1.1 Čime se sve bavi računarska grafika?

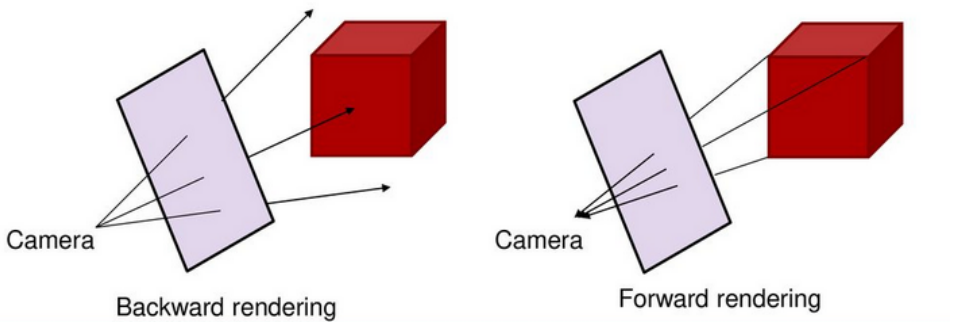
Računarska grafika se bavi pravljenjem modela objekata na sceni (geometrijskog opisa objekata na sceni i opisa kako objekti reflektuju svetlost) i modela osvetljenja scene (matematički opis izvora svetlosne energije, pravaca u kojima se ona emituje, raspodele talasnih dužina svetlosti) i, na osnovu njih, pravljenjem reprezentacija određenog pogleda na scenu (svetlost koja dolazi od nekog imaginarnog oka ili kamere na sceni).

1.2 Koje su osnovne poddiscipline računarske grafike?

- *modelovanje* – bavi se pravljenjem matematičke specifikacije tela i njegovih vizuelnih svojstava na način koji je moguće sačuvati na računaru
- *renderovanje* – to je proces kreiranja realistične dvodimenzione digitalne slike na osnovu (dvodimenzionog ili trodimenzionog) modela i (realističnog ili nerealističnog) modela ponašanja svetlosti
- *animacija* – to je proces kreiranja nizova slika koje, kada se prikažu brzo jedna za drugom, daju utisak glatkog kretanja
- *obrada slika* – bavi se zapisivanjem i obradom slika (kao što su isecanje dela slike, skaliranje slike, kombinovanje više slika, . . .), kao i rekonstrukcijom dvodimenzionih ili trodimenzionih objekata na osnovu njihovih slika.
- *virtuelna realnost* – pokušava da korisnika “ubaci” u 3D virtuelni svet, korišćenjem napredne 3D grafike i naprednih uređaja za prikaz
- *3D skeniranje* – koristi tehnologiju baziranu na pronalaženju opsega za pravljenje merljivih 3D modela
- *računarska fotografija* – korišćenje metoda iz oblasti računarske grafike i obrade slika za omogućavanja novih načina fotografskog pamćenja objekata, scena i okruženja

1.3 Koja je razlika između renderovanja unapred i renderovanja unazad?

Renderovanja unapred predstavlja proces renderovanja koji je najčešće podržan u hardveru i koji koristi OpenGL. U ovoj vrsti renderovanja primitive se transformišu od modela ka uređaju za prikaz.
S druge strane, rejtrejsing algoritam predstavlja primer koncepta *renderovanja unazad* kod koga se kreće od tačke na slici i onda se utvrđuje koje se primitive projektuju na nju.



1.4 Šta se podrazumeva pod tim da je proces modelovanja hijerarhijski?

Modelovanje obuhvata pravljenje modela, primenu materijala na modele, postavljanje modela na scenu, pozicioniranje svetla na sceni, postavljanje kamere. Dijagram stabla obezbeđuje hijerarhijski, vizuelni metod za izražavanje odnosa “sastavljen od”.

1.5 Koje dve paradigme razlikujemo u računrskoj grafici? Koje su njihove osnovne prednosti, a šta su njihovi nedostaci?

Grafika zasnovana na uzorku koristi diskretne uzorke za opisivanje vizuelne informacije. Pikseli se mogu kreirati digitalizacijom slike. U ovom pristupu pikseli su lokacije tačaka sa pridruženim vrednostima uzorka, najšešće intenziteta svetlosti, transparentnosti i drugim kontrolnim informacijama. Kada se slika definiše kao niz piksela, nju je moguće jednostavno izmeniti (izmene kreirane od strane korisnika) i obraditi (algoritamske operacije koje se izvode na slici bez intervencije korisnika). Prednosti ovog pristupa su da kada se slika jednom definiše u terminima boje na (x, y) poziciji mreže, ona se lako može modifikovati izmenom lokacije ili vrednosti boje, informacije o pikselima jedne slike se mogu iskopirati u drugu, zamenom ili kombinovanjem sa prethodnim pikselima. Mane ovog pristupa su što ne postoje dodatne informacije već važi paradigma What You See Is All You Get.

Grafika zasnovana na geometriji (skalabilna vektorska grafika ili objektno-orijentisana grafika) - kreiraju se i čuvaju matematički opisi ili modeli geometrijskih elemenata i pridruženih atributa i onda se oni uzorkuju za vizuelizaciju (vrši se rasterizacija). Korisnik najčešće ne može da direktno radi nad individualnim pikselima u geometrijski zasnovanim programima, već dok korisnik radi sa geometrijskim elementima, program iznova uzorkuje i prikazuje elemente. Renderovanje sve više kombinuje geometrijski zasnovanu grafiku i grafiku zasnovanu na uzorku, da bi se povećao kvalitet finalnog proizvoda.

1.6 Šta označava pojam interaktivne računarske grafike?

Interaktivna računarska grafika podrazumeva dinamički način prikaza slike na računaru uz aktivno učešće čoveka u stvaranju i izmeni slike, gde su rezultati odmah vidljivi. Korisnik kontroliše sadržaj, strukturu i izgled objekata i njihovih slika koje se prikazuju korišćenjem brze vizuelne povratne informacije.

1.7 Koja je razlika između adresivosti i rezolucije uređaja za prikaz?

Adresivost je broj pojedinačnih tačaka po inču koje mogu biti kreirane. Može da se razlikuje horizontalna adresivost i vertikalna adresivost.
Rezolucija je broj razlučivih od strane posmatrača ili uređaja različitih linija po inču (na primer, naizmenično crnih i belih) koje uređaj može da kreira. Rezolucija ne može biti veća od adresivosti.

1.8 Kako se generiše slika kod vektorskih sistema?

Kod njih se linija dobija tako što se digitalne koordinate krajnjih tačaka transformišu u analogni napon za elektronski zrak koji pada na površinu ekrana. Slika se osvežava obično 30 do 60 puta u sekundi (30-60 Hz). Na vektorskim monitorima moguće je iscrtati i do 100000 linija. Pritom su linije glatke, ali se obojene površine teško prikazuju. Pogodni su samo za mrežne modele.

1.9 Kakav izgled imaju kose linije u vektorskim, a kakav u rasterskim sistemima?

Kod vektorskih sistema su linije glatke, a u rasterskim sistemima kose linije su „stepenaste“.

1.10 Čemu služi frejm bafer? Čemu služi video kontroler?

Na *frejm bafer* se može gledati kao na računarsku memoriju organizovanu u vidu dvodimenzionog niza tako da svaka adresibilna lokacija (x, y) odgovara jednom pikselu. U njemu se čuva sadržaj koji se zatim prikazuje na ekranu.
Video kontroler pristupa frejm baferu i prikazuje liniju po liniju na ekranu. On je zadužen da stalno osvežava sadržaj ekrana.

1.11 Navesti bar četiri oblasti u kojima se koristi računarska grafika.

Grafički korisnički interfejsi, Interaktivna izrada crteža u nauci i tehnologiji, CAD i CAM, Simulacije, Industrija zabave, Industrijski dizajn, Virtuelna realnost…

2.1 Šta je rasterizacija?

Rasterizacija je konvertovanje projektovane primitive u skup piksela.

2.2 Kako bi izgledala vektorska, a kako rasterska slika dva pravougaonika sa stranicama paralelnim koordinatnim osama? Koja od ove dve slike bi zauzimala manje memorije?

Kod vektorske grafike je slika predstavljena kao kolekcija geometrijskih figura, a kod rasterske je ekranu pridružena matrica piksela.
Vektorska slika bi zauzimala manje memorije, zbog rezolucije i bitske dubine kod rasterske slike.

2.3 Navesti primer gde se u praksi koristi vektorska grafika, a gde rasterska.

Vektorska se koristi za fontove, logotipove, štampani materijal, animacije… Rasterska se koristi za fotografije, u veb dizajnu...

2.4 Kako ocenjujemo kvalitet dobijene rasterske slike?

Kvalitet jedne rasterske slike određuje ukupan broj piksela (rezolucija) kao i broj vrednosti za svaki pojedinačni piksel (dubina boje).

2.5 Koji su zahtevi kod crtanja linije?

Niz piksela treba da bude što bliži idealnoj liniji i treba da prolazi kroz krajnje tačke. Slika duži treba da bude nezavisna od redosleda kojim su date njene krajnje tačke. Crtanje treba da bude što je moguće brže. Sve linije treba da budu iste osvetljenosti i sve tačke svake linije treba da budu iste osvetljenosti (ili da tako izgledaju) bez obzira na njihov nagib i dužinu.

2.6 Ako je koeficijent pravca prave (a) m = 1/2, (b) m = 3, da li crtamo po jedan piksel u svakom redu ili koloni?

a) $m = ½ < 1$ → po jedan piksel u svakoj koloni
b) $m = 3 > 1$ → po jedan piksel u svakom redu

2.7 Koliko piksela (računajući i krajnje tačke) treba da bude uključeno ukoliko na rasterskom uređaju treba linijom spojiti tačke: (a) (1,2) i (53,30) (b) (1,2) i (22,50) (c) (1,2) i (31,32)

a) max(53-1, 30-2) + 1 = 53
b) max(22-1, 50-2) + 1 = 49
c) max(31-1, 32-2) + 1 = 31

2.8 Kako radi algoritam grube sile za crtanje duži?

Najjednostavnija strategija za crtanje duži određene tačkama (x_0, y_0) i (x_1, y_1) sastoji se od narednih koraka:

- izračuna se koeficijent pravca: $m = \frac{(y1 - y0)}{(x1 - x0)}$
- x se povećava za 1 počev od x koordinate početne tačke $(x_0\ y_0)$ duži
- za svako x_i računa se vrednost $y_i = mx_i + B = m(x_i - x_0) + y_0$
- osvetljava se piksel sa koordinatama $(x_i, round(y_i))$, pri čemu je $round(y_i) = \lfloor y_i + 0.5 \rfloor$

2.9 Koji su nedostaci algoritma grube sile za crtanje duži? Na koji način se oni mogu eliminisati?

Ako je $x_1 < x_0$ ništa se ne crta (rešenje: promeniti poredak tačaka ukoliko je $x_1 < x_0$). Algoritam je neefikasan zbog broja operacija i korišćenja realne aritmetike - svaki korak zahteva oduzimanje, sabiranje, množenje i zaokruživanje jer je y promenljiva realnog tipa.

2.10 Koja je ideja inkrementalnih algoritama?

U svakom koraku, izračunavanja pravimo na osnovu prethodnog koraka.

2.11 Kako funkcioniše midpoint algoritam za crtanje duži?

Bresenham je razvio algoritam za crtanje duži koji koristi samo celobrojnu aritmetiku. Ista tehnika može da se koristi i za krug. Pitteway i Van Aken su razvili midpoint algoritam koji se za duži i celobrojne krugove ponaša isto kao Bresenhamov algoritam, ali može da se uopšti na proizvoljne krive drugog reda.
Pretpostavimo da smo označili piksel P (x_p, y_p) i da treba da izaberemo između piksela jednu poziciju desno (E=east) i piksela jednu poziciju desno i jednu poziciju iznad (NE=northeast).

2.12 Koja je ključna ideja midpoint algoritma za crtanje duži?

Odluku pravimo između tačno 2 piksela.

2.13 Ukoliko je $1 < m < \infty$, između kojih tačaka vršimo odabir?

2.14 Čemu nam u midpoint algoritmu služi tačka M između tačaka N i NE?

Ako je tačka M “ispod” prave koja sadrži duž, onda je pravoj bliža NE, a inače E.

2.15 Šta je promenljiva odlučivanja?

Da bismo odredili položaj tačke M potrebno je jedino odrediti znak izraza $F(M) = F(x_p+1, y_p+ ½)$. S obzirom na to da se odluka bazira na vrednosti ove funkcije, vrednost $d = F(x_p+1, y_p+ ½)$ zovemo promenljiva odlučivanja. Ako je $d < 0$, onda se tačka M nalazi iznad prave, te biramo piksel E; ako je $d > 0$, onda se tačka M nalazi ispod prave i biramo piksel NE; ako je $d = 0$, onda je sve jedno i, po dogovoru, biramo piksel E.
Promenljiva odlučivanja je vrednost implicitne jednačine prave u središnjoj tački.

2.16 Zašto prvu vrednost promenljive odlučivanja množimo sa 2 u realizaciji midpoint algoritma?

Da bismo dobili celobrojnu vrednost.

2.17 Izračunaj prvu vrednost promenljive odlučivanja za duž koja povezuje tačke sa koordinatama (1,2) i (7,4).

$d_{start} = 2*dy – dx = 2*2 – 6 = 4 – 6 = -2$

2.18 Ako se midpoint algoritmom iscrtava duž između tačaka (2,4) i (12,9), čemu je jednaka početna vrednost promenljive odlučivanja?

$d_{start} = 2*dy – dx = 2*10 – 5 = 20 – 5 = 15$

2.19 Kako funkcioniše algoritam grube sile za iscrtavanje kruga? Koji su njegovi nedostaci?

· Iz jednačine kruga: $x^2 + y^2 = R^2$, sledi: $y = \pm\sqrt{R^2 - x^2}$

· Za $i = 1, 2, 3,\ldots$, pri čemu se x_i inkrementira za po 1, treba osvetliti tačku: $(x_i, [\sqrt{R^2 - x_i^2} + 0.5])$

Tačno (koliko je to moguće), ali neefikasno.

· Sličan neefikasan metod, korišćenjem parametarske jednačine kruga: crtati tačke

$$([R\cos\varphi_i + 0.5], [R\sin\varphi_i + 0.5]) \quad \text{za } \varphi_i = 1^\circ, 2^\circ, 3^\circ, \ldots, 90^\circ.$$

· Kako vrednost x raste, povećavaju se praznine između tačaka; to je lako rešiti korišćenjem i simetrije kruga po još jednoj osi (pravoj y = x).

2.20 Šta se menja ukoliko krug nije u koordinatnom početku?

Razlikuje se samo početni piksel – umesto (0, R), prvi uključeni piksel treba da bude (a, R+b), a umesto uslova y > x, treba koristiti uslov y-b > x-a, ili efikasnije y > x+(b-a).

2.21 Zašto se u midpoint algoritmu za crtanje kruga razmatra samo osmina kruga?

Ostali slučajevi se obrađuju simetrično; efikasnije.

2.22 Koju vrednost u algoritmu rasterizacije kruga zovemo promenljiva odlučivanja?

$d = F(x_p+1, y_p- ½) = (x_p+1)^2 + (y_p- ½)^2 – R^2$

2.23 Koja je suštinska razlika u vrednosti promenljive odlučivanja kod crtanja kruga u odnosu na crtanje duži?

Prva vrednost za d kod crtanja kruga nije celobrojna, a kod crtanja duži jeste.

2.24 U unapređenoj verziji midpoint algoritma za crtanje kruga, zašto je neophodno da svaki put ažuriramo obe promenljive ΔE i ΔSE?

Ne znamo koja će nam biti potrebna u daljem radu.

2.25 Koje veličine nazivamo razlikama drugog reda?

Vrednosti za koje se uvećavaju promenljive ΔE i ΔSE su razlike drugog reda.

2.26 Koja je vremenska složenost midpoint algoritma za crtanje: (a) duži, (b) kruga?

a) linearna po broju piksela koji će biti iscrtani

b) linearna u funkciji poluprečnika kruga

3.1 Kada za region kažemo da je 4-povezan, a kada da je 8-povezan?

Kažemo da je region *4-povezan* ako se svaka dva piksela tog regiona mogu povezati nizom piksela korišćenjem samo poteza nagore, nadole, ulevo i udesno. Nasuprot ovome, region je *8-povezan* ako se svaka dva piksela tog regiona mogu povezati nizom piksela korišćenjem poteza nagore, nadole, ulevo, udesno, gore-levo, gore-desno, dole-levo, dole-desno.

3.2 Kako se zadaje region definisan unutrašnjošću, a kako region definisan granicom?

Region definisan unutrašnjošću je najveći povezani region piksela čija je boja ista kao boja piksela P. Region definisan granicom je najveći povezani region piksela čija boja nije jednaka nekoj graničnoj vrednosti boje.

3.3 Na kakve regione se primenjuje flood-fill algoritam a na kakve boundary-fill algoritam? Kako svaki od njih funkcioniše?

FloodFill – algoritam za popunjavanje regiona definisan unutrašnjošću. Kreće se od početnog piksela u unutrašnjosti (seed-a), proverava se da li je boja piksela stara, ako jeste piksel se boji novom bojom i pomeramo se iz ovog piksela u 4 (ili 8) smerova. Granica oblasti se može sastojati od različitih boja. Ovakva implementacija je praktično neupotrebljiva.
BoundaryFill – algoritam za popunjavanje regiona definisanog granicama. Kreće se od početnog piksela u unutrašnjosti, proverava se da li je boja piksela različita od granične vrednosti boje i od nove vrednosti boje, ako jeste piksel se boji novom bojom i pomeramo se iz ovog piksela u 4 (ili 8) smerova. Rekurzivna priroda algoritma može prouzrokovati prerani prekid rada ako se nova boja pojavi u regionu koji je definisan granicom.

3.4 Navesti primer gde boundary-fill algoritam neće ispravno obojiti region.

Neće dati dobar rezultat ako je unutrašnjost regiona podeljena na 2 dela (granica koja ih deli je drugačije boje od unutrašnjosti).

3.5 Koje su složenosti ovi algoritmi?

3.6 Šta je scan linija? Koliko scan linija se razmatra u algoritmu za scan popunjavanje poligona?

Scan linija je horizontalna linija na rasterskom sistemu.

3.7 Kod scan popunjavanja poligona koje se presečne tačke scan linije sa stranicama poligona broje, a koje ne?

Kod tačaka sa neparnim stanjem treba započeti bojenje, a kod tačaka sa parnim treba ga zaustaviti.

3.8 Na koji način obrađujemo scan linije koje prolaze kroz neko teme poligona?

Za stranicu poligona računamo tačku preseka sa najmanjom y koordinatom, a ne računamo tačku preseka sa najvećom y koordinatom parnost se menja osim ako je u pitanju lokalni minimum/maksimum.

3.9 Koje se tačke boje kod horizontalnih stranica poligona?
Kod horizontalnih preseka ne bojimo nijednu tačku preseka. Efekat ovog pravila je da se donje horizontalne stranice boje, a gornje ne; takođe ne boje se i desne tačke stranica.

3.10 Šta je sliver?
Sliver je uska poligonalna oblast koja sadrži 0 ili 1 piksel za neke scan linije. Ovo je posledica toga da se boje samo pikseli koji su u unutrašnjosti ili na levim ili donjim stranicama poligona. Antialiasing tehnika se može koristiti za rešavanje ovog problema.

3.11 Kako se računa presek (i+1)-ve scan linije na osnovu na i-te scan linije?
Koristi se inkrementalni pristup da bi se izračunali preseci jedne scan linije na osnovu prethodne scan linije, bez toga da analitički računamo presek scan linije sa svakom stranicom poligona. Ideja je nalik midpoint algoritmu, samo je potrebno uvek birati tačke u unutrašnjosti poligona.

3.12 Šta označava koherentnost scan linija, a šta koherentnost stranica?

3.13 Na koji način prelazimo na celobrojnu aritmetiku – koje dve celobrojne vrednosti razmatramo i na koji način vršimo obradu?
Na celobrojnu aritmetiku prelazimo određivanjem preseka scan linije sa levom stranicom poligona. Razmatramo slučaj kada je m > 1 i razmatramo posebno celi i razlomljeni deo vrednosti:

$$x_i + \frac{x_{max} - x_{min}}{y_{max} - y_{min}}$$

3.14 Koja je vremenska složenost algoritma LeftEdgeScan?
Linearna, $O(y_{max}-y_{min})$

3.15 Koje se tačke preseka dobijaju algoritmom LeftEdgeScan za duž sa krajnjim tačkama (2, 3) i (5, 8)?
Iz algoritma: (2, 3), (3, 4), (4, 5), (5, 6), (6, 7), (7, 8)

3.16 Zašto u algoritmu LeftEdgeScan vrednost promenljive inkrement kreće od vrednosti imenilac?
To nam je važno za zaokruživanje na unutrašnji piksel. Nakon što mu dodamo brojijlac, ispitujemo da li je došlo do prekoračenja i zaokružujemo na sledeći piksel.

3.17 Kako bi se izmenio algoritam ukoliko bismo računali presek sa desnom stranicom poligona?

4.1 Na kakav kliping region se primenjuje Cohen-Sutherland-ov algoritam?
Na pravougaonik.

4.2 Na koliko regiona se u ovom algoritmu deli ravan? Šta svaki od bitova u 4-bitnom kodu tačke označava?
Ravan se deli na 9 regiona. Prvi bit (sdesna) označava da se prava nalazi iznad gornje granice regiona; drugi bit – ispod donje granice; treći bit – desno od desne granice; četvrti bit – nalazi se levo od leve granice kliping regiona.

4.3 Koji kôd imaju tačke u unutrašnjosti kliping regiona?
0000

4.4 Kod Cohen-Sutherlandovog algoritma šta važi za duž koja se secka ako je $K_1=0$ i $K_2=0$? Šta važi ako je $K_1 \& K_2 <> 0$?
Za prvu važi da se nalazi unutar kliping regiona, a za drugu da se nalazi van i trivijalno se odbacuje.

4.5 Koja je vremenska složenost Cohen-Sutherland algoritma?
Konstantna složenost.

4.6 Na kakve kliping regione se može primeniti Cyrus-Backov algoritam?
Primenljiv je na proizvoljni konveksni kliping poligon.

4.7 Koji uslov važi za presečnu tačku duži P_0P_1 i normalu stranice?
$N \cdot (P(t) - P_E) = 0$

4.8 Da bi se tačka preseka klasifikovala kao potencijalno ulazna/izlazna razmatra se odnos koja dva vektora?
Vektora $D = P_0P_1$ i vektora N – normala stranice poligona usmerena ka spoljašnosti poligona.

4.9 Šta treba da važi da bi tačka bila okarakterisana kao potencijalno ulazna/izlazna?
Potencijalno ulazna: $N \cdot D < 0$
Potencijalno izlazna: $N \cdot D > 0$, $D = P_0P_1$

4.10 Šta znači kada je $t_L < t_E$?
Tada ne postoji presek.

4.11 Koja je vremenska složenost Cyrus-Beckovog algoritma?
Složenost je linearna po broju temena poligona.

4.12 Na kakav se region može primeniti Liang-Barsky varijanta algoritma za kliping?
Na pravougaonik sa horizontalnim/vertikalnim stranicama.

5.1 Kako se može jednostavno utvrditi da li duž P_0P_1 seče ravan datu jednačinom Ax + By + Cz + D = 0?

Zamenom P_0 i P_0 u jednačinu ravni, treba dobiti rešenja suprotnog znaka.

3.2 Koliki je ugao izmedju vektora u(−8, 6) i v(3, 4)?

alfa = arccos((-8*3 + 6*4)/(10*5)) = arccos(0)

5.3 Iz kog razloga se prelazi na rad sa homogenim koordinatama? Navesti bar dve različite reprezentacije Dekartove tačke (−3, 2) homogenim koordinatama.

Bilo bi dobro da translacija, skaliranje, smicanje i rotacija imaju istu formu. Koristeći homogene koordinate sve ove transformacije imaju formu množenja matrica. Uz pomoć homogenih koordinata 2D tačka se predstavlja trojkom vrednosti (x, y, W).
(-3, 2, 6) ili (-3/6, 2/6, 1)=(½, 1/3, 1)

5.4 Koja je transformacija inverzna smicanju za faktor *a* po x osi?

Smicanje za faktor -*a*.

5.5 Navesti primer situacije u kojoj je pogodno izvršiti promenu koordinatnog sistema.

Ovaj pristup je pogodniji kada više objekata u svojim pojedinačnim koordinatnim sistemima treba objediniti u jedinstven koordinatni sistem.

5.6 Od čega se sastoji graf scene?

Čvorova objekata (kocke, valjci, lopte, . . .) koji su podrazumevano jedinične veličine i pozicionirani u koordinatnom početku; čvorova atributa (boja, tekstura, . . .); čvorova transformacija.

5.7 Čemu služe čvorovi grupisanja u grafu scene?

Ako imamo više sličnih komponenti na sceni, moguće je ponovo iskoristiti već definisane grupe objekata.