Ninth Irish Workshop on Computer Graphics (2009)
Eurographics Irish Chapter

Interactive 3D Graphics for Cancer Modelling

Yin Jie Chen[1], Sabin Tabirca[1], Mark Tangney[2]

Department of Computer Science, University College Cork, Ireland[1]
Cork Cancer Research Centre, Ireland[2]

Abstract

Using 3D graphics technology to rich traditional chart presentation will make presentations carry more information and easier be understood by audiences. The paper discusses making 3D presentation easer to use and more reusable by unprofessional people, the structure that 3D models are organized in a scene, and the 3D scene takes data from database tables by data interfaces.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.5]: Computational Geometry and Object Modeling—Computational Geometry and Object Modeling [I.3.5]: –*classification*

1. Introduction

1.1. Visualisation for Cancer Data

In recent years there has been a great deal of interest concerning three catch-phrases: nonlinear dynamics, chaos, and complexity. This interest has led to a large number of popular-science articles relating to the visualization of cells, many of these feature very high end graphical simulations [Wil97].

Technologies such as ultrasound are being used to evaluate tumours in 3D [HGGS99]. 3D microscopy imaging to display 3D biological tissue architecture during carcinogenesis [aJCR95]. Models and Simulations are being widely used, for example Mathematical models have been used to describe ontogenetic growth [WBE02]. Two-dimensional modelling has been used in the area of tumour growth and morphology [AMWd03], while simulations have been used in the area of benign avascular tumour growth [SB99].

The measure of cancer tumour size is very important in cancer research. The experiments data in mouse usually recorded into table data and presented only in 2D charts. Usually chart presents two or more lines to describe different experiment field data in different time. This is main method to comparing two or more data fields.3D charts extend the traditional 2D charts to be able to handle more data. The presentation is more material and easier to understand.

1.2. 3D Chart Form for Visualisation Cancer Data

A chart is a visual representation of data, in which the data are represented by symbols such as bars in a bar chart or lines in a line chart [JA90]. A chart can represent tabular numeric data, functions or some kinds of qualitative structures. Charts are widely used in different area to present data especially for tabular data. Tabular usually presents a serious of columns values. Each column is defined for a specific meaning in different conditions.

The common 3D chart form is just an extension of typical 2D chart. Most of 3D charts are use to increase visual effects and few of them to present three columns data.

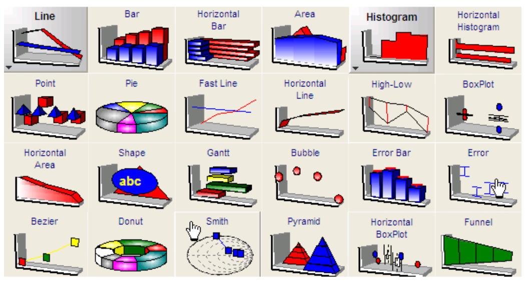


Figure 1: *3D charts example*

The figure shows some type of 3D charts(Figure 1). These types of 3D charts are only present lines, bars, pies and other geometrical objects in 3D method. It only enhances the presentation visual effect and no more significant meaning.

A lot of 3D scenes or presentation models are built to achieve this kind of requirements. Usually one scene or model only presents one type of data, and the reusability is quite poor. We try to find a way that allow user to build their scene or model to satisfy presentation that will give maximum reusability and keep easy usage at the same time.

1.3. Enhanced 3D chart in "VisEx"

VisEx is a visualisation tool that visualises abstract data to graphic presentations. VisEx allow user to create an experiment table data form database or input manually then visualization data be multiply forms that includes customization 3D methods.

Various forms of chart could be used for presenting data. These forms are usually abstract. E.g. bar chart abstract values in different length of bars, the quantity is emphasized in this presentation but the meaning of the values only presented by text. In some situations especially in cancer research and education, the meaning and value of data are both needs to be emphasized, or the data needs to be presented in different methods and comparing each other. VisEx visualisation tool can handle this type of problem. For example, a list of cancer tumour volume data needs to be presented in different time with treatment and no treatment conditions. VisEx could present cancer tumour volume by 3D sphere with different size. The sphere size will change according to time has been selected.

VisEx provides data collection and edit function that allow user to select data from different data sources, the data could be selected from local Microsoft Excel data file or from a specific database. User can only select needed data from a table even a cell a row or a column. These data is organized in VisEx for visualisation proposes and the data could be share with other researchers as well. The prepared data can be visualised in different forms in VisEx, traditional 2D charts, image sequences and customization 3D methods (scenes).

Visualised data could interactive with user selection based on a time line data filed that usually presents time. When different time line data value is selected, all relatives' presentation will change.

The most important function in VisEx is customization 3D scene. This function allow user to build their own 3D scene to present their data. VisEx provides a very easy to use platform to assemble 3D scenes. User could select 3D models from a public model library in VisEx and to set properties for each, finally to build a 3D scene.

1.4. Requirements to Enhance 3D Chart Graphics

The following points needs to be achieved to enhance traditional 3D chart and satisfied the visualisation of cancer experiment data:

- Render 3D model by different value with different aspects. The 3D model will change by apply different data value.
- The aspect for model changing could be set by user. For example user could set a data to control a sphere size, color or alpha channel.
- Application provides various presentation forms at same time. Different presentation form could emphasis different aspect of data, for example traditional line chart could emphasis the relationship between each value, but not the data meaning itself. 3D chart could emphasis the meaning at specific point but not the relationship. The multiply visualisation forms are presented together that will rich the presentation structure and easier to understand.
- The presentation template must be extendable. It is no reason to build a fixed visualisation model into a visualisation tool. Application should provide extendable structure for 3D scene graphics.
- The application must be easy to use. The visualisation tool is for cancer researchers or people work on various areas but don't have too much experience in computer or 3D graphics.

2. Main Feature of Interactive 3D Graphics

The basic idea of interactive 3D graphic is that user selects models form a public library then assemble them into a 3D scene and user can set data interfaces that connect between data value and model aspects, finally to build customization 3D scene for interactive 3D graphic presentation.

The figure shows a customization 3D chart(Figure 2).

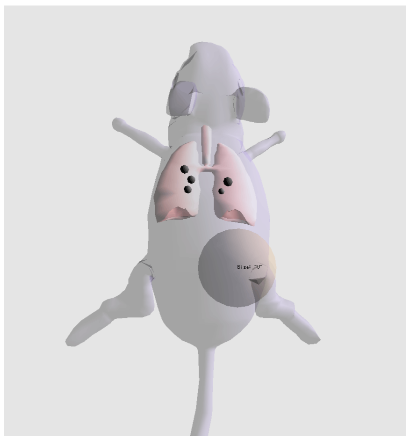
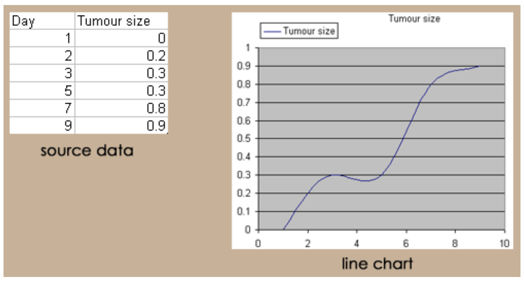


Figure 2: *A customization 3D chart*

Here is a table data that present a tumour volume in different days. The data could be presented into a traditional line chart (Figure 3).



Day	Tumour size
1	0
2	0.2
3	0.3
5	0.3
7	0.8
9	0.9

Figure 3: *Traditional visualization for a table data in Microsoft Excel*

In our interactive 3D graphic, a customization 3D scene was built for this. If this tumour is on the back of a mouse, the data will be presented in a 3D scene that contains a mouse and a tumour which presented as a sphere. The sphere size will scale by different data those in different days(Figure 4).

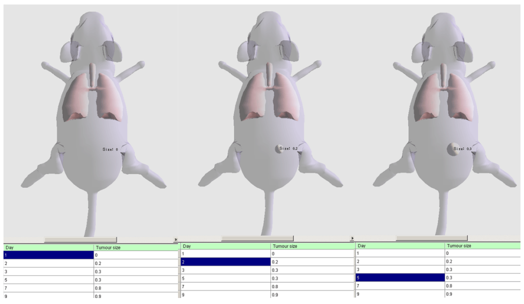


Figure 4: *Customization 3D chart visualization for a table data*

The customization 3D chart presents data more straightforward and animated. The 3D mouse brings more information of this data, for example the tumour location and proposition of this data. This is easier to understand and to compare with different data by audiences.

As a typical customization 3D chart for cancer experiment data in this case contains following features:

- A 3D mouse model or human body model presents experiment target. Some of organs need to fit into the body according to specific requirement. The organs are different in different experiment and the location, size, color might be different, and so these aspects of 3D graphics have to be resettable.
- The cancer mets need to fit into organs randomly in this case. Mets location in an organ could be random and the number of mets is from data interactively.
- The cancer tumours need to fit into 3D graphic. Tumours could be in an organ or just in a place in body for example in the back. Tumours volume grow overtime and it interact to data value by time.

3. Design of the Interactive Modeling

3.1. 3D Scene Structure

The 3D scene is organized into a few of classes(Figure 5). ScenList is the class contains a group of Scenes on the top, it provides various 3D scenes to visualisation. New scene that is built by user will be stored into the group. Scene class holds a list of Components; it organizes a 3D scene with serial components. Component class is the core class for a scene data, it contains a Model class that present 3D model aspects and DataInterface class that records information between data and model. Component is self-contain type. A component object might contain another component.

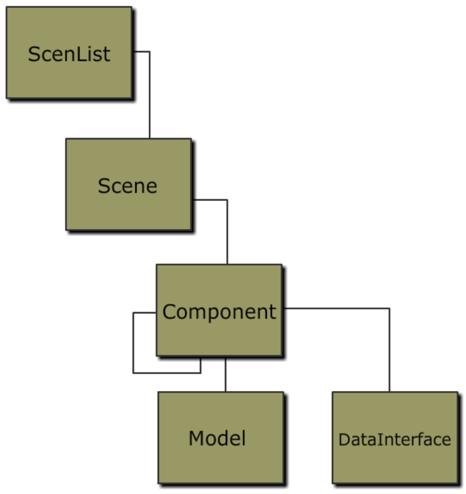


Figure 5: *Structure of 3D scene*

The 3D scene can be built by user freely and conveniently. The 3D scene data structure must have these functions:

- Dynamic add components, dynamic size. To custom a 3D scene, the components have to be added dynamically, because the components number will depend on the data. For storage, the dynamic structure will save space and more effective than static structure.
- Components can be grouped. Some of components logically need to be grouped together, like the accessories of a car should be in car group. Functionally, some time we might need do a same process to multiple components, there are need to be grouped.
- Component is self-contained. One component might contain another component; it provides ability that all components could be processed by a same approach. It will be basic aspect for group structure.

The "List" data structure is selected for dynamic structure in Csharp. Declaring a List of our component type make component easy to add, remove and visit. The component is the basic unit for 3D scene. A component contains a list of component with its own data. This structure will satisfy previous requirements. The figure shows this structure(Figure 6).

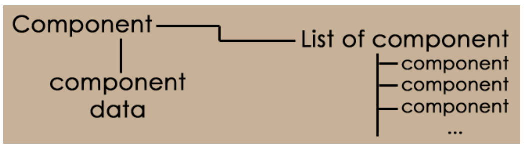


Figure 6: *Structure of component*

The class code section is shown follow(Listing 1):

```
...
public class Component : ISerializable
{
   public List<Component> components;
    // list of component
   // own data
   public Model model;
   public String componentName;
   public List<DataInterface> dataInterfaces;
   public String modelFileName;
   ...

   public Component(String comName, String filename)
   {
      componentName = comName;
      components = new List<Component>();
      ...

   public Component(Component cx)
   {
      this.componentName = cx.componentName;
      this.components = new List<Component>();
      foreach (Component c in cx.components)
      {
         this.components.Add(new Component(c));
      }
      ...
   }
   ...
}
...
```

Listing 1: *Component Class*

A 3D scene data has same structure as component; a 3D scene contains a list of components with its own data. In the application, user could build several of scenes, all scenes and their components are presented in a tree structure.

3.2. Data Processing

Processing 3D scene data is same as to processing a tree structure in this case. How to visit correct component in the tree is the main task. For adding or removing a new component, the only thing needs to do is that visit parent component then add new component or remove component under it. Because of the uniform of the component structure for components visiting, the data processing could use same function with recursive calling. The following code shows a recursive function for loading components(Listing 2).

```
private void loadComp(Component c, TreeNodeCollection tn)
{
   TreeNode tnn = new TreeNode(c.componentName);
   tn.Add(tnn);
   foreach (Component cx in c.components)
   {
     loadComp(cx, tnn.Nodes);
   }
}
```

Listing 2: *Recursive function for loading components*

The function creates a TreeNode data type that will contain a component object. Function search each of component in the component list as the parameter to call function itself again with each component and new tree node.

Recursive function is suit for tree data processing, because usually the depth of tree is unknowable, to visit data from root to leaf without more controls providing a dynamic structure for the tree.

Csharp provides a TreeNode type for tree structure, a TreeNode could add any type of object in, there is another class TreeView that is for user interface that can display a TreeNode type data in a window. There are two aspects need keep:

- Using TreeView to display components in a scene is convenient and no need writing user interface class, so components tree view need to be converted into TreeNode type.
- To store a scene object, there is no reason to add extra data in components structure, no TreeNode type. The TreeNode type needs to be converted into components tree structure.

Components data need to be read by recursive function for loading scene components data then added into TreeNode. The listing function is for is(Listing 2). The parent component needs to be found when a new component is added or removed as well. The figure shows TreeView user interface and rendered components in a scene(Figure 7).

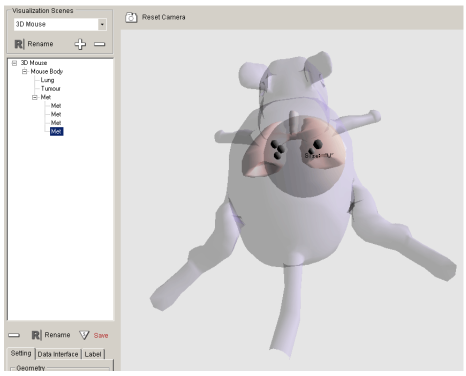


Figure 7: *TreeView and Scene*

3.3. Data Storage

When a new 3D scene is built, a Scene object is created by class, the components are dynamically added in by component objects. The scene object needs to be saved for saving whole scene data. To save an object, use serialization to serialize object. The component objects that in scene or parent component needs to be serialized as well for serialization of recursive structure. Serialization is the process of converting the state of an object into a form that can be persisted or transported. The complement of serialization is deserialization, which converts a stream into an object. Together, these processes allow data to be easily stored and transferred [Mic09]. The code section shows serialization constructor function and deserialization function GetObjectData(Listing 2).

```
public Component(SerializationInfo info,
             StreamingContext ctxt)
{
   this.model = (Model)info.GetValue("model",
                      typeof(Model));
   this.components =(List<Component>)info.GetValue(
         "components", typeof(List<Component>));
   this.componentName = (String)info.GetValue(
              "componentName", typeof(String));
   this.dataInterfaces =
            (List<DataInterface>)info.GetValue(
    "dataInterfaces", typeof(List<DataInterface>));
   ...
}

public void GetObjectData(SerializationInfo info,
                    StreamingContext ctxt)
{
   info.AddValue("model", this.model);
   info.AddValue("components", this.components);
   info.AddValue("componentName", this.componentName);
   info.AddValue("dataInterfaces", this.dataInterfaces);
   ....
}
```

Listing 3: *Serialization and deserialization functions*

The serialization constructor function loads all object data form SerializationInfo data then initializes corresponding variables. This processing is equivalent to copy objects. The GetObjectData function save object data into SerializationInfo data.

3.4. Data Interface

To visualize table field data into 3D scene model, a connection between field and specific 3D model aspect needs to be built. E.g. the first column data will control the scale aspect of 3D model then these two data needs to be connected. Data interface is the data type that handles this functionality(Figure 8).

Once the connection between data field and 3D model is linked, the model will be rendered by the data interface type with the field data value(Figure 9).

One 3D model or one component could have multiple data interfaces at same time. The number of data interface is unpredictable, so the data interface should be organised into a dynamic type that is List type as well. A component object contains a list of data interface; the interface contains its type, experiment data column and 3D model.

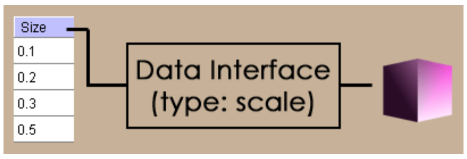


Figure 8: *Data Interface*

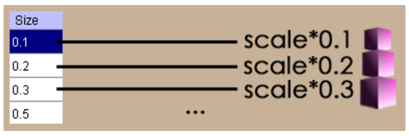


Figure 9: *3D Model Change by Linked Value*

3.5. Graphics

The graphics in application is implemented by OpenGL in Csharp. The component class contains a render function that will render the model by scene setting. When a scene is rendering, each of component will be rendered by calling render function(Listing 4).

```
public override void glDraw(){
   ...
   GL.glLoadIdentity();
   if (scene != null){
      foreach (Component comp in scene.components){
         comp.isDataLink = this.isLinkData;
         comp.render(fCameraX, fCameraY,fCameraDis,this);
      }
   }
   GL.glFlush();
}
```

Listing 4: *Scene Render Function*

The render function into component class will load each of data interface then shift model by current data value. This value will shift by a correction number; this correction number is for measure the standard unit. There is no absolute size or value for "1" in a 3D scene, because these values are measured relatively. The correction value could change the unit to fit the requirement.

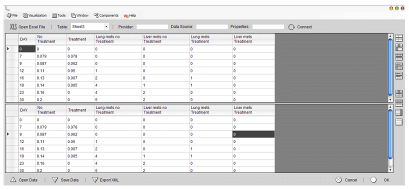
The code list shows the render function into component class(Listing 5). The switch expression will shift component according to type of the data interface with the correction values, after that the function will call render function that in model class to render model itself. Finally function call each of sub-component render function recursively.

```
public void render(...){
   ...
   foreach (DataInterface di in dataInterfaces){
      switch (di.dl_index){
            // scale all index = 0
            case 0:
               model.mScale(this.scale_x*correctValueF(di),
               this.scale_y*correctValueF(di),
               this.scale_z*correctValueF(di));
            break;
            // alpha index =1
            case 1:
            ...
            default: break;
         }...
      }
      ...
      if (model != null) model.Render();
      ...
      foreach (Component c in this.components){
         c.isDataLink = this.isDataLink;
         c.render(cx, cy, cDis,soc);
      }
}
```

Listing 5: *Component Render Function*

4. VisEx and User Interface

3D chart is a very important part of the application VisEx. VisEx is a visualisation tool that visualises abstract data to graphic presentations. VisEx allow user to create an experiment table data form database or input manually. This experiment table data could be editing and assembled form different data sources(Figure 10).



DAY	No Treatment	Treatment	Lung mets no Treatment	Liver mets no Treatment	Lung mets Treatment	Liver mets Treatment
0	0	0	0	0	0	0
7	0.079	0.078	0	0	0	0
9	0.087	0.062	0	0	0	0
12	0.11	0.05	1	0	0	0
16	0.13	0.007	2	0	1	0
19	0.14	0.005	4	1	1	0
23	0.16	0	4	2	0	0
30	0.2	0	5	2	0	0

DAY	No Treatment	Treatment	Lung mets no Treatment	Liver mets no Treatment	Lung mets Treatment	Liver mets Treatment
0	0	0	0	0	0	0
7	0.079	0.078	0	0	0	0
9	0.087	0.062	0	0	0	0
12	0.11	0.05	1	0	0	0
16	0.13	0.007	2	0	1	0
19	0.14	0.005	4	1	1	0
23	0.16	0	4	2	0	0
30	0.2	0	5	2	0	0

Figure 10: *Screenshot of Experiment Editing Window*

The experiment table data could be visualised by traditional chart or new 3D chart and each of row data could be link to single image that might be generated by other software. 3D chart presentation is organised by a scene, scene could be built by user, user could select 3D model form application model library to assemble new scene and the application model library is extendable(Figure 11).

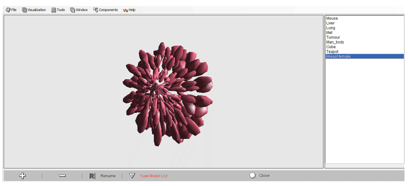


Figure 11: *Screenshot of Model Library Window*

VisEx provides some statistic functions that usually use for cancer tumour research propose. The interpolation and prediction functions also could be used for cancer research data as well(Figure 12).

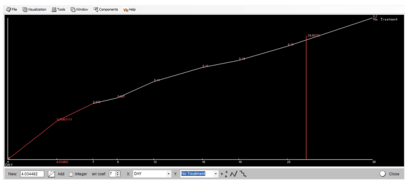
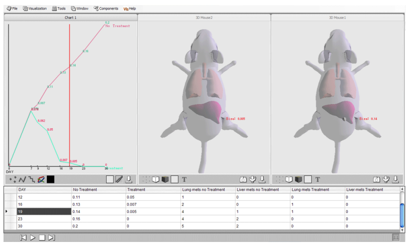


Figure 12: *Screenshot of Interpolation Window*

VisEx default models and 3D chart scenes are for cancer research proposes, but the application could be use for any proposes that needs 3D chart by adding extension models and building new scenes. The figure shows an application group visualisation windows for cancer research(Figure 13).



DAY	No Treatment	Treatment	Lung mets no Treatment	Liver mets no Treatment	Lung mets Treatment	Liver mets Treatment
12	0.11	0.05	1	0	0	0
16	0.13	0.007	2	0	1	0
19	0.14	0.005	4	1	1	0
23	0.16	0	4	2	0	0
30	0.2	0	5	2	0	0

Figure 13: *Screenshot of Visualisation Windows*

5. Conclusion

Various 3D chart presentations extend traditional chart visualisation forms. Comparing normal 2D chart and 3D chart, 3D chart present more directly for describe material type of data. Audients are easier to understand especially in education area. These 3D presentations are usually only for specific proposes and without reusability. The application VisEx provides a frame that balance between reusability and ease for use. User could build customization 3D scene for specific presentation only with simply operations. This frame could be use for various areas with extension of model library.

References

[aJCR95] AROLE J. CLEM, RIGAUT J. P.: *Computer simulation modelling and visualization of 3D architecture of biological tissues*, vol. 43. (425-442) Springer Netherlands, 1995.

[AMWd03] ADRIANA N., MOMBACH J., WALTER M., DEAVILA F.: *The interplay between cell adhesion and environment rigidity in the morphology of tumors*, vol. 322. (546-554) Elsevier Science, 2003.

[HGGS99] HUNERBEIN M., GHADIMI B., GRETSCHEL S., SCHLAG P.: *Three-dimensional endoluminal ultrasound: a new method for the evaluation of gastrointestinal tumors*, vol. 24. (445-448) Springer New York, 1999.

[JA90] JENSEN C., ANDERSON L.: *Harvard Graphics: The Complete Reference*. 1990.

[Mic09] MICROSOFT: *Serializing Objects -.NET Framework Developer's Guide*. 2009.

[SB99] STOTT E. L., BRITTON N. F.: *Stochastic Simulation of Benign Avascular Tumour Growth Using the Potts Model*, vol. 30. (183-198) Elsevier Science, 1999.

[WBE02] WEST G., BROWN J., ENQUIST B.: *A general model for ontogenetic growth*. (626) Nature, 2002.

[Wil97] WILKINSON M.: *Nonlinear Dynamics, Chaos-theory, and the "Sciences of Complexity" : Their Relevance to the Study of the Interaction between Host and Microflora*. (110-130) Old Herborn University Seminar Monograph 10: New Antimicrobial Strategies, 1997.