Z

Zero-One Laws

Nicole Schweikardt
Johann Wolfgang Goethe-University Frankfurt am Main, Frankfurt, Germany

Definition

A query language is said to have the *0-1 law* if every Boolean query that contains no constants (i.e., the query does not mention any particular element from the domain of potential data values) is almost surely true or almost surely false. The notions of being "almost surely true," respectively, "almost surely false" are defined as follows: Let σ be a fixed database schema. For each natural number n, let $DB_n(\sigma)$ be the set of all database instances of schema σ whose active domain is a subset of $\{1,...,n\}$ (i.e., all database entries belong to $\{1,...,n\}$). For a Boolean query q of schema σ let $\mu_n(q)$ be the probability that a database D chosen uniformly at random from $DB_n(\sigma)$ is a "yes"-instance of query q. In other words, $\mu_n(q)$ is the number of databases in $DB_n(\sigma)$ on which q evaluates to "yes," divided by the number of all databases in $DB_n(\sigma)$. Query q is said to be *almost surely true* (respectively, *almost surely false*), if the limit $\mu(q) := \lim_{n\to\infty}\mu_n(q)$ exists and is equal to 1 (respectively, 0).

Key Points

0-1 laws can be used as a tool for proving expressivity bounds for query languages: If q is a Boolean query for which the limit $\mu(q) = \lim_{n\to\infty}\mu_n(q)$ either does not exist or is different from 0 and 1, then q cannot be expressed by any query language that has the 0-1 law.

For example, let σ be the schema consisting of one binary relation symbol E, and let q_{even} be the query "Does the given database contain an even number of tuples?" It is not difficult to see that the limit $\mu(q_{even}) = \lim_{n\to\infty}\mu_n(q_{even})$ exists and is equal to $1/2$. Thus, query q_{even} is neither "almost surely true" nor "almost surely false" and hence cannot be expressed by a query language that has the 0-1 law.

Historically, the first query language for which a 0-1 law was proven was the relational calculus (i.e., first-order logic). The proof also shows that there exists an algorithm which, given a Boolean relational calculus query q, computes $\mu(q)$. Since then, 0-1 laws have been shown for many different query languages, among them fixed point logic, infinitary logic $L^{\omega}_{\infty\omega}$, and various fragments of second-order logic. Variants of the 0-1 law are also known for several other probability measures and for restrictions to particular classes of databases.

A common method for proving that a query language has the 0-1 law is based on so-called *extension axioms* and an *Ehrenfeucht-Fraïssé game* argument. 0-1 laws are also closely related to the theory of the *countable random graph*. For an overview of results and proof techniques, refer to the textbooks [1,2,4] and the survey [3].

To point out the limitations of the use of 0-1 laws for proving inexpressibility results, it should be noted that there do exist queries that are almost surely true but nevertheless are not expressible in the relational calculus (an example is the query q_{conn}: "Does the given database relation E form a connected graph?").

Furthermore, several query languages are known *not* to have the 0-1 law, e.g., existential second-order logic and monadic second-order logic (cf. [1,3,4]).

Cross-references

► Expressive Power of Query Languages

Recommended Reading

1. Ebbinghaus H.-D. and Flum J. Finite Model Theory, 2nd edn. Springer, Berlin, 1999.
2. Hodges W. Model Theory. Cambridge University Press, Cambridge, New York, USA, 1993.
3. Kolaitis P. and Vardi M.Y. 0-1 laws for fragments of existential second-order logic: a survey. In Proc. 25th Int. Symp. on Mathematical Foundations of Computer Science, 2000, pp. 84–98.
4. Libkin L. Elements of Finite Model Theory. Springer, New York, NY, USA, 2004.

ZF-Expression

► Comprehensions

Zoning

► Storage Security

Zoomable User Interface (ZUI)

► Zooming Techniques

Zooming Techniques

Harald Reiterer[1], Thorsten Büring[2]
[1]University of Konstanz, Konstanz, Germany
[2]Ludwig-Maximilians-University Munich, Munich, Germany

Synonyms

Zoomable user interface (ZUI); Multiscale interface; Scaling

Definition

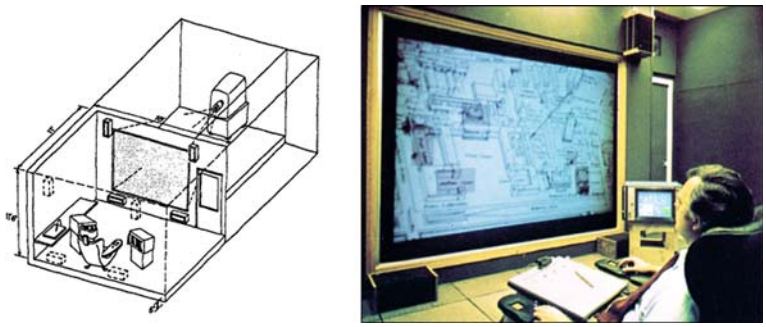
Zooming facilitates data presentation on limited screen real-estate by allowing the users to alter the scale of the viewport such that it shows a decreasing fraction of the information space with an increasing magnification. Hence the system may first present a global overview of the information space for the benefit of orientation, and in a second step, the users can then dynamically re-allocate the screen space based on the information objects they are interested in. A navigation technique commonly used in conjunction with zooming is *panning*: a movement of the viewport over the information space at a constant scale.

Historical Background

The first application to use zooming as a fundamental interface approach was the Spatial Data Management System (SDMS) [5] in 1978 (see Fig. 1). The SDMS system relied heavily on custom hardware, including an octophonic sound system and an instrumented chair equipped with pressure-sensitive joysticks, two touch-sensitive tablets and a digital lapboard. The data was presented via a rear-projected color television display. The flat information space was called Dataland. It presented the content of a database with the help of tiny pictures of faces, maps, television sets, letters, book covers, a calendar, a telephone, and a calculator. Items of similar sort have been grouped together on distinctive color backgrounds. SDMS enabled users to manage and zoom into the visual database representation with the help of a joystick, by touch commands, or by voice. This early attempt of a zoomable interface included a variety of elementary design concepts: a "what you see is what you get" representation of different multimedia data types with the help of icons, a spatial arrangement depending on the users' choice or automatic by semantic (e.g., clustering of related data), the possibility to fly over Dataland in a helicopter-like fashion (pan), and the possibility to change the granularity of the information presentation by zooming in and out.

The Computer Corporation of America (CCA) of Cambridge, Massachusetts built a "field version" of the SDMS. It consisted of three color TV monitors lined up on a table-top with a joystick, a keyboard, and a graphic tablet [9]. The system also offered a scalable data surface with the enhanced possibility to zoom in a semantic way. Therefore each data item on the surface was stored at several levels of detail. Zooming in on such a data item caused the more detailed version to appear (e.g., shape of a ship first, and then shape with further text, and finally a full picture of the ship with all details). Another important new concept was the idea of ports. There was not only one single data surface but an entire set of them. The transition points between them were called ports and shown as special pictures (icons). When zooming in on them, these ports acted as trapdoors down into other information spaces. The result was a hierarchical set of information spaces through which the user could navigate. Ports allowed presenting alternate views of the same data items, and could also be used to activate programs external to SDMS (e.g., electronic mail, text editors).

In 1993, a zoomable user interface (ZUI) called Pad [15] was developed. It introduced mostly the same fundamental design concepts for zooming as SDMS,



Zooming Techniques. Figure 1. The Spatial Data Management System [5].

but the main difference was its ability to run on conventional PCs or Minicomputers. Pad aimed to provide an alternative to the Windows paradigm. The system visualized an infinite two dimensional information plane populated by objects that users could interact with. Such Pad objects could, for instance, be text files, a clock program, or a personal calendar. Each of these entities occupied a well-defined region on the Pad surface and was visualized by means of graphics and portals. Portals showed portions of the Pad surface at different scales, and may also look recursively at other portals. One way to use a portal, for instance, would be to show a miniature overview of a large Pad object. Users can manipulate the view's scale and data representation by performing semantic zoom operations. Another important concept is portal filters that transform data into other complex views, e.g., present tabular data as a bar chart.

Only one year later, the successor to Pad, Pad++ [3], was presented, a system that also constituted the first ZUI toolkit. Pad++ aimed to serve as the basis for exploration of novel interfaces for information visualization by providing a framework for simplifying the creation of multiscale applications. It introduced some, mostly technical, enhancements over the original Pad implementation. For instance, much effort had been devoted to realizing smooth semantic zooming, even with hundreds of thousands of objects loaded into the information space. To achieve this, the rendering with Pad++ followed a "parallel lazy loading" strategy, i.e., only the portion of the database that is currently visible is loaded. One important design objective of Pad++ was to support a wide range of platforms ranging from high-end workstations to PDAs and Set-top boxes. However, an increased level of platform independency was only achieved by later ZUI toolkits such as Jazz (2000) [4] and Piccolo (2004) [2].

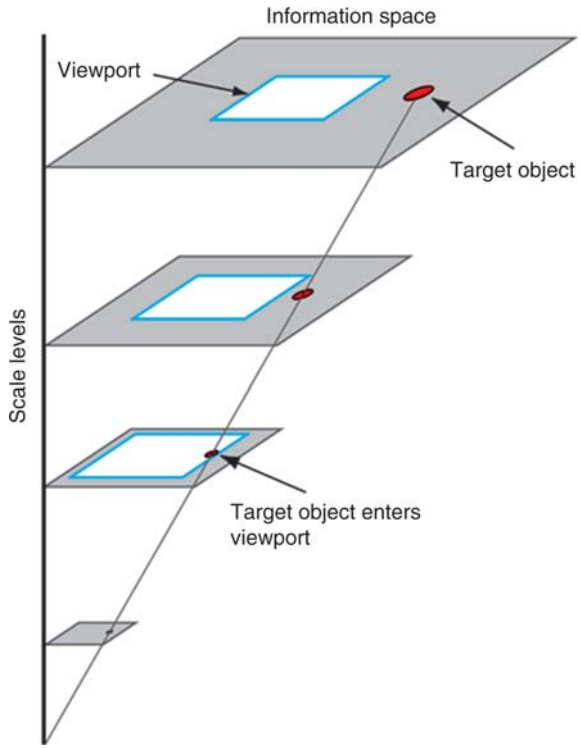
Foundations

A method of illustrating zooming is that of space-scale diagrams [7]. Figure 2 models a ZUI in which the 2D information space is shown at different magnification levels and aligned by the vertical axis representing scale. The inner rectangular outline represents the viewport, i.e., the portion of the information space that is visible. The outer gray area is the off-screen space. In the diagrammatic example, users are searching for the red target object that is off-screen (top-most scale level). Instead of panning the view around, as in the scrolling interface, users zoom out until the target object enters the viewport. In a second step, they can then access the object's details by zooming back in.

Navigation in Information Spaces

In contrast to scrolling interfaces, which are only effective for small spaces, ZUIs develop their full potential as the size of the information space grows. Even if users know the precise location of an off-screen target, in most cases a pan operation would still be a slow way of navigating. Panning only covers distance at a constant pace, while zooming allows users to view off-screen content in a non-linear fashion. This advantage is due to the special properties of multiscale interfaces, in which the shortest path between two points is usually not a straight line, but a combination of zoom and pan operations.

A common problem with ZUIs is the lack of context. Even after a short period of navigation, users lose their



Zooming Techniques. Figure 2. Space-scale diagram [7].

overview due to the continuous clipping of orientation cues during zooming. The most straightforward way to rediscover context in ZUIs is to zoom out. While this approach may help users to generate or refresh their internal model of the information space, it also implies frequent interaction that, after some time, users are likely to find tedious. Especially so in cases in which they have to zoom out extensively to regain context. While this problem seems to be inherent to ZUIs, a strategy that at least reduces the burden of frequent zooming is to provide a fast and precise interaction design. The less time-consuming and cognitively demanding the zooming is, the less it will annoy users.

A more severe type of orientation problem that cannot be solved solely by interaction has been termed desert fog [12]. Desert fog describes a condition in which users zoom into the white space between objects up to the point where the viewport goes completely blank. On the one hand, the empty screen could be indicating that there are no objects to be found in that direction, in which case users need to zoom out. Another possibility, though, is that there are indeed objects, but they are still too far away to be visible. In this case, users need to zoom in further to approach these objects. A solution to ease such desert-fog conditions is to provide navigational assistance by generating multiscale residues for all objects in view. Such landmarks are drawn across scale and indicate that a particular object exists in that direction. If no residues are seen, the users know that they have to zoom out to find other objects. To avoid visual clutter, navigational information should be clustered.

Interaction Techniques

Different interaction techniques for zooming can be distinguished. Earlier systems were often limited to centralized zooming. Users click a device (or onscreen) button to increase the scale and another button to decrease it. Since the view is only scaled and not translated, this approach implies that for targeting information objects in space, users first have to move the object to the center of the screen. Furthermore, for large information spaces, users may frequently have to interrupt the scaling operation to readjust the focus. A more elegant and effective scaling technique is point-directed zooming, which allows users to magnify and center information objects at the same time. One example is the default event handler in Jazz, and now in the Piccolo framework. While pressing the right mouse button, users scale the view by dragging the mouse right or left to zoom in or out, respectively. The zoom speed increases with the distance the mouse is dragged, and vice versa. During the operation, the point that the cursor was over when the dragging started moves to the center. Another interaction technique, which assumes that users want to return to the highest magnification level after each zoom operation, is speed-dependent automatic zooming (SDAZ) [10]. SDAZ couples rate-based scrolling with scaling in a single operation and was developed to avoid visual blur when navigating large information spaces at high speed. The users control the velocity of a continuous panning operation by dragging the pointing device. To keep the visual flow constant, the system automatically zooms out when the scrolling speed increases and zooms back in when the scrolling speed decreases.

Zoom interaction design may also be based on visual mediators such as sliders. A slider widget has the advantage that the user is provided with additional visual feedback. The position of the slider thumb

reveals the current zoom level of the view in relation to the range of scale factors available (e.g., the slider in Google Earth). Users can drag the control to increase or decrease the zoom level. Another example of a visual mediator is zoom bars [11]. Equipped with three thumbs, a zoom bar is a slider that controls the range boundaries of an axis dimension. Moving the two extreme thumbs, users can increase or decrease the upper and lower range boundary, causing a zoom in or a zoom out by changing the scale on the corresponding display axis. That way, any rectangular region can be enlarged to full diagram size. This is also the reason why zoom bars are usually limited to abstract information spaces, in which the two dimensions do not require a fixed aspect ratio. While their similarity to a regular scrollbar may support first-time users in operating the widget, a drawback is that the usability of a zoom bar deteriorates with a decreasing ratio of the physical slider size and the attribute range of the related dimension.

A similar zoom effect as with zoom bars, but without occluding the view, can be achieved with a bounding-box tool. Users drag the pointing device over the view and a focus rectangle is drawn that takes the drag starting point as a corner location and the drag distance as a diagonal. When the users release the pointing device, the defined region is magnified to fit the full size of the view. In cases such as images and text, in which a constant aspect ratio is desired, the focus region is only scaled but not distorted. A bounding-box is a visual mediator that does not require permanent display space, and thus lends itself to the application on small screens such as featured by mobile devices.

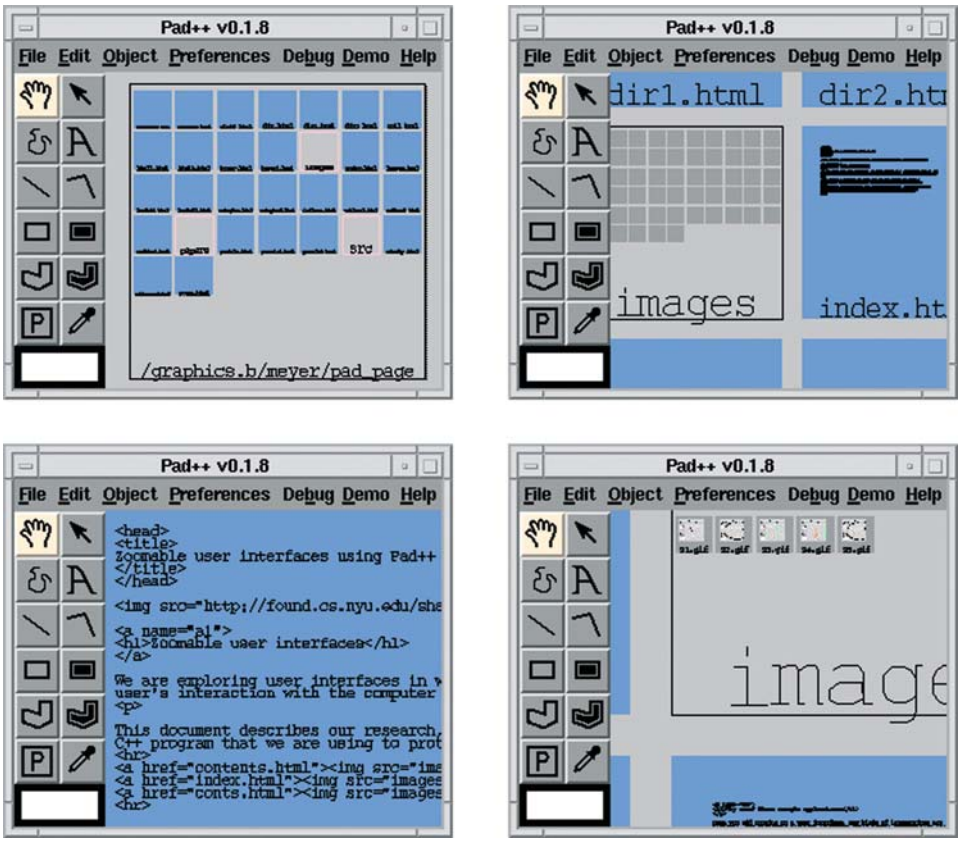
Zoom Granularity and Manipulation

Apart from the interaction design ZUIs can further be categorized by the granularity of zooming they provide. The most basic ZUI is the two-level zoom. It lets users switch between a single overview and a single detail zoom level, and thus only works for small information spaces. To manage larger data sets, several intermediate zoom levels between the minimum and maximum scale must be introduced. In older systems such as, for instance, Pad [15], navigation between the levels is accomplished by discrete jumps. This approach is easy to implement and computationally very effective, but it hampers the usability of the system. Coarse jumps can irritate and disorient the users and thus may hinder the cognitive and perceptual processing required for navigation. Accordingly, a lot of effort has been put into equipping multiscale interfaces with smooth continuous zooming.

ZUIs can offer different types of zooming. Most common is geometric zoom, in which objects are simply magnified. Zooming in, the object's size increases, and vice versa. This approach is found in many standard software applications such as PDF readers or image editors. Semantic zooming, in contrast, is a more sophisticated concept, in which objects change their appearance as the amount of screen real estate available to them changes. In the Pad++ -based directory browser shown in Fig. 3, for instance, subfolders and files are first represented by small-sized icons that only show the name of the object. Increasing the scale, the icons change their appearance to present some more detailed information, e.g., the number of images in a folder, or the structure and amount of text contained in a document. Zooming in further, images become visible and text is magnified to a readable size. At this level, users may also be provided with additional functionality to manipulate the object in focus. Overall, the goal of semantic zooming can be summarized as providing the users with the most meaningful object representation at each magnification level. The difficulty with this approach is that the appropriate representations for all scale levels must be determined in advance by an expert user. However, for complex objects, several representations may be suitable for a given portion of display size. In this case, it is hard to reliably predict the users' requirements. Systems such as DataSplash [14] try to overcome this problem by enabling users to visually program how objects behave during zooming.

Transition Between Zooming

Another important concept for ZUIs is animated transitions. Users may, for instance, click on a hyperlink to automatically move the viewport to a remote location. Or they initiate a scale manipulation via a bounding-box. In both cases the viewport needs to be adjusted. The transition between these interface states can either be instantaneous, which is fastest, or it can be animated by showing intermediate frames. The benefit of smooth and animated transitions is that they help users to maintain relations between



Zooming Techniques. Figure 3. A directory browser featuring semantic zooming (http://www.cs.umd.edu./hcil/pad++).

application states. Users are not required to consciously make connections between the changes of interface content and thus they can stay focused on the task. Smooth transitions were also found to have a positive effect on the users' ability to build a mental map of the information space. In a study [1], users were asked to navigate a virtual family tree, in which each node showed a picture of a family member and hyperlinks to connected nodes. Only one or two nodes could be seen at a time. To move the viewport, users clicked on the hyperlinks. The authors discovered that animated transitions of 1 second improved the users' ability to reconstruct the information space, with no penalty on task performance time. To determine the optimal trajectory between two locations in a multiscale interface, some prior research has investigated how to calculate the shortest path with zooming and panning [7], or the path that specifically supports smooth animations [16].

Key Applications

The most common application domain to incorporate zooming techniques is geographical information system, e.g., Google Earth. However, an increasing amount of research prototypes use ZUIs for presenting data without inherent spatial relations. Example domains include web, image, document and database browsers, browsing history widgets, slide show programs, thought organizers, 3D character animation controls, and novel desktop systems.

Future Directions

ZUIs make more efficient use of limited screen real estate, and thus are considered to have a great potential on small-sized mobile devices. Examples include mobile calendars, application explorer, image-, web- and scatterplot-browsers. The Apple iPhone includes a variety of such applications and makes extensive use of zooming techniques in combination with a touch sensitive display.

Experimental Results

Different research studies indicate that, on desktop computers, ZUIs reliably outperform scrolling interfaces in terms of performance and preference [6,13]. Similar effects can be observed for small screens as featured by mobile devices. One study, in which different interfaces were tested on a simulated handheld display, found that a two-level zoom was significantly faster for accomplishing editing and monitoring tasks than a scrolling interface. Even in cases where users failed to achieve optimal performance with the two-level zoom, they preferred it to the other experimental interfaces [8].

Cross-references

- ▶ Browsing in Digital Libraries
- ▶ Data Visualization
- ▶ Human-Computer Interaction
- ▶ Interface
- ▶ Mobile Interfaces
- ▶ Navigation
- ▶ Scientific Visualization
- ▶ Usability
- ▶ Visual Analytics
- ▶ Visual Data Mining
- ▶ Visual Interaction
- ▶ Visual Interfaces
- ▶ Visualization for Information Retrieval

Recommended Reading

1. Bederson B.B. and Boltman A. Does animation help users build mental maps of spatial information? In Proc. IEEE Symp. on Information Visualization, 1999, p. 28.
2. Bederson B.B., Clamage A., Czerwinski M.P., and Robertson G.G. Datelens: a fisheye calendar interface for PDAs. ACM Trans. Comput. Human Interact., 11(1):90–119, 2004.
3. Bederson B.B. and Hollan J.D. Pad ++ : a zooming graphical interface for exploring alternate interface physics. In Proc. 7th Annual ACM Symp. on User Interface Software and Technology, 1994, pp. 17–26.
4. Bederson B.B., Meyer J., and Good L. Jazz: an extensible zoomable user interface graphics toolkit in java. In Proc. 13th Annual ACM Symp. on User Interface Software and Technology, 2000, pp. 171–180.
5. Donelson W.C. Spatial management of information. In Proc. 5th Annual Conf. Computer Graphics and Interactive Techniques, 1978, pp. 203–209.
6. Donskoy M. and Kaptelinin V. Window navigation with and without animation: a comparison of scroll bars, zoom, and fisheye view. In CHI'97 Extended Abstracts on Human Factors in Computing Systems, 1997, pp. 279–280.
7. Furnas G.W. and Bederson B.B. Space-scale diagrams: understanding multiscale interfaces. In Proc. SIGCHI Conf. on Human Factors in Computing Systems, 1995, pp. 234–241.
8. Gutwin C. and Fedak C. A comparison of fisheye lenses for interactive layout tasks. In Proc. Graphics Interface, 2004, pp. 213–220.
9. Herot C.F., Carling R., Friedell M., and Kramlich D. A prototype spatial data management system. In Proc. 7th Annual Conf. Computer Graphics and Interactive Techniques, 1980, pp. 63–70.
10. Igarashi T. and Hinckley K. Speed-dependent automatic zooming for browsing large documents. In Proc. 13th Annual ACM Symp. on User Interface Software and Technology, 2000, pp. 139–148.
11. Jog N.K. and Shneiderman B. Starfield visualization with interactive smooth zooming. In Proc. 3rd IFIP WG2.6 Working Conference on Visual Database Systems, vol. 3, 1995, pp. 3–14.
12. Jul S. and Furnas G.W. Critical zones in desert fog: aids to multiscale navigation. In Proc. 11th Annual ACM Symp. on User Interface Software and Technology, 1998, pp. 97–106.
13. Kaptelinin V. A comparison of four navigation techniques in a 2D browsing task. In Proc. SIGCHI Conf. on Human Factors in Computing Systems, 1995, pp. 282–283.
14. Olston C., Woodruff A., Aiken A., Chu M., Ercegovac V., Lin M., Spalding M., and Stonebraker M. Datasplash. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1998, pp. 550–552.
15. Perlin K. and Fox D. Pad: an alternative approach to the computer interface. In Proc. 20th Annual Conf. Computer Graphics and Interactive Techniques, 1993, pp. 57–64.
16. van Wijk J.J. and Nuij W.A.A. Smooth and efficient zooming and panning. In Proc. IEEE Symp. on Information Visualization, 2003, pp. 15–22.