

Unstructured Points

- No known topological relationship between the data points
 - For example, temperature and humidity sensors arbitrarily placed in an environment
- Lack of inherent structure to directly apply library of visualization techniques
- Requires building the structure

THE UNIVERSITY OF
MEMPHIS

Techniques to Build Structure

- Some approaches for Visualizing Unstructured Points:
 1. Triangulation Techniques (handout 21)
 - vtkDelaunay2D
 - vtkDelaunay3D
 2. Sample unstructured points into an image dataset then use volume rendering or geometric primitives (surface rendering) techniques
 - Splatting Techniques
 - Interpolation Techniques
 - Surfaces from Unorganized Points Techniques

THE UNIVERSITY OF
MEMPHIS

Splatting Techniques

- Build topological structure by sampling unstructured points into an image dataset
- A splatting function $SF(x,y,z)$ is used to distribute the data value of each unstructured point over the surrounding region
- Then volume rendering (e.g., ray casting) or geometric techniques (e.g., iso-surface) can be used for visualization

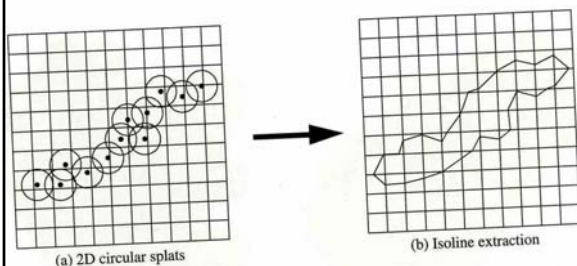
THE UNIVERSITY OF
MEMPHIS

Splatting

- Choose an appropriate footprint or splatting kernel
 - shape of the kernel function should be smooth, symmetric, and fall off to zero away from the center (e.g., Gaussian distribution function). This shape allows the splatting algorithm to blend the volume points.
- If the kernel is too big then too much blending will occur and the images will be blurred and suffer a loss of detail
- If the kernel is too small then holes or gaps between rendered points could occur resulting in a type of aliasing
- The kernel size should be proportional to the spacing of the volume (i.e., grid spacing).
 - size will generally be somewhere between one and two times the grid spacing

THE UNIVERSITY OF
MEMPHIS

2D Splatting Technique



THE UNIVERSITY OF
MEMPHIS

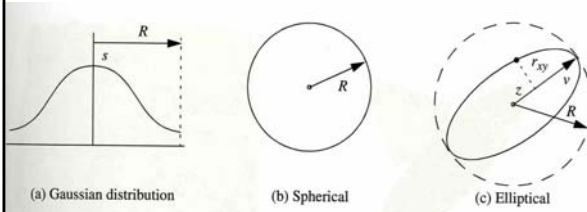
Gaussian Distribution Function

$$SF(x, y, z) = se^{-f(r/R)^2}$$

- centered around a given splat point p_i
- (x,y,z) is the current voxel sample point
- r is the distance $|(x,y,z) - p_i|$; max value (s) when $r=0$
- R is the radius of influence ($r \leq R$)
- f is the exponent scale factor ($f \geq 0.0$); controls decay rate
- s (i.e., scale factor) can be multiplied by the *scalar value* of the point p_i that is currently being splatted

THE UNIVERSITY OF
MEMPHIS

Gaussian Splating Functions



The Gaussian function becomes a circle in cross section in 2D and a sphere in 3D

THE UNIVERSITY OF
MEMPHIS


Splats

- May be accumulated using the standard implicit modeling boolean operations, e.g., form a union of splats
- May modify the shape of the splat according to a vector quantity such as surface normal or vector data

THE UNIVERSITY OF
MEMPHIS

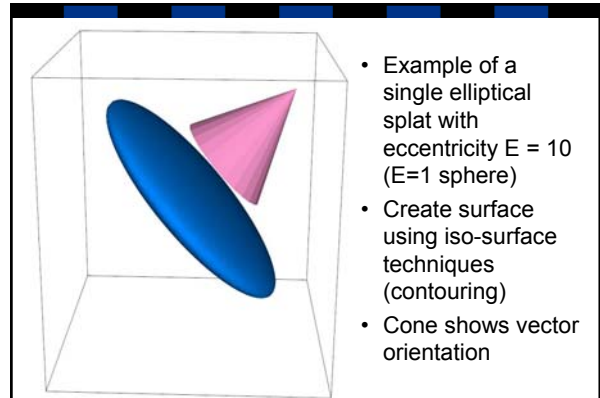
- May want to modify splat based on vector data associated with p_i
 - for example, if point normals are present (and NormalWarping is on), then the splat function becomes elliptical and the Gaussian distribution function then becomes:

$$SF(x, y, z) = se^{-f\left(\frac{r_{xy}^2}{E} + z^2\right)} \quad z = \vec{v} \cdot (p - p_i), \text{ with } |\vec{v}| = 1$$

$$r_{xy} = r^2 - z^2$$


where E is a user-defined eccentricity factor that controls the elliptical shape of the splat; z is the distance of the current voxel sample point along normal N; and r_{xy} is the distance of x in the direction perpendicular to N.

THE UNIVERSITY OF
MEMPHIS



- Example of a single elliptical splat with eccentricity $E = 10$ ($E=1$ sphere)
- Create surface using iso-surface techniques (contouring)
- Cone shows vector orientation

THE UNIVERSITY OF
MEMPHIS

Elliptical Splat

- Has the advantage that we can flatten the splat in the plane perpendicular to the point normal which facilitates bridge the space between sampled points

THE UNIVERSITY OF
MEMPHIS

vtkGaussianSplatter

- Splats points into a volume with an elliptical, Gaussian distribution
- Injects input points into a structured points (volume) dataset
- As each point is injected, it "splats" or distributes values to nearby voxels
- The distribution function is modified using scalar values (expands distribution) or normals (creates ellipsoidal distribution rather than spherical)

THE UNIVERSITY OF
MEMPHIS

vtkPolyDataReader cyber
 cyber SetFileName "\$VTK_DATA_ROOT/Data/fran_cut.vtk"
 vtkPolyDataNormals normals
 normals SetInput [cyber GetOutput]
 vtkMaskPoints mask
 mask SetInput [normals GetOutput]
 mask SetOrthRatio 50
 vtkGaussianSplatter splatler
 splatler SetInput [mask GetOutput]
 splatler SetSampleDimensions 100 100 100
 splatler SetEccentricity 2.5
 splatler NormalWarpingOn
 splatler SetScaleFactor 1.0
 splatler SetRadius 0.025
 vtkContourFilter contour
 contour SetInput [splatler GetOutput]
 contour SetValue 0 0.25
 vtkPolyDataMapper splatMapper
 splatMapper SetInput [contour GetOutput]
 splatMapper SetScalarVisibilityOff
 vtkActor splatActor
 splatActor SetMapper [splatMapper]
 eval [splatActor GetProperty] SetColor 1.0 0.49 0.25
 vtkPolyDataMapper cyberMapper
 cyberMapper SetInput [cyber GetOutput]
 cyberMapper SetScalarVisibilityOff
 vtkActor cyberActor
 cyberActor SetMapper [cyberMapper]
 [cyberActor GetProperty] SetRepresentationToWireframe
 eval [cyberActor GetProperty] SetColor 0.2510 0.8784 0.8157

THE UNIVERSITY OF MEMPHIS

VTK & Multidimensional Viz.

- Multidimensional datasets cannot be categorized by VTK types
- Requires writing custom interface code to convert source data into types defined in VTK
- VTK lacks functionality to regard source data as an n-dimensional matrix
- In some systems, pieces of data can be extracted directly from such an n-dimensional matrix and automatically converted to types such as a volume or structured grid

THE UNIVERSITY OF MEMPHIS

First, splat entire dataset & contour, display as translucent white
 Second, splat entire dataset but scale by an attribute (i.e., time late) and display as red
 Red shows clusters (late payments) in region of high "interest rate" and lower "monthly payment"

TIME_LATE, MONTHLY_PAYMENT, UNPAID_PRINCIPLE, LOAN_AMOUNT, INTEREST_RATE, MONTHLY_INCOME

THE UNIVERSITY OF MEMPHIS

Financial data in vtk field format (financial.vtk)

```

ASCII
FIELD financialData 6
TIME_LATE 1 3188 float
29.14 0.00 0.00 11.71 0.00 0.00 0.00 0.00
0.00 29.14 0.00 0.00 0.00 0.00 0.00 0.00
... (3188 total values)
MONTHLY_PAYMENT 1 3188 float
7.26 5.27 8.01 16.84 8.21 15.75 10.62 15.47
5.63 9.50 15.29 15.65 11.51 11.21 10.33 10.78
...
UNPAID_PRINCIPLE 1 3188 float
430.70 380.88 516.22 1351.23 629.66 1181.97 888.91 1437.83
453.43 851.96 1419.27 1422.11 1009.89 913.47 923.09 893.69
...
LOAN_AMOUNT 1 3188 float
441.50 391.00 530.00 1400.00 650.00 1224.00 920.00 1496.00
476.00 880.00 1480.00 1485.00 1044.00 950.00 960.00 930.00
...
INTEREST_RATE 1 3188 float
13.875 13.875 13.750 11.250 11.875 12.875 10.625 10.500
10.375 10.500 10.000 10.250 11.125 10.250 10.250 10.375
...
MONTHLY_INCOME 1 3188 unsigned_short
39 51 51 38 35 49 45 56
42 35 56 69 58 41 52 41
  
```

A field is represented using 6 arrays

This array has a single component and 3,188 tuples

Alternative is to read and parse data from an ASCII file and populate a VTK data object. See FinancialField.tcl.

THE UNIVERSITY OF MEMPHIS

vtkDataObjectReader

- Reads ASCII or binary field data files in vtk format
- Output is vtkDataObject

```

vtkDataObjectReader fdr
fdr SetFileName "$VTK_DATA_ROOT/Data/financial.vtk"
  
```

THE UNIVERSITY OF MEMPHIS

vtkDataObjectToDataSetFilter

- This class maps a data object (i.e., a field) into a concrete dataset
- It gives structure to the field by defining a geometry and topology using a subset of the data
- Associate components in the input field data with portions of the output dataset
 - a component is an array of values from the field
 - E.g., specify x-y-z points by assigning components from the field for the x, then y, then z values of the points
 - Normalization can be done such that the components distribute the data uniformly

THE UNIVERSITY OF MEMPHIS

```

vtkDataObjectToDataSetFilter do2ds
do2ds SetInput [fdr GetOutput]
do2ds SetDataSetTypeToPolyData
#format: component#, arrayname, arraycomp, minArrayId, maxArrayId, normalize
do2ds DefaultNormalizeOn
do2ds SetPointComponent 0 LOAN_AMOUNT 0
do2ds SetPointComponent 2 MONTHLY_PAYMENT 0
do2ds SetPointComponent 1 INTEREST_RATE 0 0 3187 1

```

Axes
 X -- 0, Y -- 1, Z -- 2

range of data in the component you wish to extract

Normalize between 0 and 1

THE UNIVERSITY OF MEMPHIS

vtkFieldDataToAttributeDataFilter

- maps field data into dataset attributes
- must specify which field data from the input dataset to use
- must specify either cell attribute data or point attribute data

```

vtkFieldDataToAttributeDataFilter fd2ad
fd2ad SetInput [do2ds GetOutput]
fd2ad SetInputFieldToDataObjectField
fd2ad SetOutputAttributeDataToPointData
fd2ad DefaultNormalizeOn
fd2ad SetScalarComponent 0 TIME_LATE 0

```

Specify which attribute data to output: point or cell data attributes

Array name and component of that array

THE UNIVERSITY OF MEMPHIS

vtkGaussianSplatter

```

vtkGaussianSplatter popSplatter
popSplatter SetInput [fd2ad GetOutput]
popSplatter SetSampleDimensions 50 50 50
popSplatter SetRadius 0.05
popSplatter ScalarWarpingOff

```

See next slide

THE UNIVERSITY OF MEMPHIS

ScalarWarpingOff for total population, on for those scaled by time_late

ScalarWarpingOn for both total and time_late populations

THE UNIVERSITY OF MEMPHIS

```

# construct pipeline for original population
vtkGaussianSplatter popSplatter
popSplatter SetInput [fd2ad GetOutput]
popSplatter SetSampleDimensions 50 50 50
popSplatter SetRadius 0.05
popSplatter ScalarWarpingOff
vtkContourFilter popSurface
popSurface SetInput [popSplatter GetOutput]
popSurface SetValue 0 0.01
vtkPolyDataMapper popMapper
popMapper SetInput [popSurface GetOutput]
popMapper ScalarVisibilityOff
vtkActor popActor
popActor SetMapper popMapper
[popActor GetProperty] SetOpacity 0.3
[popActor GetProperty] SetColor .9 .9

# construct pipeline for delinquent population
vtkGaussianSplatter lateSplatter
lateSplatter SetInput [fd2ad GetOutput]
lateSplatter SetSampleDimensions 50 50 50
lateSplatter SetRadius 0.05
lateSplatter SetScaleFactor 0.05
vtkContourFilter lateSurface
lateSurface SetInput [lateSplatter GetOutput]
lateSurface SetValue 0 0.01
vtkPolyDataMapper lateMapper
lateMapper SetInput [lateSurface GetOutput]
lateMapper ScalarVisibilityOff
vtkActor lateActor
lateActor SetMapper lateMapper
[lateActor GetProperty] SetColor 1.0 0.0 0.0

```

finacialfield3.tcl

THE UNIVERSITY OF MEMPHIS

Interesting finding: high interest loans with lower monthly payments are the most risky for late payment

THE UNIVERSITY OF MEMPHIS

Summary of Approach

- Map from n-dimensions to 3-dimensions, then choose attribute data for the points or cells
- Can often be an iterative process to choose the attributes that map to the x-y-z axes and which attribute or attributes to visualize
- Can combine with starplots or other glyphs which are controlled by attribute values to increase the dimensions visualized

THE UNIVERSITY OF
MEMPHIS

Alternative Visualization via Parallel Coordinates

```

# This example converts data to a field and then displays it using
# parallel coordinates
# extract data from field as a polydata (just points), then extract scalars
vtkDataObjectReader fd;
fd->SetFileName("VTK_DATA_ROOT/Data/financial.vtk");

# create data set
vtkDataObjectToDataSeriesFilter do2ds;
do2ds->SetInput(fd->GetOutput());
do2ds->SetDataSetTypeToPolyData();
do2ds->SetDefaultNormalOn();
do2ds->SetPointComponent(0, 5, Axis 0);
do2ds->SetPointComponent(1, 5, Axis 0);
do2ds->SetPointComponent(2, 5, Axis 0);
do2ds->Update();

vtkFieldDataToAttributeDataFilter fd2ad;
fd2ad->SetInput(do2ds->GetOutput());
fd2ad->SetInputFieldID(0, dataObjectField);
fd2ad->SetOutputAttributeDataToPointData();
fd2ad->SetDefaultNormalOn();

vtkDataSeriesToDataObjectFilter do2do;
do2do->SetInput(fd2ad->GetOutput());

# field data for parallel coordinates
vtkParallelCoordinatesActor actor;
actor->SetInput(do2do->GetOutput());
actor->SetTitle("Parallel Coordinates for Financial Data");
actor->SetIndependentVariablesToColumns();
(actor->GetPositionCoordinate())->SetValue(0.95, 0.05, 0.0);
(actor->GetPosition2Coordinate())->SetValue(0.95, 0.85, 0.0);
(actor->GetProperty())->SetColor(1, 0, 0);
(actor->GetLabelTextProperty())->SetColor(1, 0, 0);
(actor->GetLabelTextProperty())->SetColor(1, 0, 0);
    
```

THE UNIVERSITY OF
MEMPHIS

Programmable Sources & Filters

- Alternative to extending VTK (don't need to develop a C++ class and rebuild the libraries)
- Does not require subclassing and the function can be defined in Tcl
- Can develop new filters for execution at runtime
- Create instances of the following:
 - vtkProgrammableSource
 - vtkProgrammableFilter
 - vtkProgrammableAttributeDataFilter

THE UNIVERSITY OF
MEMPHIS

vtkProgrammableSource

- Example use of this class includes writing a function to read a data file or interface to another system
- An alternative to getting source data in VTK format

THE UNIVERSITY OF
MEMPHIS

vtkProgrammableFilter

- Can set input and retrieve output of any dataset type
- To use the filter define a function that retrieves input of the correct type, creates data, and then manipulates the output of the filter

THE UNIVERSITY OF
MEMPHIS

vtkProgrammableAttributeDataFilter

- Facilitates development of a custom procedure to manipulate attribute data (either point or cell data). Examples include:
 - generate scalars based on a complex formula
 - convert vectors to normals
 - compute scalar values as a function of vectors, texture coords, and/or any other point data attribute
- This filter takes multiple inputs (input plus an auxiliary input list), so procedures that combine several dataset point attributes can be written
- The output of the filter is the same type (topology/geometry) as the input

THE UNIVERSITY OF
MEMPHIS

```

# This example demonstrates the use of fields and use of
# vtkProgrammableDataSource. It creates fields the hard way
# (as compared to reading a vtk field file), but shows you how to
# interface to your own raw data.

package require vtk
package require vtkinteraction

set xAxis INTEREST_RATE
set yAxis MONTHLY_PAYMENT
set zAxis MONTHLY_INCOME
set scalar TIME_LATE

# Parse an ascii file and manually create a field. Then construct a
# dataset from the field.
vtkProgrammableDataSource dos
dos SetExecuteMethod parseFile

proc parseFile {} {
    global VTK_DATA_ROOT

    # Use Tcl to read an ascii file
    set file [open "$VTK_DATA_ROOT/Data/financial.txt" r]
    set line [gets $file]
    scan $line "%f %d" numPts
    set numLines [expr ($numPts - 1) / 8 + 1]

    # Get the data object's field data and allocate
    # room for 4 fields.
    set fieldData [[dos GetOutput] GetFieldData]
    $fieldData AllocateArrays 4

    # read TIME_LATE - dependent variable
    # search the file until an array called TIME_LATE is found
    while { [gets $file arrayName] == 0 } {}
    # Create the corresponding float array
    vtkFloatArray timeLate
    timeLate SetName TIME_LATE
    # Read the values
    for {set i 0} {$i < $numLines} {incr i} {
        set line [gets $file]
        set m [scan $line "%f %f %f %f %f %f %f %f"
            v(0) v(1) v(2) v(3) v(4) v(5) v(6) v(7)]
        for {set j 0} {$j < $m} {incr j} {timeLate InsertNextValue $v($j)}
    }
}

```

Source object programmable by the user; output creates vtkDataObject

Read and parse data from an ASCII file and populate a VTK data object.

```

# Add the array
$fieldData AddArray timeLate

# MONTHLY_PAYMENT - independent variable
while { [gets $file arrayName] == 0 } {}
vtkFloatArray monthlyPayment
monthlyPayment SetName MONTHLY_PAYMENT
for {set i 0} {$i < $numLines} {incr i} {
    set line [gets $file]
    set m [scan $line "%f %f %f %f %f %f %f %f"
        v(0) v(1) v(2) v(3) v(4) v(5) v(6) v(7)]
    for {set j 0} {$j < $m} {incr j} {monthlyPayment InsertNextValue $v($j)}
}
$fieldData AddArray monthlyPayment

# UNPAID_PRINCIPLE - skip
while { [gets $file arrayName] == 0 } {}
for {set i 0} {$i < $numLines} {incr i} {
    set line [gets $file]
}

# LOAN_AMOUNT - skip
while { [gets $file arrayName] == 0 } {}
for {set i 0} {$i < $numLines} {incr i} {
    set line [gets $file]
}

```

```

# INTEREST_RATE - independent variable
while { [gets $file arrayName] == 0 } {}
vtkFloatArray interestRate
interestRate SetName INTEREST_RATE
for {set i 0} {$i < $numLines} {incr i} {
    set line [gets $file]
    set m [scan $line "%f %f %f %f %f %f %f %f"
        v(0) v(1) v(2) v(3) v(4) v(5) v(6) v(7)]
    for {set j 0} {$j < $m} {incr j} {interestRate InsertNextValue $v($j)}
}
$fieldData AddArray interestRate

# MONTHLY_INCOME - independent variable
while { [gets $file arrayName] == 0 } {}
vtkIntArray monthlyIncome
monthlyIncome SetName MONTHLY_INCOME
for {set i 0} {$i < $numLines} {incr i} {
    set line [gets $file]
    set m [scan $line "%d %d %d %d %d %d %d %d"
        v(0) v(1) v(2) v(3) v(4) v(5) v(6) v(7)]
    for {set j 0} {$j < $m} {incr j} {monthlyIncome InsertNextValue $v($j)}
}
$fieldData AddArray monthlyIncome
}

```

```

# Create the dataset.
# DataObjectToDataSetFilter can create geometry using
# fields from DataObject's FieldData
vtkDataObjectToDataSetFilter do2ds
do2ds SetInput [dos GetOutput]
# We are generating polygonal data
do2ds SetDataSetTypeToPolyData
do2ds DefaultNormalizeOn
# All we need is points. Assign them.
do2ds SetPointComponent 0 $xAxis 0
do2ds SetPointComponent 1 $yAxis 0
do2ds SetPointComponent 2 $zAxis 0

# RearrangeFields is used to move fields between DataObject's
# FieldData, PointData and CellData.
vtkRearrangeFields rf
rf SetInput [do2ds GetOutput]
# Add an operation to "move" TIME_LATE from DataObject's FieldData to
# PointData"
rf AddOperation MOVE_Scalar_DATA_OBJECT_POINT_DATA
# Force the filter to execute. This is needed to force the pipeline
# to execute so that we can find the range of the array TIME_LATE
rf Update
# Set max to the second (GetRange returns [min,max]) of the "range" of the
# array called Scalar in the PointData of the output of rf"
set max [lindex [l [rf GetOutput] GetPointData] GetArray Scalar] GetRange 0] 1]

```

Define which attribute data serves as x-y-z axes

```

# Use an ArrayCalculator to normalize TIME_LATE
vtkArrayCalculator calc
calc SetInput [rf GetOutput]
# Working on point data
calc SetAttributeMode ToUsePointData
# Map Scalar to s. When setting function, we can use s to
# represent the array Scalar (TIME_LATE)
calc AddScalarVariable s $scalar 0
# Divide Scalar by Smax (applies division to all components of the array)
calc SetFunction "s / Smax"
# The output array will be called resArray
calc SetResultArrayName resArray

```

performs operations on vectors or scalars in field data arrays

```

# Use AssignAttribute to make resArray the active scalar field
vtkAssignAttribute aa
aa SetInput [calc GetOutput]
aa Assign resArray SCALARS POINT_DATA
aa Update

# construct pipeline for original population
vtkGaussianSplatter popSplatter
popSplatter SetInput [aa GetOutput]
popSplatter SetSampleDimensions 50 50 50
popSplatter SetRadius 0.05
popSplatter SetWarpingOff

vtkContourFilter popSurface
popSurface SetInput [popSplatter GetOutput]
popSurface SetValue 0 0.01

vtkPolyDataMapper popMapper
popMapper SetInput [popSurface GetOutput]
popMapper SetScalarVisibilityOff

vtkActor popActor
popActor SetMapper popMapper
[popActor GetProperty] SetOpacity 0.3
[popActor GetProperty] SetColor 9 9 9

```

FinancialField.tcl is a good starting point if your application calls for parsing data from a non-VTK formatted ASCII file

Interpolation Techniques

- Construct a function to smoothly interpolate from a set of unstructured points
- Provides a mechanism to estimate values at un-sampled points
- F_i are the prescribed function values at the scatter points (e.g. the data set values)
- Create a function $F(p)$, given a set of n points $p_i = (x_i, y_i, z_i)$ and function values $F_i(p_i)$
- Build topological structure from the n points by sampling $F(p)$ over an image dataset
- Then use techniques that can process vtkImage data (i.e., either geometric or volume rendering techniques)

THE UNIVERSITY OF
MEMPHIS

Inverse Distance Weighted Interpolation Technique

- Based on the assumption that the interpolation should be influenced most by the nearby points and less by the more distant points
- The interpolation is a weighted average of the scatter points and the weight assigned to each scatter point diminishes as the distance from the interpolation point to the scatter point increases

THE UNIVERSITY OF
MEMPHIS

Inverse Distance Weighted Interpolation Technique

Shepard's Method

Let $F_i = F_i(p_i)$

$$F(p) = \left(\sum_{i=1}^n F_i / |p - p_i|^2 \right) / \left(\sum_{i=1}^n 1 / |p - p_i|^2 \right)$$

THE UNIVERSITY OF
MEMPHIS

Inverse Distance Weighted Interpolation Technique (Disadvantages)

- Computationally inefficient since the method is global in that any interpolation involves a computation involving all data points
- Improvements include weighting functions that are subjected to a damping factor to reduce them to zero outside a certain radius of the data point
- Generates a local flat spot at each point p_i since the derivatives are zero: $\frac{dF}{dx} = \frac{dF}{dy} = \frac{dF}{dz} = 0$

THE UNIVERSITY OF
MEMPHIS

```
# create some points
#
vtkMath math
vtkPoints points
for (set i 0) ($<50) {incr i 1} {
  eval points InsertPoint $i [math Random 0 1] [math Random 0 1] [math Random 0 1]
}

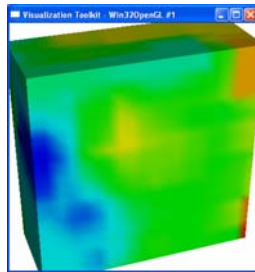
vtkFloatArray scalars
for (set i 0) ($<50) {incr i 1} {
  eval scalars InsertValue $i [math Random 0 1]
}

vtkPolyData profile
profile SetPoints points
[profile GetPointData] SetScalars scalars

vtkShepardMethod shepard
shepard SetInput profile
shepard SetModelBounds 0 1 0 1 1 5
shepard SetSampleDimensions 20 20 20
shepard Update

vtkDataSetMapper map
map SetInput [shepard GetOutput]

vtkActor block
block SetMapper map
[block GetProperty] SetColor 1 0 0
```



THE UNIVERSITY OF
MEMPHIS

Surfaces from Unstructured Points

- VTK has a class to reconstruct surfaces from points
- Can create surfaces from point clouds
- The unstructured points are assumed to lie on a surface of a 3D object of interest

THE UNIVERSITY OF
MEMPHIS

vtkSurfaceReconstructionFilter

- vtkSurfaceReconstructionFilter takes a list of points assumed to lie on the surface of a solid 3D object
- A signed measure of the distance to the surface is computed and sampled on a regular grid that bounds the input set of points
- The grid can then be contoured at zero to extract the surface
- Works well if points are close together

```
# Construct the surface and create isosurface.
#
vtkSurfaceReconstructionFilter surf
surf SetInput [pointSource GetPolyDataOutput]

vtkContourFilter cf
cf SetInput [surf GetOutput]
cf SetValue 0 0.0

vtkReverseSense reverse
reverse SetInput [cf GetOutput]
reverse ReverseCellsOn
reverse ReverseNormalsOn

vtkPolyDataMapper map
map SetInput [reverse GetOutput]
map SetScalarVisibilityOff

vtkActor surfaceActor
surfaceActor SetMapper map
```

unorganized points
of interest

Sometimes when processing unorganized points, the contouring algorithm can create a volume whose gradient vector and ordering of polygons are inconsistent. For example, the normals may point inward resulting in a black surface. vtkReverseSense cures this problem.