Generisanje građevina u računarskoj igri „Minecraft” korišćenjem Shap-e modela

Marko Milenković

Sadržaj - Napravljena je praktična implementacija Shap-e modela kao nadogradnja računarske igre „Minecraft”. Konačni rezultati su izgenerisane građevine na osnovu tekstualnog upita, sa ciljem da se smanji količina vremena koju profesionalne ekipe igrača troše na građevinske projekte u ovoj računarskoj igri. Primenjeno je uzorkovanje STF funkcije za pravljenje mreže *voxel*-a. Nakon toga je urađeno smanjeno uzorkovanje da bi se uklonile anomalije i praznine. Za sva računanja i poređenja vezana za boje i teksture je korišćen CIE lab prostor boja zbog boljeg preklapanja sa ljudskom percepcijom boja i linearnosti u poređenju sa RGB prostorom boja. Da bi se napravilo mapiranje boja na blokove korišćen je *median cut* algoritam za pronalaženje boje koja najbolje predstavlja blok. Ta predstavljajuća boja se koristi za računanje euklidske distance boja u TF od Shap-e objekta i blokova u igri, time dobijajući podudarno predstavljanje boja Shap-e generisanog objekta unutar igre korišćenjem Minecraft blokova. Konačni rezultat je zadovoljavajući, uzevši u obzir ograničenja Shap-e generativnog modela, i pokazana je praktična primena Shap-e modela.

I. Uvod

Minecraft je poznata računarska igra. Spada u kategoriju *voxel* igara [1]. Igra nema predodređen početak i kraj i pruža igraču mnogo različitih načina igranja igre, pa se zato naziva *sandbox* igrom.

Vremenom su nastale ekipe igrača koje su krenule isključivo da se bave građenjem struktura, okruženja i mapa. Rastom popularnosti igre rasla je i veličina projekata. Građevine su se pravile isključivo ručno ili uz asistenciju alatki. Nakon korišćenja tih alatki je još uvek bilo potrebno ručno izmeniti detalje. Takav pristup je doveo do toga da je potrebno izuzetno mnogo vremena da se napravi neka struktura, pogotovo ako je velikih razmera.

Cilj istraživanja je da se napravi alatka kojom bi se ubrzalo ili automatizovalo pravljenje struktura. Time bi se igračima olakšao rad i smanjila bi se količina vremena potrebna za realizaciju projekata. Uz razvoj veštačke inteligencije su se kao moguće rešenje pokazali uslovno generativni algoritmi.

M. Milenković, student 1. godine na Fakultetu tehničkih nauka, Trg Dositeja Obradovića 6, 21102, Novi Sad, Srbija / polaznik programa Računarstva u Istraživačkoj stanici Petnica, Selo Petnica, 14104 Valjevo, Srbija, E-mail: mmarkomile@gmail.com

Za generisanje objekata korišćen je OpenAI uslovni generativni model za 3D objekte *Shap-e* [2]. Shap-e generiše skup parametara implicitno zadate funkcije [3] koja se može predstaviti kao model ili *neural radiance field* (NeRF) [4]. U ovom radu je korišćen teksturirani model.

Shap-e model je izabran jer u poređenju sa drugim uslovnim generativnim modelima [5][6][7][8] brzo izvršava generaciju, a pruža zadovoljavajuće rezultate. U poređenju sa *Point-e* modelom [8], koji ima sličnu primenu, brži je i proizvodi slične ili bolje rezultate.

Shap-e proizvodi SDF (*signed distance function*) [9] i TF (*texture field*) [10] koji se koriste u uzorkovanju funkcije da bi se napravila teksturizovana mreža *voxel*-a.

Da bi se boje objekta koji Shap-e izgeneriše uvezle u Minecraft, potrebno je napraviti mapiranje boja na blokove. Najjednostavniji način da se ovo uradi je uporediti euklidske distance RGB [11] vrednosti i prosečne boje teksture svakog bloka u Minecraft-u. Međutim, ovaj pristup ne naglašava preciznost i ne poklapa se dovoljno sa ljudskom percepcijom sličnosti boja. Zato je korišćen CIE lab [12] prostor boja [13] za mapiranje boja na blokove i *median cut algoritam* [14] za pronalaženje predstavljajućih boja blokova. Uveden je i parametar standardne devijacije [15] nad teksturama blokova. Standardnom devijacijom može se predstaviti koliko je neki blok šaren. Šareni blokovi su neželjeni jer nemaju jasno definisanu boju koja ih predstavlja.

II. Priprema za obradu

A. Eliminacija nepoželjnih blokova

Da bi objekti izgledali prihvatljivo u Minecraft-u, potrebno je eliminisati određene grupe blokova iz selekcije. Zbog toga su uvedeni različiti kriterijumi: standardna devijacija boja, providnost i da li je taj blok dekorativan.

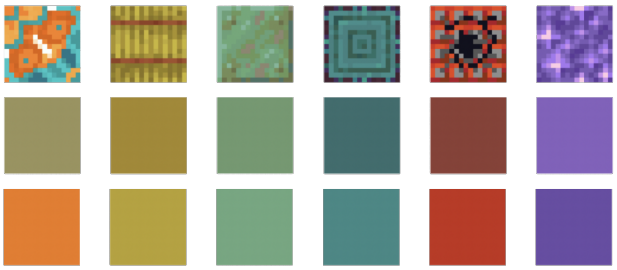
Pojedinačni blokovi su takođe eliminisani. To su problematični blokovi kao što su vrata (koja se rasprostiru preko 2 koordinate), korali (jer izgube boju kada su na suvom) i blokovi sa gravitacijom.

Polublokovi, zidovi i stepenice nisu ubačeni u selekciju. Za sada je u Minecraft-u prilično ograničena raznovrsnost takvih blokova i postoji mali broj različitih boja i materijala za njih. Zbog toga nisu jako značajni.

B. Mapiranje blokova na boje

Mapiranje blokova na boje i kasnije upoređivanje boja na izgenerisanoj TF sa blokovima koji najbliže odgovaraju je jedan od najbitnijih koraka. Korišćen je CIE lab prostor boja.

Iako računanje prosečne boje teksture bloka može u određenim uslovima pokazati zadovoljavajuće rezultate, to nije dobar način da se traži boja koja najbolje predstavlja neku sliku. *Median cut* algoritam može da pronađe boju koja najbolje predstavlja sliku. Vidi se značajna razlika kada gledamo predstavljajuću boju šarenih blokova datoj u primeru prikazanom na slici 1.



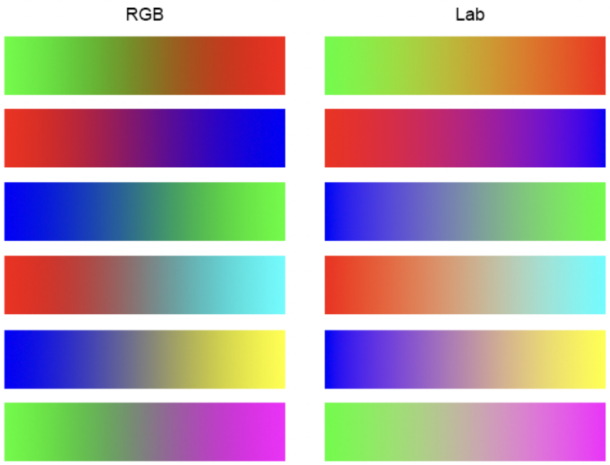
Slika 1. Neki primeri blokova (prvi red), njihovih predstavljajućih boja dobijenih prostim prosekom (drugi red) i korišćenjem *median cut* algoritma (treći red)

Posterizacija slike je pretvaranje slike sa velikim brojem tonova u sintetičku sliku sa izraženim ravnim površinama i manjim brojem tonova. *Median cut* algoritam je jedan od algoritama za posterizaciju koji pronalazi ograničenu paletu boja kojom se posterizuje slika tako da najviše liči na original. Algoritam je iskorišćen da se odabere najučestalija boja od dobijenih boja u paleti teksture nekog bloka, da bi se dobila boja koja najbolje predstavlja taj blok. Da bi se mapirao svaki blok, algoritam se pokreće na svakoj od tekstura blokova koji se koriste pojedinačno. Od parova predstavljajućih boja i naziva blokova se pravi mapiranje koje se dalje koristi za upoređivanje sa bojama iz TF.

Gradijenti između boja imaju znatno blaži prelaz u CIE lab prostoru boja u poređenju sa RGB prostorom boja i više se poklapaju sa ljudskom percepcijom. Ovo se može videti na grafičkom prikazu gradijenata boja (slika 2).

Zbog toga je za svako računanje i poređenje korišćen CIE lab prostor boja. Prebacivanje je odrađeno korišćenjem *skimage* biblioteke koja prebacuje sRGB boje u CIE XYZ (koji je derivat CIE RGB prostora boja) prostor boja [16] [17]. Postupak počinje normalizacijom boja u vrednosti od 0 do 1 (deljenjem sa 255). Zatim se vrši konverzija sRGB → XYZ.

CIE XYZ je potrebno prebaciti u CIE lab prostor boja. Da bi se to odradilo mora se definisati *illuminant* (osvetljivač) [18]. Osvetljivač predstavlja teoretski izvor svetlosti i time daje osnovu za poređenje slika. U ovom konkretnom slučaju korišćen je uobičajeni osvetljivač D_{65} jer predstavlja dnevnu svetlost u podne (6500K). Kada je izabran osvetljivač, moguće je odraditi konverziju CIE XYZ → CIE lab.



Slika 2. linearna interpolacija od leve do desne boje u RGB i CIE lab prostorima boja

III. Obrada

A. Shap-e generacija

Shap-e model je obučen da na osnovu tekstualnog upita koji je zadat na prirodnom jeziku izgeneriše objekat koji je opisan u upitu. S obzirom da je skup podataka koji je korišćen za obučavanje Shap-e modela uglavnom sačinjen iz objekata koji su poprilično jednostavni, najbolje rezultate daje kada su i tekstualni upiti jednostavni.

Iz izlazne vrednosti modela se izdvajaju SDF i TF korišćenjem funkcija predstavljenih uz Shap-e [2]. Da bi se SDF i TF iskoristile u Minecraft-u potrebno je od njih napraviti mrežu *voxel*-a.

B. Uzorkovanje funkcije

Da bi se dobila mreža *voxel*-a, odrađeno je uzorkovanje funkcije. Pravi se binarna trodimenzionalna matrica, gde je svaki element indikator postojanja *voxel*-a na datoj koordinati iz SDF. Ako je vrednost u SDF na datim koordinatama >=0 onda se postavlja na 1.

C. Smanjeno uzorkovanje

Zbog pojave veoma tankih elemenata u objektima, ponekad se može desiti da nastanu rupe u konačnom objektu kada se on prebaci iz SDF u mrežu *voxel*-a. Jedan od metoda rešavanja ovog problema je smanjeno uzorkovanje sa veće rezolucije.

Uzorak iz SDF je uzet u 3 puta većoj rezoluciji od konačne (željene) rezolucije, zatim je urađeno sažimanje pomoću srednje vrednosti [19] nad tom matricom sa dimenzijama jezgra 3x3x3. Posledica je trodimenzionalna matrica željenih dimenzija. Ovako je smanjena količina nepravilnosti u konačnom rezultatu uz minimalan uticaj na performanse.

Smanjeno uzorkovanje primenjeno nad teksturama ne utiče previše na konačan rezultat, pa iz tog razloga nije rađeno nad teksturama.

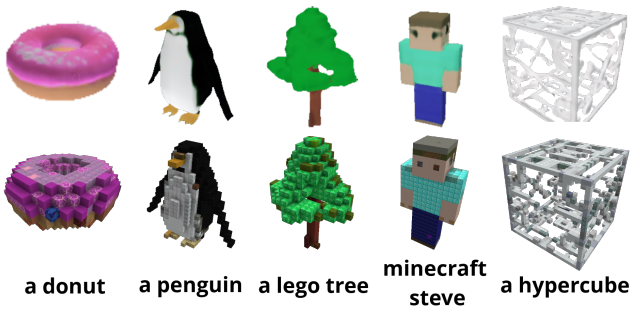
D. Postavljanje blokova u Minecraft-u

Poslednji korak je postavljanje blokova u Minecraft-u. Ovo je implementirano jednostavnom Python skriptom koja se povezuje na Minecraft server korišćenjem RCON port-a i poziva komandu „/setblock X Y Z minecraft:block”. X, Y i Z su koordinate gde se postavlja blok, a *minecraft:block* je blok koji treba da se postavi na osnovu vrednosti boje na tim koordinatima. Da bi se našao blok koji će da se postavi na osnovu boje iz TF na datim koordinatima, računamo euklidsku distancu za svaki od blokova. Koristi se blok čija predstavljajuća boja ima najmanju euklidsku distancu do proizvoljne boje iz TF.

IV. Rezultati i diskusija

A. Evaluacija rezultata

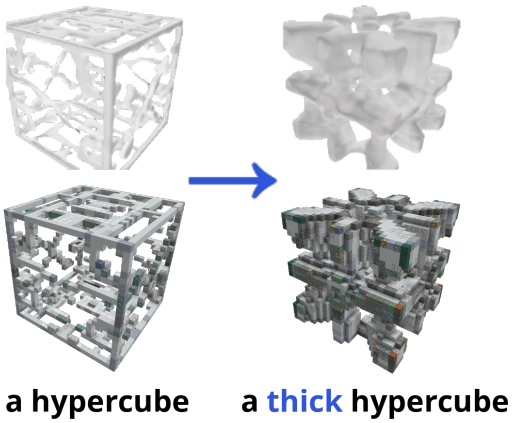
Za evaluaciju rezultata su upoređene izlazne slike Shap-e modela i odgovarajuća građevina u Minecraft-u. Može se videti izrazita sličnost (slika 3).



Slika 3. prikaz različitih primera kao Shap-e objekat (prvi red) i u Minecraft-u (drugi red) zajedno sa upitima

Vidi se da u kompleksnim strukturama kao što je „hypercube” nastaju problemi u generaciji Shap-e modela.

Nekada je potrebno prilagoditi upit da bi objekat izgledao kao što je zamišljeno i da bi se adekvatno prebacio u Minecraft. Može biti korisno dodavati pojmove kao „lego” (slika 3) kako bi izgled izgenerisanog objekta bio bolje prilagođen za prebacivanje u Minecraft. Iz *hypercube* primera može se lako zaključiti da postoji nekakav problem, pa je jedno od mogućih rešenja sakriveno u menjanju upita. Za primer razlike pri menjanju upita je dodat pridev „thick”, koji značajno menja rezultat, što se vidi na slici 4.



Slika 4. razlika pri menjanju upita

Razlika između korišćenja RGB prostora boja i CIE lab prostora boja je veoma značajna. Može se videti da je CIE lab prostor boja mnogo sličniji onome što bi se očekivalo i nema velikih skokova boje od svetlije do tamnije strane spektra (slika 5).

Slika 5. poređenje rezultata kada se koristi RGB i kada se koristi CIE lab prostor boja

Sličnost građevina i objekata iz modela je zadovoljavajuća. Primećuje se sličnost boja i nema ozbiljnih artefakta na jednostavnim upitima.

Vreme potrebno da Shap-e izgeneriše model se kreće oko 20-30 sekundi na Nvidia Tesla T4 grafičkoj. Ovakvi rezultati se mogu smatrati dovoljno dobrim za praktičnu upotrebu u igri.

V. Ograničenja i budući rad

Trenutni uslovno generativni modeli za generisanje trodimenzionalnih objekata imaju brojna ograničenja. Moguće ih je podeliti na „brze” [8] i „spore” [5][6][7]. Brzi modeli još uvek podržavaju samo jednostavne upite i ne raspoznaju komplikovanije opise i strukture. Iako modeli razumeju jednostavne opise, ograničeno podržavaju kompoziciju pojmova. Spori modeli nisu praktični za primene u igrama ili u *real-time* scenarijima, kada je potrebno da se brzo vizuelizuje rezultat.

U ovom radu nisu razmatrani nepravilni blokovi (polublokovi, stepenice, zidovi i dekorativni blokovi), ali bi bilo moguće implementirati ih u budućem radu uz nekakav sistem pravila i marching cubes algoritmom umesto uzorkovanja funkcije. Marching cubes algoritam bi mogao biti prilagođen Minecraft blokovima. Ovo bi takođe omogućilo jednostavno dodavanje podrške za nove blokove u budućnosti.

VI. Zaključak

Shap-e algoritam pruža zadovoljavajuće rezultate kada se njegovi izlazni objekti pretvore u Minecraft građevine. Napravljen je način da se ubrzaju građevinski projekti u Minecraft-u i da se igračima olakša građenje određenih objekata. Pokazano je da savremeni uslovno generativni modeli pored svojih ograničenja mogu da naprave zadovoljavajuće rezultate u kratkom vremenskom intervalu i pronađu praktičnu primenu.

Reference

[1] “Tutorials/units of Measure: Distance,” Minecraft Wiki, https://minecraft.wiki/w/Tutorials/Units_of_measure#Distance (pristupljeno 14. oktobra 2023).

[2] Jun, H. and Nichol, A., 2023. Shap-e: Generating conditional 3d implicit functions. arXiv preprint arXiv:2305.02463.

[3] Krantz, S.G. and Parks, H.R., 2002. The implicit function theorem: history, theory, and applications. Springer Science & Business Media.

[4] Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R. and Ng, R., 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM, 65(1), pp.99-106.

[5] Lin, C.H., Gao, J., Tang, L., Takikawa, T., Zeng, X., Huang, X., Kreis, K., Fidler, S., Liu, M.Y. and Lin, T.Y., 2023. Magic3d: High-resolution text-to-3d content creation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 300-309).

[6] Poole, B., Jain, A., Barron, J.T. and Mildenhall, B., 2022. Dreamfusion: Text-to-3d using 2d diffusion. arXiv preprint arXiv:2209.14988.

[7] Gao, J., Shen, T., Wang, Z., Chen, W., Yin, K., Li, D., Litany, O., Gojcic, Z. and Fidler, S., 2022. Get3d: A generative model of high quality 3d textured shapes learned from images. Advances In Neural Information Processing Systems, 35, pp.31841-31854.

[8] Nichol, A., Jun, H., Dhariwal, P., Mishkin, P. and Chen, M., 2022. Point-e: A system for generating 3d point clouds from complex prompts. arXiv preprint arXiv:2212.08751.

[9] Chan, T. and Zhu, W., 2005, June. Level set based shape prior segmentation. In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) (Vol. 2, pp. 1164-1170). IEEE.

[10] Oechsle, M., Mescheder, L., Niemeyer, M., Strauss, T. and Geiger, A., 2019. Texture fields: Learning texture representations in function space. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 4531-4540).

[11] Süsstrunk, S., Buckley, R. and Swen, S., 1999. Standard RGB color spaces. In Proc. IS&T;/SID 7th Color Imaging Conference (Vol. 7, No. CONF, pp. 127-134).

[12] “CIE 1976 (L *a*b*) colour space,” u CIE Colorimetry 15 (Third ed.), CIE, 2004, ISBN 3-901-906-33-9.

[13] Joblove, G.H. and Greenberg, D., 1978, August. Color spaces for computer graphics. In Proceedings of the 5th annual conference on Computer graphics and interactive techniques (pp. 20-25).

[14] Heckbert, P., 1982. Color image quantization for frame buffer display. ACM Siggraph Computer Graphics, 16(3), pp.297-307.

[15] Bland, J.M. and Altman, D.G., 1996. Measurement error. BMJ: British medical journal, 312(7047), p.1654.

[16] International Commission on Illumination. (1932). Commission internationale de l'éclairage ... huitième session cambridge--septembre 1931. University Press.

[17] T S. & J G. (1931). The c.i.e. colorimetric standards and their use. Transactions of the Optical Society 73–134. https://doi.org/10.1088/1475-4878/33/3/301

[18] Noboru, O. and Robertson, A.R., 2005. 3.9: Standard and Supplementary Illuminants, Colorimetry. Wiley Hoboken, NJ, USA. https://doi. org/10, 1002(04700), p.94745.

[19] “Papers with code - average pooling explained,” Papers With Code, https://paperswithcode.com/method/average-pooling (pristupljeno 15. oktobra 2023).