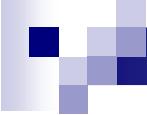


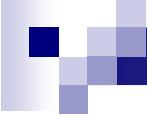
Računarska grafika

Dr. sci. Emir Skejić, vanr. prof.
Fakultet elektrotehnike Tuzla



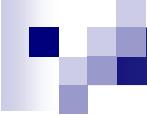
Nastavno osoblje

- Nastavnik: Dr. sci. Emir Skejić, vanr. prof.
 - Predavanja: STELEKT (četvrtak, 11:00-14:00 h)
 - Kancelarija: FE 203 (utorak, 11:30-12:30 h)
 - E-mail: emir.skejic@untz.ba
 - Web stranica predmeta: www.fe.untz.ba
- Asistent: Harun Delić, BSc inž. el.
 - Vježbe: Prema utvrđenom rasporedu nastave
 - Kancelarija: Nema
 - E-mail: harun.delic@fet.ba



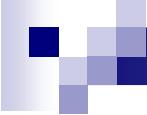
Preduslovi

- Linearna algebra
- Poznavanje:
 - Vektori (skalarni proizvod, vektorski proizvod...)
 - Matrice
 - C/C++ programski jezik



Literatura

- D. Hearn, M. P. Baker, "Computer Graphics with OpenGL", 4th Edition, Pearson, 2010, ISBN-13: 978-0136053583
- Samuel R. Buss. *3-D Computer Graphics: A Mathematical Introduction with OpenGL*. Cambridge University Press, 2003.
- M. Woo et al. *OpenGL Programming Guide*, 4th Edition. Addison Wesley, 1999.
- Bilo kakvi korisni sadržaji koji se mogu pronaći na Internetu



Obaveze studenata

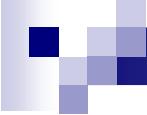
- Redovno pohađanje predavanja (dozvoljena su 2 neopravdana izostanka)
- Redovno pohađanje vježbi (nije dozvoljen nijedan neopravdan izostanak)

- Preduslov za formiranje konačne ocjene je uspjeh od 50% na završnom ispitu (ostvarenih minimalno 25 bodova)

Način bodovanja i ocjenjivanje

- Ocjena iz predmeta "Računarska grafika" izvodi se na osnovu sljedećih elemenata:

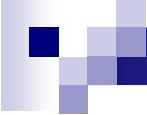
Vrsta aktivnosti	Specifična aktivnost	Bodovi
Pohađanje predavanja	Aktivnost na predavanjima	5
Prisustvo vježbama	Aktivnost na vježbama	5
1. kolokvij	Pismeni ispit	20
2. kolokvij	Pismeni ispit	20
Kviz	Test	10
Završni ispit	Projekt iz programiranja	40



Način bodovanja i ocjenjivanje

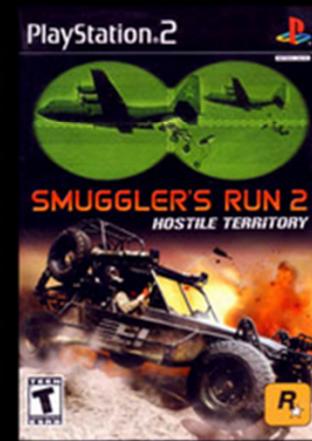
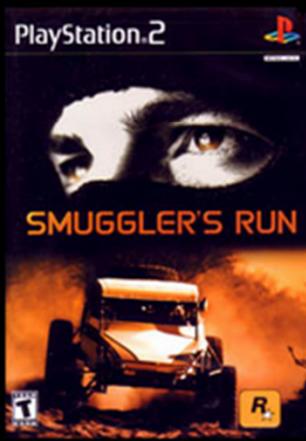
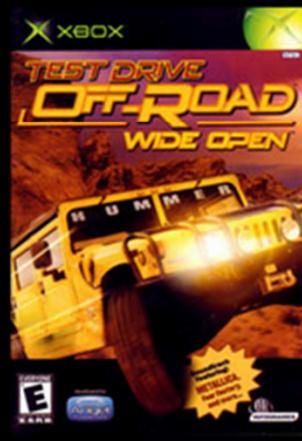
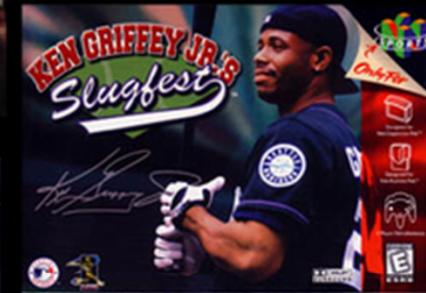
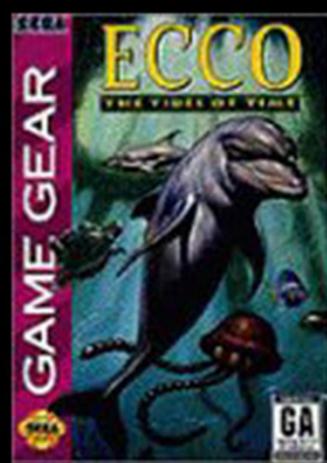
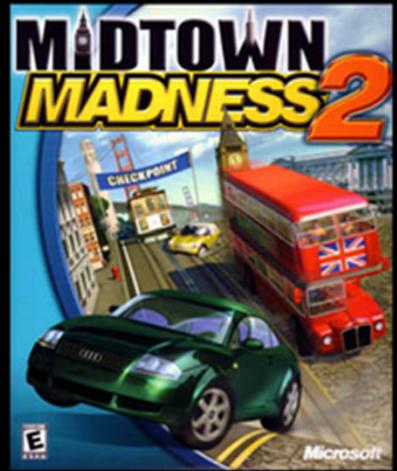
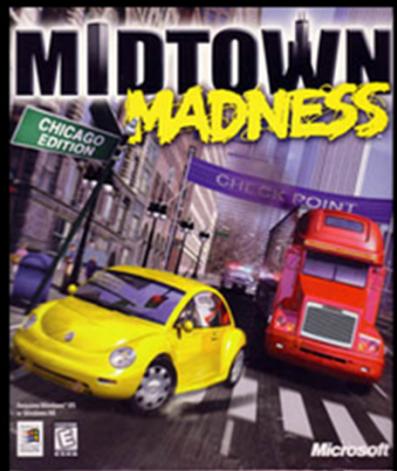
- Konačna ocjena se formira kao zbir osvojenih bodova

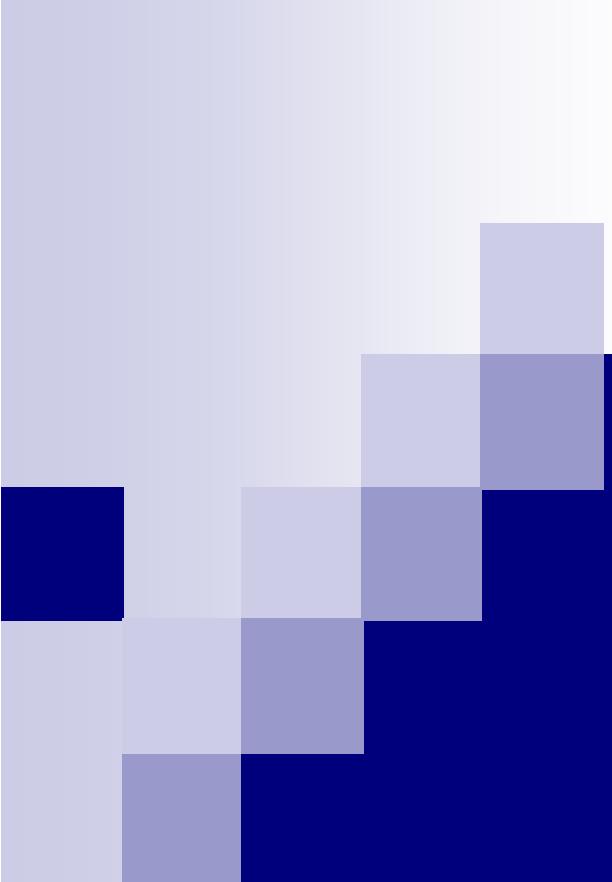
<input type="checkbox"/> Od 54 do 63	→ dovoljan (6)
<input type="checkbox"/> Od 64 do 73	→ dobar (7)
<input type="checkbox"/> Od 74 do 83	→ vrlo dobar (8)
<input type="checkbox"/> Od 84 do 93	→ odličan (9)
<input type="checkbox"/> Od 94 do 100	→ izvrstan (10)



Sadržaj kursa

1. Uvod
2. Grafički hardver
3. Vektori i matrice
4. Homogene transformacije
5. Gledanje i perspektiva
6. Odsijecanje i scan konverzija
7. Uvod u OpenGL
8. Preslikavanje tekstura
9. Osvjetljenje
10. Kubne krive
11. Zakrivljene površine



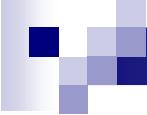


Računarska grafika

Dr. sci. Emir Skejić, vanr. prof.
Fakultet elektrotehnike Tuzla

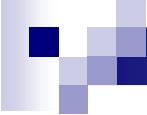
Uvod

- Računarsku grafiku možemo definirati kao granu računarskih nauka koja se bavi izgradnjom višedimenzionalnih modela objekata i njihovim *prikazom* uz upotrebu računara
- Ovo područje nauke privuklo je pažnju svih ostalih područja gdje je potrebno ostvariti prikaz
- Čovjek je *vizualno* biće, te je vješt brzo i efikasno prihvatići i obraditi veliku količinu informacija preko čula vida (brojčani rezultati ne daju zornu informaciju)
- Osnova u izgradnji virtualnih svjetova



Područja primjene

- Film, TV specijalni efekti
- Video igrice
- Naučna vizualizacija
- Geografski informacioni sistemi GIS
- Medicinska vizualizacija
- Industrijski dizajn
- Simulacija
- Komunikacija
- Itd.



Obrada slike

- Neke operacije u računarskoj grafici uključuju obradu 2D slika (bitmape)
- *Obrada slike* se primjenjuje direktno na rešetku piksela (engl. *pixel grid*) i uključuje operacije kao što su korekcija boje, skaliranje, zamagljivanje, izoštravanje, itd.
- Uobičajen primjer uključuje obradu digitalne fotografije i programe za digitalno "slikanje" (Adobe Photoshop...)

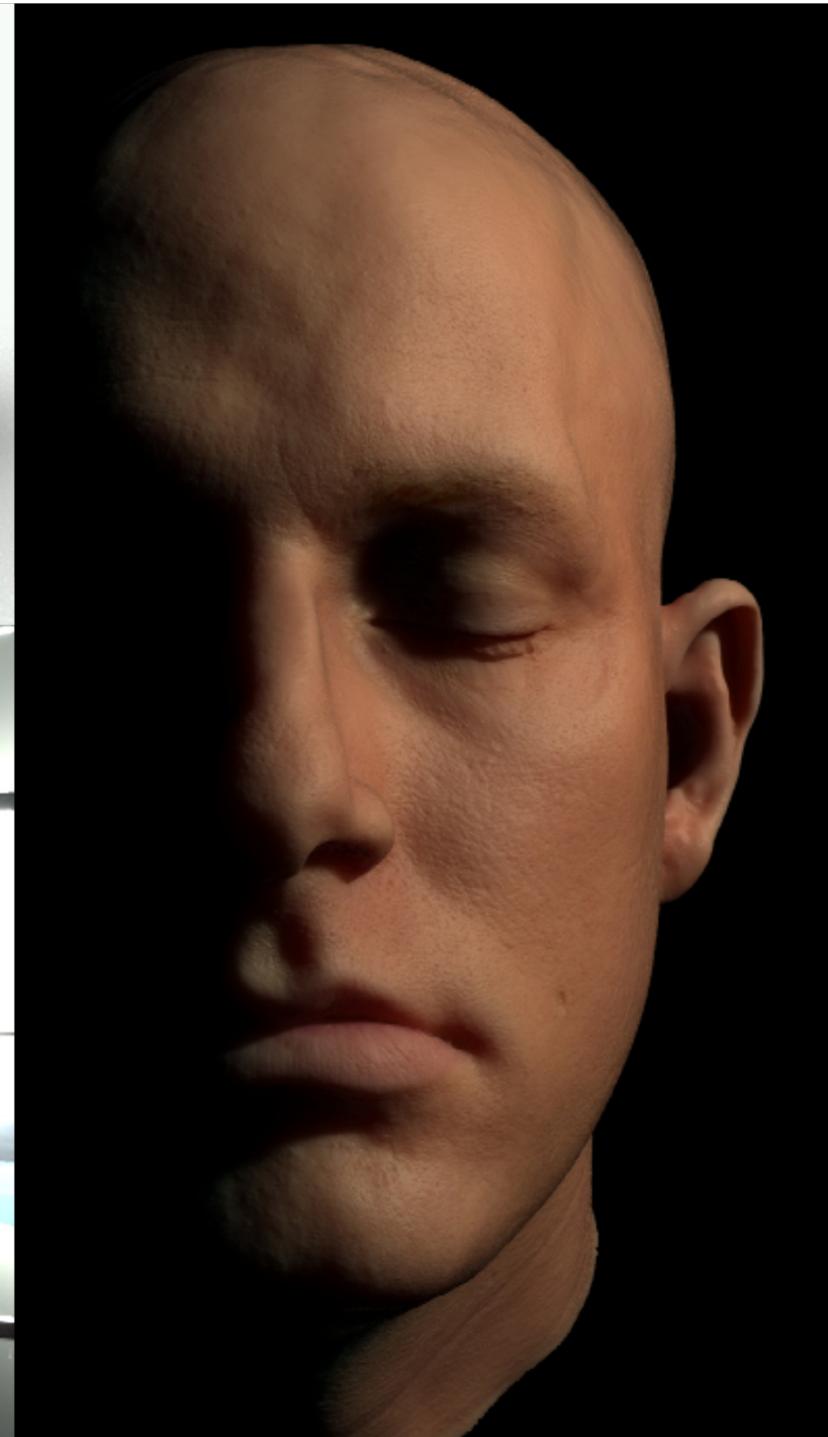


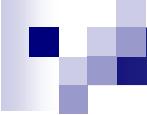
Sinteza slike

- *Sinteza slike ili stvaranje slike* više se odnosi na izgradnju slika počev od nule, nego na obradu postojećih slika
- Sinteza 2D slike iz opisa 3D scene često se naziva *rendering*

Fotorealistični rendering

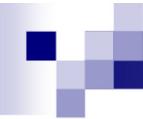
- *Fotorealistični rendering* se odnosi na rendering 3D scene na realističan način
- Moderni algoritmi fotorealističnog renderinga su u suštini simulacije zasnovane na fizikalnim svojstvima propagacije i disperzije svjetlosti kroz 3D okruženje
- U određenom smislu, ovo znači da postoji "korektna" slika koja se treba generirati na osnovu skupa ulaznih podataka. Ovo dozvoljava da predmet fotorealističnog renderinga ima jaku teorijsku osnovu (tj. nauku o svjetlosti)
- Većina modernih algoritama fotorealističnog renderinga je bazirana na algoritmu klasičnog *praćenja zrake* (engl. *ray tracing*), u kojem se prati putanja pojedinačnih svjetlosnih zraka unazad, tj. počev od oka prema izvorima svjetlosti



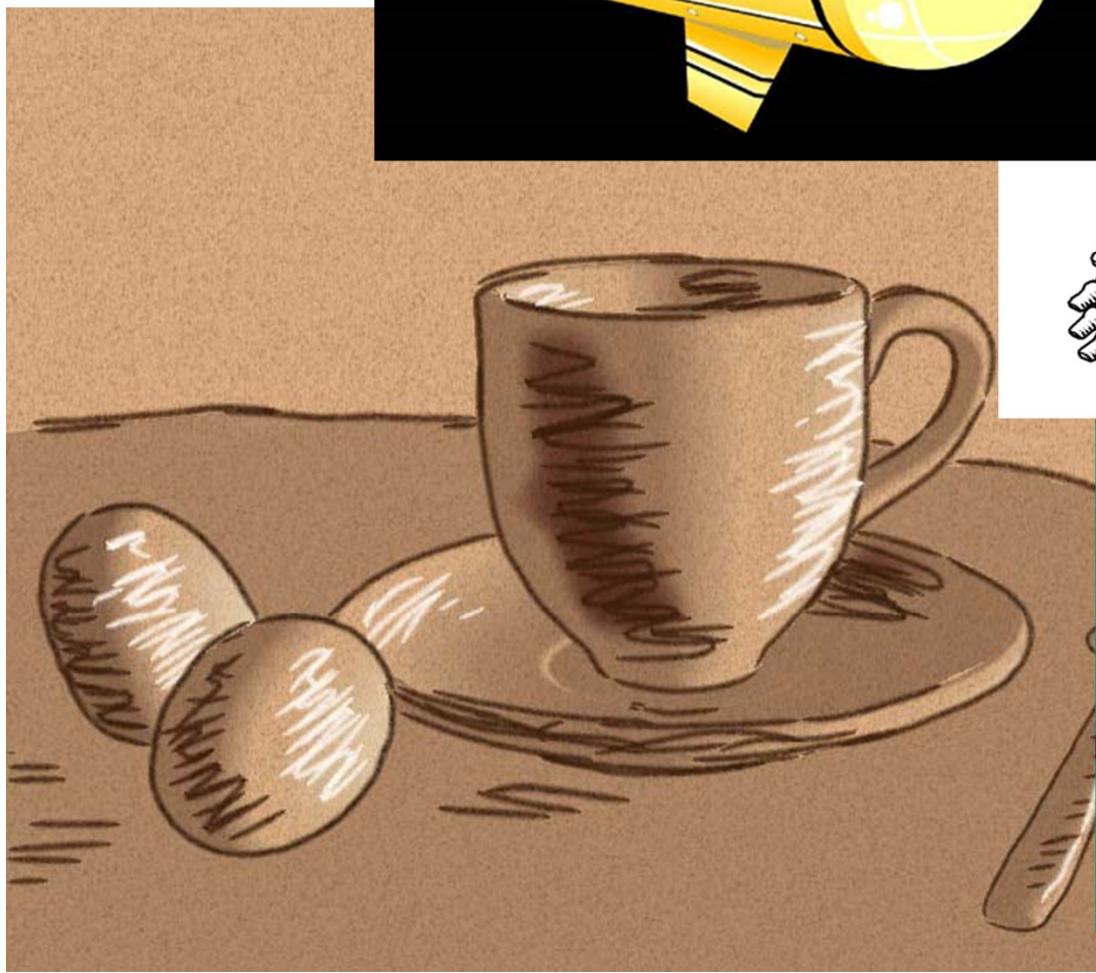
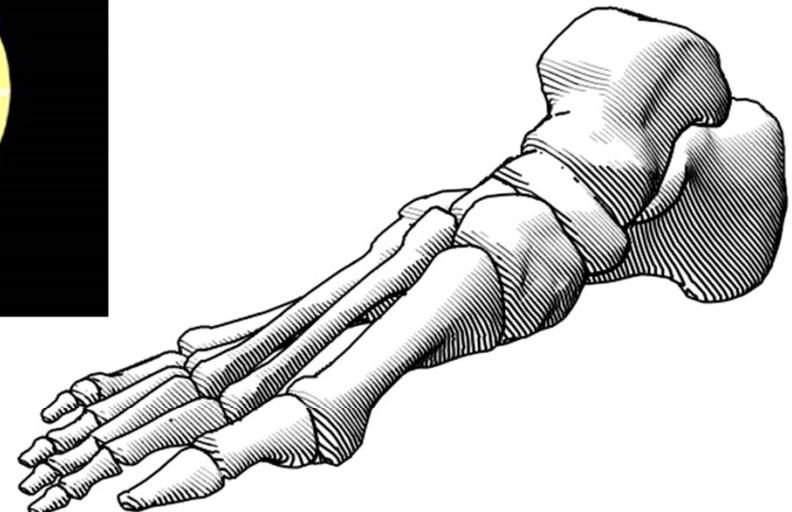
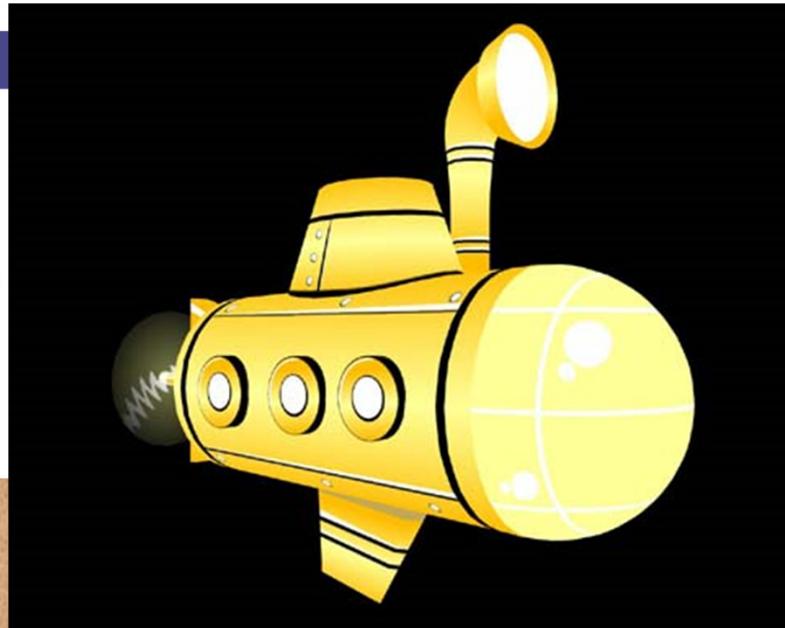


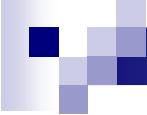
Nefotorealistični rendering

- *Nefotorealistični rendering* (engl. *Non-Photoreal Rendering* – *NPR*) se odnosi na rendering slika na druge načine...
- Ponekad se to radi da bi se postigli estetski ciljevi poput vještačkog akvarela, izrade skica olovkom, poteza kistom kod slikanja...
- U drugim prilikama, cilj je maksimizirati komunikaciju vizualnim informacijama, kao u naučnoj i medicinskoj vizualizaciji



NPR



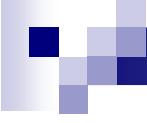


Računarska vizija

- Računarska vizija se ponekad razmatra kao zasebna disciplina od računarske grafike, premda ove dvije discipline imaju mnogo toga zajedničkog
- Centralni cilj u računarskoj viziji je uzeti skup 2D slika (obično iz videa ili skupa fotografija), te na osnovu toga zaključiti kakav je 3D opis onoga što je posmatrano
- Ovo je znatno drugačiji proces od renderinga, i više predstavlja nekakav oblik vještačke inteligencije

Računarska vizija

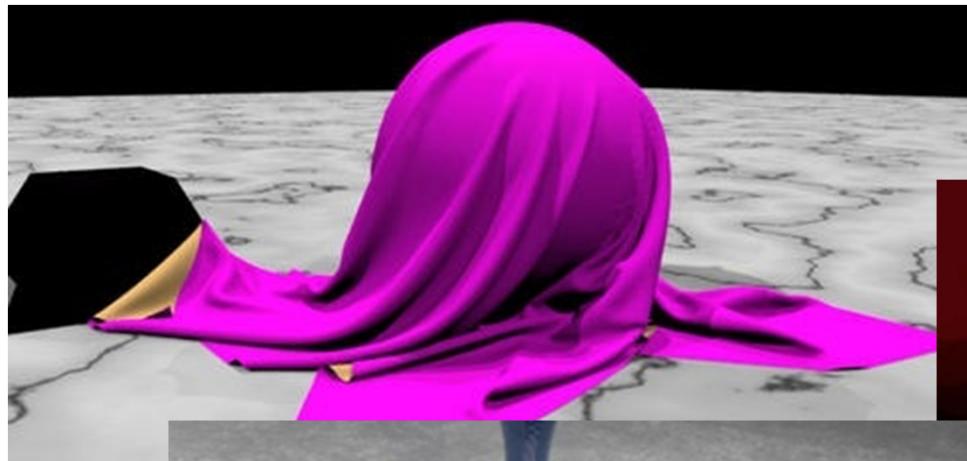




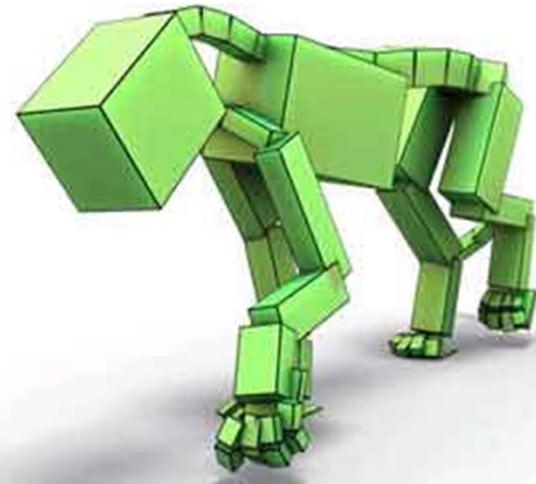
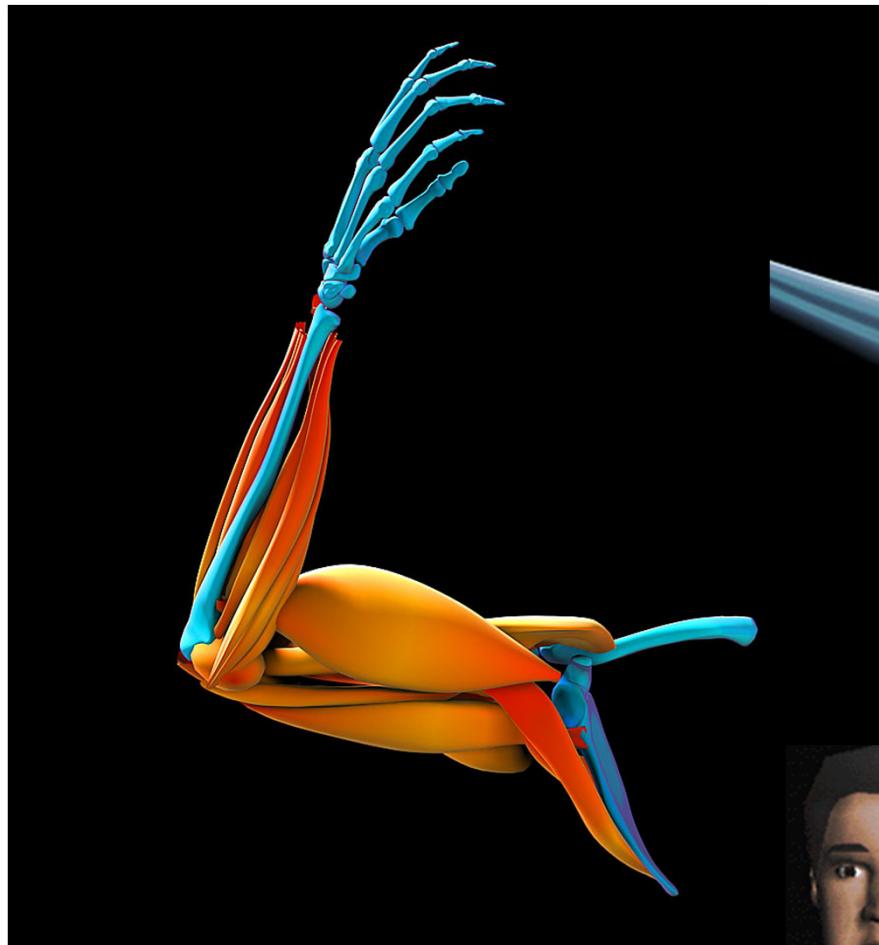
Animacija

- Animacija je samo sekvenca pojedinačnih slika
- U osnovi, predmet računarske animacije je usmjeren na promjene tokom vremena. Ovo se obično odnosi na kretanje, ali se može odnositi i na druge karakteristike koje se mijenjaju tokom vremena
- Simulacija fizike (engl. *physical simulation*) je veoma moćan alat u računarskoj animaciji i može biti korištena za stvaranje uvjerljivih animacija krutih objekata, deformabilnih objekata, gasova, tečnosti, frakturna, atomskih efekata, te čak i eksplozija i vatre
- Računarska animacija također uključuje veliki broj tehnika specifično dizajniranih za manipulaciju virtualnim likovima

Simulacija fizike

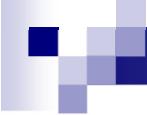


Animacija likova

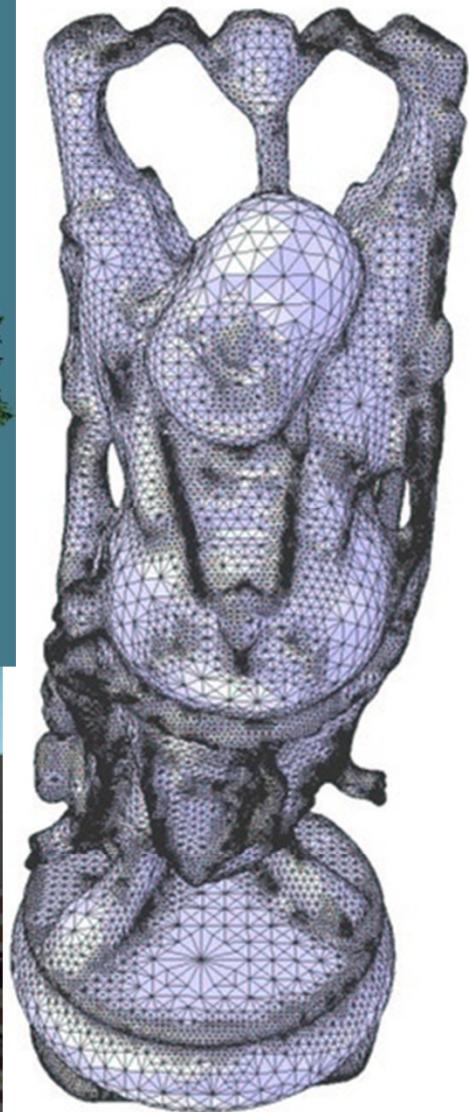
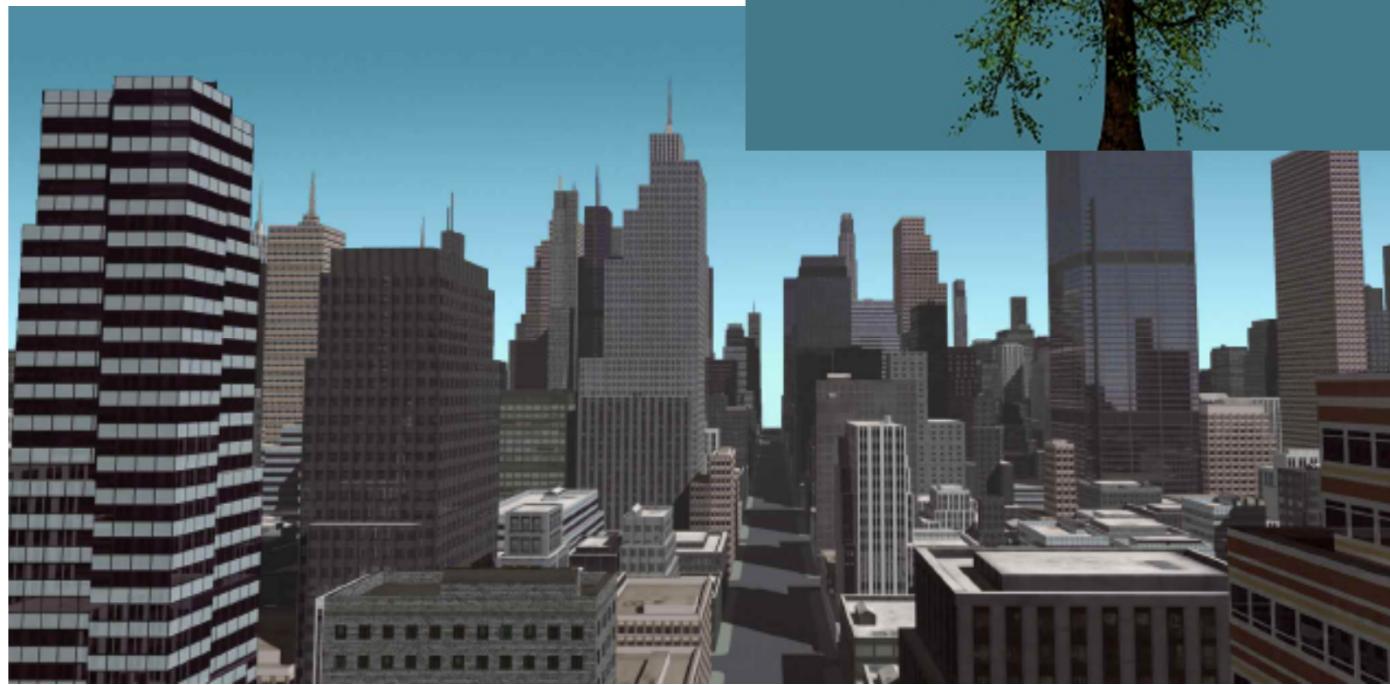


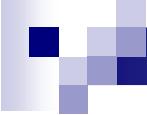
Modeliranje

- *Modeliranje* se odnosi na tehnike povezane s kreiranjem, skeniranjem, editovanjem i manipuliranjem 3D geometrijskim podacima
- Modeliranje često obavlja korisnik pomoću nekog interaktivnog programa za editovanje
- Složeniji objekti, poput stabala, mogu biti konstruirani pomoću *automatskih algoritama proceduralnog modeliranja*
- 3D modeli se također često dobivaju iz stvarnih objekata pomoću tehnika laserskog skeniranja ili računarske vizije
- Modeliranje također uključuje upotrebu zakriviljenih površina i drugih primitiva višeg reda, koji se često konvertuju u trouglove upotrebom raznih algoritama teselacije
- Još jedno važno područje modeliranja uključuje rekonstrukciju mesha za simplifikaciju površina
- Modeliranje znatno koristi *računsku geometriju* (engl. *computational geometry*)



Modeliranje



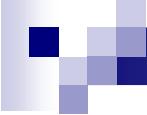


Računarska grafika

- Rendering
 - Fotorealistični
 - NPR
- Animacija
 - Fizika
 - Animacija likova (engl. *character animation*)
- Modeliranje
 - Računarska geometrija
 - Proceduralno modeliranje
 - Akvizicija podataka

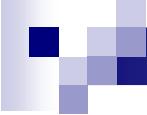
Historijski razvoj

- 1960-e:
 - Rani teoretski razvoj, ograničen uglavnom na istraživanje i vojsku
 - 1962: Sketchpad (Ivan Sutherland) – prvi program koji je u potpunosti koristio grafički interfejs
 - 1964: General Motors - DAC - sistem za dizajn automobila
- 1970-e:
 - Razvijen je "tradicionalni" grafički cjevod
 - Daljnji razvoj računarske grafike je vođen novcem iz vojnih simulacija i industrije automobilskog dizajna
- 1980-e:
 - 1980: 3D hardverska podrška na radnim stanicama
 - Razvijeni mnogi značajni temeljni algoritmi
 - Poboljšan vizualni kvalitet vođen zahtjevima iz industrije zabave (film)
 - 1985: Jednačina renderinga (James Kajiya)
- 1990-e:
 - 1990: 3D hardverska podrška na PC računarima
 - Kada je usavršena teorija o grafici, kreirani su i napredni algoritmi
 - Širi fokus na animaciju, akviziciju podatka, NPR, fiziku...
 - 1995: Preslikavanje fotona (Henrik Jensen)
- 2000-e:
 - 3D hardverska podrška na prijenosnim uređajima
 - Fotorealistični rendering evoluirao do faze u kojoj može renderirati uvjerljive slike proizvoljnih kompleksnih scena na potrošačkom hardveru
 - Spajanje računarske grafike i računarske vizije
 - Jeftin grafički hardver s velikim sposobnostima, uvelike vođen industrijom video igrica
- 2010-e:
 - Struktura i izračun fluida mogući su na prijenosnim računarima
 - Za izradu animacija koriste se veliki podaci (engl. big data)
 - Animacije se mogu napraviti preslikavanjem od izvora do ciljeva u realnom vremenu



Rasterska grafika

- Moderni grafički ekran su *rasterski*
- To znači da oni prikazuju rešetku piksela, gdje boja svakog piksela može biti nezavisno podešena
- Pojedinačni pikseli se obično formiraju od manjih crvenih, zelenih i plavih subpiksela. Ukoliko ste bili u prilici da iz velike blizine pogledate stariji TV ekran ili računarski monitor, mogli ste primijetiti uzorak subpiksela
- *Vektorski* ekran nisu prikazivali rešetku piksela, već su umjesto toga direktno crtali linije s elektronskim snopom
- Rasterska grafika također se ponekad naziva i *bitmapirana grafika*

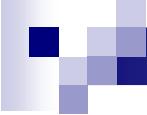


Prikazne tehnologije

- Vektorski ekran (katodna cijev)
- Rasterski ekrani
 - Katodna cijev (engl. *Cathode Ray Tube – CRT*)
 - Ekran s tečnim kristalima (engl. *Liquid Crystal Display – LCD*)
 - TFT → tanki film tranzistor
 - Organska svjetlosna dioda (engl. *Organic Light Emitting Diode – OLED*)
 - Svjetlosna elektronska cijev
 - Plazma
 - HDR (veliki dinamički raspon: TFT / bijeli hibridni LED)
- Film
- Štampa

Rezolucija i frekvencija okvira

- Video:
 - NTSC: 720 x 480 @ 30 Hz (prepleteni)
 - PAL: 720 x 576 @ 25 Hz (prepleteni)
- HDTV:
 - 720p: 1280 x 720 @ 60 Hz
 - 1080i: 1920 x 1080 @ 30 Hz (prepleteni)
 - 1080p: 1920 x 1080 @ 60 Hz
- Film:
 - 35mm: ~2000 x ~1500 @ 24 Hz
 - 70mm: ~4000 x ~2000 @ 24 Hz
 - IMAX: ~5000 x ~4000 @ 24-48 Hz
- Napomena: Hz (Hertz) = frame-ova u sekundi (engl. *frames per second – fps*)
- Napomena: Video standardi označeni sa *i* (npr. 1080i) su *prepleteni*, dok su standardi označeni sa *p* (1080p) *progresivni scan*



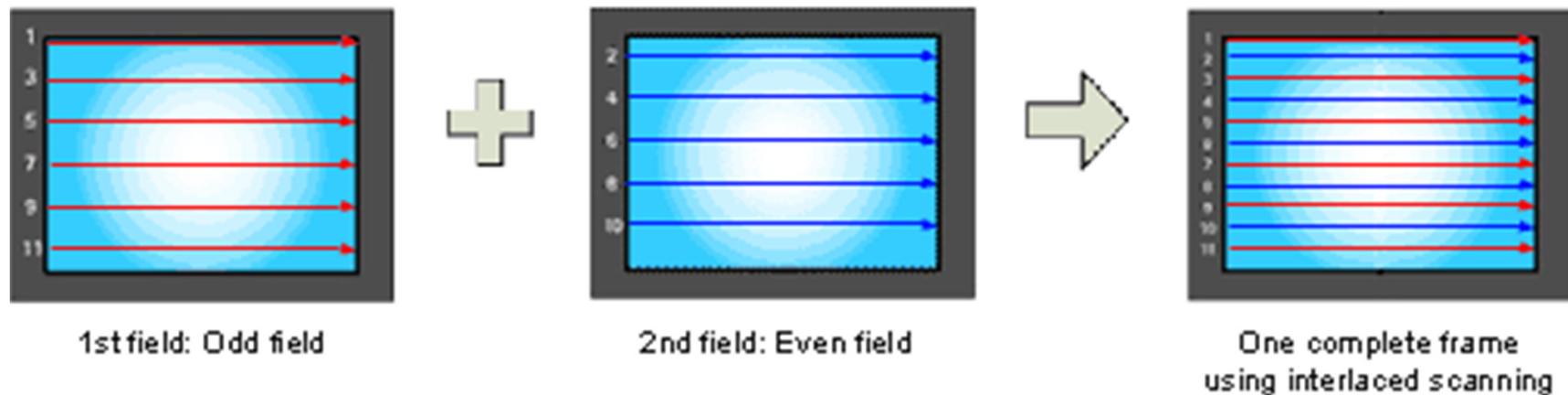
Interlacing

1/2

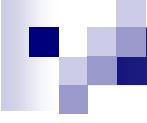
- Stariji video formati (NTSC, PAL) i pojedini HD formati (1080i) koriste tehniku pod imenom *interlacing* (preplitanje linija prikaza)
- Interlacing dijeli scan linije na neparne i parne linije, a zatim ih naizmjenično osvježava pri 30 frame-ova u sekundi. Malo kašnjenje između parnih i neparnih scan linija stvara određeno izobličenje ili "*jaggedness*". To se dešava zato što samo polovina linija "drži korak" s pokretnom slikom, dok druga polovina čeka da bude osvježena.
- Takav scan svake druge linije naziva se *interlacing*. Polje (engl. *field*) je slika koja sadrži samo polovicu linija potrebnih za kreiranje potpune slike.
- NTSC video je, naprimjer, 720 x 480 pri 30 *frame-ova* u sekundi, ali je ustvari 720 x 240 pri 60 *polja* u sekundi
- Interlacing je važno pitanje za razmatranje kada se radi s videom, naročito u animacijama kao u TV efektima i video igricama
- Općenito, računarski monitori nisu prepleteni

Interlacing

2/2



- Kod bilo kojeg video sistema postoje kompromisi. Jedan od najvažnijih faktora je propusnost (engl. *bandwidth*), mjerena u MHz-ima (za analogni video), odnosno brzina prijenosa (za digitalni video). Što je veća propusnost, skuplji je i složeniji cijeli sistem (kamera, sistemi za pohranjivanje podataka kao što su magnetofoni ili tvrdi diskovi, prijenosni sistemi kao što su sistemi kablovske televizije, te ekrani kao što su TV ekrani).
- Interlaced video dvostruko smanjuje propusnost signala, za dati broj linija i frekvenciju osvježavanja.

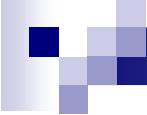


Framebuffer

- Pojam *framebuffer* se odnosi na memoriju namijenjenu za pohranjivanje slike
- To bi općenito bio 2D niz piksela, pri čemu svaki piksel pohranjuje boju (Napomena: piksel = element slike)
- Boja se tipično pohranjuje kao 24-bitna RGB vrijednost. Ovo pruža 8 bitova (256 nivoa) za crvenu, zelenu i plavu boju, za ukupno 16.777.216 različitih boja
- Vrlo često su po pikselu pohranjeni i dodatni podaci kao što je dubina (z), ili druge informacije
- Framebuffer može biti samo blok glavne memorije, mada mnogi grafički sistemi imaju namjensku framebuffer memoriju s direktnom konekcijom na hardver za video skeniranje (engl. *video scan-out hardware*) i druge specijalne karakteristike

Primitivi

- Kompleksne scene su obično izgrađene od jednostavnijih *objekata*
- Objekti su izgrađeni od pojedinačnih *primitiva*
- Najuobičajeniji i opštenamjenski 3D primitiv je trougao
- Tačke i linije također su korisni primitivi
- Složeniji oblici kao što su n-stranični poligoni, sfere, krive, zakrivljene površine te fraktali također mogu biti smatrani za primitive. Međutim, oni se samo automatski *teseliraju* u trouglove u fazi prije renderinga, te tako nisu pravi primitivi



3D modeli

- Osnovni 3D model može se sastojati od jednostavnog niza trouglova
- Svaki trougao pohranjuje 3 vrha (engl. *vertex*)
- Svaki vrh sadrži xyz poziciju, te vjerovatno i neke druge informacije (boja, normala...)

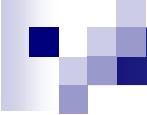
3D modeli

```
class Vector3 {  
    float x,y,z;  
};
```

```
class Vertex {  
    Vector3 Position;  
};
```

```
class Triangle {  
    Vertex Vert[3];  
};
```

```
class Model {  
    int NumTris;  
    Triangle *Tri;  
};
```

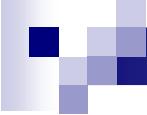


Tradicionalni grafički cjevovod 1/2

- Niz funkcija koje se izvode jedna za drugom ili istovremeno
- Pretvaraju prividnu scenu u sliku
- Najsporija operacija određuje brzinu izvođenja
- Optimiziran za rad s trouglovima
- Prikaz grafičkog cjevovoda nepromjenljive funkcionalnosti

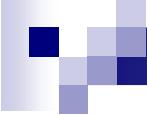
Tradicionalni grafički cjevovod 2/2

- U *tradicionalnom grafičkom cjevovodu* (engl. *pipeline*), svaki primitiv se obrađuje kroz sljedeće korake:
 - Transformacija
 - Osvjetljavanje
 - Odsijecanje
 - Scan konverzija
 - Obrada piksela



Transformacija

- Proces transformacije se odnosi na linearu transformaciju iz 3D prostora u 2D prostor pogleda
- Konačno, pozicija svakog vrha mora biti transformirana iz definiranog *prostora objekta* u *koordinate uređaja* (prostor piksela)
- Ovo često uključuje kombinaciju rotacija, translacija, skaliranja i perspektivnih transformacija
- Naučit ćemo kako predstaviti cjelokupnu ovu materiju s 4×4 homogenim matricama

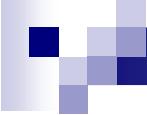


Osvjetljavanje

- Operacije osvjetljavanja se primjenjuju na svaki vrh da bi se izračunala njegova boja
- Kod naprednjeg renderinga, operacije osvjetljavanja se izračunavaju prije po pikselu, nego po vrhu
- Mogu biti definirane raznolike vrste svjetala kao što su tačkasta svjetla, usmjerena svjetla, spotlight, itd.
- Naprednije operacije osvjetljavanja mogu uzeti u obzir sjene, refleksiju, translucenciju i mnoge raznolike optičke efekte

Odsijecanje

- Neki trouglovi biće u potpunosti vidljivi na ekranu, a neki mogu biti potpuno izvan pogleda
- Neki će možda presijecati ivicu ekrana te zahtijevaju specijalno rukovanje
- Prostor vidljiv kamerom obrazuje volumen koji se naziva *volumen pogleda*. Trouglovi koji presijecaju granice volumena pogleda moraju biti *odsječeni*.
- Povezani postupak *selektivnog odbacivanja* (engl. *culling*) odnosi se na određivanje koji primitivi su potpuno nevidljivi
- Izlaz postupka odsijecanja/cullinga je skup vidljivih trouglova koji leže unutar dimenzija prikaznog uređaja



Scan konverzija

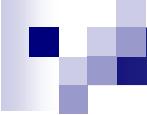
- Proces *scan konverzije* (ili *rasterizacije*) uzima 2D trouglove kao ulaz i daje kao izlaz egzaktne piksele pokrivene trouglom
- Podaci "po vrhu" trougla, npr. boja, se *interpoliraju* preko trougla, tako da svaki piksel može imati jedinstvenu boju

Obrada piksela

- Izlaz procesa scan konverzije je "snop" pojedinačnih xy piksela, plus dodatni podaci za piksel poput interpolirane dubine (z), boje ili drugih informacija
- Faza "obrada piksela" uključuje operacije koje se događaju po pikselu kako bi se izračunala finalna boja koja biva renderirana u *framebuffer*
- Obično se koristi *zbuffer* tehnika da bi se osiguralo da piksel bude renderiran samo ako ga ne blokira postojeća površina
- Ostala obrada, kao što je teksturiranje i operacije s transparentnosti, događa se po pikselu
- U nekim sistemima, čitav proces osvjetljavanja se računa po pikselu, umjesto po vrhu

Rendering scene

- Kod tradicionalnog zbuffered grafičkog cjevovoda, trouglovi mogu biti renderirani bilo kojim redoslijedom bez utjecaja na konačnu sliku
- Kompleksni efekti poput transparentnosti često zavise od redoslijeda renderinga, te može biti potreban dodatni oprez
- Ipak, ovo je dobar temeljni pristup, tj. to je pristup usvojen od OpenGL-a i ugrađen u mnoge moderne hardverske grafičke ploče
- Postoje napredniji algoritmi renderinga (scan-line algoritam, praćenje zrake, itd.) koji trouglove ne renderiraju jedan po jedan, i zahtijevaju da bude procesirana čitava scena. O ovome ćemo kasnije također učiti.



OpenGL

- OpenGL je standard za interaktivni rendering dizajniran za lakoću upotrebe i prenosivost kod širokog spektra sistema
- On je u opštem slučaju ograničen na stvari koje mogu biti načinjene pri interaktivnim brzinama i ne uključuje osobine naprednog renderinga kao što je praćenje zrake
- Baziran je na tradicionalnom grafičkom cjevovodu gdje se trouglovi obrađuju jedan po jedan
- Mi ćemo u okviru ovog kursa učiti o OpenGL-u, ali će se većina predavanja fokusirati na temeljnu teoriju i nije zavisna od bilo kojeg određenog interfejsa. Većina onoga što budete naučili može biti primijenjena na Java3D i Direct3D, kao i na druge interfejse.

Primjer u OpenGL-u

- Evo jednog primjera kojim se vrši rendering obojenog trougla (Napomena: Ovo ne uključuje bilo kakav "setup" kôd)

```
glBegin(GL_TRIANGLES);
    glColor3f(1.0, 0.0, 0.0); // crvena boja
    glVertex3f(-4.0, -2.0, 0.0);
    glColor3f(0.0, 1.0, 0.0); // zelena boja
    glVertex3f(4.0, -2.0, 0.0);
    glColor3f(0.0, 0.0, 1.0); // plava boja
    glVertex3f(0.0, 5.0, 0.0);
glEnd();
```

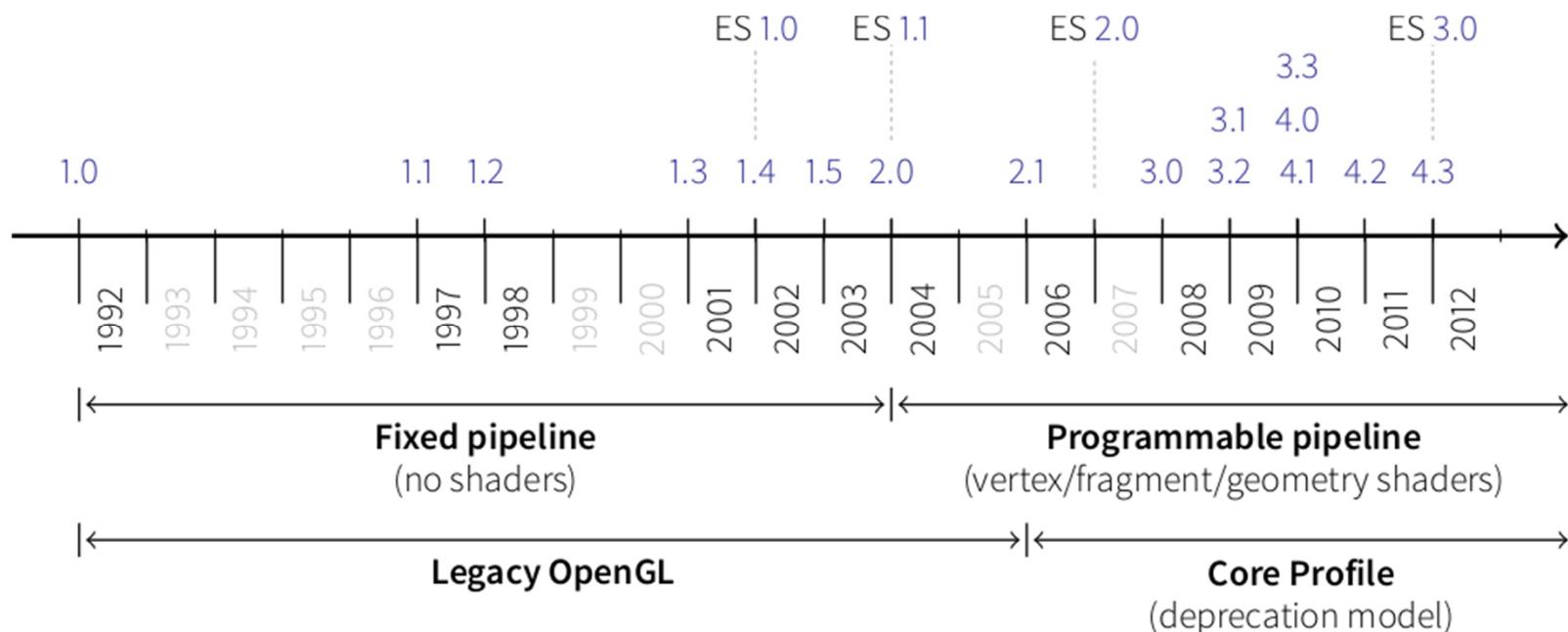
- Detaljniji uvod o tačkama, linijama, trouglovima, spojenim trouglovima (engl. *strips*) i lepezama (engl. *fans*) u OpenGL-u možete pronaći u poglavljju 1 knjige Samuela R. Bussa "3-D Computer Graphics: A Mathematical Introduction with OpenGL".

Organizacija OpenGL programa

- Evo jednog grubog primjera kako jedan tipičan GL program može biti organiziran:

```
main() {
    // Initialization code:
    // - Open window
    // - Load & define texture maps, etc.
    while(not finished) {
        // Frame set up code:
        // - Clear screen
        // - Specify camera position & properties
        // - Specify lighting information
        for(each object) {
            // Object set up code
            // - Specify object position & orientation (matrix)
            for(each material) {
                // Specify material properties
                // - Lighting properties
                // - Texture properties
                for(each primitive) {
                    // Render primitive (glBegin, glVertex3f...)
                }
            }
        }
        // Frame completion code (glSwapbuffers()...)
    }
    // Shutdown code
}
```

Moderni OpenGL 1/2

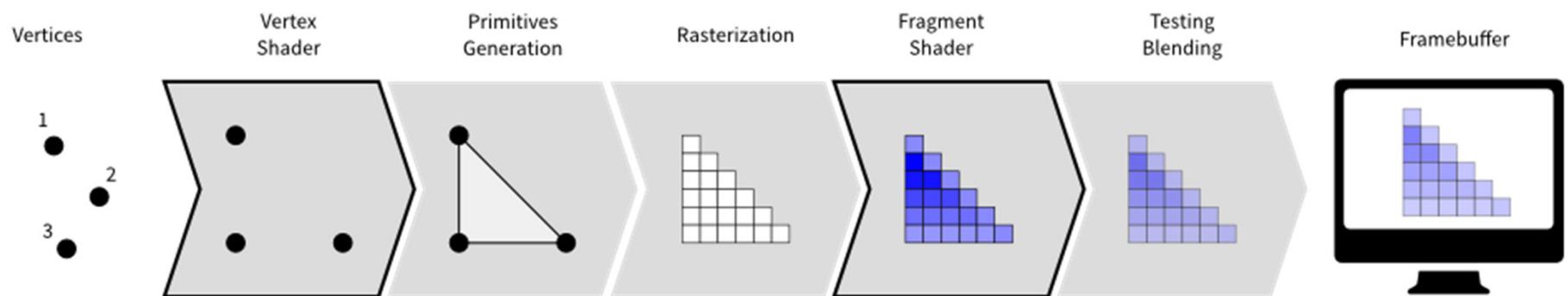


- OpenGL je evoluirao tokom godina, a velika promjena se dogodila 2003. godine uvođenjem dinamičkog cjevovoda (OpenGL 2.0), tj. upotrebom shadera koji omogućuju direktni pristup GPU-u.

Moderni OpenGL

2/2

- Prije ove verzije, OpenGL je koristio fiksni cjevovod i još uvijek se može pronaći puno tutoriala koji koriste ovaj fiksni cjevovod.
- Ovo uvodi neke radikalne promjene u načinu programiranja OpenGL-a i čini ga težim za programiranje, ali daleko moćnijim.



- Shaderi su dijelovi programa koji su izgrađeni na GPU-u i izvršavaju se tokom cjevovoda renderinga.