

Semantic Zoom

With Immersive Detail View for ExplorViz

Jens Bamberg

Master Thesis
December 3, 2024

Software Engineering Group
Department of Computer Science
Kiel University


Advised by
Prof. Dr. Wilhelm Hasselbring
M. Sc. Malte Hansen

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Weiterhin erkläre ich, dass die digitale Fassung dieser Arbeit, die dem Prüfungsamt per E-Mail zugegangen ist, der vorliegenden schriftlichen Fassung entspricht.

Kiel, 4. 12. 2024



Abstract

The demand for large software projects is growing and therefore the use of software visualizations is a precious tool for onboarding new developers and introducing new software features. The use of visualizations is an effective method for accelerating the comprehension of data. Given the limited visual processing power of the human brain, the amount of visible objects can become overwhelming. It is therefore necessary to reduce the number of visible objects at once without losing access to important details. In light of these considerations, we introduce two new features, namely semantic zoom and immersive view, to the domain of software visualization, with ExplorViz serving as a case in point. ExplorViz is a 3D city metaphor software visualization tool that displays both dynamic and static data. Semantic zoom is a common technique employed in 2D graph representations for the aggregation of data. In this work, the semantic zoom feature is applied to a 3D visualization. The feature in question conceals, reveals, and modifies the 3D objects of the visualization in accordance with the position of the camera within the environment. The immersive view offers a comprehensive, unobstructed representation of the object in question, free from the distractions inherent to traditional visualizations.

A user and a system's performance evaluation provides insight into the usability and performance improvements. The system's performance evaluation indicates a higher rate of frames per second on average in large landscapes compared to a system without the semantic zoom feature. However, the frames per second exhibited more fluctuations and could even reach almost zero in some cases. A user evaluation was conducted with 16 participants to examine the user performance while performing tasks in the visualization. The results of the user evaluation indicate no improvement in the time required to complete the tasks. However, the participants rated the interaction with the visualization positively. The newly introduced immersive view provided further detailed information and was considered beneficial by the participants.

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Document Structure	2
2	Goals	5
2.1	G1: Integration of Semantic Zoom and Level of Detail	5
2.1.1	G1.1: Identification of Objects and Metrics That Are Suitable for Semantic Zoom and Level of Detail	5
2.1.2	G1.2: Addition of Different Levels of Detail	5
2.1.3	G1.3: Implementation of Semantic Zoom for Appropriate Objects	5
2.1.4	G1.4 Optional: Improve Performance Using WASM for Goal 1.3	6
2.2	G2: Adding an Immersive View	6
2.2.1	G2.1: Structuring and Designing of an Immersive View	6
2.2.2	G2.2: Immersive View Implementation	6
2.3	G3: Performance Evaluation of Goal 1	6
2.4	G4: Usability Evaluation by Users for Goal 1 and Goal 2	6
3	Foundations and Technologies	7
3.1	<i>Semantic Zoom</i> and <i>LoD</i>	7
3.2	<i>Three.js</i>	7
3.3	<i>Ember.js</i>	8
3.4	<i>OpenTelemetry</i>	9
3.5	<i>ExplorViz</i>	10
4	Related Work	13
5	Semantic Zoom	19
5.1	Analysis of the Current <i>ExplorViz</i> State	19
5.2	Fundamental Appearance Options	19
5.3	New Designs, Metrics and Aggregations	21
5.4	Concept and Pre-Implementation Thoughts	22
5.5	Implementation of Semantic Zoom	23
5.5.1	Appearances	23
5.5.2	Algorithms To Decide on Level of Detail	24
5.5.3	Programmer Interface	33
5.5.4	Known Problems and Limitations	34

Contents

5.6	User Settings and Parameters	34
6	Immersive View	37
6.1	General Concept	37
6.2	Analyze of Attachment Point for ExplorViz	38
6.3	Implementation of Immersive View	39
6.3.1	Current Data Model	39
6.3.2	Example Implementation Based on the <i>Immersive Class View</i>	39
6.3.3	Entering/Exiting	41
7	Evaluation	43
7.1	Goals	43
7.2	Performance Evaluation	43
7.2.1	Setup	43
7.2.2	Evaluation Results	46
7.2.3	Discussion	46
7.2.4	Threats to Validity	47
7.3	User Evaluation	47
7.3.1	Setup	48
7.3.2	Pretest	48
7.3.3	Introduction to ExplorViz	48
7.3.4	General Questions	49
7.3.5	Questions and Assignments for <i>Semantic Zoom</i>	49
7.3.6	Questions and Assignments for <i>Immersive View</i>	51
7.3.7	Evaluation Results	52
7.3.8	Discussion	60
7.3.9	Threats to Validity	62
8	Conclusion	65
8.1	Future Work	66
A	Cheat Sheet	69
	Bibliography	75

Introduction

Software developers spent more time reading code compared to writing code [Scalabrino et al. 2018]. Reading is required to comprehend the system in order to extend or maintain it. A software visualization tool is designed to facilitate a more comprehensive understanding of complex software projects. This is particularly crucial when new features are introduced, existing features are modified, or new developers are integrated into a team. Visualization tools can assist in the maintenance of software, as developers build an internal representation of the code architecture in their mind [Teyseyre and Campo 2009]. The software visualization assists in the creation of a comprehensive representation of the entire system, that can be easily shared among developers. The images are constructed using a variety of visual objects that represent systems or components [Teyseyre and Campo 2009]. This approach allows the new developer to avoid the necessity of examining the entirety of the source code, offering a comprehensive overview of the structure instead. Visualization of dynamic runtime information provides insights into the communication patterns within a software system that are not observable in static code analysis. The use of software visualization is beneficial in aiding comprehension of software architecture, as well as facilitating cognitive processing through the utilization of the visual cortex [Teyseyre and Campo 2009]. There is much research regarding a 2D visualization [Wiens et al. 2017] or even software visualization [Alnabhan et al. 2018], but a 3D representation is gaining more popularity since the computation power has increased in the past years [Teyseyre and Campo 2009]. *ExplorViz* is one of those 3D software visualization tools. The analyzed software with *ExplorViz* is structured in a treemap structure, resulting in a 3D Code-City displayed on the *frontend*. The city metaphor is a known technology in terms of 3D data visualization, like [Balogh et al. 2016] showed with a 3D-Websearch-Tool.

1.1 Motivation

Understanding software is a time-consuming and costly process, especially when the code was written by someone else [Al-Saiyd 2017]. A software visualization tool reduces the comprehension time [Teyseyre and Campo 2009]. The objective of software developers is to be presented with information in a precise manner on their screens. This includes a trade-off between the simplification of the view and the insight to be gained from

1. Introduction

the information, which must be carefully considered to ensure optimal usability [Voinea and Telea 2006]. In the current state of *ExplorViz*, packages are opened manually by clicking on them or collectively via a context menu. The displayed information is static and remains unaltered when interactions occur within the virtual environment. In this context, *Semantic Zoom* serves as a solution. For instance, the packages could be opened and closed automatically, depending on the developer's view, without requiring direct interaction. Another example is the problem of unreadable labels if they are too far away, which can result in flickering artifacts. Interesting properties that are weighted more than other properties will be displayed more often than common information. Identifying such interesting properties represents another challenge. All modern 2D map visualization tools like Google Maps ¹ provide a *Semantic Zoom* feature. As an illustration, one may choose to magnify the image in order to observe the street names more closely, or alternatively, to reduce the magnification in order to view the cities or even the country names. This functionality is available on various online map services.

The implementation of a *Semantic Zoom* feature is even more advised, since a study by Luck & Vogel in 1997, shows how the human visual memory works [Liverence and Franconeri 2015]. An experiment was conducted using up to 12 objects. They were first visible and then disappeared in order to alter one object's shape or color. All objects reappeared and the participants were required to identify the object that had been changed. The outcome was that the participants could only keep track of up to four objects and their features in their *Visual Working Memory (VWM)*. This indicates, that the *VWM* capacity is limited. The objective of *Semantic Zoom* is to address this issue by reducing the level of detail at distant objects and enhancing the level of detail at nearby objects, while ensuring that the visibility of other objects is not compromised [Liverence and Franconeri 2015].

In addition to the proposed *Semantic Zoom*, a new detailed view without context distraction is introduced. This view is called *Immersive View*.

Attentional blink also plays a role in determining the amount of visible information. The attentional blink was tested by detecting two targets in a rapid serial visual presentation of distractors. It was difficult to notice the second target if the first target occurred within 500ms prior [Victor 2000]. This result demonstrates that the brain not only has a limited *VWM* but also requires time to process the information. That's where *Semantic Zoom* and *Immersive View* can help to maximize the potential of the *VWM* and work on the edge of the attentional blink.

1.2 Document Structure

The introduction in the first Chapter provides a brief overview of the topic and its motivation. Chapter 2 followed by this Chapter describes the goals that are pursued. It is separated into two main goals with several sub-goals and the evaluation of the goals. In Chapter 3

¹www.google.de/maps

1.2. Document Structure

technologies and the baseline of the project are introduced. Chapter 4 describes related works and useful papers. After that the Chapter 5 and Chapter 6 describe the concept and the implementation of the *Semantic Zoom* and the *Immersive View*. The evaluation for both implementations is presented in Chapter 7. The *Semantic Zoom* feature is evaluated by a performance measurement and a user evaluation. The *Immersive View* is only evaluated by a user evaluation. The final Chapter 8 states the results and an outlook.

Goals

2.1 G1: Integration of Semantic Zoom and Level of Detail

2.1.1 G1.1: Identification of Objects and Metrics That Are Suitable for Semantic Zoom and Level of Detail

It should be noted that not all 3D objects displayed in *ExplorViz* are suitable for semantic zooming. This includes those that display only a binary value. The objective of goal 1.1 is to identify all suitable objects. They must then be grouped into different categories and sorted by priority.

2.1.2 G1.2: Addition of Different Levels of Detail

All identified objects from goal 1.1 can be utilized in goal 1.2 to modify their visual representation. It is essential to store the appearance of these objects in a generic way, as this allows an algorithm in goal 1.3 to determine the optimal rendering strategy. This raises the question of whether the appearance change is discrete or continuous.

2.1.3 G1.3: Implementation of Semantic Zoom for Appropriate Objects

The identified objects from goal 1.1 and goal 1.2 provide a basis for the *Semantic Zoom*, yet it is lacking in the ability to make decisions regarding the displayed state. It is necessary to determine whether the state of objects is modified only in accordance with the distance to the camera, or whether additional factors should be taken into account. What is the impact on performance if the distance of all objects to the camera is calculated? A review of existing literature may provide insight into the optimal approach for addressing this issue. In any case, the system should include parameters that allow users to modify the sensitivity and alter these values in the *frontend*. The automatic handling of zoom should not interfere with the user's actions and can be disabled. If the performance of the *Frames per Second (FPS)* in the visualization drops below 30 *FPS* in a decent-sized landscape, the system is not considered useful.

2. Goals

2.1.4 G1.4 Optional: Improve Performance Using WASM for Goal 1.3

The introduced algorithm for rendering decisions can be optimized for enhanced performance in scenarios involving a large number of objects when utilizing *WebAssembly*.

2.2 G2: Adding an Immersive View

2.2.1 G2.1: Structuring and Designing of an Immersive View

The default view can be characterized by a high degree of information density, with data from nearby system components. In contrast, the *Immersive View* is designed to provide a closed view with a singular focus on a designated object. This results in a context switch with minimal or no connection to the previous view. The objective is to develop a structure that specifies which objects are compatible with the *Immersive View* and to establish a fundamental design framework.

2.2.2 G2.2: Immersive View Implementation

The *Immersive View* modifies the default view to present a first-person perspective, displaying the functions and variables of a class (separated by public and private) in a shelf-like arrangement. This primarily focuses on the *Immersive View* of class objects, with a conceptual similarity to Google Maps¹ and its transition from the 2D map to the street view perspective.

2.3 G3: Performance Evaluation of Goal 1

This goal compares the approach introduced in goal 1.3 with the base approach in terms of performance. This is particularly the case in the context of very large landscapes. Potential parameters for evaluation include one or two landscapes with a fixed initial camera position, fixed initial orientation, fixed *Field of View (FOV)*, and fixed zoom speed in discrete steps.

The performance metrics are as follows: Number of displayed objects and meshes, *FPS*, as well as the *Central Processing Unit (CPU)* utilization and memory consumption.

2.4 G4: Usability Evaluation by Users for Goal 1 and Goal 2

The impact of a software feature on usability and productivity can be evaluated through a user survey. The users are tasked with performing specific functions, and the time required to complete these tasks is documented. It is anticipated that there will be a marginal enhancement in the acquisition of information displayed on the screen.

¹www.google.de/maps

Foundations and Technologies

3.1 *Semantic Zoom and Level of Detail (LoD)*

Semantic Zoom has multiple definitions: It is mostly compared against the geometric zoom or physical zoom, which only scales the objects, but do not present further information. *Semantic Zoom* is described to change the structure of objects that are to be displayed. It changes the shape or appearance in any way to display other data on different spatial scale¹. Many papers that describe *Semantic Zoom* are only focused on 2D representation of data [Wiens et al. 2017], [Buering et al. 2006].

Level of Detail (LoD) displays objects with different numbers of polygons. There are three major *LoD* variants. The first one is the discrete *LoD*. The application receives multiple preprocessed versions of an object with a varying number of polygons. Subsequently, the software determines the optimal choice. As the angle of the object is not known in the pre-processing step, it is not possible to implement any optimizations in that regard.

The next significant *LoD* variant is the continuous *LoD* variant. The data structure for an object is streamed at run time and provides enhanced granularity. Refinements can be streamed via a slow internet connection.

The last major *LoD* is the View-Dependent *LoD*. It is a combination of continuous *LoD* and the viewing angle. It is possible for larger objects to have a greater degree of granularity of polygons that are in closer proximity to the viewer, while simultaneously reducing the number of polygons in the far distance from the viewer. However, despite this variation, the object remains coherent [Luebke 2003].

In general *LoD* focuses on the reduction of meshes and polygons for an object.

3.2 *Three.js*

The *Three.js*² library is used to create 3D scenes within *Web Browsers*. It utilizes the WebGL technology as its rendering engine [Dirksen et al. 2014]. WebGL introduces the OpenGL 2.0 SE into the realm of *Web Browsers* [Parisi 2012]. All major *Web Browsers* are supported by *Three.js*, like Google Chrome, Safari and Firefox. Furthermore, it has a fallback mode,

¹https://infovis-wiki.net/wiki/Semantic_Zoom

²<https://threejs.org/>

3. Foundations and Technologies

where it renders the scene on a *Hypertext Markup Language (HTML)* canvas. The library provides an easy-to-use *Application Programming Interface (API)* for the manipulation of 3D renderings, which was first made available in 2010 [Dirksen et al. 2014]. The software developer initiates a new scene and adds a camera to it. While not mandatory, the addition of a camera can prove beneficial in instances where the camera is to remain fixed to another object. The presence of a light source is essential for visibility within the scene. Any 3D object can be added to the scene, and its construction is based on the combination of a geometry and a material. There are several fundamental geometries available, as well as an array of materials. The former describes the shape of the object, while the latter offers insight into its design and surface structure. The mesh is characterized by both its current position on the three-dimensional grid and its orientation. It is possible for each mesh to contain sub-meshes that are independent of the scene coordinate system. The coordinate system is a right-handed cartesian coordinate system, wherein the x-axis is aligned with the horizontal line or the width of an object, the y-axis represents the vertical line or the height of an object, and the z-axis describes the depth. The renderer is tasked with processing the images of the current view. It is responsible for mapping the 3D grid onto the 2D plane screen. It requires the camera object and the scene, as illustrated in Figure 3.1. A group is defined as a list of meshes that are represented as a singular entity, eliminating the necessity for a unique parent object. Unlike a child-parent relationship, which is established when one mesh is added to another, a group is a self-contained unit.

The Debug tool³ is a third-party Chrome *Web Browser* extension, that is utilized for the purpose of debugging the current *Three.js* rendering. It shows metrics, and meshes, and enables the user to hide an object dynamically.

3.3 Ember.js

*Ember.js*⁴ is a *JavaScript* framework that helps to create a Single-Page WebApp. The initial release occurred in 2011, and the current version is *Ember.js* 5.8. *Ember.js* is build on several core concepts. There are templates, which contain HTML. It is used as the scaffold for the *User Interface (UI)*. Components are usually small HTML fragments. In parallel to the hbs file (in which the HTML fragment is stored, there is a *JavaScript* file), that make the *HTML* UI interactive. The next major concept is the Router. It manages the navigation of the web application, such that on different paths, different pages within the application can be rendered. Finally, there is the Service. It handles the stored objects of *Ember.js* and its lifetime. Service data can be made available in different parts of the application.

³<https://github.com/oslabs-beta/BACE>

⁴<https://emberjs.com/>

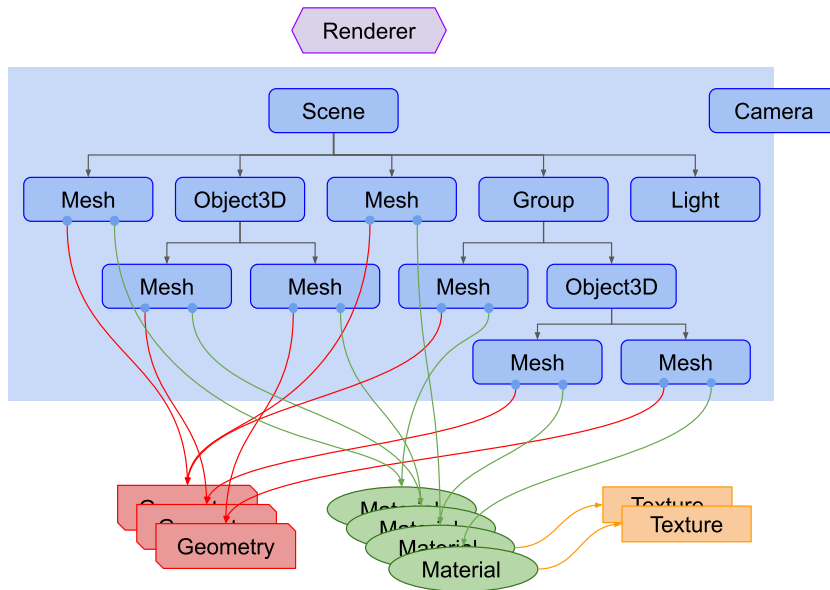


Figure 3.1. Structural composition of a *Three.js* scene that is rendered by the renderer including the camera, mesh and group objects^a.

^a<https://threejs.org/manual/en/fundamentals>

3.4 OpenTelemetry

OpenTelemetry presents itself as a "vendor-neutral open source observability framework for instrumenting, generating, collecting, and exporting telemetry data such as traces, metrics, and logs."⁵ These data are used to observe the current state of the system. For example, it helps to identify service slowdowns or failures. *OpenTelemetry* collects logs, metrics, and traces. Given the existence of numerous vendors who collect data in varying formats, *OpenTelemetry* is regarded as a standard for the collection and transmission of telemetry data. It replaces the previously utilized OpenCensus and OpenTracing. The standard uses the *OpenTelemetry protocol (OTLP)*. In *OpenTelemetry*, a span includes information about the *name*, *parent span id*, *start and end time*, *span context*, *attributes*, *span events*, *span links*, *span status*. The *parent span id* can be empty in case of a root span. The *span context*, *name* and *timestamps* are mandatory. Traces⁶ represent a series of events that have occurred within the application and are represented by a collection of spans. Spans contain tasks that have been carried out within a trace. Metrics⁷ are measurements of systems states. The metric event is the instant that a measurement happens. It can be used for either alerting a user

⁵<https://opentelemetry.io/docs/>

⁶<https://opentelemetry.io/docs/concepts/signals/traces/>

⁷<https://opentelemetry.io/docs/concepts/signals/metrics/>

3. Foundations and Technologies

when there is an unexpected value or to query real-time data. Logs⁸ on the other hand represent text based information that contains timestamps.

OpenTelemetry can instrument a variety of programming languages, with a focus on distributed tracing [Hansen and Hasselbring 2024]. Using the automatic instrumentation, the application can be traced changing only its configuration prior to the startup. It does not require any internal imports. A third-party tool called "inspectIT Ocelot" adds internal function calls to the automatic instrumentation [Hansen and Hasselbring 2024]. The *OpenTelemetry* collector is the component responsible for receiving the *OTLP* data and can export that data to a multitude of other services, including *Prometheus* and *Jaeger*. *OpenTelemetry* is used to provide (dynamic) data for *ExplorViz*

3.5 ExplorViz

ExplorViz is a system visualization tool that enables the visualization of both dynamic (live monitoring traces) and static information (such as code structure) of a system [Hasselbring et al. 2020], [Fittkau et al. 2017]. The *backend* structure is illustrated in Figure 3.2 and is composed of numerous discrete services. Each monitored application transmits *OpenTelemetry* traces via *gRPC* to the *OpenTelemetry* collector which then exports the data via *Apache Kafka*. The analysis component of *ExplorViz* prepares the data for the visualization in the *UI* [Hansen and Hasselbring 2024]. *ExplorViz* employs a web-based application as its *frontend*, aiming to reach a broad audience and provide an optimal user experience through collaboration. The *frontend* accesses multiple backend services, including the user, collaboration, and span services. This separation enhances flexibility. The collaboration service leverages *WebSocket* technology to enable real-time communication. In this project, the *Three.js* library is utilized within the *Ember.js* library to create an integrated experience. *ExplorViz frontend* initially presents a selection of a landscape. Each landscape contains all of the recorded traces and presents them in a visual format [Fittkau et al. 2017]. The communication between system components is illustrated by yellow curved lines and black arrows, as can be observed in the Figure 5.3 (a).

The visualization can be displayed on multiple end-user devices, including personal computers, tablets, and smartphones. Any browser capable of supporting *WebXR* can be used to enter a virtual or augmented reality perspective [Krause-Glau et al. 2022]

⁸<https://opentelemetry.io/docs/concepts/signals/logs>

3.5. ExplorViz

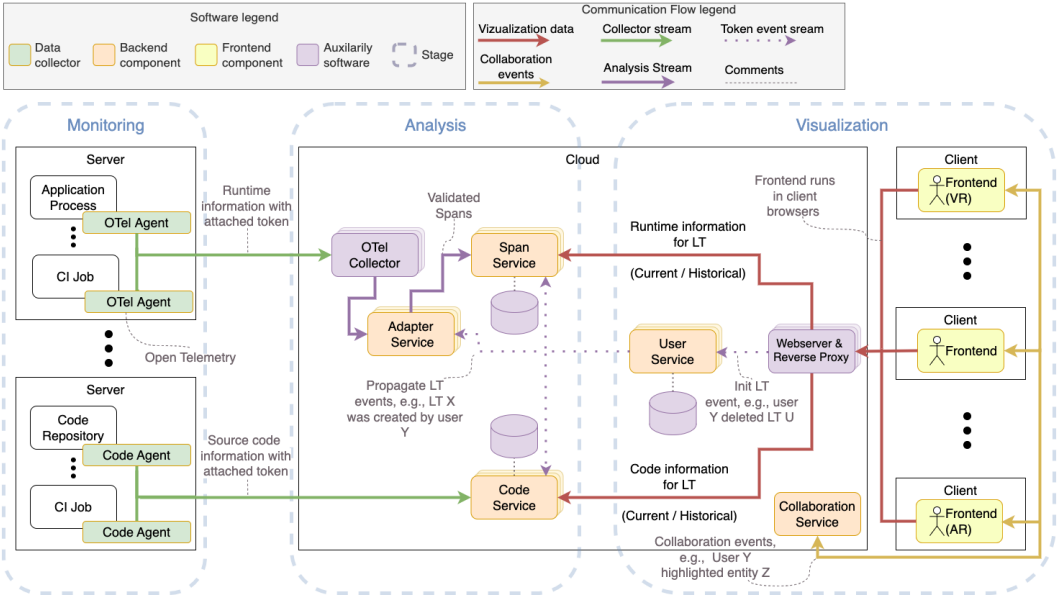


Figure 3.2. Conceptual structure of all components in ExplorViz^a as well as the *OpenTelemetry* collector and the instrumenting agent at the monitored application.

^a<https://explorviz.dev/3-architecture/>

Related Work

In order to gain a more comprehensive understanding of existing related work, the search is focused on identifying *Semantic Zoom* features within existing software, with particular attention to any software visualization that employs *Semantic Zoom*. However, the latter did not yield any suitable results and was therefore excluded from further consideration. The papers that were included in this chapter serve to motivate and inform the design of the proposed semantic zoom feature.

IslandViz IslandViz [Misiak et al. 2018] is a software visualization tool for OSGi-based applications, which are built on top of Java. In contrast to other software visualizations that use the city metaphor, IslandVis utilizes an island-based metaphor. The ocean represents an entire software. An island represents a bundle and is divided into subregions. Classes are represented by buildings that are joint in regions. These regions are Java packages. This approach bears a resemblance to *ExplorViz* using the city metaphor. The line of code metrics is encoded in the height of the buildings. An arch between two buildings represents the package dependencies [Misiak et al. 2018]. It is unclear whether a type of semantic zoom or other zoom-based alteration of objects is present.

Ontology Graph A *Semantic Zoom* feature for a 2D ontology graph is described in [Wiens et al. 2017]. The graph $G(V, E)$ contains vertices and edges. The graphical representation of such networks is frequently not straightforward due to the presence of intersecting edges or a high degree of visual complexity. The graph is therefore challenging to interpret, and the capacity of the human brain to process data is limited [Wiens et al. 2017]. The described "Visual Appearance Layer" has two distinct directions of operation. In one direction, it removes information from the graph, including details such as vertices or edge labels. In the other direction, it aggregates information, combining multiple edges between the same vertices for instance [Wiens et al. 2017]. The Topological Layer describes a simplification of the topological graph structure. Discrete topological levels of detail are assigned to each class. These levels allow the user to adjust the visualization to a specified level of detail [Wiens et al. 2017]. The published approach follows the visual information-seeking mantra proposed by Shneiderman, which is defined as follows: "overview first, zoom and filter, then details-on-demand" [Wiens et al. 2017]. This mantra offers an interesting perspective on the *Semantic Zoom* and the *Immersive View* implementation for *ExplorViz*, both of which regulate the flow of information to the human eye. A study was conducted with

4. Related Work

12 participants to evaluate the effects of the new *Semantic Zoom* feature. The participants were divided into two groups. Each individual was required to complete tasks with and without the use of the *Semantic Zoom* feature. The order of the tasks differed depending on the group. Following the completion of the tasks, the participants were asked to rate them on a number of criteria, including readability, visual clarity, information clarity, navigation support, and layout stability. The results demonstrated that the version with semantic zoom outperformed the version without, as indicated by the ratings [Wiens et al. 2017]. The evaluation method in question can be employed in the assessment of this work with regard to the user evaluation and group separation. The rating system for tasks is designed with a different methodology. As the ontology graph is based on the assessment of tasks using a range of criteria, this study aims to enhance the productivity of tasks by measuring the time taken for each task and comparing it between a version with semantic zoom and one without. The proposed Visual Appearance Layer bears resemblance to the registration of *LoDs* per 3D object, as outlined in Chapter 5. However, there is currently no aggregation of information, resulting in each object operating independently of one another.

Zoomable Multi-Level Tree (ZMLT) Another paper with a similar focus to [Wiens et al. 2017] is the Zoomable Multi-Level Tree. The authors [Luca et al. 2019] set out seven properties that their "Zoomable Multi-Level Tree (ZMLT)" algorithm must fulfill. However, the algorithm is still designed for a 2D representation [Luca et al. 2019].

The seven features that need to be covered are [Luca et al. 2019]:

1. Appropriate representation of the abstract tree originated from the graph
2. Appropriate layout
3. Show Real path from the original graph only
4. Persistent: if an information is visible in this level. It has to be visible further down the tree
5. Overlap-free for labels
6. Crossing-free edges
7. Compact: enough drawing space for all labels

The paper creates sub-graphs based on the original graph using the Steiner tree problem to identify the most important nodes. $V_1 \subset V_2 \subset \dots V_n = V$ V_1 has the most important nodes In an iterative step for all sub-trees, it draws an overlap-free graph.

OntoTrix OntoTrix is a visualization technique to navigate large datasets and their relations [Bach et al. 2011]. It uses adjacency matrices to visualize dense portions of the graph structure underlying the ontology, thereby enhancing the visual scalability. But does not show structural aspects within these dense graph parts any longer [Bach et al. 2011]. The Lin Log model is introduced in the paper. It is an energy algorithm designed by Andreas Noack to provide an efficient way to minimize the energy in a force-directed cluster graph [Bach et al. 2011]. Given that the paper is primarily concerned with layout rather than level of detail, it is not a central component of this work. Nevertheless, the establishment of cross-free communication lines may prove beneficial.

EvoSpaces EvoSpaces [Alam and Dugerdil 2007] is about a software system visualization as a 3D city that can be discovered. In terms of data representation, the EvoSpace city bears resemblance to *ExplorViz*. A software package is organized within a district, analogous to a foundation component in *ExplorViz*. The visualization employs the use of city halls, houses, apartment blocks, and skyscrapers to display a variety of software components, including header files, code, and other relevant elements. The EvoSpace system offers the capability to render building walls transparent, thereby exposing the internal file structure. In this model, each floor represents either methods, functions, or other types of elements. The authors do not indicate whether the different views are automatically triggered at specific zoom levels [Alam and Dugerdil 2007]. The paper provides some basic ideas for the later introduced *Immersive View* for classes in Chapter 6. The paper's method of displaying file content through transparent walls enables the user to enter an *Immersive View* with context. The internal class structure of the file is represented by three floors, which is similar to the three rows in Chapter 6. These rows provide a summary of the class content, ranging from inheritance, class variables, to class methods.

Polyfocal Projection & Fisheye Views The paper [Keahey 1998] provides an introduction to polyfocal projection, fisheye views, and focus + context. It introduced the concept of nonlinear magnification, as illustrated in the accompanying Figure 4.1 for 2D. The paper describes how many focus+context techniques expand the spatial area while simultaneously reducing the surrounding areas through compression. However, it does not incorporate additional detail within the acquired space.

The implementation of this system displays an atlas with images placed on top. In instances where the focus + context feature is utilized, the images are rendered in a larger size, and a per-pixel Z-buffer rendering is employed. The interesting part for our topic is the mentioned two-layer view. Layer 1 has only basic details, while layer 2 has advanced details and is only shown when a magnification is triggered in that area. Layer 2 is pulled in front of layer 1 and has a seamless handover between the two layers. As an illustrative example, a country map with an underlying city map was introduced. When the area is magnified, the city map, which is of greater detail, is only pulled in front. For further details, please refer to figure 12 in [Keahey 1998]. Another noteworthy section is an example

4. Related Work

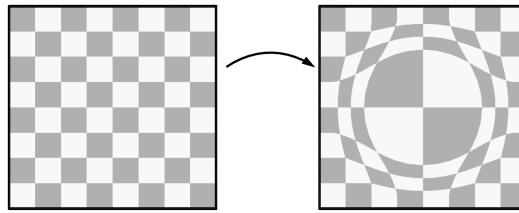


Figure 4.1. "Single Image Magnification" Figure 10 from [Keahey 1998].

of semantic zoom in conjunction with level of detail, termed "semantic levels of detail." For their interactive atlas, three levels of detail are provided. Level 0 is the image of a represented castle, level 1 is an iconic representation of a castle, and level 2 is a colored square [Keahey 1998]. The concept of the "semantic levels of detail" is also employed in this work regarding the *Semantic Zoom* approach in Chapter 5.

Semantic Zoom for Scatter Plots The paper [Buering et al. 2006] focuses on the interaction with a scatter plot on small screens and describes the use of semantic zoom and fisheye distortion to enhance the user experience. The *Semantic Zoom* feature can be used to abbreviate paragraphs once zoomed out or present images as thumbnails with varying resolution for each *LoD*. The paper combines the geometric zoom with *Semantic Zoom* to change from the scatter plot overview to a detailed view. A two-step interaction was implemented. The first step is to zoom into a specific region of the scatter plot. Data points that fall outside the view are removed. The resulting view is a sparser plot, where the new gaps can be utilized for other purposes. The second step utilizes the new gaps to present the user with additional information about the data point. For instance, the data point may transition from a gray dot to a gray box, with text displayed within the box [Buering et al. 2006]. The fisheye distortion is optimal in terms of providing details and the surrounding context in a single view. The paper implements a scatter plot interface for the user with a detail+context based fisheye view [Buering et al. 2006]. An experiment was conducted using PDA handheld devices on a scatter plot with 10 tasks. A total of 24 participants, aged between 19 and 33 years, took part in the experiment. The experiment compared the time needed to complete the tasks using either the semantic zoom feature or the fisheye interface [Buering et al. 2006]. The results demonstrated no statistically significant difference in task completion times. The setup of the experiment can be used as a basis for the evaluation in Chapter 7.

Google Maps Although Google Maps¹ is not a 3D application by default, it is evolving into a 3D representation with its "Earth" feature. The "Earth" or globe feature is a realistic bird's-eye view of the world based on images. While the Google Maps 2D representation

¹<https://www.google.de/maps>

only displays an abstract view of the streets. They use *Semantic Zoom* in combination with landmarks, all sorts of labels and *PoI*'s such as shops. The labels change their font size, number of repetitions in the visible part of the view, and the position of the label. This is especially the case for large objects, such as a road or a border, that extend beyond the user's view after zooming in. A similar approach can be used for the communication lines and their communication direction arrows that float above the line in Chapter 5. The 3D "Earth" functionality limits their *Semantic Zoom* features to *PoIs* and labels, as it is not feasible to obscure minor roadways in the context of an abstract representation.

Semantic Zoom

5.1 Analysis of the Current ExplorViz State

ExplorViz displays a comprehensive list of available landscapes on its *UI*. The user is given the option to either open an existing landscape or to create a new one. Each landscape is a container for traces and spans of distributed systems. If the correct authentication credentials are provided, any system is able to transfer data to a landscape. The foundation is the ground-level element, which represents a system. A further component is constructed on this foundation. In the case of a Java application, it is a package. It is possible for multiple packages to be stacked on top of one another. The next significant component is a class element, which can be found at any level of a package or component. In its entirety, the data is structured in a tree-like manner for each application, as illustrated in Figure 5.1. The user has the option to determine the depth of the tree structure. A context menu is available via the right mouse button, and the display of additional information is dependent upon the location of the mouse cursor. Further options are concealed within the settings menu, accessible via the button situated in the top right corner.

From the perspective of a software developer, the *ExplorViz frontend* employs *Ember.js* and *Three.js*. *Ember.js* is utilized to regulate the settings, while *Three.js* is responsible for displaying the data obtained from the *API* of the *ExplorViz backend*. The 3D environment is comprised of a scene, a camera, and 3D objects. The 3D objects are represented by the code through class meshes. Each class has different attributes and functions, and is derived from the *Three.js* "Mesh" class. The *ForceGraph* is responsible for the layout and is therefore fed with all the telemetry data. Upon a refresh of data, the *ForceGraph* is completely recreated, which can result in the sudden movement of 3D objects and a change in the user's view.

5.2 Fundamental Appearance Options

The following list shows basic operations to alter the appearance to encode data. The concept of visual variables draws its inspiration from the study of sign systems, as conducted by Bertin. The original identification of seven methods for encoding information visually has been expanded to a total of 12. The utilization of visual variables on a marker in a map or analogous visualization is a fundamental aspect of this process [Roth 2017].

5. Semantic Zoom

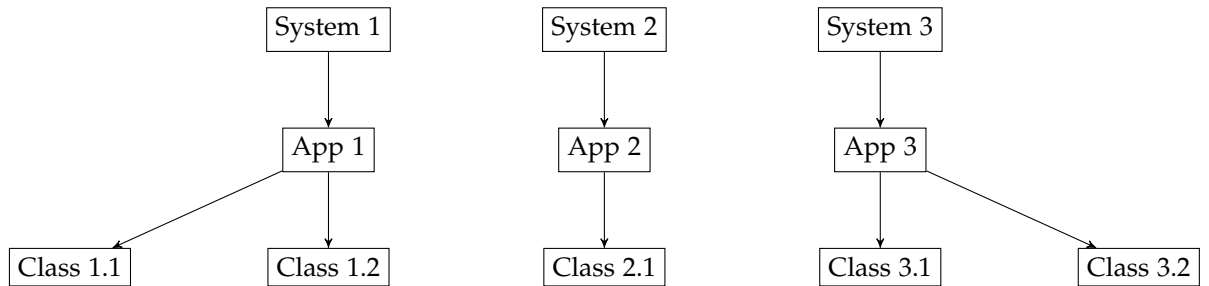


Figure 5.1. ExplorViz underlying tree data visualized for three instrumented applications.

1. Location - Primarily utilized in coordination systems to represent the spatial components of information.
2. Size - Changing the scale of an object to encode data.
3. Shape - It characterizes the external form in terms of its shape and structure. This form may be abstract or highly distinctive.
4. Orientation - The orientation of the object in a given direction has the potential to encode data, thereby indicating the direction of flow.
5. Color hue - Defines the primary color wavelength.
6. Color value - This color scheme is frequently employed in situations where the color value progresses in a single direction, traversing a multitude of color hues. Its usage is analogous to that of a heat map.
7. Texture - The term "texture" is used to describe the pattern that is created on the surface of an object when it is filled.
8. Color saturation - Saturated colors possess a single, pronounced peak within the visible spectrum, whereas desaturated colors are distributed more evenly across the visible spectrum.
9. Arrangements - The arrangement may be perfectly ordered, as in a grid, or it may be completely random.
10. Crispness - Is the sharpness of a marker.
11. Resolution - This refers to the spatial resolution at which a marker is displayed.
12. Transparency - The alpha channel serves as the primary control for blending objects in and out, and its value determines the degree of transition.

5.3. New Designs, Metrics and Aggregations

Bertin's theory states that the visual variables are processed by the eye itself, negating the necessity for cognitive brain processing, which consequently results in accelerated recognition. The color hue and the shape are of equal importance and, as a result, cannot be compared to any highlighted markers. There is no dominant visual variable in color hue or shape. The effect is not the same when it comes to color value or size. In the case of a darker color on a white background, it is observed that the color in focus attracts more attention than a lighter color. The same phenomenon can be observed with regard to the size of the elements in question. The perception of an object's size affects the amount of attention that is directed towards it. In general, larger objects tend to attract more attention from observers than their smaller counterparts [Roth 2017].

All of the mentioned visual variables in the previous list can be used in our 3D rendering, although the *Resolution* and *Arrangements* can be neglected.

5.3 New Designs, Metrics and Aggregations

The following metrics have been integrated into the system to demonstrate the proof of concept. These metrics are either newly introduced in the overall rendering process or exist in some form prior to this implementation.

1. The height of a class mesh is modified in accordance with the request count of the dynamic analysis.
2. Method meshes are incorporated into class meshes with the objective of demonstrating the number of methods present within a given class. The use of alternating colors serves to differentiate between methods. The height of each individual method mesh indicate the *Lines of Code (LoC)* relative to the other methods in the same class.
3. Class labels change the size of their text.
4. Labels exceeding a predefined length are trimmed and subsequently appended with three dots, signifying that the original label was of a greater length.
5. The communication lines are capable of modifying the line thickness in order to reduce the degree of view blockage.
6. The communication lines undergo a transformation in their curve, thereby increasing the arc.
7. In the event of a limited number of transmissions, communication lines may be hidden.
8. The visibility of communication lines or direction indicators is reduced when the distance between the observer and the relevant reference point is too great.

5. Semantic Zoom

The initial item 1 is a boolean value that is triggered at an early stage. Item 2 is independent from 1 and is displayed whenever there is a class object visible. Item 3 alongside with 4 are connected. The labels are available in multiple levels, as the cutting length can be adjusted, and the font size is dependent on the distance to the object. The width of the lines in reference to item 5 is subject to a proportional reduction or expansion in accordance with the degree of zoom, occurring in discrete steps. Item 7 is linked to the crucial importance of the underlying communication. For example, a communication that occurs infrequently is given less weight than one that occurs frequently.

5.4 Concept and Pre-Implementation Thoughts

Given that *ExplorViz* is a 3D visualization that allows users to hide and show different parts of the underlying tree through human interaction, it is crucial that the new feature does not override the existing function. Accordingly, the methodology differs slightly and does not directly interact with the underlying data, but rather with the 3D representation of said data. This approach offers the advantage that any 3D object within the scene, regardless of whether it is part of the graph, can possess the capability to alter its appearance based on the level of zooming.

Three.js incorporates a built-in feature for *LoD*, which can override the mesh depending on the distance to the camera. However, the new meshes must be known prior to their actual display, and the feature does not allow for dynamic creation of new meshes or any other function calls, which limits its utility for dynamic changes. As a discrete *LoD*, a new implementation with these missing features must be found.

The next significant topic for consideration is the run time. The initial version involved the calculation of different *LoD* during the creation process. However, this resulted in an increase in memory usage and the corruption of some flexibility, such as a dynamic change dependent on a property during run time. Consequently, a more dynamic approach was introduced, which also permits developers to seamlessly integrate their own *LoD*. In this approach, the processing of the new mesh is embedded within the run time. This reduces memory consumption but increases calculation time.

This led to the implementation of additional measures aimed at reducing the processing time at an earlier stage. One such measure was the use of clustering. It involves reducing the number of checks performed on each individual 3D object to determine whether an update was required. Initially, only elements within the camera's field of view were updated, but this changed as described in Section 5.5.2.

5.5. Implementation of Semantic Zoom

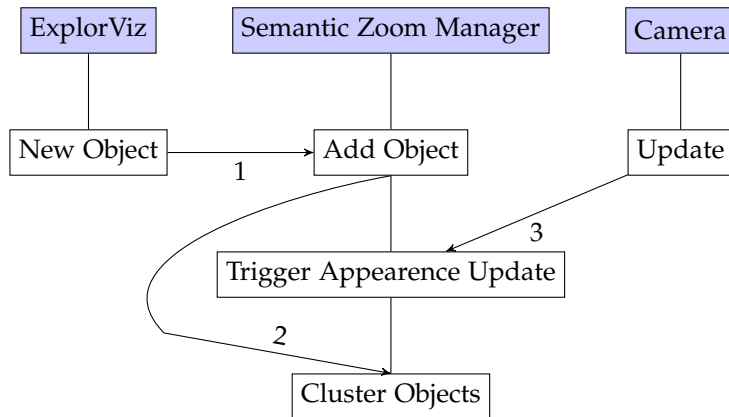


Figure 5.2. Workflow for zoomable objects that are created by *ExplorViz*, managed by the *Semantic Zoom* manager and updated by the camera.

5.5 Implementation of Semantic Zoom

As the fundamental data forms a tree, it is feasible to integrate a *Semantic Zoom* functionality based on this data set. However, there are some inherent limitations, as outlined in Section 5.4. There were multiple conceptual approaches available to implement the desired functionality. The Figure 5.2 provides a general overview of the workflow. Any new 3D object created by *ExplorViz* for the *frontend* will be added to the *Semantic Zoom* manager. This represents the initial stage of the process. Once all objects have been incorporated, the manager initiates the clustering procedure, as illustrated in Figure 5.2 of the workflow *Semantic Zoom*. Given the potential for new objects to be generated, they must also be included in a cluster. As the camera is moved, a function of the manager is triggered, which then determines the appearance level of each cluster. Thereby the entire cluster receives an update to display the appropriate *LoD*. The source code of the *Semantic Zoom* implementation is available for review in this archive: [Bamberg 2024b].

5.5.1 Appearances

It is possible for any 3D object to exhibit a multitude of visual manifestations. These manifestations are organized in a sequence beginning with the value of zero and continuing through the positive integer x . The value of zero (0) represents the original state and should also be the smallest in terms of boundary box and shape complexity. Each subsequent appearance is larger and more complex than the previous one, progressing from 0 to x . The value of x may vary for each 3D object. Upon request for an appearance above level $> x$, the appearance of x is returned. In the event that an appearance between 0 and x is not defined, it will be bypassed in the process of fulfilment of the request. An appearance

5. Semantic Zoom

may be either a "recipe class" or a function that can be called. In the event that the object in question is a "recipe class", it will contain a recipe that describes the manner in which its own appearance may be altered. More information can be found in Section 5.5.1

The appearances are called consecutively. This implies that following a level change from level 4 to 3, all changes will restart from the beginning once more. For example, the sequence would be 0, 1, 2, 3. In the event of an update from level 3 to 5, only the changes for levels 4 and 5 will be triggered.

Recipe Class

The "recipe class" can be considered the equivalent of a registered function call, but it is mutually exclusive with regard to each other. The "recipe class" class represents an effort to provide software developers with a straightforward interface for modifying an existing 3D object and then saving or restoring it. The proposed solution is a straightforward one: It allows the user to save the object's current shape and color, which can be restored later. This is achieved by saving fundamental information such as scale, width, height, depth, position, and color for a box mesh. A similar process is employed for spheres. However, when the shape in question is more complex or a combination of multiple shapes, the mesh is saved along with the material. The latter requires more memory and time to recover.

5.5.2 Algorithms To Decide on Level of Detail

This section is divided into three parts. The initial section introduces a clustering algorithm and its implementation. The subsequent section describes the creation of an array used to determine the current appearance level. Finally, it presents the combination of these elements in the manager class.

Clustering

A simplistic approach would be to calculate the distance of each object from the camera and then determine its appearance based on that information. The Figure 5.3 illustrates the center points of both visible and invisible objects within the scene, represented as a dot. For the purpose of this illustration, communication lines on top have been excluded. However, this approach may result in suboptimal performance, which is not an ideal scenario for the user. Consequently, the implementation of a clustering method to group objects together can significantly reduce the distance calculation and shift some of the calculations to a preprocessing stage. This work focuses on centroid-based clustering, as the provided center point can be used as a trigger point. Therefore, any clustering method that does not utilize centroids is unsuitable for this use case. While a density-based cluster like DBSCAN has its own advantages, it is not effective for non-convex structures and provides no centroids. Figure 5.4 provides a summary of well-known clustering methods with examples and their

5.5. Implementation of Semantic Zoom

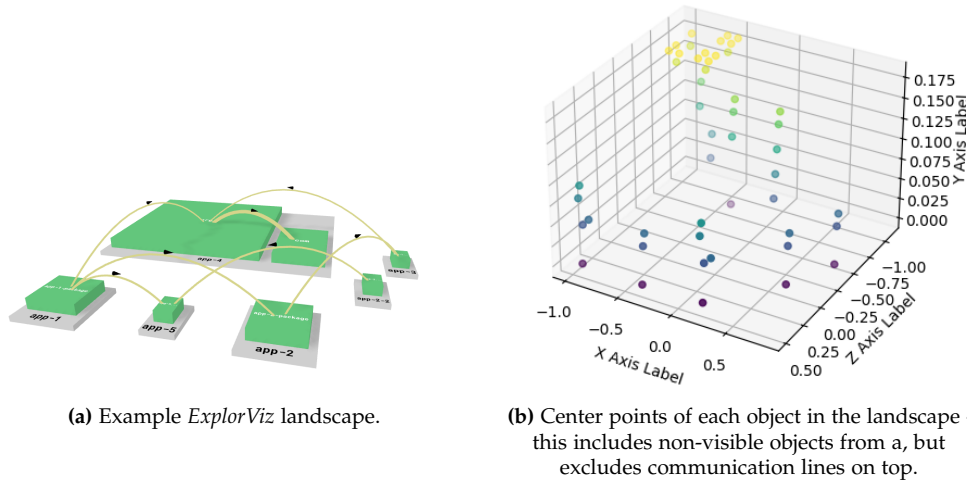


Figure 5.3. From 3D view to object center Point view.

run time. Two algorithms have been implemented: k-means and mean shift. Both provide centroids and have different run time behavior.

The implemented clustering algorithms have been designed with an interface that allows for straightforward replacement with any alternative clustering method that employs centroid-based operations. The interface includes a function that enables the clustering of an array of objects. In this case, the 'clusterMe' function accepts any array of 'SemanticZoomableObject' and returns a map, where the key is a centroid 3D vector and the value is an array of the 'SemanticZoomableObject'. The function 'addMe' has two parameters: the first is the existing map from the previous clustering process, and the second is an array of new data points. The 'counterSinceLastReclusteringOccured' indicates the number of additions that have occurred without reclustering.

K-Means Clustering

The k-means clustering algorithm serves as the fundamental approach. The run time of k-means is given by $O(nkdi)$, where n is the size of the dataset, k is the number of clusters, and i is the maximum iteration count, and d is the dimension of the input data vector n^1 . The k-means method requires the predefinition of the number of clusters and the placement of the center points of k clusters at any random position. The naive sharding centroid approach involves the placement of the cluster points based on the sum of each vector and subsequent sorting. It then proceeds to divide the data into equally sized shards, a

¹https://en.wikipedia.org/wiki/K-means_clustering

5. Semantic Zoom

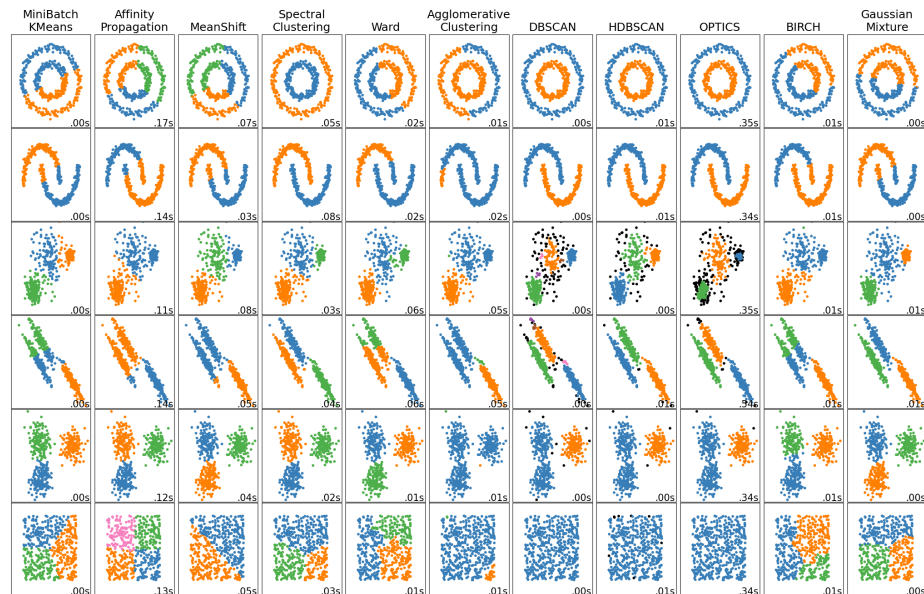


Figure 5.4. Illustration of different clustering algorithms and their runtime^a.

^ahttps://scikit-learn.org/1.5/auto_examples/cluster/plot_cluster_comparison.html

process that can be achieved by jumping through the number of rows n . The step size is determined by n/k .

Figure 5.5 show the application of k-means on the dataset of an example *ExplorViz* landscape. The red marks annotate the centroid position of each cluster. The clustering process used parameter $k = 3$ for the illustration. The implemented version employs a percentage of all objects within the scene as the k value.

Mean Shift Clustering

The Mean Shift clustering algorithm requires the bandwidth parameter, which is used to pull nearby data points into its cluster. By pulling data points towards each other, it automatically generates separated clusters. In contrast to the k-means clustering algorithm, a predefined number of clusters is not required. The runtime complexity is defined by $O(in^2)$, where n is the size of the dataset and i is the number of iterations [Ren et al. 2014]. Therefore, it is less efficient for larger data sets compared to k-means.

Adaptations to the Cluster Algorithm

It is possible for an object to be associated with multiple *Point of Interests (PoIs)*. *PoI* is a coordinate in the three-dimensional room that is associated with an object. To illustrate, a

5.5. Implementation of Semantic Zoom

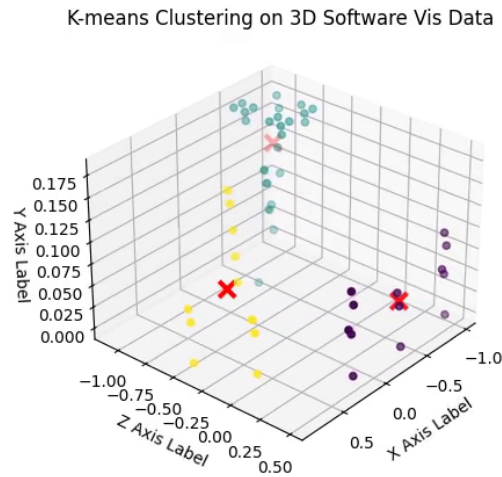


Figure 5.5. The positions of the objects mentioned in Figure 5.3 clustered by k-means with $k = 3$. Each color represents a cluster and the red 'x' is the center point of the cluster.

line may employ multiple *PoIs* to trigger another appearance when one of the numerous points is in close proximity to the camera. Consequently, it is essential to modify the outcome of the clustering algorithms in a manner that allows for a single object to be clustered in multiple clusters.

fastAddToCluster is a function that rapidly adds additional objects to the existing cluster centroids. It is necessary when new 3D objects are generated due to new metric data. However, such a process would be too time-consuming if it were to interfere with the user's interactions. In its initial implementation, the algorithm identifies the nearest neighbor clustering centroid and attaches itself to that cluster without modifying the position of the centroid in question.

Creation of the Zoom-Level-Array

The objective of Algorithm 1 is to adapt the zoom level array in a dynamic manner, based on the elements that are provided. The algorithm requires the *DiscreteLevel* array, which contains integers from 0 to 100. This represents the percentage at which the specified level should be triggered. Furthermore, the parameter "objects" contain all 3D objects present within the scene. Each element is classified according to its mesh type, such as *FoundationMesh*, *ComponentMesh*, *ClassMesh*, and *ClassCommunicationMesh*. For each group, the algorithm identifies the two elements with the greatest and least values. This is achieved through the utilisation of the bounding box functionality inherent to the *Three.js* framework.

5. Semantic Zoom

The bounding box algorithm returns a vector from the lowest to the highest corner, with the length of the vector serving as the size. The predefined levels at which a change in the appearance should occur are employed in a subsequent step. In the next step, the distance between the object and the camera is calculated so that the object's vector covers x percent of the screen based on the *FOV*. The average of the previous step for the smallest and largest element in every group is calculated. The most computationally demanding aspect of this process is the extraction of the smallest and largest objects within each category of 3D objects. This is achieved through a looping operation over all registered objects.

Algorithm 1 Build process for the Zoom-Level-Array

Require: objects: Array
Require: camera: THREE.Camera
Require: DiscreteLevels: Array<number>
zoomLevelMap \leftarrow [inf]
distinctMeshClassNames \leftarrow Set<string>
smallestMap \leftarrow Map<string,number>
biggestMap \leftarrow Map<string,number>
for $index \leftarrow 0$ to $|objects| - 1$ **do**
 add name of objects[index] to distinctMeshClassNames
 if objects[index] is smallest so far **then**
 smallestMap.set(name of objects[index], size of objects[index])
 end if
 if objects[index] is largest so far **then**
 biggestMap.set(name of objects[index], size of objects[index])
 end if
end for
for $index \leftarrow 0$ to $|DiscreteLevels| - 1$ **do**
 summedTotal $\leftarrow 0$
 for $indexmeshname \leftarrow 0$ to $|distinctMeshClassNames| - 1$ **do**
 distanceS \leftarrow calculateDistancesForCoveragePercentage(smallestMap.get(distinctMeshClassNames[indexmeshname]), camera, DiscreteLevels[index])
 distanceL \leftarrow calculateDistancesForCoveragePercentage(biggestMap.get(distinctMeshClassNames[indexmeshname]), camera, DiscreteLevels[index])
 summedTotal \leftarrow summedTotal + distanceS + distanceL
 end for
 average \leftarrow summedTotal / $|distances|$
 zoomLevelMap.push(average)
end for

The resulting Zoom-Level-Array of Algorithm 1 contains the distances between which an appearance level trigger should be initiated. The index of the array serves to indicate the

5.5. Implementation of Semantic Zoom

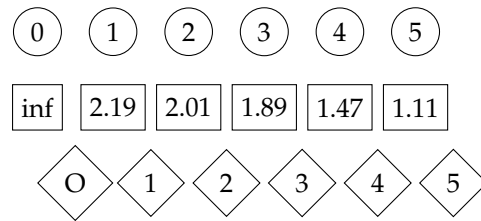


Figure 5.6. Example Zoom-Level-Array with the index of the array (top), the value of each position in the array (middle), and the final discrete *LoD* selection (bottom).

appearance level, while the value of that index represents the distance. The array begins with an index of $i = 0$, and the subsequent value of $i = 1$ serves as the demarcation point for appearance level 1. The value of *inf* for index 0 is triggered when the camera is located at a distance between infinity and $i = 1$. Consequently, the value in question must fall within the range defined by the values of $i = 0$ and $i = 1$. In general, the values decrease as the index i increases. The lowest value that can be stored in the array is 0, as there is no negative distance that can be assigned to an object. Refer to Figure 5.6 for an overview of the array structure. The circled numbers in the upper row represent the array index. The values represented in the rectangle (middle row) are the values of the array and indicate the distance at which the respective value is triggered. The bottom row, comprising diamonds, represents the appearance level that is displayed. The process is initiated with a value of zero, which represents the original appearance. To illustrate, a distance value of 3.0 is sorted between the array index (top row) 0 and 1, thereby resulting in the original appearance, as evidenced in Figure 5.6, marked by the first diamond on the left bottom row. This implementation draws inspiration from reference [De Carlo et al. 2022] in chapter 3, yet it employs a distinct approach, omitting the use of tuples spanning from "from" to "to."

The utilization of a reduction function to identify the nearest and yet smallest value relative to an appearance is illustrated in Algorithm 2.

Interface `SemanticZoomableObject`

All 3D meshes that are intended to be alterable in terms of their visual representation must incorporate an interface called "semanticZoomableObject." For this purpose, a preexisting class is available that has already been designed to implement the specified interface and can be enhanced through the use of *JavaScript's* mixin technology. The interface offers a range of methods and properties that facilitate the manipulation of a provided recipe. It enables the alteration of the appearance to a more refined level, the retrieval of the current level, and the registration of any novel appearance level. Additionally, it provides callback functions that can be overridden. A function may be registered for the purpose of executing code both before and after an alteration of the visual shape has occurred. An excerpt of the relevant interface is provided in Listing 5.1. The function "getPoI()" is employed by

5. Semantic Zoom

Algorithm 2 Distance to Level Function

Require: distanceCamToCluster > 0

Require: zoomLevelArray: Array

target ← Reduce of zoomLevelArray with the anonymous function and the startvalue of zoomLevelArray[0]

(closestSoFar, CurrentValue) =>

if *closestSoFar* > *currentValue* && *currentValue* > *distanceCamToCluster* then

 return *currentValue*

else

 return *closestSoFar*

end if

targetLevel ← Find index of *target* in zoomLevelArray

return *targetLevel*

the clustering algorithm to incorporate multiple instances of the object within multiple clusters.

Listing 5.1. Excerpt of the SemanticZoomableObject interface

```
1 export interface SemanticZoomableObject {
2   callBeforeAppearanceAboveZero: (currentMesh: Mesh | undefined) => void;
3   callBeforeAppearanceZero: (currentMesh: Mesh | undefined) => void;
4   showAppearance(
5     i: number,
6     fromBeginning: boolean,
7     includeOriginal: boolean
8   ): boolean;
9   getCurrentAppearanceLevel(): number;
10  setAppearance(i: number, ap: Appearance | (() => void)): void;
11  getNumberOfLevels(): number;
12  saveOriginalAppearance(): void;
13  setCallBeforeAppearanceAboveZero(
14    fn: (currentMesh: Mesh | undefined) => void
15  ): void;
16  setCallBeforeAppearanceZero(
17    fn: (currentMesh: Mesh | undefined) => void
18  ): void;
19  useOriginalAppearance(yesno: boolean): void;
20  getPoI(): Array<THREE.Vector3>;
21 }
```

5.5. Implementation of Semantic Zoom

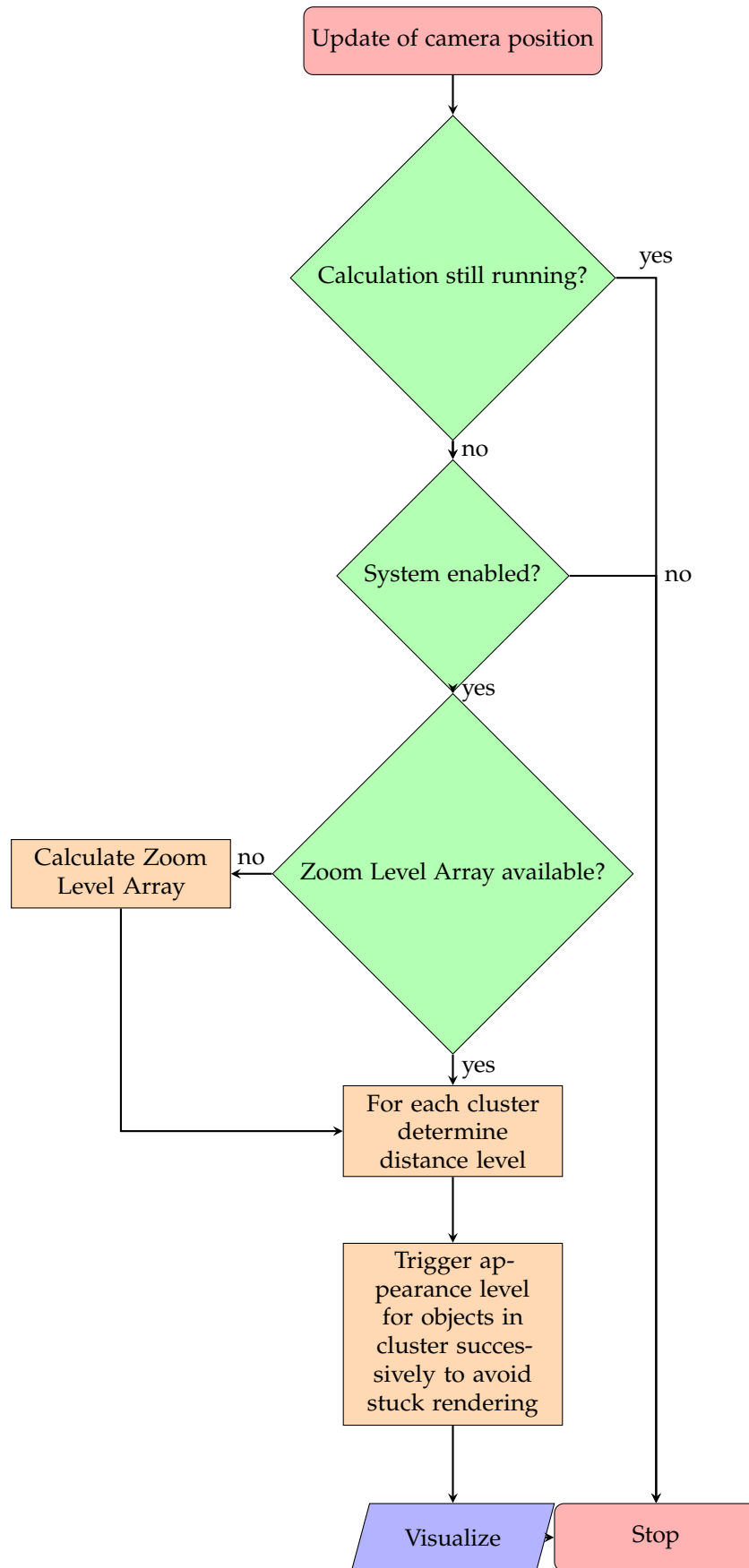
Semantic Zoom Manager

The cumulative result is the generation of a function that is triggered with each update to the camera. For further information, refer to Algorithm 3. The function is invoked with the *Three.js* camera object as a parameter. The initial assessment is to verify whether the zoom functionality is enabled. Subsequently, the Zoom Level Array is queried. In the event that the array was not constructed in a preceding phase, it is constructed at this juncture. The subsequent step involves a process of iteration over all clusters. It is also necessary to define the clusters in advance. The distance between the camera and the cluster center point is used to determine the level of appearance for the cluster members (see Algorithm 3). Only if an object in the cluster is visible, it will receive the command to change its appearance to the corresponding level. This function relies on the distance of clusters to the camera; it does not involve the angle between the camera and the cluster centroid. The whole process is illustrated in Figure 5.7. In a preprocessing step, the clustered vectors are sorted by their y-axis coordinate, such that the values of the lower objects trigger their appearance change at the beginning. This is of significant importance with regard to the "auto open/close" feature for packages. The function responsible for returning *Point of Interests* within Section 5.5.2 has the potential to result in the 3D object undergoing a process of adaptation, whereby it assumes two distinct appearances simultaneously. This can lead to an undefined state, which must be avoided. The manager is responsible for monitoring this process and ensuring that the most appropriate appearance level is consistently displayed, either that of the closer view or, in the majority of cases, the more detailed view.

Frustum

The frustum in *Three.js* can be employed to ascertain whether an object is within the field of view of a designated camera. Once an object is outside the field of view, no further updates are required with regard to its appearance level. This feature is beneficial when processing each object individually. However, this approach entails a pre-processing clustering step that can yield suboptimal results when clustering and frustum detection are employed concurrently. Once the center point of a cluster is outside of the frustum, the entire cluster is no longer updated. If the camera is positioned in close proximity to a few objects, it is highly probable that the cluster center point of the clustered objects will be outside of the view. Consequently, the user may not perceive any changes in appearance, despite the anticipated change. In light of these observations, the frustum section was found to be unsuitable for further consideration.

5. Semantic Zoom



32

Figure 5.7. Flowchart, demonstrating the *Semantic Zoom* manager's action when the camera position receives an update.

Keeping the Manager Clean

Upon the creation of a new mesh, it is appended to the *Semantic Zoom* manager. As the insertion of a mesh without the removal of existing ones would result in a memory issue, any mesh that is removed via the dispose function is automatically removed from the manager. The removal of non-existent items from the manager also enhances performance, as the list of objects can be iterated more rapidly. If the manager is enabled and the user switches to another landscape, all objects are removed from the manager.

Algorithm 3 Decision on Level of Appearance

Require: cam: THREE.Camera

```

if Semantic Zoom Manager is not enabled then
  Return
end if
if alreadyCreatedZoomLevelMap == False then
  Create Zoom Level Array
  alreadyCreatedZoomLevelMap = True
end if
for each cluster in clusters do
  if Check if cluster center point in close proximity to the camera then
    Continue
  end if
  X ← Distance between Camera and cluster
  Y ← find corresponding appearance level for distance X
  for each object3D in cluster do
    if object3D is visible then
      Trigger Appearance Level Y
    end if
  end for
end for

```

5.5.3 Programmer Interface

The developer has the option of extending any new mesh by calling certain functions in the constructor of the new object. The example in Listing 5.2 illustrates the setting of another shape/function call that is triggered when the object receives a request to change its appearance level to 1:

Listing 5.2. Register New Appearance Altering Function for Level 1

```
1 this.setAppearance(1, <function>);
```

The function call in Listing 5.3 saves the current shape of the object:

5. Semantic Zoom

Listing 5.3. Save the Current Shape and Texture

```
1 this.saveOriginalAppearance();
```

It is mandatory for the new object to be inherited from the original BaseMesh. In the event that there is already another inheritance, the use of *JavaScript* mixins can be employed to assist in this process.

5.5.4 Known Problems and Limitations

It is acknowledged that there are constraints associated with the nesting of semantic zoomable objects. Consequently, if a mesh is generated by one object that is also capable of the *Semantic Zoom* feature, it will be registered at the manager as well. However, there is no immediate trigger for the newly created mesh and its target appearance. Following an update of the camera, the newly created mesh will assume the same appearance as the original object. However, that level may already have been reached. Therefore, it is not possible to descend below that level. This behavior is observed only when the new object is in close spatial proximity to the original object. If the newly created object is situated at a considerable distance, this behaviour does not manifest. However, it is still possible that a specific requested appearance level may never be triggered.

Another challenge is the immediate incorporation of new objects into the process. In some instances, the opening of a component results in the generation of new objects that are not directly registered within the *Semantic Zoom* manager. Consequently, these objects do not undergo the visual alterations that would otherwise be expected.

5.6 User Settings and Parameters

Figure 5.8a illustrates the configurations for the various discrete levels and parameters for the clustering algorithm. The initial switch activates the *Semantic Zoom* functionality, which is analogous to the switch in the context menu. The second switch enables the auto open/close feature. When this option is selected, the "open all components" feature is deactivated. The third option allows the user to select between the k-means clustering and the shift mean clustering. The initial slider, labeled "Predefined Zoom Sets," incorporates six preset configurations that modify the levels below. Once set, the switch above turns on and only turns off again if any custom level slider is changed. In this instance, a fixed number of five levels have been defined. The percentage of the preceding level is increased for each subsequent level. The values represent the average dimensions of objects required to occupy a specified percentage of the screen. This percentage is represented by the value of a slider in the settings, denoted by the value X . If the values assigned to the levels are not increasing monotonically, the *frontend* performs a correction. In this case, the previous value is simply set to $X - 1$. If the current value is zero, the value is incremented to prevent

5.6. User Settings and Parameters

any level from having a value of zero or below. The same approach is applied to values above 100.

The last option in Figure 5.8a is used by the k-means clustering algorithm. It calculates the number of clusters by simply using all objects, that are part of the semantic zoom feature and takes a percentage of Y . The value Y is set by the settings option "Relative # of clusters". Using a value of 100% leads to the same count of clusters as there are objects in the view. These clusters can still have different center points other than the original center point of each object. So there is no option to get exactly the position of each 3D object and provide the level of detail based on each object's position. It is always only an approximation.

The Figure 5.8b illustrates the provided context menu. The final two options have been included to facilitate the activation of the system, which can be enabled via "Semantic Zoom enable". The last option, "Show SemanticZoom Center Points", is employed for debugging purposes and displays a red X at the centroid positions of the clusters.

5. Semantic Zoom

The image shows two parts of the ExplorViz interface related to Semantic Zoom settings.

(a) Semantic Zoom settings in ExplorViz frontend: This panel is titled "Semantic Zoom" and contains several settings:

- Semantic Zoom:** On (green button)
- Auto Open/Close of Components:** On (green button)
- Use k-Means instead of Shift-Mean:** On (green button)
- Use Predefined Set:** On (green button)
- Predefined Zoom Sets:** Slider from 1 to 6, currently set at 3.
- Level 1:** Slider from 1 to 100, currently set at 20.
- Level 2:** Slider from 1 to 100, currently set at 50.
- Level 3:** Slider from 1 to 100, currently set at 60.
- Level 4:** Slider from 1 to 100, currently set at 70.
- Level 5:** Slider from 1 to 100, currently set at 80.
- Relative # of clusters:** Slider from 1 to 100, currently set at 30.

(b) Available options in the right-click context-menu of ExplorViz: This menu is located in the top right corner and contains the following options:

- Reset View
- Open All Components
- Hide Communication
- Enable Heatmap
- Pause Visualization
- Open Sidebar
- Enter AR
- Semantic Zoom disable (Beta)
- Show SemanticZoom Center Points

(a) *Semantic Zoom* settings in *ExplorViz* frontend, which can be accessed via the gear icon on the top right corner.

(b) Available options in the right-click context-menu of *ExplorViz*.

Figure 5.8. Illustration of the various settings available for the *Semantic Zoom* features within the *ExplorViz* UI.

Immersive View

The visual information-seeking mantra proposed by Shneiderman is used as the baseline. The objective is to create a visualization of the interior of a class or file in a manner similar to that of Google Street View. As with the transition from Google Maps to Google Street View, the immersive view will be initiated when zooming in on a class or file. The concept of an immersive view is a general one, designed to provide a focused view. In this implementation, it is combined with the structure of a "class", resulting in the creation of an "*Immersive Class View*".

6.1 General Concept

A new scene is created, replacing the previous main scene. Given that the camera control object in *Three.js* is constrained by a maximum zoom level, it is necessary to utilize the specified point in order to gain access to another scene. Upon reaching the maximum zoom level and subsequent scrolling, the initial stage of the two-stage process is completed. In the event of another scroll occurring without any other movement, the preceding scene will fade out and the subsequent scene will appear. In addition, the outcome is influenced by the area in which the cursor is positioned. To alert the user to an upcoming change, the object in question begins to flash. The object should transition into an *Immersive View*, which is supported by the system and is in closest proximity to the cursor. To facilitate a prompt return to the preceding scene, the original scene is retained in parallel with the new scene. The new scene and camera can be employed for a variety of visualizations. However, in the case of the "Immersive Class View", the new scene is a view from within a sphere. With the exception of rotation around the fixed camera position, the camera remains stationary. Accordingly, the information must be organized in a circular configuration surrounding the camera.

Once within the *Immersive View*, the user is able to utilize the mouse wheel to zoom and enlarge objects. Should the user employ a negative zoom factor that is beyond the limits of the system, the camera will be halted at a virtual barrier. At this point, another scroll out will result in the user exiting the immersive view and returning to the previous scene. This concept enables the user to scroll into an *Immersive View* of an *Immersive View*. The entire process can be completed with the mouse alone. The process of entering an *Immersive View* is both rapid and seamless. A double click on any *Immersive View*-enabled object serves as

6. Immersive View

an alternative means of initiating the process. Furthermore, the escape key on the keyboard can be utilized to exit the view. The current work does not prioritize the integration of a touchscreen control mechanism.

As illustrated in Figure 6.1, the *Immersive Class View* provides an overview of the internal structure of an object-oriented class. The upper section of the diagram contains information about the name, implemented interfaces, and extended classes. The subsequent row incorporates the class variables, highlighting extended classes or implemented variables. The lower section of the diagram contains the class methods, along with their parameters and return types. Any extended method is indicated by a black box in the background if it is a method. A method that is enforced by an interface is boxed by a green rectangle.

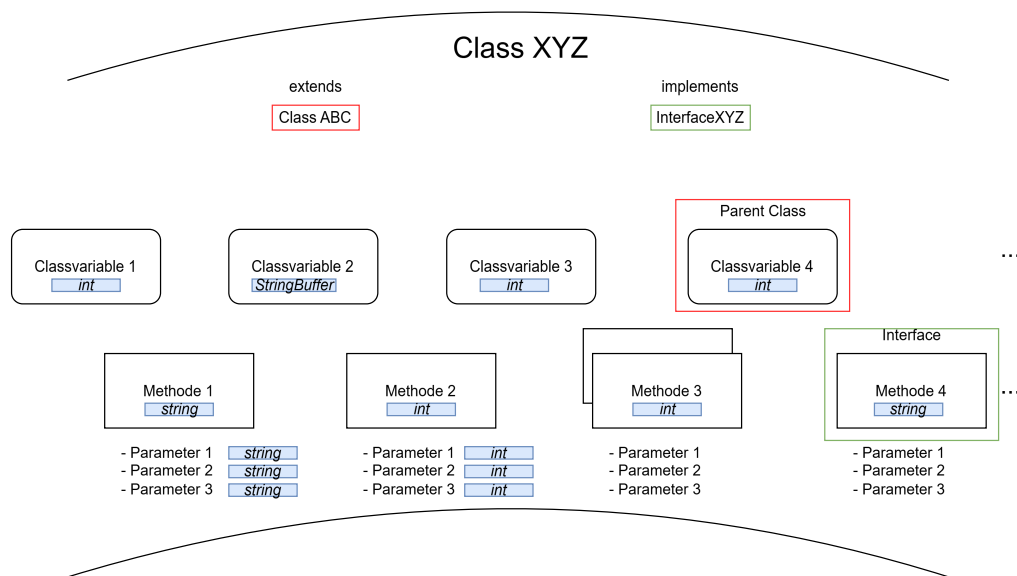


Figure 6.1. Preview of the *Immersive Class View* as a base concept.

6.2 Analyze of Attachment Point for ExplorViz

As mainly described in Section 5.1, the new feature extends the code in the following sections:

1. Any "BaseMesh" can be extended by an immersive viewable object
2. The scene renderer can replace its current scene and camera
3. Another camera control can be installed

4. Tracking the mouse actions in the original scene

6.3 Implementation of Immersive View

The initial step is to examine and expand the data model which serves as the foundation for the project. The second section addresses the initiation of the immersive visualization of an object. The fundamental functionality is limited to the actions necessary for navigation within a *Web Browser* environment, utilizing conventional input devices such as a mouse or keyboard. It does not consider the specific input modalities of touchpads or touchscreens. Once the mouse cursor has been positioned over an object that is capable of the *Immersive View* and three consecutive zoom actions have been initiated, the view is triggered. All three steps are visually highlighted, indicating to the user that the view is about to change. The third section addresses the construction of an *Immersive View*, illustrated through the example of a class. Here, the camera control has been updated to a *PointerLock* control, enhancing the *Immersive View* experience. The final section covers the steps involved in entering and exiting an *Immersive View*. The source code of this implementation alongside the *Semantic Zoom* implementation is available for review in this archive: [Bamberg 2024b].

6.3.1 Current Data Model

The existing model is illustrated in Figure 6.2 using white boxes only. At this stage, it is only capable of storing the methods of a class; there is no further indication as to whether these methods are public or private. In addition, it lacks information about class variables.

As illustrated in the Figure 6.2, the data model has been expanded to include both gray and white boxes. This process is part of the preprocessing stage. At this point in time, the extended data set is not yet available; therefore, dummy data is employed in its place. The addition of class variables and the protection type allows for a more detailed view of a class. The use of the *LoC* per method indicates the modularization of code. For each method, the following information is stored: the parameter names and corresponding types, the return value, and, in addition, data about the extended classes and the implemented interfaces. A method can be marked as overridden or abstract. The overloading of a method can be stored by referencing the others using a linked list.

6.3.2 Example Implementation Based on the *Immersive Class View*

The initial primary component is an interface that must be incorporated into each class, which displays a 3D object and aims to provide an *Immersive View*. The interface seen in Listing 6.1 provides the necessary methods for entering, exiting, and constructing an *Immersive View*. To facilitate the development process, a pre-existing class that implements the required interface has been provided, thus eliminating the need for developers to implement the interface themselves. Instead, they must only define the build method.

6. Immersive View

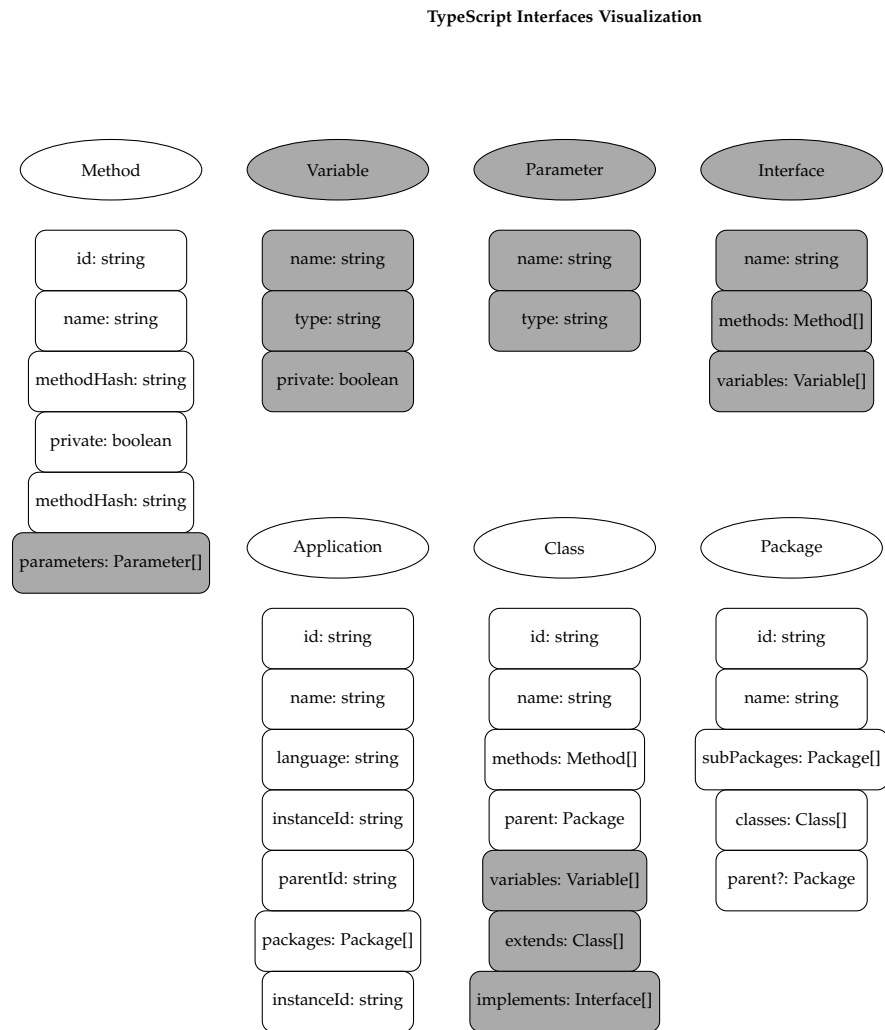


Figure 6.2. The data structure of *ExplorViz* is populated by both the dynamic and static data analysis services. The grey-highlighted boxes represent a novel addition for the *Immersive Class View*.

6.3. Implementation of Immersive View

The build method accepts two parameters. The initial parameter is the camera, and the subsequent parameter is the scene. As every utilized 3D object in *ExplorViz* is derived from the *BaseMesh* class, further inheritance was not feasible. However, a method known as "mixin" enables multiple inheritances. Consequently, any class capable of displaying a 3D object can inherit from the class and construct an *Immersive View*. The mixin class has already prepared the scene and mirrored the original cameras' properties. The scene is initially empty, except for a light source. The *Immersive View* manager is responsible for maintaining a record of the original camera and camera control, and for initiating the *Immersive View* of any registered 3D object. The *Immersive View* is tasked with triggering the exit process at the manager class.

Listing 6.1. Excerpt of the ImmersiveViewCapable interface

```
1 interface ImmersiveViewCapable {
2   buildSceneandCamera(
3     Ocamera: THREE.Camera,
4     Oscene: THREE.Scene
5   ): Array<THREE.Camera | THREE.Scene>;
6   enterImmersiveView(camera: THREE.Camera, scene: THREE.Scene): void;
7   exitImmersiveView(camera: THREE.Camera, scene: THREE.Scene): void;
8 }
```

The example view is currently an *Augmented 2D* representation, with the 3D effect employed primarily for aesthetic purposes [Teyseyre and Campo 2009]. However, this may evolve in the future, with the incorporation of historical data.

The utilization of this script¹ facilitates the bending of objects for enhanced visibility within the sphere.

6.3.3 Entering/Exiting

The scene is generated each time the user attempts to access the *Immersive View*. Two methods are provided as the default means of entry. One possible method for entering the *Immersive View* is to hover the cursor over an object that is capable of displaying an *Immersive View* and scroll into it, with the mouse positioned on top. To provide some visual feedback, the object begins to pulse in a predefined color. This feature serves to alert the user that they are in the process of entering another view. By using the mouse to rotate or pan, the user can simply cancel the process. The other possibility for entering a view is a simple double click on the object. It should be noted that all methods mentioned so far are only usable in the *Web Browser* using mouse and keyboard.

The exit is seamlessly integrated into the mouse control and may be executed with a simple zoom-out command by rotating the mouse wheel, or by pressing the escape button.

¹<https://github.com/Sean-Bradley/Bender/blob/main/src/client/bender.ts>

Evaluation

The evaluation is subdivided into 3 categories. The first category is a performance evaluation of *Semantic Zoom* compared to the version without *Semantic Zoom*. The last two are used for the user experience evaluation of *Semantic Zoom* and *Immersive View*.

7.1 Goals

1. Does using *Semantic Zoom* affect system performance?
2. Does the *Semantic Zoom* feature improve the user experience and increase productivity?
3. Is the *Immersive View* that is provided useful in the sense that it provides new information that is not otherwise available?
4. Can the user relate to the approach mentioned in a paper “overview first, zoom and filter, then details-on-demand”?

7.2 Performance Evaluation

In order to evaluate the impact of the *Semantic Zoom* feature on performance, a performance evaluation was conducted, comparing the version with and without *Semantic Zoom*. In the majority of cases [Shariff et al. 2019], performance is assessed through the use of *Frames per Second* [Hamzaturrazak et al. 2023], *CPU* usage, and *RAM* measurements. A *Web Browser* automated by Selenium is used to perform a series of actions with the *Semantic Zoom* feature enabled and disabled, and records a time series of data. Despite Selenium’s self-definition as a tool for other purposes and its referral to JMeter for performance analysis, it remains a suitable tool for testing a 3D application. In contrast to Selenium, JMeter does not execute any browser code, such as *JavaScript*, or render applications. Instead, it is designed to test the connection and the *backend* performance.

7.2.1 Setup

A computer which runs Microsoft Windows 11 with an Intel(R) Core(TM) i7-8650U CPU @ 1.90GHz with 4 cores was utilized for the performance evaluation. The computer has

7. Evaluation

an L1 cache of 256 KB, L2 cache of 1 MB and a L3 cache with 8 MB. The built-in GPU is an Intel(R) UHD Graphics 620. The Selenium setup consists of a Python 3.13.0 and a Selenium 4.25.0 installation. The controlled Google Chrome *Web Browser* runs on version 129.0.6668.90. To eliminate the 60 FPS limit enforced by Chrome in the default mode, the parameter in Listing 7.1 is passed to Chrome prior to the test by Selenium.

Listing 7.1. Chrome Parameter To Disable Frame Rate Limit

```
1 --disable-gpu-vsync --disable-frame-rate-limit
```

In order to have a more diverse selection of landscapes, the tool developed by [Bugla 2024] at Kiel University was employed to generate landscapes of varying dimensions. These landscapes can be found on *GitHub*¹. The provider for these landscapes is a *Node.js* Server running in a *Docker* container.

The FPS is calculated by determining the interval between render calls. In *JavaScript*, an array of all render call occurrences is stored in memory. This list is subsequently retrieved by Selenium. The heap size measurement is obtained directly from Selenium by retrieving the variable in Listing 7.2

Listing 7.2. JavaScript Access Memory

```
1 performance.memory
```

It contains a JSON object with *totalJSHeapSize*, *usedJSHeapSize* and *jsHeapSizeLimit*. The *usedJSHeapSize* is used in the evaluation process.

The Selenium test executes operations within the context of the landscape "XXXL world with high communication." This landscape consists of numerous application foundations and a substantial volume of communication between applications. The trace generator was utilized with the following settings:

- ▷ 35 apps,
- ▷ max package depth of 5,
- ▷ 200 classes,
- ▷ 10 methods,
- ▷ 200 communication calls,
- ▷ 2411 seed,
- ▷ true random communication style

The following list enumerates the actions performed by Selenium.

¹<https://github.com/ExplorViz/deployment/blob/main/demo-supplier/demo-data/petclinic-distributed-structure.json>

7.2. Performance Evaluation

1. Zoom-In 100px for 10 times
2. Zoom-In 100px for 20 times
3. Move Camera Left by 500px
4. Rotate Camera Left by 100px for 2 times
5. Wait for 1 Second
6. Zoom-Out 100px for 10 times
7. Wait for 1 Second
8. Zoom-In 100px for 10 times
9. Wait for 1 Second
10. Rotate Camera Up by 25px for 2 times
11. Wait for 1 Second
12. Move Camera Left by 500px
13. Rotate Camera Left by 100px for 2 times

All of the described actions are carried out by the Selenium driver at a resolution of 1680x1050 pixels. The mouse movements that are included in these actions are always initiated from the center point of the canvas and then directed toward the desired direction. The actions are initiated at the starting point designated by *ExplorViz*. In the subsequent action, the camera is zoomed in on the packages. Once the camera is positioned between the packages, a series of camera movements are executed. The next step involves zooming out and then back in again. This step represents a relatively minor zooming action in comparison to the initial zooming in. The final step comprises further camera movement and rotation. The final position of the camera is in close proximity to the packages, in contrast to the initial position where the packages were distant.

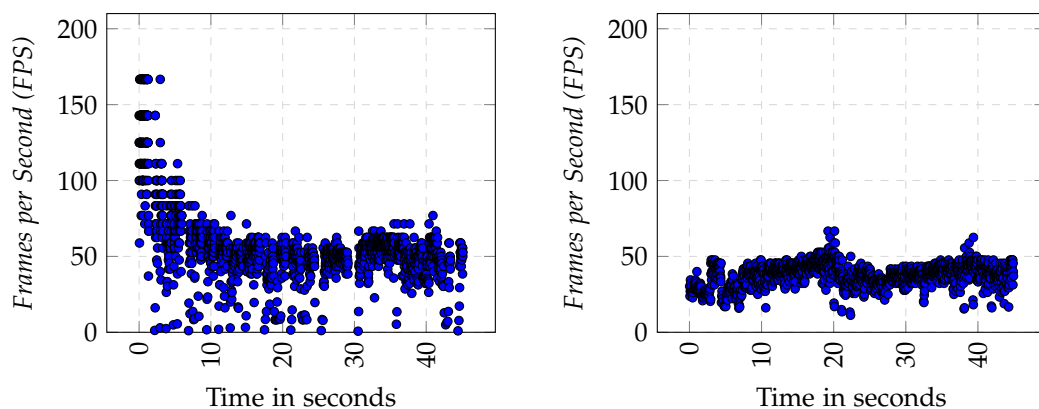
The tests are conducted with either the *Semantic Zoom* feature or the "open all components" functionality. The predefined zoom level sets are set to 3 with the auto open/close functionality. The *FOV* is set to 75 on the perspective camera. The communication line thickness is set to 0.5 with an arrow size of 1. The curviness factor is set to 1. No debugging feature is activated, including *FPS* counter, axes helper, or light helper is activated. The remaining settings are set to their default value, as they do not impact the render performance.

The Python script and the CSV results can be accessed via the following archive: [Bamberg 2024a].

7. Evaluation

7.2.2 Evaluation Results

The Figure 7.1a and Figure 7.1b contain the *FPS* data over time of both tests. The memory consumption over time is displayed in Figure 7.2a and Figure 7.2b. The data recordings in question represent a single observation of the system's behavior and, as such, do not constitute a definitive representation of the system's overall trend.



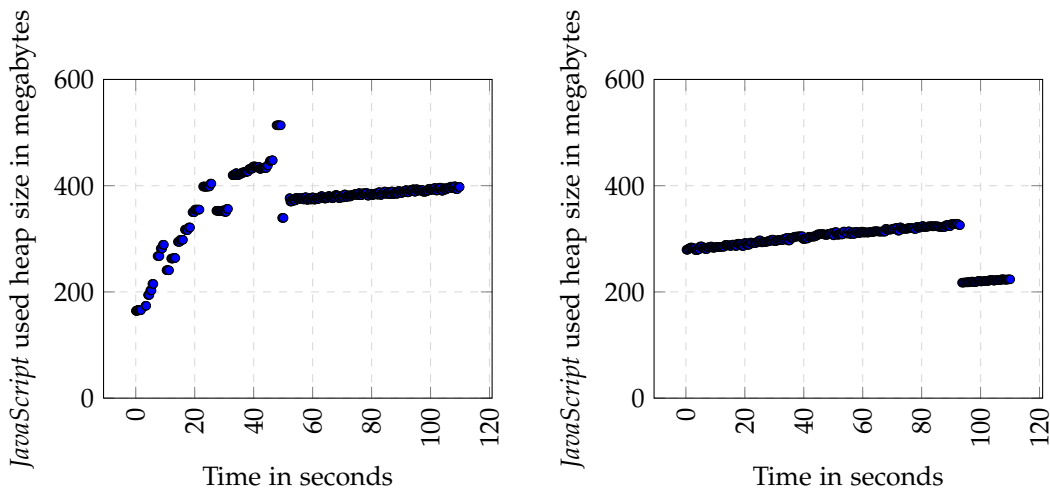
(a) FPS scatter plot of Selenium's interaction with the landscape "XXXL world with high communication" with *Semantic Zoom* enabled.

(b) FPS scatter plot of Selenium's interaction with the landscape "XXXL world with high communication" without *Semantic Zoom* enabled.

Figure 7.1. Comparison of FPS scatter plots with and without *Semantic Zoom* enabled.

7.2.3 Discussion

The data set obtained from the recorded performance evaluation demonstrates notable discrepancies. In the initial startup phase, the version without *Semantic Zoom* displays superior performance in comparison to the other version. This phenomenon can be explained by the removal of the communication direction indicator, which significantly impacts the computational resources required for processing. Upon further interaction with *ExplorViz*, the version with *Semantic Zoom* was observed to have a higher frame rate, on average, in comparison to the version without. The version without *Semantic Zoom* maintains a relatively *FPS* throughout the entire interaction, although it does not exceed 50 *FPS*. Only on rare occasions. In contrast, the versions with *Semantic Zoom* exhibit a notable decline in *FPS*, reaching nearly zero, and a surge to almost 200 fps. These fluctuations can be attributed to the auto open/close feature, as each open/close call demands a substantial computational burden. This was identified during the development phase. On exceptionally large landscapes, the *Semantic Zoom* version is capable of elevating the *FPS* counter to a level that is visually acceptable for the human eye. The memory consumption is marginally higher when the *Semantic Zoom* feature is in use, with a greater degree of fluctuation.



(a) Memory scatter plot of Selenium's interaction with the landscape "XXXL world with high communication" with *Semantic Zoom* enabled.

(b) Memory scatter plot of Selenium's interaction with the landscape "XXXL world with high communication" without *Semantic Zoom* enabled.

Figure 7.2. Comparison of memory scatter plots with and without *Semantic Zoom* enabled.

In the version without *Semantic Zoom* enabled, the memory consumption is increasing steadily, followed by a sudden drop. This phenomenon is likely associated with the garbage collector.

7.2.4 Threats to Validity

Performance Test The performance test was running on the same computer that also ran the demo supplier, which fed the *frontend* with pre-recorded data, and the node developer environment. The performance might be affected by that. The Javascript function "performance.memory" is considered to be not an accurate measurement tool, as it can overestimate the memory usage². It should be noted that the performance test was only conducted on a single occasion; consequently, there is some potential for variation in the results.

7.3 User Evaluation

In order to evaluate the usability and efficacy of the system in relation to specific tasks, a user evaluation is conducted. The user evaluation is open to all interested parties. To evaluate the system, users are presented with a task that they must complete. This task

²<https://developer.mozilla.org/en-US/docs/Web/API/Performance/memory>

7. Evaluation

primarily entails searching for a property within the 3D environment. As proposed by [Wiens et al. 2017], participants are divided into groups to compare the versions with and without *Semantic Zoom*. In addition to fulfilling the assigned tasks, participants are required to record the time required to complete them [Hansen et al. 2013]. Furthermore, users are invited to complete a survey describing their experience with the software. The complete survey and the results can be found in the archive: [Bamberg 2024a].

7.3.1 Setup

All participants completed the evaluation in a laboratory facility at Kiel University on the following system. A Windows 10 computer equipped with an Intel Core i5-6500 CPU at 3.2GHz providing 4 cores and 16GB of DDR3 memory. For 3D rendering, it uses the NVIDIA GeForce GTX 1070 with 8GB memory. The system SSD is a Samsung 850EVO 500GB model. The browser is Google Chrome in version 130.0.6723.92. The input device is a QWERTZ keyboard and a default mouse. Two NEC MultiSync EA243WM 24-inch displays with a resolution of 1920x1200 each were utilized to facilitate the simultaneous observation of the survey and *ExplorViz* by the participants. The LimeSurvey tool is employed to document the responses of each participant. This tool is an online platform for developing surveys, hosted by the university. As previously outlined in Section 7.2.1, the use of diverse landscape sizes was employed to afford the user a multitude of potential scenarios.

7.3.2 Pretest

In order to ascertain whether there are any logical or system failures prior to the actual evaluation, an external party has been designated to run the evaluation and provide feedback. One of the most strongly recommended features to be implemented was the automatic updating of the content following a change in the settings. It is essential that all tasks include concrete instructions, ensuring that all participants have a uniform underlying basis. The settings have been updated with the objective of enhancing the user experience. It is recommended that the implementation of a visual indicator be considered to demonstrate the current state of the *Semantic Zoom* feature. One proposed solution is the inclusion of an icon in the top right corner of the screen. A bug that randomly displayed pop-ups while in the *Immersive View* needed to be fixed, as it obstructed the central view of the user.

7.3.3 Introduction to ExplorViz

As part of the preparatory process, each participant is provided with a brief introduction to *ExplorViz*. This is necessary to ensure that all participants possess a similar level of knowledge, thereby facilitating more accurate results during the time-based tasks. The initial stage comprises a summary of essential functions and settings, presented in the form of a cheat sheet. This document is included in the appendix, referenced as Figure A.1. The subsequent stage involves a brief tutorial, during which the participant becomes acquainted

with the environment through exploration of the original version and, subsequently, the extended version of *ExplorViz* with *Semantic Zoom*. The designated landscape is the "VISSOFT 23 - Study Sample."

7.3.4 General Questions

The objective of the general questions was to be able to differentiate between test results based on the participant's level of experience.

1. Do you have experience in computer programming?
2. Have you used ExplorViz before?

7.3.5 Questions and Assignments for *Semantic Zoom*

The participants are divided into two groups based on the tasks they are assigned. One task requires the use of *Semantic Zoom*, whereas the other does not permit its use. The assignments are mixed, such that every participant experiences the *Semantic Zoom* feature. The time taken to complete a task is recorded.

In the final stage of the introduction, the participant is required to describe the differences between the two versions. The response provides an insight into what the participant initially noticed, and therefore carries greater value.

All assignments have a preparation stage that is not included in the time recording. This stage involves loading the appropriate landscape and adjusting the settings as described.

1. Group A: Find the class 'PetDetails' and name the underlying application name in landscape Distributed Petclinic Sample with semantic zoom.
2. Group B: Find the class 'PetDetails' and name the underlying application name in landscape Distributed Petclinic Sample without semantic zoom.
3. Group A: Name all classes (full name) in the sub package "model" of the application "app-4" in landscape Artificial Software Landscape with semantic zoom.
4. Group B: Name all classes (full name) in the sub package "model" of the application "app-4" in landscape Artificial Software Landscape without semantic zoom.
5. Group A: There is a major update of class Vet in package vet of Application app-4. Which classes might be affected by that? Name all potentially affected classes by looking at the communication lines. Use the Artificial Software Landscape without semantic zoom.
6. Group B: There is a major update of class Vet in package vet of Application app-4.

7. Evaluation

Which classes might be affected by that? Name all potentially affected classes by looking at the communication lines. Use the Artificial Software Landscape with semantic zoom.

7. Group A: Look out for the application nexus-code. There is a class called CacheInvalidator on the highest level. Only one communication line should be visible from and to the class CacheInvalidator . What direction is the communication? Inbound or outbound? Use the landscape Tracegen - XL world with high communication and disabled semantic zoom.
8. Group B: Look out for the application nexus-code. There is a class called CacheInvalidator on the highest level. Only one communication line should be visible from and to the class CacheInvalidator. What direction is the communication? Inbound or outbound? Use the landscape Tracegen - XL world with high communication and enabled semantic zoom.

The following questions are free text questions.

1. What changes did you notice between the version with and without semantic zoom?
2. Do you have any suggestions for improving the user experience while using the semantic zoom feature?

The following questions can be rated from *Haven't noticed/No answer*, 0 *No advantage/No*, 1 *Very low*, 2, 3, 4, 5 *Very high*.

1. How useful is the change of height of classes to indicate the request count?
2. How useful is the change of the font size on the class labels?
3. How useful is the indicator of the number of methods at the class object?
4. How much did you like the shrinking of the communication lines while zooming in?
5. How much did you like the feature where packets open and close automatically depending on the zoom level?
6. Did the objects you zoomed into change? Did you expect them to be triggered?
7. Do you think the semantic zoom feature is useful in daily life?
8. Do you like the overall reduced view compared to the regular view (all components open) without semantic zoom?

The following questions can be answered with *with semantic zoom*, *without semantic zoom*, or *no difference*.

1. Which version (with or without semantic zoom) was a smoother experience for the eye regarding steady FPS?
2. Which version do you prefer in terms of interacting with the 3D world?

7.3.6 Questions and Assignments for *Immersive View*

The assignments in this chapter are not subject to a specified time frame. They begin with the following question:

How intuitive is the entering? Try to "enter" a class (Hint: there are 2 methods)

Which can be answered with *yes* or *no*. The next question was only shown if the participant failed to enter a class. It is a free text question:

If you found a way, describe the method. If you did not find a way, skip it. The next question targeted the exiting of the view:

Did you find a way to exit the immersive view?

With the answer options: *yes* or *no* The following question is gaming towards the user experience while interacting with the view. As a hint, it stated two ways to enter the view.

Try both methods to enter the immersive view. Which method felt more natural to you?

Possible answer options: *scroll*, *double click*. The following questions are split into 3 parts:

Is the data provided in a structured way? Provide a suitable topic for each section to the best of your knowledge.

The answer should describe *top*, *middle*, and *bottom* row. The next question is a free text question again and focuses on future works:

Do you know any other object where an immersive view can be helpful to get more information?

The question:

7. Evaluation

How useful is the immersive view in your opinion?

could be answered with one of the following options: *Haven't noticed/No answer*, 0 *No advantage/No*, 1 *Very low*, 2, 3, 4, 5 *Very high*. The final free text question targeted the general feedback:

Do you have any suggestions to improve the immersive view? Entering/Display/Exiting?

7.3.7 Evaluation Results

The evaluation results are divided into three sections. The first section concerns the general knowledge of the participants, while the second section concerns *Semantic Zoom* related questions. The third section concerns the *Immersive View*. The answers to the free text questions have been shortened and/or summarized. The complete results can be found in the following archive: [Bamberg 2024a].

General

A total of 17 individuals participated in the evaluation. One participant was unable to complete the evaluation due to a timeout of the survey tool. The issue was subsequently addressed by modifying the structure of the survey, thereby facilitating greater interactivity with the tool. Of the remaining 16 participants, nine selected Group A, and seven selected Group B. The distribution of users who had previously utilized *ExplorViz* was nearly balanced. 9 of the 16 participants had previously used *ExplorViz*, while the remaining seven had no prior experience with it.

The majority of participants have a background in computer science, which is reflected in the higher level of advanced knowledge compared to other values, as illustrated in Figure 7.3. The participants self-identified their category based on their own understanding, without any pre-defined criteria.

Semantic Zoom

Each participant was required to complete four tasks, two of which were conducted with the *Semantic Zoom* feature enabled and two without. The *Semantic Zoom* feature was either enabled or disabled based on the selected group. The following Figure 7.4 presents the mean time required to complete the task without checking for correct answers. It should be noted that no participants skipped the answering process. The number of incorrect answers was only relevant for Tasks 2 and 3 and did not affect the overall outcome; therefore, it is not displayed in any chart.

7.3. User Evaluation

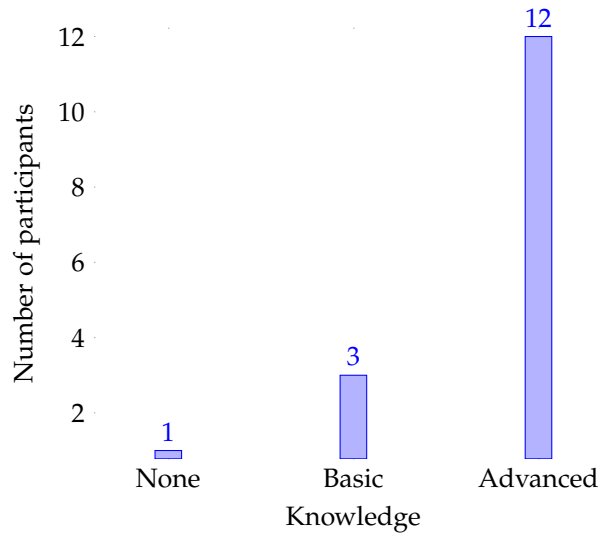


Figure 7.3. Self-rated programming knowledge of the participants.

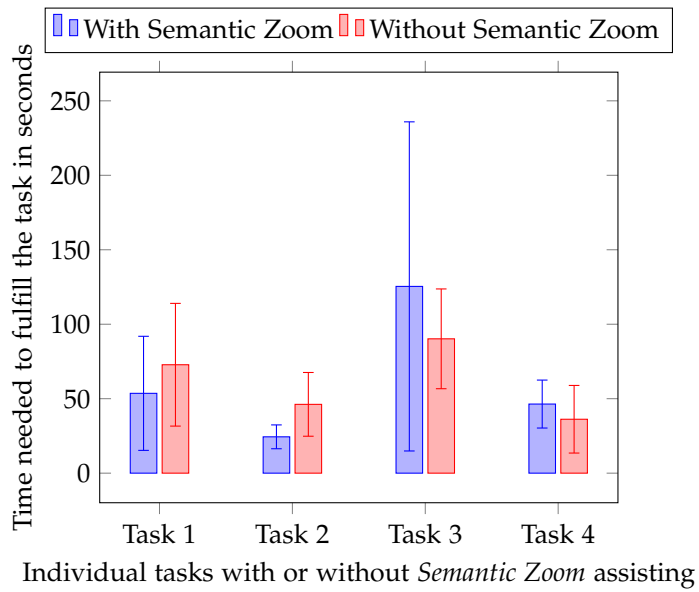


Figure 7.4. Avg time needed for each task.

7. Evaluation

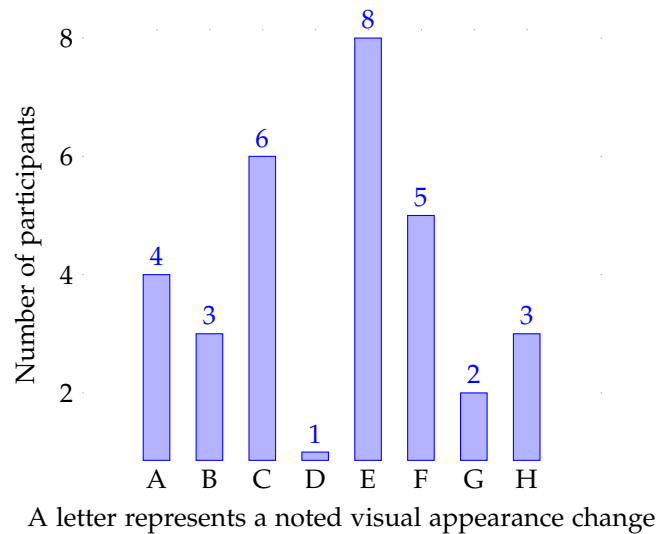


Figure 7.5. First noticed differences by the participants before the tasks begin.

The objective of the inquiry was to ascertain the initial impression of the participants when presented with the two versions, one with and one without the inclusion of the *Semantic Zoom*.

What changes did you notice between the version with semantic zoom and without, while zooming in and out? Describe them as best as you can. You can still use the two browser tabs to compare them.

The results are interpreted and cumulated in order to provide a brief overview of the findings. Each finding gets a unique letter to trace the value in the presented Figure 7.5.

Participants 7, 5, 12, (13) noticed the change of the thickness of a communication line (A). Participants 7, 2, and 9 observed the absence of the direction indicator on the communication lines (B). Participants 7, 3, 4, 8, 10, and 16 mentioned the automatic open and closing of packages (C). The *Immersive View* was already noticed by participant 3 (D). MethodMesh indicators at the classes were observed by participants 7, 6, 8, 9, 11, 12, 13, and 15 (E). The use of ellipses to abbreviate class names has been identified by participants 7, 8, 11, 12, and 15 (F). The alteration in height of the class is documented by participants 7 and 15 (G). The *Immersive View* was referenced by participants 13, 14, and 15. (H).

Another question focused on suggestions to improve the *Semantic Zoom* feature.

Do you have any suggestions for improving the user experience while using the semantic zoom feature?

One participant observed that the packages were opened and closed in a manner

that was not foreseen. One participant expressed disapproval of the hiding of direction indicators on communication lines. The abrupt transition between two discrete levels proved overwhelming for the eye, necessitating a gradual and seamless integration. A technical issue has been identified with the custom selection of the individual *LoD* setting, whereby the sliders are capable of accepting values that fall outside the specified range. One participant expressed concern about the numerous changes between the zoom levels. Another participant requested the full name of the class instead of a partially visible abbreviation. In the closed application state, the font size should be adapted to the distance of the camera to enhance readability.

As illustrated in Figure 7.8, the majority of participants expressed a preference for the version that included the *Semantic Zoom* feature over the version that did not. However, the display performance did not yield a definitive outcome, with a balanced distribution of responses across the three options, with four participants indicating a preference for each option and eight participants indicating that they could not distinguish between the versions.

The results for the ratings of the third list in Section 7.3.5 can be found in Figure 7.6 and Figure 7.7.

Immersive View

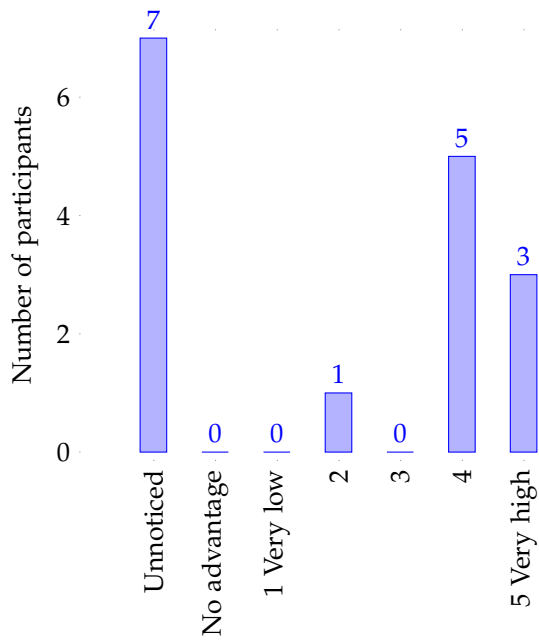
All participants indicated that they were able to gain access to the *Immersive View*. However, one participant did not provide a description of the method used to gain access. Thus, 15 out of 16 participants were able to access the view independently, without the assistance of the provided hint. Of the participants, 13 were able to successfully navigate to the *Immersive View* via the zoom method. However, it should be noted that not all of them accurately described the precise steps required to achieve this. In some instances, participants were initially misled by clicking on the object before following the correct procedure. In terms of the double-click method, seven participants were able to successfully access the *Immersive View* through this approach. However, it should be noted that this method may not be entirely accurate, as one participant stated, "I clicked on the class and zoomed in." This suggests that the double-click may not be the optimal solution for all users. Every participant was able to exit the *Immersive View*. The preferred enter and exit method can be found in Figure 7.9 and the overall rating for the *Immersive View* is shown in Figure 7.10.

The participants were asked for further usability of the *Immersive View* regarding other objects.

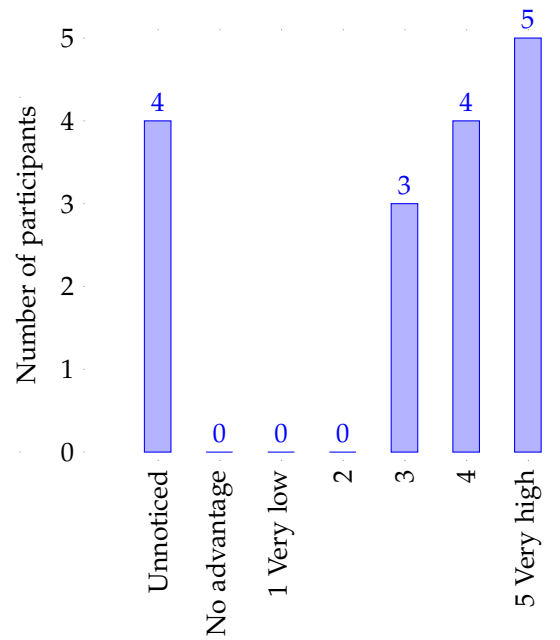
Do you know any other object where an immersive view can be helpful to get more information?
--

Some responses concentrate on the implementation of an immersive perspective for a different object, whereas other responses prioritize the expansion of the existing *Immersive Class View*. Two responses indicated an interest in developing an *Immersive View* at the package/application level. Another respondent expressed interest in developing an

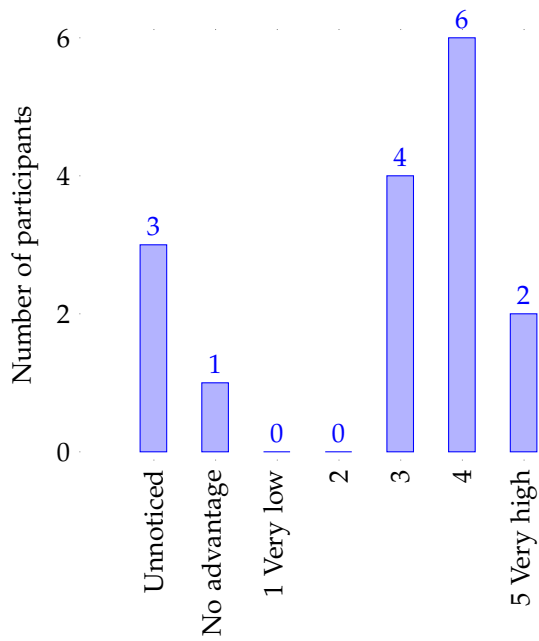
7. Evaluation



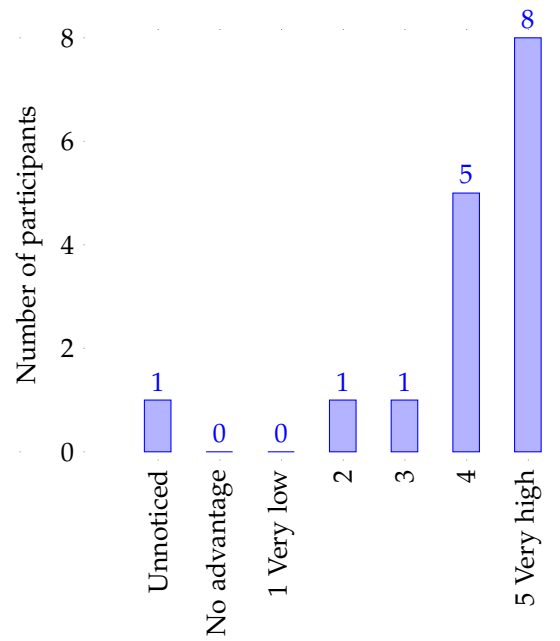
(a) How useful is the change of height of classes to indicate the request count?



(b) How useful is the change of the font size on the class labels?

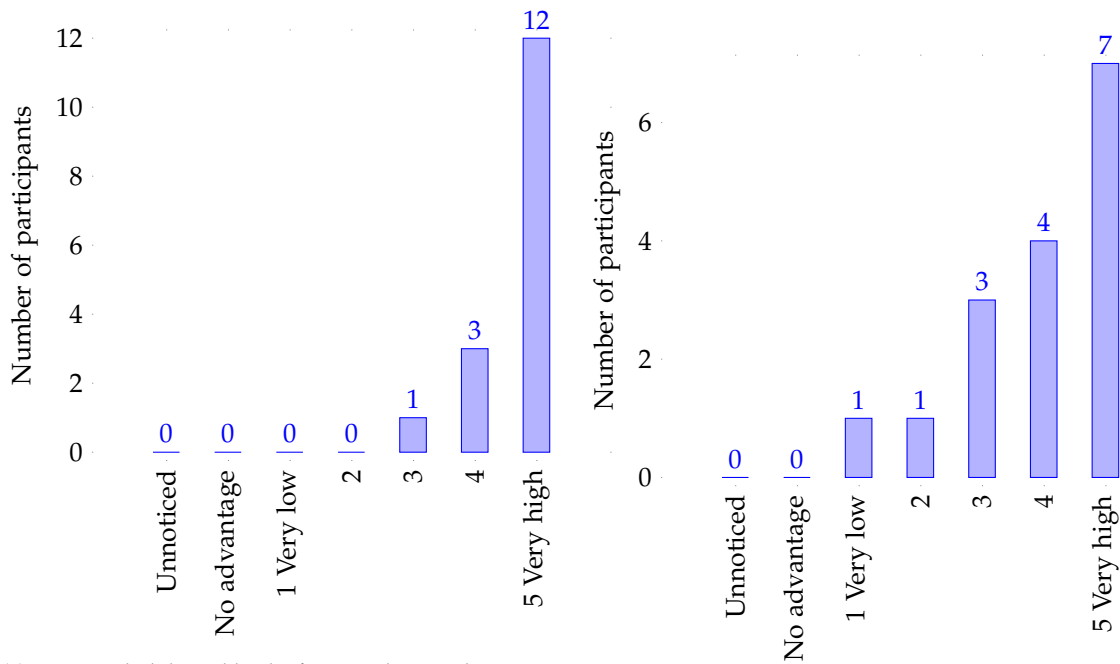


(c) How useful is the indicator of the number of methods at the class object?



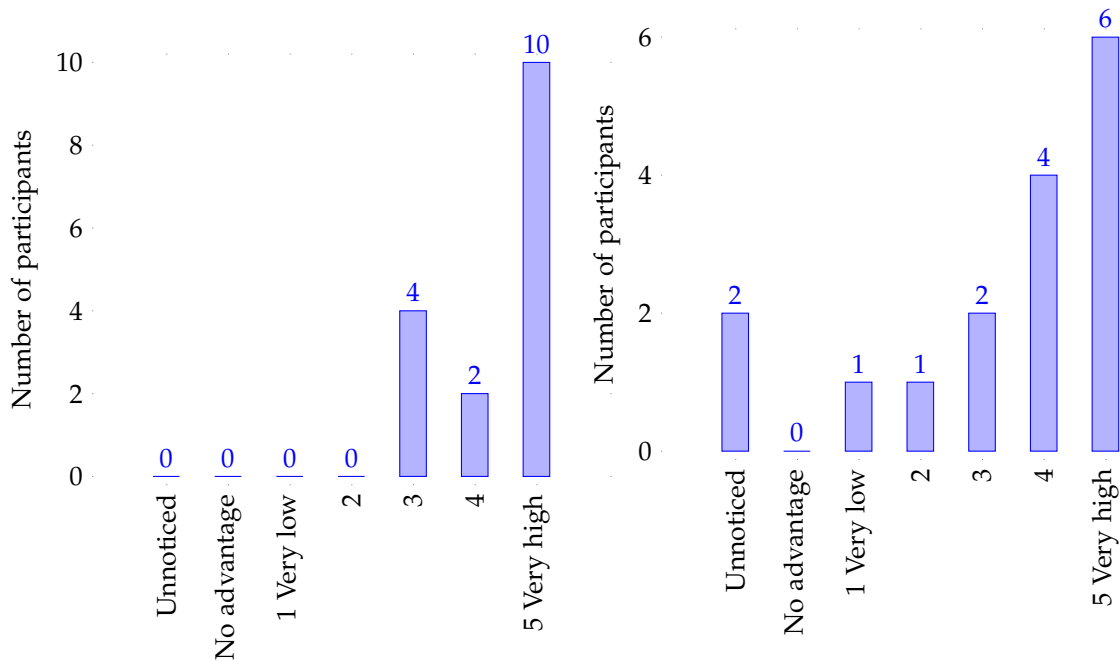
(d) How much did you like the shrinking of the communication lines while zooming in?

Figure 7.6. First set of ratings.



(a) How much did you like the feature where packets open and close automatically depending on the zoom level?

(b) Did the objects you zoomed into change? Did you expect them to be triggered?



(c) Do you think the semantic zoom feature is useful in daily life?

(d) Do you like the overall reduced view compared to the regular view (all components open) without semantic zoom?

Figure 7.7. Second set of ratings.

7. Evaluation

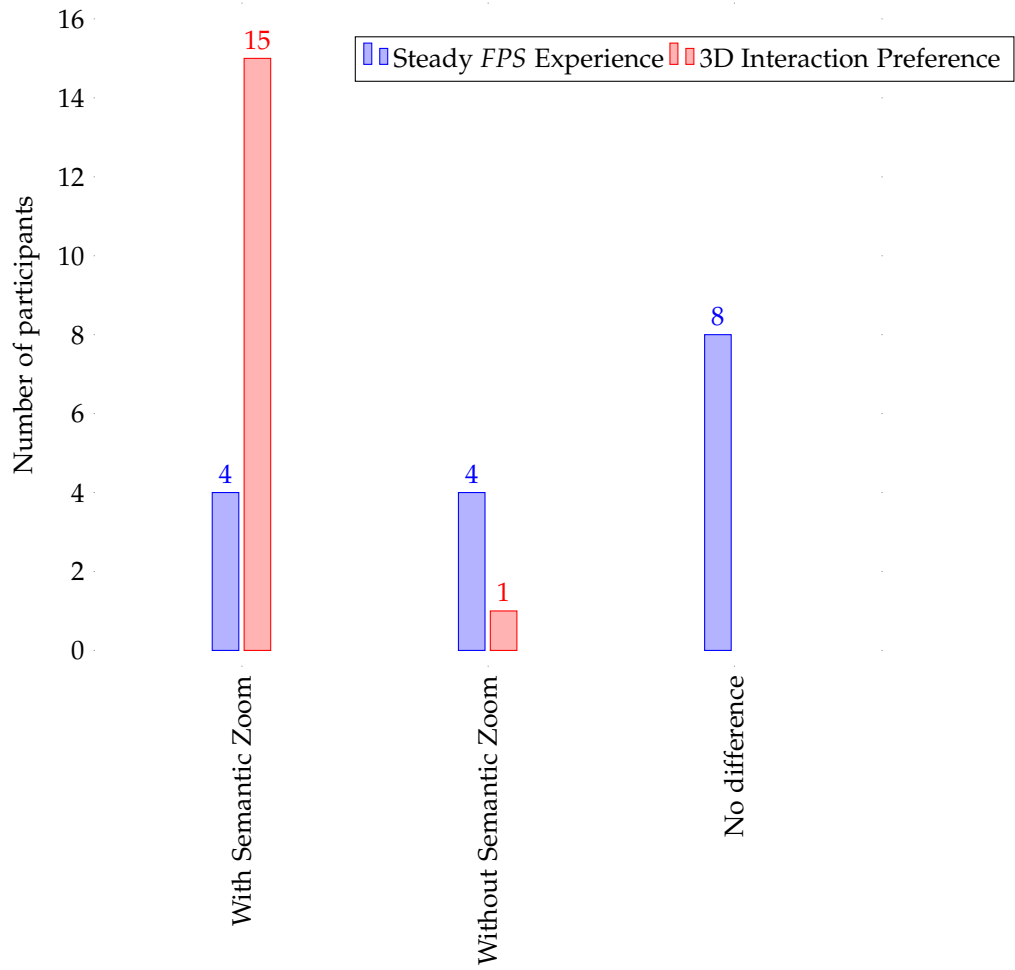


Figure 7.8. FPS and interaction preference between *ExplorViz* with *Semantic Zoom* and without.

7.3. User Evaluation

Immersive View for communication lines. Some respondents proposed enhancements to the *Immersive Class View*. Based on the communication, the classes should display links to other classes that can be triggered. This would enable the user to transition from one *Immersive Class View* to another without having to return to the original view. Another improvement, based on a polymorphism indicator in the *Immersive Class View*, was suggested. This could indicate a method that has been overloaded during the compilation process or overridden during runtime.

Those participating in the study were invited to propose enhancements to the process of entering, displaying, and exiting the *Immersive View*.

Do you have any suggestions to improve the immersive view? Entering/Display/Exiting?
--

The responses encompass all three options. With regard to enhancements for the entry of the immersive view, the following points have been identified:

- ▷ The transition between the original view and the immersive view should be smooth and seamless.
- ▷ The distance between the user and the object should be taken into account, with the entering process only initiated when the appropriate proximity has been reached.

The view or functionality could be improved in the following cases:

- ▷ Enhancements to the table representation of the data.
- ▷ The ability to switch between windows (disable mouse catch).
- ▷ An improved design.
- ▷ The option to automatically open a code editor when zooming into a function.
- ▷ The display of source code within the immersive view.
- ▷ The presentation of all values within the field of view of the camera.

Improvements for exiting:

- ▷ Reduce the timeout of the virtual barrier to enable the user to exit the view in a more expedient manner.
- ▷ Remove the stopper in its entirety.

7. Evaluation

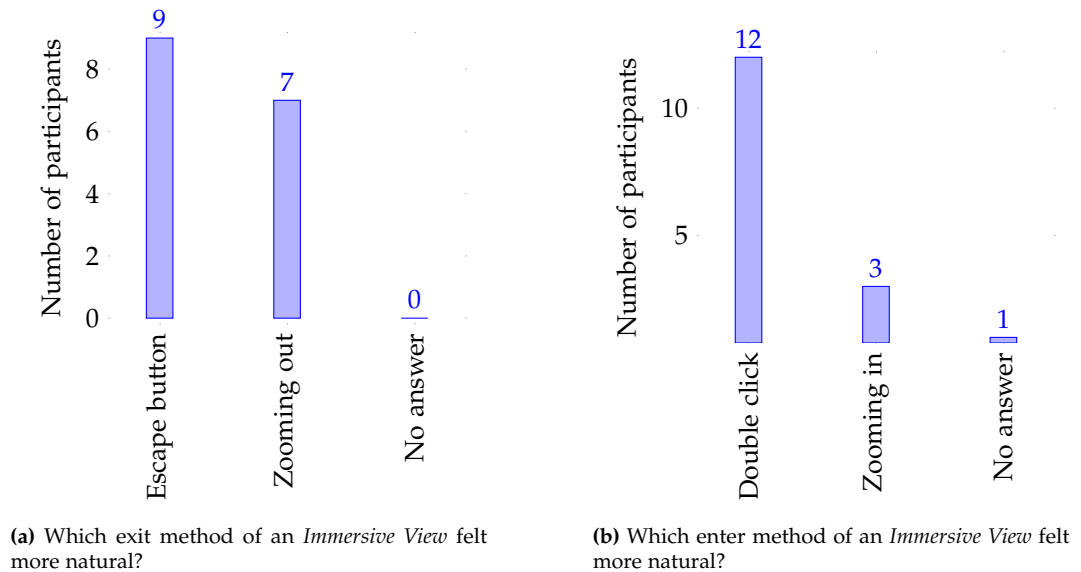


Figure 7.9. Preferred way to exit or enter the *Immersive View*.

7.3.8 Discussion

In the user evaluation of the *Semantic Zoom* tasks, the mean time required by participants with and without *Semantic Zoom* differed in favor of the *Semantic Zoom* feature for tasks 1 and 2. The mean completion time for task 1 with the use of *Semantic Zoom* was 53 seconds, whereas the mean completion time for the group without was 72 seconds. This represents a difference of 19 seconds, or an increase of 35%. A similar trend was observed in the case of Task 2. The group utilizing the *Semantic Zoom* feature completed the task in an average of 24 seconds, while the other group required 46 seconds, resulting in a difference of 22 seconds or 91%.

The results for tasks 3 and 4 are reversed. The group utilizing *Semantic Zoom* required 125 seconds to complete task 3, whereas the other group only required 90 seconds. This yields a difference of 35 seconds, or 38%. Task 4 yielded comparable outcomes, with the group utilizing *Semantic Zoom* requiring 46 seconds, in contrast to the group without this feature, which required only 36 seconds. The discrepancy is 10 seconds, or 27%. After two tasks, the parties swapped the groups, as evidenced by the results. There is a correlation between the time needed and the participants. One party consisted of participants who completed the tasks more rapidly than the other party. There is no evidence that the *Semantic Zoom* feature enhances user performance when using *ExplorViz*.

The results of the question "Which version do you prefer in terms of interacting with the 3D world?" demonstrated a notable trend indicating that *Semantic Zoom* enhances

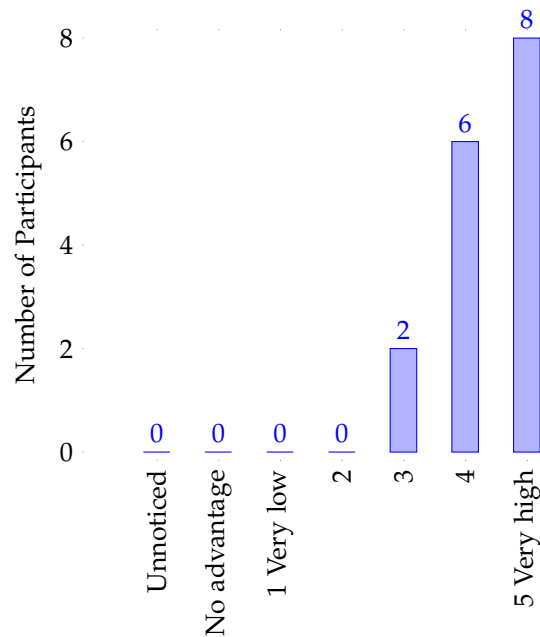


Figure 7.10. How useful is the immersive view in your opinion?

user usability when utilized with *ExplorViz*. 15 participants indicated a preference for the version incorporating *Semantic Zoom*, while only 1 participant expressed a preference for the version without this functionality.

8 of the 16 participants were able to identify the introduction of method indicators at the class objects at first glance. The change in the class height indicating the request count was only noted by 1 participant, which suggests that either the height scale was insufficiently large or that participants are unable to discriminate between size differences effectively. In such instances, a continuous *LoD* may prove beneficial, as it would permit a moderate scaling of the height.

These related results can be observed in the following question: "How useful is the change of height of classes to indicate the request count?" 7 of the 16 participants did not notice the change, while 8 ranked it between 4 and 5 (very high).

The question regarding the change of the font size was either not noted (4) or was noted and identified as useful (12). In combination with the written feedback, further development in optimizing font size and abbreviations is advised. This helps the user to obtain the full information at once, which is beneficial in terms of efficiency.

Although the method indicator on a class was initially identified by a significant number of participants, only 2 rated it with 5, 6 rated it with a 4, and 4 rated it with a 3. 4 did not notice it or identified no advantage for the visualization. It is possible that this feature may

7. Evaluation

not be useful enough to progress to the next stage of development.

The reduction of the thickness of the communication lines was met with considerable approval, as evidenced by the 13 participants who rated it with a 4 or 5. Only one individual did not take note of the alteration. The reduction of the line thickness proved beneficial in situations where navigation was required in a dense area, as it facilitated better orientation.

15 participants rated the automatic opening and closing of packets with either a 4 or a 5. This feature was the most highly rated by participants. Nevertheless, the feedback indicated that there is still room for improvement. In some instances, participants anticipated the opening of other packages, rather than the ones that did open. The clustering algorithm, which serves as an abstraction layer, constrained the options to create a system with 100% accuracy. The system's reliability was evaluated by examining whether the targeted packages altered their appearance as anticipated. The majority of participants rated the system with a 4 or 5 (11), followed by 3 who rated it with a 3, and 2 who rated it with either a 1 (very low) or 2.

The proposed enhancements for *ExplorViz* vary considerably, ranging from unexpected behavior and bugs to feature requests and visual improvements. It is notable that each suggestion is distinctly different from the others, which is somewhat unexpected.

The majority of participants indicated that the *Immersive View* was a useful feature. 14 out of the 16 participants rated the feature in question a 4 or a 5. This indicates that the development of the *Immersive View* can be continued and enhanced. The majority of participants designated the double click as the preferred method for entering the *Immersive View*. However, while a clear preference was observed with regard to entering the *Immersive View*, no majority was reached with respect to the preferred exit method. 9 out of 16 participants voted for the escape button, while 7 out of 16 voted for the zooming out method.

The participants identified the communication lines as a potential candidate for a further *Immersive View* object. However, they did not provide any information regarding the type of content that could be displayed within the view. The blocking barrier that prevents users from unintentionally exiting the view while zooming out requires a shorter timeout period, as it was perceived negatively by multiple participants. The context switch confuses users due to the absence of a transition. Therefore, a feature that facilitates a smooth transition is necessary. One potential solution is to initiate the transition by moving the camera into the objects.

7.3.9 Threats to Validity

It is important to mention the threats to validity as there are multiple factors that can cause validity problems.

Probands The number of 16 participants is considered very low. It might show a trend regarding the user performance using *ExplorViz* with or without *Semantic Zoom*. To show if

7.3. User Evaluation

the tasks can be performed faster, we need multiple hundreds of participants. 9 out of 16 participants had used ExplorViz before and therefore had an advantage compared to the other participants. The distribution among the groups with *ExplorViz* knowledge is 4 (A) against 5 (B), which is very well distributed.

Evaluation Setup All participants had access to a two screen setup, yet not all of the participants took the advantage and therefore lost precious time while performing time critical tasks. The participants need a pretask where the time is recorded but dropped, to get insight of how a task looks like. Some participants might improve in how to control the systems after performing the first tasks. The tutorial beforehand was used as such a buffer, but there was no pressure regarding the time.

Relevance of the Assignments All assignments focused on the fulfillment of a small task, where the user had to extract up to 6 objects. All participants required less than 5 minutes per task.

Conclusion

This work aims to enhance the usability and performance of the *ExplorViz* tool by reducing the amount of information presented depending on the position in the visualization, in consideration of the restrictions of the human brain and visual processing. To address these issues, the *Semantic Zoom* and *Immersive View* features were developed. The *Semantic Zoom* enables the concealment and revelation of information based on the distance between the camera and the object in question. In contrast, the *Immersive View* facilitates a comprehensive and undistracted representation of the object, presenting information in a clear and structured manner.

The implementation process of *Semantic Zoom* involved extending the available 3D object with additional metrics and developing a user-friendly programmer interface for modifying 3D objects. Two clustering algorithms were employed to reduce computational complexity and shift the majority of the computational workload to a preprocessing step. Both the k-means and mean shift clustering algorithms are centroid-based. A manager instance was utilized to switch between the discrete *Level of Details (LoDs)* based on the camera distance to the cluster centroids. Each cluster contains 3D objects that are capable of adjusting their visual presentation. The *Semantic Zoom* feature represents a non-invasive extension to *ExplorViz*, which can be enabled or disabled. Further detailed settings, such as those concerning sensitivity and clustering, may be configured according to the user's preferences.

The *Immersive View* was implemented by creating a new scene that is decoupled from the regular *ExplorViz* view in order to eliminate distracting factors. Each 3D object that is visible in the original scene can be extended by an *Immersive View*. Software developers have access to an easy-to-use *API* that allows them to create their own *Immersive View* for further objects. All 3D objects in *ExplorViz* are capable of being extended if they extend the "BaseMesh". The *Immersive Class View* is employed as a proof of concept for the *Immersive View*. It displays inner class details such as the name, inheritance, variables, and functions with their respective parameters and return types. The data is presented in a structured manner around a sphere, with the user located at the center point of the sphere. The camera control was changed to a first-person view.

An evaluation was conducted with the objective of assessing the performance of the system and its usability, with the latter being evaluated utilizing a user evaluation. The results for the *Semantic Zoom* feature demonstrate no statistically significant impact on user performance when completing tasks. However, the survey indicated positive results

8. Conclusion

regarding usability improvements. Of the 16 respondents, 15 expressed a preference for interacting with *ExplorViz* using *Semantic Zoom*, while only 1 indicated a preference for the version without *Semantic Zoom*. Participants were able to distinguish between versions with and without *Semantic Zoom* and rated most newly-introduced features positively. The feedback included requests for bug fixes, stylistic improvements, and new features, such as modifying the font sizes on all 3D objects. The same ratings were observed for the *Immersive View*. Participants reported no difficulty interacting with the *Immersive View*, which includes entering, moving around, and exiting.

8.1 Future Work

The received feedback outlines potential opportunities for enhancement of the new *ExplorViz* features. Additionally, it includes ideas that emerged during the development process but were not incorporated due to time constraints.

Extend pool of *Semantic Zoom* objects In the context of the work, only a few potential 3D objects were identified that are suitable for the *Semantic Zoom* feature. To enhance the functionality further, it would be beneficial to integrate additional 3D objects and discrete *LoDs*.

Non-Linear Magnification The paper [Keahey 1998] introduced the concept of non-linear magnification, which enables the user to focus on a specific area within the 3D environment without magnifying the entire scene to the same degree. This approach allows for a more detailed spatial view of a particular region while maintaining the contextual information surrounding it. The newly acquired spatial area can then be utilized with *Semantic Zoom* to provide additional 3D object data and insights into the system.

Continuous *Level of Detail* Another proposal is to introduce a continuous *LoD* in place of a discrete *LoD*, as this is perceived as a more naturalistic approach compared to the sudden appearance of objects, as suggested by a participant in the evaluation process. The continuous *LoD* can facilitate the blending in and out of objects, rather than an abrupt manifestation. A different implementation approach is required for this feature, as the current approach is based on a discrete *LoD* and therefore cannot be altered without making significant changes to the implemented structure.

Real Data in the *Immersive Class View* The *Immersive Class View* employs synthetic data to illustrate the functionalities of the view. The necessary data are not yet supplied by the *backend*. To obtain this data, the static analysis must extract this type of information and transmit it to the *frontend*. The *frontend* has already been expanded by a minor amount of class-specific data, as evidenced in Section 6.3.1.

8.1. Future Work

Immersive View With Context The current approach of presenting an *Immersive View* in isolation from the original scene results in a sudden and disruptive shift in context. An alternative strategy is to integrate the *Immersive View* into the original scene as a miniature version, effectively blurring the context while assisting in orientation and minimizing distractions from surrounding information.

View Code in the Immersive View Once the static analysis has provided information about the code of a class, it can also be used to display the code. For example, the code of the method can be displayed when the user is in the *Immersive View* and looks directly at the method.

Appendix A

Cheat Sheet

ExplorViz

List of different landscapes

Alias	Created
Expanding Sample (Expanding structure and increasing, unrelated timestamps)	19.5.2020, 10:28:08
VISOFT 23 - Study Sample	19.5.2020, 10:28:08
Petclinic Sample (Random traces and increasing, unrelated timestamps (with random gaps))	19.5.2020, 10:28:08

Name of the current landscape

Distributed Petclinic Sample
Active mode: Runtime

Main menu button

Settings available via the gear at the top right corner

Semantic Zoom

- Semantic Zoom
- Auto Open/Close of Components
- Use k-Means instead of Shift-Mean
- Use Predefined Set
- Predefined Zoom Sets

Level 1: 1 to 100 (slider at 5)

Level 2: 1 to 100 (slider at 30)

Level 3: 1 to 100 (slider at 40)

Level 4: 1 to 100 (slider at 50)

Level 5: 1 to 100 (slider at 60)

Relative # of clusters: 1 to 100 (slider at 40)

On/Off toggles for all settings.

Classes, height indicate the request count

Method indicator

Show more details:
1: Very early
6: Very late

Set a custom trigger point for each level
Low: Very early
High: very late

Change cluster number k when using k-means
Low: a few trigger points
High: very concrete trigger point

Context menu available via right click in the landscape

- Reset View
- Open All Components
- Hide Communication
- Enable Heatmap
- Pause Visualization
- Open Sidebar
- Enter AR
- Semantic Zoom disable (Beta)
- Show SemanticZoom Center Points

petclinic-visits-service

Communication Line with direction indicator

Opened Application with is Components and Classes

petclinic-visits-service

dto

PetDetails

OwnerDetails

PetType

visits

springframework

web

filter

hikari

zaxxer

pool

org

com

Figure A.1. Cheat sheet used during the user evaluation.

Glossary

Apache Kafka Apache Kafka is a distributed event streaming platform, that can store, process and export datastreams. 10

backend The backend of software typically operates on a server in the background and provides data. 10, 19, 43, 66

Docker Docker is a platform that facilitates the packaging and distribution of applications as containerized software. 44

Ember.js Single-page web app written in JavaScript. 8, 10, 19

ExplorViz ExplorViz software visualization tool. 1, 2, 5, 10, 13, 15, 19, 22, 23, 25, 26, 36, 40, 41, 45, 46, 48, 49, 52, 58, 60–63, 65, 66

frontend Software's frontend that is communicating with the user. 1, 5, 10, 19, 23, 34, 36, 47, 66

GitHub Online Service for Software development using Git. 44

gRPC gRPC is a highly efficient Remote Procedure Call Framework initially developed by Google. 10

Immersive Class View An Immersive View for an object-oriented class. viii, 37–40, 55, 59, 65, 66

Immersive View Immersive View provides a focused perspective of a specific part of the overall system without contextual distractions. viii, 2, 3, 6, 13, 15, 37, 39, 41, 43, 48, 51, 52, 54, 55, 59, 60, 62, 65–67

Jaeger Jaeger is a distributed tracing platform. 10

JavaScript A scripting language used to develop interactive websites for web browsers. 8, 29, 34, 43, 44, 47

Node.js Open-Source JavaScript runtime outside the Browser. 44

OpenTelemetry OpenTelemetry collects logs, metrics and traces. 9–11

Glossary

Prometheus Prometheus is a monitoring system with a time series database. 10

Semantic Zoom Semantic Zoom displays different features on different zoom levels. vii, viii, 2, 3, 5, 7, 13, 14, 16, 17, 23, 32–34, 36, 39, 43, 45–49, 52–55, 58, 60–62, 65, 66

Three.js A JavaScript library for rendering 3D environments in the browser using WebGL. vii, 7–10, 19, 22, 27, 31, 37

Web Browser Software designed to display web pages accessed through HTTP. 7, 8, 39, 41, 43, 44

WebAssembly Binary code that can be executed by the web browser. 6

WebXR WebXR provides an API for developers of web applications to access augmented reality or virtual reality. 10

Acronyms

API Application Programming Interface. 8, 19, 65

CPU Central Processing Unit. 6, 43

FOV Field of View. 6, 28, 45

FPS Frames per Second. 5, 6, 43–46, 58

HTML Hypertext Markup Language. 8

LoC Lines of Code. 21, 39

LoD Level of Detail. vii, 7, 14, 16, 22, 23, 29, 55, 61, 65, 66

OTLP OpenTelemetry protocol. 9, 10

PoI Point of Interest. 17, 26, 27, 31

RAM Random Access Memory. 43

UI User Interface. 8, 10, 19, 36

VWM Visual Working Memory. 2

Bibliography

- [Alam and Dugerdil 2007] S. Alam and P. Dugerdil. Evospaces visualization tool: exploring software architecture in 3d. In: *14th Working Conference on Reverse Engineering (WCRE 2007)*. 2007, pages 269–270. DOI: 10.1109/WCRE.2007.26. (Cited on page 15)
- [Alnabhan et al. 2018] M. Alnabhan, A. Hammouri, M. Hammad, M. Atoum, and O. Al-Thnebat. 2d visualization for object-oriented software systems. In: *2018 International Conference on Intelligent Systems and Computer Vision (ISCV)*. 2018, pages 1–6. DOI: 10.1109/ISACV.2018.8354085. (Cited on page 1)
- [Bach et al. 2011] B. Bach, E. Pietriga, I. Liccardi, and G. Legostaev. Ontotrix: a hybrid visualization for populated ontologies. In: *Proceedings of the 20th International Conference Companion on World Wide Web. WWW '11*. Hyderabad, India: Association for Computing Machinery, 2011, pages 177–180. DOI: 10.1145/1963192.1963283. (Cited on page 15)
- [Balogh et al. 2016] G. Balogh, T. Gergely, Á. Beszédes, and T. Gyimóthy. Using the city metaphor for visualizing test-related metrics. In: *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*. Volume 2. 2016, pages 17–20. DOI: 10.1109/SANER.2016.48. (Cited on page 1)
- [Bamberg 2024a] J. Bamberg. *Evaluation Results - Semantic Zoom With Immersive Detail View for ExplorViz*. Zenodo, Nov. 2024. DOI: 10.5281/zenodo.14228962. (Cited on pages 45, 48, 52)
- [Bamberg 2024b] J. Bamberg. *Explorviz-frontend with semantic zoom feature*. Nov. 2024. DOI: 10.5281/zenodo.14229523. (Cited on pages 23, 39)
- [Buering et al. 2006] T. Buering, J. Gerken, and H. Reiterer. User interaction with scatterplots on small screens - a comparative evaluation of geometric-semantic zoom and fisheye distortion. *IEEE Transactions on Visualization and Computer Graphics* 12.5 (2006), pages 829–836. DOI: 10.1109/TVCG.2006.187. (Cited on pages 7, 16)
- [Bugla 2024] M. M. Bugla. Ein ansatz zur generierung von opentelemetry-traces auf grundlage von synthetischen anwendungsstrukturen. Bachelor's Thesis. Kiel University, Sept. 2024. URL: <https://oceanrep.geomar.de/id/eprint/60834/>. (Cited on page 44)
- [De Carlo et al. 2022] G. De Carlo, P. Langer, and D. Bork. Advanced visualization and interaction in glsp-based web modeling: realizing semantic zoom and off-screen elements. In: *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems. MODELS '22*. Montreal, Quebec, Canada: Association for Computing Machinery, 2022, pages 221–231. DOI: 10.1145/3550355.3552412. (Cited on page 29)
- [Dirksen et al. 2014] J. Dirksen et al. *Three.js essentials*. Packt Publishing, 2014. (Cited on pages 7, 8)

Bibliography

- [Fittkau et al. 2017] F. Fittkau, A. Krause, and W. Hasselbring. Software landscape and application visualization for system comprehension with explorviz. *Information and Software Technology* 87 (2017). DOI: <https://doi.org/10.1016/j.infsof.2016.07.004>. (Cited on page 10)
- [Hamzaturrazak et al. 2023] M. Hamzaturrazak, E. M. A. Jonemaro, and A. Pinandito. Performance analysis of 3d rendering method on web-based augmented reality application using webgl and opengl shading language. In: *Proceedings of the 8th International Conference on Sustainable Information Engineering and Technology*. SIET '23. Badung, Bali, Indonesia: Association for Computing Machinery, 2023, pages 637–643. DOI: 10.1145/3626641.3626949. URL: <https://doi.org/10.1145/3626641.3626949>. (Cited on page 43)
- [Hansen and Hasselbring 2024] M. Hansen and W. Hasselbring. *Instrumentation of software systems with opentelemetry for software visualization*. 2024. URL: <https://arxiv.org/abs/2411.12380>. (Cited on page 10)
- [Hansen et al. 2013] M. Hansen, R. L. Goldstone, and A. Lumsdaine. *What makes code hard to understand?* 2013. URL: <https://arxiv.org/abs/1304.5257>. (Cited on page 48)
- [Hasselbring et al. 2020] W. Hasselbring, A. Krause, and C. Zirkelbach. Explorviz: research on software visualization, comprehension and collaboration. *Software Impacts* 6 (2020). DOI: <https://doi.org/10.1016/j.simpa.2020.100034>. (Cited on page 10)
- [Keahey 1998] T. Keahey. The generalized detail in-context problem. In: *Proceedings IEEE Symposium on Information Visualization (Cat. No.98TB100258)*. 1998, pages 44–51. DOI: 10.1109/INFVIS.1998.729558. (Cited on pages 15, 16, 66)
- [Krause-Glau et al. 2022] A. Krause-Glau, M. Hansen, and W. Hasselbring. Collaborative program comprehension via software visualization in extended reality. *Information and Software Technology* 151 (2022), page 107007. DOI: <https://doi.org/10.1016/j.infsof.2022.107007>. (Cited on page 10)
- [Liverence and Franconeri 2015] B. M. Liverence and S. L. Franconeri. Resource limitations in visual cognition. In: *Emerging Trends in the Social and Behavioral Sciences*. John Wiley Sons, Ltd, 2015, pages 1–13. DOI: <https://doi.org/10.1002/9781118900772.etrds0287>. (Cited on page 2)
- [Luca et al. 2019] F. D. Luca, I. Hossain, K. Gray, S. Kobourov, and K. Börner. *Multi-level tree based approach for interactive graph visualization with semantic zoom*. 2019. URL: <https://arxiv.org/abs/1906.05996>. (Cited on page 14)
- [Luebke 2003] D. Luebke. *Level of detail for 3d graphics*. Morgan Kaufmann, 2003. (Cited on page 7)
- [Misiak et al. 2018] M. Misiak, A. Schreiber, A. Fuhrmann, S. Zur, D. Seider, and L. Nafeie. Islandviz: a tool for visualizing modular software systems in virtual reality. In: *2018 IEEE Working Conference on Software Visualization (VISSOFT)*. 2018, pages 112–116. DOI: 10.1109/VISSOFT.2018.00020. (Cited on page 13)
- [Parisi 2012] T. Parisi. *Webgl: up and running*. " O'Reilly Media, Inc.", 2012. (Cited on page 7)

- [Ren et al. 2014] Y. Ren, U. Kamath, C. Domeniconi, and G. Zhang. Boosted mean shift clustering. In: *Machine Learning and Knowledge Discovery in Databases*. Edited by T. Calders, F. Esposito, E. Hüllermeier, and R. Meo. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pages 646–661. (Cited on page 26)
- [Roth 2017] R. E. Roth. Visual variables. *International encyclopedia of geography: People, the earth, environment and technology* (2017), pages 1–11. (Cited on pages 19, 21)
- [Al-Saiyd 2017] N. A. Al-Saiyd. Source code comprehension analysis in software maintenance. In: *2017 2nd International Conference on Computer and Communication Systems (ICCCS)*. 2017, pages 1–5. DOI: 10.1109/CCOMS.2017.8075175. (Cited on page 1)
- [Scalabrino et al. 2018] S. Scalabrino, M. Linares-Vásquez, R. Oliveto, and D. Poshyvanyk. A comprehensive model for code readability. *Journal of Software: Evolution and Process* 30.6 (2018). e1958 smr.1958, e1958. DOI: <https://doi.org/10.1002/smr.1958>. (Cited on page 1)
- [Shariff et al. 2019] S. M. Shariff, H. Li, C.-P. Bezemer, A. E. Hassan, T. H. Nguyen, and P. Flora. Improving the testing efficiency of selenium-based load tests. In: *2019 IEEE/ACM 14th International Workshop on Automation of Software Test (AST)*. 2019, pages 14–20. DOI: 10.1109/AST.2019.00008. (Cited on page 43)
- [Teyseyre and Campo 2009] A. R. Teyseyre and M. R. Campo. An overview of 3d software visualization. *IEEE Transactions on Visualization and Computer Graphics* 15.1 (2009), pages 87–105. DOI: 10.1109/TVCG.2008.86. (Cited on pages 1, 41)
- [Victor 2000] J. D. Victor. How the brain uses time to represent and process visual information¹published on the world wide web on 16 august 2000. *Brain Research* 886.1 (2000). Towards 2010, A brain Odyssey, The 3rd Brain Research Interactive, pages 33–46. DOI: [https://doi.org/10.1016/S0006-8993\(00\)02751-7](https://doi.org/10.1016/S0006-8993(00)02751-7). (Cited on page 2)
- [Voinea and Telea 2006] L. Voinea and A. Telea. Multiscale and multivariate visualizations of software evolution. In: *Proceedings of the 2006 ACM Symposium on Software Visualization. SoftVis '06*. Brighton, United Kingdom: Association for Computing Machinery, 2006, pages 115–124. DOI: 10.1145/1148493.1148510. (Cited on page 2)
- [Wiens et al. 2017] V. Wiens, S. Lohmann, and S. Auer. Semantic zooming for ontology graph visualizations. In: *Proceedings of the 9th Knowledge Capture Conference. K-CAP '17*. Austin, TX, USA: Association for Computing Machinery, 2017. DOI: 10.1145/3148011.3148015. (Cited on pages 1, 7, 13, 14, 48)