

# Virtualna okruženja

Igor S. Pandžić, Krešimir Matković, Mirko Sužnjević

Uvod

# Dobro došli na prvo predavanje

- O kolegiju
  - Teme i ciljevi kolegija
  - Praktična organizacija, ocjenjivanje
- Osnove računalne grafike
  - Virtualna scena, modeliranje
  - Isrtavanje, protočni sustav

# Virtualna okruženja

- Predavači
  - Prof. dr. sc. Igor S. Pandžić
  - Prof. dr. sc. Krešimir Matković
  - Doc. dr. sc. Mirko Sužnjević
- Asistenti
  - Sara Vlahović, mag. ing.
  - Kuzma Mustač, mag. ing.
  - Lovro Boban, mag. ing.
- Zavod na telekomunikacije, C zgrada 7/8. kat
- Web: [http://www.fer.hr/predmet/virokr\\_c](http://www.fer.hr/predmet/virokr_c)

# Konzultacije

- Nastavnici
  - Pitanja u vezi gradiva, rješavanje eventualnih sporova u vezi laboratorija i bodovanja ispita
  - Obratiti se nastavniku tek ako od asistenta niste dobili zadovoljavajući odgovor
  - Tjedni termin konzultacije: neposredno iza predavanja
- Asistenti
  - Pitanja u vezi laboratorija, projekta, bodovanja ispita
  - Tjedni termin konzultacije: utorak 14-16h uz najavu mailom
  - Zavod za telekomunikacije, ured C07-16 ili *online* (Microsoft Teams)
- Demonstratori
  - Konzultacije u vezi laboratorijskih vježbi (e-mailom i po dogovoru)
  - Na raspolaganju je laboratorij C08-22
    - Za rezervaciju termina javiti se asistentima

# Konzultacije

- Što se može riješiti e-mailom, riješite e-mailom!
  - Asistenti: vo@fer.hr, ime.prezime@fer.hr
  - Nastavnici: ime.prezime@fer.hr

# Predavanja

- Uvod (osnove računalne grafike)
- Grafički procesor
- Specijalni efekti
- Ubrzavanje iscrtavanja
- Umrežena virtualna okruženja
- Virtualna stvarnost
- Proširena stvarnost
- Virtualni ljudi
- Vizualizacija (prof.dr.sc. Krešimir Matković)
- Virtualna produkcija – pozvano predavanje tvrtka Stype

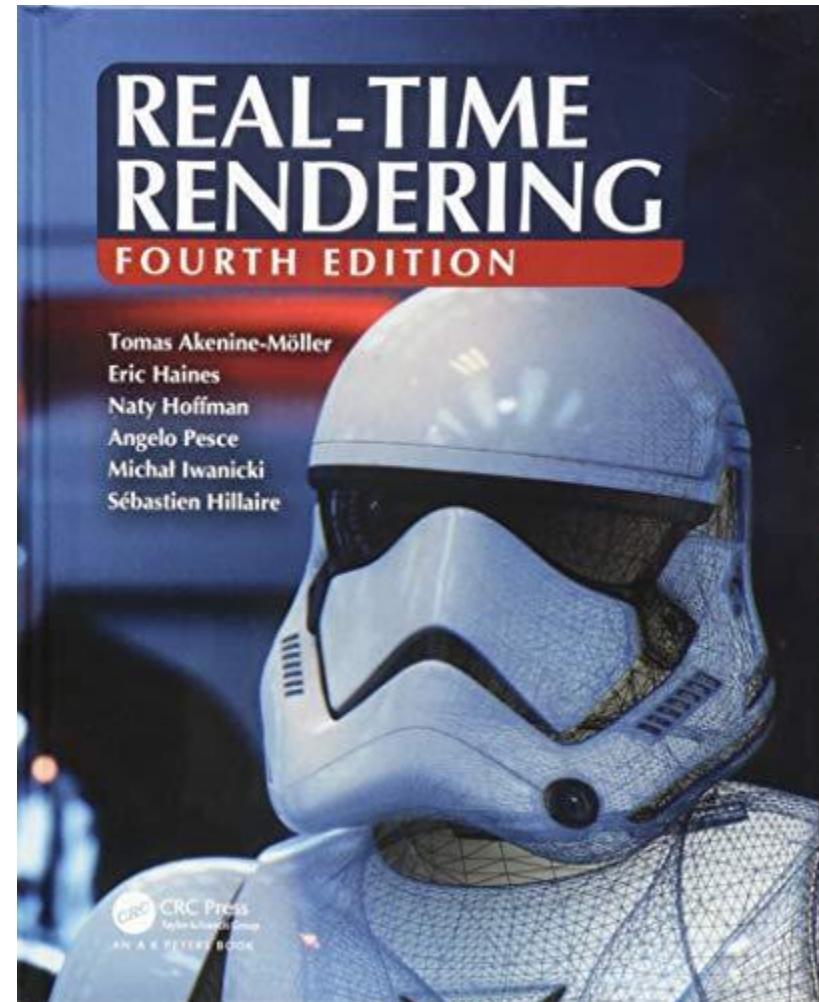
# Udžbenik

"Virtualna okruženja:  
Interaktivna 3D grafika i  
njene primjene", Igor S.  
Pandžić, Tomislav Pejša,  
Krešimir Matković, Hrvoje  
Benko, Aleksandra  
Čereković, Maja Matijašević;  
Element Zagreb 2011,  
Manualia Universitatis  
Studiorum Zagabiensis



# Dodatna literatura

- “Real-Time Rendering”,  
Tomas Akenine-Möller, Eric  
Haines, Naty Hoffman, A.K.  
Peters Ltd., 4th edition,  
ISBN 978-1138627000, 2019
- Resources:  
[www.realtimerendering.com](http://www.realtimerendering.com)



# Laboratorijske vježbe (sati, okvirno)

- Specijalni efekti (3)
- Ubrzavanje iscrtavanja (3)
- Umrežena virtualna okruženja (3)
- Proširena stvarnost (3)
- Virtualni ljudi (3)

# Napredne aktivnosti u okviru lab. vježbi

- Napredni zadaci ponuđeni u većini vježbi
  - Poticaj za studente koji žele više
  - Mogu donijeti bodove za aktivnost prema procjeni asistenta
- Sudjelovanje u izradi materijala
  - Za posebno motivirane studente
  - Bodovi za aktivnost

# Upute i materijali za lab. vježbe

- Upute za izvođenje vježbe, kao i svi materijali za samostalno izvođenje vježbe nalaze se na web stranici predmeta najkasnije tjedan dana prije početka svake vježbe
- Upute sadrže točan opis traženih rezultata vježbe
- Vježbe se rade samostalno, bez dolaska u laboratorij
- Demonstratori raspoloživi e-mailom i po dogovoru
- Na raspolaganju je laboratorij C08-22
  - Za rezervaciju termina javiti se asistentima

# Raspored predavanja i laboratorija

- Raspored predavanja i preporučeni raspored vježbi na stranici predmeta
- Napomena:
  - *Pregledati i prodiskutirati sa studentima na predavanju*

# Predaja i ocjenjivanje vježbi

- Rezultati vježbi i izvještaji o izvođenju se predaju korištenjem aplikacije Moodle
- Nema roka za predaju pojedine vježbe
  - U rasporedu je preporučen optimalni ritam rada
  - Molimo vas da predajete redovito radi opterećenja asistenata
- **Paziti na krajnji rok za predaju svih vježbi**
  - Vježbe koje do tada nisu predane nose 0 bodova
  - Nema nadoknada
- Ocjenu za svaku laboratorijsku vježbu formira asistent na temelju rezultata vježbi i izvještaja

# Demonstratori

- Potrebno je pet demonstratora: po jedan demonstrator za svaku vježbu
  - Demonstrator/-ica „specijalizira“ jednu vježbu i na raspolaganju je studentima za konzultacije oko „njegove/njene“ vježbe (e-mailom i po dogovoru)
    - Na raspolaganju je laboratorij C08-22
    - Ostale vježbe radi kao i ostali
    - Za napredne: moguća suradnja u unapređivanju materijala
- Privilegije demonstratora
  - Bonus tri boda za sudjelovanje u nastavi
- Pripreme za demonstratore prema dogovoru s asistentom
- Prijave se šalju na [vo@fer.hr](mailto:vo@fer.hr) do **12.10. u 16h**

# Projekt

- Rad u grupi
- Tema po želji, vezano uz predmet
- Prezentacija na predavanju
- Fakultativno, ali **preporučeno**
- Upute na stranici predmeta
  - *Pregledati i prodiskutirati sa studentima na predavanju*

# Ocjenvivanje – ukupna ocjena predmeta

- Detaljno opisano u VO-ocjenjivanje.pdf na web-u predmeta i u izvedbenom planu predmeta
  - *Zajednički pročitati i prokomentirati sve detalje na predavanju*

# Preporuke za (među)ispite

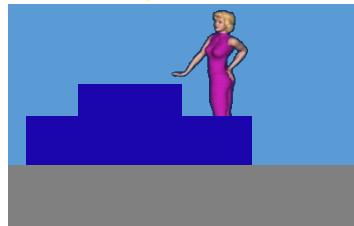
- Predavanja i knjiga su osnova uspjeha ☺
- Materijali s predavanja (slajdovi) su na web-u
  - Namijenjeni su pisanju bilježaka tokom predavanja
  - **NEMOJTE SPREMATI ISPIT SAMO PO SLAJDOVIMA**



Pitanja?

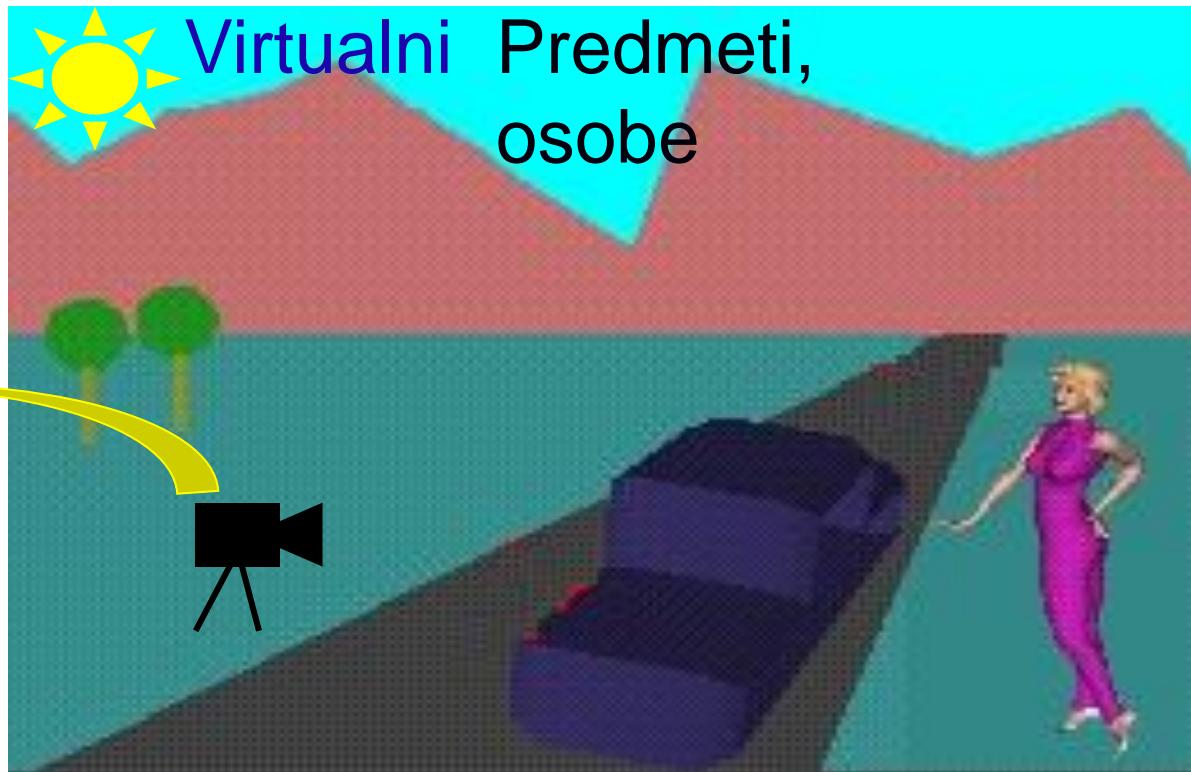
# Virtualna scena

Virtualno Svjetlo



Stvarna Slika

Virtualni Materijali

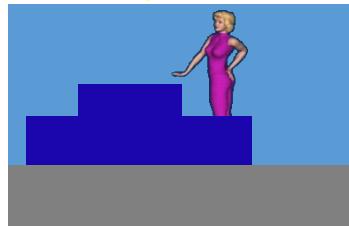


Virtualna Kamera

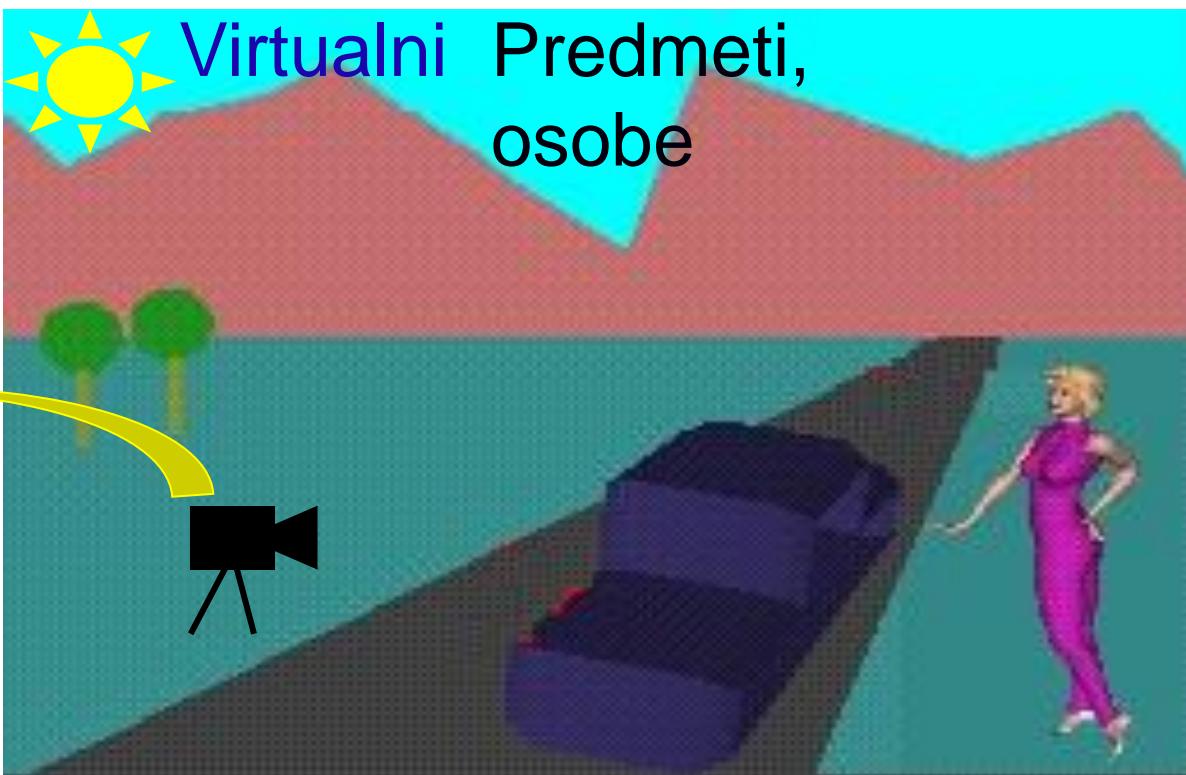
# Virtualna scena

Virtualno Svjetlo

**RENDERING**



Stvarna Slika



Virtualni Materijali

Virtualni Predmeti,  
osobe

# Uvod u 3D grafiku I: Modeliranje

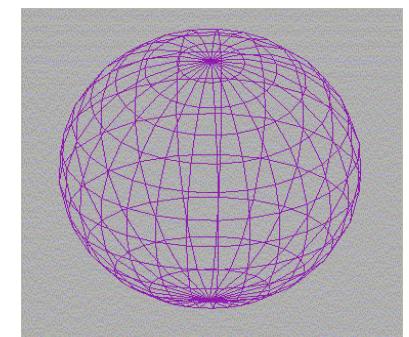
- Modeliranje i digitalni prikaz predmeta
- Model kamere
- Model osvjetljenja
  - Model izvora svjetlosti
  - Model odbijanja svjetlosti
  - Model materijala

# Modeliranje i digitalni prikaz predmeta

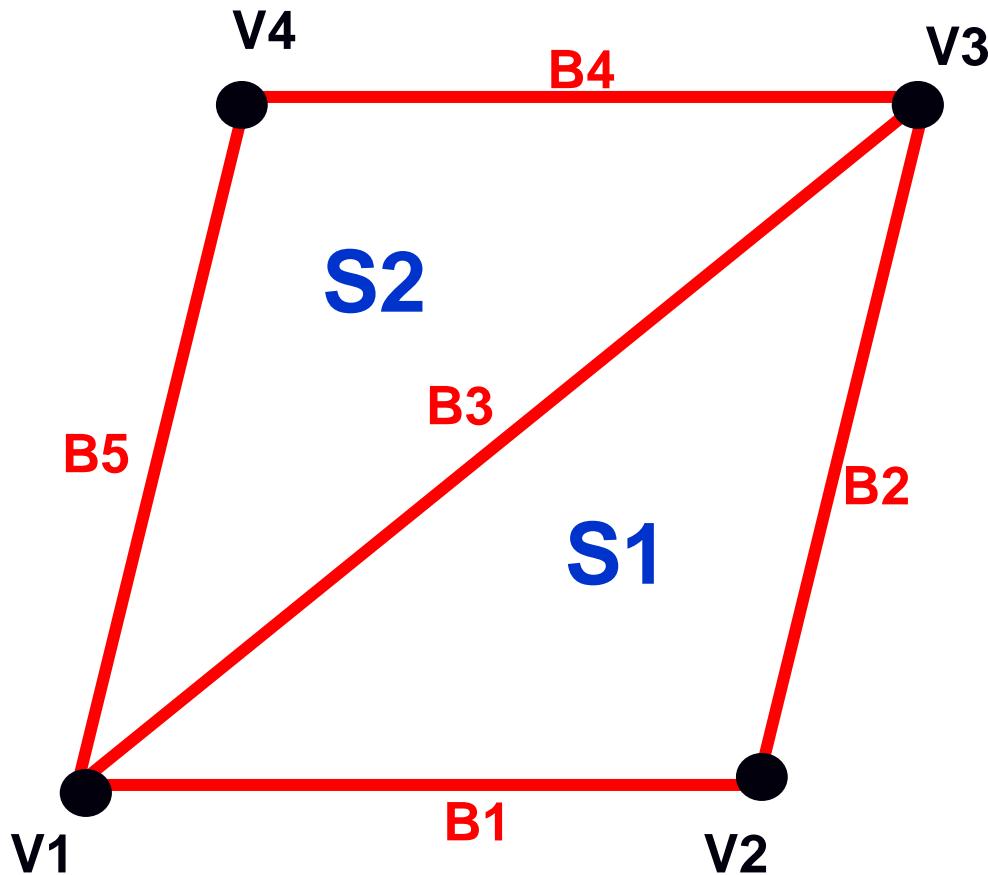
- Poligoni
- Konstruktivna geometrija čvrstih tijela (CSG)
- Parametarske krivulje i plohe
- Razdjelne plohe
- Brišuće plohe
- Volumenske reprezentacije
- Fraktali
- Sustavi čestica

# Prikaz geometrije poligona (1/2)

- Najčešći pristup
- Vrlo općenit pristup: sve se može pretvoriti u poligone
- Aproksimativna metoda
- Nije intuitivno za ručno modeliranje
- Koristi se za interno spremanje podataka
- Često se drugi oblici prikaza pretvaraju u poligone u zadnji čas prije prikaza
- Grafičko sklopovlje prilagođeno za rad s poligonima (najčešće trokutima)



# Prikaz geometrije poligonima (2/2)

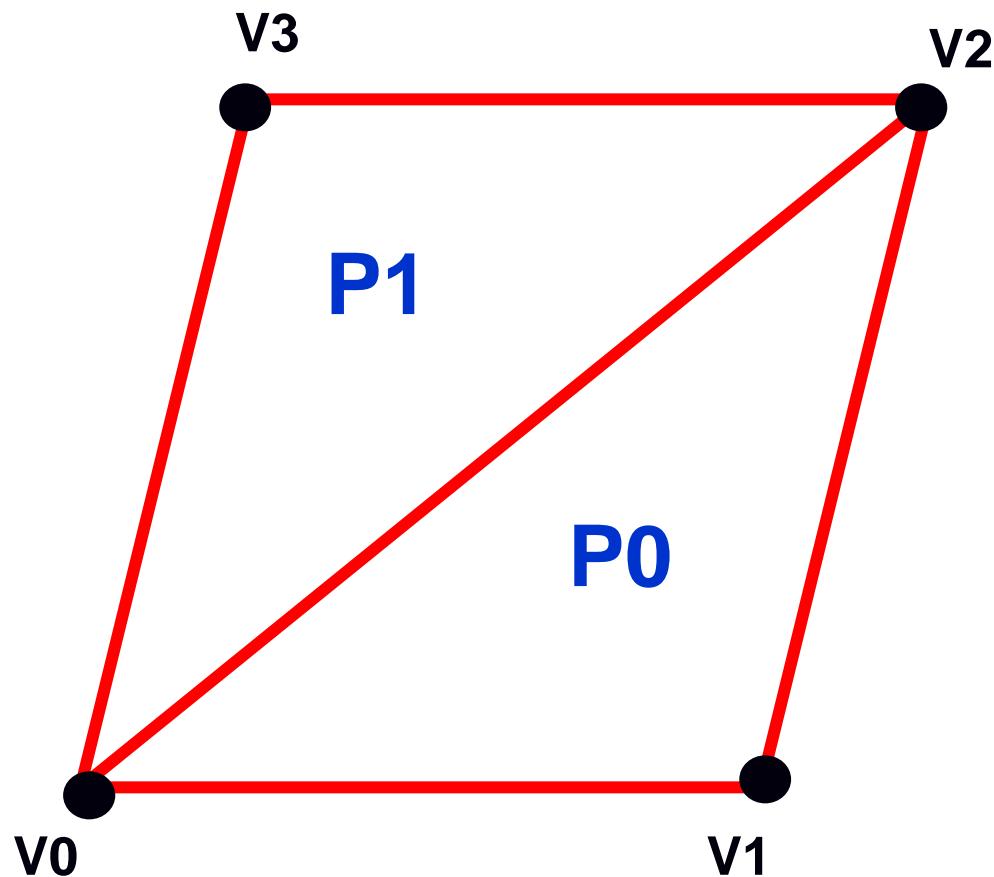


- Vrh – brid – stranica
- $B_1 = V_1, V_2$
- $B_5 = V_4, V_1$
- $S_1 = V_1, V_2, V_3$
- $S_2 = V_1, V_3, V_4$

# Mreža poligona

- Polygon mesh, indexed face set
- Uz manje razlike, gotovo svi formati i programska sučelja koriste ovakvu osnovnu strukturu
- Definiraju se vrhovi i poligoni, te eventualno normale, koordinate teksture i boje

# Zapis vrhova i trokuta

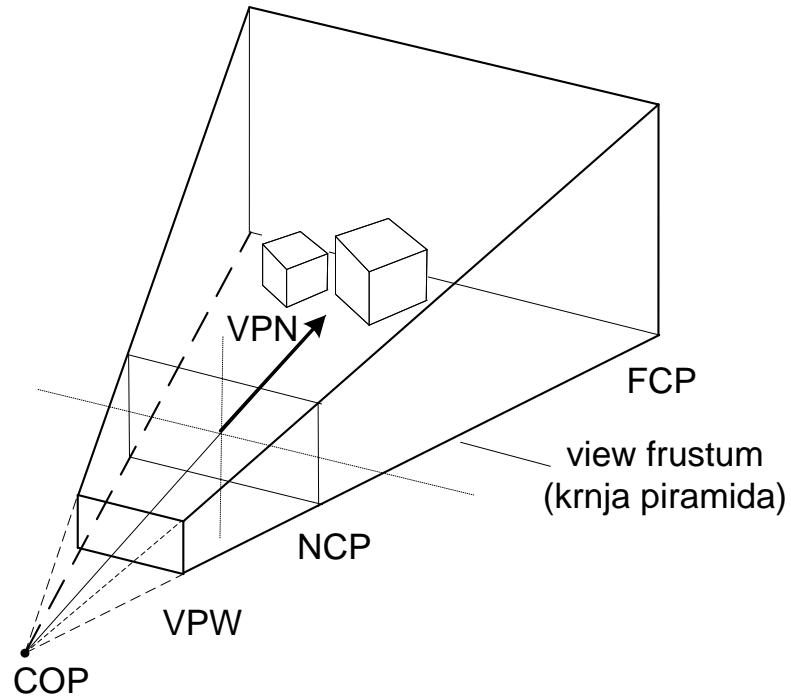


BROJ VRHA	KOORDINATE
0	x y z
1	x y z
2	x y z
3	x y z

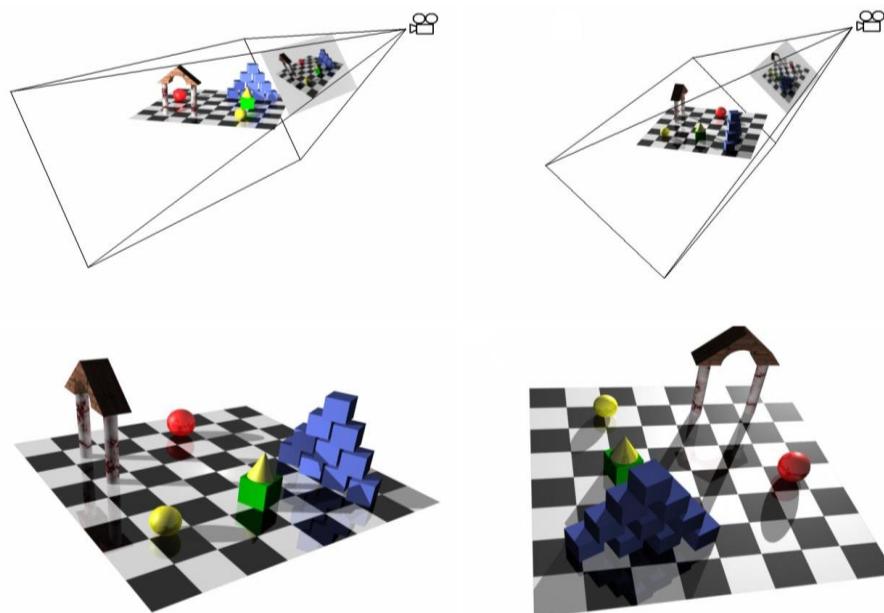
#T	INDEKS VRHA
0	0 1 2
1	0 2 3

# Model kamere

- Određuje pogled u scenu koji će se iscrtati

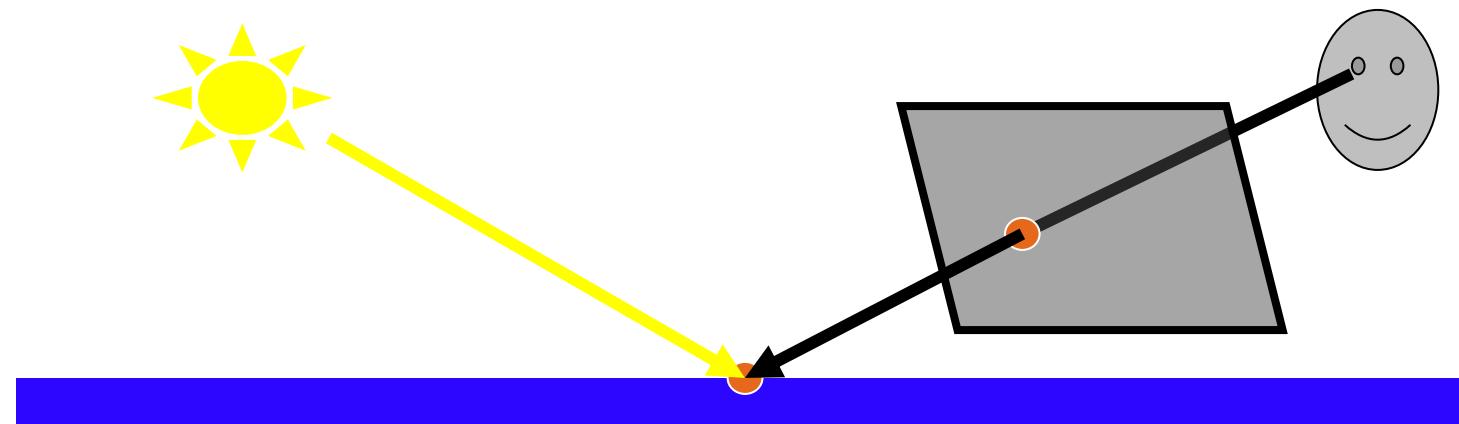


COP - centar projekcije (engl. center of projection)  
NCP/FCP - bliska i daleka odrežujuća ploha (engl. near/far clipping plane)  
VPW - projekcioni prozor (engl. view-plane window)  
VPN - normala na projekcionu plohu (engl. view-plane normal)



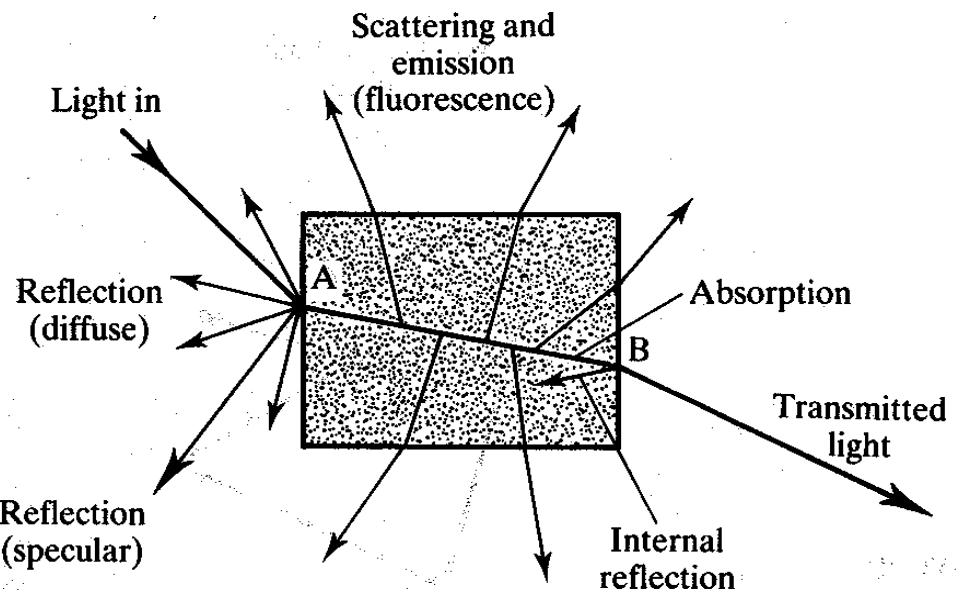
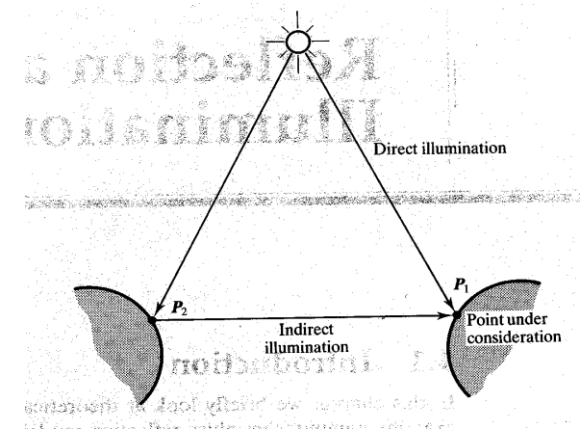
# Model osvjetljenja

- Kada predmet iz virtualne scene projiciramo na ekran, koje boje će biti svaka točka?
- Ovisi o:
  - Materijalu predmeta
  - Svjetlima
  - Relativnim položajima kamere, svjetla i predmeta



# U stvarnosti...

- Svjetlosna tijela
- Globalno osvjetljenje
  - Mekane granice svjetla/sjene
  - Razlijevanje boje (color bleeding)
  - Odrazi
- Odbijanje, lom...



# Na računalu

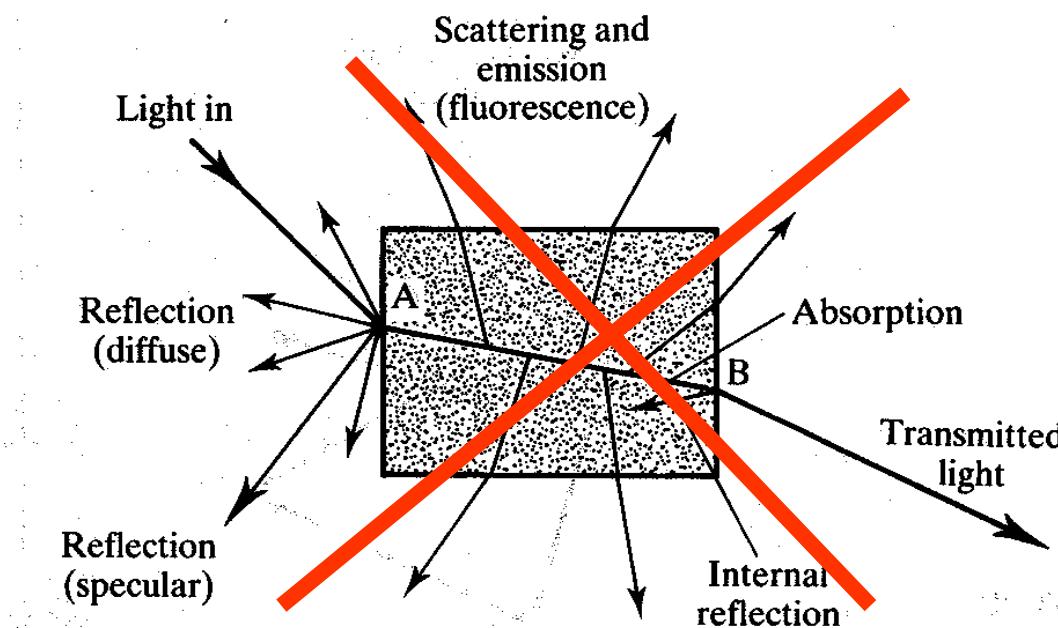
- Za većinu svjetlosnih efekata postoje algoritmi, no često su skupi
- Za realno vrijeme uvijek aproksimacije
  - Jednostavni modeli izvora svjetlosti
  - Zanemarivanje (dijela) globalnih efekata
  - Zanemarivanje dijela lokalnih efekata

# Modeli izvora svjetlosti

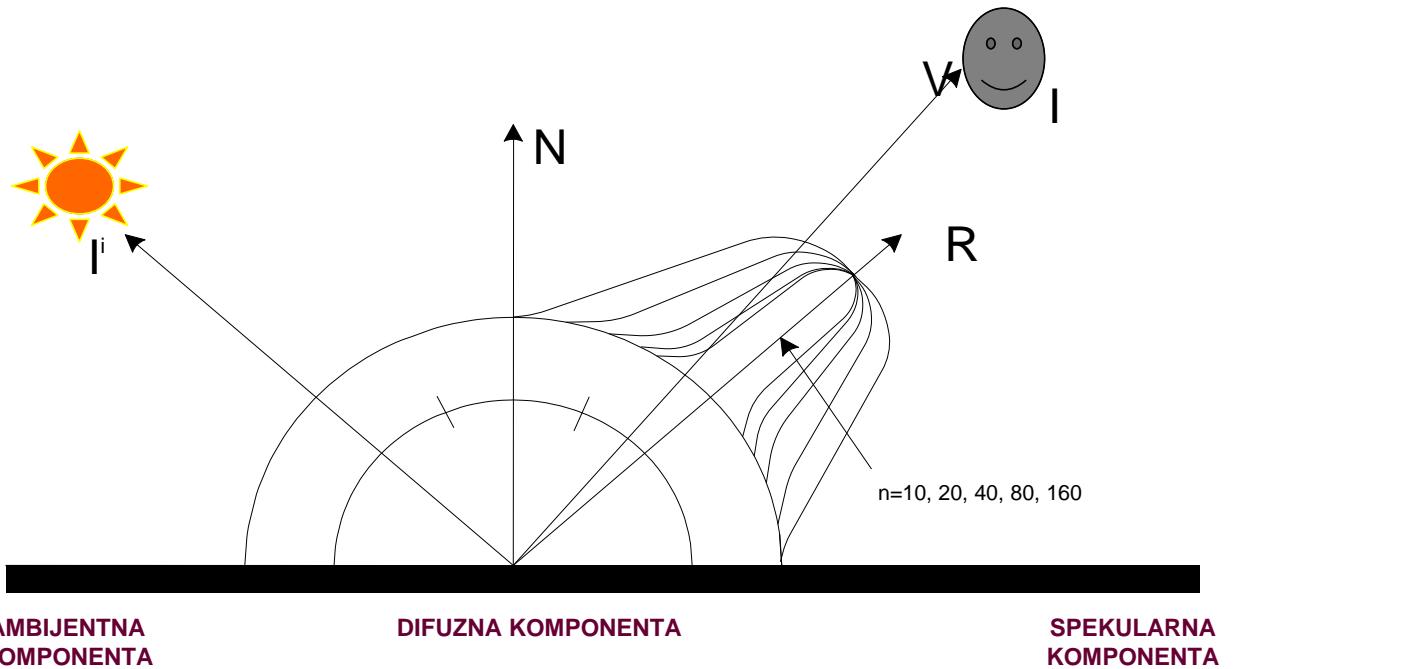
- Ambijentno svjetlo
  - Prisutno svuda u sceni
  - Grubo aproksimira globalno osvjetljenje
- Usmjereno svjetlo
- Točkasto svjetlo
- Reflektor

# Model odbijanja svjetlosti (Phong)

- Bui-Tuong Phong, 1975.
- Najčešći model za RT grafiku
- Jednostavan, ali dobra aproksimacija



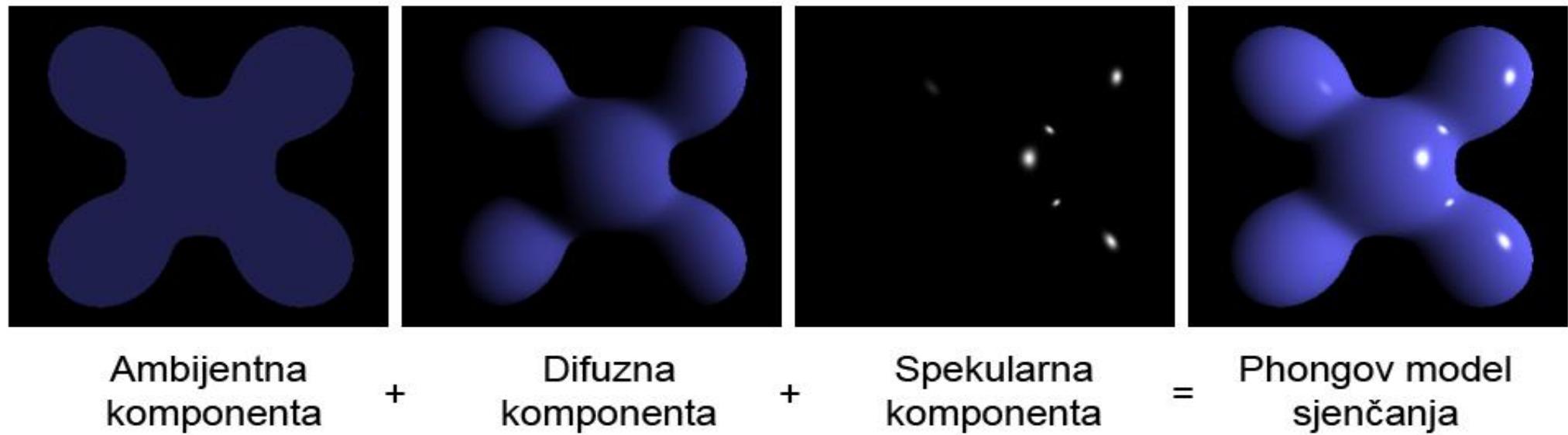
# Phongov model odbijanja svjetlosti (1/2)



$$I = I_a k_a + I_i k_d (L \bullet N) + I_i k_s (R \bullet V)^n$$

KOEFICIJENTI MATERIJALA

# Phongov model odbijanja svjetlosti (2/2)



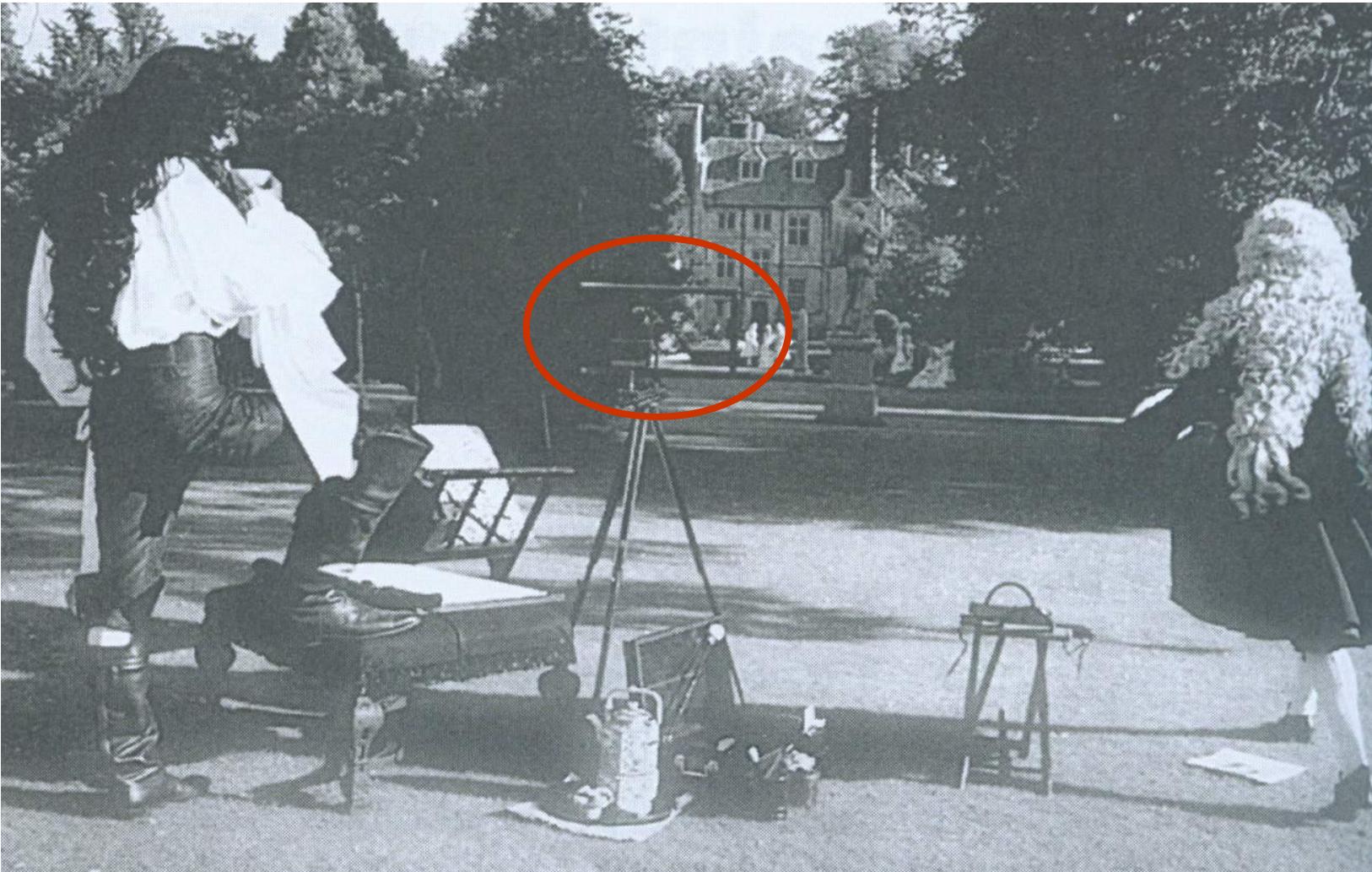
# Model materijala

- Opisuje kako materijal odbija svjetlost
- Dobra vijest: ovo već znate! Osnovni model materijala je sadržan u modelu odbijanja svjetlosti
  - Koeficijenti odbijanja ambijentne, difuzne i spekularne komponente  $k_a$ ,  $k_d$  i  $k_s$
  - Spekularni faktor  $n$
  - Faktor prozirnosti, ako se ona simulira

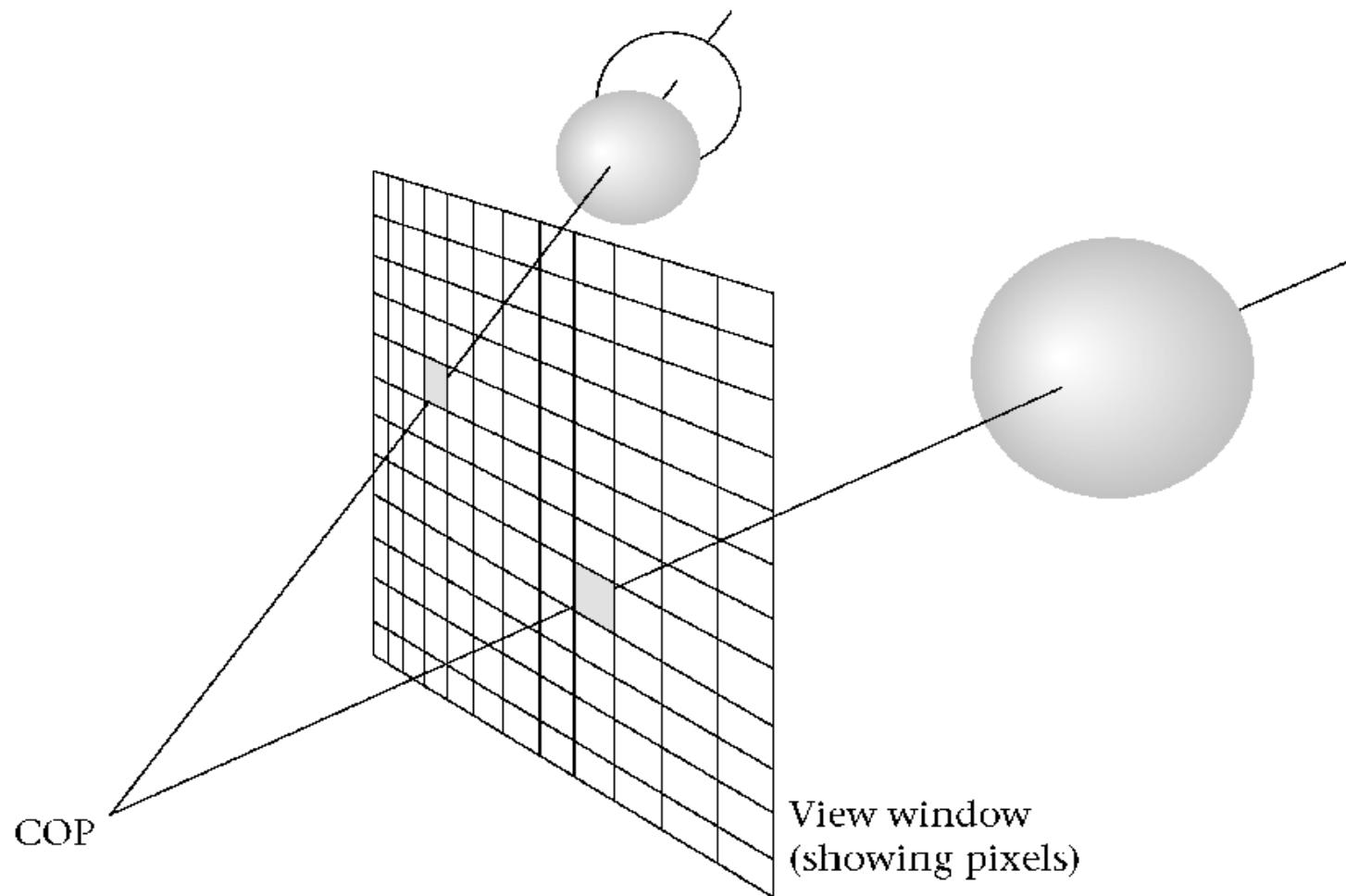
# Uvod u 3D grafiku II: IsCRTavanje

- Grafički protočni sustav u realnom vremenu
  - Aplikacijska faza
  - Geometrijska faza
  - Faza rasteriziranja
- Teksturiranje
- Grafičko sklopolje: spremnici

# Kada slikari „varaju”...



# Projekcija



# Grafički protočni sustav u stvarnom vremenu

- Engl. Graphics Rendering Pipeline
- Niz funkcija koje se izvode jedna za drugom, a koje virtualnu scenu pretvaraju u sliku
- Funkcije koje se izvode istovremeno, kao na pokretnoj traci; najsporija funkcija usko grlo
- Dio funkcija izveden u sklopolju
- Neke funkcije se mogu paralelizirati
- Optimizirano za rad s trokutima

# Osnovne faze grafičkog protočnog sustava

- Aplikacijska faza
  - Operacije specifične za aplikaciju; priprema primitiva (trokuta)
- Geometrijska faza
  - Transformacije, projekcije, osvjetljenje: što, gdje i kako iscrtati
- Faza rasteriziranja
  - Priprema trokuta i pronalazak točaka u pojedinom trokutu
- Faza procesiranja točaka
  - Konačno iscrtavanje, tj. „punjenje“ točaka
- Osnovne faze dijele se na funkcijeske faze
- U implementaciji, funkcijeske faze se preslikavaju u implementacijske faze, ne nužno jedan na jedan



# Aplikacijska faza (1/2)

- „Puni” ostatak protočnog sustava elementima za iscrtavanje u odgovarajućem obliku
  - To su najčešće trokuti, točke i linije
  - Ovo je osnovni zadatak aplikacijske faze
- Sve „pametne” stvari se rade ovdje:
  - Logika same aplikacije
  - Animacija
  - Simulacija
  - Ulaz/izlaz
  - Detekcija sudara
  - Ostalo...

# Aplikacijska faza (2/2)

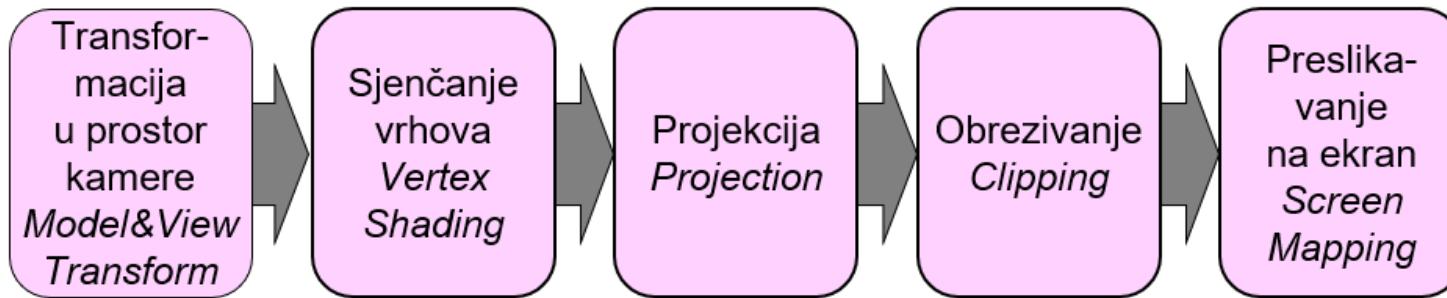
- Uvijek izvedena programski: najveća kontrola
- Najčešće nema podfaza (osim možda korištenjem više procesora)
- Može tako utjecati na ukupnu brzinu iscrtavanja
  - Npr. odabirom trokuta za iscrtavanje (engl. *culling*)

# Geometrijska faza

- Ulaz: 3D poligoni, svjetla, kamera
- Izlaz: 2D poligoni u ekranskim koordinatama i sa određenim bojama
- Zahtjevna faza: oko 100 FP operacija po vrhu u klasičnim izvedbama
- Izvedba najčešće sklopovska, na grafičkom procesu (engl. Graphics Processing Unit, GPU)

# Geometrijska faza: funkcije faze

- Transformacija u prostor kamere (engl. model & view transform)
- Sjenčanje vrhova (engl. vertex shading)
- Projekcija (engl. projection)
- Obrezivanje (engl. clipping)
- Preslikavanje na ekran (engl. screen mapping)



# Tipični ulaz u geometrijsku fazu

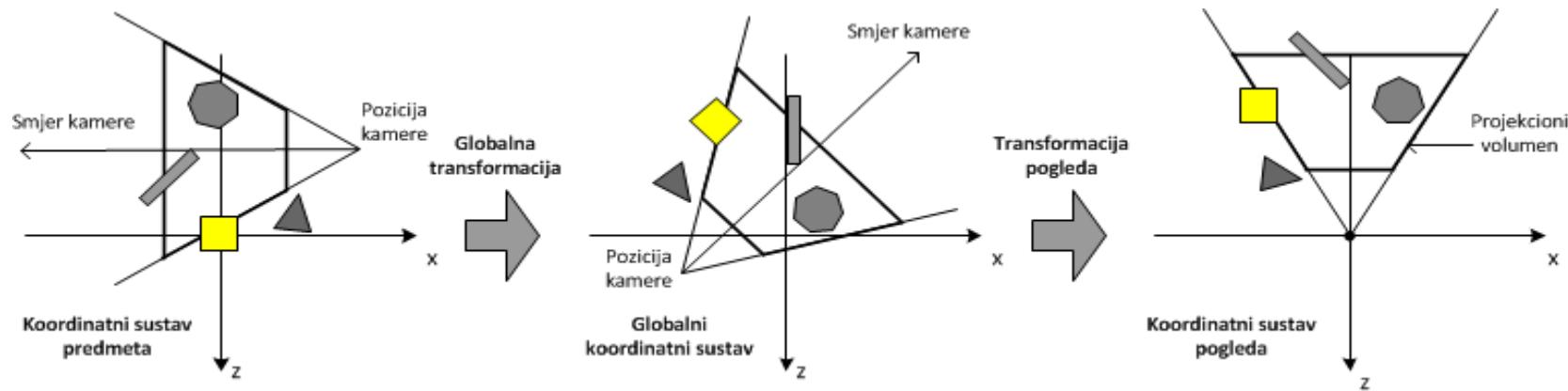
- Niz trokuta
  - Određeni vrhovima (vertex)
  - Mogu se zadati normale za svaki vrh
  - Može se zadati fiksna boja za trokut ili za svaki vrh
  - Parametri materijala
- Većina operacija izvodi se na vrhovima, dakle 3D točkama ( $x,y,z$ )

# Transformacija u prostor kamere (1/2)

- Vrhovi su zadani u *koordinatnom sustavi predmeta*
- Svakom predmetu pridružena je *globalna transformacija*, koja određuje njegov položaj i orientaciju u *globalnom koordinatnom sustavu* (zajednički za čitavu scenu)
- Kamera ima svoj *koordinatni sustav pogleda* – definira se *transformacija pogleda* koja transformira iz globalnog sustava u sustav pogleda

# Transformacija u prostor kamere (2/2)

- KS pogleda:
  - Kamera je u ishodištu
  - Gledamo u smjeru  $-z$  osi
  - $y$  os je gore,  $x$  os je desno
- Transformacija u dva koraka

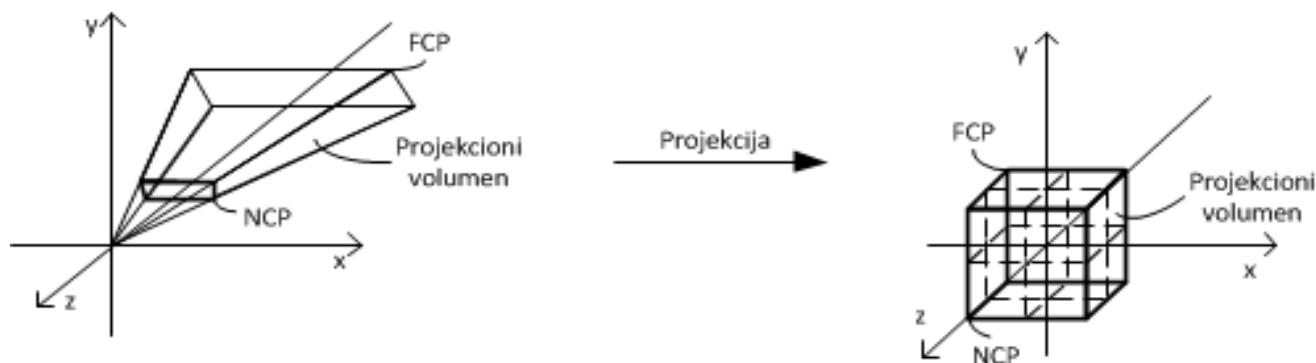


# Sjenčanje vrhova (engl. vertex shading)

- U klasičnoj izvedbi, osvjetljenje se može računati npr. modelom s prošlog predavanja
  - Normala, materijal, svjetla, položaj/kamere → (jednadžba sjenčanja) → boja vrha/trokuta
- Boja može biti i direktno zadana (nema osvjetljenja, slab 3D efekt)
- Moderan GPU – sjenčanje vrhova potpuno programabilno:
  - Proizvoljni modeli osvjetljenja i sjenčanja
  - Pomicanje/stvaranje/brisanje vrhova

# Projekcija

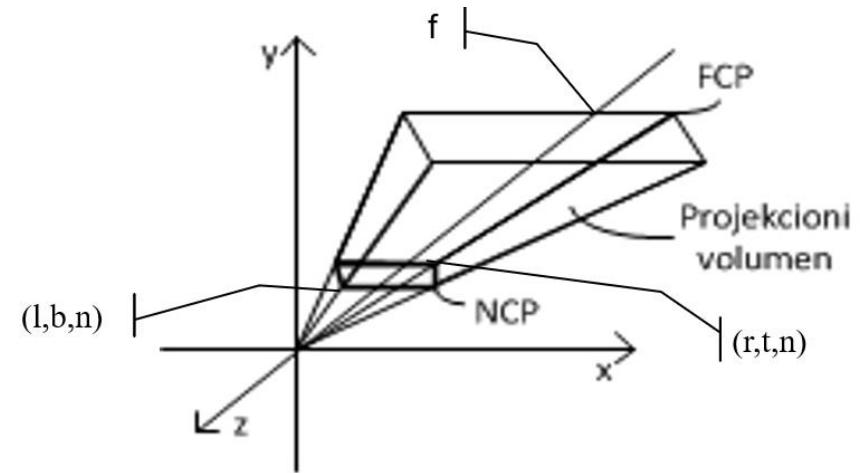
- Projekcioni volumen se transformira u jediničnu kocku (-1 -1 -1) (1 1 1)
  - x i y su normalizirane projicirane koordinate
  - z je norm. dubina, još će nam zatrebati



- Obavlja se tako da se vrhovi množe matricom projekcije

# Matrica projekcije, vidni kut, *aspect ratio*

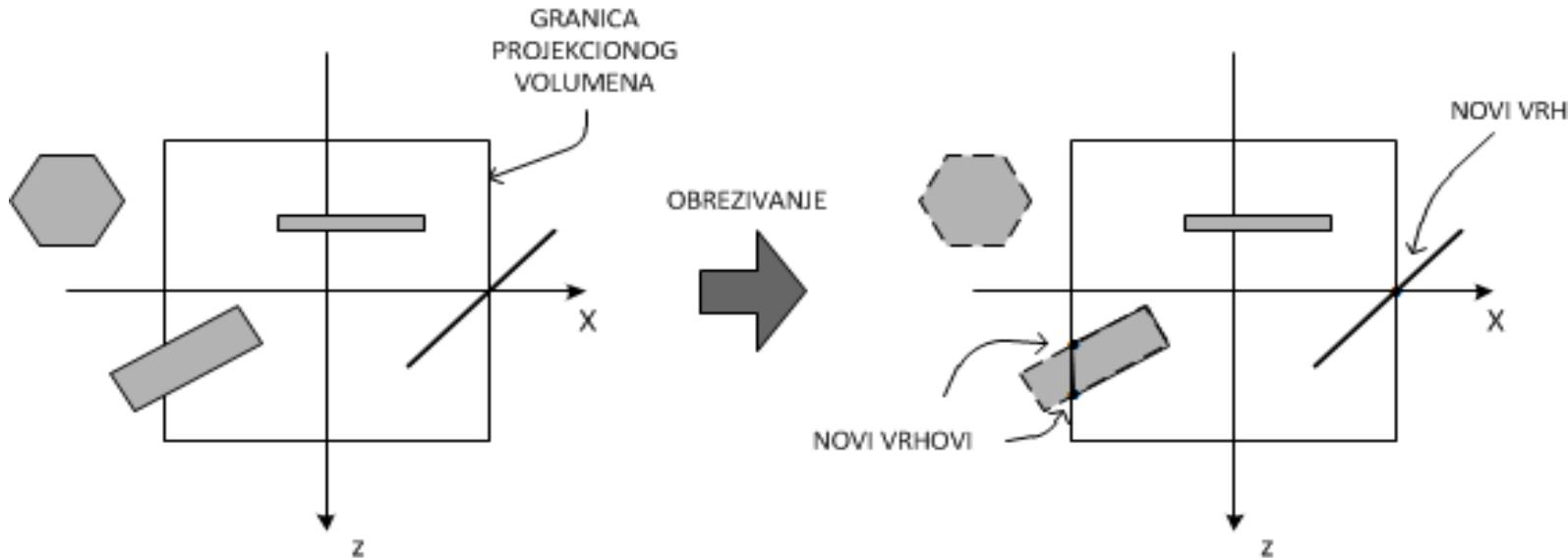
$$P_p = \begin{bmatrix} \frac{2n}{r-l} & 0 & -\frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & -\frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & 1 & 0 \end{bmatrix}$$



- Vidni kut (engl. *Field of View, FOV*)
  - Objekt širine w na udaljenosti d vidi se pod kutem:
$$\phi = \arctan\left(\frac{w}{2d}\right)$$
  - Veza vidnog kuta i matrice Pp: 
$$\phi_h = \arctan\left(\frac{r-l}{2n}\right)$$
- Omjer širina/visina (engl. *aspect ratio*): 
$$A = \frac{r-l}{t-b} = \frac{\phi_h}{\phi_v}$$

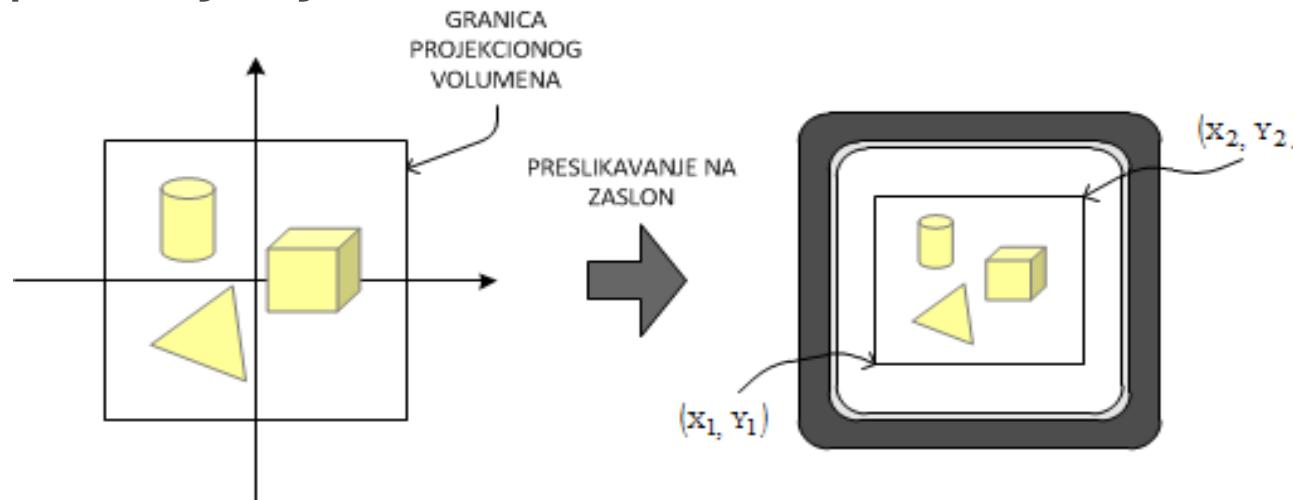
# Obrezivanje (engl. clipping)

- Odbacuju se trokuti koji su izvan jedinične kocke
- Od trokuta koji presijecaju granice jedinične kocke nastaju novi, krnji trokuti



# Preslikavanje na ekran

- x i y se transformiraju iz intervala -1,1 u koordinate ekrana
  - Jednostavna linearna operacija  $x_s = ax_n + b$
- z ostaje nepromijenjen

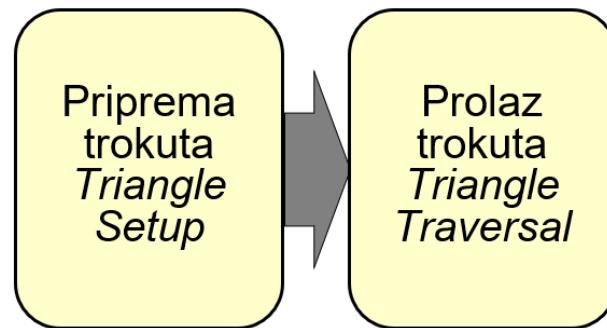


# Faza rasteriziranja

- Engl. rasterisation, scan conversion
- Pretražuje koja točka je asocirana s kojim poligonom odnosno trokutom, odnosno koliko je preklapanje pojedinog trokuta po točkama
- Najčešće se prekrivenost računa po centru piksela, ali za neke tehnike može i na osnovu drugih
- Ulaz su transformirani i projicirani vrhovi s podacima o sjenčanju istih
- Izlaz su fragmenti s podacima i o sjenčanju i o dubini

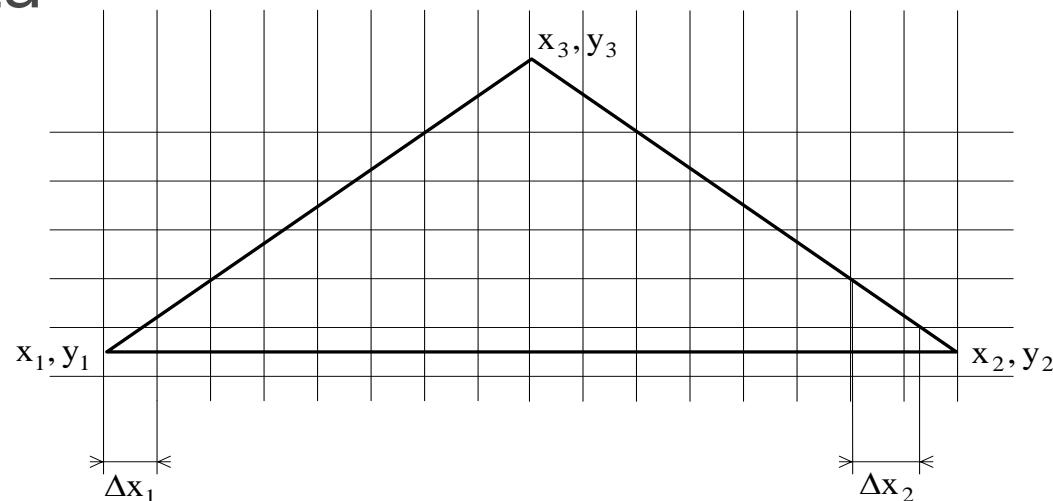
# Faza rasteriziranja: funkcija faze

- Priprema trokuta (engl. triangle setup)
- Prolaz trokuta (engl. triangle traversal)



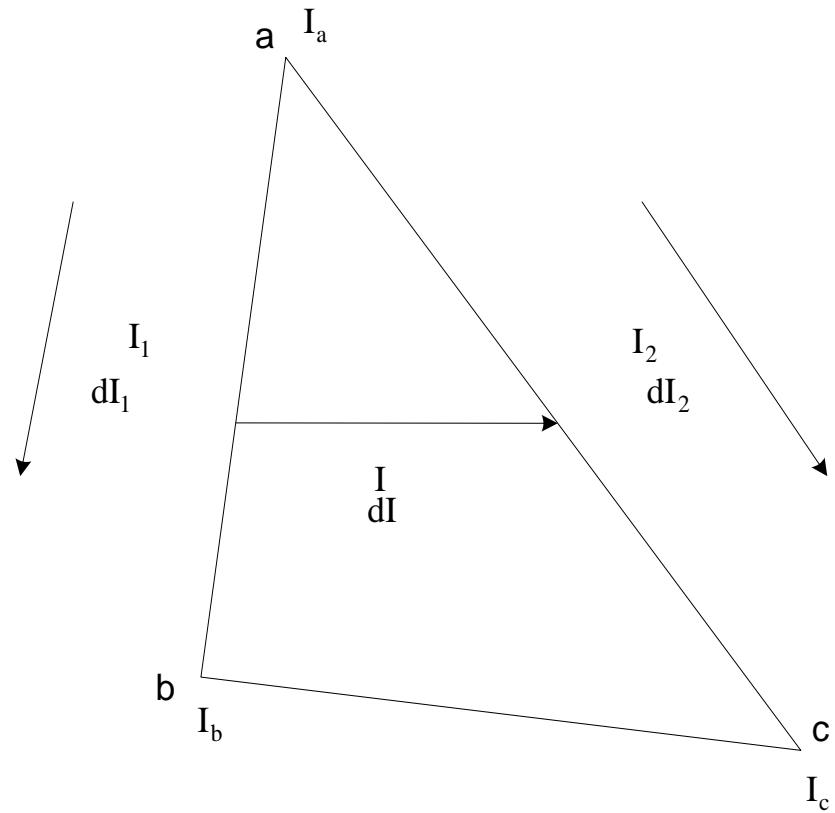
# Priprema i prolaz trokuta

- Fazi pripreme – izračun  $\Delta$  koordinata
- Faza prolaza – prekrivenost, izračun fragmenta
- Prolaz trokuta redak po redak, od lijevog do desnog ruba
- Pri prijelazu u novi redak, novi lijevi i desni rub se određuju dodavanjem  $\Delta$  koordinata



# Prolaz trokuta

- Svi podaci u vrhu interpoliraju se na prethodno objašnjrenom principu (korištenjem  $\Delta$ )
- Fragment: skup interpoliranih podataka potrebnih za sjenčanje jedne točke
  - To mogu biti koordinate točke, dubina, boja, koordinate teksture, normala, prozirnost, specifični podaci za pojedine efekte sjenčanja...

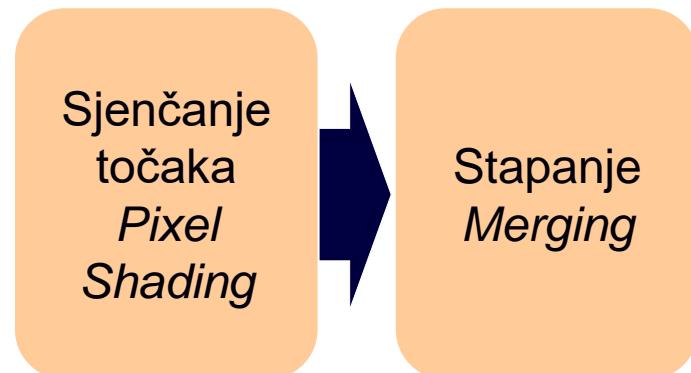


# Faza procesiranja točaka

- Engl. pixel processing
- Upisuje boju u pojedine točke zaslona (piksele)
- Određuje vidljivost točaka
- Obavlja razne dodatne funkcije (detalji kasnije)
  - Teksturiranje, prozirnost, anti-aliasing, efekti zamućenosti...
- Vrlo zahtjevna, gotovo uvijek izvedba na grafičkom procesoru

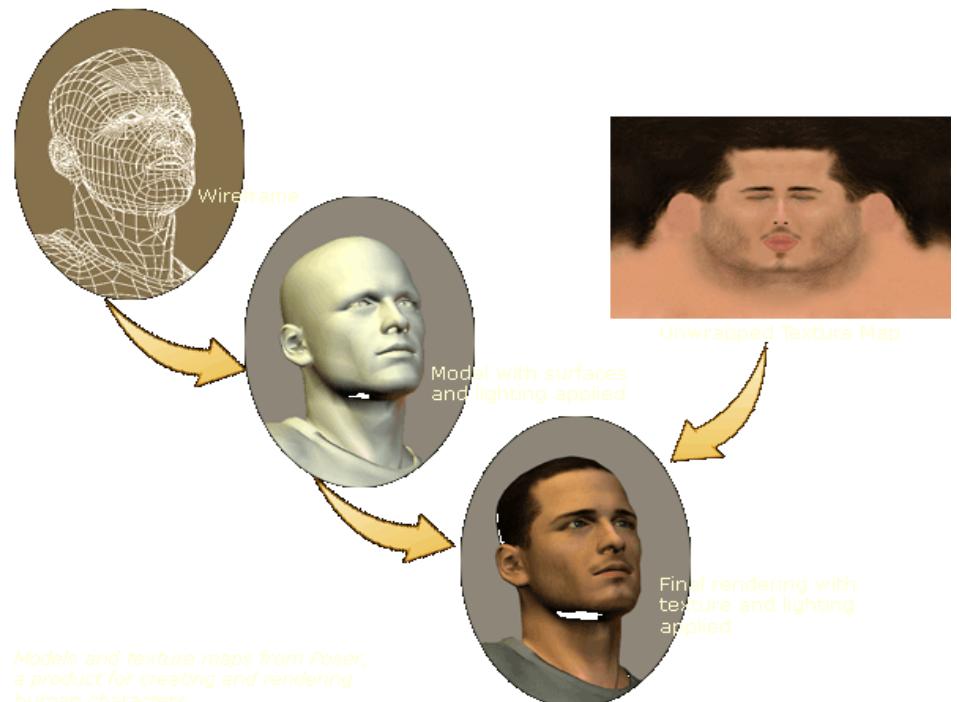
# Faza procesiranja točaka: funkcija faze

- Sjenčanje točaka (engl. pixel shading)
- Stapanje (engl. merging)



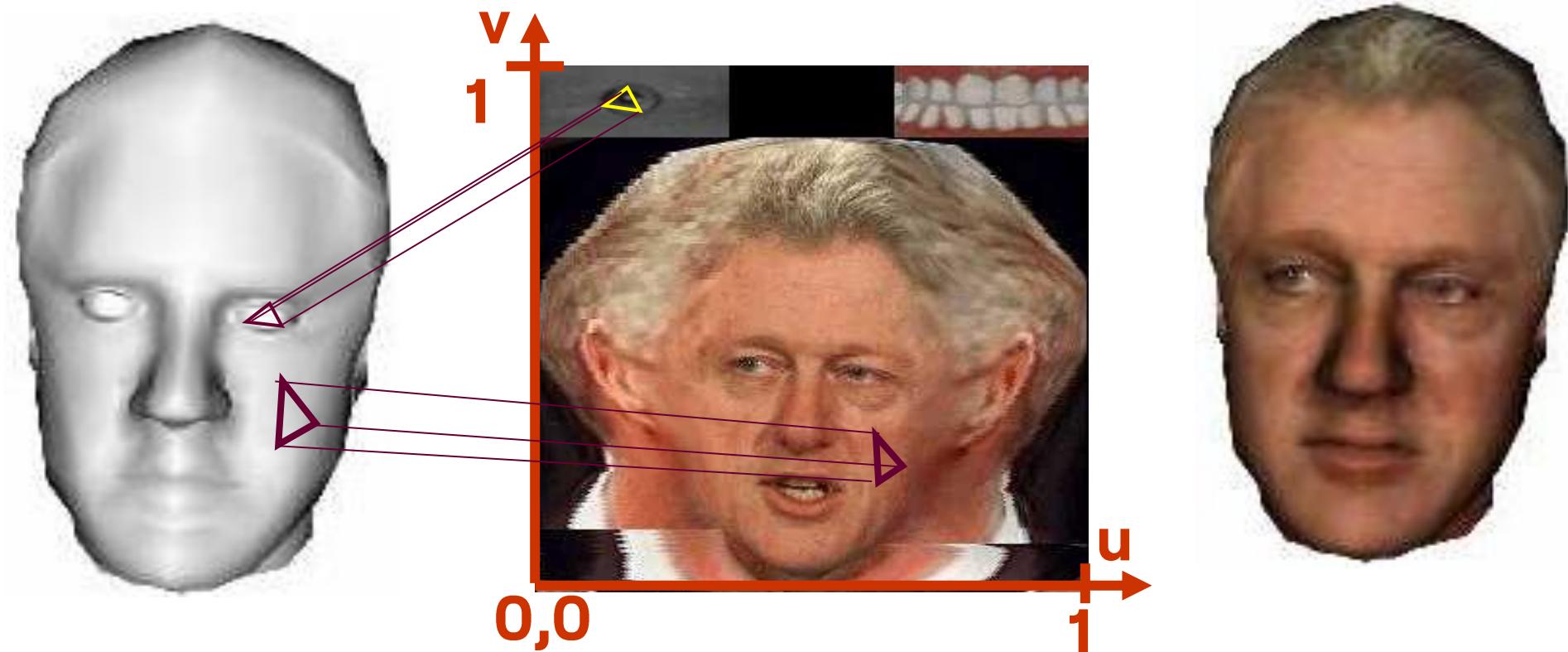
# Sjenčanje točaka

- Iz interpoliranih podataka računa se boja točke
- Programabilno, razni efekti: klasične tehnike:
  - Plošno sjenčanje – ista boja za cijeli trokut
  - Gouraudovo sjenčanje – interpolacija boje
  - Phongovo sjenčanje – interpolacija normale
  - Teksturiranje – „lijepljenje“ slike na trokute



# Teksturiranje (engl. texture mapping)

- U osnovi: „lijepjenje“ slike na geometriju
- Za svaki vrh imamo koordinate tekstuure  $u, v$



# Stapanje

- Izračunata boja točke stapa se s postojećom bojom točke u spremniku boja
- Konfiguiranjem ove faze postižu se razni efekti
  - Npr. efekt maske: oblik iscrtan u zasebnom spremniku (spremniku maske) određuje područje ekrana u kojem se točke iscrtavaju
- Faza stapanja odgovorna je i za određivanje vidljivosti metodom Z-spremnika

# Metoda Z-spremnika

- Prije crtanja, sve vrijednosti u Z-spremniku postavljaju se na maksimalnu vrijednost (1.0)
- Prilikom crtanja svake točke svakog poligona:
  - Ako je dubina točke poligona koji crtamo veća od vrijednosti u Z spremniku za točku ekrana, znači da je na ekranu već nacrtan poligon koji je bliži od ovoga kojeg upravo crtamo, dakle preskačemo
  - U protivnom upisujemo točku u spremnik boje („ekran”), a dubinu u Z-spremnik

# Grafičko sklopolje: spremnici

- Spremnik okvira (frame buffer)
  - Spremnik boje (color buffer)
  - Z-spremnik (Z-buffer)
  - Dvostruko spremanje (double buffering)
  - Stereo spremnici
  - Spremnik šablone (stencil buffer)

# Virtualna okruženja

Igor S. Pandžić, Tomislav Pejša, Mirko Sužnjević

Grafički procesor (GPU)

# Pregled predavanja

- Uvod
- Grafički protočni sustav
  - Procesor za sjenčanje (engl. shader)
  - Tok podataka kroz protočni sustav
  - Evolucija programabilnog grafičkog sklopolja
- Programski jezici za sjenčanje
  - Povijest
  - Efekti
  - Primjer programa za sjenčanje
  - Kombiniranje više svjetala i materijala
  - GPGPU

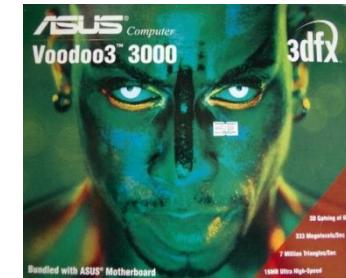
# Uvod – Povijesni razvoj grafičkih kartica

- Rani dani grafičkih kartica (1976 – 1995)
  - 1976 – RCA “Pixie” čip
    - Izlazni video signal rezolucije 62x128 piksela
    - Korišten u konzoli RCA Studio II s rezolucijom 64x32
  - 1977 – Television Interface Adapter (TIA) 1A čip – korišten u konzoli Atari 2600
  - 1979 – Intel iSBX 275 Video Graphics Controller Module mogao prikazati rezoluciju 256x256 u boji ili 512x512 monokromatsku (1000 USD)
  - 1981 – Evans & Sutherland – grafički sustav za vojni CT5 simulator leta od 20 milijuna USD
  - 1986 – prva ATI kartica OEM Color Emulation Card
  - 1989 – formiran S3 koji će dugo dominirati tržištem te proizvodi prvi čip S3 911 s 1MB RAM te podrškom za 16 bita boje.
  - 1992 – OpenGL 1.0



# Uvod – Povijesni razvoj grafičkih kartica

- 1995 – 1999 Dominacija 3Dfx Voodoo kartica
  - 1996 – 3Dfx izdaje Voodoo grafičku karticu fokusiranu na 3D
    - Kompletna revolucija na tržištu, nestala dominacija S3-a koji je kontrolirao 50% tržišta tada te na vrhuncu popularnosti 3Dfx drži 80 – 85% 3D tržišta
    - Veliki broj kartica postao zastario preko noći
  - 1996 – VideoLogic razvija tehnologiju u kojoj se miče potreba za velikim Z-spremnikom kroz odbacivanje sve geometrije koja se ne vidi prije izračunavanja tekstura, sjena i osvjetljenja
  - 1997 – ATI izdaje Rage II karticu koja se može mjeriti sa Voodoo karticama, a Nvidia RIVA 128 (Real-time Interactive Video and Animation accelerator)
  - 1998 – 3Dfx izdaje Voodoo 2 karticu koja ponovno ima najbolje 3D performanse, ali tržišna prednost se počinje topiti i ATI dostiže 27% tržišta, a Nvidia izdaje visoko popularnu karticu TNT s 32 -bitnom bojom u odnosu na 16-bitnu od Voodoo 2 kartice, a S3 se vraća sa Savage3D karticom.
  - 1999 – izlaze Savage4 od S3, Riva TNT2 od Nvidia, Voodoo 3 od 3Dfx-a te G400 od Matroxa,
  - 1999 – izlazi i Nvidia GeForce 256
    - Prvi potrošački grafički sklop s geometrijskom fazom
    - Tvrta ga naziva "grafički procesor" (engl. Graphics Processing Unit, GPU), naziv ostaje





# Uvod – Povijesni razvoj grafičkih kartica

- 2000 – Kreiranje duopola na tržištu ATI i Nvidia
  - ATI kupuje ArtX Inc, za oko 400 milijuna USD
  - VIA kupuje S3 za oko 165 milijuna USD
  - Nvidia kupuje 3Dfx za oko 70 milijuna USD
  - ATI objavljuje svoju prvu Radeon karticu
- 2001 – udio drugih proizvođača (Matrox i S3/VIA) na tržištu grafičkih kartica marginalan
- 2006 – “pobjeda” Nvidie
  - AMD kupuje ATI za 5.4 milijarde USD
  - Nvidia uvodi generičku arhitekturu procesora za sjenčanje (shader model)
- 2007 – počinje era primjene GPU-ova za “generalne” poslove (GPGPU)
  - Izlazi CUDA SDK od Nvidie koji omogućuje iskorištavanje arhitekture grafičkog procesora za generalne visoko paralelizirane poslove
  - Nvidia izdaje Tesla liniju GPGPU



# Uvod – Povijesni razvoj grafičkih kartica



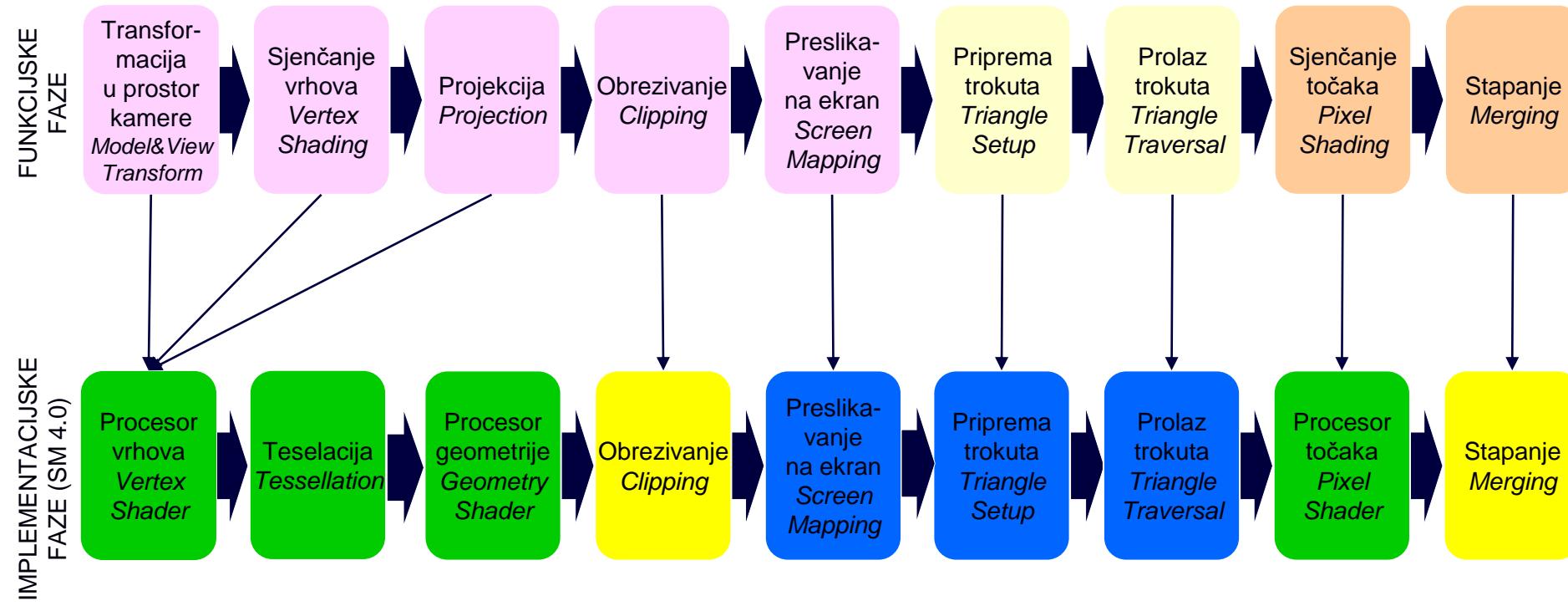
- NVIDIA GeForce256, 1999. g.
  - Prvi potrošački grafički sklop s geometrijskom fazom
  - Tvrтka ga naziva “grafički procesor” (engl. Graphics Processing Unit, GPU), naziv ostaje
- Nakon toga, idući veliki trend: programabilnost
  - Dovodi do današnjih općenitih, potpuno programabilnih procesorskih elemenata (engl. shader)

Geometrijska faza

Rasterizacija

Procesiranje piksela

# Protočni sustav na grafičkom procesoru



Programabilno

Podesivo

Fiksno

# Jedinstvena procesorska jezgra (*Common Shader Core*)

- Procesori (shaderi) se programiraju u jezicima za sjenčanje sličnima C-u
  - HLSL, Cg, GLSL
- Programi se prevode u asemblerSKI jezik neovisan o hardveru, što tvori *virtualni stroj*
- Jedinstven model za sve procesore (shadere)
  - Od inačice Shader Model 4.0
  - Ranije su procesori vrhova i točaka imali različite programske modele, a procesor geometrije nije ni postojao

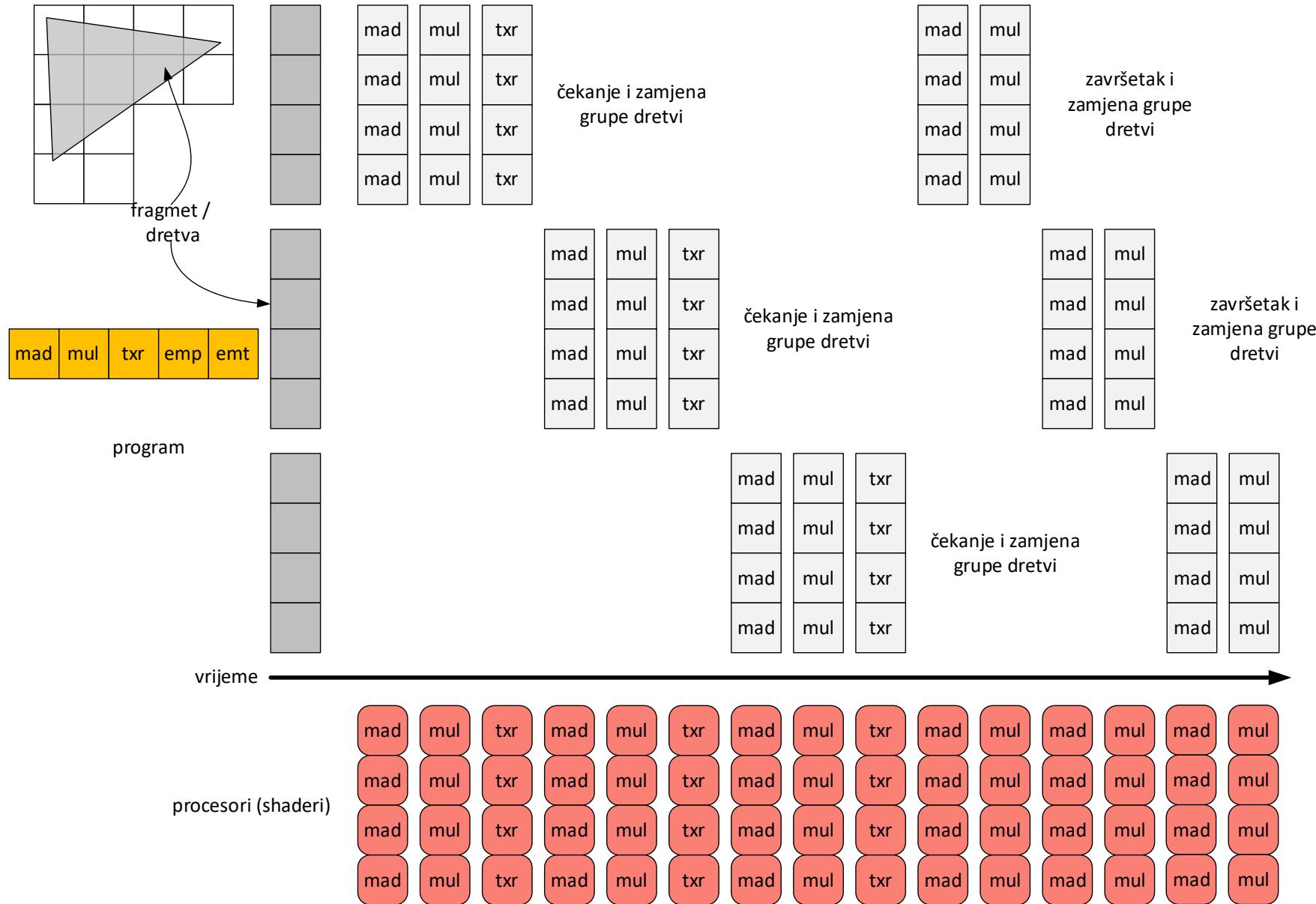
# Paralelno procesiranje

- GPU razdvaja logiku izvođenja instrukcija i podatke
- Single Instruction Multiple Data (SIMD) arhitektura
- Primjer
  - Imamo 2000 piksela za koje treba procesirati fragmente te jednu procesorsku jezgru
  - Obradujemo prvi kroz procesor (shader)
  - Obrada je brza kroz nekoliko aritmetičkih operacija na vrijednostima u lokalnim registrima
  - Nakon toga potrebno je dohvatiti teksturu zapisanu u vanjskoj memoriji što je jako zahtjevno vremenski (stotine ciklusa procesora)
  - Kako bi to bilo brže svakom fragmentu dodjeljuje se malo prostora u lokalnim registrima te umjesto da čekamo na teksturu procesoru se daje na obradu drugi fragment (od dvije tisuće)
  - Prebacivanje između obrade fragmenata je brzo jer ih ne povezuju podaci (ništa iz prvog ne utječe na drugi)
  - Prebacuje se od fragmenta do fragmenta sve dok svi nisu obrađeni, a do tada je i tekstura dostavljena iz memorije
  - Na ovaj način obrada jednog fragmenta je dulja, ali ukupna obrada svih fragmenata je puno brža – kašnjenje sakriva se prebacivanjem na drugi fragment

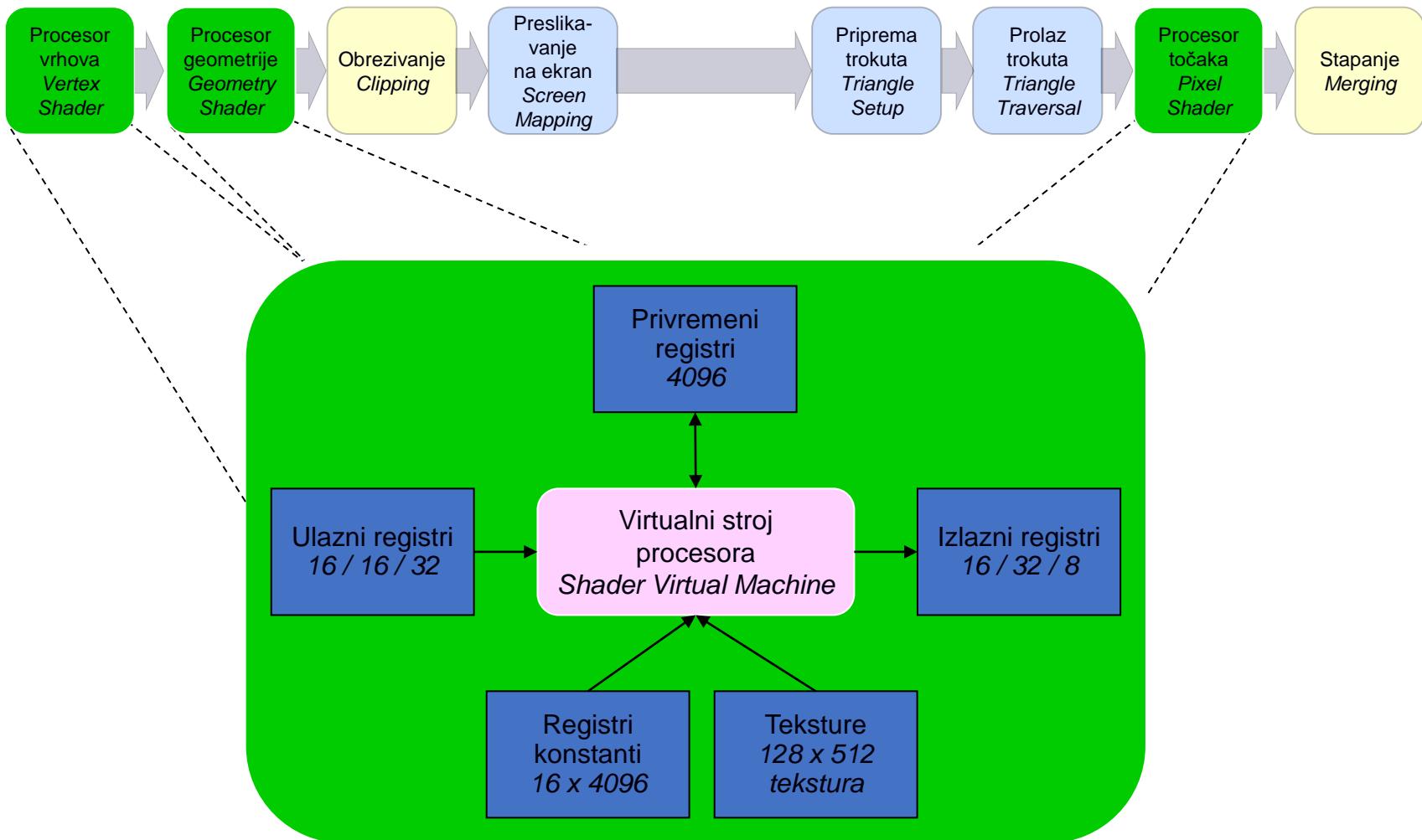
# Paralelno procesiranje

- Obrada jednog fragmenta se naziva dretva (thread) i sastoji se od dijela memorije za ulazne podatke za procesor, te memorijskog prostora koji je potreban da procesor završi izračun
- Grupe dretvi su grupirane u valove (engl. wavefronts) u AMD terminologiji odnosno grupe (engl. Warps) u NVIDIA terminologiji
- Koristeći SIMD procesiranje svaka grupa dretvi je raspoređena na određeni broj jezgri procesora (od 8 do 64)
- Svaka dretva je mapirana na SIMD traku
- Primjer
  - Na NVIDIA GPU grupa dretvi može imati 32 dretve
  - Na našem primjeru od 2000 fragmenata imamo  $2000/32 = 62,5$  63 grupe dretvi
  - Zatim se počinje izvoditi prva grupa dretvi na svih 32 procesora
  - Kada dođe do zahtjeva za teksturom isti se događa na svih 32 procesora u isto vrije
  - Cijela grupa dretvi se mijenja drugom dok se ne dostavi tekstura za prvu grupu dretvi

# Primjer



# Jedinstvena procesorska jezgra SM4.0



# Tipovi podataka i operacije

- 4x SIMD arhitektura
- Osnovni tip podataka – 4x32-bit cjelobrojni ili *floating point* vektori (novije kartice podržavaju i 64-bitu)
  - 2D, 3D i 4D vektori su vrlo česti u grafici – koordinate, normale, boje, kvaternioni, reci transf. matrica
  - Cijeli brojevi – indeksi, brojači, bitovne maske
- Operacije
  - Množenje i zbrajanje skalara i vektora – 1 takt
  - sqrt, pow, log, sin, cos... – do 4 takta

# Memorija (1/2)

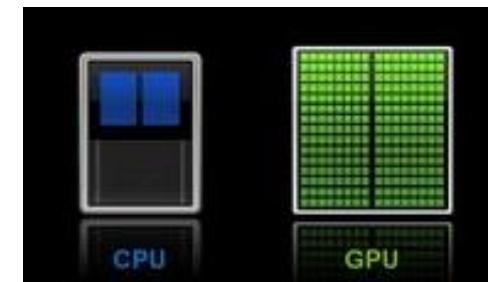
- Ulazni i izlazni registri
  - Varijabilni parametri – specifični za pojedini primitiv (vrh, trokut ili fragment)
  - Ulaz – podaci za obradu (npr. koordinate, normala, teksturne koordinate)
  - Izlaz – rezultati obrade (npr. koordinate, teksturne koordinate, boja)
- Registri konstanti
  - Konstante ili uniformni parametri – podaci konstantni u jednom pozivu iscrtavanja
  - Npr. transformacijske matrice, svjetla

# Memorija (2/2)

- Privremeni registri
  - = registri opće namjene
  - Međurezultati operacija
- Teksture
  - Memorijski spremnici
    - Nema pristupanja proizvoljnim memorijskim lokacijama
  - Najsporiji oblik
    - Registri su fizički dio GPU
    - Teksture se u memoriji na graf. kartici ili memoriji sustava
  - Dohvat podataka (*uzorkovanje*)
    - Kontinuirane teksturne koordinate (0-1)
    - Odgovaraju jednoj ili više memorijskih lokacija
    - Ako ih je više, sadržaj se interpolira (filtriranje tekstura)

# Kontrola toka

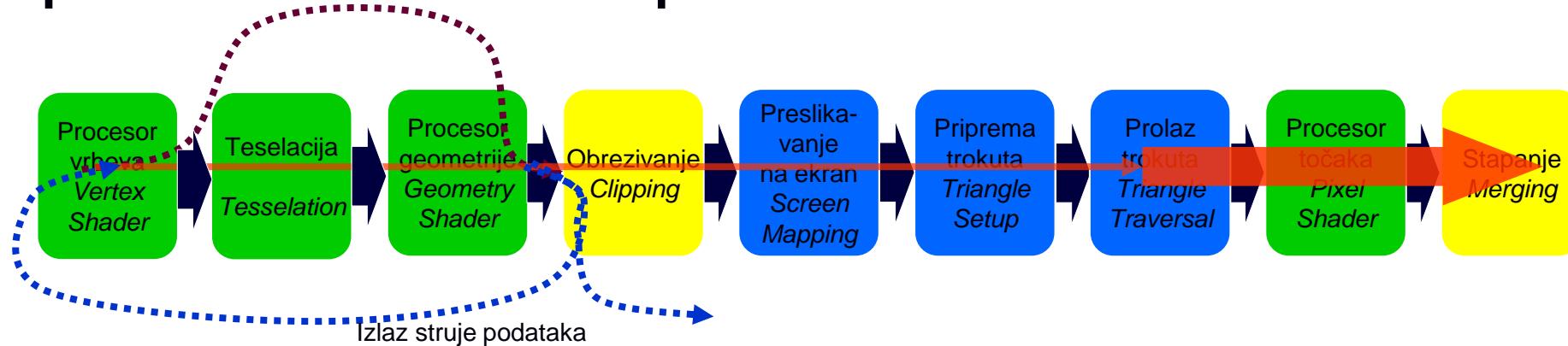
- Kao i na CPU – if-else, for, while
- Zapravo je stvar složenija
- Statička kontrola toka
  - Samo *konstante* u uvjetu – tijek izvođenja isti u jednom prolazu (dobre performanse)
- Dinamička kontrola toka
  - Varirajući podaci u uvjetu – tijek izvođenja može biti različit za različite primitive
  - Slabije performanse zbog *granularnosti*
  - GPU obrađuje po nekoliko desetaka ili stotina primitiva u paraleli – aktivne grane izvođenja moraju se izvesti za sve primitive



# Model izvođenja

- U pripremnom koraku:
  - Učitavanje programa za sjenčanje (preko grafičkog API-ja)
  - Postavljanje vrijednosti konstanti, alokacija memorijskih resursa (tekstura, spremnika vrhova...)
- U petlji iscrtavanja:
  - Poziv funkcije za iscrtavanje (engl. draw call)
  - 1 poziv iscrtavanja = 1 prolaz geometrije kroz protočni sustav
  - Pritom će procesori za sjenčanje izvesti učitane programe na svim primitivima
  - Rezultat se zapisuje u 1 ili više memorijskih spremnika (npr. spremnik boje, teksturu...)

# Tok podataka kroz protočni sustav



- Umnožavanje podataka interpolacijom pri prolazu trokuta
- Procesor geometrije te teselacija se ne moraju koristiti
- Izlaz struje podataka (engl. Stream Output)
  - Povrat u sustav za iterativnu obradu
  - Izlaz iz sustava, praktično za opće primjene grafičkog procesora

Programabilno

Podesivo

Fiksno

# Procesor vrhova (1/2)

- Ulaz: vrh sa pripadajućim podacima
  - Minimalno: koordinate
  - Dodatno: normala, teksturne koordinate, boja...
  - Rad samo na pojedinom vrhu (nema pristupa drugim vrhovima)
- Operacije na ulaznim podacima
  - Minimalno: transf. u prostor kamere, projekcija
  - Sve ostalo ovisno o željenom efektu (npr. proračun osvjetljenja za Gouraudovo sjenčanje)
  - Ne može brisati niti dodavati vrhove
- Izlaz
  - Minimalno: položaj nakon projekcije
  - Izlazni registri se interpoliraju u prolazu trokuta, interpolirane vrijednosti za svaku točku ulaze u procesor točaka

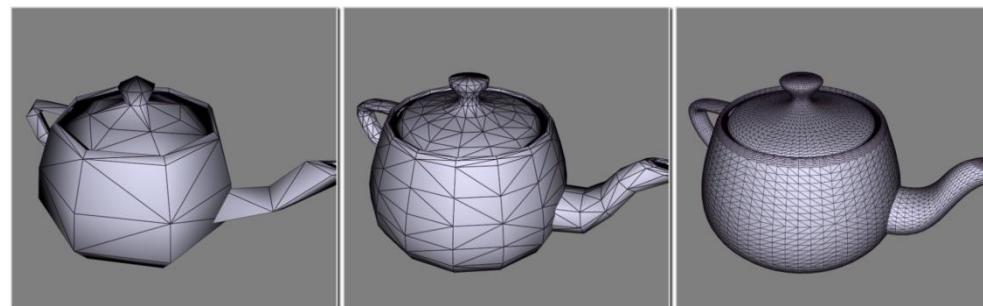
# Procesor vrhova (2/2)

- Ukupni efekt je rezultat suradnje procesora vrhova i točaka (i geometrije)
- Neki primjeri efekata
  - Animacija kože pomoću kostura (engl. skinning)
  - Interpolacija oblika (engl. morphing)
  - Efekti objektiva kao npr. riblje oko
  - Proceduralne animacije (npr. zastave, valovi...)



# Procesor teselacije

- Teselacija nam omogućava iscrtavanje zakrivljenih površina
- Latinski *tessella* – mali kamen ili pločica u mozaiku
- Predstavlja uzorak koji se može proširiti u svakom smjeru
- Prednosti
  - Sama teselacija štedi na resursima pošto su parametri teselacije manji nego dostava svih trokuta vezanih za zakrivljene površine
  - Dodatno može sprječavati zagušenje sabirnice između CPU-a i GPU-a za likove čiji se oblik mijenja u svakom okviru
- Uvijek se sastoji od tri koraka
  - DirectX terminologija: hull shader, tessellator i domain shader
  - OpenGL terminologija: tessellation control shader, tessellator i tessellation evolution shader

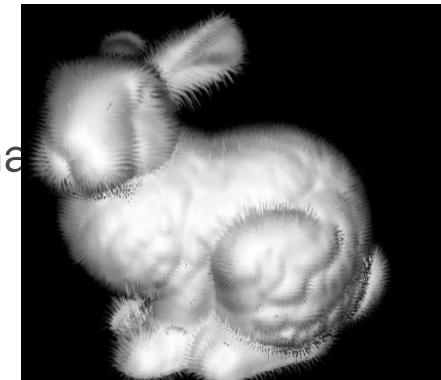


# Procesor teselacije

- Hull shader
  - Ulaz je poseban set kontrolnih točaka koji definiraju zakrivljenu površinu (primjerice Bezier površina)
  - Može modificirati ulazne kontrolne točke, dodavati ih ili uklanjati
  - Teselatoru šalje tip površine (trokut, četverokut ili isocrta – poseban oblik traka koji se koriste za renderiranje kose) te faktore teselacije
  - Unutarnji faktori teselaciji određuju koliko teselacije se dešava unutar jedne jedinice (trokuta)
  - Vanjski faktori teselacije govore na koliko se dijeli svaki vanjski brid
  - Domain shaderu se šalju transformirani set kontrolnih točaka te kontrolne podatke
- Teselator generira mesh s te ga šalje domain shaderu
- Domain shader svaki vrh iz teselatora prilagođavaja na temelju kontrolnih točaka te generira izlazne vrijednosti vrhova

# Procesor geometrije

- Uporaba je izborna
- Ulaz
  - Objekt (trokut, crta, točka) s pripadajućim vrhovima (+ dodatno susjedni vrhovi sa shader modelom 5 čak do 32 dodatne kontrolne točke)
- Izlaz
  - Nula ili više objekata: procesor može brisati, dodavati i mijenjati primitive (skupa operacija, mogućnost učinkovite teselacije dodana u Shader Modelu 5.0)
- Primjeri korištenja
  - Generiranje čestica raznih veličina i oblika u sustavima čestica
  - IsCRTavanje siluete u efektu krzna
  - Generiranje proceduralnih objekata (npr. *metaballs*)
  - Generiranje fraktalne geometrije



# Izlaz struje podataka

- Engl. stream output
- Izlaz procesora vrhova/geometrije može se zapisati izravno u memorijski spremnik
- Već u idućem prolazu se može koristiti kao spremnik vrhova – iterativna obrada podataka!
- Faza rasterizacije se može potpuno isključiti (bolje performanse)
- Korisno za GPGPU

# Procesor točaka (fragmenata) (1/2)

- Ulaz: fragment
  - Fragment: točka trokuta sa podacima za sjenčanje (**koordinate, dubina**, boja, teksturne koord., normala...)
  - Podaci fragmenta – određeni interpolacijom u prolazu trokuta
  - Procesor točaka sjenčanjem računa boju fragmenta
  - Ako fragment preživi određivanje vidljivosti (Z-buffer), boja se zapisuje u spremnik boje (postaje bojom piksela)
- Operacije
  - Izračun modela osvjetljenja, teksturiranje, efekt magle, razni specijalni efekti...

# Procesor točaka (fragmenata) (2/2)

- Izlaz: boja
  - Može i izbrisati fragment ili mu promijeniti dubinu (z)
  - Višestruki spremnici (engl. Multiple Render Targets, MRT)
    - Više boja na izlaz, zapisuju se u različite spremnike
    - IsCRTavanje više slika u jednom prolazu, stapanje u jednu završnu sliku
- Nema pristupa susjednim točkama, ali ima diferencijalima podataka (ddx, ddy)
  - Zato što se točke obrađuju u blokovima 2x2 – 8x8
  - Korisno za mipmapping ili anti-aliasing
  - Nedostupni unutar grane dinamičkog grananja

# Stapanje

- Zadatak: odrediti konačnu boju točke (piksela) koja će se vidjeti na ekranu
  - Ulaz: boja više fragmenata koji se preslikavaju u točku, Z-spremnik, spremnik maske...
- Određivanje vidljivosti metodom Z-spremnika
- Operacije sa spremnikom maske
- Nije programabilna, ali velika fleksibilnost u postavkama (rasterske operacije, ROP)
  - Množenje, zbrajanje, oduzimanje, min/max, bitovne operacije na vrijednostima boja i prozirnosti
  - Najčešće za miješanje boje
  - Neovisno miješanje višestrukih spremnika (MRT)

# Procesor računanja (engl. compute shader)

- Koristi se za općenite izračune – GPU računarstvo ili GPGPU
- Procesor računanja je uveden u DirectX 11
- Platforme za GPGPU
  - CUDA
  - OpenCL
  - ...
- Nema predefiniranu poziciju u protočnom sustavu GPU
- Temelji se na istoj standardiziranoj jedinstvenoj procesorskoj jezgri
- Jedna od prednosti procesora računanja je pristup GPU memoriji što je puno brže od komunikacije između GPU i CPU
- Koriste se i za čestične sustave, procesiranje mesheva kao što je animacija lica, sjene, itd...

# Evolucija programabilnog grafičkog sklopolja

- Povijest ideje programibilnosti u procesu iscrtavanja
  - Shade trees 1984.
  - RenderMan Shading Language, kasne 80te, koristi se i danas
- Shader Model 1.1, 2001. g., DX8.0 i OpenGL, NVIDIA GeForce 3
  - Procesor vrhova, vrlo ograničeni procesor točaka
  - Programiranje u asembleru
- SM 2.x, 2002. g., DX9.0 i OpenGL
  - Potpuno programabilni procesor točaka
  - Povećanje svih ograničenja resursa (broj instrukcija, teksura, registara...)
  - Jezici više razine (HLSL, GLSL, Cg)
  - Statička kontrola toka
- SM 3.0, 2004 g., DX9.0c i OpenGL (2005. g. Xbox 360, 2006. g. Playstation 3)
  - Dinamička kontrola toka
  - Daljnja povećanja resursa
- SM 4.0, 2007 g., DX10.0 i OpenGL
  - Jedinstvena procesorska jezgra
  - Izlaz struje podataka
  - Procesor geometrije
  - Asembler više nije podržan osim za debugging
  - Daljnja povećanja resursa
- SM 5.0, 2009 g., DX11.0 i OpenGL
  - Teselacija
  - Elementi OOP u HLSL-u
- SM 6.0, 2016 g., DX12.0
  - Razdvojeni kompajliranje i optimizacija
  - Proceduralne tekture, mijenjanje veličine piksela
- Trenutno SM 6.6, video o najnovijim mogućnostima kartica koje implementiraju dani model  
<https://www.youtube.com/watch?v=5rYBLjUmGkA>

# Programski jezici za sjenčanje (1/2)

- Program za sjenčanje (engl. *shader*) – izvodi se na procesoru vrhova/teselacije/geometrije/točaka
- *Shaderi* se pišu u jezicima za sjenčanje (engl. shading languages) i prevode u *assembler*
- Jezici slični C-u, prilagođeni specifičnostima grafičkog sklopolja
  - *Stream processing*, vektorske operacije, nema slobodnog pristupa memoriji
  - Glavni tip podataka 4x32-bit vektor, no postoje i skalarni tipovi, cijelobrojni tipovi, matrice
  - Operacije množenja, zbrajanja, oduzimanja...
  - Ugrađene funkcije – mat. i log. operacije, uzorkovanje tekstura, izračun gradijenta (samo u procesoru točaka)
  - Definiranje funkcija i struktura (kao u C-u)
  - Preprocesor – #include, #define, #ifdef itd. (kao u C-u)

# Programski jezici za sjenčanje (2/2)

- Razvili se iz ekvivalentnih jezika za *offline* iscrtavanje (Pixar PhotoRealistic RenderMan)
- Za prvu generaciju (prije SM2.0) morao se koristiti *assembler*
- Cg (NVIDIA, 2003.)
  - Prvi visokorazinski jezik za sjenčanje
  - kratica – *C for Graphics*
- HLSL (Microsoft, 2003.)
  - *High Level Shading Language*
  - Gotovo identičan Cg-u, no namijenjen korištenju uz Direct3D
- GLSL (OpenGL ARB, 2004.)
  - *OpenGL Shading Language*
- Od SM4.0 *shaderi* se više ne mogu pisati u *assembleru*

# Virtualna okruženja

Igor S. Pandžić, Tomislav Pejša, Mirko Sužnjević

Grafički procesor (GPU)

# Efekti (1/3)

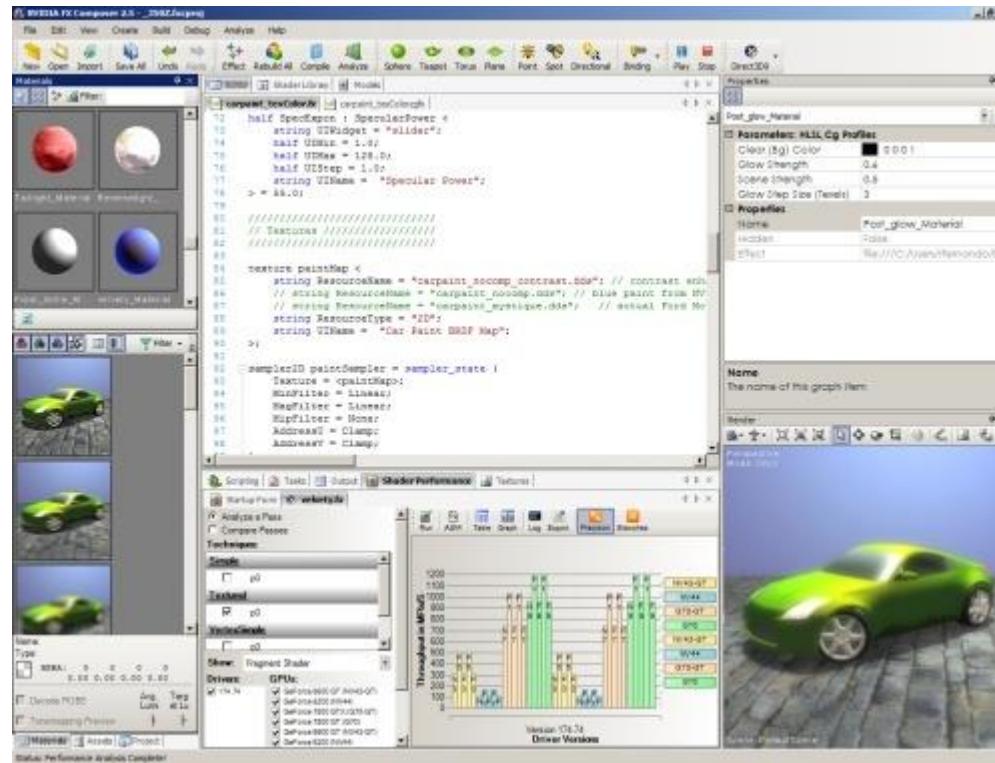
- Pojedini *shader* nije koristan sam za sebe
  - Željeni rezultat dobiva se kombiniranjem više *shadera* i postavki protočnog sustava
- Efekt: skup programa za sjenčanje i svih potrebnih postavki protočnog sustava (npr. uključen/isključen Z-spremnik)
- Zapisuje se u jedinstvenoj datoteci korištenjem jezika za efekte (HLSL FX, CgFX, COLLADA FX)
- Globalni parametri efekta
  - Kroz njih korisnik upravlja svojstvima efekta
  - Moguće ih postavljati kroz sučelje alata za razvoj efekata
  - Npr. efekt Phongovo sjenčanje – parametri materijala, svjetla

# Efekti (2/3)

- *Shaderi* su organizirani u *prolaze* (engl. pass)
  - 1 prolaz iscrtavanja kroz grafički protočni sustav
  - Sastoji se od programa za sjenčanje vrhova i točaka (te eventualno geometrije) + postavke protočnog sustava
- Više prolaza čini *tehniku*
  - Slijed prolaza potrebnih za realizaciju efekta
  - Često može biti više tehnika (npr. za različite generacije grafičkog sklopolja)

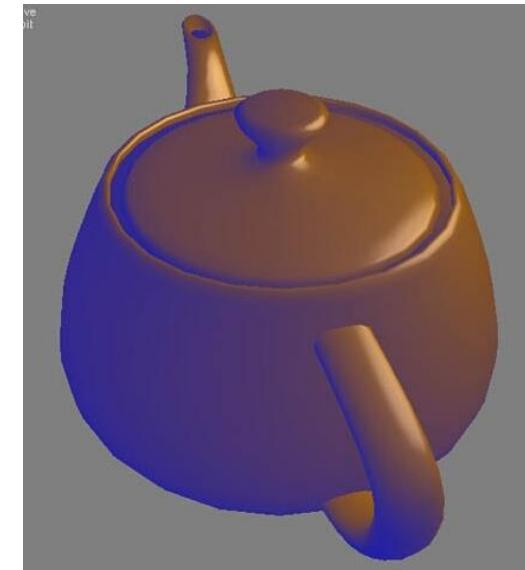
# Efekti (3/3)

- Alati za razvoj efekata
  - NVIDIA FX Composer, AMD RenderMonkey



# Primjer efekta

- Goochovo sjenčanje
  - Vrsta ne-foto realističnog sjenčanja
- Topli tonovi u smjeru svjetla, hladni od svjetla
  - Osnovni topli i hladni ton definira korisnik
  - Interpolacija boje u ovisnosti o kutu normale prema svjetlu
- Primjer pisan u jeziku HLSL FX



# Konstante (uniformni parametri)

```
float4x4 WorldXf      : World  
float4x4 WorldITXf     : WorldInverseTranspose  
float4x4 WvpXf       : WorldViewProjection
```

- **Sintaksa:**

```
tip ime : semantika
```

- **Semantika**

- Definira značenje parametra
- Aplikacija prema semantici određuje na koju vrijednost inicijalizirati konstantu

# Varijable koje zadaje korisnik

```
float3 Lamp0Pos : Position
<
    string Object = "PointLight0";
    string UIName = "Lamp 0 Position";
    string Space = "World";
> = {-0.5f, 2.0f, 1.25f};
float3 WarmColor
<
    string UIName = "Gooch Warm Tone";
    string UIWidget = "Color";
> = {1.3f, 0.9f, 0.15f};
float3 CoolColor <...
```

- Podatke u <> koristi aplikacija, ne sam shader
  - Definiraju kako i pod kojim imenom prikazati parametre u sučelju alata

# Program za sjenčanje vrhova (1/2)

- Ulazni parametri procesora vrhova

```
struct vertexIn
{
    float3 Position : POSITION;
    float3 Normal      : NORMAL;
};
```

- Izlazni parametri proc. vrhova / ulaz u proc. Točaka

```
struct vertexOut
{
    float4 HPosition : POSITION;
    float3 LightVec : TEXCOORD1;
    float3 WorldNormal : TEXCOORD2;
};
```

- Semantika ima analogan smisao – govori protočnom sustavu koji podatak primitiva zapisati u koju varijablu
- Semantika TEXCOORD ostala iz povijesnih razloga

# Program za sjenčanje vrhova (1/2)

```
vertexOut std_vs( vertexIn IN )
{
    vertexOut OUT;
    float4 No = float4(IN.Normal, 0);
    OUT.WorldNormal = mul(No, WorldITXf).xyz;
    float4 Po = float4(IN.Position, 1);
    float4 Pw = mul(Po, WorldXf);
    OUT.LightVec = (Lamp0Pos - Pw.xyz);
    OUT.HPosition = mul(Po, WvpXf);
    return OUT;
}
```

- Računa normalu i vektor svjetla u globalnom koordinatnom sustavu
- Transformira vrh u prostor pogleda i projicira ga

# Program za sjenčanje točaka

```
float4 gooch_PS(vertexOutput IN)
{
    float3 Ln = normalize(IN.LightVec);
    float3 Nn = normalize(IN.WorldNormal);
    float ld़n = dot(Ln, Nn);
    float mixer = 0.5 * (ld़n + 1.0);
    float4 result = lerp(CoolColor, WarmColor, mixer);
    return result;
}
```

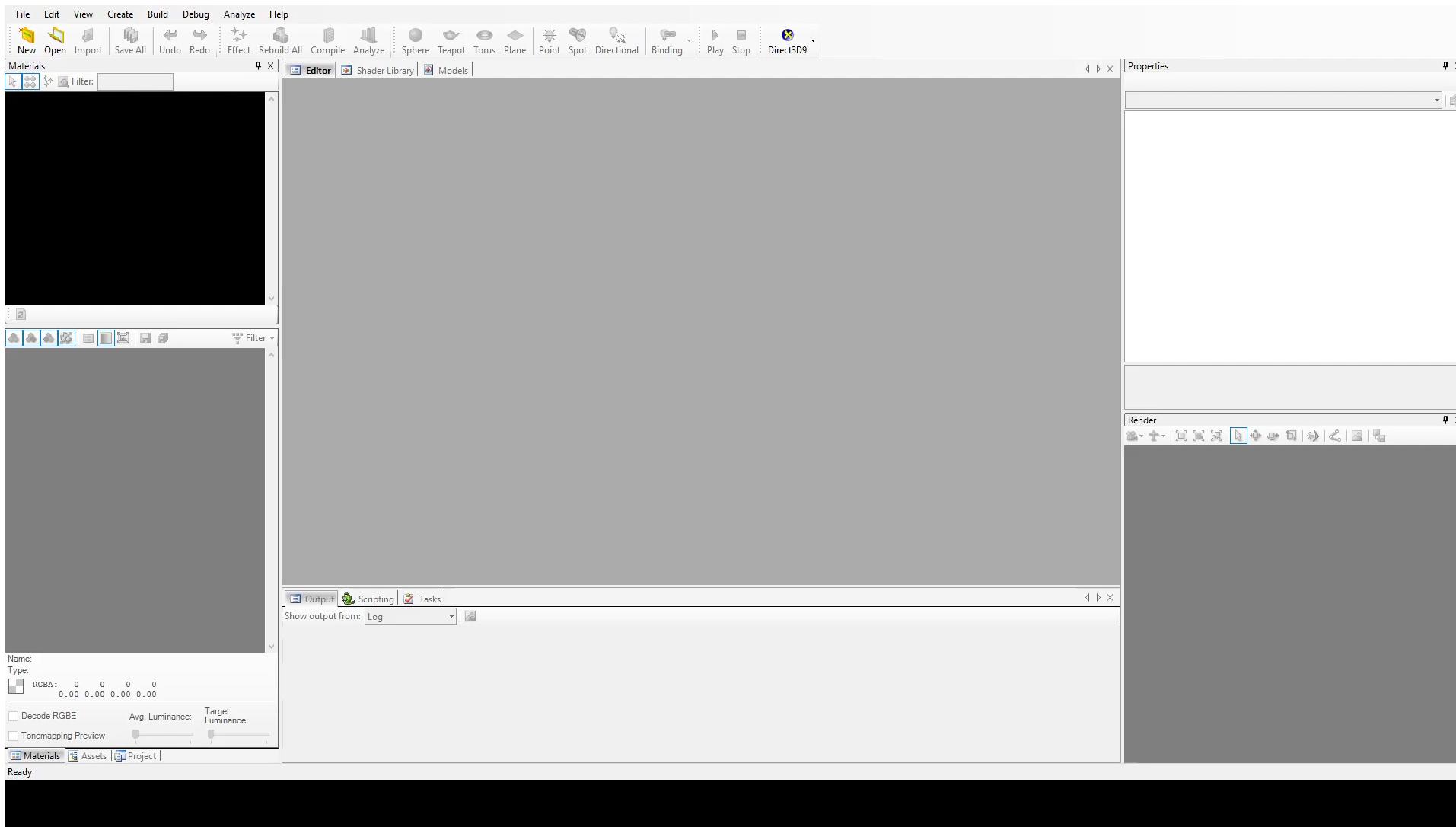
- Interpolira između toplog i hladnog tona u ovisnosti o kutu između normale i smjera svjetla

# Definicija tehnike

```
technique Gooch < string Script = "Pass=p0;" >
{
    pass p0 <string Script = "Draw=geometry;" >
    {
        VertexShader = compile vs_2_0 std_VS();
        PixelShader = compile ps_2_a gooch_PS();
        ZEnable = true;
        ZWriteEnable = true;
        ZFunc = LessEqual;
        AlphaBlendEnable = false;
    }
}
```

- Naš efekt ima jednu tehniku, koja ima jedan prolaz
- Standardna uporaba Z-spremnika, prozirnosti nema

# Primjer efekta



# Kombiniranje više svjetala i materijala

- *Shaderi* nam omogućuju proizvoljne kombinacije broja i tipova svjetala, materijala i jednadžbi sjenčanja
- Kombinacije mogu varirati i unutar iste scene
- Kako ostvariti sve te kombinacije unutar programa za sjenčanje?
  1. Dinamičko grananje
  2. Übershader
  3. Višeprolazno osvjetljenje
  4. Odgođeno sjenčanje
- Definicije:
  - $M$  tipova svjetala
  - $N$  tipova materijala
  - $L$  svjetala po predmetu

# Dinamičko grananje

```
float4 my_shader( . . . )
{
    float4 color;
    for( i = 0; i < L; ++i )
    {
        color += light_compute( light[i],
            objMaterial );
    }
    return color;
}
```

# Übershader

- Zaseban *shader* za svaku kombinaciju
- Broj kombinacija:  $\frac{(M + L)!}{L!M!} \times N$
- Primjer: Half-Life 2 (2004.) – 1920 kombinacija!
- Nije praktično *shader* za svaku varijantu pisati zasebno
- Piše se jedan složeni *übershader*:
  - Preprocesorske naredbe za razne kombinacije
  - Višestruko prevođenje – različite varijante *shadera*
- Nepraktično za umjetnike

# Višeprolazno osvjetljenje

- Zaseban prolaz iscrtavanja za svaki izvor svjetlosti
- Aditivno miješanje rezultata u spremnik boja u fazi stapanja
- Broj *shadera*: MxN
- *Shaderi* su jednostavnji, ali performanse su slabije (aditivno miješanje, ponavljanje proračuna)

# Odgodeno sjenčanje (1/3)

- Sjenčanje se radi *nakon* određivanja vidljivosti Z-spremnikom
- 1. prolaz:
  - Iscrta se scena bez sjenčanja, ali uz uključen Z-spremnik
  - MRT – podaci vidljive geometrije u svakoj točki se zapisuju u *G-spremnike* (dubine, normale, teksturne koordinate)
- Daljnji prolazi:
  - Iscrta se jedan pravokutnik koji prekriva cijeli zaslon, a na njega se „lijepi“ slika scene
  - U procesoru točaka – uzorkuju se G-spremnići i obavlja sjenčanje

# Odgodeno sjenčanje (2/3)

- Odlične performanse (sjenčaju se samo vidljivi fragmenti)
- Gotovo neograničen broj svjetala – puno „življe” i realnije scene
- Broj *shadera*:  $M+N$  ; *shaderi* za materijal (1. prolaz) odvojeni od *shadera* za svjetla (2. prolaz) – olakšan posao umjetnicima
- Nedostaci:
  - Velike potrebe za memorijom
  - Nema MSAA (no moguće implementirati u *pixel shaderu*)

# Odgodeno sjenčanje (3/3)

- Primjer: S.T.A.L.K.E.R. (2007.)

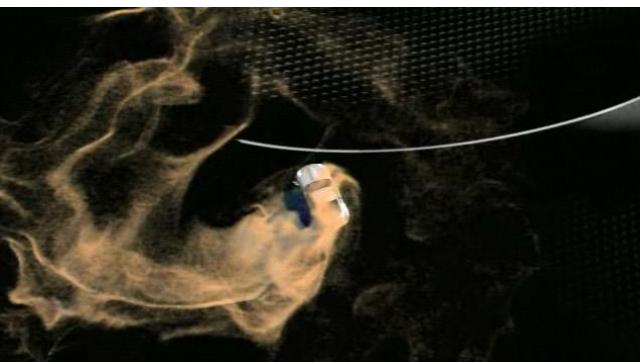


# GPGPU (1/2)

- Općeniti proračuni na grafičkom procesoru
- Strujna obrada podataka (engl. stream processing)
  - Struja – podaci; jezgre – operacije na podacima
  - Paralelna, nezavisna obrada velikog broja struja
  - IsCRTavanje je primjer struje obrade podataka!
- GPGPU sustavi i jezici:
  - NVIDIA CUDA
  - AMD APP SDK
  - OpenCL
  - DirectCompute

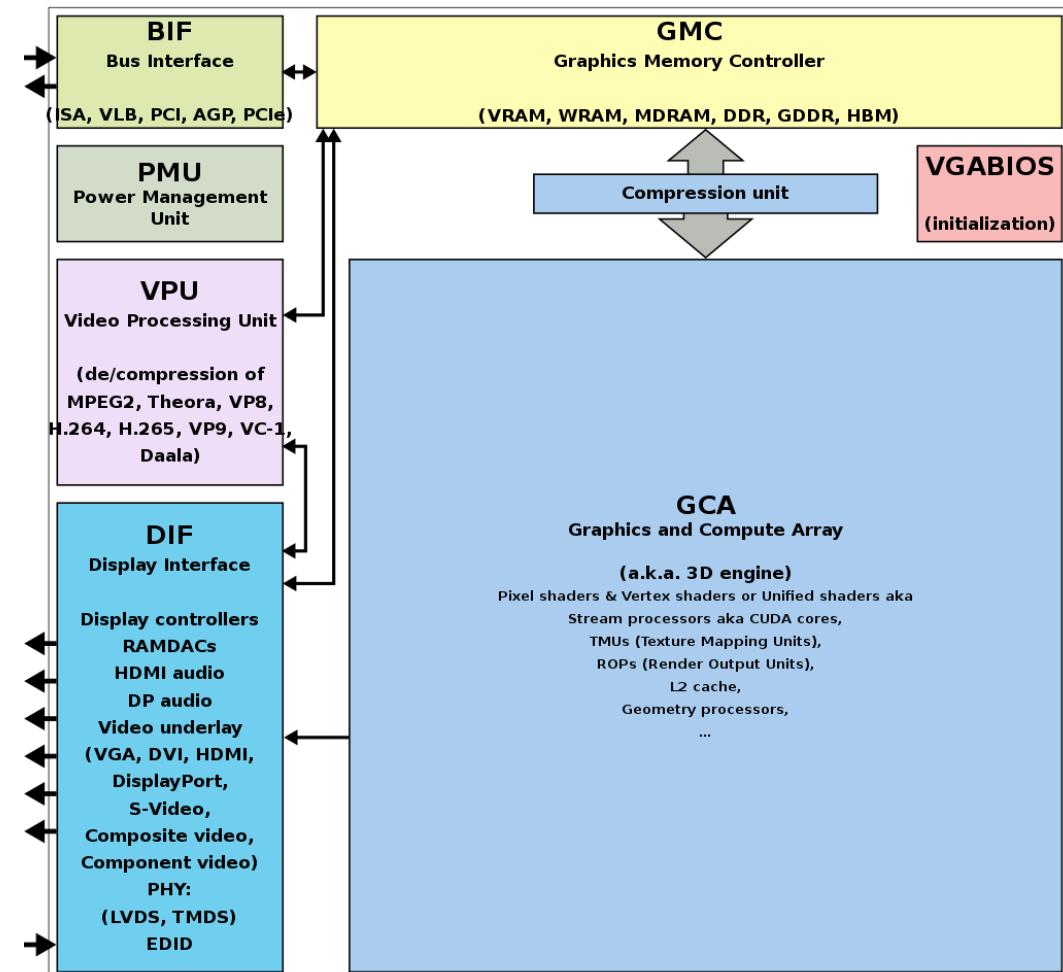
# GPGPU (2/2)

- Danas – dedicirano grafičko sklopolje za GPGPU
- Primjene:
  - Fizikalne simulacije
  - Ray tracing
  - Obrada slike, zvuka i videa
  - Kriptografija i kriptoanaliza
  - Baze podataka
  - Neuronske mreže
  - Linearna algebra
  - Sortiranje i pretraživanje



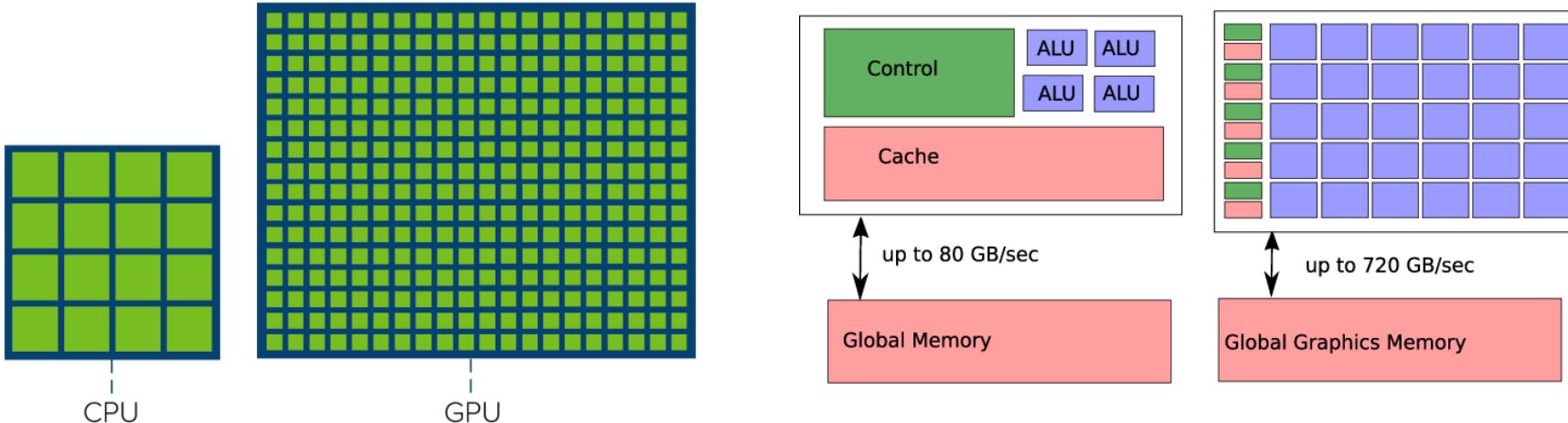
# Grafičko sklopovlje – generička arhitektura

- Sabirnica (engl. bus)
- Memorija
- Jedinica za upravljanje napajanjem
- Jedinica za procesiranje videa
- Sučelje za prikaz slike
- VGABIOS
- Jedinica za procesiranje



# Arhitektura jezgri ALU i CPU vs GPU

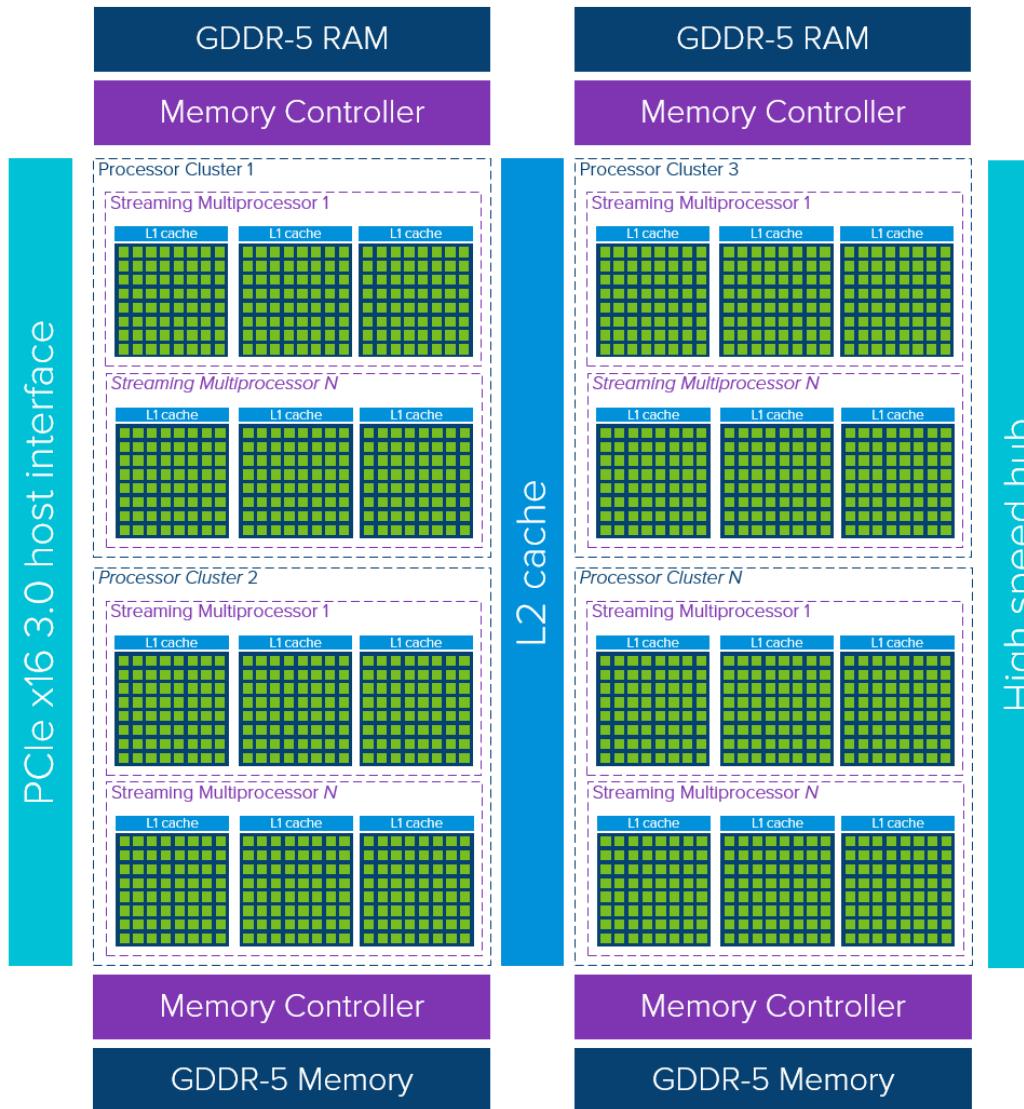
- ALU (Arithmetic Logic Unit) – komad sklopolja optimiziran za izvođenje programa za jedan entitet (primjeri vrh ili fragment) (za GPU primjene)
  - Osnovne su funkcije za obradu brojeva s pomičnim zarezom i cijelim brojevima
  - Imaju funkciju za dodavanje i množenje u jednom koraku
  - Često imaju i druge funkcije poput usporedbe podataka, pohrane, čitanja, matematičkih operacija poput sinusa i kosinusa
- CPU – cilj izvršiti instrukciju što brže uz mogućnost mijenjanja operacija
- GPU – cilj optimizacija odnosno izvršenje velikog broja operacija istovremeno
- CPU ima puno manje jezgri od GPU-a
- MythBusters [video GPU vs CPU](#)



# Organizacija današnjih GPU

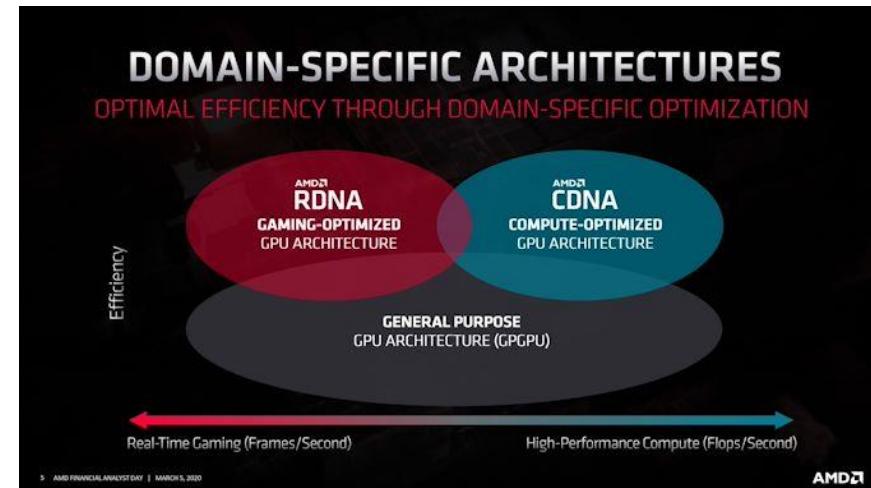
- Čip podijeljen na grupe jezgri koje rade neovisno jedni od drugih a sastoje se od više ALU-a
  - Compute Units (AMD)
  - Streaming Multiprocessors (Nvidia) - SM
- Više jezgri po pojedinom SM-u
- SM-ovi izvode istu operaciju na različitim elementima
- Svaki SM ima L1 predmemoriju te dijele L2 predmemoriju
- L2 predmemorija se veže sa CPU-om, memorijom sustava i drugim uređajima putem sabirnica (primjerice PCIe)
- Visoka propusnost
  - Ugrađena L2 predmemorija kojoj se može paralelno pristupiti
  - Najčešće spojeni na vrlo brze memorije koje se nalaze na samoj grafičkoj kartici DDR5, DDR6

# Organizacija današnjih GPU - primjer



# Mikroarhitektura GPU čipa

- Fizički raspored samog čipa, odnosno njegovih elemenata kao i implementacija programske i sklopoveske podrške za izvršavanje određenih instrukcija
- Arhitekture Nvidia  
  - Pascal 2016
  - Volta 2017
  - Turing 2019
  - Ampere 2020
- Arhitekture AMD-a  
  - Terascale 2007
  - Graphics Core Next (GCN) 2012
  - Radeon DNA 2019
  - CDNA 2019 GPGPU
  - Radeon DNA 2



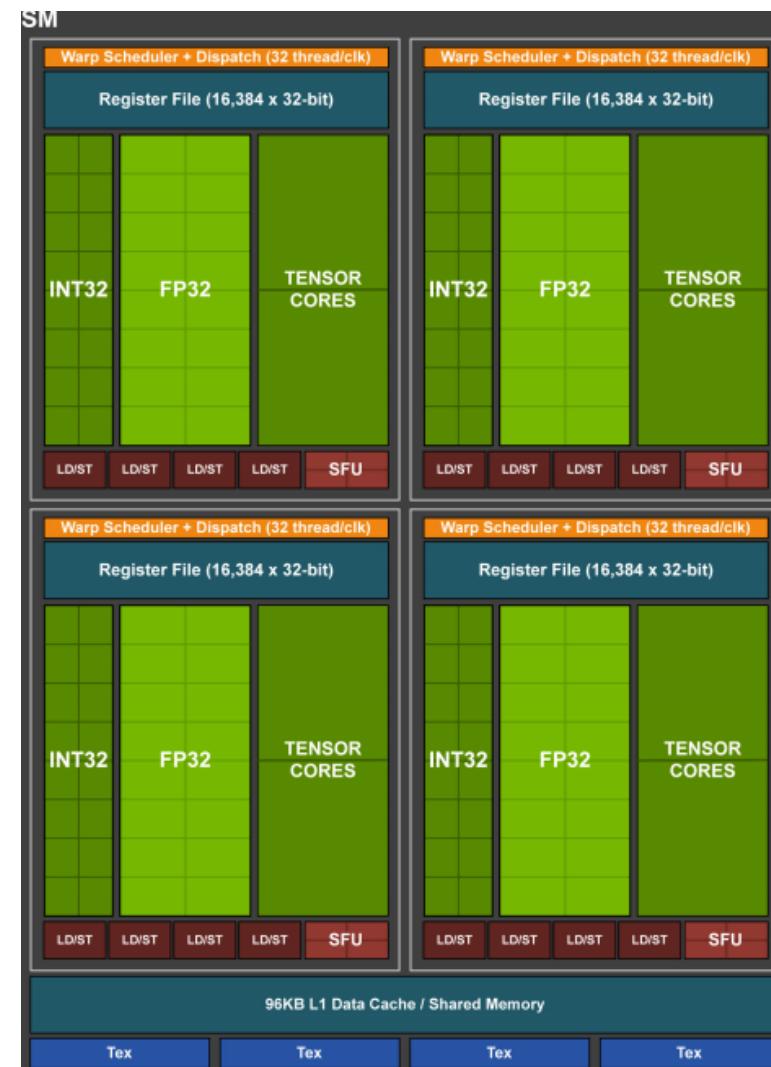
# Pascal arhitektura

- Svaki SM ima 4 bloka od kojih svaki ima 32 ALU-a (svaki blok može procesirati 4 grupe dretvi s 32 dretve po grupi – na slici prikazana samo 2)
- CUDA jezgra – izvodi instrukciju
- SFU – izvode operacije nad brojevima s pomičnim zarezom (32 bita) poput korijenovanja, sinusa, kosinusa, logaritmiranja i potenciranja
- LD/ST – Load/Store jedinice koje pristupaju memoriji
- DP unit – Double Precision – 64 bitne operacije
- 2 L1 registra (24kB svaki)
- 8 teksturnih memorija
- Vjerojatno se koristi ranglista prioriteta za prebacivanje grupa dretvi



# Turing arhitektura

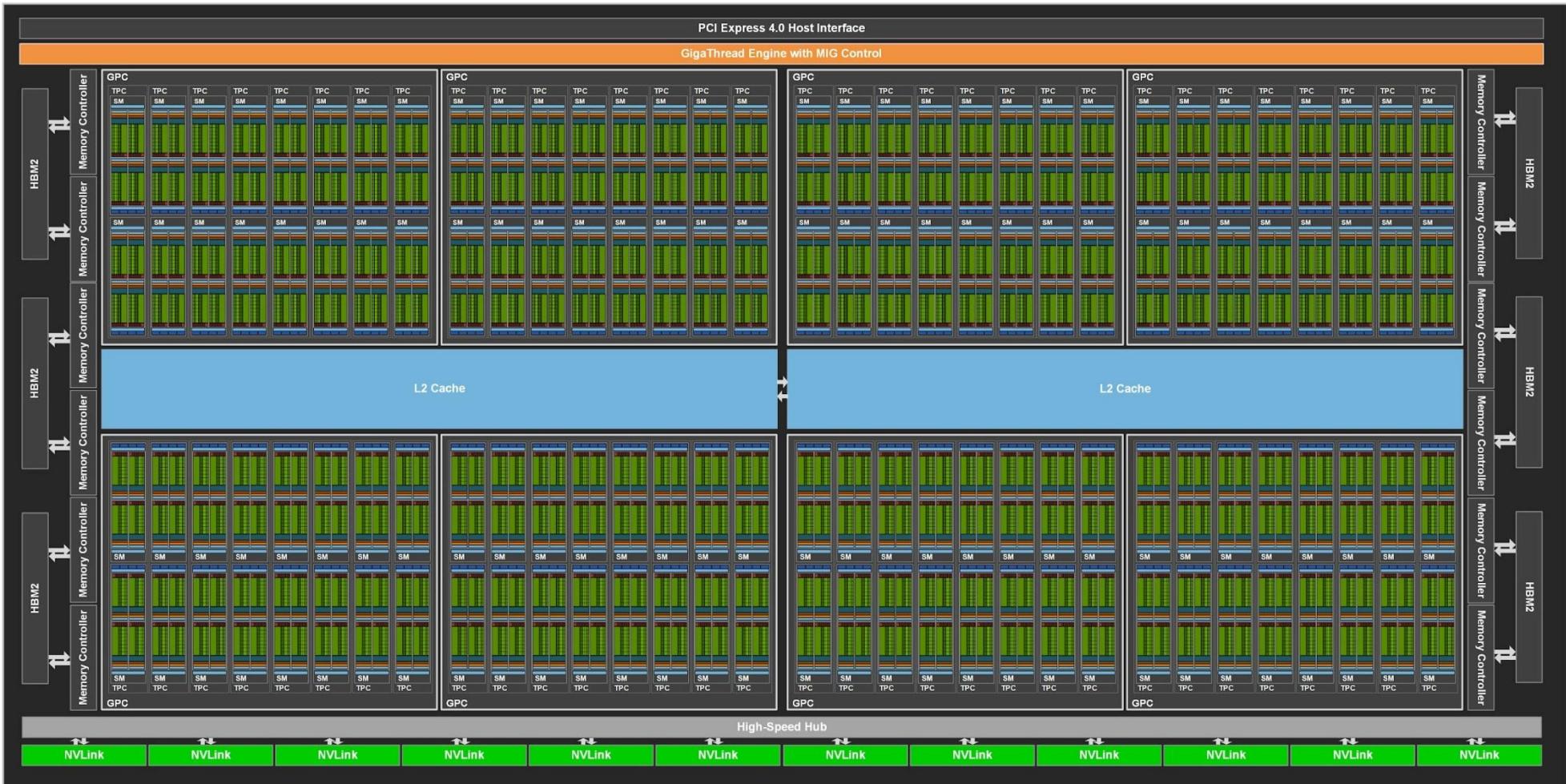
- Omogućava sklopovski izračun praćenja zrake
- INT32 – jezgre za operacije nad cijelim brojevima
- FP32 – jezgre za operacije nad brojevima s pomičnim zarezom
- Tensor jezgre – specijalne jezgre koje su namjenjene prvenstveno operacijama s matricama te primjeni u AI području odnosno za GPGPU kartice



# Ampere arhitektura

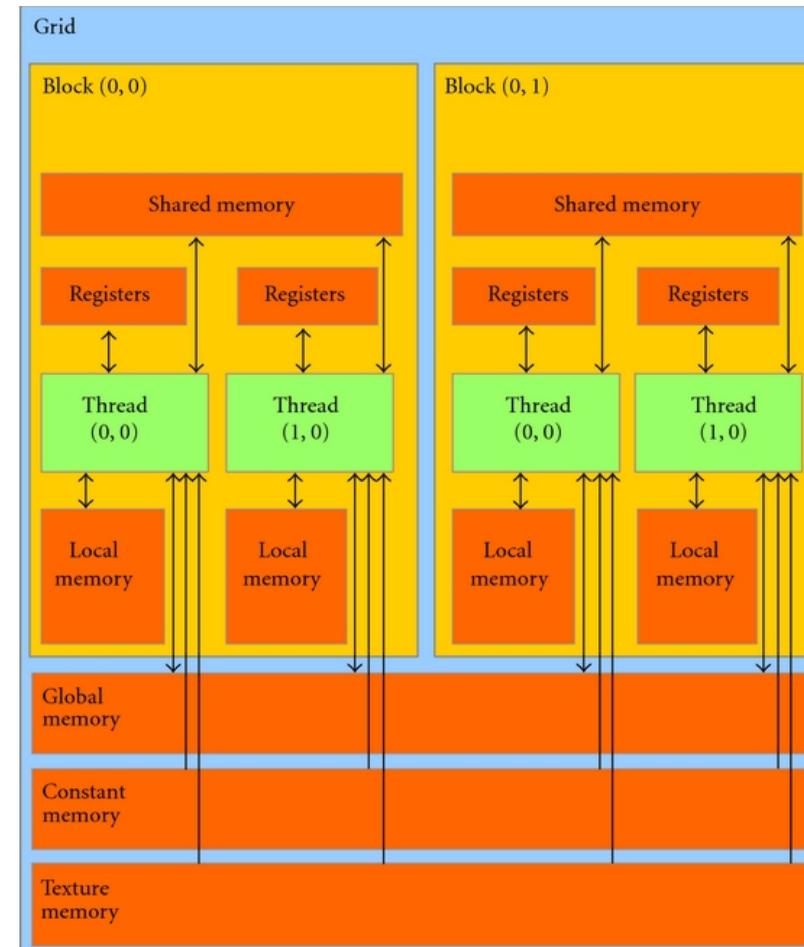


# Primjer Ampeere GA100 kartice



# Organizacija memorije

- Registri – najbrža memorija
- Lokalna memorija – memorija u koju se prelivaju podaci, može ići iz registara u L1, u L2 te u globalnu memoriju sustava (DRAM)
- Globalna memorija – današnji GPU imaju u prosjeku 2GB te do 150x sporija od operacija na registrima
- Dijeljena memorija
  - Unutar SM-a
  - Malo kašnjenje ( $\sim 5\text{ns}$ )
  - Dijeli sklopolje sa L1 memorijom
- L1/L2 predmemorija – drži podatke o pristupu lokalnoj i globalnoj memoriji
- Konstante i tekstuure



# Virtualna okruženja

Igor S. Pandžić, Tomislav Pejša

“Specijalni” efekti

# Što je specijalno?

- Većina prirodnih vizualnih efekata može se simulirati na računalu
  - ako imate dovoljno vremena
- Sve više efekata je moguće realizirati u stvarnom vremenu
- Specijalni efekt – sve što nije rutinski dio grafičkog protočnog sustava
- Dolaskom *shadera* pojam sve više gubi smisao – svi efekti su sada „specijalni“

# Pregled predavanja

- Metode teksturiranja
- Tehnike panoa
- Zrcaljenje
- Sjene

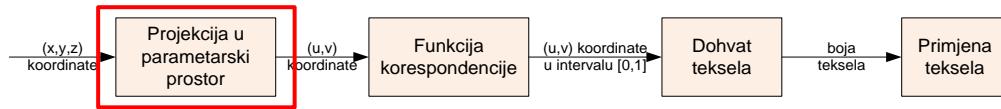
# Metode preslikavanja teksture

- Poopćeno teksturiranje
- Animacija teksture
- Preslikavanje materijala
- Preslikavanje prozirnosti (alpha mapping)
- Preslikavanje svjetlosti (light mapping)
- Preslikavanje okoline (environment mapping)
- Preslikavanje neravnina (bump mapping)

# Poopćeno teksturiranje, projekcija

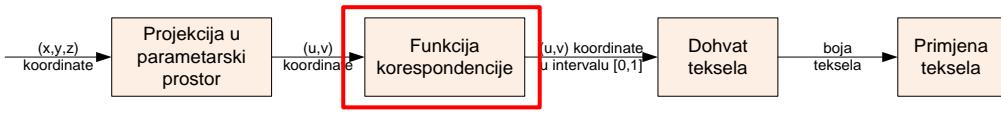


- Teksturiranje (texture mapping) = preslikavanje *uzorka* na geometriju predmeta
- Uzorak može doći iz bilo kakvog izvora, uklj. slike
- Uzorak se može koristiti na puno različitim načina, uklj. jednostavno preslikavanje uzorka na boju



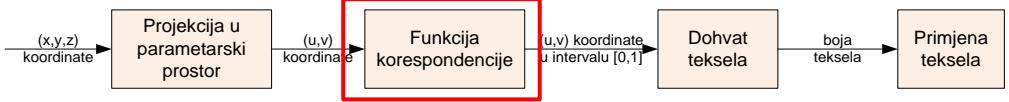
# Projekcija u parametarski prostor

- $u,v$  koordinate obično zadane za svaki vrh (prilikom modeliranja), te se interpoliraju
- Više tekstura na predmetu (multitexturing) = više parova  $u,v$  koordinata po vrhu
- 3D teksture  $(u,v,w)$  – npr. mramor, drvo
- 1D teksture – npr. bojenje terena po visini

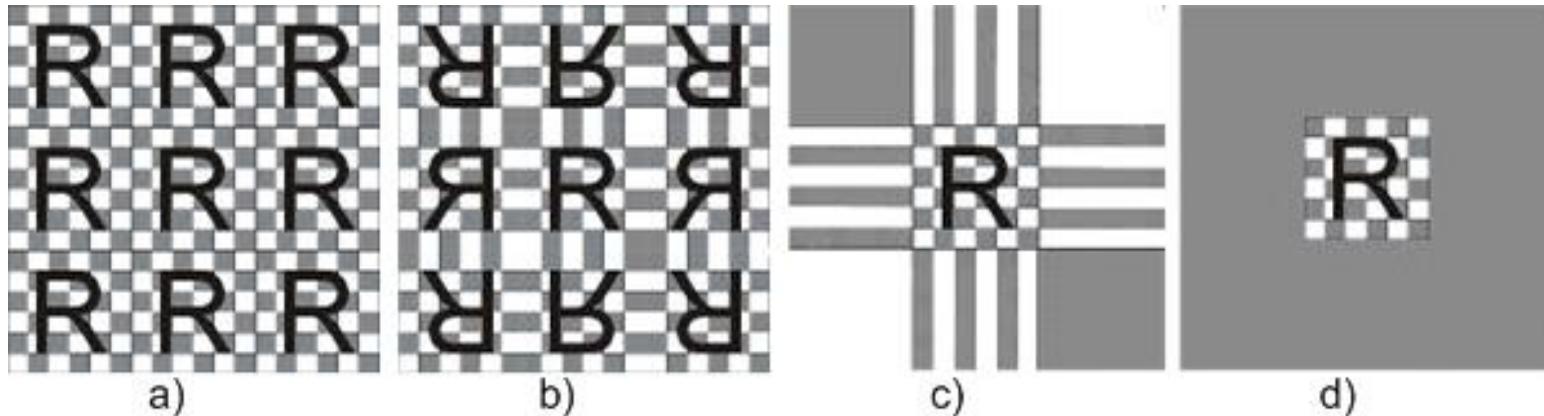


# Funkcija korespondencije (1/2)

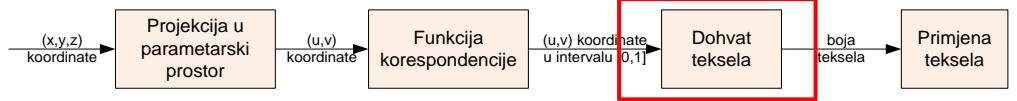
- Preslikavanje  $u,v$  koordinata u prostor tekstuure (vrijednost ograničena na  $[0,1]$ )
- 2D slika –  $(0,0)$  donji lijevi kut;  $(1,1)$  gornji desni kut
- Koordinate neovisne o dimenzijama – lako zamijeniti sliku (npr. manja razlučivost radi štednje memorije)



# Funkcija korespondencije (2/2)

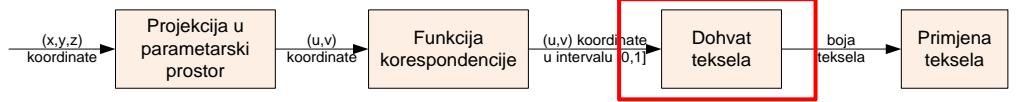


- Funkcije korespondencije:
  - Ponavljanje (wrap, repeat, tile)
  - Zrcaljenje (mirror)
  - Ponavljanje ruba (clamp)
  - Ograničavanje (border)
- Na  $u,v$  koordinate možemo primijeniti matričnu transformaciju u shaderu (npr. translacija ili rotacija teksture)



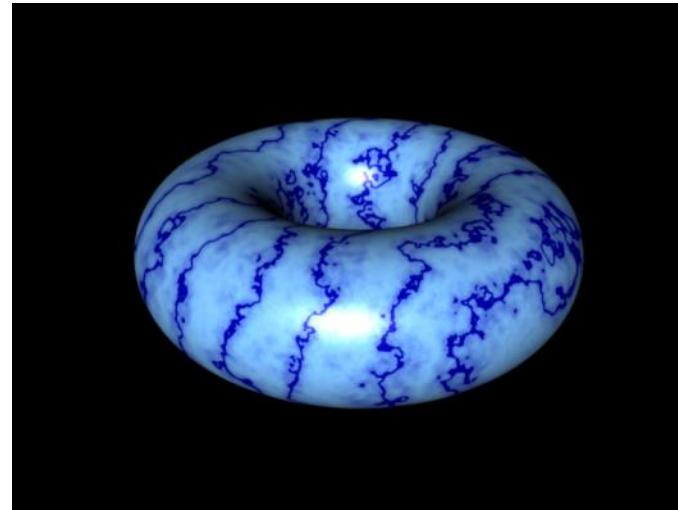
# Dohvat teksela (1/2)

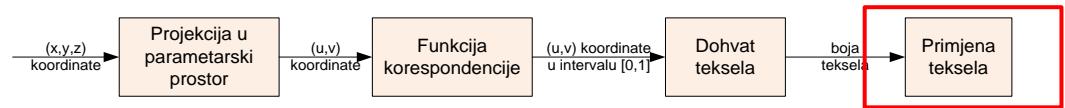
- = uzorkovanje teksture
- Teksel – točka teksture na u,v koordinatama:
  - RGB – 24-bit (8-bit po komponenti boje)
  - RGBA – 24-bit + 8-bit  $\alpha$  kanal
  - HDR – visoki dinamički raspon; više od 8-bit po komponenti
- (Ne mora se koristiti kao boja, nego npr. normala, parametar materijala...)



# Dohvat teksela (2/2)

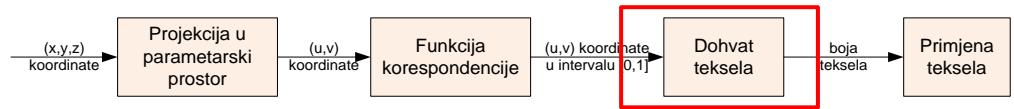
- Uzorkovanje = čitanje memorijске lokacije
- Moguće koristiti filtriranje – čitanje susjednih lokacija i interpolacija rezultata
- No, teksel se može i izračunati, npr. matematičkom operacijom (proceduralne teksture)





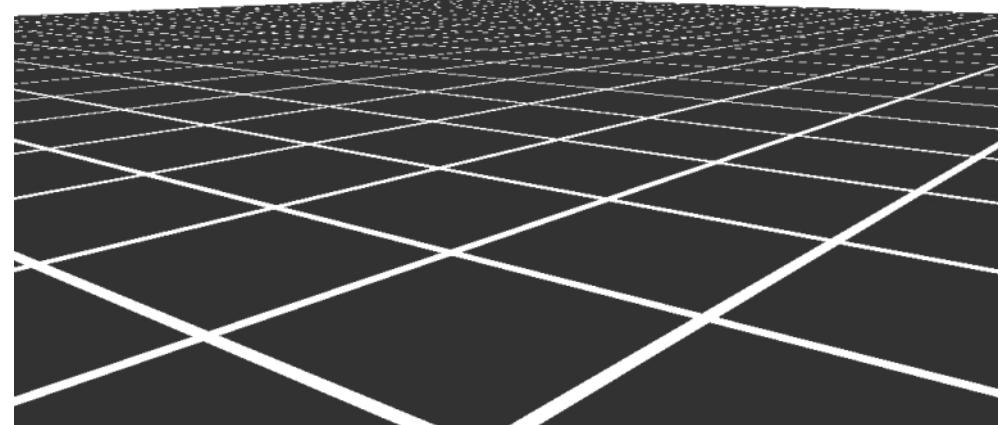
# Primjena tekscela

- Kako vrijednost tekscela utječe na konačnu boju točke koja se sjenča?
  - Izravna primjena kao boje
  - Težinsko miješanje s bojom osvjetljenja
  - Parametar materijala (difuzna ili spekularna tekstura, tekstura sjaja)
  - Normala (v. preslikavanje normala)
  - ...



# Filtriranje tekture

- Ako je tekstura = slika, javljaju se problemi poduzorkovanja ili naduzorkovanja (ovisno o udaljenosti od kamere)
- Idealno – korespondencija piksel-teksel 1:1
  - ◆ Slika uvećana (predmet bliže kamери) → efekt blokova
  - ◆ Slika umanjena (predmet dalje od kamere) → aliasing



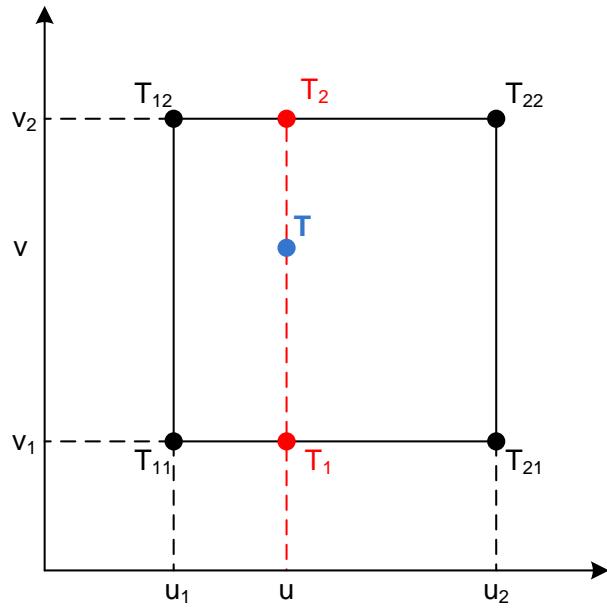
# Metoda najbližeg susjeda

- Dohvaća se teksel najbliži u,v koordinatama
- Naglašen efekt blokova

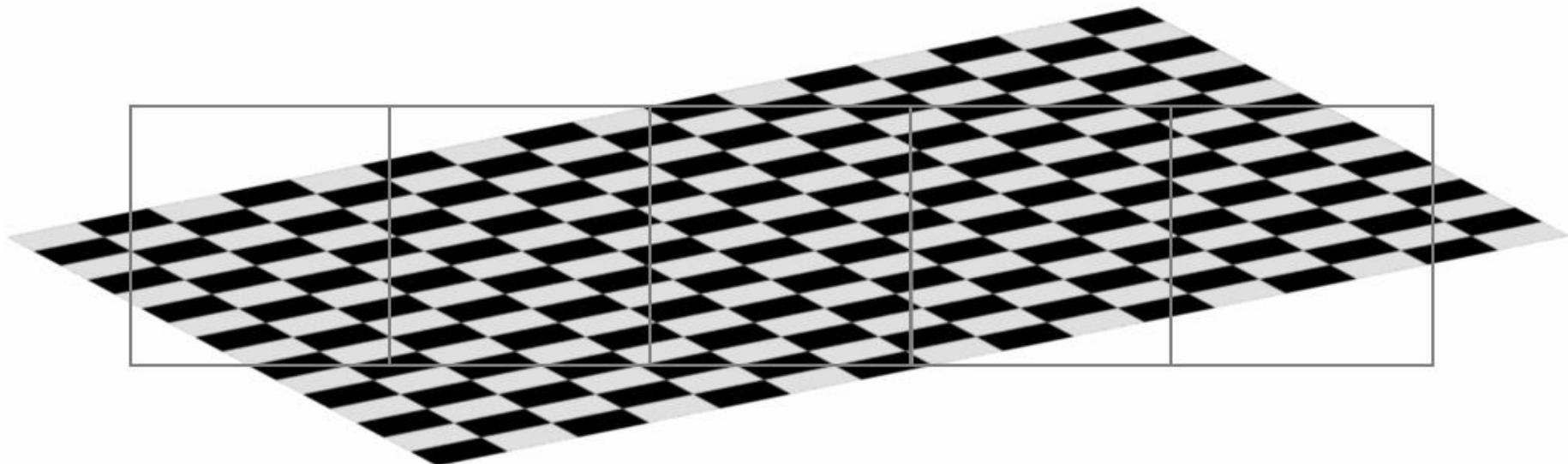


# Bilinearna interpolacija

- Interpolacija 2x2 najbližih texsela
- 2 uzastopne linearne interpolacije
- Zamućena slika, ublažen efekt blokova
- Sklopopovski podržana



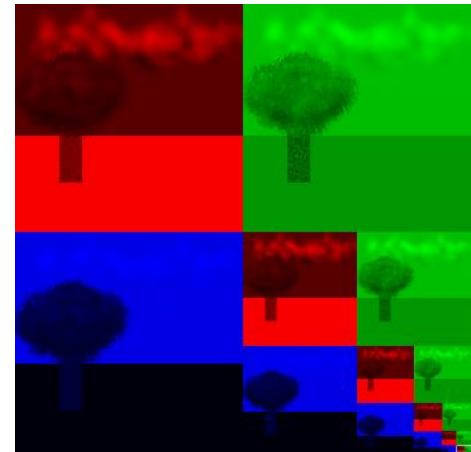
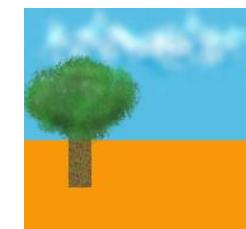
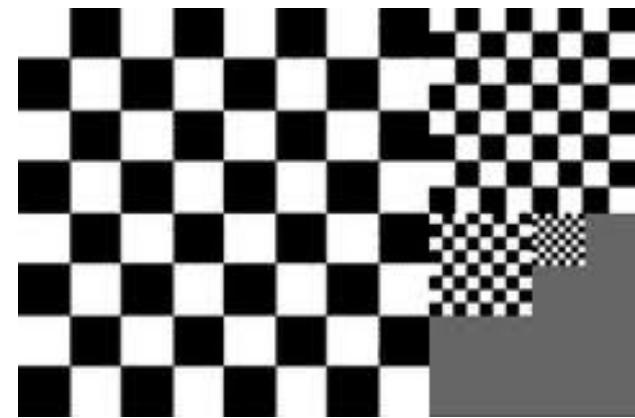
# Mipmapiranje (1/3)



- 1 piksel prekriva više teksele
- Ispravno bi bilo zbrojiti njihove utjecaje, ali ne možemo u stvarnom vremenu

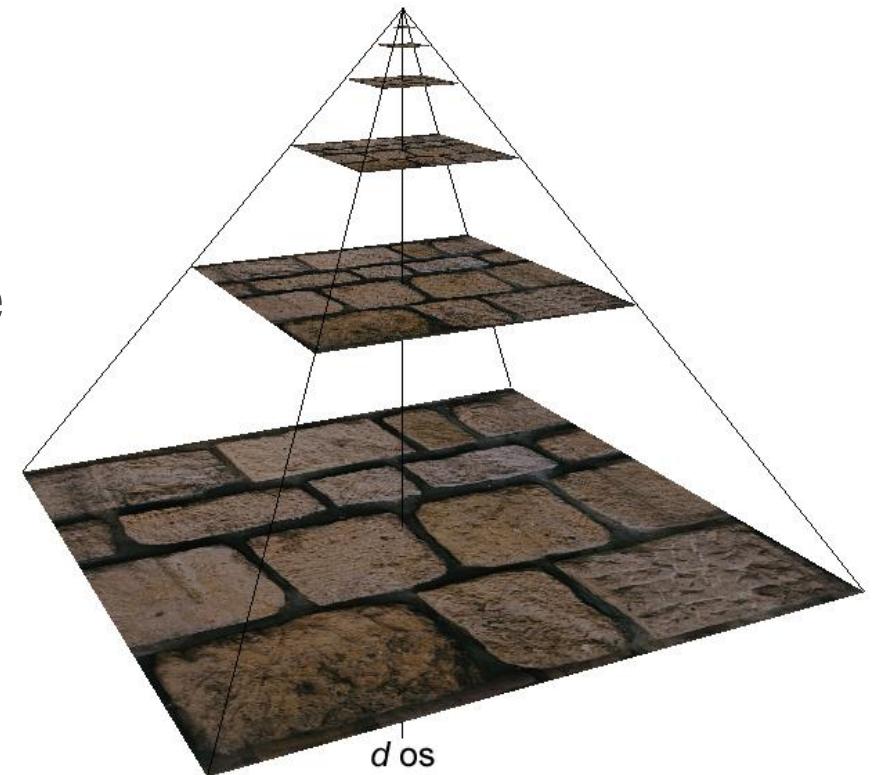
# Mipmapiranje (2/3)

- Mipmap – originalna slika + lanac umanjenih, filtriranih slika (svaka 2x manja od prethodne)
- Uzorkuju se texseli one slike čija veličina najbolje odgovara veličini piksela
  - ◆ Mipmap slike moguće pohraniti zajedno (datoteka veća za samo 1/3 jer  $\frac{1}{4} + \frac{1}{16} + \frac{1}{64} + \frac{1}{256} + \dots \rightarrow \frac{1}{3}$ )
  - ◆ Može ih se generirati dinamički (ugrađena sklopovska podrška)



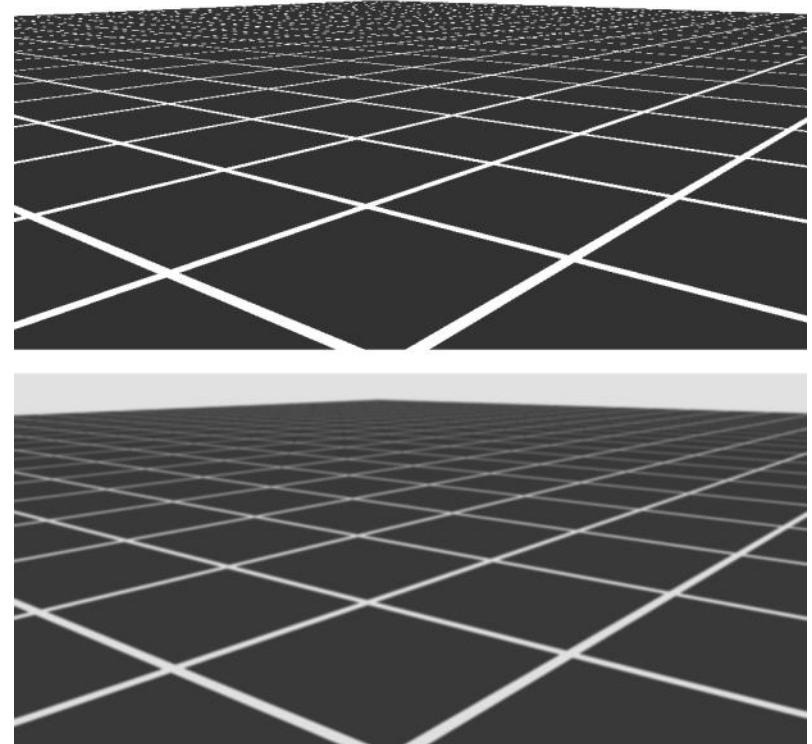
# Mipmapiranje (3/3)

- Parametar  $d$  (razina detalja, level od detail) – mjera prekrivanja bloka teksela pikselom
- Za uzorkovanje mipmap teksture trebamo  $u, v$  i  $d$  (veći  $d \rightarrow$  odabire se manja slika)
- 2 načina izračuna  $d$ :
  1. Najdulja stranica četverokutaprojekcije piksela na teksturu
  2. Iz diferencijala  $u$  i  $v$  duž osi zaslona  $x$  i  $y$  ( $du$  i  $dv$ )



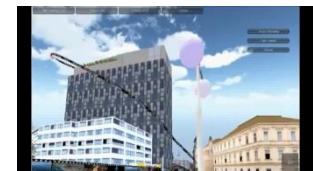
# Trilinearna interpolacija

1. Odredi vrijednost  $d$
2. Pomoću  $d$ , odredi dvije susjedne mipmap slike
3. Uzorkuj obje slike na  $u, v$  (uz korištenje bilinearne interpolacije)
4. Linearno interpoliraj rezultat po  $d$



# Animacija tekstuра

- Promjena slike tekstuра u vremenu
1. Vremenski niz slika koje se izmjenjuju:
    - Efekti vatre, dima, eksplozija...
    - Slike pohranjene kao 3D tekstura (w koord. = vrijeme) ili polje tekstuра (SM4.0+)
    - Gladak prijelaz između tekstuра težinskim miješanjem (npr. mrštenje lika)
  2. Animiranje teksturnih koordinata:
    - Translacija, rotacija, skaliranje i smicanje koordinata u programu za sjenčanje
    - Tok duž površine (slap, rijeka, kapi vode), efekti dima i vatre...



# Preslikavanje materijala

- Umjesto boje, teksture pohranjuju parametre materijala
- Parametri definirani za svaku točku predmeta umjesto konst. za čitav predmet
- Teksture materijala (za Phongov model osv.):
  - Tekstura difuzne boje (diffuse color map) –  $k_d$
  - Tekstura spekularne boje (specular color map) –  $k_s$
  - Tekstura sjaja (gloss map) –  $n$

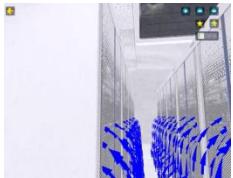
# Preslikavanje prozirnosti (1/2)

- Alpha mapping
- $\alpha$  kanal tekture definira prozirnost u svakoj točki predmeta
- Efekt naljepnice:
  - Slika je neki nepravokutan oblik (npr. krug), ostali dijelovi prozirni
  - Moguće dodavati u scenu dinamički rupe od metaka, mrlje od krvi itd.)



# Preslikavanje prozirnosti (2/2)

- Primjena prozirnosti na sam predmet:
  - Predmet proziran gdje je  $\alpha = 0$
  - Prikaz složenog predmeta jednim poligonom (npr. vegetacija)
  - Uz animaciju teksture – plamen, dim, eksplozije...



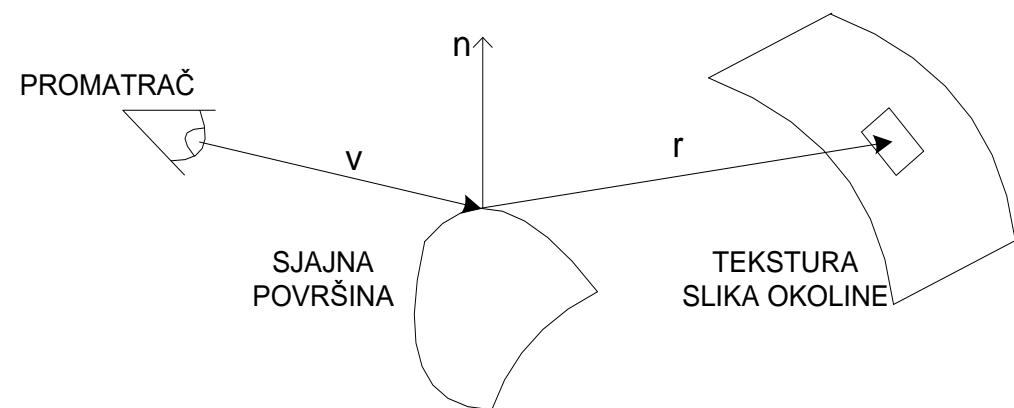
# Preslikavanje svjetlosti (light mapping)

- Simulacija efekata difuzne svjetlosti
- Osvjetljenje se računa unaprijed (npr. isijavanje) i pohranjuje u teksture svjetlosti (light map)
  - ◆ (Moguće zato što difuzna svjetlost ne ovisi o položaju promatrača)
  - ◆ Teksture svjetlosti koriste se u realnom vremenu



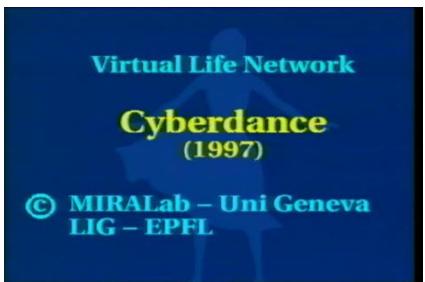
# Preslikavanje okoline (environment mapping)

- Simuliranje zrcaljenja na oblim površinama
- Ideja – slika okoline (gledano iz pozicije predmeta) pohrani se u teksturu, koja se „lijepi“ na predmet u realnom vremenu
- u,v koordinate ne možemo izračunati unaprijed, zrcaljenje ovisi o položaju promatrača (sjetite se spekularnog odsjaja!)



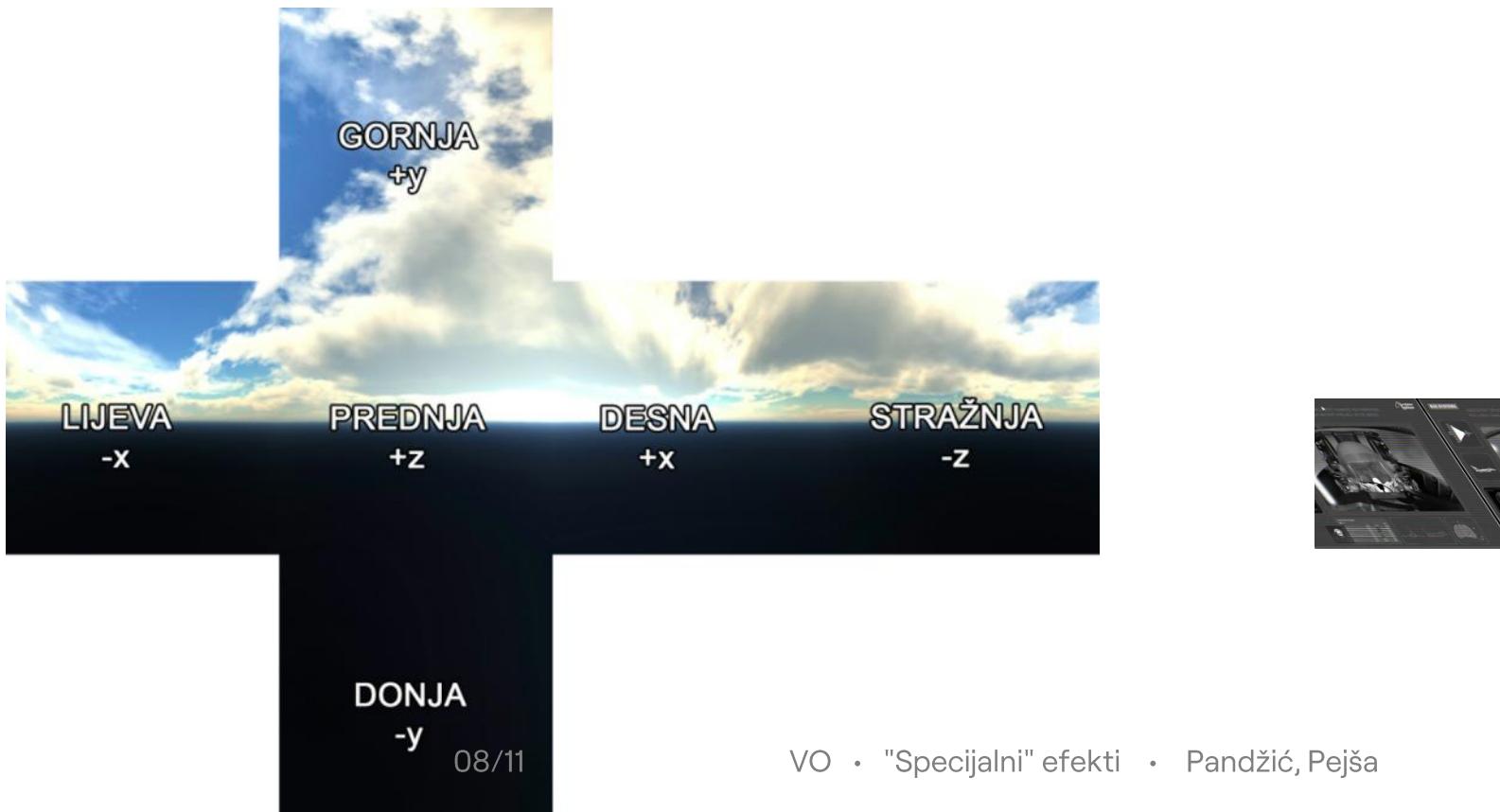
# Kuglasto preslikavanje

- Projekcija okolina na kuglu oko predmeta



# Kockasto preslikavanje (1/2)

- Projekcija okoline na 6 strana kocke oko predmeta
- Rezultat – tekstura od 6 dijelova (cube map)

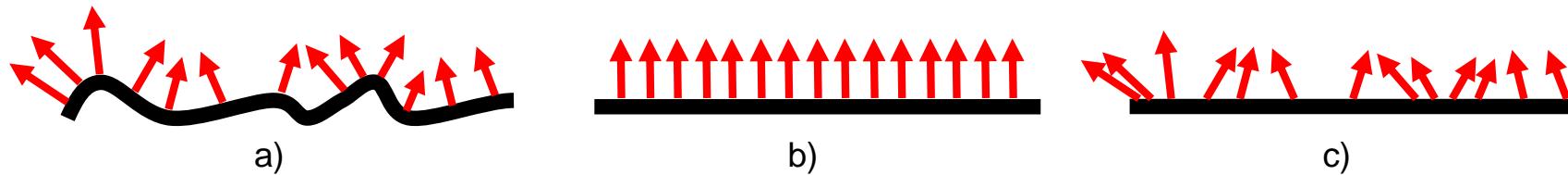


# Kockasto preslikavanje (2/2)

- ◆ Postupak iscrtavanja:
  1. Vektor r se normalizira
  2. Određuje se najveća komponenta r (x, y ili z), te njen predznak; iz njih se određuje koju stranu uzorkovati
  3. Uzorkuj odgovarajuću stranu (u,v koordinate – 2 manje komponente r)
    - (U jezicima za sjenčanje – texCUBE, textureCube)
- ◆ Prije SM4.0 – 6 prolaza za iscrtavanje kockaste teksture
- ◆ SM4.0+ – dovoljan jedan prolaz, ako umnožimo geometriju 6x u procesoru geometrije

# Preslikavanje neravnina (bump mapping)

- Skup tehnika za simuliranje neravnih, hrapavih površina
- Phongov model osvjetljenja – difuzna i spekularna komponenta ovise o normali
- Poremećajem normale dobivamo privid neravnina
- Tekstura neravnina – pohranjuje podatke potrebne za poremećaj normala
- (Iluzija se gubi pod oštrim kutem)



# Preslikavanje normala (normal mapping)

- Tekstura neravnina pohranjuje poremećene normale – tekstura normala
- Uzorkuje se u procesoru točaka, dobivena normala se koristi za proračun osvjetljenja



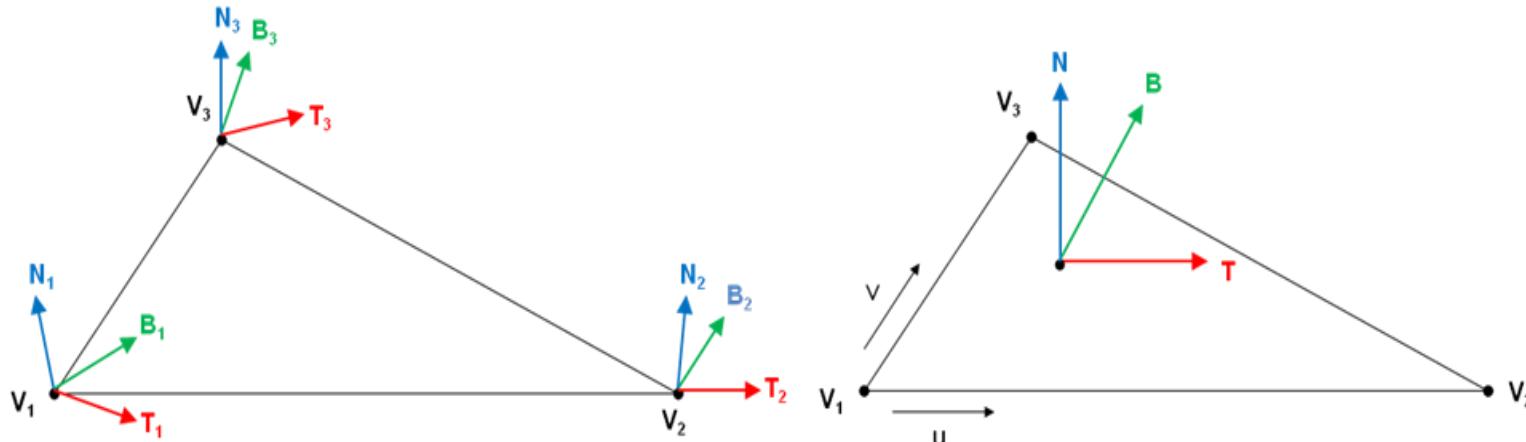
# Koordinatni sustav za normale

- U kojem koord. sustavu zadati normale?
- Globalni koordinatni sustav:
  - Nepraktičan, ne smijemo pomicati niti deformirati predmet
- Koordinatni sustavi predmeta:
  - Moguće rigidne transformacije
  - Nije moguća deformacija predmeta (npr. skinning kod likova)
- Najbolje rješenje – koordinatni sustav tangente!

# Koordinatni sustav tangente (1/2)

- Zadan trima vektorima:
  - Vektor tangente T (na površinu predmeta)
  - Vektor bitangente B (na površinu predmeta)
  - Vektor geometrijske normale N
- N već znamo odrediti (okomit na poligon)
- T i B – vektori u smjeru rastućih u,v koordinata:
  - a. Najčešće zadani u svakom vrhu (kao i sama normala), te se interpoliraju duž površine
  - b. Mogu se računati dinamički u proc. točaka iz diferencijala u i v (funkcije ddx i ddy)

# Koordinatni sustav tangente (2/2)



- Koord. sustav tangente varira duž površine – normale ostaju ispravne prilikom deformacija
- $T$  i  $B$  općenito nisu okomiti  $\rightarrow$  distorzija normale (no ne predstavlja problem)
- Pitanje: Zašto je tekstura normala plava?

# Preslikavanje neravnina i okoline

- Environment bump mapping
- u,v koordinate za uzorkovanje teksture okoline odredimo iz poremećene normale
  - ◆ Rezultat – zrcaljenje na neravnoj površini (npr. vode)
  - ◆ Povijesna važnost (1999.) – prvi primjer zavisnog uzorkovanja tekstura u GPU



# Preslikavanje paralakse, reljefa

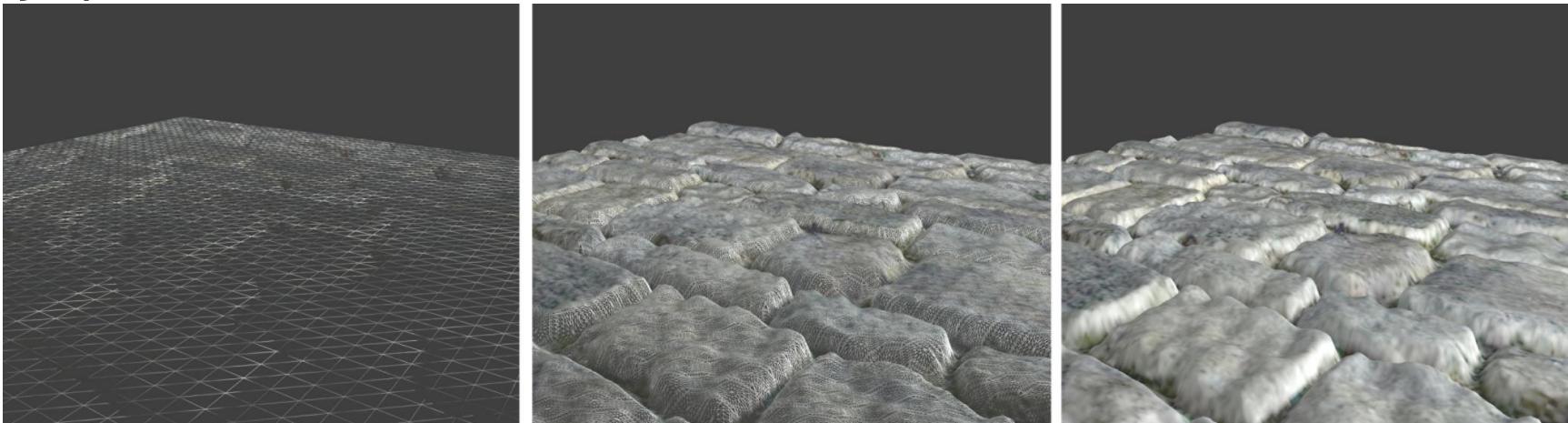


# Preslikavanje pomaka (displacement mapping) (1/2)

- Prethodne tehnike – iluzija neravnina, trikovi sa sjenčanjem
- Preslikavanje pomaka – tekstura visina se koristi za deformiranje same geometrije predmeta
- Jednostavnija izvedba (SM3.0):
  - Ulaz – „ravna“ geometrija
  - (Vertex shader) Za svaki vrh se uzorkuje tekstura visina, te se vrh pomiče
  - Nije „pravi“ bump mapping (ne povećava detaljnost geometrije)
  - Korisno za animaciju neravnina (npr. valovi na moru)

# Preslikavanje pomaka (displacement mapping) (2/2)

- „Pravo“ preslikavanje pomaka:
  - (Geometry shader) Uzorkuje se tekstura visina, te se stvaraju dodatni trokuti za neravnine
  - Potrebna sklopovska podrška za teselaciju (SM5.0)
- Odlični vizualni rezultati
- Detekcija presjeka i sudara postaje problem – vrši se u aplikaciji, prije unošenja pomaka

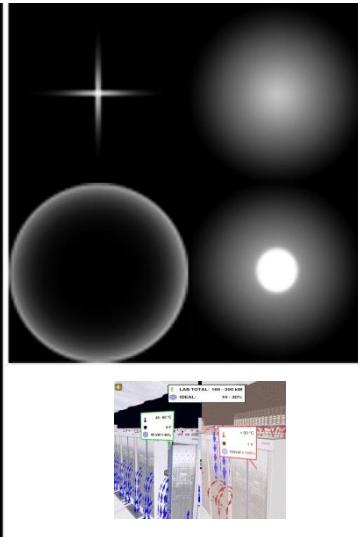
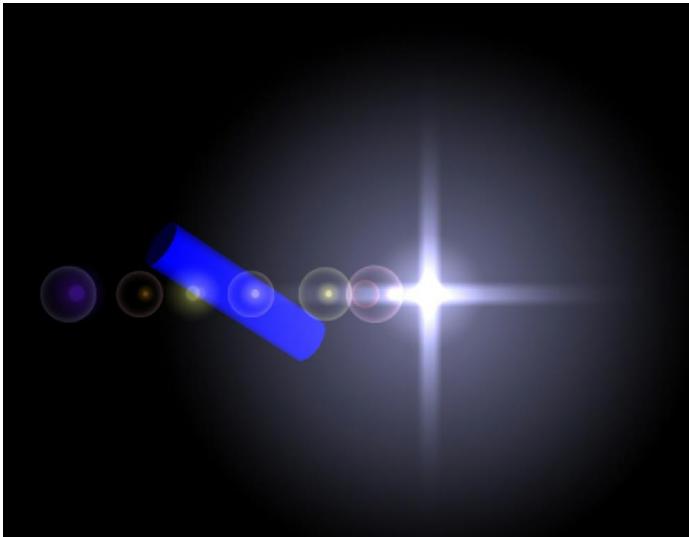


# Tehnike panoa (billboard)

- Za automatsku orijentaciju predmeta prema kameri
- Predmet se uvijek vidi iz istog kuta
- Npr. tekst, vegetacija, oblaci...
- Na pravokutnik (pano) se lijepi 2D slika predmeta (sprite)
- Često se koriste preslikavanje prozirnosti i/ili animacija tekstu

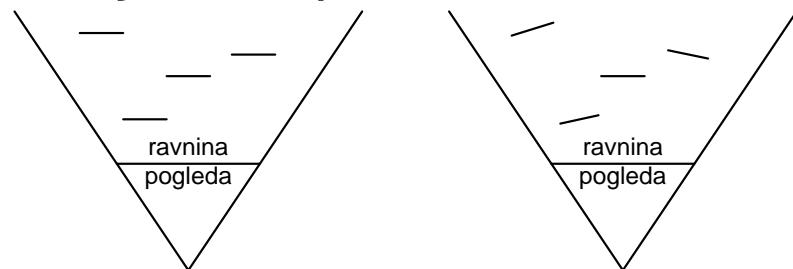
# Pano poravnat sa zaslonom

- Uvijek ista orijentacija prema zaslonu
- Normala uvijek okomita na ravninu projekcije
- Vertikalna os (up axis) = vertikalna os kamere
- Primjeri – tekst, odbljesak na objektivu (lens flare), sustavi čestica...



# Globalno orijentirani pano

- Vertikalna os djelomično globalno orijentirana – predmet ne može rotirati oko osi pogleda
- Za predmete koji su fizički prisutni u sceni, a nisu radijalno simetrični – oblaci, dim, eksplozije...
- Normala na pano:
  - a. Okomita na ravninu proj.  
– rubni panoi izgledaju „iskriviljeno”
  - a. Usmjerena prema kameri



# Osni pano

- Predmet prema kameri rotira oko fiksne osi
- Primjer – simulacija drveća

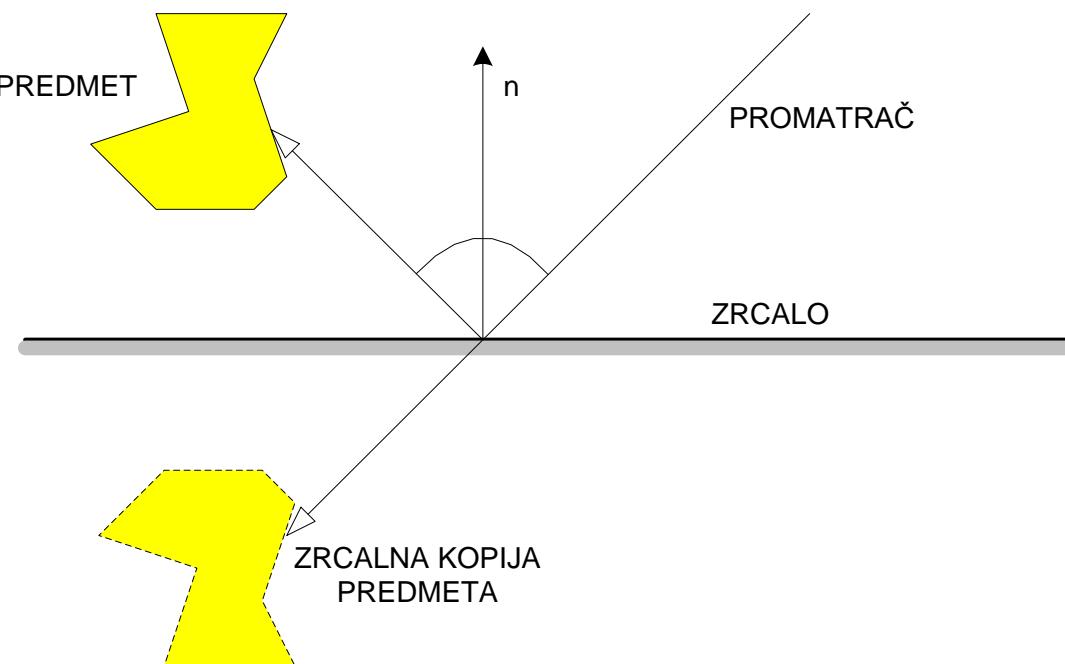


# Zrcaljenje

- Jednostavno – zraka se odbija simetrično na normalu (idealno zrcalo)
- Praćenje zrake (ray tracing) daje dobro rješenje, ali skupo
- Preslikavanje okoline prikladno samo za udaljene predmete

# Zrcaljenje na ravnini (1/3)

- Pojednostavljen ali često koristan slučaj: jedno zrcalo u sceni, i to ravno
- Ideja: konstruiramo kompletну kopiju scene zrcaljenjem oko ravnine zrcala, zatim crtamo scenu i kopiju

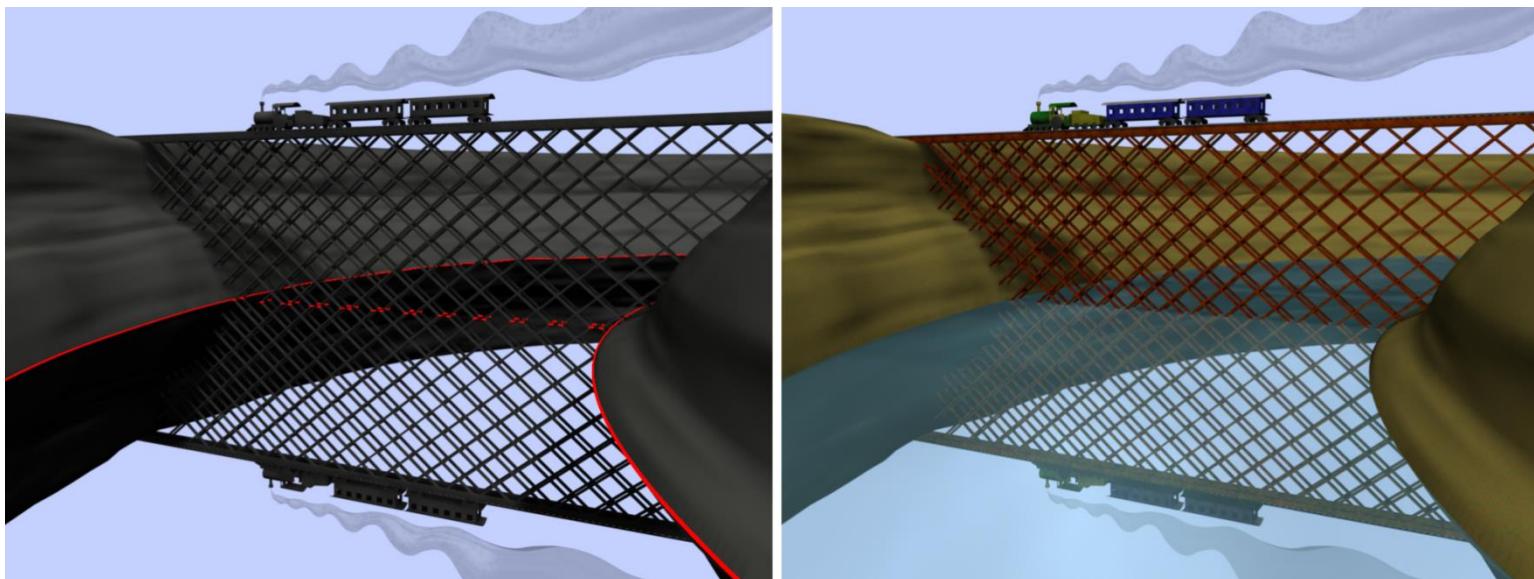


# Zrcaljenje na ravnini (2/3)

- Najjednostavniji slučaj: ravnina  $y=0$ 
  - Zrcaljenje izraženo matricom  $S(1, -1, 1)$
- Za općenu zrcalnu plohu zadalu točkom  $P$  i normalom  $N$ 
  - $F = T(-P) R(N, (0, 1, 0))$  preslikava plohu na  $y=0$
  - Konačna zrcalna matrica je  $M = F S(1, -1, 1) F^{-1}$

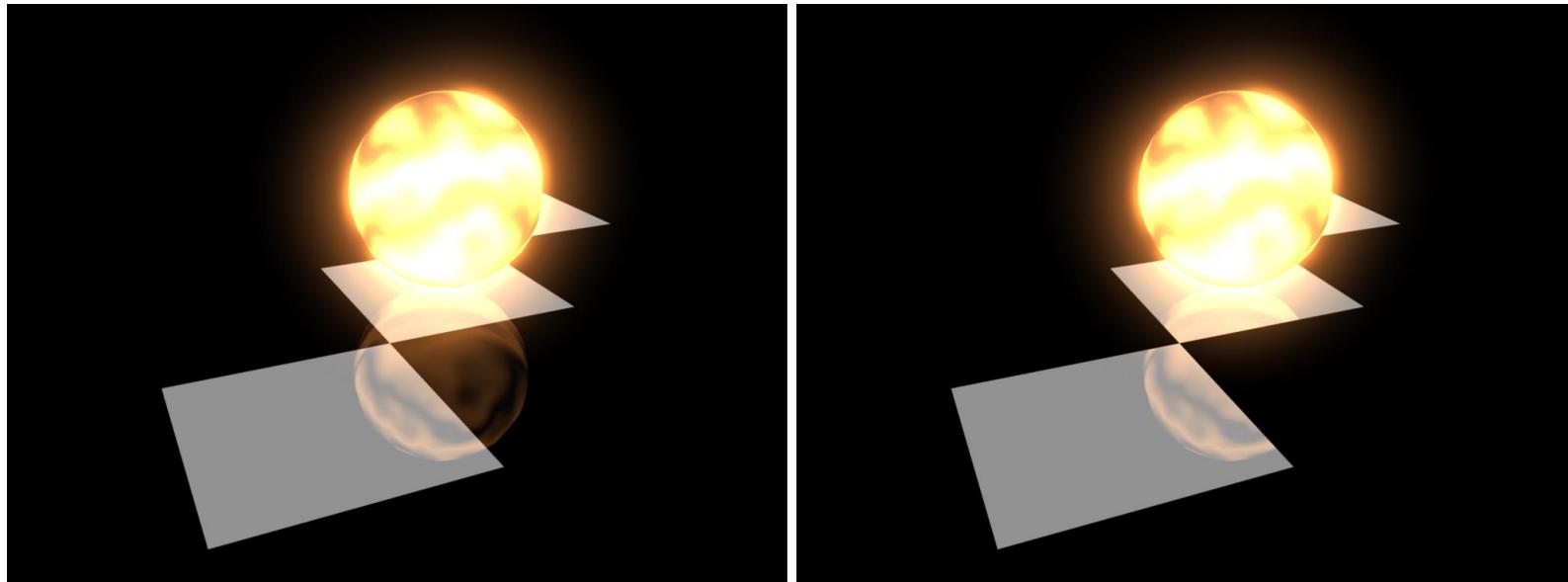
# Zrcaljenje na ravnini (3/3)

1. IsCRTavanje zrcaljene geometrije (transformirane matricom M)
2. IsCRTavanje izvorne geometrije
3. IsCRTavanje zrcalne plohe
  - ◆ Faktor prozirnosti djeluje kao faktor zrcaljenja



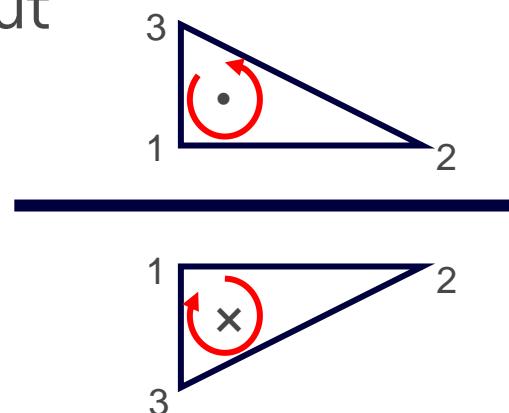
# Ograničenje zrcaljenja na dio ravnine

- IsCRTavanje zrcalne plohe u spremnik predloška
- IsCRTavanje zrcaljene geometrije samo po predlošku
- Ostatak postupka isti



# Ostali problemi

- Geometrija ispod plohe se ne smije zrcaliti
  - Sve ispod zrcala treba odrezati, direktno ili korištenjem proizvoljne odrežujuće plohe (arbitrary clipping plane)
- Odbacivanje stražnjih poligona
  - Zrcaljenje obrće redoslijed vrhova poligona → odbacivanje daje pogrešan rezultat!
  - Isključiti odbacivanje, ili zadati obrnut redoslijed vrhova



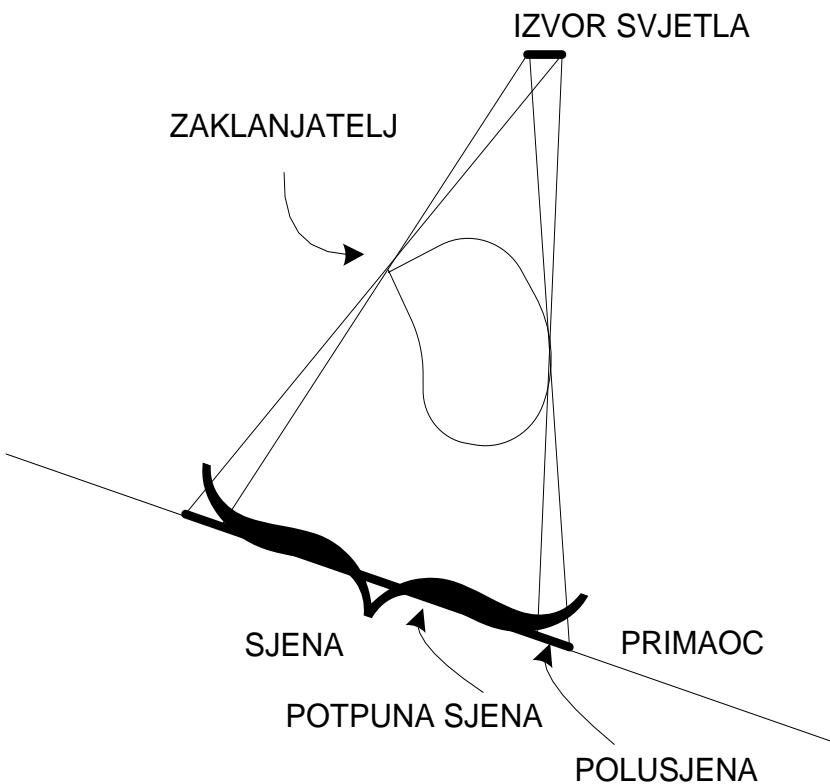
# Sjene

- Daju osjećaj o položaju predmeta (čak i kada su jako pojednostavljene)
- Sjena nastaje jer predmet zaklanja svjetlo
- Jednostavno, ali:
  - U sceni postoji velik broj potencijalnih zaklanjatelja
  - Zaklonjenost treba utvrditi u svakoj točki – skupo
  - Može biti više svjetala



# Nomenklatura sjena

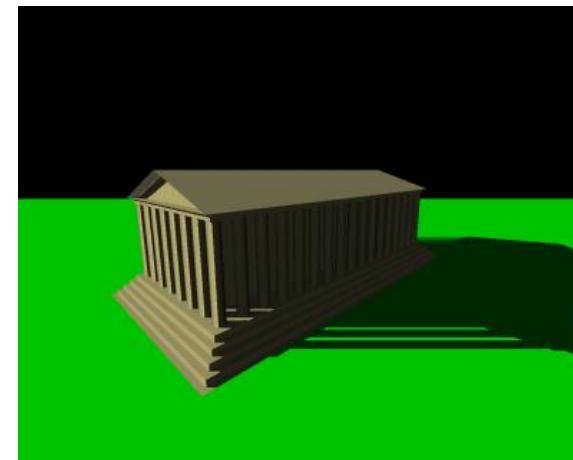
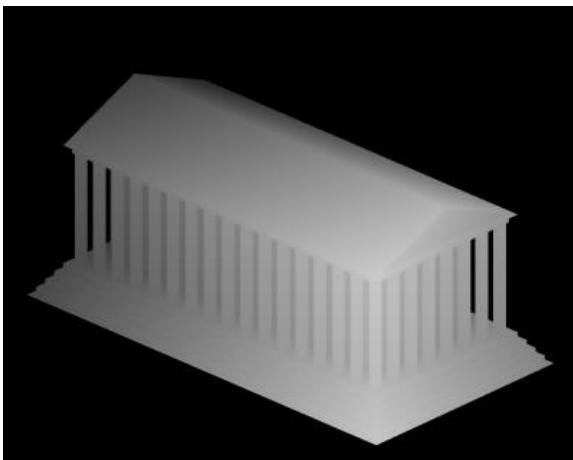
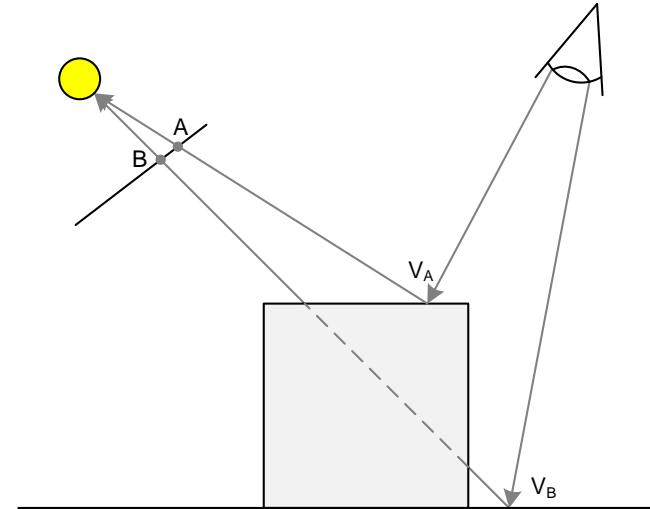
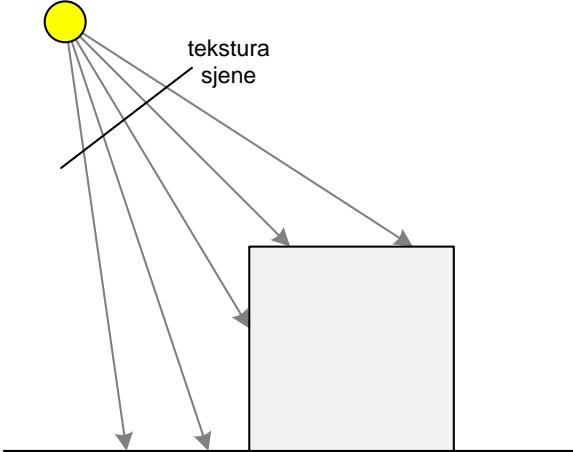
- Sjene su mekane jer izvori svjetla nisu točkasti
- Oštре sjene u prirodi rjeđe, no u grafici se često koriste (bolje išta nego ništa)



# Metoda teksture sjena (shadow map) (1/3)

- Ideja – samo točke vidljive iz perspektive svjetla trebaju biti osvijetljene
- Bacanje zrake prema svjetlu i provjera presjeka s geometrijom? – skupo, nepraktično!
- Umjesto toga koristimo *teksturu sjena*

# Metoda teksture sjena (2/3)



# Metoda teksture sjena (3/3)

1. Iscrtati scenu iz perspektive svjetla u Z-spremnik (samo dubine) → tekstura sjena (TS)
2. Iscrtati scenu iz perspektive kamere:
  - U svakoj točki uzorkovati TS
  - Ako je  $z > z_{TS}$  → točka je u sjeni, samo ambijentalno osvj.
  - (Prethodno točku transformirati u KS svjetla)
  - Samo 1 projekcijska ravnina za TS:
    - Prikladno za usmjereni svjetla (npr. Sunce)
    - Za točkasta svjetla potrebna kockasta TS (generira se u 6 prolaza)
  - Dobra skalabilnost – vrijeme izvođenja linearno s brojem poligona

# Virtualna okruženja

Igor S. Pandžić, Tomislav Pejša

Tehnike ubrzavanja iscrtavanja

# Kako iscrtavati brže?

- Mjera brzine iscrtavanja – broj slika u sekundi (engl. frames per second, FPS)
- Za interaktivnu 3D grafiku – min. 20–30 FPS
  - (Za brze akcijske igre poželjno i do 60 FPS)
- Tehnologija grafičkog sklopolvlja napreduje strahovito brzo – no, sklopolje nikad neće biti „dovoljno brzo“
- Npr. model Boeing 777 – 500.000.000 trokuta
  - GeForce GTX580 (2010), 2e9 tri/s: procjena 4fps
  - GeForce GTX1080 (2016), 11e9 tri/s: procjena 22fps

# Napredak tehnologije

- Napredak tehnologije nije rješenje – potrebne bolje metode ubrzavanja iscrtavanja
- Obradit ćemo metode:
  - Optimalan zapis poligona
  - Selektivno odbacivanje poligona
  - Tehnike razina detalja
  - Optimizacija protočnog sustava

# Optimalan zapis poligona

- Naivan pristup – u protočni sustav šaljemo svaki trokut zasebno (3 vrha po trokutu)
- Mnogi vrhovi su dijeljeni među susjednim trokutima te se obrađuju višekratno – neučinkovito!
- Možemo smanjiti broj vrhova organizacijom trokuta u spojenu strukturu:
  - Trake trokuta
  - Lepeze trokuta
  - Mreže trokuta

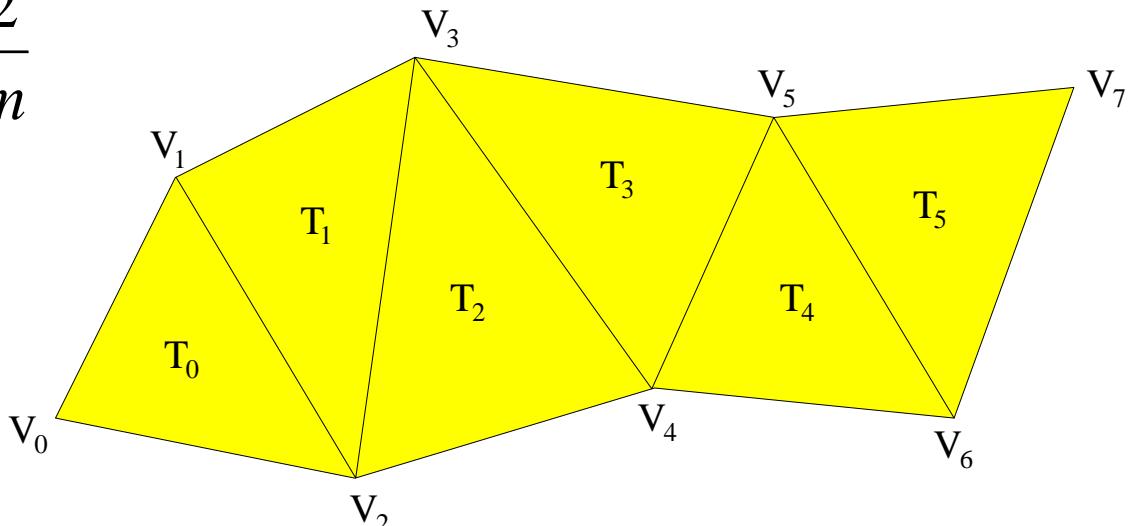
# Trake trokuta (triangle strip) (1/2)

- 1. trokut zadan s 3 vrha
- Idući nasljeđuje 2 vrha prethodnog
- $V_0, V_1, V_2, V_3\dots$
- Prosječan broj vrhova po trokutu ( $m - \# \text{ trokuta}$ ):

$$\bar{v} = \frac{3 + (m-1)}{m} = 1 + \frac{2}{m}$$

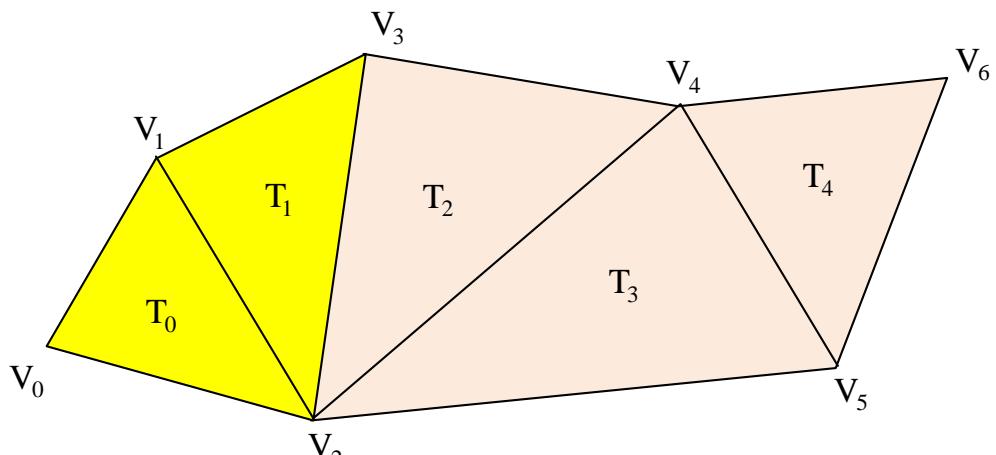
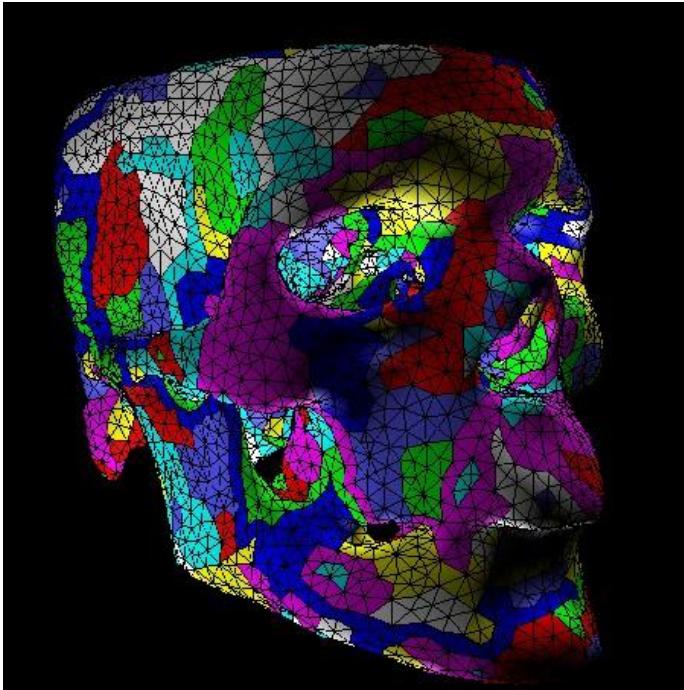
$$m \rightarrow \infty \Rightarrow \bar{v} \rightarrow 1$$

$$m = 10 \Rightarrow \bar{v} = 1.2$$



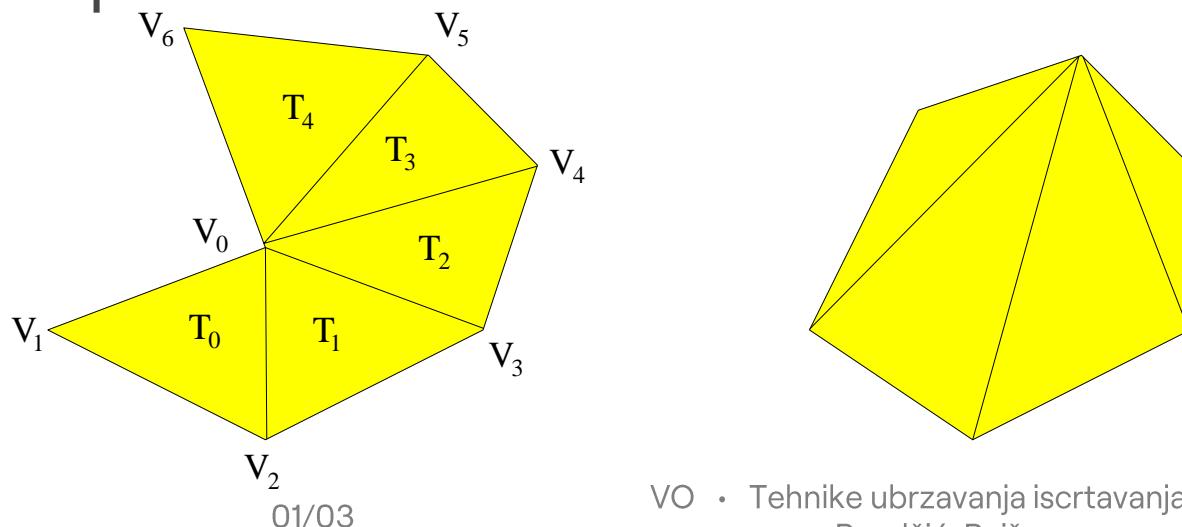
# Trake trokuta (2/2)

- Nije učinkovito u g.p.s. slati svaku traku vrhova zasebno (pozivi iscrtavanja su skupi)
- Možemo spojiti više traka u jednu, obrtanjem redoslijeda 2 zadnja vrha
- Primjer:
  - $V_0, V_1, V_2, V_3, V_2, V_4, V_5, V_6$



# Lepeze trokuta (triangle fan)

- 1 središnji vrh ( $V_0$ ) + m vrhova ( $V_1-V_m$ ) oko njega koji tvore lepezu
- Prosj. broj vrhova po trokutu kao i za traku trokuta, no rjeđe se pojavljuju
- Korisno za pretvorbu  $n$ -terokuta u trokute



# Mreže trokuta

- Najvažnija struktura za prikaz 3D geometrije
- GPU prilagođen radu s mrežama trokuta
- Skup vrhova + slijed indeksa
- Vrh – koordinate, normala, teksturne koord. itd.
- Slijed indeksa definira trokute
- Pri iscrtavanju predajemo vrhove i indekse u g.p.s.

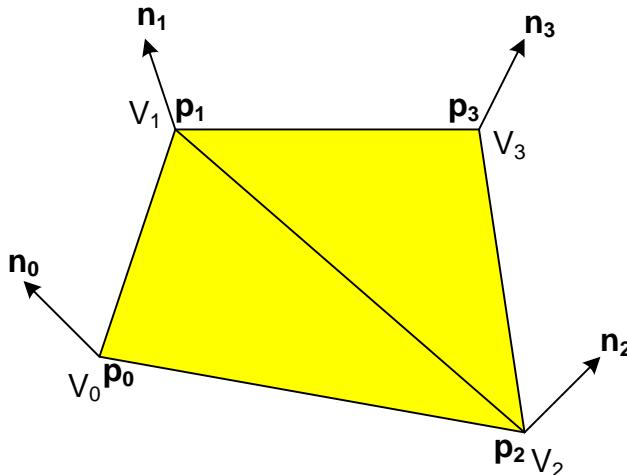
# Iscrtavanje mreže trokuta (1/3)

- 2 načina predaje vrhova i indeksa:
  - Spremnik vrhova (vertex buffer)
  - Struje vrhova (vertex streams)
- Spremnik vrhova (+ opc. spremnik indeksa):
  - Slijed podataka na susjednim mem. lokacijama
  - Podaci za 1 vrh pohranjeni zajedno (1 blok)
  - Format vrha definira podatke vrha
- Struje vrhova (+ opc. struja indeksa):
  - Polja podataka – svako polje sadrži podatke određene vrste
  - Npr. struja koordinata, struja normala, struja teksturnih koord. itd.

# Iscrtavanje mreže trokuta (2/3)

- Pri iscrtavanju specificiramo g.p.s. koji tip primitiva se iscrtava:
  - Lista točaka
  - Lista linija
  - Lista polilinija
  - Lista trokuta
  - Traka trokuta
  - Lepeza trokuta
- ◆ O tome ovisi kako će g.p.s. interpretirati sadržaj spremnika/struja vrhova
- ◆ Najčešće se koristi lista trokuta u spremniku vrhova i indeksa

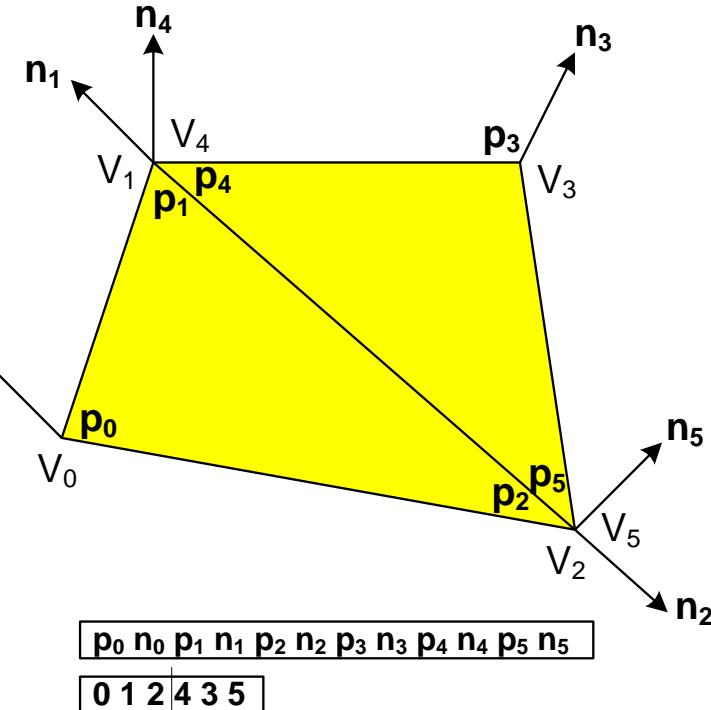
# Iscrtavanje mreže trokuta (3/3)



	Lista trokuta	Traka trokuta																										
Struje vrhova	<table border="1"><tr><td><math>p_0</math></td><td><math>p_1</math></td><td><math>p_2</math></td><td><math>p_1</math></td><td><math>p_3</math></td><td><math>p_2</math></td></tr><tr><td><math>n_0</math></td><td><math>n_1</math></td><td><math>n_2</math></td><td><math>n_1</math></td><td><math>n_3</math></td><td><math>n_2</math></td></tr></table>	$p_0$	$p_1$	$p_2$	$p_1$	$p_3$	$p_2$	$n_0$	$n_1$	$n_2$	$n_1$	$n_3$	$n_2$	<table border="1"><tr><td><math>p_0</math></td><td><math>p_1</math></td><td><math>p_2</math></td><td><math>p_3</math></td></tr><tr><td><math>n_0</math></td><td><math>n_1</math></td><td><math>n_2</math></td><td><math>n_3</math></td></tr></table>	$p_0$	$p_1$	$p_2$	$p_3$	$n_0$	$n_1$	$n_2$	$n_3$						
$p_0$	$p_1$	$p_2$	$p_1$	$p_3$	$p_2$																							
$n_0$	$n_1$	$n_2$	$n_1$	$n_3$	$n_2$																							
$p_0$	$p_1$	$p_2$	$p_3$																									
$n_0$	$n_1$	$n_2$	$n_3$																									
Spremnik vrhova	<table border="1"><tr><td><math>p_0</math></td><td><math>n_0</math></td><td><math>p_1</math></td><td><math>n_1</math></td><td><math>p_2</math></td><td><math>n_2</math></td><td><math>p_1</math></td><td><math>n_1</math></td><td><math>p_3</math></td><td><math>n_3</math></td><td><math>p_2</math></td><td><math>n_2</math></td></tr></table>	$p_0$	$n_0$	$p_1$	$n_1$	$p_2$	$n_2$	$p_1$	$n_1$	$p_3$	$n_3$	$p_2$	$n_2$	<table border="1"><tr><td><math>p_0</math></td><td><math>n_0</math></td><td><math>p_1</math></td><td><math>n_1</math></td><td><math>p_2</math></td><td><math>n_2</math></td><td><math>p_3</math></td><td><math>n_3</math></td></tr></table>	$p_0$	$n_0$	$p_1$	$n_1$	$p_2$	$n_2$	$p_3$	$n_3$						
$p_0$	$n_0$	$p_1$	$n_1$	$p_2$	$n_2$	$p_1$	$n_1$	$p_3$	$n_3$	$p_2$	$n_2$																	
$p_0$	$n_0$	$p_1$	$n_1$	$p_2$	$n_2$	$p_3$	$n_3$																					
Struje vrhova + struja indeksa	<table border="1"><tr><td><math>p_0</math></td><td><math>p_1</math></td><td><math>p_2</math></td><td><math>p_3</math></td></tr><tr><td><math>n_0</math></td><td><math>n_1</math></td><td><math>n_2</math></td><td><math>n_3</math></td></tr><tr><td>0</td><td>1</td><td>2</td><td>1</td><td>3</td><td>2</td></tr></table>	$p_0$	$p_1$	$p_2$	$p_3$	$n_0$	$n_1$	$n_2$	$n_3$	0	1	2	1	3	2	<table border="1"><tr><td><math>p_0</math></td><td><math>p_1</math></td><td><math>p_2</math></td><td><math>p_3</math></td></tr><tr><td><math>n_0</math></td><td><math>n_1</math></td><td><math>n_2</math></td><td><math>n_3</math></td></tr><tr><td>0</td><td>1</td><td>2</td><td>3</td></tr></table>	$p_0$	$p_1$	$p_2$	$p_3$	$n_0$	$n_1$	$n_2$	$n_3$	0	1	2	3
$p_0$	$p_1$	$p_2$	$p_3$																									
$n_0$	$n_1$	$n_2$	$n_3$																									
0	1	2	1	3	2																							
$p_0$	$p_1$	$p_2$	$p_3$																									
$n_0$	$n_1$	$n_2$	$n_3$																									
0	1	2	3																									
Spremnik vrhova + spremnik indeksa	<table border="1"><tr><td><math>p_0</math></td><td><math>n_0</math></td><td><math>p_1</math></td><td><math>n_1</math></td><td><math>p_2</math></td><td><math>n_2</math></td><td><math>p_3</math></td><td><math>n_3</math></td></tr><tr><td>0</td><td>1</td><td>2</td><td>1</td><td>3</td><td>2</td></tr></table>	$p_0$	$n_0$	$p_1$	$n_1$	$p_2$	$n_2$	$p_3$	$n_3$	0	1	2	1	3	2	<table border="1"><tr><td><math>p_0</math></td><td><math>n_0</math></td><td><math>p_1</math></td><td><math>n_1</math></td><td><math>p_2</math></td><td><math>n_2</math></td><td><math>p_3</math></td><td><math>n_3</math></td></tr><tr><td>0</td><td>1</td><td>2</td><td>3</td></tr></table>	$p_0$	$n_0$	$p_1$	$n_1$	$p_2$	$n_2$	$p_3$	$n_3$	0	1	2	3
$p_0$	$n_0$	$p_1$	$n_1$	$p_2$	$n_2$	$p_3$	$n_3$																					
0	1	2	1	3	2																							
$p_0$	$n_0$	$p_1$	$n_1$	$p_2$	$n_2$	$p_3$	$n_3$																					
0	1	2	3																									

# Učinkovitost indeksirane geometrije

- Spremnići vrhova i indeksa orijentirani na učinkovitost iscrtavanja, a ne pohrane
- Npr. ako želimo da isti dijeljeni vrh ima više normala (npr. oštri bridovi kocke) – potrebno višestruko definirati vrh (po 1 za svaku normalu)

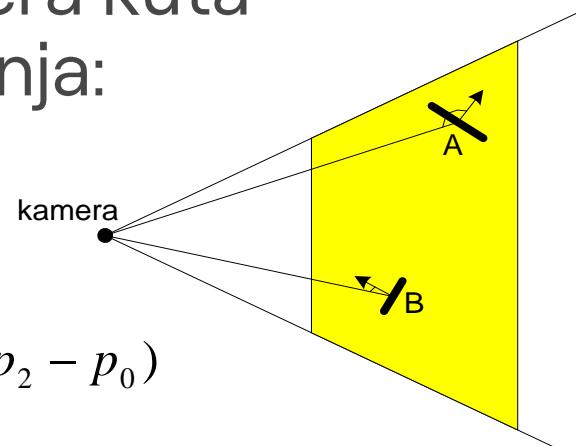


# Selektivno odbacivanje poligona (culling)

- Osnovna ideja: poligone koji nisu u nekom trenutku vidljivi na slici ne trebamo iscrtavati
- Najvažnije metode:
  - Odbacivanje stražnjih poligona (backface culling)
  - Odbacivanje po projekcionom volumenu (view-frustum culling)
  - Portalno odbacivanje (portal culling)
  - Odbacivanje prekrivenih poligona (occlusion culling)

# Odbacivanje stražnjih poligona

- Standardni dio geometrijske faze g.p.s.
- Poligoni okrenuti od kamere nisu vidljivi
- Okrenutost poligona određena redoslijedom njihovih vrhova (2 konvencije – u smjeru kazaljke na satu ili obrnuto)
- Moguća implementacija – provjera kuta normale poligona i smjera gledanja:



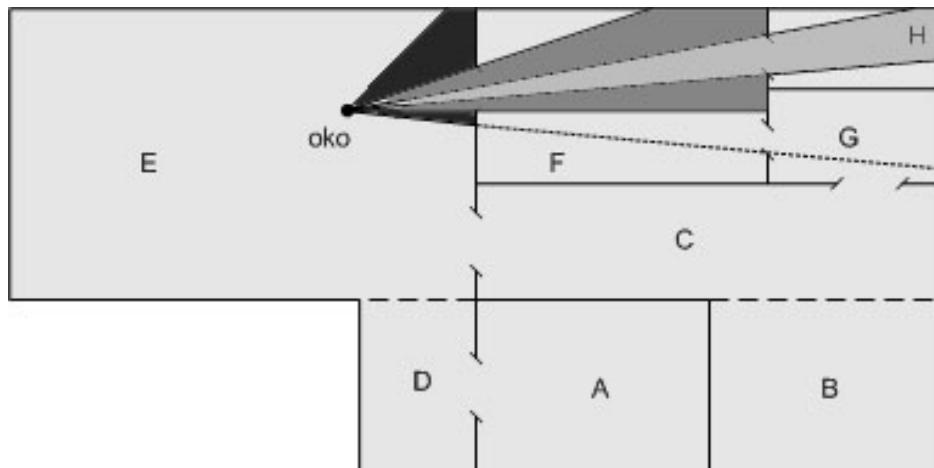
$$n = (p_1 - p_0) \times (p_2 - p_0)$$

# Odbacivanje po projekcionom volumenu

- Sve što je izvan projekcionog volumena, nevidljivo je
- Za provjeru vidljivosti koriste se hijerarhije obujmice, BSP i oktalna stabla...
- Npr. hijerarhija obujmica:
  1. Ako je obujmica potpuno izvan projekcionog volumena, odbacuje se sve u njoj
  2. Ako je potpuno unutra, sve se crta
  3. Ako projekcioni volumen siječe obujmicu, provjerava se sljedeća niža hijerarhijska razina obujmica
- Izvodi se u aplikacijskoj fazi

# Portalno odbacivanje (1/2)

- Koristi se za scene arhitekture sa sobama
- Scena se dijeli na ćelije (sobe)
- Ćelije imaju portale (vrata) prema drugim ćelijama
- Za svaku ćeliju gradimo graf susjednosti (podaci o portalima i susjednim ćelijama)

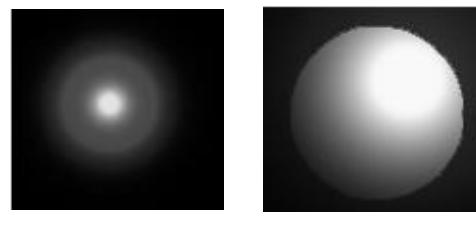
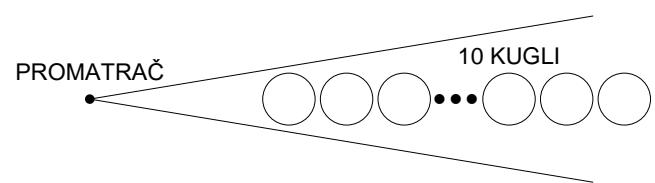


# Portalno odbacivanje (2/2)

- IsCRTavanje je rekURzivno:
  1. IsCRTaj trenutnu ćeliju (uz korištenje trenutnog projekcionog volumena)
  2. Na temelju vidljivih portala odredi nove, sužene projekcione volumene
  3. IsCRTaj vidljive susjedne ćelije
- Izvodi se u aplikacijskoj fazi

# Odbacivanje putem prekrivenosti

- Predmeti često prekriveni drugim predmetima
- Neće se vidjeti zbog Z-spremnika, no Z-spremnik se primjenjuje tek u fazi rasterizacije
- Dubinska složenost:



DUBINSKA  
SLOŽENOST

KONAČNA SЛИKA - VIDI  
SE SAMO PRVA  
KUGLA

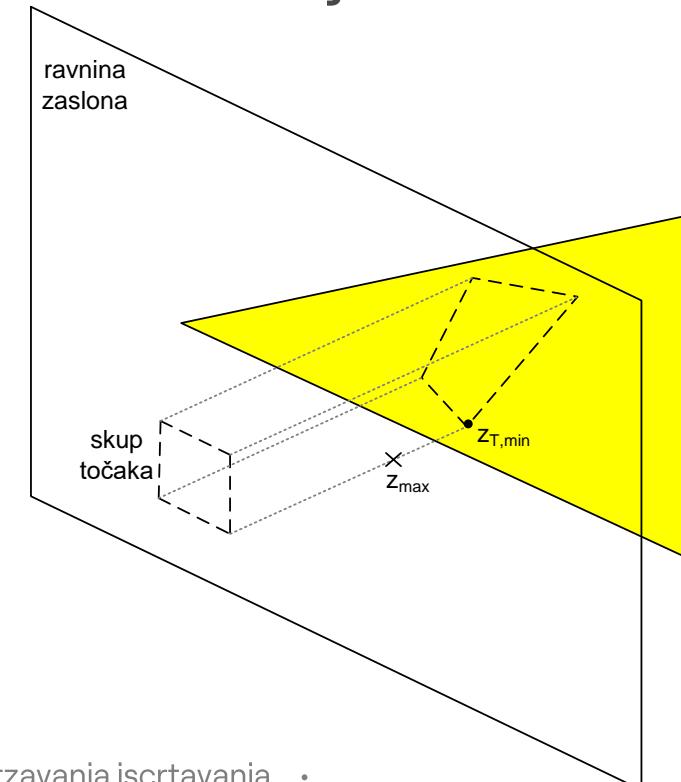
- Ideja: prekrivenu geometriju odbaciti što ranije
- Dosta složene tehnike, no neke su podržane sklopovalni
- Učinak ovisi o redoslijedu iscrtavanja – najbolje od naprijed prema natrag (front-to-back)

# Sklopovska provjera prekrivenosti (hardware occlusion queries)

- GPU podržava način iscrtavanja u kojem se provjerava vidljivost skupa poligona (najčešće obujmica) s obzirom na Z-spremnik
- Algoritam:
  1. Iscrtati obujmicu u načinu provjere prekrivenosti (uključen Z-test, isključen Z-write)
  2. Dohvatiti broj vidljivih piksela – ako je veći od nekog praga, iscrtati predmet (uz uključen Z-write)
- Najbolje u redoslijedu front-to-back, uz korištenje hijerarhije obujmica
- Izvodi se u aplikacijskoj fazi (uz djelomičnu sklopovsku podršku)

# Z-odbacivanje (Z-cull)

- Ugrađeno u fazi prolaza trokuta
- Rasterizacija se radi u skupovima do  $8 \times 8$  točaka
- Ispituje je li prekriven isječak trokuta (koji odgovara trenutnom skupu)
- Postupak  $Z_{MAX}$ :
  1. Odredi  $\sim z_{T,MIN}$
  2. Dohvati  $z_{MAX}$  iz Z-spremnika
  3. Ako  $z_{T,MIN} > z_{MAX}$ , isječak je prekriven



# Rani-Z (early-Z)

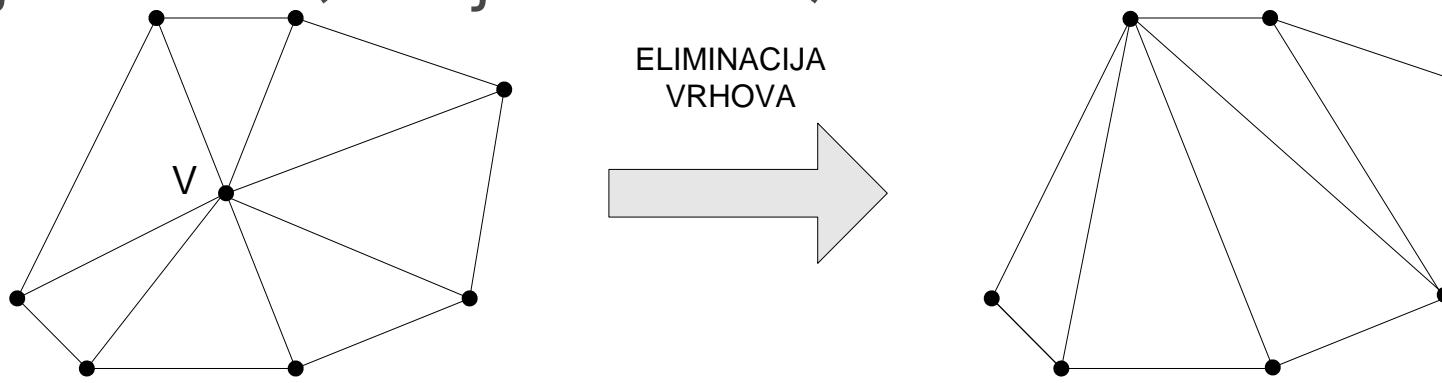
- Nakon generiranja fragmenata, a prije sjenčanja
- Uspoređuje dubinu fragmenta  $z_F$  s dubinom u Z-spremniku – ako je veća, odbacuje fragment
- Metode Z-cull i early-Z:
  - Ugrađene u graf. sklopolje i uključene „po defaultu“
  - Smanjuju opterećenje procesora točaka
  - Posebno učinkovite kod višeprolaznog iscrtavanja (jer se Z-spremnik postavlja u 1. prolazu)
  - Automatski se isključuju ako pixel shader mijenja dubinu fragmenta  $z_F$

# Tehnike razina detalja

- Engl. Level Of Detail – LOD
- Ideja: smanjiti razinu detalja (broj poligona) kada je predmet udaljen od kamere
  - Razlika se neće primijetiti
- Ako brzina padne, smanji razinu detalja
- Podtehnike:
  - Generiranje – stvaranje jednostavnijih inačica osnovnog modela predmeta
  - Odabir – odluka koja verzija će se iscrtavati
  - Zamjena – zamjena trenutne inačice drugom

# Pojednostavljivanje mreže trokuta

- Smanjiti broj trokuta u modelu, nastojeći pritom što manje promijeniti izgled modela
- Osim za LOD, koristi se i za pojednostavljivanje vrlo složenih modela, npr. dobivenih skeniranjem
- Eliminacija vrhova (starija metoda)



# Eliminacija bridova (edge collapse)

- Novija i jednostavnija metoda
- Općeniti postupak:
  1. Izračunaj funkciju troška za sve moguće eliminacije bridova; poredaj ih po trošku
  2. Izvedi operaciju s najmanjim troškom
  3. Ponovo izračunaj trošak gdje se mijenja; ponovi 2

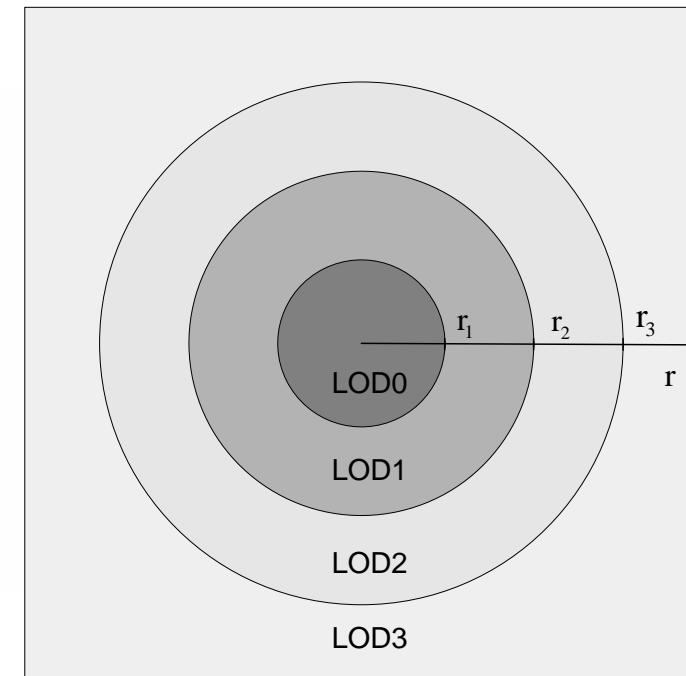


# Odabir razina detalja

- Kako odabrati LOD razinu koja će se u nekom trenutku iscrtavati?
- U upotrebi 2 metrike:
  - Udaljenost predmeta od kamere
  - Površina projekcije obujmice

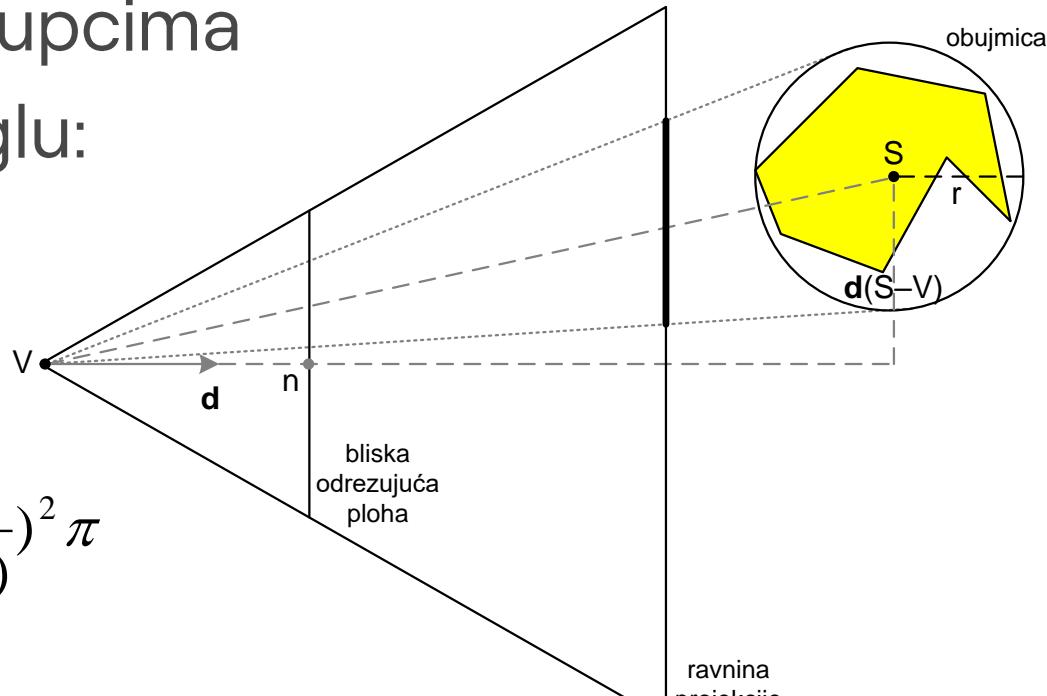
# Udaljenost predmeta od kamere

- Najjednostavnija i najčešće korištena metoda
- Svaka razina  $LOD_i$  ima raspon udaljenosti od kamere  $r_i$  do  $r_{i+1}$  na kojoj je aktivna



# Površina projekcije obujmice

- Kao i za udaljenost od kamere, definiraju se rasponi površina  $P_i$  do  $P_{i+1}$
- Površina projekcije se računa približnim, ali brzim postupcima
- Npr. za kuglu:



$$P = \left( \frac{nr}{d(S-V)} \right)^2 \pi$$

# Zamjena razina detalja

- Mora biti što neprimjetnije
- Nagla zamjena modela rezultira efektom skokova (popping)
- Glavne metode:
  - Diskretne razine detalja
  - Miješanje razine detalja
  - $\alpha$  razine detalja
  - Geomorfne razine detalja

# Diskrete razine detalja (1/2)

- Koristi se više verzija istog modela, sa sve manjim i manjim brojem poligona
- Kad se ispune uvjeti (npr. udaljenost od kamere), jedna inačica se zamjenjuje drugom



# Diskrete razine detalja (2/2)

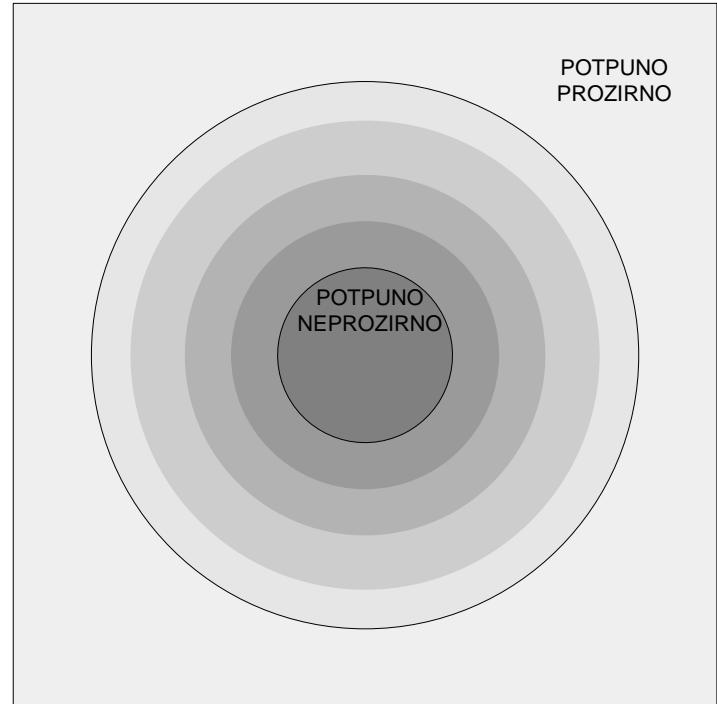
- ◆ Jednostavna tehnika
- ◆ Vrlo dobra sklopovska podrška – različite LOD inačice modela mogu se pohraniti zajedno u video mem.
- ◆ Izražen efekt skokova, pogotovo ako r „oscilira“ oko granice raspona
- ◆ Potonje moguće ublažiti histerezom – granice promjene u jednom smjeru drukčija nego u drugom

# Miješanje razina detalja

- Ideja: kratko vrijeme iscrtavamo obje inačice istodobno, uz uključen alpha blending
- Postupak:
  1. Iscrtaj LOD1 bez prozirnost u spremnik boje, uz uključen Z-spremnik
  2. Iscrtaj LOD2 u spremnik boje uz uključenu „over“ funkciju prozirnosti
    - Pritom linearno povećavati  $\alpha$  od 0 do 1
  3. Kad je LOD2 iscrtan uz  $\alpha = 1$ , počni iscrtavati LOD1, uz uključen Z-test, ali isključen Z-write
    - Pritom linearni smanjivati  $\alpha$  od 1 do 0
- Nedostatak – kratkotrajan gubitak performansi zbog istodobnog iscrtavanja 2 inačica predmeta

# $\alpha$ -razine detalja

- Predmet postaje prozirniji do nestajanja
- Nije potrebno raditi dodatne LOD inačice
- Jednostavna metoda
- Eliminira efekt skokova
- Ubrzanje se postiže tek po nestajanju predmeta



# Geomorfne razine detalja

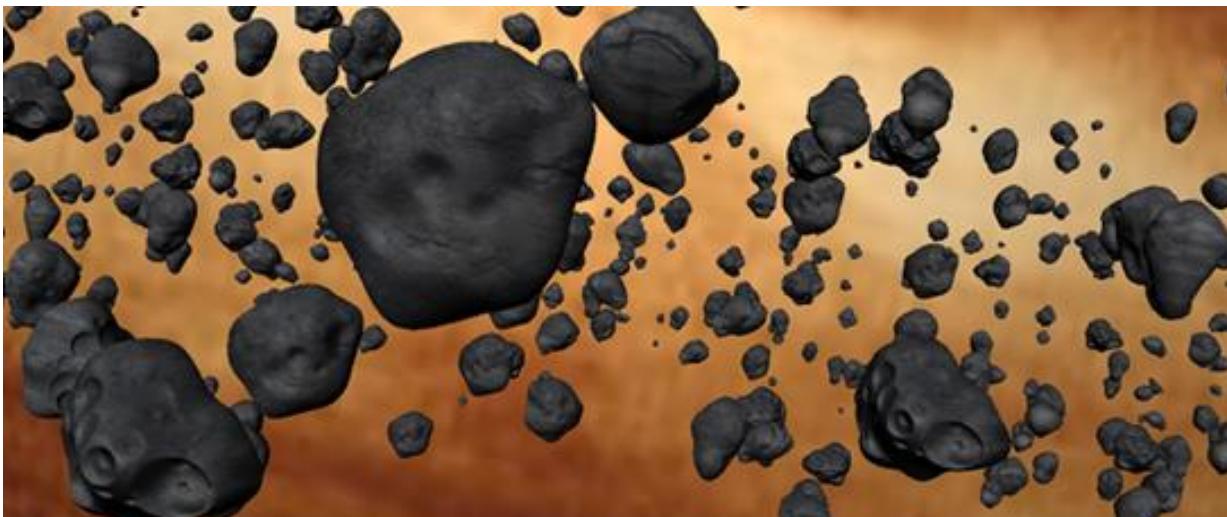
- Koristi se niz modela proizведен eliminacijom bridova
- Podjela vrhova – inverz eliminacije bridova
- Možemo zamisliti da je složenija LOD inačica nastala nizom podjela vrhova u jednostavnijoj inačici
- Ideja: za svaki vrh upamtimo njegov „vrh-roditelj”, te interpoliramo vrhove između susjednih LOD inačica
- Gladak prijelaz između dviju inačica modela
- Predmet neprestano mijenja oblik, što može zasmetati

# Srodne tehnike

- Ideja LOD – zamijeniti resurs (model) jednostavnijom inačicom / aproksimacijom ovisno o relativnom položaju kamere
- To se može poopćiti na druge tehnike iscrtavanja
- Npr. zamjena materijala/tekstura niže kvalitetnima:
  - Isključivanje tekture detalja
  - Isključivanje tekture okoline
  - Zamjena shadera jednostavnijom inačicom
  - ...

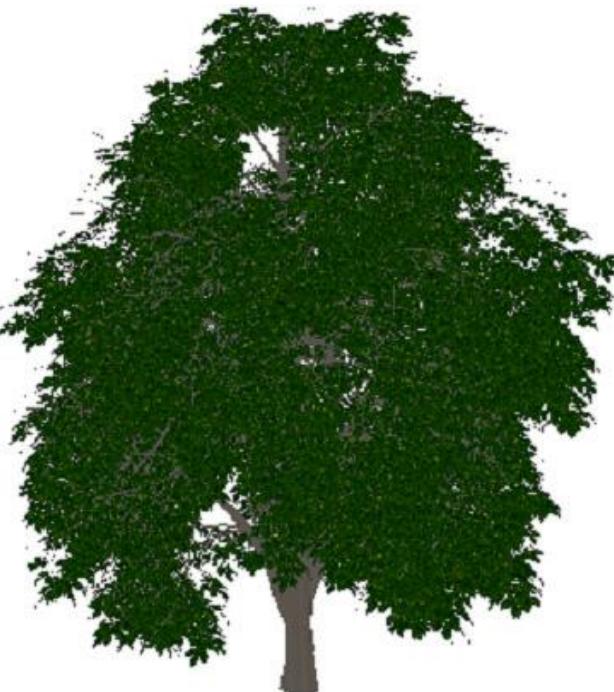
# Varalice (impostors)

- Složeni predmet iscrtamo u teksturu
- Predmet zamijenimo panoom (billboard), i na njega nalijepimo teksturu
- Kod većih promjena relativnog položaja kamere potrebno ponoviti postupak

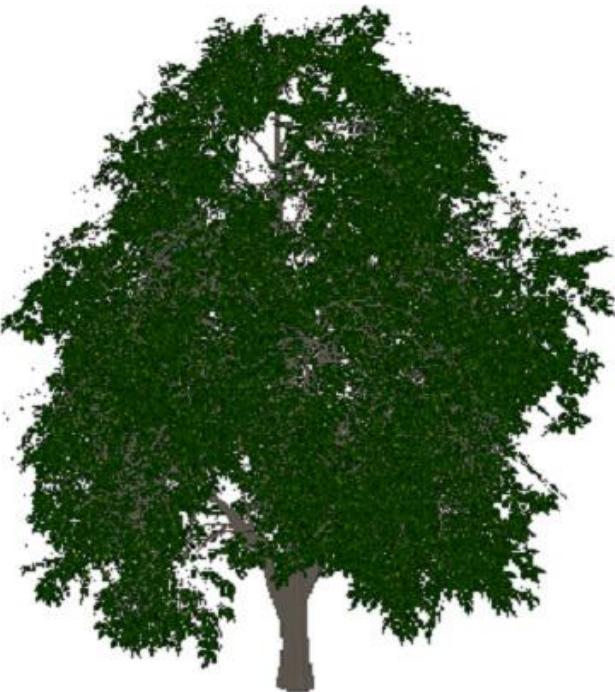


# Oblaci panoa (billboard clouds)

- Složen predmet zamjenjujemo skupom panoa
- Svaki pano sadrži sliku nekog dijela predmeta
- Nije potrebno višekratno iscrtavati predmet



01/03



VO • Tehnike ubrzavanja iscrtavanja •  
Pandžić, Pejša

# Optimizacija protočnog sustava

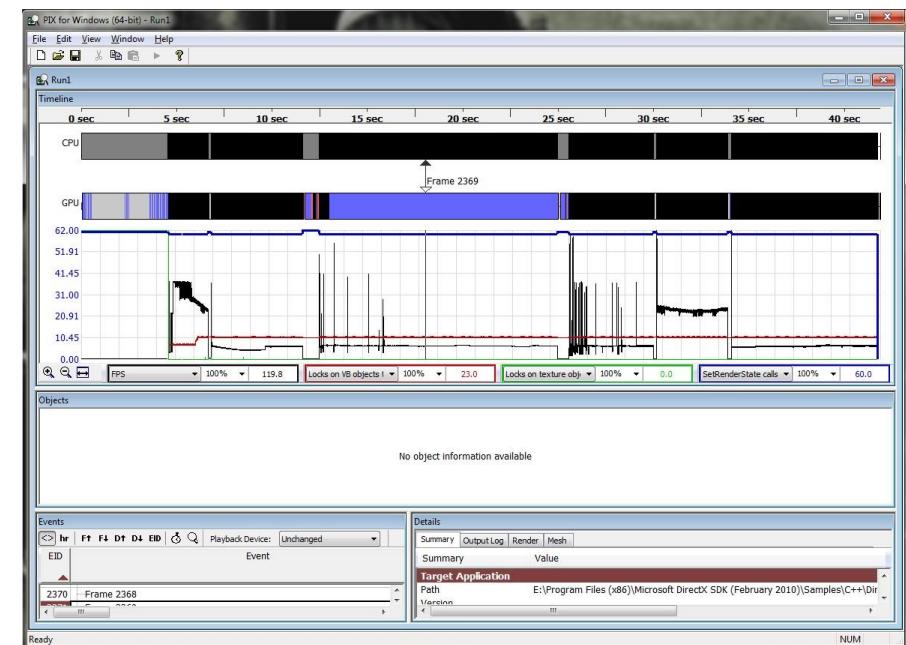
- Najsporija faza stvara usko grlo (kao kod pokretnih traka)
- Postupak
  1. Pronaći usko grlo
  2. Ubrzati tu fazu
  3. Ponoviti postupak
- Ako ne možemo ukloniti usko grlo, u ostalim fazama se može obaviti više posla i povećati kvaliteta

# Mjerenje performansi (1/2)

- Brzina iscrtavanja – broj slika u sekundi (frames per second, FPS)
- Brzina geom. faze – broj vrhova u sekundi
- Brzina rast. faze – broj točaka u sekundi
- Te mjere nisu same dovoljne:
  - Usko grlo se često seli već unutar jedne slike
- Složeni testovi (benchmark):
  - Prethodno isključiti dvostruko spremanje

# Mjerenje performansi (2/2)

- Alati za profiliranje korištenja CPU/GPU:
  - NVIDIA PerfKit
  - AMD GPU PerfStudio
  - PIX for Windows
  - gDEBugger
- ◆ Omogućuju vremensko praćenje raznih statistika:
  - Broj poziva iscrtavanja, čitanja tekstura, izvođenja shadera
  - Potrošnja memorije
  - Iskorištenost CPU-a
  - ...



# Traženje uskog grla (1/3)

- Testirati svaku fazu posebno
- Aplikacijska faza:
  - 100% na CPU
  - Koristiti programe za prikaz tereta procesora:
    - Ako je procesor skoro 100% opterećen, tu smo!
    - ... osim ako glavna petlja nije radno čekanje
  - Koristiti alate za profiliranje kôda:
    - AMD CodeAnalyst, VS Team System Profiler...
  - Eliminirati ostale faze slanjem „praznih“ naredbi
    - Npr. za iscrtavanje koristiti null driver
    - Ako nema ubrzanja, našli smo krivca!

# Traženje uskog grla (2/3)

- Geometrijska faza:
  - Najčešće 100% na GPU
  - Glavne operacije – dohvati i sjenčanje vrhova
  - Testiranje dohvata vrhova:
    - Povećati format vrha (npr. dodati „prazne“ teksturne koord.)
    - Ako brzina padne, to je to!
  - Testiranje sjenčanja vrhova:
    - Dodati naredbe u vertex shader
    - Ako brzina padne, to je to!

# Traženje uskog grla (3/3)

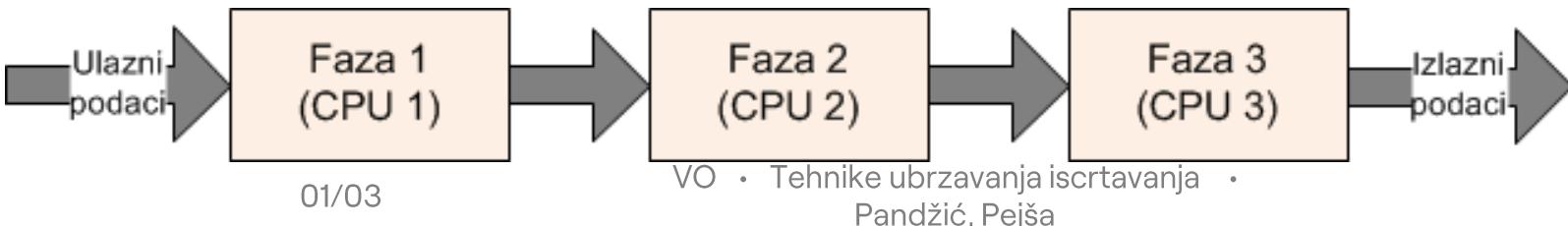
- Faza rasterizacije:
  - 100% na GPU
  - Glavne operacije – sjenčanje točaka i ROP
  - Testiranje sjenčanja točaka:
    - Dodati naredbe u pixel shader ili smanjiti razlučivost
    - Promjena brzine => tu je usko grlo
  - Testiranje ROP:
    - Smanjiti dubinu boje u spremniku boja (npr. 32-bit => 16-bit)
    - Ako brzina poraste, tu je problem!

# Optimizacija aplikacijske faze

- Opća pravila optimizacije koda i pristupa memoriji
  - Npr. izbjegavati dijeljenje, spremati podatke u memoriju redoslijedom korištenja...
  - Koristiti optimizacijske opcije compilera
  - ...
- 2 važne strategije:
  - Koristiti paralelizam
  - Optimizirati promjene stanja

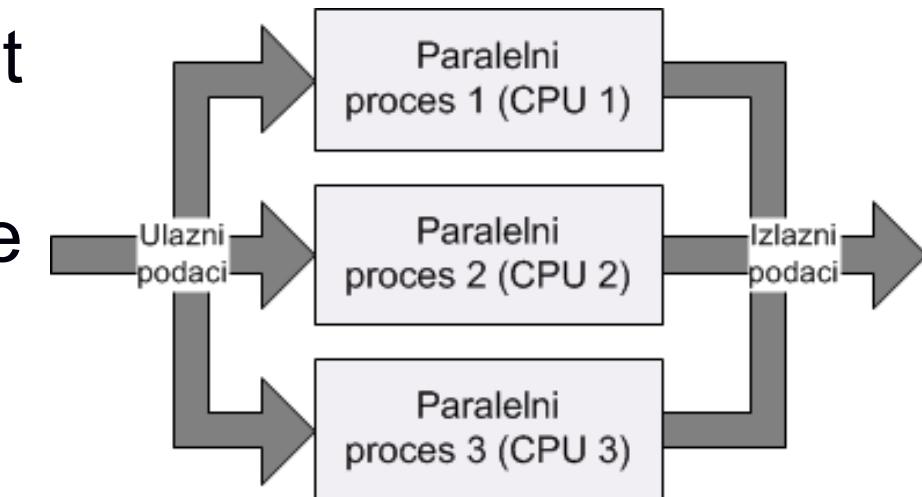
# Paralelizam u aplikacijskoj fazi (1/2)

- Grafički algoritmi se često daju paralelizirati
- Višeprocesorska protočnost (vremenski paralelizam):
  - Aplikacijsku fazu podijeliti u protočne podfaze
  - Svaka podfaza se izvodi na zasebnoj CPU jezgri
  - Npr. u sustavu OpenGL Performer:
    - APP – logika aplikacije
    - CULL – odbacivanje poligona
    - DRAW – iscrtavanje
  - Nedostatak – akumulacija kašnjenja zbog protočnosti



# Paralelizam u aplikacijskoj fazi (2/2)

- Paralelno izvođenje (prostorni paralelizam):
  - Pojedine algoritme zamijeniti višedretvenim inačicama
  - Nema akumulacije kašnjenja
  - Neke algoritme nije moguće učinkovito paralelizirati
- ◆ Napomena: mogućnost paralelnog pristupa grafičkom API-ju iz više dretvi uvedena tek u DirectX 11!



# Optimizacija promjene stanja (1/3)

- Operacije promjene stanja:
  - Zadavanje spremnika vrhova i spremnika indeksa
  - Zadavanje tekstura
  - Učitavanje i konfiguracija shadera
  - Poziv iscrtavanja
- Te operacije su skupe:
  - Izvode se na CPU
  - Zahtijevaju pražnjenje g.p.s. (npr. priručnih memorija) → prazan hod!
- Učestale promjene stanja su danas glavni uzrok loših performansi!

# Optimizacija promjene stanja (2/3)

- Rješenje – bolje grupirati predmete u sceni!
- Spajanje predmeta (batching):
  - Više manjih predmeta spojiti u jedan
  - Što ako imaju različite materijale i teksture?
  - Dodati ID predmeta u format vrha, te u shader grananje na temelju ID
  - Time se opterećenje seli u druge faze g.p.s.

# Optimizacija promjene stanja (3/3)

- Instanciranje:
  - Ako se isti predmet pojavljuje mnogo puta (npr. drveće)
  - Predmet možemo iscrtati N puta jednim pozivom iscrtavanja
  - Podaci specifični za instancu (npr. transformacije) u posebnom spremniku
- Organizacija scene s obzirom na stanje:
  - Neki predmeti dijele materijale, shadere i/ili teksture
  - Grupirati ih i iscrtavati slijedno!
  - Manje učinkovito (svaki predmet i dalje ima svoj spremnik vrhova)
  - Najpraktičnije realizirati u sklopu grafa scene

# Optimizacija geometrijske faze

- Transformacije, osvjetljenje, obrezivanje, projekcija i preslikavanje na ekran
- U načelu ne optimiziramo izravno
- Smanjiti količinu geometrije u g.p.s.  
(odbacivanje, LOD)
- Koristiti mreže trokuta s indeksima, trake  
trocuta i sl.

# Optimizacija faze rasterizacije

- Koristiti odbacivanje stražnjih poligona
- Isključiti Z-spremnik ako nije potreban:
  - Npr. crtanje pozadine
- Koristiti tehnike smanjenja dubinske složenosti:
  - Z-odbacivanje i rani-Z ne isključivati bez razloga
  - Uvesti preliminarni prolaz radi inicijalizacije Z-spremnika
- Miješanje boja koristiti samo kad je potrebno
- Koristiti kompresiju tekstura
- Smanjiti broj svjetala ili pojednostaviti sjenčanje
- Imati više varijanti pixel shadera
- Koristiti jednostavniji anti-aliasing
- Smanjiti razlučivost iscrtavanja

# Virtualna okruženja

*Umrežena virtualna okruženja*

*Prof. dr. sc. Igor S. Pandžić*

*Prof. dr. sc. Maja Matijašević*

*dr. sc. Mirko Sužnjević*

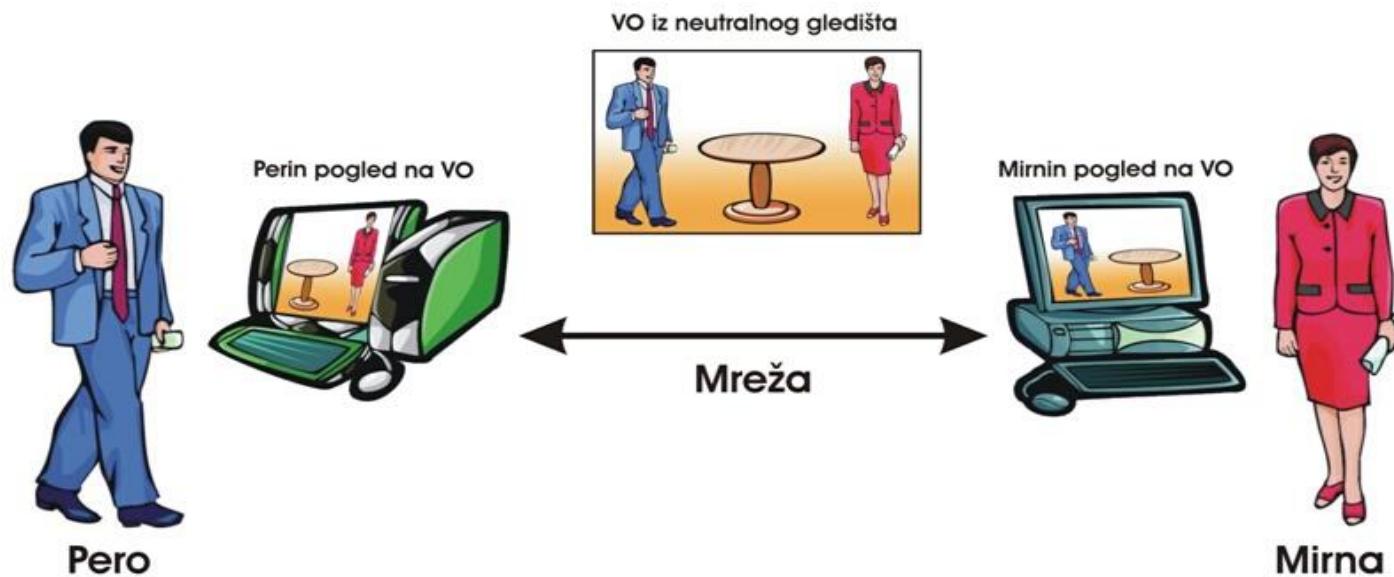
# Sadržaj predavanja

Zavod za telekomunikacije

- ◆ Uvod
  - Definicija
  - Tehnički izazovi
  - Osnovni model UVO
- ◆ Upravljanje zajedničkim dinamičkim stanjem
- ◆ Oblikovanje programskog rješenja UVO
- ◆ Virtualni svjetovi
  - Mrežni zahtjevi
  - Korisnička populacija
  - Poslovni modeli

# Umrežena virtualna okruženja

Zavod za telekomunikacije



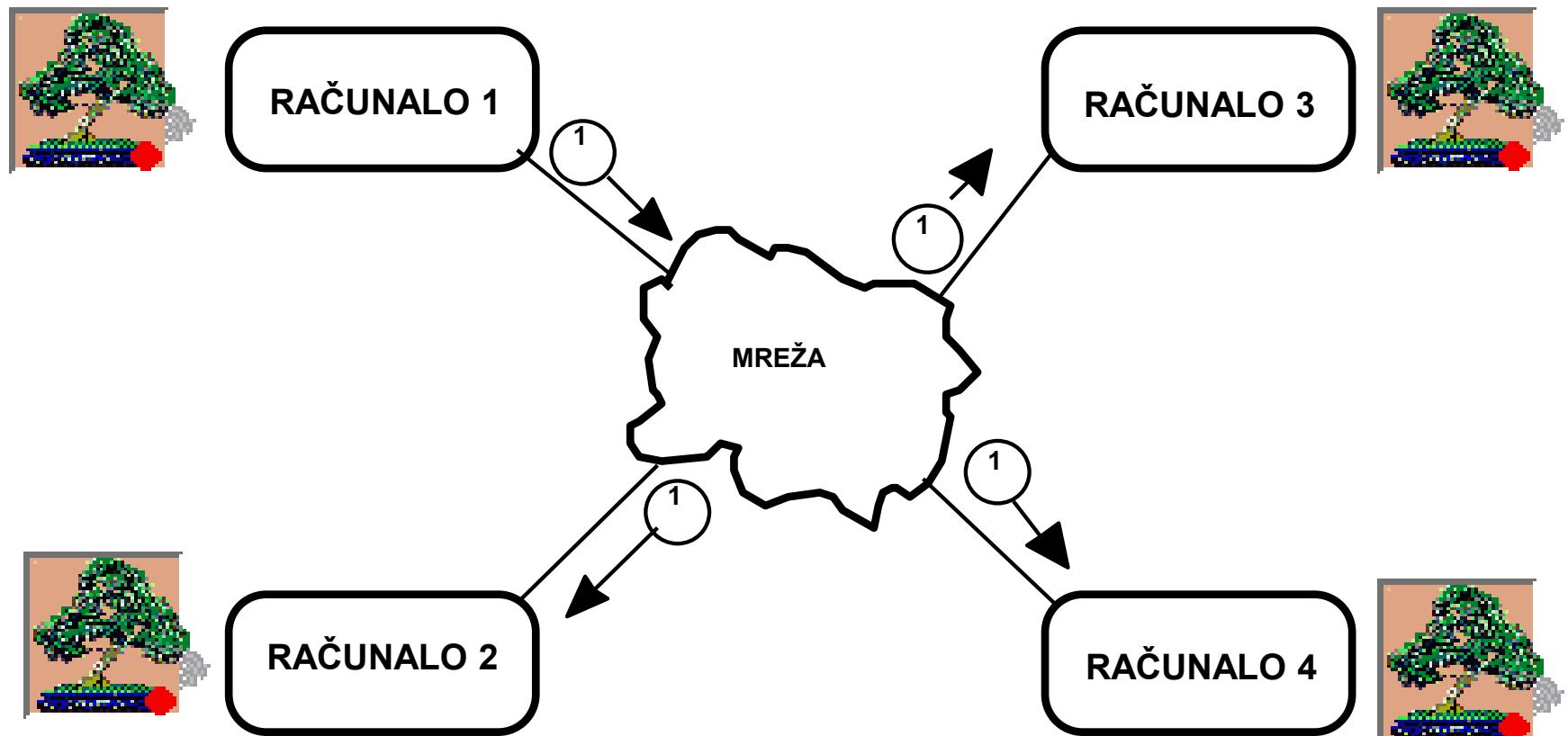
- ◆ Fizički udaljeni korisnici sudjeluju u zajedničkom virtualnom okruženju
- ◆ Svako računalo ima lokalnu kopiju okruženja
- ◆ Svaki korisnik upravlja svojim 3D prikazom i okruženjem
- ◆ Sve kopije okruženja se međusobno sinkroniziraju putem mreže
- ◆ Korisnici vide jedni druge jer su grafički prikazani u okruženju

# Kako korisnici doživljavaju UVO-a?

Zavod za telekomunikacije

- ◆ Doživljaj zajedničkog prostora
  - Korisnici osjećaju da su u istom (stvarnom ili zamišljenom) prostoru
- ◆ Doživljaj zajedničkog prisustva
  - Svaki od korisnika ima svoju reprezentaciju unutar virtualnog svijeta
- ◆ Doživljaj zajedničkog vremena
  - Događaji (izgledaju kao da) se događaju u isto vrijeme
- ◆ Mogućnost komunikacije
- ◆ Mogućnost interakcije
  - S virtualnim svjetom i drugim korisnicima

# Kako radi umreženo virtualno okruženje



# Primjena

Zavod za telekomunikacije

- ◆ Virtualni svjetovi – popularni u 2000-tima, danas ih ponovno popularizira virtualna stvarnost
  - Druženje i društvene mreže (Facebook Horizon - <https://www.oculus.com/facebookhorizon/>)
  - Virtualna telekonferencija/zajednički rad (Terf - <https://www.3dicc.com/>)
  - Učenje/obuka na daljinu (Second Life - <https://secondlife.com/>)
  - Virtualni svjetovi za djecu (Club Penguin - ugašen)
  - ...
- ◆ Umrežene višekorisničke igre – primarna svrha je zabava



UMREŽENA VIRTUALNA OKRUŽENJA

- ◆ Svrha
- ◆ Perzistentnost
  - Virtualni svjetovi su uglavnom perzistentni – postoje i mijenjaju se i dok korisnik nije u njima
  - Kod igara je puno češće iznova stvaranje virtualnog svijeta iz pohranjenih podataka (primjerice svaka bitka u Call of Duty se odvija na istoj mapi koja se iznova kreira)
- ◆ Mijenjanje 3D svijeta
  - Virtualni svjetovi dopuštaju dodavanje novih slika, tekstura, zvukova, videa i 3D objekata (primjerice Second Life)
  - Igre ne dopuštaju dodavanje novih informacija u svoj klijent (iako mogu dopuštati mijenjanje svijeta primjerice Fortnite)
- ◆ Karakteristike mrežnog prometa – jako ovisne o „promjenjivosti“ 3D svijeta jer ako korisnici mogu dodati nove informacije, drugi korisnici moraju te iste informacije preuzeti

- ◆ Prikaz korisnika
- ◆ Interakcija korisnika u virtualnom okruženju
- ◆ Podrška za prirodnu komunikaciju
- ◆ Karakteristike tehnologije i načina umrežavanja
- ◆ Prilagodljivost veličini (engl. *scalability*)

# Prikaz korisnika

- ◆ Lik korisnika unutar virtualnog okruženja se obično naziva avatar
- ◆ Od riječi avatāra iz sanskrita koja znači inkarnaciju ili manifestaciju nekog od bogova u tijelu čovjeka ili životinje
- ◆ Prvi put avatar se koristi u ovom značenju 1985 u igri *Ultima IV: Quest of the Avatar*
- ◆ Oblici avatara variraju od jednostavnih geometrijskih do složenih humanoidnih avatara visokih detalja
- ◆ Karakteristike avatara često definiraju i mogućnosti interakcije kao i posredno na količinu podataka koje se prenose mrežom
- ◆ Današnji avatari najčešće imaju niz gesti i izraza lica kroz koje korisnici mogu komunicirati s drugima (primjerice ples)

Zavod za telekomunikacije



# Interakcija korisnika u virtualnom okruženju

- ◆ Načini interakcije zadani karakterom aplikacije
- ◆ Učestali načini interakcije
  - Kretanje
  - Geste
  - Stvaranje zvukova
  - Interakcije s predmetima
  - Interakcije s drugim korisnicima
- ◆ Mehanizmi za upravljanje pravom pristupa – ako više korisnika pokušava udariti virtualnu loptu kako odrediti tko je uspio?
- ◆ Vremenski okvir interakcije?



# Podrška za prirodnu komunikaciju

Zavod za telekomunikacije

- ◆ Većina UVO-a podržava pismenu komunikaciju
- ◆ Komunikacija putem gesti je također vrlo česta
  - Mahanje
  - Ples
  - Potvrda i negacija
  - ....
- ◆ U posljednje vrijeme UVO-a također donose ugrađenu podršku za glasovnu komunikaciju, a ista može biti i ograničena pozicijom avatara unutar UVO (ukoliko su avatari dva korisnika predaleko neće se čuti)

# Karakteristike tehnologije i načina umrežavanja



Zavod za telekomunikacije

- ◆ Korisnici istog UVO-a mogu koristi pristupne uređaje i mreže heterogenih karakteristika
- ◆ Pristupni uređaji imaju definirane minimalne uvjete (minimalnu konfiguraciju) koja podržava pristup određenom UVO
- ◆ Karakteristike mrežnih parametara variraju ovisno o tipu UVO te o njegovim karakteristikama
- ◆ „Sakrivanje“ mreže i njenih utjecaja je najveći tehnički izazov UVO-a
- ◆ Najvažniji mrežni parametri koji utječu na UVO:
  - Propusnost
  - Kašnjenje
  - Kolebanje kašnjenja
  - Gubitak paketa

# Prilagodljivost veličini - skalabilnost

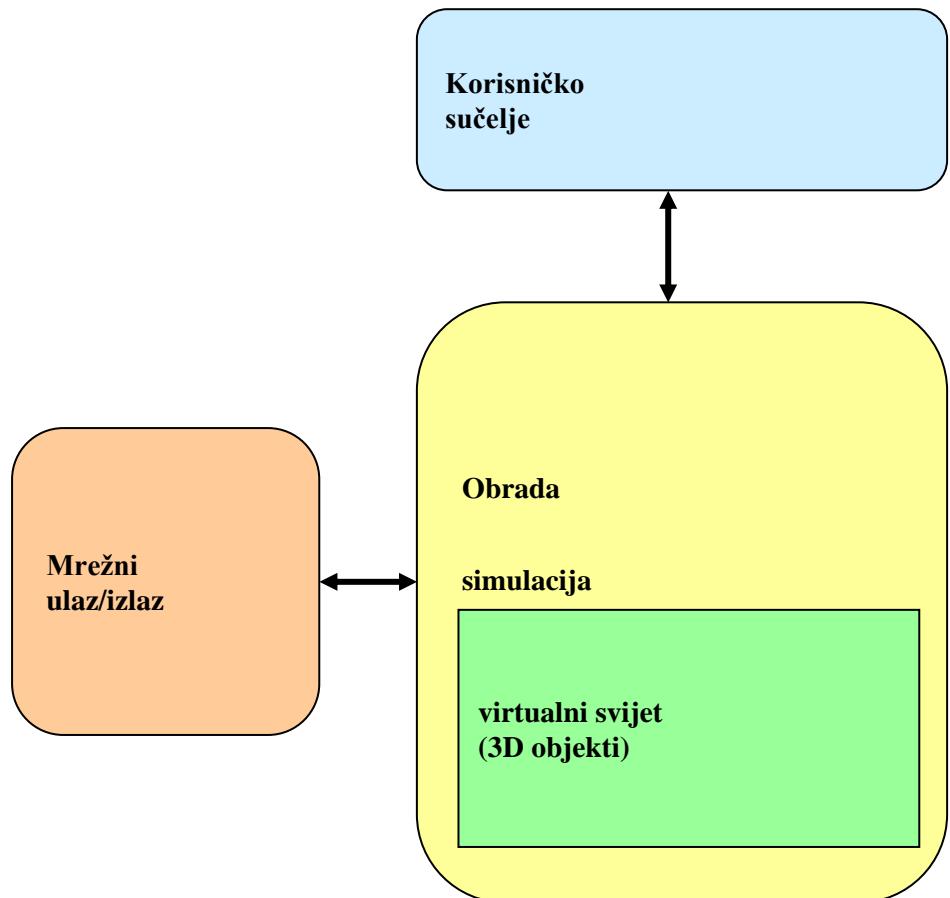
Zavod za telekomunikacije

- ◆ Skalabilnost – koliko neki sustav može rasti, a da se pri tom ne naruši njegova funkcija
- ◆ UVO-a mogu biti vrlo popularna – mehanizmi za osiguranje skalabilnosti su vrlo važni
  - Second Life – na vrhuncu popularnosti imao 1 milijun korisnika
  - World of Warcraft – na vrhuncu popularnosti 12 milijuna korisnika
- ◆ Parametri koji utječu na skalabilnost:
  - Složenost grafičkog prikaza svijeta i avatara
  - Količina podataka koju generira svijet
  - Obrada ulaza i izlaza
  - Odražavanje dinamičkog zajedničkog stanja

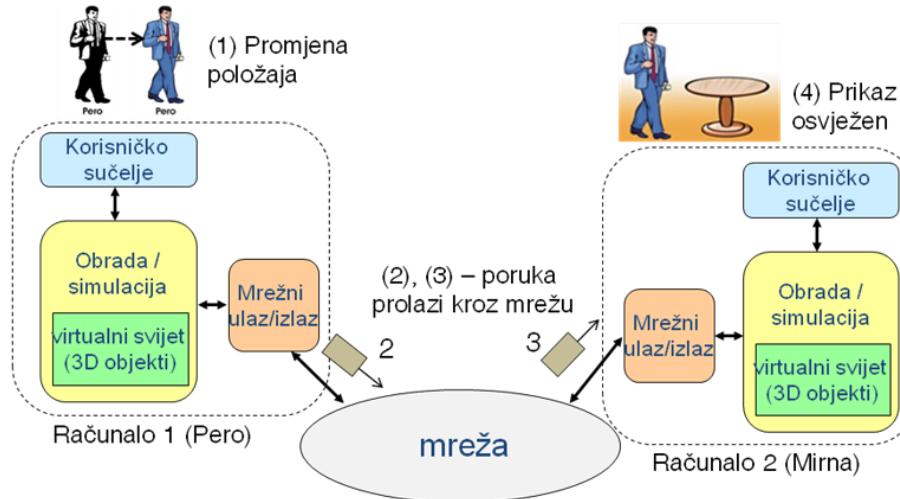
# Osnovni model UVO

Zavod za telekomunikacije

- ◆ Korisničko sučelje, ulazni i izlazni uređaji
- ◆ Skup 3D geometrijskih objekata (oblik, boja i sl.), i drugih komponenata (npr. svjetlo, zvuk) u računalu
- ◆ Obrada – sadrži simulaciju vezana uz primjenu (početne postavke, simulacijska petlja), obrada 3D grafike i zvuka
- ◆ Mrežna povezanost



# Usklađivanje instanci UVO putem mreže

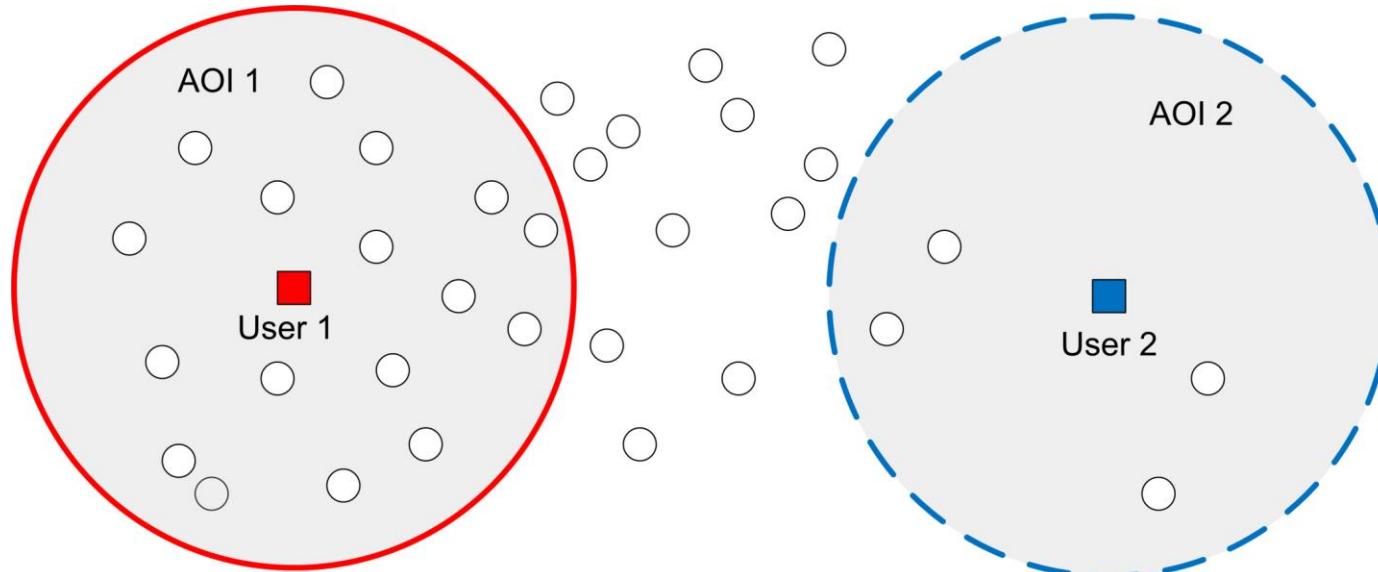


- ◆ Svaki korisnik ima pokrenutu instancu VO
- ◆ Ne moramo slati elemente koji se ne mijenjaju (npr. teksture, gotovi 3D objekti)
- ◆ Problemi
  - Promet raste s brojem korisnika – skalabilnost
  - Kašnjenje nije isto za sve korisnike - nekonzistentnost

# Filtriranje prema području interesa

Zavod za telekomunikacije

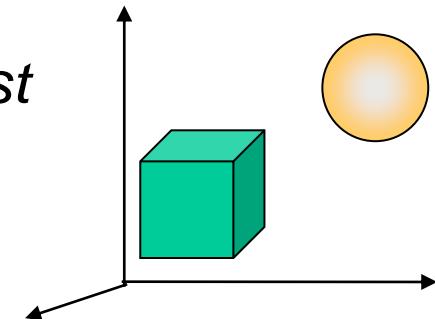
- ◆ engl. Area Of Interest Management, AOIM
- ◆ Prosljeđuju se samo relevantne poruke
- ◆ Umjesto eksponencijalnog rasta prometa s brojem korisnika, linearan rast: ključan preduvjet za postojanje modernih UVO s desecima tisuća istovremenih korisnika



# Upravljanje dinamičkim zajedničkim stanjem

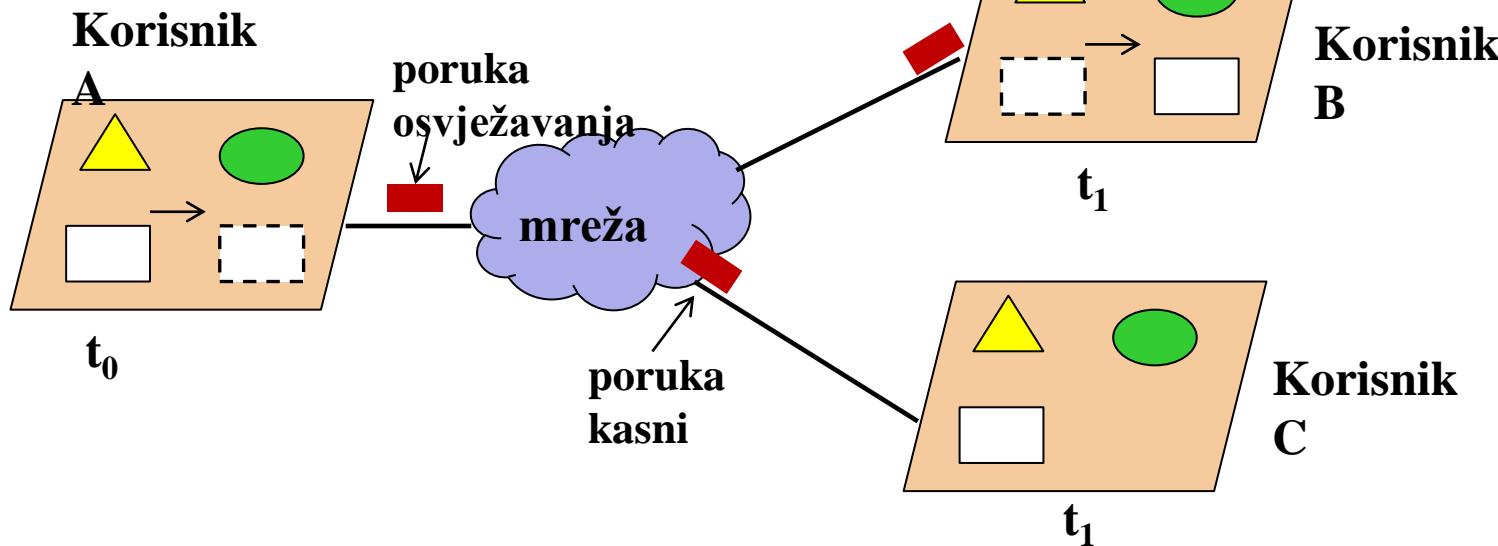
# Upravljanje dinamičkim zajedničkim stanjem

- ◆ Zahtjev: održati konzistentno stanje (isto stanje na svim replikama virtualnog svijeta)
- ◆ Zajedničko stanje = skup varijabli stanja svih pojedinačnih objekata u UVO
  - varijable za jedan objekt: položaj, orijentacija, brzina, izgled, itd.
- ◆ Osnovni problem:  
*odnos konzistentnost – propusnost*



# Problem nekonzistentnosti

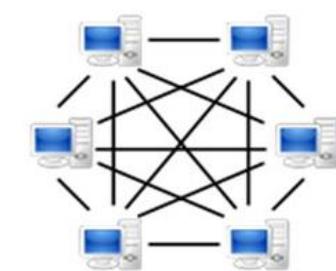
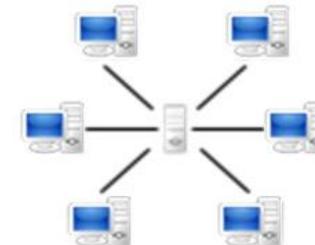
Zavod za telekomunikacije



- Ključni izazov: održavanje raspodijeljenog zajedničkog stanja konzistentnim, ili približno konzistentnim, uz promjene koje nastaju (asinkrono, dinamički i raspodijeljeno)
- Zahtjeve na konzistentnost treba uskladiti s namjenom aplikacije
- Češće slanje poruka osvježavanja skraćuje potencijalni vremenski interval u kojem može doći do odstupanja, ali više opterećuje mrežu i proces za slanje i primanje poruka

# Rješenja problema upravljanja zajedničkim stanjem

- ◆ Na razini arhitekture
  - Centralizirani repozitorij informacija
  - Raspodijeljeni (distribuirani) repozitorij informacija
- ◆ Na razini protokola
  - Jaka konzistentnost - svakom trenutku stanje svake replike UVO-a je istovjetno
    - Koristi se protokol koji osigurava konzistentnost u distribuiranom okruženju
    - Protokol konzistentnosti je na aplikacijskoj razini
  - Slaba konzistentnost – stanje svake replike UVO-a može biti različito u istom trenutku
    - Razlika se pokušava minimizirati kako bi njen utjecaj bio neprimjetan
    - Koristi se učestala regeneracija (slanje) stanja
    - Koriste se tehnike predviđanja i *dead reckoning*



# Arhitektura raspodijeljene aplikacije

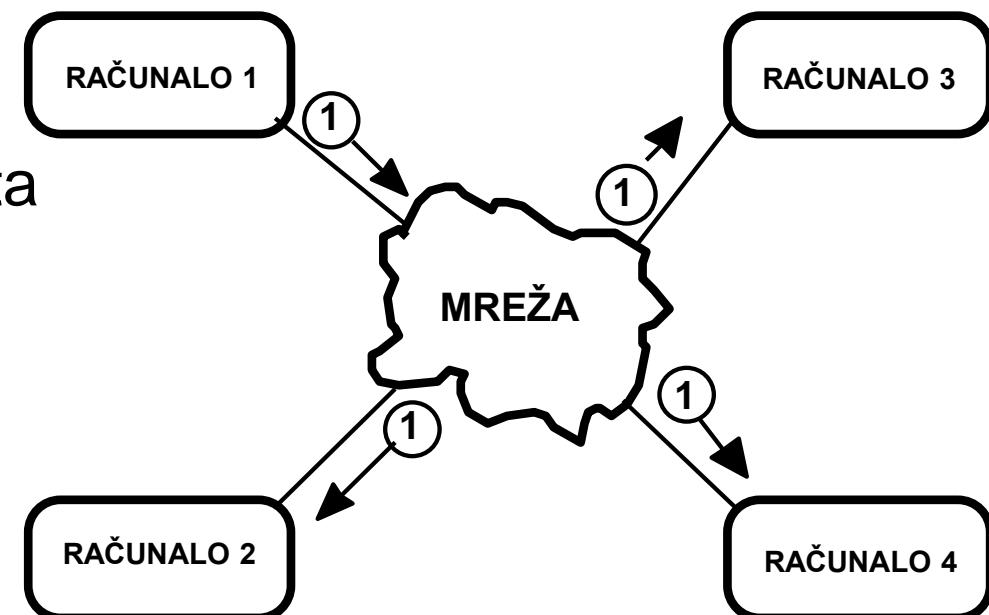
## UVO

### ◆ Zahtjevi:

- Učinkovit prijenos poruka
- Podrška za AOIM
- Upravljanje sjednicom
- Trajni zapis stanja
- Kontrola pristupa, naplata

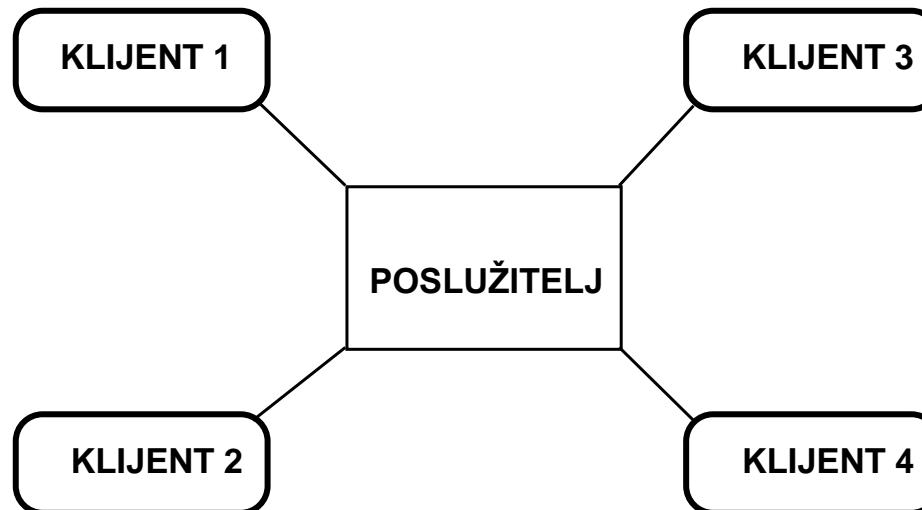
### ◆ Rješenja:

- Klijent/poslužitelj
- Više poslužitelja
- Ravnopravni procesi  
(engl. peer-to-peer)



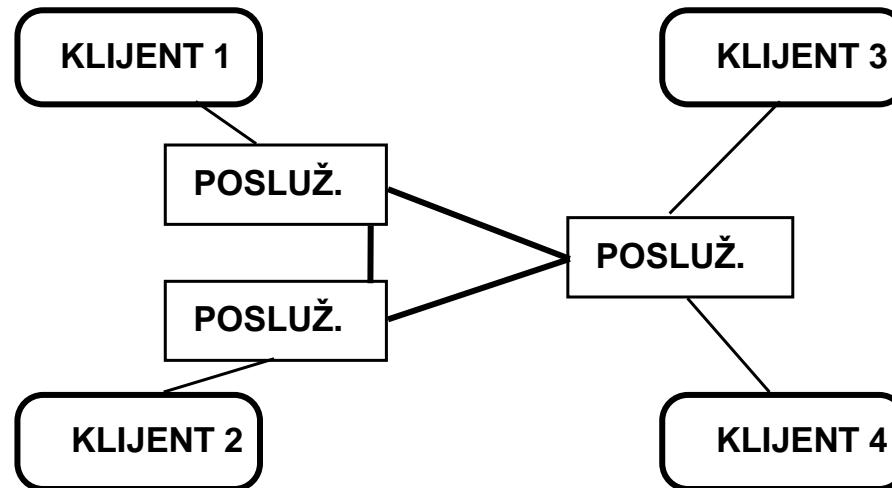
# Klijent/poslužitelj

Zavod za telekomunikacije



- ◆ Vrlo učinkovita arhitektura
- ◆ Filtriranje prometa, kontrola sjednice, trajni zapis, kontrola pristupa i naplata izvedeni na poslužitelju
- ◆ Poslužitelj usko grlo: ograničena prilagodljivost veličini

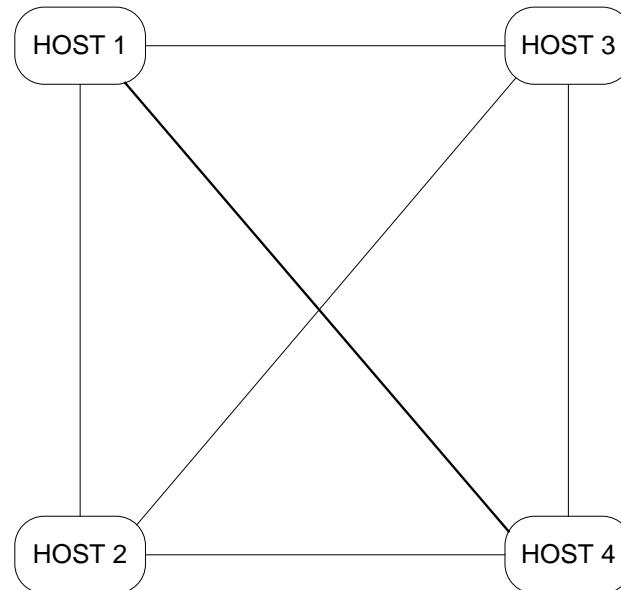
# Više poslužitelja



- ◆ Većina dobrih strana arhitekture klijent/poslužitelj
- ◆ Povećava se kašnjenje
  - potreba za vrlo brzom vezom između poslužitelja
- ◆ Izvedba relativno složena

# Ravnopravni procesi (peer-to-peer)

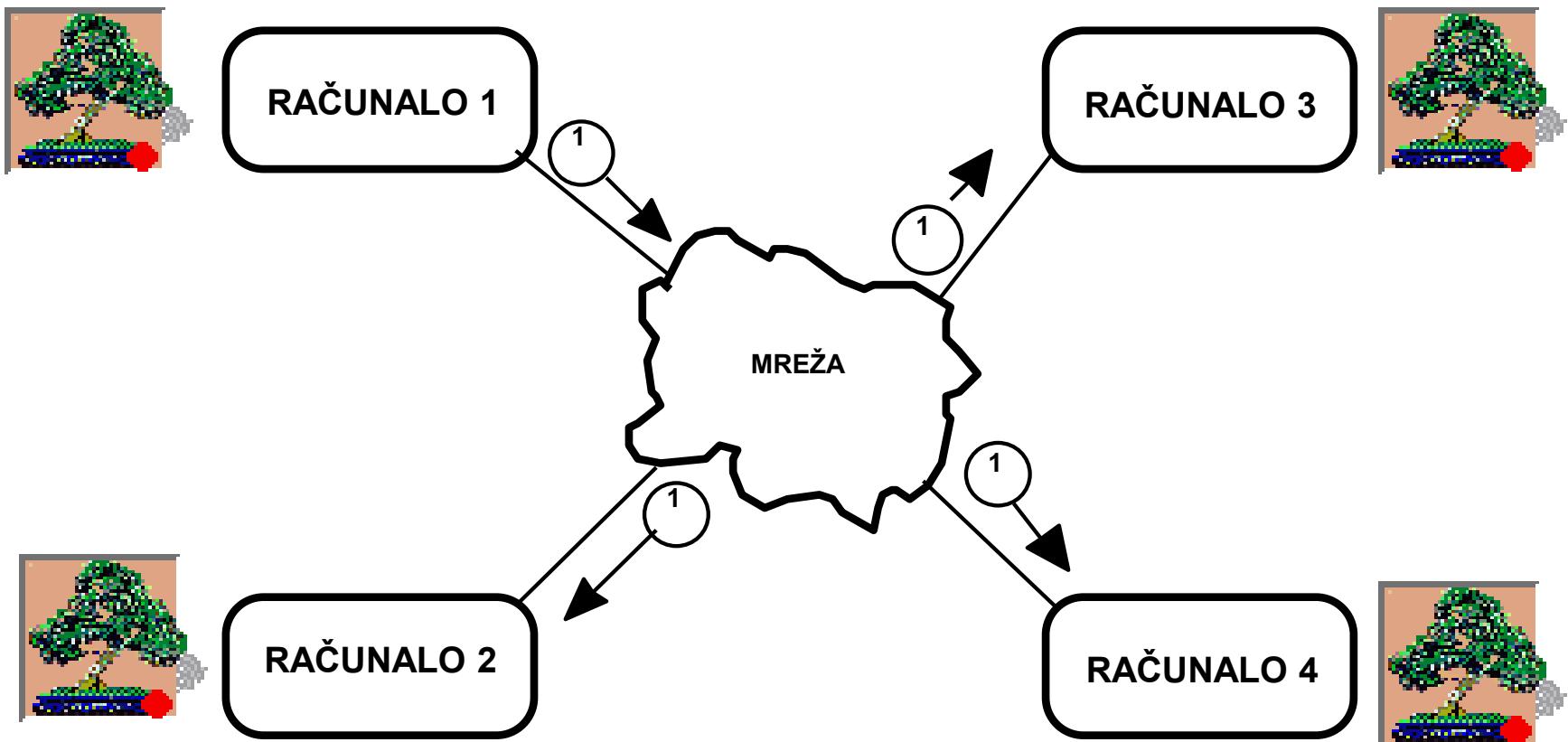
Zavod za telekomunikacije



- ◆ Direktna komunikacija između svih čvorova
- ◆ Prvi sustavi UVO (npr. Doom) koristili su ovo rješenje
- ◆ Problemi: trajni zapis, kontrola pristupa, AOIM, upravljanje sjednicom, kontrola varanja, migriranje informacija u slučaju odspajanja jednog korisnika

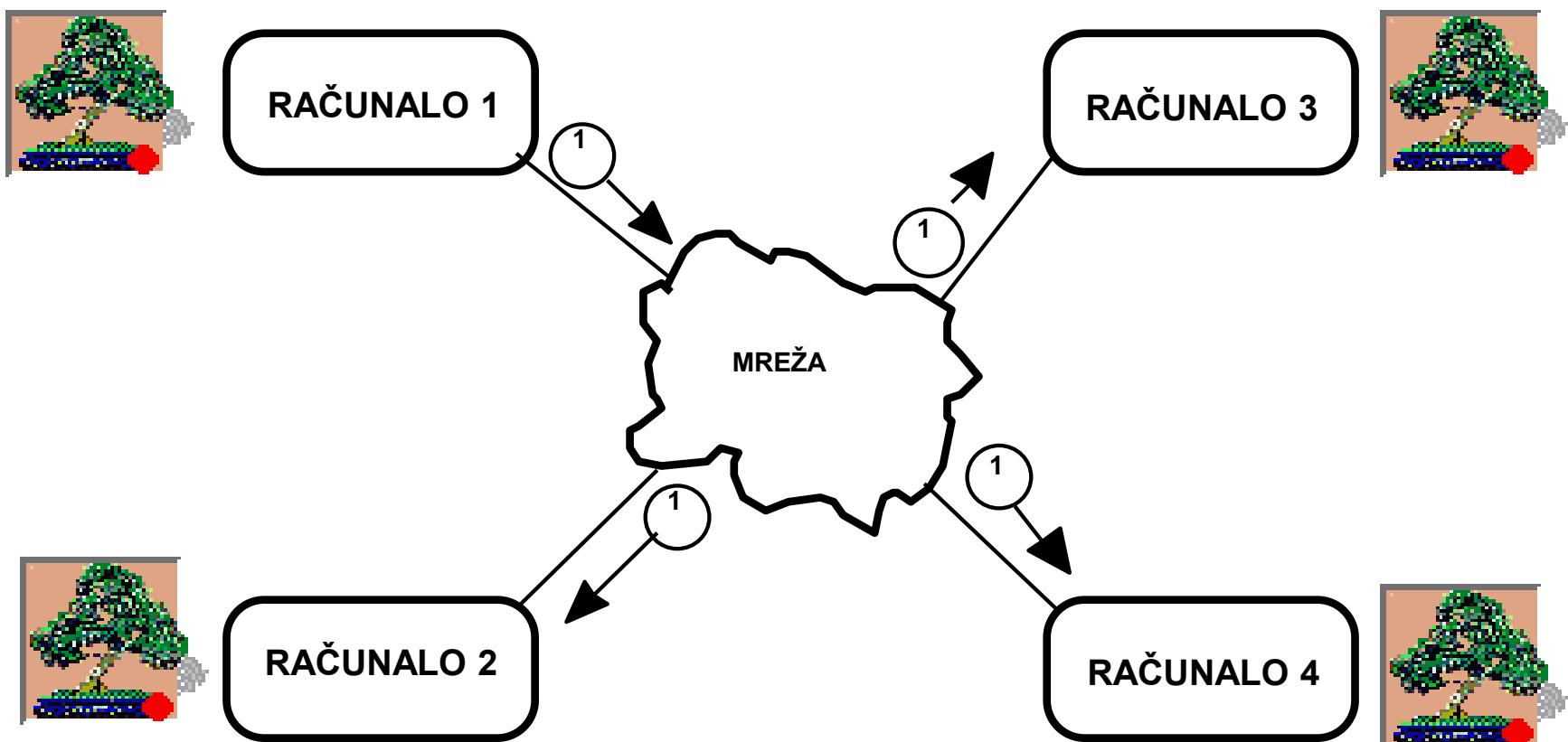
# Jaka konzistentnost

Zavod za telekomunikacije



# Slaba konzistenost

Zavod za telekomunikacije



# Protokol za konzistentnost

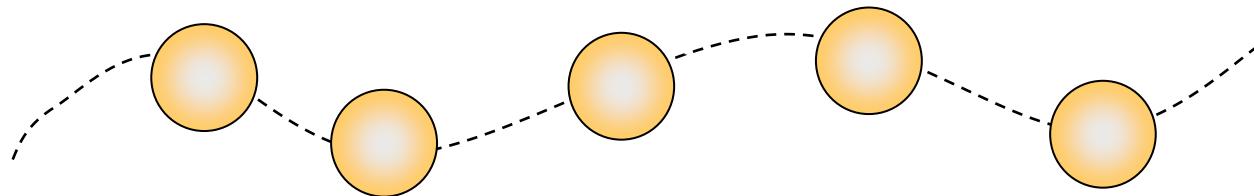
Zavod za telekomunikacije

- ◆ Jaka konzistentnost (zajednički podaci)
  - Ograničena kašnjenjem najsporijeg klijenta – informacija o promjeni mora doći i do tog klijenta
  - Ograničena nivoom interaktivnosti samog UVO-a
    - Primjerice Time To Kill nekih oružja u Call of Duty franšizi je 200 ms - nemoguće omogućiti takvu interaktivnost u sustavu s jakom konzistentnosti
- ◆ Slaba konzistentnost (replicirani podaci)
  - Koristi učestala osvježenja – ograničena propusnošću
  - Ograničena arhitekturom (skalabilnošću) – kako primiti i poslati osvježenja za veliki broj korisnika u slučaju P2P arhitekture?

# Učestala regeneracija stanja

Zavod za telekomunikacije

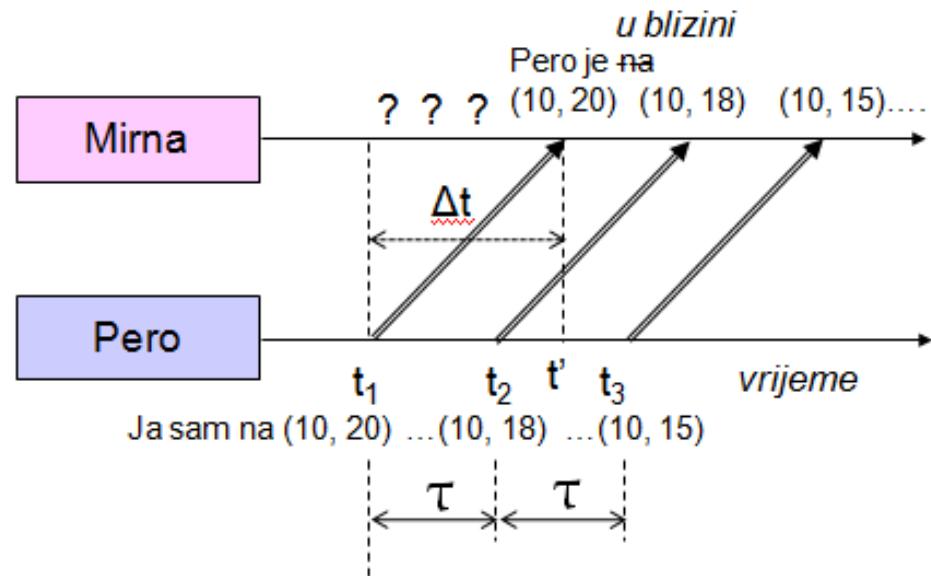
- ◆ Vlasnik entiteta razašilje poruke osvježavanja u pravilnim vremenskim razmacima, koristeći za dostavu nepouzdani protokol



- ◆ Svaka poruka osvježavanja sadrži puni opis stanja entiteta (npr. položaj, orijentaciju, izgled itd.)
- ◆ Svaki primatelj ima lokalni spremnik
- ◆ Primatelji ne potvrđuju primitak poruka, a izgubljene poruke osvježavanja se nadomještaju novijima [primjer 1]
- ◆ Nema očuvanja globalnog redoslijeda poruka osvježavanja

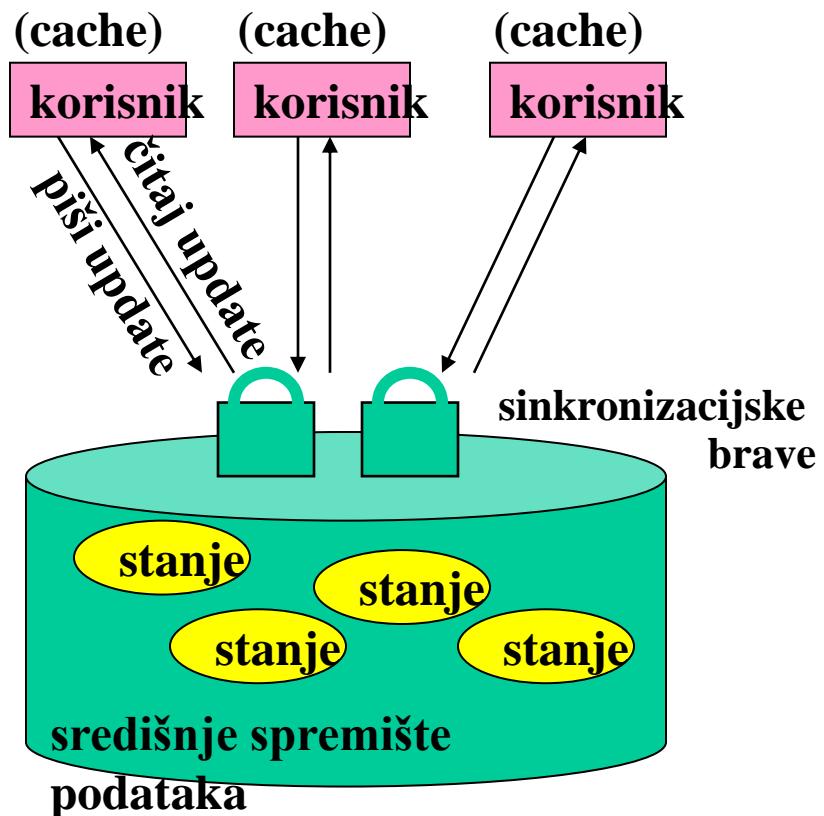
# Odnos konzistentnosti i propusnosti

- ◆ Primjer pokazuje nekonzistentnost uzrokovana kašnjenjem
- ◆ Za maksimalnu konzistentnost, Pero bi trebao čekati da Mirna potvrди primitak, pa tek onda slati iduću poruku: gubitak propusnosti i interaktivnosti
- ◆ Nemoguće je postići visoku dinamičnost u okruženju i visoku konzistenciju



# Središnji repozitorij informacija s zajedničkim podacima

- ◆ Klijent – poslužitelj uz jaku konzistenost
- ◆ Središnji repozitorij sadrži sve podatke o zajedničkom stanju UVO
- ◆ Svi čvorovi u svakom trenutku imaju identični pogled na zajedničko stanje
- ◆ Središnji repozitorij nadzire čitanje i pisanje stanja, kao i poredak pri osvježavanju stanja



# Prednosti i nedostaci središnjeg repozitorija s zajedničkim podacima



Zavod za telekomunikacije

- ◆ Prednosti
  - Jednostavan programski model
  - Garantirana konzistentnost
  - Nema vlasništva nad podacima, središnji poslužitelj se brine o osvježavanju stanja i redoslijedu
- ◆ Nedostaci:
  - Nepredvidivost što se tiče trajanja i čekanja na pristup podacima i osvježavanje
  - Zamjetna količina dodatne obrade (*overhead*) – pouzdani komunikacijski protokol i/ili učestalo slanje stanja od strane poslužitelja”

# Raspodijeljeni repozitorij s zajedničkim podacima



Zavod za telekomunikacije

- ◆ Peer to peer uz jaku konzistentnost
- ◆ Očuvanje konzistentnosti se sada prebacuje na protokol za očuvanje
- ◆ Sve replike su uvijek u istom stanju te imaju kompletnu informaciju o cijelom virtualnom svijetu
- ◆ Primjer korištenja: strategije u stvarnom vremenu (engl. Real Time Strategies) poput Starcraft 2
- ◆ Prednosti: bolja otpornost sustava, nema kašnjenja uzrokovanih komunikacijom sa središnjim sustavom, omogućuje dobru skalabilnost s obzirom na mrežni promet (deterministička simulacija)
- ◆ Mane: najsporiji klijent usporava sustav, ne može se korisiti za sustave s visokom razine interakcije

# Središnji repozitorij s raspodijeljenim podacima



Zavod za telekomunikacije

- ◆ Klijent – poslužitelj sa slabom konzistentnošću
- ◆ Svaki klijent posjeduje repliku stanja UVO-a (najčešće samo lokalnu repliku određenu područjem interesa)
- ◆ Poslužitelj posjeduje centralnu kopiju koja ima „stvarno“ stanje prema kojem se osvježavaju replike na klijentima
- ◆ Poslužitelj sljedeće dvije funkcije:
  - Osigurava jednoznačni poredak poruka osvježavanja stanja
  - Razašilje novo stanje svim zainteresiranim klijentima
- ◆ Najčešća varijanta u današnjim sustavima

# Raspodijeljeni repozitorij s raspodijeljenim podacima

Zavod za telekomunikacije

- ◆ Peer to peer s slabom konzistentnošću
- ◆ Ne postoji kompletna „točna” replika
- ◆ Moguća i parcijalna replikacija virtualnog svijeta – kombinirani podaci sa svih replika daju ukupnu sliku svijeta
- ◆ Za neke primjene dodatna obrada kao kod središnjeg repozitorija može biti neprihvatljiva, ili pak ne trebaju absolutnu konzistentnost
  - primjer: simulator leta – “glatki” pomak, odnosno prikaz promjene položaja bez trzaja ili zamrzavanja slike, je važniji od absolutne točnosti položaja u svakom trenutku

# Primjer 1: učestalost osvježavanja

Zavod za telekomunikacije

Pretpostavimo da izvor šalje 25 poruka osvježavanja svake sekunde. Na temelju toga, prikaz se može osvježiti svakih:

$$1/25 = 0.04 \text{ s} = 40 \text{ ms}$$

Jedna izgubljena poruka utječe na prikaz u trajanju od 40 ms, što za najveći broj primjena, ljudima nije uočljivo.

Dizajner aplikacije može pretpostaviti da će visoka učestalost poruka osvježavanja učiniti male nekonzistencije nevidljivima za korisnike.

Primjeri aplikacija koje koriste učestalo osvježavanje:

- Silicon Graphics Dogfight simulator leta
- Višekorisničke igre na PC-u: Overwatch, Battlefield, World of Warcraft

# Eksplisitno vlasništvo nad objektima

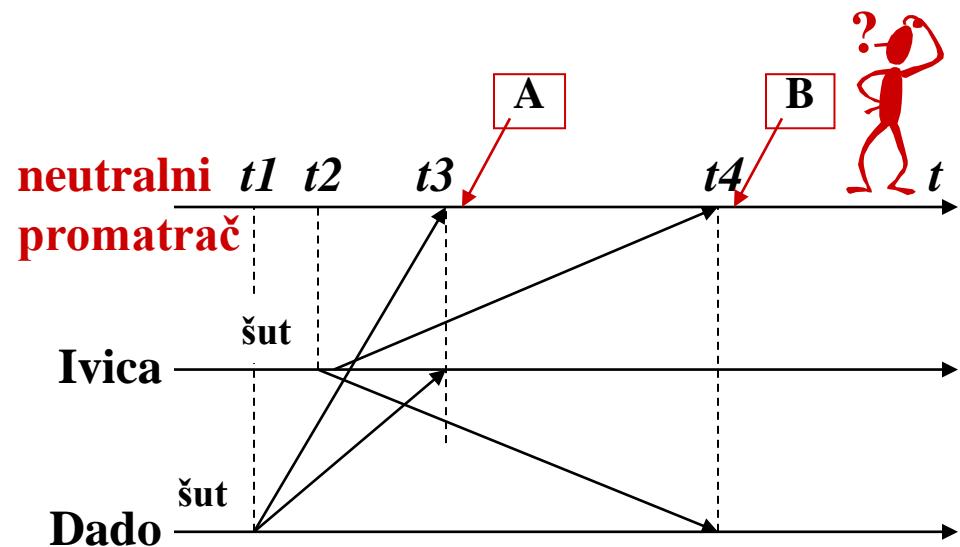
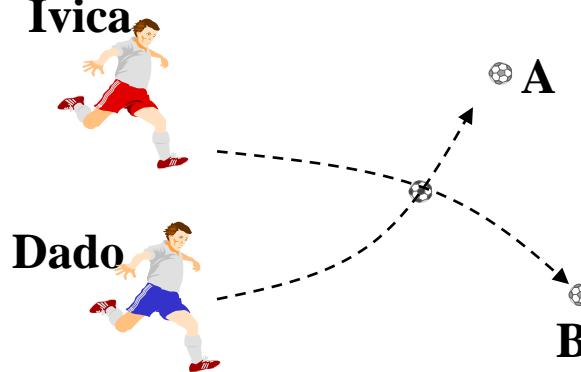
Zavod za telekomunikacije

- ◆ Kada nema potpune konzistentnosti dolazi do problema vlasništva nad objektima
- ◆ Treba spriječiti da više igrača istovremeno mijenja stanje entiteta [primjer 2, primjer 3]
  - Na primjer, promjena položaja je pisanje variabile stanja položaja entiteta ( $x, y, z$ )
- ◆ Uvodi se eksplisitno vlasništvo nad entitetom
  - Tipičan primjer je korisnikov *avatar*, čiji je vlasnik korisnik koji njime upravlja
  - Drugim objektima upravlja poslužitelj *lock manager*
- ◆ *Lock manager* osigurava da svaki entitet u zadanim trenutku ima samo jednog vlasnika

# Primjer 2: nema vlasništva

Zavod za telekomunikacije

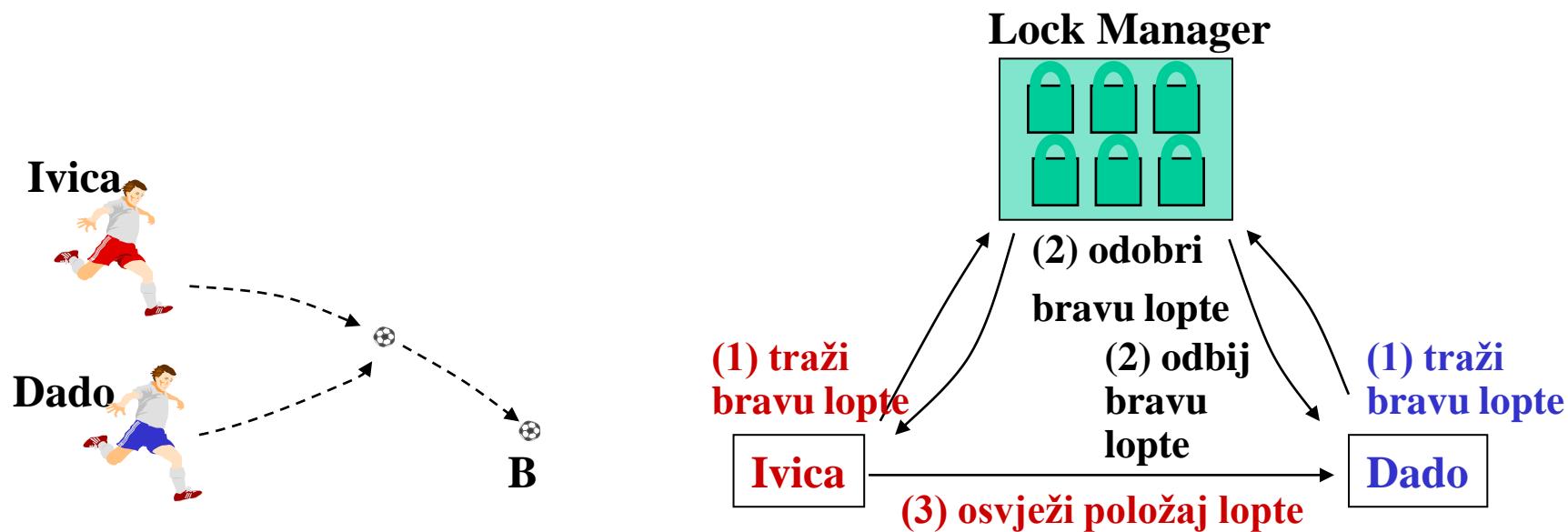
Ivica i Dado igraju nogomet i pokušavaju istovremeni šutnuti loptu. Svaki host šalje poruku osvježavanja stanja lopte (položaj). Neki promatrači mogu prvo primiti Ivičinu poruku, a drugi Dadinu. Osim ako se Ivica i Dado slože oko toga tko je zapravo šutnuo loptu, svaki će nastavljati osvježavati položaj neovisno o drugome. Promatrači će vidjeti titranje lopte između novih položaja ovisno o pristizanju poruka osvježavanja od Ivice i Dade.



# Primjer 3: eksplicitno vlasništvo

Zavod za telekomunikacije

Ivica i Dado izdaju zahtjev za vlasništvom poslužitelju *lock manager* prije osvježavanja stanja lopte. Ovisno o politici upravljanja bravama (npr. FIFO, round-robin, prioriteti, itd.), *lock manager* daje bravu jednom od korisnika, u ovom primjeru, Ivici. Ivica onda može mijenjati položaj lopte dok god je u vlasništvu brave.



# Prednosti i nedostaci distribuiranih pristupa s učestalom regeneracijom stanja



Zavod za telekomunikacije

## ◆ Prednosti

- Pristup jednostavan za izvedbu: nema centralnog poslužitelja, protokola konzistencije, niti (u nekim slučajevima) brava
- Može se podržati velik broj korisnika

## ◆ Nedostaci:

- Značajni zahtjevi na propusnost [primjer 4]
- Različita učestalost osvježavanja za različite entitete
- Posljedice mrežnog kašnjenja i kolebanja kašnjenja
  - problemi s kauzalnošću, odnosno, uzročno-posljedični odnosima u virtualnom svijetu [primjer 5]
  - povremeno “smrzavanje” slike, neujednačenost pokreta

# Primjer 4: zahtjevi na propusnost

Zavod za telekomunikacije

Pretpostavimo da izvor šalje 30 poruka osvježavanja u sekundi i da je veličina poruke (PDU) 144 byte. (To je inače stvarna veličina PDU za potpuno artikulirani ljudski lik u simulaciji prema standardu Distributed Interactive Simulation).

Za svaku PDU imamo:

$$144 \text{ byte} * 8 \text{ bit/byte} * 30 \text{ 1/s} = 34\,560 \text{ bit/s}$$

Dakle, na 10 Mbit/s LAN (i zanemarujući *overhead*, koji inače nije beznačajan!), možemo imati najviše 289 virtualnih ljudi.

Ako koristimo modemsku vezu s 56 kbps, možemo imati samo 1 virtualni lik.

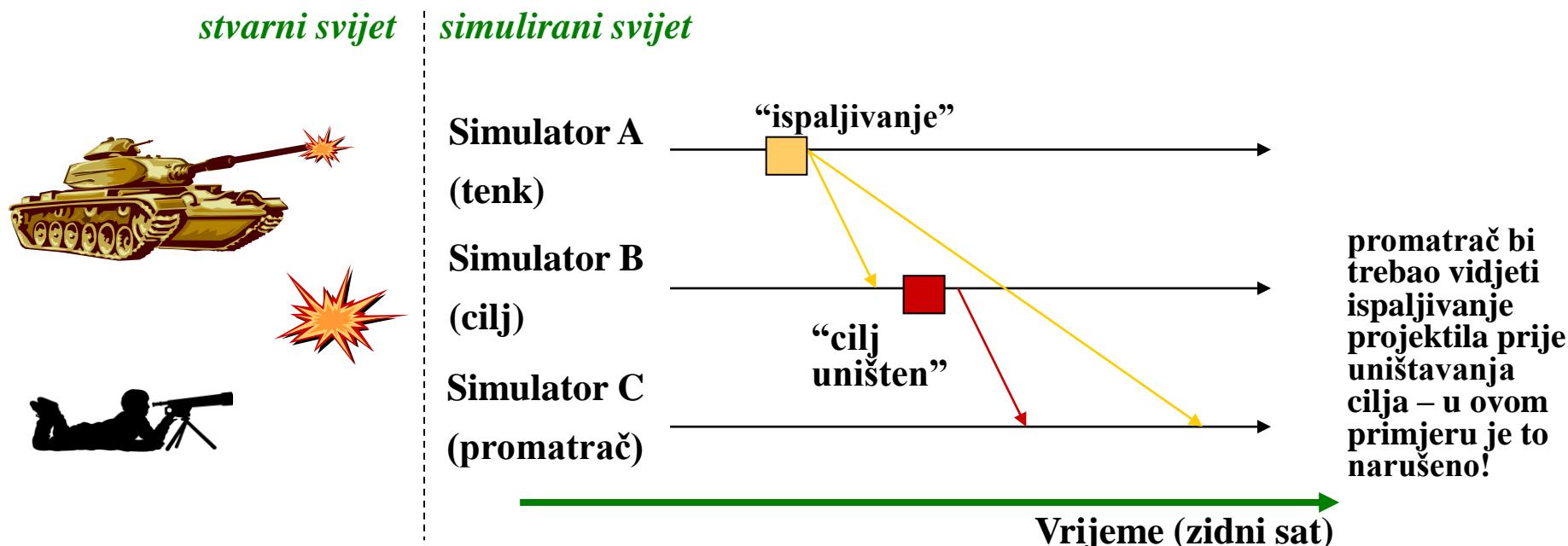
Da bi se moglo podržati više korisnika, koristi se smanjena učestalost poruka osvježavanja, kao i kompresija poruka. Vidljivo je da se uz osvježavanje "naslijepo" ne mogu podržati UVO s velikim brojem entiteta.

# Primjer 5: problem s kauzalnošću

Zavod za telekomunikacije

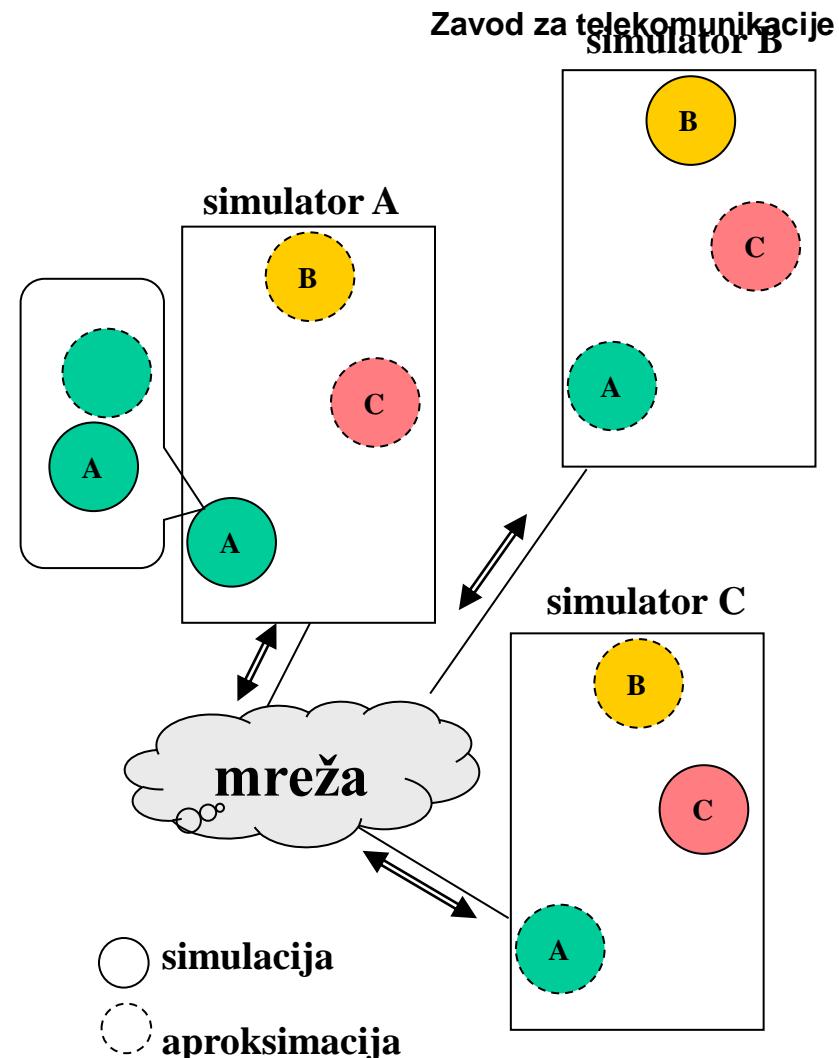
U UVÖ, kao i u stvarnom svijetu, korisnici očekuju da vrijedi kauzalnost, odnosno, uzročno-posljedični odnosi.

U UVÖ se taj odnos ne može baš uvijek očuvati zbog nepredvidivog kašnjenja u mreži.



# Mrtva procjena (Dead reckoning) 1/2

- ◆ Na strani svakog čvora koristi se i simulacija i aproksimacija trenutnog stanja
- ◆ Aproksimacija je jednostavnija za računanje
- ◆ Svaki čvor za vlastite entitete šalje poruke osvježavanja samo ako razlika između simulacije i aproksimacije prijeđe zadani prag
- ◆ Svaki čvor radi predikciju (predviđanje) stanja udaljenih entiteta, npr. položaja i orientacije, na temelju lokalno pohranjene informacije



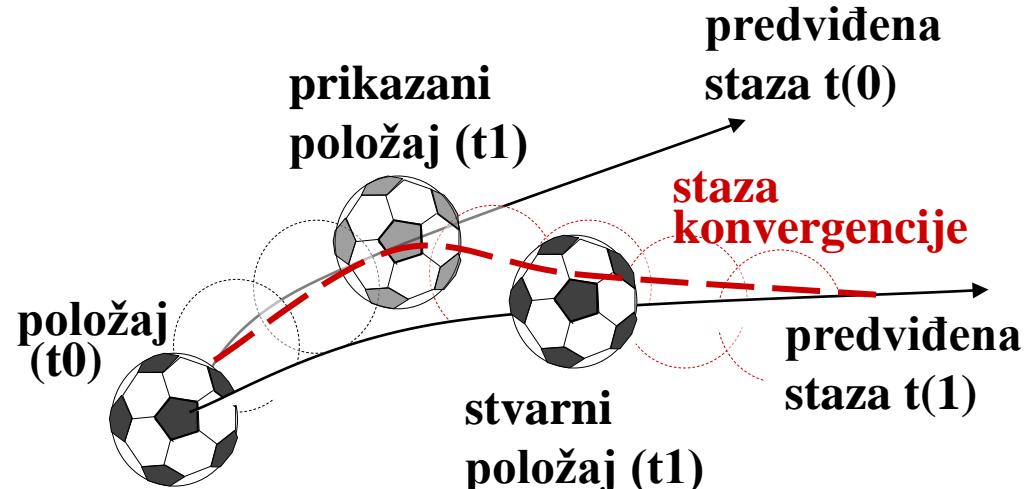
# Mrtva procjena (Dead reckoning) 2/2

Zavod za telekomunikacije

- ◆ Pristup je pogodan za UVÖ s velikim brojem sudionika
- ◆ Algoritam mrtve procjene sastoji se od predikcije i konvergencije
- ◆ Uvjet za primjenu metode je (barem djelomična) predvidivost kretanja entiteta kako bi se mogla primijeniti predikcija

# Algoritam mrtve procjene

- ◆ **Predikcija:**  
izračunavanje  
sadašnjeg stanja na  
temelju prethodno  
primljenih poruka  
osvježavanja
- ◆ **Konvergencija:**  
korekcija staze  
dobivene predikcijom  
na temelju novo-  
primljenih poruka  
osvježavanja  
("izglađivanje")



# Prednosti i nedostaci primjene mrtve procjene

## ◆ Prednosti:

- Smanjen promet i zahtjevi na propusnost
- Svaki čvor računa algoritam mrtve procjene neovisno o drugima
- Moguće je podržati velik broj korisnika
- Otpornost na gubitke paketa

## ◆ Nedostaci:

- Slaba konzistentnost – nema garancije da će svi čvorovi imati identično stanje istog entiteta
- Distribuirane simulacije su složene za izvedbu, održavanje i vrednovanje

# Oblikovanje programskog rješenja UVO

- ◆ Strukturiranje virtualnog prostora
- ◆ Arhitektura raspodijeljene aplikacije UVO
- ◆ Vrste mrežnog prometa i komunikacijski protokoli
- ◆ Način distribucije poruka
- ◆ Izvedba simulacijske petlje UVO

# Strukturiranje virtualnog prostora

Zavod za telekomunikacije

- ◆ Podijeliti okruženje na jedinice prihvatljive po:
  - broju korisnika u svakoj jedinici
  - složenosti geometrije svake jedinice
  - preciznosti koordinata
- ◆ Primjer: World of Warcraft
  - Korisnici podijeljeni na veći broj poslužitelja, po nekoliko tisuća korisnika po poslužitelju (engl. žargon *shard*)
  - Nema interakcije s korisnicima na drugom poslužitelju
- ◆ Primjer: Second Life
  - Virtualni svijet podijeljen u ćelije 256 x 256 m
  - Za svaku ćeliju odgovoran zaseban poslužitelj

# Vrste mrežnog prometa u UVO

Zavod za telekomunikacije

Učitavanje  
Poruke sustava  
Dogadjaji  
Osvježavanje stanja  
Tekst  
Zvuk  
Video

3D predmeti  
Teksture (slike)  
Modeli ponašanja  
Modeli prikaza korisnika

Predmeti  
Korisnici

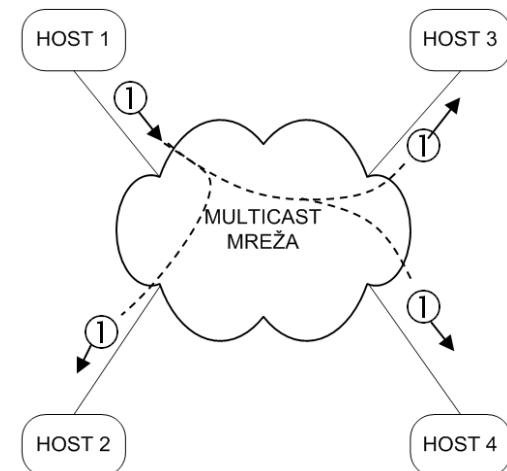
Animacija  
Deformacija  
Položaj tijela  
Izraz lica  
Kretanje

# Mrežni zahtjevi i protokoli za pojedine tipove podataka

Vrsta prometa	Količina podataka	Potrebna pouzdanost	Protokol
Učitavanje	  	  	TCP
Poruke sustava		  	TCP
Događaji		 	TCP
Osvježavanje stanja	 		UDP
Tekst		 	TCP
Zvuk	 		UDP (RTP)
Video	  		UDP (RTP)

# Način distribucije poruka

- ◆ Jednoodredišna komunikacija
  - Najčešći način povezivanja
  - Neučinkovitost se u praksi kompenzira izvedbom AOIM na poslužitelju
- ◆ Višeodredišno razašiljanje
  - Učinkovitije, ali složenija izvedba



# Izvedba simulacijske petlje UVO

Zavod za telekomunikacije

- ◆ Zahtjev:

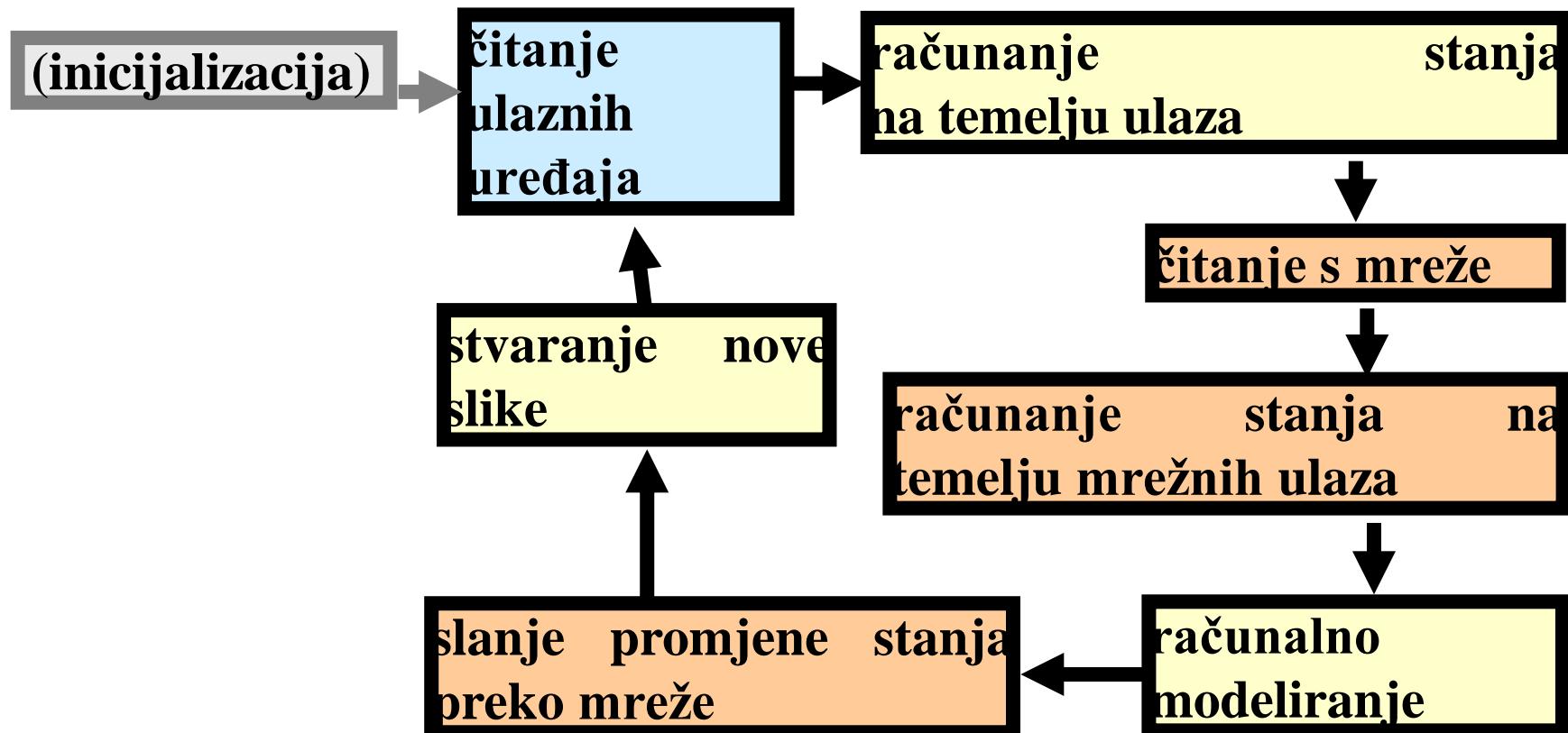
- Omogućiti nesmetano i efikasno izvođenje većeg broja modula

- ◆ Rješenja:

- ~~Jednonitno programsko rješenje~~
  - Višenitno programsko rješenje

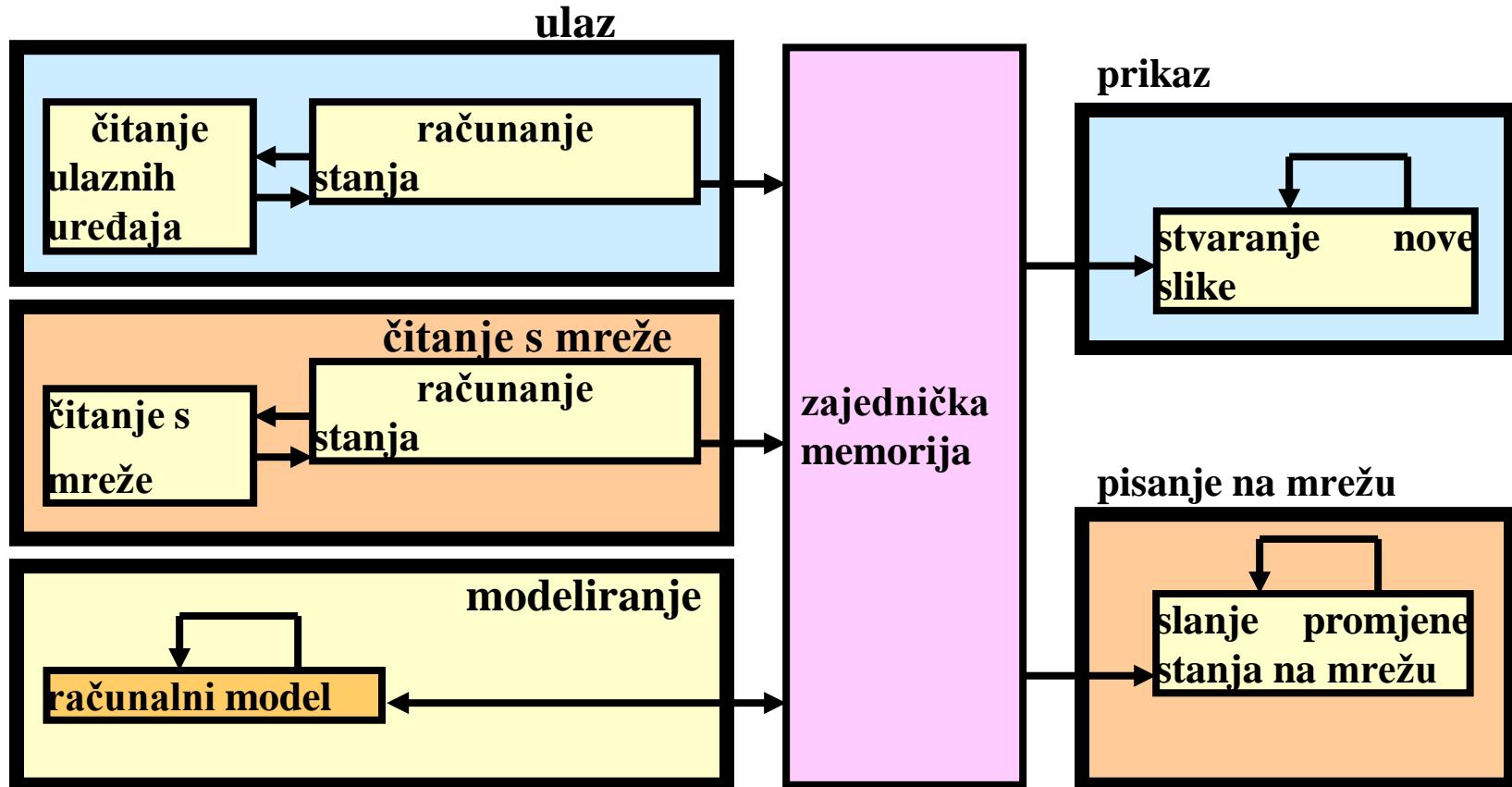
# Jednonitno programsko rješenje

Zavod za telekomunikacije



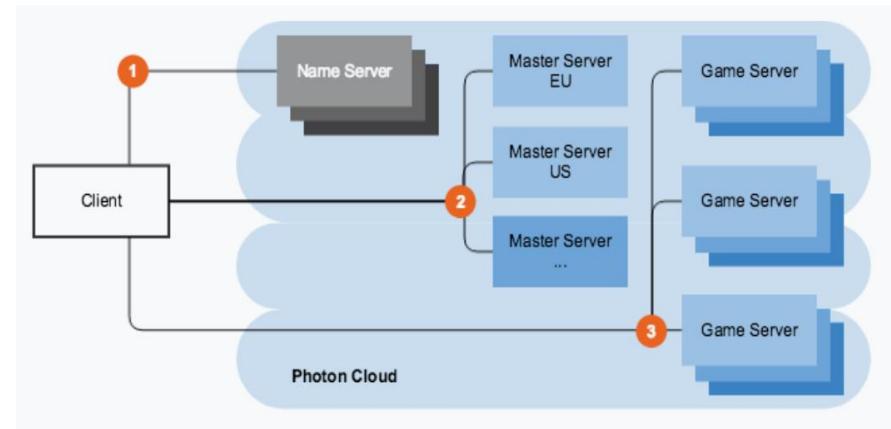
# Višenitno programsko rješenje

Zavod za telekomunikacije



# Izrada višekorisničke aplikacije u Unity sustavu za izradu igara

- ◆ Trenutno unutar Unity postoji sustav pod nazivom Unet koji je zastario te je u razvoju novi sustav
- ◆ Jednostavno se može zamijeniti postojećim bibliotekama
  - Photon Unity Networking (PUN)
  - Mirror Networking (otvorenog koda)
- ◆ PUN omogućuje
  - Sinkronizaciju objekata kroz dodavanje komponente PhotonView objektu koji se sinkronizira
  - Poziv udaljenih procedura
  - Podizanje događaja
- ◆ Detaljnije u okviru zadaće



# Virtualni svjetovi

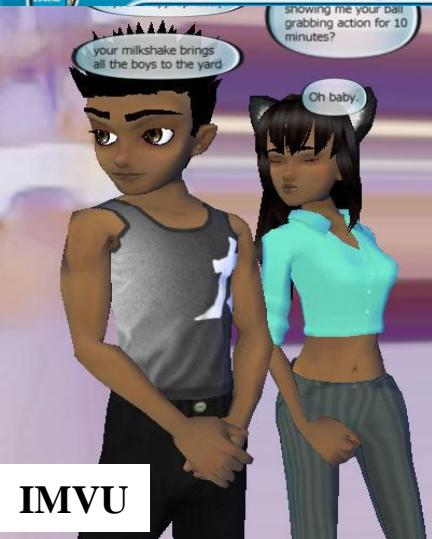
# Virtualni svjetovi

Zavod za telekomunikacije

- ◆ engl. Virtual Worlds



# Virtualni svjetovi - primjeri



# Virtualna ekonomija

Zavod za telekomunikacije

- ◆ Virtualnim novcem kupuju se dobra i usluge
- ◆ Primjeri usluga: kampiranje, rad u trgovini, zabava (često “za odrasle”), izrada sadržaja
- ◆ Primjeri dobara: zgrade, vozila, nakit, odjeća, animacije, biljke, životinje, umjetnine
- ◆ Virtualni novac se kupuje i prodaje
  - Primjer: SL LindeX: 12.12.2018. tečaj 253,90 L\$ za 1\$
  - Pomoću dane valute trgovci unutar Second Lifea mogu zarađivati kreirajući sadržaje (odjeću, obuću, kuće itd.)

# Tehnička svojstva

Zavod za telekomunikacije

- ◆ Pored ranije opisanih osnovnih svojstava, karakteristike specifične za virtualne svjetove:
- ◆ Zajednički, perzistentni svijet za sve korisnike
- ◆ Korisnici stvaraju svijet!
- ◆ Goleme količine sadržaja koji se stalno mijenja
  - Korisnici kupuju “zemlju”, grade, trguju, programiraju
- ◆ Kada se korisnik kreće kroz svijet, mora stalno učitavati nove ili osvježene sadržaje

# Mrežni zahtjevi

Zavod za telekomunikacije

- ◆ Osvježavanje stanja korisnika: slično kao u igrama
- ◆ Daleko veći promet generira učitavanje modela svijeta/predmeta i njihovo osvježavanje prilikom promjena
- ◆ Načini na koje se smanjuje promet:
  - Lokalno spremanje (engl. *cache*) smanjuje promet pri drugoj i idućim posjetama istom prostoru
  - UDP za sve osim kritičnih poruka, npr. autentikacija
  - Kompresija (JPEG, zlib)
  - Primjer: Second Life – koristi sve navedene tehnike

# Primjer: Second Life

Zavod za telekomunikacije

- ◆ Mjerenja prilikom posjete Veleposlanstvu Švedske
  - Prva posjeta: blokiranje, dugo čekanje na učitavanje predmeta
  - Druga posjeta: normalan rad, prosjek prometa na klijentu 640 kbit/s

Smjer poruka	Tip prometa	Promet (kb), 1. posjeta	Promet (kb), 2. posjeta
Poslužitelj - klijent	Učitavanje predmeta, tekstura...	11563	2792
	Osvježavanje stanja, događaji	91	50
	Poruke sustava i ostalo	276	151
Klijent -poslužitelj	Učitavanje (zahtjevi)	71	14
	Osvježavanje stanja, događaji	101	87
	Poruke sustava i ostalo	112	69

- ◆ Zbog mrežnih i računalnih zahtjeva jedan poslužitelj služi oko 40 klijenata
  - Usporedba s igrama: stotine ili tisuće klijenata po poslužitelju

# Korisnička populacija

- ◆ Virtualni svjetovi su vrhunac popularnosti imali u 2000-tima
  - Club Penguin – čak 200 milijuna registriranih računa – ugašen 2017., a nasljednik Club Penguin Island nije uspio te je najavljeno gašenje
  - Second Life – milijun korisnika u 2007 godini – još uvijek radi, ali s vrlo malim brojem korisnika (oko 45 000)
  - Današnja popularnost puno manja, ali još uvijek postoji veliki broj virtualnih svjetova
- ◆ Minecraft – igra ili virtualni svijet?
  - Prosinac 2017 – 74 milijuna aktivnih korisnika
- ◆ Virtualna i proširena stvarnost donose nove oblike virtualnih svjetova
  - Facebook Horizons
  - Holoportation

# Lista virtualnih svjetova (nepotpuna)

Zavod za telekomunikacije

- ◆ Active Worlds
- ◆ AltspaceVR
- ◆ Anyland
- ◆ Avakin Life
- ◆ Bigscreen
- ◆ Ceek
- ◆ Cisco Spark in VR
- ◆ Community Garden
- ◆ Cryptovoxels
- ◆ DiveReal
- ◆ Edorble
- ◆ EmbodyMe
- ◆ Endless Riff
- ◆ Engage
- ◆ Eventual VR
- ◆ Ever, Jane
- ◆ Facebook Spaces
- ◆ Geekzonia
- ◆ Guru Gedara
- ◆ High Fidelity
- ◆ Hyperfair VR
- ◆ IMVU
- ◆ Inlight Spark
- ◆ JanusVR
- ◆ Kiss or Kill
- ◆ Kitely
- ◆ LivCloser
- ◆ LiveLike
- ◆ Mark Space
- ◆ Mozilla Hubs
- ◆ Myst Online: Uru
- ◆ Live
- ◆ NeosVR
- ◆ Neutrans
- ◆ Occupy White Walls
- ◆ Oculus Rooms
- ◆ Pluto VR
- ◆ Rec Room
- ◆ Rumii
- ◆ Sansar
- ◆ Second Life
- ◆ Sinespace
- ◆ Somnium Space
- ◆ STYLY
- ◆ SurrealVR
- ◆ Topik
- ◆ There.com
- ◆ TheWaveVR
- ◆ Twinity
- ◆ Center
- ◆ VirBELA
- ◆ Virtual Paradise
- ◆ Virtual Universe
- ◆ VRChat
- ◆ vTime
- ◆ Worldopoly
- ◆ Worlds Adrift

# Poslovni modeli

Zavod za telekomunikacije

- ◆ Većina virtualnih svjetova koristi kombinacije triju modela
  - Ovlašavanje, kanal za prodaju
  - Pretplate
    - Često besplatno osnovno članstvo, naplata za "premium" članstvo
  - Prodaja virtualnih dobara
    - Virtualni novac koji se može trošiti unutar svijeta
    - Dodaci za avatare
- ◆ Ograničenja poslovnog modela s oglašavanjem
  - Svjetovi za djecu – mnogi roditelji nisu skloni izlaganju djece reklamama
  - Tvrтke nisu sklone oglašavanju u nekontroliranom okruženju
  - Troškovi: stalna prisutnost npr. u SL zahtjeva zaposlenika

# Primjeri poslovnih modela

Zavod za telekomunikacije

Svijet	Vlasnik	Ciljani korisnici	Područje	Poslovni model	Korisnici (2008)	Rast 07-08
BarbieGirls	Mattel	Djevojčice	SAD, svijet	Prodaja igračaka, pretplate	3,3M	26%
ClubPenguin	Disney	Djeca	SAD, svijet	Pretplate	10,8M	23%
CyWorld	SK Telecom	Tinejdžeri, 20te	Koreja	Virtualna dobra	13,7M	-1%
Habbo Hotel	Sulake, FI	Tinejdžeri	Europa, svijet	Oglašavanje, virtualni novac	6,3M	52%
NeoPets	Viacom	Djeca	Svijet	Premium članstvo, oglašavanje, virtualna dobra	6,1M	-16%
ActiveWorlds		Odrasli	SAD, svijet	Pretplate, hosting	0,1M	181%
Lively	Google	Odrasli	Svijet	Oglašavanje	0,6M	Ugašen
IMVU		Odrasli	SAD, svijet	Premium članstvo, oglašavanje	4,8M	22%
Second Life	Linden Labs	Odrasli	Svijet	Pretplate, virtualni novac i dobra, oglašavanje	1,6M	-45%

# Literatura

Zavod za telekomunikacije

- ◆ S. Singhal & M. Zyda, *Networked Virtual Environments: Design and Implementation*, Addison-Wesley, 1999 (poglavlja 4, 5)
- ◆ Abdelkhalek, A.; Bilas, A. and Moshovos, A. (2001), Behavior and Performance of Interactive Multiplayer Game Servers, Proc. Int. IEEE Symposium on the Performance Analysis of Systems and Software.
- ◆ Gao Huang, Meng Ye, Long Cheng, "Modeling System Performance in MMORPG", in IEEE CSG Workshops, 2004, pp. 512-518.
- ◆ Seay, A. F., Jerome, W. J., Lee, K. S., and Kraut, R. E. 2004. Project massive: a study of online gaming communities. In *CHI '04 Extended Abstracts on Human Factors in Computing Systems* (Vienna, Austria, April 24 - 29, 2004). CHI '04. ACM, New York, NY, 1421-1424.
- ◆ Kumar, Chhugani, Kim, Kim, Nguyen, Dubey: "Second Life and the New Generation of Virtual Worlds", IEEE Computer, 2008
- ◆ D-Lib Magazine, December 2005, Volume 11, Nr. 12 (John Kirriemuir)
- ◆ ESA: Entertainment Software Association, <http://www.theesa.com/>
- ◆ DFC Intelligence, <http://www.dfcint.com>
- ◆ "Telepresence report", Howard S. Lichtman, Human Productivity Lab
- ◆ Virtual Worlds Management, <http://www.virtualworldsmanagement.com/>
- ◆ JP Morgan Global Equity Research, 2009

# Virtualna okruženja

*Umrežena virtualna okruženja –  
digitalne igre*

*Prof. dr. sc. Igor S. Pandžić*

*Prof. dr. sc. Maja Matijašević*

*dr. sc. Mirko Sužnjević*

# Sadržaj predavanja

Zavod za telekomunikacije

- ◆ Uvod
- ◆ Tržište i poslovni modeli
- ◆ Funkcijska arhitektura
- ◆ Karakteristike mrežnog prometa
- ◆ Utjecaj mrežnih degradacija
- ◆ Skalabilnost

# Konceptualne razlike između igara i virtualnih svjetova

## Fixed Synthetic



World of Warcraft  
EverQuest  
Uru

Puzzle Pirates  
EVE Online

Habbo Hotel

There.com  
Second Life  
Active Worlds

## Games

(Goal-oriented/Ludic)

- ◆ Postoji cilj
- ◆ Postoje formalizirana struktura
- ◆ Postoje ograničenja na kako se ti ciljevi mogu postići  
(mehanike)

## Metaverses

(Open-ended/Paidiaic)

- ◆ Ne postoji krajnji cilj
- ◆ **Postoji niz mogućih aktivnosti i opcija za različite društvene interakcije**

Pearce, C. (2011). *Communities of play: Emergent cultures in multiplayer games and virtual worlds*. MIT Press.

# Tehničke razlike između igara i virtualnih svjetova



Zavod za telekomunikacije

- ◆ Perzistentnost
  - Virtualni svjetovi su uglavnom perzistentni – postoje i mijenjaju se i dok korisnik nije u njima
  - Kod igara je puno češće iznova stvaranje virtualnog svijeta iz pohranjenih podataka (primjerice svaka bitka u Call of Duty se odvija na istoj mapi koja se iznova kreira)
- ◆ Mijenjanje 3D svijeta
  - Virtualni svjetovi dopuštaju dodavanje novih slika, tekstura, zvukova, videa i 3D objekata (primjerice Second Life)
  - Igre ne dopuštaju dodavanje novih informacija u svoj klijent (iako mogu dopuštati mijenjanje svijeta primjerice Fortnite)
- ◆ Karakteristike mrežnog prometa – jako ovisne o „promjenjivosti“ 3D svijeta jer ako korisnici mogu dodati nove informacije, drugi korisnici moraju te iste informacije preuzeti

# Terminologija

Zavod za telekomunikacije

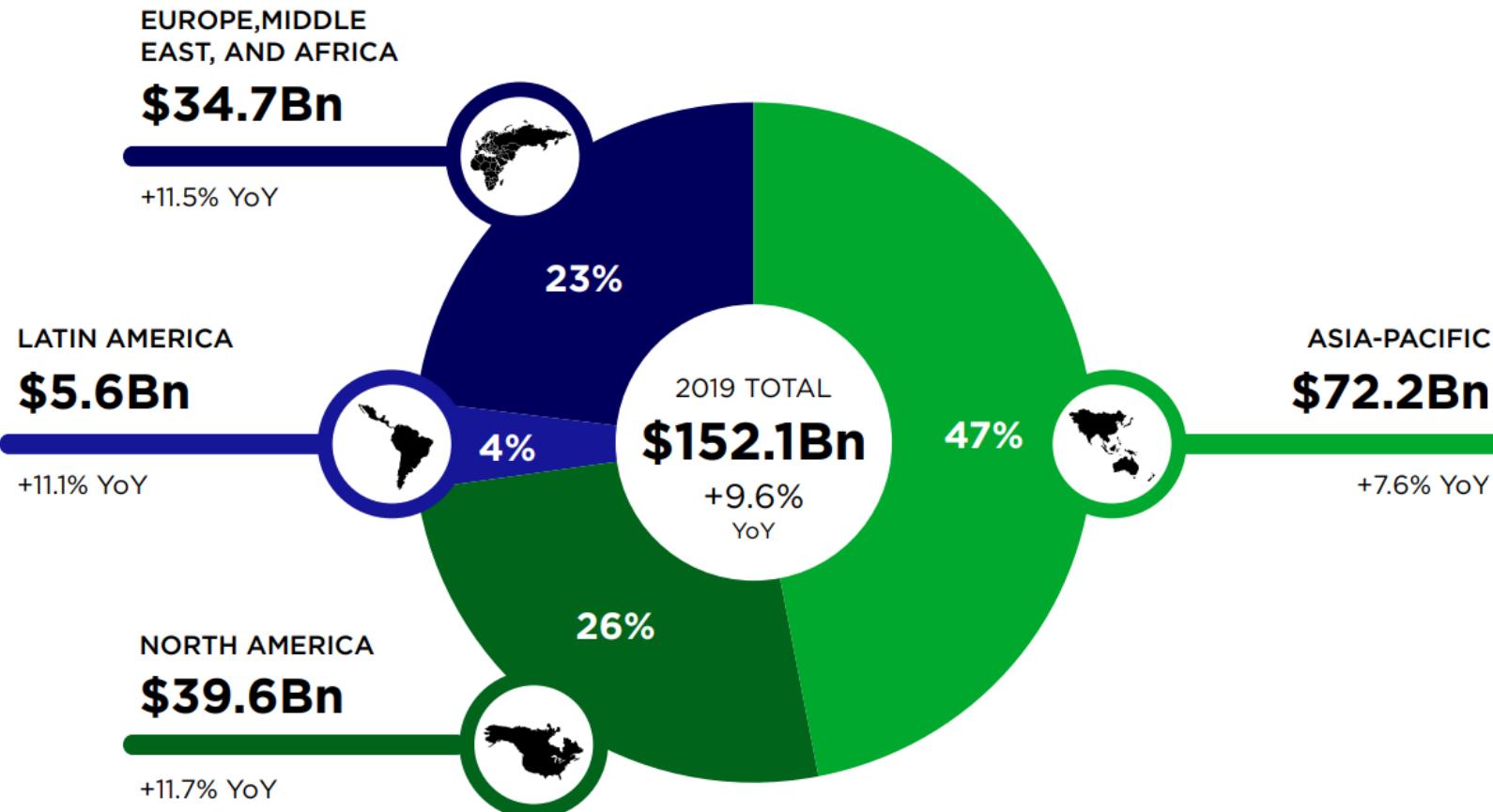
- ◆ Digitalne igre - igre koje se igraju na elektroničkom uređaju kroz korisničko sučelje, čiji prikaz na ekranu kontrolira program
  - “Video games” – igre na konzoli
  - “Computer games” – igre na računalu
- ◆ Višekorisničke igre – više od jednog igrača može istovremeno igrati u istoj virtualnoj okolini
- ◆ Umrežene igre – potrebna mrežna povezanost
- ◆ Fokus predavanja – umrežene višekorisničke igre kao dio umreženih virtualnih okruženja

# Tržište i poslovni modeli

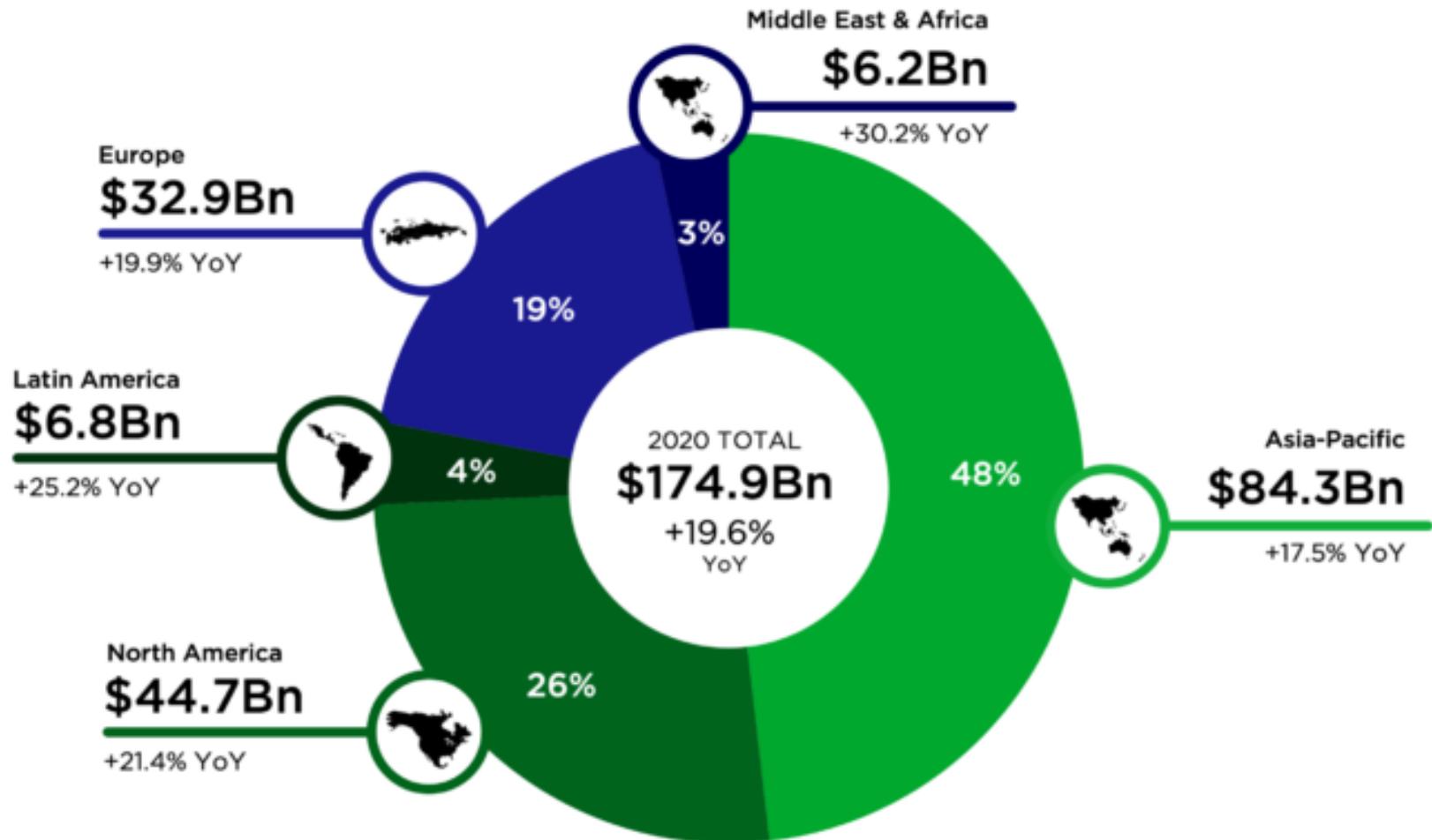
# Tržište igara – procjene vrijednost 2019.



Zavod za telekomunikacije

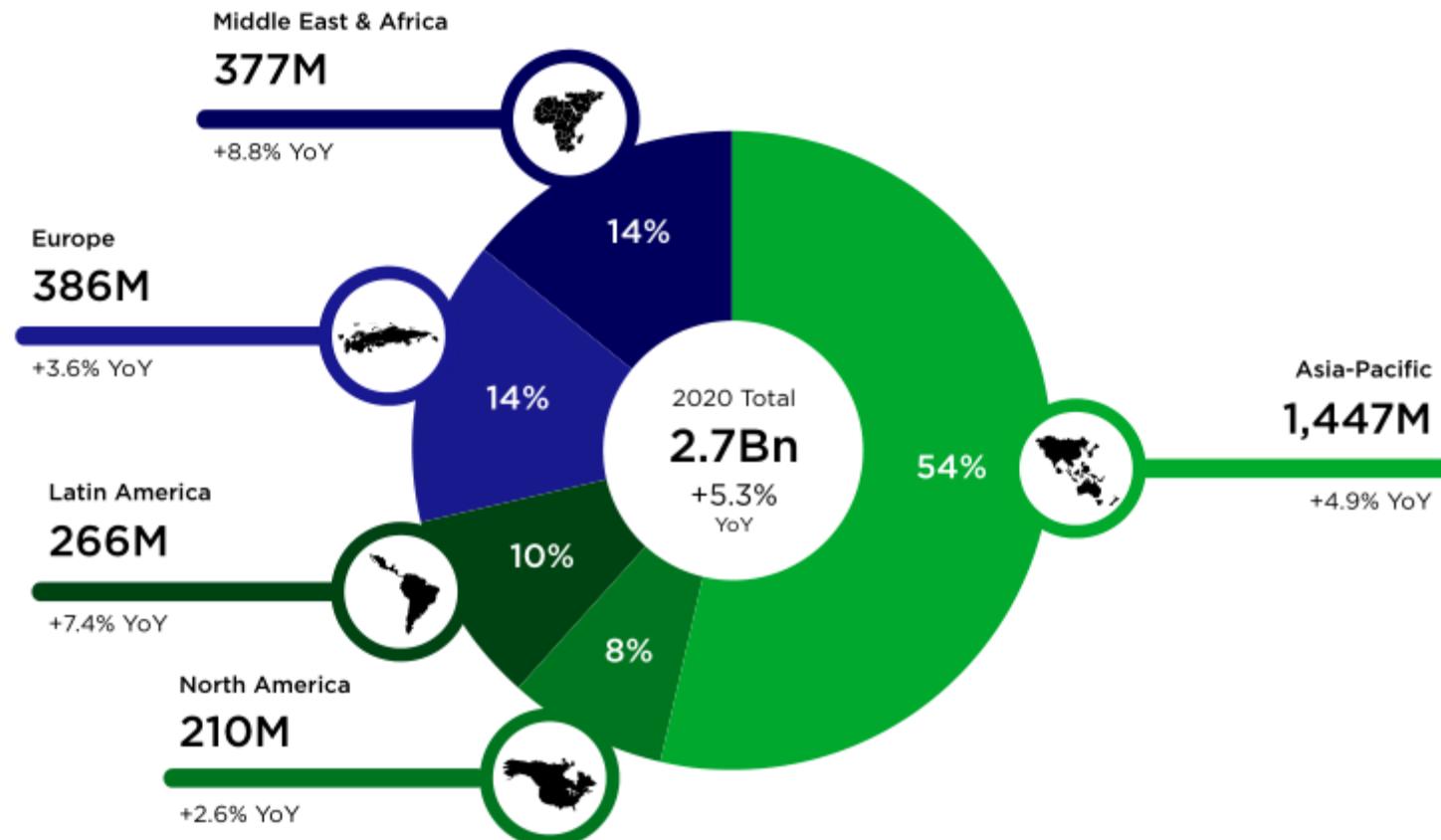


# Tržište igara – procijene vrijednosti danas



# Tko igra igre?

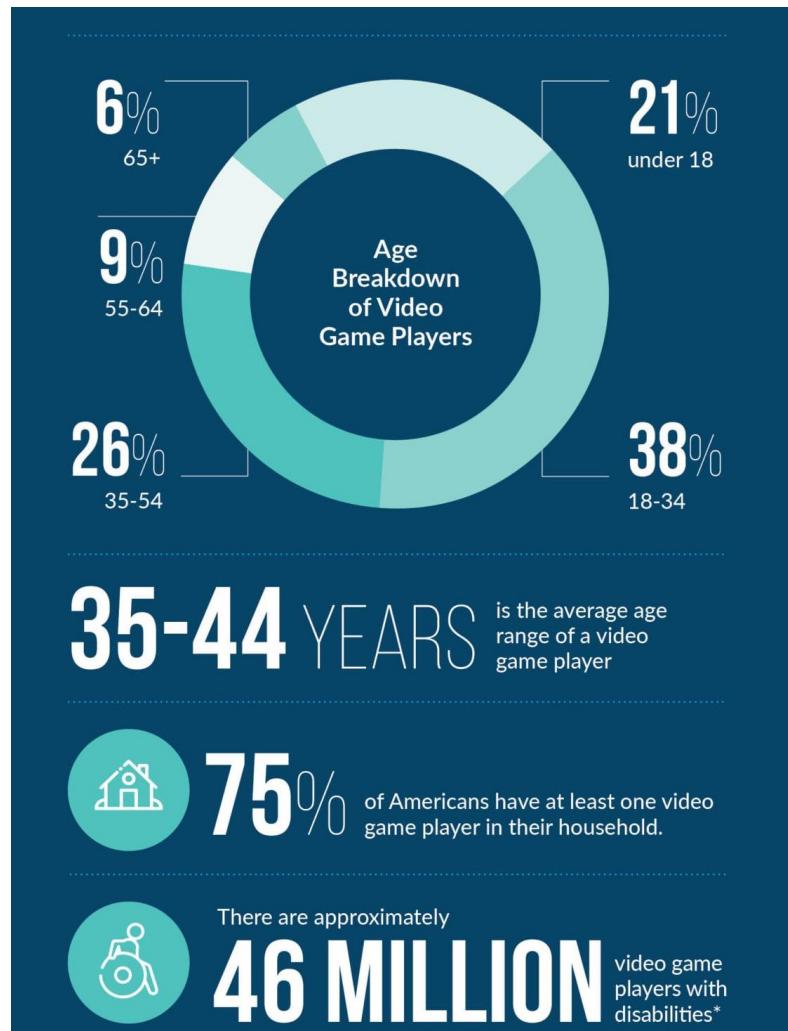
Zavod za telekomunikacije



# Tko igra igre (u SAD-u)?

**214.4 milijuna  
ljudi**

**70%**  
**populacije**  
**ispod 18 godina**



Zavod za telekomunikacije

**64%**  
**populacije**  
**iznad 18 godina**

Izvor: <https://www.theesa.com/esa-research/2020-essential-facts-about-the-video-game-industry/>

# Porast korištenja igara preko Interneta

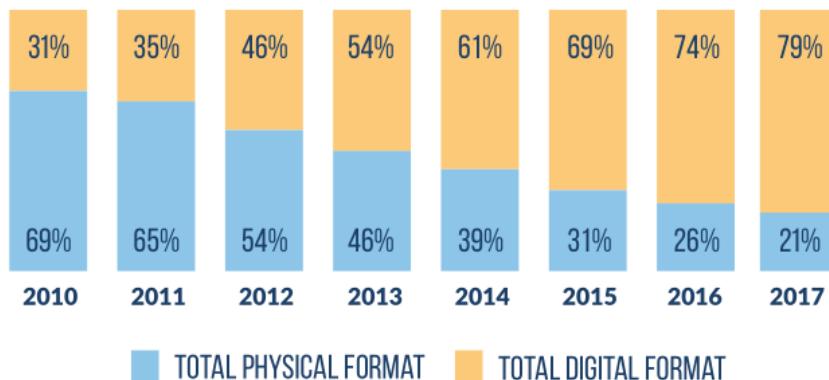
- ◆ Više igrača
- ◆ Zaštita autorskih prava (Digital rights management – DRM)
- ◆ Društvene igre
- ◆ Igre na pokretnim uređajima
- ◆ Distribucija sadržaja
- ◆ Igre na konzolama (XboX one – 300 000 poslužitelja, prethodna generacija XboX live samo 30 000)



Zavod za telekomunikacije

## RECENT DIGITAL\* AND PHYSICAL SALES INFORMATION

\*Digital format sales include subscriptions, digital full games, digital add-on content, mobile apps, and social network games.



### Peak concurrent players

**Fortnite – 12,3 million**

**League of Legends – 8 million**

**Dota 2 – 1,2 million**

**World of Tanks – 1,1 million**

**Steam – 16 (5 u igrama) million**

# Igre na pokretnim uređajima – veliki rast



## 2012-2021 GLOBAL GAMES MARKET

REVENUES PER SEGMENT 2012-2021 WITH COMPOUND ANNUAL GROWTH RATES



Source: ©Newzoo | April 2018 Quarterly Update | Global Games Market Report  
newzoo.com/globalgamesreport



## Top Grossing iPhone - Games

UNITED STATES

DEVICE: iPhone  
COUNTRY: United States  
DATE: Dec 17  
GROUP BY: Day

CAGR  
2012-2021  
**TOTAL +11.0%**

**MOBILE +26.8%**

**PC +3.1%**

**CONSOLE +2.3%**

#	FREE	PAID	GAME	PUBLISHER	PRICE	REVENUE	NEW INSTALLS
1	-	-	Fortnite	Epic Games	Free	\$1,451,543	12,479
2	-	-	Candy Crush Saga	King	Free	\$1,407,996	15,346
3	-	-	Pok��mon GO	Niantic, Inc.	Free	\$1,113,188	16,059
4	-	-	Clash of Clans	Supercell	Free	\$761,353	14,802
5	-	-	Brawl Stars	Supercell	Free	\$738,513	16,462
6	-	-	Toon Blast	Peak Games	Free	\$653,109	14,257
7	-	-	Golf Clash	Playdemic	Free	\$344,148	15,485
8	-	-	ROBLOX	Roblox Corporation	Free	\$317,298	16,232
9	-	-	Slotomania™ Vegas Casino Slots	Playtika Ltd	Free	\$277,483	11,843
10	-	-	MARVEL Contest of Champions	Kabam	Free	\$178,757	12,758

**1,4 miliona \$ DNEVNO –  
samo na iPhoneu u SAD-u**

# Zašto igre na mobitelima idu online?

Zavod za telekomunikacije

- ◆ Najviše zarađuju višekorisničke online igre
- ◆ Sve igre koje najviše zarađuju su višekorisničke

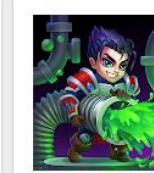
## Najprodavanije igre

Pogledajte više

 MONOPOLY	 Minecraft	 Football Manager 21 MOBILE	 Farming Simulator 2020	 NBA 2K20	 Bloons TD 6	 Don't Starve: Pocket Edition	 The House of Da Vinci	 Monument Valley 2
Monopoly - Board game Marmalade Game Studio	Minecraft	Football Manager 21 SEGA	Farming Simulator GIANTS Software	NBA 2K20 2K, Inc.	Bloons TD 6 ninja kiwi	Don't Starve: Pocket Klei Entertainment Inc.	The House of Da Vinci Blue Brain Games s.r.o.	Monument Valley 2 ustwo games
★★★★★ 33,00 HRK	★★★★★ 59,00 HRK	★★★★★ 79,99 HRK	★★★★★ 55,00 HRK	★★★★★ 49,00 HRK	★★★★★ 40,00 HRK	★★★★★ 33,00 HRK	★★★★★ 42,00 HRK	★★★★★ 43,00 HRK

## Igre s najvećim prihodom

Pogledajte više

 Coin Master Moon Active	 Brawl Stars Supercell	 Roblox	 Empires & Puzzles Small Giant Games	 Gardenscapes Playrix	 PUBG MOBILE - Call of Duty: Warzone	 Hero Wars – Hero Fights	 State of Survival: Survivors	 Homescapes Playrix
★★★★★	★★★★★	★★★★★	★★★★★	★★★★★ 1*	★★★★★	★★★★★	★★★★★	★★★★★ 1*

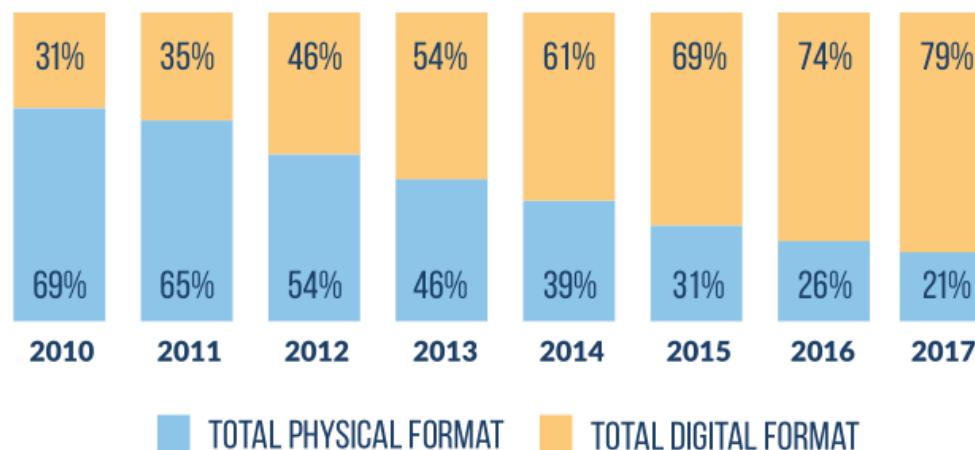
# Distribucija igara

Zavod za telekomunikacije

- ◆ Digitalna distribucija postaje dominantan način prodaje igara
- ◆ Prodaja za pokretne uređaje je kompletno digitalno distribuirana
- ◆ Steam je u veljači 2015 imao 125 milijuna korisnika, 67 milijuna mjesечно aktivnih korisnika, 33 milijuna dnevno aktivnih korisnika i 16 milijuna istovremenih aktivnih korisnika

## RECENT DIGITAL\* AND PHYSICAL SALES INFORMATION

\*Digital format sales include subscriptions, digital full games, digital add-on content, mobile apps, and social network games.



# Tržište – poslovni modeli

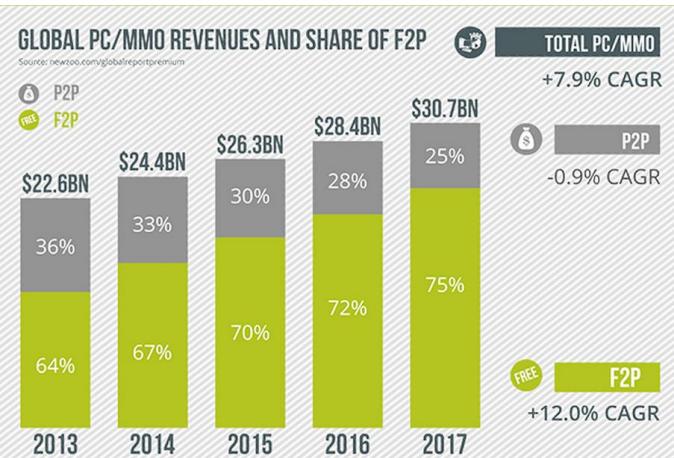
Zavod za telekomunikacije

- ◆ *Pay-to-Play (P2P)*
  - Prodaja igre (Guild Wars)
  - Pretplata (World of Warcraft)
  - Prodaja dodataka na igru
    - Manji – (Downloadable Content - DLC) (Mass Effect 3)
    - Veći – ekspanzije (Age of Conan: Rise of the Godslayer)
- ◆ *Free-to-Play (F2P)*
  - Besplatni dio igre ograničen (Star Wars: The Old Republic)
  - Prodaja virtualnih dobara koji utječu na performanse (Travian)
  - Prodaja kozmetičkih dodataka (Dota 2)
  - “Porez” na trgovinu između igrača (Diablo 3)
- ◆ *Hibridni modeli*
  - Igra se plaća, ali postoji i dodatne mogućnosti unutar igre

# Tržište – udio pojedinog modela

Zavod za telekomunikacije

- ◆ F2P model preuzima tržište
- ◆ Mnoge „premium“ igre imaju F2P komponente (primjerice World of Warcraft)
  - Prvi plaćeni jahaći konj unutar WoW igre je imao cijenu od 25 USD (dvije mjesечne pretplate)
  - Nakon izdavanja se stvorio rep čekanja od 140 000 ljudi koji su htjeli ga kupiti (čekalo se i do 7 sati) – 3,5 milijuna USD



FREE-TO-PLAY		PREMIUM	
TITLE	REVENUE	TITLE	REVENUE
LEAGUE OF LEGENDS	\$2.1B	PUBG	\$714M
DUNGEON FIGHTER ONLINE	\$1.6B	OVERWATCH	\$382M
CROSSFIRE	\$1.4B	CS:GO	\$341M
WORLD OF TANKS	\$471M	DESTINY 2	\$218M
DOTA 2	\$406M	GRAND THEFT AUTO V	\$118M
ROBLOX	\$310M	BATTLEFIELD 1	\$113M
MAPLESTORY	\$279M	MINECRAFT	\$92M
HEARTHSTONE	\$217M	GUILD WARS 2	\$87M
BLADE & SOUL	\$178M	DIVINITY: ORIGINAL SIN 2	\$85M
FIFA ONLINE 3	\$163M	RAINBOW SIX: SIEGE	\$67M
<b>TOTAL:</b>	<b>\$7.1B</b>	<b>TOTAL:</b>	<b>\$2.2B</b>



Izvori: SuperData research <https://www.superdataresearch.com/market-data/market-brief-year-in-review/>  
<https://newzoo.com/globalgamesreport>

# Funkcijska arhitektura

# Arhitektura umreženih igara

Zavod za telekomunikacije

- ◆ Peer-to-Peer (P2P)
  - Dobra skalabilnost
  - Nedostaci: loša kontrola varanja, distribucija stanja virtualnog svijeta, problem konzistencije
  - Danas se rijetko koristi “samo P2P” (*Demigod*)
- ◆ Klijent – poslužitelj
  - Klasičan pristup
  - Dobra kontrola
  - Usko grlo – sve ide kroz poslužitelj
    - Više poslužitelja na strani proizvođača (World of Warcraft)
    - Farme poslužitelja na strani proizvođača (EvE online)
    - Klijent postaje poslužitelj (Warcraft III)
    - Namjenski (engl. dedicated) poslužitelji koje mogu održavati igrači (Call of Duty)

# Demigod – zašto je P2P loš u praksi

Zavod za telekomunikacije



- ◆ Prva igra nastala na konceptu DOTA-e
- ◆ Potpuna P2P arhitektura
- ◆ Dosta dobre ocjene kritičara
- ◆ Inicijalno privučeno puno igrača iako je igra naplaćivana
- ◆ Tehnički problemi
  - Problemi s Network Address Translator (NAT) poslužiteljima – poslužitelji za probijanje NAT-a nisu mogli obraditi veliki broj zahtjeva (veliko kašnjenje u obradi zahtjeva)
  - Veliki broj korisnika koji su bili ilegalni (18 000 regularnih i 100 000 ilegalnih)
  - Veliki broj korisnika su bili od spojeni od svojih mečeva
  - Vrlo loša iskustvena kvaliteta – katastrofalno lansiranje igre što je dugoročno upropastilo

# Raspored funkcija unutar igre na klijenta i poslužitelja

- ◆ Osnovne funkcije
  - Unos komandi (uvijek na klijentu)
  - Izračun stanja virtualnog svijeta
  - IsCRTavanje stanja virtualnog svijeta (renderiranje virtualne scene)
- ◆ Sve funkcije na klijentu (nema poslužitelja)
- ◆ IsCRTavanje stanja virtualnog svijeta na klijentu, logika izračuna stanja virtualnog svijeta na poslužitelju
  - Tradicionalni način rasporeda funkcija
  - Poslužitelj šalje osvježenja stanja
- ◆ Logika izračuna stanja virtualnog svijeta te isCRTavanje virtualnog svijeta na poslužitelju – igre u oblaku (engl. cloud gaming)
  - Poslužitelj šalje video strujanje (dodatna funkcija kodiranja videa na poslužitelju)
  - Visoki zahtjevi na propusnost mreže
- ◆ Hibridni – dio na klijentu, a dio funkcija na poslužitelju

# Izračun stanja virtualnog svijeta

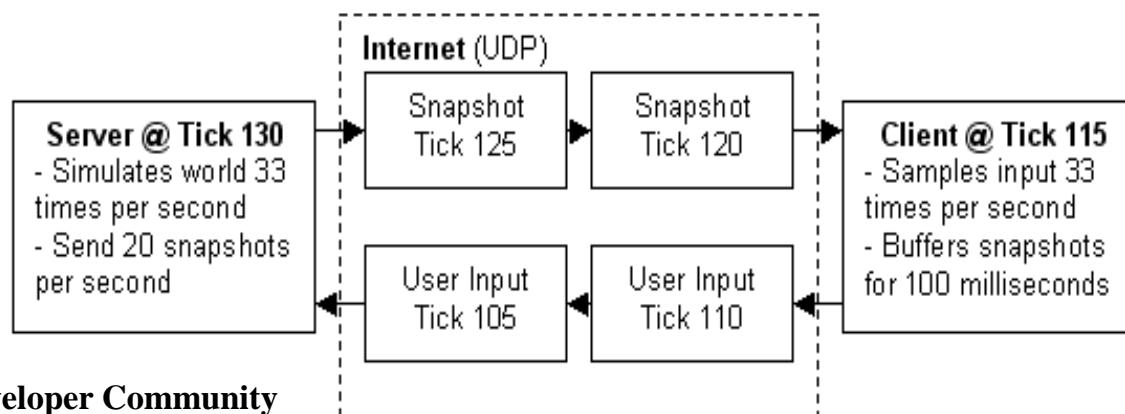
Zavod za telekomunikacije

- ◆ Svakih period  $t$  server izračunava novo stanje svijeta na temelju primljenih unosa klijenata
- ◆ Taj period definira broj otkucaja (engl. tickrate) – koliko puta u sekundi se treba izračunati stanje svijeta – mjeri se u Hz
- ◆ Brzina izračuna stanja na poslužitelju ne mora biti ista kao i broj poslanih osvježenja – primjerice ako se stanje nije promijenilo ne mora se slati osvježenje
- ◆ Igre interpoliraju pozicije i stanje virtualnog svijeta između dva osvježenja stanja radi fluidnog prikaza, ali na razini poslužitelja postoje samo stanja izračunata svakog period  $t$
- ◆ Brzina prikaza igre na računalu je izražena u broju sličica u sekundi (engl. framerate)

# Mrežno funkcioniranje Source pokretačkog sustava (tradicionalni koncept)

Zavod za telekomunikacije

- ◆ Source pokretački sustav (engl. game engine) pokreće igre poput Counter Strike Source (CSS), Team Fortress 2 (TF2), Left 4 Dead (L4D) itd.
- ◆ Simulacija se izvršava na poslužitelju.
  - Tickrate 66 za CSS, a 30 za TF2 i L4D – ovisno i interaktivnosti igre
  - Poslužitelj šalje razlike između pojedinih snimaka, a ne cijelo stanje
- ◆ Klijent određuje mrežne parametre
  - Dojavljuje mrežnu propusnost – dizajnirano za propusnost od 40-60 Kb/s
  - Dojavljuje koliki broj osvježenja želi po sekundi (20 je predefinirani broj)
  - Klijent ne šalje zasebno svaku komandu već ih šalje određenom brzinom (30 puta u sekundi je predefiniran broj) te je obično više od jedne naredbe u jednom paketu
  - Klijent može zahtijevati kompletну snimku stanja

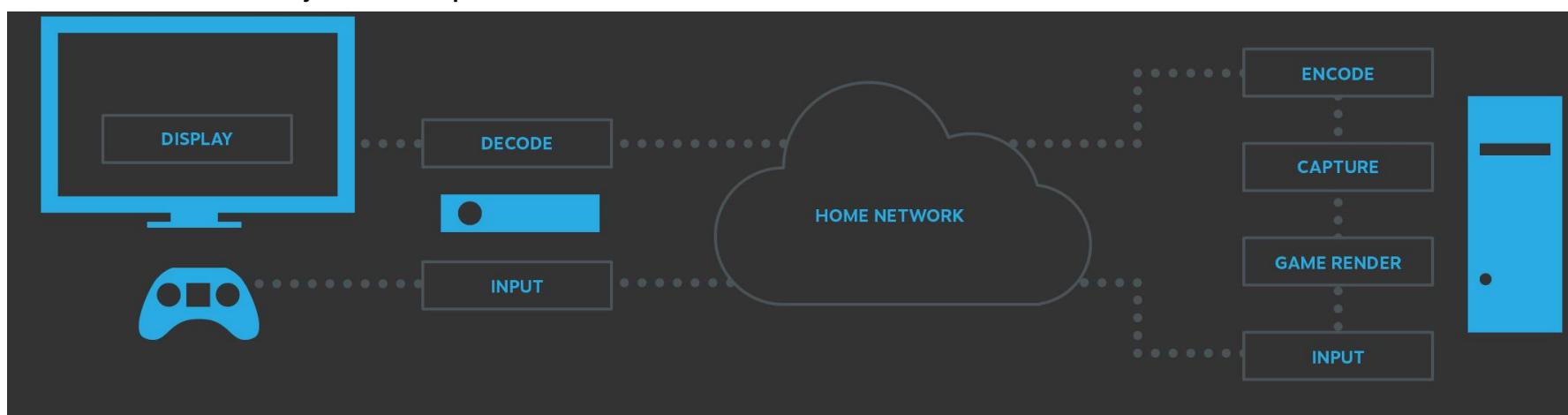


Izvor: Valve Developer Community

# Koncept igranja u oblaku

Zavod za telekomunikacije

- ◆ Primjer na platformi Steam In-Home Streaming
- ◆ “Oblak” je snažno kućno računalo
- ◆ Osnovne funkcije:
  - Unos komandi
  - Računanje stanja igre
  - Renderiranje igre
  - Snimanje i kodiranje videa
  - Dekodiranje videa i prikaz



Izvor: [https://support.steampowered.com/kb\\_article.php?ref=3629-RIAV-1617](https://support.steampowered.com/kb_article.php?ref=3629-RIAV-1617)

# Karakteristike mrežnog prometa

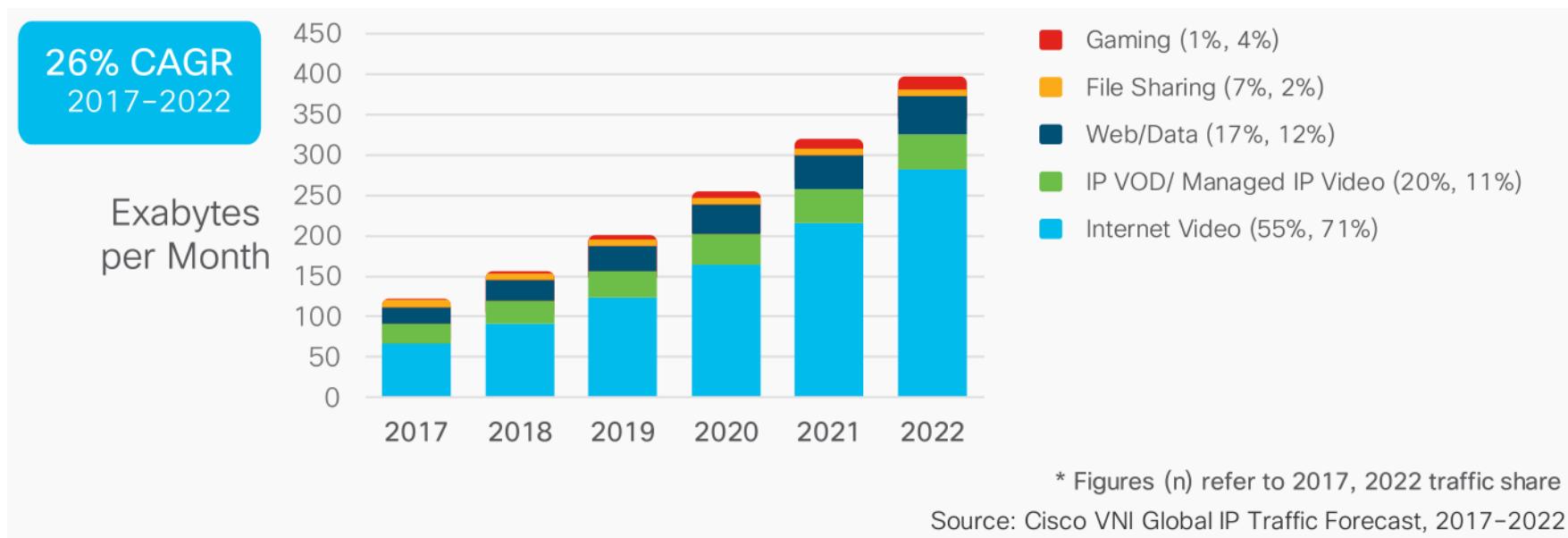
# Tipovi informacija u mrežnom prometu

Zavod za telekomunikacije

- ◆ Koje informacije prenosi mrežni promet igara?
  - Izlazni promet na klijentu - naredbe igrača
  - Izlazni promet na poslužitelju - osvježenja stanja virtualnog svijeta
  - Tekstualna komunikacija
  - Audio komunikacija među igračima
    - Neke igre imaju ugrađene sustave za VoIP komunikaciju
    - Mnogi igrači koriste i posebne aplikacije (TeamSpeak, Discord, Skype...)
  - Igre u pravilu NE prenose 3D podatke – to je karakteristično za umrežene virtualne svjetove (npr. Second Life), informacije o 3D objektima su pohranjene na klijentima
  - Video – igre u oblaku
    - Strujanje igračih sjednica ([www.twitch.tv](http://www.twitch.tv))
- ◆ Karakteristike prometa ovise o funkcionalnoj arhitekturi

# Globalni trendovi vezani za promet igara

- ♦ Promet igara je mali dio sveukupnog prometa (1%), dok mu je stopa rasta 4%

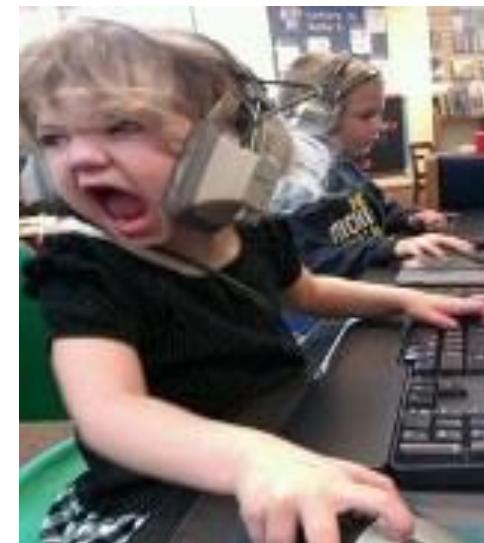


# Karakteristike prometa

- ◆ Mrežni tokovi igara:
  - Dugo trajanje
  - Visoka brzina paketa
  - Mala veličina paketa
  - Niski zahtjevi na mrežnu propusnost
  - Korištenje i UDP-a i TCP-a
  - **Karakteristike jako ovise o tipu igre**
- ◆ Zahtjevi :
  - Osjetljivi na kvalitetu mreže
  - Različita zahtijevana pouzdanost
- ◆ Izuzetak su igre u oblaku

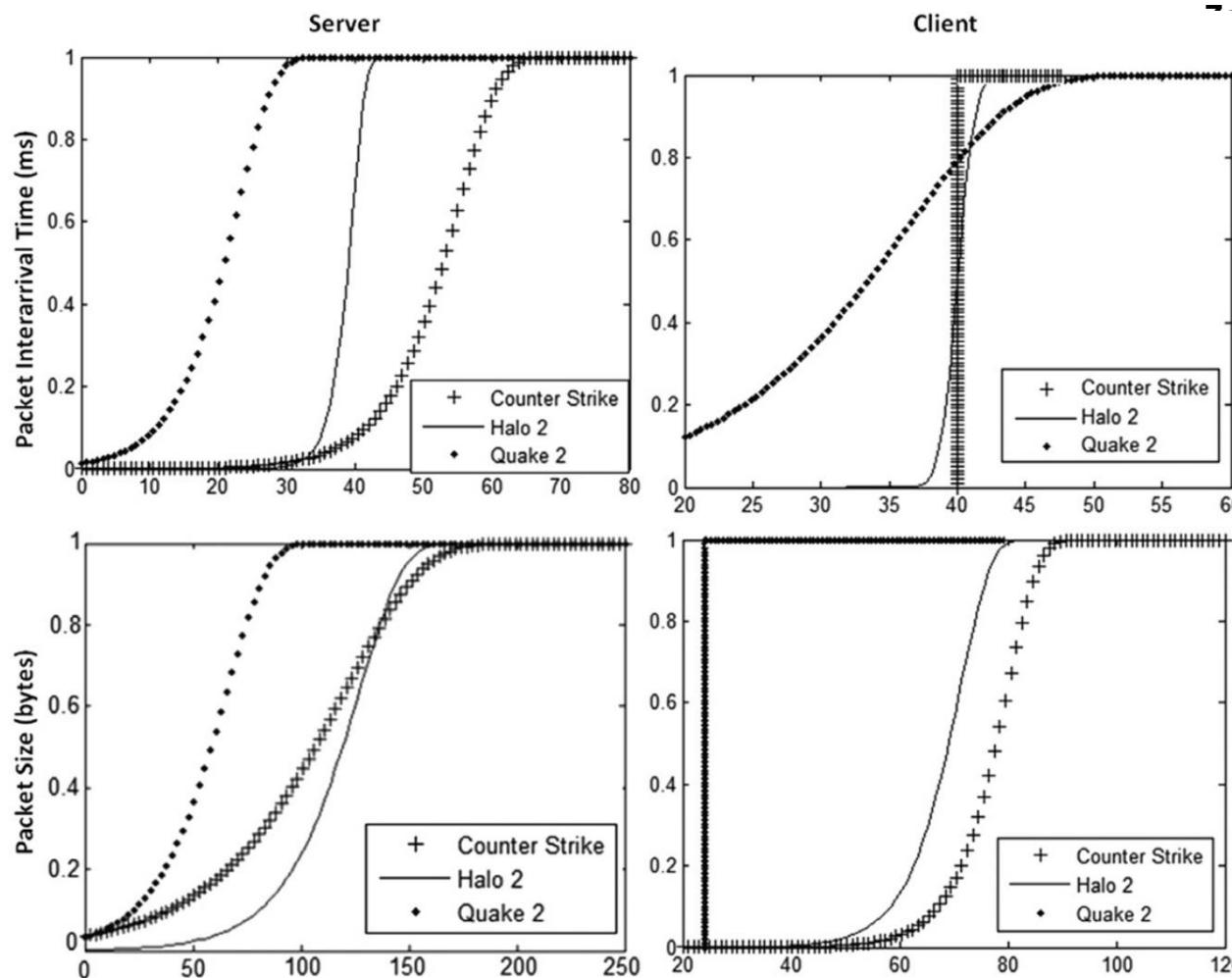
*120 hours of World of Warcraft*

by Elizabeth Harper Jul 24th 2007 at 8:10PM

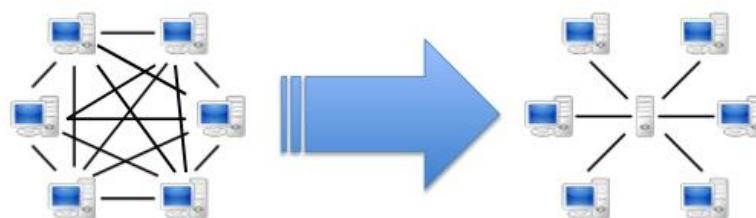


# Karakteristike FPS igara

za telekomunikacije



# Evolucija mrežnog prometa? – Ne baš...



## StarCraft I (1998-2010)

**1 - 5 kbps**

**(2-8 igrača)**



## StarCraft II (2010-present)

**2 - 3 kbps**

**(neovisno o broju  
igrača)**

M. Claypool, D. LaPoint, and J. Winslow, “Network Analysis of Counter-strike and Starcraft,” in Proceedings of the 22nd IEEE International Performance, Computing, and Communications Conference (IPCCC), USA, April 2003.

C-S. Lee, “The Revolution of StarCraft Network Traffic” in Proceedings of the 11th Annual Workshop on Network and Systems Support for Games NetGames 2012

# Revolucija mrežnog prometa igara? Da\*



- ◆ Igre u oblaku
- ◆ Fundamentalno različite karakteristike prometa
  - Vrlo visoki zahtjevi na propusnost veze
  - Viši zahtjevi na mrežno kašnjenje nego "standardne" igre
- ◆ \*Bez većeg komercijalnog uspjeha
- ◆ Ako igranje u oblaku zaživi promet igara će postati jedna od najznačajnijih kategorija prometa u Interneta



Zavod za telekomunikacije

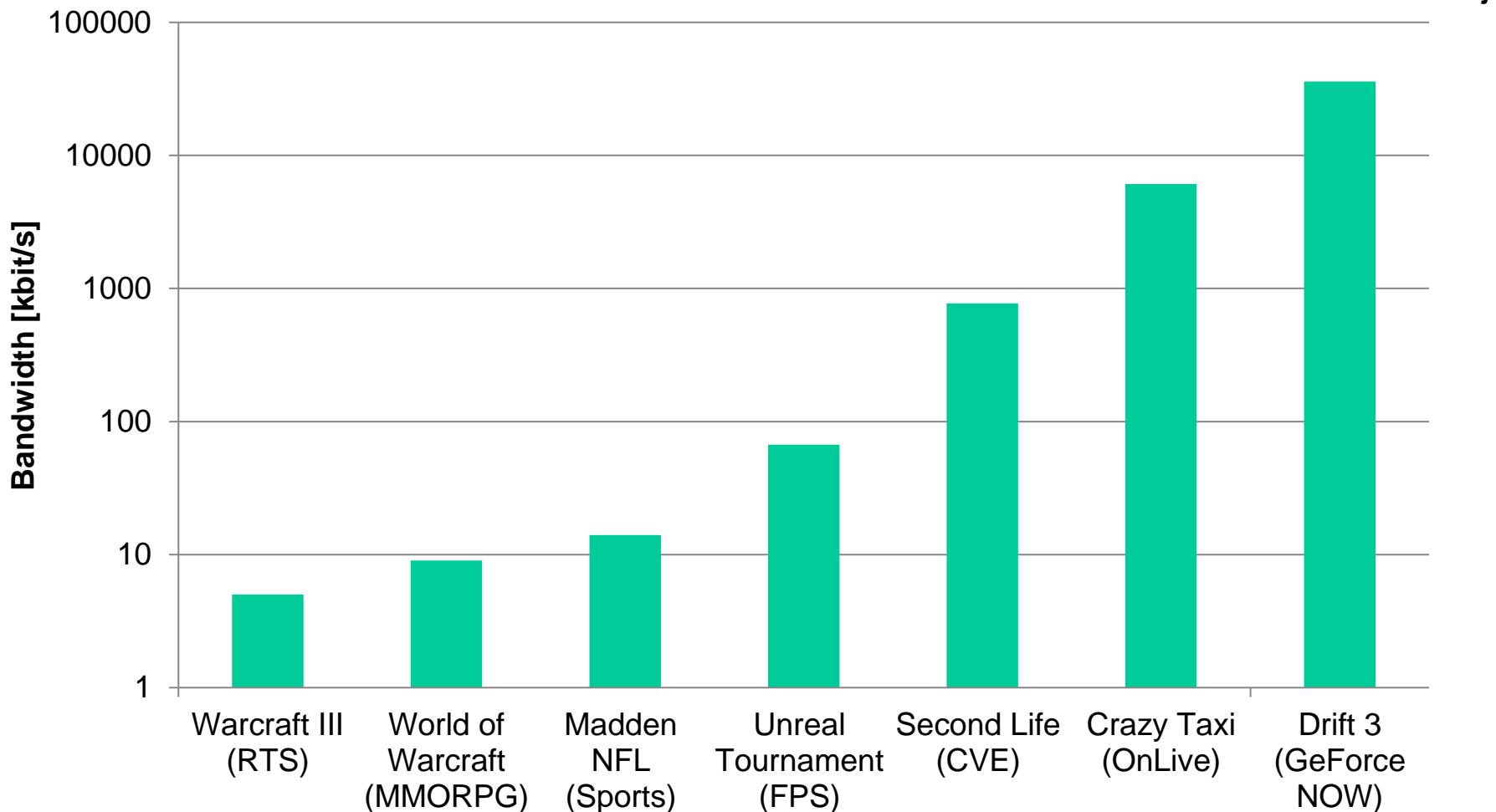


RTP/UDP flows of the OnLive Streaming Protocol

Direction	RTP SSRC	RTP Payload Type	Flow description
Downstream	0x00000000	100	QoS monitoring flow
Downstream	0x00010000	100	OnLive Control
Downstream	0x00030000	100	Audio stream (CBR Codec)
Downstream	0x00040000	100	Cursor position
Downstream	0x00050000	101	Audio stream (VBR Codec)
Downstream	0x00060000	96	Video stream
Downstream	0x00080000	100	Voice Chat (Sound from other players)
Upstream	0x0000XXXX	100	User input (keyboard and mouse buttons)
Upstream	0x0001XXXX	100	Cursor movement
Upstream	0x0004XXXX	100	OnLive Control ACK
Upstream	0x0008XXXX	100	Voice Chat (Microphone from the user)

# Mrežna propusnost – razlika među tipovima igara

Zavod za telekomunikacije

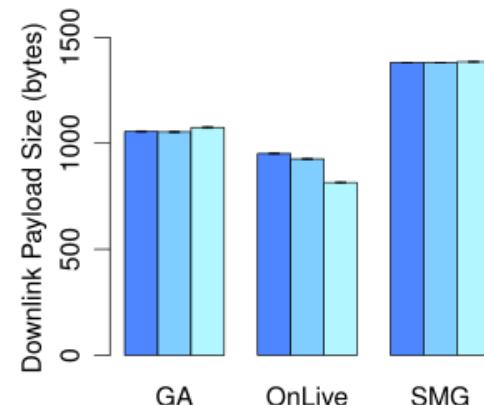
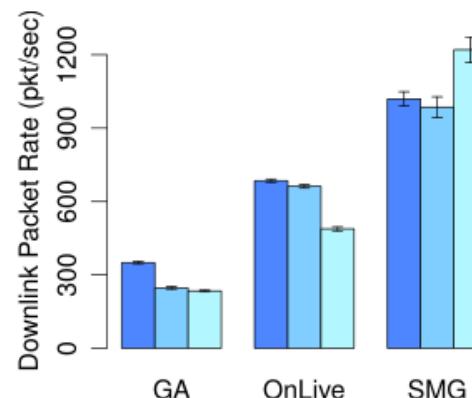
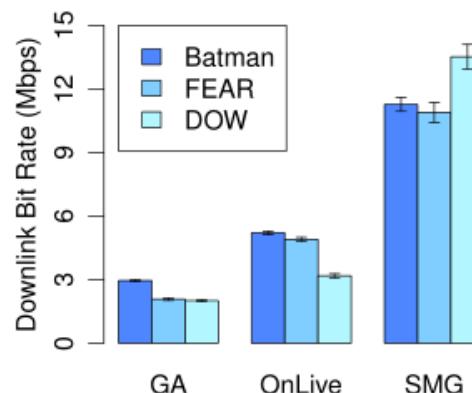


# Razlike u propusnosti kod igara u oblaku

Razlika zbog tipa igre – različite karakteristike videa (prostorne i vremenske)

		<i>pes2012</i>	<i>unreal3</i>	<i>crazytaxi</i>	<i>aircombat</i>	<i>4elements</i>
Downstream	Total time (s)	249.41	261.16	200.56	239.68	236.59
	Number of packets	149004	174265	13867	153798	108619
	Avg. packets / sec	597.41	667.25	691.39	641.66	459.09
	Avg. packet size (B)	915.57	975.05	1014.09	955.65	722.58
	Bit rate (Mbps)	4.37	5.21	5.61	4.91	2.65
Upstream	Total time (s)	249.48	261.31	200.69	239.83	236.72
	Number of packets	8947	13943	6825	14677	14849
	Avg. packets / sec	35.86	53.35	34.01	61.19	62.72
	Avg. packet size (B)	168.49	157.8	170.08	154.81	154.91
	Bit rate (Mbps)	0.0048	0.067	0.046	0.075	0.077

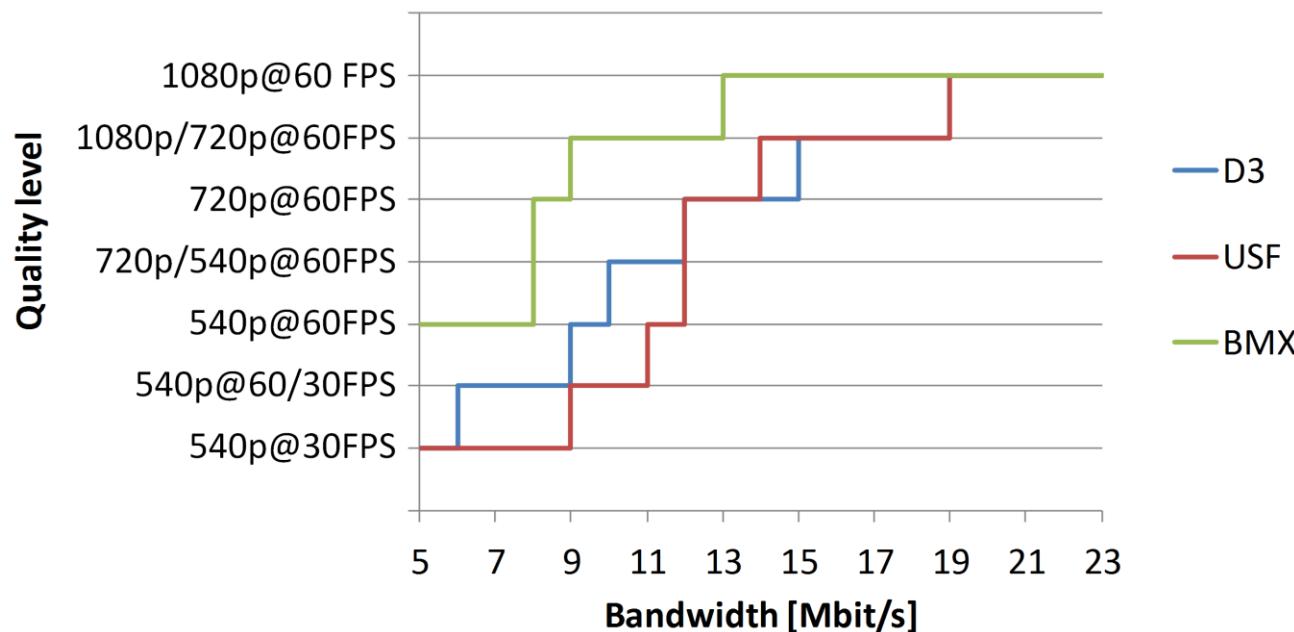
Razlika u implementaciji platforme



# Razlike zbog prilagodljivog strujanja

Zavod za telekomunikacije

- ◆ Razlike po platformama
  - Steam InHome Streaming – ne mijenja se rezolucija ni brzina osvježavanja ekrana s padom propusnosti
  - GeForce NOW – traži 30 Mbit/s za punu kvalitetu, ali implementira adaptacijski algoritam koji snižava rezoluciju pa potom frame rate na 30 FPS

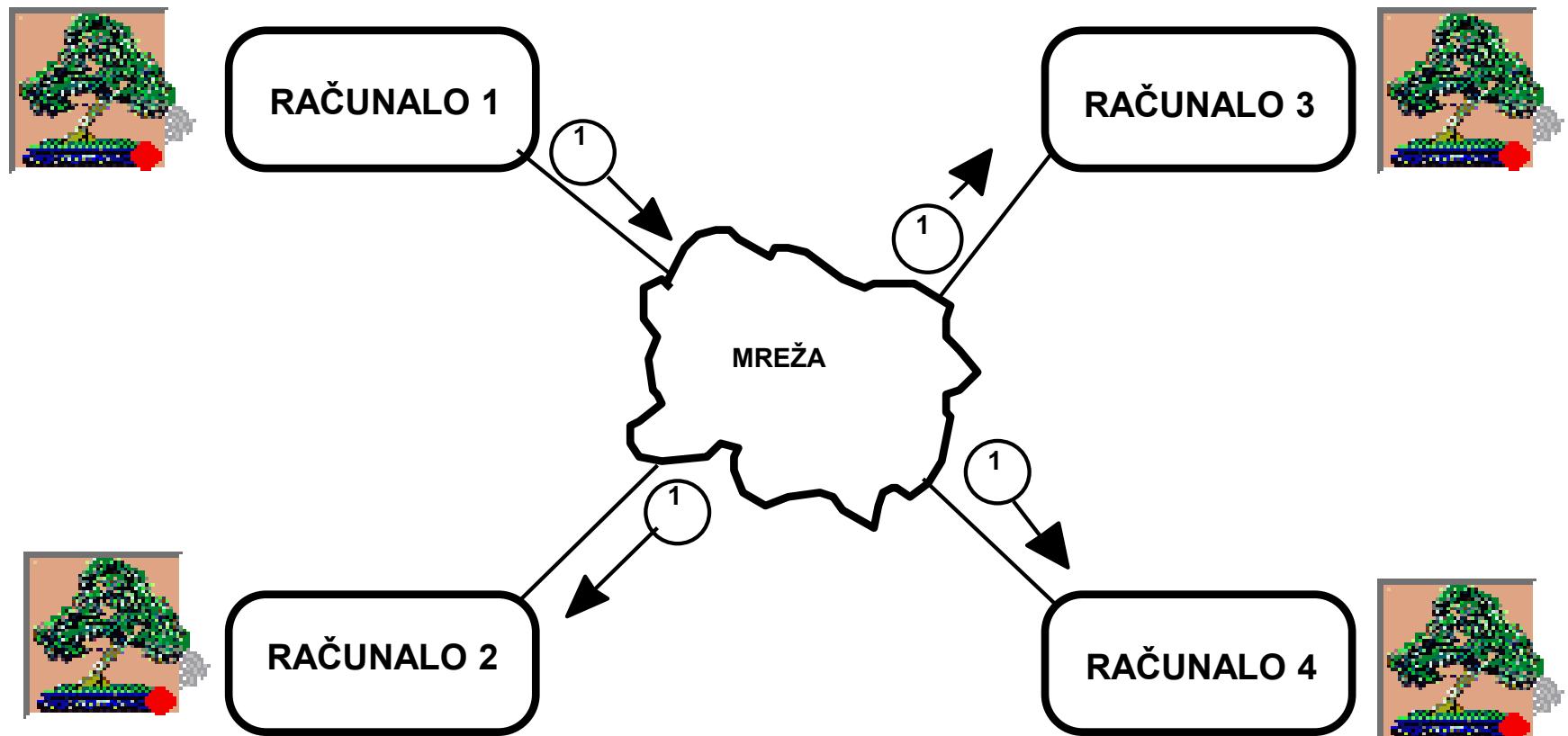


# Utjecaj mrežnih degradacija

# Mrežno kašnjenje

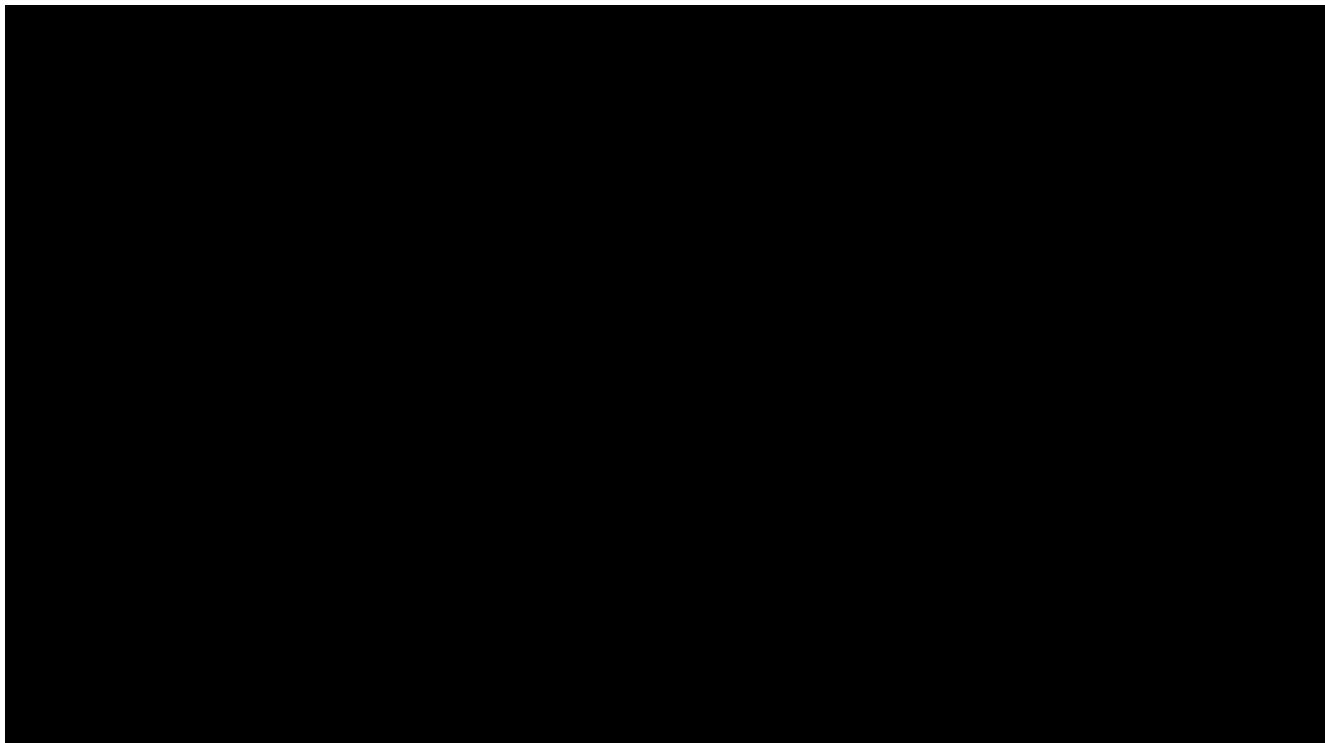
- ◆ Osnovne komponente mrežnog kašnjenja
  - Propagacijsko – ograničenje: brzina svjetlosti
  - Procesorsko – kašnjenje zbog obrade u usmjeriteljima
  - Transmisijsko – kašnjenje uzrokovano slanjem podataka na prijenosni medij
  - Kašnjenje u redovima čekanja – kašnjenje uzrokovano čekanjem u međuspremnicima
- ◆ Kašnjenje je UVIJEK prisutno! (kao i razlika u kašnjenju pojedinih igrača)
- ◆ Mrežno kašnjenje je jedan od najvažnijih parametara kvalitete umreženih igara – potrebno je održati iluziju da virtualni svijet nije distribuiran u mreži

# Kako radi umreženo virtualno okruženje



# Rezultat – paralelni svjetovi pomaknuti u vremenu

- ◆ Primjer FPS sa 4 igrača (jedan služi kao poslužitelj, a ostali imaju 50, 100, i 150 ms kašnjenja)



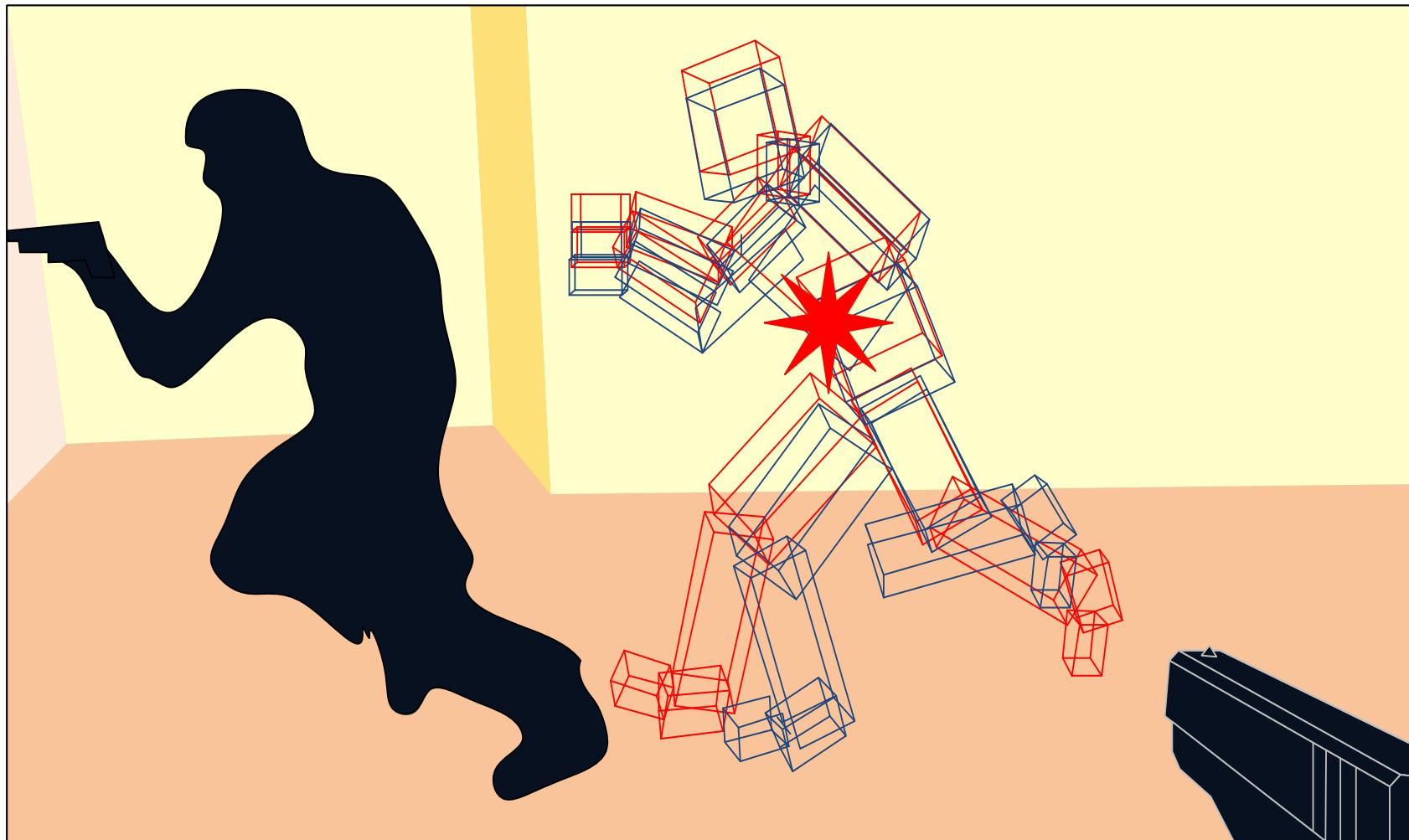
Video URL: <https://www.youtube.com/watch?v=xyCQtUFOJmA#t=728>

# Mehanizmi za borbu protiv kašnjenja

Zavod za telekomunikacije

- ◆ Predikcija na strani klijenta
  - Predikcija kretanja samog igrača - primjerice izvršavanje komande prije nego što ju je poslužitelj autorizirao
  - Predikcija kretanja drugih entiteta (dead reckoning) - na temelju dosadašnjeg vektora kretanja izračunava se novo stanje iako osvježenje stanja nije došlo na vrijeme
- ◆ Mehanizmi na strani poslužitelja
  - Veća geografska granulacija poslužitelja (što manje propagacijsko kašnjenje)
  - Vremenski odmak u izračunu stanja nakon dolaska komandi kako bi se kompenziralo klijente s većim kašnjenjem
  - “Premotavanje vremena” – poslužitelj čuva prošla stanja te za pojedinog klijenta premotava vrijeme kako bi znao koje točno stanje klijent vidi u pojedinom trenutku

# Kompenzacija kašnjenja na strani poslužitelja – premotavanje vremena



# Skalabilnost

- ◆ Skalabilnost – koliko neki sustav može rasti, a da se pri tom ne naruši njegova funkcija
- ◆ Primjer - **Massively** Multiplayer Online Role-Playing Games
  - Veliki broj korisnika koji dijele jedan virtualni svijet
  - WoW – 12 miliona korisnika (na vrhuncu popularnosti)
- ◆ Glavni problemi skalabilnosti:
  - Izračunavanje stanja virtualnog svijeta
  - Održavanje konzistentnosti stanja virtualnog svijeta
  - Kontrola varanja
  - Svi poslužitelji na strani proizvođača
- ◆ Dva arhitekturna rješenja
  - Velike poslužiteljske farme
  - “Komadanje” virtualnog svijeta (engl. žargon: *server “shards”*)

# Koliko masivno?

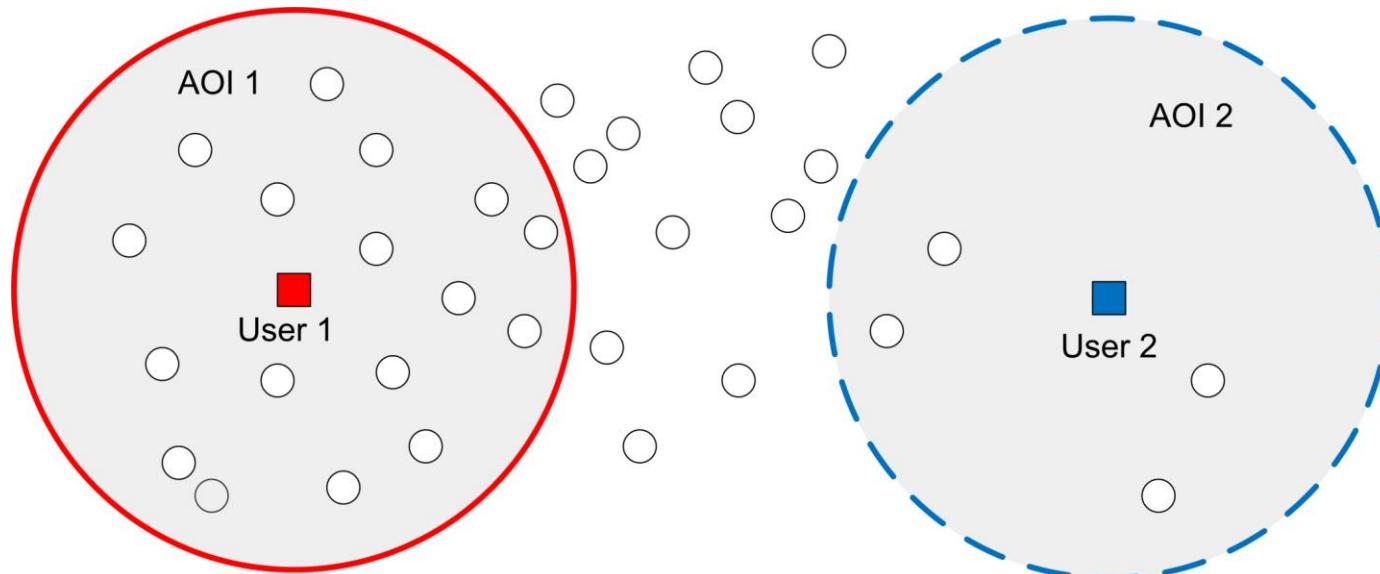
Zavod za telekomunikacije

- ◆ World of Warcraft-a (2009.):
  - 13 250 poslužitelja
  - 75 000 CPU jezgri
  - 11,5 TB RAM
  - 11 podatkovnih centara po cijelom svijetu (USA, Europe, China, Australia...)
- ◆ EvE online (2013.):
  - Supercomputer “Tranquility”, u Londonu, UK
  - 3 936 GB RAM
  - 2 574 GHz CPU snage

# Filtriranje prema području interesa

Zavod za telekomunikacije

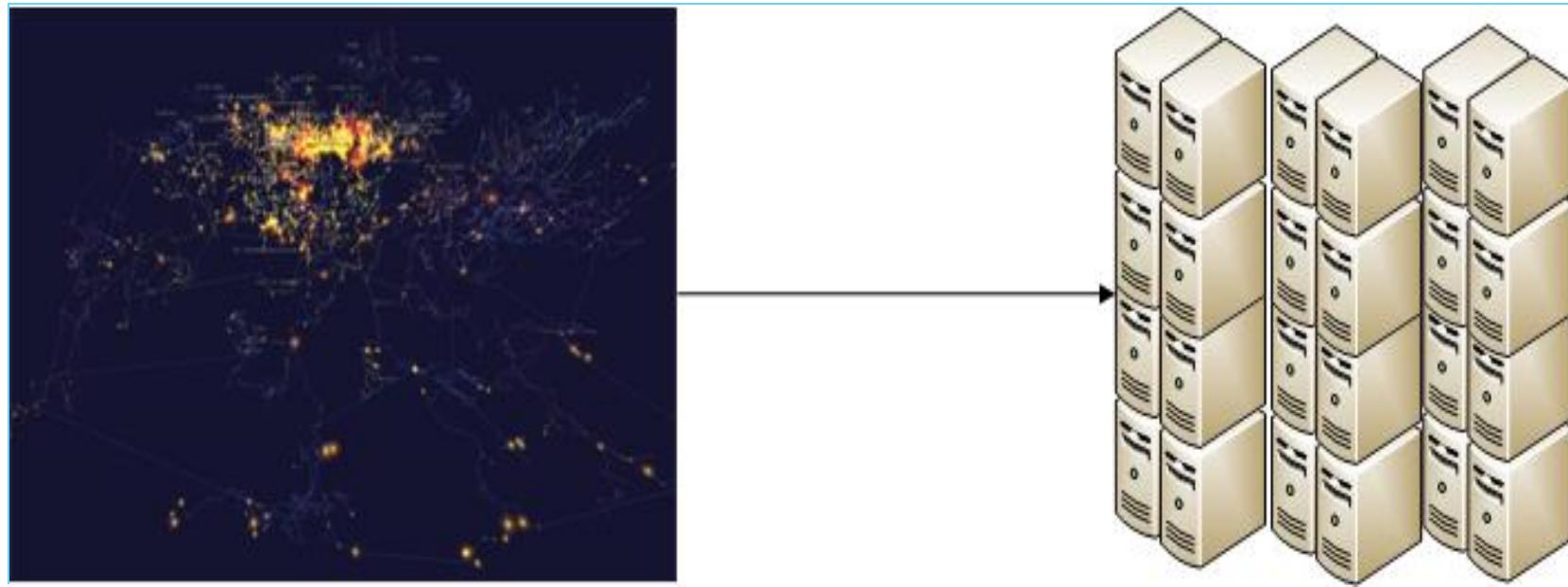
- ◆ engl. Area Of Interest Management, AOIM
- ◆ Prosljeđuju se samo relevantne poruke
- ◆ Umjesto eksponencijalnog rasta prometa s brojem korisnika, linearan rast: ključan preduvjet za postojanje modernih UVO s desecima tisuća istovremenih korisnika



# Farma poslužitelja (*grozd*)

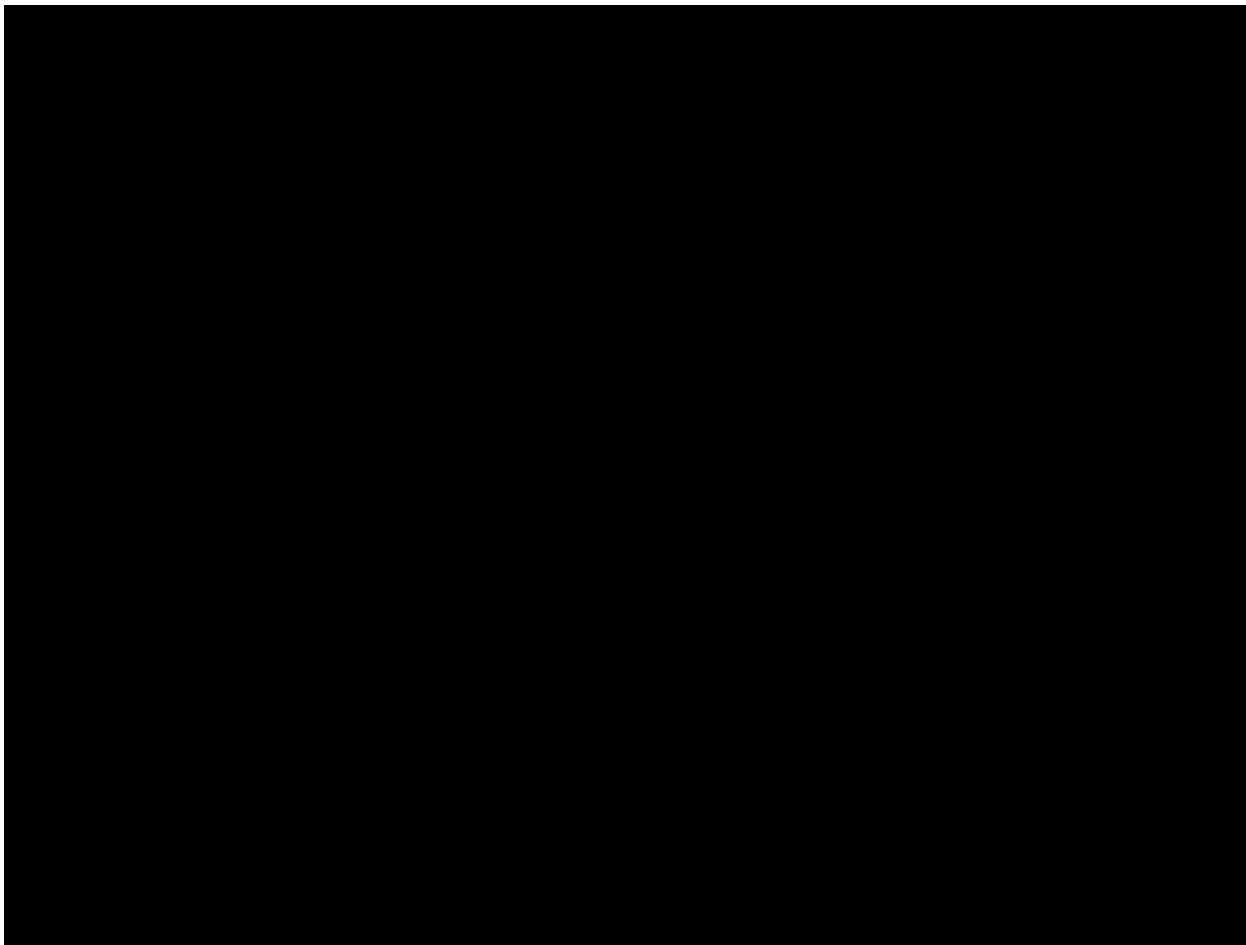
Zavod za telekomunikacije

- ◆ Svi korisnici unutar istog virtualnog svijeta (EvE online, World of Tanks)
- ◆ Velike farme poslužitelja
- ◆ Problemi s izračunom stanja virtualnog svijeta (moguć veliki broj korisnika na jednom mjestu)



# The Asakai incident

Zavod za telekomunikacije

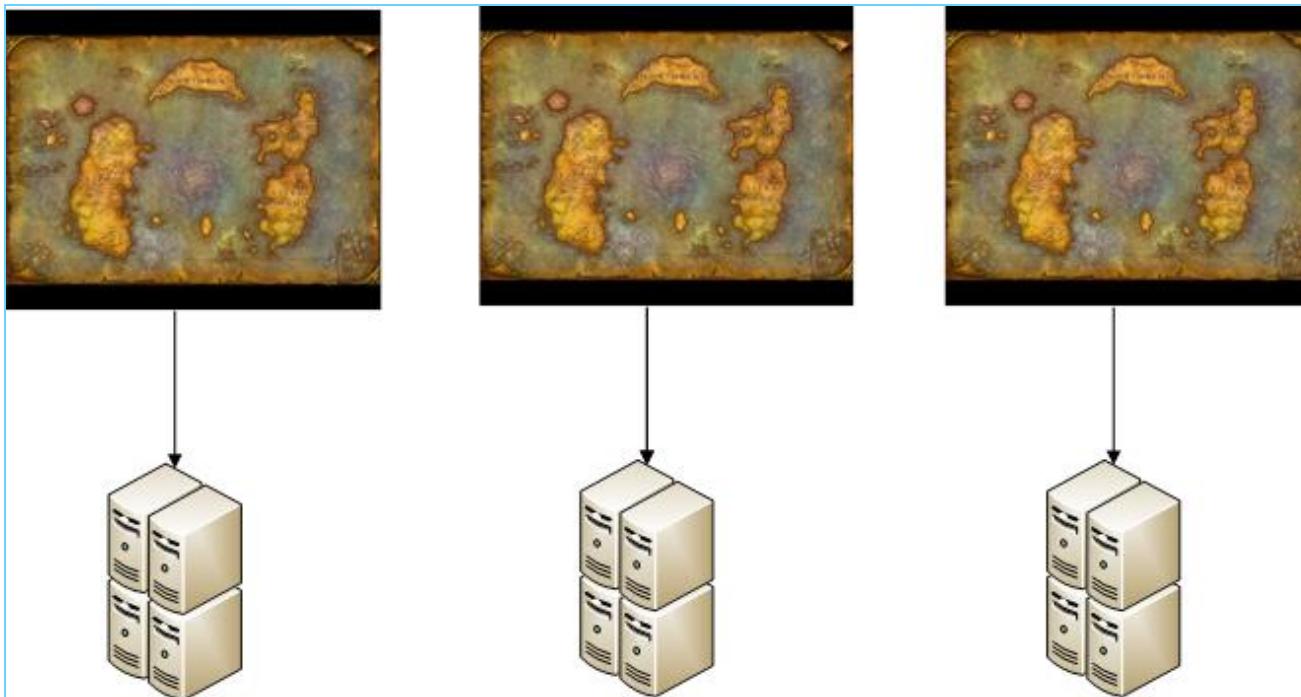


**Video URL:** [https://www.youtube.com/watch?v=\\_iQw3YcLoQU](https://www.youtube.com/watch?v=_iQw3YcLoQU)

# Komadanje svijeta

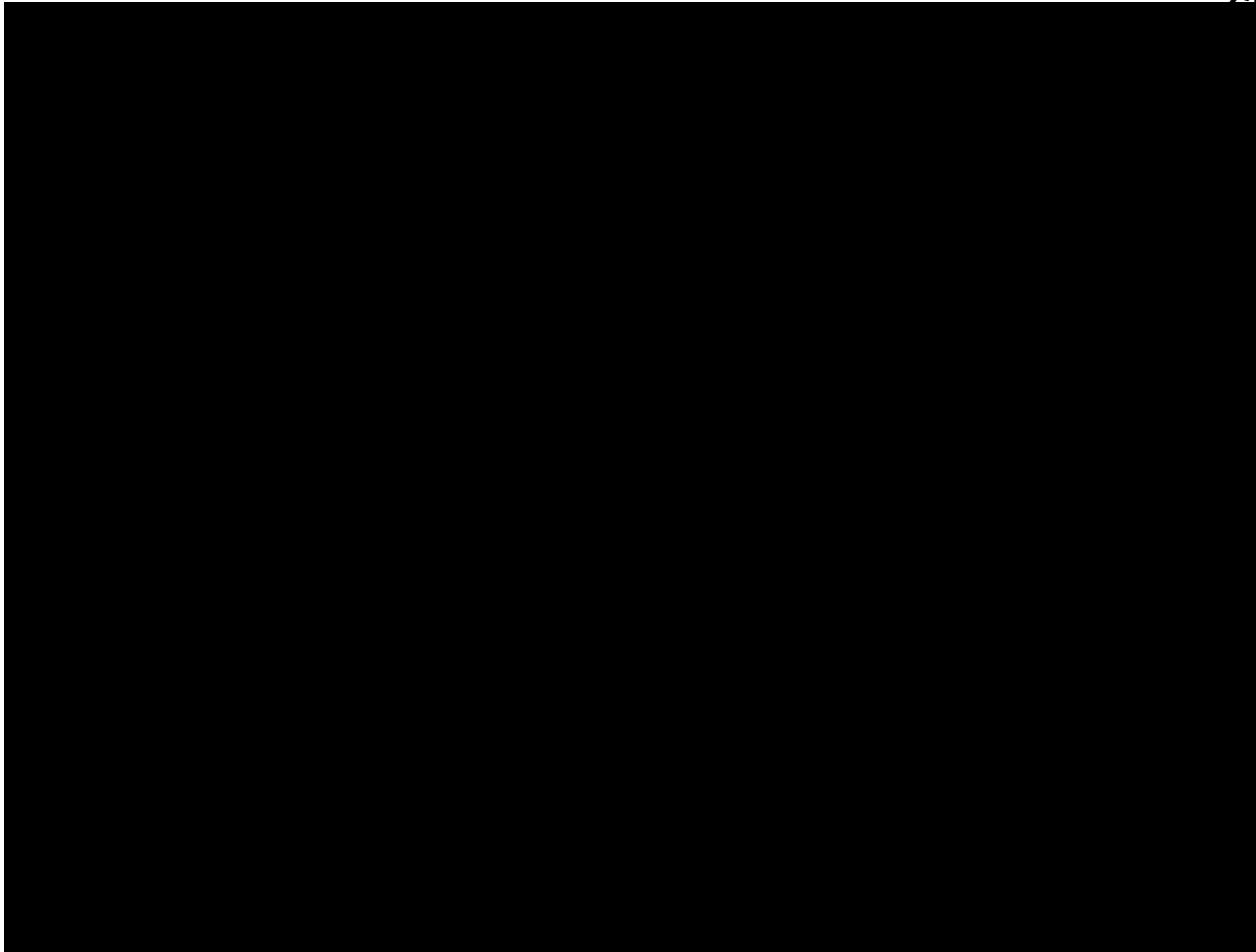
Zavod za telekomunikacije

- ♦ Repliciran cijeli virtualni svijet na svakom “komadu” (engl. shard)
- ♦ Razdvojeni korisnici
- ♦ Smanjenje skale (sa 10 miliona na nekoliko tisuća)
- ♦ Osjetljivost na velika okupljanja igrača ostaje



# War without the warchief

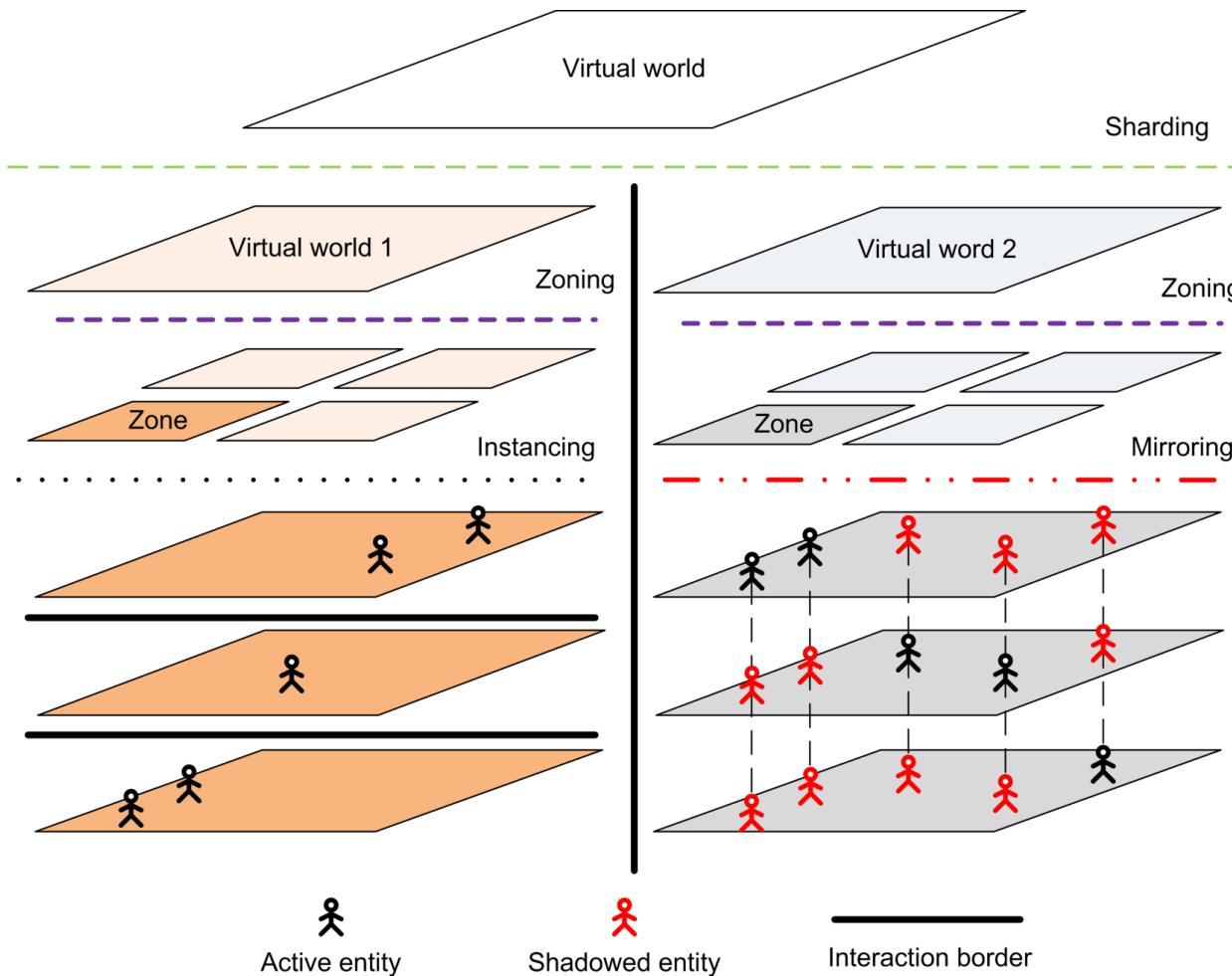
Zavod za telekomunikacije



**Video URL:** <https://www.youtube.com/watch?v=ZzsIiSTnQfI>

# Tehnike za skalabilnost

Zavod za telekomunikacije



# Tehnike skalabilnosti

Zavod za telekomunikacije

- ◆ **Zoniranje** – podjela virtualnog svijeta na geografska područja koja će se samostalno obraditi odvojenim strojevima
- ◆ **Zrcaljenje** – način distribucije opterećenja umnožavanjem iste zone igre na nekoliko poslužitelja. Svaki replicirani poslužitelj izračunava stanje za podskup entiteta koji se naziva aktivni entiteti, dok se preostali, koji se nazivaju entitetima sjene (koji su aktivni u drugim poslužiteljima koji sudjeluju), sinkroniziraju preko poslužitelja
- ◆ **Instanciranje** – pojednostavljenje zrcaljenja koje distribuira opterećenje sesije pokretanjem više paralelnih instanci visoko naseljenih zona. Igrači u različitiminstancama ne mogu komunicirati

Izvor: Prodan, Radu, and Vlad Nae. "Prediction-based real-time resource provisioning for massively multiplayer online games." *Future Generation Computer Systems* 25.7 (2009): 785-793.

# Izlazak Warlords of Draenor ekspanzije za World of Warcraft



Zavod za telekomunikacije

- ◆ 13.11.2014
- ◆ Veliki problemi s poslužiteljima
- ◆ Usluga je bila djelomično ili u potpunosti nedostupna 4 dana
- ◆ Riješen problem kroz ograničavanje broja igrača na pojedinom poslužitelju – uzrokovalo redove čekanja ponekad i po nekoliko sati

***Skalabilnost u velikim umreženim virtualnim okruženjima je još uvijek neriješen problem!***

# Izlazak Shadowlands ekspanzije za World of Warcraft



Zavod za telekomunikacije

- ◆ Ekspanzija izašla 27.11.2020.
- ◆ Jako mali problemi na početku
- ◆ Princip dinamičkog instanciranja svake zone za određeni broj igrača
- ◆ Funkcionira isto kao sharding, ali je dinamički te igrači jednostavno mogu otići u drugu instancu zone, ako ih pozove igrač koji je već tamo

# Virtualna okruženja

Igor S. Pandžić, Mirko Sužnjević

Virtualna stvarnost

# Pregled predavanja

- Uvod
  - Definicija
  - Povijest
  - Tržišni trendovi
- Uređaji za VR
  - Ulazni uređaji
  - Izlazni uređaji

# Definicije

- Sam termin Virtualna stvarnost (engl. Virtual Reality – VR) je osmislio Jaron Lanier, direktor VPL Research kompanije **1989. godine** kako bi opisao sve projekte u kojima je radio s virtualnim okruženjima pod isti termin. Njegova kompanija je među prvima prodavala naočale, rukavice kao i cijela odijela za sustave virtualne stvarnosti
- **George Coates (1992):** Virtual Reality is electronic simulations of environments experienced via head mounted eye goggles and wired clothing enabling the end user to interact in realistic three-dimensional situations.
- **P. Greenbaum (1992):** Virtual Reality is an alternate world filled with computer-generated images that respond to human movements. These simulated environments are usually visited with the aid of an expensive data suit which features stereophonic video goggles and fiber-optic gloves.
- **Myron Krueger (1991):** ....The term (virtual worlds) typically refers to three-dimensional realities implemented with stereo viewing goggles and reality gloves.

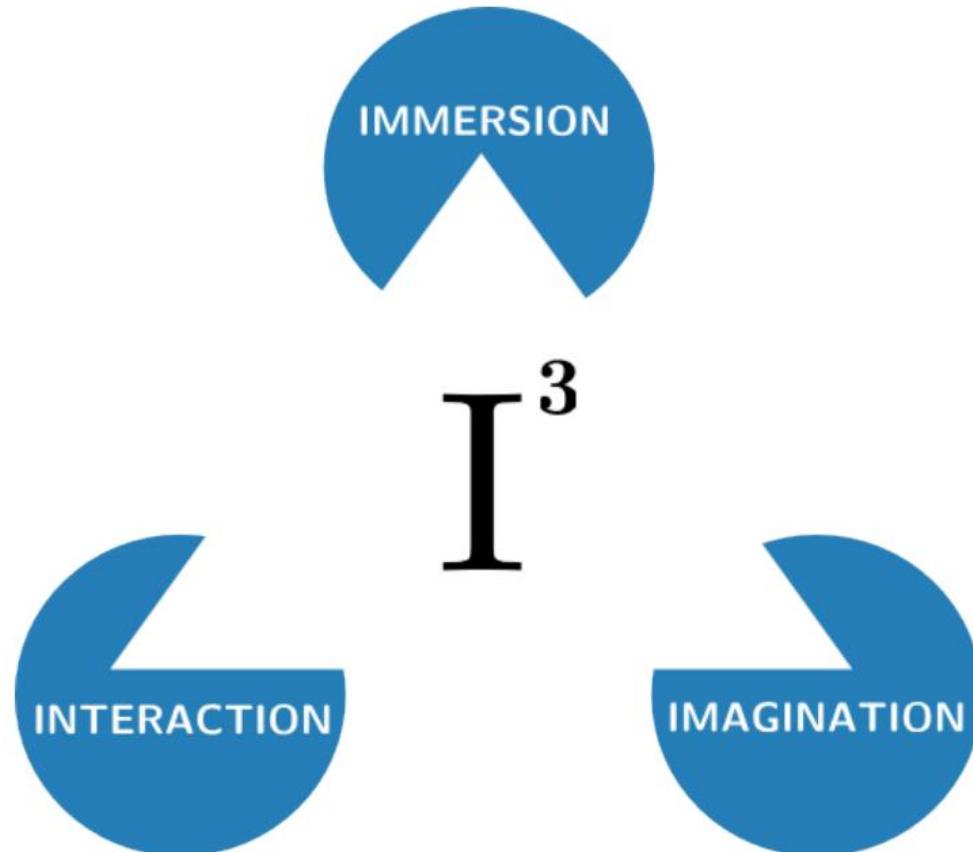
# Definicije

- **Mirriam Webster:** an artificial environment which is experienced through sensory stimuli (such as sights and sounds) provided by a computer and in which one's actions partially determine what happens in the environment
- *Generalna definicija: Virtualna stvarnost je pojam za računalne simulacije kojima je cilj stvoriti osjećaj prisutnosti korisnika u virtualnom okruženju*

# Prisutnost

- Prisutnost (engl. presence) – osjećaj prisutnosti u određenoj lokaciji/prostoru
- Teleprisutnost (engl. telepresence) – osjećaj prisutnosti negdje kroz komunikacijski medij
- Virtualna prisutnost – osjećaj prisutnosti u nekoj simuliranoj okolini – forma teleprisutnosti
- Virtualna prisutnost: “Is experienced by a person when sensory information generated only by and within a computer compels a feeling of being present in an environment other than the one the person is actually in” (Sheridan, 1992, pg.6)

# Trokut virtualne stvarnosti



# Prisutnost i/ili imerzija

- Prisutnost
  - Mentalna – potiskivanje sumnje u svijet (razmišljanje kao da je osoba u tom svijetu)
  - Tjelesna – tijelo ulazi u virtualni medij (refleksi na virtualni svijet)
- U kontekstu virtualne stvarnosti često se miješaju
  - Imerzija je objektivan opis tehnologije – karakteristike ekrana, brzina osvježavanja, prirodna sučelja i slično – imerzija je senzorski parametar koji je ovisan o razvijenosti tehnologije
  - Prisutnost je subjektivno iskustvo koje se može jedino izmjeriti kroz korisničke studije – prisutnost je kognitivni parametar ovisan o samom korisniku

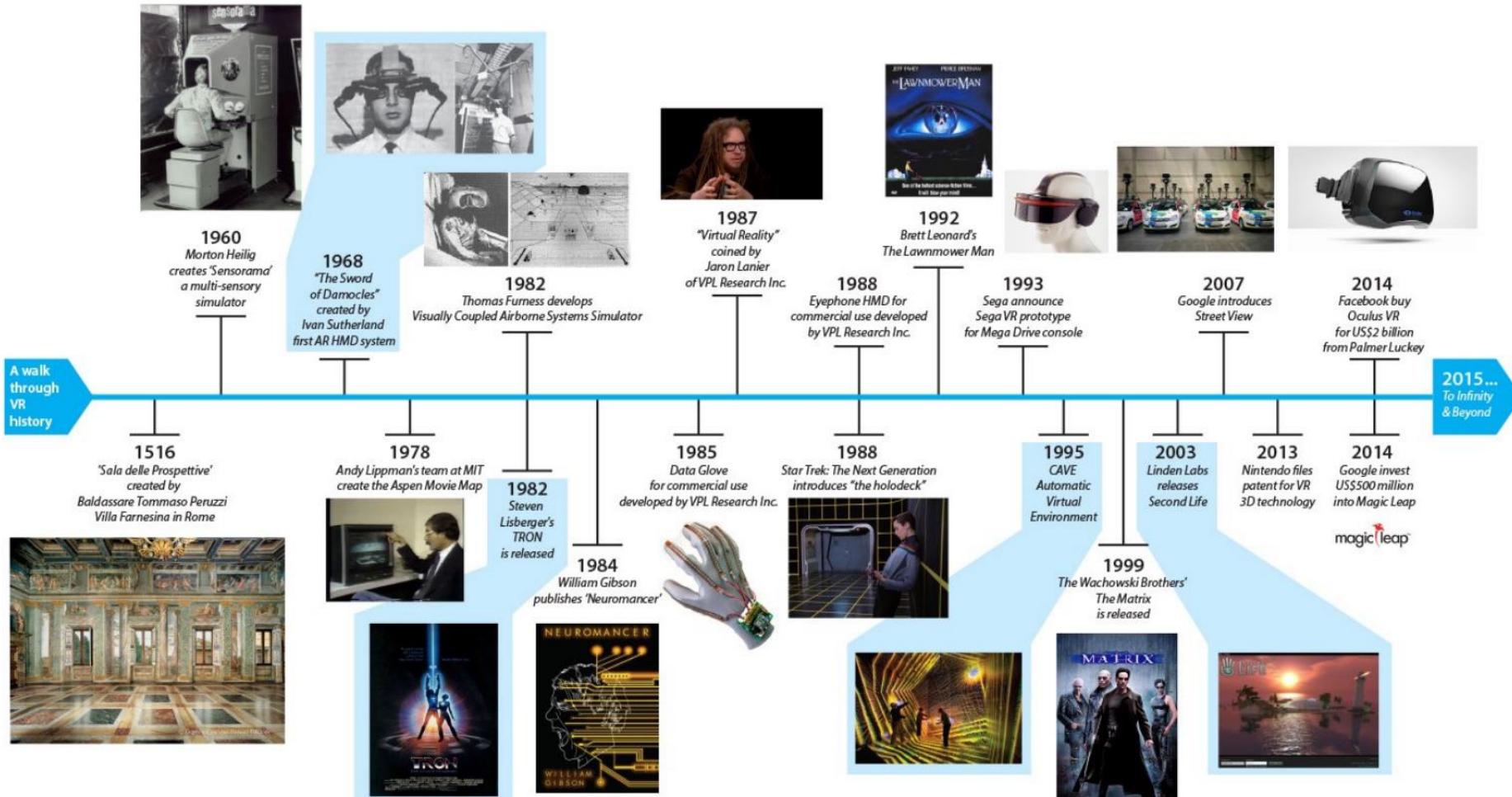


# Povijest

- 1956. Morton Heilig napravio prvi VR stroj – Sensorama
  - Stroj je koristio audio, video, vibracije, vjetar i mirise
- 1960. prvi HMD patent od istog izumitelja
- 1961. prvi HMD prototip koji je imao praćenje položaja glave – koristila ga je vojska za kontrolu udaljenih kamera
- 1968. Ivan Sutherland kreira prvi HMD za virtualnu stvarnost koji je bio točno lociran korištenjem mehaničkih i ultrazvučnih slijednika glave
- 1979. razvijen prvi magnetski slijednik položaja – omogućio vrlo bitne kasnije pomake u VR tehnologijama
- 1989. NASA razvija sustav za trening astronauta
- ...
- 2012 – prva kickstarter kampanja za Oculus HMD
- 2014 – Facebook kupuje Oculus za 2 milijarde dolara
- 2016 – izlazi HTC Vive sustav koji uključuje kontrolere za interakciju u virtualnom svijetu
- 2018 – novi HMD-ovi izlaze kao samostalni uređaji (bez poveznice s kompjutorom), varifokalne (prilagodljive) leće, praćenje pogleda...

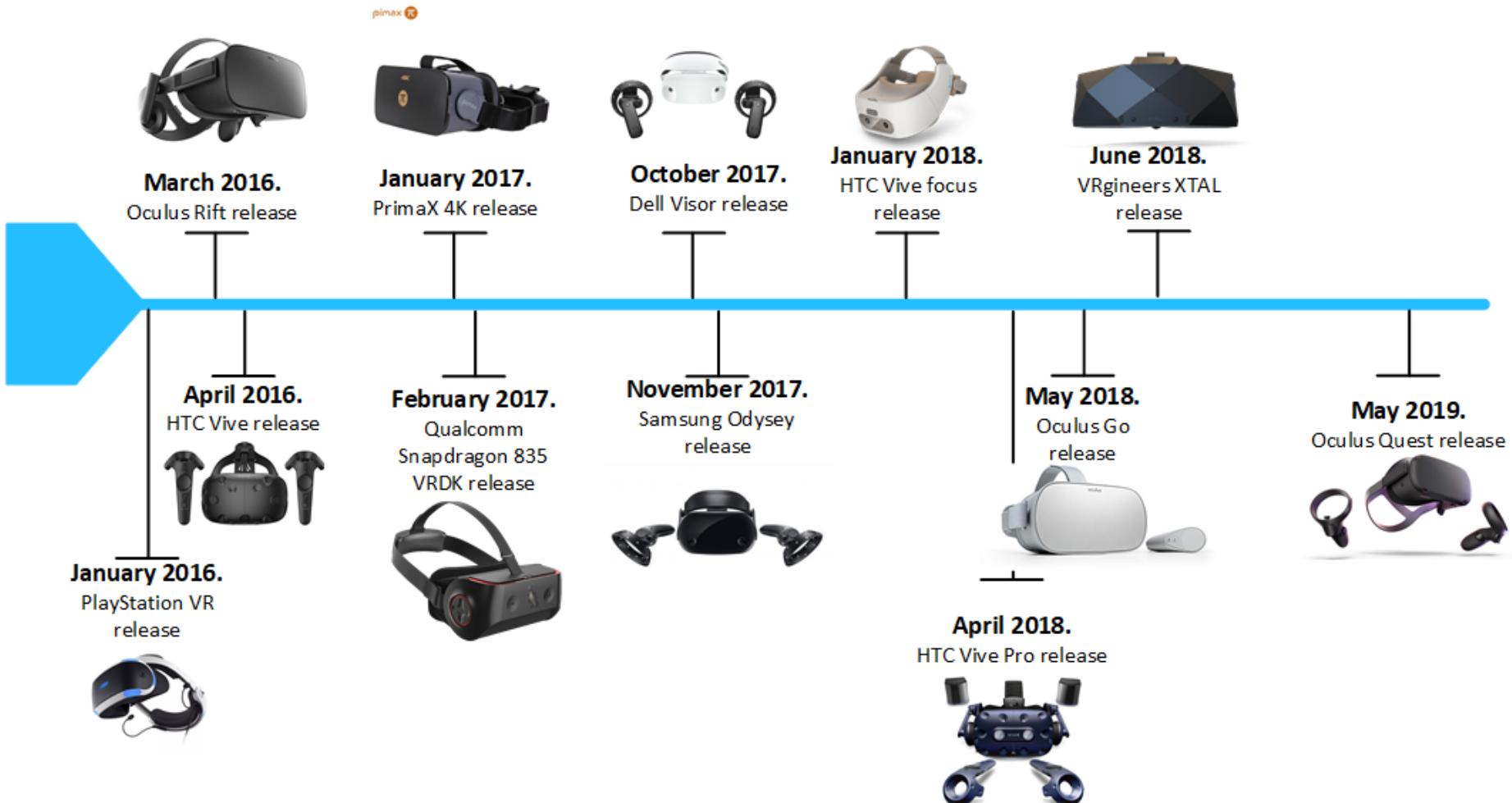


# Povijest

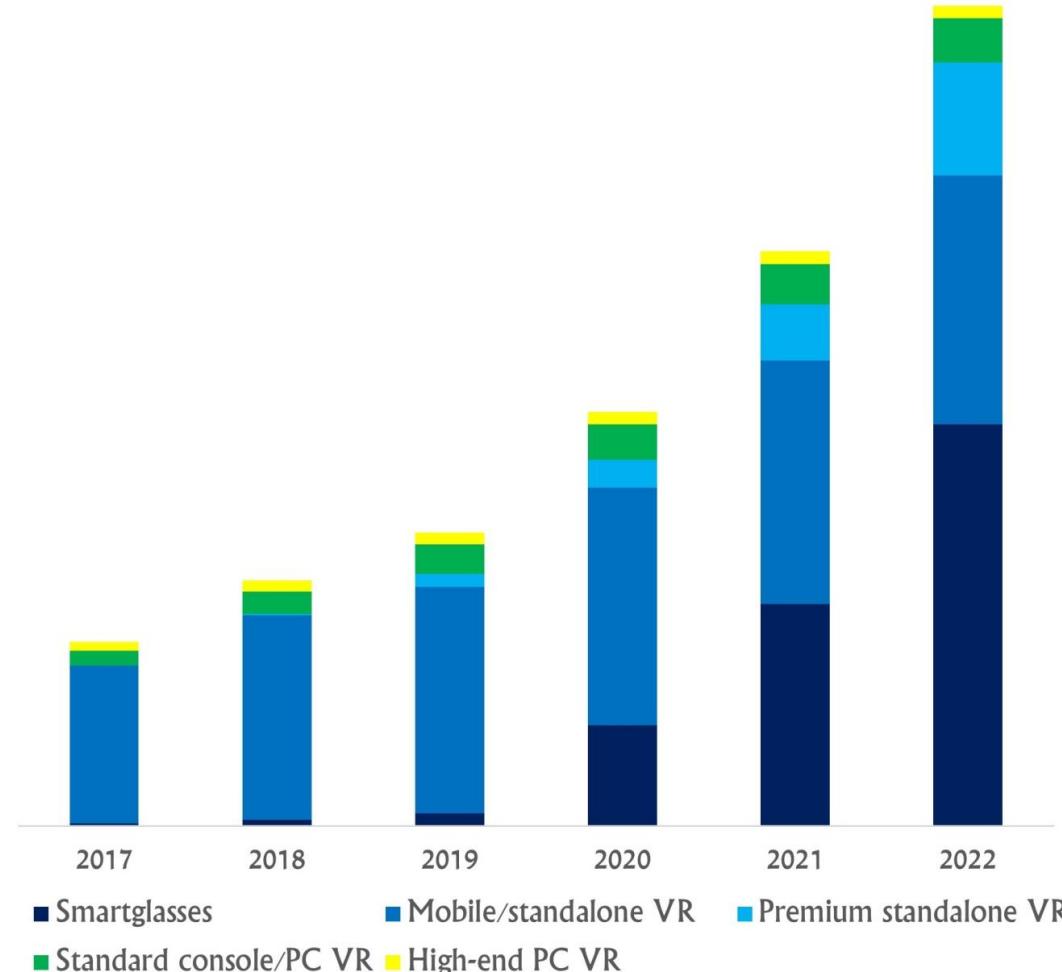


[http://dsc2018.org/Docs/Keynotes/OLIVER\\_KEYNOTE.pdf](http://dsc2018.org/Docs/Keynotes/OLIVER_KEYNOTE.pdf)

# Povijest



# Tržište – predviđanja



# Primjer usluge

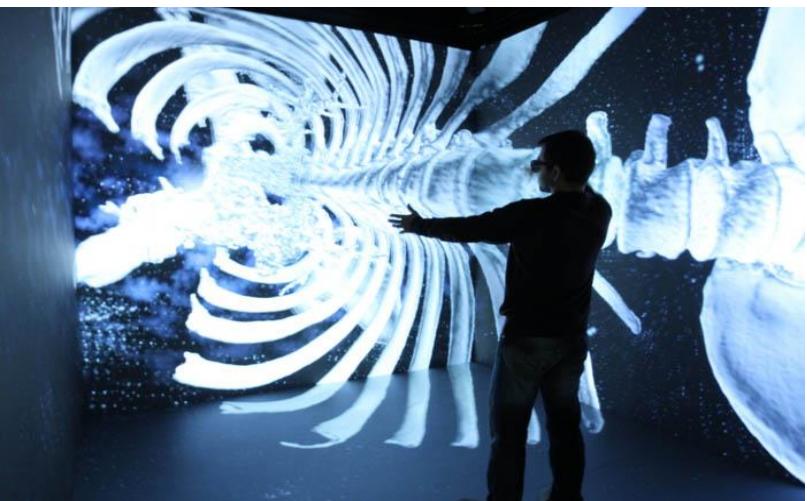
- Igra EvE Valkyre
  - Igrač je stavljen u ulogu pilota svemirskog lovca
  - U svakom trenutku iz kokpita pilotskog može gledati u svim smjerovima
  - <https://youtu.be/3SI1CB4rcf8?t=42>
- Aplikacija za proučavanje ljudskog tijela – Body VR
  - Putovanje kroz krvotok
  - Putovanje kroz pojedinu stanicu
  - Promatranje funkcije proteina i stanične jezgre
  - Proučavanje pojedinog sustava unutar tijela (primjerice probavni)
  - <https://www.youtube.com/watch?v=9zTsDXMyBEY>

# Princip virtualne stvarnosti

- Korisnik u zatvorenoj petlji



# Sustavi za virtualnu stvarnost



# Moderni sustavi za virtualnu stvarnost

- ◆ Najprodavaniji Sony PlayStation VR
- ◆ Danas za PC najpopularnija dva sustava
  - HTC Vive
  - Oculus Rift
- ◆ Osnovne komponente
  - Naočale
  - Kontroleri
  - Senzori za praćenje



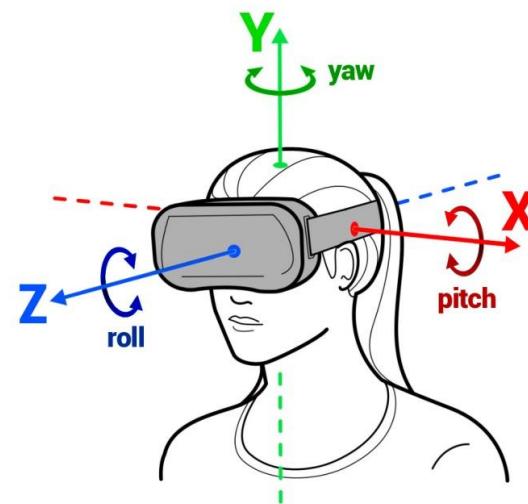
# Uređaji za virtualnu stvarnost

- Ulazni – uređaji preko kojih se informacije prenose od korisnika ka računalnoj simulaciji
  - Senzori pozicije – magnetski, optički, mehanički...
  - Senzori sile – spaceball
  - Senzori položaja tijela – rukavice, kontrolери
  - Senzori pokreta – pedale, hodalice, trake za trčanje u mjestu
  - Ostalo
- Izlazni – uređaji preko kojih se informacije iz računalne simulacije prezentiraju korisniku
  - Vizualni
  - Zvučni
  - Haptički
  - Ostalo – mirisi, toplina, vjetar...



# Tehnički izazov – praćenje pozicije

- Mijenjanje kuta gledanja korisnika potrebno praćenje tri stupnja slobode (engl. degrees of freedom – DOF)
- Mijenjanje kuta gledanja korisnika i kretanje korisnika u prostoru potrebno praćenje 6 DOF



# Senzori pozicije

- Inercijski senzori: (engl. Intertial measurement unit – IMU)
- Nalaze se na naočalama ili kontrolerima
  - Akcelerometar – senzor koji detektira ubrzanje
  - Žiroskop – uređaj koji zadržava svoju os rotacije (u idealnom sustavu zauvijek)
  - U elektroničkim uređajima: optički žiroskop i vibracijski žiroskop
  - Magnetometri



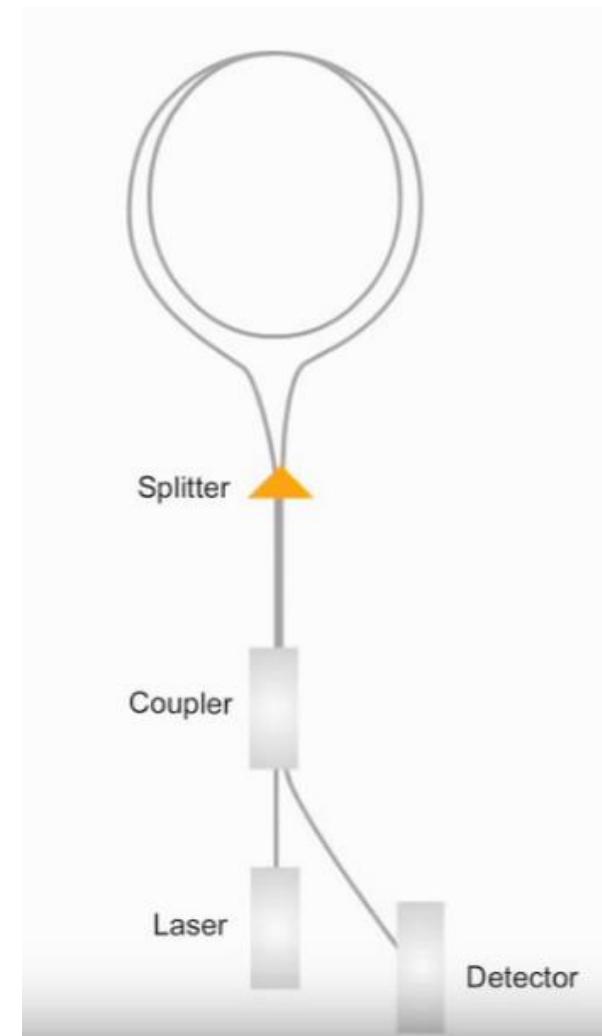
# Akcelerometar

- Pizoelektrični – sastavljeni od mikroskopskih kristalnih struktura koje se naprežu prilikom ubrzanja te generiraju određenu električnu struju – struja se može izmjeriti te time i ubrzanje
- Pizootporni – mjeri se otpor između kristalnih struktura koji se mijenja tijekom ubrzanja
- Kapacitivni – dvije mikrostrukture imaju određeni kapacitet među njima u slučaju ubrzanja koje pomjera jednu od struktura kapacitet se mijenja te se može izmjeriti

# Optički žiroskop

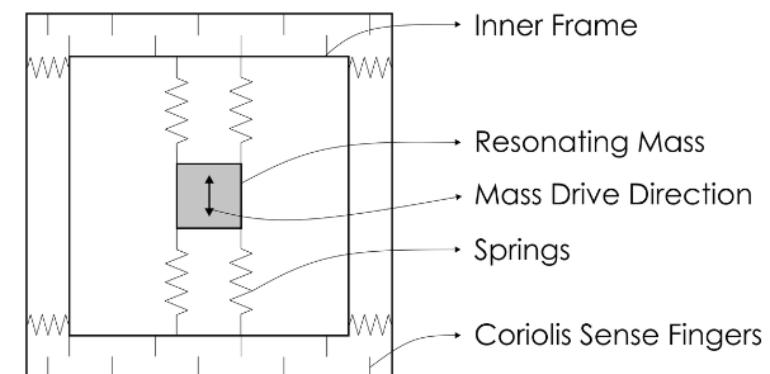
- Funkcioniranje
  - Šalju se dvije zrake svjetla u suprotnim smjerovima u zatvorenoj petlji
  - Mjeri se vrijeme koje je potrebno da se svaka zraka vrati
  - Na temelju razlike u vremenu povratka zraka može se deducirati smjer rotacije
  - Brže dolazi svjetlo koje ima kraću putanju tj. ono koje se kreće suprotno od smjera rotacije

<https://youtu.be/aO0pCbKSpvE?t=92>



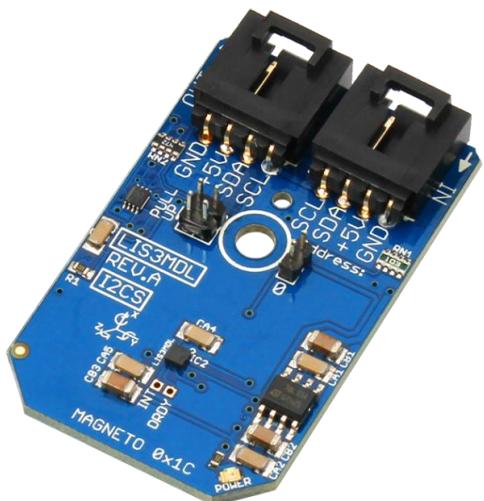
# Vibracijski žiroskop

- Vibracijski žiroskop je MEMS (engl. Micro-machined Electro-Mechanical Systems)
- Funktioniranje
  - Vibracijska struktura sadrži mikro masu koja je povezana s vanjskim dijelom kroz par opruga
  - Vanjski dio je povezan s elektroničkom pločom s parom opruga koje su ortogonalne na prethodne
  - Testna masa se cijelo vrijeme sinusoidno kreće po prvom paru opruga
  - Bilo kakva rotacija sustava će inducirati Coriolisovo ubrzanje mase koje će gurnut i masu po osi na kojoj je drugi par opruga.
  - Ako se masa gura od osi rotacije gura u jednom smjeru, a ako se gura prema osi rotacija gurat će u drugom smjeru zbog Coriolisove sile



# Magnetometar

- Magnetometar – mjeri magnetsko polje te na temelju njega poziciju
- Mjeri magnetsko polje zemlje na sve tri osi
- Kompas
- Problemi
  - Osjetljivost na druga magnetska polja koje generiraju željezni te magneti od nikla



# Elektromagnetski sljednici (engl. tracker)

- Sastoje se od odašiljača i prijemnika
- Odašiljači generiraju elektromagnetsko polje koje se prima u prijemnicima te se njegove promjene analiziraju i na temelju toga procjenjuju pokreti korisnika
- Prednosti
  - Rad u stvarnom vremenu
- Jednostavno postavljanje
- Nedostaci
  - Kabeli (noviji sustavi bez njih)
  - Kratak domet
  - Osjetljivost na metal u okolini
  - Mali broj istovremenih slijednika

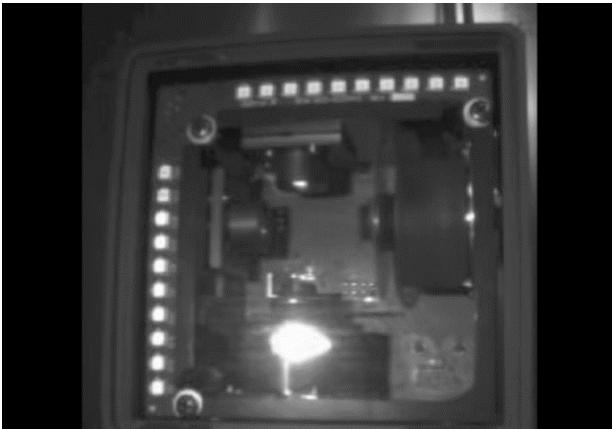
# Optički sljednici

- Sustavi sa više kamera (obično infracrvene engl. infrared – IR)
- Na predmetima koje slijedimo oznake od sjajnog materijala
- Oznake se slijede u 2D u slici svake kamere
- Kombiniranjem 2D položaja u slikama svih kamera, uz znanje o položajima i karakteristikama kamera, precizno se utvrđuje 3D položaj oznake
- Constealltion sustav iz Oculus Rifta
  - Svake naočale imaju set IR LED sijalica
  - Lako se identificira položaj zbog poznatog oblika odnosno rasporeda LED sijalica
  - Više ne koristi za ovu svrhu – kamere na naočalama sada ovako pra
- PlayStation VR – koristio vidljivi spektar svjetla



# Optički sljednici

- HTC Vive – Lighthouse sustav
- Senzori za praćenje nisu zapravo senzori te nisu povezani s računalom
- Emitiraju dva široka snopa IR svjetla jedan za drugim
- Snop gore-dolje, potom snop lijevo-desno
- Senzori koji se nalaze na samim naočalama te na kontrolerima mjeru vrijeme između snopova te na temelju toga izračunavaju poziciju



<https://www.youtube.com/watch?v=5yuUZV7cgd8>  
<https://www.youtube.com/watch?v=oqPaaMR4kY4>

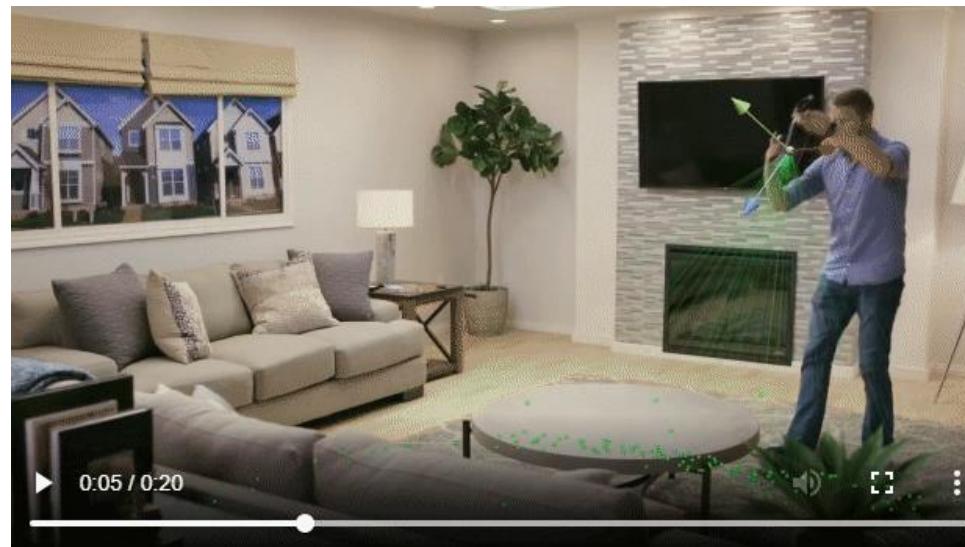
# Optički slijednici

- Dubinske kamere (engl. range camera)
- Za svaku točku daje boju i udaljenost od kamere, tj. kartu dubine (engl. depth map)
- Više principa rada:
  - lasersko mjerjenje,
  - stereo slike,
  - strukturirano svjetlo.
- Karta dubine olakšava praćenje tijela; postaje moguće u stvarnom vremenu jednom kamerom
- Popularizacija 2010. g.
  - Microsoft Kinect
  - Microsoft Kinect 2 (prekinuta proizvodnja)
  - Intel RealSense
  - Azure Kinect



# Optički slijednici

- SLAM (Simultaneous Location And Mapping ) sustavi
- Ne koriste senzore koji su odvojeni od uređaja
- Detektiraju poziciju naočala na osnovu statičkih elemenata u prostoriji – pomak tih elementa u kombinaciji s podacima s IMU određuje položaj naočala koristeći dubinske kamere
- Sličan način rada poput Constellation sustava koristeći ili IR ili vidljivo svjetlo gdje kamere na naočalama prate LED diode na kontrolerima

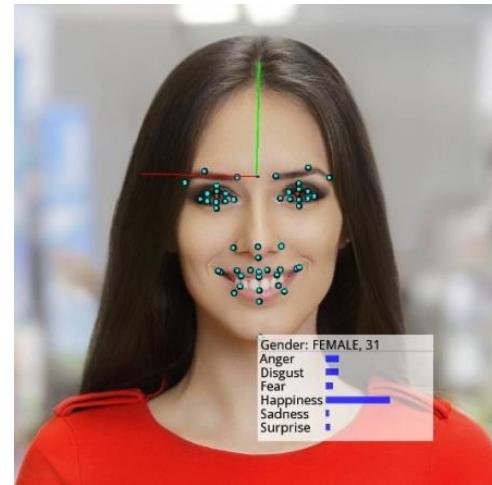


# Prednosti i mane

- Constellation prednosti :
  - Niska cijena
  - Visoka točnost
  - Radi u gotovo svim okolinama
- Constellation mane:
  - Senzori moraju biti povezani s PC-em
  - Velika USB propusnost može uzrokovati probleme na matičnim pločama
  - Ograničen vertikalni FOV
- PSVR prednosti:
  - Niska cijena
  - Koristi postojeće kontrolere (PS Move)
- PSVR mane:
  - Niska točnost
  - Nema podrške za veći prostor (sobu)
- Lighthouse prednosti :
  - Nije potrebna veza s PC-em
  - Visoka točnost
  - Širok kut praćenja
- Lighthouse mane:
  - Relativno skupo za proizvodnju i integraciju
  - Potrebni vanjski senzori koji se montiraju u sobi
  - Površine koje reflektiraju svjetlo uzrokuju probleme
- SLAM prednosti:
  - Nije potreban dodatni senzor
  - Vrlo niska cijena
  - Vrlo lagano postavljanje
- SLAM mane:
  - Ne radi u mraku
  - Ne prati kontrolere ako je nešto između njih i naočala (primjerice leđa ili ruka)

# Optički slijednici pokreta lica

- Unos pokreta lica za animaciju avatara
- Prije korišteni markeri, a danas tehnologija obrade slike omogućava mapiranje lica na temelju običnih kamera (Visage Technologies SDK)
- Unos pokreta lica za upravljanje – praćenjem smjera pogleda (engl. gaze control)
- Neke naočale već podržavaju kontrolu pogledom (FOVE VR), a neki ga uvode (HTC Vive Pro Eye)



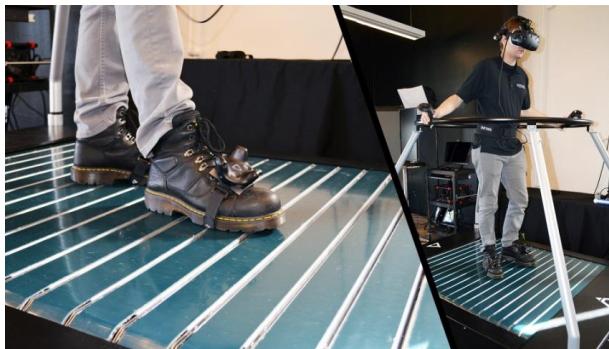
# Senzori položaja tijela

- Položaj šaka i prstiju
  - Posebni senzori – Leap Motion
  - Kontroleri
  - Procjena na temelju pritisnutih gumba na kontroleru – Oculus Rift
  - Rukavice za virtualnu stvarnost
  - I ulazni i izlazni uređaji – haptički povrat informacija



# Senzori kretanja

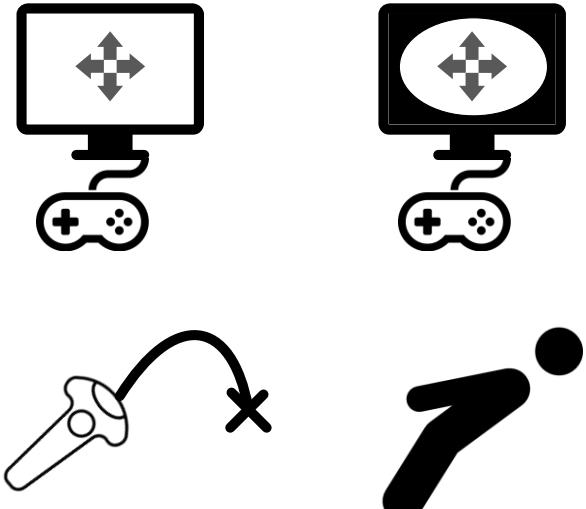
- Položaj i kretanje tijela – platforme za kretanje
  - Virtux Omni, KAT Walk VR, Infinadeck...
  - Klizanje obuće – posebna obuća
  - Pomične trake
  - Visoka cijena – još nije u rangu osobnih uređaja
  - Koriste se u zabavnim parkovima



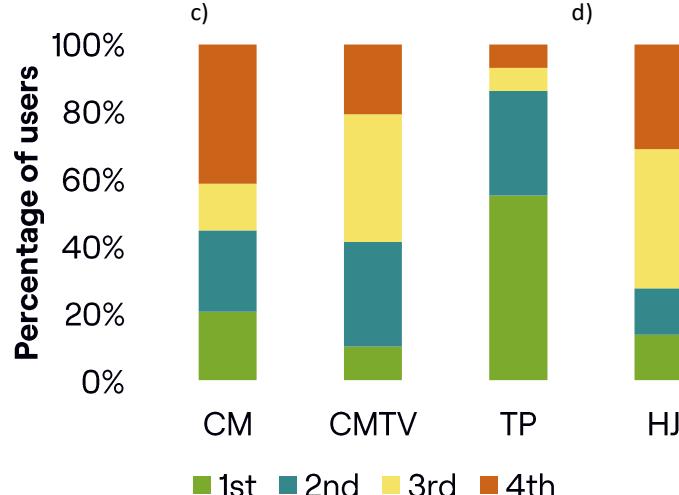
# Ograničenja virtualne stvarnosti – kinetoza

- Kinetoza ili bolest kretanja je normalan odgovor na pokret kad više osjetnih sustava tijela (vidni, sustav ravnoteže, osjet za položaj u prostoru) različito doživljavaju pokret
- Kinetoza se javlja prilikom putovanja različitim prijevoznim sredstvima, ali i u virtualnoj stvarnosti – oči nam govore da se krećemo, ali tijelo ne detektira taj pokret
- Simptomi – neugoda, mučnina u stomaku, glavobolja, znojenje, povraćanje, dezorientacija...
- Kod sustava za virtualnu stvarnost – simulacijska bolest (engl. simulator sickness)
  - Kretanje u virtualnoj stvarnosti jako često uzrokuje simulacijsku bolest
  - Utječu parametri poput kašnjenja u simulaciji, frekvencije osvježavanja ekrana, rezolucije ekrana, reprezentacije avatara u odnosu na stvarni položaj tijela
  - Veliki problem za sustave virtualne stvarnosti – najčešće su današnje igre i aplikacije stacionarne odnosno imaju diskretno kretanje (teleportiranje)

# Kretanje u virtualnoj stvarnosti



- Korisnicima najmanje smeta teleportiranje!

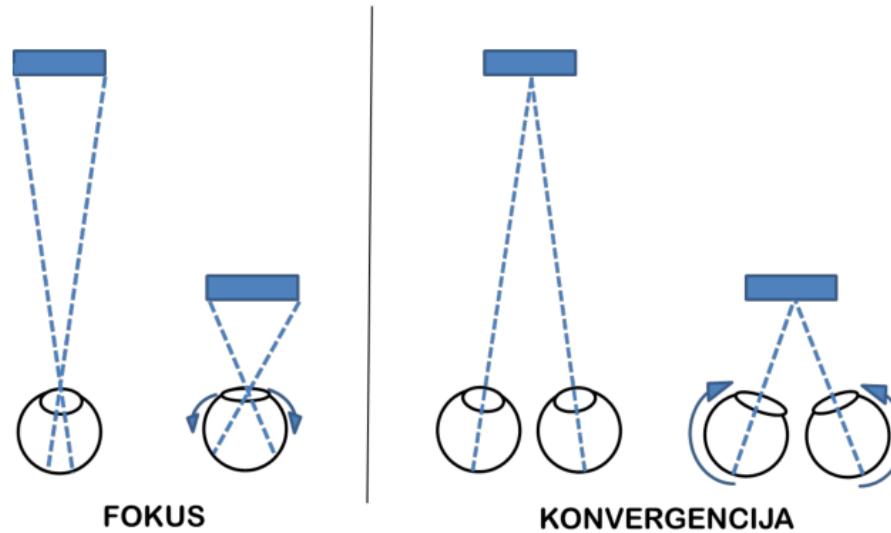


# Izlazni uređaji

- Vizualni izlazni uređaji
  - Zaslon na glavi (HMD)
  - Projekcijski sustavi – stereoskopska projekcija
- Zvuk
- Haptički izlazni uređaji
  - Taktilni izlazni uređaji
  - Uređaji za povrat sile
- Pomične platforme
- Ostalo
- Miris, vjetar, toplina

# Zašto se vidi 3D

- Prekrivanje predmeta
  - Sjene
  - Stereoskopska slika:
- Svako oko gleda iz druge perspektive
- Okulomotorni faktori: fokus, konvergencija
- Perspektiva: udaljeni predmeti manji
- Paralaksa gibanja: prilikom kretanja, bliži predmeti se prividno kreću brže od udaljenih



# Simulacija 3D vida na računalu

- Prekrivanje predmeta: Z-spremnik i sl.
- Sjene: postoje tehnike simulacije u stvarnom vremenu
- Stereo slika: više mogućih rješenja
- Fokus i konvergencija problematični
  - Računalo ne zna na koju dubinu korisnik fokusira
  - Model kamere je idealan, nema fokusa
  - Fizički, slike svih predmeta su na istoj udaljenosti od očiju
  - U stereo slici, nema konvergencije: bliski predmeti se vide dvostruko (ukoliko se prisilimo gledati "u daljinu", predmeti "iskaču" iz ekrana)
- Perspektiva i paralaksa gibanja dobivaju se virtualnom kamerom s perspektivnom projekcijom

# Zasloni na glavi

- Dva mala ekrana, po jedan za svako oko
  - Optički sustav ispred očiju, inače ekran preblizu
- Važna svojstva
  - Veličina, težina, udobnost
  - Rezolucija (danas standard 2160 x1200 pixel-per-eye)
  - Frekvencija osvježavanja (danas standard 90 sličica po sek.)
  - Vidni kut
- Montira se i sustav sljednika, tako da računalo generira sliku iz perspektive korisnika

# Stereoskopski prikazi

- Stereo zasloni s izmjenjivanjem – dvije slike se izmjenjuju (obično 120 Hz)
  - Zaklopne naočale (engl. shutter glasses)
  - Mogu zatvoriti svako staklo posebno polariziranjem
  - Otvaranje sinkronizirano s pojavom slika na ekranu: desno oko otvoreno uvijek kada je desna slika na ekranu, i obrnuto
- Autostereoskopski zasloni
  - Mikro-prizme ugrađene u zaslon odbijaju sliku lijevo i desno
  - Ljeva i desna slika istovremeno prikazane na ekranu, optički sustav ih razdvaja
  - Vidimo 3D bez posebnih naočala
  - Gledatelj treba biti u određenom položaju u odnosu na zaslon (obično u sredini)
- Stereoskopska projekcija
  - Dva projektorja s polarizirajućim filterima
  - Polarizacija u okomitim smjerovima
  - Gledatelji nose naočale s polariziranim staklima
  - Jedno oko horizontalno, drugo vertikalno polarizirano
  - Svako oko vidi samo sliku s projektorja iste polarizacije
  - Relativno jednostavan i jeftin sustav (još jeftinije s filterima u boji umjesto polarizacije)

# Zvuk

- Daje dodatne informacije (govor, zvučni signali)
- Doprinosi stvaranju ugodaja
- 3D zvuk – čovjek ima vrlo dobru sposobnost lokalizacije zvuka
  - To nije samo posljedica stereo slušanja!!
  - Head-Related Transfer Function (HRTF): Zvučni signali iz različitih smjerova različito se prenose u uho, uz složene efekte odbijanja na ramenima, vanjskim dijelovima uha, oko glave...
- Moguće je simulirati lokalizirani 3D zvuk

# Haptički izlazni uređaji

- Taktilni izlazni uređaji
  - Simulacija dodira, najčešće vibracijama (gotovo svi današnji kontroleri)
  - Montiraju se mikroelementi koji vibriraju ili se griju, te se time korisniku daie dojam dodira
- Uređaji za povrat sile
  - Motorima na ruku i tijelo – potrebno uporište
- Pomične platforme
  - Sustavi za pomicanje čitave platforme na kojoj stoji ili sjedi korisnik ili njih više
  - Često u zabavnim parkovima – korisnici obično zatvoreni u “vozilo” na platformi, s integriranim velikim ekranom u vozilu



# Programiranje za VR

- Umnogome pojednostavljeno kroz razne sustave za razvoj igara (engl. Game engine) te SDK sustave koje nude pojedini proizvođači – jednostavno “build for VR”
  - Unity
  - Unreal engine
  - Cry Engine
  - Godot (Open Source)
- Još ne postoje standardi za programska sučelja za VR
- Postoji Open Source Virtual Reality SDK

Video Game Engines with VR support

Platform	Unity	Unreal Engine	CryEngine	Godot	Platform-specific SDK
Steam VR	Yes <sup>[1]</sup>	Yes <sup>[2]</sup>	Yes, CryEngine V <sup>[3]</sup>	Yes <sup>[4]</sup>	
Oculus PC (Rift and Rift S)	Yes <sup>[5]</sup>	Yes <sup>[2]</sup>	Yes <sup>[6]</sup>	Yes <sup>[4]</sup>	Oculus PC SDK <sup>[4]</sup>
Oculus Mobile	Yes	Yes	No	No	Oculus Mobile SDK <sup>[4]</sup>
Windows Mixed Reality	Yes, Unity 2017.2 and later <sup>[7]</sup>	Yes, Unreal Engine 4 and later	Unknown	Unknown	
Open Source Virtual Reality	Yes <sup>[8]</sup>	Yes <sup>[9]</sup>	Yes <sup>[10]</sup>	Yes, via OpenHMD	OSVR SDK <sup>[4]</sup>

# Programski paketi za razvoj VR u sustavu Unity (fokus na interakciju)

- Virtual Reality Toolkit VRTK
- Kolekcija korisnih skripti koja omogućuje jednostavniju izgradnju VR aplikacija
- Dostupan iz Asset Store kao i s githuba
  - <https://assetstore.unity.com/packages/tools/integration/vrtk-virtual-reality-toolkit-vr-toolkit-64131>
  - <http://github.com/thestonefox/vrtk>  
Pokriva sljedeće funkcionalnosti
    - Kretanje u virtualnom prostoru
    - Osnovne interakcije – dodirivanje, uzimanje i korištenje objekata
    - Interakciju s Unity3D UI elementima
    - Podrška za fiziku tijela u 3D prostoru
    - 2D i 3D kontrole poput gumba, poluga, vrata, itd.
- <https://www.youtube.com/watch?v=vH5zHo6ql84>

# Programski paketi za razvoj VR u sustavu Unity (fokus na interakciju)

- NewtonVR
- Omogućuje korisnicima da podignu, bace i drže objekte
- Masa objekata se uzima u obzir u izračunu interakcije
- Dostupan iz Asset Stora i githuba
  - <https://www.assetstore.unity3d.com/en/#!/content/75712>
  - <https://github.com/TomorrowTodayLabs/NewtonVR>
- Podrška za SteamVR i Oculus SDK

# Virtualna okruženja

Igor S. Pandžić, Mirko Sužnjević

Proširena stvarnost

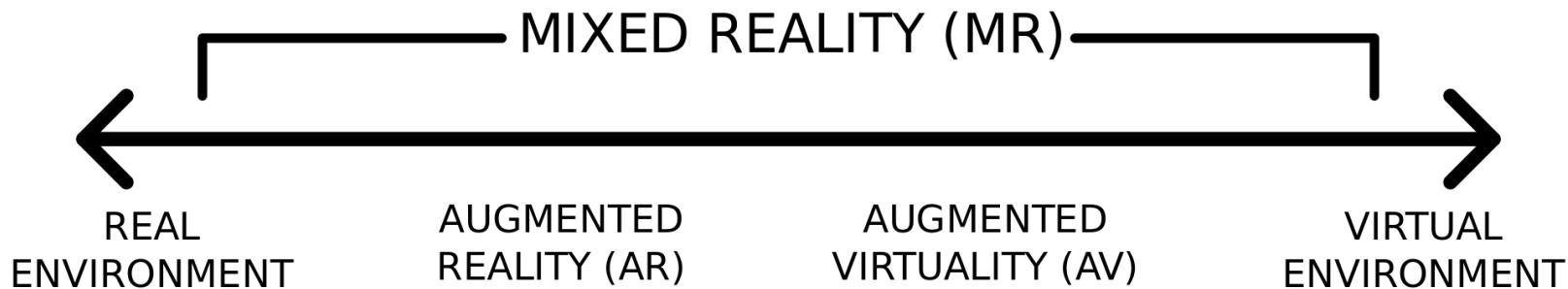


# Pregled predavanja

- Uvod
  - Definicija
  - Povijest
  - Tržišni trendovi
- Ključni aspekti
- Tehnički koncepti
  - Miješanje slike
  - Registracija (poravnanje)
  - Slijedenje
  - Tehnologije za prikaz
- Razvoj aplikacija za AR

# Definicije

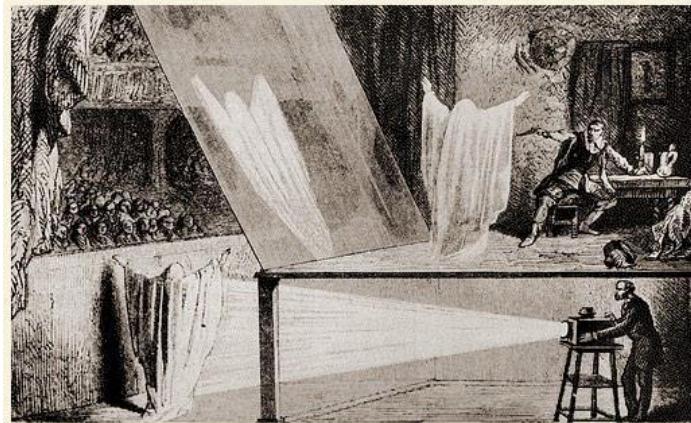
- Terminologija u području proširene stvarnosti nije još "konvergirala"
- Prije Microsoft Hololensa



- Nakon izlaska Microsoftovog Hololens uređaja najčešće korištena terminologija je :
- Proširena stvarnost (engl. Augmented Reality – AR) – preklapanje postojeće stvarne slike s 2D i 3D virtualnim objektima (primjerice Pokemon GO)
- Miješana stvarnost (engl. Mixed Reality – MR) – proširena stvarnost sa mogućnostima postavljanja virtualnih 3D objekata u stvarnu scenu i interakcije s njima kroz pozicioniranje i preslikavanje (engl. Simultaneous Localization And Mapping - SLAM)
- Proširena stvarnost v2 (engl. Extended reality) – ono što je prije predstavljala miješana stvarnost

# Povijest

- 1862 – Pepper's ghost
- 1901 – The Master Key koncept AR naočala
- 1958 – prve implementacije Heads Up Display (HUD) u avionima
- 1961 – Philco Headsight – gledanje kroz udaljenu kameru
- 1968 – Shuterland prvi naočale za virtualnu stvarnost (prozirne)
- 70ih i 80ih Nastavljen razvoj kaciga za pilote
- 90ih – Boing stvara termin “AR” te koristi naočale za proces sastavljanja
- 90ih se intenzivira daljnje istraživanje

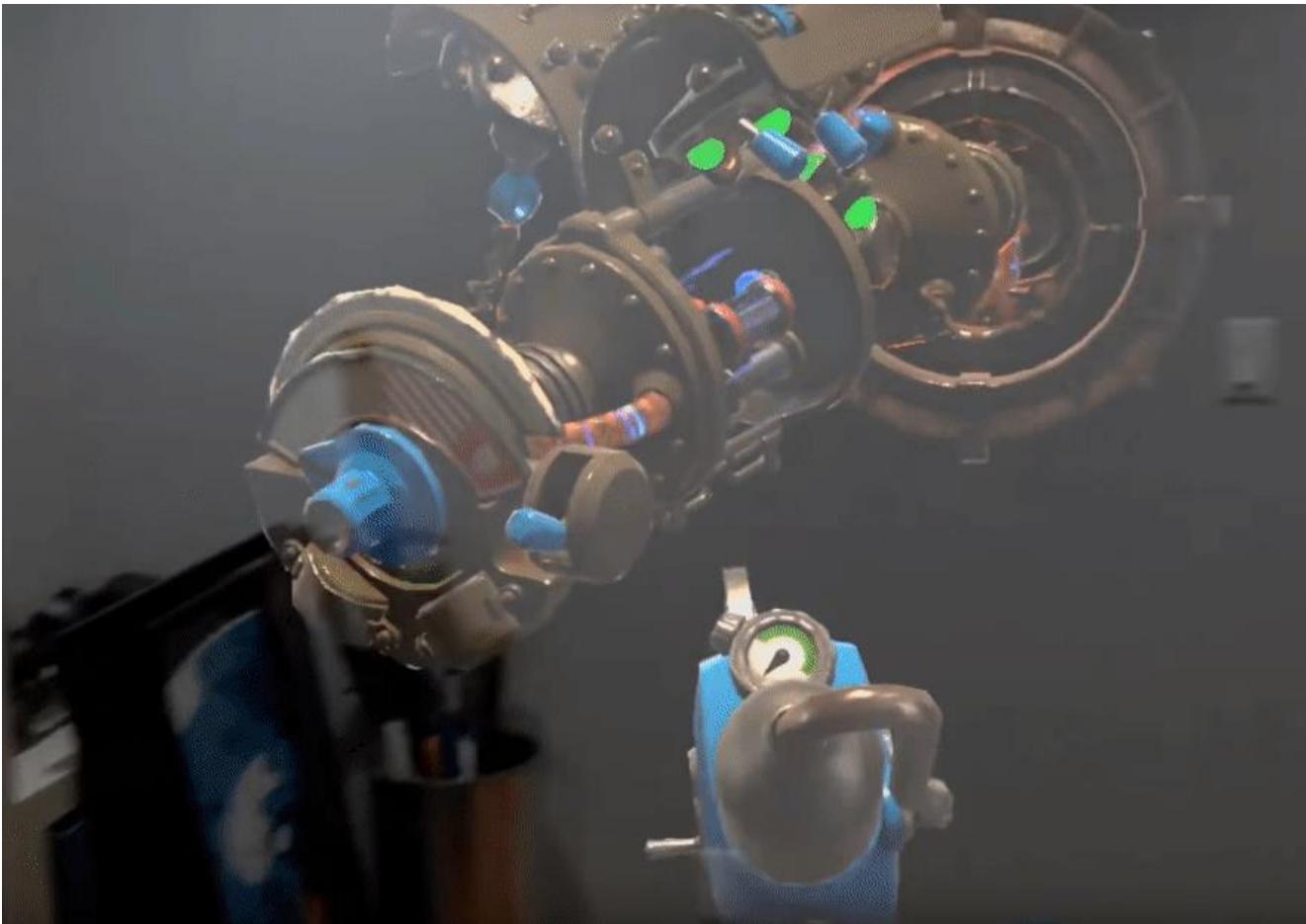


# Povijest

- 2005 – AR na mobilnim telefonima
- 2008 – AR u preglednicima u okviru Adobe Flash tehnologije
- 2011 – Google Glass naočale – velika medijska bura, ali na kraju mali komercijalni uspjeh i veliki problemi vezani za privatnost
- 2016 – Microsoft HoloLens 1
- 2017 – AR.js jednostavno uvođenje AR-a na temelju markera u web primjene (a mjesec dana kasnije i na temelju projekt Tangoa i AR bez markera)
- 2018 – Magic Leap naočale
- 2019 – Google Glass Enterprise edition 2
- 2020 – Nreal naočale

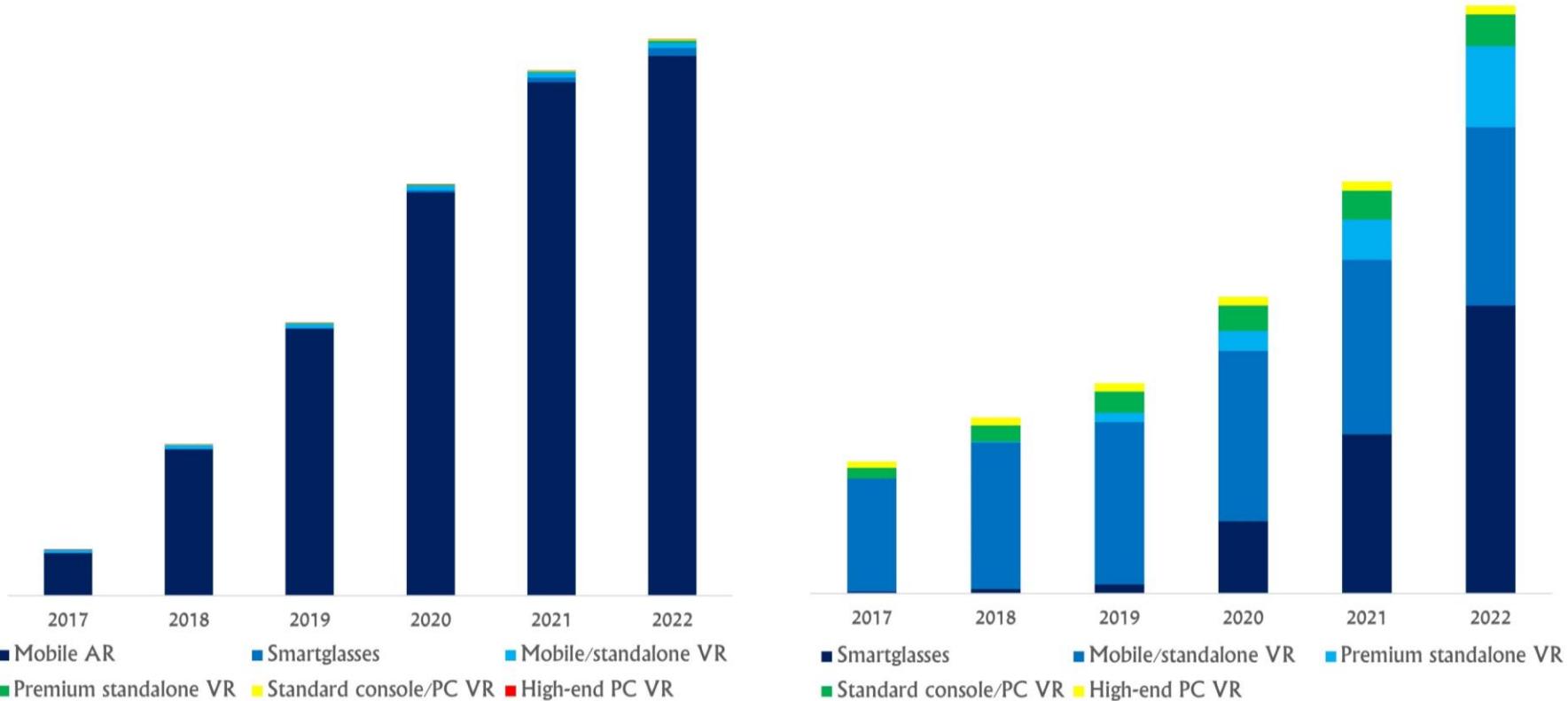


# Demo Magic Leap aplikacije



# Tržište

- Puno veći potencijal nego VR

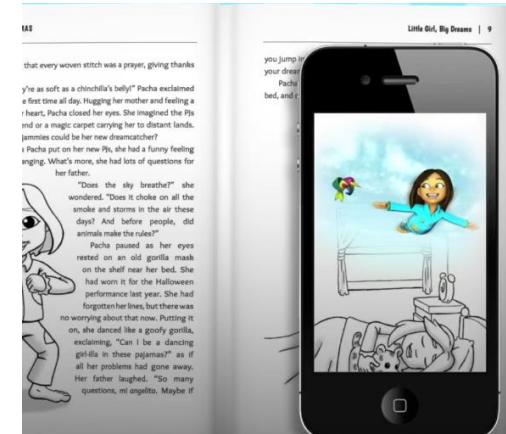


Digi-Capital™

© 2018 Digi-Capital. All rights reserved. No publication, adaptation, modification, reproduction or compilation without written permission from Digi-Capital

# Tržište

- Vrlo veliki prostor za primjene
  - Igre – Pokemon GO
  - Edukacija – dodavanje interaktivnosti u knjige
  - Proizvodnja
  - Marketing – Pepsi AR reklama
  - Sustavi za sastavljanje u proizvodnji
  - Primjena za pakiranje i skladištenje
  - Medicina
  - Muzeji
  - Turističke svrhe
  - Holoprotacija
  - Vojne primjene



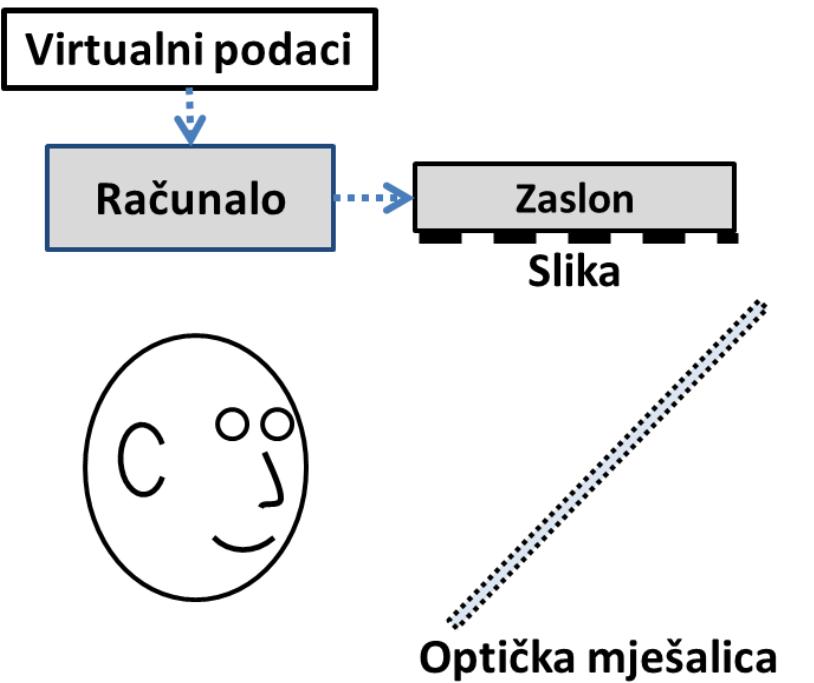
# Ključni aspekti

- Virtualne informacije se postavljaju u stvarni svijet.
- Virtualne informacije se registriraju na odgovarajućoj poziciji unutar stvarnog svijeta.
- Registracija se realizira s obzirom na perspektivu osoba u stvarnom svijetu koja se može mijenjati.
- Osobe koje su u AR iskustvu mogu komunicirati s virtualnim informacijama.

# Osnovni tehnički koncepti

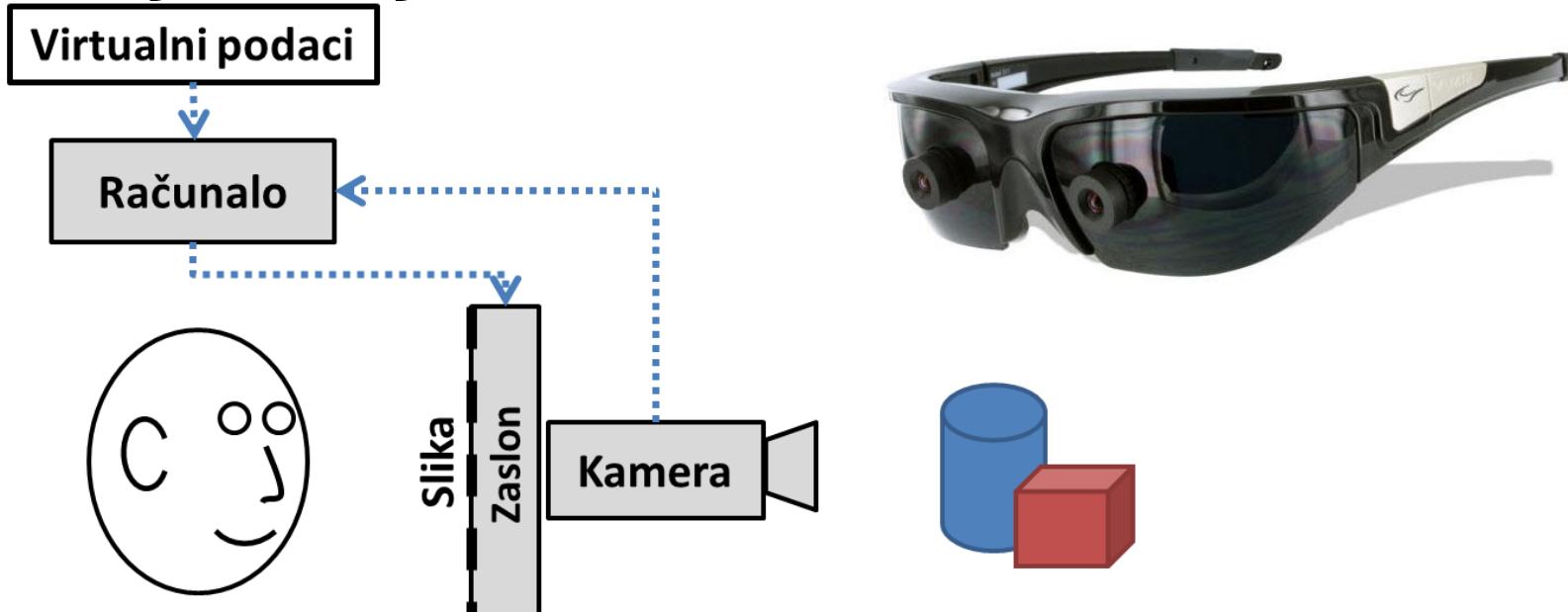
- Miješanje slike
  - Optičko
  - Video
  - Projekcijsko
- Registracija objekata
- Slijedenje
  - Pozicije i orijentacije korisnika
  - Pozicije virtualnih 3D objekata u stvarnom svijetu
    - Rješenja s markerima
    - Rješenja bez markera
- Prikaz
  - Zaslon na glavi
  - Zaslon u ruci

# Optičko miješanje



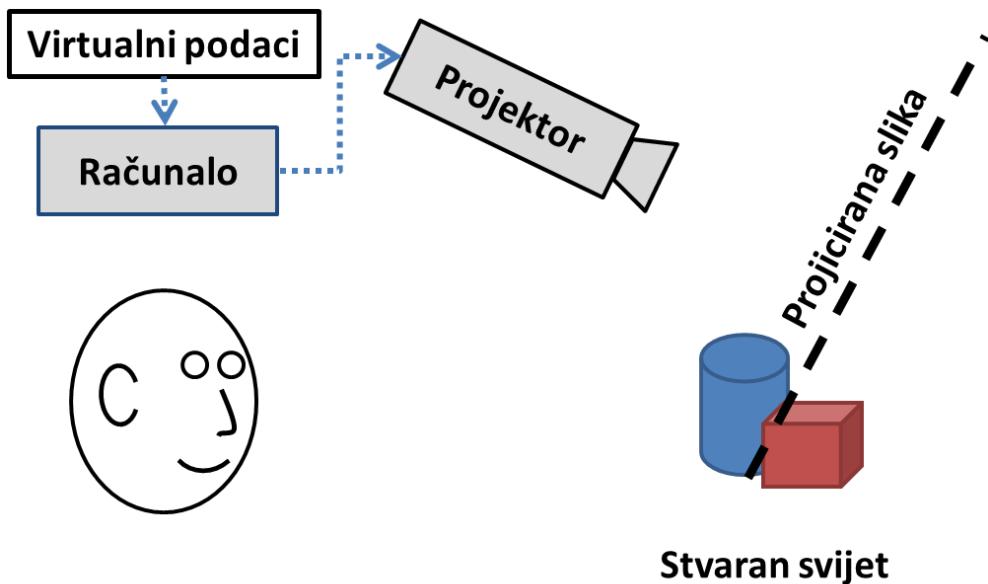
- Optička mješalica (engl. optical combiner) je polu-prozirno ogledalo, tako da korisnik vidi dvije slike
- Po jedan monitor za svako oko – stereo slika
- Prednosti:
  - Bolja vidljivost stvarnog svijeta
  - Lakše
- Mane:
  - Svjetlina – teže koristiti na otvorenom

# Video miješanje



- Nema direktnе stvarne slike; video signali stvarnog svijeta (iz kamere) i virtualnog (s računala) se miješaju
- Prednosti:
  - Video se može kontrolirati -> mogu se lako kombinirati grafički elementi i video
- Mane:
  - Percepcija stvarnog svijeta je degradirana
  - Kašnjenje – snimanje i prikaz videa uvode određeno kašnjenje u percepciji stvarnog svijeta

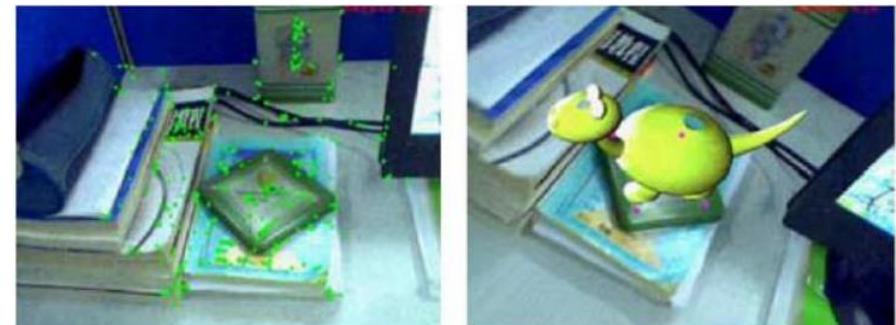
# Projekcijsko miješanje



- Virtualna slika se projicira na predmete u stvarnoj okolini
- Složeno za neravne površine
- Problemi osvjetljenja i prekrivanja

# Registracija ili poravnavanje

- Poravnavanje stvarnih i virtualnih predmeta u 3D
- Središnji problem proširene stvarnosti!
- Položaj promatrača i svih predmeta u sceni mora biti poznat što omogućava
  - Postavljanje virtualnih predmeta u isti koordinatni sustav sa stvarnim predmetima
  - Dinamičko iscrtavanje iz perspektive promatrača
- Potrebna je velika preciznost – ljudsko oko detektira pomake manje od jedne kutne minute



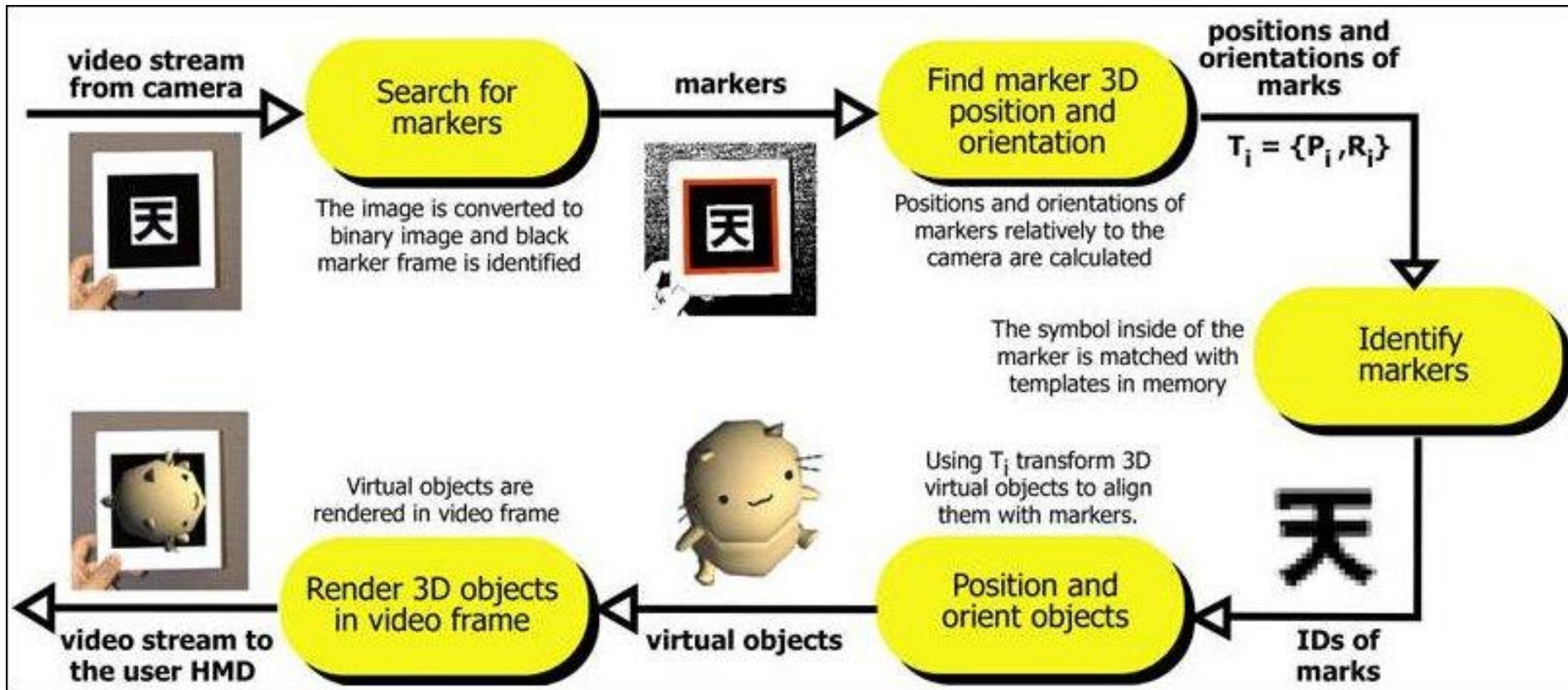
# Pogreške poravnavana

- Statičke pogreške
  - Greške slijedenja
  - Pogrešni parametri virtualne kamere (vidni kut, razmak među očima, pozicija slijednog elementa, parametri perspektivne projekcije...)
  - Optičko izobličenje
  - Mehaničke nepreciznosti opreme
- Dinamička pogreška
  - Nastupa zbog kašnjenja
  - Slijedenje, prijenos podataka, iscrtavanje
  - Ukupno kašnjenje ~100ms

# Slijedeње

- Postupak dobivanja pozicije/orijentacije predmeta u stvarnom vremenu
  - Sustavi slijedeњa prvo razvijeni za VR
  - Ovi sustavi obično nedovoljno precizni jer VR ima niže zahtjeve
- Tehnike slijedeњe – obrađeno na prethodnom predavanju
  - Aktivne
    - Mehaničke
    - Magnetske
    - Elektromagnetske
    - Mrežne (WiFi, bazne stanice, GPS)
  - Pasivne
    - Inercijski senzori
    - Optički (markeri, statični elementi, SLAM algoritmi)
  - Hibridni

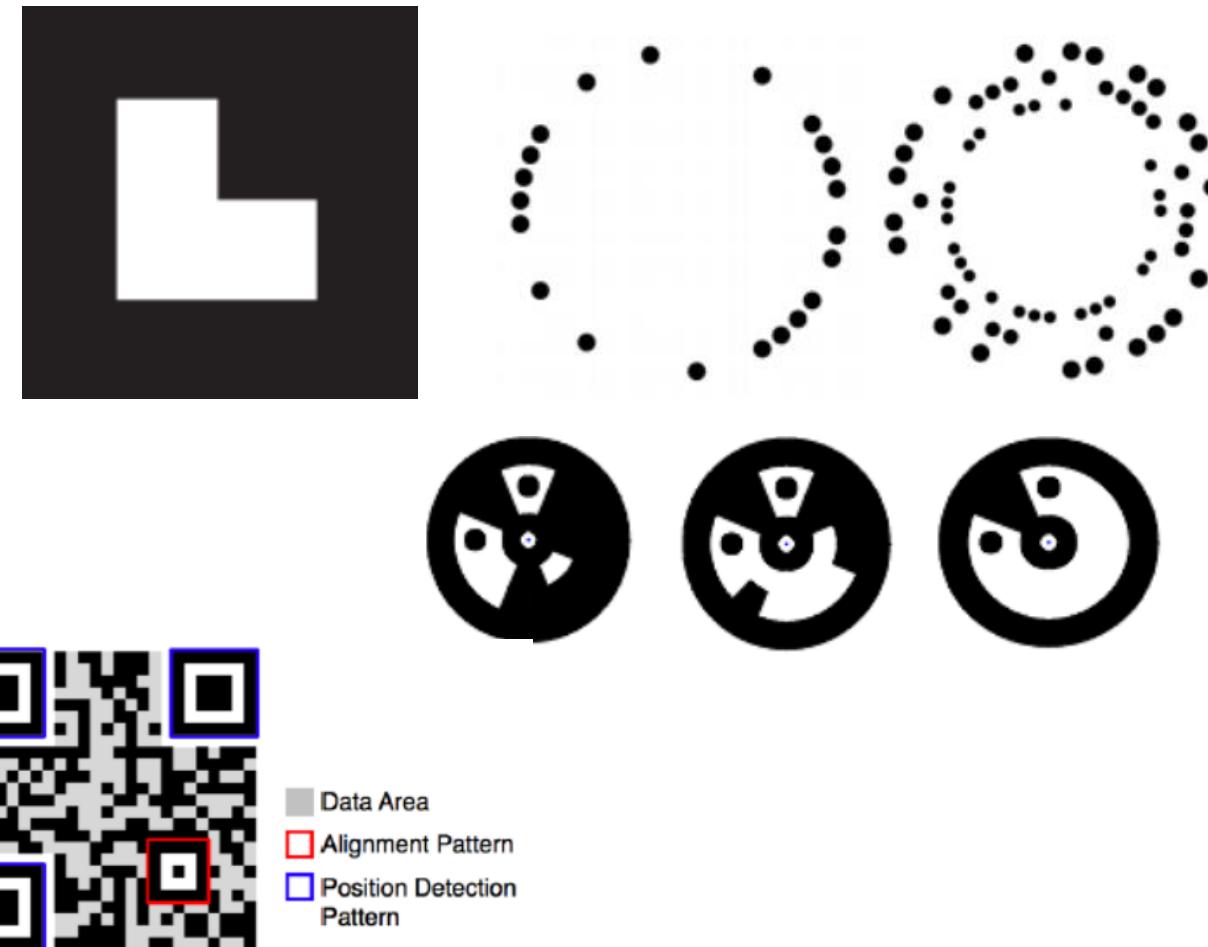
# Optičko slijedenje s markerima



<http://www.hitl.washington.edu/artoolkit/documentation/userarwork.htm>

# Tipovi markera

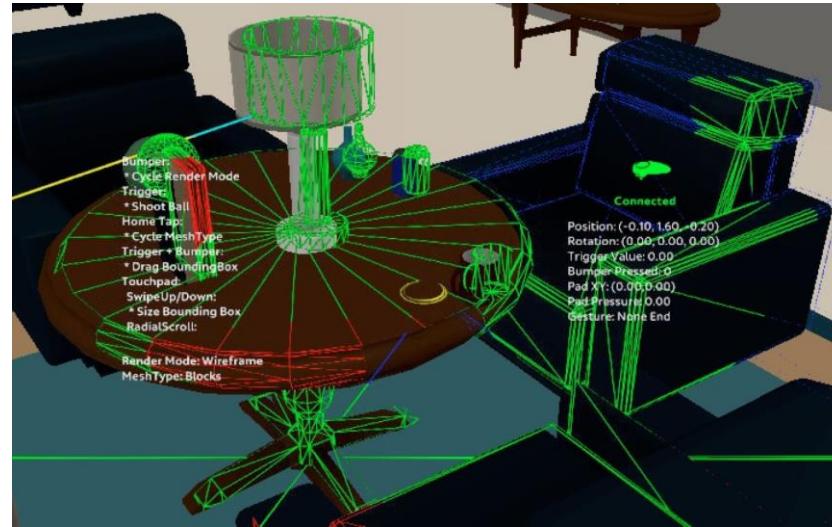
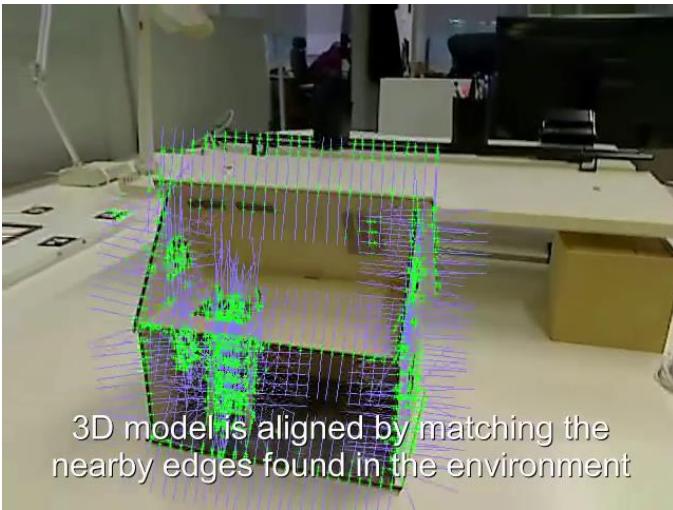
- Pravokutni
- Kružni
- Točkasti
- Abecedni
- QR kodovi
- Infracrveni
- Ugniježđeni



<http://www.cs.bham.ac.uk/~rjh/courses/ResearchTopicsInHCI/2014-15/Submissions/yan--yan.pdf>

# Bezmarkersko slijedenje

- Ne treba prethodno poznavanje prostora
- Temelji se na korištenju dubinskih kamera
- Temeljeno na SLAM algoritmima
- Magic Leap koristi i kombinaciju dubokih neuronskih mreža kako bi poboljšao praćenje



# Zaslon na glavi

- Naočale za proširenu stvarnost
  - Naočale za prikaz virtualnog sadržaja preko stvarnog svijeta
  - Naočale za pozicioniranje virtualnog sadržaja u stvarnom svijetu

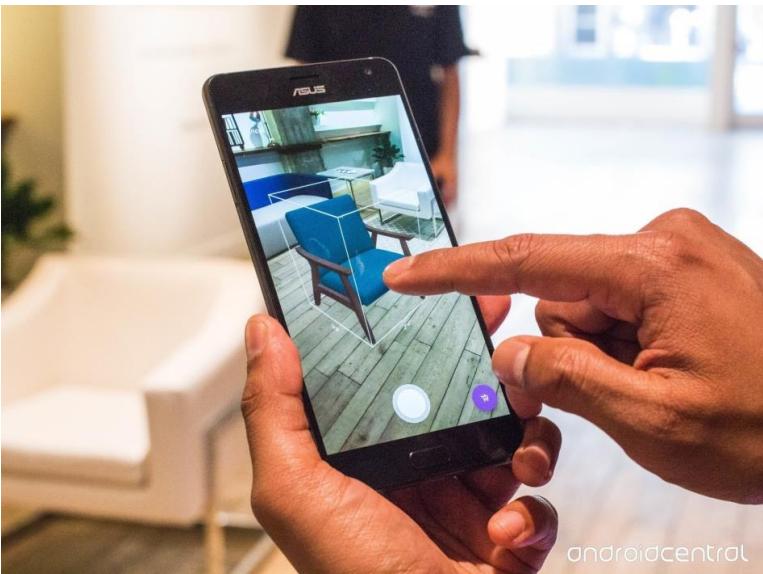
Tethered AR headset	FOV	Country	Release year	Price in USD (MSRP*)	Buy
Epson MOVERIO BT-300	23°	Japan	2016	\$699	<a href="#">Buy</a>
EverySight Raptor	-	Israel	2018	\$649	
Google Glass Enterprise Edition	-	US	2017	\$1,800	
Kopin SOLOS	10.68°	US	2016	\$499	
Meta 2	90°	US	2017	\$1,495	
ODG R-7	30°	US	2017	\$2,750	
Toshiba dynaEdge AR100 Viewer	-	Japan	2018	\$1,899	
Vuzix Blade Smart Glasses	-	US	2018	\$1,000	
ThirdEye Gen X1	40°	US	2017	\$1,299	<a href="#">Buy</a>
Vuzix M300	20°	US	2016	\$999	



Izvor: [https://www.aniwaa.com/best-of/vr-ar/best-augmented-reality-smartglasses/#What\\_are\\_the\\_best\\_AR\\_smartglasses](https://www.aniwaa.com/best-of/vr-ar/best-augmented-reality-smartglasses/#What_are_the_best_AR_smartglasses)

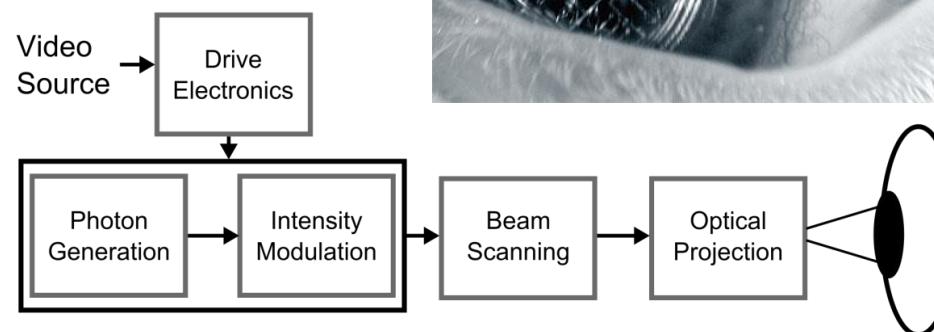
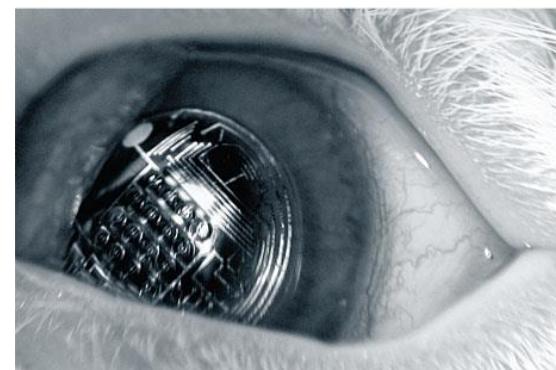
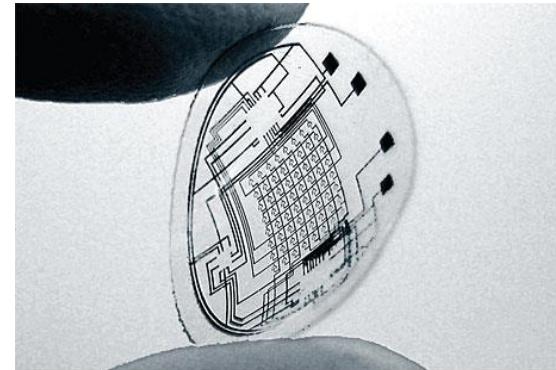
# Zaslon u ruci

- Pokretni uređaj – mobitel ili tablet
- Velika podrška u smislu programske podrške
- Trenutno najraširenija uporaba proširene stvarnosti



# Buduće tehnologije za prikaz

- Prikaz u retini
  - Fotoni projicirani direktno u oko
  - Dobra svjetlina (dobro ponašanje po danu na otvorenom)
- Prikaz na kontaktnoj leći
  - Pitanja privatnosti i društvene prihvatljivosti
  - Veliki tehnički izazovi
    - Rezolucija
    - Napajanje



# Programiranje proširene stvarnosti – alati

- Vuforia
  - Prepoznaće veliki broj objekata uključujući okvire, cilindre i igračke, kao i slika
  - Podržava prepoznavanje teksta uključujući oko 100.000 riječi
  - Omogućuje stvaranje prilagođenih VuMarks, koji izgledaju bolje od tipičnog QR-koda
  - Omogućuje stvaranje 3D geometrijske karte bilo kojeg okruženja pomoću značajke Smart terena
- ARToolkit
  - Podrška za Unity3D i OpenSceneGraph.
  - Podržava i jednu i dvije kamere.
  - Podrška za GPS i kompase za stvaranje AR aplikacija koje se temelje na lokaciji.
- Google ARCore
- Apple ARKit
- Maxst
- Specijalizirani SDK-ovi za HoloLens i Magic Leap

# Virtualna okruženja

Virtualni ljudi

# Virtualni ljudi

- Simulacija ljudi na računalu
- Modeliranje ljudskih likova
  - 3D model
  - Stvaranje modela
- Animacija



[izvor]

# Realizam virtualnih likova

FOTO-REALISTIČNI

ČOVJEKOLIKI

MASKOTE

SIMBOLIČNI



[\[izvor\]](#)



[\[izvor\]](#)



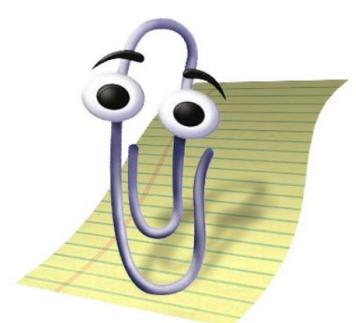
[\[izvor\]](#)



[\[izvor\]](#)



[\[izvor\]](#)



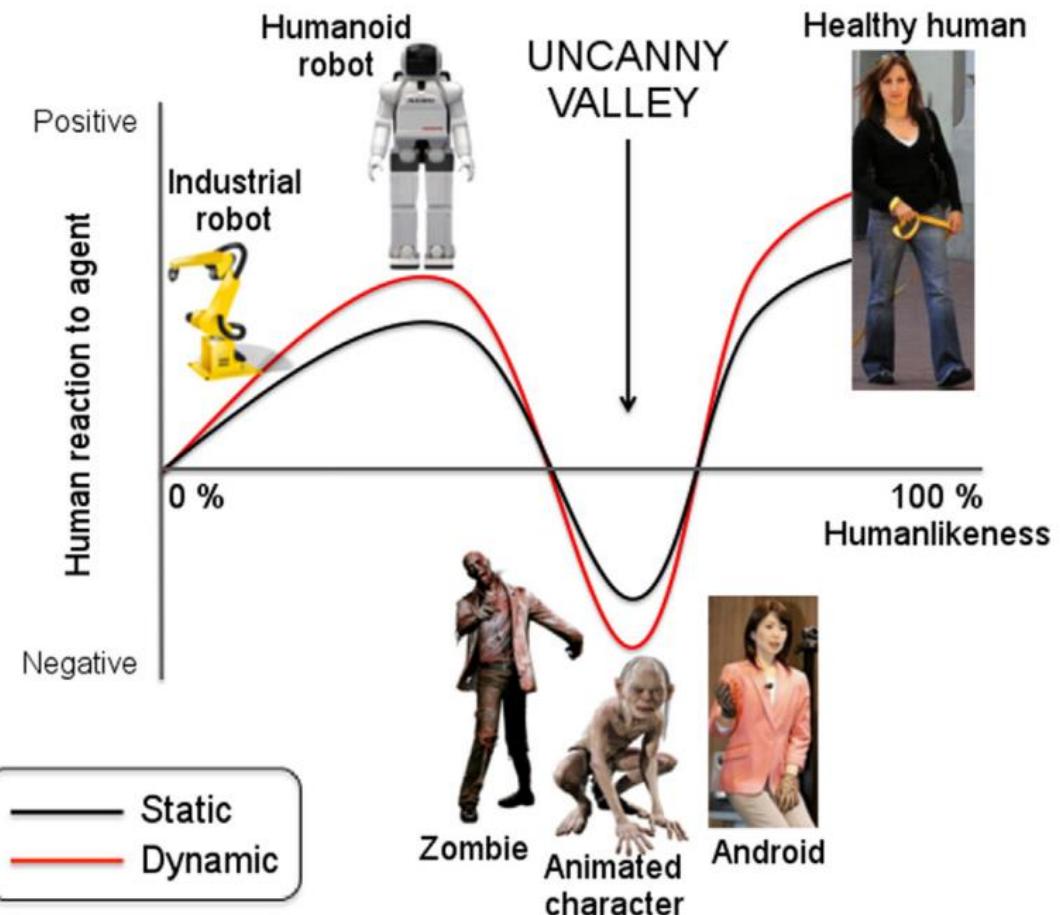
[\[izvor\]](#)

- Foto-realizam je vrlo teško postići
  - Statične slike je lakše učiniti realističnima, dok je kod animacije riječ o većem izazovu
  - Ovisno o aplikaciji, foto-realizam je često nepotreban, a ponekad i zastrašujuć

# Realizam virtualnih likova

- „*Uncanny valley*“ ili jeziva dolina
  - Humanoidni objekti koji su vrlo slični pravim ljudima, ali kod kojih potpuni realizam nije postignut, u promatračima mogu izazvati negativne osjećaje (jeza, odbojnost)
  - Npr. beba Billy u Pixarovom kratkometražnom filmu *Tin Toy* (1988)

*“Pixar took a lesson from ‘Tin Toy.’ We have to nail the human form or not even go there.”*  
Thalia Wheatley

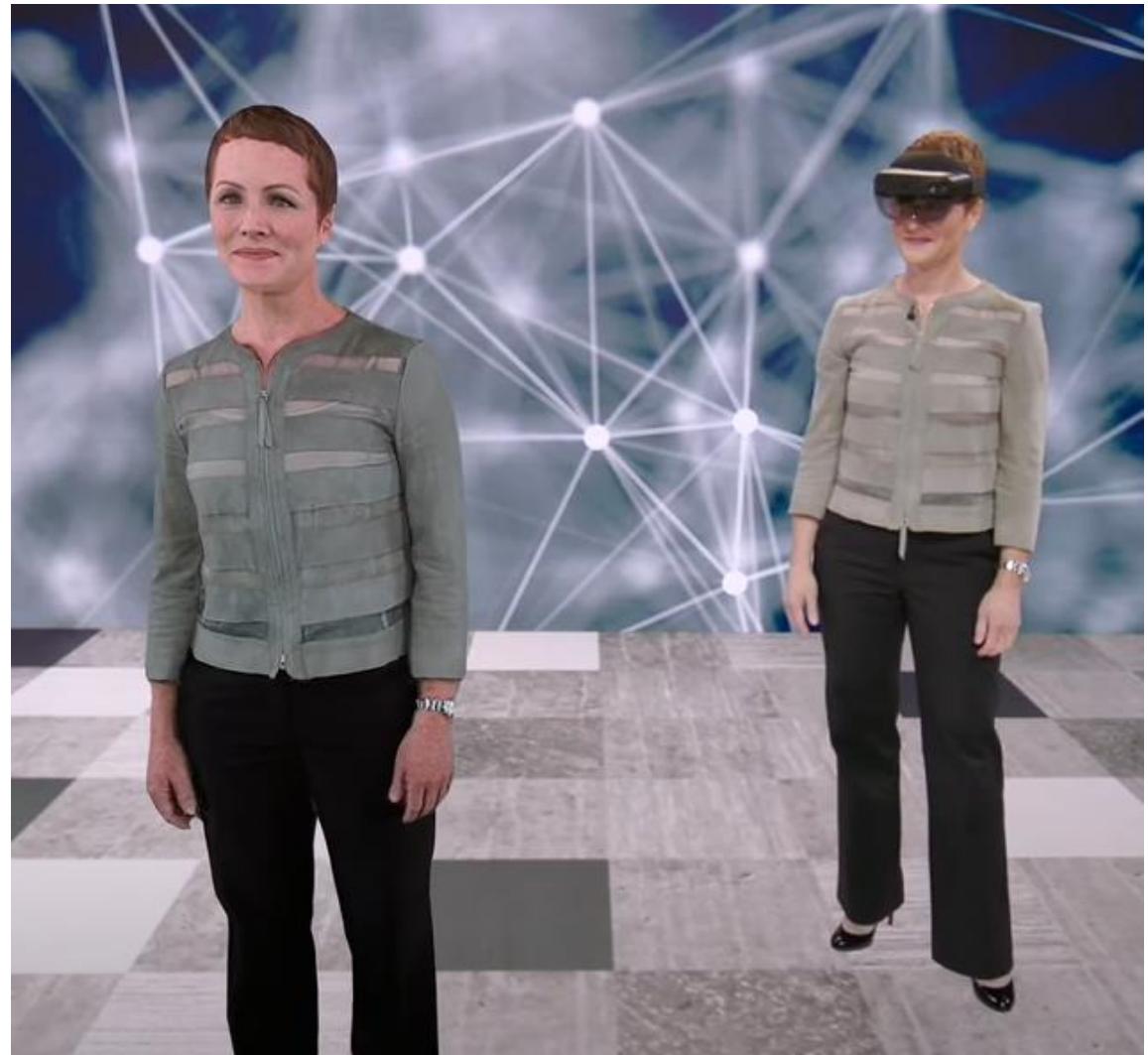


Urgen, B. A., Kutas, M., & Saygin, A. P. (2018). Uncanny valley as a window into predictive processing in the social brain. *Neuropsychologia*, 114, 181–185.

# Primjene virtualnih ljudi

- Filmove, TV, igre
- Ergonomija
- Komunikacije
  - Marketing
  - Osobne poruke/zabava
  - Broadcasting (npr. vijesti)
  - Servisi, prodaja, podrška korisnicima...
- Učenje
  - Podučavanje djece s teškoćama
  - Trening, korporativna komunikacija
- Utjelovljeni razgovorni agenti
  - Korisnik razgovara s računalom

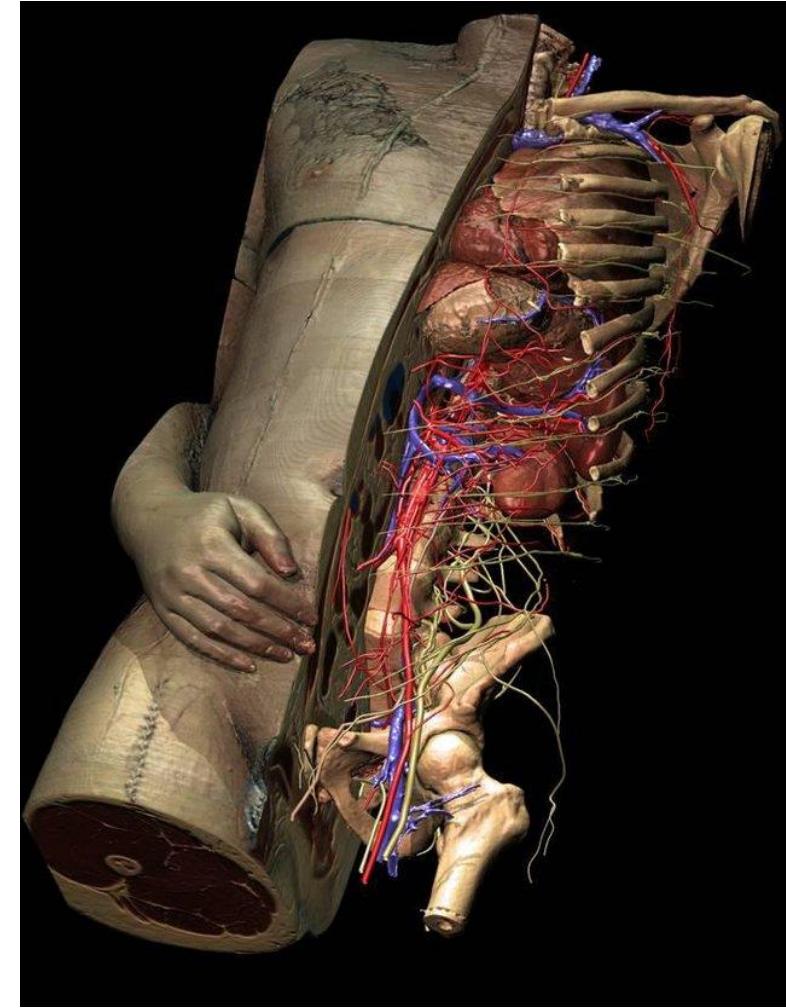
*Microsoftova implementacija virtualnog prevodioca korištenjem tehnologija umjetne inteligencije i volumetrijskog prikaza*



[\[originalni video\]](#)

# 3D model

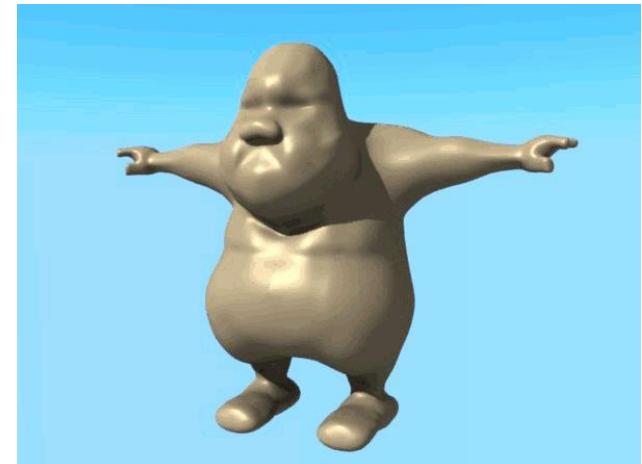
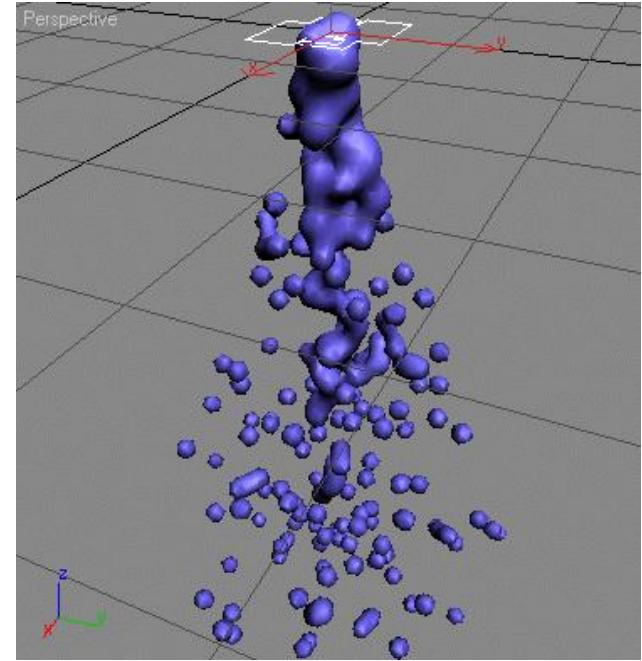
- Služi za definiciju oblika i izgleda
- Ljudsko tijelo je izuzetno složeno
  - Kosti, mišići, živci, žile, žljezde, masno tkivo, vezivno tkivo, koža...
- Za sada ne postoji jedinstveni i potpuni model
- Zahtjevi na model ovise o primjeni
  - Medicinske primjene zahtijevaju visoku preciznost i razinu detalja
  - U animaciji se mogu koristiti aproksimacije
- Pri modeliranju treba voditi računa o animaciji
  - Mogućnost otvaranja usta, okretanja očiju itd.



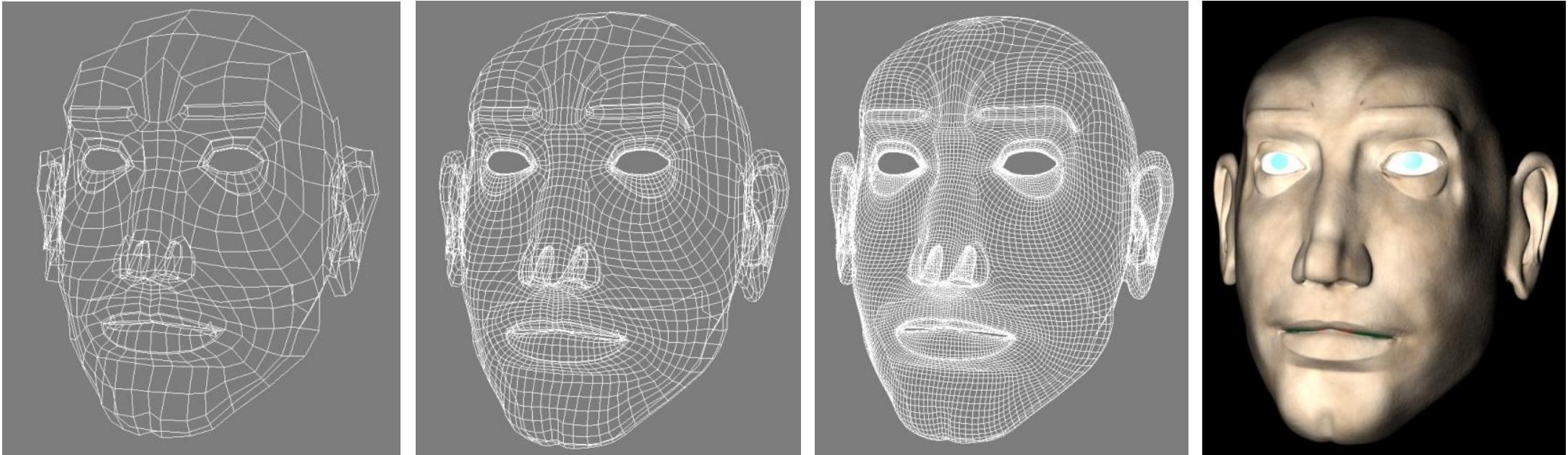
Hacker, Silke, and Heinz Handels. "Representation and visualization of variability in a 3D anatomical atlas using the kidney as an example." Medical Imaging 2006: Visualization, Image-Guided Procedures, and Display. Vol. 6141. International Society for Optics and Photonics, 2006.

# Parametarske plohe

- Stapajuće plohe: mekani predmeti, „metaballs”
  - Računski zahtjevne, složena manipulacija
  - Ponekad u uporabi za modele tijela jer mogu dobro prikazati zaobljenost tijela
- Splines, pogotovo NURBS
  - Dobro izražavaju glatke, zaobljene površine
  - Relativno jednostavno modeliranje i manipulacija
  - Danas se primarno koriste za modeliranje objekata s tvrdim površinama, a ne za modeliranje likova



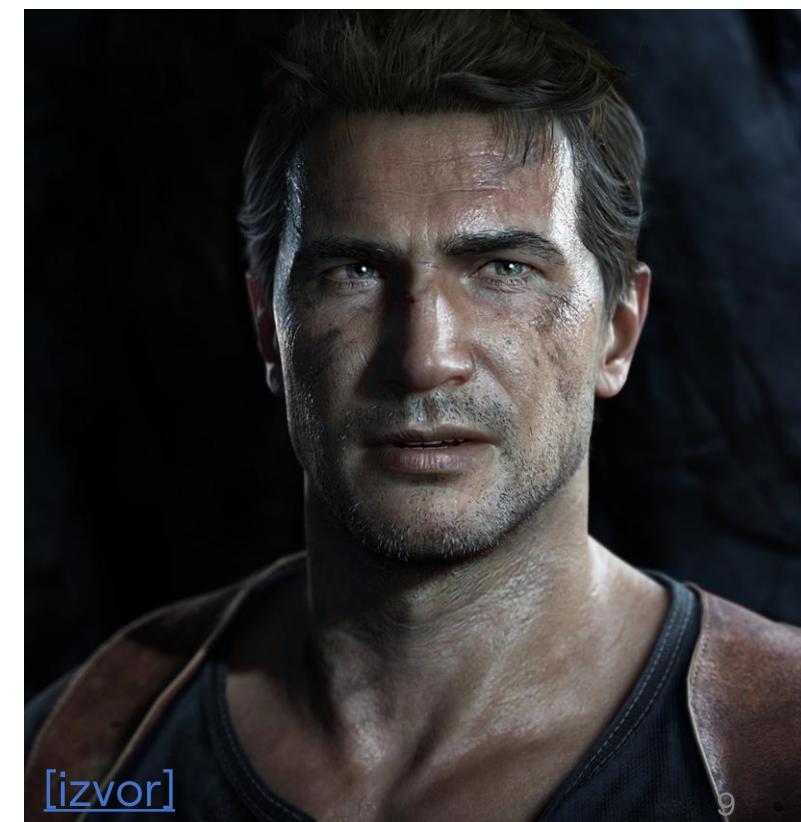
# Razdjelne plohe



- Omogućuju jednostavnu kontrolu glatkoće dijelova modela
- Kontroliranjem je moguće dobiti oštре ili glatke rubove dijelova modela

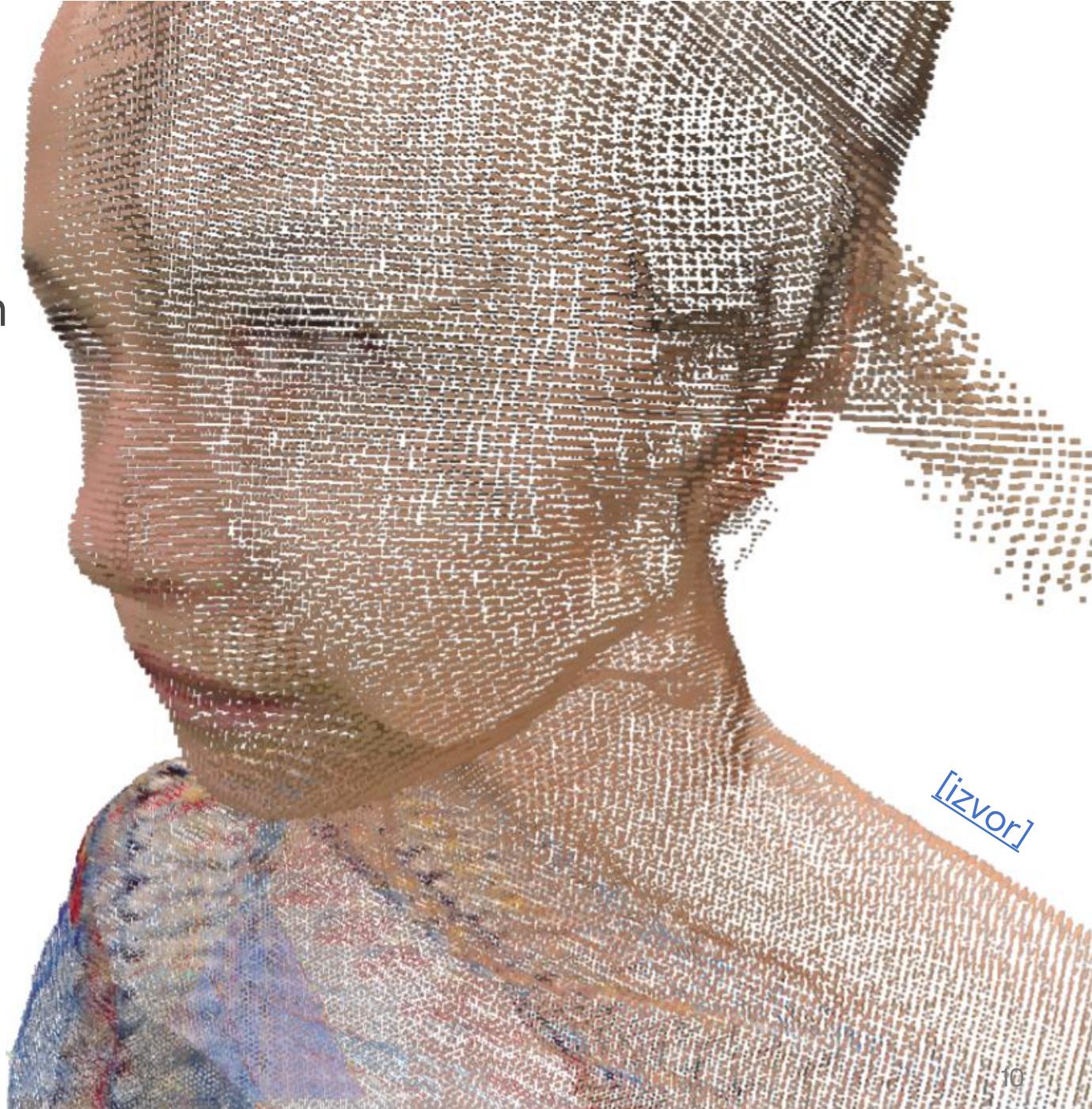
# Mreže poligona

- Kao i u ostalim područjima grafike, tako i u modeliranju ljudskih likova, poligoni su izuzetno popularni zbog univerzalnosti
- Najzastupljenija vrsta prikaza
- Ogroman broj modela lica
- Za tijelo se koristi i graf scene koji definira hijerarhiju dijelova tijela



# Oblaci točaka

- Skup točaka u trodimenzionalnom prostoru
- Točke su definirane svojim koordinatama, bez topoloških podataka
- Najčešće nastaju 3D skeniranjem
- Rekonstrukcija površine (engl. *surface reconstruction*)
  - Proces kojim se oblak točaka može pretvoriti u drugi format (npr. mreža poligona, NURBS)

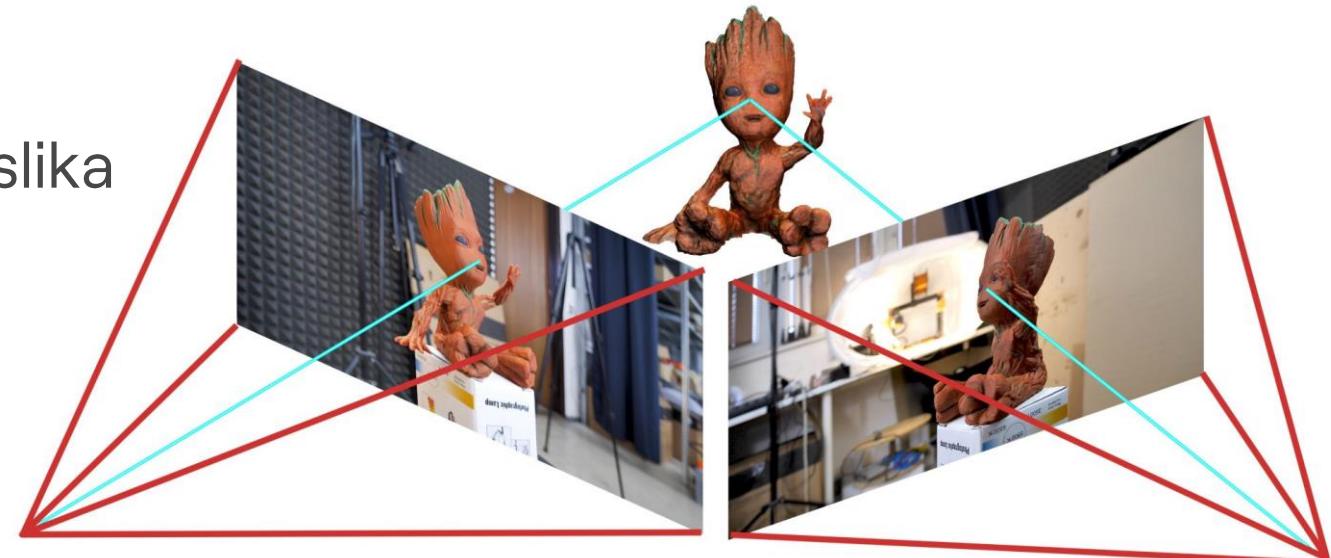


# Modeliranje ljudskog lica

- Ručno digitaliziranje
- Fotogrametrija
- Lasersko skaniranje
- Ručna izrada
- Modifikacija postojećih modela

# Fotogrametrija

- Razne metode dobivanja 3D oblika iz 2D slika
- Često se koriste dvije ortogonalne slike u kojima se identificiraju 3D točke i koordinate se zatim prenose na računalo
- Metode:
  - Poluautomatska obrada slika
  - Strukturirano svjetlo



[izvor]

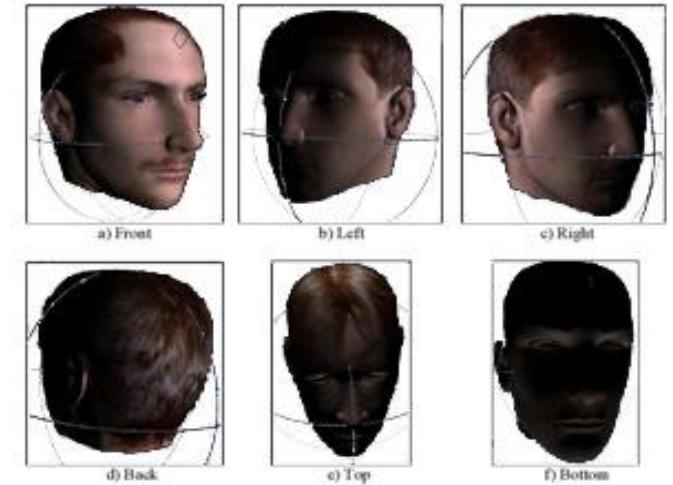
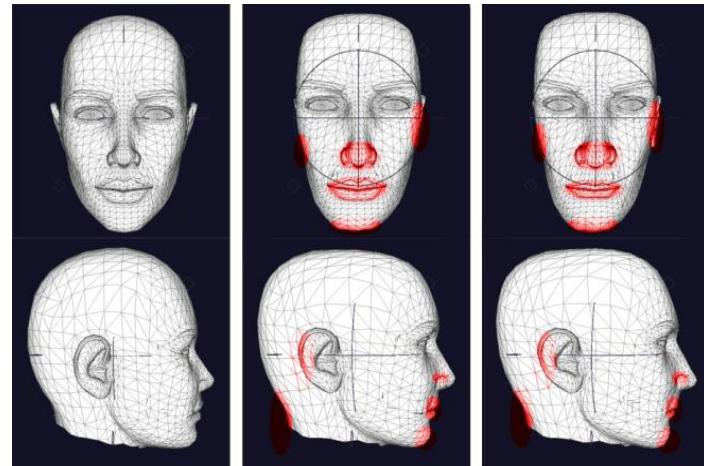
# Fotogrametrija

- Za potrebe igara i filmova osobe i predmeti snimaju se u posebnim fotogrametrijskim studijima koji sadrže velik broj (nekoliko desetaka ili čak više od stotinu) sinkroniziranih kamera, postavljenih u pravilnim razmacima oko snimanog objekta



# Poluautomatska obrada slika

- Automatski ili ručno se biraju karakteristične točke na licu
- Točke se prenose na univerzalni model lica koji se time deformira; dodaje se tekstura



# Automatske korespondencije

- Prepoznatljivi uzorci se identificiraju u 2 ili više slika (ili videozapisu) i koriste za korespondenciju



# Strukturirana svjetlost

- Projektor projicira na predmet
- Iz svake crte dobija se krivulja koja predstavlja profil
- Iz krivulja se određuje reljef predmeta
- Mnoge suvremene dubinske kamere (iPhone X) umjesto crta na objekt projiciraju unikatni uzorak točkica, ali u infracrvenom spektru kako ne bi bile vidljive golim okom



[izvor]

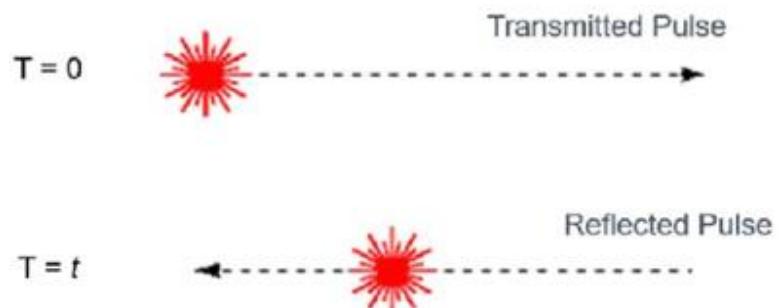
# Lasersko skeniranje

- Pojavljuje se ranih 90tih (tvrtka Cyberware)
- Danas vrlo velik izbor uređaja, od ručnih skenera do skenerskih kabina i daljinskih skenera za velike objekte
- Vrlo gusti i detaljni podaci (dubina i boja)

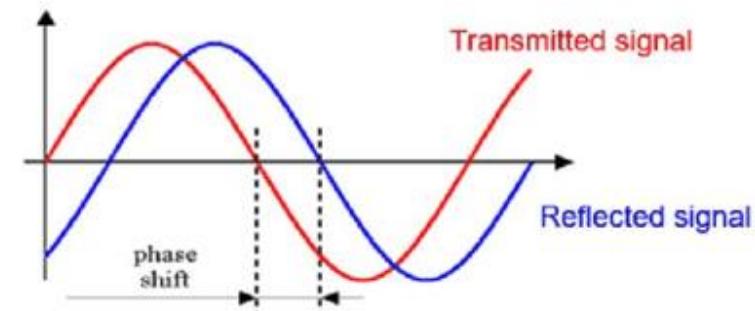


# Lasersko skeniranje – tehnologije

- **Time-of-flight (ToF):** određivanje udaljenosti između senzora i objekta mjeranjem vremena koje je proteklo između odašiljanja laserskog signala i njegovog povratka na senzor nakon što se odbio od snimanog objekta
- **Phase-shift:** određivanje udaljenosti između senzora i objekta na temelju faznog pomaka između odaslanog i reflektiranog signala



Time-of-flight



Phase-shift

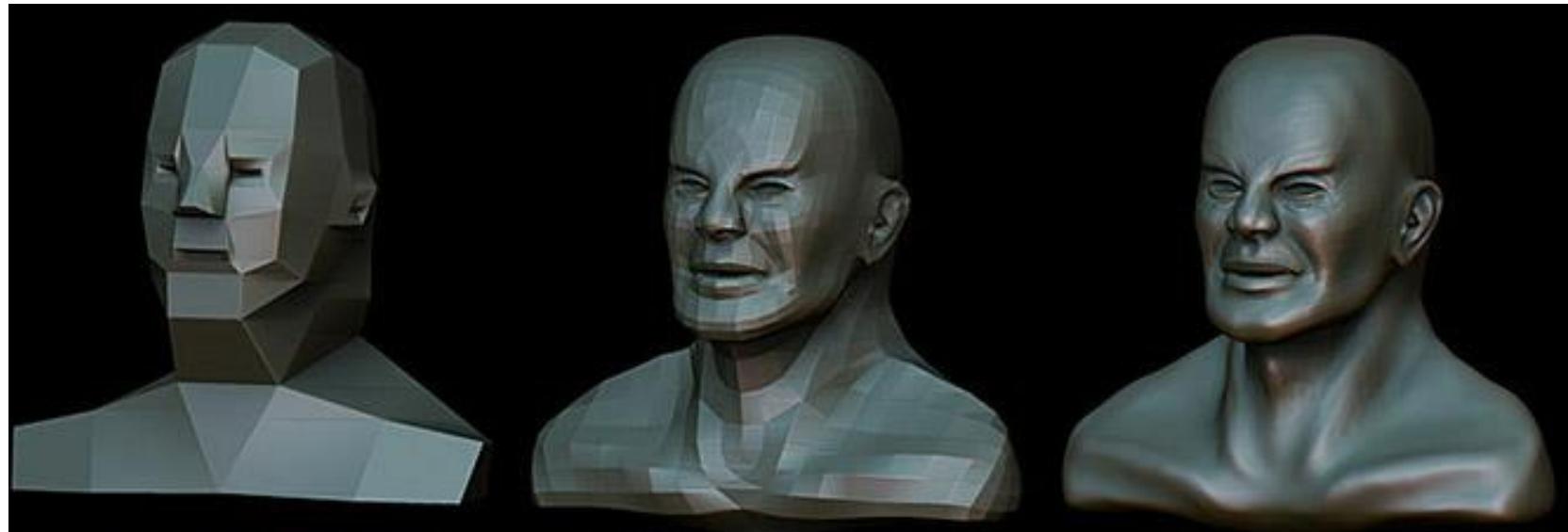
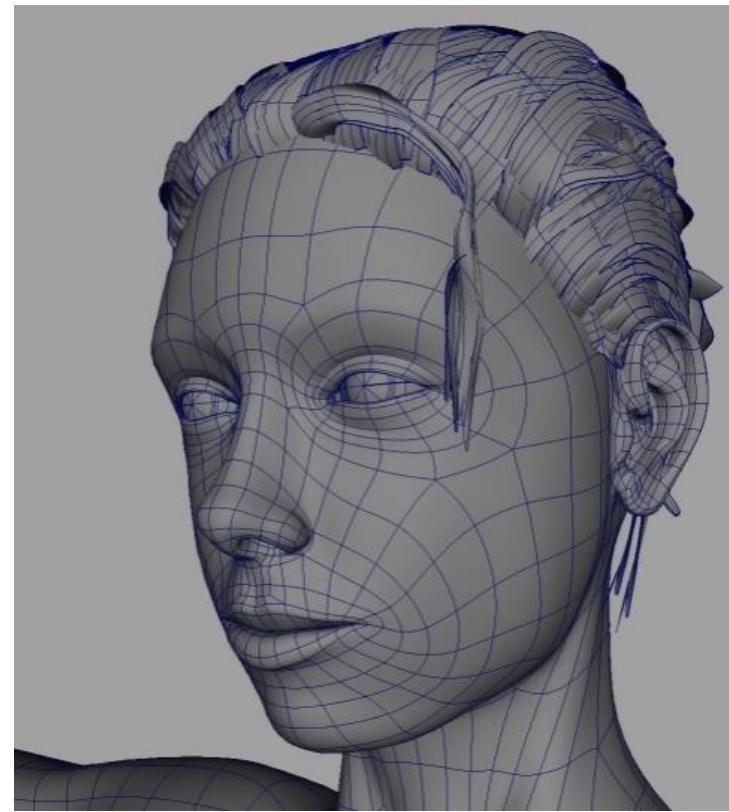
# Lasersko skeniranje

- Disperzija (npr. na kosi) i prekrivenost
  - Dio točaka nedostaje
  - Rješava se interpolacijom
- Oscilacije dobivene površene
  - Potrebno je izglađivanje filtriranjem
- Loši podaci na polovima
- Mrežu točaka je potrebno optimizirati



# Ručna izrada

- Najraširenija metoda
  - Najčešće se koriste mreže poligona
  - Slažu se jednostavni oblici
  - Modeliranje postepenim rafiniranjem

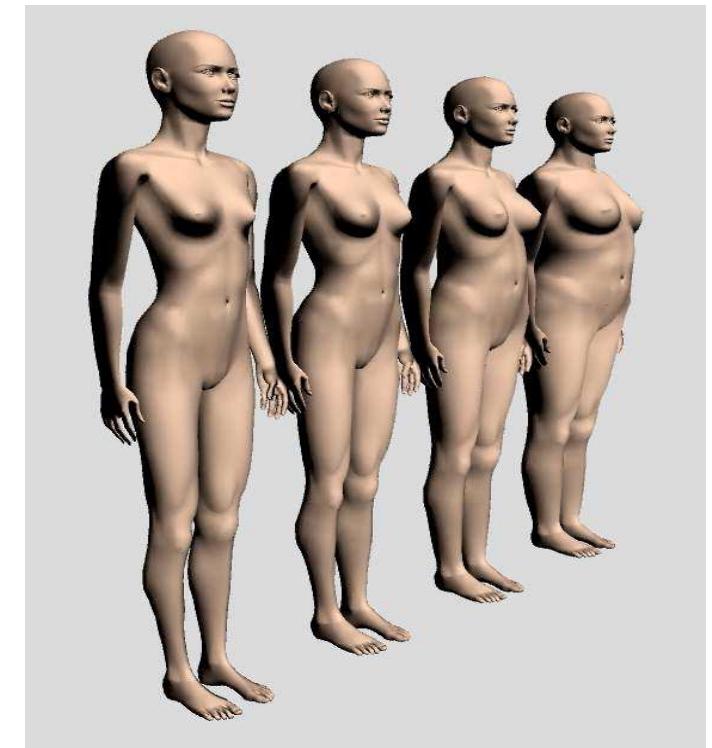
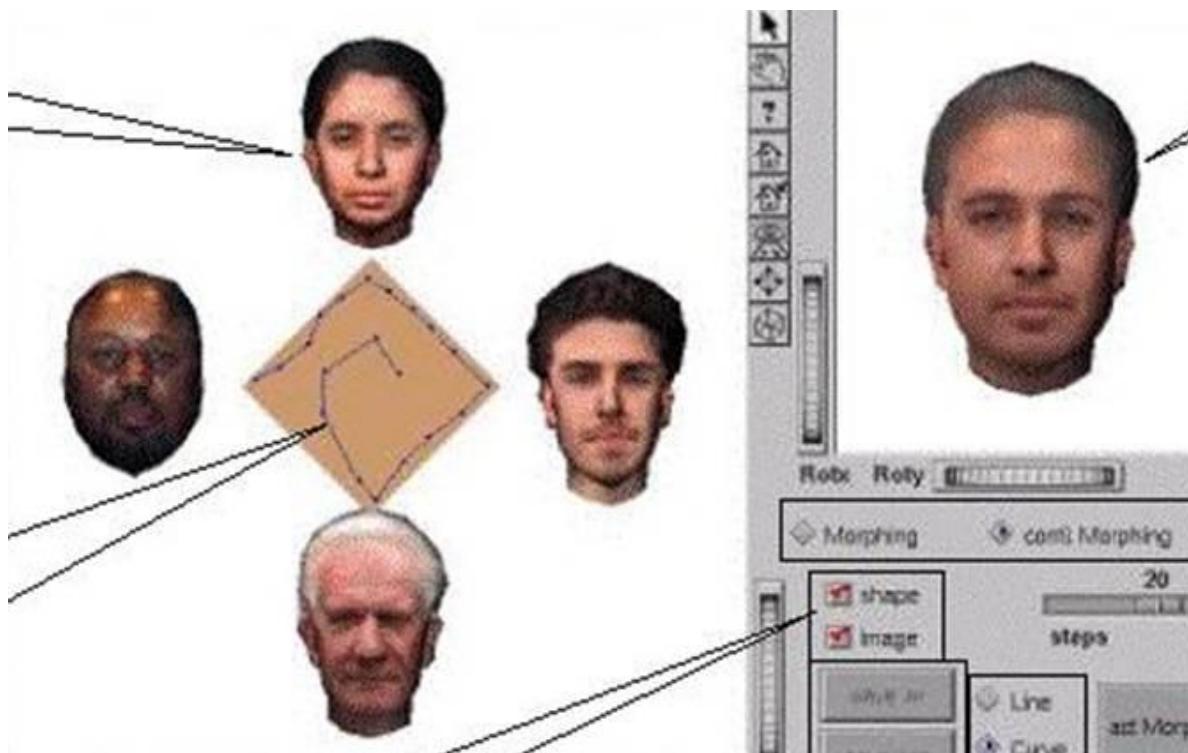


# Modifikacija postojećih modela

- Interpolacija
- Deformacija univerzalnog modela
- Lokalne deformacije
- Statistički modeli populacije

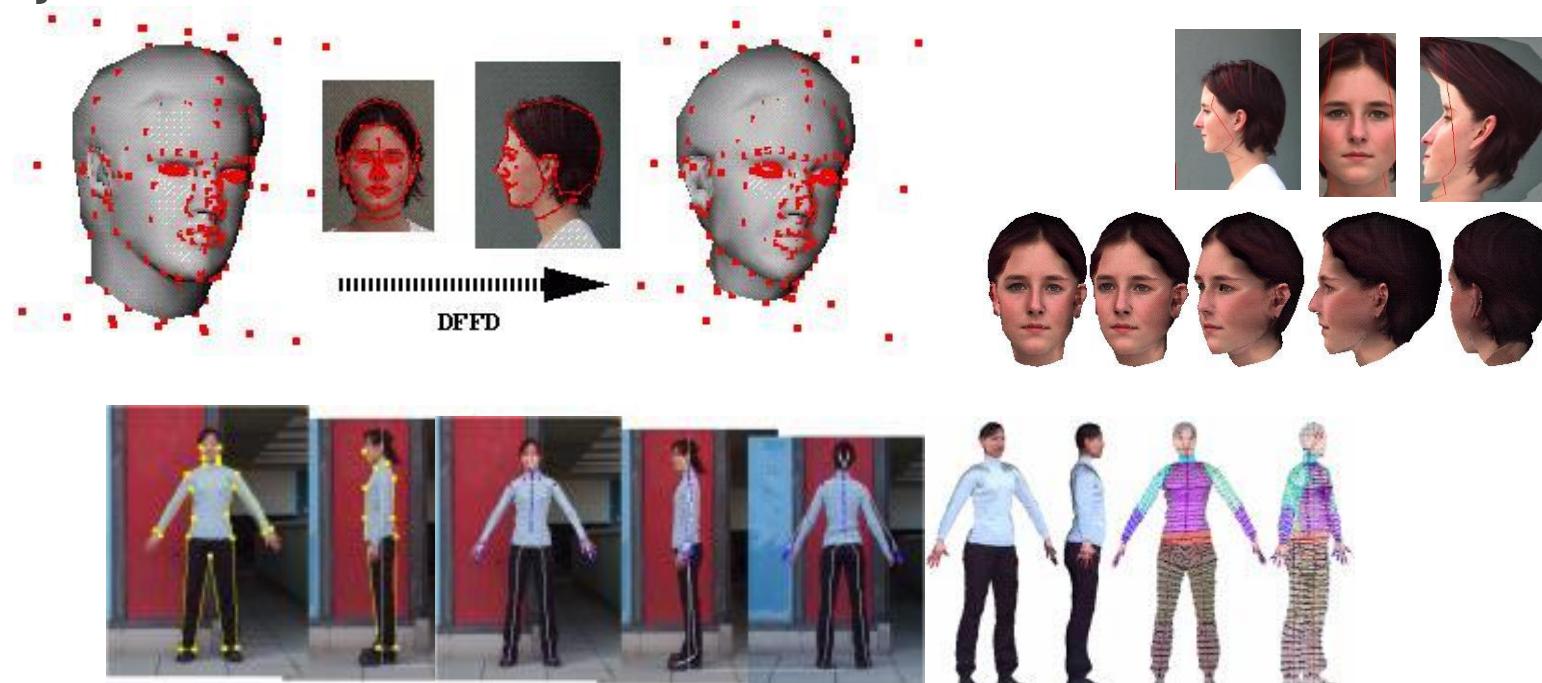
# Interpolacija

- Potrebna je jednaka topologija modela



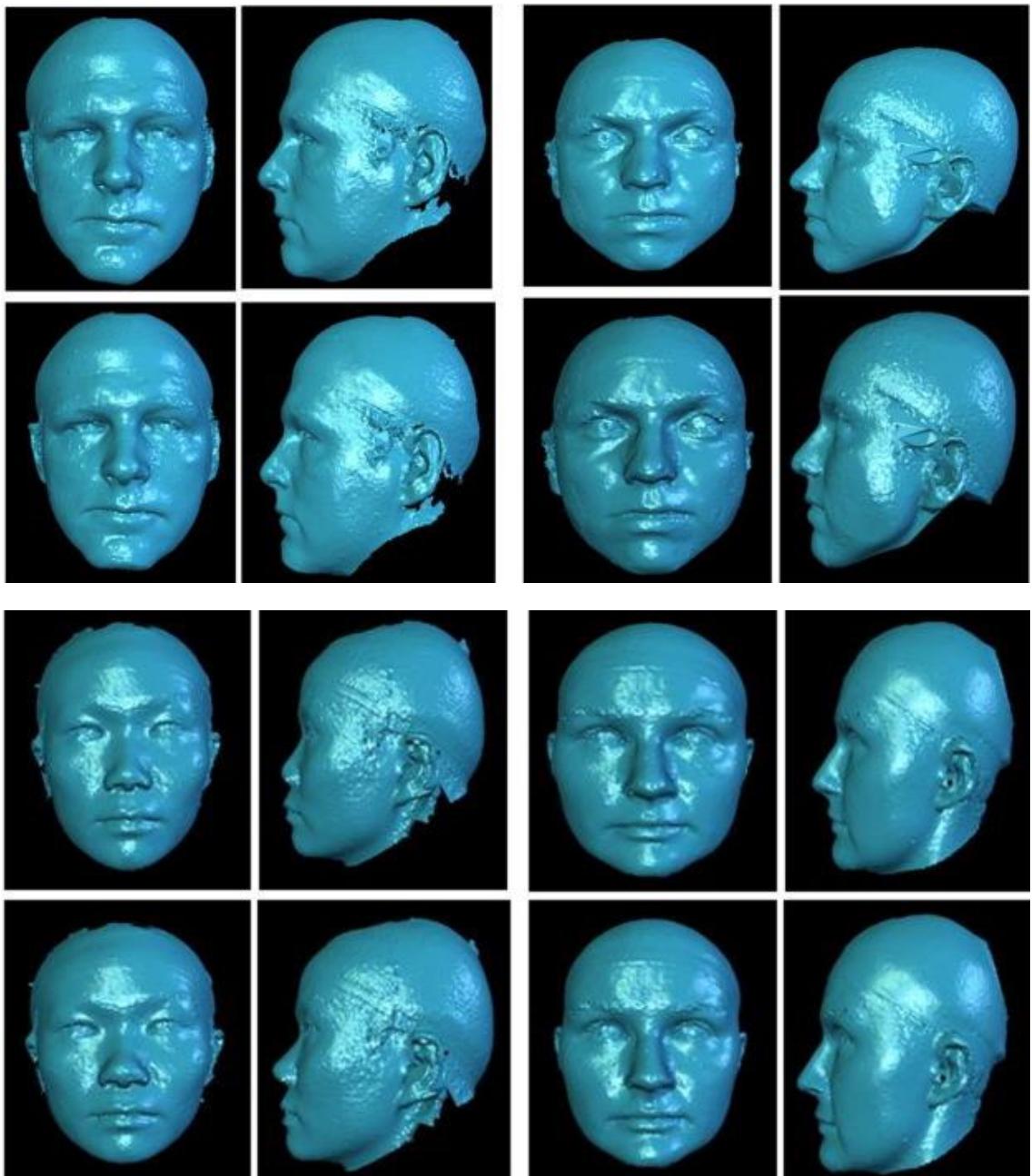
# Deformacija univerzalnog modela

- Univerzalni model je relativno neutralno lice sa definiranim karakterističkim točkama
- Deformacija prema predlošku, tekstura
- Univerzalni model sadrži funkcionalni model: novi model je odmah spremam za animaciju



# Lokalne deformacije

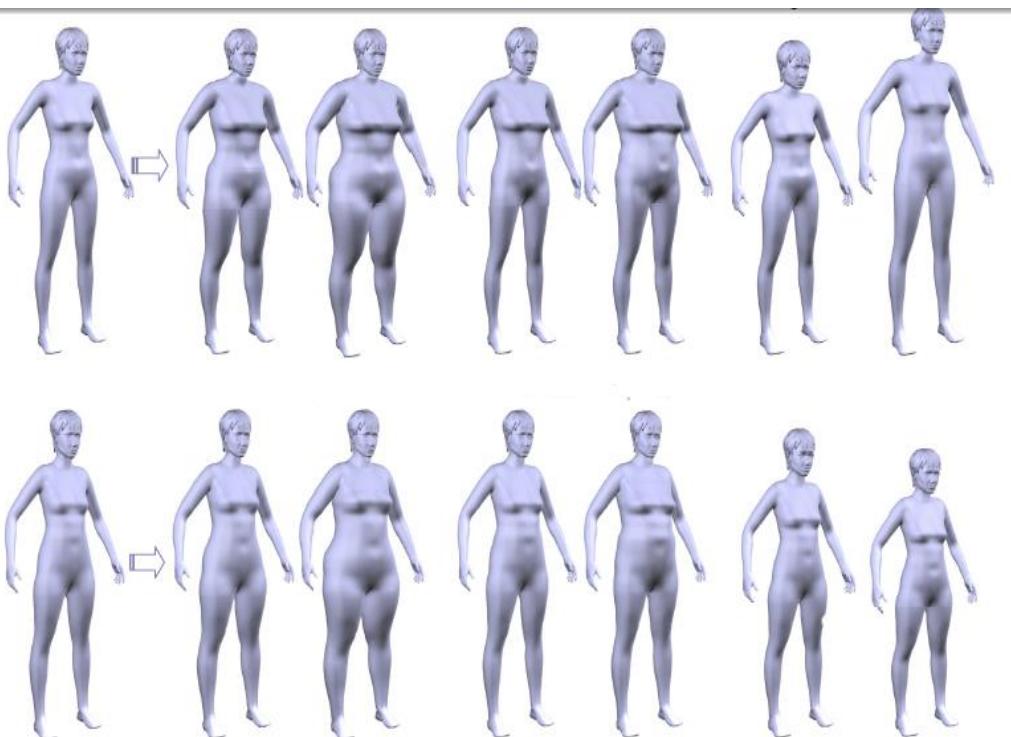
- Selekcija i deformiranje dijela modela
- Koristi se pri ručnom modeliraju
- Razne metode
- Primjer: Free form deformations



Arsalane, Zaghili, Majda Aicha, and Oufkir Ayat Allah. "3D facial attractiveness enhancement using free form deformation." Journal of King Saud University- Computer and Information Sciences (2020)..

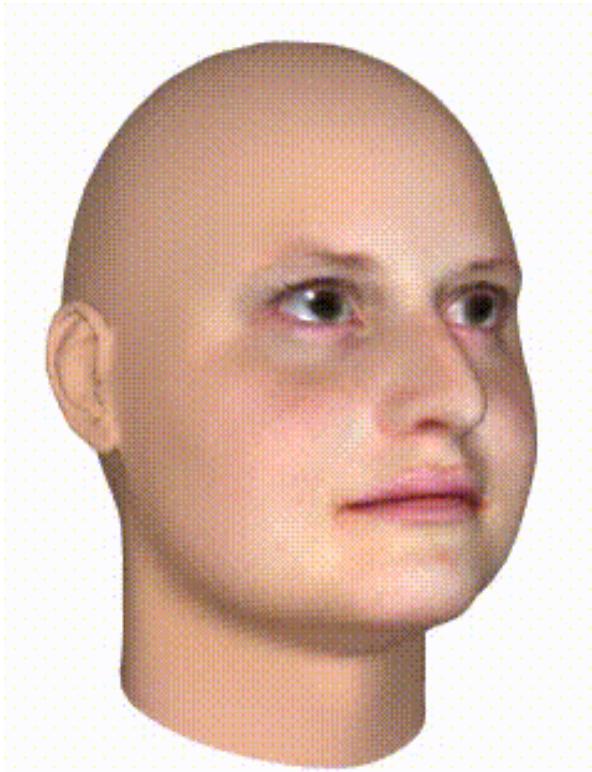
# Statistički modeli populacije

- Počevši od manjeg broja postojećih lica, varijacijom parametara lica dobiva se čitava populacija



# Integrirani alati

- Današnji komercijalni alati koji su dostupni na tržištu objedinjuju predložene metode kako bi se na što lakši način omogućilo modeliranje virtualnih likova



# FaceGen

Easily create realistic 3D human faces

[\[izvor\]](#)

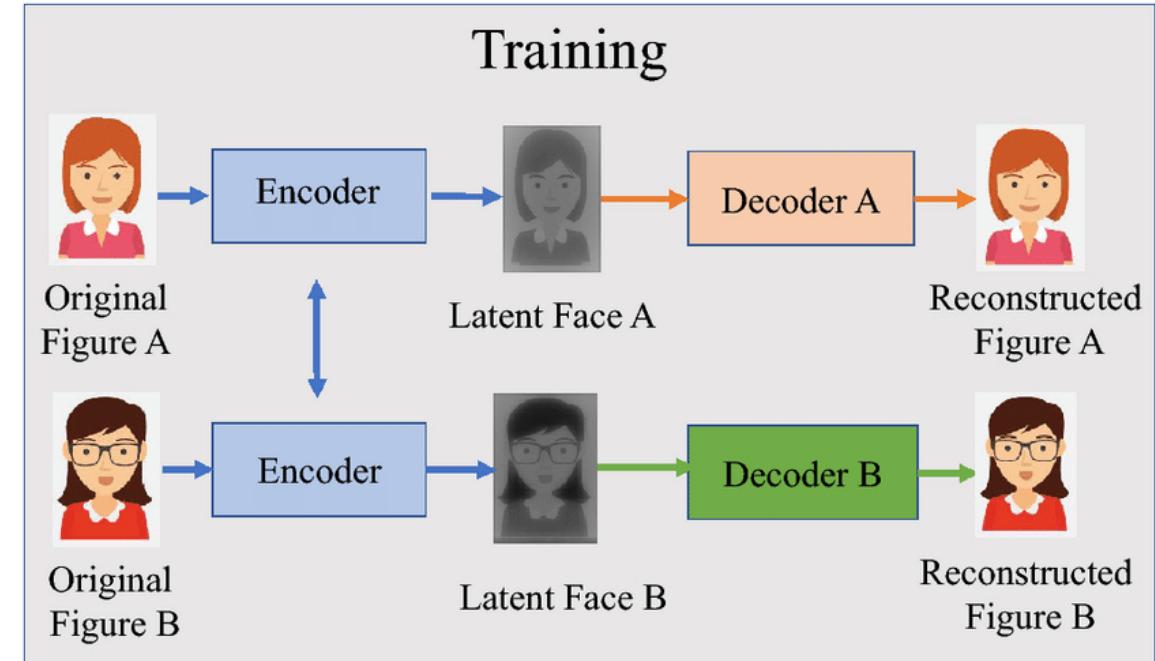
# Deepfakes

- Naziv *deepfake* je kombinacija pojmoveva *deep learning* (duboko učenje) i *fake* (lažno)
- Sintetizirani prikazi koji se najčešće sastoje od lica jedne osobe „zalijepljenog“ preko lica druge osobe
  - pravi *deepfake* podrazumijeva korištenje metoda strojnog učenja

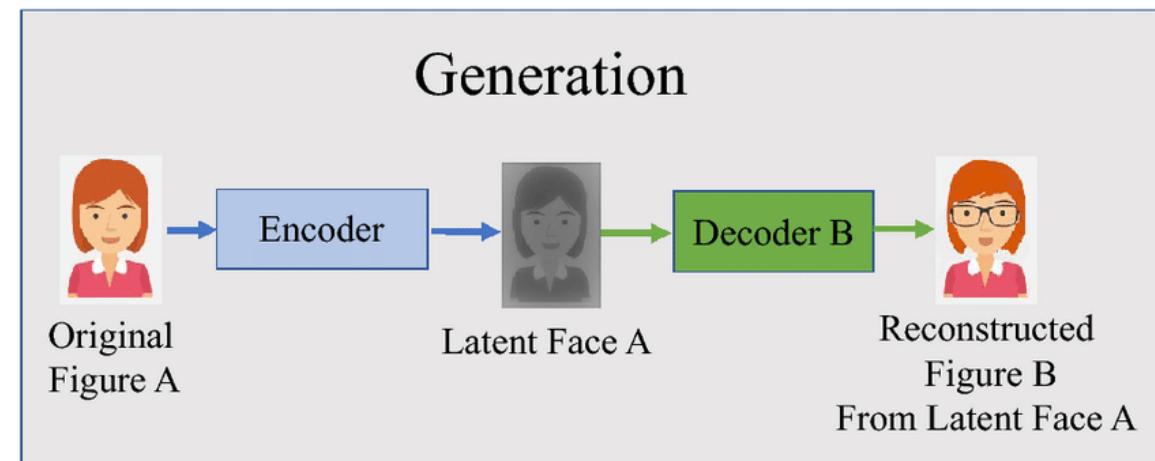


# Deepfakes

- Proces kreiranja *deepfake* sadržaja: treniranje i generiranje
  - Treniranje neuronske mreže na stvarnim snimkama osoba kako bi se postiglo razumijevanje toga kako one izgledaju iz različitih kutova i u različitim uvjetima osvjetljenja
  - Isti enkoder koristi se za obje osobe, na temelju čega nastaju „latentna lica”, zasnovana na specifičnim značajkama
  - Pri generiranju *deepfakea* latentno lice osobe A ulazi na dekoder B



(a) Training Phase



(b) Generation Phase

# Deepfakes



- Problemi s *deepfake* prikazima

- Mogu biti vrlo realistični
- Zbog dostupnosti potrebne tehnologije, može ih kreirati bilo tko
- Ovaj se princip može iskoristiti i za iznimno realistične sintetizirane glasove
- Problemi: krađa identiteta, lažni dokazi, širenje lažnih vijesti, pornografija...
- Kako prepoznati *deepfake*?
  - Postoji li razlika u osvjetljenju, manji broj detalja na određenim dijelovima lica, promotriti odraze u naočalama, podudaranje naušnica...
  - *watermark* i *block chain* tehnologija kao potencijalna rješenja za detekciju sintetiziranog sadržaja

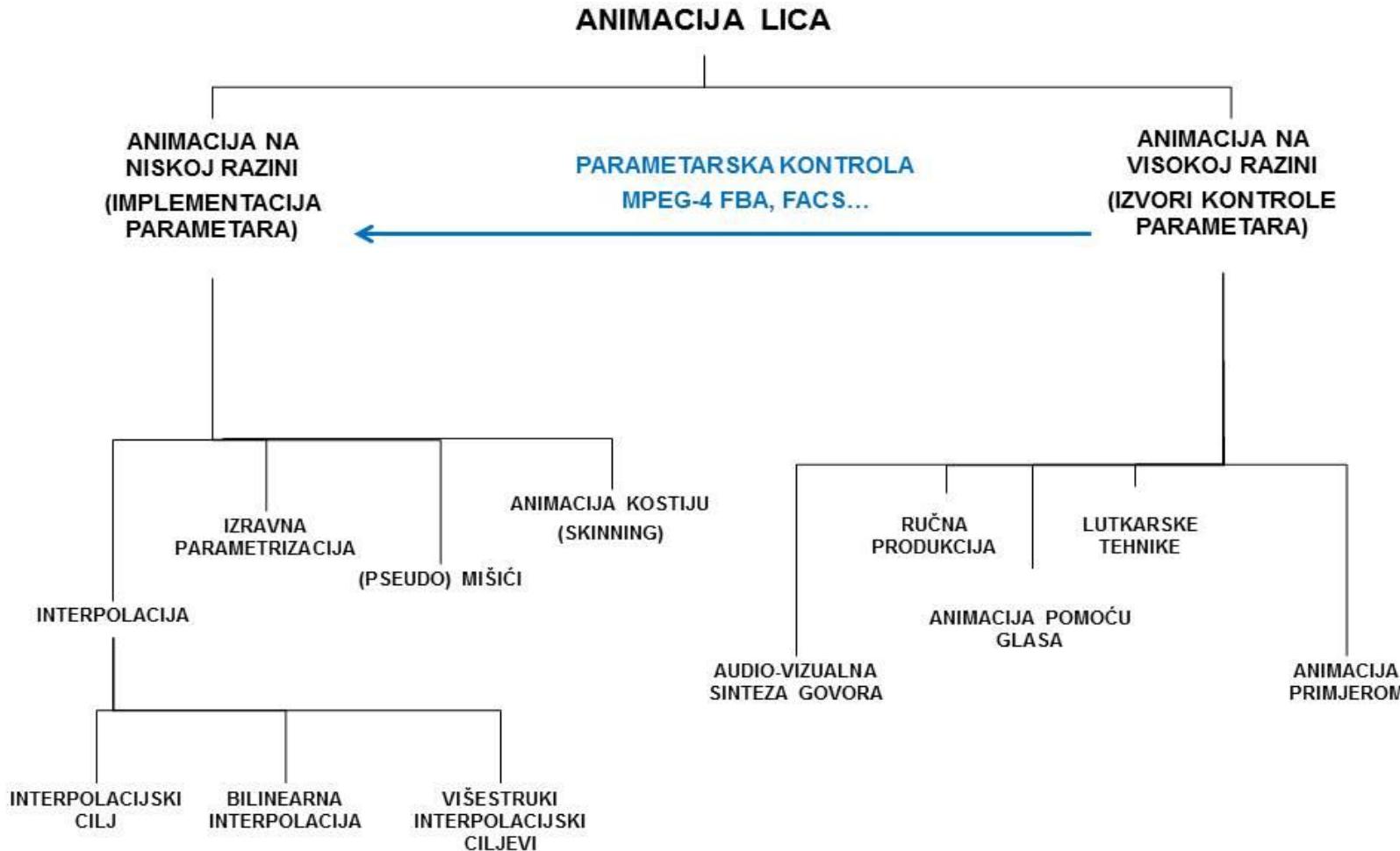
# Animacija

- 3D model je potrebno pripremiti za animaciju
  - Definira se skup parametara koji deformira model, npr. za tijelo se koristi model kostura čije se gibanje prenosi na površinu modela
- Područja animacije virtualnih ljudi u predavanju
  - Animacija lica
  - Animacija tijela
  - Animacija odjeće
  - Animacija kose

# Animacija lica

- Izuzetno složena struktura kostiju, mišića i tkiva
- Animacija lica se dijeli na dva područja
- Animacija na niskoj razini
  - Parametrizacija pokreta lica
  - Algoritmi za pomicanje geometrije lica (otvaranje usta, dizanje obrva, itd.)
  - Bitno je stvoriti dobar set parametara za animaciju
- Animacija na visokoj razini
  - Stvaranje animacijske sekvence
  - Vremenski slijed animacijskih parametara niske razine stvara potpunu animacijsku sekvencu

# Animacija lica



# Animacija lica na niskoj razini

- Razni načini implementacije skupa parametara koji pomiču lice
  - Treba odabratи smislen skup parametara
  - Za svaki parametar, pomak dijela lica
  - Miješanje parametara može biti problem
- Metode
  - Interpolacija
  - Direktna parametrizacija
  - Pseudo-mišići
  - Simulacija mišića

# Interpolacija

- Najjednostavnija i najviše korištena metoda
- Iz zadanih krajnjih položaja lica interpolacijom položaja se stvaraju novi položaji
  - Krajnji položaj = interpolacijski cilj



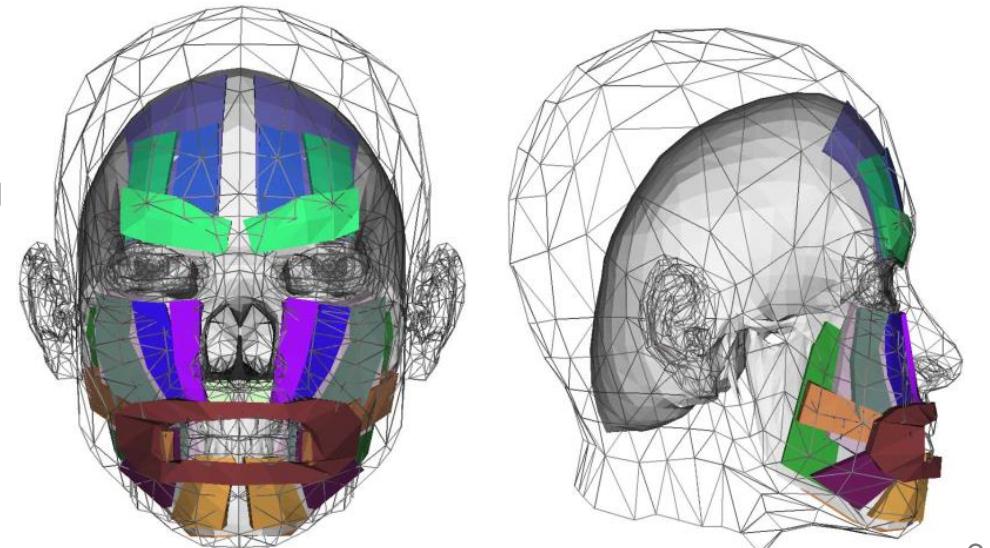
# Interpolacija

- Interpolacijski ciljevi se pripremaju ručno ili automatski
- Bilinearna interpolacija: dobivanje novih međupoložaja uzimanjem središnjih vrijednosti više interpolacijskih ciljeva
- Težinsko kombiniranje više ciljeva može rezultirati nepravilnostima



# Ostale metode

- Izravna parametrizacija: matematički modeli koji oponašaju stvarne pokrete lica, no bez modeliranja biomehanike lica
- Pseudo-mišići
  - Definicija pojedinih područja lica (usta, oči, obrve, itd.) koja se pokreću odgovarajućim parametrima
- Animacija kostiju (skinning)
  - Simulirane „kosti“ se spajaju s modelom

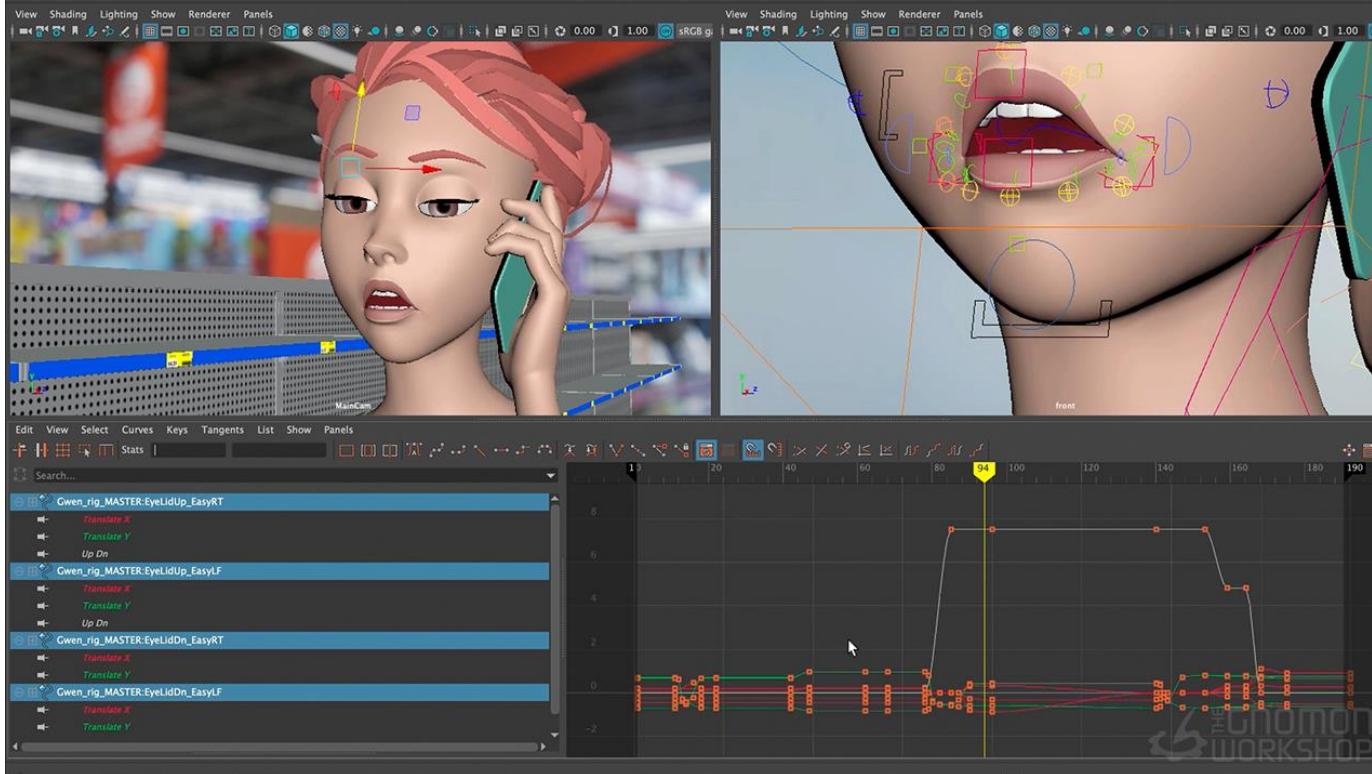


# Animacija lica na visokoj razini

- Izvori slijedova animacijskih parametara
  - Ručna produkcija
  - Animacija iz teksta
  - Animacija pomoću glasa
  - Lutkarske tehnike
  - Animacija primjerom

# Ručna produkcija

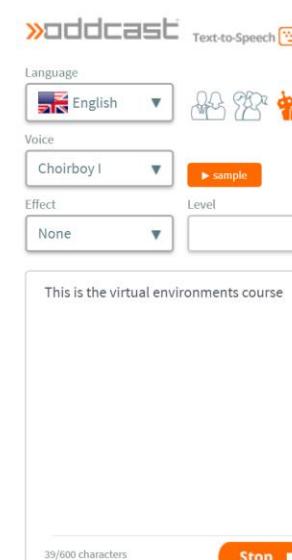
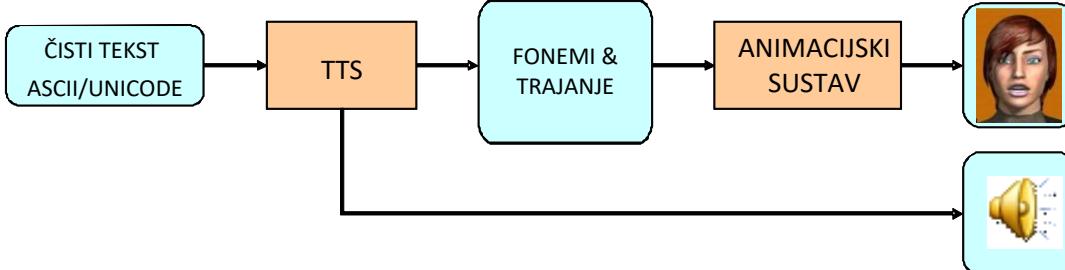
- Najprimitivniji način, najviše vremena
- Vrlo efektivna metoda za završno dotjerivanje



[izvor]

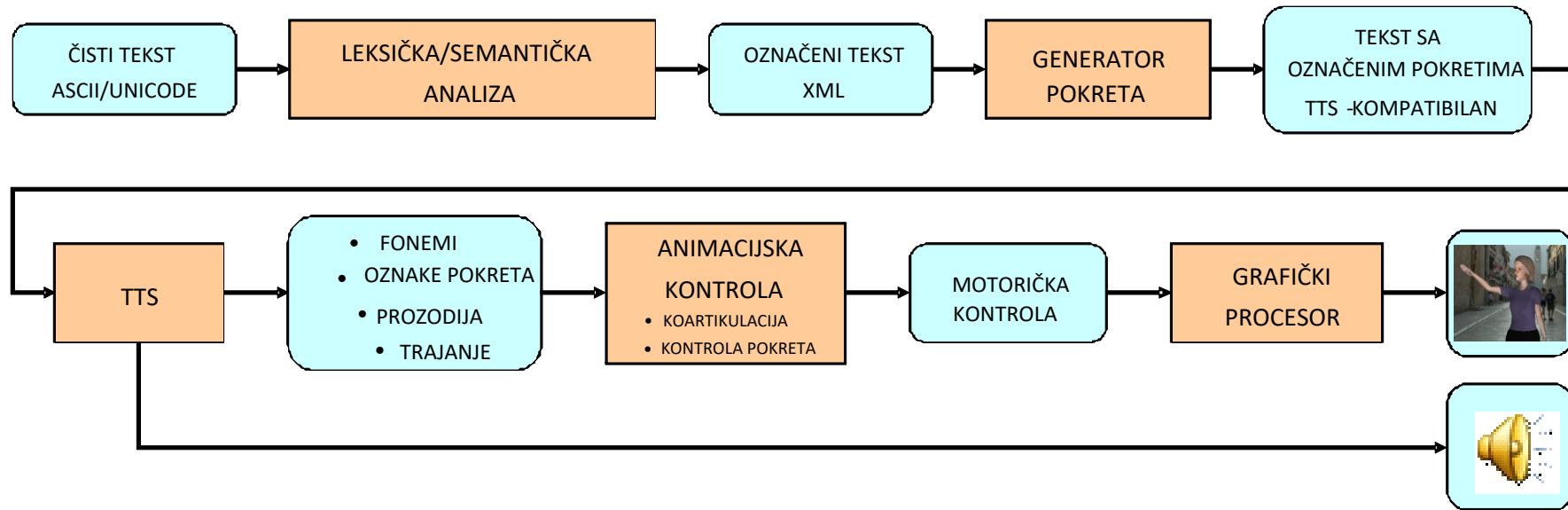
# Audio-vizualna sinteza govora

- Primjer: [link](#)
- Automatska animacija dobivena iz teksta
  - Uz govor, često se animiraju geste lica
- Vizualna sinteza govora (VTTS)
  - Na bazi vremena fonema, uz sintetički govor generira se i animacija usana



# Audio-vizualna sinteza govora

- Potpuna audio-vizualna sinteza
  - Pristup temeljen na pravilima
  - Pristup iz podataka

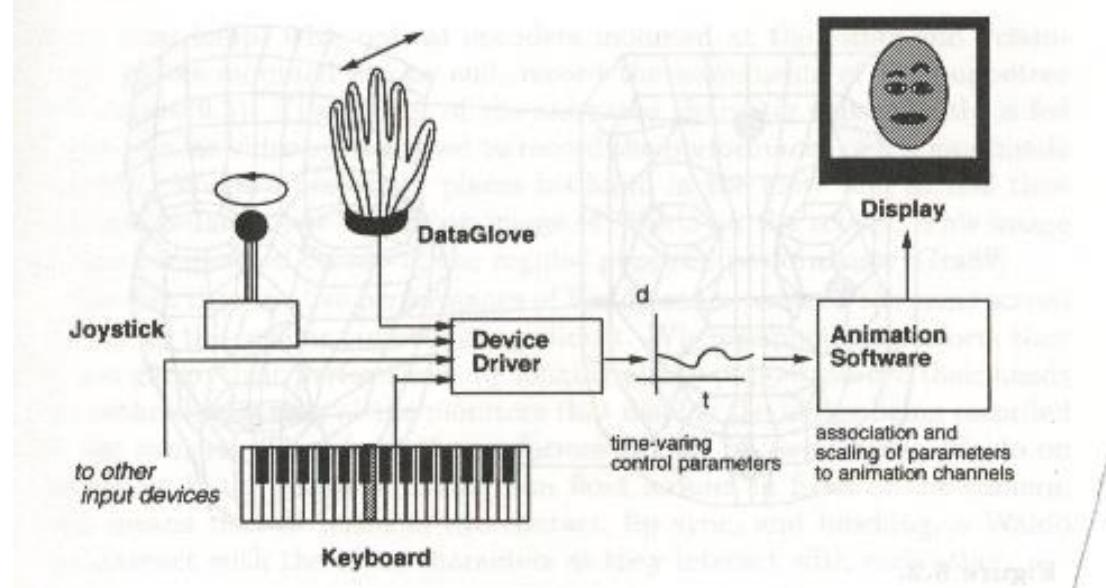


# Animacija pomoću glasa

- Automatska sinkronizacija usana na postojeći glas (engl. lip sync)
- Zasniva se na analizi zvuka
- Primjeri metoda za sinkronizaciju
  - Jednostavno praćenje glasnoće
  - Uporaba metoda za prepoznavanje govora
  - Linearna predikcija

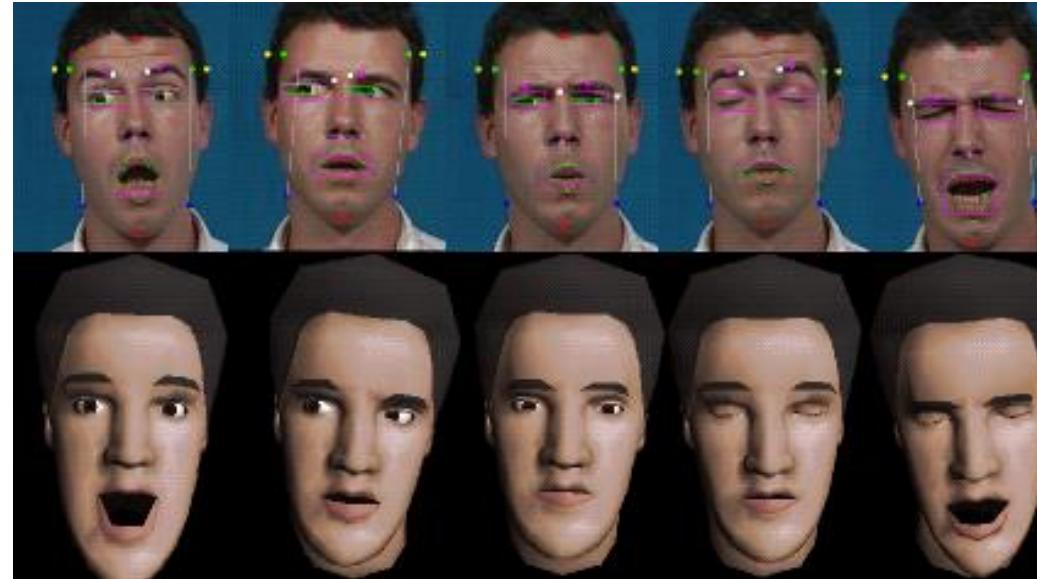
# Lutkarske tehnike

- Klasično se koriste u produkciji filmova
  - Odgovarajućim uređajima lutkom se upravlja na daljinu (engl. *animatronics*)
- Umjesto upravljanja stvarnom lutkom, upravlja se licem na računalu



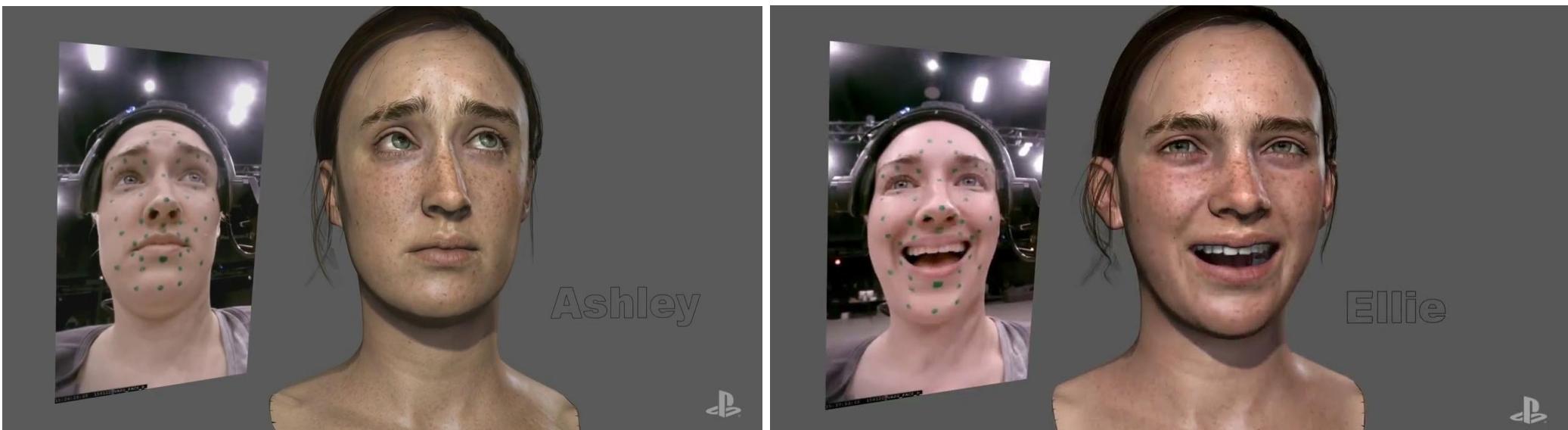
# Animacija primjerom

- Pokreti lica glumca preslikavaju se direktno na virtualno lice
- Razne metode:
  - Slijedjenje pomoću markera
  - Slijedjenje iz čiste video slike



# Slijedjenje s markerima

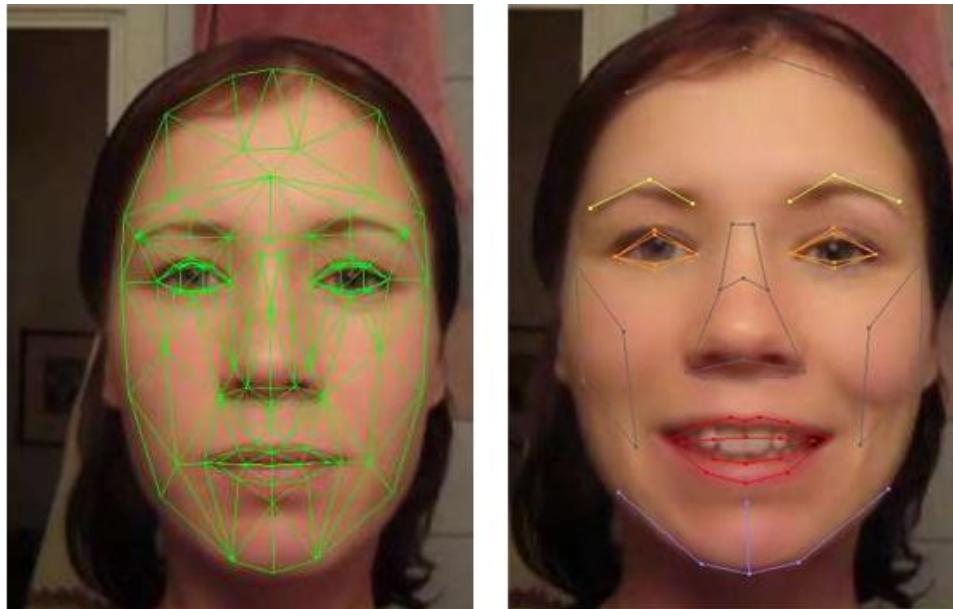
- Obično IR kamere, reflektivni markeri
- Skupo, osjetljivo, složena instalacija
- Rezultati često nisu u stvarnom vremenu
- Rezultati mogu biti vrlo visoke kvalitete



*The Last of Us: Part II - Slijedjenje lica pomoću markera  
[izvorni video]*

# Slijedjenje iz čiste video slike

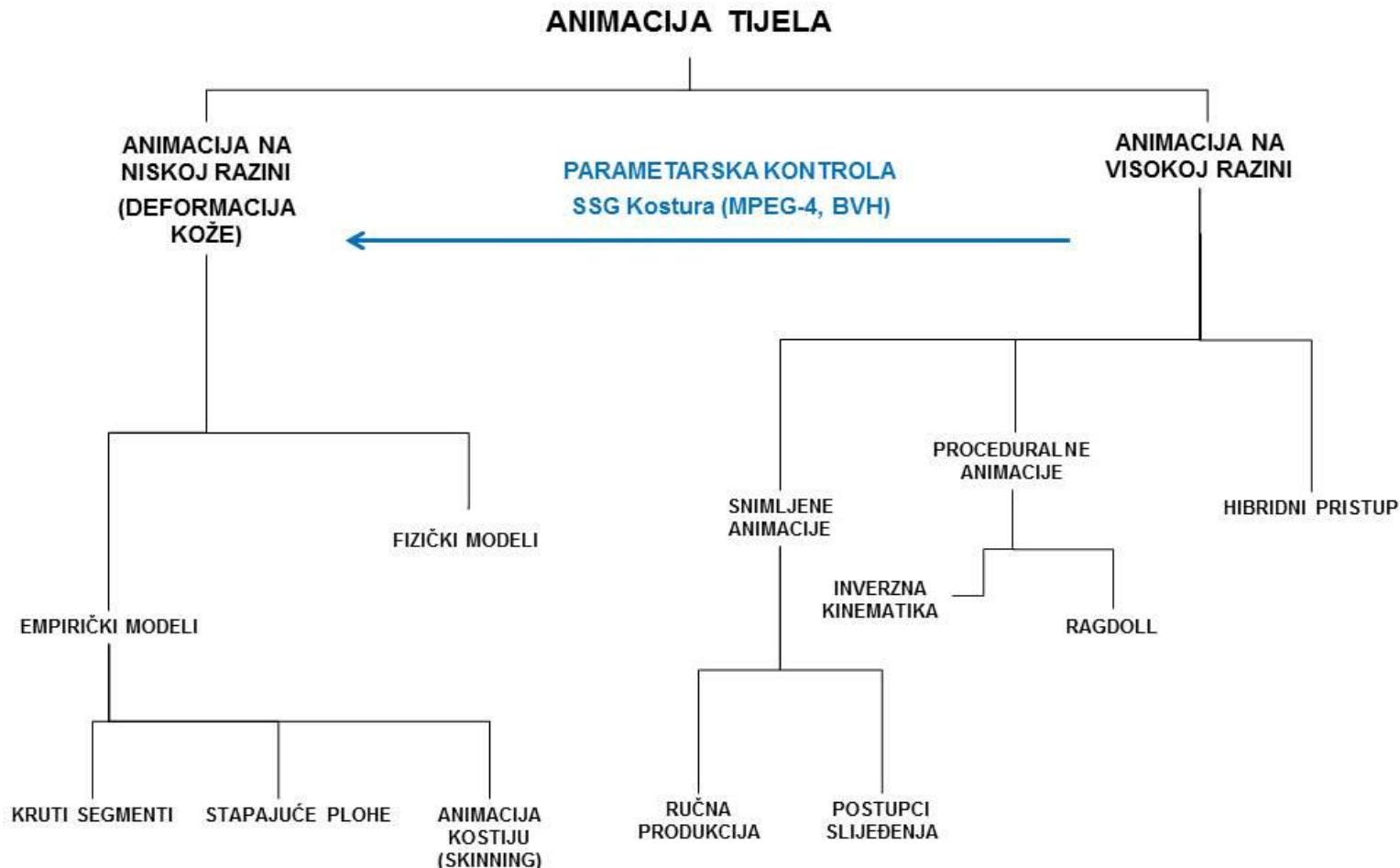
- Slijedjenje karakterističnih točaka lica (engl. *facial feature tracking*)
- Donedavno ideal, danas postoji vrlo dobri sustavi
- Osjetljivost na osvjetljenje, boju kože, bradu, naočale i slično



# Animacija tijela

- Gotovo uvijek se zasniva na pojednostavljenom modelu kostura
- Animacija tijela na niskoj razini
  - Veza između kostiju i kože, parametri koji omogućuju pokretanje geometrije tijela
- Animacija tijela na visokoj razini
  - Stvaranje animacijske sekvence
  - Vremenski slijed animacijskih parametara niske razine stvara potpunu animacijsku sekvencu

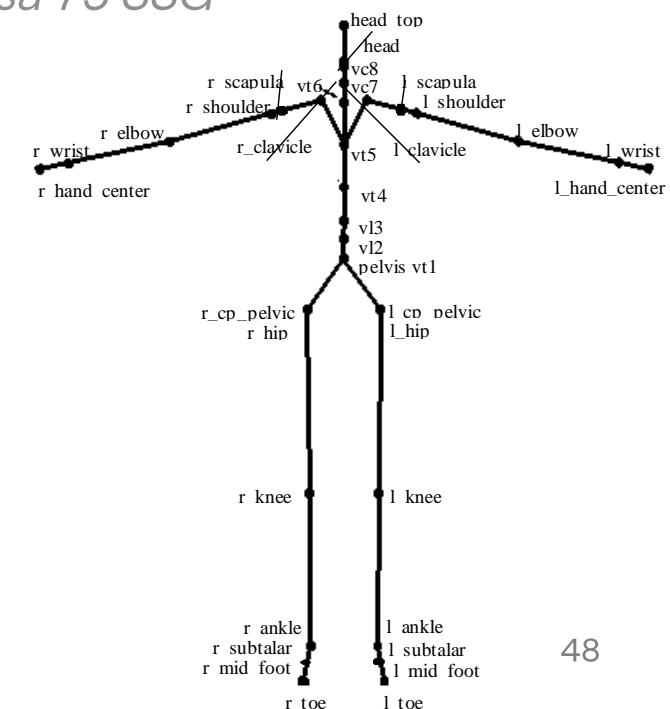
# Animacija tijela



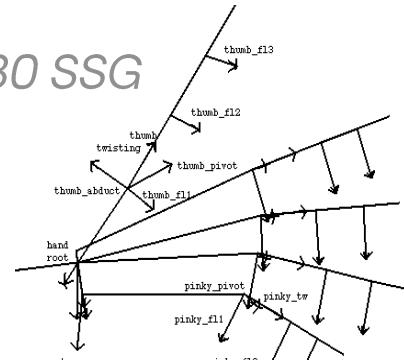
# Kostur

- Kostur služi za upravljanje tijelom
- SSG == stupnjevi slobode gibanja
- Dobra aproksimacija anatomskega kostura: 75 SSG + 2x30 SSG za ruke; uglavnom rotacijski SSG
- Anatomski kostur ima daleko najviše SSG
- Često se koriste i grublje aproksimacije

Kostur sa 75 SSG



Kostur ruke sa 30 SSG



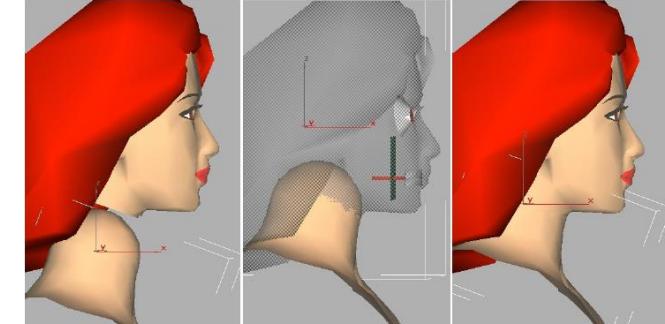
# Animacija tijela na niskoj razini

- Tijelo nije kruto, nego se pri pokretu deformira na vrlo složen način
  - Interakcija kostiju, mišića, masnog tkiva, veznog tkiva, kože
  - Definirani su postupci kojima se pokreti kostura prenose na površinu 3D modela
- Metode
  - Empirički modeli
    - Metoda krutih segmenata
    - Slojeviti model
    - Animacija kostima (skinning)
  - Fizički modeli



[\[izvorni video\]](#)

# Kruti segmenti i slojeviti modeli

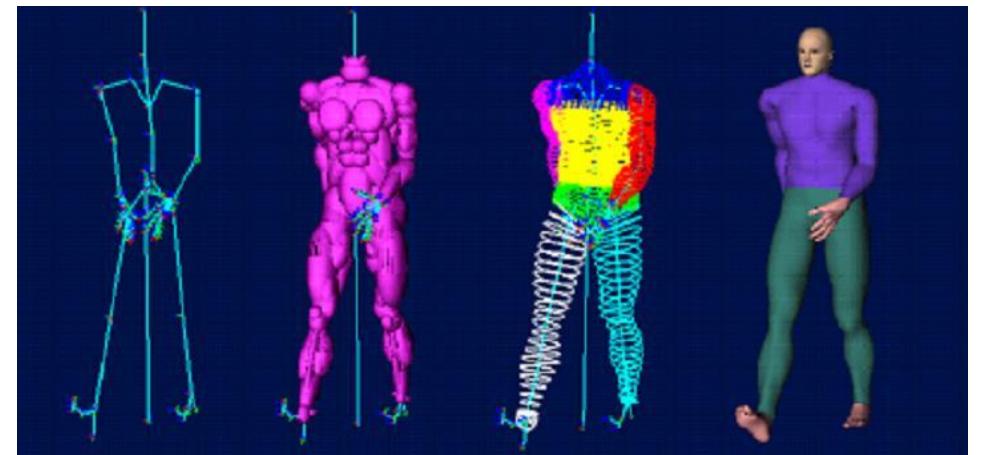


- Metode krutih segmenata

- Najjednostavniji pristup deformacije kože – segmenti tijela su kruti oblici spojeni na kosti
- Nedovoljno realistični rezultati
  - Rješavaju se pažljivim preklapanjem segmenata u zglobu

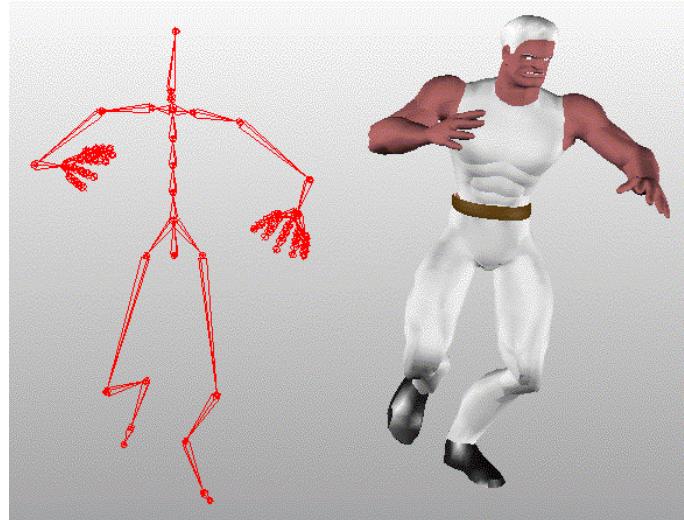
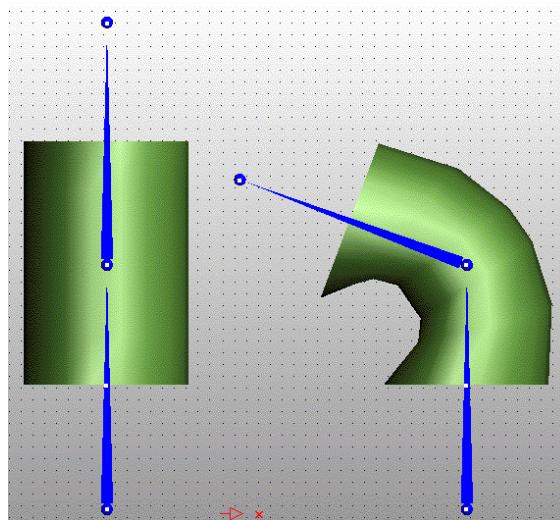
- Slojeviti model

- Simulacija slojeva između kostiju i kože
  - Npr. stapajuće plohe



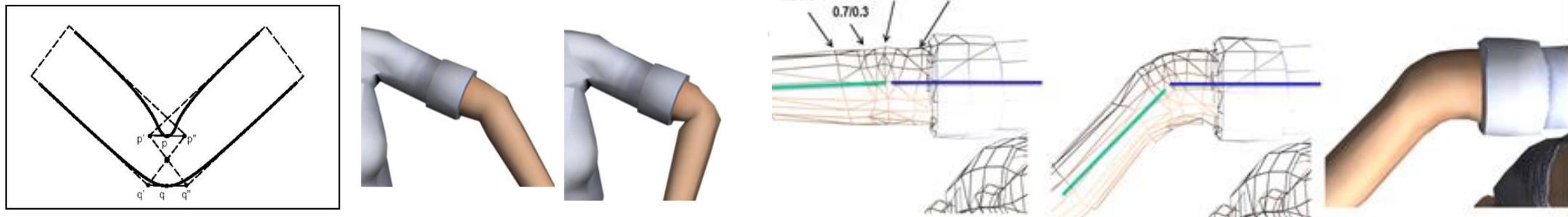
# Animacija kostima (*skinning*)

- *Rigging* – proces kreiranja unutarnjeg digitalnog kostura za neku poligonalnu mrežu
- *Skinning* – proces vezanja površinske 3D mreže na unutarnji kostur tako da zglobovi mogu imati utjecaj na vrhove mreže
- Najrašireniji postupak animacije tijela
- Koža se deformira pomicanjem kostura, ali deformacija se izvodi težinskim miješanjem



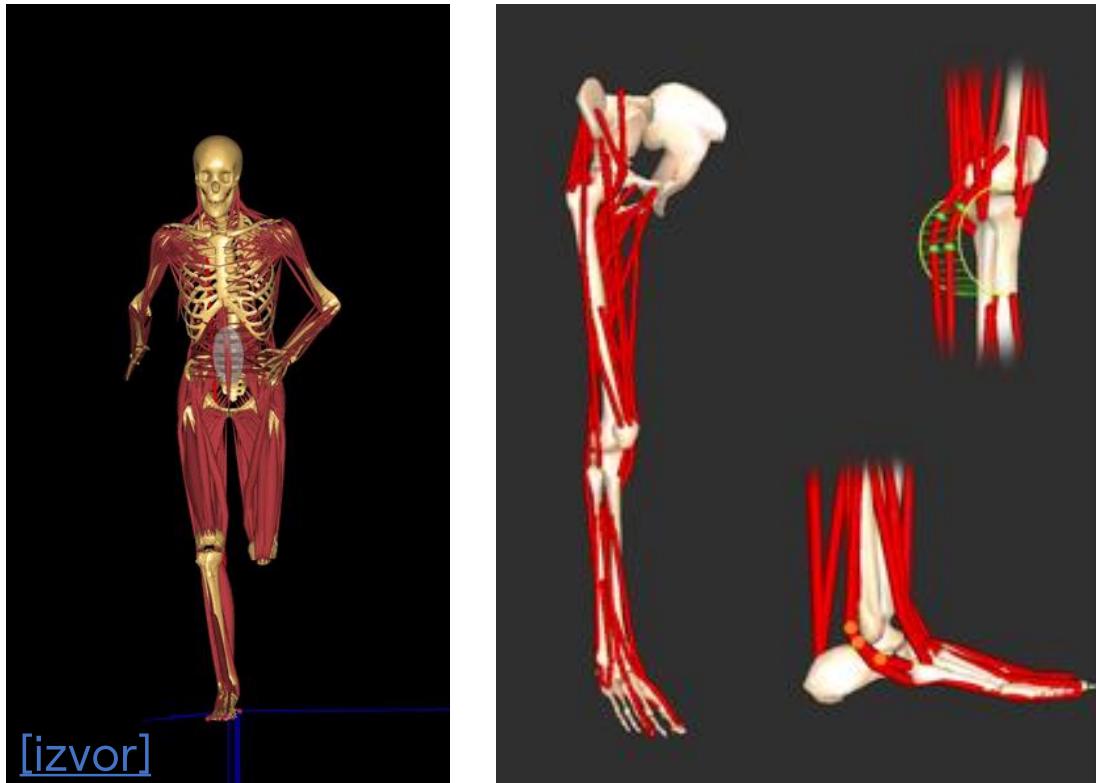
# Animacija kostima – osnovna ideja

- Vrhove poligona pomicu jedna ili više kostiju (obično do 4 kosti)
- Kosti koje utječu na pomak imaju svoju težinu za svaki vrh
  - Težine se pripisuju ručno ili automatski
- Konačna pozicija vrha se određuje miješanjem težina i pomaka kostiju



# Fizički modeli

- Simulacija stvarnih mišića i stvaranje biomehaničkog modela tijela
- Računalno izuzetno zahtjevno
  - Pribjegava se aproksimaciji



# Animacija tijela na visokoj razini

- Izvori slijedova animacijskih parametara
  - Snimljene animacije (ručna izrada, tehnike slijedeњa)
  - Proceduralne animacije
  - Hibridni pristup

# Snimljene animacije

- Snimanje ili stvaranje animacijskih sekvenci u svrhu kasnije reprodukcije
  - Vrlo prirodne animacijske sekвенце
- Ručna izrada
  - Animacija ključnim položajima
- Metode slijedeњenja
  - Film, računalne igre
  - Problem: primjena na različite anatomije tijela, poravnanje sa ravniom (npr. hodanje)



[\[izvorni video\]](#)

# Proceduralne animacije

- Uporaba matematičkih modela koji simuliraju fizikalne zakone i sile
  - Jednostavne za kontrolu
- Inverzna kinematika
  - Primjer: iz zadanog položaja šake može se odrediti položaj i poza čitavog tijela
- Ragdoll fizika
  - Animacija padova
  - Animacija borbi i scena u koje su uključene fizikalne sile



# Hibridni pristup

- Modifikacija snimljenih animacija kako bi se dobile nove animacije
  - Iz ulaznih parametara i baze snimljenih animacija dinamički se generiraju sekvence novih animacija
  - Velik potencijal za interaktivne primjene
- Temelj hibridnog pristupa – miješanje pokreta (*engl. motion blending*)
  - Postupak kojim se iz dvije ili više zadanih animacijskih sekvenci uz pomoć težinske funkcije dobiva nova animacijska sekvenca
  - Miješaju se animacije koje su slične

# Hibridni pristup – miješanje pokreta

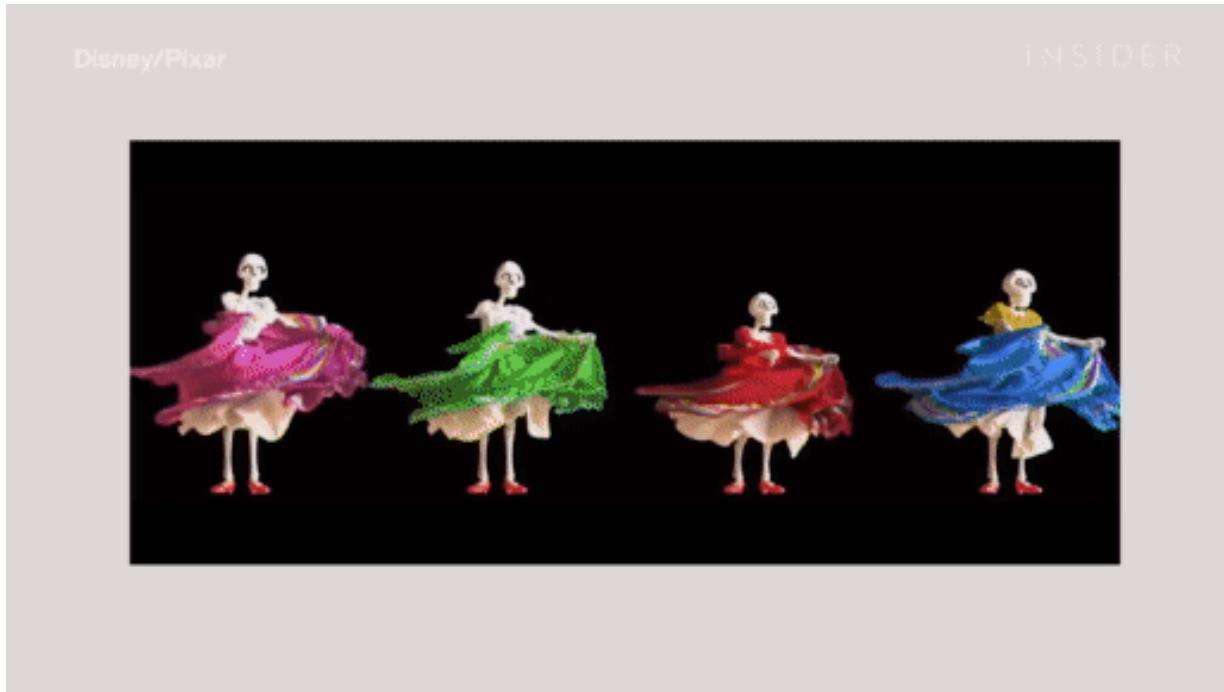
- Specifikacija animacijske sekvence, odnosno kretanje u prostoru
$$M(f) = (p(f), q_1(f), \dots, q_n(f))$$
- Rezultat miješanja  $B(t)$  se dobiva iz sekvenci  $M_1, \dots, M_N$  i N-dimenzionalne težinske funkcije  $w(t)$  koje određuje doprinos sekvence
- Zbroj težina za bilo koje vrijeme je 1
- Tranzicija – ako je  $N=2$ ,  $w_1$  opada od 1 do 0,  $w_2$  raste od 0 do 1
- Interpolacija :  $w_i = \text{const.}$

# Hibridni pristup – karakteristike

- Snimljene animacije su spremljene u baze
- Problemi
  - Pronaći slične sekvence i okvire za miješanje i zapisati ih u odgovarajuće strukture kako bi se dobio odziv u stvarnom vremenu
  - Omogućiti parametarsku kontrolu rezultata miješanja
  - Poravnanje u 3D prostoru, npr. kretanje, kolizija s objektima
- Rješenja
  - Grafova pokreta – strukture koje sadrže slične okvire za miješanje
  - Registracijske krivulje – poravnanje animacijskih sekvenci na zajedničku koordinatnu os
  - Parametrizirani grafovi pokreta i prostor pokreta – parametrizacija prostora

# Animacija odjeće

- Vrlo često, odjeća se uopće ne simulira kao takva nego postoji jedna površina tijela na kojoj se bojom razlikuje odjeća
- Fizikalna simulacija odjeće – sustavi čestica



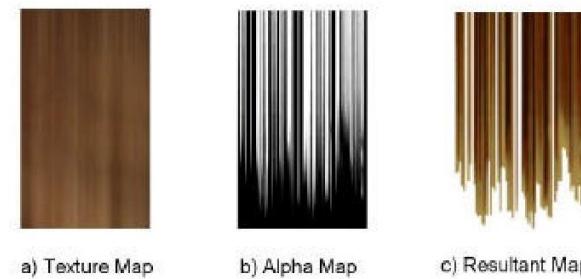
[\[izvorni video\]](#)

# Animacija kose

- Kruti model kose
- Postoje modeli za animaciju kose
  - Model sa trakama
  - Sustavi čestica
  - Volumenske teksture
  - Simulacija pojedinih vlasa
    - Simuliraju se poput elastičnih opruga



[izvor]



Koh, Chuan Koon, and Zhiyong Huang. "Modeling and animation of human hair in strips." SIGGRAPH. 2000.