Segmentacija i klasifikacija elemenata na stranici dokumenta

Stefan Nožinić
student II godine master studija računarskih nauka
Departman za matematiku i informatiku
Prirodno-matematički fakultet
Univerzitet u Novom Sadu.

Tip rada: seminarski rad

Mentor: prof. dr Miloš Radovanović

22. novembar 2022.

Motivacija

- OCR
- Konverzija PDF u druge formate

Problem

Realtime Procedural Terrain Generation

Realtime Synthesis of Eroded Fractal Terrain for Use in Computer Games

Jacob Olsen, xenorg@imada.sdu.dk
Department of Mathematics And Computer Science (IMADA)
University of Southern Denmark

October 31, 2004

Abstract

The main goal of this paper is to provide a[n] overview of a variety of methods for synthesi[s] of eroded terrain for use in computer games, V[R] worlds and the like. Traditionally, such softwar[e] uses either predefined terrains or runtime gen[er]erated data based on simple fractal noise tech[n]niques.

In recent years, the advances in processing powe[r] of average home computers have made it pos[s]sible to simulate erosion processes near-realtim[e] by putting emphasis on speed at the expense o[f] physical correctness. This paper presents a fas[t] method to synthesize natural looking fractal ter[r]rain and then proceeds to evaluate and sugges[t] optimizations for two of the most commonly use[d] erosion algorithms [1, 2]. With some criteria fo[r] applicability in computer games in mind, a new and much faster algorithm is then proposed. Fi[-]nally, a few issues regarding terrain modification for maximum playability are discussed.

Figure 1: A rendered view of a synthesized ...

Definitions

Data representation

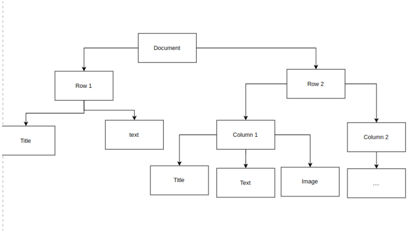
[I]n the algorithms described in this paper, terrain [w]ill be represented by two-dimensional height [m]aps using floating point values between 0 and 1. [U]nless otherwise stated, all examples use square [m]aps with side length $N = 2^9 = 512$, giving a [t]otal of $N^2 = 2^{18} = 262144$ cells, each cell con[t]aining a height value.

[T]he height map is denoted H and the individual [c]ells are addressed as $h_{i,j}$, where i and j are coor[d]inates ranging from 0 to 511. Some calculations [w]ill address cells outside this range; in this case, [m]odulo is used to wrap the coordinates around so [t]hat the right neighbour of a right-most cell will [b]e the left-most cell in the same row etc.

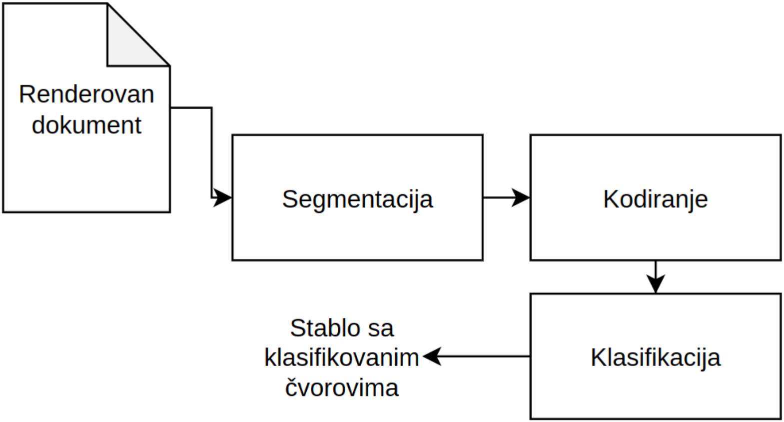
[A]ll implementations were done i Java, and all cal[c]ulation times are from tests executed on a fairly [s]tandard 2.4 GHz Pentium 4 PC.

Defining erosion

[T]he effects of erosion are difficult to describe [m]athematically: The term erosion covers many [n]aturally occurring phenomena, and different ter[r]ain types and climates will produce many differ[e]nt kinds of changes to a landscape. For sim[p]licity, a set of desirable traits (from a computer [g]ame development perspective) that will be used [t]o measure how eroded a height map is, is defined. [O]verall, most types of erosion dissolve material [f]rom steep slopes, transport it downhill and then [d]eposit the material at lower inclinations. This ...



Reprezentacija celog sistema



Ulazni podaci

- Ulazni podaci su dokument reprezentovan kao sekvenca stranica koje su slike u RBG formatu.
- Dokument je prošao proces uklanjanja šumova ako je skeniran.

Izlazni podaci

- Stablasta struktura
- Svaki čvor stabla ima dodeljeni region stranice ulaznog dokumenta.
- Svaki čvor stabla ima labelu koja označava semantičko značenje tog dela dokumenta.

Segmentacija dokumenta

Abstract

The main goal of this paper is to provide an overview of a variety of methods for synthesis of eroded terrain for use in computer games, VR worlds and the like. Traditionally, such software uses either predefined terrains or runtime generated data based on simple fractal noise techniques.

In recent years, the advances in processing power of average home computers have made it possible to simulate erosion processes near-realtime by putting emphasis on speed at the expense of physical correctness. This paper presents a fast method to synthesize natural looking fractal terrain and then proceeds to evaluate and suggest optimizations for two of the most commonly used erosion algorithms [1, 2]. With some criteria for applicability in computer games in mind, a new and much faster algorithm is then proposed. Finally, a few issues regarding terrain modifications for maximum playability are discussed.

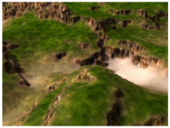


Figure 1: *A rendered view of a synthesized, eroded terrain created with the techniques discussed in this paper.*

Definitions

Data representation

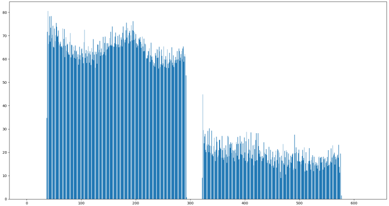
In the algorithms described in this paper, terrain will be represented by two-dimensional height maps using floating point values between 0 and 1. Unless otherwise stated, all examples use square maps with side length $N = 2^9 = 512$, giving a total of $N^2 = 2^{18} = 262144$ cells, each cell containing a height value.

The height map is denoted H and the individual cells are addressed as $h_{i,j}$, where i and j are coordinates ranging from 0 to 511. Some calculations will address cells outside this range; in this case, modulo is used to wrap the coordinates around so that the right neighbour of a right-most cell will be the left-most cell in the same row etc.

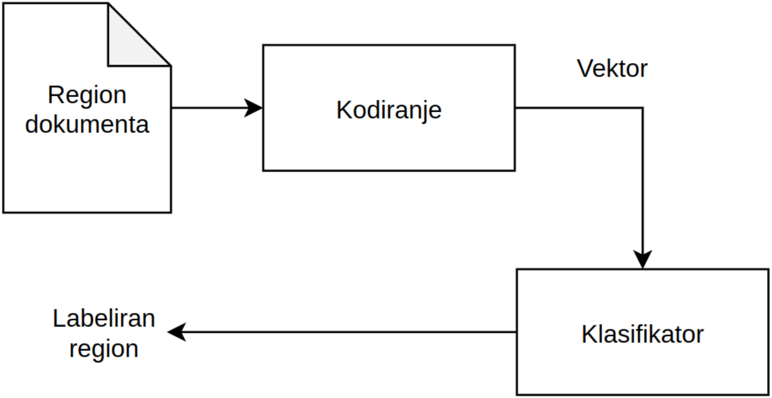
All implementations were done i Java, and all calculation times are from tests executed on a fairly standard 2.4 GHz Pentium 4 PC.

Defining erosion

The effects of erosion are difficult to describe mathematically: The term erosion covers many naturally occurring phenomena, and different terrain types and climates will produce many different kinds of changes to a landscape. For simplicity, a set of desirable traits (from a computer game development perspective) that will be used to measure how eroded a height map is, is defined. Overall, most types of erosion dissolve material from steep slopes, transport it downhill and then deposit the material at lower inclinations. This tends to make steep slopes even steeper, and flatten out low-altitude terrain when the transported material is deposited. To aid in the analysis of the changes in inclination, the slope map S is defined



Proces klasifikacije



Načini kodiranja

- `simple`: Element je kodiran kao vektor [E.x, E.y, E.h, E.w], odnosno njegova pozicija i veličina su predstavljeni kao vektor.
- `img_attrs`: Element je kodiran kao vektor: [E.h * E.w, E.w / (E.h*E.w), E.h / E.w, prosečna vrednost piksela u E]
- `pixels`: [R(x,y) za svako (x,y) na slici R veličine LxL]
- `histogram`: Elementi su sume vrednosti piksela po kolonama i vrstama.

Random decision forest

- 100 stabala za klasifikaciju
- kreiranje svakog stabla bilo ograničeno sa maksimalnom dubinom od 2
- Kriterijum podele je bio vrednosti *Gini* koeficijenta

Neuronska mreža

- jedan skriven sloj veličine 100 sa *ReLU* aktivacionom funkcijom
- Trenirana standardnim BP algoritmom

Rezultati

	Metod kodiranja	Klasifikator	Tačnost (%)
1	histogram	RF	71.792
2	img_attrs	RF	72.034
3	img_attrs	one rule	63.153
4	pixels	NN	38.451
5	simple	NN	42.345
6	simple	RF	70.422
7	simple	one rule	63.216

Tabela: Tačnost klasifikatora u odnosu na metode kodiranja

Zaključak

- *Random decision forest* je pokazao najbolje rezultate za izuzetno jednostavan metod kodiranja
- zadovoljavajući rezultati na 7 klasa.
- U Praktičnoj primeni, pogrešno klasifikovani elementi bi mogli da se modifikuju ručno.
- Neke pogrešno klasifikovane elemente ne treba smatrati podjednako lošim kao neke druge pogrešno klasifikovane elemente.

Poboljšanja

- Upotrebiti konvolucionu neuronsku mrežu.
- Proširiti skup za obuku.
- Određivanje matrice konfuzije za date klase.

Pitanja

?