om te joj se dodaje materijal i tekstura.

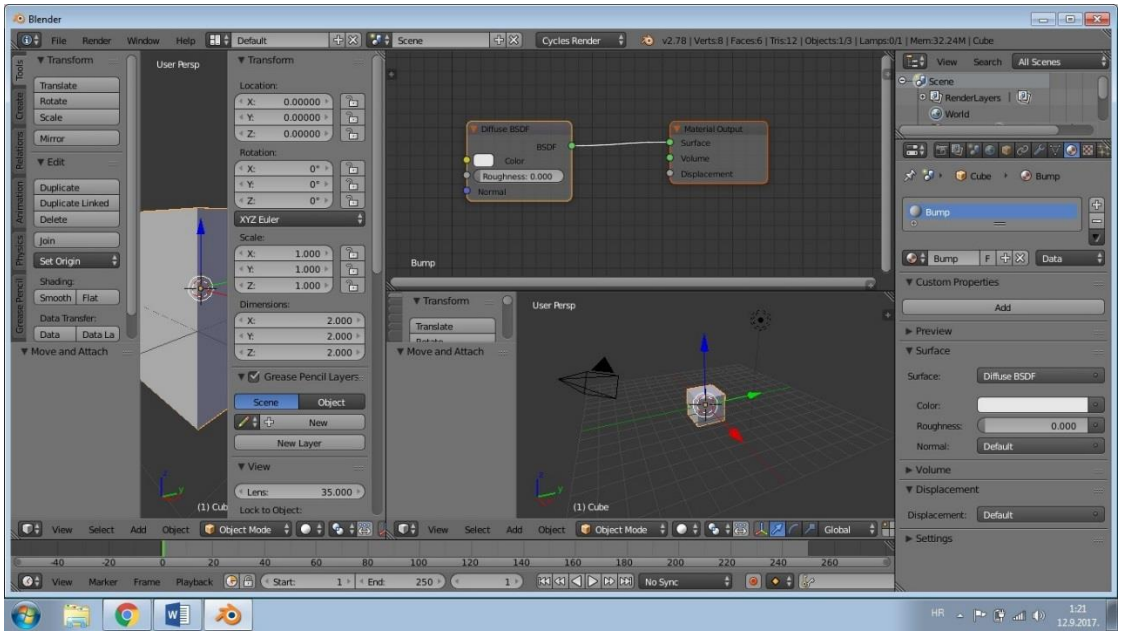


Sl. 3.12. Kreiranje modela kocke u 3D i 2D obliku

Detaljnije o UV *Unwrapping*-u te materijalima i teksturama je objašnjeno ranije u radu. Slika koja je korištena za teksturiranje je unaprijed pripremljena kako bi odgovarala dimenzijama stranica kocke, a njen izgled se može vidjeti na slici 3.13.

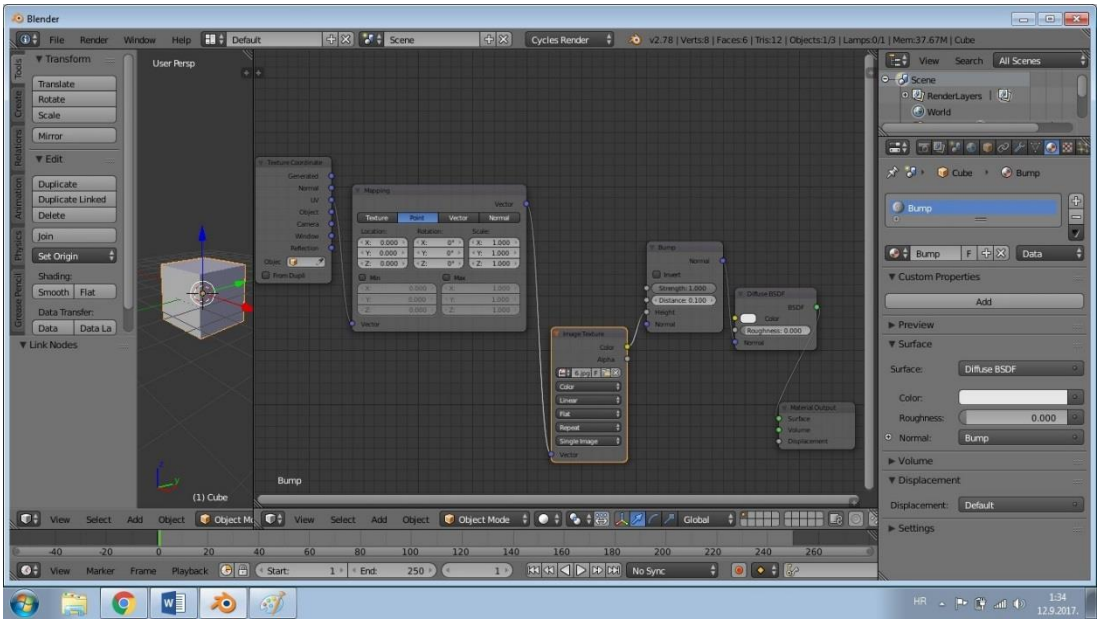


Sl. 3.13. Slika korištena za Bump mapping



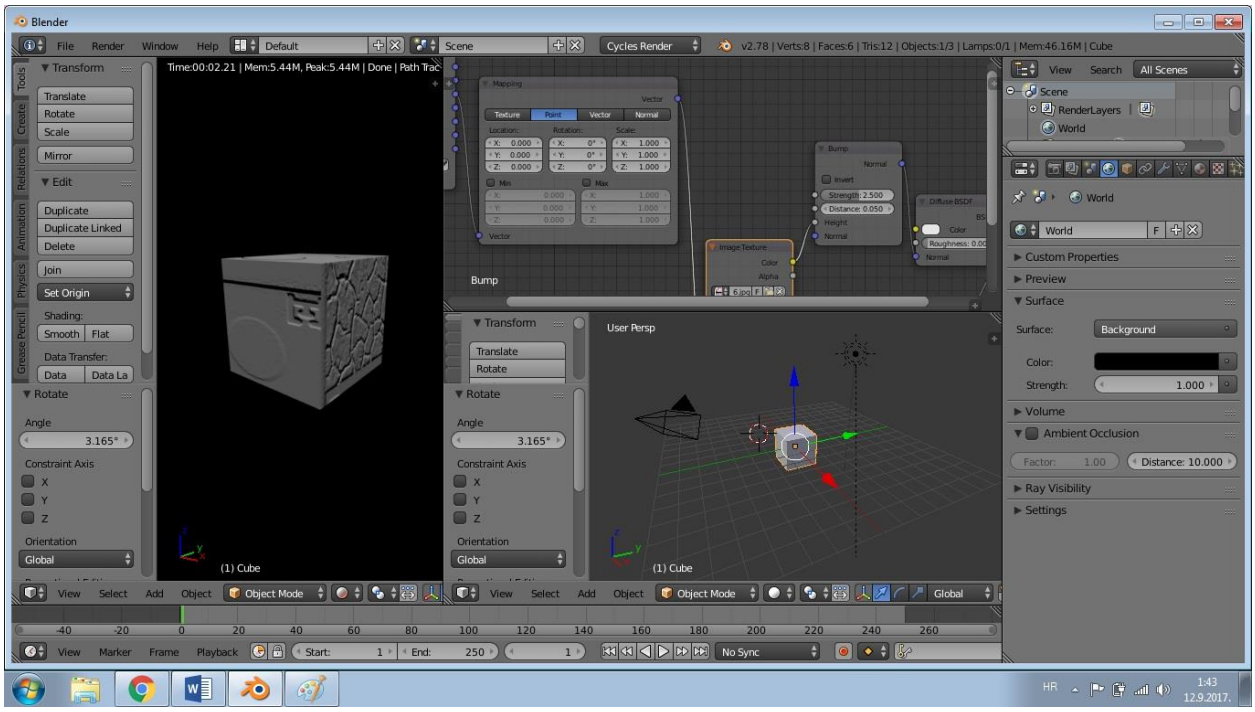
Sl. 3.14. Prikaz rada u Node Editor-u

Prozor koji je bio u UV/Image Editor-u se prebaci u Node Editor. U Node Editor-u se uz pomoć klika na „SHIFT + A“ dodaju sljedeći čvorovi: *Input – Texture Coordinate*, *Vector – Mapping*, *Texture – Image Texture* (unutar kojeg se odabere *Open Image* i zatim izabere željena slika čiju teksturu će kocka primiti), *Vector – Bump* (Sl. 3.15.).



Sl. 3.15. Raspored čvorova (engl. nodes) unutar Node editor-a

Kada je sve posloženo i pravilno spojeno potrebno je vratiti se u glavni prozor gdje se kocka nalazi u *Object Mode*, a zatim se vrši renderiranje.



Sl. 3.16. Model kocke nakon renderiranja

Jasno je vidljivo da ono što je na slici bilo crno ili bliže crnoj boji sada na kocki predstavlja udubine, a ono što je bilo bijelo ili se približavalo bijeloj boji predstavlja izbočine. Ljudsko oko upravo tako i vidi predmete, ono što je ispred ili izbočeno imati će svjetliju boju, a ono što je iza ili udubljeno imati će tamniju nijansu. Ova tehnika radi upravo na tom principu te daje iluziju ljudskom oku. Koristeći bijelu boju za rubove postiže se efekt izbočenosti, a crnu udubljenosti.

3.2. Shader-i u Cycles rendereru

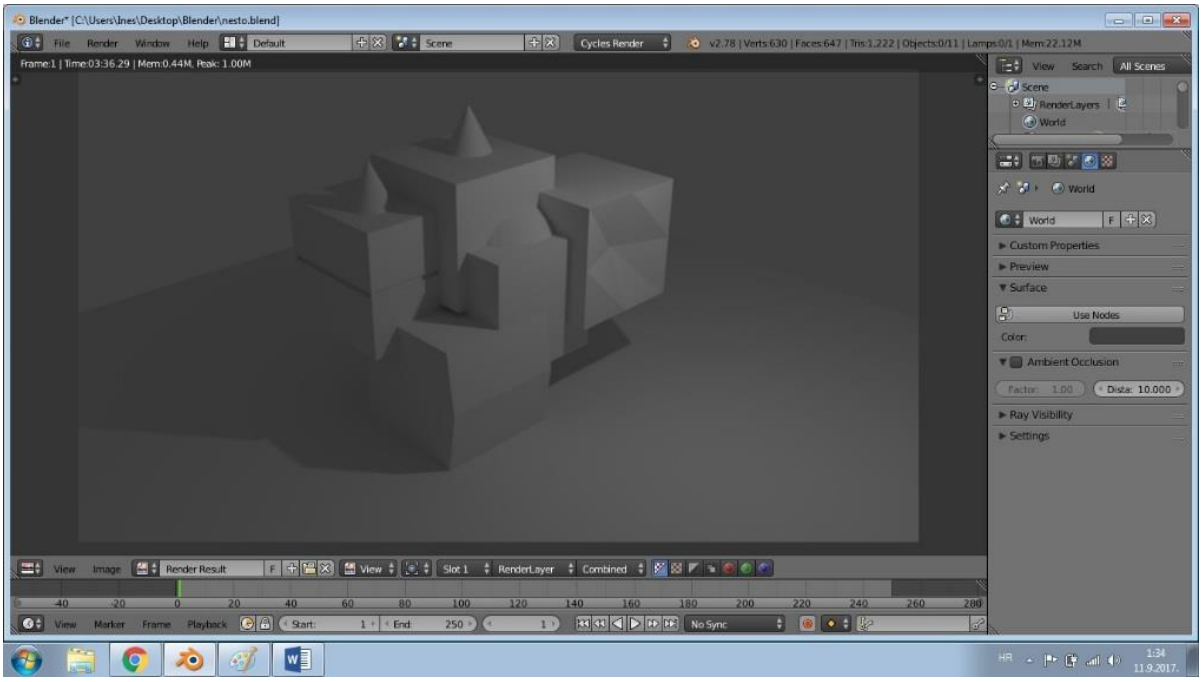
3.2.1 Anisotropic

Anisotropic shader se koristi kada se objektu želi dodati sjajni odsjaj. Tangente koje se koriste za sjenčanje su izvedene iz aktivne UV mape, a ako UV mapa nije dostupna, tangente su automatski generirane koristeći sferno mapiranje na temelju *mesh bounding box*-a. Ponaša se identično kao *Glossy shader*, ali refleksiju skreće u jednom smjeru. Koristi se za realističan prikaz brušenih metala ili materijala gdje se svjetlo ne reflektira ravnomjerno. Najpoznatiji primjer jeste dno tave ili neke druge posude za pečenje hrane.

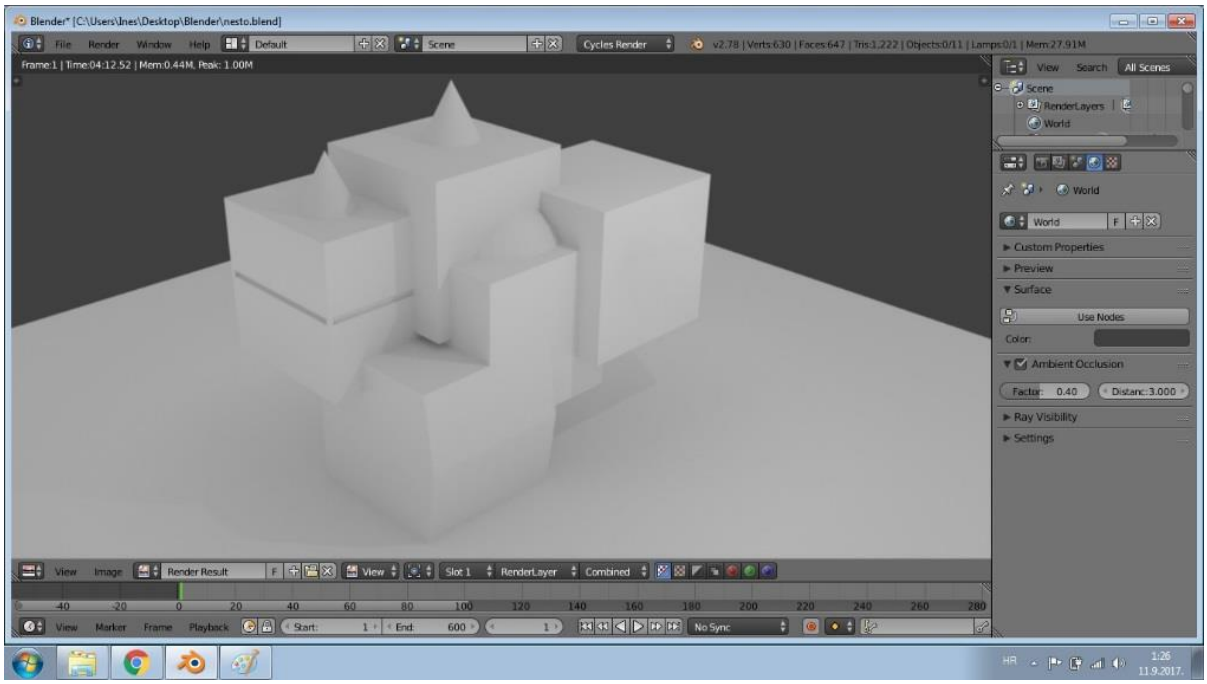
Sl. 3.17. Primjer korištenja Anisotropic shader-a

3.2.2 Ambient occlusion

Ambient occlusion ili "okluzija ambijenta/okoline" je tehnika sjenčanja i renderiranja koja se koristi za izračunavanje izloženosti svake točke scene na osvjetljenja koja dolaze iz cijele okoline. Ova tehnika izračunava tamno sjenčanje na uglovima i pukotinama. Koristi se za naglašavanje kontaktnih točaka, simulira sjene i lažira osvjetljenja okoline. Obično se koristi za primjenu na cijeloj sceni, ali je korisna tehnika i za primjenu na samo jednom objektu. U stvarnom svijetu ovakvo nešto ne postoji te ova tehnika zapravo nema nikakve veze sa osvjetljenjem. *Ambient occlusion* je jednostavan trik za renderiranje koji uljepšava izgled scene i pokušava ju učniti što realnijom.



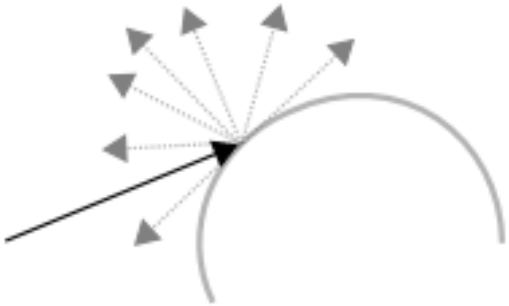
Sl. 3.18. Renderirani model prije korištenja Ambient Occlusion shader-a



Sl. 3.19. Renderirani model poslije korištenja Ambient Occlusion shader-a

3.2.3 Diffuse BSDF[1]

Diffuse shader je *shader* pri čijem korištenju objekt prima svjetlost koja se širi bez vidljivih refleksija. Jednostavno rečeno, *Diffuse shader* određuje boju objekta kada do njegove površine dopire bilo koja vrsta svjetlosti. Koristi se kod objekata koji ne posjeduju svojstvo reflektiranja ili za miješanje sa drugim *shader*-ima. Takve površine su naprimjer zidovi, papir ili pijesak.



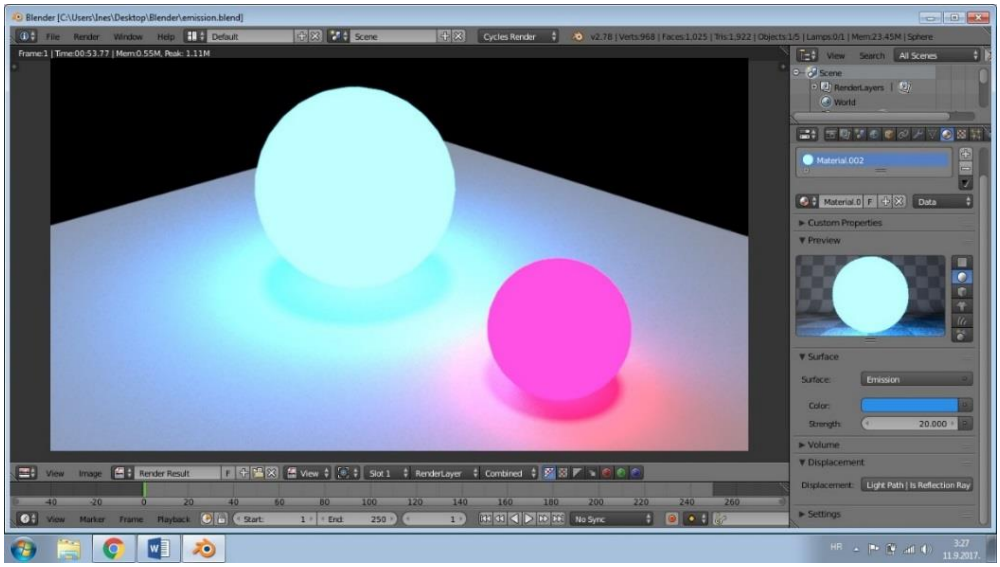
Sl. 3.20. Geometrija širenja zrake svjetlosti za Diffuse shader [16]

Sl. 3.21. Izgled objekta nakon primjene Diffuse shader-a [16]

[1] BSDF je oznaka za „*Bidirectional Scattering Distribution Function*“. Prijevod na hrvatski jezik glasi: „Dvosmjerna distribucijska funckija raspršenja“.

3.2.4 Emission

Ovaj *shader* čini da objekt isijava svjetlost iz samoga sebe te na taj način osvjetljava ostale kao i samu scenu. Vrlo je jednostavan za korištenje te se zapravo i upotrebljava samo za objekte koji isijavaju svjetlost (žarulja, iskra, vatra, svijeća).



Sl. 3.22. Izgled objekta nakon primjene Emission shader-a

3.2.5 Glass BSDF

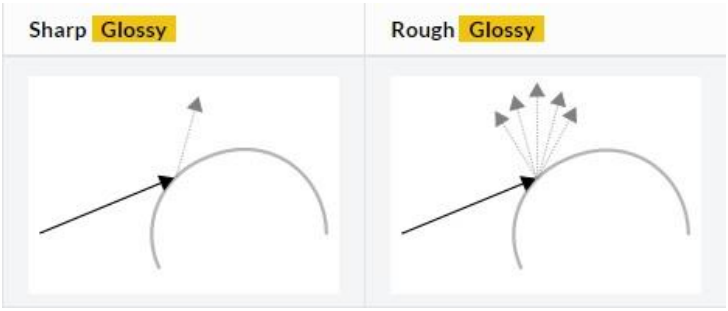
Glass shader čini da objekt poprimi teksturu pravog stakla. Prilikom doticaja površine objekta svjetlost se savija i reflektira prema indeksu loma svjetlosti koji određuje koliko će se svjetlost prelomiti prilikom prolaska kroz površinu objekta na kojem je primijenjen *Glass shader* te koliko nastale refleksije vidljive.

Sl. 3.23. Geometrija širenja zrake svjetlosti za oštar Glass efekt (lijevo) i za hrapav Glass efekt (desno) [16]

Sl. 3.24. Izgled objekta za oštar Glass efekt (lijevo) i za hrapav Glass efekt (desno) [16]

3.2.6 Glossy BSDF

Koristi se za dodavanje refleksije objektima u kojima se tada može vidjeti odraz svjetlosti, drugih objekata, okoline itd. Jako često se primjenjuje, a posebice u kombinaciji sa *Diffuse shader*-om. Tom prilikom se stvaraju materijali koji su u čestoj upotrebi u svakodnevnom životu, kao što su plastika, metal, keramika, drvo. Koristeći samo *Glossy shader* može se uspješno prikazati zrcalo.



Sl. 3.25. Geometrija širenja zrake svjetlosti za sjajan Glossy efekt (lijevo) i hrapav Glossy efekt (desno)[16]

Sl. 3.26. Izgled objekta za sjajan Glossy efekt (lijevo) i hrapav Glossy efekt (desno) [16]

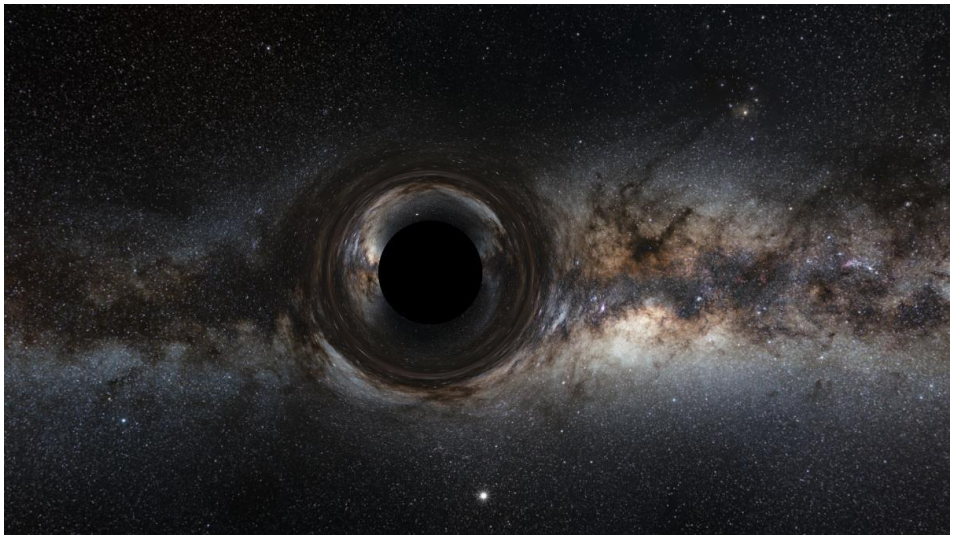
3.2.7 Hair BSDF

Kao što samo ime kaže, *Hair shader* služi za vjerodostojno i realistično prikazivanje kose, dlaka, krzna i slično. Prikazati kosu se može i na druge načine (kompliciranije, s mnogo geometrije), ali *Hair shader* to čini daleko lakše i jednostavnije tako što se bazira na cjelinu, umjesto na svaku dlaku pojedinačno. Iako pogledavši izbliza ovaj efekt čini dosta „ružan“ posao, činjenica da iz normalne perspektive ili normalne udaljenosti od ljudskog oka krajnji rezultat izgleda skoro pa savršeno čini da je ovaj *shader* omiljena tehnika za fotorealistično prikazivanje kose, dlaka i krzna u Blenderu.

Sl. 3.27. Izgled objekta nakon primjere Hair shader-a

3.2.8 Refraction BSDF

Ponaša se identično kao *Glass shader*, jedina razlika je što ovaj *shader* nema komponentu refleksije. Koristi se kada je potrebno svjetlost prelomiti o objekt ali ne i reflektirati. Iako na prvu pomisao pomalo nejasno što zapravo čini, ovaj *shader* služi za prikazivanje pojava kao što su deformacije uzrokovane toplinom ili za prikaz crnih rupa, što opravdava nejasnoću s početka obzirom da ovo nisu tako česte pojave.



Sl. 3.28. Prikaz crne rupe u svemiru uz pomoć Refraction shader-a

3.2.9 Subsurface scattering

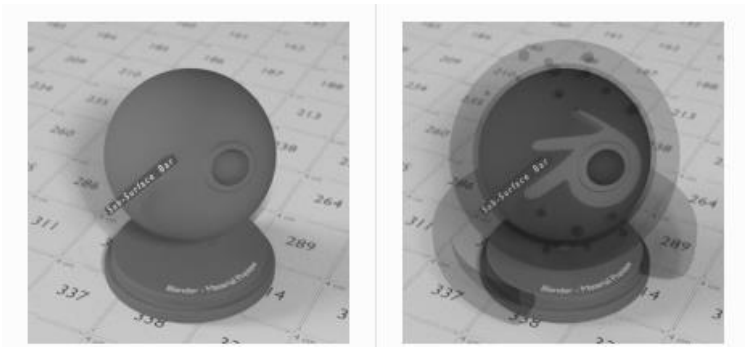
Mnogi organski i neorganski materijali nisu totalno neprozirni na površini što omogućava svjetlosti da se ne odbije u potpunosti od gornje površine objekta, nego prodire duboko u površinu. Svjetlost se tu rasprši, te izlazi van na drugačijoj lokaciji od prethodne, prihvaćajući svojstva i boju unutrašnjosti. *Subsurface scattering* (SSS) se koristi kod prikazivanja ljudske ili životinjske kože, vanjske ljuske voća i povrća, za prikaz voska ili nekih drugih tvari u obliku želea (med) i glavni je alat za prikaz istih. Fotorealizam navedenih primjera ne bi bio moguć bez ovog *shader*-a. Važno je naglasiti da su SSS i *Diffuse shader* isti alati, razlika je u tome koliko daleko svjetlo može prodrijeti i raspršiti se u unutrašnjosti objekta prije nego što je apsorbirana ili odbijena nazad u prostor.

Sl. 3. 29. Renderirani prikaz lica bez korištenja SSS (lijevo) i s korištenim SSS (desno)

3.2.10 Transparent BSDF

Koristi se za efekt transparentnosti, bez refrakcije. Svjetlost prolazi ravno kroz površinu, kao da geometrija ne postoji, te ovaj *shader* na put svjetla utječe drugačije od drugih dvosmjernih distribucijskih funkcija raspršenja. Potrebno je naglasiti da samo čisti bijeli *Transparent shader* će za rezultat imati potpunu transparentnost.

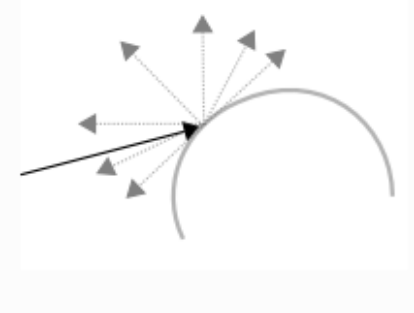
Sl. 3.30 Geometrija širenja zrake svjetlosti za Transparent shader [16]



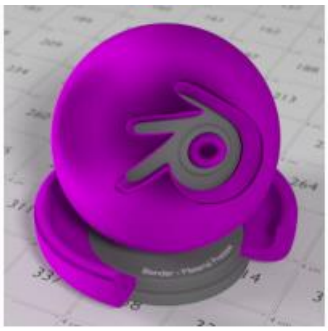
Sl. 3.31. Izgled objekta nakon korištenja Transparent shader-a [16]

3.2.11 Velvet

Kada se u Blenderu želi realistično prikazati odjeća, tada se koristi *Velvet shader*. Nije koristan u samostalnoj upotrebi, no u kombinaciji sa drugim *shader*-ima daje poprilično dobre rezultate.



Sl. 3.32. Geometrija širenja zrake svjetlosti za Velvet shader [16]



Sl. 3.33. Izgled objekta nakon korištenja Velvet shader-a [16]

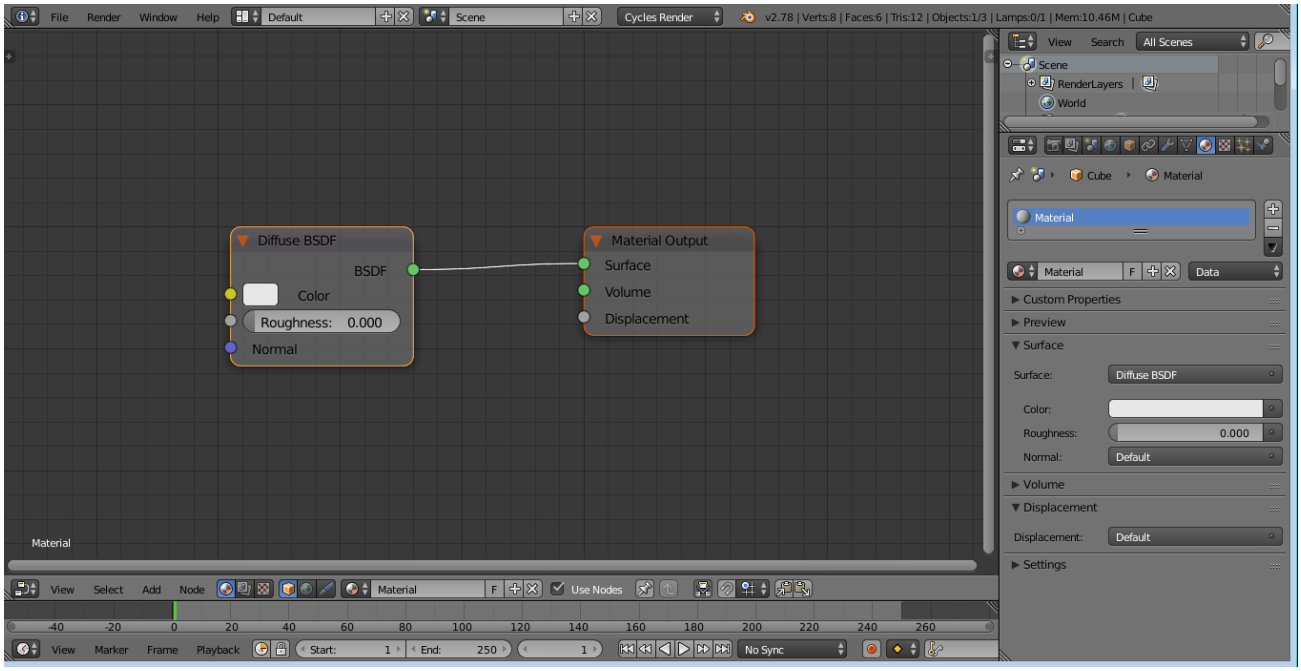
3.2.12 Volume absorption i Volume scatter

Volume absorption je *shader* koji omogućava da se svjetlo apsorbira prilikom prolaska kroz volumen. Najčešće služi za prikazivanje vode ili stakla, a uz *Volume scatter* (*shader* koji čini da se svjetlost prilikom prolaska kroz volumen raspršava) služi i za vjerodostojni prikaz dima. Tipična upotreba *Volume scatter shader*-a je kreiranje magle.

Sl. 3.34. Izgled objekta nakon korištenja Volume Scatter (lijevo), Volume Absorption (sredina), Volume Scatter i Volume Absorption (desno) [16]

3. Node Editor

Node Editor je editor koji olakšava rad u Blenderu na način da korisnicima pruža mogućnost olakšanog rješavanja složenih zadataka uz pomoć "paketa" čije funkcije pretvaraju određene ulaze u izlaze. On također pruža korisnicima organiziraniji uvid u tijek procesa kreiranja željenog modela ili scene. Vrsta je sučelja koja je fleksibilna za rad i kombinira najbolje od svijeta programiranja i svijeta korištenja Blendera.



Sl. 3.35. Izgled Node Editor-a

Dijelovi glavnog izbornika *Node editor*-a su:

View – izbornik koji mijenja pogled *Node editor*-a;

Select – izbornik koji dopušta biranje jednog čvora ili grupe čvorova;

Add – dodavanje čvorova;

Node – akcije sa odabranim čvorom;

Material, Compositing, Texture – grupe čvorova;

Use Nodes – govori programu da li da koristi čvorove prilikom računanja boja ili renderiranja gotove slike ili ne;

Use Pinned – kada uključena, ova opcija čini da editor zadrži materijal ili teksturu čak i kada korisnik odabere drugi objekt.

Sl. 3.36. Glavni izbornik Node Editor-a

Riječ *node* je engleskog podrijetla i znači čvor, pa će u daljnjem tekstu ovog rada biti korištena riječ čvor zbog lakšeg opisivanja i objašnjenja. Čvorovi predstavljaju vizualni izraz matematičkih operacija. Složene operacije su podijeljene na jednostavne čvorove te se upravo tu očituje olakšano rješavanje koje korisniku daje slobodu i fleksibilnost. Čvorovi služe za izgradnju materijala i za postavke modela ili scene nešto prirodnijim putem. Umjesto liste postavki za određeni materijal, uz pomoć čvorova možemo vizualizirati na koji način su te postavke upotrijebljene, kako se neke informacije mogu ponovno upotrijebiti te koji procesi će se dogoditi za vrijednosti ulaza koje se unose u čvorove. Čvorovi se međusobno vežu, a te veze među njima znače da se informacija prenosi sa jednog na drugi. Iako svaki čvor ima svoju specifičnu ulogu i posao, kombinirati se mogu na stotine načina te tako pružaju veliki broj različitih rezultata. Dijelovi čvorova su:

Title

Prikazuje naziv/vrstu čvora.

Sockets

Priključnice za ulazne i izlazne vrijednosti čvorova. To su mali obojani krugovi s obje strane čvora. Priključnice koje se ne koriste se mogu sakriti sa naredbom „CTRL+H“. Žuti krug označava da se treba unijeti informacija o boji kao ulaz ili će ona biti izlaz iz čvora. Sivi krug označava informacije o numeričkim vrijednostima dok plavi krug označava informacije o

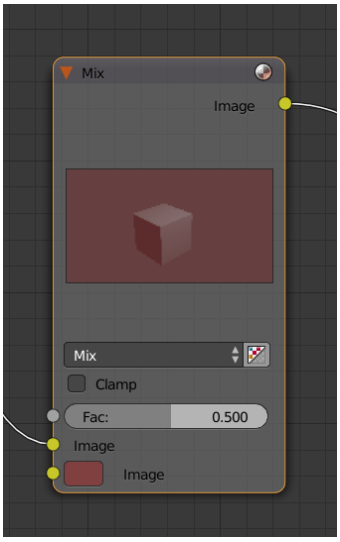
vektorima, koordinatama i normalama. Zeleni krug se koristi za *shader*-e u Cycles-u. Ulazi (engl. *Inputs*) se nalaze na lijevoj donjoj strani čvora i oni pružaju čvoru potrebne podatke za izvršenje funkcije. Svaka ulazna priključnica, osim zelene za *shader*-e, kada nije spojena ima početne zadane vrijednosti koje se mogu urediti preko sučelja numeričkog, vektorskog ili ulaza za boju. Izlazi (engl. *Outputs*) se nalaze na desnoj gornjoj strani. Oni se dalje mogu povezati na ulaze sljedećih čvorova u nizu čvorova koji se također naziva i *node tree* (u daljnjem tekstu stablo čvorova).

Properties

Mnogi čvorovi imaju postavke koje utječu na način komunikacije ulaza i izlaza. Ove postavke se nalaze u sredini čvora, dakle ispod izlaza i iznad ulaza.

Preview

Neki čvorovi prikazuju pretpregled izlaznih podataka za određene naredbe. Pregled može biti promijenjen uz pomoć ikone u desnom gornjem kutu čvora, odmah pored naslova čvora.



Sl. 3.37. Izgled čvora

Mrežom čvorova su definirani materijali, svjetla i pozadine i ona za *output* daje vrijednosti, vektore, boje i *shader*-e. Postoji nekoliko vrsta *shader*-a koji su dostupni kao čvorovi:

- BSDF *shader* (opisuje refrakciju, refleksiju i apsorpciju svjetla na površini objekta);
- *Emission shader* (opisuje emisiju ili zračenje svjetla na površini objekta ili u volumenu);
- *Volume shader* (opisuje raspršenost svjetla unutar volumena);
- *Backround shader* (opisuje zračenje svjetlosti iz okoline).

Postoje takozvane grupe čvorova (engl. *Node Groups*) koje pojednostavljuju stablo čvorova dopuštajući geometrijsku instanciju i skrivanje pojedinih dijelova stabla. Geometrijska instancija je renderiranje u istom trenutku više kopija iste mreže jednog objekta. Jednostavnije rečeno, grupiranje čvorova dopušta da određeni skup čvorova bude tretiran kao samo jedan čvor. Takva grupa čvorova može ponovno biti upotrijebljena unutar istog projekta (tada se naziva *NodeGroups*) ili u drugim datotekama (tada se prilikom dodavanja naziva *NodeTrees*). Grupa čvorova se pravi na sljedeći način: izbornik *Group – Make Group*, zatim prečac „CTRL-G“. Imati će zelenu naslovnu traku i unutar nje će se nalaziti svi odabrani čvorovi. Obrnuti postupak od ovog bi bio razgrupiranje (engl. *Ungroup*): izbornik *Group – Ungroup*, prečac „ALT-G“.

Unutar *Node Editor*-a postoji koristan alat za organiziranje čvorova koji okuplja povezane čvorove u jedan zajednički prostor i on se naziva *Frame Node* (engl. *frame* – okvir). Ovi okviri su korisni kada postoji puno čvorova koji su nepregledno i zbunjujuće poredani i mogu se koristiti samo za grupe čvorova koje se neće ponovno upotrebljavati. Čvorovi se pridružuju u okvir na sljedeći način: izbornik *Node – Join in new Frame*, prečac „CTRL-J“. Ukoliko se želi dodati novi čvor već postojećem okviru potrebno je označiti željene čvorove i kliknuti prečac „CTRL-P“. Ukloniti okvir se može uz pomoć izbornika *Node – Remove from Frame*, ili označiti čvorove koji se žele ukloniti i kliknuti na prečac „ALT-P“.

Za organizaciju se također koristi i čvor zvan *Reroute Node* (engl. *reroute* – usmjeriti). Čvor se ponaša i izgleda slično kao priključnica na druge čvorove u kojoj podržava jedan ulaznu konekciju, a dozvoljava višestruke izlazne konekcije. Dodaje se na već postojanu konekciju zadržanim pritiskom „SHIFT“ te lijevim klikom miša kojim obuhvaćamo vezu.

4. PRAKTIČNI DIO

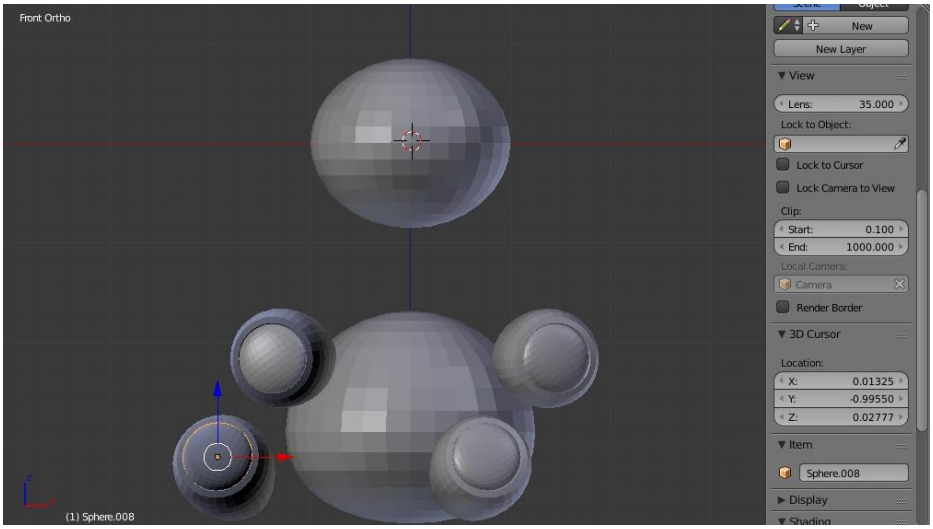
U ovom dijelu rada su prikazani opisani *shader*-i i njihova praktična primjena. U prvom primjeru korišteni su *shader*-i *Diffuse, Glossy, Emission* i *Hair*. S obzirom da je prva verzija plišanog medvjedića izrađena na osobnom prijenosnom računalu slabe snage i brzine, bilo je nemoguće u Blenderu izvršiti sve željene akcije. Za drugu verziju ovog primjera, ali i za sljedeća dva primjera u nastavku, korišteno je računalo Fakulteta elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. Ovo računalo posjeduje jače i bolje komponente koje omogućavaju lakši i brži rad u Blenderu, a samim time i bolje mogućnosti izrade fotorealističnih slika.

Za drugi primjer korišten je Node Editor kako bi se pokazala i takva vrsta manipulacije *shader*-ima. *Shader*-i korišteni u ovom primjeru su: *Diffuse, Emission, Glass, Glossy, Subsurface Scattering, Transparent, Volume Absorption.* Prikazana je staklena čaša s pivom. Ovaj primjer je dosta zahtjevniji od prvog, zahtjeva veći nivo znanja o Blenderu, više vremena te računalo veće brzine i snage.

Treći primjer koristi *Velvet shader* za prikaz odjeće, u ovom slučaju ručnika obješenog na zid. Osim *Velvet shader*-a korišteni su i *Diffuse, Hair* i *Translucent.*

4.1. Primjer: Plišani medvjedić (prva verzija)

Na slici 4.1. i 4.2. se može vidjeti geometrijski dio rada u Blenderu potreban za fotorealistični prikaz plišanog medvjedića. Svaki dio geometrije je izrađen posebno, a zatim spojen u cjelinu koja izgledom podsjeća na medvjedića. Ta cjelina je potom razbijena na manje dijelove: oči, njuška, tijelo, šape. Razlog tome je što se svaki od tih dijelova i kod stvarnog modela razlikuje po makar jednoj karakteristici u odnosu na druge dijelove, a u Blenderu se svakom od tih dijelova dodjeljuje različit *shader*.



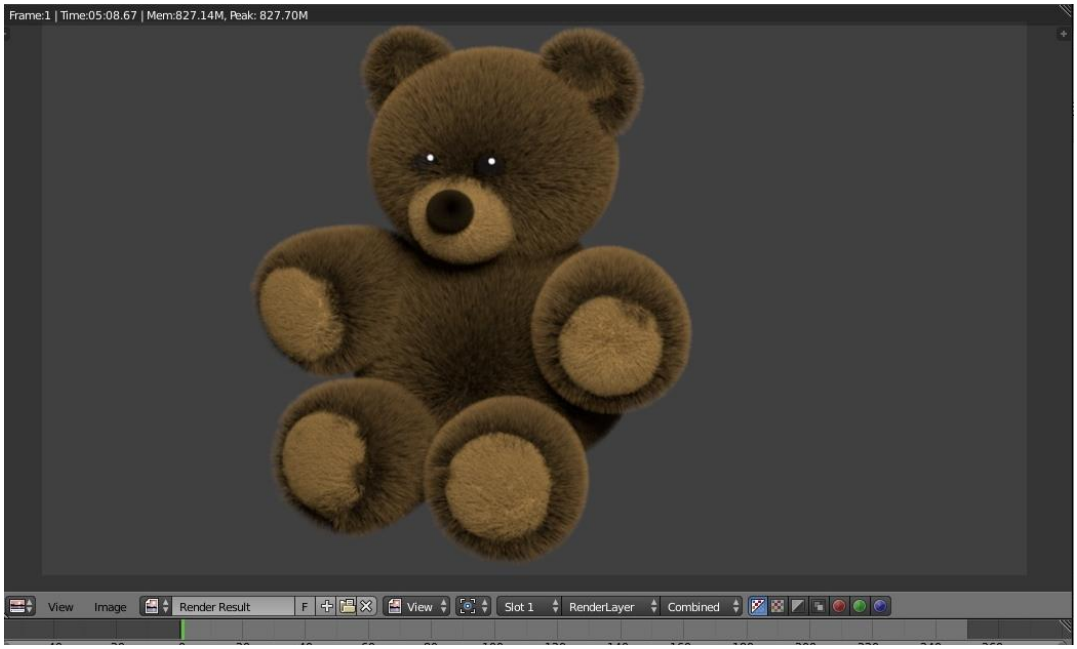
Sl.4.1. Geometrijski prikaz dijelova medvjedića

Sl. 4.2. Odabir i dodjela shader-a odgovarajućim dijelovima medvjedića

Oči medvjedića su napravljene kao kombinacija dva *shader*-a: *Diffuse* i G*lossy*. *Diffuse shader*-u je pridodana crna boja, dok je za *Glossy* izabrana siva. Na ovaj način medvjedić će imati crne oči sa sivim odsjajem. Za njušku, tijelo i šape medvjedića je korišten klasični *Diffuse shader*, u boji krzna medvjedića (po želji je odabrana tamno smeđa boja za tijelo i vrh njuške, svijetlo smeđa za njušku i šape).

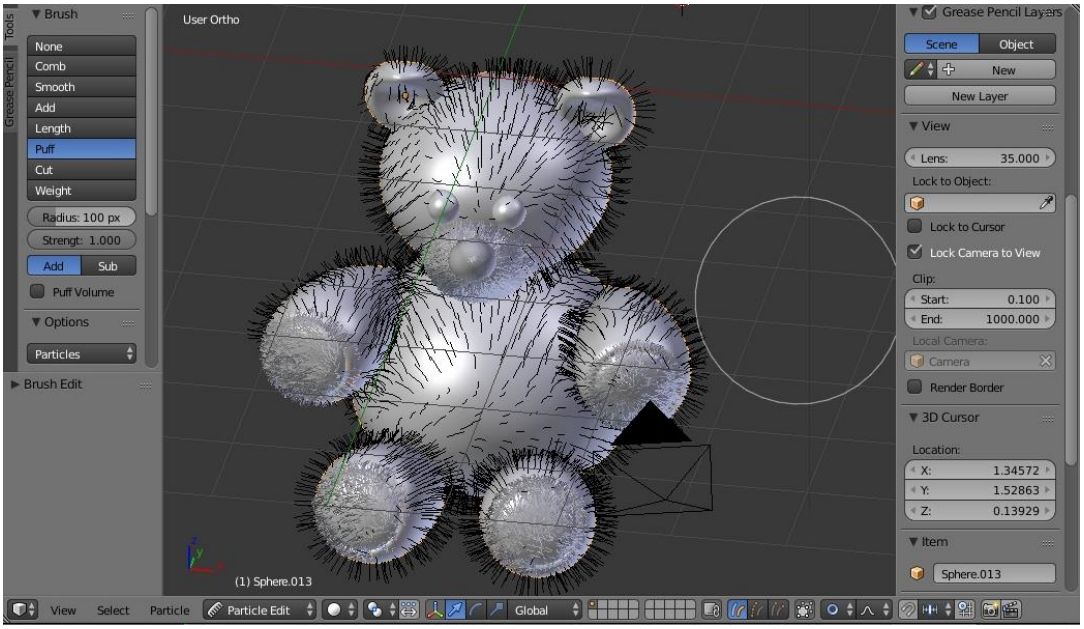
Sl. 4.3. Postavke Hair shader-a

Preostalo je dodati medvjediću ono što ga i čini plišanom igračkom, a to je krzno. Za fotorealistični prikaz krzna koristi se *Hair shader*. On koristi boju površine na koju se dodaje, no to ne mora nužno biti tako. U naredbi *Color* se izabere opcija *ColorRamp* gdje se može podesiti boja korijena i vrhova dlake krzna. U ovom slučaju za boju su također izabrane tamnosmeđa i svijetlosmeđa. U istom prozoru se klikom na naredbu *Fac – Hair info – Intercept* osigura da boje ravnomjerno prelaze jedna u drugu od početka (korijena) do kraja (vrha). Sljedeći korak je u prozoru *Particle* dodati novi *Particle System*, vrstu postaviti na *Hair*. Vrlo je važno u ovom dijelu dobro procijeniti i podesiti postavke kako bi se što realističnije prikazao objekt, a to podrazumijeva mijenjanje broja dlaka i dužine dlaka. Unutar *Particle System*, u opciji *Children*, dodaju se dječje dlake (engl. *Children Hair*) koje za rezultat imaju gušće krzno prepuno manjih dlačica koje upotpunjavaju realistično izgled.



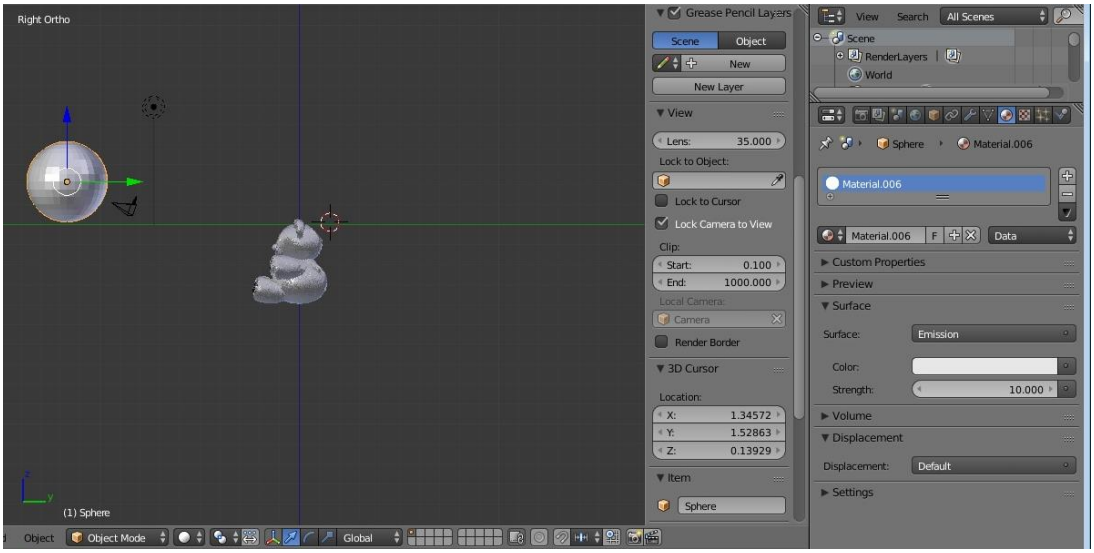
Sl. 4.4. Renderirani prikaz medvjedića

Rezultat, odnosno renderirana slika medvjedića je prikazana na slici 4.4., ali fotorealizam i dalje nije na zadovoljavajućoj razini.



Sl. 4.5. Prikaz korištenja naredbi Comb i Puff

Kako bi se on doveo na zadovoljavajuću razinu, koriste se naredbe *Comb* i *Puff* u *Particle Edit* načinu rada (sl. 4.5.). *Comb* (hrv. češalj) češlja dlake u smjeru pomicanja miša, dok *Puff* daje volumen dlakama te ih na taj način tako „počešljane“ izdiže.



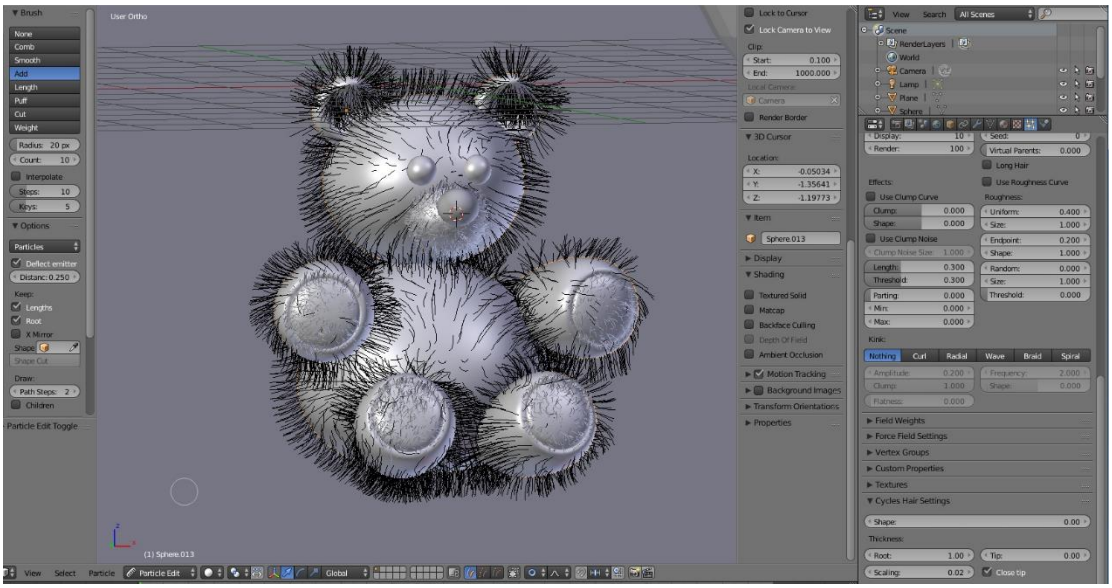
Sl. 4.6. Dodavanje izvora svjetlosti za prikaz odsjaja u očima

Kako bi se u očima medvjedića stvorio sjajni odsjaj koji ne izgleda lažno, na scenu nasuprot medvjediću je dodano geometrijsko tijelo kugla (sl. 4.6.). Odsjaj se stvara na način da se kugli pridodijeli *Emission shader* koji isijava svjetlost. Također je dodana i podloga i pozadina za koje se koristio *Diffuse shader*. Rezultat je prikazan na slici 4.7.

Sl. 4.7. Rezultat renderiranja na osobnom prijenosnom računalu

4.2. Primjer: Plišani medvjedić (druga verzija)

U drugoj verziji je maknuta kosa koja je ulazila u oči medvjediću na način da je prvo skraćena naredbom *Length – Shrink* te je počešljana naredbom *Comb* u smjeru suprotnom od položaja očiju (Sl. 4.8.). Dopunjena je kosa u području ušiju sa naredbom *Add* u *Particle Edit* načinu rada za *Hair shader*. Na temelju prve renderirane slike dorađeni su detalji u smislu dužine, gustoće i boje krzna, na isti način kao što je objašnjeno u prvom dijelu.



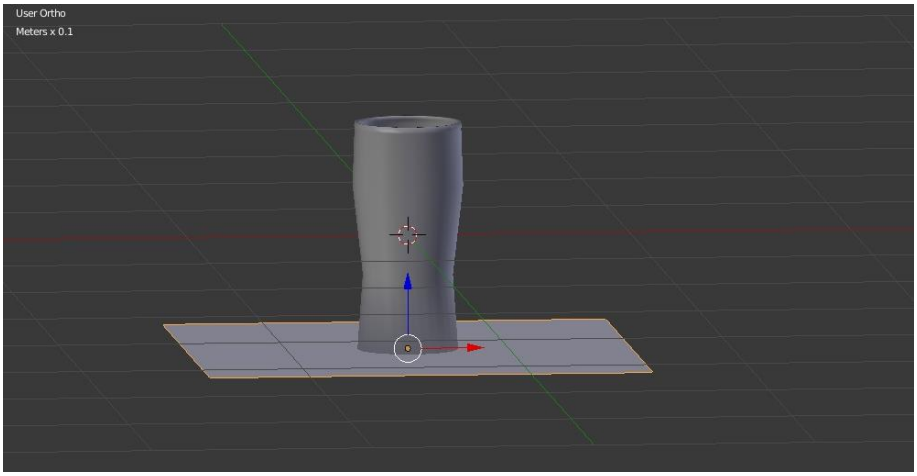
Sl. 4.8. „Češljanje“ medvjedića

Na kraju je izvršeno finalno renderiranje, a konačni rezultati je prikazan na slici 4.9..

Sl. 4.9. Krajni rezultat fotorealističnog prikaza medvjedića

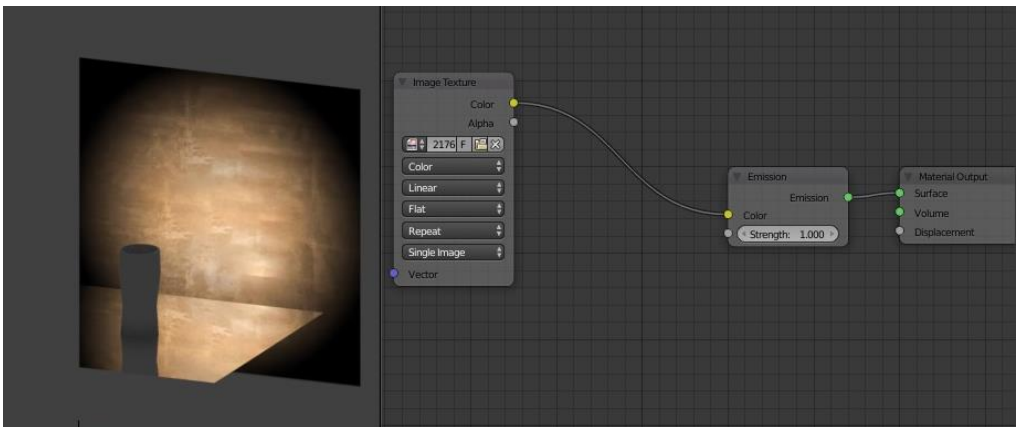
4.3. Primjer: Čaša s pivom

Osnovna geometrija potrebna za izradu staklene čaše ispunjene pivom prikazana je na slici 4.10., a sastoji se od cilindra koji je oblikovan u čašu te podloge koja je vrlo važan dio za fotorealistični prikaz. Za podlogu je odabran *Glossy shader* koji, kada je u opciji *Color* postavljena vrijednost bijele boje na „1“, podlogu čini čistim ogledalom.



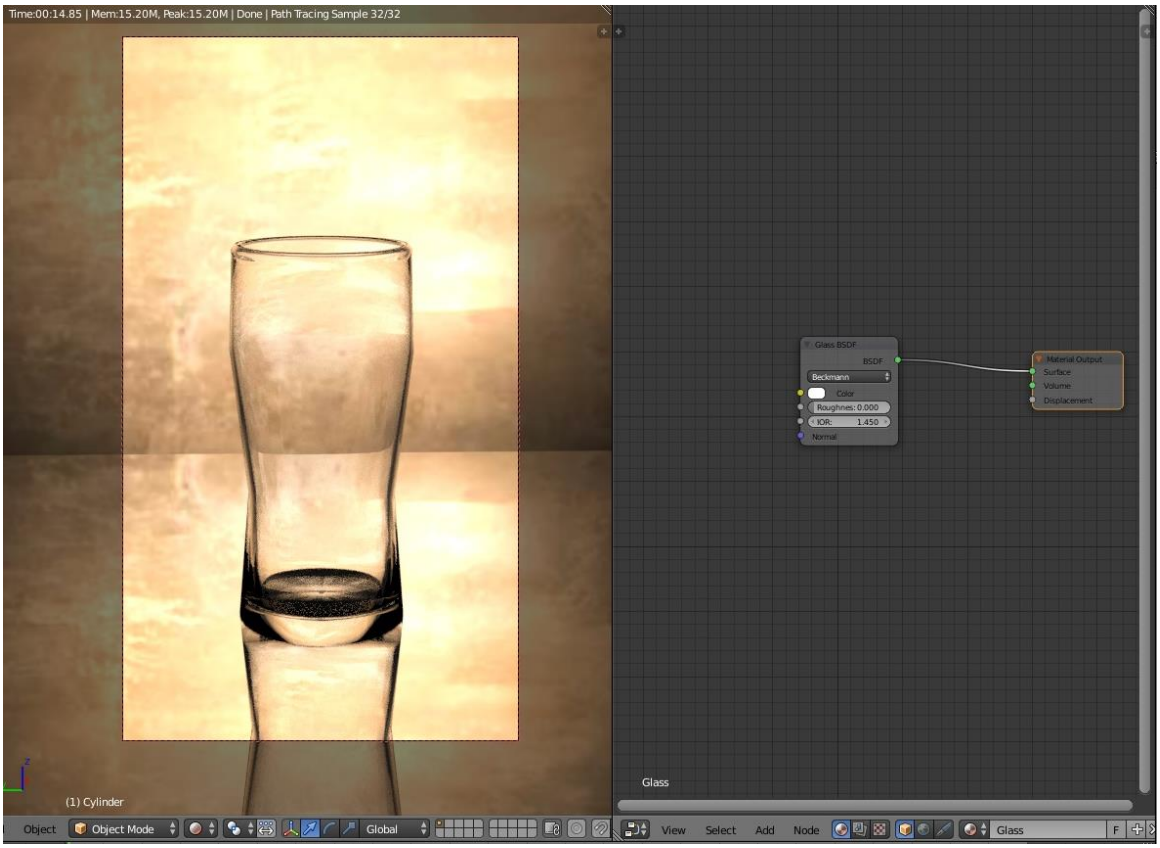
Sl. 4.10. Geometrijski prikaz čaše i podloge

U nastavku ovog primjera, umjesto korištenja *shader*-a klasično u desnom prozoru Blendera, korišten je *Node Editor*. Za pozadinu je odabrana slika po izboru, postavljena kao *Image Texture* na površinu okomitu na položaj čaše. Odabrana je neutralna pozadina crnog obruba sa zlatnom „kuglom“ u središtu. *Shader* korišten za ovu pozadinu je *Emission*, a njegova uloga je da osvjetljava kako čašu, tako i podlogu na kojoj se ona nalazi. Kao rezultat se dobije podloga u kojoj se zrcali izgled pozadine (sl. 4.11.)



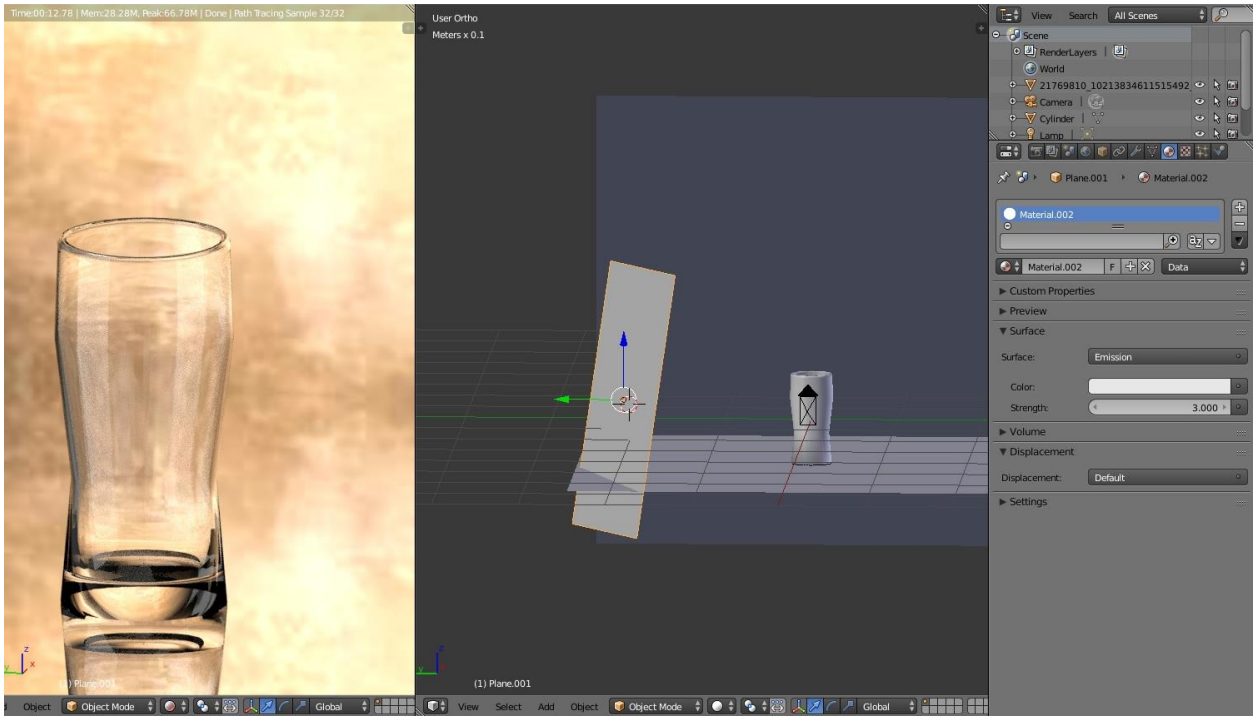
Sl. 4.11. Node editor za pozadinu

Za modeliranje same čaše je korišten *Glass shader*. U *Node Editoru* (sl. 4.12.) je u opciji *Color* postavljena vrijednost bijele boje na „1“. Ovom postavkom se dobije čisto prozirno staklo koje je i potrebno za prikaz stakla čaše.

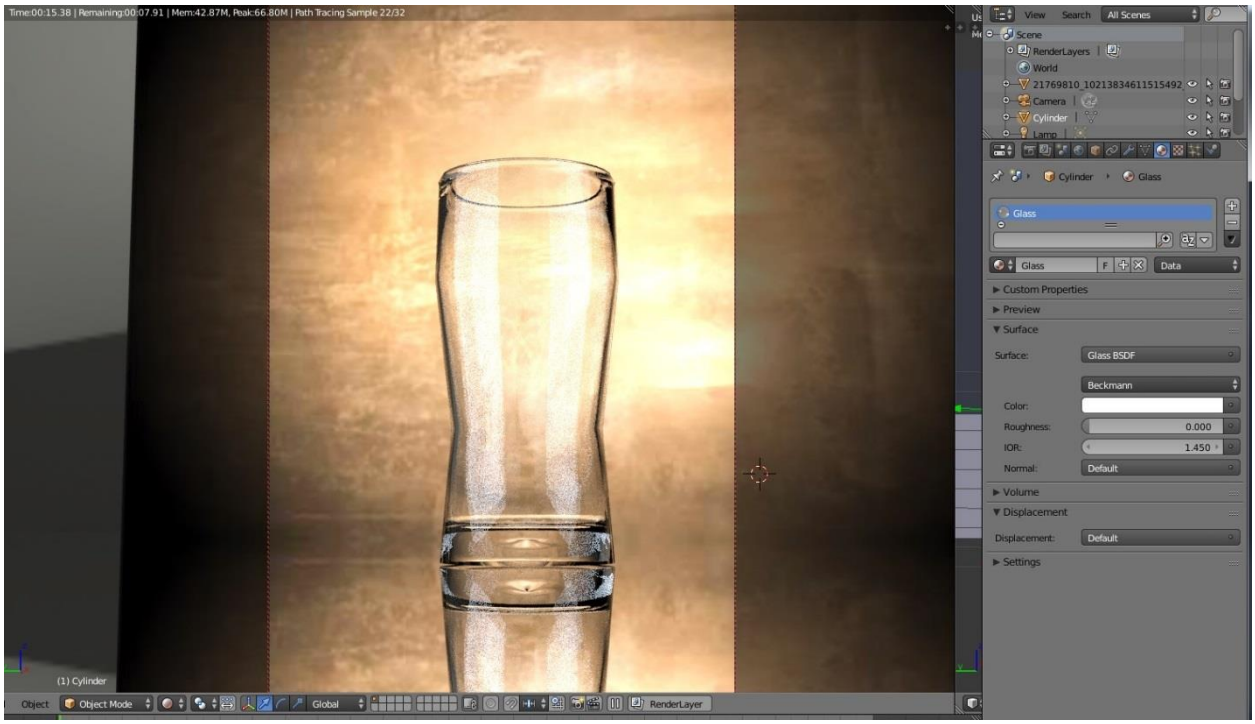


Sl. 4.12. Node editor za čašu

Kada je staklo napravljeno, potrebno mu je dodati detalje koji će ga činiti realističnim. Za efekt odsjaja na čaši korišteni su pomoćni objekti, ploče sa *Emission shader*-om, postavljeni okomito ili pod oštrim kutom u odnosu na čašu (sl. 4.13.). U postavkama *Emission shader*-a u opciji *Color* boja je promijenjena na svijetlo plavu umjesto bijelu, kako bi odsjaj ne bi izgledao suviše bijelo, odnosno neprirodno. Za dodatni efekt je u prozoru *World* u desnom kutu Blendera podešena opcija *Surface – Background - Environment Texture* te je odabrana slika prikaza lažnog studija sa osvjetljenjima u obliku reflektora. Rezultat dodavanja ovih osvjetljenja je vidljiv na slici 4.14..

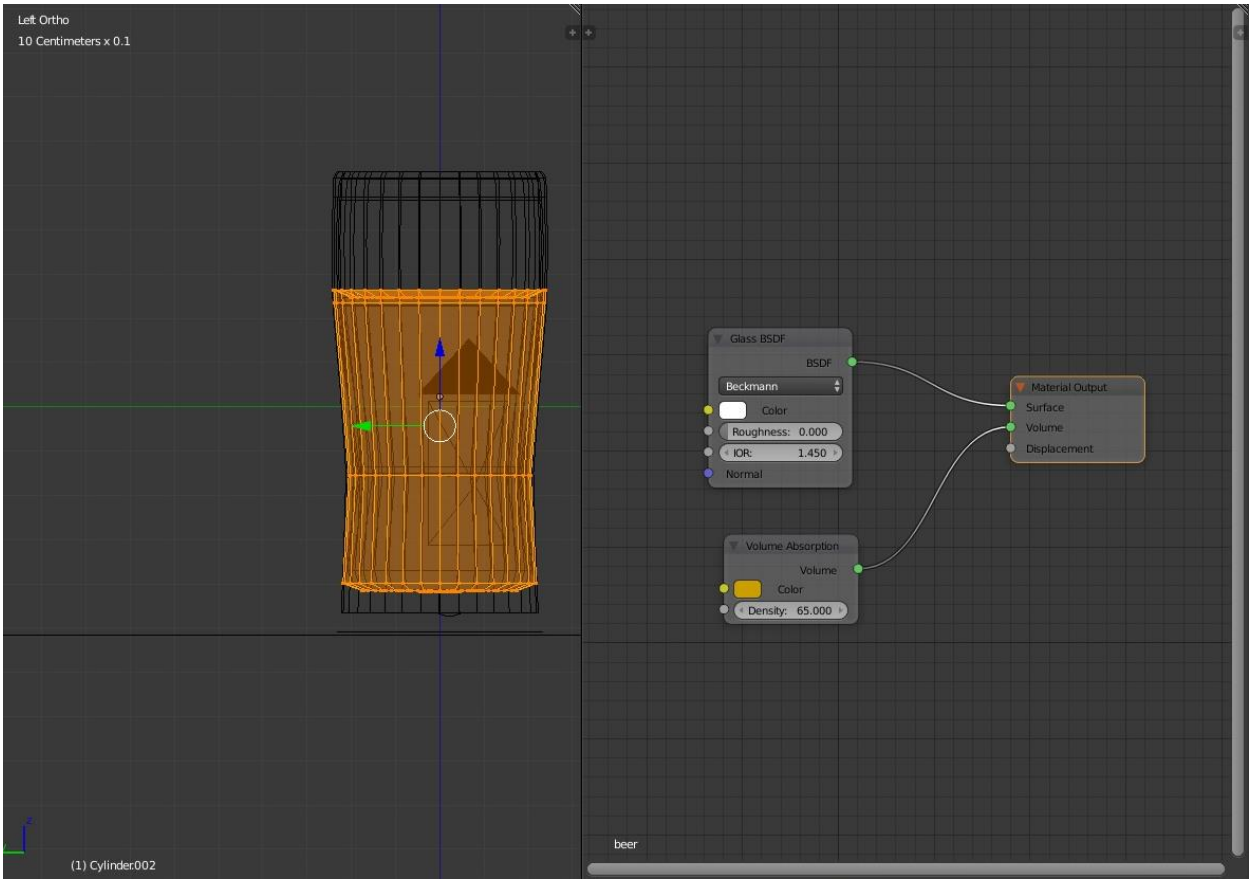


Sl. 4.13. Dodavanje dodatnog osvjeljenja čaši

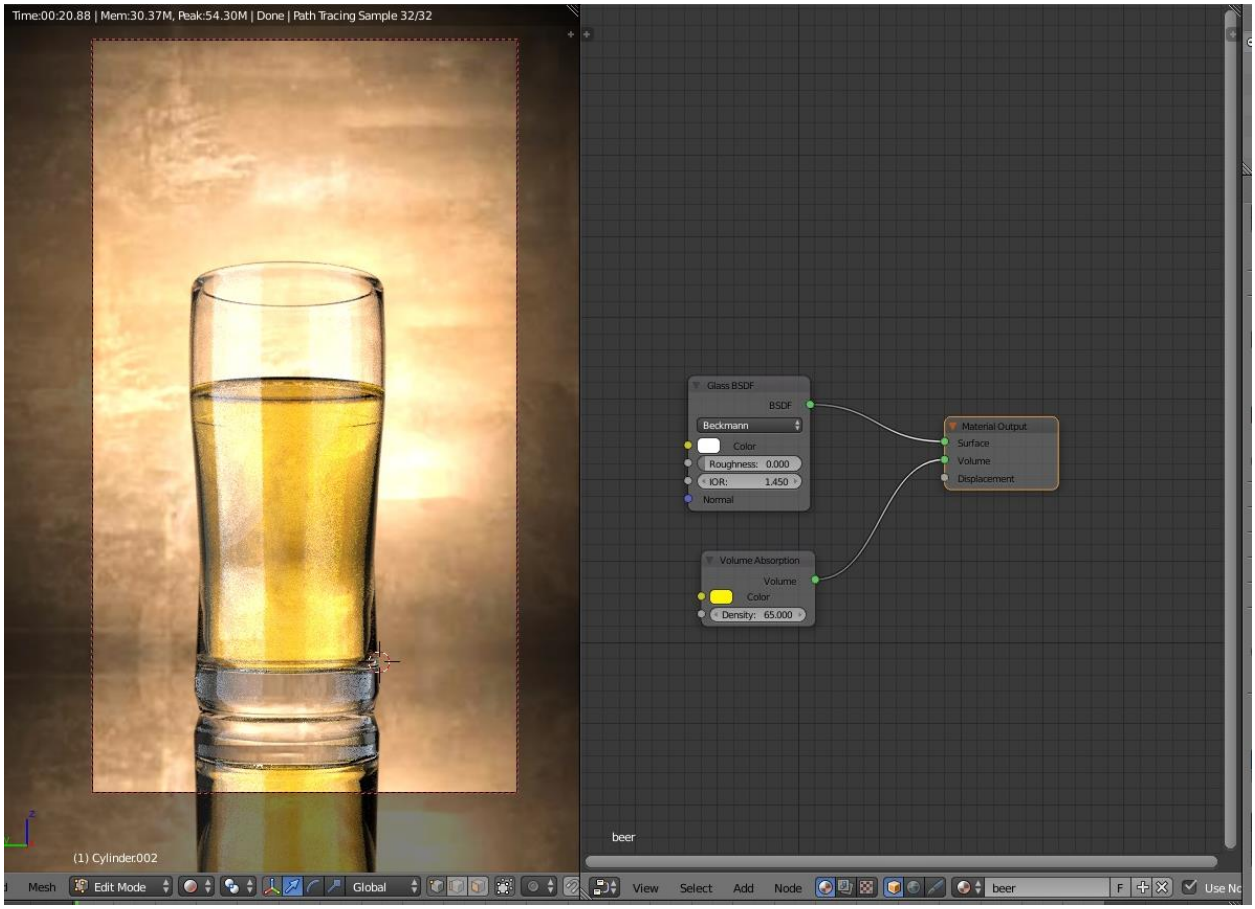


Sl. 4.14. Prikaz sjena na čaši uzrokovan raznim osvjetljenjima na sceni

Kada su čaša, pozadina i osvjetljena stvorena i postavljena, slijedi korak u kojem se stvara tekućina. Na slici 4.15. se može vidjeti i geometrijski oblik tekućine za koji je vrlo važno napomenuti da ne smije izlaziti izvan granica čaše, nego se vrhovi moraju nalaziti točno unutar stjenke čaše. Tekućina je također rađena sa *Glass shader*-om istim postavkama kao za čašu, no kako bi ona stvarno izgledala kao takva korišten je *Volume Absorption shader*. Ovaj *shader* čini da se svjetlo apsorbira unutar materijala i na taj način stvara iluziju tekućine te on utječe na volumen tekućine, dok *Glass shader* utječe na izgled površine tekućine. U *Node Editor*-u u prozoru *Volume Absorbtion* je boja postavljena na zlatno-narančastu obzirom da je pivo tekućina koja će biti prikazana. Opcije *Density* i *Saturation* u ovom prozoru nude mogućnost pojačavanja gustoće i zasićenosti boje, što jednom riječju možemo nazvati jačina boje (nema velike razlike između ove dvije opcije, postavkom jedne postavljena je i druga).

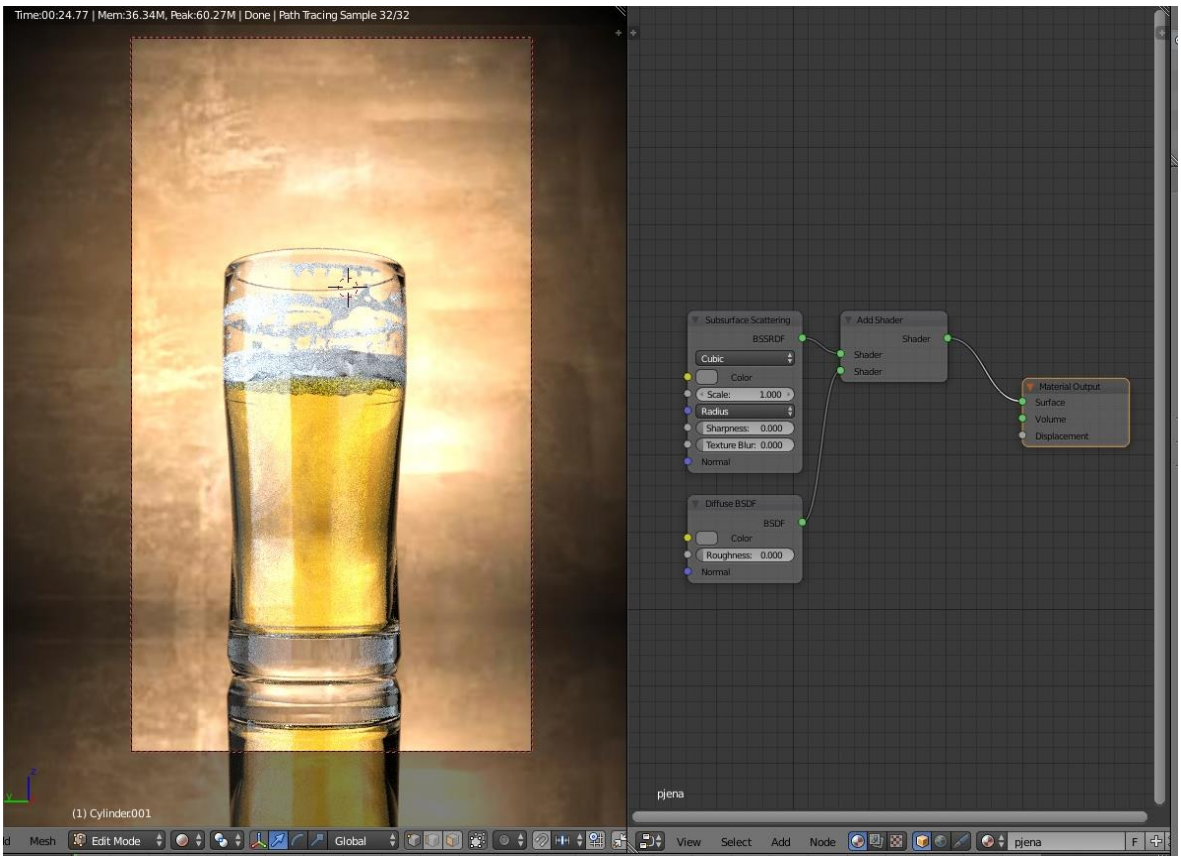


Sl. 4.15. Geometrijski prikaz tekućine (lijevo) i Node editor za tekućinu (desno)



Sl. 4.16. Node editor za tekućinu (desno) i rezultat (lijevo)

Kreiranje pjene, koja se gotovo uvijek u stvarnom životu nalazi na vrhu piva i na stjenkama čaše, je možda i najzahtjevniji dio ovog procesa, ali je svakako ono što pivo čini pivom i pri tome je nemoguće izbaciti ili preskočiti ovaj dio modeliranja. Geometrijsko oblikovanje neće biti objašnjeno zbog prevelike količine informacija ali i zato što su veličina i oblik pjene proizvoljni i kreiraju se po osobnim željama. Naglasak se postavlja na pravilan odabir shader-a kako bi pjena izgledala što realističnije, a taj odabir uključuje kombinaciju *Diffuse shader-a* koji će pjeni dati bijelu boju i *Subsurface Scattering shader*-a koji čini da svjetlost najviše osvjetljava početne dijelove površine pjene i polako blijedi ka unutrašnjosti. Ova dva *shader*-a se povezuju sa *Add shader*-om koji zbraja količinu, odnosno jačinu svjetlosti svakog pojedinačnog *shader*-a, pa je potrebno pripaziti na postavke boje i za jačinu odabrati prikladne vrijednosti u oba čvora kako se ne bi dobila presvijetla pjena nakon renderiranja (sl. 4.17).



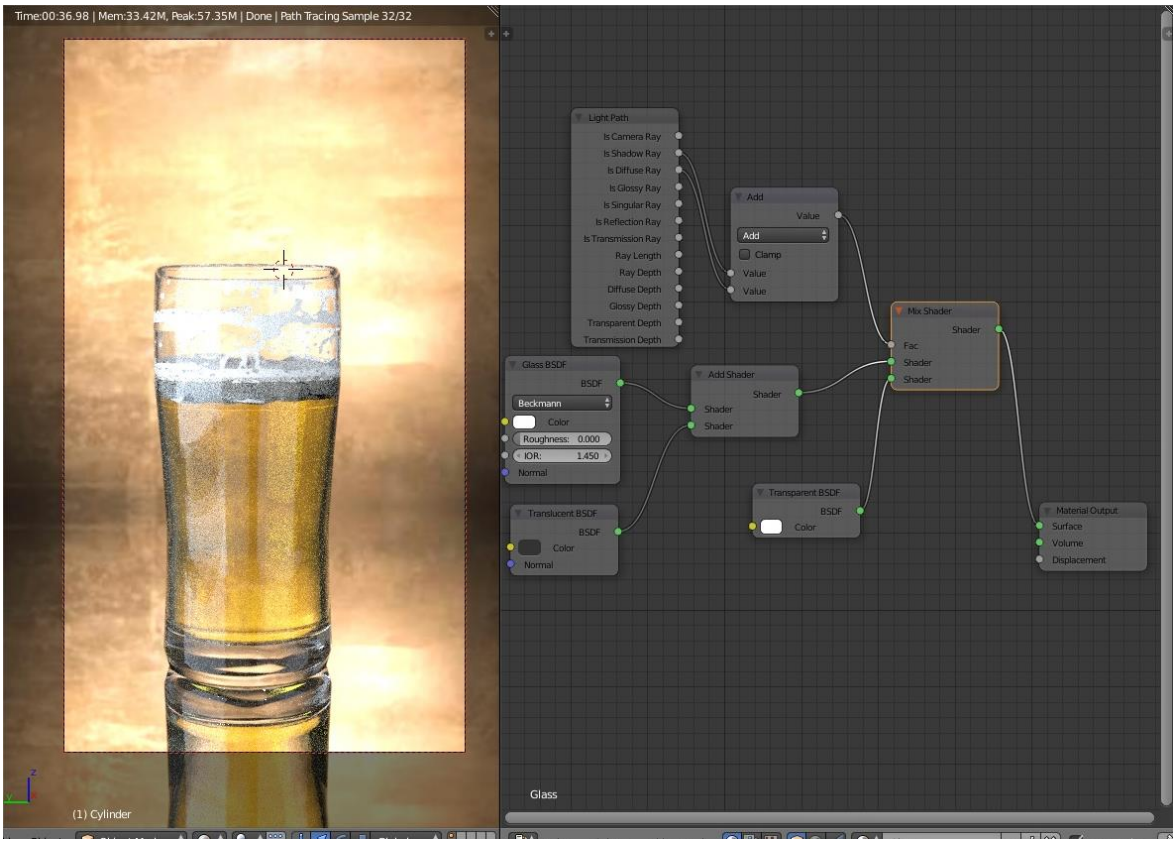
Sl. 4.17. Node editor za pjenu

Kreirati pjenu na stjenkama čaše podrazumijeva nešto drugačiji proces. Unutrašnjost gornjeg dijela čaše koji se nalazi iznad pjene se pretvori u zaseban geometrijski oblik i na njemu se izvrši teksturiranje uz pomoć slike prikazane na sl.4.18..

Sl. 4.18. Slika korištena za prikaz pjene na stjenkama čaše

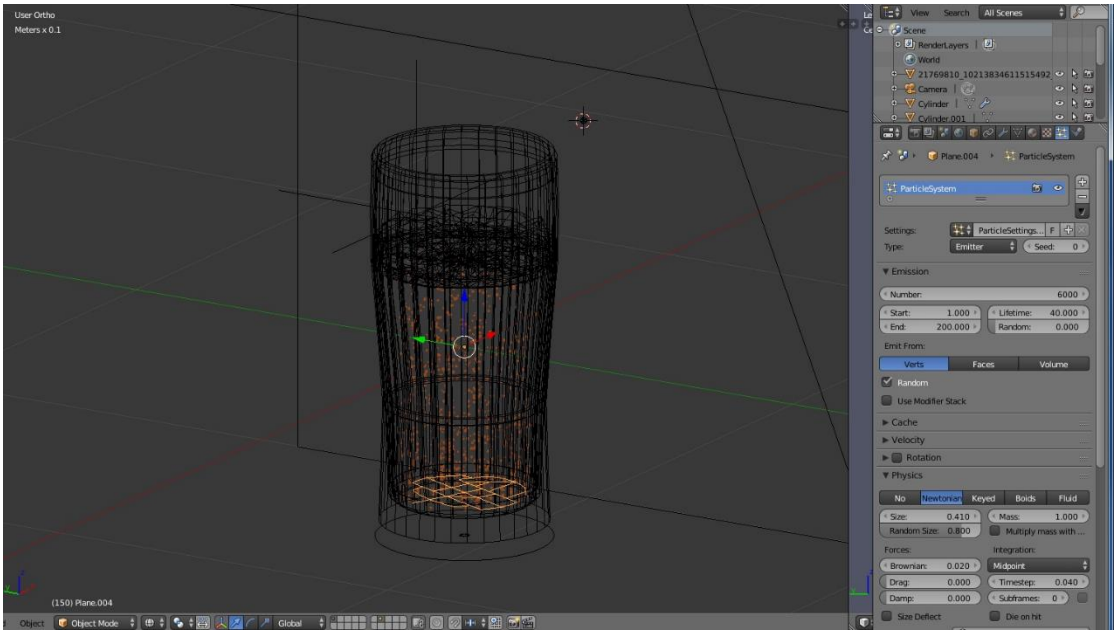
Sada je potrebno napraviti da svi dijelovi tog geometrijskog oblika budu nevidljivi osim teksture (pjene). U Node Editor-u je dodan *Transparent shader* koji će u kombinaciji sa *Diffuse shader-*

om taj efekt i postići. Uz ova dva, u *Mix shader* je potrebno spojiti i čvor *Image Texture*. S obzirom da svjetlost prolazi kroz čašu i na svojem putu stvara sjene, kada stigne na unutarnje stjenke čaše gdje se nalazi tekstura koja nije čisto staklo, stvorit će se sjene i taj dio čaše će izgledati pomalo zamućeno. Iz ovog razloga ponovno se uređuje struktura čaše. Uz *Glass shader*, koji je već prisutan, dodaje se *Transparent shader*. Zatim se dodaje novi input *Light Path* koji uz pomoć *Math* čvora kombinira *Diffuse* i *Shadow* zraku, odnosno govori programu da: ako je *shadow* (hrv. sjena), tretiraj to kao *transparent* (hrv. prozirno); ako je *diffuse* (u našem slučaju pjena), tretiraj to kao *transparent*; sve ostalo što je vidljivo tretiraj kao *Glass* (hrv. staklo).



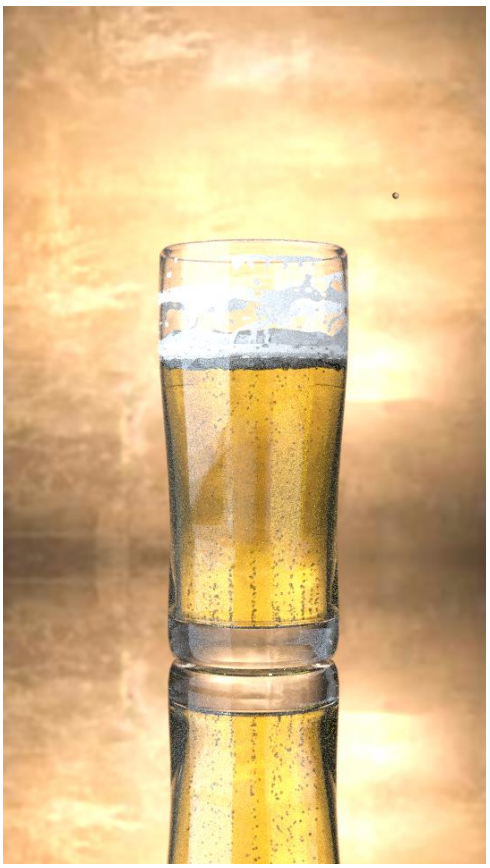
Sl. 4.19. Nadopunjeni Node editor za čašu

Za kraj je ostalo još prikazati mjehuriće. Otvori se novi Particle System, u opciji *Field Weights* se podese postavke gravitacije od kojih ovisi kako brzo i u kojem smjeru će „putovati“. Ostatak postavki je irelevantan jer se ne dotiče direktno teme ovo završnog rada te iz tog razloga neće biti detaljno objašnjene. Postavke se mogu vidjeti na slici 4.20. Ono što se treba spomenuti jeste da su mjehurići napravljeni od sićušnih kuglica na koje je primijenjen *Glass shader*.



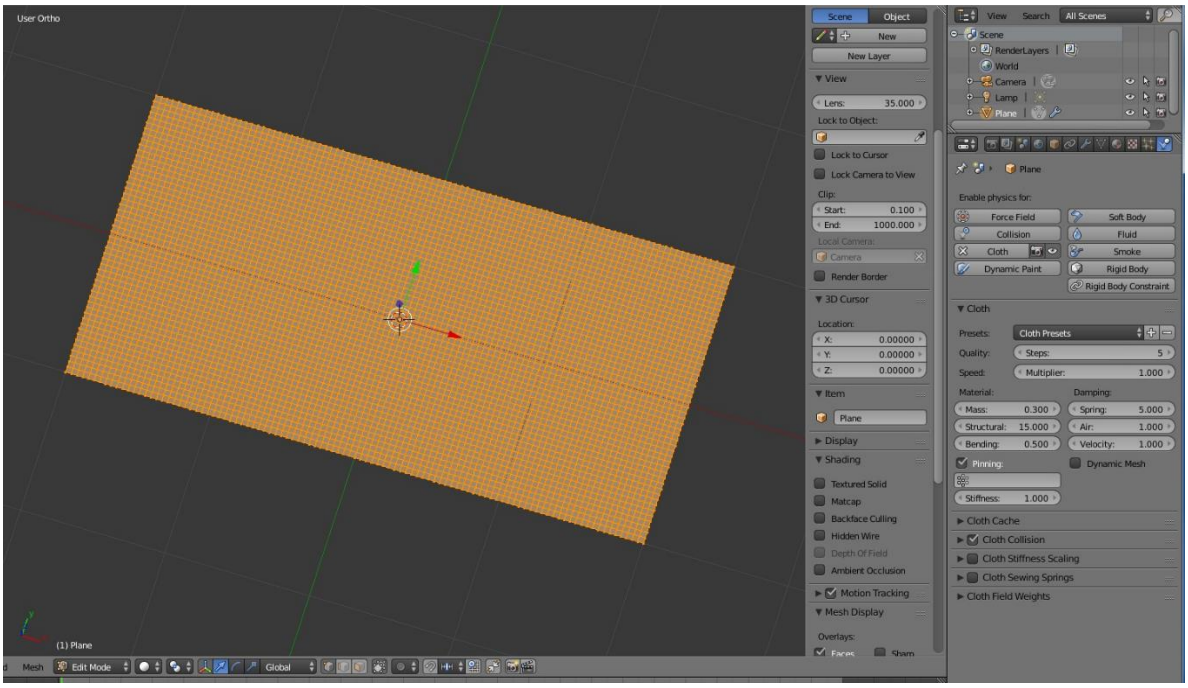
Sl. 4.20. Postupak dodavanj mjehurića

Krajnji rezultat je vidljiv na slici 4.21.



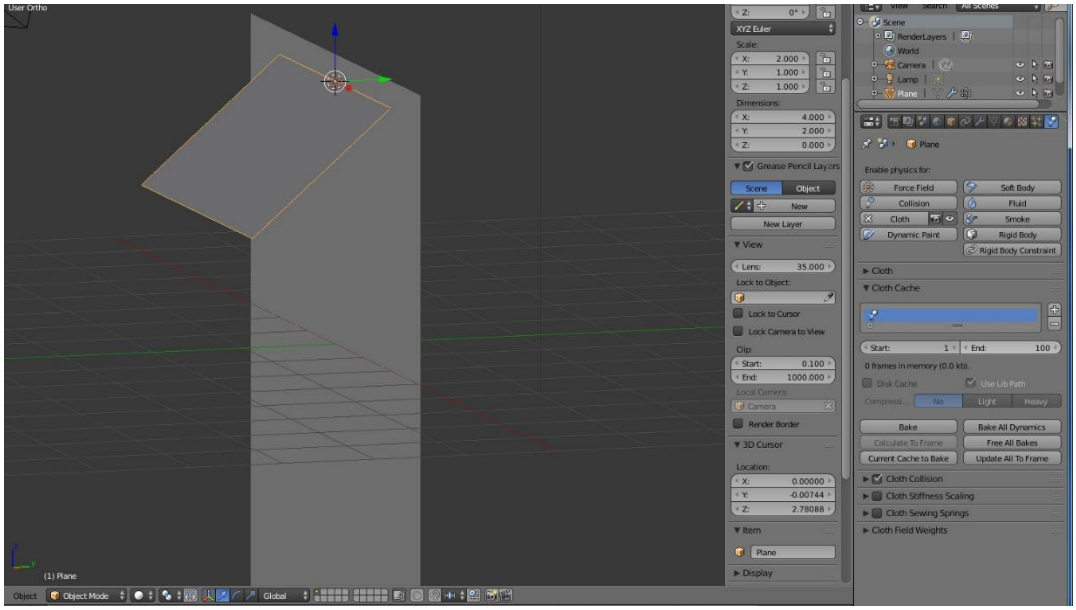
Sl. 4.21. Renderirani prikaz čaše s pivom

4.4. Primjer: Ručnik



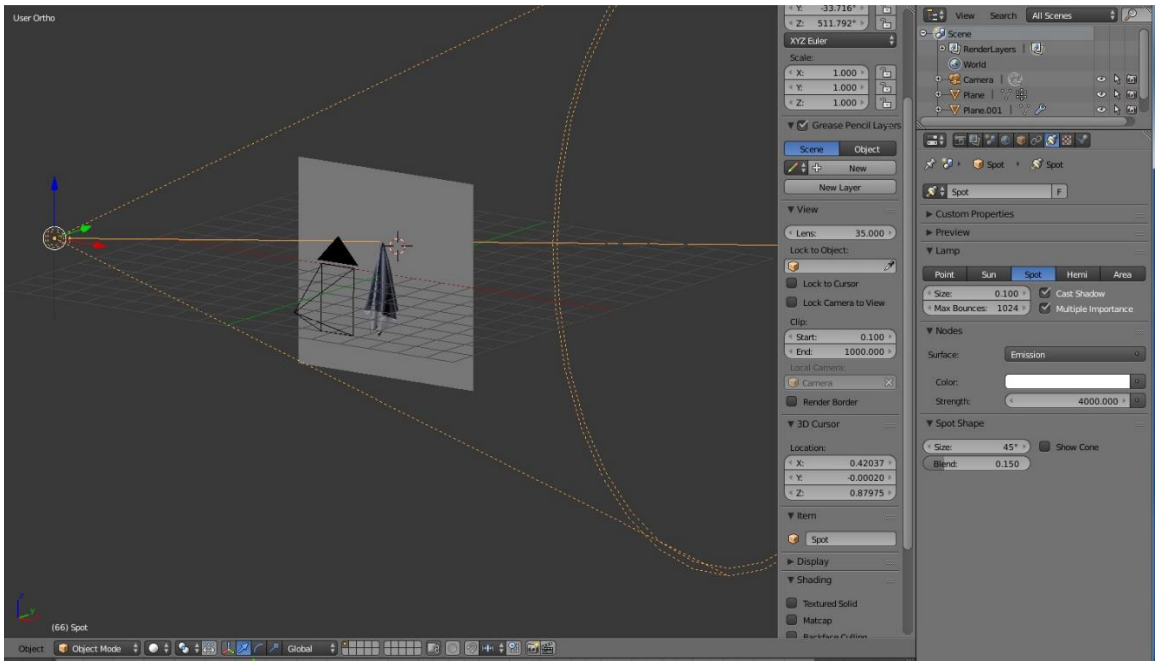
Sl. 4.22. Geometrijski prikaz ručnika

Na slici 4.22. je prikazan geometrijski oblik ručnika, pravokutnik podijeljen na veliki broj sitnih kvadratića. U ovom slučaju je iznimno bitno da su to upravo kvadratići zbog pravilnog oblika kako bi se u kasnijem procesu dobilo ispravno simuliranje koje je u svim smjerovima jednako.
U ovom primjeru koristi se izbornik *Phisycs* koja se nalazi na desnoj strani Blendera, odabrana je opcija *Cloth* (hrv. tkanina) te su postavljene sve potrebne postavke kako bi rezultat simulacije izgledao što realnije stvarnom modelu ručnika. Kao i u prethodnom primjeru i iz istog razloga, te postavke neće biti detaljno opisane.



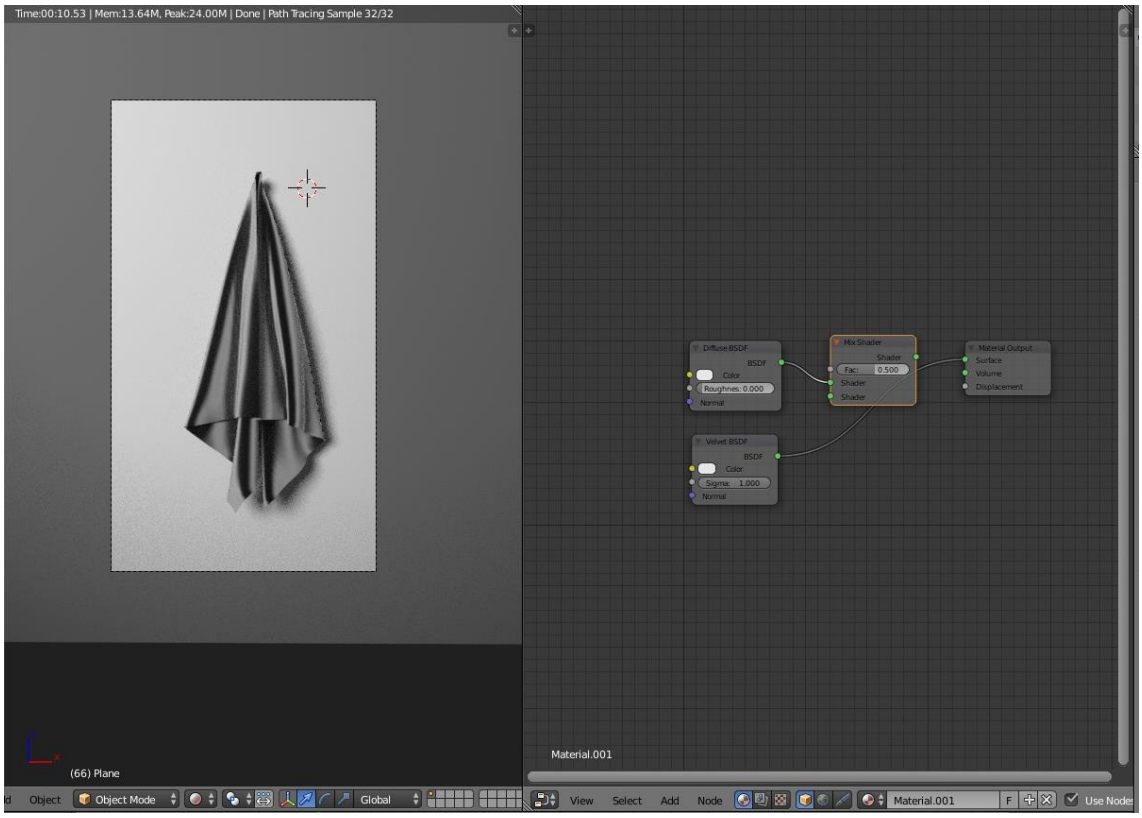
Sl. 4.23. Izgled scene prije simulacije

U sljedećem koraku je na scenu dodan zid (sl. 4.23.) kako bi ručnik nakon simulacije ostao u položaju u kakvom inače visi na zidu i kako bi se osiguralo pravilno osvjetljenje, odnosno sjenčanje ručnika. Osvjetljenje i cjelokupna scena nakon simulacije je vidljiva na slici 4.24.



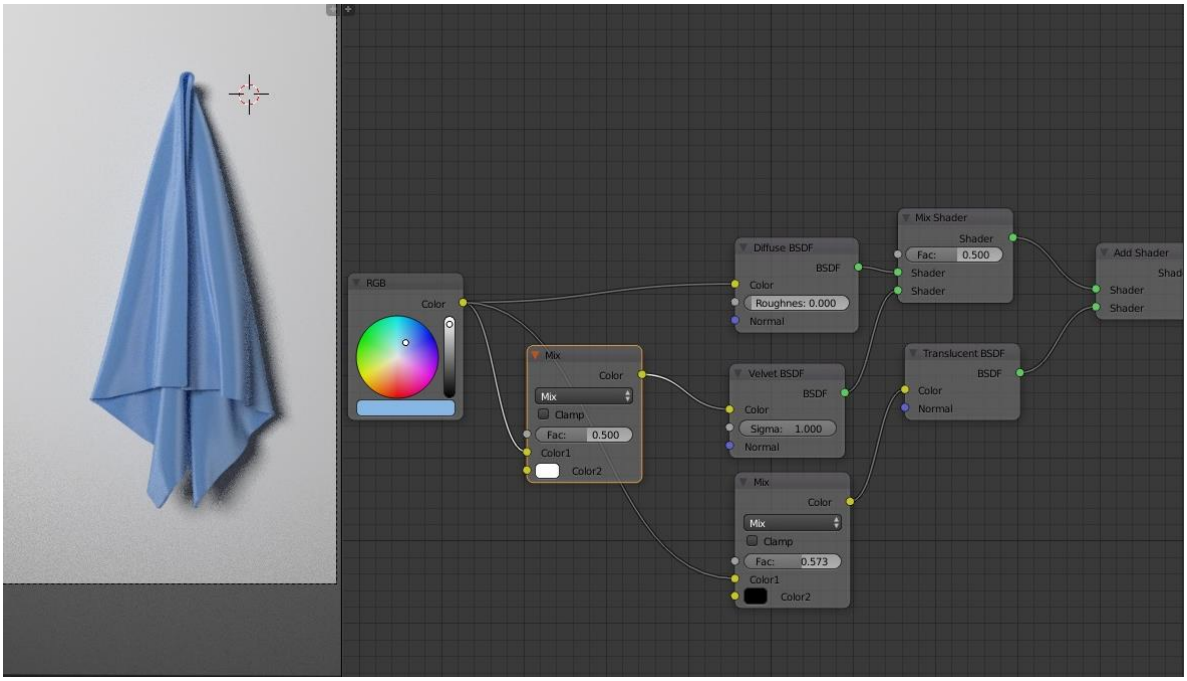
Sl. 4.24. Izgled scene nakon simulacije

Za prikaz tkanina se koristi *Velvet shader* (hrv. baršun). Baršun je mekani materijal proizveden najčešće od svile te iz tog razloga posjeduje specifični odsjaj koji se ublažava kombiniranjem *Velvet shader*-a sa *Diffuse shader*-om. *Diffuse shader* čini da materijal posjeduje određenu boju, dok *Velvet shader* daje površini tog materijala odsjaj na rubovima prigodan za prikaz tkanina.



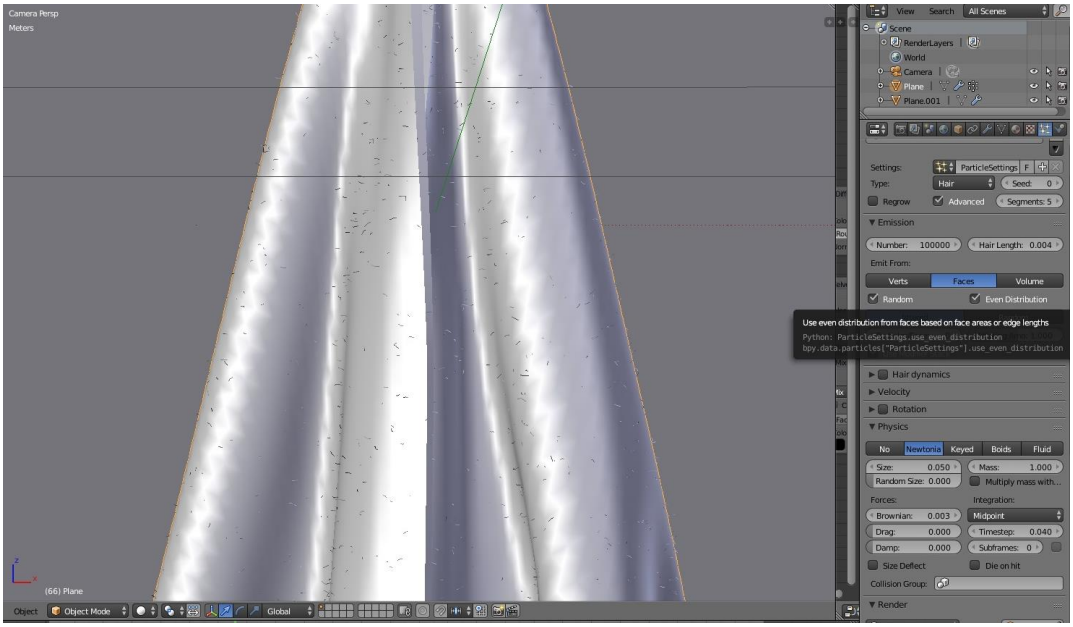
Sl. 4.25. Node editor za ručnik

U nastavku je dodan *Translucent shader* kako bi se postigao prozračni efekt tkanine. *Translucent shader* se u mrežu čvorova dodaje uz pomoć *Add shader*-a te je potrebno pripaziti na jačinu svjetlosti koja nastaje kao rezultat zbrajanja svjetlosti svih čvorova u *Add shader*-u. S obzirom da se radi o većem broju *shader*-a spojenih u jedan, a svaki od njih zahtjeva podešavanje boje, proces dobivanja željene boje bio bi dugotrajan i kompliciran. U svrhu olakšavanja korisnicima, Blender nudi mogućnost odabira *RGB* i *RGB Mix* čvorova za ulaze. *RGB* postaje čvor u kojem se podešava glavna boja modela, dok *RGB Mix* čini da se ta boja miješa sa odabranom bojom, u odabranom omjeru, za odabrani shader. U našem slučaju bijela boja se miješa sa plavom bojom, u mjeru 0.5 : 0.5, ostavljajući efekt na *Velvet shader*-u. Još jedan *RGB Mix* čvor je dodan kako bi se povezao sa *Translucent shader*-om i učinio te dijelove ručnika tamnijim odabirom miješanja plave boje sa crnom (sl. 4.26).



Sl. 4.26. Nadopunjeni Node editor za ručnik

Posljednji korak podrazumijeva dodavanje *Hair shader*-a u *Partycle System* načinu. Sve postavke se namjeste tako da odgovaraju veličinama dlaka tkanine od koje je ručnik napravljen, ne uključujući vrhove ručnika koji su zaobljeni.



Sl. 4.27. Izgled ručnika nakon dodavanja Hair shader-a

Također se isključi *Hair shader* za dvije pravokutne površine s lijeve i desne strane pravokutnika od kojeg je ručnik napravljen, koje su karakteristične za izgled ručnika i bez njih bi ručnik izgledao jednostavno lažno.

Sl. 4.28. Rezultat s pogrešno okrenutim ručnikom (lijevo) i rezultat s ispravno okrenutim ručnikom (desno)

Ovo nije krajnji rezultat ovog primjera jer se potkrala sitna greška u procesu modeliranja. Naime, prilikom postavljanja zida kao potpore, on je postavljen sa pogrešne strane ručnika što je jasno vidljivo iz slike 4.28.. Ručnik je potrebno zarotirati 180° stupnjeva kako bi bio naslonjen s ispravne strane. Nažalost, tako rotirani ručnik ne prikazuje sve detalje koji su izrađeni u procesu modeliranja, no i dalje je zadržan fotorealizam i potvrđena je činjenica da sve što je primjenjeno na model u jednoj perspektivi je također vidljivo i u svim ostalim. Na izrazito jednostavan i banalan način je potvrđena iznimna snaga i moć 3D modeliranja koje iz dana u dan sve više napreduje i zadržava svoje mjesto u vrhu tehničkog napretka današnjice. Krajnji rezultat je prikazan na slici 4.28. desno.

5. ZAKLJUČAK

U ovom završnom radu predstavljeni su *shader*-i Cycles Enginea u Blenderu. Blender je 3D programski paket koji sadrži mnogobrojne funkcije, a u ovom radu je pažnja posvećena funkciji fotorealističnog prikaza objekata koristeći metodu koja se temelji na praćenju puta zraka svjetlosti kroz scenu (engl. *Ray tracing*). Ukratko su opisani osnovni koncepti 3D računalne grafike potrebni za konstrukciju, manipulaciju i prikazivanje 3D objekta na zaslon računala te najpoznatiji alati za 3D modeliranje i renderiranje. Nakon toga je opisan sam program Blender sa naglaskom na Cycles Engine i *shader*-e. Za kraj je kroz praktični dio prikazana usvojena teorija i to na tri primjera: plišani medvjedić, staklena čaša s pivom i ručnik. U primjerima je prikazan rad opisanih *shader*-a.

Sama tema rada je poprilično široka i zahtjeva opširno pisanje u smislu obuhvaćanja svih opcija koje Blender i Cycles Engine nude, kako bi se na kraju uspješno opisao rad *shader*-a koji su samo jedna od bezbroj opcija koje ovaj program nudi. Prepreka koja se ovim putem sama od sebe nameće je previše informacija na zadanu temu, a rezultira potragom za važnim informacijama u moru nevažnih i konstantnim postavljenjem pitanja: „Što je bitno, a što je nebitno?“. Osim ovoga pojavila se prepreka u pogledu rada samog programa. Nemogućnost izvršenja pojedinih opcija programa uzrokovana je nedostatkom dovoljno jakog i brzog procesora u osobnom računalu. Problem je riješen na način da se dio praktičnog rada koji se nije mogao izvršiti na osobnom prijenosnom računalu izvršio na računalu u laboratoriju Fakulteta elektrotehnike, računarstva i informacijskih tehnologija u Osijeku.

LITERATURA

[1] K. C., Finney, 3D Game Programming All in One, Premier Press, Boston, MA, 2004.

[2] P., Shirley, S., Marschner, Fundamentals of Computer Graphics, A.K. Peters, Natick, MA, 2009.

[3] Topić, Ivana, Diplomski rad - obrada podataka laserskog skaniranja RapidForm-om, Sveučilište u Zagrebu, Geodetski fakultet, Zagreb, svibanj 2005.

[4] Blender 2.79 manual [online], Blender foundation, 2017. , dostupno na: https://docs.blender.org/manual/en/dev/ , [12.9.2017]

[5] C.R., Parker, Tessellation WebQuest, Regular Polygon Tessellations, [online], srpanj 2004., dostupno na: http://palmettostateroots.org/tess/Regular_Tessellations.html , [10.9.2017]

[6] Custom 3D models, [online], dostupno na: https://jeannieinabottle3dartistry.files.wordpress.com/2012/03/custom-3d-models-mesh-face-3d-modeling-9778fabelar1.jpg , [10.9.2017.]

[7] Topić, Ivana, Diplomski rad - obrada podataka laserskog skaniranja RapidForm-om, [Online], Hrvatska znanstvena Bibliografija, Sveučilište u Zagrebu, Geodetski fakultet, Zagreb, svibanj 2005., dostupno na: https://bib.irb.hr/datoteka/198888.itopic.pdf ,[8.9.2017.]

[8] About POV-ray, [online], Persistance of Vision Raytracer, 2003.-2004., dostupno na: http://lib.povray.org/about.php , [8.9.2017.]

[9] F.A., Lohmuller, Introduction to the Scene Description Language on the POV-Ray Raytracer, [online], 2014., dostupno na: http://www.f-lohmueller.de/pov_tut/basic/povtuto1.htm , [8.9.2017.]

[10] News Room, [online], Autodesk, 2017., dostupno na: http://www.autodesk.co.uk/adsk/servlet/autoindex?siteID=452932&id=5446881

[11] Autodesk Maya 2011, [online] , WordPress.com, 2011. dostupno na: https://jewelry3dms.wordpress.com/software/maya-2011/ , [6.9.2017.]

[12] [online], Blender Community, vBulletin Solutions, 20017., http://blenderartists.org/forum/ , [19.9.2017.]

[13] GNU General Public Licence, [online], Free Sotware Foundation, lipanj 2007., dostupno na: https://www.gnu.org/licenses/gpl-3.0.en.html , [16.6.2017.]

[14] Computer Entrepreneur Award, Edwin E. Catmull, [online], IEEE, 2017., dostupno na: https://www.computer.org/web/awards/entrepreneur-edwin-catmull, [18.6.2017.]

[15] J., Blinn, Biography, [online], Jim Blinn, 2017., dostupno na: http://www.jimblinn.com/biography/ , [18.6.2017.]

[16] The Cycles Shader Encyclopedia, [online], Greg Zaal, 2015., dostupno na: https://www.blenderguru.com/articles/cycles-shader-encyclopedia , [14.9.2017.]

SAŽETAK

U ovom završnom radu opisan je dio 3D programskog paketa Blendera zvan Cycles Engine. Blender Cycles Engine je objektivno, realno i fizički temeljen program napravljen za animiranje i animacije, što znači da stvara slike na temelju praćenja puta zrake svjetlosti kroz scenu. Objašnjena je *Ray tracing* metoda na osnovu koje radi Cycles Engine te je opisan postupak fotorealističnog renderiranja u Blenderu. Opisani su tipovi *shader*-a koji postoje unutar Cycles Enginea kao glavna tema ovog završnog rada. U praktičnom dijelu su izrađene fotorealistične digitalne slike uz pomoć kojih je prikazan rad opisanih *shader*-a na temelju tri primjera poredanih po složnosti: plišani medvjedić, ručnik, staklena čaša s pivom.

Ključne riječi: Blender, Cycles Engine, 3D modeliranje, fotorealizam, *Ray tracing*, *shader*

ABSTRACT

In this final paper, a part of the Blender's 3D software package called Cycles Engine is described. Blender Cycles Engine is a 3D program designed for animation, which means it creates images based on the Ray tracing method (tracking the path of light rays through the scene). The Ray tracing method is described, aswell as the photorealistic rendering method. The types of shaders that exist within the Cycles Engine are described as the main topic of this final paper. In the practical part, photorealistic digital images were created, showing the work of described shaders. The shaders are presented by examples in three images in the following complexity order: teddy bear, towel, beer glass.

Keywords: Blender, Cycles Engine, 3D Modeling, Photorealism, Ray tracing, Shader

ŽIVOTOPIS

Ines Džebić je rođena 19. ožujka 1995. godine u Osijeku. Osnovnu školu Orašje u Orašju završava 2009. godine i iste godine se upisuje u Srednju školu fra Martina Nedića u Orašju, smjer opća gimnazija. Srednju školu završava 2013. godine i upisuje se na sveučilišni preddiplomski studij elektrotehnike na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. Ines se bavi sportom te je fakultet predstavljala na sveučilišnom, državnom i međudržavnom nivou. Potpredsjednica je Studentskog zbora Fakulteta elektrotehnike, računarstva i informacijskih tehnologija i članica je Fakultetskog vijeća istog.

PRILOZI

U prilogu završnog rada se na CD-u uz elektroničku verziju rada u Word i PDF datotekama također nalaze i izrađene fotorealistične digitalne fotografije prikazane u četvrtom poglavlju, u .blend i .jpg formatu.