Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 655
DOI: https://doi.org/10.24867/31EF01Tapai

PROCEDURALNO GENERISANJE OKRUŽENJA VIDEO IGRE

PROCEDURAL GENERATION OF VIDEO GAME ENVIRONMENTS

Kristina Tapai, Bojan Banjanin, *Fakultet tehničkih nauka, Novi Sad*

Oblast – GRAFIČKO INŽENJERSTVO I DIZAJN

Kratak sadržaj – *U radu su predstavljene mogućnosti procedrualne generacije okruženja video igara koristeći gejm endžin Unreal Engine. Kreiran je sistem koji generiše teren koristeći gotove 3D modele, kao i sistem koji generiše modele koji se koriste za kreiranje terena. Predstavljeni su sistemi koji proceduralno postavljaju modele i sisteme čestica na generisani teren. Analizirane su performase kreiranih procedrualnih sistema.*

Ključne reči: *Proceduralna generacija u video igrama, dizajn okruženja, real-time rendering, kompjuterski generisana grafika.*

Abstract – *The paper presents the possibilities of procedural generation of video game environments using Unreal Engine game engine. Two systems were created, a system that generates terrain using ready-made 3D models, as well as a system that generates models used to create a terrain. Systems that procedurally place models and particle systems on the generated terrain were created. The performances of the created systems were analyzed.*

Keywords: *Procedural generation in video games, environment design, real-time rendering, computer generated graphics*

1. UVOD

Jedan od problema u razvoju video igara jeste potreba da se velika količina kvalitentog sadržaja proizvede za što kraće vreme. Proceduralna generacija sadržaja je pristup razvoju video igara koji se bavi upravo ovim problemom. Upotrebom proceduralne generacije određeni delovi igre kreiraju se pomoću algoritama, odnosno njihovo kreiranje se automatizuje, čime se ubrzava proces. Pored toga proceduralna generacija u video igrama otvara nove mogućnosti vezano za dizajn i mehaniku igre, poput adaptivnog sadržaja i beskrajnih mapa. Procedrualna generacija je oblast koje se još uvek razvija, otkrivajući nove načine za upotrebu ovog pristupa razvoja video igara, kao i rezultate kombinacije procedrualne generacije sa drugim oblastima računarskih nauka.Osnovu proceduralne generacije predstavljaju algoritmi, odnosno setovi definisanih matematičkih ili logičkih pravila [1].

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Bojan Banjanin, docent.

Algoritmi funkcionišu na osnovu parametara i nasumičnih vrednosti čime se osigurava različitost generisanog sadržaja [1].

Gejm endžin Unreal Engine 5 nudi različite pristupe preoceduralne generacije sadržaja za video igre. Procedrualna generacija se može postići kreiranjem algoritama unutar C++ skripti ili korišćenjem sistema za vizuelno kodiranje Blueprint. Gejm endžin sadrži i komponente namenjene za proceduralno generisanje modela poput komponenti Procedruarl Mesh i Dynamic Mesh. Unutar Blueprint systema nalaze se Geometry Script kartice koje omogućavaju procedrualnu generaciju modela i njihovu manipulaciju. Sistem za proceduralnu generaciju, Procedrual Content Generation (u daljem tekstu PCG), omogućava kreiranje sistema za proceduralno postavljanje aseta unutar nivoa [2].

2. KREIRANJE ALGORITAMA ZA PROCEDURALNU GENERACIJU OKRUŽENJA

2.1. Generacija okruženja koristeći gotove modele

Proceduralna generacija okruženja se može upotrebiti za postavljanje delova okruženja, odnosno gotovih 3D modela, koji se međusobno uklapaju u celinu tako da se dobije konačan izgled okruženja. Ovim načinom je moguće automatizovati proces postavljanja objekata na nivo, ili je moguće dobiti nivoe različitog izgleda pri svakom pokretanju nivoa. Korišćenjem unapred kreiranih modela zadržava se kontrola nad izgledom elemenata okruženja [3].
Kako bi se postigla generacija terena pripremljeni su modeli, unutar programa za 3D modelovanje Blender, koje je moguće međusobno uklapati postavljanjem modela na kvadratnu mrežu. Kreirano je 7 različitih modela koji će se postavljati na polja mreže (slika 1). Veličina ovih modela po x i y osi je jednaka, kako ne bi došlo do razmaka između modela kada se postave na mrežu. Kreiran je i jedan model koji će se postavljati tako da se eliminišu vertikalni razmaci između polja (slika 1).

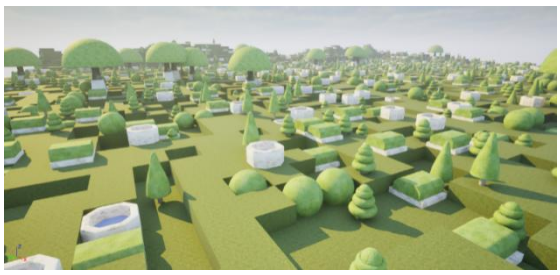
Slika 1. *Pripremljeni modeli*

Pored generacije terena na nivo će biti proceduralno postavljeni sistemi čestica kojima će se simulitati leptiri. Pripremljeni su 3D model i teksture potrebne za

animiranje materijala leptira koristeći program Blender. Teksture boje za model leptira kreirani su proceduralno koristeći program Substance Designer. Ovaj program omogućava kreiranje tekstura generisanjem različitog šuma i paterna jednostavih oblika i njihovu manipulaciju. Nakon pripreme modela i tekstura, unutar Unreal Engine-a je kreiran sistem čestica koristeći Niagara Particle System. Podešavanjem emitera postignuto je da svaki sistem čestica generiše dve čestice na čijem mestu se renderuje model leptira, čiji su životni ciklusi beskonačani. Čestice se kreću u prostoru određenog radijusa. Na kretanje leptira utiče šum koji dodaje turbolentnost kretanju čestica čime se postiže realističnije kretanje leptira. Model leptira se okreće u pravcu kretanja čestice.

Nakon pripreme potrebnih elemenata, kreiran je algoritam za postavljanje svih elemenata na nivo koristeći Blueprint sistem za vizuelno kodiranje. Unutar ovog sistema korisnik kreira kod povezujući dostupne kartice koje izvršavaju određenu logiku.Pri pokretanju nivoa, koristeći petlje, kreiraju se tačke u prostoru na jednakoj udaljenosti po x i y osi. Ova udaljenost odgovara veličini pripremljenih modela po x i y osi. Veličinu mreže korisnik definiše pre početka generacije okruženja. Vertikalna pozicija prve tačke, odnosno pozicija po z osi, određuje se nasumično u okrviru vrednosti od 0 do maksimalne visine, koju korisnik definiše pre početka generacije terena. Kako bi se dobio teren bez velikih razlika u visini susednih polja sledeća tačka u redu može imati jednaku vrednost kao prethodna tačka, ili vrednost koja je za jedan korak veća ili manja, ukoliko ta vrednost spada u definisani okvir mogućih vrednosti visine. Visina polja u svakom sledećem redu mreže je ograničena visinom prethodnog polja u tom redu i polja u prošlom redu koje dodiruje to polje. Kada se odredi visina polja vrši se selekcija modela koji će se postaviti na to polje. Svakom prethodno pripremljenom modelu je dodeljen redni broj, kao i šansa da taj broj bude izabran. Nakon određivanja pozicije i izbora rednog broja modela na nivo se postavlja instanca odgovarajućeg modela. Korišćenjem instanci modela smanjuje se broj poziva iscrtavanja, tako što se instrukcije za model pamte i koriste za ponovno iscrtavanje tog modela, umesto da se instrukcije prosleđuju ponovo za svaku kopiju modela. Jedina mana korišćenja instanci jeste da se sve kopije modela iscrtavaju zajedno, odnosno nije moguće sakriti modele koje su udaljeni od kamere, niti je moguće koristiti različite nivoe detalja (engl. Level of Detail) za različite kopije modela [3]. U slučaju da je odabran model ravnog tla na poziciju tog polja postavlja se pripremljeni sistem čestica. Visina sistema čestica je za 20cm viša od visine polja tla kako bi se izbeglo preklapanje modela. Nakon postavljanja modela na polje popunjavaju se vertikalni razmaci između polja ukoliko polja nisu iste visine. To je postignuto postavljanjem pripremljenog modela ukoliko je razlika između visina dva susedna polja različita od nule. U zavisnosti od predzaka ove razlike, odnosno da li je naredno polje više ili niže od prethodnog, model se postavlja na adekvatnu visinu.

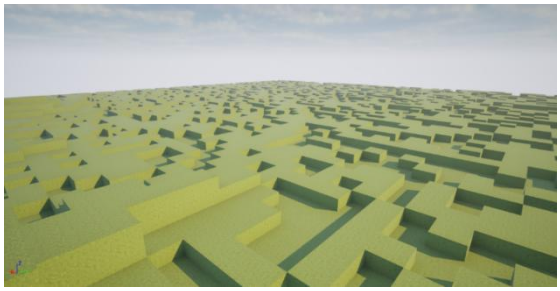
Izgled okruženja dobijenog korišćenjem predstavljenog algoritma prikazano je na slici 2.



Slika 2. *Izgeld okruženja generisanog korišćenjem gotovih modela*

2.2. Generacija okruženja koristeći gotove modele i PCG sistem

Bazirno na prethodno opisanom algoritmu postignuta je generacija terena koristeći samo model ravnog tla (slika 3). Time se izbacuje potreba upotrebe logike izbora modela koji će se postavljati na polje unutar algoritma. Pošto se za generaciju terena koristi isti model za sva polja, sistem čestica se postavlja na svako treće polje u mreži.



Slika 3. *Izgled generisanog okruženja*

Ukrasni modeli okruženja se u ovom slučaju postavljaju na generasani teren korišćenjem PCG sistema. PCG sistem se zasniva na korišćenju kartica kako bi se na jednostavan način kreirali sistemi za postavljanje aseta unutar nivoa prateći definisana pravila. Korišćenjem PCG sistema moguće je koristiti različite nivoe detalja modela u zavisnosti od udaljenosti modela od kamere čime se poboljšavaju performanse [4]. Koristeći detekciju površine, PCG sistem generiše tačke na površini generisanog terena. Veličina i učestalost tačaka je podešena tako da svako polje mreže terena predstavlja jednu tačku. Kako bi se postigla adekvatna učestalost postavljanja modela uklonjene su tačke tako da se nijedna tačka ne dodiruje sa drugom tačkom. Za svaku dobijenu tačku generiše se jedan od definisanih modela. Svaki model ima definisanu vrednost šanse za izbor. PCG sistem se poziva unutar Blueprint-a, nakon završene generacije tla. Izgled okruženja dobijenog korišćenjem predstavljenog algoritma prikazano je na slici 4.

Slika 4. *Izgeld okruženja generisanog korišćenjem gotovih modela i PCG sistema*

2.3. Generacija okruženja koristeći komponentu Procedural Mesh

Komponenta Procedural Mesh omogućava proceduralnu generaciju modela koristeći Blueprint sistem. Generisani model se može upotrebiti kao model tla na koji se mogu dodati ukrasni modeli kako bi se generisalo okruženje video igre. Proceduralnim generisanjem modela, štedi se vreme potrebno za kreiranje modela, generisani model zauzima manje memorije u odnosnu na unapred kreirani model koji je potrebno skladišti, ali se smanjuje kontrola nad konačnim izgledom modela u odnosu na ručno kreirani model [5].

Pre kreiranja sistema za generaciju okruženja pripremljen je materijal koji će se primeniti na model tla. Kako bi se izbeglo definisanje UV koordinata generisanog modela, materijal je kreiran tako da koristi projekciju tekstura na osnovu usmerenja verteksa modela u odnosu na ose sveta. Ovo je postignuto koristeći karticu World Aligned Texture.

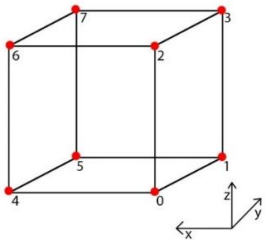
Izgled komponente Procedural Mesh je moguće kontrolisati karticom Create Mesh Section pomoću koje se prosleđuju koordinate verteksa, kao i redosled indeksa verteksa koji obrazuju trouglove modela koji se procedurlano generiše. Kako bi se dobile pozicije i redosled verteksa koristiće se algoritam Marching Cubes. Potrebno je generisati 3D mrežu tačaka na jednakoj međusobnoj udaljenosti. Svaka tačka ima svoje stanje i može biti uključena ili isključena. Algoritam Marching Cubes proverava stanje 8 tačaka mreže koje obrazuju jednu kocku, i razdvaja tačke suprotnog stanja trouglovima. Verteksi trouglova se postavljaju na jednakoj razdaljini između tačaka suprotnog znaka. Trouglovi su okrenuti tako da je prednja strana trougla okrenuta prema uključenoj tački. Ova operacija se ponavlja za sve tačke u 3D mreži.

Kako bi se dobio adekvatan izgled terena, potrebno je generisati mrežu tačaka na jednakom međusobnom rastojanju, gde su donje tačke isključene, a kako se povećava vrednost z koordinate tačke, tačke imaju šansu da postanu uključene. Jednom kada se tačka uključi, sve tačke iznad nje moraju da budu uključene kako ne bi došlo do delova terena koji lebde. Tačke gornjeg reda mreže su uvek uključene kako ne bi došlo do rupa na generisanom terenu.

Algoritam je započet korišćenjem 3 petlje, čiji indeksi iteracije se množe sa vrednošću razmaka tačaka, kako bi se dobila mreža tačaka. Nakon toga se u niz upisuju vrednosti koordinata svake tačke, i određuje se stanje svake tačke na osnovu logike koja je prethodno predstavljena.

Moguće je odrediti veličinu mreže po svakoj osi zasebno, kao i međusobno rastojanje tačaka pre početka generacije terena. Jedini uslov je da mora postojati barem 2 tačke na svakoj osi kako bi se dobila barem jedna kocka.

Nakon određivanja pozicije i stanja tačaka u mreži potrebno je izračunati slučaj svake kocke u mreži kako bi se odredili trouglovi koje je potrebno iscrtati unutar te kocke. Slučaj kocke se računa sabiranjem vrednosti uključenih tačaka koje obrazuju kocku. Vrednost uključene tačke zavisi od pozicije te tačke u kocki, gde je vrednost tačke jednaka 2^n, gde je n jednak numeričkoj vrednosti temena kocke koje odgovara poziciji uključene tačke. Time se dobija 256 različitih kombinacija uključenih i isključenih temena kocke. Vrednosti temena kocke prikazani su na slici 5.

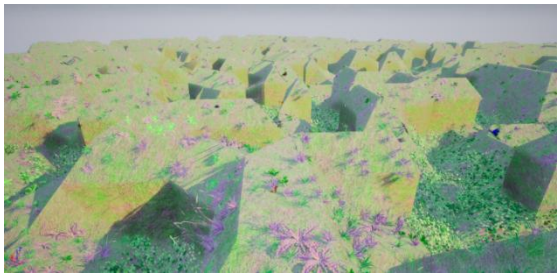


Slika 5. *Numeričke vrednosti temena kocke*

Kako jednu kocku čine 8 tačaka mreže, poslednje tačke iz svakog reda, poslednji redovi, kao i najviše tačke mreže ne mogu da predstavljaju početnu tačku kocke. Ostale tačke mreže se uzimaju kao početna tačka kocke i na osnovu njih se računaju slučajevi za svaku kocku. Nakon proračuna slučaja, u niz se upisuju vrednosti koordinati verteksa, adekvatni indeksi verteksa, i vektori normala verteksa za taj slučaj. Pozicija verteksa se računa na osnovu koordinate početnog temena kocke, dodavanjem polovine ili cele vrednosti rasojanja tačaka na poziciju početne tačke po odgovarajućoj osi, kako bi se dobila odgovarajuća pozicija verteksa. Redosled indeksa verteksa je takav da se dobiju trouglovi okrenuti ka uključenim tačkama. Vektori normala verteksa utiču na izgled osvetljenja, kao i na projekciju tekstura koje će se primeniti na model. Iz tog razoga je potrebno podesiti da usmerenje vektora normala bude u adekvatnom smeru u odnosu na svet.

Nakon izvršavanja proračuna za svaku kocku u mreži, kartici Make Mesh Section se prosleđuju dobijeni nizevi verteksa, indeksa i normala, i generisanom modelu se dodeljuje priprmenljeni materijal. Nakon toga se na poziciji svake treće tačke na najvišoj poziciji postavlja sistem čestica leptira. Sistem se postavlja na poziciji najviših tačaka kako ne bi došlo do preklapanja modela leptira i tla.

Kako bi se postavili ukrasni modeli kreiran je PCG sistem. Sistem detektuje površinu modela i nakon toga generiše različite modele u zavisnosti od visine dobijenih tačaka. Ukoliko je pozicija tačke niža od 1m, na toj poziciji će se generisati modeli deteline, dok na tačkama koje su na visini većoj od 1m generisaće se modeli drugih biljaka. Pripremljeni PCG sistem se poziva unutar Blueprint-a nakon postavljanja sistema čestica. Izgled okruženja dobijenog korišćenjem ovog algoritma prikazan je na slici 6.



Slika 6. *Izgled okruženja generisanog koristeći komponentu Procedural Mesh*

2.4. Performanse algoritama

Prethodno kreirani algoritmi su testirani po sledećim parametrima:

- maksimalna veličina okruženja koju je moguće generisati
- vreme potrebno da se nivo pokrene
- broj frejmova u sekundi pri radu igre
- vreme potrebno za izvršavanje naredbi za procesor
- vreme potrebno za prosleđivanje podataka za renderovanje
- vreme potrebno za izvršavanje naredbi za grafičku karticu

Dobijene vrednosti su predstavljene u tabeli ispod.

Tabela 1. *Izmerene performanse algoritama*

	Generacija koristeći gotove modele	Generacija koristeći gotove modele i PCG sistem	Generacija koristeći komponentu Procedural Mesh
Maksimalna veličina (tačke/kocke)	105x105x90	200x200x99	60x60x3
Vreme pokretanja nivoa (sekund)	0,18	0,13	0,25
Broj frejmova u sekundi	97	90	65
Vreme naredbi procesora (milisekund)	10	10	8,5
Vreme prosleđivanja podataka za renderovanje (milisekund)	7	7	15
Vreme naredbi grafičke karte (mislisekund)	8,4	8,7	14,4

Sve vrednosti, osim vrednosti maksimalne veličine, merene su pri generaciji terena veličine 20 tačaka, odnosno kocki, po x i y osi i 3 tačke, odnosno kocke, po z osi. Razmak između tačaka kocke je bio podešen na 200 kako bi odgovarao veličini modela drugih algoritama.

U slučaju prvog algoritma, maksimalna veličina terena je ograničena brojem grananja pri izboru modela koji će se postaviti na nivo. Iz tog razloga vidimo da se korišćenjem PCG sistema taj problem eliminiše i time postiže mogućnost kreiranja većeg okruženja. Maksimalna veličina okruženja generisanog koristeći komponentu Procedural Mesh je ograničena brojem verteksa i indeksa verteksa u nizu. Pošto veličina nizeva zavisi od oblika terena nije moguće precizno odrediti maksimalnu veličinu okruženja koju je moguće generisati. Utvrđeno je da je pri vrednosti datoj u tabeli generisanje uvek uspešno, iako postoji šansa za generaciju većeg terena.

3. ZAKLJUČAK

Na osnovu rezultata predstavljenih algoritama za proceduralnu generaciju okrženja može se uvideti raznikolikost pristupa proceduralne generacije elemenata okruženja video igre. Promenom parametara algoritama moguće je dobiti potpuno različite rezultate za kratko vreme. Moguće je uticati na performanse igre odabirom algoritma koji u kombinaciji sa ostalim sistemima igre postižu da upotreba komponenata računara bude ujednačena. U zavisnosti od potrebe igre, proceduralna generacija se može prilagoriti radi dobijanja najboljeg rešenja za dati problem.

4. LITERATURA

[1] A. Ikeda, „Unraveling the Mysteries of Procedural Generation in Gaming," tokengamer.io, Avg. 10, 2023. [Online]. Dostupno na: https://tokengamer.io/unraveling-the-mysteries-of-procedural-generation-in-gaming/ (Pristupljeno: Sept. 21, 2024.)

[2] J. DiLaura, „A Look at Unreal Engine Procedural Generation of Content," interactiveimmersive.io, Sept. 22, 2023. [Online]. Dostupno na: https://interactiveimmersive.io/blog/unreal-engine/a-look-at-procedural-content-generation-in-unreal-engine/#:~:text=The%20framework%20allows%20artists%20and,or%20trees%20to%20a%20forest (Pristupljeno: Sept. 21, 2024.)

[3] A. Moskalev, „Wave Function Collapse for procedural generation in Unity," pvs-studio.com, Jan. 24, 2023. [Online]. Dostupno na: https://pvs-studio.com/en/blog/posts/csharp/1027/ (Pristupljeno: Sept. 21, 2024.)

[3] K. North, „Boost Performance with Instanced Static Meshes in Unreal Engine", gamedevexp.com, [Online]. Dostupno na: https://gamedevexp.com/unreal-engine-instanced-static-mesh/ (Pristupljeno Sept. 21, 2024.)

[4]Epic Games, „Procedural Content Generation Overview," dev.epicgames.com, [Online]. Dostupno na: https://dev.epicgames.com/documentation/en-us/unreal-engine/procedural-content-generation-overview (Pristupljeno: Sept. 21, 2024.)

[5] Autodesk, „Procedural generation: Creating infinite algorithmic realities," autodesk.com, [Online]. Dostupno na: https://www.autodesk.com/solutions/procedural-generation (Pristupljeno: Sept. 21, 2024.)

Kratka biografija:

Kristina Tapai rođena je u Zrenjaninu 2000. god. Diplomski rad na Fakultetu tehničkih nauka iz oblasti grafičkog inženjerstva i dizajna odbranila je 2023.god.
kontakt: tapaikristina@gmail.com

Bojan Banjanin rođen je u Novom Sadu 1986. godine. Doktorirao je na Fakultetu tehničkih nauka 2018. godine, a od 2023. godine je u zvanju docenta. Oblasti interesovanja su Multimedije i razvoj video igara.
kontakt: bojanb@uns.ac.rs