void MULTIVIS\_RENDER::OriginalSamples(float4\* Points, PointProperty\* Property, int\* SampleNum)

{

int i, j, k, offset, SampleCellNumber,SampleCount;

int dx, dy, dz, TempOffset, flag[7], tt, flagValid, CellPointNumber;

float x, y, z, PoissionMean;

float4\* SampleCellIndex;

SampleCellIndex = (float4\*)malloc(sizeof(float4)\*GRIDWIDTH\*GRIDHIGHT\*GRIDDEPTH);

int CellSize;

CellSize = CUBESIZE / GRIDWIDTH;

SampleCellNumber = 0;

for (k = 0; k < (GRIDDEPTH-1); k++)

for (j = 0; j<(GRIDHIGHT-1); j++)

for (i = 0; i < (GRIDWIDTH-1); i++)

{

offset = k \* GRIDWIDTH \* GRIDHIGHT + j \* GRIDWIDTH + i;

x = i \* CellSize - GRIDWIDTH \* CellSize / 2;

y = j \* CellSize - GRIDHIGHT \* CellSize / 2;

z = k \* CellSize \* (-1) + GRIDDEPTH \* CellSize / 2;

//test if this cell include the iso-value

tt = 0;

flagValid = 1;

for (dz = 0; dz<2; dz++)

for (dy = 0; dy<2; dy++)

for (dx = 0; dx < 2; dx++)

{

if (MinorRadius < MajorRadius)

{

if (GridValue[offset] == -1)

flagValid = 0;

TempOffset = (k + dz) \* GRIDWIDTH \* GRIDHIGHT + (j + dy) \* GRIDWIDTH + i + dx;

if (TempOffset != offset)

{

if ((GridValue[offset] < IsoValue) && (GridValue[TempOffset] < IsoValue) || (GridValue[offset] > IsoValue) && (GridValue[TempOffset] > IsoValue))

flag[tt] = 0;

else

{

flag[tt] = 1;

}

if (GridValue[TempOffset] == -1)

{

flagValid = 0;

}

tt++;

}

}

else

{

TempOffset = (k + dz) \* GRIDWIDTH \* GRIDHIGHT + (j + dy) \* GRIDWIDTH + i + dx;

if (TempOffset != offset)

{

if (GridValue[offset] \* GridValue[TempOffset] < 0)

flag[tt] = 1;

else flag[tt] = 0;

tt++;

}

}

}

if ((flag[0]|| flag[1] || flag[2] || flag[3] || flag[4] || flag[5] || flag[6]) && (flagValid != 0))

{

SampleCellIndex[SampleCellNumber] = make\_float4(x, y, z, 1);

SampleCellNumber++;

}

}

PoissionMean = (float)SampleNumber / (float)SampleCellNumber;

std::default\_random\_engine generator;

std::poisson\_distribution<int> distribution(PoissionMean);

SampleCount = 0;

for (offset = 0; offset < SampleCellNumber; offset++)

{

CellPointNumber = distribution(generator);

for (i = 0; i < CellPointNumber; i++)

{

x = floor(SampleCellIndex[offset].x) + urand();

y = floor(SampleCellIndex[offset].y) + urand();

z = floor(SampleCellIndex[offset].z) + urand();

Points[SampleCount]= make\_float4(x, y, z, 1);

Property[SampleCount].x = x;

Property[SampleCount].y = y;

Property[SampleCount].z = z;

SampleCount++;

}

}

\*SampleNum = SampleCount;

free(SampleCellIndex);

}