ORRIL: A Simple Building Blocks Approach to Zoomable User Interfaces

Mike Bennett[1,2]

Media Lab Europe[1]
Sugar House Lane,
Bellevue, Dublin 8,
Ireland

mikeb@medialabeurope.org

Fred Cummins[2,1]

Dept. of Computer Science[2]
University College Dublin,
Belfield, Dublin 4,
Ireland

fred.cummins@ucd.ie

Abstract

Zoomable User Interfaces (ZUI) can help people perceive and interact with large collections of information. These large collections complicate the task of creating ZUIs. In this paper the ORRIL (Objects, Regions, Relations and Interface Logic) framework is defined and presented as a technique for aiding ZUI design and creation. ORRIL makes explicit the data that appears in a zoomable information space, while simultaneously emphasising the relationships between user actions and transforms of the data. This is important for modeling and clarifying the processes that occur within a ZUI application. From the perspective of an implementer it may be used as an underlying model when designing and implementing a framework for building ZUIs.

Keywords--- **Zoomable Interfaces, multiscale interfaces, information visualization, user interfaces, interactive graphics, audio display.**

1. Introduction

Zoomable User Interfaces (ZUI) are used [9][1] to help users perceive and interact with large collections of information. However, the process of creating ZUIs remains a complex problem. The interplay between the information in the zooming space, the changing semantic content of the information based on scale and novel interaction techniques result in making designing and implementing ZUIs more challenging than for two-dimensional interfaces.

An important aspect of constructing ZUIs is designing for the display of information at user controllable levels of detail, i.e. at different scales. This method of displaying varying information based on scale in ZUIs is called semantic zooming [12]. The choice of what information to display is often dependent on the position of the user's viewport along the Z axis within the zoomable information space [7][11] (Figure 1). This requirement means an implementer may have to explicitly place many separate but related versions of the information at varying scales within the zoomable information space. Alternatively, or in conjunction, an implementer could construct a generative function that automatically produces an information representation appropriate to the scale.

In this paper we introduce the ORRIL (*O*bjects, *R*egions, *R*elations and *I*nterface *L*ogic) framework as a technique for aiding ZUI design and creation. ORRIL makes explicit the data that appears in a zoomable information space, while simultaneously emphasising the relationships between user actions and transforms of the data.

Section 2 covers a brief review of prior work. Ten requirements are put forward as needed for constructing ZUIs. Examples of the requirements are show in an implemented ZUI. Section 3 introduces and defines

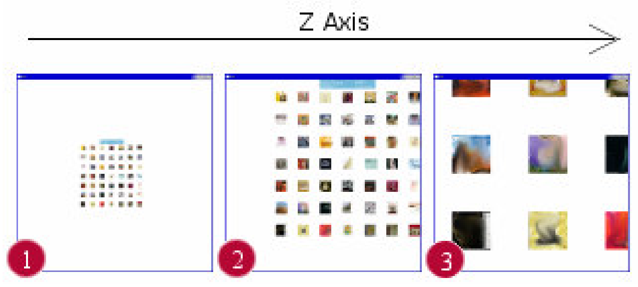

Figure 1. *Three consecutive screen shots captured when zooming in. As the distance traveled along the Z axis increases the semantic content of the displayed information increases.*

ORRIL. The use of ORRIL is elaborated upon in three examples. Section 4 shows how ORRIL fulfills the ten ZUI construction requirements. Section 5 concludes and discusses future directions.

2. Constructing ZUIs

2.1. Prior work

Research into the construction of ZUIs has primarily focused in the areas of ZUI developer toolkits and novel ZUI authoring tools.

Results of the work on developer toolkits include low-level APIs and libraries such as Pad [12], Pad++ [5], Jazz [4], SATIN [10], Tabula Rasa [6] and Piccolo [3]. Benefits of the toolkit approach include a high degree of application independence; and therefore greater flexibility in potential end user interfaces. The key disadvantage is the requirement for a firm knowledge of technical issues, e.g. strong programming skills.

In the area of novel authoring tools MuSE [8] and the related formalism of Space-Scale Diagrams [7] are noteworthy. Space-Scale Diagrams are a visual technique for understanding scale in ZUIs. MuSE uses Space-Scale Diagrams as part of a domain independent prototype authoring system for creating ZUIs. Other ZUI authoring tools are often domain specific. Examples of these include Tioga-2 [1], a database-centric visualization tool, and Counterpoint [9], a tool for creating zoomable slide show presentations. The advantage of these tools is they make powerful visualization techniques available in a user-friendly manner, i.e. little or no programming. The disadvantage is often the domain specific nature of the authoring tools.

2.2. Decomposing ZUI creation

The necessities for constructing ZUIs can be decomposed into ten requirements. From these we extrapolate out to a more general framework for ZUI creation. The requirements identified by us are:

- *R1: Render* a display that a user can perceive and interact with.
- *R2: Place* on the display at least one viewport into a very large three-dimensional information space.
- *R3: Allow* movement of the viewport in the information space so that it can pan and zoom.
- *R4: Constrain* the viewport such that is it impossible to rotate it.
- *R5: Position* data, such as text, images and audio, at specific spatial locations within the information space.

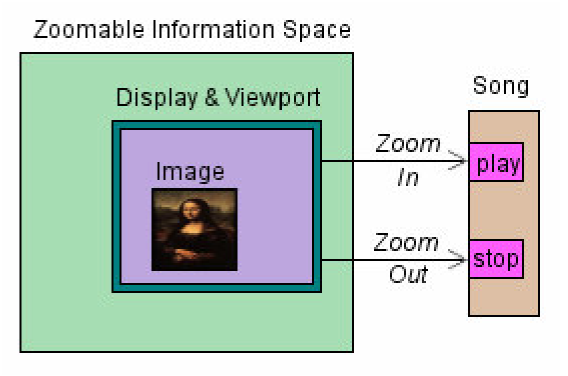

Figure 2. *Overview of process and structure in Media Dive.*

- *R6: Transform* the data based on the viewport position and other occurrences, e.g. user actions.
- *R7: Create* a mapping between which occurrences trigger what transforms of the data.
- *R8: Define* areas within the information space where the mappings exist, i.e. not all transforms will be relevant to all locations or data.
- *R9: Encode* the nature of the transforms that occur on data.
- *R10: Enable* transforms and mappings to be altered.

2.3. An example of the Requirements

Media Dive (Figure 2) is a prototype ZUI application we designed and implemented for browsing large collections of audio, such as songs. The visual interface (Figure 3) displays many images, each of which is associated with a song. A user browses the songs by panning and zooming. If a user zooms in to an image the associated song will begin to play and if a user zooms out the song will stop.

Looking at the Media Dive from the perspective of the ten requirements we see there is a display (R1: Render) with a viewport (R2: Place) into a very large information space where images and songs are spatially organized (R5: Position). The user can pan around and zoom (R3: Allow and R4: Constrain) in to the images thereby triggering the start of songs (R6: Transform and R7: Create). Zooming in to a specific image triggers a particular song (R8: Define). Triggering a song causes it to play or stop (R9: Encode), e.g. read the audio data, process it and send it out on an audio channel.

Note that Requirement 10 is not currently used in Media Dive. However Media Dive could be extended so that a user could reorganise the layout of the songs by

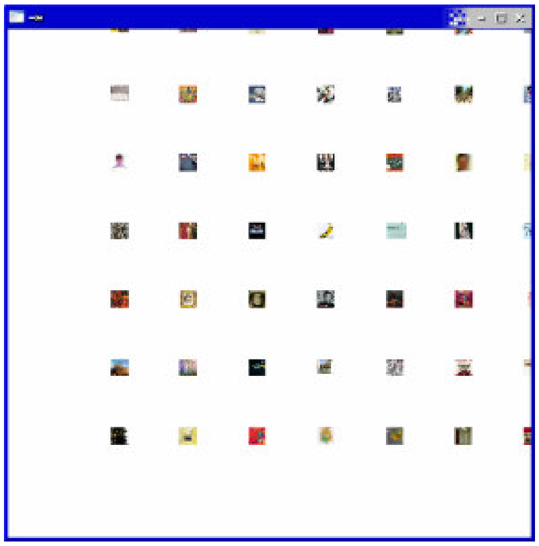

Figure 3. *Media Dive interface showing thirty six songs. Each image/dot represents a song that may be zoomed towards to hear it playing.*

direct manipulation. This would mean the mapping between zooming in to a particular area in the information space and what song is played would need to be altered (R10: Enable) in response to the user's actions.

2.4. Grouping the Requirements

Analysing the ten requirements enables further classification of them into four broader and interdependent groups (Table 1).

Group	***Requirements***
Display	R1: Render, R2: Place, R3: Allow, R4: Constrain
Data	R5: Position, R10: Enable
Interaction	R7: Create, R8: Define
Results	R6: Transform, R9: Encode

Table 1. *The ZUI Requirement Groups.*

Fulfilling the requirements within the *Display Group* would result in a basic zoomable information space where the fundamental ZUI operations of zooming and panning a viewport are possible. Of course this would be useless if there was no data within the display. Therefore the *Data Group* in conjunction with the Display Group is required to create a limited but usable ZUI application.

For more complex applications there is a need to monitor user actions and respond to them accordingly. As well as reacting to external occurrences, e.g. updating the display to indicate a file has loaded. Partially realising this need is possible with the *Interaction Group*, which would enable the creation of a ZUI application that monitors but does not respond to user actions and external occurrences. A complete application necessitates also using the *Results Group* to add and define interface and program behaviours.

Within a complete ZUI application the interplay between the groups is, in a simplified form, that which is shown in Figure 4. There is a display that enables interaction, which often results in the transformation of the data and that in turn causes display updates.

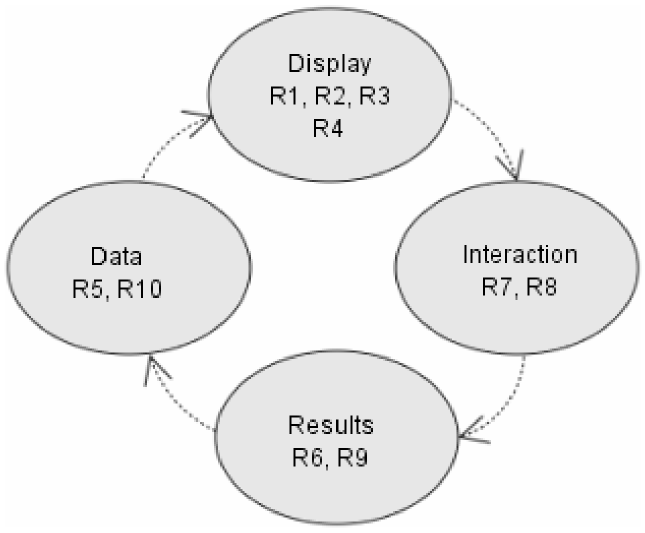

Figure 4. *Simplified interplay and flow of dependencies between the Requirement Groups.*

3. ORRIL

3.1. Defining ORRIL

ORRIL (Objects, Regions, Relations and Interface Logic) is an abstraction of our approach to ZUIs into four basic components (Figure 5). This abstraction is useful as a framework for understanding ZUIs. From the perspective of an implementer it may be used as an underlying model when designing and implementing a framework for building ZUIs. Alternatively it is useful for modeling and clarifying the processes that occur within a ZUI application.

The four ORRIL components are Objects, Regions, Relations and Interface Logic. The definition of each is as follows:

- *Objects* represent a basic perceptual unit within a zoomable information space, e.g. an image, a piece of audio or a block of text.

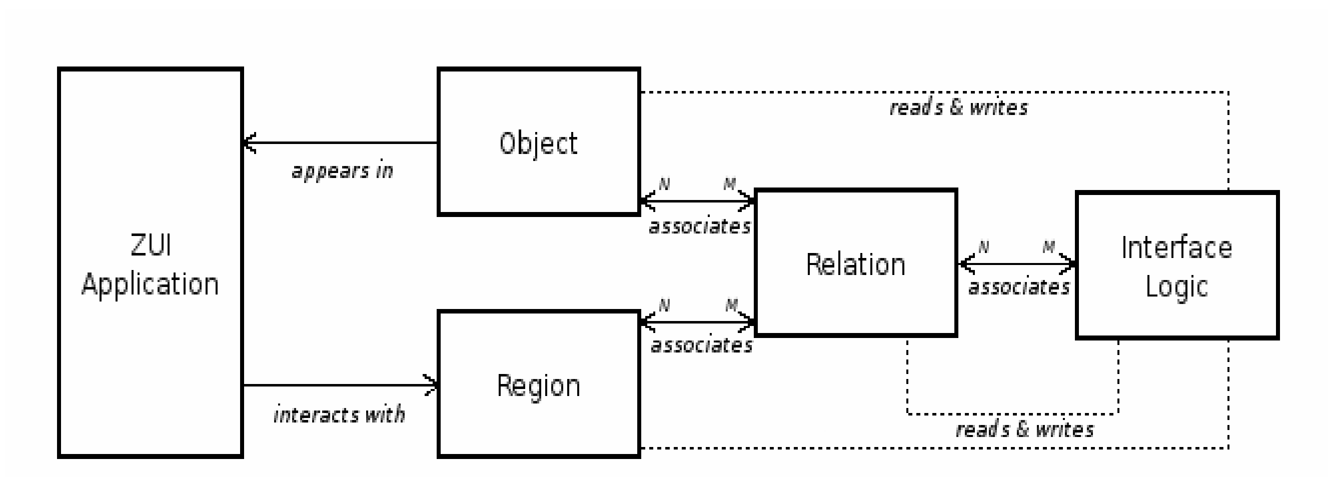

Figure 5. *The ORRIL framework and the relationships between its components.*

- *Regions* denote three-dimensional areas where user actions may be captured, e.g. movement of a viewport, continuous updates of a pointer's position, a key press, etc.

- *Relations* define the mappings and associations between Regions, Objects and Interface Logic.

- *Interface Logic* represents transforms that can occur.

Our focus with ORRIL was to make explicit the data that appears in a zoomable information space, while simultaneously emphasising the relationships between user actions and transforms of the data.

This is beneficial because it helps reduce the complexity associated with having many different components for creating a ZUI. Reducing the number of different types of components means users do not have to keep track of numerous levels and layers of abstraction. Clearly delimiting the role of each component enables users to think about ZUI creation at varying levels of complexity.

3.2. Three examples of using ORRIL

If a user wants to quickly create a ZUI, or is unfamiliar with them, they could think only in terms of Objects. Each Object could be associated with a single image. By placing Objects within a zoomable information space they would have control over where the images appear, thus creating a basic ZUI application.

The following is a more complex example where all four ORRIL components are used. In Section 2.3 Media Dive was analysed from the perspective of the Requirements - here it will be presented in the ORRIL framework (Figure 6).

Media Dive requires that songs and images are placed within a large zoomable information space. With ORRIL this is done by associating each song and each image with an Object. Each Object is then given a specific location within the zoomable information space. If the viewport displays an Object location then the Object's image will be displayed in the viewport.

When a Media Dive user zooms in to an image the song associated with the image begins to play. This is done by creating one Region for each image. Each Region is given a location with the result that it is placed in front of an image Object. When the viewport enters a Region a user event occurs that eventually leads to the song starting.

The user event will lead to nothing if it is not linked

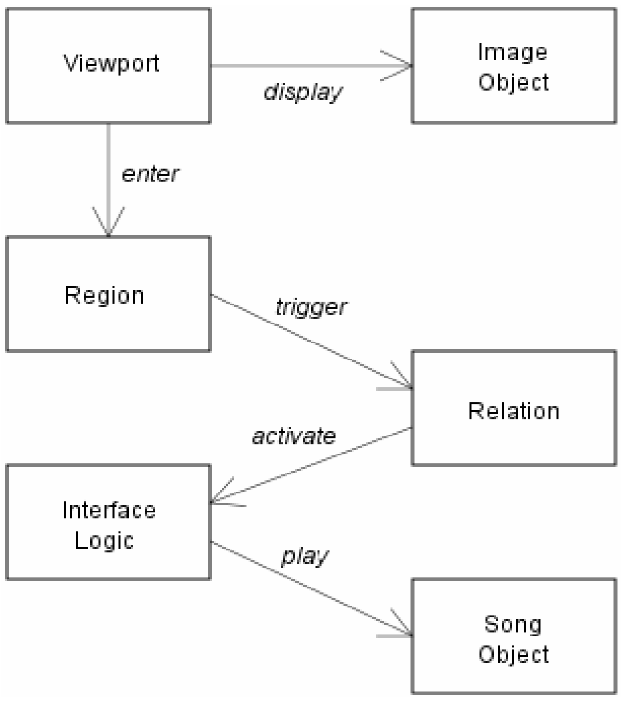

Figure 6. *A snapshot of the ORRIL framework applied to Media Dive.*

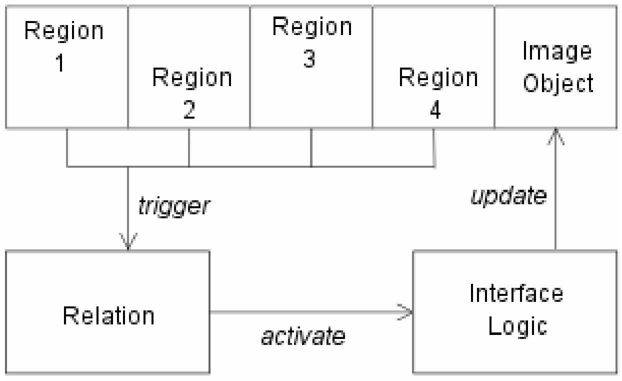

Figure 7. *Semantic zooming as outlined by ORRIL.*

with a transformative function, i.e. starting or stopping the song. A Relation maps the user event to a transformative function. The transformative function is encoded in the Interface Logic.

A Region captures the user event. The user event is mapped via a Relation to Interface Logic. The Interface Logic is a function that acts upon a song Object. The function acting on the song causes it to start playing. The music stops when the same Region detects a user leave event. The user leave event is mapped to a function for stopping the music playing.

The final example (Figure 7) covers using ORRIL to outline the process of semantic zooming. In this case ORRIL is used to explain what occurs when zooming towards a single image. As the zoom occurs the image continuously gets bigger and the content of the image is updated in discreet steps. Figure 7 shows four distinct Regions in front of the image Object. When the user controlled viewport enters Region 1 the image Object is updated by the activated Interface Logic. Transitioning from Region 1 to Region 2 causes another activation of the Interface Logic, which again updates the image. This also occurs when transitioning from Region 2 to Region 3, and from Region 3 to Region 4.

4. From Requirements to ORRIL

Returning to the analysis at the end of Section 2 we note that the four ZUI Requirement Groups do not map directly to the four ORRIL components.

Instead ORRIL is built on the core assumption that the Display Group is the default environment in which ZUIs exist. This assumption means that the very large information space, the display with a viewport and zooming and panning (R1: Render to R4: Constrain) are defaults that are not explicitly captured in ORRIL. That is each component of ORRIL exists in the context of a standard ZUI. In Figure 5 the Display Group can be thought of as belonging to the ZUI Application.

ORRIL	*Requirements*
Objects (1)	R5: Position
Regions (2)	R8: Define
Relations (3)	R7: Create, <P> R10: Enable
Interface Logic (4)	R6: Transform, R9: Encode, <P> R10: Enable

Table 2. *Mapping between ORRIL components and the Requirements. <P> means the Requirement following <P> is only partially met.*

Table 2 shows how the ORRIL components relate to and fulfill the remaining six Requirements. What follows is an elaboration on how each ORRIL component contributes to meeting the Requirements.

1. Objects related directly to R5: Position. There is a partial relationship with the Data Group because Objects do not meet R10: Enable, i.e. Objects can be transformed but they cannot transform other ORRIL components.

2. Regions meet the requirement of R8: Define. A partial relationship exists between Regions and the Interaction Group. The relationship is only partial because Regions cannot be used to create new mappings between ORRIL components, i.e. an ORRIL Relation needs to be used as well.

3. Relations enable the creation of new ORRIL mappings therefore they meet the R7: Create and partially meet the R10: Enable Requirements. R10: Enable is partially met because altering a Relation alters a mapping but not a transform. The Interaction Group is now completely fulfilled because Relations meet R7: Create and Regions fulfill R8: Define.

4. Interface Logic directly meets the R6: Transform and the R9: Encode Requirements, which also means it fulfills the Results Group. The remaining unfulfilled part of R10: Enable is met by Interface Logic because transforms can alter transforms and mappings. This completes the Data Group.

Therefore the four ORRIL components fulfill all the ZUI creation Requirements that we specified in Section 2.

5. Conclusions and Future Directions

In this paper we have presented the ORRIL framework as a technique for aiding ZUI design and creation. As part of this we outlined ten requirements and the processes needed when constructing ZUIs. In Section 2.3 we showed where the Requirements occur in a prototype ZUI application, i.e. Media Dive.

Based on the Requirements and Requirement Groups we analysed ORRIL in Section 4. The results of this indicate that ORRIL meets all the Requirements for a suitable framework for specifying ZUIs.

In Section 3.2 the three examples of using ORRIL imply that it is sufficiently expressive for describing a large range of ZUIs.

Future directions for our work include evaluating ORRIL by implementing it as part of an application for rapidly prototyping high-fidelity [13] ZUIs. This work has already begun and takes the form of a new tool called Nutmeg.

Further work by the authors will include analysing ORRIL in conjunction with Space-Scale Diagrams. This will provide a means of evaluating ORRIL by doing a contrastive analysis with a closely related but not completely equivalent framework.

Acknowledgements

This research has been supported by Media Lab Europe, University College Dublin, and the HEA (Higher Education Authority) in Ireland.

References

[1] Aiken, A., Chen, J., Stonebraker, M., and Woodruff, A. Tioga-2: A Direct Manipulation Database Visualization Environment. *Proc. of the 12th International Conference on Data Engineering*, 1996, p. 208-217.

[2] Bederson, B.B. PhotoMesa: A Zoomable Image Browser Using Quantum Treemaps and Bubblemaps. *Proc. of USIT 2001, ACM Symposium on User Interface Software and Technology*, p. 71-80.

[3] Bederson, B. B., Grosjean, J. and Meyer, J. *Toolkit Design for Interactive Structured Graphics*. Tech Report HCIL-2003-01, Computer Science Department, University of Maryland, College Park, MD.

[4] Bederson, B. B., Meyer, J., and Good, L. Jazz: An Extensible Zoomable User Interface Graphics ToolKit in Java. *Proc. of USIT 2000, ACM Symposium on User Interface Software and Technology*, CHI Letters 2(2): p. 171-180.

[5] Bederson, B. B., and Hollan, J. D. Pad++: A Zooming Graphical Interface for Exploring Alternate Interface Physics. *Proc. of UIST'94, ACM Symposium on User Interface Software and Technology*, p. 17-26.

[6] Fox, D. *Tabula Rasa: A Multi-scale User Interface System.* (1998) Doctoral dissertation, New York University, New York, NY

[7] Furnas, G., and Bederson, B. B. Space-Scale Diagrams: Understanding Multiscale Interfaces. *Proc. of ACM CHI 1995*, p234-241.

[8] Furnas, G., and Zhang, X. MuSE: A Multiscale Editor. *Proc. of USIT'98, ACM Symposium on User Interface Software and Technology*, p107-116

[9] Good, L., and Bederson, B. B. Zoomable User Interfaces as a Medium for Slide Show Presentations. *Information Visualization* 1(1) 2002, Palgrave Macmillan, p. 35-49.

[10] Hong, J., and Landay, J. SATIN: A Toolkit for Informal Ink-based Applications. *Proc. of USIT 2000, ACM Symposium on User Interface Software and Technology*, CHI Letters 2(2): p. 63-72.

[11] Lieberman, H. A Multi-Scale, Multi-Layer, Translucent Virtual Space. *Proc. of the IEEE International Conference on Information Visualization*, London, September 1997.

[12] Perlin, K. and Fox, D. Pad: An alternative approach to the computer interface. *Proc. of the 20th Annual ACM Conference on Computer Graphics*, SIGGRAPH 1993, p. 57-64.

[13] Walker, M, Takayama, L., and Landay, J. High-Fidelity or Low-Fidelity, Paper or Computer? Choosing Attributes When Testing Web Prototypes. *Group for User Interface Research*, Computer Science Division, University of California, Berkeley.