ELEKTROTEHNIŠKI VESTNIK 84(1-2): 30–38, 2017
IZVIRNI ZNANSTVENI ČLANEK

# Proceduralno generiranje tropskega otoka in koralnega grebena

**Oskar Korošec, Iztok Lebar Bajec**

*Univerza v Ljubljani, Fakulteta za računalništvo in informatiko, Večna pot 113, 1000 Ljubljana, Slovenija*
*E-pošta: oskar.korosec@gmail.com, ilb@fri.uni-lj.si*

**Povzetek.** V računalniški grafiki pogosto nastane potreba po prikazu večjih območij terena. Ročno oblikovanje le-tega je običajno časovno potratno. S proceduralnim generiranjem pa lahko ob minimalnih ročnih posegih ali celo brez njih ustvarimo večja območja terena z verodostojnim videzom v razmeroma kratkem času. V članku predstavljamo proces preoceduralnega generiranja tropskega otoka. Začne se z generiranjem začetnega terena, sledi mu preoblikovanje s simuliranjem procesov hidravlične in termične erozije katerima sledi še simulacija rasti koralnega grebena, ki se pogosto nahaja v neposredni okolici tropskega otoka. Na teren dinamično nanašamo teksture glede na njegove lokalne značilnosti in med simulacijo omogočamo vizualizacijo sprememb terena v realnem času. Končni rezultat je verodostojen model tropskega otoka z okoliškim koralnim grebenom in ustreznimi teksturami.

**Ključne besede:** proceduralno generiranje, generiranje terena, termalna in hidravlična erozija, koralni greben, simulacija, GPE

**Procedural generation of a tropic island and coral reef**

In computer graphics there is a frequent need for displaying large vistas of a naturally looking terrain. Designing such terrain by hand is typically time consuming. With procedural generation, on the other hand, larger areas of a naturally looking terrain can be generated with or with no minimal intervention in a relatively short time. In this work we present a process of procedural generation of a tropical island with an associated corral reef. We start by generating a heightmap for the base terrain. The heightmap is then transformed by simulating processes of hydraulic and thermal erosion to achieve a more natural look of the terrain. As coral reefs often grow around tropical islands, we also simulate their growth as part of the last step. Real-time visualization is enabled during simulation, so that one can observe evolution of the terrain. Here we dynamically apply textures to the terrain based on its local characteristics. The result is a naturally looking model of a textured tropical island and corral reef.

**Keywords:** procedural generation, terrain generation, thermal and hydraulic erosion, coral reef, simulation, GPU

## 1 UVOD

Članek je povzet po diplomskem delu prvega avtorja [1], ki prikaže metode za proceduralno generiranje tropskega otoka, primernega za uporabo v računalniški igri. Omejimo se na generiranje in teksturiranje terena, brez postavljanja morebitnih drugih objektov, kot so drevesa in skale. Ker je v pri večjih terenih proceduralno generiranje mnogokrat računsko intenzivna operacija, posamezne korake implementiramo na grafični procesni enoti (GPE), ki je zaradi svoje visokoparalelne arhitekture zmožna hitrega procesiranja velike količine podatkov. Najprej generiramo začetni teren, nato pa za bolj verodostojen videz simuliramo hidravlično in termično erozijo. Ker v tropskih morjih v okolici otokov običajno rastejo koralni grebeni, izvedemo tudi simulacijo rasti le-teh. Na teren lepimo teksture, ustrezne njegovim značilnostim.

Enega pomembnejših prispevkov na področju proceduralnega generiranja je dal Ken Perlin. Predstavil je način generiranja šuma, ki ga danes imenujemo Perlinov šum [2]. Perlinov šum se uporablja na veliko področjih, eno izmed njih pa je proceduralna gradnja terena. Perlin je pozneje predstavil še šum *Simplex* [3], ki je za izračun hitrejši in še dodatno izboljša videz. V delu za gradnjo začetnega terena uporabljamo slednjega, ker je ta še danes najboljši kompromis med hitrostjo izračuna, pomnilno potratnostjo in kakovostjo [4], [5].

Musgrave, Kolb in Mace [6] so bili med prvimi, ki so predstavili simuliranje erozije. Opisujejo tako termično kot tudi hidravlično erozijo in način prikaza generiranega terena. Hidravlična erozija simulira pretakanje vode po terenu, ki spodjeda material, ga prenaša v svojem toku in ga nato odlaga. Termična erozija pa zajema dejavnike, ki drobijo površino terena v drobir, ta pa nato drsi po strmih pobočjih. Simuliranje hidravlične erozije sta predstavila tudi Beneš in Forsbach [7]. Mei, Decaudin in Hu [8] pa so prvi predstavili implementacijo hidravlične erozije iz [7] na GPE. Jákó in Tóth [9] sta dodatno opisala še implementacijo termične erozije na GPE. Generiranje terena s pomočjo hidravlične in termične erozije na GPE je v svojem diplomskem delu opisal tudi Maške [10]. V tem delu uporabljamo drugačen pristop k generiranju začetnega terena kot

*Prejet 13. februar, 2016*
*Odobren 15. marec, 2017*

drugi in po simulaciji erozije simuliramo še rast koralnega grebena ter na teren nanesemo teksture.

Virov, ki prikazujejo simuliranje rasti koral z namenom vizualizacije, v računalniški grafiki nismo našli. Po drugi strani pa smo našli vira, ki sta model rasti koral in s tem koralnega grebena, namenjen znanstvenim raziskavam [11], [12]. Z nekaj poenostavitvami smo ta model priredili svojim potrebam.

## 2 POSTOPEK

Za predstavitev terena smo uporabili višinsko karto (angl. *heightmap*). To je rastrska slika, kjer vsak piksel pomeni višino v pripadajoči točki na terenu.

### 2.1 Generiranje začetnega terena

Ker je bil cilj generirati teren v obliki otoka, smo želeli, da bo teren na sredini višinske karte višji kot ob robovih. To smo dosegli z generiranjem višinske karte, ki definira teren v obliki stožca z nelinearno klančino:

$$h_{\mathrm{A}}(x,y) = \begin{cases} \kappa_{\mathrm{H}} \left(1 - \left(\frac{r(x,y)}{\kappa_{\mathrm{R}}}\right)^{\kappa_{\mathrm{P}}}\right); & \text{če } r(x,y) < \kappa_{\mathrm{R}} \\ 0; & \text{sicer,} \end{cases} \tag{1}$$

kjer $\kappa_{\mathrm{H}}$ označuje višino stožca, $\kappa_{\mathrm{R}}$ polmer stožca in $\kappa_{\mathrm{P}}$ potenco, ki nadzoruje obliko klančine,

$$r(x,y) = \sqrt{(x - x_{\mathrm{C}})^2 + (y - y_{\mathrm{C}})^2} \tag{2}$$

pa razdaljo točke $(x, y)$ do središča osnovne ploskve stožca $(x_{\mathrm{C}}, y_{\mathrm{C}})$.

Za izboljšanje verodostojnosti terena smo v naslednjem koraku generirali dodatno višinsko karto, ki smo jo potem združili s prvo. Za generiranje te smo uporabili šum Simplex [3], ki ga lahko implementiramo v poljubnem številu dimenzij in s katerim lahko dosežemo hribovit videz. Pri svoji rešitvi uporabljamo C# implementacijo, ki je del prosto dostopnega paketa MinePackage[1]. Za boljši videz terena smo uporabili več oktav šuma[2]. Izraz oktava v glasbeni teoriji označuje interval med višinama dveh tonov, od katerih ima eden dvojno oziroma polovično frekvenco drugega. V primeru funkcije šuma z višanjem frekvence šuma (skala šuma) zmanjšujemo tudi amplitudo. Na eni višinski karti smo uporabili več oktav šuma tako, da smo vse oktave sešteli:

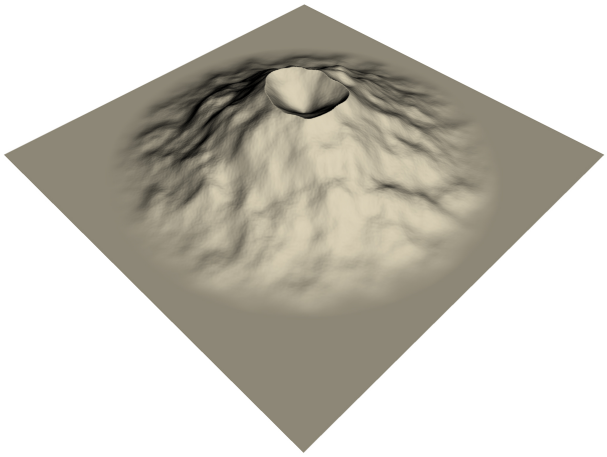
$$h_{\mathrm{B}}(x,y) = \sum_{i=0}^{N} \eta(x2^i, y2^i) \frac{1}{2^i}, \tag{3}$$

kjer je $\eta$ vrednost šuma Simplex v koordinatah $(x, y)$ in $N$ število oktav.

Pri združevanju višinskih kart stožca in šuma smo se zgledovali po orodju Worldmachine[3], kjer je mogoče več višinskih kart združiti v eno na več načinov. Najboljše rezultate smo dobili, ko smo višinski karti združili s potenciranjem:

$$h_{\mathrm{C}}(x,y) = h_{\mathrm{A}}(x,y)^{1+h_{\mathrm{B}}(x,y)\kappa_{\mathrm{N}}}, \tag{4}$$

kjer je $h_{\mathrm{A}}$ višina v višinski karti stožca, $h_{\mathrm{B}}$ pa višina v višinski karti šuma Simplex. Vpliv šuma določa parameter $\kappa_{\mathrm{N}}$.



Slika 1: Začetni relief terena

Ker smo želeli, da otok dobi videz vulkana, smo dodali možnost generiranja vulkanskega kraterja. Generirali smo ga tako, da smo vse vrednosti v združeni višinski karti $h_{\mathrm{C}}$, ki so višje od določenega praga, čez prag preslikali navzdol. Postopek prikazuje enačba (5), kjer je $h_0$ končna višinska karta začetnega terena, $\kappa_{\mathrm{C}}$ pa višina praga in s tem višina, na kateri se ustvari vulkanski krater. Slika 1 pa prikazuje relief, generiran z uporabo končne višinske karte začetnega terena.

$$h_0(x,y) = \begin{cases} 2\kappa_{\mathrm{C}} - h_{\mathrm{C}}(x,y); & \text{če } h_{\mathrm{C}}(x,y) > \kappa_{\mathrm{C}} \\ h_{\mathrm{C}}(x,y); & \text{sicer} \end{cases} \tag{5}$$

### 2.2 Erozija

Kljub temu, da že začetni teren spominja na otok, je njegov videz še nepreprič ljiv. Simuliranje naravnih procesov, kot sta hidravlična in termična erozija, nam omogoča videz izboljšati. Hidravlično erozijo smo implementirali po viru [8], termično pa po [9]. Vira opisujeta metodi simuliranja teh procesov na GPE.

*2.2.1 Hidravlična erozija:* Simuliranje hidravlične erozije se začne s simuliranjem pretakanja vode po površini terena. Na podlagi hitrosti vode in naklona terena določimo količino raztopljenega sedimenta in njegov prenos. V predelih hitrejšega vodnega toka voda teren odnaša, v počasnejših pa ga odlaga. S tem se teren preoblikuje in dobi želen erodiran videz. Voda med simulacijo ves čas izhlapeva.

Za opis simulacije bomo uporabljali naslednje pomembne količine: višina terena, količina tekoče vode,

[1] sourceforge.net/projects/minepackage
[2] www.redblobgames.com/maps/terrain-from-noise
[3] www.world-machine.com

količina razstopljenega sedimenta, odtočni tokovi vode in hitrost vode. Pri tem se vsaka od količin hrani za vsako točko (celico) terena. S $t$ dodatno označujemo trenutni čas in s $\tau$ časovni korak ene iteracije simulacije. Vsaka iteracija je razbita na več zaporednih korakov:

1) dodajanje vode,
2) pretakanje vode po terenu in izračun vektorjev hitrosti ter posodobitev količine tekoče vode,
3) izračun stopljenega in odloženega sedimenta,
4) prenos sedimenta,
5) difuzija sedimenta,
6) izhlapevanje vode.

Vsak korak uporabi količine iz prejšnjega koraka in jih posodobi. Nekatere količine se izračunajo v več podkorakih, zato te indeksiramo še z $'$, $''$ itd., da med seboj ločimo vrednosti v različnih podkorakih. Sosednje celice trenutne celice ozačujemo z L (leva), R (desna), T (gornja), B (spodnja).

Za simuliranje pretakanja vode uporabljamo prirejen model, ki povezuje sosednje celice z navideznimi cevmi, po katerih se med njimi izmenjuje voda [13]. Ob začetku simulacije privzemamo, da je višina vode po celotnem terenu ničelna; $d_0 = 0$. Dež pomeni nastanek nove vode, kar za obravnavano celico $(x, y)$ izračunamo kot

$$d_t{}' = d_t + r\tau, \tag{6}$$

kjer $r$ označuje količino dežja, ki prispe v celico na enoto časa. Zaradi velikosti terena namreč ni smiselno simulirati dežja kot posamezne kaplje, temveč kot uniformno rast količine tekoče vode po posamezni celici terena.

Za vsako celico hranimo njene odtočne tokove $L, R, T, B$ do štirih sosednjih celic, njeni pritočni tokovi pa so ustrezni odtočni tokovi obravnavani celici sosednjih celic. V vsaki iteraciji simulacije se tok vode v odtočnih tokovih pospeši zaradi razlike v hidrostatičnem tlaku med celicami, količina tekoče vode pa se nato izračuna s kopičenjem vseh tokov v virtualnih ceveh.

Odtočni tok v smeri proti levi celici za obravnavano celico izračunamo po enačbi:

$$L_t{}' = \max\left\{0, L_t + \tau\kappa_\mathrm{A}\frac{\kappa_\mathrm{g}\Delta h_t^\mathrm{L}}{\ell}\right\}, \tag{7}$$

kjer $\kappa_\mathrm{A}$ pomeni površino prereza cevi, $\ell$ dolžino cevi med dvema sosednjima celicama, $\kappa_\mathrm{g}$ gravitacijski pospešek in $\Delta h_t^\mathrm{L}$ razliko višin med obravnavano in levo sosednjo celico $(x-1, y)$, ki se izračuna po enačbi:

$$\Delta h_t^\mathrm{L} = h_t + d_t{}' - h_t^\mathrm{L} - d_t^\mathrm{L}{}'. \tag{8}$$

Odtočni tokovi v smeri sosed R, T, B se izračunajo na enak način. Ker izračun odtočnih tokov upošteva višinske razlike do vsake sosede individualno, obstaja možnost, da bi iz obravnavane celice odteklo več vode, kot je ta vsebuje. S tem bi količina tekoče vode v celici po izračunu postala negativna. Težavo se rešuje z omejitvijo moči vsakega izmed odtočnih tokov v odvisnosti od količine tekoče vode v celici in predvidene moči vseh odtočnih tokov [8], [10]. Pri odtočnem toku v smeri proti levi celici se tako odtočni tok v naslednjem koraku simulacije izračuna kot:

$$L_{t+\tau} = L_t{}' \min\left\{1, \frac{d_t{}'\ell^2}{\tau\sum \mathbf{O}_t{}'}\right\}, \tag{9}$$

kjer $\mathbf{O}_t{}'$ pomeni množico vseh odtočnih tokov obravnavane celice $\left\{L_t{}', R_t{}', T_t{}', B_t{}'\right\}$, izračunanih z uporabo enačbe (7), $\sum \mathbf{O}_t{}'$ pa njihovo skupno vsoto.

Pretakanje vode po navideznih ceveh se odraža v spremembi količine tekoče vode

$$d_t{}'' = d_t{}' + \frac{\sum \mathbf{I}_{t+\tau} - \sum \mathbf{O}_{t+\tau}}{\ell^2}\tau, \tag{10}$$

kjer sta

$$\mathbf{I}_{t+\tau} = \left\{R_{t+\tau}^\mathrm{L}, L_{t+\tau}^\mathrm{R}, B_{t+\tau}^\mathrm{T}, T_{t+\tau}^\mathrm{B}\right\}$$

in

$$\mathbf{O}_{t+\tau} = \left\{L_{t+\tau}, R_{t+\tau}, T_{t+\tau}, B_{t+\tau}\right\}$$

po vrsti množici pritočnih in odtočnih tokov obravnavane celice v naslednjem časovnem koraku. Pri simuliranju pretakanja vode moramo upoštevati robne pogoje, ki izhajajo iz dejstva, da je teren končen. Iztekanje vode iz terena omejimo tako, da vsaki cevi, ki nima sosednje celice, postavimo vrednosti ustreznih odtočnih in pritočnih tokov na 0. Na primer: za celice brez levega soseda $(0, y)$ postavimo tako levi odtočni tok $L$, kot tudi levi pritočni tok $R^\mathrm{L}$ na 0.

Voda med pretakanjem zaradi erozije pretvori del terena v sediment

$$S = \kappa_\mathrm{S}\left\|\mathbf{v}_t\right\|\sin\alpha, \tag{11}$$

ki ga nato prenaša in odlaga po terenu. V enačbi (11) $\kappa_\mathrm{S}$ pomeni konstanto kapacitete sedimenta, $\alpha$ pa lokalni naklon terena v celici. Tega za obravnavano celico izračunamo kot $\alpha = \arccos(\hat{\mathbf{n}}_t \cdot \mathbf{k})$. Pri tem $\hat{\mathbf{n}}_t$ pomeni normalo na teren, ki jo pridobimo kot

$$\mathbf{m} = \mathbf{i}\frac{h_t^\mathrm{R} - h_t^\mathrm{L}}{\kappa_\mathrm{M}} + \mathbf{j}\frac{h_t^\mathrm{T} - h_t^\mathrm{B}}{\kappa_\mathrm{M}} + \mathbf{k}, \quad \hat{\mathbf{n}}_t = \frac{\mathbf{m}}{\|\mathbf{m}\|}, \tag{12}$$

kjer je $\kappa_\mathrm{M}$ razdalja med sosednjimi celicami terena, **i**, **j**, **k** pa po vrsti enotski vektorji, ki kažejo v smereh osi x, y in z. Hitrost vode v obravnavani celici $\mathbf{v}_t = \langle v_\mathrm{x}, v_\mathrm{y}\rangle$, ki jo potrebujemo v enačbi (11), pa izračunamo s pomočjo odtočnih in pritočnih tokov celice:

$$v_\mathrm{x} = \frac{R_{t+\tau}^\mathrm{L} - L_{t+\tau} + R_{t+\tau} - L_{t+\tau}^\mathrm{R}}{\ell(d_t{}' + d_t{}'')}, \tag{13}$$

$$v_\mathrm{y} = \frac{T_{t+\tau}^\mathrm{B} - B_{t+\tau} + T_{t+\tau} - B_{t+\tau}^\mathrm{T}}{\ell(d_t{}' + d_t{}'')}. \tag{14}$$

Na zelo položnih delih, kjer se $\alpha$ približuje 0, je vrednost $S$ zelo nizka. Na takšnih predelih proces erozije ne bo imel občutnega učinka. To lahko omilimo tako, da $\alpha$ navzdol omejimo s poljubno minimalno vrednostjo. Na podlagi vrednosti $S$ in raztopljenega sedimenta iz

predhodne iteracije simulacije, $s_t$, se odločimo, ali se bo del sedimenta odložil ter pretvoril nazaj v teren ali ne. Ko velja $S > s_t$, se zgodi proces erozije in del terena se raztopi, v nasprotnem primeru se del raztopljenega sedimenta odloži in pretvori nazaj v teren. Novo višino terena in količino raztopljenega sedimenta izračunamo po enačbah:

$$h_{t+\tau} = \begin{cases} h_t - \kappa_{\mathrm{E}}(S - s_t); & \text{če } S > s_t \\ h_t + \kappa_{\mathrm{D}}(S - s_t); & \text{sicer} \end{cases} \tag{15}$$

$$s_t{}' = \begin{cases} s_t + \kappa_{\mathrm{E}}(S - s_t); & \text{če } S > s_t \\ s_t - \kappa_{\mathrm{D}}(S - s_t); & \text{sicer,} \end{cases} \tag{16}$$

kjer je $\kappa_{\mathrm{E}}$ konstanta topljenja materiala, $\kappa_{\mathrm{D}}$ pa konstanta odlaganja materiala.

Raztopljeni sediment prenaša vodni tok, kar opisuje advekcijska enačba $\frac{\delta s}{\delta t} + (\mathbf{v}_t \nabla s) = 0$. Enačbo rešujemo z Eulerjevim korakom nazaj v času, kar pomeni, da količini raztopljenega sedimenta v obravnavani celici $(x, y)$ priredimo vrednost, ki je odvisna od vodnega toka v njej

$$s_t{}''(x, y) = s_t{}'(x - v_{\mathrm{x}}\tau, y - v_{\mathrm{y}}\tau), \tag{17}$$

kjer koordinate $(x - v_{\mathrm{x}}\tau, y - v_{\mathrm{y}}\tau)$ omejimo na območje terena. Kot je omenil že Maške [10], postopek ne deluje zadovoljivo, saj je mogoče, da se v enem koraku količina sedimenta iz ene celice preslika v več celic. Opazili smo tudi, da se sediment pri nižjih hitrostih vode večkrat prenaša v ravnih črtah. To je zato, ker se sediment v območjih nizke hitrosti prenaša med celicami, ki so neposredne sosede. To daje polju sedimenta grob in nenaraven videz, zato ga mi dodatno gladimo s simuliranjem difuzije sedimenta.

Difuzijo simuliramo z dvodimenzionalno Laplacevo enačbo, ki jo numerično rešujemo z Jakobi iteracijami [14]. Izračun iteracije prikazuje enačba

$$s_{t+\tau} = \frac{s_t^{\mathrm{L}}{}'' + s_t^{\mathrm{R}}{}'' + s_t^{\mathrm{T}}{}'' + s_t^{\mathrm{B}}{}''}{n_{\mathrm{W}}}, \tag{18}$$

kjer $n_{\mathrm{W}}$ pomeni število celic izmed L, R, T in B, ki držijo vodo ($d_t{}'' > 0$). Robne pogoje smo poenostavili tako, da se koordinate celic omejijo na območje terena. Na primer: za vse celice, ki ležijo na levem robu terena $(0, y)$, se koordinate leve sosednje celice $(x-1, y)$ omejijo na $(0, y)$. Celicam, ki ne vsebujejo vode, nastavimo vrednost $s_t{}''$ na 0.

V zadnjem koraku vsake iteracije simulacije hidravlične erozije upoštevamo, da se količina tekoče vode spreminja tudi zaradi izhlapevanja. V našem delu predpostavljamo, da je temperatura ozračja med simulacijo konstantna, zato hitrost izhlapevanja določamo s konstanto $\kappa_{\mathrm{T}}$. Proces izhlapevanja opisuje enačba

$$d_{t+\tau} = d_t{}''(1 - \kappa_{\mathrm{T}}\tau). \tag{19}$$

*2.2.2 Termična erozija:* Hidravlična erozija oblikuje grob teren z veliko ostrimi robovi, kar je opazno predvsem na pobočjih ob vodnih strugah, ki se nenehno poglabljajo. S simuliranjem termične erozije, ki modelira proces razgradnje kamnin v drobir zaradi temperaturnih sprememb in njegovo nadaljne razsipanje, teren zgladimo in izboljšamo njegov videz. Podobno kot za hidravlično lahko tudi za termično erozijo uporabljamo model virtualnih cevi, le da v tem primeru cevi prenašajo drobir (terena) namesto vode. Vendar se običajno [9][10] zaradi boljšega rezultata pri termični eroziji namesto von Neumannove okolice, ki zaobjema zgolj najbližje štiri sosede obravnane celice, uporablja Moorova okolica, ki zajema vseh osem najbližjih sosednjih celic $\mathbf{N} \in \{\mathrm{L}, \mathrm{R}, \mathrm{T}, \mathrm{B}, \mathrm{TL}, \mathrm{TR}, \mathrm{BL}, \mathrm{BR}\}$. To bi lahko sicer storili že pri hidravlični eroziji, vendar menimo, da bi to povzdignilo kompleksnost implementacije brez večjih izboljšav pri rezultatu.

Količina drobirja, ki jo lahko v eni iteraciji prenesemo v vsako od sosednjih celic, je omejena z $M_t = \frac{H_t}{2}\tau{\kappa_{\mathrm{M}}}^2$, kjer je $H_t = h_t - \min\left\{h_t^i |\ i \in \mathbf{N}\right\}$ razlika v višini terena v obravnavani celici in najnižjo višino terena v njej sosednjih celicah. Če to vrednost presežemo, začne algoritem oscilirati [9]. Nadaljnji proces smo glede na vir [9] nekoliko poenostavili; vir upošteva tudi lokalno trdnost terena, ki je v našem delu ne.

Drobir se pod vplivom gravitacije razsipa med niže ležeče sosednje celice, ki z obravnavano celico tvorijo kot naklona, večji od kota mirovanja $\kappa_\alpha$. To je kritičen kot pobočja, ki ga lahko neki zrnati material še tvori. Če klančina preseže kot mirovanja, material zdrsne.

Kote naklona med obravnavano in posamezno sosednjo celico $i \in \mathbf{N}$ izračunamo kot $\alpha_t^i = \arctan\frac{h_t - h_t^i}{\kappa_{\mathrm{M}}}$. Na podlagi teh sestavimo množico, ki vsebuje vse sosednje celice z naklonom, večjim od kota mirovanja $\kappa_\alpha$:

$$\mathbf{R}_t = \left\{i \in \mathbf{N} |\ \alpha_t^i > \kappa_\alpha\right\}. \tag{20}$$

V vsako celico iz množice $\mathbf{R}_t$ na podlagi višinske razlike z obravnavano celico prenesemo temu proporcionalno količino drobirja, v preostale celice ga ne prenašamo. Na primer: količino drobirja, ki se prenese v levo sosednjo celico, če ta spada v množico $\mathbf{R}_t$, izračunamo z enačbo:

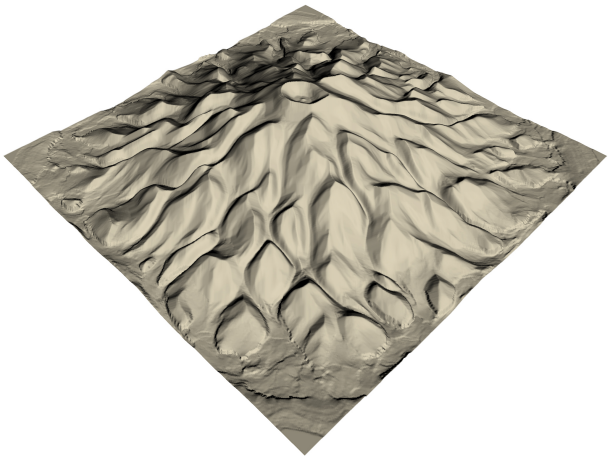
$$L_t = M_t \frac{h_t^{\mathrm{L}}}{\sum_{i \in \mathbf{R}_t} h_t^i}. \tag{21}$$

Zaradi vzporednega izvajanja simulacije drobirja ne prestavimo v celice takoj, temveč za ta proces uporabimo navidezne cevi. Končno višino vsake celice izračunamo podobno kot pri hidravlični eroziji, in sicer tako, da k trenutni višini prištejemo vse pritočne tokove in odštejemo vse odtočne:

$$h_{t+\tau} = h_t + \sum \mathbf{M}_t^{\mathrm{in}} - \sum \mathbf{M}_t^{\mathrm{out}}, \tag{22}$$

le da tokrat z

$$\mathbf{M}_t^{\mathrm{out}} = \{L_t, R_t, T_t, B_t, TL_t, TR_t, BL_t, BR_t\}$$



Slika 2: Relief terena po hidravlični in termični eroziji

označimo drobir, ki se iz obravnavane celice razsipa v sosednje celice po ustreznih virtualnih ceveh (odtočni tokovi drobirja), z

$$\mathbf{M}_t^{\text{in}} = \langle R_t^{\text{L}}, L_t^{\text{R}}, \ldots, TL_t^{\text{BR}} \rangle$$

pa pritočne tokove drobirja (odtočne tokove drobirja usreznih sosed). Primer terena po končani simulaciji prikazuje slika 2.

### *2.3 Glajenje morskega dna*

Teren pod morsko gladino ima po simulaciji erozije enak videz kot teren nad morsko gladino, saj simulaciji erozije ne upoštevata višine morske gladine. Med izvajanjem simulacij smo ugotovili, da ima rezultat rasti koralnega grebena bolj naraven videz, če pred njim teren pod morsko gladino še dodatno zgladimo. To ponovno počnemo z Jakobijevo iteracijo, podobno kot smo to storili s sedimentom pri hidravlični eroziji:

$$h_{t+\tau} = \frac{h_t^{\text{L}} + h_t^{\text{R}} + h_t^{\text{T}} + h_t^{\text{B}}}{4}. \tag{23}$$

Jakobijevo iteracijo računamo samo za celice, ki ležijo pod vodno gladino in ne ležijo tik ob robu terena. Drugim celicam vrednosti ne spreminjamo.

### *2.4 Rast koralnega grebena*

V tropskih morjih okoli otokov, predvsem takšnih vulkanskega nastanka, velikokrat rastejo korale, ki skozi čas tvorijo koralne grebene [15]. Korale rastejo v plitvinah toplejših morij. Za rast potrebujejo svetlobo, saj ožigalkarji živijo v sožitju s fotosinteznimi algami zooksantele (angl. *zooxanthellae*) [15]. Sestavljene so iz kolonij majhnih polipov, katerih apnenčasti zunanji skeleti se nalagajo in tvorijo koralne grebene. K tvorbi grebena pripomorejo tudi drugi organizmi, ki živijo v koralnem ekosistemu. S simuliranjem rasti koralnih grebenov želimo videz otoka še bolj približati resničnosti in ga narediti zanimivejšega.

Model rasti koralnega grebena, predstavljen v nadaljevanju, temelji na modelu Nakamure in Nakamorija [12]. Avtorja sta predstavila model rasti koralnega grebena na podlagi intenzitete svetlobe in koncentracije anorganskega ogljika v vodi, za dvodimenzionalne simulacije rasti pa v pogledu s strani. Ker za naše potrebe potrebujemo pristop, s katerim je mogoče rast simulirati v pogledu iz zraka, smo njun model priredili in poenostavili.

V opisu bomo uporabljali dve pomembni količini: višina terena (morskega dna) in koncentracija anorganskega ogljika v vodi. Obe količini se hranita za vsako točko oziroma celico terena. Vsaka iteracija simulacije pa je razbita na dva zaporedna koraka; rast koral in difuzijo koncentracije anorganskega ogljika.

Kalcifikacija koral je v veliki meri odvisna od stopnje fotosinteze, ki jo lahko izračunamo na podlagi globine vode

$$g_t = w_t - h_t, \tag{24}$$

kjer $w_t$ označuje globalno višino morske gladine v trenutni iteraciji simulacije. Korale zaradi plimovanja in drugih dejavnikov namreč običajno rastejo le do določene globine vode $\kappa_{\text{G}}$. Stopnjo kalcifikacije koral izračunamo z uporabo enačbe

$$c_t = \begin{cases} O_t^2 \kappa_{\text{B}} e^{-g_t \kappa_{\text{A}}}; & \text{če } g_t < \kappa_{\text{G}} \\ 0; & \text{sicer,} \end{cases} \tag{25}$$

kjer je $O_t$ vrednost koncentracije anorganskega ogljika v vodi v trenutni iteraciji simulacije, $\kappa_{\text{B}}$ konstanta hitrosti kalcifikacije in $\kappa_{\text{A}}$ konstanta absorpcije vode. V prvi iteraciji simulacije je vrednost $O_t$ vsake celice postavljena na $O_0$.

Korale za rast porabijo določeno količino anorganskega ogljika, zato moramo vrednost njegove koncentracije v vodi zmanjšati v sorazmerju s stopnjo kalcifikacije. Spremembo izračunamo po enačbi

$$O_t{}' = \max\left\{0, O_t - \frac{c_t \tau}{g_t} \kappa_{\text{U}}\right\}, \tag{26}$$

kjer $\kappa_{\text{U}}$ nadzoruje stopnjo porabe anorganskega ogljika. S tem, da v enačbi (26) stopnjo kalcifikacije delimo z višino vode, dosežemo, da se pri enaki stopnji rasti koral koncentracija anorganskega ogljika v vodi zmanjša primerno količini vode nad to točko na terenu.

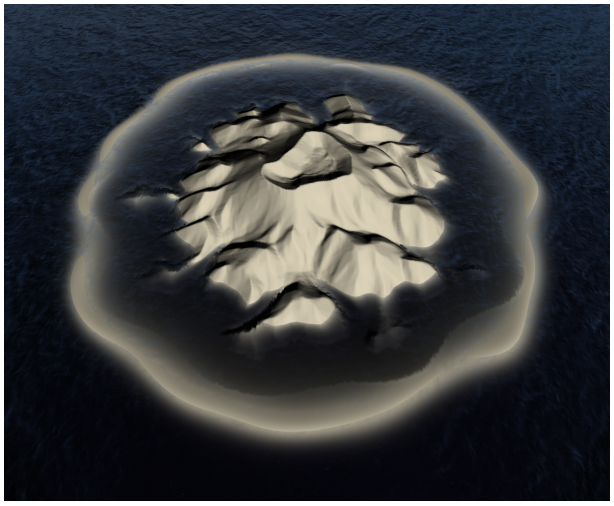
Na koncentracijo anorganskega ogljika v vodi vplivata tudi plimovanje in mešanje vode. Ker v našem primeru simulacija teče v časovnem razponu več tisoč let, in to na velikem terenu, lahko ta vpliv prikažemo s preprosto difuzijo, podobno kot smo to storili s sedimentom pri hidravlični eroziji. Difuzijo anorganskega ogljika modelira enačba

$$\nabla^2 O = \frac{\delta^2 O}{\delta x^2} + \frac{\delta^2 O}{\delta y^2} = 0, \tag{27}$$

ki jo numerično rešujemo z Jakobijevo iteracijo

$$O_{t+\tau} = \frac{O_t^{\text{L}\prime} + O_t^{\text{R}\prime} + O_t^{\text{T}\prime} + O_t^{\text{B}\prime}}{n_{\text{U}}} \tag{28}$$

kjer $n_{\text{U}}$ pomeni število najbližjih sosednjih celic v von Neumannovi okolici, ki ležijo pod morsko gladino



Slika 3: Koralni greben okoli otoka. Morska gladina je prikazana za boljšo predstavo.

($g_t > 0$). Pri robnih celicah, celicah, ki ležijo tik ob robu terena in zato nimajo levega ali desnega, ali zgornjega, ali spodnjega soseda, Jakobijeve iteracije ne računamo, temveč $O_{t+\tau}$ priredimo vrednost $O_0$. Jakobijeve iteracije prav tako ne računamo pri celicah, ki so nad gladino morja, a v nasprotju s prej v teh primerih $O_{t+\tau}$ priredimo vrednost 0, ker celice ne izpolnjujejo pogojev za nastanek koral.

V našem delu podatkov o višini koralnega grebena ne hranimo ločeno, temveč ga štejemo kot teren. Rast grebena (in s tem dvig višine terena) je neposredno odvisna od stopnje kalcifikacije. Višino terena po iteraciji simulacije izračunamo z enačbo

$$h_{t+\tau} = h_t + c_t\tau. \tag{29}$$

Rast koralnega grebena in posledično sprememba višine terena vpliva tudi na globalno višino morske gladine. Na spremembe v višini morske gladine seveda vplivajo tudi drugi dejavniki. V našem delu po vzoru Nakamure in Nakamora [12] privzemamo konstantno rast višine morske gladine. Globalno višino morske gladine ob koncu iteracije rasti koralnega grebena tako izračunamo po enačbi

$$w_{t+\tau} = w_t + \kappa_{\mathrm{W}}\tau, \tag{30}$$

kjer je $\kappa_{\mathrm{W}}$ konstanta rasti višine morske gladine. Vpliv simulacije rasti koralnega grebena na končni videz proceduralno generiranega otoka prikazuje slika 3.

## 2.5 Implementacija

Simuliranje naravnih procesov, kot sta hidravlična in termična erozija, je običajno računsko zahtevna operacija. Čas izvajanja simulacije je pomemben še posebno takrat, ko želimo rezultate opazovati v realnem času, zato v teh primerih posegamo po GPE. V naši implementaciji uporabljamo okolje Unity in senčilnike *ComputeShader*[4]. Simulacijo izvajamo nad višinsko karto, zato podatke o njej shranjujemo v teksturi. Za hranjenje višinske karte v osnovi potrebujemo zgolj en barvni kanal, zato je teksura višinske karte običajno sivinska. V naši implementaciji za posamezen barvni kanal uporabljamo 32-bitno predstavitev s plavajočo vejico (*float*).

Vsak od predstavljenih algoritmov hrani podatke o nekaterih količinah (npr. višina vode), ki se uporabljajo med posameznimi koraki in iteracijami simulacije. Ti podatki so vezani na posamezne točke v višinski karti, zato jih hranimo na enak način kot višinsko karto, tj. v teksturi. Za shranjevanje vsake količine tako kot za višinsko karto potrebujemo en barvni kanal. Ker v naši implementaciji uporabljamo teksture s štirimi kanali, ki se v senčilniku preslika v *float4*, nam to omogoča v eni teksuri shranjevati podatke o več različnih količinah.

Razvili smo tri senčilnike, pri čemer prvi izvaja hidravlično erozijo, drugi termično erozijo, tretji pa skrbi za rast koral. Vsak senčilnik simulacijo pripadajočega procesa izvaja v več iteracijah, v posamezni iteraciji pa določene teksture obdela večkrat. Senčilnik ComputeShader je sestavljen iz več funkcij, med katerimi lahko poljubne označimo kot jedrne. Poženemo ga tako, da poženemo eno od jedrnih funkcij. Z uporabo več jedrnih funkcij lahko teksturo z enim senčilnikom obdelamo v več različnih korakih. Naša koda je organizirana temu primerno, napisana pa je v jeziku HLSL različice DirectX 11.

V prvem koraku naše implementacije generiramo višinsko karto začetnega terena in jo shranimo v obliki teksture. Na tej točki je pripadajoča tekstura shranjena v glavnem pomnilniku računalnika, zato jo pred začetkom izvajanja simulacije naravnih procesov prenesemo v pomnilnik GPE. V pogonu Unity so teksture, ki so shranjene izključno v pomnilniku GPE, objekti tipa RENDERTEXTURE. V senčilnikih uporabljamo le takšen tip tekstur; tekstur ne prenašamo med glavnim pomnilnikom računalnika in pomnilnikom GPE, saj je to časovno potratna operacija. Ker se vizualizacija prav tako izvaja v senčilnikih, je prenašanje nepotrebno. Ker pa rezulatatov posameznega koraka iteracije ne smemo pisati v isto teksturo, ker bi zaradi vzporednega izvajanja s tem lahko uničili rezultate preostalih niti, moramo poleg vsake teksture hraniti še njeno kopijo. Vsaki jedrni funkciji moramo tako pred izvajanjem nastaviti potrebne vhodne in izhodne teksture. Jedrne funkcije berejo trenutno stanje simulacije iz vhodnih tekstur in rezultate obdelave zapišejo v izhodne teksture. Posledično bi morali po vsaki končani jedrni funkciji rezultate iz izhodnih prekopirati nazaj v vhodne teksture. Temu kopiranju smo se izognili z izmeničnim menjavanjem vhodnih in izhodnih tekstur. Izhodne teksture prejšnje jedrne funkcije nastavimo kot vhodne teksture naslednje. Menjavanje izvajamo samo v sklopu posamične iteracije. Kadar se

[4] docs.unity3d.com/Manual/ComputeShaders

menjavanje ne izvede sodo-krat, teksture vseeno kopiramo.

Iteracije simulacije izvajamo v zanki, v kateri v določenem vrstnem redu poženemo jedrne funkcije ustreznega senčilnika. V namen nadzora simulacije smo implementirali preprost grafični uporabniški vmesnik. Z njim lahko spreminjamo določene parametre in vklapljamo ter izklapljamo posamezne korake simulacije. Vrstni red zagona senčilnikov načeloma ni pomemben, a moramo paziti, da tistega, ki je namenjen rasti koral, ne uporabljamo hkrati s hidravlično oz. termično erozijo. Koralnega grebena namreč ne hranimo ločeno, temveč kot teren, erozija (tako hidravlična kot termična) pa teren preoblikuje in posledično spremeni videz koralnega grebena, kar ni zaželeno.

### *2.6 Vizualizacija*

Višinska karta v vsakem pikslu hrani višino pripadajoče točke na terenu. Za prikaz terena uporabljamo metodo imenovano odmiki (angl. *displacement mapping*) [16]. Njena težava je, da za prenos podrobnosti višinske karte na model potrebuje veliko oglišč. V optimalnem primeru bi bil model predstavljen z enakim številom oglišč, kot ima višinska karta pikslov. V praksi to pomeni modele z izjemno visokim številom oglišč in posledično velikim številom trikotnikov. Problem lahko rešimo z uporabo strojne teselacije (angl. *hardware tessellation*) [17], ki model v osnovi predstavi z majhnim številom oglišč in trikotnikov, te pa nato z uporabo senčilnikov dinamično deli na več manjših, pri čemer ustvari več oglišč in več podrobnosti. V naši implementaciji uporabljamo strojno teselacijo fiksne stopnje.

Osvetlitev izračunavamo z Lambertovim modelom [18, str. 267–268]

$$c = (\hat{\mathbf{l}} \cdot \hat{\mathbf{n}}) c_{\mathrm{m}} c_{\mathrm{l}}, \tag{31}$$

kjer osvetljenost v neki točki izračunamo na podlagi vpadnega kota svetlobe $\hat{\mathbf{l}}$ in normale na teren v tej točki $\hat{\mathbf{n}}$. Parametra $c_{\mathrm{m}}$ in $c_{\mathrm{l}}$ po vrsti pomenita barvo materiala v točki in barvo vira svetlobe. Za osvetljevanje uporabljamo smerno oziroma oddaljeno luč [18, str. 264], ki nima pozicije. Luč definira samo vektor smeri, barvo in intenziteto. V enačbi (31) je intenziteta že všteta v barvo luči.

Naš cilj je bilo generiranje tropskega otoka, k čemur veliko pripomorejo teksture. Osnovne štiri teksture, ki jih uporabljamo, so: trava (zelene površine), skalovje, pesek in morsko dno. Teksture smo dobili na spletni strani Textures[5]. V senčilniku na podlagi višine in naklona terena ter višine vode v vsaki točki terena vzorčne teksture združujemo in ob upoštevanju osvetlitve izračunamo barvo v tej točki. Logika združevanja tekstur temelji na opažanjih iz narave in osebne predstave o videzu tropskega otoka. Strma pobočja so redko poraščena, medtem ko na ravninah prevladujejo zelene površine. Za boljši videz smo v višjih in bolj strmih predelih teksturo trave rahlo posvetlili ter dodali rumen odtenek. Na obalah je pogosta peščena plaža, zato smo pod gladino vode in na ožjem pasu obale nad njo nanesli teksturo peska, vendar le tam, kjer je obala dovolj položna. Na koncu smo pod gladino vode dodali še teksturo morskega dna. Za doseganje mehkih prehodov med teksturami smo si pomagali s funkcijo SMOOTHSTEP, ki je del jezika HLSL. Funkcija nam omogoča definicijo poljubno širokega pasu, v katerem se dve teksturi mešata, zunaj pasu pa je izbrana ena od njiju. Pri nanašanju teksture skalovja smo upoštevali tudi, kje je med hidravlično simulacijo tekla voda. V dodatni teksturi hranimo števec prisotnosti vode v določeni točki na terenu v zadnjem krajšem časovnem obdobju. Števec med simulacijo hidravlične erozije povečamo, če je v pripadajoči točki prisotna voda, in zmanjšamo, če ni. Števec je navzdol omejen z 0 in navzgor s poljubno vrednostjo. Prisotnost teksture skalovja je višja tam, kjer ima števec višjo vrednost. S tem vizualno dosežemo, da na območjih, kjer so bile vodne struge, nekaj časa ne raste trava. Podobno v isti dodatni teksturi hranimo tudi števec količine rasti koralnega grebena, kjer tega povečujemo premo sorazmerno z rastjo koralnega grebena. Kjer ima ta števec višjo vrednost, je večja tudi prisotnost teksture morskega dna.

Ker je otok v naravnem okolju obkrožen z morjem, smo dodali prikaz tudi slednjega. Morja v okolici tropskih otokov so običajno zelo čista, zato v plitvih predelih pogosto vidimo dno. Intenziteta svetlobe v neki točki v globini morja je funkcija dolžine poti svetlobnih žarkov skozi vodno maso [11]. V naši aplikaciji na teren in morje v pretežni meri gledamo od zelo daleč, zato pri izračunu barve morske gladine zanemarjamo dejstvo, da se svetlobni žarki, ko iz zraka prehajajo v vodo, lomijo, kar vpliva na dolžino njihove poti skozi vodno maso. Barvo morske gladine tako določamo zgolj s spreminjanjem prosojnosti osnovne, temno modre barve v odvisnosti globine vode v obravnavani točki

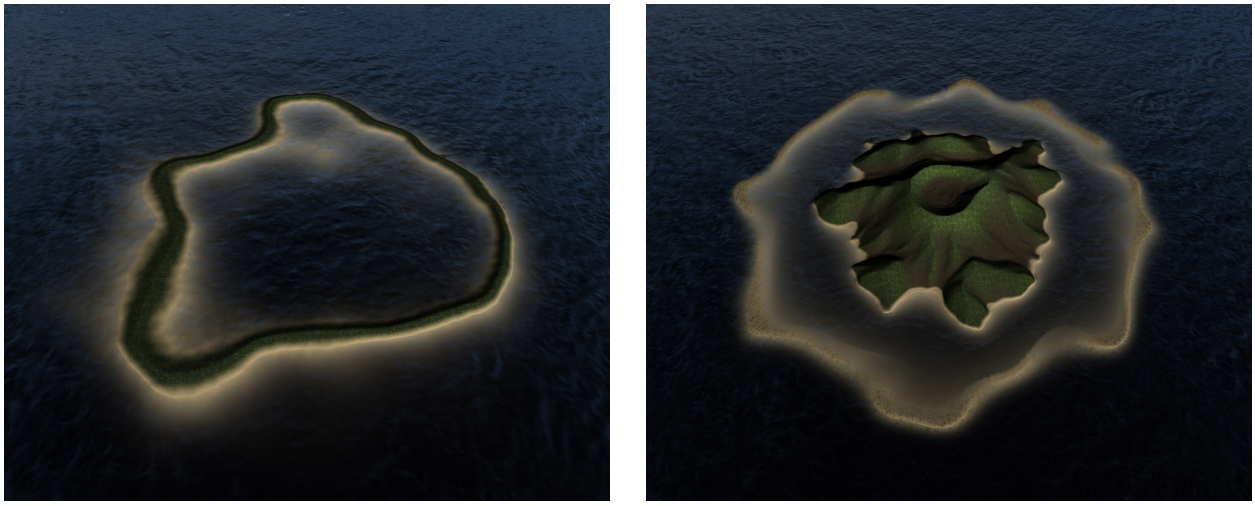
$$c_{\mathrm{w}}.a = e^{-g_t \kappa_{\mathrm{A}}}, \tag{32}$$

kjer $\kappa_{\mathrm{A}}$ nadzoruje moč slabljenja svetlobe (motnost vodne mase). Izračun opravlja ločen senčilnik, ki skrbi zgolj za izračun barve vsake točke na površini, ki pomeni morsko gladino.

## 3 REZULTATI

Uporabljamo višinske karte v velikosti 512×512 točk. Začetni teren se, predvsem zato, ker je opravilo prepuščeno centralni procesni enoti, v povprečju zgradi v 2,2 s. Ker se ta korak izvede samo enkrat v celotnem procesu, to ni moteče. Simulacije naravnih procesov brez vizualizacije se izvajajo hitro; 1000 iteracij hidravlične erozije se v povprečju izvede v 4 ms, 1000 iteracij termične erozije v 1 ms in 1000 iteracij rasti koralnega grebena v 2 ms. Vizualizacija dosega povprečno hitrost

[5] www.textures.com



Slika 4: Atol, generiran s pomočjo vulkanskega kraterja brez simuliranja posledic erozije (levo), vulkanski otok generiran s simulacijo hidravlične erozije, termične erozije in rasti koralnega grebena

200 fps (slik na sekundo). V celoti generiranje otoka želenega videza z vizualizacijo traja od 10 do 20 sekund, kar je odvisno od želene velikosti koralnega grebena. Brez vizualizacije pa generiranje traja približno 2,3 s, pri čemer simuliranje naravnih procesov traja 35 ms. Vse vrednosti veljajo za računalnik s procesorjem Intel Core i7 6700K, 16 GB delovnega pomnilnika in grafično kartico Nvidia GeForce GTX 1070, pri čemer smo v pogonu Unity 5 imeli nastavljen prikaz z najvišjo stopnjo podrobnosti.

Predstavljen postopek torej uspešno generira začetni teren, na katerem simulira hidravlično in termično erozijo ter nato še rast koralnega grebena. Končni teren z nanesenimi teksturami in izrisom morske gladine ima prepričljiv videz tropskega otoka. S spreminjanjem parametrov lahko dobimo veliko različnih oblik otoka, kot prikazujeta primera na sliki 4.

Z videzom tropskega otoka smo zadovoljni, hidravlična in termična erozija skupaj oblikujeta razgiban teren brez večjih vidnih napak. Dodan korak difuzije sedimenta se dobro obnese, vendar ni fizično natančen. Transport sedimenta in logika nalaganja sedimenta potrebujeta še nekaj izboljšav. Težava modela hidravlične erozije je v tem, da je težko uravnovesiti vse konstante v simulaciji. Simulacija hitro postane nestabilna, kar povzroča večje napake na terenu. Maške [10] je v svojem diplomskem delu dosegel boljši transport in nalaganje sedimenta. Hitrosti izvajanja pa zaradi uporabe različne strojne opreme ne moremo primerjati z njegovo.

Simulacija rasti koralnega grebena, ki je primarni dosežek našega dela, daje prepričljive rezultate in se izvaja hitro. Videz grebena je verodostojen, saj se ob obali otoka ustvari razgiban in občasno prekinjen greben. Naravni grebeni so mnogokrat prekinjeni in različno oddaljeni od obale. Tako kot pri hidravlični eroziji pa smo tudi pri simulaciji rasti koralnega grebena imeli težave z uravnovešanjem konstant, saj ima že zelo majhna sprememba velik vpliv na končni rezultat.

Dinamično nanašanje tekstur deluje hitro in daje prepričljiv videz. Vzorčne teksture lahko zamenjamo in tako na preprost način spremenimo videz otoka. Videz otoka lahko spreminjamo tudi s spreminjanjem parametrov v senčilniku, kot sta na primer gostota ponavljanja posamezne vzorčne teksture in naklon terena, pri katerem namesto teksture trave nanesemo teksturo skale.

Gladina morja veliko pripomore k prepričljivosti vizualizacije otoka. Senčilnik morske gladine se dobro obnese, saj nam daje tudi vizualno informacijo o globini vode, kar je še posebno koristno v okolici koralnega grebena. Prikaz sicer ni fizikalno natančen, saj zanemari kot pogleda pri izrisu globine in jo vedno izrisuje, kot da bi bil pogled od zgoraj navzdol. V praksi pa to ne povzroča težav, saj teren večino časa opazujemo iz zraka.

## 4 SKLEP

V delu smo predstavili postopek, ki na podlagi vhodnih parametrov generira teren v obliki otoka, na katerem v realnem času nato simulira hidravlično in termično erozijo ter za tem še rast koralnega grebena. Na teren dinamično nanaša teksture in izrisuje gladino morja. Zadani cilj, da ima generiran teren prepričljiv videz tropskega otoka z okoliškim koralnim grebenom, smo po našem mnenju dosegli.

Končni teren je uporaben predvsem v strateških igrah, kjer nanj večinoma gledamo iz oddaljenosti. Teren namreč nima dovolj podrobnosti za igre s pogledom iz prve osebe, kjer se igralec po terenu sprehaja in ga raziskuje. Čas, potreben za generiranje otoka želenega videza, je pri uporabi v računalniški igri lahko težava, vendar bi jo z izboljšanjem implementacije generiranja začetnega terena lahko odpravili. Generiranje začetnega terena bi lahko pospešili tako, da bi ga namesto v jeziku

C# implementirali v nižjenivojskem jeziku (npr. C++) ali pa na senčilniku.

Terenu bi lahko dodatno izboljšali videz s tem, da bi za njegovo predstavitev uporabili več plasti z različnimi trdotami, kar bi morali nato upoštevati tudi pri simulacijah erozije [19]. Funkcije, ki generirajo začetni teren, pa smo že zasnovali tako, da lahko izhod ene uporabimo kot vhod druge. S tem imamo več možnosti pri gradnji začetnega terena; nismo omejeni zgolj na obliko stožca.

## LITERATURA


[1] O. Korošec, *Proceduralno generiranje tropskega otoka*. Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, 2016, diplomsko delo.

[2] K. Perlin, "An image synthesizer," *ACM SIGGRAPH Computer Graphics*, vol. 19, no. 3, pp. 287–296, Jul. 1985.

[3] ——, "Noise hardware," in *Real-Time Shading Languages – SIGGRAPH Course Notes*, M. Olano, Ed., 2002.

[4] W. L. Raffe, F. Zambetta, and X. Li, "A survey of procedural terrain generation techniques using evolutionary algorithms," in *2012 CEC, IEEE Congress on Evolutionary Computation*. IEEE, 2012, pp. 1–8.

[5] T. J. Rose and A. G. Bakaoukas, "Algorithms and approaches for procedural terrain generation-a brief review of current techniques," in *VS-GAMES 2016, 8th International Conference on Games and Virtual Worlds for Serious Applications*. IEEE, 2016, pp. 1–2.

[6] F. K. Musgrave, C. E. Kolb, and R. S. Mace, "The synthesis and rendering of eroded fractal terrains," *SIGGRAPH Comput. Graph.*, vol. 23, no. 3, pp. 41–50, Jul. 1989.

[7] B. Beneš and R. Forsbach, "Visual simulation of hydraulic erosion," *Journal of WSCG*, vol. 10, no. 1-2, pp. 79–86, 2002.

[8] X. Mei, P. Decaudin, and B.-G. Hu, "Fast hydraulic erosion simulation and visualization on GPU," in *Proceedings of the 15th Pacific Conference on Computer Graphics and Applications*. IEEE, 2007, pp. 47–56.

[9] B. Jákó and B. Tóth, "Fast hydraulic and thermal erosion on GPU," in *Eurographics 2011 - Short Papers*, N. Avis and S. Lefebvre, Eds. The Eurographics Association, 2011, pp. 57–60.

[10] L. Maške, *Simulacija in vizualizacija erozije terena z uporabo GPE*. Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, 2013, diplomsko delo.

[11] T. Nakamura and T. Nakamori, "A geochemical model for coral reef formation," *Coral Reefs*, vol. 26, no. 4, pp. 741–755, 2007.

[12] ——, "A simulation model for coral reef formation: Reef topographies and growth patterns responding to relative sea-level histories," in *Sea Level Rise, Coastal Engineering, Shorelines and Tides*. Nova Science Publishers, 2011, pp. 251–262.

[13] J. F. O'Brien and J. K. Hodgins, "Dynamic simulation of splashing fluids," in *Proceedings of Computer Animation '95*. IEEE, 1995, pp. 198–205.

[14] J. M. Cecilia, J. M. García, and M. Ujaldón, "CUDA 2D stencil computations for the Jacobi method," in *International Workshop on Applied Parallel Computing*. Springer, 2010, pp. 173–183.

[15] M. Dermastia, T. Kiauta, and T. Turk, *Oxfordova ilustrirana enclikopedija žive narave*, 2nd ed., M. J. Coe, Ed. Ljubljana: DZS, 1995.

[16] L. Szirmay-Kalos and T. Umenhoffer, "Displacement mapping on the GPU – state of the art," *Computer Graphics Forum*, vol. 27, no. 6, pp. 1567–1592, 2008.

[17] M. Nießner, B. Keinert, M. Fisher, M. Stamminger, C. Loop, and H. Schäfer, "Real-time rendering techniques with hardware tessellation," *Computer Graphics Forum*, vol. 35, no. 1, pp. 113–137, 2016.

[18] E. Angel and D. Shreiner, *Interactive Computer Graphics: A Top-Down Approach with Shader-Based OpenGL*, 6th ed. USA: Addison-Wesley Publishing Company, 2011.

[19] O. Šťava, B. Beneš, M. Brisbin, and J. Křivánek, "Interactive terrain modeling using hydraulic erosion," in *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association, 2008, pp. 201–210.



**Oskar Korošec** je diplomiral (2016) na Fakulteti za računalništvo in informatiko v Ljubljani. Zaposlen je v podjetju v zasebnem sektorju, kjer se ukvarja z razvojem mobilnih aplikacij in iger.

**Iztok Lebar Bajec** je diplomiral (2000), magistriral (2002) in doktoriral (2005) na Fakulteti za računalništvo in informatiko. Na isti fakulteti je trenutno izredni profesor. Raziskovalno se ukvarja z modeliranjem in simulacijo skupinskega vedenja ter večvrednostno logiko v kvantnih celičnih avtomatih.