GERAÇÃO PROCEDURAL DE TERRENOS VIRTUAIS COM APARÊNCIA NATURAL UTILIZANDO GPU

Aluno: Alex Seródio Gonçalves

Orientador: Dalton Solano dos Reis

FURB
UNIVERSIDADE DE BLUMENAU

Roteiro

- Introdução
- Objetivos
- Fundamentação teórica
- Trabalhos correlatos
- Descrição da ferramenta
- Resultados
- Conclusões

Introdução

- Eventos naturais que causam mudanças na paisagem são comumente estudados através de simulações computacionais.
- Um fator importante a se considerar nestes casos é o cenário no qual a simulação será executada.
- Terrenos com aparência natural gerados de forma procedural podem ser utilizados, proporcionando maior variedade de cenários.

Objetivos

- Disponibilizar uma ferramenta para geração de terrenos virtuais com aparência natural utilizando modelos estocásticos e físicos executados em GPU.

- Os objetivos específicos são:
  - analisar a performance dos algoritmos implementados comparando sua execução em CPU e GPU;
  - avaliar a qualidade dos terrenos gerados com base em terrenos reais.

Fundamentação teórica

Fundamentação Teórica

Modelos estocásticos e geração procedural

- Objetos gráficos são normalmente modelados através de equações determinísticas.
- Porém, modelos determinísticos não são adequados para modelar objetos naturais.
- Nestes casos, modelos estocásticos (não determinísticos) são mais adequados para representar as imperfeições naturais.

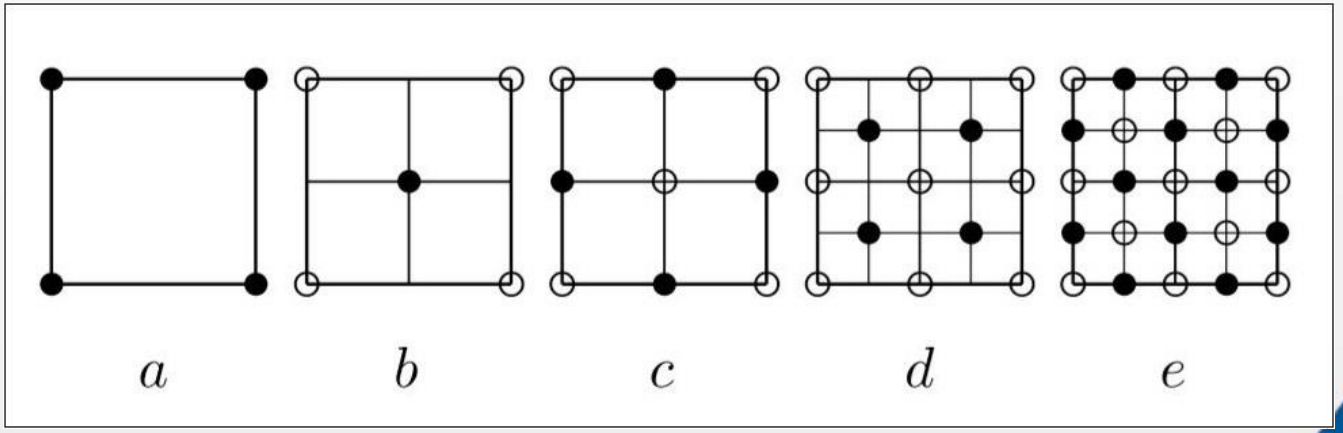
Fundamentação Teórica

Modelos estocásticos e geração procedural

Diamond-square

(FOURNIER; FUSSELL; CARPENTER, 1982)

- *diamond step*: partindo do vértice central do quadrado, calcula o valor deste vértice como a média dos quatro vizinhos na diagonal somada a um valor aleatório;
- *square step*: partindo do vértice central do quadrado, calcula o valor dos vizinhos na horizontal e vertical como sendo a média de seus vizinhos nos mesmos eixos mais um valor aleatório.



Fonte: Olsen (2004, p. 3).

Fundamentação Teórica

Modelos físicos e algoritmos de erosão

- Utilizados para simular processos físicos do mundo real por um determinado período.

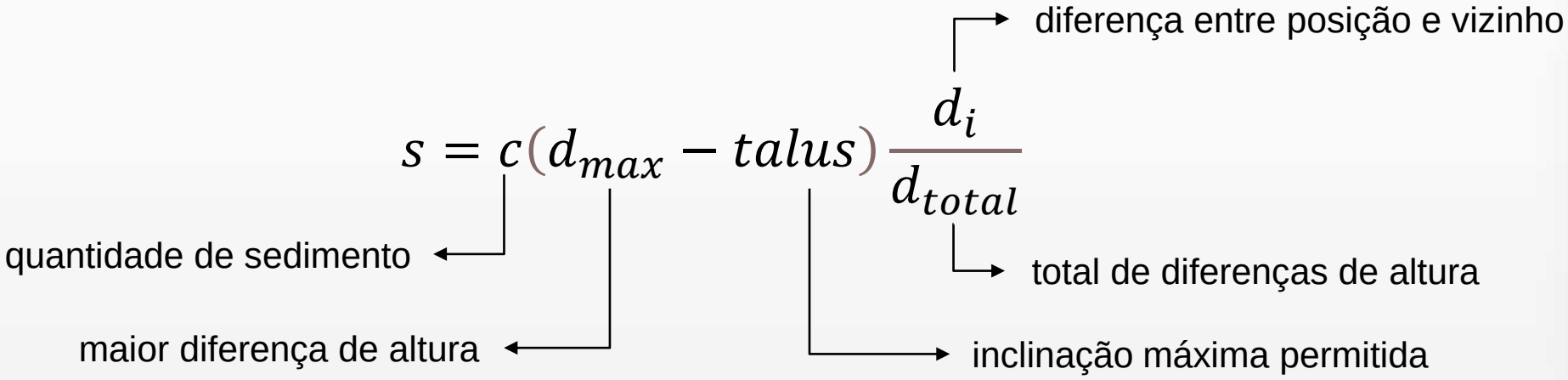
- Para terrenos geralmente trata-se de algoritmos de erosão, com os mais comuns sendo:
  - Erosão térmica: move sedimentos de áreas íngremes para áreas planas;
  - Erosão hidráulica: move sedimentos através do fluxo de escoamento de água da chuva.

Fundamentação Teórica

Modelos físicos e algoritmos de erosão

Erosão térmica

(OLSEN, 2004)



Fundamentação Teórica

Modelos físicos e algoritmos de erosão

Erosão hidráulica

(DIEGOLI NETO, 2017) e (BENEŠ; FORSBACH, 2002)

a) aparecimento de água da chuva

$$w_{i,j} = w_{i,j} + r$$

quantidade de chuva

Fundamentação Teórica

Modelos físicos e algoritmos de erosão

Erosão hidráulica

(DIEGOLI NETO, 2017) e (BENEŠ; FORSBACH, 2002)

a) aparecimento de água da chuva

$$w_{i,j} = w_{i,j} + r$$

quantidade de chuva

b) água transforma solo em sedimento

$$h_{i,j} = h_{i,j} - s(w_{i,j} - h_{i,j})$$

fator de solubilidade

Fundamentação Teórica

Modelos físicos e algoritmos de erosão

Erosão hidráulica

(DIEGOLI NETO, 2017) e (BENEŠ; FORSBACH, 2002)

a) aparecimento de água da chuva

$$w_{i,j} = w_{i,j} + r$$

quantidade de chuva

b) água transforma solo em sedimento

$$h_{i,j} = h_{i,j} - s(w_{i,j} - h_{i,j})$$

fator de solubilidade

c) água e sedimento são transportados

$$\Delta w_{i,j} = w_{i,j} + \min(w, \Delta a)\frac{d_i}{d_{total}}$$

w – média dos vizinhos

Fundamentação Teórica

Modelos físicos e algoritmos de erosão

Erosão hidráulica

(DIEGOLI NETO, 2017) e (BENEŠ; FORSBACH, 2002)

a) aparecimento de água da chuva

$$w_{i,j} = w_{i,j} + r$$

quantidade de chuva

b) água transforma solo em sedimento

$$h_{i,j} = h_{i,j} - s(w_{i,j} - h_{i,j})$$

fator de solubilidade

c) água e sedimento são transportados

$$\Delta w_{i,j} = w_{i,j} + \min(w, \Delta a)\frac{d_i}{d_{total}}$$

w – média dos vizinhos

d) sedimento é depositado e água evapora

$$\Delta s = (w_{i,j} - h_{i,j}) - ((w_{i,j} - h_{i,j})(1 - e))$$

$$w_{i,j} = w_{i,j} - \Delta s$$

$$h_{i,j} = h_{i,j} + s(\Delta s)$$

fator de evaporação

Trabalhos Correlatos (1/3)

Realtime Procedural Terrain Generation

Olsen (2004)

- Apresenta e analisa métodos de geração procedural de terrenos erodidos em tempo real.
- Usa os algoritmos *diamond-square*, diagrama de Voronoi, erosão térmica e hidráulica.
- Define a métrica *erosion score.*
- Alcançou resultados adequados para a utilização em jogos.

Trabalhos Correlatos (2/3)

Simulação de dinâmica de relevo

Diegoli Neto (2017)

- Apresenta o desenvolvimento de um simulador de deslizamentos de terra de forma simplificada.
- Utiliza algoritmos de erosão térmica e hidráulica com quatro camadas diferentes de solo.
- Atingiu uma visualização condizente com a realidade, porém com pouca precisão.

Trabalhos Correlatos (3/3)

Arches

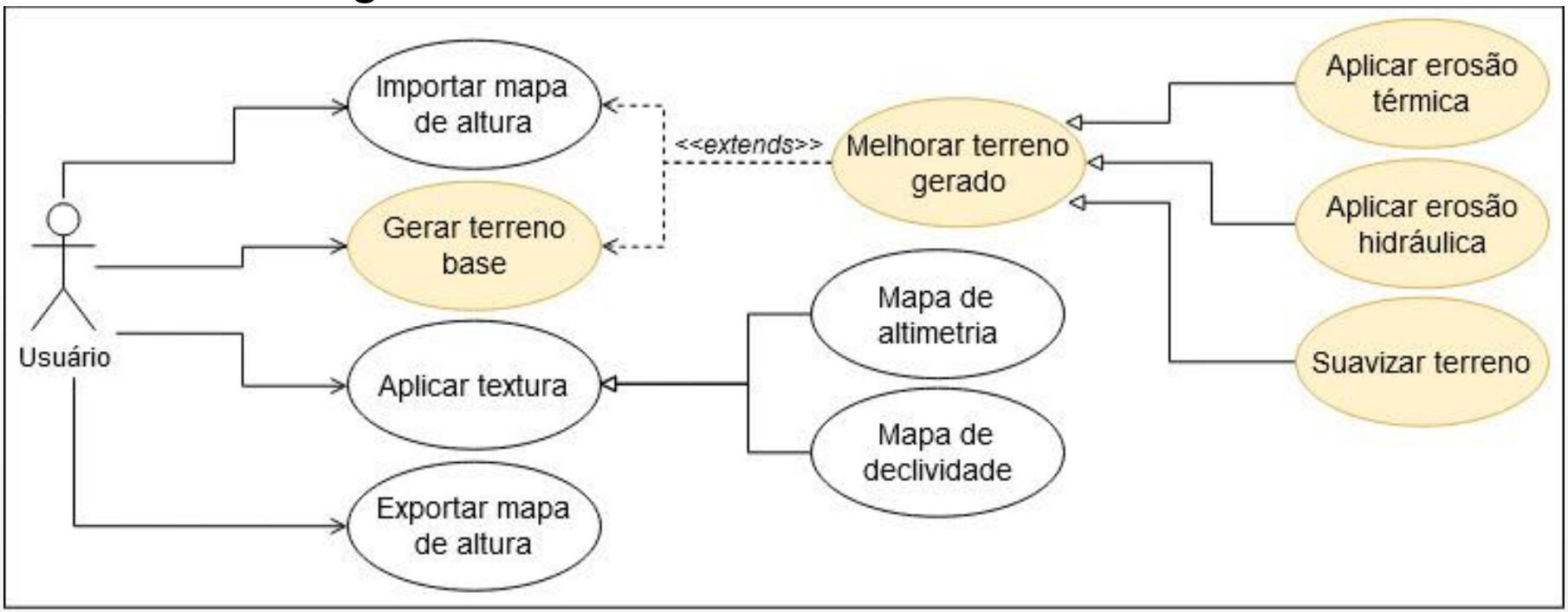
Peytavie *et al*. (2009)

- Apresenta a modelagem de terrenos com características de relevo complexas.
- Trabalha com diferentes tipos de materiais organizados em pilhas.
- Possui algoritmos para estabilização de materiais e criação de formações rochosas.
- Apresentou uma abordagem original para representação de estruturas complexas.

Descrição da ferramenta

Especificação

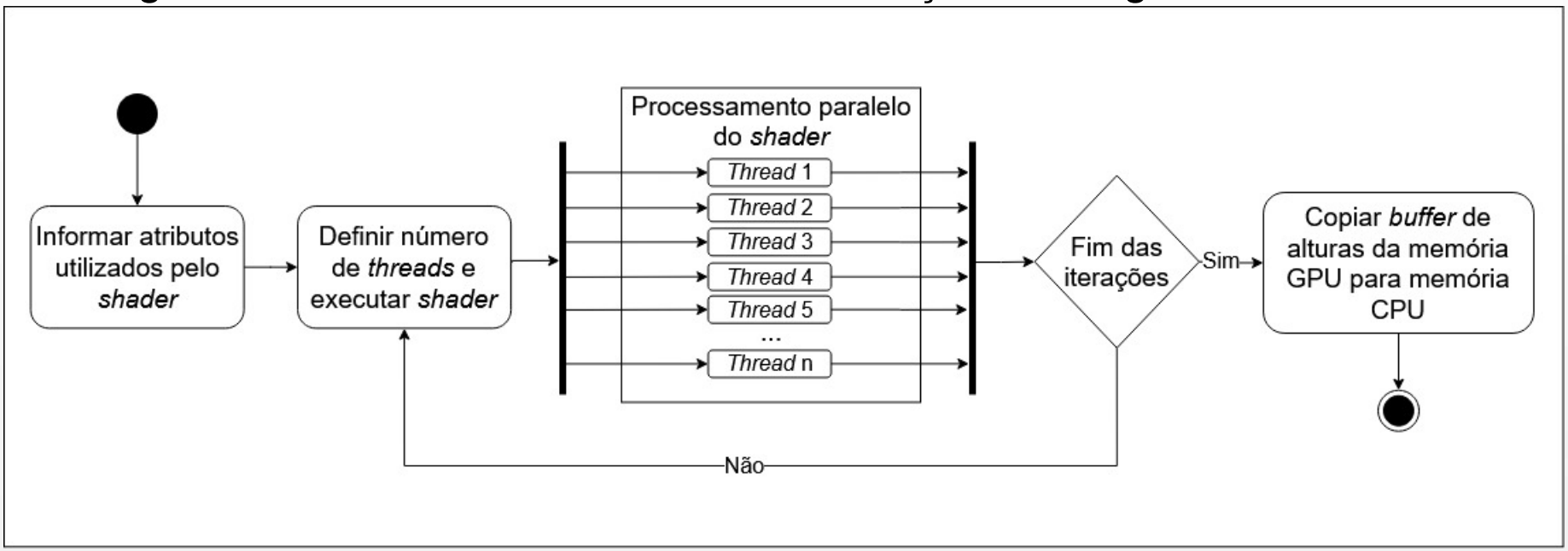
Diagrama de casos de uso da ferramenta



Executado em GPU com *compute shaders*

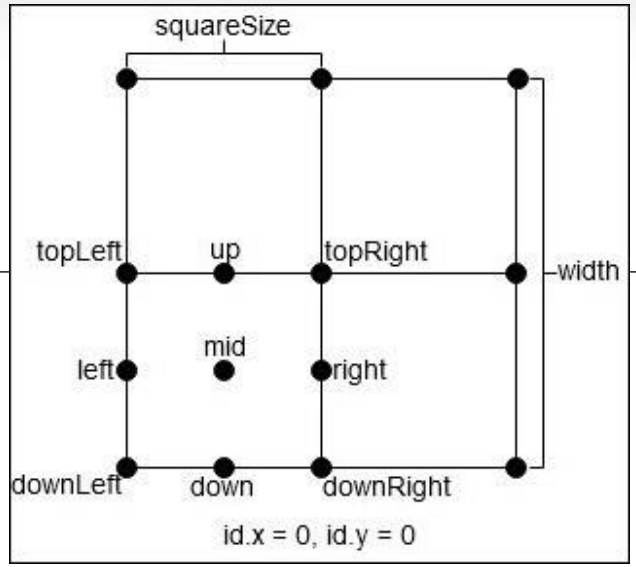
Especificação

Diagrama de atividades do fluxo de execução dos algoritmos em GPU



Implementação

Diamond-square




```
uint col = id.x * squareSize;
uint row = id.y * squareSize;

uint rowSize = width + 1;
uint halfSize = squareSize / 2;

uint mid = (row + halfSize) * rowSize + (col + halfSize);
uint topLeft = row * rowSize + col;
uint topRight = row * rowSize + (col + squareSize);
uint bottomLeft = (row + squareSize) * rowSize + col;
uint bottomRight = (row + squareSize) * rowSize + (col + squareSize);

uint up = topLeft + halfSize;
uint down = bottomLeft + halfSize;
uint left = mid - halfSize;
uint right = mid + halfSize;
```

```
// diamond step
map[mid] = (map[topLeft]+map[topRight]+map[bottomLeft]+map[bottomRight])/4 + rand()*H;

// square step
map[up]= (map[topLeft]+map[topRight]+map[mid]+map[up+halfSize])/4 + rand()*H;
map[down]= (map[bottomLeft]+map[bottomRight]+map[mid]+map[down-halfSize])/4 + rand()*H;
map[left]= (map[topLeft]+map[bottomLeft]+map[mid]+map[left-halfSize])/4 + rand()*h;
map[right]= (map[topRight]+map[bottomRight]+map[mid]+map[right+halfSize])/4 + rand()*H;
```

Erosão térmica

```
uint currentPosition = id.x + (id.y * width);
getNeighbors(currentPosition);

float dMax = 0;
float dTotal = 0;
float di;

for(uint i = 0; i < NEIGHBORHOOD_SIZE; i++) {
   di = heightmap[currentPosition] - heightmap[neighbors[i]];
   if (d > talus) {
      dTotal += di;
      if (di > dMax)
         dMax = d;
   }
}
for(uint j = 0; j < NEIGHBORHOOD_SIZE; j++) {
   di = heightmap[currentPosition] - heightmap[neighbors[j]];

   if (di > talus && isValidPosition(id)) {
      float sediment = c * (dMax - talus) * (di / dTotal);

      heightmap[currentPosition] -= sediment;
      heightmap[neighbors[j]] += sediment;
   }
}
```

Análise dos Resultados

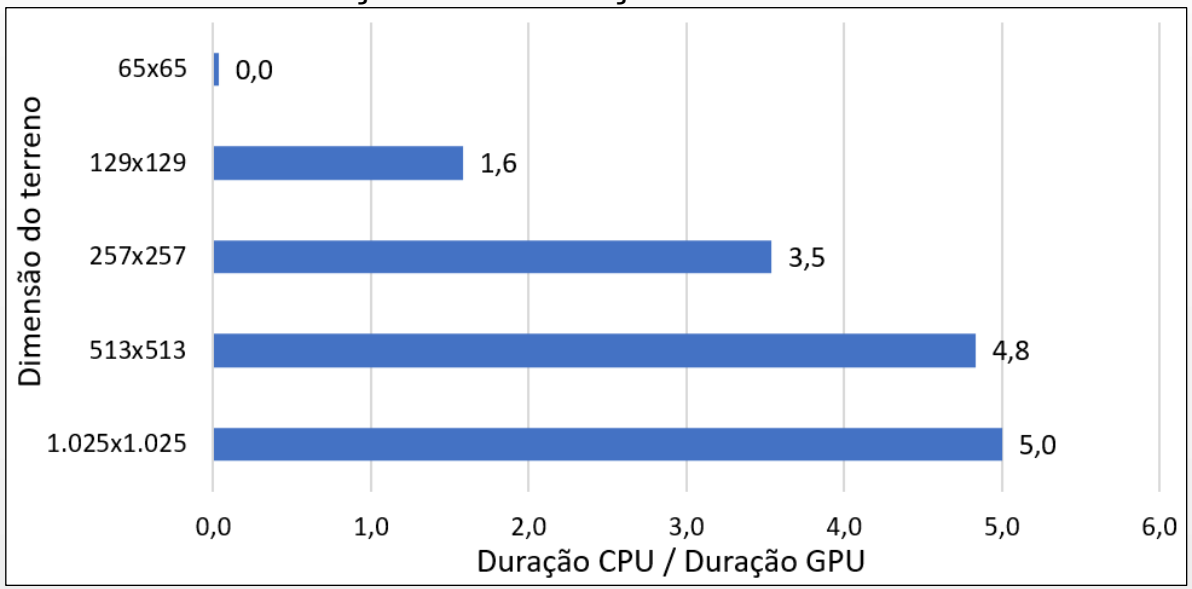
Performance

Diamond-square

Tempo de execução em CPU e GPU

Dimensões	CPU (ms)	GPU (ms)	Nº total *threads*
65x65	0,02	0,62	127
129x129	1,36	0,86	255
257x257	7,36	2,08	511
513x513	31,6	6,54	1.023
1.025x1.025	131,24	26,24	2.047

Diferença de execução em CPU e GPU

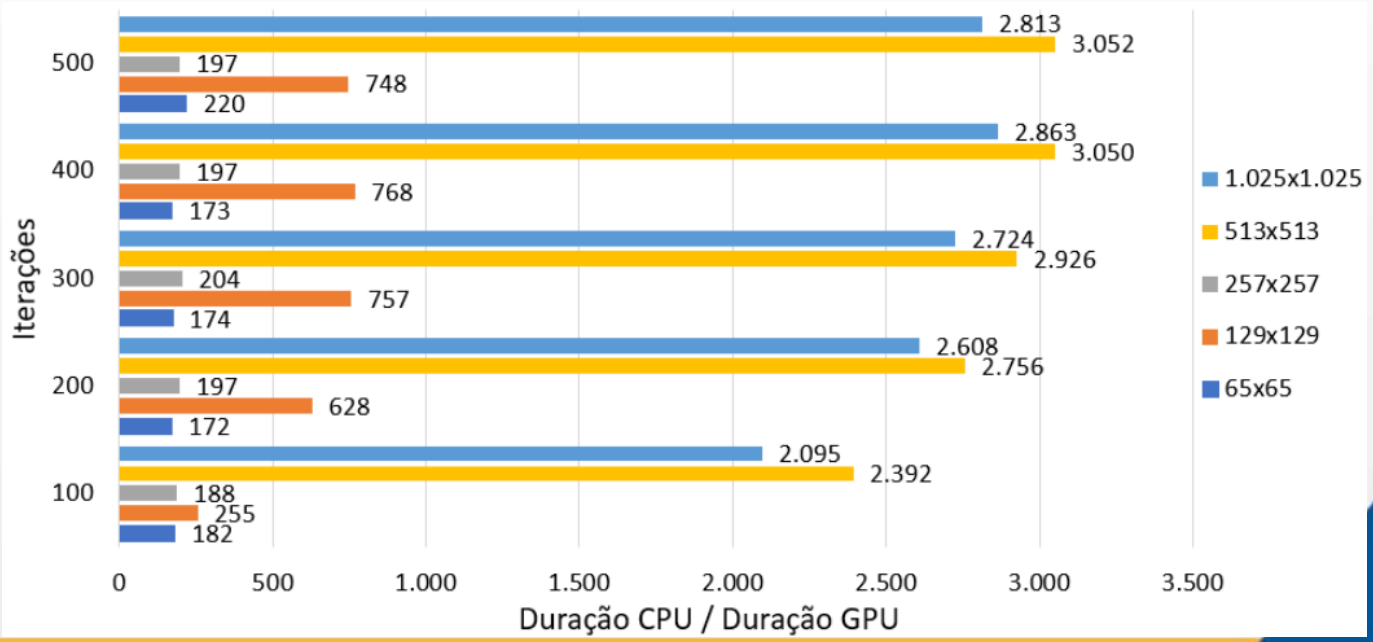


Erosão térmica

Tempo de execução em CPU e GPU

	100 iterações		200 iterações		300 iterações		400 iterações		500 iterações	
Dimensões	CPU	GPU	CPU	GPU	CPU	GPU	CPU	GPU	CPU	GPU
65x65	182	1	344	2	523	3	692	4	880	4
129x129	765	3	1.508	2	2.271	3	3.070	4	3.739	5
257x257	2.636	14	5.312	27	8.171	40	10.430	53	12.984	66
513x513	9.567	4	19.290	7	29.261	10	39.656	13	48.828	16
1.025x1.025	37.706	18	75.620	29	111.701	41	151.762	53	180.006	64

Diferença de execução em CPU e GPU



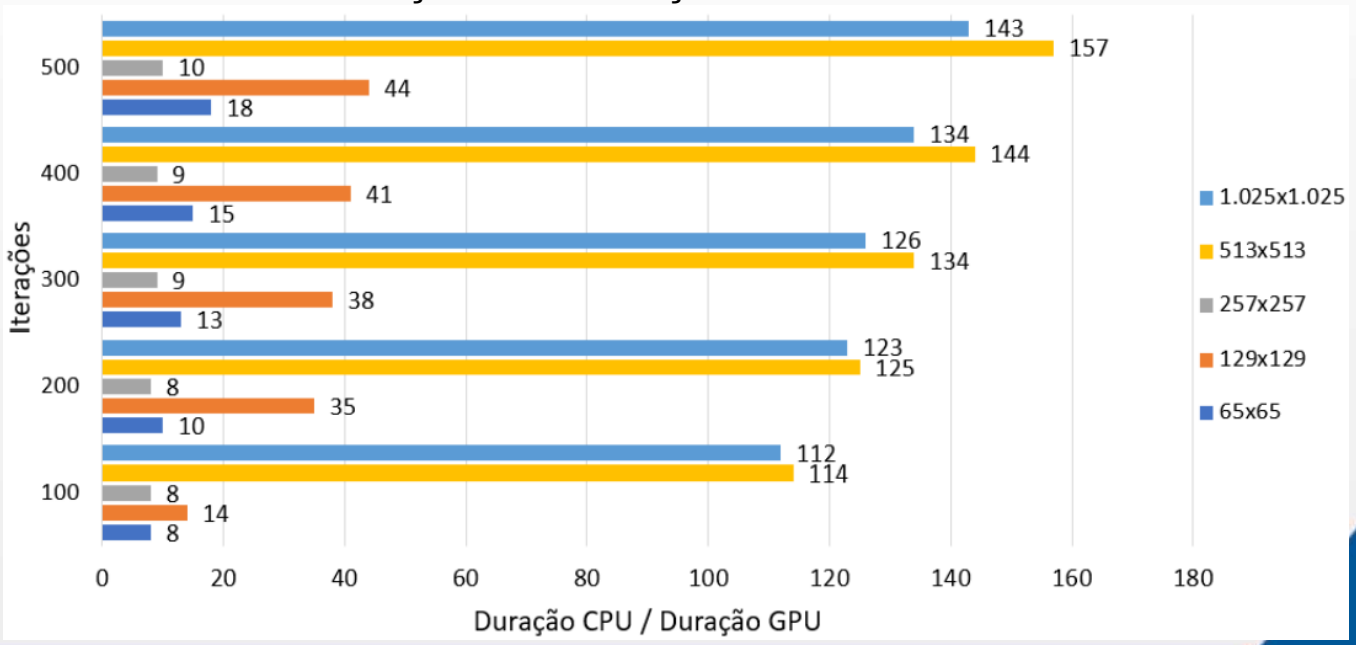
FURB
UNIVERSIDADE DE BLUMENAU

Erosão hidráulica

Tempo de execução em CPU e GPU

	100 iterações		200 iterações		300 iterações		400 iterações		500 iterações	
Dimensões	CPU	GPU	CPU	GPU	CPU	GPU	CPU	GPU	CPU	GPU
65x65	44	5	87	9	132	10	177	12	235	13
129x129	175	13	344	10	518	14	703	17	920	21
257x257	683	84	1.362	164	2.054	241	2.711	312	3.601	376
513x513	2.848	25	5.620	45	8.645	64	11.796	82	15.407	98
1.025x1.025	10.955	98	21.830	178	32.360	256	43.879	328	55.879	391

Diferença de execução em CPU e GPU



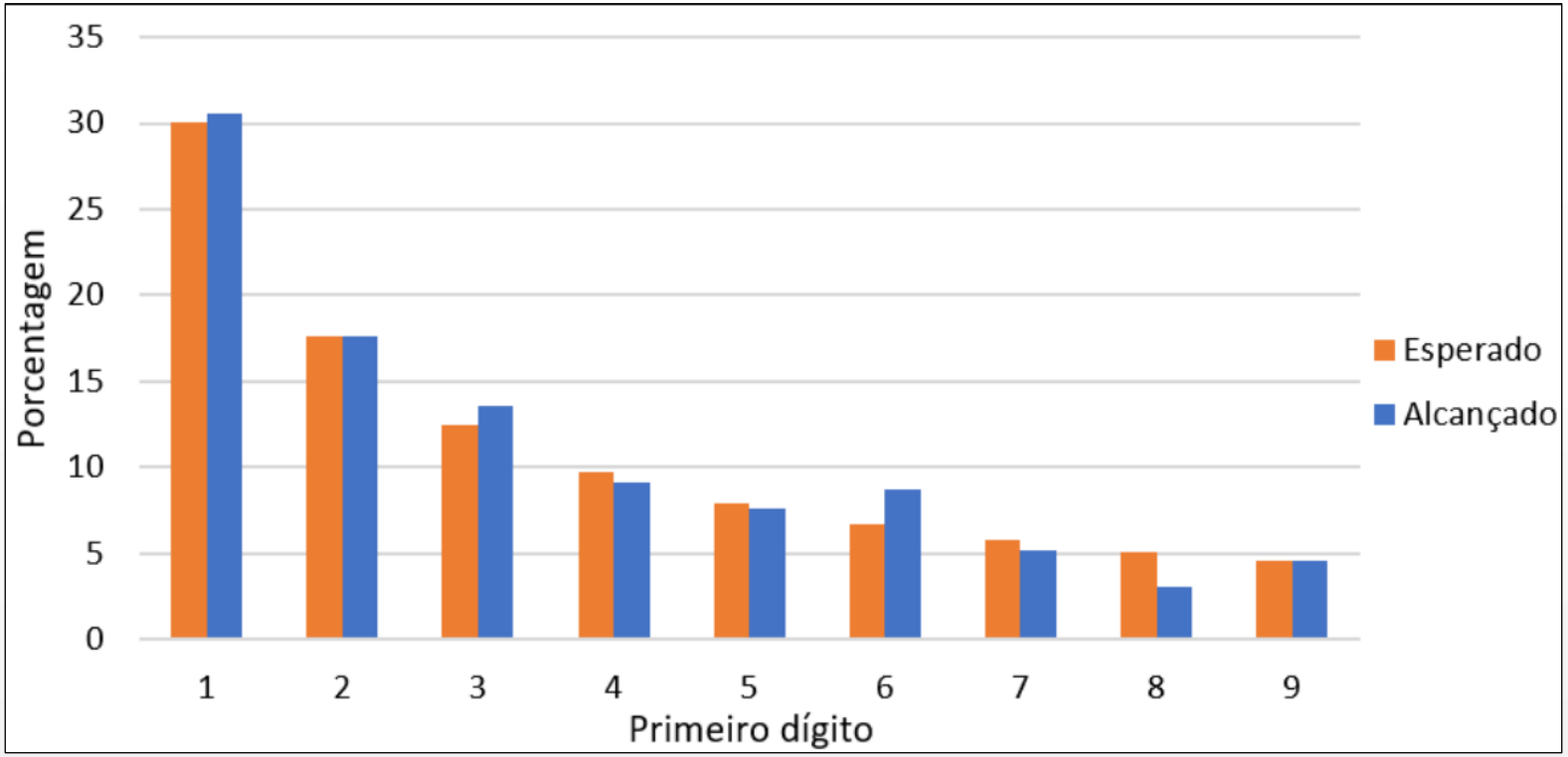
FURB
UNIVERSIDADE DE BLUMENAU

Análise dos Resultados

Naturalidade

Terrenos reais

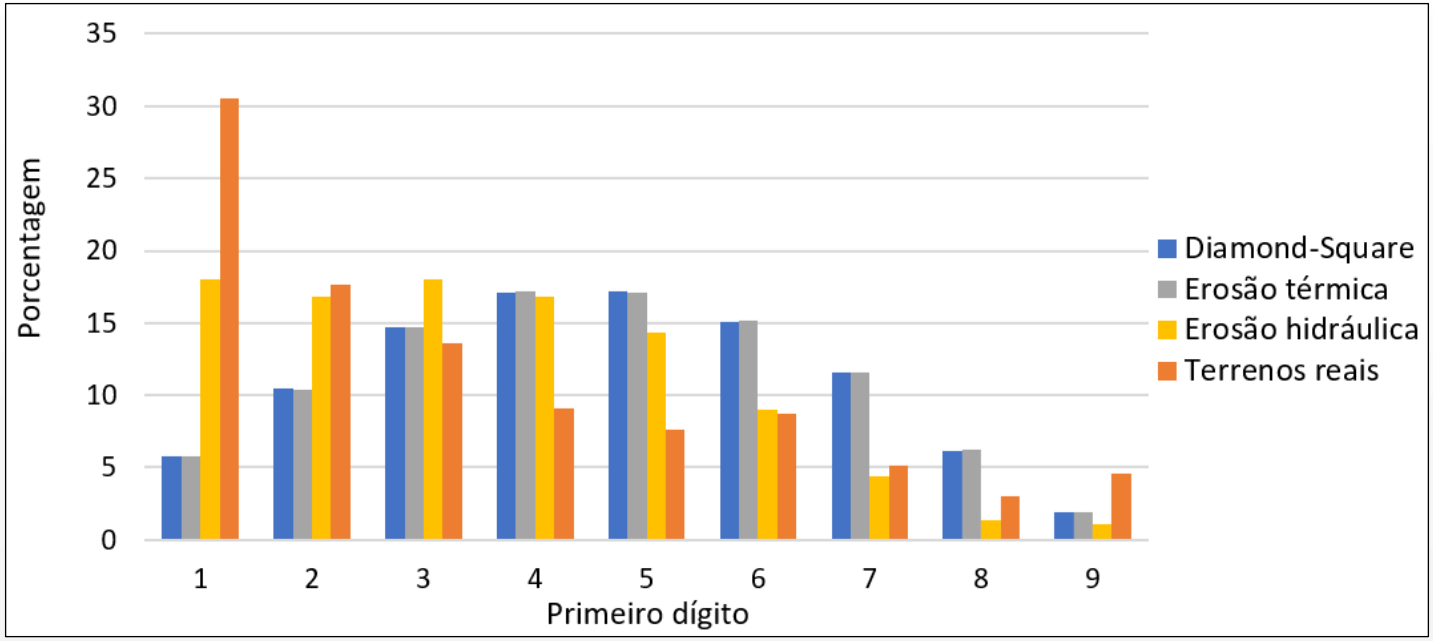
Distribuição da lei de Benford em 30 terrenos reais



Os terrenos foram coletados através do site terrain.party
Alcançaram em média 0,697 no *erosion score*.

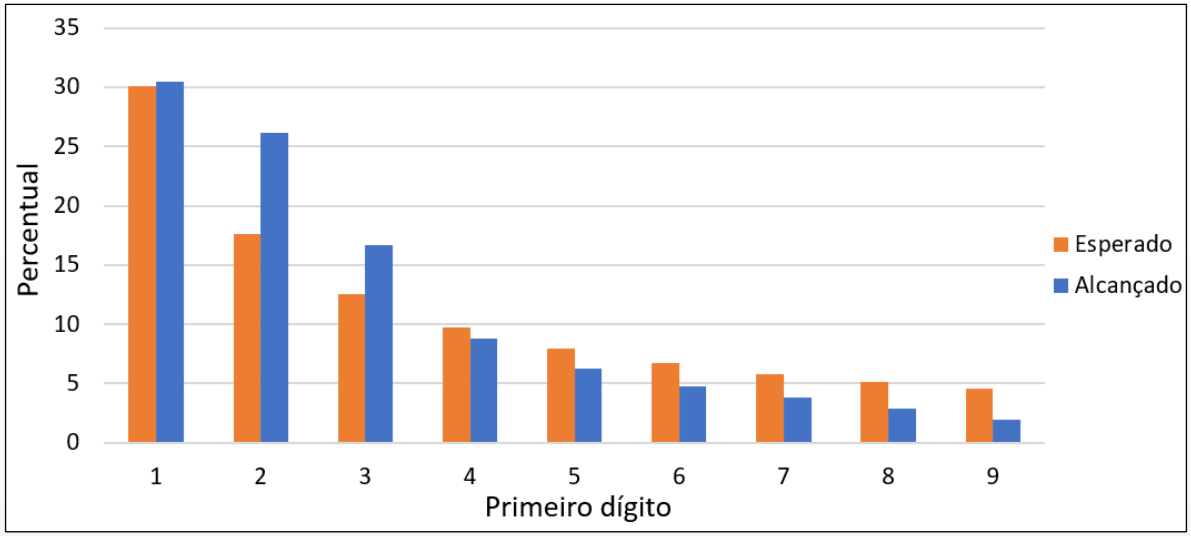
Terrenos gerados pela ferramenta

Lei de Benford e *erosion score* 50 terrenos gerados com os três algoritmos



Modelos	Erosion score
Diamond-square	0,431
Erosão térmica	0,403
Erosão hidráulica	0,487
Terrenos reais	0,697

Terrenos gerados com filtro de melhores resultados



- Alcançaram em média 0,484 no *erosion score*.
- Problemas:
  - tempo de execução varia imprevisivelmente entre 2 segundos a até 6 minutos.
  - diversificação da distribuição muito baixa.

Conclusões

- O objetivo principal foi alcançado, porém com limitações na naturalidade dos terrenos.
- As versões em GPU dos algoritmos se mostraram melhores em todos os cenários.
- As métricas *erosion score* e lei de Benford utilizadas para medir a naturalidade dos terrenos se mostraram adequadas.

Conclusões

- A geração de terrenos com aparência natural foi desenvolvido mas não totalmente alcançado.

- O trabalho apresenta uma análise comparativa entre terrenos gerados e reais não vista em nenhum dos autores estudados.

Sugestões de melhorias

a) incluir diferentes camadas de solo que influenciem nas transformações de relevo;
b) utilizar outros algoritmos de erosão como Beyer (2015) para erosão hidráulica;
c) analisar mais terrenos reais a fim de coletar outras informações de naturalidade;
d) expandir os estudos sobre a lei de Benford, *erosion score* e outras métricas.

Apresentação prática