Douglas Cunningham, Petra Hofstedt, Klaus Meer, Ingo Schmitt (Hrsg.): INFORMATIK 2015
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2015

Game Event Lenses: Focus+Context Visualizations of Computer Game Data

Lars Schütz[1]

Abstract: This paper introduces Game Event Lenses. They enable explorative and interactive focus+context visualizations of computer game events that may form huge and complex datasets including various attributes from the close game environment. The Data State Reference Model (DSRM) is the foundation of the presented approach. It is based on different data states and operators. This work features a concept that is designed to completely simulate the DSRM on a graphics card by using its memory and related shader programs. Their new general purpose computing and traditional rendering capabilities allow user interactions and frequently generated images in real time.

Keywords: Information visualization, graphic systems and interfaces, computer game data.

1 Introduction

Computer games are versatile, technically complex and widely spread. They can be differently perceived. Developers might create new virtual worlds and adventures with dedication and passion. They and additional publishers try to make a profit with their games while players typically consume them to have fun or to simply pass time. There are more concrete motivational points when it comes to games.

Developers and publishers try to analyze the behavior of players in solo and group activities [EDC13]. By doing this, they possibly gain insight into specific trends and preferences, e. g., favorite in-game weapons. Furthermore, they might detect if the players really play their game how they meant it to be played. Such obtained information can result in further adjustments to the game and its mechanics that even might get into the game after its initial release. Individual short term offers for in-game items that can be bought with real money or layout changes of specific in-game world areas can be seen as only two related of many more imaginable examples.

Other motivations from a developer's point of view refer to the general game design. Weak points regarding the overall game balance can become obvious by analyzing each player's progress and performance in a game. For example, several players have to fight enemies and no one is able to beat them because their attack skill is numerically too weak. Those kind of shortcomings can possibly be seen when collected game data is examined. Such an analysis can lead to game balance adjustments, e. g., lowering the overall difficulty level.

[1] Anhalt University of Applied Sciences, Department of Computer Science and Languages, 06366 Köthen, Germany / Otto von Guericke University, Faculty of Computer Science, 39106 Magdeburg, Germany, l.schuetz@inf.hs-anhalt.de

Players view games from another perspective. They may take part in electronic sports tournaments where the best strategies and tactics typically lead to a tournament win. In this respect, they possibly have a high interest in the analysis of their performances. But also ordinary players compare themselves by browsing player profiles and statistical overviews about achievements and owned items in various games. A game can almost always be seen as a competitive environment.

Generally speaking, the analysis of computer game data involves different directions and perspectives. The afore-mentioned approaches only give a basic overview. A separate discipline called Game Analytics evolved during the last years that deals with such questions [EDC13]. It combines several other subjects like Data Mining, Machine Learning, Statistics and Visualization to gain insight into game related data.

1.1 Challenges

The extraction and communication of usable information from huge datasets is a central issue, because it is difficult to deduce structural information and interrelationships from raw data representations. It can be a very time-consuming and complex task. The visualization of the raw data may help in this respect. Additionally, supporting tools are needed that process data in a timely manner.

A consequent problem is the possible information overload when many data elements are visualized at the same time. That complicates the recognition and interpretation of facts. Specific visualization techniques may help to address that challenge.

All those issues affect visualization in general and they are applicable to the visual analysis of computer game data in particular. This paper examines possibilities to overcome those problems and therefore provides a useful contribution to the specific area of Game Analytics as well as to the discipline called Information Visualization [Sp01].

1.2 Outline

The visualization of raw data makes the illustration of information possible and more accessible. It is seen as one chance to address the previously mentioned challenges. Section 2 clarifies what visualization actually means and how initial data can be processed by introducing the visualization transformation and the Data State Reference Model (DSRM) [CR98].

Section 3 refers to (computer) game events in the context of the DSRM. Game events are defined, their components are identified and it is outlined how they can be processed.

Visualizations of game events in particular and of data in general still might lead to cluttered information as previously mentioned. This hinders the discovery of new insights into the data. Regarding this, lenses are presented in Section 4. They enable a user to focus on specific parts of a visualization.

To process and to visualize the game events with respect to the integration of lenses a performant graphics system is needed. Section 5 shows the conceptual idea of such a system and delivers introductory examples based on a collected sample dataset of game events.

Finally, Section 6 concludes this paper and proposes future work.

2 Visualization and the Data State Reference Model

A visualization is typically understood as an external graphical representation of concepts or data [Wa04]. In this respect, it stands for a final result of some kind of process. In contrast to that, another point of view refers to a visualization as a human process itself that leads to a mental model of data [Sp01]. This approach emphasizes the human brain with its perceptive and cognitive abilities. We can combine both views and see a visualization as an interface between them [Wa04]. Humans with their comprehensive faculty and decision making abilities are on the one side. Computers that process huge amounts of data, e. g., images from the internet, to further abstractions by using specific algorithms are on the other side. A visualization is then the foundation for the knowledge acquisition. It carries information that is based on the given data that should be easier to understand. This whole idea is illustrated in Fig. 1.

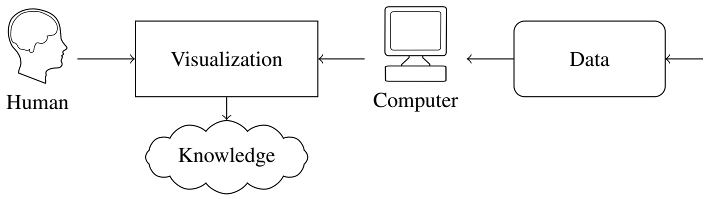


Fig. 1: Visualization acts as an interface between the human and the computer-processed data and enables the discovery of knowledge

It now needs to be clarified how visual representations or image data D_I are generated starting with available raw data D_R before we are actually able to derive knowledge from them. The involved process is the so called visualization transformation that is basically the mapping $f : D_R \times P_1 \times P_2 \times \cdots \times P_n \rightarrow D_I$ with $P_i \mid 1 \leq i \leq n, n \in \mathbb{N}$ being the visualization parameters and $p \in P_i$ being the visualization parameter values [JMG02]. This parameterized form gives us more control and flexibility because each set P_i can be applied to different steps in the visualization transformation depending on how it is actually defined.

The DSRM describes how raw data can be transformed into image data. It is based on different data states and operators. The following four states, also called stages, need to be distinguished:

- *Raw data D_R* are the input data for the visualization.
- *Analytical abstractions D_A* are calculated properties or meta data.

- *Visual abstractions D_V* are abstracted data in portrayable form that carry visual properties, e. g., color hue values.
- *Image data D_I* are the final visual representations, e. g., pixel data.

The DSRM works like a pipeline. It consists of two types of operators that propagate the data through the model:

- *State operators* leave the structure of the data untouched because the output is on the same stage as the input. There are operators O_R on D_R, operators O_A on D_A, operators O_V on D_V and operators O_I on D_I.
- *Transformation operators* transform data from one stage to their succeeding stage. They can be distinguished into the following operators:
  - *Raw data transformations T_R* abstract raw data and create new structures.
  - *Visualization transformations T_V* map analytical abstractions to visual abstractions.
  - *Visual mapping transformations T_A* render visual abstractions to obtain image data.

The first two states are part of the data domain while the remaining states are part of the view domain. The same principle applies to the operators. This kind of classification enables a distinction between data-oriented or semantic operators and view-oriented or graphical operators [TFS08]. The schematic view of the DSRM is depicted in Fig. 2.

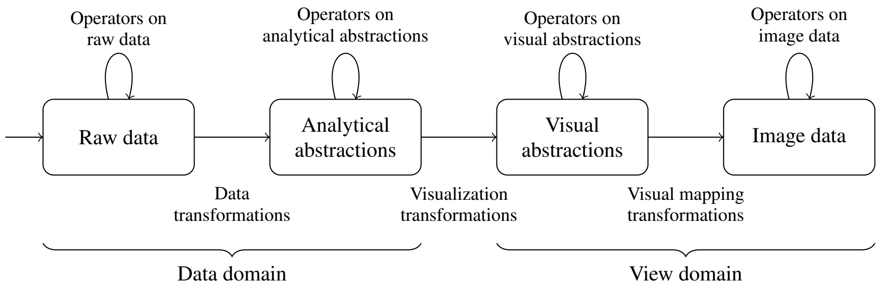


Fig. 2: The DSRM transforms raw data via analytical abstractions and visual abstractions into image data by using operators

3 Game Events in the Context of the Data State Reference Model

The following examinations apply the DSRM to game events. Aspects that relate to the data domain and the view domain are considered. But first of all the most important terms shall be clarified.

A *(computer) game* is an interactive experience that offers increasingly challenging sequences of patterns to a player that can learn and potentially master those [Ko04]. We can conclude that the gaming experience can turn out differently for various persons while the patterns may be the same. That relates to the individual learning aptitude of each player and to the fact that they interact with or within the game in different ways, e. g., in terms of chronological order.

The term *event* is versatile and actually defined differently in various disciplines. For example, events are essential when it comes to interrupt processing and the process life cycle in operating systems [Ma13]. In that context, events trigger temporary and final interrupts of process executions to handle new or already existing processes. Those kind of events let a system react to them. In discrete event-oriented simulations events are instant occurrences that alter a system's state [To11]. Such a state comprises a collection of the system's properties at a specific time. We can find different general concepts for an event that are sometimes used interchangeably. On the one hand, there are event types or event classes that describe different kind of events and determine a unified structure for similar events while, on the other hand, there are event instances or event objects that are a concrete realization or instantiation of an event type [BD10]. In this paper, an event stands for an event instance unless stated otherwise.

A *(computer) game event* can now be defined as a detectable and immediate event that contains parts of the complete game state description and is triggered by game mechanics or internal requests. This definition focuses on the following aspects:

- *Detection:* A game event can be detected by the game itself. It can further be detected by either nobody or at least one player.
- *Duration:* A game event occurs instantly and does not span a specific period of time.
- *State description:* A game event represents properties of game related elements. It is an excerpt from the game's current complete internal state description.
- *Game mechanics or requests:* On the one hand, at least one action of a player triggers an event. Respectively, an action is caused by interacting with the game, e. g., pressing a button to move a virtual avatar. That refers to game mechanics in the sense of gameplay. On the other hand, the game's internal processes trigger events, too. Those are either game mechanics with no players involved, e. g., jumping of a non-player character, or procedures that are relevant to the system itself, e. g., getting the number of currently logged in players in an online game.

3.1 Game Events in the Data Domain

Game events consist of different components that represent the game's partial state description. They form the input of the DSRM. We propose the following classification:

- The *event type* describes the event's class or the type of the corresponding event instance.

- The *timestamp* informs about the point in time the event happened, e. g., by stating date and time information.
- The *spatial data* describe several spatial information, e. g., the position or place of the happening in in-game world coordinates or orientations of game entities.
- The *sender* holds information about the event's trigger.
- The *receiver* may store associated data if the event affects a game related entity. The receiver can also be the sender at the same time.
- The *event type data* are specific information based on the event type.

These components can result in many game events with different concrete attributes. They are flexible and suit different fields of application, e. g., a gameplay or business context. Tab. 1 shows two examples for a game event to further substantiate the appropriateness of the mentioned components.

Component	Gameplay game event	Business game event
Event type	Attack	Transaction
Timestamp	2015-03-07T23:01:14Z	2015-01-18T11:11:11Z
Spatial data	Area: 23, x: 234, y: 102, z: 20	Auction house: 15
Sender	Player: 3, level: 80, class: 4	Account: 66
Receiver	Player: 5, health points: 34; Player: 14, health points: 56	Account: 66
Event type data	Damage to player 5: 11; Damage to player 14: 5	Item: 54, quantity: 1, price: $7.99

Tab. 1: Two sample game events from different contexts

It is obvious that just one event can hold plenty of data depending on the degree of detail. There is no general rule. Each game may handle events differently. Some components may be optional. The shown arrangement and hierarchical structure of the components are not mandatory, e. g., the spatial data could be part of the sender's data instead. Each component further divides itself into several attributes, e. g., the receiver's health points. The proposed components are a working approach to describe the raw data but each visualization system should be aware of the complexity and possible different structures.

It is now clear how the raw data looks like. Several different operators can be used to further process the game events depending on the goal or task. Data cleansing, data reduction or data transformation are typical activities with a lot of different specific operators, e. g., automatic or manual selection of events (O_R or O_A), dealing with missing attribute values (O_R or T_R), detection and handling of outliers (O_R), normalization and standardization (O_R or O_A), attribute selection by principal component analysis (T_R), aggregation or discretization (O_R, T_R or O_A) [HKP12]. The operators may change the structure of the data so that analytical abstractions result from their application, or the data is simply passed through to the visualization transformation operators.

3.2 Game Events in the View Domain

The visualization transformation operators choose at least one of the visual variables from Bertin [Be11] to encode the analytical abstractions. Those variables are called *position*, *size*, *shape*, *brightness*, *color hue*, *orientation* and *texture*. It is possible to pick a visual variable for an attribute of a game event that should be transformed by observing its level of measurement. There are concepts that classify the degree of appropriateness for a visual variable, e. g., see MacEachren [Ma04]. Each game event or its further abstraction is represented by a geometric primitive with different levels of detail, e. g., a game event might have a position and a specific shape to encode its spatial data and type components.

The visual mapping transformation includes typical rendering operators, e. g., transformation and shading of geometric primitives, perspective or parallel projection and rasterization. The visual abstractions of the game events in form of geometric primitives are transformed into image data. Game events and possible further abstractions of them are now visualized. Operators for exemplary graphical manipulations on the image data are the translation, the rotation or the scaling of the viewable area.

4 Visualization Lenses for Game Events

The interpretation of visualizations can still be difficult because of some kind of information overload. Visualized game events may overlap each other. Too many game events may be illustrated at the same time. Therefore it is sometimes useful to concentrate on specific areas of a graphical representation. That area can include more detailed information while the surrounding remains simple and does not distract. That leads to focus+context visualizations by means of visualization lenses.

A visualization lens is an interactive and parameterizable visualization tool that spatially selects and modifies the underlying original representation of a visualization and helps to enrich, suppress or alter the corresponding content [To14]. The spatial selection represents the focus the user is interested in. The remaining part specifies the context that helps to orientate oneself. That enables a user to focus on different regions one after the other. In order to do this, the interaction with the lens is necessary that can be achieved by controlling its properties *shape*, *position*, *size* and *orientation* [To14].

A lens needs to be integrated into the visualization transformation. The focus region can be visualized by an additional DSRM pipeline that is attached to the main DSRM pipeline [To14]. A selection of the data in the focus region is needed. It is possible to select data from any stage of the DSRM, but a proper inverse projection of the selected data elements to their corresponding representations from previous states might be needed depending on the first operator in the lens pipeline. A management of unique identifiers for data elements or the concept of half spaces [TFS08] are resolutions for this problem. The lens determines the content of its focused region by realizing a composition $f = o_n \circ o_{n-1} \circ \cdots \circ o_1$ of $n \in \mathbb{N}_{\geq 1}$ operators o_i so that f can be interpreted as at least one state or transformation operator of the DSRM. The final output or the lens effect is joined with the main pipeline. Fig. 3 shows the described idea.

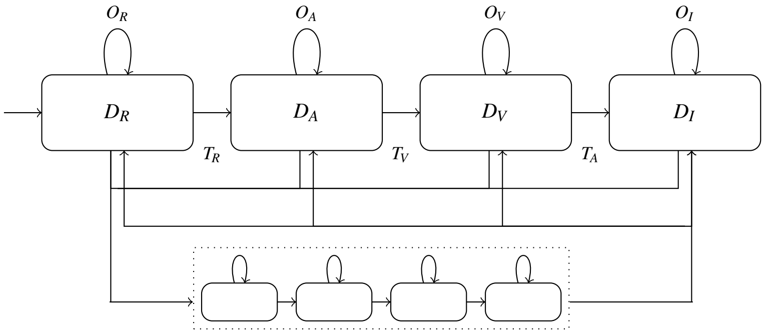


Fig. 3: A separate lens pipeline is attached to the main pipeline

Finally, the overall visualization of game events including the lens and its rendering is generated in three steps:

1. Create and render the base or context visualization by omitting the lens region.
2. Create and render the lens or focus region.
3. Optionally render the lens itself to show its properties, e. g., by drawing its boundary.

5 Graphics System and Prototypical Implementation

A software application is needed to explore game events. It is mainly responsible for the graphical representation and the processing of user interactions that alter the visualization parameter values. It should be able to process thousands of events in real time.

The proposed graphics system is based on the Open Graphics Library (OpenGL) [Sh13]. It is a cross-platform application programming interface and enables the control of the rendering process through the use of shader programs that operate on the graphics processing unit (GPU) and therefore profits from the GPU's parallel computing capabilities [Ow05]. OpenGL combines different shader types in a pipeline whereby each one has its distinctive functionality [Sh13]. These are in order the *vertex shader* (VS), the *tessellation control shader* (TCS), the *tessellation evaluation shader* (TES), the *geometry shader* (GS) and the *fragment shader* (FS).

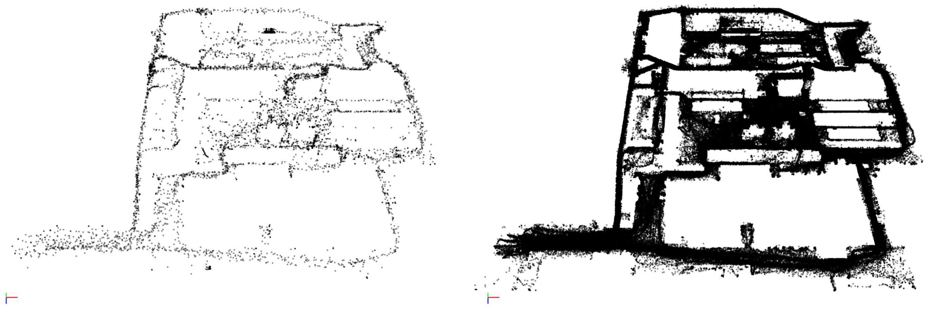
In the presented concept the shader programs are used to simulate the DSRM's operators. Traditionally, vertices of geometric primitives are transformed into pixel data using at least a vertex shader and a fragment shader. This approach is similar to what the DSRM's graphical operators do. Consequently, those are covered and only the semantic operators need to be handled. Since August 2012 respectively OpenGL Version 4.3 the OpenGL pipeline is additionally equipped with *compute shaders* (CS) [Sh13]. They allow general purpose computations on the data that can be visualized afterwards. No extra interoperability efforts in terms of context switches are needed. A compute shader is used to simulate a semantic operator. Fig. 4 shows the proposed applicability of the shaders.

$$\begin{tikzcd}
{} \arrow[r] & D_R \arrow[loop above, "\text{CS}"] \arrow[r, "\text{CS}"'] & D_A \arrow[loop above, "\text{CS}"] \arrow[r, "\text{VS, TCS, TES, GS or CS}"'] & D_V \arrow[loop above, "\text{VS, GS or CS}"] \arrow[r, "\text{VS, TCS, TES, GS or FS}"'] & D_I \arrow[loop above, "\text{CS}"]
\end{tikzcd}$$

Fig. 4: Conceptual applicability of the shader types to simulate the DSRM's operators

In order to use GPU-based operators the data they work on need to be stored on the graphics card. All game events and further abstractions are coherently packed using *shader storage buffer objects* because shader programs can read from and write to them [Sh13]. That means that initially m game events with a maximum number of n attributes are stored in the sequence $(v_a^e) = (v_1^1, v_2^1, \dots, v_n^1, v_1^2, v_2^2, \dots, v_n^2, \dots, v_1^m, v_2^m, \dots, v_n^m)$ where v_a^e is the value of the attribute a that belongs to the specific event e. The visualization parameter values can be stored in *uniforms* or in *uniform buffer objects* [Sh13].

As a proof of concept two lens examples were developed and integrated into a software prototype that enables a user to interactively explore an example dataset, e. g., by moving the mouse to change the lens' position. The visualization design of the following examples is not addressed consciously and almost arbitrary, because only the feasibility of the shader-based approach is considered. That should definitely be examined and discussed separateley, e. g., in terms of choosing color hues or picking different shapes. The dataset originates from the computer game *Counter-Strike: Global Offensive*. The game events were collected on a private dedicated server applying a custom-built server plugin. 782000 events with 33 attributes for each event of overall 827 game rounds between two teams of computer controlled bots were logged. That corresponds to the play time of about 20.5 h. As a basic example Fig. 5 shows the spatial positions of each game event's sender component. This is the base visualization with no omitted events because there is no active lens.



(a) 13 game rounds (b) 827 game rounds

Fig. 5: Processed game events reveal the spatial position of their sender component

The first lens example allows to explore the category of the used weapon when a shot was fired by a player in relation to the event's position. The goal is to reveal preferred weapon types. The lens starts operating on D_R and uses a CS $o_1 : D_R \to D_A$ (selects and passes the related events through), a VS $o_2 : D_A \to D_V$ (sets the position, and sets the color hue based on the weapon's category), a GS $o_3 : D_V \to D_V$ (changes the shape from point to triangle or square to encode the bot's team attribute) and a FS $o_4 : D_V \to D_I$ (passes the color through). Fig. 6 shows this example for one team by using a cuboid-shaped lens.

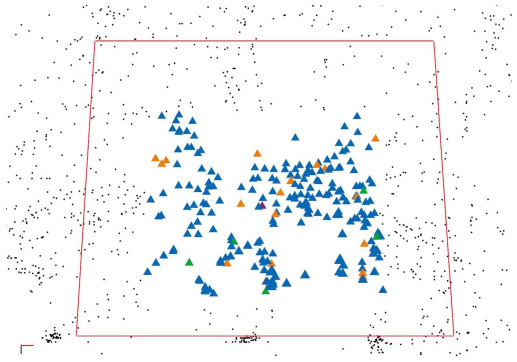


Fig. 6: Used weapon categories (encoded by different color hues) of one team when a shot was fired

The second lens example shows the positions of virtual player deaths and kills and the connections between them. The goal is to show dangerous and advantageous positions, and the related view directions at the same time. The lens starts operating on D_R and uses a CS $o_1 : D_R \to D_A$ (selects and passes the related events through), a VS $o_2 : D_A \to D_V$ (sets the position, and sets the color hue based on the team attribute), a GS $o_3 : D_V \to D_V$ (changes the shape from point to triangle (kill) or cross (death)) and a FS $o_4 : D_V \to D_I$ (passes the color through). The lens also uses a VS $o_5 : D_A \to D_V$, a GS $o_6 : D_V \to D_V$ (changes the shape from point to line) and a FS $o_7 : D_V \to D_I$ in a second pass to draw the connections. Fig. 7 shows this example with a sphere-shaped lens.

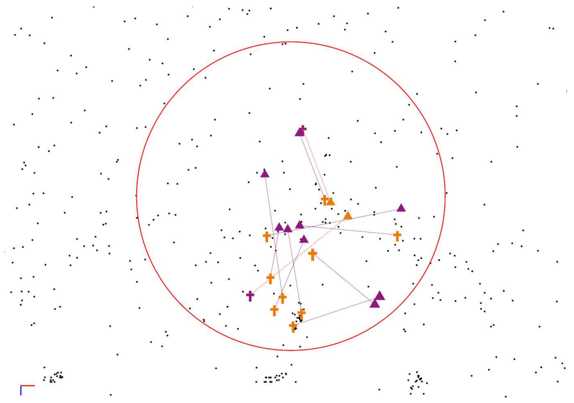


Fig. 7: Death and kill positions of both teams (encoded by shapes and color hues) and their relations

From a user's point of view performance is an important criteria in interactive applications. Loading times or processing times in general need to be minimized if a user interacts with the system and when the visualization almost constantly changes, e. g., when adjusting the lens' position. An interactive experience is still possible with a maximum latency of about 66.667 ms that correspond to about 15 frames per second whereby a higher frame rate respectively a lower latency is preferable [TC07]. The runtime performance heavily depends on the used operators and their time complexity. A generalization of the performance is not possible. We additionally need to distinguish between iterative and parallel problems, because a GPU-accelerated operator can be more suitable than a CPU-based operator for some problems and vice versa. That makes the considerations difficult. But a mandatory process is the detection of game events in the focus region. These need to be updated when a single lens property changes. A simple brute force search algorithm implemented on the GPU was tested in this regard to give at least a performance hint. The runtime between two frames was measured on the GPU for a duration of 10 s using *OpenGL timer queries* [Sh13] while the position of the lens constantly changed. The average time taken to test 750000 initial game events on a moderate desktop computer[2] is 12.143 ms using a cuboid-shaped lens and 12.207 ms using a sphere-shaped lens. This leaves about 50 ms for other operators.

6 Conclusion and Future Work

Computer games provide numerous data in form of game events that are interesting for developers, publishers and players. The visualization of those events is able to reveal insights into facts from the game environment. In this context, the DSRM is an established and appropriate model for the visualization transformation of game events.

The proposed Game Event Lenses provide a controllable opportunity to explore game events. It is possible to focus on different areas of the visualization while a contextual overview is still intact. But there are remaining issues that could be examined in the future, e. g., visual occlusions of information in three dimensional virtual scenes with corresponding lenses as seen in the examples. Information that are displayed inside the focus area but do not belong to it might hinder the recognition of facts. This should be addressed.

The Game Event Lenses base on the DSRM and therefore focus on operators that process data on different stages. The shader based concept is appropriate to simulate the DSRM because an operator can be easily simulated by a shader program. This GPU-based approach is suitable for general purpose computations that deal with parallel problems. That is not tied to game events only. But issues remain. One problem concerns the linkage of several operators. Currently only the programmer of the graphics system is in total control of a sequence of operators. Well defined interfaces and interaction mechanisms are needed if an ordinary user should be able to design and use its own sequences.

Nevertheless, the lens examples of the prototypical implementation demonstrate the aptitude of the presented idea for at least basic or simple visualizations to start with the visual exploration of game events.

[2] Intel i7-960 @ 3.20 GHz x8, 24 GB RAM, NVIDIA GeForce GTX 480 with 1536 MB RAM, NVIDIA driver version 343.22 on Arch Linux 64 bit with kernel version 3.17.1

References

[BD10] Bruns, R.; Dunkel, J.: Event-Driven Architecture: Softwarearchitektur für ereignisgesteuerte Geschäftsprozesse. Springer-Verlag, Berlin, Heidelberg, Germany, 2010.

[Be11] Bertin, J.: Semiology of Graphics: Diagrams, Networks, Maps. Esri Press, Redlands, CA, USA, 2011. Translated by Berg, W. J. from: Jacques Bertin. Sémiologie graphique: Les diagrammes, Les réseaux, Les cartes. Gauthier-Villars, Paris, France, 1967.

[CR98] Chi, E. H.-H.; Riedl, J.: An Operator Interaction Framework for Visualization Systems. In: Symposium on Information Visualization, InfoVis 1998, Research Triangle Park, NC, USA, Proceedings. IEEE Computer Society, pp. 63–70, 1998.

[EDC13] El-Nasr, M. S.; Drachen, A.; Canossa, A., eds. Game Analytics: Maximizing the Value of Player Data. Springer-Verlag, London, UK, 2013.

[HKP12] Han, J.; Kamber, M.; Pei, J.: Data mining: Concepts and Techniques. Morgan Kaufmann Publishers, Waltham, MA, USA, 3rd edition, 2012.

[JMG02] Jankun-Kelly, T. J.; Ma, K.-L.; Gertz, M.: A Model for the Visualization Exploration Process. In (Moorhead, R.; Gross, M.; Kenneth, I. J., eds): IEEE Visualization 2002, Boston, Massachusetts, USA, Proceedings. pp. 323–330, 2002.

[Ko04] Koster, R.: A Theory of Fun for Game Design. Paraglyph Press, Phoenix, AZ, USA, 2004.

[Ma04] MacEachren, A. M.: How Maps Work: Representation, Visualization, and Design. The Guilford Press, New York, NY, USA, 2004.

[Ma13] Mandl, P.: Grundkurs Betriebssysteme: Architekturen, Betriebsmittelverwaltung, Synchronisation, Prozesskommunikation. Springer Vieweg, Wiesbaden, Germany, 3rd edition, 2013.

[Ow05] Owens, J. D.; Luebke, D.; Govindaraju, N.; Harris, M.; Krger, J.; Lefohn, A.; Purcell, T. J.: A Survey of General-Purpose Computation on Graphics Hardware. In: Eurographics 2005: State of the Art Reports. pp. 21–51, 2005.

[Sh13] Shreiner, D.; Sellers, G.; Kessenich, J.; Licea-Kane, B.: OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 4.3. Addison-Wesley, 8th edition, 2013.

[Sp01] Spence, Robert: Information Visualization. Addison-Wesley, Harlow, England, 2001.

[TC07] Thropp, J. E.; Chen, J. Y. C.: Review of Low Frame Rate Effects on Human Performance. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, 37(6):1063–1076, 2007.

[TFS08] Thiede, C.; Fuchs, G.; Schumann, H.: Smart Lenses. In (Butz, A.; Fischer, B.; Krüger, A.; Oliver, P.; Christie, M., eds): Smart Graphics: 9th International Symposium on Smart Graphics, SG 2008, Rennes, France, Proceedings. Springer-Verlag, Berlin Heidelberg, Germany, pp. 178–189, 2008.

[To11] Tominski, C.: Event-Based Concepts for User-Driven Visualization. Information Visualization, 10(1):65–81, 2011.

[To14] Tominski, C.; Gladisch, S.; Kister, U.; Dachselt, R.; Schumann, H.: A Survey on Interactive Lenses in Visualization. In (Borgo, R.; R., Maciejewski; Viola, I., eds): EuroVis - STARs. The Eurographics Association, pp. 43–62, 2014.

[Wa04] Ware, C.: Information Visualization: Perception for Design. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition, 2004.