Scene Graph: Implementing Graph in Graphical Data Processing

Hafizh Adi Prasetya - 13511092
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13511092@std.stei.itb.ac.id

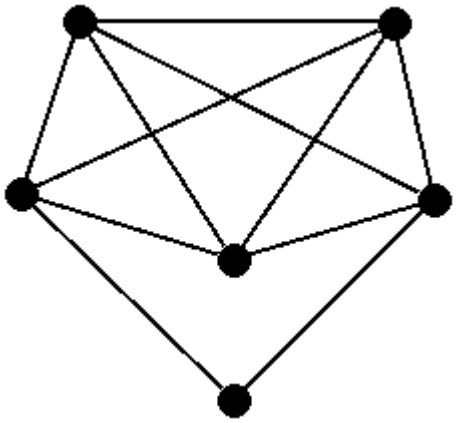
Graph is a common data representation, commonly used in mathematics. It is studied heavily in discrete mathematics. But extensively, graph is also used in the field of informatics as a data structure, covering a lot of application in different contexts. This is because the versatile, unique, and flexible set of quality it has. One of the more common application for graph as a data structure nowadays is something called Scene Graph. Scene graph are basically used in processing graphics, like in vector-based graphical editor such as Adobe Illustrator, or in 3D model rendering. With the boom of 3D models in many aspects of current industry, especially with animation and gaming industry, Scene Graph become one of the more popular and important application of the graph.

Index Terms—Graphs, Scene Graphs, Graphical Data Structure, 3D.

I. INTRODUCTION

Graph is a classical data representation commonly used in mathematics and computer science. Graph theory originated back in 1736, when Leonhard Euler writes his paper on the *Seven Bridges of Konigsberg*.[1] In his paper, he addresses the problem about the ways to cross a certain arrangement of bridges each once. He then models this problem into a representation of points and lines, which then he called a Graph.

A graph itself can most simply be defined as a representation of a set of objects where some pairs of the objects are connected by links. The interconnected objects are represented by mathematical abstractions called *vertices*, and the links that connect some pairs of vertices are called *edges*.[2] A valid graph contains at least one vertex, which means that a graph doesn't necessarily have to have edges. It is possible that a graph consists solely of vertices, with no edges. In this case, the graph may represent a collection of objects with no relation whatsoever.



Picture I.1 A simple graph with 6 vertices and 10 edges

In common practice, the vertices of the graph represent objects, while the edges that connects them represents the existence of a relation between two objects. For example if vertices A and B is connected by an edge, it means that in a certain way, A is connected with B. A more contextual example is when a graph represents a website. The vertices A and B represents page A and B in the website, while an edge connecting A and B means that there is a link from page A to page B, or the other way around. But despite the common practice, this is not always true. The representation of vertices and edges will vary on the usage and the intention of the graph user.

Of course, graph theory spans wider than a simple graph. A more advanced graph can have direction on the edges, signifying a one-sided relationship vertices have. Edges can also have values, that shows how significant a certain relationship between vertices. Example is when vertices A and B represents place, and the edge connecting them represents the road. The value of edge A-B could represent the distance between place A and place B.

Graphical processing application, especially ones that are vector-based, mostly process images or graphics in discrete objects. For example, a face vector graphic may consist of eyes, nose, lips, ears, hairs, and other objects. Of course it is also possible to make the face into one object, but doing so creates a more complex function (as vector graphics represent objects in form of mathematical

function), and sometimes will be harder to handle. That is why, these little objects will most likely be easier to work with. In this context, graph can provide a way to organize, sort and modify those little objects, to create a bigger, more complex object.

This is also true in the rendering of 3D objects, you can divide big objects or scenery into smaller elements, for example the lighting, the fog, the humidity, and other stuffs. This makes it easier to render these elements individually rather than as one big chunk of scenery. The organization of these elements is where graph theory comes in.

Picture I.2 An example of 3D graphics, with elements such as units, shadows, fires, and other

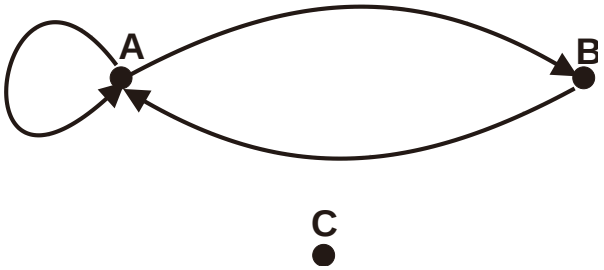
It is best to keep in mind that the word 'graphical' does not refer to the affinity of being a graph, but rather the affinity of visual.

II. BASIC THEORIES

II.1 Graph Terminology

Vertices are the points in the graph, usually representing the objects of the data. A vertex can be connected to other vertices by lines. These lines are called **edges.** An edge could have an arrow, representing the direction of the relationship. Other than direction, an edge could have a value attached to it. This value represents the significance of the relationship. It is possible that an edge connect a vertex to itself. In this case, the edge is referred as a **loop**. It is also possible that two vertices are connected by more than one edge. This case is referred as them having **multiple edges**.

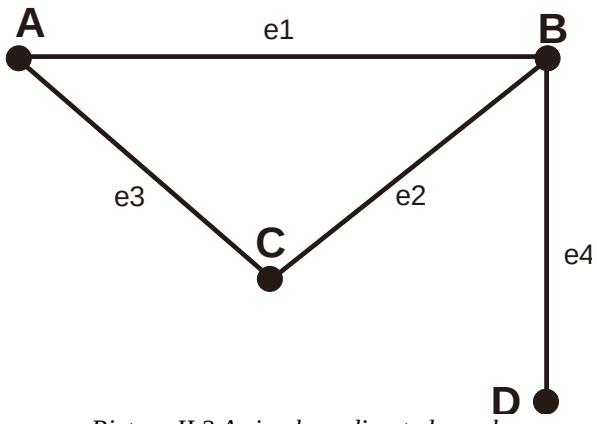
Two vertices are called **adjacent** to each other if they are connected directly by an edge, whether the edge it directed, weighed, or neither. An edge could be caled **incident** with a vertex if the edge connect the vertex and any other vertex. In a case where a vertex has no incident edge (and in consequence, no adjacent vertex), a special name is given. These kinds of vertices is called **isolated vertex**.



Picture II.1 A graph with vertices A, B,and C

In the picture above, we can see that three vertices exist. Vertex A, vertex B, and an isolated vertex C. A loop exists in vertex A, while multiple edges connects vertex A and vertex B. By previous definitions, we can call A and B adjacent, and B are incident with the edge that connect it to A. A vertex is said to have a certain **degree**. The degree of a vertex is equal to the number of edges that are incident with the vertex. Naturally, an isolated vertex has zero degree. Vertex A has a degree of three, vertex B has a degree of two, and vertex C, being an isolated vertex, has a degree of zero.

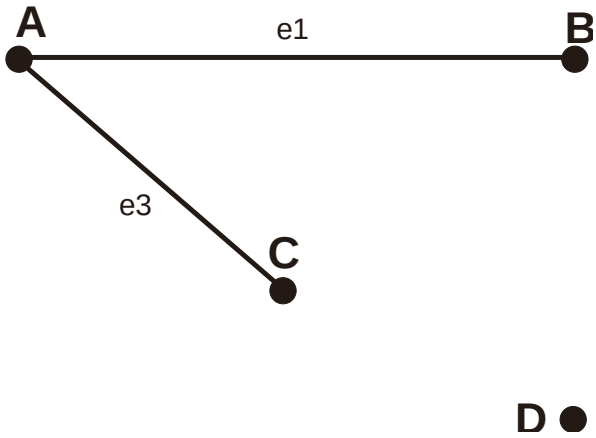
The **path** from vertex A to vertex B is defined as row alternating between a vertex and the edge that is incident with it, from vertex A to vertex B. In the special case when the path starts and ends at the same vertex, the path is called a **cycle**. Lastly, when a path exist between vertex A and vertex B, the two vertices can be called **connected**.



Picture II.2 A simple undirected graph

Picture II.2 shows an example of paths and loops. The path from A to D can be written as A-e1-B-e4-D. This however, is not a cycle, since it starts on vertex A and ends on vertex D. A cycle is shown on the path A-e1-B-e2-C-e3-A, since it starts and ends on vertex A.

A **subgraph** is a graph which all vertices and edges are member of a bigger graph. Thus, graph B is called a subgraph of graph A if all vertices and edges in graph B exists also in graph A.



Picture II.3 A subgraph of graph in picture II.2

II.2 Classification of Graphs

A graph can be classified in lot of ways. Based on the existence of a loop or multiple edges, a graph can be classified into two. **A simple graph** is a graph that contains no mutiple edges or loop. A graph with either one is called an **unsimple graph**.

Looking at the pictures, we can see that picture II.1 depicts an unsimple graph, because of the loop that exists in vertex A, and also the multiple edges connecting it to vertex B. On the other hand, Picture II.2 shows a simple graph, where no loops and multiple edges exists.

Before, it is already explained how an edge of a graph can have a direction. A graph with a directed edge is called a **directed graph**, or **digraph**. When a directed graph contains a multiple edge with different direction, the graph is further differentiated by the name **multidirected graph**.

As per se, we can conclude that the graph in picture II.1 is a multidirected graph, while the one in picture II.2 is a regular graph.

II.3 Computer Graphics

III. Scene Graph as an Application of Graph in the Visual Graphic Field

III. 1 Scene Graph

According to the website Dr. Dobbs, *"Scene graphs are data structures used to hierarchically organize and manage the contents of spatially oriented scene data. Traditionally considered a high-level data management facility for 3D content, scene graphs are becoming popular as general-purpose mechanisms for managing a variety of media types. MPEG-4, for instance, uses the Virtual Reality Modeling Language (VRML) scene graph programming model for multimedia scene composition, regardless of whether 3D data is part of such content.*[3]"

Despite such definition, in reality, the definition of a scene graph is somewhat hazy, since in the implementation process, programmers tweak the definition to adapt with the situation, taking only the very basic principle of the scene graph. This means that there is no such clear consensus on what a scene graph is.

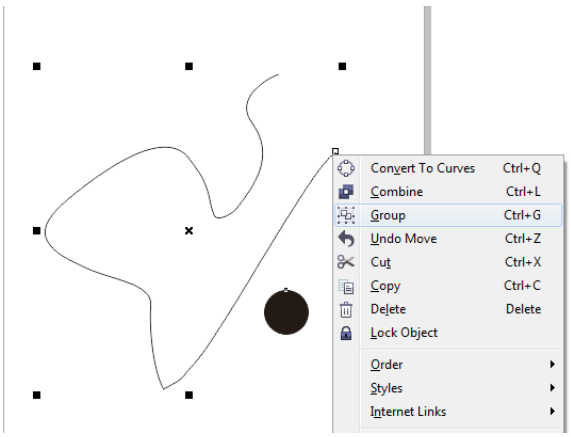
Basically, a scene graph is a graph, a collection of vertices and edges, creating a hierarchic relation between the vertices. Although sometime the graph can take the form of a tree, in most cases, the graph does not necessarily fulfill the condition of a tree, thus for convenience, it is referred as a graph (as a tree is always a graph). A vertex in a scene graph usually have many child, but often only one parent. When a process is applied to the parent vertex, it will be also applied to all the child of that vertex. This way, object processing can somewhat be simplified.

III.2 Scene Graphs in Graphics Editing Tools

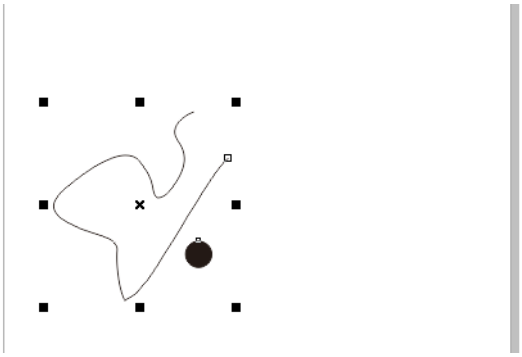
Graphic editing tools can mostly divided into two. The first one are vector-based graphic editing tools, like Adobe Illustrator or CorelDRAW. The second is raster-based editing tools, like Adobe Photoshop. Despite their differences in basic principle, scene graph can be applied to both.

Firstly, in vector-based graphics, usually, a vertex of the graph will represent a single unit or function in the overall graphics. This function can represent a single object, for example a triangle, a circle, or a bezier path. Giving a process to a vertex the graph means giving a process to the object.

What about the edges? Usually, in vector-based graphic editing, objects can be manipulated in groups. You can select a number of objects and group them. In the principle of graphs, this is the same as connecting a number of vertex. As a result, when a number of objects are grouped, the process applied to one objects will be applied to all other objects. The matter of ungrouping them is simply removing the edges between these vertices representing the objects.

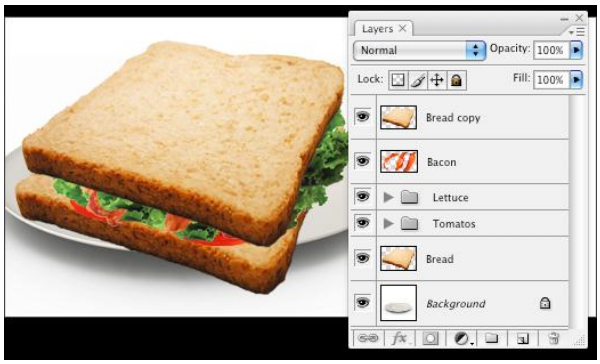


Picture III.1 Grouping in CorelDRAW X5



Picture III.2 Resizing a group of object

The case would be different with raster-oriented graphics editing tools. Most of the time, these tools would not be dealing with objects of the grouping of them. But, the concept that may ever-be-so familiar with raster graphic editing is layer. A layer acts like a sheet of paper, where you can put objects and edit it. On the graph principle, this layer acts as a vertex. Selecting two layers as one connects them with the edge, and so on.



Picture III.3 An example of layer usage in Adobe Photoshop[4]

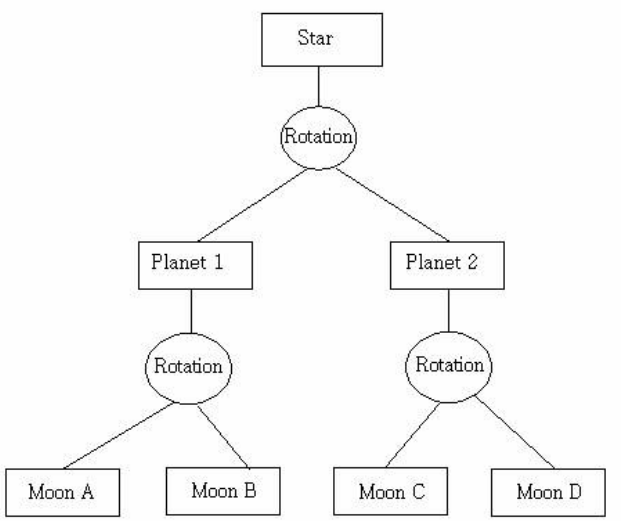
Basically, the structural shape between groups and layers are the same, but it may be sometimes useful to divide them, when you're making a tool that handle both layer and groups. In this case, you can make a different subclass, layers and groups from the class vertex.

III.3 Scene Graph in Games and 3D Application

Scene graph are greatly useful in the modelling of a 3D scenery, especially in bigger scale where the number of objects are very large, making it harder to apply process to them one-by-one. In most 3D graphics application, vertex in the scene graph generally represent an object or an entitiy in the overall scenery.

For an example, a racing game with drivers and cars. In this case, the vertices of the graph are 'drivers' and 'cars'. When the driver get into the car, the vertices are then connected with an edge, and the driver would be treated as an extension to the car. So, for example if the car crashed and set aflame, the driver, who is an extension of the car would be set aflame also.

Scene graphs is especially useful in modeling big 3D environment, essentially in the rendering process. Now let's take a look at a simple scene graph, and the process done to render it.



Picture III.4 An example of a scene graph depicting a simple universe with a star at its center[5]

The rendering process will start at the top vertex. The star will be rendered as the center of the universe. The vertex rotation below it consists of a rotation matrix that will be applied to the star, and all things rendered after it. After finishing the rotation rendering, Planet 1 will be rendered (already having the rotation quality). Now more than rotating with the star, the planet will have an extra rotation on its own, around its own orbit. After the rotation of the planet is rendered, Moon A and B are drawn, having the rotation of the star and also Planet 1.

Finishing the subgraph of Planet 1, the iteration will then move to Planet 2. The exact same process will happen, only different in the way that the rotation of Planet 2 is different with Planet 1, and the moons will be Moon C and D. After the rendering of Moon D is finished, the overall rendering is finished.

As we can see, the graph quickens the rendering process significantly, if we compare it to singlehandedly rendering every objects then applying process to that object without managing them hierarchically.

III.4. Implementation of Scene Graphs

There can be a lot of variation on the data structure used in implementing the scene graph on the source code. The most simple being the linked list. Processing the vertex basically consist of linearly iterating every cell in the list. This can be convenient for a small scale scene graph.

In bigger scale however, linear iterations can take a lot of time, so a more complex data structure is needed. For this need, usually the data structure used is a tree, where a

higher vertices is referred as group nodes, and the lowest level vertices is referred as leaf nodes. Leaf nodes usually represents objects that are being rendered (in the case of 3D rendering), while group nodes are the transformation matrix that will be applied to the leaf nodes. When a leaf node have more than one parent, the transformation that it undergo will be the combination of all its parents.

IV. Conclusion

In conclusion, scene graph is one of the more popular and practical usage of the graph theory nowadays. In graphic editing, graphs can represent group of objects or layer to further ease the editing of a graphical image. Whereas in the field of 3D modelling, a good scene graph design can significantly minimize the effort taken to render a 3D scenery, especially large scale 3D sceneries.

References

[1] Biggs, N.; Lloyd, E. and Wilson, R. (1986), *Graph Theory, 1736-1936*, Oxford University Press

[2] Trudeau, Richard J. (1993). *Introduction to Graph Theory* (Corrected, enlarged republication. ed.). New York: Dover Pub.. pp. 19. ISBN 978-0-486-67870-2. Retrieved 8 August 2012. "A graph is an object consisting of two sets called its *vertex set* and its *edge set*."

[3] Walsh, Aaron E. *Understanding Scene Graph***.** December 17, 2012 (20.00) < http://www.drdobbs.com/jvm/understanding-scene-graphs/184405094>

[4] December 17, 2012. (20.30) <http://dp.hightechhigh.org/~mmctighe/photoshop4kids/layering.html>

[5] Foster, Garett. *Understanding and Implementing Scene Graphs*. December 17, 2012 (21.00). < http://archive.gamedev.net/archive/reference/programming/features/scenegraph/index.html>

[6] Munir, Rinaldi. 2008. *Diktat Kuliah IF 2091 Struktur Diskrit* 4th ed. Program Studi Teknik Informatika STEI ITB.

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 29 April 2010

Ttd

Hafizh Adi Prasetya/13511092