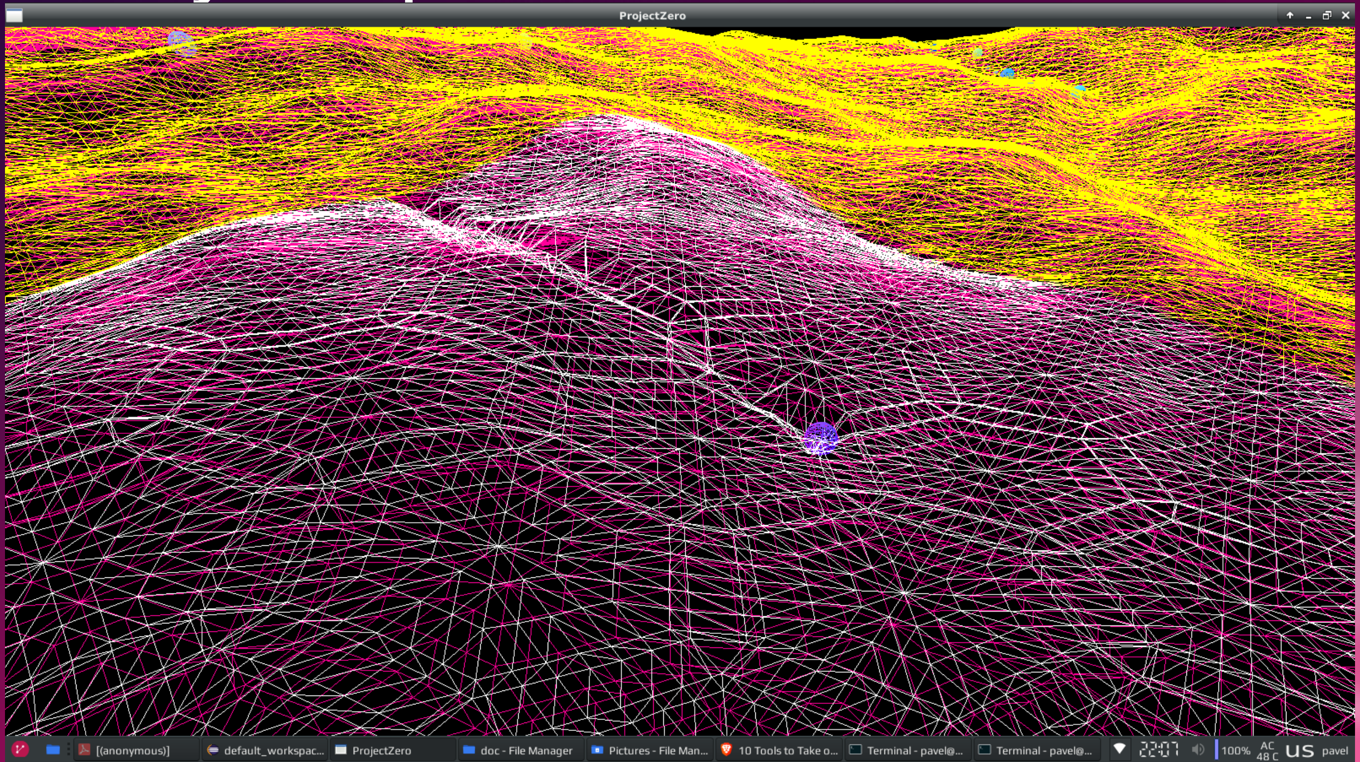
Deferred Snow Deformation in Rise of the Tomb Raider

Authors of the technique:
Anton Kai Michels and Peter Sikachev

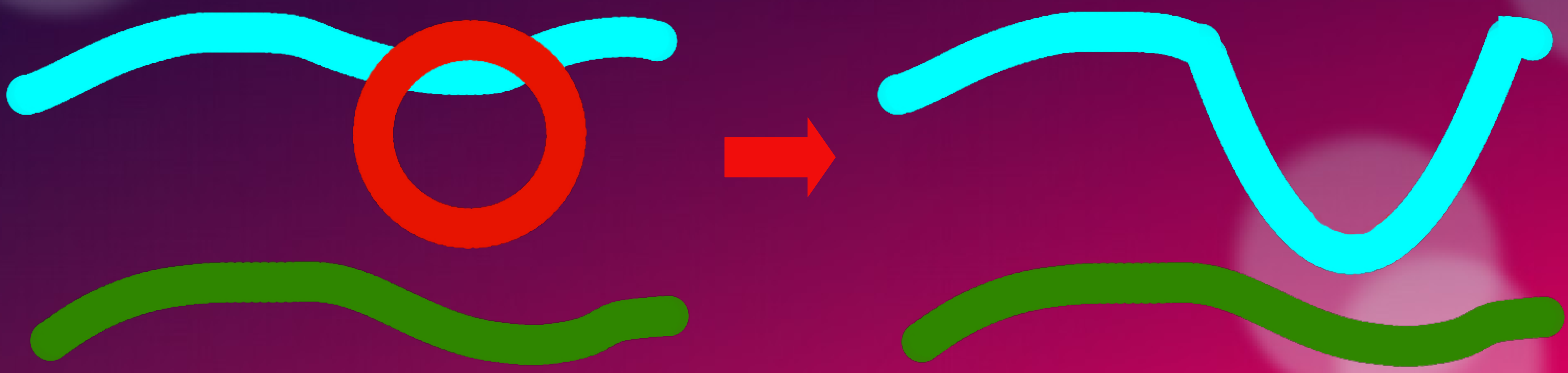
Idea

- Decoupling the deformation logic from geometry
- Scale well with large amount of objects
- Low memory footprint



Basic approach

- Render terrain and snow into heightmaps
- Render dynamic objects into deformation map
- During render, clamp the values
- Sample the deformation map during snow render

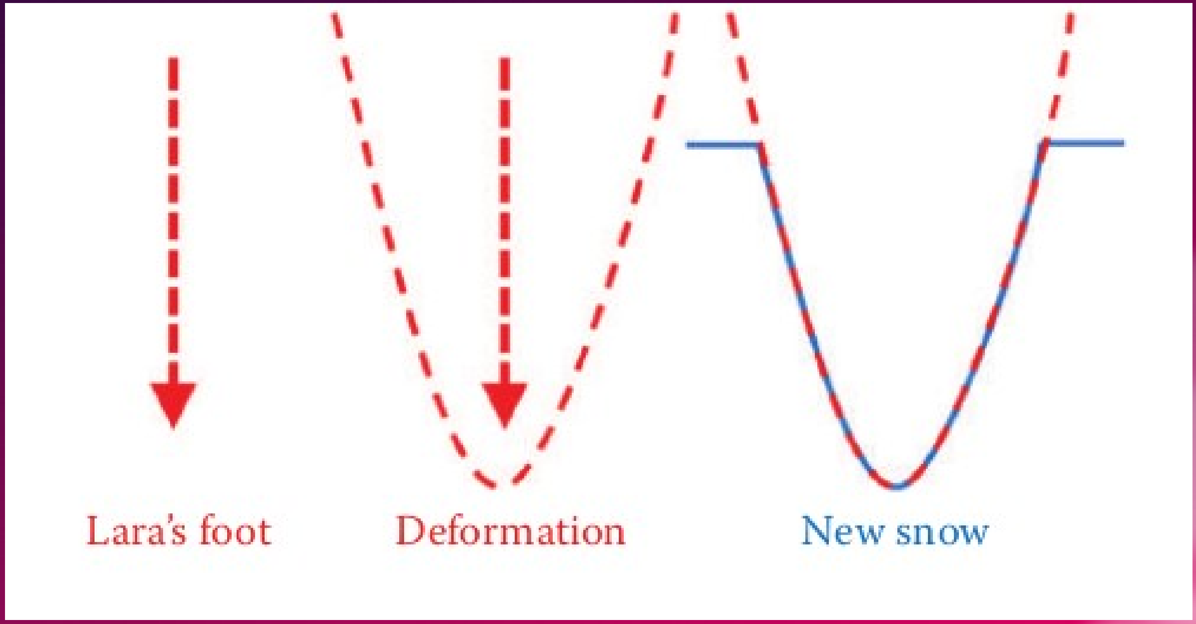


Deferred deformation

- Single uint32 texture
- Approximate objects by deformation points and parabolic shape
- Store deformation point height and deformation height
- deformation height = deformation point height + (distance to point)^2 × artist’s scale.

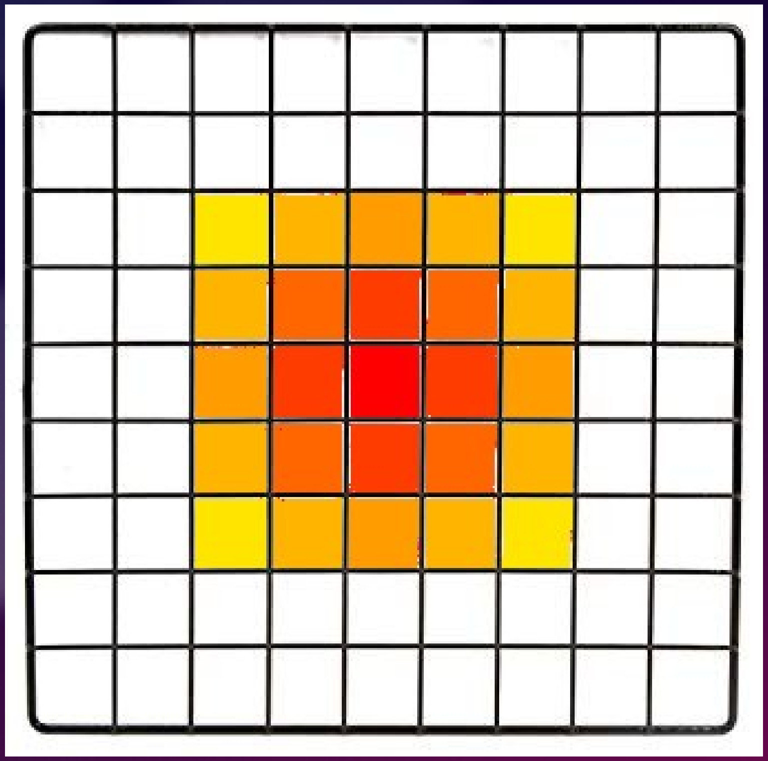
Deferred deformation

- Gather all deformation points
- Write deformation data around the points into the texture
- Compute shader – atomic operations → uint32 texture

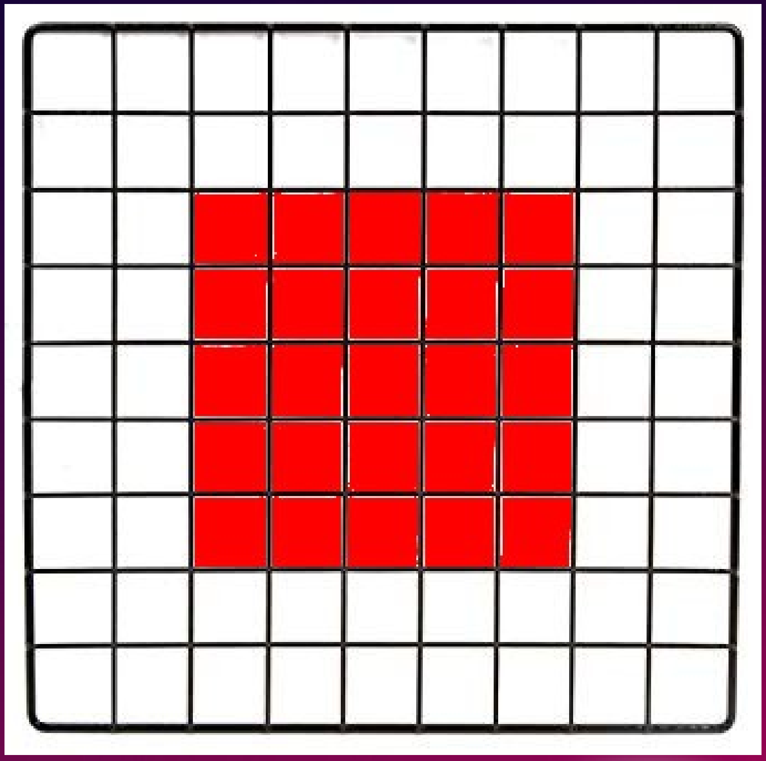


Deferred deformation

Deformation height

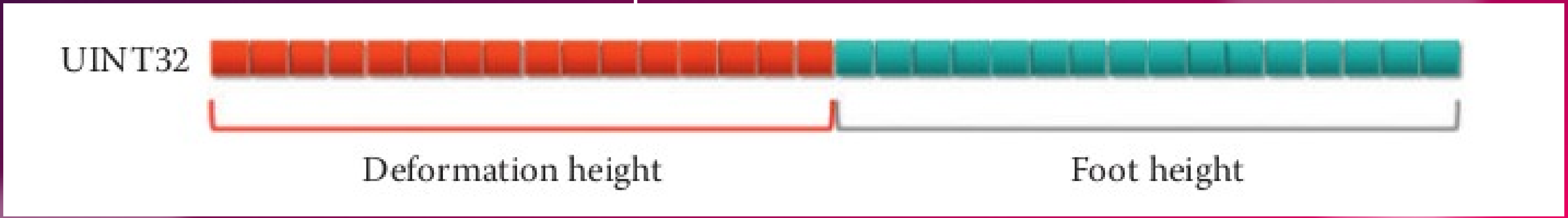


Deformation point height



deformation height = deformation point height + (distance to point)^2 × artist's scale.

Bit allocation in the texture – compatible with atomic minimum

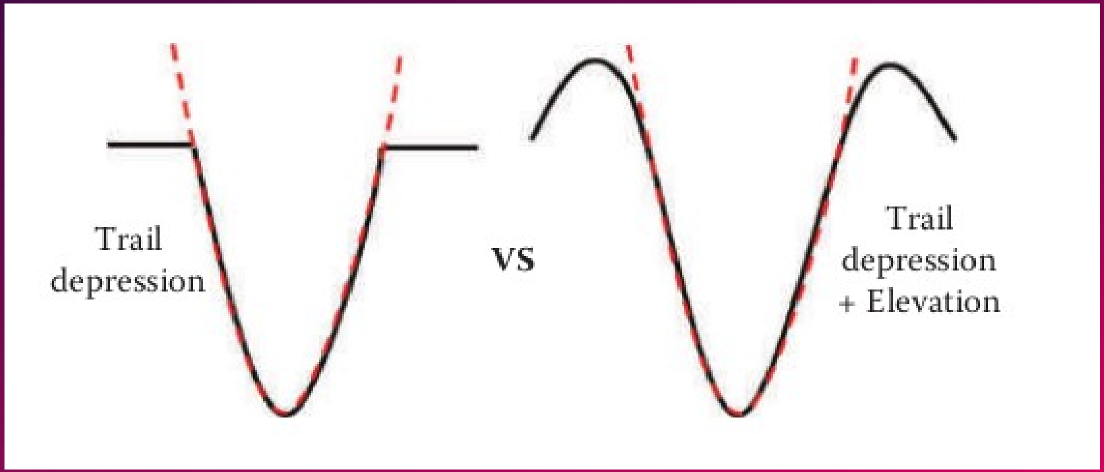


Snow Rendering without elevation

- Sample deformation texture during snow render
- Snow height = min(deformation height, snow height)

Problems:

- Mapping the texture to the world
- Sharp deformation edges on the edge of the texture
- No elevation on the edges of the deformation



Elevation

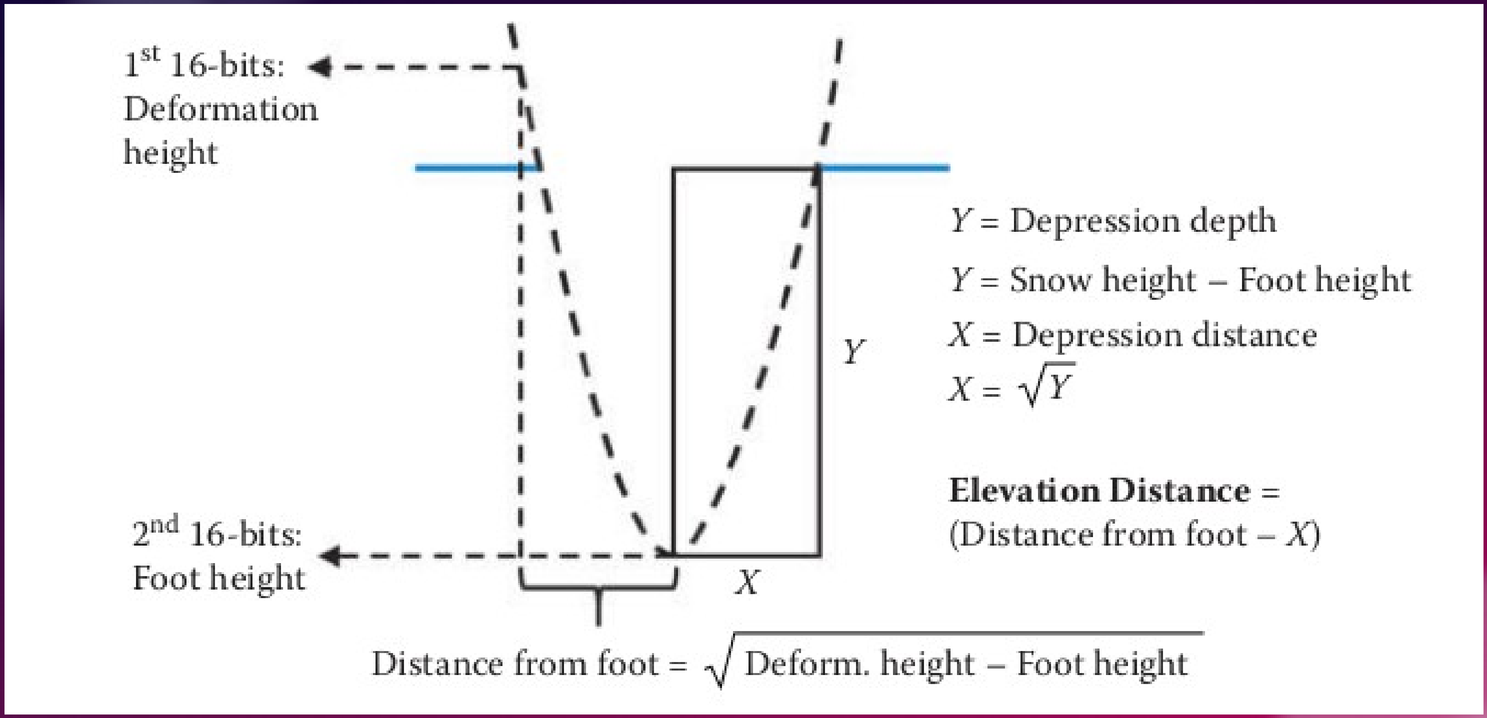
- During snow render pass we have three pieces of information:

  snow height, deformation point height and deformation height

  deformation height = deformation point height + (distance to point)^2 × artist's scale.

Elevation

- deformation height = deformation point height + (distance to point)^2 * artist's scale



deformation height – deformation point height = (distance to point)^2 * artists scale
distance to point = sqrt(deformation height – deformation point height)/sqrt(artists scale)

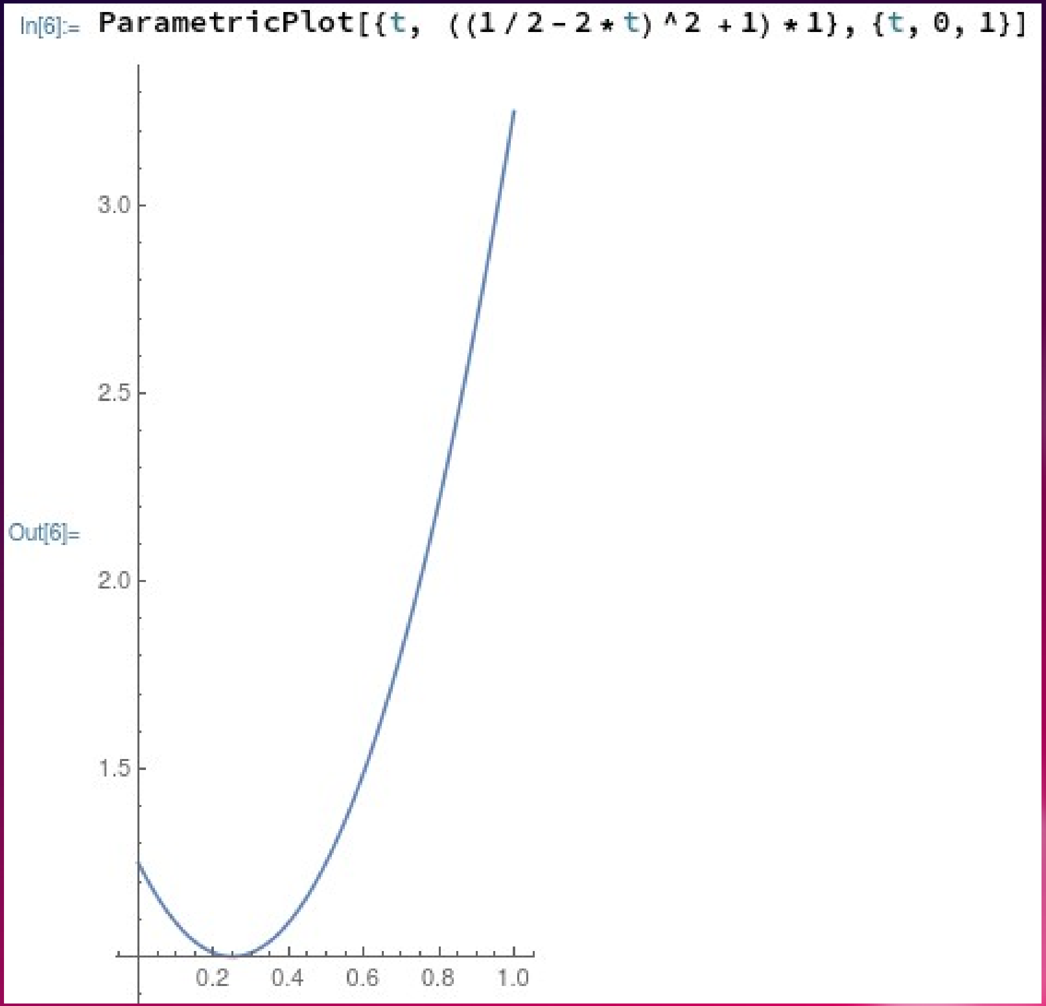
snow height – deformation point height = (depression distance)^2 * artists scale
Depression distance = sqrt(snow height – deformation point height)/sqrt(artists scale)

Elevation

- We computed: depression distance, depression depth and distance to point
- Elevation distance = distance to point – depression distance
- Maximum elevation distance = depression depth * artists scale
- Max elevation height = maximum elevation distance * artists scale
- Ratio = elevation distance / maximum elevation distance
- Elevation = ((0.5-2*ratio)^2+1)*height

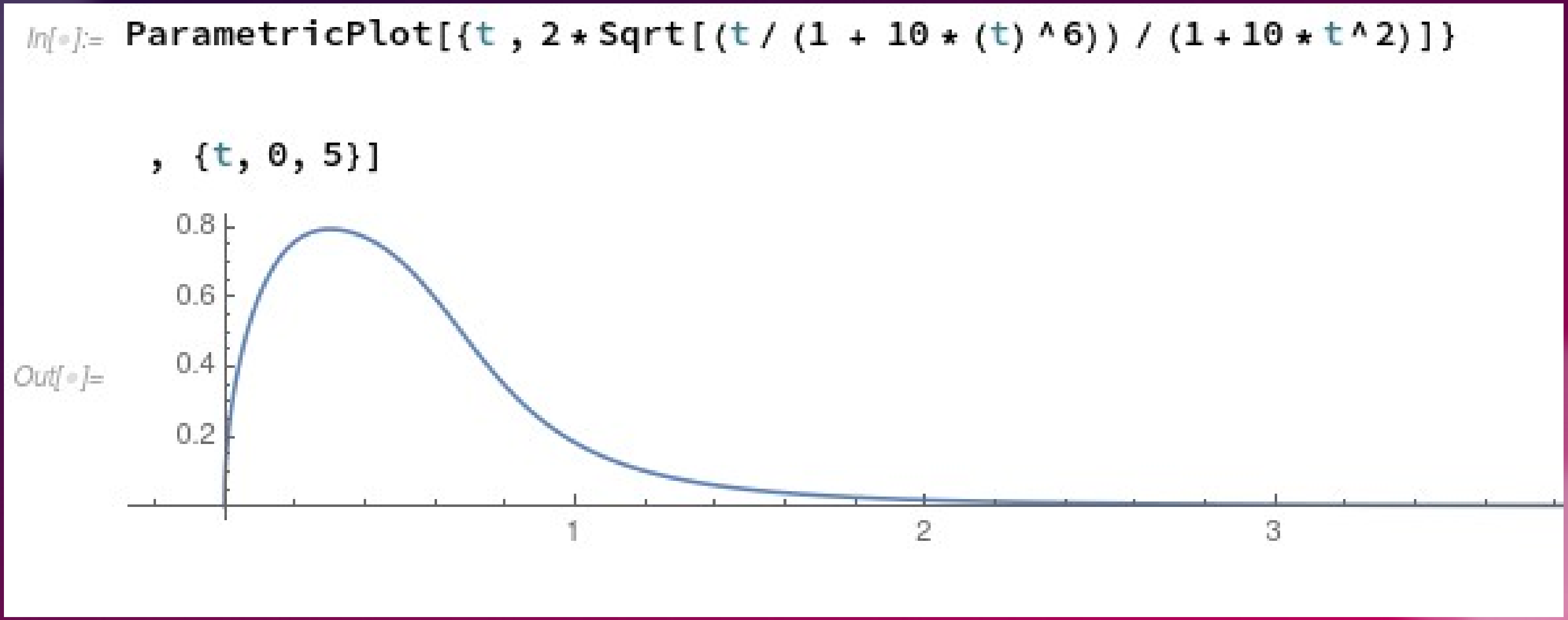
Elevation

- Elevation = ((0.5-2*ratio)^2+1)*height ???
- Ratio is in range
  [0, 1+]

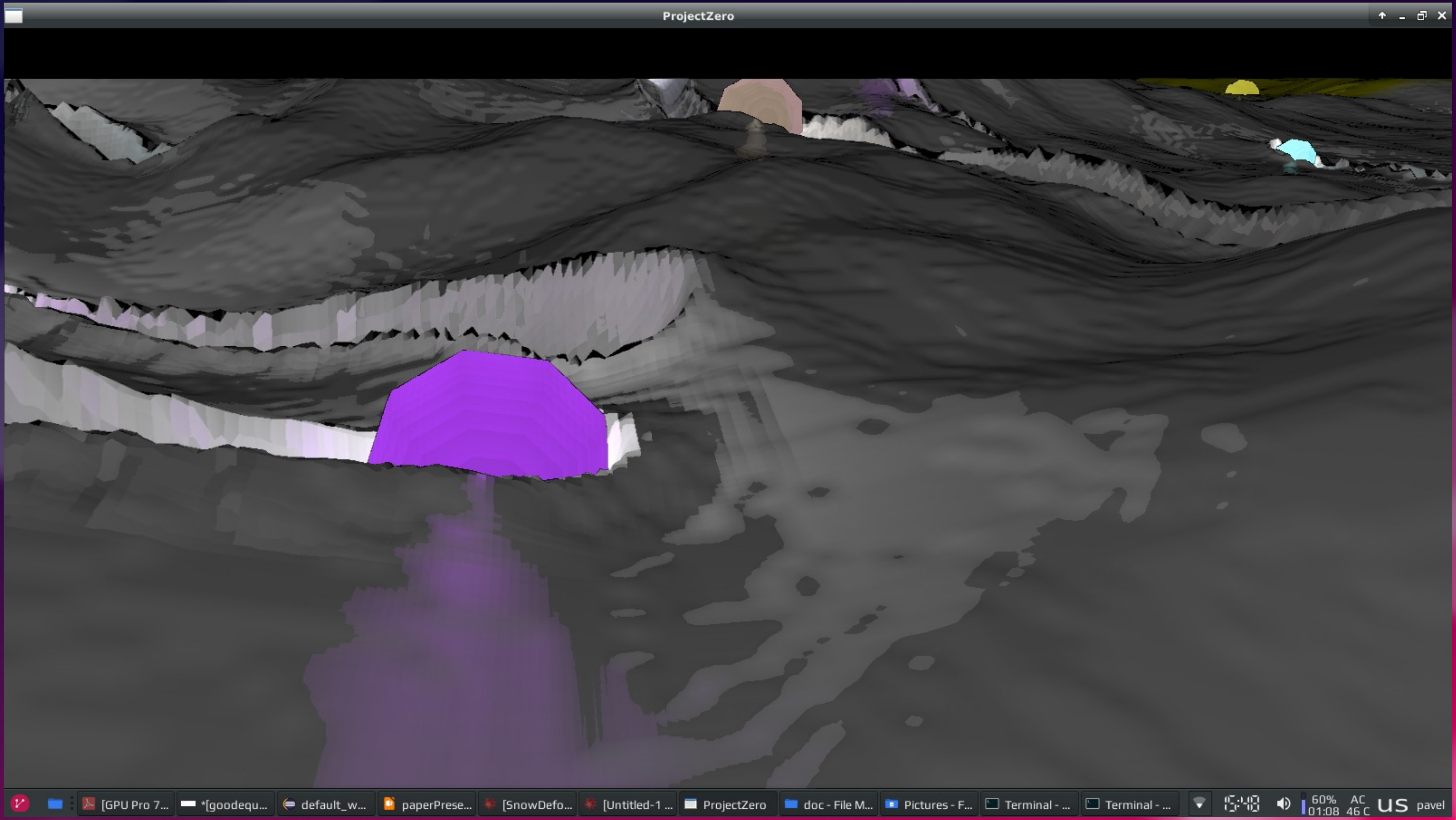


Elevation

- What worked: 2*Sqrt[(t/(1 + 10*t^6)) / (1 + 10*t^2)
- Ratio is in range [0, 1+]

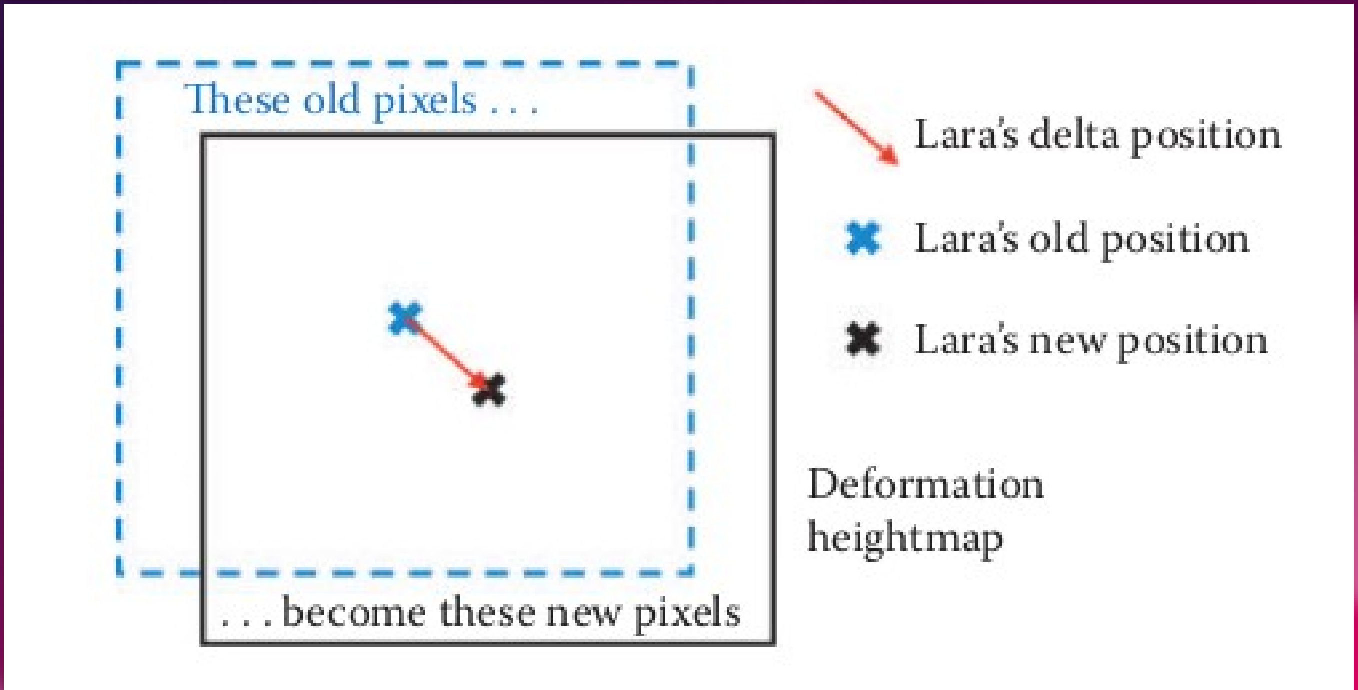


Elevation


ProjectZero
[GPU Pro 7...
*[goodequ...
default_w...
paperPrese...
[SnowDefo...
[Untitled-1 ...
ProjectZero
doc - File M...
Pictures - F...
Terminal - ...
Terminal - ...
60%
01:08
AC
46 C
US
pavel

Mapping the texture to the world

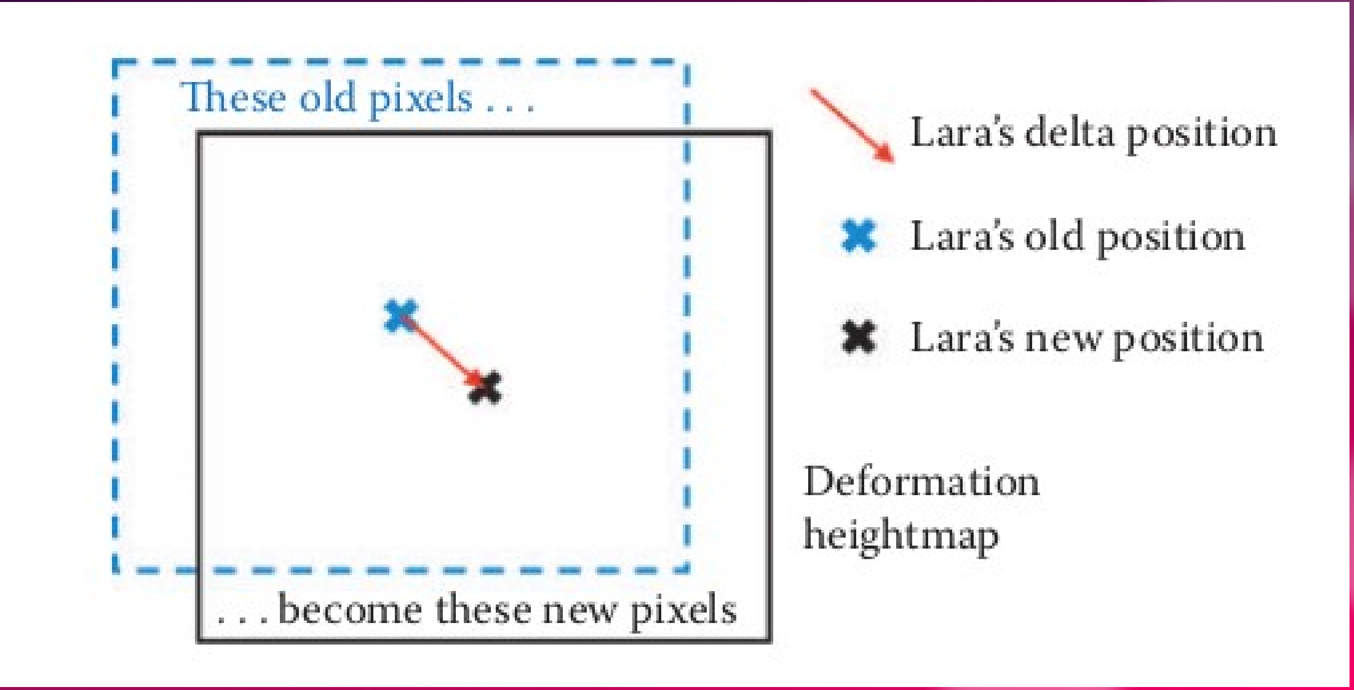
- The deformation texture has 1024*1024 pixels
- Deformations are needed only around the character
- Sliding window approach



Mapping the texture to the world

```
float2 Modulus(float2 WorldPos, float2 TexSize) {
  return WorldPos - (TexSize * floor(WorldPos/TexSize));
}
```

- Modulo access to the texture
- New problem: the deformation values are not reset



Filling the trail

- Options:
- Reset the value to some max value
- Gradually increase the value - how fast?

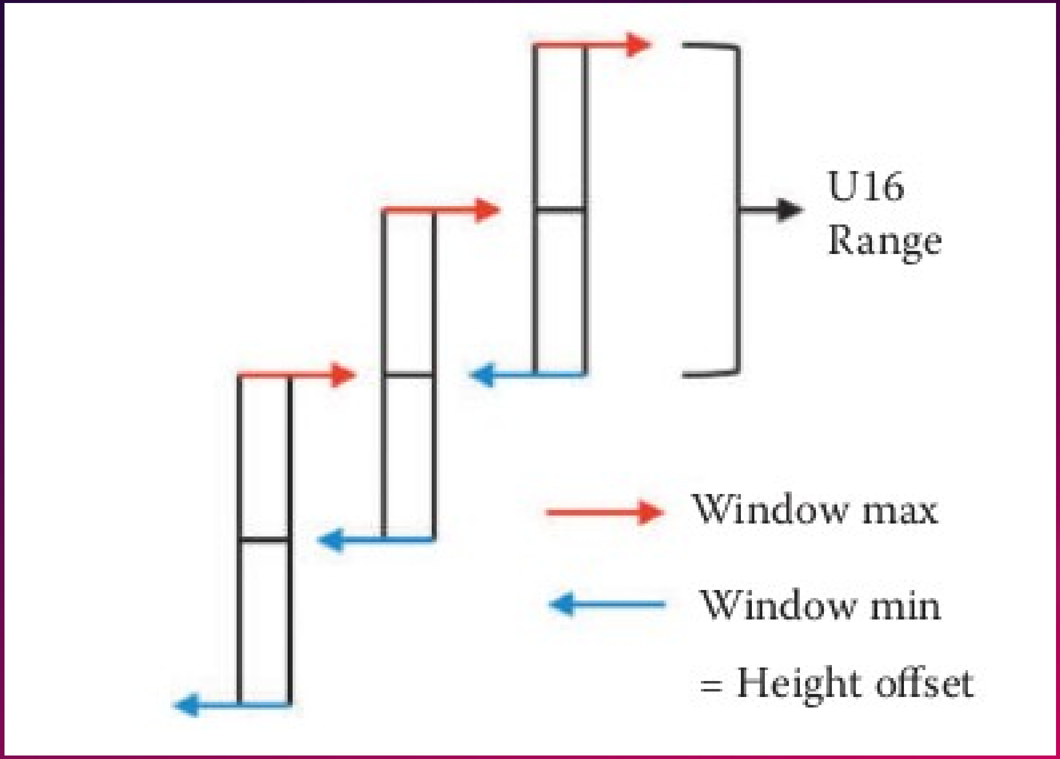
Filling the trail

- Options:
- Reset the value to some max value
- Gradually increase the value - how fast?

- The closer to the edge the faster – smooth filling at the edges
- Suitable for compute shader

Vertical sliding window

- Why? Limited precision of int
- How? When the main character crosses a threshold, move half the range up or down



Summary

- Decoupled geometry and deformation
- Low memory demands
- Produces elevation at edges
- Works with uneven ground (almost)

- complex
- Uneven ground requires small objects
- Hard to balance the precision and scale

Q&A

- https://youtu.be/QSYkwdlDN8s?t=42

- Source: GPU Pro 7: Advanced Rendering Techniques