PROJECT

Report

Apr 20 2023
CS 805 Computer Graphic

Hoon Seok Kim
200452816

Perlin Noise

This project shows several examples how to apply Perlin noise to the image. Perlin noise function was added to Assignment 2 codes (*mymodel.h* and *proj.cpp*) by adding separate '*noise.h*' file and '*noise.cpp*' file. Perlin noise script was modified from the one used for generating random 3D object before. Perlin noise function generate different random values depending on seed values and grid size. The grid size was fixed by 1 since we are using grid size 1 for the image.

Perlin noise was applied as follows.

1. Plane (projected view) => Perlin noise to shape => Hand drawn square
2. Plane (projected view) => Perlin noise to shading => Patterned square
3. Plane (projected view) => Large Perlin noise to shape => Random island map generator
4. Plane in 3D space => Perlin noise to shape and shading => Terrain
5. Sphere => Perlin noise to shape => Bumpy sphere
6. Sphere => Perlin noise to shape, outline and shading => Melting sphere
7. Sphere + Plane => Perlin noise to everywhere => Melting ice cream

Two source files, two header files: Proj.cpp is the main .cpp

1. Projected View Plane: Hand drawn square (Change straight line to natural drawn line)

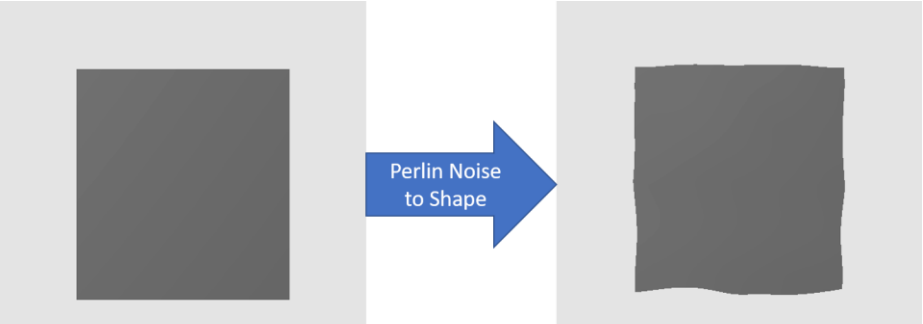

Figure 1 Create hand drawn square image by Perlin Noise

Seed values of random numbers for this simulation.

```
const unsigned int SEED_X1 = 1273472206;
const unsigned int SEED_X2 = 4278162623;
const unsigned int SEED_Y1 = 1440014778;
const unsigned int SEED_Y2 = 2524485263;
const unsigned int SEED_Z1 = 2813546167;
const unsigned int SEED_Z2 = 3305132234;
const unsigned int SEED_Q0 = 1498573726;
const unsigned int SEED_Q1 = 3476519523;
const unsigned int SEED_Q2 = 3905844518;
```

P = Po + t*V.
't' was distorted. Perlin noise was split into 3 directions since it is vector.
However, it is not a vector. Thus, magnitude of Perlin noise was used for distorting 't'.

```
t = t * (1.0 + sqrt(noisex*noisex + noisey*noisey + noisez*noisez)/12.0);
```

2. Projected View Plane: Bumpy plane (Add noise to shading)

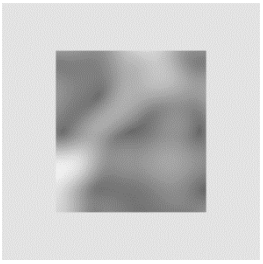

Figure 2 Noise to shading

Seed values of random numbers for this simulation.

```
// Seed setup for random value. Changin this results different result
const unsigned int SEED_X1 = 1273472206;
const unsigned int SEED_X2 = 1278162623;
const unsigned int SEED_Y1 = 1440014778;
const unsigned int SEED_Y2 = 1524485263;
const unsigned int SEED_Z1 = 1813546167;
const unsigned int SEED_Z2 = 1305132234;
const unsigned int SEED_Q0 = 1498573726;
const unsigned int SEED_Q1 = 1476519523;
const unsigned int SEED_Q2 = 1905844518;
```

Add Perlin noise to Normal Vectors for shading variation to each direction.

```
// N[0] => normal vector to 'x' direction, N[1] => normal vector to 'y' direction, N[2] => normal vector to 'z' direction
// N[0], N[1], N[2] was normalized again.
N[0] = N[0] * (1.0f + 2.0*noisex);
N[1] = N[1] * (1.0f + 2.0*noisey);
N[2] = N[2] * (1.0f + 2.0*noisez);
```

Additional Perlin noise to shading for more variation.

```
C = (unsigned char)(unsigned int) (C * (1.0 + sqrt(noisex * noisex + noisey * noisey + noisez * noisez)/0.5));
```

3. Projected View Plane: Random map generation (add large noise to ‘t’)

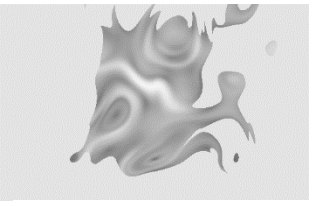

Figure 3 Large noise to shape

```
// Seed setup for random value. Changin this results different result
const unsigned int SEED_X1 = 273472206;
const unsigned int SEED_X2 = 278162623;
const unsigned int SEED_Y1 = 440014778;
const unsigned int SEED_Y2 = 524485263;
const unsigned int SEED_Z1 = 813546167;
const unsigned int SEED_Z2 = 305132234;
const unsigned int SEED_Q0 = 498573726;
const unsigned int SEED_Q1 = 476519523;
const unsigned int SEED_Q2 = 905844518;
```

Apply different seed values

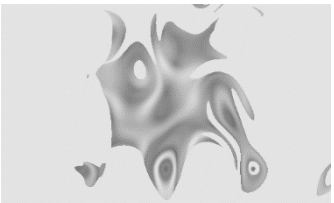

Figure 4 Map by different seed

```
// Seed setup for random value. Changin this results different result
const unsigned int SEED_X1 = 4273472206;
const unsigned int SEED_X2 = 4278162623;
const unsigned int SEED_Y1 = 4440014778;
const unsigned int SEED_Y2 = 4524485263;
const unsigned int SEED_Z1 = 4813546167;
const unsigned int SEED_Z2 = 4305132234;
const unsigned int SEED_Q0 = 3498573726;
const unsigned int SEED_Q1 = 3476519523;
const unsigned int SEED_Q2 = 3905844518;
```

Extra: Adding excessive noise to shading

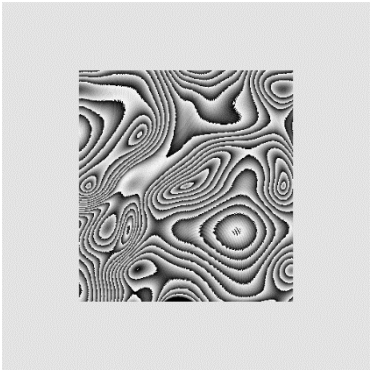

Figure 5 Massive noise to shading

4. Plane in 3D space: Generating terrain (move plane coordinates)

Add Perlin noise to shape and shading, creates wrinkled paper.

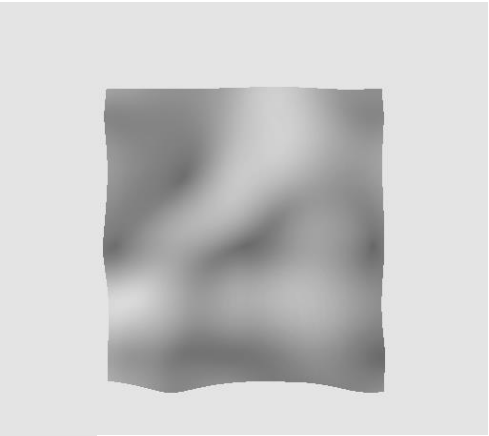

Figure 6 Wrinkled paper

Translate and rotate the plane so it looks like existing in 3D space

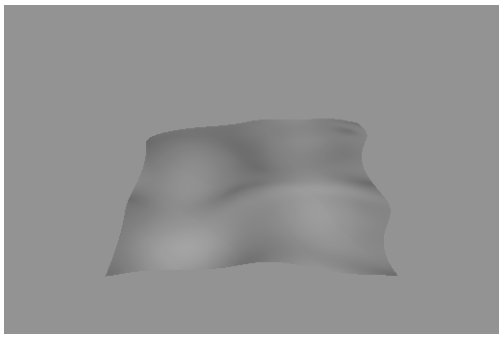

Figure 7 Terrain

```
POLY4 obj2 = {     0.0, 1.0, 0.0,            /* v0 */
                   0.0, 1.0, 2.0,            /* v1 */
                   2.0, 1.0, 2.0,            /* v2 */
                   2.0, 1.0, 0.0,            /* v3 */
                   0.0, 1.0, 0.0,            /* normal of the polygon */
                   0.8f };                   /* diffuse reflection coefficient */
```

Reduce noise of N from 2.0 to 1.2, and noise to ‘t’ and intersection point for distorting shape

```
N[0] = N[0] * (1.0f + 1.2*noisex);
N[1] = N[1] * (1.0f + 1.2*noisey);
N[2] = N[2] * (1.0f + 1.2*noisez);

t = t *(1.0 + sqrt(noisex * noisex + noisey * noisey + noisez * noisez) / 0.5);
float rayPoint[3]={(P0[0]+t*V[0])*(1.0+1.0*noisex), (P0[1]+t*V[1])*(1.0+1.0*noisey), (P0[2]+t*V[2])*(1.0+1.0*noisez)};
```

Change shading noise

```
C = (unsigned char)(unsigned int) (C * (1.0 + sqrt(noisex * noisex + noisey * noisey + noisez * noisez)/0.6));
```

5. Sphere: Looks bumpy Sphere (distorting ‘t’)

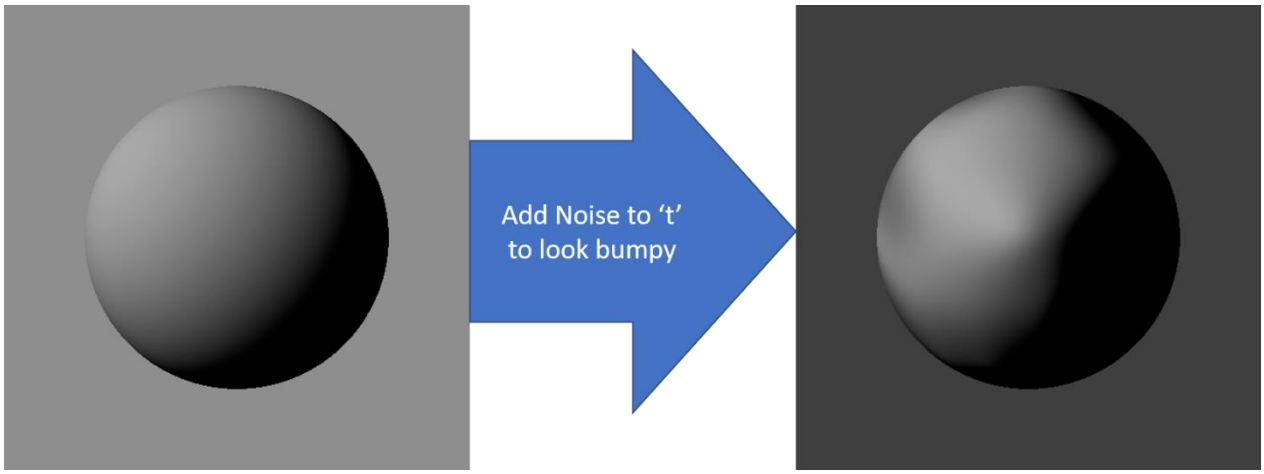

Figure 8 Make bumpy sphere by adding Perlin noise to sphere

```
float LRP[3] = { -10.0, 12.0, 7.0 };             /* light position */

t = t * (1.0 + 0.35 * sqrt(noisex * noisex + noisey * noisey + noisez * noisez));
```

Bumpier by adding noise to normal vectors of sphere and more noise to ‘t’

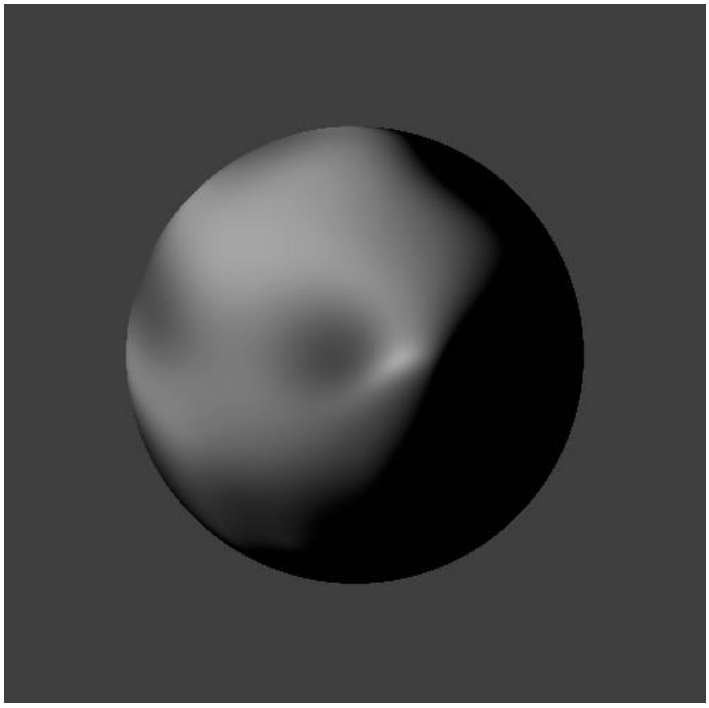

Figure 9 Bumpier sphere

```
t = t * (1.0 + 0.5 * sqrt(noisex * noisex + noisey * noisey + noisez * noisez));

N[0] = ((P0[0] + t * V[0]) * (1.0 + 0.5*noisex) - obj1.x);
N[1] = ((P0[1] + t * V[1]) * (1.0 + 0.5*noisey) - obj1.y);
N[2] = ((P0[2] + t * V[2]) * (1.0 + 0.5*noisez) - obj1.z);
```

6. Melting Sphere: Distort even outline of sphere and Shading

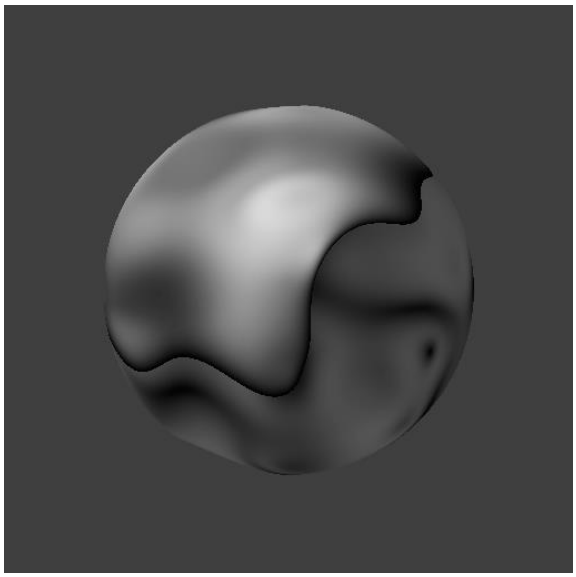

Figure 10 Melting sphere

Seed value setup and light position

```
// Seed setup for random value. Changin this results different result
const unsigned int SEED_X1 = 2273472206;
const unsigned int SEED_X2 = 2278162623;
const unsigned int SEED_Y1 = 1440014778;
const unsigned int SEED_Y2 = 1524485263;
const unsigned int SEED_Z1 = 2813546167;
const unsigned int SEED_Z2 = 2305132234;
const unsigned int SEED_Q0 = 3498573726;
const unsigned int SEED_Q1 = 3476519523;
const unsigned int SEED_Q2 = 3905844518;

float LRP[3] = { -7.0, 16.0, 7.0 };             /* light position */
```

Distort radius of sphere for even distorting outline of sphere: rr is radius * radius

```
rr = ((P0[0] + t * V[0]) * (1.0 + 0.45 * noisex) - obj1.x) * ((P0[0] + t * V[0]) * (1.0 + 0.45 * noisex) - obj1.x)
+ ((P0[1] + t * V[1]) * (1.0 + 0.45 * noisey) - obj1.y) * ((P0[1] + t * V[1]) * (1.0 + 0.45 * noisey) - obj1.y)
+ ((P0[2] + t * V[2]) * (1.0 + 0.45 * noisez) - obj1.z) * ((P0[2] + t * V[2]) * (1.0 + 0.45 * noisez) - obj1.z);
// re-estimate thc
thc = sqrt(rr - dd);
// If r is lager than or equal with d, it meet
if ( dd <= rr) {
       // re-estimate t
       t = tca - thc;
       // distort t
       t = t * (1.0+ 0.7*sqrt( noisex*noisex + noisey*noisey + noisez*noisez));

       // Additional noise to normal vector
       N[0] = ((P0[0] + t * V[0]) * (1.0 + 0.55*noisex) - obj1.x);
       N[1] = ((P0[1] + t * V[1]) * (1.0 + 0.55*noisey) - obj1.y);
       N[2] = ((P0[2] + t * V[2]) * (1.0 + 0.55*noisez) - obj1.z);
```

Distort shading value even for the dark side for looking like ice cream

```
// Add noise to C(shading)
if (Ctemp > 1.0) {
        C = (unsigned char)(unsigned int)(Ip * kd * (N[0] * L[0] + N[1] * L[1] + N[2] * L[2]));
        C = (unsigned char)(unsigned int) (C * (1.0 + sqrt(noisex * noisex + noisey * noisey + noisez * noisez)/1.0)); }
// if angle bet. light and normal is greater than or equal with 90.
// force total black.
else {
// Add noise even to black part
       C = (unsigned char) 1;
        C = (unsigned char)(unsigned int)(C * (1.0 + sqrt(noisex * noisex + noisey * noisey + noisez * noisez) / 0.01)); }
```

7. Final Result: Melting ice cream (Meltint sphere + Terrain)

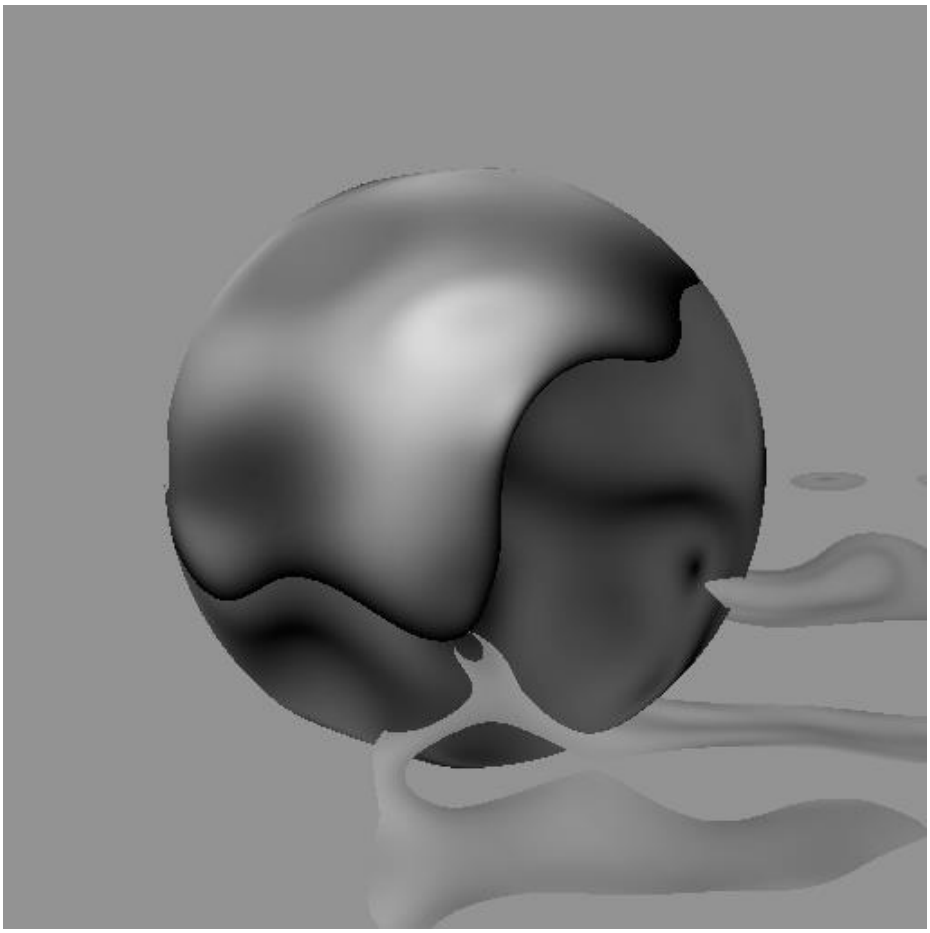

Figure 11 Melting ice cream

Submitted C++ codes reflect this result.