OrderWarp

1st Henrique Lourenço Ferreira
Instituto Superior Técnico
Lisbon, Portugal
ferreira.henrique27@gmail.com

***Abstract*—The creation of platforms that support Big Data and Streaming domains is nowadays an important topic. However, the excessive amount of data forces the visualization's system to find the best data aggregation techniques to explicitly display information, and that at the same time does not impact the overall performance of the system. These techniques have to be applied in runtime, since that is when data is received in Streaming systems. To deal with this issue, there is a concept called Graceful degradation, a technique that depicts information with different levels of aggregation and detail for different time periods. We present OrderWarp, a system that displays ordinal big data applying the abovementioned technique using WebGL technologies for performance enhancement. The system's visualization uses different task-oriented idioms accompanied by animated transitions to represent changes in aggregations between periods. Using a binning strategy, the aggregation method ensures the visualization is able to always represent the whole dataset, and run indefinitely. The study confirms the performance boost in Big Data Streaming visualizations, resulting of the system's architecture. It also finds the most effective idioms to represent ordinal data in these domains, as well as the best transitions between said idioms.**

***Index Terms*—Big Data, Streaming, Ordinal Data, Graceful Degradation, Information visualizations, Idioms and Transitions**

I. Introduction

Information analysis is extremely important for data recognition and decision-making in multiple areas of scientific study, playing an important role in the development of said areas. The advancement of information technologies, such as smartphones or IoT devices, originated a growth in the amount of accessible and valuable data, created and stored by increasingly more entities. When the dimensions of these datasets defy, computationally, its representation, it is safe to say that the dataset belongs to the **Big Data** domain.

Understanding information present in a dataset with a few thousand records without using a visualization, can be an excessively time-consuming task, and if this dataset enters the Big Data domain, the analysis is unpractical or even impossible. This means that Big Data visualizations are obliged to find adequate aggregation techniques that represent the whole dataset. If the aggregation is too small, the visualization is forced to draw more representative elements of the data, and the drawing time will affect the system's computational performance, or the overdrawing of elements will make the visualization illegible, in both cases its analysis is compromised. Oppositely, if the aggregation is too large the visualization might lose the ability to present useful information. The aggregation level is therefore dependent of intended detail.

The ease of producing data also means that at every moment new data is being generated and plenty of systems experience an increase in traffic. Decision-making, in these situations, has to done in real-time. This domain is called **Streaming**, and it is characterized by the data's moment of creation being during the observation of the visualization, whose main concern is the representation of the data as soon as it is received and without it constantly changing the visualization with new data.

Joining both domains, the result is an analysis with the advantages of the two, which addresses an even bigger computational challenge. Since, in Streaming visualizations, the data can not be processed before the start of the visualization, since it has not yet been created, it is necessary to group it during the visualization's runtime. The moment where the grouping occurs is also important to study. If grouped too soon, the system could, again, lose important information. If too belatedly, the visualization will hold too many visual elements. Moreover, the moment where it was created, should be explicit in the visualization. Newer data is usually the focus of Streaming visualizations, yet older data can present important information too, so both data periods should be represented. This implies that more data needs to represented, hence the system requires, once again, an aggregation technique that should group data according to its age.

With all this in mind, Pires et al. [13] presented the **Graceful degradation** technique, whose objective is to represent Big Data Streaming data in different LoD, with different visualization techniques, in separate timespans, from the beginning of the visualization to real-time. The work only took into account quantitative data, and to further the value of the technique, in this work's case, the studied data type is the **ordinal data**. Ordinal data is defined as data divided in categories, where these present a defined discrete order. One example of this type of data is seen in *Likert* scales, being the most common in social sciences' studies [9]. Understanding the best visual representations for ordinal data in the Graceful degradation technique is the focus of this work.

II. State-of-the-art

Big Data's inherit conditions provokes complications regarding visual noise and large image perception identified in Gorodov et al. [5] for unprepared visualizations. LiveRac [11] and Ferreira et al. [6] present the full dataset with different LoD's. The first work displays quantitative data, while the last works in geospatial data. Alternatively, Daae Lampe and Hauser [3] and Repke [16] apply a modified Kernel density algorithm used in geospatial and graph data types respectively. For multivariable data, Sansen et al. [14] uses a parallel coordinates with Bézier curves to reduce intersection clutter.

In Streaming visualizations, it is important to analyze works that explore Time-series, where the data's timestamp is fundamental for the visualization. The majority of visualizations explain age through one finite axis, that, in some idioms such as stream graph used in [2, 8], pass this idea with clarity. However, this idiom is not suitable for every data type, nor does it show additional details. Mansmann et al. [10] creates a data path for their data, displaying with more detail the newer data. Since it addresses categorical data, the authors were unable to find a technique which aggregated all the data in one idiom, forcing data to leave the system.

Regarding performance, in Big data visualizations, the system should scale with higher amounts of data. A common response is to apply pre-processing techniques to a dataset [18, 14, 15] which groups data and in turn reduces its access time. Zhicheng Liu and Heer [18] accomplishes this using multidimensional projections, Sansen et al. [14] applies extensive cluster algorithms, trading pre-processing time with interactivity, Tim Repke [16] uses machine learning algorithms accompanied by the t-SNE algorithm. Even the visualizations which apply these techniques still need some form of real-time aggregation due to the users infinite possible queries [14]. Sansen et al. [14] presents the on-demand processing module and Sye-Min Chan et al. [15] implements a technique for predictive caching, that foresees the next data necessary for the most likely succeeding interaction. Pre-processing techniques require prior knowledge of the dataset, which is not known in Streaming domains. The focus in these visualizations should be fast and iterative data treatment techniques. Traub et al. [17] implements a variation of the M4 algorithm to extract necessary data in the specific moments, and Fujiwara et al. [7] uses the incremental PCA algorithm to reduce the new data's dimensions. These solutions work with high flows of data, therefore they are effective for both studied domains.

A. *VisBig*

VisBig is a set of works, where the goal is to study and extend the concept of Graceful degradation. The first proposed work was Pires et al. [13], which serves as base for OrderWarp.

Graceful degradation's technique displays data along a path, passing through different levels of detail. The LoD's represent different timespans which are depicted in different connected idioms, suitable for the respective timespan. The final idiom should present an aggregation technique that allows the grouping of all data since the beginning of the visualization. The system was called VisMillion and analyzed the presented technique for quantitative data, leaving other data types unstudied.

Connecting the idioms are horizontal transitions [1], responsible for transforming the information in the first idiom to the second, while explaining the changes in the aggregation levels. These modules play a pivotal part in the conservation of the user's perception of data between idioms. For this reason, Pereira et al. [12] studied the effectiveness of developed transitions between the scatter plot idiom to other idioms.

None of the studied works address ordinal data in big data streaming visualizations. Our goal is to find the best possible different task-oriented idioms for ordinal data as well as their transitions, and create a performance stable system that can display all data in a dataset and run indefinitely, using the Graceful degradation technique.

III. Solution

With ordinal data in mind, we propose OrderWarp, a system that extends the Graceful degradation technique to this data type, by restructuring VisMillion's [13] architecture to WebGL technologies, and creating a set of connected idioms and transitions between them, that can run indefinitely and maintain all received information inside the visualization separated through different LoD's. Therefore, the system commits to implement a visualization displaying ordinal Big data in real-time, grouping data along a timeline through different idioms that can provide information in separate timespans, until it reaches a final idiom which agglomerates the totality of data since the beginning of the visualization. In the next sections, we describe our solution.

A. *Architecture*

To achieve the previous goals the resulting architecture can be seen in figure 1. The visualization is composed by a limitless number of different **modules**, all commanded by the **Modules manager**. The **Modules manager** is responsible for linking the idioms by sending and receiving the data to the respective idioms, updating the current module's timings and other relevant information of the visualization, such as its start time or the number of ordinal values. This module is responsible for the update of the common time axis.

Each module is created given a width and a timespan configurable by the user in order to accommodate the needs of the visualization, this way the user can alter the detail of each module. Using these properties, each module is charge of calculating the position of an element inside itself through the equation $p = vModule + b$, where the velocity ($vModule$) is the division of the width by the timespan. Each module is responsible for an **idiom** that is the visual depiction of data. The module commands the data while within its timespan by storing, sending it to the idiom and returning it back to the Modules manager.

A module can also be a **transition** module, which is the entity responsible for continually transforming the data in the form of the previous module to the next module's requirements. This is accomplished by the creation of bins, the structures responsible for grouping data. Since the transition is the link between two modules, the position of the data also has to decelerate from the first module's velocity to the second module's velocity. Opposite to the other modules, the module's timespan (Δt) is not given as an input by the user, but rather calculated with the remaining available visualization width (w) divided by the number of existing transitions, by the following equations:

$$a = \frac{vModule2^2 - vModule1^2}{2w} \tag{1}$$

$$\Delta t = \frac{vModule2 - vModule1}{a} \tag{2}$$

The **visualizations** are representations of the data using different techniques for different analysis tasks. These representations are purely visual, which means a module can change visualization in runtime, making the system more dynamic.

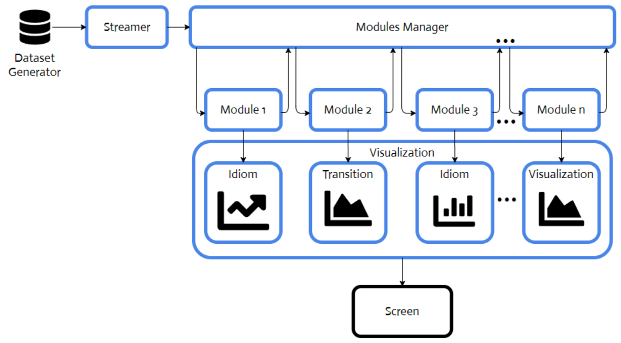

Fig. 1. OrderWarp's architecture

B. Migration

VisMillion [13] and VisMillion and change [12] were conceived to support Big Data. In Pereira et al. [12], the authors came to the conclusion that its visualization could reach 1000 dots per second. After careful consideration, both works decided to implement the visualization using the D3.js JavaScript library for manipulation of documents based on data. D3.js makes it easy to bind data to HTML elements, and defining animation and interaction behaviors for them. This library usually uses SVG technology, a XML based vector image format. Another possible alternative is canvas, which, instead of drawing according to vectors, draws elements pixel by pixel inside the canvas' dimensions. SVG's alternatives are a better and simpler solution for the representation of larger objects, since it draws the element as a whole and has already built-in manipulation and interactivity functions. The manipulation of these elements takes time, which makes this solution ineffective for visualizations with plenty of elements. Canvas draws by pixel, this means the scale of the dimension of elements does not alter the drawing phase of the system, perfect for Big data visualizations. The downside of this technology is that it requires much more scripting, leading to larger implementation times and increased lines of code.

OrderWarp's proposed solution uses the THREE.js library, used to create, manipulate and display 3D or 2D elements, which is also built on top of WebGL. WebGL is a JavaScript API that accelerates the rendering of graphic elements in a web browser by shifting the drawing of the elements to the GPU's. WebGL, alongside the canvas technology, is shown by Kee et al. [4] as the best solution for displaying complex visualizations.

The first phase of implementation, after restructuring the architecture of the visualization, was to implement the visualization techniques studied in the VisBig organization [13, 12, 1] in the best way possible, within a WebGL environment. The process was done by creating a THREE.js scene which is the subject displayed in the canvas through the view of a virtual orthographic camera. A scene holds the objects or meshes, which will be the visual elements displayed in the visualization. To animate a scene, the application updates every object inside the scene and then renders it continuously.

C. Dataset generator

Finding an ordinal big data dataset, capable of testing all tasks our visualization sets out to accomplish, is not easy. Therefore, we created our own dataset generator capable of creating sending and manipulating data through the use of python server script.

The server and client use the *Socket.IO* library, which creates a WebSocket channel between them. After creating the channel, this technology maintains the connection, lowering the latency of every communication between the entities. The channel is also bidirectional, which allows the server to send data to the visualization, and it returns performance measures for later analysis.

Both ordinal and quantitative data can be generated in the server, with the data values assigned by one of multiple functions of randomness, which can be changed prior or during runtime. After creating a data point, the server sleeps until it needs to create a new one. However, the sleep function consumes time itself, so for bigger flows the server becomes unable to create points at a sufficient rate. In these cases, the server sends multiple data points corresponding to the spent time. The data is received by the **Streamer** module, which is the platform of communication between the visualization and the server.

D. Elements

The visual representations of data, displayed in the visualization, are individual elements created by both idioms and transitions which manipulate these representations to convey information as intended by them. All elements are instances of the THREE.js library, and the required elements were the following:

- **Dots:** Simple, small rectangular planes that represent individual data points.
- **Lines:** Rectangular plane, where its height and angle of rotation is given by two x and y coordinates. The line thickness is also modifiable.
- **Rectangles:** Rectangular plane, with a single position. In this case the rectangle's size or color is modifiable. It is also possible to draw borders surrounding the rectangle, accomplished by the creation of lines for each border. These means, the rectangles are comprised of five geometries instead of one.
- **Polygon:** This element is needed to create non-rectangle polygons, and it is possible through a buffer geometry, by providing a list of vertices positions. This element allows more flexibility, since it does not restrict the shape of the visual representation, yet its manipulation is much more complex.

On a preliminary test phase, while using the dots elements, we verified that the performance of the system in terms of fps's, would easily be heavily reduced. To improve performance of this element, we substituted the dot mesh with *instanced meshes*. These elements reduce the number of manipulated objects by joining every dot created between two updates, in a single instance. From that point on, the manipulation of the instanced mesh impacts every dot inside it. This limits the alteration of the dot's properties, reflected in the incapability of changing the dots' opacity.

Whether it be in drawing or scripting, the amount of elements in the visualization dictates the performance of the visualization. All elements are stored in a scene's array where one element's removal takes $O(n)$, and in OrderWarp this event happens very often. In order not to impact the performance of the visualization, the system stores every element sent for removal, and when a new one is required, instead of creating a new element, it reuses freed meshes.

E. Bins

The concept of Graceful degradation is only possible with the help of bins. They are responsible for reducing the amount of data in the visualization by representations of the same, lowering the computational load and allowing more information to be displayed. Bins are agglomerations of all the data received during a period. When no more points will be added to the bin, it will compute and hold various properties of the grouped data, simplifying its access later on. Examples of these properties are its timestamp, and the number of data points inside the bin.

The transition modules generate bins continually, receiving all data during the bins' interval whether it is single data points or other bins. In case of more than one transition, the second transition's bins have to be a multiple of the previous one, to avoid two same size bins receiving different amounts of smaller bins. An example of this can be seen in figure 2 shows in a) two 9 second bins receiving a different number of smaller bins.

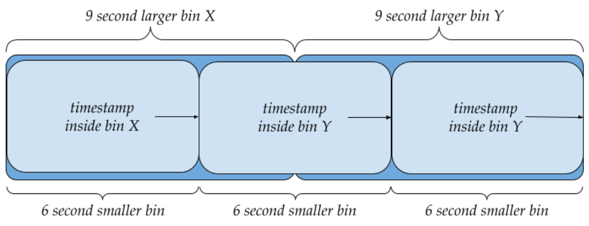

Fig. 2. Depiction of the difference between non-multiple and multiple bin duration times.

F. Idioms

Idioms are the visual representation of data, who's job is to allow the user to accurately understand the information present in the data, and take insightful information about said data. The idiom's suitability relies on the tasks relevant to analyze, such as the evolution of the data or the comparison between ordinal values, while also taking into account their volume or in which timeframe they are being studied. All idioms share an x-axis which encodes the data's timestamp, which is constantly updated in real-time, forcing the visual elements to move towards the end of the visualization. The y-axis also encodes the ordinal value for the whole visualization. The implemented idioms are explained below.

1) Cleveland plot: The Cleveland plot was proposed as a variation of the quantitative scatter plot. In both idioms the visual representation is achieved by drawing an individual dot for every data point received. Oppositely to scatter plot, in Cleveland plot the dots are restrained into invisible lines representing each existing ordinal value. In terms of implementation, these idioms are the same, as the ordinal values are converted to a position as if they were quantitative.

Since the dots represent individual data points, the number of elements being drawn every update is considerably high, impacting the visualization's performance more than other solutions. To reduce the number of elements, this idiom implements the abovementioned instanced meshes, which does not reduce the number of visual meshes raising the concern of dot clutter when the dots overlap due to a higher data flow resulting in an unclear data's distribution.

This idiom is a great solution to understand the arrival of data points and its distribution over time, as each data point is represent in one dot. Since it applies no aggregation techniques, it is only suitable as the beginning idiom. It is also more effective for smaller timespans, computationally and visually, since longer time periods will result in more elements to draw, and bigger visual agglomerations.

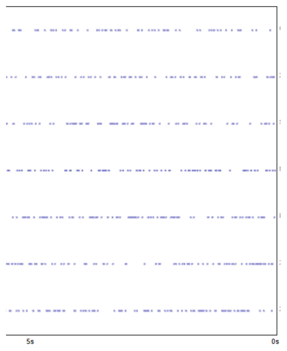

Fig. 3. Cleveland plot: The dots move from right to left, representing its timestamp along the idiom. Each "row" of dots represents one ordinal value.

2) Heat map: The common heat map resembles a table of colors, where rows and columns encode different attributes of the data and their intersection returns a number encoded with a color inside a spectrum. In our heat map, the position on the x-axis encodes a time interval. The number of points inside this interval yields the saturation of a rectangle called a cell. The cell's timestamp dictates the interval's position, which also moves towards the idiom's end.

The number of points inside a cell varies with the data's distribution, but also with the overall data flow of the visualization. This means the value scale needs to adapt to the idiom's data, increasing the maximum value if it was surpassed and lowering it when the value no longer accurately reflects the data in the idiom. The change is done gradually, in order to avoid visualization leaps that can compromise the user's comprehension of the data.

This idiom is suitable for various time periods, but presumably most suitable as a replacement of the Cleveland plot, since it creates a small aggregation of the data that can still distinguish its distribution, trading performance with temporal discretization of the data.

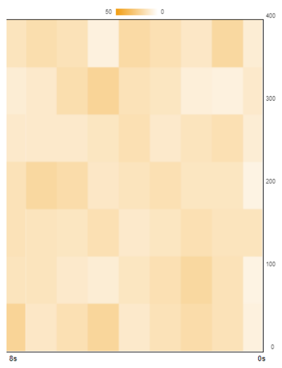

Fig. 4. Heatmap: Each cell's color encodes the data point quantities in a timespan. Since it represents a time interval, it moves along the data's path.

3) Ordinal line chart: As explained before, ordinal data shares a lot of characteristics with quantitative data, so it is understandable that an idiom is suitable for both data types. Since a mean value does not make sense in ordinal data, Pires et al. [13]'s solution of a mean line chart needs rethinking. The goal of the idiom is to understand the distribution of points and its evolution through time. To keep these functionalities, each ordinal value will have its own line chart encoding the number of data points inside a bin. Each line segment is connected to the previous bin's position, forming a line whose slope encodes the evolution of the data. This allows value comparison with the other lines if each line chart presents the same scale. Otherwise, one position would encode different values, confusing the information's analysis.

Like the heatmap, the idiom's scale needs to be adjusted to the values inside the idiom. If the amount of points surpasses the maximum scale value, the scale's upper limit increases to the new value plus a fraction of $\frac{1}{10}$ this maximum, avoiding a possible line overlap. If the maximum number of points in the idiom decreases to less than 70% of the scale's upper limit, then its value also decreases to a new maximum. These techniques avoid a constant change of scales.

If the idiom's timespan is too little, the evolution analysis might be insignificant, and since the idiom renews the depicted data, it can not serve as an accumulating idiom. This makes the idiom most suitable for middle modules with somewhat of a time magnitude.

4) Stacked bars: Stacked bars' goal is to effectively show proportions of the data values between every existing value group encoded with a different color and their values comprised in stacked lengths of a single bar, and then compared to other stacked bars. The bigger the size of the stacked bar, the bigger the proportion of total values in the value group of that specific bar. In OrderWarp each bar corresponds to an ordinal value, and the groups are encoded in the x-axis representing time intervals defined as **"Eras"**. The length of each full bar is the width of the idiom, and the Eras are encoded with a color from a list which is reused through time. Each Era has the same timespan, and every new data point in the idiom will belong to the latest one. A new Era is created when the last one has surpassed the idiom's beginning date. Since the time intervals move along the idiom, the first and last Eras will not be fully represented most of the time has their beginning has surpassed the idioms end or their end is yet to enter the idiom. This means these Eras will not have all their data points present for comparison.

This idiom is suited for middle idioms, since the Eras need a considerable interval to represent relevant visual cues. The idiom is also not capable of accumulating data, so it should not represent the latest module.

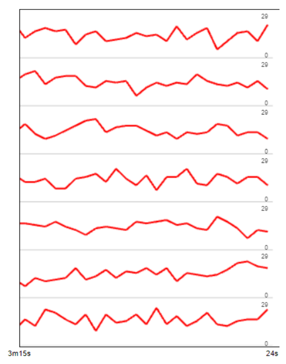

Fig. 5. Ordinal line chart: There is one line for each ordinal bin, updated after every bin duration of the previous transition.

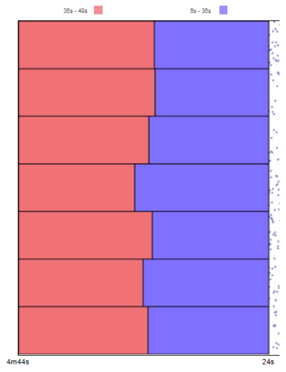

Fig. 6. Stacked bars: Each bar in the vertical axis represents an ordinal value, and the colored bars encode time periods called Eras. The Eras' width represents their percentage of records on the idiom for that ordinal value.

5) Bar chart: A bar chart is an idiom composed by bars where its size represents a quantity on one axis, and each bar is placed side by side for value comparison. With ordinal data each value is depicted with a bar, rather than an interval of values [13]. In our implementation, to keep a visual continuity the bar chart is horizontal, which means there are two x-axes, the common time visualization axis and one which identifies the number of data points in the bars.

To represent the arrival of new data, the bars gradually increase in size to their new values on the idiom. The scale updates the idiom's the maximum value, when the current hits 90% of the previous. To avoid constant change of the scale, instead of changing to the new maximum, it adds 30% to it. When the scale changes, the bars' length changes with it by gradually decreasing in size, since the maximum has increased.

This idiom is only suitable for a last idiom period, since there is no visual connection between the elements to the left boundary of the idiom, which maintains the data path idea. For a last period idiom, it works perfectly, as the constant update of bar size and scales allows the data to be grouped indefinitely since the start of the visualization.

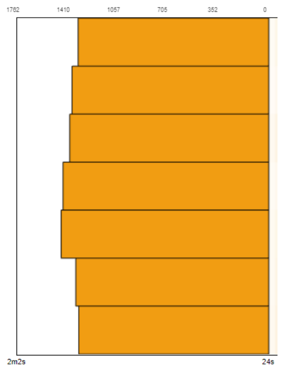

Fig. 7. Bar chart: The bars are placed horizontally, connected to the beginning of the idiom. Their width enlarges when data points arrive. The bars' sizes represent the number of data points since the beginning of the visualization.

G. *Transitions*

The transitions referred in this work are **horizontal transitions**, which represent the continuity between two idioms, by transforming the elements that represent the data in the former idiom **(A)** to the next idiom's **(B)** element properties [1]. The goal to perceptibly show that the information is the same in different aggregation levels. Part of this transformation is also reducing the element's velocity from idiom A (vA) to idiom B (vB), and does so using the movement equations:

$$\Delta t = |timestamp - idiomStartTime| \tag{3}$$

$$p = offsetA + vA + a\frac{\Delta t}{2} \tag{4}$$

Two different transitions were developed for each idiom combination, to provide alternatives and explore the properties of an effective transition, while avoiding an extensive combinatorial explosion of transitions, which would limit the implementation and analysis time. Some combinations were not studied because either the idioms were not suitable in their positions or were previously studied in [12]. Each combination of idioms also has a default transition, which is a simple fade out of the elements from idiom A. The picked transitions in plenty of cases followed similar or identical logics, and for that reason these transitions will be explained as one.

The first transition is only used in Cleveland plot to ordinal line chart. In **Growing Bars** the dots converge on top of a bar in the left boundary of the transition, causing the bar to grow to the value it will have in the ordinal line chart. This bar is attached to a line which connects to the next idiom's latest line, creating a seamless connection where there is always a line being produced, and growing with each entering dot, giving the idea that they are being grouped by pilling up in a bar whose height is its number of dots. Again, in this combination, there is a second transition called **Grouped Dots**. Similarly to Pereira et al. [12]'s scatter plot to line chart. The dots converge at the beginning of the respective line segment, giving the idea that the dots group together on the group's future on the ordinal line chart. The lines' opacity grows the closer they get to the next idiom.

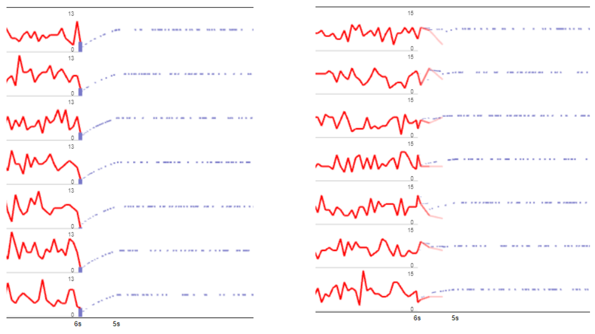

Fig. 8. Cleveland plot to Ordinal line chart transitions

Pilot Lines is a variation of the once again Pereira et al. [12]'s tested transition. As explained before, instanced meshes do not allow changing the size of a single point without computational drawbacks, and thus it is unfeasible to replicate the transition. Our variation of this transition, gradually scales down the instanced meshes' height until the first third of the transition. By scaling the height to zero, the dots convey the idea that they are being merged into a single point, however this is only the case for quantitative data, as the ordinal values follow a line and the scaling will almost be unperceivable. After the first third, a single rectangle is created with the size the dots would occupy in the next idiom. Once these rectangles hit the idiom boundary, just like stacking dots, they enlarge a horizontal bar that will "pushed" to the next idiom.

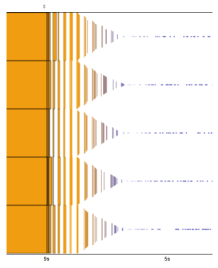

Fig. 9. Cleveland plot to bar chart's Pilot lines

On all combinations ending in either a heatmap or an ordinal line chart, a **Morphing** transition was applied. In these transitions, the elements that exit the first idiom gradually transform into the elements present on the second. This could be done by changing color, opacity, size and/or rotation. If more than one element is needed to represent the next element, then the first will morph into a portion of the second rather than the totality. The element morphing eases the visual transition, and the proportion clarifies how much data is being grouped.

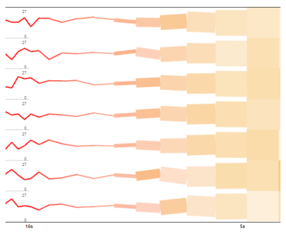

Fig. 10. Example of the Morphing transition

In ordinal line chart to heatmap combination, a different transition was implemented called **Line squeeze**. In this transition the lines will move towards the boundary while horizontally straightening, and after they reach their position in the heatmap, they increase in height almost as if they were squeezed against the boundary. Multiple lines will likely represent one cell, so each line will depict just a portion of said cell.

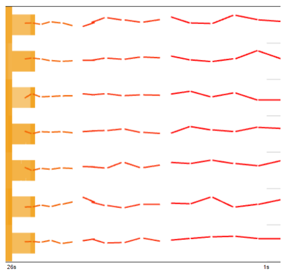

Fig. 11. Ordinal line chart to heatmap line squeeze

When starting with idioms whose elements are rectangles, there is an alternative transition of **Squares**. In order to understand the quantity of points inside the expelled element, it divides itself in smaller squares proportional to the total received points in that interval. If the ending idiom is the ordinal line chart, then the created squares will transform into small segments of a line by rotating and resizing themselves gradually. If instead the ending transition is a stacked bars or bar chart, then the squares will enter and increase the size of a bar, which will then seem to be pushed to the next idiom. The combination of the stacked bars to heatmap starts the same way with a rectangle being split into squares, but in this case they will enter a cell which starts with no opacity, until it reaches the intended opacity. There is also another transition called **Dissolving Lines**, beginning in an ordinal line chart and ending in a stacked bars/bar chart, which as the same logic as the Squares transition for the same ending idiom, the only difference is that instead of splitting the element once into squares, it ”dissolves” the lines into segments while entering the transition.

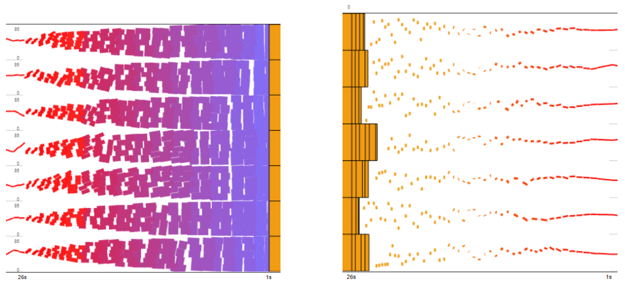

Fig. 12. Example of the Squares and Dissolving Lines transitions

Oppositely, if the ending idiom represents data through rectangles, then one alternative is a **Stacking Bars**. Here, the starting elements, if not already rectangles, transform into them. After transforming, the resulting rectangles make their way to the end of the transition, where they stack on top others rectangles, giving the idea that they are being aggregated into a single bar that represents the bin.

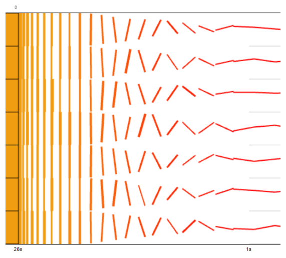

Fig. 13. Example of the Stacking Bars transition

IV. Evaluation

The evaluation of OrderWarp was done in two fronts, the measure of the system's performance and the confirmation of the effectiveness of the visual representations through user testing.

A. *System's performance*

Without a stable performance, the information comprehension is affected through visual leaps, frame freezing or worst-case scenario a system crash. To ensure the good functioning of the whole system, two tests were done in a visualization with three idioms and two transitions, with the same conditions. In the first test the idioms were Cleveland plot, an ordinal line chart and a bar chart, from left to right respectively, with Growing Bars and Stacking bars as transitions. In the second, a heatmap, a stacked bars and a final bar chart composed the visualization, separated by two Stacking Bars transitions. The idioms positions were chosen based on the theorized most suitable positions, and the combinations were chosen arbitrarily, ending in a bar chart to ensure the visualization depicts the full dataset. A study of all the combinations of idioms was also done to find out which transitions impact the performance the most.

All the performance tests were executed during 10 minutes with 100, 1000 and 10000 records per second data flows, using Google Chrome version 94.0.4606.81 in a Windows 10 environment with an Intel(R) Core(TM) i5-8400 CPU @2.80GHz, 16Gb of RAM memory in a 1920x1080 resolution, with NVIDIA GeForce GTX 1660 GPU. This duration ensures that any complications relative to time will likely be recognized.

Two measures were identified as essential to maintain system performance. The first being fps, with a set lower limit of 30 to ensure a fluid visualization. The values retrieved from the visualization were a fps average in 10 second intervals, guaranteeing no punctual spikes. The second measure was the stored memory of the system, which is the amount of structures required for the visualization to run. If this value is too high it might compromise the system's functionality, but more importantly if it increases linearly through time, the system will crash at some point in the future, thus the system can not run indefinitely.

Regarding the fps measure, all the tests stabilized quickly at some point in the analysis, always keeping the same fps value. For the two first tests, the fps value was much greater than the given limit, with the worst performing visualization out of the two, implementing the Cleveland plot. The lowest recorded value was 103.2 fps and the average value was 104.8 ± 3.8. When substituting the first idiom with the heatmap, the values rose, as expected, and it was possible to see that, since no idioms created elements depending on the data flow, the fps values were similar for all data flows. The transitions were tested with just two idioms and wider transition spans, and for this reason the values for Cleveland plot starting transitions suffered a decrease in fps for all cases. However, the Pilot Lines transitions had lower fps than the limit only for 10000 points data flow.

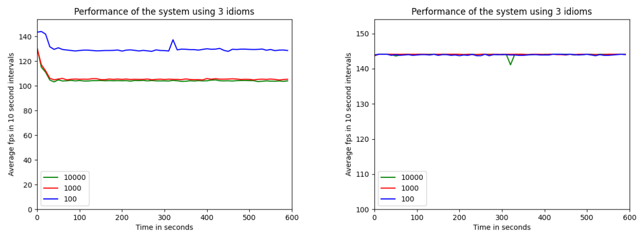

Fig. 14. Fps's evolution through time results for the first and second tests.

For the memory usage, it was clear in every test that, even though larger data flows will hold higher memory values as expected, its evolution was constant through time regardless of the data flow. No surprising information was discovered when studying the memory usage in the transitions, since the ones who implemented more visual elements return higher values.

B. Discussion

In both measures, the system fulfilled its goals, so we can conclude that system's architecture in WebGL technologies allows the visualization of Big Data Streaming for as long as required.

C. User Testing

To evaluate the effectiveness of the implemented representations, we conducted usability tests to a group of users. The study was done via questionnaires created with Google forms. On total, there were 13 questionnaires, an initial user profile questionnaire, followed by one questionnaire for each possible combination of idioms. Each user could complete any amount of different testing questionnaires out of all 12 throughout a two-week period. The questionnaires were tested online, to reach more people and not require the presence of a guide. The tests were assigned using an ID, corresponding to a row in a 12 by 12 *Latin square* distribution, that ensures no bias in the order attribution.

1) Questionnaires: The 12 questionnaires all followed the same structure. The users were first presented with a description of the purpose of the study and their role in it. After this there was an explanation of the OrderWarp accompanied by a video [1] covering the system step by step. All user testing videos were recorded with a starting data flow of 100 points per second.

The next questionnaire sections focused on the analysis of each idiom in the combination individually. These sections start off with another video showing only the functioning of the target idiom, followed by objective questions that allow the quantification of the idiom's effectiveness. There were then two common questions to evaluate the cognitive demand of the idioms, which asked - *"How confident are you on the previous answers?"* and - *"How many times did you have to rewatch the video?"*. Finally, the user is encouraged to identify any problems with the idiom and/or offer suggestions. To avoid duplicate answers in questionnaires with the same idiom, users could skip these section if already answered.

The next two questionnaire sections serve to evaluate the two proposed transitions for the questionnaire's combination. As before, there was a testing video followed by three questions - *How clear is it that the data is being aggregated*; *How fluid is the transition*; and *Which of these is statements is true regarding the transition's logic* with the first two presenting a Likert scale, and the next one, three different written options. The users were then asked suggestions again, the previous cognitive demand questions, and a final transition preference question between both alternatives.

2) Participants: 24 people between the ages of 18 to 30 (91.7%) and 50 to 60 (8.3%) responded to the questionnaires. 16 identified as male, 7 as female and one as non-binary. The users' experience with visualizations was dispersed, a third used it occasionally, 37.5% every week, 12.5% every day, and the rest never used or only once a month. Almost every user knew the bar chart, line chart, and scatter plot idioms, and a smaller considerable proportion was familiar with a heatmap and stacked bars.

3) Results: The results are split in idioms and transitions

3.A) Idioms: For the **Cleveland plot** idiom, the displayed video [2] showed an almost binomial distribution on one of the values, and then suddenly an agglomeration of points emerges

[1]https://youtu.be/RcwU2Bc3KT8
[2]https://youtu.be/uKTqZ0lQWDg

on a distinct ordinal value. The first question (Q1) targeted the user's distinction of the distribution of the ordinal values. The next focused on the recognition of agglomerations (Q2), followed by identification of which values were affected by this agglomeration(Q3). The results are shown in table I.

Question	Q1	Q2	Q3
Results	100%	100%	83.3%

TABLE I
CLEVELAND PLOT EFFICIENCY RESULTS

Q1 and Q2 proved the visualization to be quite effective in its proposed tasks. Yet the identification of agglomerations return lower results, but still positive, with 83% answering Q3 correctly.

Heatmap's video [3] followed the same logic as before, with a different dataset and agglomeration instant. Since the analyzed tasks are identical, so were the three first questions, adapted to the new dataset. The idiom's section presented two extra questions, the first (Q4) researched the comprehension of the lack of records in the idiom. The second's (Q5) purpose was to understand the scale changing perception with a Likert scale.

Question	Q1	Q2	Q3	Q4	Q5
Results	100%	95.7%	87.0%	82.6%	4 (1)

TABLE II
HEATMAP EFFICIENCY RESULTS

The results can be found in table II. The three first questions' results showed similar findings as before, with a slight decrease in agglomeration identification balanced by an increase in the affected values' recognition. 82.6% perceived the idiom's scale correctly when no values were shown, and considered its change easy to understand, with 4 (1).

The test of the **ordinal line chart** was different. The video's [4] dataset was designed so that one of the values presented a larger amount of points, testing their comparison with the previous Q1. Two of the other values tested the analysis of the idiom's evolution (Q2), with one irregularly increasing linearly and the other decreasing. The last question focused on understanding if the idiom accurately perceives the exact value of a line (Q3). The results can be found in table III.

Question	Q1	Q2	Q3
Results	95.7%	78.3%	56.5%

TABLE III
ORDINAL LINE CHART EFFICIENCY RESULTS

The results were positive for the first couple of questions, however, the third identified a problem on either the logic or the implementation of the scale, with only 56.5% finding the correct answer.

Stacked bars goal is to understand the proportions of points arrived in different time intervals called Eras, which were the test subject of the questions. Stacked bars video [5] showed a normal distribution of the data through various Eras, and then, a sudden agglomeration on the S value.

[3]https://youtu.be/2KNuAX6ZmBI
[4]https://youtu.be/ozKu4JvQ1-A
[5]https://youtu.be/ZVYVDIICzAY

The first question's (Q1) objective was to study the comparison between Eras, by asking to identify the most common one. The second question's (Q2) goal was to identify an outlier Era in an ordinal value, by identifying the one where the agglomeration was present. The third question (Q3), asked the user to identify the percentage value of an Era. In the last question, when asked the true or false question on whether the first and last idioms were fully represented (Q4), the users had to understand that these two Eras' had missing values. Oppositely, the scale of the proportions was well distinguished, with a vast majority correctly identifying the percentage of records in an Era (Q3). In the last question (Q4), the study found that only 20% of users understood that the timespan of the idiom did not represent the full interval of the first and final Era. Furthermore, a number of users commented that this idiom was not intuitive. The results can be found in table IV.

Question	Q1	Q2	Q3	Q4
Results	72.7%	59.1%	86.4%	20.0%

TABLE IV
STACKED BARS EFFICIENCY RESULTS

This idiom presented somewhat of negative results. 72.7% found the most common Era for most ordinal values, but 40.9% said that it was not clear in which Era this did not occur. Furthermore, the identification of the individual values was clear for most users, with 86.4% finding the correct answer, however, only 20.0% understood that the Eras were not fully represented in the idiom.

Bar chart's video [6] demonstrates a simple binomial distribution of points around the L value. The idiom was then tested with just two questions, the first (Q1), tested the efficiency of the idiom on comparison by asking to identify the most common value. The second (Q2) asked to identify the number of records in an ordinal value. The results can be found in table V.

Question	Q1	Q2
Results	100%	88.9%

TABLE V
BAR CHART EFFICIENCY RESULTS

Both questions showed good results in the two measures with 100% and 88.9% correct answers respectively.

An additional analysis on the comparison of the confidence values for each idiom and the amount of times the users rewatched the videos, represented in the box plots on figure 15, shows the cognitive demand of the users per idiom.

3.B) Transitions: As explained in section IV-C1, the transitions sections asked three questions, which targeted the transitions' aggregation, fluidity and logic. To understand if there were statistically significant differences between the first two measures, each combination of transitions underwent a *Wilcoxon signed-rank test* except for one transition that found a non-normal difference median distribution and was tested with the *sign test*. For the logic measure, the question's result type had to be tested with the *McNemar's Test*. However,

[6]https://youtu.be/FrvASuI2RE4

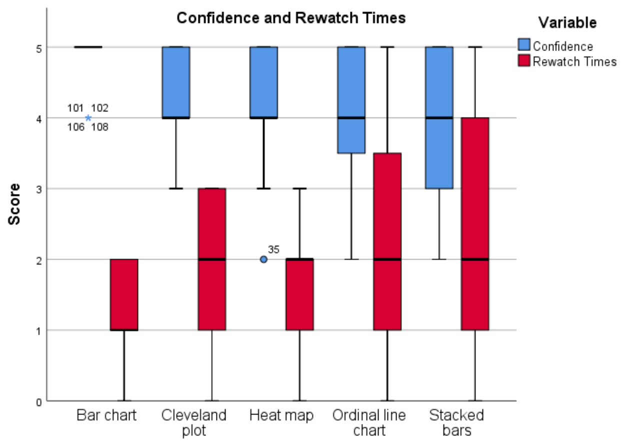

Fig. 15. Answer confidence and rewatch times comparison between idioms

no significant differences were found in any combination, and consequently will not be shown in the following results. Furthermore, in this work, any transition with a median value of 3 or lower in any of the measures is considered not to be effective enough to represent the changes between idioms.

For the Cleveland plot beginning transitions, only the first combination with the ordinal line chart was compared statistically, since it was the only one with two transitions. The **Growing Bars** transition saw statistically better results with $p < 0.0005$ in both measures and $z = -3.678$ for fluidity alone. The aggregation measure was the only one which used the *sign test*. For the other two idiom combinations, there was only the **Pilot Lines** transitions, which proved to be effective in both combinations and measures.

Heatmap's beginning transitions, resulted in a statistical significance between Line Morphing and Squares measures, when addressing the ordinal line chart as the ending idiom. The fluidity measure saw significant differences, deeming **Line Morphing** more suitable. For the next two combinations, **Stacking bars** was picked as more suitable. However, when combining with stacked bars, no significant differences were found, the decision was done through user preference. In bar chart's case, the transition was significantly better in both measures.

Starting with the ordinal line chart, the transition picked for the heatmap combination was **Cell Morphing**, which returned statistical significant improvements over Line Squeeze in both measures. For stacked bars and bar chart, the same transitions were implemented and analyzed, but no statistical distinctions were found, and so the user preference chose the **Stacking Bars** transitions.

In the last beginning idiom, both combinations ending in heatmap and line chart were only distinguished by user preference, with the resulting transitions being **Squares** and **Line Morphing**, respectively. However, the second transition's fluidity values did not surpass the given median limit, thus it should be substituted or rethought. From stacked bars to bar chart, the **Squares** transition saw statistical differences in the aggregation measure, which declared it the most suitable. The fluidity values barely surpassed the median limit.

D. Discussion

Addressing idioms, Cleveland plot, heatmap, ordinal line chart and bar chart, all showed positive results in the questionnaires, with most questions answered correctly by most part of the users, proving to be good representations for ordinal data on the Graceful degradation technique. Cleveland plot generates performance issues and visual clutter when the data flow is too high, and so, we suggest changing idiom to heatmap in these cases. The good results in both the Cleveland plot and its transitions, demonstrates that instanced meshes can represent well data even with its limitations.

Ordinal line chart's had good results in evolution understanding and value comparison, yet the identification of the exact values return poor results, and should be rethought if used in a real-context. Oppositely, stacked bars idiom had lower results in all questions, except for proportion value distinction. The idiom was found confusing by users in the feedback question, which affect their data comprehension and should either not be considered an adequate idiom or also rethought. It was possible to verify that the idioms, which were achieved better results and were less demanding cognitively, were the most known idioms in the profile questionnaire.

The most suitable transitions of the suggested idiom's combinations can be found in table VI. The ones with no statistical differences are identified with an asterisk. The transitions that got better results implement less visual elements for the most part, and in further work this should be taken into consideration. It is important to say that the depicted transitions are not necessarily the best transitions, just the best found alternatives.

Idiom	HM	OL	BC
CP	-	Growing Bars	Pilot Lines
HM	-	Line Morphing	Stacking Bars
OL	Cell Morphing	-	Stacking Bars*

TABLE VI
RESULTING TRANSITIONS TABLE

V. Conclusion

To extend the Graceful degradation technique to ordinal data, we implemented various idioms and transitions between them, focused on the analysis of this data type. The visualization allowed a flexible selection of connected task-oriented idioms, as well as their width and timespan. From this selection, the visualization is capable to adapt to different datasets, providing different levels of detail. All the representations were implemented using WebGL, migrating the initial work to a performance enhancer technologies, rendering objects with the help of the GPU.

The evaluation phase split the tests in two. The first proved the proposed architecture provided a smooth visualization that could display information indefinitely. The second part tested the designed representations through user testing, finding the idioms that could effectively depict ordinal data and those who could not, and the transitions which portrayed the change in aggregation level.

The new developed system lacks user interactivity, which is a very valuable in information systems, and should be treated as a crucial feature to implement. Some idioms still require improvements, in the ordinal line chart's case it would be a restructuring of its scale. If stacked bars is to be kept, its logic as to be deeply altered. More studying of the transitions is required to understand the impact of its width and timespan on the users comprehension.

References

[1] Filipa Castanheira, Daniel Mendes, and Daniel Gonçalves. "FastViz - Visualizando Big Data em Evolução Dinâmica". In: 2020.

[2] S. Cheng, K. Mueller, and W. Xu. "A framework to visualize temporal behavioral relationships in streaming multivariate data". In: *2016 New York Scientific Data Summit (NYSDS)*. 2016. DOI: 10.1109/NYSDS.2016.7747808.

[3] O. Daae Lampe and H. Hauser. "Interactive visualization of streaming data with Kernel Density Estimation". In: *2011 IEEE Pacific Visualization Symposium*. 2011. DOI: 10.1109/PACIFICVIS.2011.5742387.

[4] Remco Chang Daniel E. Kee Liz Salowitz. ""Comparing Interactive Web-Based Visualization Rendering Techniques"". In: (2012).

[5] Mohammad S.Alam Evgeniy Yur'evich Gorodov Vasiliy Vasil'evich Gubarev. "Analytical Review of Data Visualization Methods in Application to Big Data". In: Russia: Journal of Electrical and Computer Engineering, 2013. DOI: 10.1155/2013/969458.

[6] N. Ferreira et al. "Visual Exploration of Big Spatio-Temporal Urban Data: A Study of New York City Taxi Trips". In: *IEEE Transactions on Visualization and Computer Graphics* (2013). DOI: 10.1109/TVCG.2013.226.

[7] T. Fujiwara et al. "An Incremental Dimensionality Reduction Method for Visualizing Streaming Multidimensional Data". In: *IEEE Transactions on Visualization and Computer Graphics* (2020). DOI: 10.1109/TVCG.2019.2934433.

[8] Y. Hashimoto and R. Matsushita. "Heat Map Scope Technique for Stacked Time-series Data Visualization". In: *2012 16th International Conference on Information Visualisation*. DOI: 10.1109/IV.2012.53.

[9] Valen E Johnson and James H Albert. *Ordinal data modeling*. Statistics for Social Science and Behavorial Sciences. New York: Springer, 1999. DOI: 10.1007/b98832. URL: https://cds.cern.ch/record/1608780.

[10] Florian Mansmann et al. "StreamSqueeze: a dynamic stream visualization for monitoring of event data". In: *Visualization and Data Analysis 2012*. International Society for Optics and Photonics. SPIE. DOI: 10.1117/12.912372.

[11] Peter McLachlan et al. "LiveRAC: Interactive Visual Exploration of System Management Time-Series Data". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '08. Florence, Italy: Association for Computing Machinery, 2008. ISBN: 9781605580111. DOI: 10.1145/1357054.1357286.

[12] Tiago Pereira, Daniel Mendes, and Daniel Gonçalves. "VisMillion and change". In: 2019.

[13] Gonçalo Pires, Daniel Mendes, and Daniel Gonçalves. "VisMillion: A novel interactive visualization technique for real-time big data". In: *2017 IEEE International Congress on Big Data (BigData Congress)*. INESC-ID Lisboa, 2019. DOI: 10.1109/BigDataCongress.2017.49.

[14] Joris Sansen et al. "Visual Exploration of Large Multidimensional Data Using Parallel Coordinates on Big Data Infrastructure". In: *Informatics* 3 (2017). ISSN: 2227-9709. DOI: 10.3390/informatics4030021. URL: http://dx.doi.org/10.3390/informatics4030021.

[15] Sye-Min Chan et al. "Maintaining interactivity while exploring massive time series". In: *2008 IEEE Symposium on Visual Analytics Science and Technology*. DOI: 10.1109/VAST.2008.4677357.

[16] Ralf Krestel Tim Repke. "Topic-aware Network Visualization to Explore Large Email Corpora". In: EDBT/ICDT Workshops 2018, 2018.

[17] Jonas Traub et al. "I2: Interactive Real-Time Visualization for Streaming Data". In: EDBT, 2017.

[18] Biye Jiang Zhicheng Liu and Jeffrey Heer. "imMens: Real-time Visual Querying of Big Data". In: Eurographics Conference on Visualization (EuroVis) 2013, 2013. DOI: 10.1111/cgf.12129.