A customizable engine for data and information visualization

By

Matthew Kampschmidt

Project submitted in partial fulfillment of the requirements for the degree of Master of Science in Information Technology

Rochester Institute of Technology

B. Thomas Golisano College of Computing and Information Sciences

Thursday, November 22, 2007

Statement of Problem

The world of information visualization is a rich and varied one. Many projects have demonstrated unique, sometimes useful, and sometimes beautiful ways of displaying information digitally. Despite this wide variety of ways of displaying data, the personal computer still relies on the same paradigms for displaying digital info. One example of these paradigms is visible in the current model computers use to display and interact with files. Today's personal computers rely on what is known as the WIMP interaction model: a model distinguished by windows, icons, menus and pointers (Johnson 1999). These WIMP interfaces (Figure 1) present users with an organizational metaphor. This metaphor organizes files into artificial groupings known as folders, each of which might be nested in other folders. These files are opened using an incredible number of applications, each of which can open and edit a subset of the files. Between the WIMP interface and this variety of file viewers, the user has little chance to see how information scattered across the system is related. Only in more recent years has the world begun to see some interlinking between different applications and new ways in which to visualize these files. The problem is that the aforementioned model was designed in an era when there were relatively few files to deal with. Today's computers are bursting with information. People are increasingly turning to new organizational tools and methods to make sense of the madness that is the personal computer environment. This Masters Project will take the information represented by these files as well as other sources of data and display them in a new and possibly more useful way.

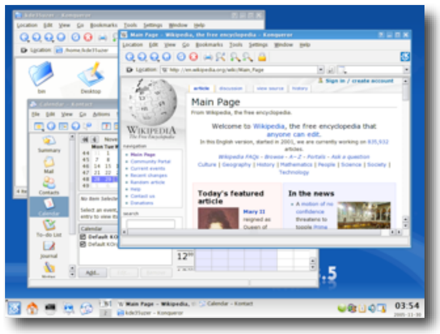


Figure 1: A screenshot of the KDE window manager and its implementation of the WIMP model.

Significance of the Problem

Recent innovations in software and hardware in the computer world are allowing users, designers, and programmers increasing freedom in the ways that they can deal with information and visualization on the computer. Processing hardware is capable of supporting a far greater load than in the past, memory can hold more data in active storage at a given

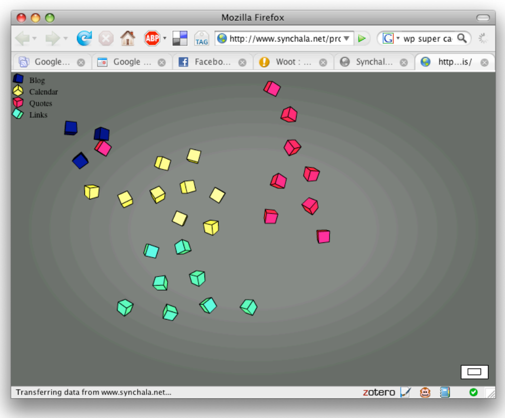
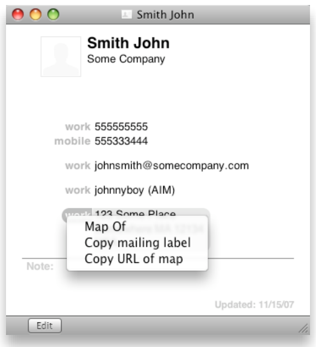

Mozilla Firefox
http://www.synchala.net/pro
wp super ca
Blog
Calendar
Quotes
Links
Transferring data from www.synchala.net...
zotero

Figure 2: A demonstration of a 3D clustering data visualization rendered on the fly.

moment, and graphics hardware can create live 3D renderings of extraordinary complexity on the fly (Figure 2). New software tools allow designers and programmers to work together to create interfaces that better resemble real life or demonstrate unique ideas. In some cases these tools allow developers to give users an opportunity to define for themselves how their information is organized and presented. Unfortunately much of this visualization is static and lifeless despite the power of the software and hardware behind it. When glancing at a date book or folder, users have only a few ways to distinguish between what is and isn't important. Therefore, the purpose of this project is to create a customizable system capable of displaying multiple types of information. It will employ a 3D metaphor and will use movement behavior, color, size, and shape to differentiate and draw attention to these files, folders, and ideas. It will be a Rich Internet Application (RIA) that can access a variety of local and remote data sources to provide information for the environment and link the information together in new and unique ways (Adobe – Business, 2007). This project is not intended to be a complete solution to the digital visualization problem, but rather a customizable environment within which others can create ways to visualize their data. For this reason, the project will incorporate approaches for users to customize the appearance and information sources that the application displays.

Creating a tool like this is important for a variety of reasons. First and foremost is, as previously stated, to deal with the "info glut" of our world. There are several factors that influence the severity of this problem. In the case of information types that stand by themselves the problem is simple: large quantities of information are hard for humans to sort through, largely because the human mind is better at dealing with small chunks of information at any given moment (Ross, 2006). By applying different visual metaphors to this data a user can perhaps find information more efficiently than if they were to simply read through a textual listing. The WIMP model is one such visual metaphor, but as previously stated it is more effective with small sets of files. When the data a user is sifting through is not simply a listing but branches out (as with hierarchical information) the problem of data sifting grows. This type of info might be extensively forked and the user must not only sort through a single list, but may have to backtrack to get to other pieces of the information being displayed. This can cause the person a delay in processing time, thus reducing the effectiveness of the search (Hick, 1952). With a three dimensional visualization it might be possible to provide peripheral visual cues to help people understand and remember what pieces of the data they have examined.

Perhaps the most important complication in today's data-rich world is the interrelatedness of data. Current methods for accessing information provide limited ways of viewing and accessing related types of information. For example, an address book might allow the user to click on a persons address and get a map of that location (Figure 3). Users might be better served if they were able to see not just the attendee's address book record, but perhaps a visual representation of people, maps, and events related to what they are viewing.


Smith John

Smith John
Some Company

work 555555555
mobile 555333444

work johnsmith@somecompany.com

work johnnyboy (AIM)

work 123 Some Place

Note:

Updated: 11/15/07

Figure 3: An example of an address book application allowing the user to pull up contextual data, e.g. maps.

These issues all contribute to the "info glut" situation in which the world finds itself. Solving them in a creative way could exponentially increase the effectiveness of the world's computer-using population.

Literature Search

The first step in information visualization is getting information into the application. While it is certainly possible to provide static data to a visualizer it is far more useful, and it is the goal of this project, to provide a visualizer with a dynamically generated data model drawn from a variety of sources. This is however a very complex problem. Practitioners in the field of information visualization have been struggling with solutions to this issue for some time. One researcher, Chaomei Chen, has written a book entitled "Information Visualization – Beyond the Horizon", which contains a chapter on this exact subject entitled "Extracting Salient Structures" (Chen, 2006). This chapter discusses various algorithms for analyzing the contents of files and for taking information about those files and then “clustering and classifying” them. This book will be useful in developing modules for this application to generate dynamic data models to visualize.

Once you have data to visualize you must find ways to animate it. The work of Craig Reynolds on what he calls “boids” (simulated creatures) is a seminal piece of research, which is now widely quoted and utilized throughout the computer industry for gaming, visualization, animation, and more (Reynolds, 1987). His work, “Flocks, Herds, and Schools: A Distributed Behavioral Model”, published in the conference proceedings of SIGGRAPH 1987, discusses how to simulate real world flocking behaviors in a simulated environment. This work includes information about the different variables involved in such flocking, as well as many uses of flocking behaviors in computer simulations. This work will be useful to this capstone project in that it will provide an excellent starting point for the animation of groups of data items in visually appealing and useful ways. For example, if a given data item were more important than other data items it might more heavily influence the overall behavior of the group. In this situation the user might then notice the cluster and have their attention drawn to the more important item.

Another way to draw and focus the attention of the user is through the use of color. While the aim of this project is not to provide a conclusive solution to the use of color for peripheral attention capture, it will be useful to provide some effective defaults within the application. To this end, another resource for this project is the work on color known as “Interaction of Color” (Albers, 2006). This book contains the findings of a set of studies by Joseph Albers in the 1960’s on the interaction of color, it’s effects on the human mind, and how best to teach and learn these concepts. By employing some of the concepts illustrated in this book, the user will be able to better distinguish data items from each other, because each item becomes more distinct. These concepts will also be useful in determining ways to make certain items stand out temporarily in order to catch the users attention.

It is important not only to catch and focus the user’s attention, but also to allow them to interact with what they are being attracted to. Therefore it will be important to develop an interface that makes it as easy as possible to navigate and use the data they are given. Two very important rules of human computer interaction, Fitts’ Law [as extended for two dimensions] (Fitts, 1954) and Hick’s Law (Hick, 1952), will be essential to this task. These laws define, respectively, the amount of time it takes a user to select something with a pointing device and the amount of time it takes to make a choice given a set of choices. Others have extended the work of these researchers, such as in Card, Moran, and Newell's work on Human Computer Interaction (1983). These resources, as a group, will inform many of the decisions on how to structure the visual output of this application and how the end-user will interact with it.

Additionally, as this project is primarily a customizable visualization engine, a number of examples of former visualization projects will be examined in detail to inform the design and architecture of the application being developed. One such example will be the work of designer Ben Shneiderman. His book, "Designing the User Interface: Strategies for Effective Human-Computer Interaction" (Shneiderman, 1998), contains numerous examples of visualization projects and details on what made each useful. Another example is the article "Information visualization using 3D interactive animation" (Card et al, 1993). It contains information on prior work in the field of 3D interactive animation as well as examples of that work. This capstone will also explore some of the fieldwork done using software environments such as Processing (Exhibition, 2007) and Papervision3D (Demos, 2007). These are software environments that enable rapid development of a variety of visualizations and could yield some ideas about unique and useful visualization schemes.

All of these resources and more will be studied and the resultant information will be synthesized to assist in creating a new and useful engine for developers and end-users to visualize a variety of data types.

Plan of Work

This project will draw from a variety of fields. These fields include building relational data models, generation of behavioral effects based on relational data, resources on visual behavior patterns and styles as well as color theory, and the development of plug-in-based software. Therefore the first step in this project will be to do extensive research into these topics in addition to that already denoted in this paper. In order to build an effective application on top of the knowledge gleaned from research into the aforementioned fields it will be important to select an appropriate software suite. The prime candidates for this type of project are Adobe Flex (Adobe Flex, 2007) running in the Adobe Integrated Runtime

(Adobe Integrated Runtime, 2007) utilizing the Papervision3D API (Papervision3D, 2007).

Once there is a solid foundation on which to base decisions about the design of the engine it will be time begin to generate sketches and storyboards of the interface and interaction model. With these sketches and storyboards in hand it will be easier to keep development moving forward without too much feature bloat. The process of generating these sketches will also help to refine the purpose and capabilities of the engine specifications.

The nature of this application dictates that it be developed using a fairly strict Model/View/Controller (MVC) (Gamma et al, 1994) design pattern. There will be three very distinct modules within the application, and each lines up well with a part of an MVC pattern. The model portion of the application will entail a data input module, which will be designed with a plug-in architecture. The view component will be a 3D rendered visualization. It will pull information from the model and display it using internal rules about presentation. These rules will be augmented with resources supplied by visualizer plug-ins.

Once the initial sketches are complete, the next step will be to further refine the capabilities and limitations of the engine. This will be accomplished by developing and documenting a set of application programming interfaces (API's) which developers will use to extend the capabilities of the engine. These API's will dictate and constrain the types of data the application will be able to process as well as how they can relate to each other. This API for data input will need thought because any data input must be in a form that can be processed by the relational/behavioral parsers within the engine. The API's will also dictate the types of visual formatting plug-ins developers will be able to create as well as the resources they will be required to use. For example, if a developer wishes to represent a data type using a special 3D model, they would be required to provide that model, either built-in or over the Internet, as a resource in their plug-in in a very specific format. If the application were to use Papervision3D for it's 3D rendering engine, this format would be Collada (Collada, 2007). As much of the visualization will be predetermined within the engine, the API for visualization will likely mostly contain the ability to add new models to the visualizer and/or new preset color/behavior/model combinations. The documentation for these interfaces will contain a listing and description of a set of methods and functions that expose the data bindings of the engine as well as a set of required methods and functions for developers to implement in their plug-ins.

Once the data API's are designed, it will be necessary to actually implement and test them. This will require developing the necessary interfaces and classes that would include a data model manager as well as service managers. Once these classes are written, a test module will be created which will consist of one data plug-in and accompanying sample data as well as a simple output class to allow querying of the data model. The test module will then be run to verify that this module works as defined.

While the basic data manager will receive data in a relatively unstructured and unformatted configuration, the engine will require the data to be in a structured and linked configuration. This means that the application will need a data model manager to take the data from the original model and reorganize it into a relational structure. This will likely be a fairly difficult task and may require some extra research. Because the work required for this module may be outside the scope of this project, some of its functionality may scaffolded for revisiting later. The essential idea is that this portion of the application will use the basic information contained in the original data model and generate a second tier model that contains all of the original data items but each item will be assigned extra information such as hierarchy, related keywords, etc. This model will also contain placeholders for visualization information such as assigned behaviors and visual properties as well as accessors and mutators for those properties. These placeholders will be discussed in the following section. The manager for this model will contain methods and functions to allow the user to add, modify, and delete the data items contained therein as well as the properties of those data items. In order to verify that this module works, a test class will be written which will traverse this relational data model and trace out all data items for a given hierarchy level, all data items related to a given item, all items containing a certain keyword, etc.

As mentioned in the previous paragraph, the relational data model will have a placeholder for "assigned behaviors". A behavior manager module will generate these assigned behaviors. This manager will analyze the data items in the relational model and assign them behaviors based on their individual properties and their relationships to other data items. Some examples of such behaviors are: flocking, following, "jitter", bouncing, pulsing, etc. An interface class will define the basic type of behavior and it will be extensible to implement more complex behaviors. While users and developers will be able to customize these behavior using the interface and plug-ins (visual API), respectively, there will be preset assignments built in to this module. This module will also contain methods and functions to detect and utilize visualization plug-ins to provide new behaviors and visual models. To verify that behaviors are properly assigned, the test module for the previous relational module will be used to check that this module is functioning properly.

With all of the back end software now written, it will be necessary to turn to the visual output of the application. The first module for the visual output will be the visualization rendering component. This component will query the relational data model for data items and will utilize the behaviors and properties of each item to appropriately render them onscreen in an animated 3D environment. Each item will appear as dictated in the data model and will move around the screen in 3D space accordingly. To achieve these goals this component will use a rendering engine such as Papervision3D and will dynamically create 3D representative objects based on and linked to each data item. These objects will exhibit the stored behaviors and visual properties of the item and will interact with other items accordingly. Additionally, a user will be able to select each item and manipulate it to access the data stored within. This manipulation may be something as simple as clicking the object to launch a file viewer or as complex as accessing an item modification user interface to modify or even delete the item.

In order to control and navigate through this data, users will need some sort of simple but effective interface that can search, sort, and traverse the data within this visualization. To this end the next step will be to implement an overlay user interface and interaction model. This interface will enable the user to do tasks such as filtering the displayed data items, causing them to sort themselves differently, or possibly allowing the user to drill down from one hierarchical level of data to another. If time permits, one additional module will be developed. This module will enable the user to select an item or set of items and alter their default behaviors and relationships using an onscreen user interface.

The last major steps will be debugging and documentation. The application will be presented to a small group and they will be asked to attempt to “break” the application. They will be given a simple set of instructions and asked to find certain pieces of information. If any major issues are discovered they will be corrected and submitted for another round of testing. Once this iterative process is complete and the application is deemed finished, the documentation from the project will be compiled and organized into a final capstone report along with published deliverables.

Timeline

November 26 - December 10

1) Research
2) Draw sketches of user interface and storyboard basic interaction models

December 11 - 23

3) Develop and document API's
4) Implement data API's

January 2 - January 14

5) Design and develop relational data manager
6) Design and develop behavior manager

January 15 - 23

7) Design and implement a visualizer component

January 24 - 30

8) Design and implement a basic user interface

January 31 - February 6

9) Design and implement data model modification UI

February 7 - 20

10) Debug entire application and review systems abilities

February 21 - 27

11) Formalize documentation of API's and application user interface

February 28 - Mar 1

12) Prepare and deliver capstone defense

Deliverables

- A complete information visualization application including:
  - A rich internet application capable of accepting data and displaying it in a 3D semantic visualization

    - A set of plug-ins for importation and visualization of some local resource (e.g. local file system) and some remote resource (e.g. Amazon item search results, Flickr images, Google API)
- Source code and assets
- Process blog
- A documented API for creation of data and visualization plug-ins
- A final capstone report

Final Requirements

In order to keep the project on track and not allow feature overflow, a set of conditions for completion have been predetermined. The initial work will involve implementing a 3D visualization environment using the Papervision3D rendering engine in Flex. This environment will be a plug-in based application with a preset API allowing specific forms of input and output. This API will limit the ability to incrementally alter the capabilities of the environment without rewriting the API. During the completion of this basic environment, visualization and data import plug-ins will be written that allow for display of a variety of types of data. The data import plug-ins will include: a local file-system access plug-in capable of traversing a computer or servers file-system, and a plug-in to access some type of online resource such as Amazon.com product information or the Google API for retrieving email, events, and address book entries. The project will be finished once the application is capable of retrieving a data set from these plug-ins, correlating that data into a relational data model, assigning behaviors based on those relationships and on inherent properties, and displaying it for the user to search and navigate as defined by internal rules of the engine and visualization plug-ins.

Works Cited

Adobe - Business : Rich Internet Applications. Retrieved November 22, 2007, from http://www.adobe.com/resources/business/rich_internet_apps/

Adobe Flex - Wikipedia, the free encyclopedia. Retrieved November 16, 2007, from http://en.wikipedia.org/wiki/Adobe_Flex

Adobe Integrated Runtime - Wikipedia, the free encyclopedia. Retrieved November 16, 2007, from http://en.wikipedia.org/wiki/Adobe_integrated_runtime

Albers, J. (1975). *Interaction of Color*. Yale University Press.

Card, S. K., Newell, A., & Moran, T. P. (1983). *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates, Inc.

Card, S. K., Mackinlay, J. D., & Robertson, G. G. (1993). Information visualization using 3D interactive animation. *Commun. ACM*, *36*(4), 57-71.

Chen, C. (2006). *Information Visualization: Beyond the Horizon*. Springer.

Collada. Retrieved November 22, 2007, from http://www.collada.org/mediawiki/index.php/Main_Page

Demos « Papervision3D. Retrieved November 16, 2007, from http://blog.papervision3d.org/category/demos/

Exhibition \ Processing 1.0 (BETA). Retrieved November 16, 2007, from http://processing.org/exhibition/index.html

Fitts, P. M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. , 381-91.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. M. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional.

Hick, W. E. (1952). On the rate of gain of information. *Quarterly Journal of Experimental Psychology*, (4), 11-26.

Hyman, R. (1953). Stimulus information as a determinant of reaction time. *Journal of experimental psychology*, *45*(3), 188-96.

Johnson, S. (1997). *Interface Culture: How New Technology Transforms the Way We Create and Communicate*. Harper San Francisco.

Papervision3D. Retrieved November 16, 2007, from http://blog.papervision3d.org/

Reynolds, C. W. (1987). Flocks, Herds, and Schools: A Distributed Behavioral Model. *Computer Graphics*, *21*(4), 25-34. Retrieved November 22, 2007, from http://http://www.red3d.com/cwr/papers/1987/boids.html

Ross, P. (2006). The Expert Mind: Scientific American. *Scientific American*. Retrieved November 22, 2007, from http://www.sciam.com/article.cfm?chanID=sa006&colID=1&articleID=00010347-101C-14C1-8F9E83414B7F4945

Shneiderman, B. (1998). *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison Wesley.

Additional Resources

Card, S. K., Mackinlay, J., & Shneiderman, B. (1999). Readings in Information Visualization: Using Vision to Think. Morgan Kaufmann.

Deller, M., Ebert, A., Bender, M., Agne, S., & Barthel, H. (2007). Preattentive visualization of information relevance. Proceedings of the international workshop on Human-centered multimedia, 47-56.

Gershon, N., Eick, S. G., & Card, S. (1998). Information visualization. interactions, 5(2), 9-15.

Judelman, G. (2004). Aesthetics and inspiration for visualization design: bridging the gap between art and science. Information Visualisation, 2004. IV 2004. Proceedings. Eighth International Conference on, 245-250.

Lau, A., & Moere, A. V. (2007). Towards a Model of Information Aesthetics in Information Visualization. Proceedings of the 11th International Conference Information Visualization, 87-92.

Moere, A. V. (2004). Time-Varying Data Visualization Using Information Flocking Boids. Proceedings of the IEEE Symposium on Information Visualization (INFOVIS'04) - Volume 00, 97-104.

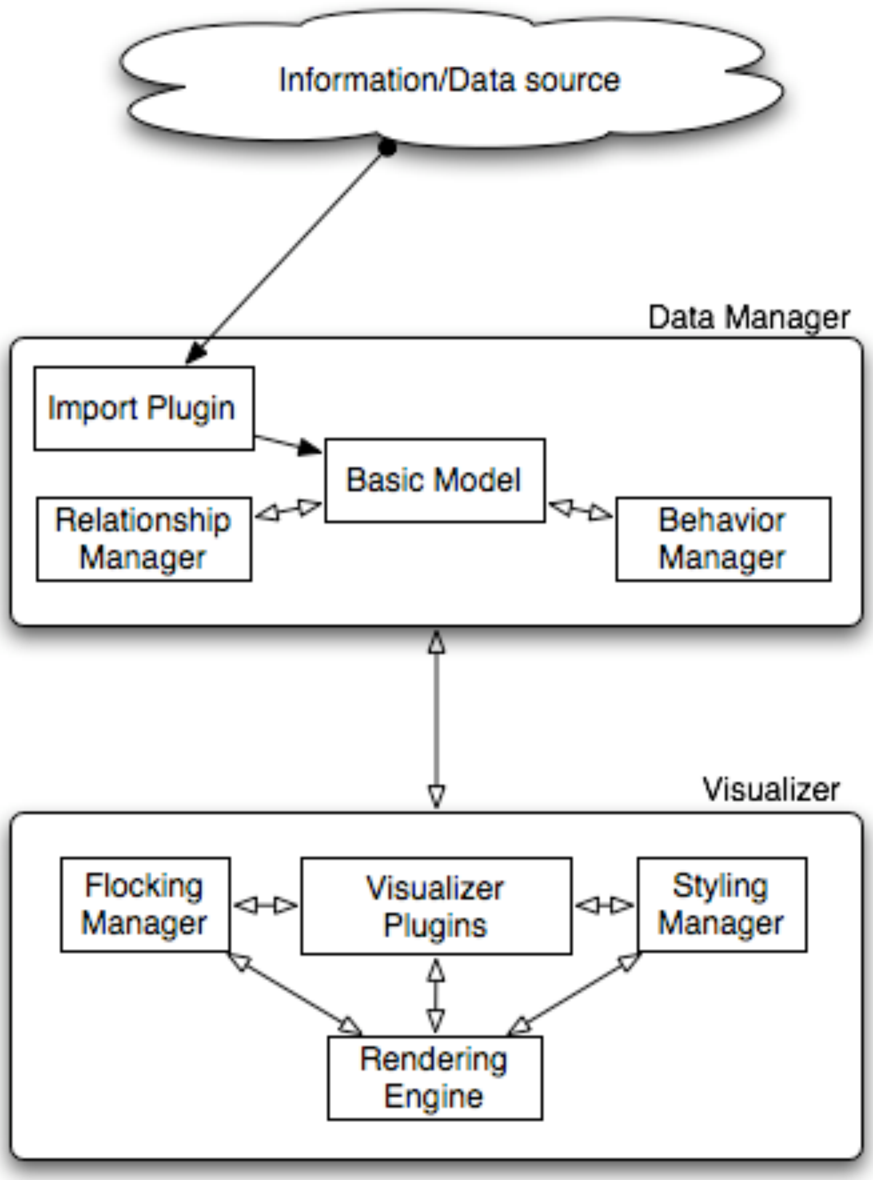
Raskin, J. (2000). The Humane Interface: New Directions for Designing Interactive Systems. Addison-Wesley Professional.

Thissen, F. (2003). Screen Design Manual: Communicating Effectively Through Multimedia. Springer.

Tufte, E. R. (1990). Envisioning Information. Graphics Press.

Addendum

Application Flow Diagram


Information/Data source
Data Manager
Import Plugin
Basic Model
Relationship Manager
Behavior Manager
Visualizer
Flocking Manager
Visualizer Plugins
Styling Manager
Rendering Engine