
Chasm: A Tiered Developer-Inspired 3D Interface Representation

Chadwick A. Wingrave

Center for Human Computer
Interaction
Virginia Tech
2202 Kraft Drive
KWII Building (0106)
Blacksburg, VA 24061 USA
cwingrav@vt.edu

Abstract

3D interface implementation is complex and made more difficult with each additional feature. After investigating the envisioned behaviors of 3D interface developers in language, artifacts and interviews, a tiered representation based upon developer artifacts and language was created. Chasm allows for development as tiered cohesive *concepts* and the execution as flows similar to developer envisioned behavior. Chasm has been used by multiple developers in case studies.

Keywords

3D interaction, User Interface Description Language, Model-Driven Engineering, User Interface Management System

ACM Classification Keywords

D.2.2 [Software Engineering]: Design Tools and Techniques---Computer-aided software engineering (CASE), Object-oriented design methods, User interfaces; H.5.2. [Information Interfaces and Presentation] User Interfaces--- Graphical User Interfaces.

Introduction

I am not interested in content creation, but interaction creation. This entails behaviors in the environment, of the user or of the interface, brought about by changes by the user, environment, interface and time. Also, I

am not interested in rapid prototyping of a simple interface. What I am interested in is a representation allowing for deep access to interaction details and for broad ideas to be representable. A representation where new ideas can be created and expressed, possibly even inspired, simply by working in the representation. Interfaces have been studied and many ideas explored in the field of 3D interaction. Despite this, there is a lack of a truly interactive experiences where the benefit is due to the interaction and not the visual experience [1].

While 3D interfaces have always been hard to implement [2], this does not explain why added features are disproportionately more complicated to implement. However, looking more closely at how 3D interfaces advance, we see the following: Each additional feature increases the actions and state of the user, environment and interface to be considered during implementation. Since each action is considered in regard to each state, linear growth of actions and state result in non-linear growth of implementation complexity.

Despite this complexity growth, 3D interaction behaviors are readily envisioned by 3D interface developers. The problem is the difficulty developers have implementing these ideas. Logically then, an implementation based on developer's envisioned behavior of 3D interfaces would scale at least as well as developer understanding. Through an investigation of developer artifacts, interviews and language (collected by developers rerepresenting video of 3D interaction techniques as language), a list of five problems facing developers was created and discussed in [4]:

- Limited Understanding
- Distance of Mapping
- Complexity
- Reimplementation over Reuse
- Hard and Broad Problems

The created development representation based upon developer artifacts has been called Concept-Oriented Design and implemented in a system called Chasm [4]. Chasm is named for its ability to bridge the divide between developer's envisioned behavior and the machine which runs it. Concept-Oriented Design (COD) has as its unit of implementation a concept. In Chasm, a concept implements a single cohesive concern in four tiers of developer understanding, shown in Figure 1. Each successive tier represents a different type of information. At runtime, the Chasm system interleaves the cohesive functionality with other concepts for the desired envisioned behavior. Chasm has been used in multiple development case studies [4].

WIM Example

An example implementation of a moderately complicated 3D interaction technique, the world-in-miniature (WIM) [3], shows the benefits of a Chasm representation. The developer's envisioned behavior of the WIM technique might be described as: "A tracker's movement causes the virtual hand to move. When a button is pressed, check for selection of an object in the WIM (a proxy object). If an object is selected, move the full-scale object in the environment as the proxy object moves. When the button is released, release the proxy object from the user's hand and stop moving the full-scale object." This description is shown as a flow at the top of Figure 2.

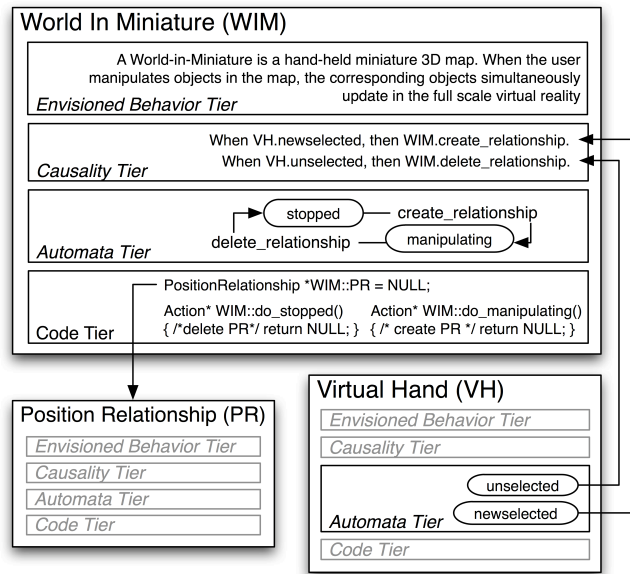


Figure 1: The four tiers of Chasm represent developer understanding as they implement a 3D interface, shown here for the WIM [3] technique. Decomposing development in this way creates cohesive concepts which are executed by Chasm as flows of events as shown in Figure 2.

Though this flow explains the WIM's functioning, it is limited in its utility for development. First, the flow does not separate into cohesive concerns to simplify development. For example, the button's roles are spread across two distant steps (2 and 6). Second, there is no clear division between a discrete event and a continuous relationship. For example, moving the hand and the full-scale object (steps 1 and 5) are continuous while checking for selection is discrete (step 3). Third, implementation of the flow requires a complicated developer mental model in order to keep

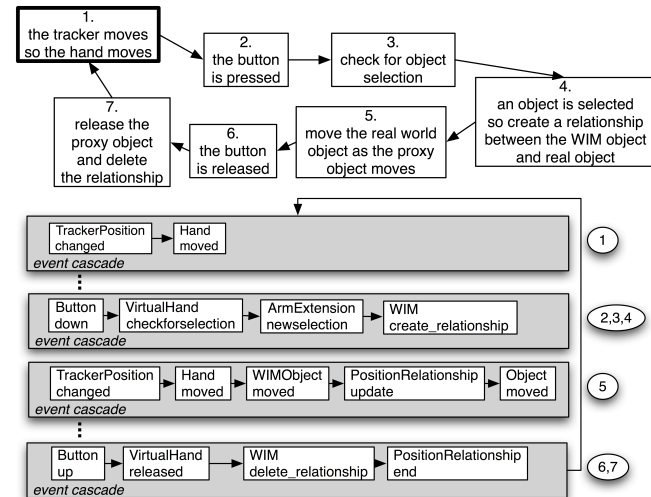


Figure 2: A flow (top) of the WIM technique, useful understanding how the interaction progresses, involves many different ideas which the developer has to maintain during implementation. The processing of events (bottom) in Chasm integrates the behaviors of multiple concepts as the flow up top but allows development as cohesive ideas (Figure 1). The numbers on the right connect events to the steps up top.

all requirements of the hand, button, selection technique and object-to-object relationships consistent.

In contrast, a Chasm representation is based on abstractions in conversational domain language. WIM was described by Stoakley et. al. in the following two sentences, "A World-in-Miniature is a hand-held miniature 3D map. When the user manipulates objects in the map, the corresponding objects simultaneously update in the full scale virtual reality" [3]. Notice how these two sentences simply describe a complex interaction technique by depending on abstractions in domain knowledge.

The WIM implementation is spread across four tiers of a Chasm concept shown in Figure 1. First, the language used to describe the WIM is placed in the Envisioned Behavior Tier. This informs the deeper tiers with a high-level, easily understood, language description. The language description implies that the virtual hand (VH) and Position Relationship (PR) concepts, previously implemented in other systems, are reusable here. Second, in the Causality Tier the WIM technique creates causal relationships to the VH concept's states to connect to its functionality. Third, the Automata Tier forms its custom state machine representation using the actions caused by the VH concept and two created states: manipulating (moving a proxy object) and stopped (not moving the proxy object), that represent the WIM's flow of behavior as described by Stoakley et. al.'s second sentence. Finally, the Code Tier implements in C++ the behavior when these two states are entered. The manipulating state creates the PR to update the full-scale object, and the stopped state deletes the PR.

This implementation is executed in Chasm as shown at the bottom of Figure 2 as cascades of events. The implementation is cohesive as in Figure 1 and the flow is interleaved by Chasm to match the flow at the bottom of Figure 2. Additional features to the WIM, such as the scaling and scrolling of the SSWIM technique [5], have full access to the internals of the WIM implementation by responding directly to the states in its flow.

Chasm's Non-Functional Requirements

An evaluation of Chasm along non-functional requirements shows its utility. *Comprehensibility* is

increased by the tiered representation, as developers know where to look for the understanding they need. *Simplicity* is created by dividing envisioned behavior along language abstractions and reducing the scope of developer mental models. *Reusability* occurs through cohesive concepts which are composable to form higher levels of meaning. *Evolvability* of systems is achieved by swapping the cause of a stimulus or response in the Causality Tier. *Portability* to new platforms, environments, and toolkits is less onerous, as only the Code Tier needs reimplementation.

Concluding Remarks

A brief introduction of the development of Chasm discussed its origins in developer representations. The WIM technique in Chasm shows implementation as cohesive ideas and execution as flows similar to developer understanding. Non-functional requirements show the utility of the approach with [4] referenced for a full discussion of the validation of Chasm in multiple case studies with multiple developers.

References

- [1] Brooks, F. What's Real about Virtual Reality? In *Computer Graphics and Applications* 19,6 (1999).
- [2] Herndon, K. P., Van Dam, A. and Gleicher, M. Workshop on the Challenges of 3D Interaction. *SIGCHI Bulletin*, 26, 4, (1994).
- [3] Stoakley, C. and Pausch, R. Virtual Reality on a WIM: Interactive Worlds in Miniature, *CHI*, (1995).
- [4] Wingrave, C. A., Bowman, D. A. Tiered Developer-Centric Representations for 3D Interfaces: Concept-Oriented Design as Chasm. Submitted to IEEE VR 2008.
- [5] Wingrave, C., Hacıahmetoglu, Y. and Bowman, D. Overcoming World in Miniature Limitations by a Scaled and Scrolling WIM. *3D User Interfaces*, (2006).