SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 443

Napredne metode korištenja video zapisa u računalnoj grafici

Krešimir Špes

Zagreb, Lipanj 2012.

ZAHVALA

Ovim putem želim se zahvaliti svojim roditeljima na strpljenju i podršci prilikom studija, mentorici prof. dr. sc. Željki Mihajlović te kolegama Ivanu Braliću i Mariju Antiću na velikoj pomoći oko izrade grafičkih modela i video materijala za potrebe ovog rada.

Sadržaj

UVOD

Digitalni video zapisi i pripadajuće tehnologije su relativno dugo prisutne u većini medija, no ipak unatoč velikoj popularnosti i masovnog konzumiranja video sadržaja, one se i danas najčešće koriste u najjednostavnijem obliku - dvodimenzionalnom prikazu video slike.

Video zapis koji u suštini nije ništa drugo nego dobro sažeta slijedna arhiva slika ima ogromni potencijal u računalnoj grafici, pogotovo za potrebe aplikacija koje rade u stvarnom vremenu.

U takvim grafičkim aplikacijama, inženjeri uvijek vode bitku između kvalitete prikaza i brzine iscrtavanja slike u prihvatljivoj frekvenciji. Zbog toga je važno optimalno iskorištenje raspoloživih memorijskih i procesorskih resursa sklopovlja na kojem se program izvodi.

Zbog takvih problema često se u 3D vizualizaciji koriste rješenja koja nisu fizikalno točna, ali daju bolji vizualni dojam uz određena ograničenja i žrtve kvalitete prikaza.

Takvih primjera je mnogo, čak bi mogli reći da većina tehnika koje se koriste u grafičkim aplikacijama koje rade u stvarnom vremenu na neki način koriste unaprijed iscrtane slike ili slijed slika za postizanje željenog efekta u stvarnom vremenu.

Kvaliteti takvih rješenja znatno pridonosi animacija tih slika. Tako će na primjer najljepše izgledati voda animirana realističnim izračunima rasprostriranja valova i iscrtana programima za sjenčanje što za sobom povlači najviši utrošak računskog vremena.

Najlošije će izgledati statična slika vode iscrtana u perspektivi u obliku poligona.

Dobro i često kompromisno rješenje jest animacija teksture vode koja je nalijepljena na taj poligon. To može biti obična translacija slikovnih elemenata ili izmjenjivanje više slika površine vode u raznim stadijima.

Što glađu animaciju želimo postići, to trebamo više slika odnosno više memorijskog prostora koji je ograničeni resurs, pogotovo u ugradbenim računalnim sustavima.

Kako bi smanjili računske zahtjeve za prikaz takvih grafičkih efekata uz minimalne

vizualne žrtve i optimalni utrošak memorijskih i računalnih resursa najbolje nam mogu poslužiti video tehnologije koje su u suštini dizajnirane upravo za tu svrhu.

U sljedećim poglavljima pokazati će se kako video tehnologije za mnoge grafičke efekte mogu pružiti najbolji kompromis između navedenih zahtjeva.

Prethodni radovi

Video tehnologija je relativno slabo iskorištena u radovima iz područja računalne grafike. Uglavnom se video tehnologije koriste za 2D plošni prikaz i rjeđe (koliko se uspjelo pronaći drugih radova), prikaz video slike na 3D plohi ili sjenčanje 2D video slike u smislu dodavanja nekog filtra na samu sliku.

Djelomično i indirektno se ovom tematikom bavi moj završni rad na Fakultetu Elektrotehnike i Računarstva - “Prikaz Video zapisa na teksturama objekata” [1].

U radu “Integrating Stereoscopic Video in 3D Games” [34], video slika se koristi u svrhe stereoskopskog prikaza i integracije sa stereoskopskom scenom što vuče paralele s ovim radom jer so opisuje jedna napredna tehnika obrade video slike.

VIDEO TEHNOLOGIJE

Znanstveno je dokazano da ljudsko oko ima donji prag primjetnosti frekvencije izmjene slika od 25 Hz što se postavlja za minimalnu frekvenciju izmjene slika za animacije u računalnoj grafici.

Sve ispod tog praga će primjetno trzati i neće se činiti realističnim. Sve iznad tog praga je poželjno, ali nedovoljno primjetno da bi opravdalo povećanje procesnih i resursnih zahtjeva u većini primjena.

Video tehnologije su algoritmi za kompresiju i dekompresiju slike koji primarno iskorištavaju nedostatke ljudskog oka kako bi iz video zapisa izbacili ili jače degradirali informacije koje ljudsko oko ne vidi odnosno slabije primjećuje s ciljem racionalnijeg korištenja računalnih resursa.

Drugu vrlo važnu činjenicu koje video tehnologije iskorištavaju i zbog čega ostvaruju najveću kompresiju jest kodiranje pokreta između slika u slijedu. U većini slučajeva, sadržaj dvaju slijednih slika u video zapisu se ne razlikuje puno, stoga video tehnologija zapisuje u svakom segmentu zapisa samo razlike u odnosu na prethodnu sliku. Prilikom kodiranja pokreta se također koriste karakteristike ljudskog oka kako bi se dodatno komprimirao zapis uz minimalnu degradaciju vizualnog doživljaja.

Memorijski Zahtjevi

Kako bi izbjegli vizualnu repetitivnost efekta, poželjno je postaviti bar nekoliko sekundi različitih slika te ukoliko je potrebno prikazivati slijed slika u petlji, napraviti ih tako da se ne primjećuje nagli prijelaz između zadnje i početne slike.

Na primjer, animirana tekstura vode u zapisu od 24 bita po slikovnom elementu, 512 x 512 razlučivosti i 5 sekundi trajanja na frekvenciji 25 Hz zahtijeva 94 Mb memorijskog prostora!

Ukoliko bi teksturu vode umjesto serijom sljednih slika računski generirali, memorijski utrošak bi bio znatno manji, ali bi zbog simulacije valova, refleksije i sličnih fizikalnih pojava na površini vode morali utrošiti puno veću količinu računskog vremenu u odnosu na vrijeme potrebno za učitavanje sljedeće teksture u nizu. Očita je potreba za racionalnijom pohranom slika animacije.

Problem sljednog kodiranja video slike

Način kodiranja video slike omogućuje visoku razinu kompresije, ali otežava nasumični dohvat pojedine slike van slijeda. No pošto se video uglavnom prikazuje sljedno, taj nedostatak je relativno malen.

Naime ako bi htjeli dohvatiti sliku u video zapisu van slijeda, odnosno preskočiti na određeni vremenski segment video zapisa, potrebno je dekodirati sve slike koje prethode željenoj slici, slijedno od početka zapisa kako bi se sve razlike između slika ispravno propagirale. U velikim video zapisima ovo postaje prilično dugotrajan i nepraktičan proces, stoga gotovo sve video tehnologije nude neko kompromisno rješenje koje ublažuje taj nedostatak. Najčešće rješenje jest korištenje ključnih slika (eng. Keyframe) koje su zapravo kompletne slike distribuirane manje više ravnomjerno kroz video zapis. Između svakih N slika koje sadrže informacije o razlici u odnosu na prethodnu se ubaci jedna kompletna slika kako bi za dohvat slike van slijeda bilo potrebno dekodirati samo prvu ključnu sliku prije željene te sve slike koje sadrže razlike do željene slike.

Frekvencija kodiranja ključnih slika varira zavisno od potrebe, obično je to svakih 64 ili 128 slika. No neke tehnologije ubacuju dodatne ključne slike ukoliko se dogodila značajnija promjena između dvije slike.

Korištenje ključnih slika je kompromis jer time povećavamo veličinu konačnog video zapisa te usporavamo proces dekodiranja, ali dobivamo mogućnost dohvata slike van slijeda u konstantnom vremenu. Ukoliko se video zapis prenosi u stvarnom vremenu preko internetske veze, ovakvo rješenje značajno smanjuje propusne zahtjeve kada korisnik želi preskočiti na određeni vremenski segment video zapisa.

Ukoliko želimo dekodirati neku sliku van slijeda nailazimo na problem – gdje se fizički u video zapisu ta slika nalazi? Pošto veličina svake slike ovisi o količini promjene u odnosu na prethodnu, nemoguće je analitički odrediti gdje se u zapisu određena slika nalazi.

Ukoliko video tehnologija podržava indeks ključnih slika, postupak traženja slike se svodi na konstantnu složenost. U protivnom je potrebno provesti neku vrstu pretraživanja zapisa poput binarnog algoritma za pretraživanje[2].

Transformacije video slike

Najveći nedostatak korištenja video tehnologija umjesto slijednog zapisa slika jest računska zahtjevnost dekodiranja slike. Obično video zapis sadrži podatke zapisane u nekoj prostornoj transformaciji, npr Fourierovoj[3] ili diskretnoj kosinusnoj transformaciji[4], pa je potrebno utrošiti računsko vrijeme za transformaciju slikovnih elemenata video slike iz transformiranog u normalni prostor.

Osim pohrane informacija u transformiranom prostoru, većina video tehnologija koristi YUV[5] sustav boja ili neke varijante tog sustava poput YCbCr[6] zbog toga što su ti sustavi slični karakteristikama ljudskog oka dok je npr. RGB[7] sustav koji se koristi u 2D/3D prikaznim tehnologijama prilagođen prikazu na zaslonima.

Zbog takvog sustava boja, video tehnologije obično smanje rezoluciju kromatskih komponenti zbog optimalne kompresije jer je oko osjetljivije na promjenu svjetline (luminance) u slici nego na promjenu boje (krominance).

Zbog te razlike između sustava boja potrebno je nakon transformacije, prebaciti boje slikovnih elemenata u RGB sustav boja.

Taj proces se može ubrzati korištenjem multimedijskih ekstenzija procesora ili programa za sjenčanje na grafičkoj kartici, ali treba voditi računa da takvo rješenje nije sklopovski prenosivo.

Stoga je uputno pri odabiru korištenja video zapisa za postizanje efekta odrediti je li računski zahtjevnije koristiti video tehnologiju ili simulirati efekt programski.

Video i općenito tehnike ubrzavanja prikaza efekta bazirane na animiranom slikovnom slijedu uputno je koristiti samo onda kada je programskom simulacijom suviše složeno, ako ne i nemoguće postići istu ili sličnu razinu kvalitete prikaza.

Također treba voditi računa za što je određena video tehnologija optimizirana, najbolje rezultate ćemo dobiti kodiranjem stvarnih slika ili tekstura prirode u RGB sustavu boja. Pošto su video tehnologije optimzirane za kodiranje manjih pokreta, bitno je ne kodirati značajno različite slike u slijedu jer time nećemo uštedjeti na prostoru, a i računska složenost dekodiranje sljedeće slike će biti velika. U ekstremnim slučajevima kodiranjem potpuno nasumičnih slika u video zapis, mogli bi dobiti i suprotan efekt - porast prostornih i računskih zahtjeva!

Odabir video tehnologije

Praktičnost i efikasnost korištenja video rješenja uvelike ovisi o odabranoj video tehnologiji. Različite tehnologije variraju u omjeru kompresije i brzine dekodiranja.
Također je bitno kakvu kontrolu nad brzinom i slijedom prikaza omogućuje tehnologija.
Ukoliko kanimo koristiti video tehnologije u komercijalnom projektu, važno je istražiti kakva se patentna ograničenja nameću jer većina video tehnologija zahtjeva plaćenu licencu za korištenje.
Iako su u pravilu komercijalne tehnologije brže i efikasnije, postoji par potpuno besplatnih i open-source rješenja poput Theora[8], WebM[9] ili Dirac[10] video tehnologija koje su dovoljno brze i optimizirane za većinu primjena.

Prenosivost

Ukoliko smo se odlučili za korištenje video tehnologije, treba uzeti u obzir na kojim se sve platformama treba izvršavati program kojeg izrađujemo. Često su tehnologije vezane za određeno sklopovlje ili operativni sustav. Tako je za Microsoft Windows platformu dostupan velik broj tehnologija dok ih za ugradbene sustave ima znatno manje. Kada planiramo distribuirati programsko rješenje za više platformi, ako je moguće, uputno je koristiti tehnologiju koja na tim platformama daje najbolje rezultate. Postoje neka programska rješenja poput FFmpeg projekta[11] koja olakšavaju probleme prenosivosti i korištenja različitih tehnologija.

Sklopovska Ograničenja

Brzina dekodiranja video slike najviše zavisi o razini korištenja sklopovski ubrzanih funkcija. Neke tehnologije na određenim platformama imaju posebne sklopovske dekodere koje je moguće iskoristiti za brzo dekodiranje (npr. H.264[12]) dok se neke tehnologije oslanjaju na razne procesorske multimedijske ekstenzije poput MMX[13] ili SSE[14] ekstenzija.

Važno je voditi računa o ciljanim platformama. Tako na primjer video tehnologija može raditi jako brzo na x86 arhitekturi, ali jako sporo na ARM arhitekturi jer ne koristi nikakve ARM multimedijske ekstenzije.

Proces dekodiranja možemo ubrzati ako znamo koju informaciju iz video zapisa trebamo. Ako trebamo samo luminantnu komponentu slike, onda je u pravilu proces dekodiranja brži. Naime, zbog toga što većina video tehnologija uglavnom koristi YUV sustav boja koji odvaja luminantnu komponentu od kromatskih, možemo direktno izvući luminatnu komponentu i time izbjeći računski intenzivnu konverziju takvog sustava boja u RGB ili neki drugi sustav prilagođen sustavu boja zaslona na kojem se prikazuje konačna slika.

IMPLEMENTACIJA

Za potrebe ovog rada odabrana je Theora video tehnologija zbog toga što nije zaštićena patentima. Theora je efikasna tehnologija po kvaliteti i brzini u usporedbi s MPEG4[15]. Iako zaostaje za suvremenim komercijalnim tehnologijama poput H.264, dovoljno je brza i efikasna za većinu primjena.

Trenutno njen glavni nedostatak jest manjak ARM optimizacije za dekodiranje što ograničava primjenu u ugradbenim sustavima.

Također bitan nedostatak je programska kompleksnost korištenja Theora aplikacijskog sučelja. Potrebno je dosta znanja o prikazu i sinkronizaciji video zapisa kako bi se mogao pravilno prikazati video zapis. Theora API[16] je zapravo sučelje pomoću kojeg programer može dohvatiti pojedine slike iz video zapisa, a kako i kada će ih dekodirati, pretvoriti u potreban sustav boja i prikazati je odgovornost programera koji implementira Theora tehnologiju.

Zbog tog nedostatka, direktno korištenje Theora programskog sučelja nije toliko raširenо u aplikacijama, već se koriste jednostavnija sučelja poput na primjer DirectShow-a[17] ili GStreamer[18]-a preko kojih se dekodira Theora zapis.

U sklopu mog Završnog rada na Fakultetu Elektrotehnike i Računarstva “Prikaz Video zapisa na teksturama objekata”, razvijeno je programsko rješenje koje programeru korisniku znatno olakšava i optimira dekodiranje i pravovremeni prikaz Theora video zapisa bez dodatnih komplikacija i ograničenja. Glavno obilježje koje to programsko rješenje omogućuje je dekodiranje videa u zasebnoj procesnoj dretvi, dodavajući dekodirane slike u red čekanja s kojeg korisnik pravovremeno skida slike i prikazuje na glavnoj dretvi. Tu se iskorištava činjenica da se jednostavnije slike brže dekodiraju od kompleksnijih te se time minimizira problem odbačenih slika zbog nedostatka vremena za dekodiranje jer su u drugoj dretvi već dekodirane i čekaju u redu kada dođe vrijeme za njihov prikaz.

To programsko rješenje je slobodno distribuirano u zajednici otvorenog koda te je od trenutka kada je napravljeno u sklopu završnog rada steklo popriličnu popularnost - do današnjeg dana iskorišteno je u petnaestak raznih komercijalnih i nekomercijalnih projekata.

JEDNOSTAVNE TEHNIKE

U sljedećim poglavljima ilustrirano je par jednostavnijih načina korištenja video zapisa za postizanje raznih efekata.

Jednostavnom tehnikom korištenja videa u grafičkim aplikacijama možemo smatrati onu tehniku kod koje se ne modificiraju slikovni elementi video slike već direktno sudjeluju u formiranju izlaznih slikovnih elemenata na zaslonu ekrana.

Direktan prikaz u prostoru

Slika 1: Prikaz video slike u prostoru na 3D sceni

Osim direktnog 2D prikaza na ekranu ova tehnika je jedna je od najjednostavnijih načina korištenja videa u grafičkoj aplikaciji. Ova tehnika je ujedno i baza za sve ostale naprednije tehnike.

Video slika se preuzme iz dekodera i pošalje u obliku teksture na grafičko sklopovlje te se kao tekstura koristi za preslikavanje na plohama poligona.

Na slici 1 možemo vidjeti prikaz unutrašnjosti sobe s televizorom na čijem se zaslonu prikazuje video slika. Scena je izrađena u 3D alatu koji je u teksture objekata scene unaprijed izračunao i zapisao statično osvjetljenje koje dolazi iz plohe televizora.

Dodatno u ovom demo programu iz svake video slike se izračuna prosječna boja svih slikovnih elemenata i ta se boja koristi za boju ambijentalnog svijetla kako bi se dobio dojam televizora kao izvora svjetla.

Plošni prikaz animacije

Svaka scena se sastoji od statičkih i dinamičkih elemenata. Tako na primjer možemo prikazati oblake statičnom slikom preslikanom na kocku koja okružuje prostor u kojemu se krećemo. Tu tehniku koristimo kako bi smanjili računske zahtjeve za prikaz oblaka u odnosu na realističnije metode poput prikaza oblaka *sustavom čestica*.

Statična slika oblaka je prilično loša aproksimacija jer se oblaci kreću, mijenjaju oblik i slično. Iako u prilog ide činjenica da je nebo uvijek u zadnjem planu, nekoga tko promatra 3D scenu u pravilu neće puno smetati statičnost neba.

Oblaci se relativno sporo kreću i deformiraju te kada bi te promjene kodirali pomoću video tehnologije, znatno bi se uštedjelo na memorijskom prostoru zbog toga što su video tehnologije optimizirane za kodiranje malih pokreta.

Drugi primjer korištenja ove tehnike je plošna animacija vode u prostoru. Na sličan način se animacija vode unaprijed iscrta u 3D modelerskom programu te slike animacije pohrane u video zapis. Nedostatak ove tehnike je što je slika vode iscrtana iz jednog gledišta i ukoliko u našem grafičkom programu dozvoljavamo velike oscilacije gledišta, moglo bi se dogoditi da voda iz nekog kuta neće dobro izgledati. Stoga je ovakva aproksimacija ograničena na situacije gdje je donekle fiksirano gledište poput simulacije letenja (voda je uvijek okomito ispod aviona) ili u situacijama gdje poglede na vodu koji znatno odstupaju od originalnog gledišta blokira neki drugi objekt te time ne dozvoljava iscrtavanje vode u nepovoljnom gledištu. Primjer ovakvog rješenja možemo vidjeti na slici 2.

Slika 2: Unaprijed iscrtan prikaz vode iz fiksnog gledišta.

Refleksija okoline

Slika 3: Refleksija

Video tehnologije se vrlo dobro mogu primijeniti kod prikaza refleksije. Na slici 3 možemo vidjeti primjer trodimenzionalnog objekta koji reflektira okolinu pomoću Cube Map[19] tehnike koja scenu iscrtava u 6 tekstura iz 6 različitih gledišta postavljenih u centar objekta čije projekcione plohe tvore kocku.

Nakon tih šest iscrtavanja te se teksture koriste prilikom iscrtavanja objekta kako bi se dobio dojam refleksije svjetlosti na tom objektu.

Ovakva metoda je iznimno računski zahtjevna zbog velikog broja iscrtavanja scene stoga se često koriste unaprijed iscrtane statičke slike prostora za taj efekt. No ukoliko pomaknemo položaj objekta, takve teksture više ne vrijede za novu poziciju. Tu se može koristiti video tehnologija kao kompromisno rješenje za određene primjene ovog efekta.

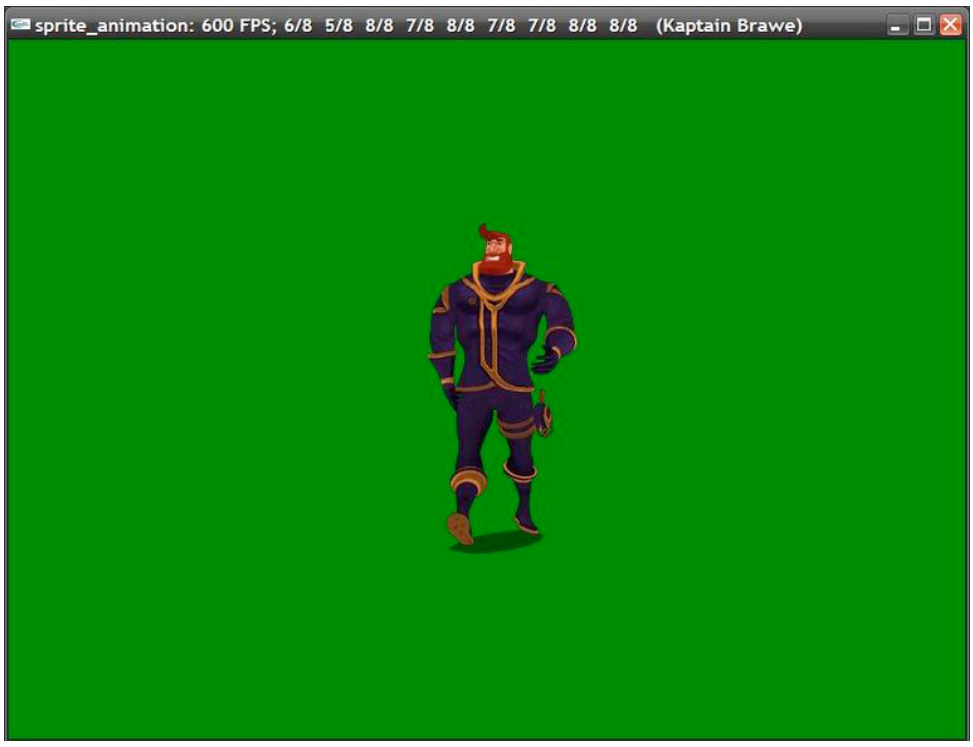
Možemo u video kodirati tih 6 tekstura na način da animiramo okolinu ili scenu iz različitih gledišta ukoliko se objekt animirano pomiče u prostoru.

Tih 6 tekstura je najbolje u video kodirati u jednu veću video sliku, na primjer svih 6 slika vertikalno nanizanih u jednu sliku te prilikom dekodiranja razlomiti na 6 tekstura.

NAPREDNE TEHNIKE

U sljedećim poglavljima prikazane su odabrane napredne tehnike.
Za razliku od jednostavnih, napredne tehnike modificiraju slikovne elemente ili služe kao apstraktni ulaz za grafički efekt.

Dekodiranje kanala prozirnosti



Slika 4: Prikaz poluprozirne video slike

Niti jedna popularna video tehnologija ne podržava kanal prozirnosti (eng. alpha channel) te ukoliko želimo koristiti video sliku za prikaz poluprozirnih elemenata, moramo na neki način dekodirati kanal prozirnosti iz video slike. Najbolje rješenje jest kodiranje tog kanala u samu video sliku na način da je s jedne strane video slike prava RGB slika a na drugoj kanal prozirnosti. Onda prilikom pretvorbe iz YUV sustava boja u RGB možemo pretvoriti samo lijevu polovicu slike a iz druge polovice uzeti samo luminantnu komponentu kao kanal prozirnosti i to sve spojiti u RGBA sliku dvostruko manje širine od video slike.
Na slici 4 možemo vidjeti prikaz 2D lika koji vrti animaciju hodanja te je iscrtan

pomoću poluprozirnog kanala na zelenu pozadinu. Kada slika ne bi sadržavala taj kanal, oko lika bi se iscrtao crni pravokutnik.
Na slici 5 možemo vidjeti kako izgleda jedna takva slika u video zapisu.

Slika 5: Primjer jedne video slike s kodiranim poluprozirnim kanalom na desnoj strani

U demo programu koji ilustrira ovu tehniku uzet je za primjer lik iz računalne igre koji mora biti visoke rezolucije pošto je predmet fokusa tijekom igranja.
Za animaciju hodanja uzmimo rezoluciju 250x300 slikovnih elemenata (RGBA), 20 slika po sekundi i 8 smjerova. Uz laganu upotrebu kalkulatora dolazimo do minimalno potrebnih 46 MB memorije i to samo za animaciju kretanja!
Koristeći video tehnologiju i opisanu metodu za dobivanje kanala prozirnosti, značajno su smanjeni memorijski zahtjevi 2D animacije uz malo povećan utrošak procesorskog vremena.
U ovom primjeru koriste se 8 video datoteka od kojih se samo jedna dekodira u danom trenutku, ovisno o smjeru kretanja lika.
Jedna video datoteka ima 20 slika te je sveukupna veličina tih datoteka 1.7 MB.
Za glatki prikaz, potrebno je unaprijed dekodirati 2 - 4 slike, što za 8 smjerova zahtjeva dodatnih 4.3 MB za pohranu što sveukupno daje 6 MB video podataka učitanih u radnu memoriju.
Jedini realan nedostatak ove metode je što zahtjeva konstantan utrošak

procesorskog vremena za dekodiranje slika. No, taj utrošak je čak i pri ovako velikim slikama relativno mali.

Demo program uz prikaz 60 slika po sekundi na modernom PC računalu troši svega oko 10% procesorskog vremena.

Koristeći ovu metodu moguće je napraviti jako puno slika uz mali utrošak memorijskog prostora jer su video tehnologije optimizirane za pokretnu sliku, tj. spremaju se samo oni segmenti slike koji su se promijenili u odnosu na prošlu i zato te datoteke zauzimaju daleko manje prostora nego kod prve metode.

Ovom metodom za animaciju lika uspjeli smo smanjiti zahtjeve za memorijskim prostorom s 46 MB na svega 6 MB za isti efekt. Za to smo morali donekle žrtvovati kvalitetu prikaza slike te utrošiti nešto više procesorskog vremena.

Treba uzeti u obzir da većina video tehnologija zapravo degradira slikovne informacije. Razina kvalitete odnosno omjer kvalitete slike i veličine izlazne video datoteke se u svim video tehnologijama može odrediti prilikom kodiranja videa. U ovakvim slučajevima poželjna je veća kvaliteta prikaza slike kako se ne bi značajnije degradirao kanal prozirnosti.

Dakle, ovom metodom smo napravili veliku uštedu, ali najveća prednost je u tome što uz neznatni porast u veličini video datoteka možemo kodirati više slika po sekundi ili dužu sekvencu što bi bilo daleko memorijski zahtjevnije konvencionalnom metodom pohrane animacije lika.

Ovom metodom možemo također kodirati puno različitih animacija u jednu video datoteku te ovisno o kontekstu prikazivati samo određene video slike u petlji.

Parcijalna animacija 2D scene

Kada imamo potrebu za animacijom scene koja je većinom statična, upotreba video tehnologije daje izvrsne rezultate!

Slika 6: Animirani glavni izbornik u igri "Legenda o Kristalnoj Dolini"

Svaka računalna igra ima barem jednu statičnu scenu – glavni izbornik. Pošto je taj izbornik u pravilu prva slika koju korisnik vidi i na temelju čega nesvjesno donosi prvi dojam o samom proizvodu, dobra je praksa potruditi se da ta scena ostavi što bolji dojam kvalitete i dorađenosti.

To se najjednostavnije ostvaruje raznim animacijama na sceni poput na primjer čestičnih sustava, animacijom pozadinskih i prednjih planova itd.

Ovisno o vrsti animacije, prostorni i računski utrošak varira. Tako na primjer u računalnoj igri Legend of Crystal Valley[20] hrvatskog studija za razvoj računalnih igara Cateia Games[21] na glavnom izborniku se mogu vidjeti dvije vrste animacija: 3D lik koji vrti neutralnu animaciju na sceni te animirana 2D animacija vode koje se izmjenjuju u petlji. Kako to izgleda, može se vidjeti na slici 6.

Animacija vode je u ovoj igri izvedena pomoću 16 slika vode u razlučivosti 1024x176 što ukupno troši 8.25 MB memorije. Ukoliko animaciju vode kodiramo u Theora video zapis dobivamo datoteku od svega 166 kB! Usporedivši to s PNG[22] zapisom koji troši 6.1 MB ili čak JPG[23] zapisom koji troši 5.7 MB očigledno je da

smo uštedjeli dosta prostora. No diskovni prostor najčešće nije glavni problem već je to količina potrebne radne memorije za prikaz efekta. Ako računamo da su nam dovoljne dvije unaprijed dekodirane video slike za glatku animaciju, trošimo 1 MB umjesto 8.25 MB za pohranu cijele animacije u radnoj memoriji. Uzevši u obzir da je izvorna animacija imala relativno malo slika, upotrebom video tehnologije nismo dobili neko spektakularano poboljšanje. Prednost animiranja ovog efekta uz pomoć video zapisa bila bi izraženija kada bi animacija vode imala više slika jer je potrebna količina radne memorije za animaciju ovog efekta uz pomoć videa konstantna, dok za klasičnu metodu potrošnja raste linearno s porastom broja slika animacije.

Sljedeća animacija koju ćemo pokušati optimizirati jest animacija 3D lika na sceni. Taj lik je animiran skeletalnom animacijom[24] koja traje desetak sekundi i vrti se u petlji.
Skeletalna animacija je vrlo efikasan način animacije 3D likova i ne zauzima puno memorijskog prostora i neosjetljiva je na rezoluciju ekrana. Uzevši ove dvije činjenice u obzir, očito je da će bilo kakav pokušaj animiranja ovog lika video tehnologijom rezultirati u većoj potrošnji radne memorije. No, razlog zašto bi se odlučili za animiranje ovog lika video tehnologijom umjesto 3D skeletalnom animacijom jest potencijalno manja procesorska složenost prikaza ukoliko se radi o vrlo složenom liku. Na taj način imamo konstantne računske zahtjeve i možemo prikazati lik sa znatno više detalja poput npr. realistično animirane kose, izraza lica itd.
Drugi razlog bi bio u tehnološkim ograničenjima. Neki sustavi nemaju sklopovlje koje ubrzava 3D iscrtavanje ili nije dovoljno brzo za prikaz složenih 3D modela.
Upotrebom video tehnologije za tu animaciju dobili bi znatno ubrzanje.

	Broj slika	**Veličina**	**Theora**	**2 slike**	**4 slike**	**8 slika**
Voda	16	8.25 MB	171 kB	1.03 MB	2.06 MB	4.13 MB
3D lik	375	259.4 MB	274 kB	0.66 MB	1.31 MB	2.63 MB

Tablica 1: Optimizacijska tablica za scenu izbornika iz igre "Legenda o Kristalnoj Dolini"

Na tablici 1 možemo vidjeti memorijske uštede korištenjem Theora video zapisa u gore navedenom primjeru u odnosu na originalnu implementaciju.
Dakako, mogli bi cijelu scenu kodirati u video zapis umjesto pojedinih elemenata što bi također bilo prilično dobro kodirano i prostorno opimizirano. No takva tehnika ima i nekoliko nedostataka. Prvo, u tom slučaju umjesto manjeg slikovnog segmenta, trebali bi dekodirati cijelu sliku što za sobom povlači veću računsku složenost procesa dekodiranja i pretvorbe YUV u RGB prostor boja.
Drugo, time gubimo mogućnost isključivanja nekih animiranih elemenata scene.
Treće, video tehnologijom gubimo na kvaliteti slike u korist manje izlazne datoteke te bi slojevit prikaz scene bio vizualno atraktivniji jer statične elemente možemo kodirati algoritmima koje manje ili uopće ne degradiraju kvalitetu slike.
Ti nedostataci postaju posebno izraženi uzevši u obzir današnji trend porasta razlučivosti zaslona. Na primjer treća generacija Apple iPad tablet uređaja[25] ima razlučivost ekrana 2048x1536 slikovnih elemenata što je izrazito računski zahtjevno za dekodiranje video slike, stoga je bolje slojevito prikazivati scenu te samo određene animacije prikazati pomoću video zapisa.

Prilikom korištenja ove tehnike treba pažljivo odabrati koje animacije na sceni ćemo kodirati video tehnologijom. Na primjer pozadinski plan u gore navedenom primjeru možemo prikazati samo jednostavnom translacijom teksture oblaka te animaciju grane stabala jednostavnom rotacijom slike grane za par stupnjeva. Uzevši u obzir da stablo nije potpuno u kadru, jednostavna rotacija je dovoljno dobra aproksimacija kretanja stabla na vjetru.

Slika 7: Proširena scena s glavnog izbornika iz igre "Legenda o Kristalnoj Dolini"

Originalna scena iz gore navedene igre je od animacija sadržavala samo animirani 3D lik i animiranu vodu.

Zbog jasnije ilustracije prednosti kodiranja ovakve scene video tehnikom, ta slika je proširena s par dodatnih elemenata: animacije grane stabla u prednjem planu te animacije oblaka i ptica u pozadinskom planu.

Na slici 7 možemo vidjeti kako izgleda proširena scena.

Prikaz osvjetljenja u animiranoj sekvenci

Slika 8: Primjer svjetlosne teksture iz demo programa

Realistično osvjetljenje je najkompleksniji proces u grafičkim programima koji rade u stvarnom vremenu. Koliko god su računalni resursi napredovali s vremenom, i dan danas nemamo dovoljno računskih resursa u osobnim računalima za računanje realističnog osvjetljenja u realnom vremenu. Pod pojmom realistično osvjetljenje smatraju se metode poput algoritma praćenja zrake (eng. Ray tracing) [26] i *postupka isijavanja (eng. Radiosity)*[27] kod kojih se računa refleksija svjetlosti, apsorpcija, meke sjene i slično.

U takvim aplikacijama, sve metode osvjetljenja su u manjoj ili većoj mjeri aproksimacija stvarnog fizikalnog modela osvjetljenja. Najjednostavnije metode su izvedene u tzv. fixed-function[28] arhitekturi grafičkog sklopovlja, dok puno bolje rezultate dobivamo korištenjem raznih algoritama za izračun sjena.

Najbolje je kada se to kombinira s osvjetljenjem izračunatim uz pomoć programa za sjenčanje. No sve te metode ne uzimaju u obzir kompleksnije realne efekte.

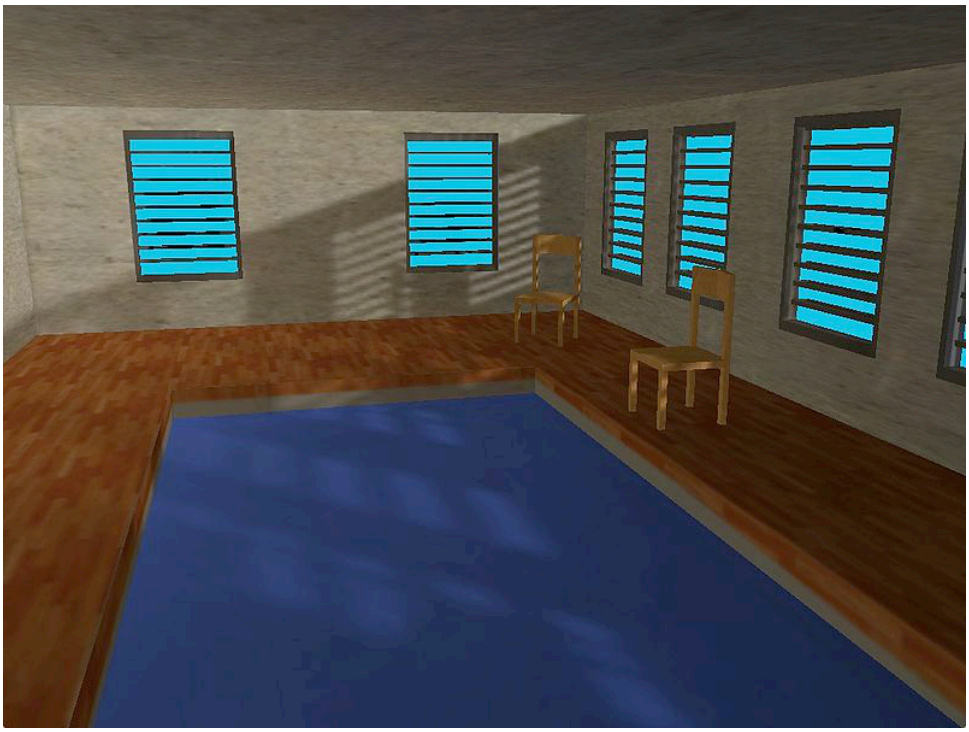
Često korištena metoda je unaprijed izračunato osvjetljenje scene u obliku tekstura koje se preslikavaju na poligone, takozvane svjetlosne teksture (eng. light map). Primjer takve teksture moguće je vidjeti na slici 8.

One se obično generiraju pomoću 3D modelerskih programa ili specijaliziranih programa za računanje isijavanja koji su dizajnirani za tu svrhu ili su dio većeg paketa alata koji dolazi uz neki grafički pokretač (eng. Engine) poput na primjer Unreal Engine-a[29].

Glavni nedostaci te metode su veliki memorijski zahtjevi te statičnost osvjetljenja.

Memorijske zahtjeve ne možemo riješiti video tehnologijom jer ukoliko imamo puno površina u sceni, bit će potrebno puno teksturnog prostora za osvjetljenje. Taj problem možemo donekle minimizirati korištenjem nekog algoritma za podjelu prostora i onda koristiti samo one svjetlosne teksture koje se koriste u kadru.

Statičnost takvog osvjetljenja možemo donekle riješiti video tehnologijom, ali samo u ograničenim okvirima. Ako imamo izvor svjetla koji nije statičan, ali se njegovo gibanje može svesti na konačan i ponovljivi slijed pozicija / rotacija, onda možemo u video kodirati izračunatu mapu osvjetljenja za svaku uzorkovanu orijentaciju svjetla u njegovom gibanju. Takav mehanizam izvrsno funkcionira kodiran u video zapis zbog već navedenih svojstva za dobro kodiranje promjene slike u video tehnologijama.



Slika 9: Svjetlosna mapa pomnožena sa slikovnim elementima scene.

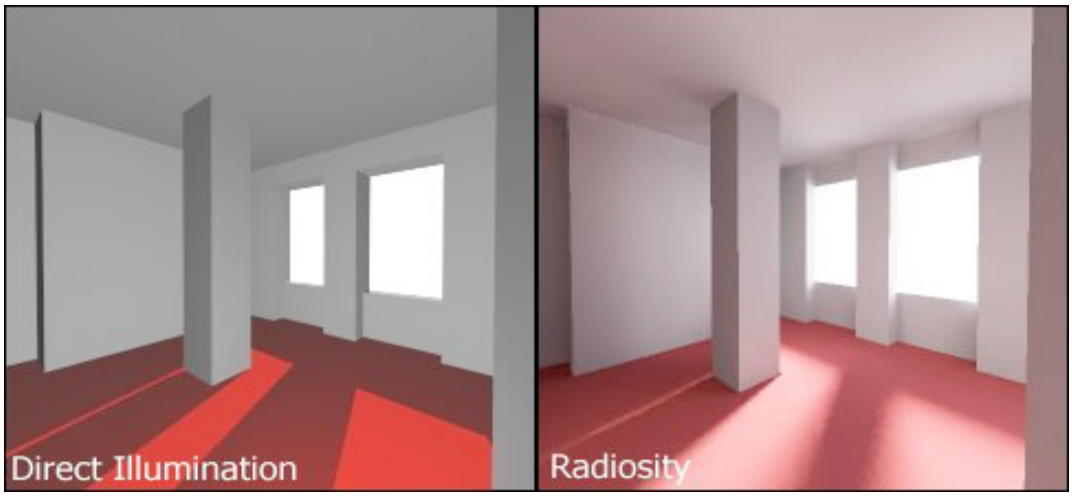
Za potrebe ovog rada razvijen je demo program koji ilustrira ovu tehniku. U 3D modelerskom programu je napravljena scena s par izvora svjetala koji se gibaju i izračunati su njihovi svjetlosni doprinosi u obliku tekstura za svaku orijentaciju svjetla u njegovoj kružnoj putanji. Na zahtjev korisnika, određena svjetla se mogu isključiti ili uključiti kako bi se ilustrirala tehnika. Mogu se primijetiti lijepe i realne sjene u prostoru. Izlazna slika demo programa se može vidjeti na slici 9.

Kao što je već rečeno, ova tehnika animiranja svjetla je ograničena zbog potrebe za kružnom putanjom izvora svjetla koji se animira, no pošto video tehnologijom imamo relativno mali prirast veličine kodirane datoteke i konstantne zahtjeve za memorijom, možemo kodirati iznimno duge sekvence animiranja svjetla.

Ovu tehniku je najbolje koristiti u dva slučaja:

1) Kada imamo potrebu za ponovljivom animacijom izvora svjetla ili objekta koji značajno sudjeluje u osvjetljenju, poput ventilatora ispred izvora svjetla.
2) Za animaciju svjetla u nekoj preferiranoj animiranoj scenskoj sekvenci, u kojem slučaju nije bitno da se putanja svjetla ponavlja.

Globalno osvjetljenje u animiranoj sekvenci



Slika 10: Usporedba klasičnog GPU osvjetljenja s realističnijom tehnikom

Zbog izrazito velike računske složenosti izračuna realnog osvjetljenja u 3D sceni, čak i uz današnje sve naprednije grafičko sklopovlje, i dalje se u aplikacijama koje rade u stvarnom vremenu uglavnom koriste jednostavnije aproksimativne tehnike. Te tehnike se svode na jednostavne izračune direktnog osvjetljenja kod kojeg se računa doprinos svjetla ovisno o kutu pod kojim pada na normalu određene plohe. Doprinos ambijentalnog osvjetljenja se računa jednostavnim zbrajanjem ambijentalne boje svjetla s izračunatim direktnim osvjetljenjem. U takvom modelu nema sjena, refleksije i sličnih efekata.

Korištenjem naprednijih grafičkih tehnika i algoritama te programa za sjenčanje uspjelo se u takav model ubaciti razne efekte koje nalazimo kod realnog izračuna svjetla. No koliko god te tehnike dobro izgledale danas, niti jedna se ne može mjeriti s fizikalno točnim tehnikama praćenja zrake svjetla poput postupka isijavanja.

Na slici 10 možemo vidjeti usporedbu jednostavnijeg modela osvjetljenja koji koristi tehniku volumnih sjena[30] i postupka isijavanja. Čak i u ovako jednostavnom primjeru realni izračuni svjetla ostavljaju daleko bolji dojam.

Unatoč konstantnom povećanju procesorske snage, daleko smo još od korištenja postupka isijavanja u stvarnom vremenu. No postoji jedan segment računalne grafike u kojem se mogu koristiti unaprijed iscrtane slike umjesto onih generiranih u stvarnom vremenu – animirane sekvence.

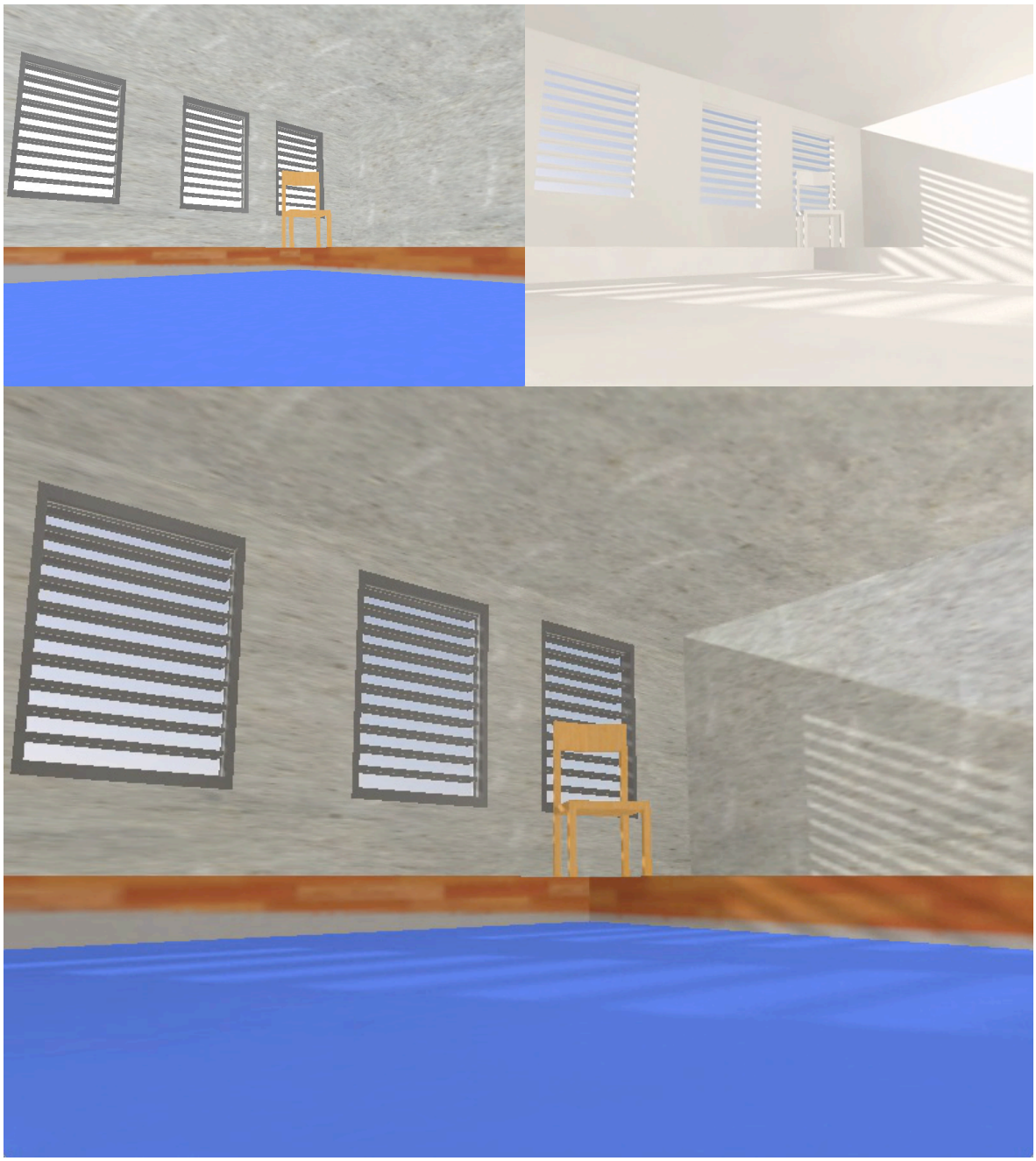
Animirane sekvence

Takve sekvence se često koriste u računalnim igrama i obično su realizirane na jedan od sljedeća dva načina:

- prikazane u stvarnom vremenu gdje su objekti animirani unaprijed definiranom putanjom.
- unaprijed iscrtana sekvenca pohranjena u obliku video zapisa.

Obje metode imaju svoje prednosti i mane. Prva metoda je neosjetljiva na rezoluciju prikaza, ali zahtjeva dodatno programiranje animiranih elemenata što u kompleksnim scenama iziskuje značajnu količinu rada dok kod video tehnike dizajner s profesionalnim alatima može postići u pravilu ljepše animacije u manje vremena. Druga velika prednost animacije u stvarnom vremenu leži u tome da nije potrebna velika količina diskovnog prostora za pohranu animacije pošto se pohranjuju samo pozicije, orijentacije i stanja objekata u svakom trenutku.
Prikaz animacije videom može dati vrlo lijepe rezultate, ali uz poveći utrošak diskovnog prostora. Korištenjem tehnika za realni izračun svjetla, svaka slika video zapisa se unaprijed izračuna i garantira dobar i fizikalno točan izgled 3D scene.
Kako bi smanjili prostorne zahtjeve moramo dakako koristiti video tehnologije za pohranu takvog zapisa te ugađanjem kompresijskih parametara odabrati dobar kompromis između veličine datoteke i izlazne kvalitete slike. Rezolucija je također problem pošto reprodukcija video zapisa jako ovisi o njoj. Ukoliko video prikazujemo na većem ekranu a imamo manju rezoluciju video zapisa, rezultatna slika će izgledati mutno dok u prvoj tehnici nemamo tih problema.
Isto tako, nije poželjno dopustiti preveliku degradaciju kvalitete zbog moguće pojave raznih artefakata uzrokovanih kompresijom.

Hibridna metoda osvjetljenja scene u sekvencama



Slika 11: Scena bez osvjetljenja, osvjetljenje u video zapisu, izlazna slika

Prethodno navedene tehnike za animiranje sekvenci su poslužile kao uvod za ovo poglavlje u kojem se opisuje spoj tih dviju tehnika čime ćemo umanjiti njihove nedostake a povećati prednosti.

Ideja te tehnike je da se osvjetljenje cijele scene u svakoj slici animirane sekvence unaprijed izračuna i pohrani u video zapis, a sama scena se prikazuje iscrtavanjem poligona bez osvjetljenja. Te dvije slike se onda spoje u izlaznu sliku

dajući lijepe rezultate.

Primjer ulaznih i izlazne slika za ovu tehniku moguće je vidjeti na slici 11.

Važno je napomenuti da je ovakva tehnika ograničena samo za animirane sekvence gdje je pozicija i orijentacija svakog objekta točno određena, pošto se sadržaj video zapisa ne može mijenjati. Ukoliko se napravi promjena u poziciji nekog objekta u danom trenutku animacije, potrebno je ponovo izračunati osvjetljenje u cijeloj sekvenci.

Prednosti ove tehnike su višestruki. Smanjili smo računske zahtjeve za prikaz scene izbacivši izračun osvjetljenja i većinu, ako ne i sve programe za sjenčanje te smo time oslobodili računalne resurse za dekodiranje video slike.

Nakon iscrtavanja scene bez osvjetljenja, sljedeći korak je dekodiranje video slike i množenje slikovnih elemenata te slike sa slikovnim elementima iscrtane scene.

Kodirajući samo osvjetljenje u video sliku dobivamo mnogo bolju kompresiju uz jednake razine kvalitete slike zbog toga što u video slici nema nikakvih tekstura i sitnijih detalja na površinama objekta. To pogoduje statičnoj kompresiji, a zbog toga će i količina promijenjenih slikovnih elemenata iz slike u sliku biti manja što najviše pridonosi kompresiji.

Na primjer, sekvenca korištena u demo programu koja sadrži samo osvjetljenje zauzima 3.9 MB prostora dok veličina te iste sekvence naraste na 6.2 MB ako u video kodiramo i teksture na sceni.

Slika 12: Rubni artefakti kod većih razlika u rezoluciji

S obzirom da je većina detalja u teksturama objekta, možemo si priuštiti i veće kompresijske omjere pošto će njihov vizualni utjecaj biti manji jer utječe samo na osvjetljenje.

No svaka aproksimativna tehnika ima svoje mane i ograničenja, pa tako i ova.

Prvi problem leži u rezoluciji. Ukoliko je razlika između rezolucije video slike i ciljne rezolucije velika, primijetit će se prijelazi u osvjetljenju slikovnih elemenata na rubovima objekata kao što možemo vidjeti na slici 12.

Na drugi problem nailazimo kod izračuna svijetla. Naime, u video sliku ne ubacujemo teksture objekata te samim time ne znamo kakvu boju svjetla treba reflektirati određena ploha pogođena svjetlom. Stoga je potrebno iscrtati scenu alatom koji kao ulaz prima stvarnu scenu s teksturama a na izlaz daje samo osvjetljenje ili namjestiti dodatne fiktivne izvore svjetla prilikom izračuna osvjetljenja koji bi nadomjestili te nedostatke. Druga opcija je upotrijebiti reflektivnu boju odnosno teksturu u programu za iscrtavanje. Ovaj problem je u potpunosti rješiv ukoliko se koriste odgovarajući grafički alati, no svejedno treba biti svjestan problema i voditi računa o njemu prilikom korištenja ove tehnike ako želimo dobiti najbolje vizualne rezultate.

Projekcijsko teksturiranje

Slika 13: Primjer osvjetljenja s projektivnom teksturom

Kod tehnike osvjetljenja projekcijskom teksturom, zrake svjetlosti koje bojaju slikovne elemente poligona objekta umjesto fiksne boje za difuznu komponentu svjetla, boju mogu uzeti iz projekcijske teksture. Zraka koja spaja točku izlaznog slikovnog elementa u prostoru i poziciju izvora svjetla se projicira na plohu projekcijske teksture te se slikovni element s projicirane koordinate uzme za boju svjetla koje obasjava izlazni slikovni element.

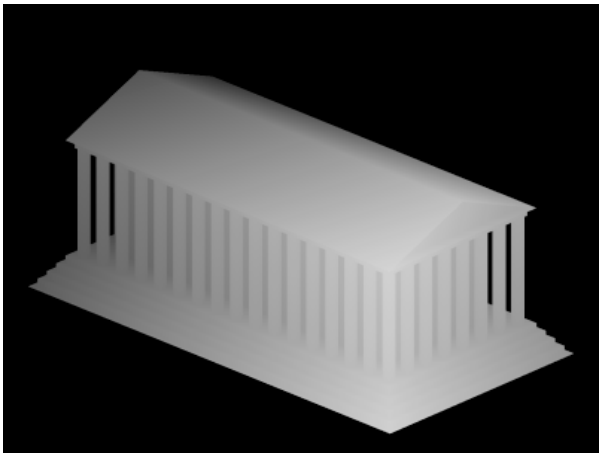
Primjer objekta obasjanim projekcijskom teksturom može se vidjeti na slici 13.

Kada je potrebno animirati tu teksturu, također možemo koristiti video tehnologije.

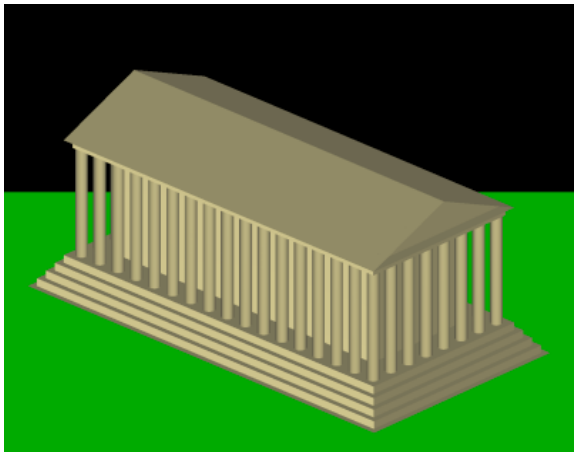
Postoje par situacija u kojima ima smisla animirati projekcijsku teksturu videom, najćešći primjeri su:

1) Kada ispred izvora svjetla stoji objekt koji se animira i time baca sjenu na scenu, npr. ventilator.
2) Kada se pozicija izvora svjetla animira ispred statičnog ili animiranog objekta.
3) Ukoliko želimo projicirati neku sekvencu, npr. simulirajući projektorsku sliku u prostoru.

Ubrzavanje mape sjene



Slika 14: Dubinska mapa iz gledišta svjetla



Slika 15: Scena iz gledišta svjetla

Algoritam mape sjena[31] (eng. Shadow mapping) temelji se na dubinskim teksturama koje se generiraju iscrtavanjem scene iz gledišta izvora svjetla zapisivanjem dubinske komponente slikovnih elemenata u teksturu. Dubinska komponenta označava koliko je određeni slikovni element udaljen od projekcijske plohe.

Nakon generiranja te teksture uz pomoć programa za sjenčanje scena se iscrtava iz gledišta glavne kamere na način da se prilikom računanja osvjetljenja u sceni provjeri pozicija slikovnog elementa u prostoru te odredi da li se nalazi u sjeni ili ne. Tu provjeru radimo na temelju prethodno iscrtane dubinske mape i pozicije svjetla u prostoru.

Primjer scene i dubinske mape iz gledišta izvora svjetla možemo vidjeti na slikama 14 i 15.

Ubrzavanje ovog algoritma je moguće korištenjem unaprijed iscrtane dubinske mape. Druga mogućnost je osvježavanje sadržaja te mape samo kada se promjeni stanje scene vidljive iz gledišta danog izvora svjetla što može uzrokovati promjena stanja objekta u vidnom polju izvora svjetla, ulazak nekog objekta u vidno polje ili promjena pozicije odnosno orijentacije izvora svjetla.

Ukoliko imamo animiranu sekvencu ili svjetlo koje mijenja poziciju, možemo kodirati sve dubinske mape iz gledišta izvora svjetla u video zapis te time ubrzati sam algoritam. Ukoliko imamo nedeterminističku situaciju poput gibanja izvora svjetla gdje je nepoznato hoće li neki objekt ući u kadar ili ne, možemo kombinirati dubinsku mapu dobivenu iz video zapisa s dodatnim iscrtavanjem takvih objekata u tu mapu.

U ovoj primjeni, ključna je kvaliteta slike jer odluka da li je neki slikovni element u sjeni ili nije uvelike ovisi o vrijednosti slikovnih elemenata u dubinskoj mapi.
U ovoj primjeni nam ne ide u korist kompresija na temelju ljudskog vizualnog sustava koju koristi većina video tehnologija pošto nam informacije kodirane video slikom na ovaj način ne reprezentiraju realnu sliku.

Kako bi postigli mekše sjene najjednostavnije je koristiti volumni izvor svjetla umjesto točkastog. U kontekstu algoritma mape sjene to znači korištenje više tekstura za nekoliko uzorkovanih pozicija s plohe izvora svjetla koji se koriste za određivanje u kolikom postotku je određeni slikovni element u sjeni. Tu se radi optimizacije može iskoristiti jedna tekstura s 3 dubinske mape, svaka zapisana u drugi kanal boja teksture. U kontekstu video zapisa, možemo isti efekt postići korištenjem kromatskih komponenti videa, ali trebamo voditi računa da video tehnologija ne smanji dimenziju kromatskih komponenti u tom slučaju.
Alternativa je podjeliti teksturu u više podtekstura, svaka iscrtana iz različitih gledišta uzorkovanih s plohe svjetla. Ta tehnika se također može kodirati u više kanala. Tako podjelom teksture na 2x2x3 podteksture dobivamo prilično lijepe meke sjene.

Preslikavanje neravnina

Tehnika preslikavanja neravnina je trik koji korištenjem programa za sjenčanje daje dojam znatno veće rezolucije geometrije trodimenzionalnog tijela nego što to stvarno jest, optimizirajući time cijeli sustav prikaza slike tog trodimenzionalnog tijela.

Naime, objekt od 200.000 poligona možemo smanjiti na znatno manji broj poligona te fine detalje koji su se degradacijom broja poligona izgubili prebaciti u teksturu koja sadrži vektore normala na površinama poligona tog lika.

Na slici 16 možemo vidjeti primjer kako korištenjem takve tehnike uz pomoć grafičkog sklopovlja dobivamo fine detalje na površini 3D objekta uz manji broj korištenih poligona za prikaz samog 3D objekta.

Slika 16: Doprinos normal mape u prikazu finih detalja na površini 3D lika

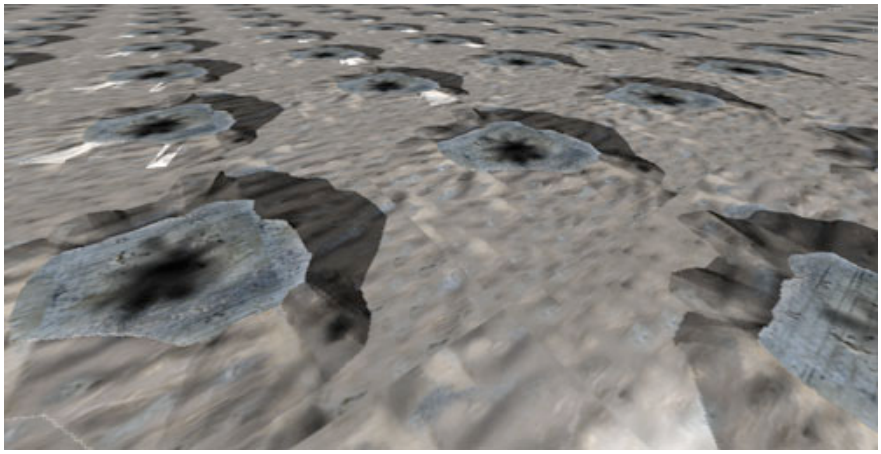
Kako je mapa normala ništa drugo nego RGB tekstura, možemo koristiti video tehnologiju za animiranje takve teksture i time postići razne efekte na površini objekta koji se prikazuje tehnikom preslikavanja neravnina.

Slika 17: Primjer mape normala

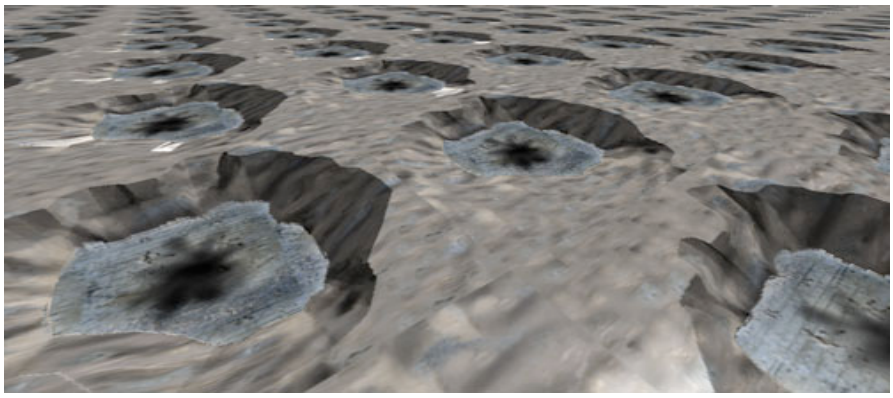
Na slici 17 možemo vidjeti primjer jedne mape normala u RGB zapisu. Treba voditi računa na to da boje u mapi normala nisu realne boje u prirodi, tj. nisu u sustavu boja kakvo ljudsko oko vidi stoga će video tehnologije više degradirati slike nego u ostalim slučajevima. Pogotovo ako se koristi smanjivanje rezolucije kromatskih komponenti. Prilikom korištenja ove tehnike uputno je normalizirati vektore normal mape nakon dekodiranja video slike u samoj teksturi ili u programu za sjenčanje.

Kada je poželjno animirati mapu normala?
Objekti mogu biti statično teksturirani, a animacija rješena geometrijski. U tom slučaju nije potrebna animacija teksture pa tako ni mape normala.
No postoji par slučajeva gdje je takva animacija potrebna. Na primjer ukoliko animiramo izraz lica pomoću teksture, možemo animirati i mapu normala teksture lica.

Preslikavanje neravnina uz učinak paralakse



Slika 18: Preslikavanje neravnina korištenjem normal mape



Slika 19: Preslikavanje neravnina korištenjem tehnike mapiranja paralakse

Mapiranje paralakse (eng. Parallax mapping) je naprednija varijanta tehnike preslikavanja neravnina kod koje se uvodi još jedna tekstura pored mape normala – visinska tekstura. Na temelju tih dvaju tekstura na plohi se difuzna tekstura preslikava tako da daje dojam volumena.

Na slikama 18 i 19 možemo vidjeti vizualnu razliku između tih dvaju tehnika.

U ovom slučaju video tehnologijom možemo kodirati mapu normala ili visinsku mapu ili oboje. Tako na primjer animacijom možemo spuštati ili povisivati određene dijelove teksture.

Slika 20: Par slika iz animirane sekvence mape normala i visinske mape

Na primjer ovako bi se moglo realizirati modeliranje animacije udubine u metalnoj površini. U početku metalna površina je ravna, a postepeno se deformira pod udarcem i raste visinska mapa te se mapa normala mijenja.

Kao ilustraciju ove tehnika razvijen je demo program koji prikazuje animiranu video sekvencu mape normala i visinske mape te ih koristi kao ulaz za razne tehnike preslikavanja neravnina.

Na slici 20 možemo vidjeti par slika iz tog video zapisa. Slično kao i u primjeru za kanal prozirnosti, mapa normala je u svaku video sliku zapisana na lijevoj strani slike a visinska mapa na desnoj. U procesu dekodiranja se te dvije slike razdvoje u dvije zasebne teksture. Razlog spajanja tih dvaju tekstura u jedan video zapis umjesto korištenje dvaju video zapisa leži u sikronizaciju. Jednostavnije je dekodirati jedan zapis nego brinuti o sikronizaciji dva zapisa.

Slika 21: Tehnika mape normala

Slika 22: Tehnika preslikavanja paralakse

Na slikama 21 i 22 možemo vidjeti kako mapa normala i visinska mapa utječu na prikaz demonstracijskog programa. Na slici 21 koristi se lijevi dio video slike u kojem se nalazi mapa normala dok se na slici 22 koriste obje mape kako bi se postigao učinak paralakse što se najbolje može vidjeti oko rubova slova na toj slici koji daju snažniji dojam neravnine nego kod tehnike mapiranja normala.

Volumne teksture

Video tehnologije možemo koristiti za animaciju volumnih tekstura za postizanje raznih efekata poput animacije krutog 3D tijela ili animacija kretanja oblaka.

Volumne teksture se od klasičnih tekstura razlikuju po reprezentaciji slikovnih elementa. Umjesto plošnih slikovnih elemenata, volumne teksture sadrže *voxele*, volumne elemente s bojom koji se mogu na razne načine interpretirati.

Najveći problem volumnih tekstura su memorijski zahtjevi koji rastu s kubnom funkcijom prostorne dimenzije. Tako 256x256x256 volumna tekstura s 24 bitnom bojom zauzima čak 48 MB radne memorije! 1024x1024x1024 tekstura zahtjeva već 3 GB radne memorije i daleko je od praktične primjene.

Ako hoćemo animirati volumne teksture pomoću videa, treba osmisliti sistem kako kodirati treću dimenziju u plošnu video sliku. To se najjednostavnije može napraviti raščlanjivanjem treće dimenzije po segmentima u plošnu teksturu i sastavljanje nakon dekodiranja video slike. Tako će se 64x64x64 volumna tekstura kodirati u video dimenzija 512x512 ili 4096x64, ovisno kako želimo prostorno smjestiti segmente volumne teksture.

Očit nedostatak video tehnologije u ovoj primjeni jest u ograničenju rezolucije volumne teksture koju možemo na ovaj način dekodirati. Već samo 64x64x64 daje visoke dimenzije video slike. Računavši da moramo provesti računski intenzivan proces dekodiranja, prostorne transformacije i transformacije sustava boja, dolazimo brzo do granice u realnim primjenama u okviru suvremenih računalnih mogućnosti. No, svejedno u 64x64x64 ipak dosta toga stane. Možda nije praktično za prikaz 3D volumnog lika, ali za neke efekte poput oblaka moglo bi poslužiti.

Druga mogućnost jest indirektno korištenje video slike u volumnim teksturama tako da se sprema zadnjih N dekodiranih slika u volumnu teksturu i na taj način se koriste za neke efekte zamućivanja (eng. blur).

Video kao sažeta reprezentacija volumne teksture

Volumne teksture, pogotovo one visoke rezolucije, zauzimaju mnogo diskovnog prostora iako se slojevi unutar 2D teksture ne razlikuju puno jedan od drugoga, poput trodimenzionalne slike tijela slikanog medicinskim skenerom. U medicinskim uređajima se javlja potreba za visokom rezolucijom 3D skenova što nekomprimirano daje ogromne datoteke.

Ukoliko komprimiramo svaki sloj nekim algoritmom za kompresiju slika poput PNG ili JPG uštedjeli bi smo nešto prostora, ali ne značajno.

Ukoliko si možemo dopustiti određenu degradaciju slike, možemo svaki sloj komprimirati JPG kompresijom uz određene gubitke, no znatno manju datoteku bi dobili korištenjem video tehnologije.

Iskoristivši činjenicu da se često slojevi volumne teksture znatno ne razlikuju, možemo slojeve volumne teksture pretvoriti u slijed 2D tekstura i komprimirati ih video tehnologijom, dakako uz određene gubitke u kvaliteti.

Rastavivši 1024x1024x1024 volumnu teksturu u 1024x1024 video od 1024 slika, dobivamo video razlučivosti 1024x1024 u trajanju od 41 sekunde (25Hz), što za video tehnologije ne predstavlja nikakav problem.

Postupak dekodiranja volumne teksture kodirane videom se svodi na dekodiranje svih slika iz video zapisa i konstruiranje volumne teksture iz toga. Time naravno nismo uštedjeli na radnoj memoriji, ali smo znatno uštedjeli na diskovnom prostoru.

Ovisno o načinu korištenja volumne teksture, moguće je uštedjeti i na radnoj memoriji, tako da prilikom iscrtavanja informacije iz volumne teksture, koristimo slojeve slijedno. Prolaskom kroz video sliku sloj po sloj u određenom video segmentu potrebnom za iscrtavanje informacije iz volumne teksture u danom trenutku konstruiramo izlaznu sliku. Takav postupak je znatno sporiji za korištenje i ne dopušta nasumično čitanje slikovnih elemenata te time znatno ograničava primjenu.

DALJNJA POBOLJŠANJA

Konačni cilj ovog rada je potaknuti čitatelja na razmišljanje i razvoj vlastitih tehnika korištenja video zapisa u računalnoj grafici.
Vjerojatno ima još mogućnosti za korištenje video tehnologije za razne efekte koji su ograničeni samo granicama ljudske mašte. U ovom radu su opisane odabrane reprezentativne tehnike kako bi čitatelju rasplamsali maštu za korištenjem raznig mogućnosti koje video tehnologije pružaju u kontekstu grafičkih aplikacija.

Moguće je poboljšanje u demonstracijskom programu hibridne metode osvjetljenja. U sadašnjoj verziji zbog ograničenog vremena prikazan je jednostavniji prostor i animacija kamere u tom prostoru. Bolja demonstracija bi bila neka kompleksnija i duža sekvenca s animiranim likovima i okolinom na kojoj bi se onda bolje mogla demonstrirati moć ove tehnike.

Programsko rješenje dano uz ovaj diplomski rad koristi isključivo Theora tehnologiju te bi kao moguće poboljšanje bilo dobro razdvojiti dekoderski podsustav od upravljačkog kako bi se lakše mogla dodati podrška za neku drugu video tehnologiju. Naime, u trenutku kada je originalno programsko rješenje bilo razvijano, Theora je bila tehnologija s dobrom budućnošću no s vremenom su došla i druga rješenja poput Dirac-a i WebM-a koja nude bolje kompromise između resursnih zahtjeva te isto nisu ograničena licencom.
Bitna činjenica jest da sve opisane tehnike nisu ograničene tehnologijom. Potreban je samo mehanizam koji će dohvaćati slike iz video zapisa i dovoditi ih korisniku na daljnju obradu za korištenje u grafičkom efektu.

ZAKLJUČAK

U ovom radu, kroz opis odabranih tehnika, demonstrirane su brojne prednosti korištenja video tehnologija u grafičkim aplikacijama koje rade u stvarnom vremenu, bez obzira da li se radi o uštedi memorijskog prostora ili poboljšanju prikaza slike.

Dok god grafičko sklopovlje ne bude dovoljno brzo da može s lakoćom za bilo koju scenu izračunati realno osvjetljenje u stvarnom vremenu, bit će potrebe za razne aproksimativne tehnike za izračun osvjetljenja, prikaz sjena itd.

Korištenjem video tehnologije u nekim slučajevima možemo smanjiti procesorske zahtjeve te dobiti istu ili bolju kvalitetu izlazne slike. Time smo oslobodili procesorsko vrijeme za računanje dodatnih efekata ili poboljšanje kvalitete postojećih uz održavanje dovoljnog broja slika u sekundi.

Donedavno su postojala dva glavna problema zbog čega se ovakve tehnike nisu više koristile u praksi – licenciranje video tehnologije i procesorski zahtjevi za dekodiranje video slike.

Pojavom open source video tehnologija i povećanjem procesorske moći računala oba problema su u većini primjena ovakvih tehnika danas otklonjena.

Stoga se nadam da će se zbog toga kao i utjecajem ovog rada ovakve tehnike sve više koristiti u praksi.

POJMOVNIK

Kako bi čitatelj manje upućen u pojmove u računalnoj grafici lakše razumio tekst ovog rada, potrebno je objasniti nekoliko korištenih pojmova:

Video – slijed slika koji čini neku cijelinu. U većini slučajeva, pojam video povezuje i druge multimedijske elemente poput zvuka i podnatpisa. Zbog velikih potreba za prostorom za pohranu sirovog video materijala, gotovo uvijek se video slika komprimira raznim video tehnologijama.

2D Tekstura – dvodimenzionalni zapis slikovnih elemenata u obliku tablice. Slikovni elementi mogu biti reprezentirani raznim sustavima boja, od kojih je najčešći RGB. Pored informacije o boji, teksture mogu također sadržavati informaciju o prozirnosti svakog slikovnog elementa što zahtjeva još jedan kanal boje u teksturi.
Memorijski prostor za pohranu nekog slikovnog elementa ili grupe slikovnih elemenata ovisi o načinu kodiranja. Najčešće se svaki kanal kodira s 8 bitova što rezultira u 3 (RGB) odnosno 4 (RGBA) bajta po slikovnom elementu. Postoje mnogi drugi načini kodiranja poput RGB565, blokovskih kompresijskih metoda no one nisu bitne za ovaj rad. Takve teksture se preslikavaju na poligone u prostoru na način da se svakom vrhu poligona pridjeli određena koordinata u teksturnom prostoru i prilikom iscrtavanja tog poligona se razapnu slikovni elementi teksture unutar koordinata na vrhovima tog poligona.

Volumna (3D) tekstura – trodimenzionalna matrica koja sadrži slikovne elemente. Može se predočiti kao N 2D tekstura naslaganih jedna na drugu. Koristi se za razne vizualizacije i napredne efekte. Ovisno o interpretaciji, može reprezentirati ulaz za neki efekt ili diskretiziranu volumnu reprezentaciju nekog objekta, što se na primjer dosta koristi u medicinskim skenerima. Prilikom korištenja na poligonima, vrhovima poligona se dodaje treća dimenzija za teksturne koordinate te se prilikom preslikavanja razapinje trokut unutar prostora teksture, interpolirajući slikovne elemente kroz sve tri dimenzije teksture.

Voxel – nedjeljivi trodimenzionalni kvant prostora. 3D sustavi građeni od voxela najtočnije simuliraju stvarni svijet koji je sastavljen od atoma. No zbog ogromnih zahtjeva za memorijskim prostorom koji takav prikaz prostora traži, još uvijek dominira poligonalna reprezentacija prostora.

Čestični sustavi – sustavi koji upravljaju česticama koje se na različite načine prikazuju na ekranu. Najčešće je riječ o plošno projiciranim teksturama u prostoru. Sustav upravlja skupom čestica koje se ponašaju na određen način simulirajući time razne efekte koje bi bilo teško ili nemoguće riješiti na drugi način. Najbolji primjer je primjena fizikalnih zakona poput gravitacije na sustav čestica čime možemo simulati kišu, vatromet, kretanje oblaka i slične efekte.

Postupak isijavanja – (eng. Radiosity method) algoritam za izračun realnog osvjetljenja u poligonalnim prostorima. Svaka ploha u prostoru se podijeli u određen broj podploha (najčešće proporcionalno dimenzijama plohe). Svjetlost se iz izvora svjetla propagira na podplohe po fizikalnim zakonima, pazeći pritom da se ne osvjetljavaju podplohe čija je površina zaklonjena nekim predmetom na putu do izvora svjetla.
Nakon što se svjetlost distribuira, svaka osvjetljena podploha postaje izvor svjetla i kao takav reflektira određenu količinu svjetla obasjavajući okolne podplohe. Time se postiže vrlo realistično osvjetljenje. Nedostatak ovog algoritma je visoka složenost izračuna i veliki broj iteracija potreban za dobar izgled osvjetljenja.
Nakon izračuna osvjetljenja, ta informacija se može pohraniti u svjetlosnu teksturu i koristiti kao statično osvjetljenje na sceni. Moguća je i animacija takvog svjetla što je prikazano u tekstu ovog rada.

SAŽETAK

U ovom radu su prikazane odabrane jednostavne i napredne tehnike korištenja video slike u računalnoj grafici.
Cilj ovog rada je pokazati kako upotrebom video tehnologije povećati kvalitetu izlazne slike uz minimalni utrošak računskih i prostornih resursa.
Tehnike su odabrane s ciljem da čitatelju najbolje predoče mogućnosti ovakvih tehnika i potaknu na razmišljanje o prilagodbi opisanih tehnika za vlastite potrebe i razvijanje novih tehnika koje koriste video tehnologije.
Za potrebe demonstracije tih tehnika razvijeno je programsko rješenje koje koristi Theora video tehnologiju te nekoliko demo programa koji ilustriraju neke opisane tehnike.

Ključne riječi

Theora, Video, grafika, OpenGL, metode, tehnike, kompozitna animacija, osvjetljenje, volumne teksture, sjene, refleksija

ABSTRACT

This paper describes a number of simple and advanced techniques that use video in computer graphics.

The goal of this paper is to show the reader how using video technologies one can increase the level of image quality with minimal cost to computational and spatial resources.

The techniques were carefully chosen to best demonstrate the possibilities of using such techniques and inspire the reader to consider adjusting them to his own needs and even develop new techniques that use video technologies.

For demonstrative purposes, a software solution was developed that uses Theora video technology as well as several demonstration programs that illustrate some of the described techniques.

Keywords

Theora, Video, graphics, OpenGL, methods, techniques, composite animation lighting, volumetric textures, shadows, reflection

LITERATURA

[1] - **"Obrada i prikaz video zapisa na teksturama objekta", Krešimir Špes, FER 2010.**
http://www.zemris.fer.hr/predmeti/irg/Zavrsni/09Spes/index.html

[2] - **Binarni algoritam pretraživanja**
http://en.wikipedia.org/wiki/Binary_search_algorithm

[3] - **Fourierova analiza**
http://en.wikipedia.org/wiki/Fourier_analysis

[4] - **Diskretna kosinusna transformacija**
http://en.wikipedia.org/wiki/Discrete_cosine_transform

[5] - **YUV sustav boja**
http://en.wikipedia.org/wiki/YUV

[6] - **YCbCr sustav boja**
http://en.wikipedia.org/wiki/YCbCr

[7] - **RGB sustav boja**
http://en.wikipedia.org/wiki/RGB_color_model

[8] - **Theora video tehnologija**
http://theora.org/

[9] - **WebM video tehnologija**
http://www.webmproject.org/

[10] - **Dirac video tehnologija**
http://diracvideo.org/

[11] - **FFmpeg projek**t
http://ffmpeg.org/

[12] - **H.264 video tehnologija**
http://en.wikipedia.org/wiki/H.264/MPEG-4_AVC

[13] - **MMX procesorske ekstenzije**
http://en.wikipedia.org/wiki/MMX_(instruction_set)

[14] - **SSE procesorske ekstenzije**
http://en.wikipedia.org/wiki/Streaming_SIMD_Extensions

[15] - **MPEG4 video tehnologija**
http://en.wikipedia.org/wiki/MPEG-4

[16] - **Theora programsko sučelje i dokumentacija**
http://theora.org/doc/

[17] - **DirectShow multimedijsko programsko sučelje**
http://en.wikipedia.org/wiki/DirectShow

[18] - **GStreamer multimedijsko programsko sučelje**
http://gstreamer.freedesktop.org/

[19] - **Cube map tehnika**
http://en.wikipedia.org/wiki/Cube_mapping

[20] - **Legend of Crystal Valley - računalna igra hrvatskog razvojnog studija Cateia Games**
http://locv.cateia.com/

[21] - **Cateia Games - hrvatski studio za razvoj računalnih igara**
http://www.cateia.com/

[22] - **PNG - standardizirani format za zapis slike**
http://hr.wikipedia.org/wiki/PNG

[23] - **JPG - standardizirani format za zapis fotografija s visokom razinom kompresije**
http://en.wikipedia.org/wiki/JPEG

[24] - **Algoritam animiranja likova na sceni pomoću virtualnih kostura**
http://en.wikipedia.org/wiki/Skeletal_animation

[25] - **Apple iPad tablet uređaj**
http://www.apple.com/ipad/

[26] - **Algoritam praćenja zrake svjetlosti (Ray Tracing)**
http://en.wikipedia.org/wiki/Ray_tracing_(graphics)

[27] - **Radiosity algoritam za izračun realističnog osvjetljenja**
http://en.wikipedia.org/wiki/Radiosity_(computer_graphics)

[28] - **Grafička protočna arhitektura "Fixed function pipeline"**
http://www.opengl.org/wiki/Fixed_Function_Pipeline

[29] - **Unreal Engine**
http://www.unrealengine.com/

[30] - **Algoritam volumnih sjena**
http://en.wikipedia.org/wiki/Shadow_volume

[31] - **Algoritam teksturnog mapiranja sjena**
http://en.wikipedia.org/wiki/Shadow_mapping

[32] - **MSU video tehnologija koja ne degradira kvalitetu slike**
http://compression.ru/video/ls-codec/index_en.html

[33] - **OpenGL programsko sučelje za upravljanje grafičkim sklopovljem**
http://www.opengl.org/

[34] - **Integrating Stereoscopic Video in 3D Games,** Jonas Schild, Sven Seele, Maic Masuch, Ifip International Federation For Information Processing, Stranice: 124-135

TEHNIČKA DOKUMENTACIJA

Programska implementacija korištena u sklopu ovog rada je objavljena pod licencom otvorenog koda. U sklopu tog projekta implementirani su svi demo programi opisani o ovom radu.
Izvorni kod se može skinuti na stranici diplomskog rada ili direktno na stranicama projekta na sourceforge.net-u.

Link: http://libtheoraplayer.sourceforge.net/

Projekt je pisan u C++-u i koristi Ogg, Vorbis i Theora biblioteke.
Za uspješno prevođenje projekta potrebno je skinuti unaprijed prevedene verzije tih biblioteka s Interneta ili ih prevesti iz njihovih izvornih kodova. Nakon toga potrebno je postaviti lokacije za header i lib datoteke tih projekata u sustavu za prevođenje kojeg koristite.
Detaljnije upute za kompajliranje projekta možete naći na danoj web stranici projekta.
Radi jednostavnosti i prenosivosti, svi demo programi su pisani također u C++-u te koriste OpenGL[33] grafičko sučelje pošto je OpenGL dostupan na skoro svim platformama.
U trenutku pisanja ovog rada, projekt se može kompajlirati i izvesti na Windows, Linux, Mac i iOS platformama. Podrška za Android platformu je u planu.

Dokumentacija programskog sučelja je dana na sljedećoj adresi:
http://libtheoraplayer.sourceforge.net/api/html/index.html
Sav izvorni kod je dizajniran po strogim objektno orijentiranim pravilima i pisan tako da se može izvoditi višedretveno.