Lukáš Polok, Radek Bartoň, Petr Chudý, Pavel Smrž, Přemysl Kršek, Petr Dittrich

Terrain Rendering Algorithm Performance Analysis

#0 Content

1. **Introduction**
2. **Motivation**
3. **Optimizing options**
4. **Our contribution**
5. **Results**
6. **Conclusions**

Lukáš Polok, Radek Bartoň, Petr Chudý, Pavel Smrž, Přemysl Kršek, Petr Dittrich

#1 Introduction

Terrain is a major part of information the flight guidance equipment offers to the pilot

Lukáš Polok, Radek Bartoň, Petr Chudý, Pavel Smrž, Přemysl Kršek, Petr Dittrich

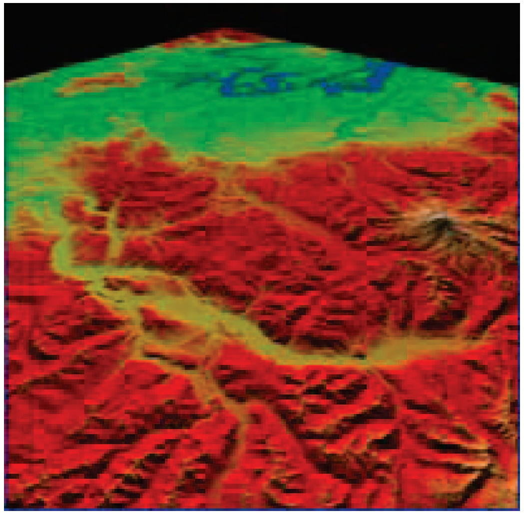
#2 Motivation

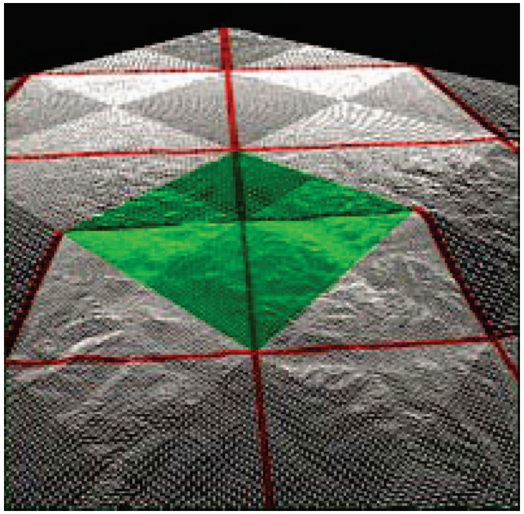
- Realistic real-time terrain rendering is easy on todays consumer GPUs
- But what about low-power?
- Consider the number of GPU cores
  - NVIDIA Riva TNT (1998): 6
  - NVIDIA GeForce256 (1999): 12
  - NVIDIA GeForce GTX 690 (2012): 3392
  - NVIDIA Tegra 2 (2010): 8
- Performance sets us back a decade ago!

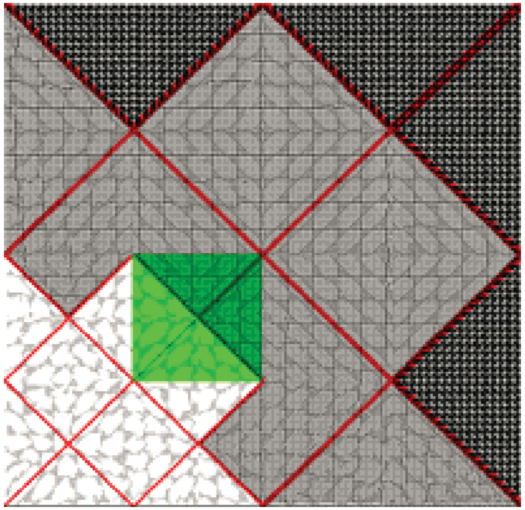
Lukáš Polok, Radek Bartoň, Petr Chudý, Pavel Smrž, Přemysl Kršek, Petr Dittrich

#3 Optimizing options

- Use a fast modern terrain rendering algorithm
  - Mostly impossible without modifications because of the missing support for vertex texturing
  - We use a modification of „Seamless Patches for GPU-based Terrain Rendering“ (Livny Y., Kogan Z. and El-Sana J., 2008)







Lukáš Polok, Radek Bartoň, Petr Chudý, Pavel Smrž, Přemysl Kršek, Petr Dittrich

#3 Optimizing options (2)

- Compromise
  - Set low level of detail (reduces number of primitives)
  - Do not use textures (reduces memory bandwidth)
  - Do not use complicated lighting (reduces shading)

⇩

  - Then it will not look really nicely,
    but it is still OK for flight guidance

Lukáš Polok, Radek Bartoň, Petr Chudý, Pavel Smrž, Přemysl Kršek, Petr Dittrich

#3 Optimizing options – Profiling

Operation		Time [mS]
Vertex buffer swap		3.52
Index buffer swap		4.01
Shader reconfig		2.90
Vertex attrib reconfig		7.54
Vertex attrib pointer reconfig		23.38
Tile upload (batch size)	1	185.78
	2	152.96
	4	138.12
	8	159.77
	16	167.38
	32	167.63
	64	156.29
	128	155.50
	256	154.46
	512	153.44
	1024	150.62
Standard deviation		0.314375

a **LOT** of time

Lukáš Polok, Radek Bartoň, Petr Chudý, Pavel Smrž, Přemysl Kršek, Petr Dittrich

#3 Optimizing options – Profiling: Discussion

- **A considerable amount of time is spent when moving terrain tiles from CPU to GPU**
- **This would be usually done by using toroidal buffers implemented using textures**
- **But on embedded GPUs we have no vertex textures**
- **Have to send tiles directly as geometry in VBO**
- **There is a lot of tiles and no way to reuse (interpolate) between them**
- **High number of buffers to update, with high frequency**

Lukáš Polok, Radek Bartoň, Petr Chudý, Pavel Smrž, Přemysl Kršek, Petr Dittrich

#4 Our contribution

- **Managing tile data in VBO is complicated, as it is essentially a 1D array**
- **Keeping a single VBO for each tile is simple, but it incurs high overhead of bus latency**
- **Our contribution**
  - **Allocation policy for keeping data for many tiles in a single block of memory.**

Lukáš Polok, Radek Bartoň, Petr Chudý, Pavel Smrž, Přemysl Kršek, Petr Dittrich

#4 Our contribution (2)

- Memory allocation policy
- Design goals
  - Must be fast
  - Must be able to keep data in two memory spaces (the GPU and CPU space) in sync
  - Must optimize for batching many small transfers into less large ones

Lukáš Polok, Radek Bartoň, Petr Chudý, Pavel Smrž, Přemysl Kršek, Petr Dittrich

#4 Our contribution (3)

- Memory allocation policy
- Algorithm
  - Works in two passes
  - In the first pass, it queries the terrain rendering algorithm for number and size of data to be transmitted to GPU
  - On GPU side, It allocates all the data in one contiguous block and places it on the lowest memory addess where it can fit
  - On CPU side, a buffer is kept for assembling the data which makes it possible to later transfer them at once
  - In the second pass, the terrain rendering algorithm fills the data inside the prepared buffer

Lukáš Polok, Radek Bartoň, Petr Chudý, Pavel Smrž, Přemysl Kršek, Petr Dittrich

#5 Results

- **The allocation policy is not perfect, i.e. it results in some wasted space**
  - **Theoretically a lot (100% with infinite number of tiles)**
  - **Benchmarks were run to find out how it behaves in real world**

Block size distribution	Space (average)
Constant (S)	154.231 %
Gaussian with center at S	186.608 %
Uniform between 0 and S	82.573 %

Lukáš Polok, Radek Bartoň, Petr Chudý, Pavel Smrž, Přemysl Kršek, Petr Dittrich

#5 Results (2)

- Runtime complexity
- To perform allocation of N blocks with M blocks being already allocated:

$$O(N) + [O(N \log N)] + O(\log M)$$

  - The first term is block IDs and sizes collection
  - The second term is memory alignment (requires sorting to be efficient, but is not always required)
  - The third term is the search for a block of free space of given minimal size (can be diminished by hierarchical representation of the blocks)

Lukáš Polok, Radek Bartoň, Petr Chudý, Pavel Smrž, Přemysl Kršek, Petr Dittrich

#5 Results (3)

- Results in real-world rendering loop
- Our system architecture is layered
  - Terrain dataset
  - Tile reader
  - Tile interpolator
  - Tile cache
  - GPU allocator
  - GPU
- Improvements of 10s of milliseconds per frame were reached while flying by the terrain.

Lukáš Polok, Radek Bartoň, Petr Chudý, Pavel Smrž, Přemysl Kršek, Petr Dittrich

#6 Conclusion

- A simple novel memory allocation algorithm was presented
  - Great speed (O(1) per allocated memory block)
  - Handles data in two memory spaces (GPU and CPU)
  - Naturally batches memory blocks for fast transfers
- The usefulness was demonstrated for terrain rendering on mobile GPUs
- It can be used for much more (rendering GUI)

Lukáš Polok, Radek Bartoň, Petr Chudý, Pavel Smrž, Přemysl Kršek, Petr Dittrich

Thank you for your attention!