Water erosion on heightmap terrain

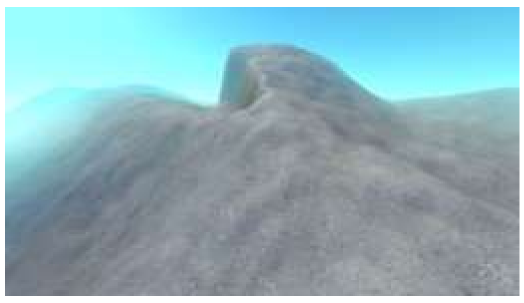
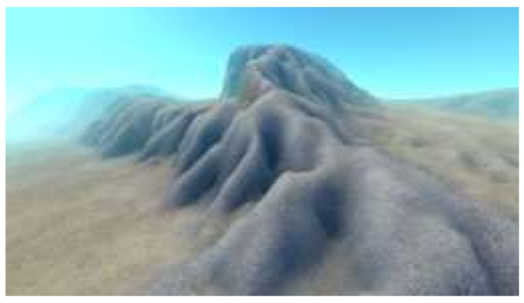
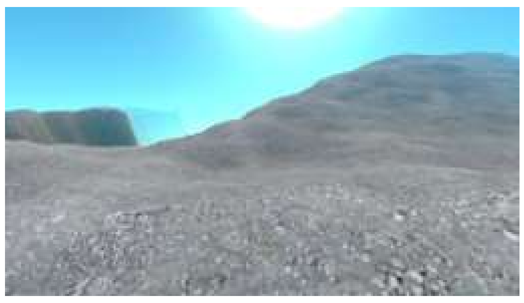
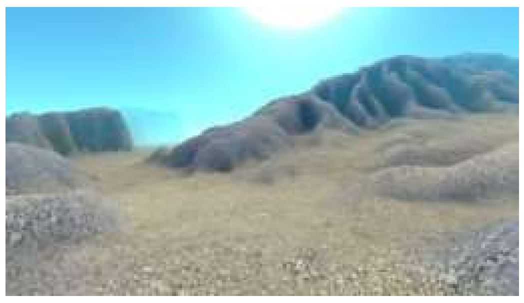
By E-DOG | *Published:* OCTOBER 8, 2011

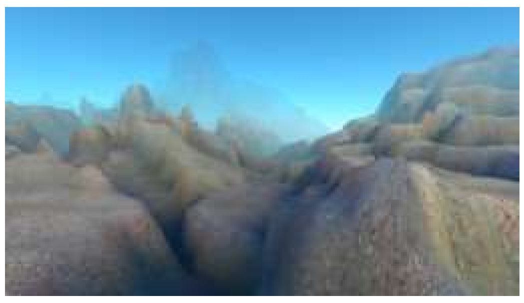
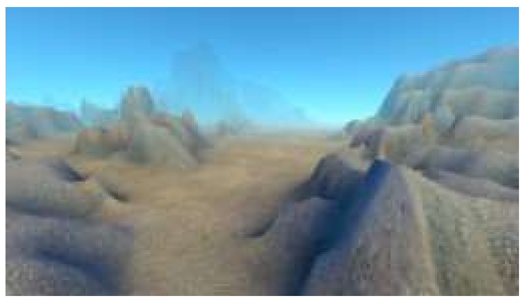
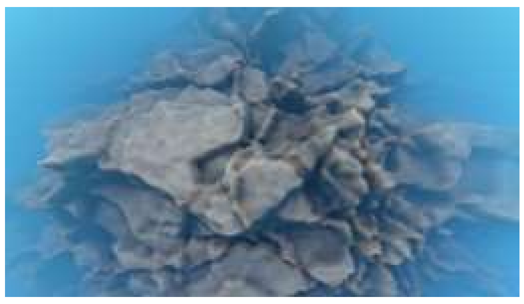
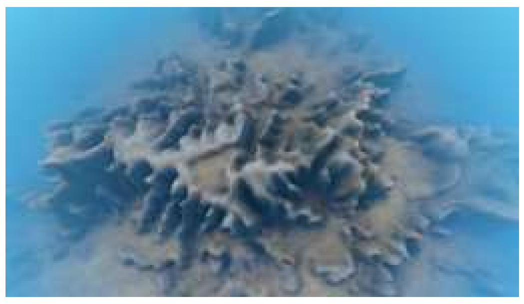
I'm back from vacation, and I decided to write an article as a warm-up exercise.

This article is for those who know what a heightmap is, and who are familiar with some basic heightmap generation. If you aren't one, you can still enjoy the screenshots and skip most of the technobabble here 🙂

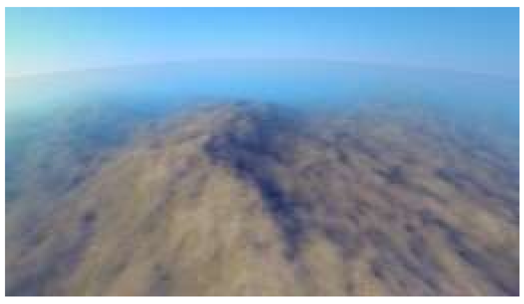
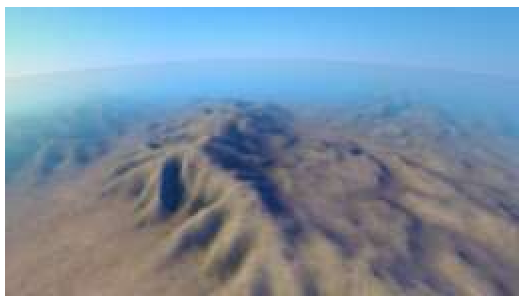
Examples

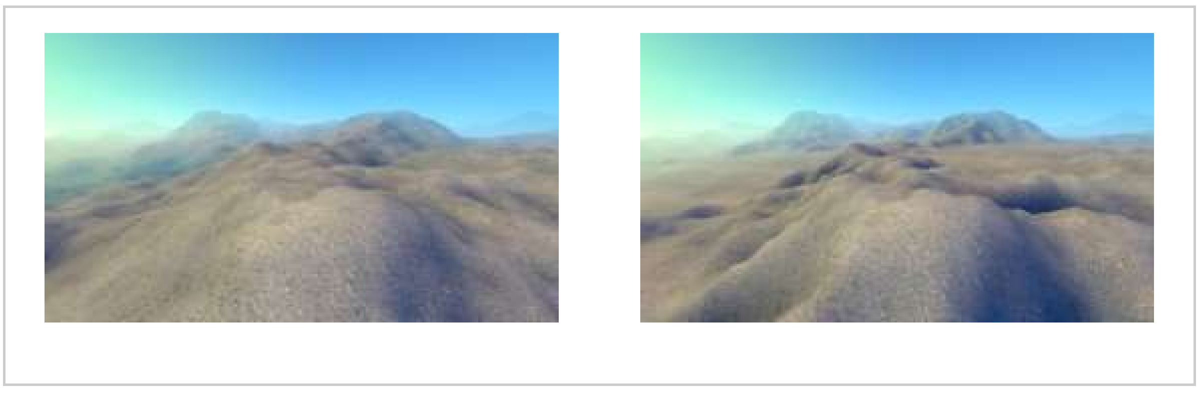
Here is some fault-formed terrain:

before erosion	after erosion
	
	

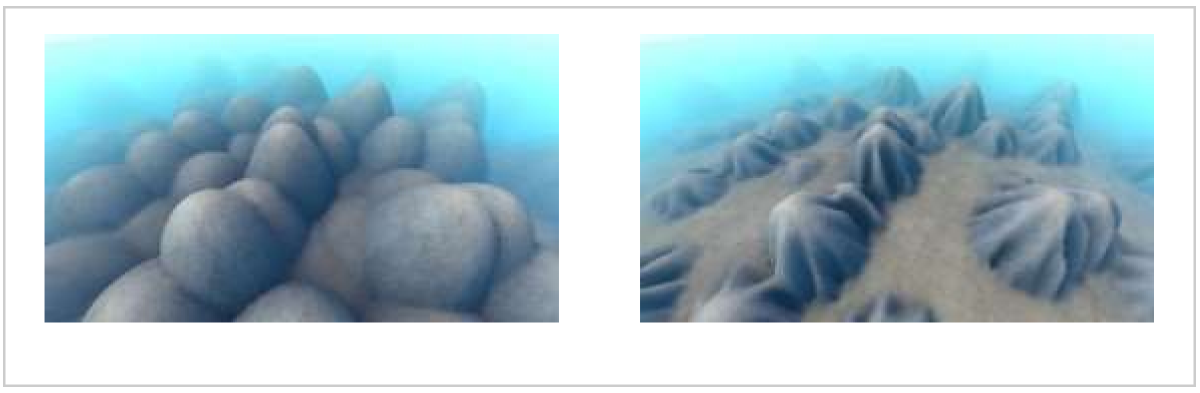
	
	

And here is what my erosion method can do to a simple classic fractal-noise-generated terrain:

before erosion	after erosion
	



It also turns artificial forms into more natural looking ones:



Applications

As you see from the examples above, erosion can make generated terrain look much more interesting and natural. It looks even better when you add grass by slope and/or ground type.
It's very straightforward to integrate since it modifies existing heightmap. It can be applied several times too, both for increased effect or after other terrain modifiers.
It's useful for gameplay as well, since it makes lower areas more flat and thus easier to walk/navigate/build on.

It has some drawbacks of course.
It's a full-terrain method, not suitable for "infinite" generated worlds. It works on maps of limited size.
It can be rather slow on large heightmaps. Especially if you want heavy erosion. So it's probably better as offline method in your editor if your heightmap is large.

Understanding water erosion

The water starts somewhere on the terrain (as rain, water spring, whatever), then flows downhill, dissolving the soil, carrying it, and depositing it as sediment. Eventually it gets to a low point and evaporates.
The trick here is understanding (and carefully implementing) the erosion/deposition process. This involves sediment carry capacity of the flow.
At any time, the water flow can only carry a limited amount of dissolved soil. This amount depends on the surface slope, the speed of the flow and the amount of water.
If less is carried than possible, erosion happens, removing soil from the terrain and adding it to the flow.
If more is carried, deposition happens, dropping extra carried soil as sediment.
As you can see, the process can switch fast between erosion and deposition as the flow accelerates, or drops down a steep slope, or comes to a flat area.
The formula I use:
`q=max(slope, minSlope)*v*w*Kq`
Where *slope* is just a tangent, *minSlope* and *Kq* are constant parameters, *v* is flow speed and *w* is amount of water in the flow.

Droplets

I use droplet model to simulate water erosion. That is, I pick a starting point on the terrain, start with zero velocity and carried soil, and proceed moving this point, changing its dynamic variables and the terrain in the process, until all its water evaporates or it flows to a pit it can't get out of.
Some tricky things here are:

- I use height gradient as the downhill direction. If the gradient is zero, I pick a random direction. I also add some inertia just for fun.
- I then sample next height in the downhill direction. If it happens to be higher than the current one, we're in the pit and should either drop sediment to fill it and flow on, or die trying. When the flow goes on after filling the pit, its speed is reset to zero.
- When erosion happens, don't remove more than the height difference (between current and next position). That is, don't dig a pit in the place the water flows from, cut it flat at most. Digging a pit makes the process unstable, creating nasty spikes.
- I deposit to heightmap using bilinear weights. This way it fills pits better while being somewhat smooth.
- I erode the heightmap with a bell-like "brush", to make smoother (yet somewhat wider) channels. That's to make the final result look better too.
- A single droplet effect on the terrain is hardly noticeable (unless you use some crazy parameters). Use at least a thousand droplets for testing. Their power is in numbers.
- Erosion and deposition speed is affected by special parameters. So only some (not all) extra soil is dropped when carry capacity is exceeded, for example.
- The flow speed is accelerated by height delta on each step, and some water is evaporated as well.

Droplet sources

I use "rain" erosion in the examples, starting droplets randomly all over the terrain.
However, other effects can be achieved with non-uniform droplet distribution, like creating streams from sources, or raining more/less in certain areas.

Code sample

Here is the code “as is”. It won’t compile since it uses some external stuff, but I hope it’s easy enough to figure it out.
You can do with it whatever you want.

```
void HeightmapData::genDropletErosion(unsigned iterations, ErosionParams &params)
{
  float Kq=params.Kq, Kw=params.Kw, Kr=params.Kr, Kd=params.Kd, Ki=params.Ki,
    minSlope=params.minSlope, Kg=params.g*2;

  TempData<Point2[HMAP_SIZE*HMAP_SIZE]> erosion;

  float flt=0;
  erosion.fillMem32(*(uint32_t*)&flt);

  static const unsigned MAX_PATH_LEN=HMAP_SIZE*4;

  int64_t t0=get_ref_time();

  #define DEPOSIT_AT(X, Z, W) \
  { \
    float delta=ds*(W); \
    erosion[HMAP_INDEX((X), (Z))].y+=delta; \
    hmap   [HMAP_INDEX((X), (Z))]  +=delta; \
    params.deposit(scolor, surface[HMAP_INDEX((X), (Z))], delta); \
  }

  #if 1
    #define DEPOSIT(H) \
      DEPOSIT_AT(xi  , zi  , (1-xf)*(1-zf)) \
      DEPOSIT_AT(xi+1, zi  ,    xf *(1-zf)) \
      DEPOSIT_AT(xi  , zi+1, (1-xf)*   zf ) \
      DEPOSIT_AT(xi+1, zi+1,    xf *   zf ) \
      (H)+=ds;
  #else
    #define DEPOSIT(H) \
      DEPOSIT_AT(xi  , zi  , 0.25f) \
      DEPOSIT_AT(xi+1, zi  , 0.25f) \
      DEPOSIT_AT(xi  , zi+1, 0.25f) \
      DEPOSIT_AT(xi+1, zi+1, 0.25f) \
      (H)+=ds;
  #endif

  uint64_t longPaths=0, randomDirs=0, sumLen=0;

  for (unsigned iter=0; iter<iterations; ++iter)
  {
    if ((iter&0x3FFF)==0 && iter!=0) show_splash("Calculating erosion", (iter+0.5f)/iterations);

    int xi=game_rnd()&(HMAP_SIZE-1);
    int zi=game_rnd()&(HMAP_SIZE-1);

    float xp=xi, zp=zi;
    float xf= 0, zf= 0;

    float h=HMAP(xi, zi);
    float s=0, v=0, w=1;
    vec4f scolor=zero4f();

    float h00=h;
    float h10=HMAP(xi+1, zi  );
    float h01=HMAP(xi  , zi+1);
    float h11=HMAP(xi+1, zi+1);

    float dx=0, dz=0;

    unsigned numMoves=0;
    for (; numMoves<MAX_PATH_LEN; ++numMoves)
    {
      // calc gradient
      float gx=h00+h01-h10-h11;
      float gz=h00+h10-h01-h11;
      //== better interpolated gradient?

      // calc next pos
      dx=(dx-gx)*Ki+gx;
      dz=(dz-gz)*Ki+gz;

      float dl=sqrtf(dx*dx+dz*dz);
      if (dl<=FLT_EPSILON)
      {
        // pick random dir
        float a=frnd()*TWOPI;
        dx=cosf(a);
        dz=sinf(a);
        ++randomDirs;
      }
      else
      {
        dx/=dl;
        dz/=dl;
      }

      float nxp=xp+dx;
      float nzp=zp+dz;

      // sample next height
      int nxi=intfloorf(nxp);
      int nzi=intfloorf(nzp);
      float nxf=nxp-nxi;
      float nzf=nzp-nzi;

      float nh00=HMAP(nxi  , nzi  );
      float nh10=HMAP(nxi+1, nzi  );
      float nh01=HMAP(nxi  , nzi+1);
      float nh11=HMAP(nxi+1, nzi+1);

      float nh=(nh00*(1-nxf)+nh10*nxf)*(1-nzf)+(nh01*(1-nxf)+nh11*nxf)*nzf;

      // if higher than current, try to deposit sediment up to neighbour height
      if (nh>=h)
      {
        float ds=(nh-h)+0.001f;

        if (ds>=s)
        {
          // deposit all sediment and stop
          ds=s;
          DEPOSIT(h)
          s=0;
          break;
        }

        DEPOSIT(h)
        s-=ds;
        v=0;
      }

      // compute transport capacity
      float dh=h-nh;
      float slope=dh;
      //float slope=dh/sqrtf(dh*dh+1);

      float q=maxval(slope, minSlope)*v*w*Kq;

      // deposit/erode (don't erode more than dh)
      float ds=s-q;
      if (ds>=0)
      {
```

```
    // deposit
    ds*=Kd;
    //ds=minval(ds, 1.0f);

    DEPOSIT(dh)
    s-=ds;
  }
  else
  {
    // erode
    ds*=-Kr;
    ds=minval(ds, dh*0.99f);

    #define ERODE(X, Z, W) \
    { \
      float delta=ds*(W); \
      hmap             [HMAP_INDEX((X), (Z))]-=delta; \
      Point2 &e=erosion[HMAP_INDEX((X), (Z))]; \
      float r=e.x, d=e.y; \
      if (delta<=d) d-=delta; \
      else { r+=delta-d; d=0; } \
      e.x=r; e.y=d; \
      scolor=params.erode(scolor, surface[HMAP_INDEX((X), (Z))], s, delta); \
    }

    #if 1
      for (int z=zi-1; z<=zi+2; ++z)
      {
        float zo=z-zp;
        float zo2=zo*zo;

        for (int x=xi-1; x<=xi+2; ++x)
        {
          float xo=x-xp;

          float w=1-(xo*xo+zo2)*0.25f;
          if (w<=0) continue;
          w*=0.1591549430918953f;

          ERODE(x, z, w)
        }
      }
    #else
      ERODE(xi  , zi  , (1-xf)*(1-zf))
      ERODE(xi+1, zi  ,    xf *(1-zf))
      ERODE(xi  , zi+1, (1-xf)*   zf )
      ERODE(xi+1, zi+1,    xf *   zf )
    #endif

    dh-=ds;

    #undef ERODE

    s+=ds;
  }

  // move to the neighbour
  v=sqrtf(v*v+Kg*dh);
  w*=1-Kw;

  xp=nxp; zp=nzp;
  xi=nxi; zi=nzi;
  xf=nxf; zf=nzf;

  h=nh;
  h00=nh00;
  h10=nh10;
  h01=nh01;
  h11=nh11;
}
```

```
    if (numMoves>=MAX_PATH_LEN)
    {
      debug("droplet #%d path is too long!", iter);
      ++longPaths;
    }

    sumLen+=numMoves;
  }

  #undef DEPOSIT
  #undef DEPOSIT_AT

  int64_t t1=get_ref_time();
  debug("computed %7d erosion droplets in %6u ms, %.0f droplets/s",
    iterations, get_time_msec(t1-t0), double(iterations)/get_time_sec(t1-t0));
  debug("  %.2f average path length, %I64u long paths cut, %I64u random directions picked",
    double(sumLen)/iterations, longPaths, randomDirs);
}
```

About parameters

Here are the default parameters I use:

```
ErosionParams()
{
  Kq=10; Kw=0.001f; Kr=0.9f; Kd=0.02f; Ki=0.1f; minSlope=0.05f; g=20;
}
```

Kq and minSlope are for soil carry capacity (see the formula above).
Kw is water evaporation speed.
Kr is erosion speed (how fast the soil is removed).
Kd is deposition speed (how fast the extra sediment is dropped).
Ki is direction inertia. Higher values make channel turns smoother.
g is gravity that accelerates the flows.

Other simulation methods

Droplet model is not the only one. The same flow process can be simulated with iterations on the grid, for example. However I find droplets results more interesting.
Also, there are methods to fake the erosion result. The fake isn't as good as a "real" one, but it can be much faster. See for example "Realtime Procedural Terrain Generation" by Jacob Olsen.

Other erosion types

This article covers some water erosion only. There are other erosion types though. Rivers can do some interesting things to the terrain, for one.
Coastal erosion is completely different, as well as glacial and thermal ones.
Wind erosion can be done with droplet-like method, but is very different in nature too.

This entry was posted in Arcane Worlds and tagged Arcane Worlds, article, code, dev, screenshots. Bookmark the permalink. Both comments and trackbacks are currently closed.