International Journal of Service and Knowledge Management
International Institute of Applied Informatics
2019, Vol. 3, No. 1, 19 – 34

A Layered Canvas Synchronization Mechanism for an Adaptable Presentation System

Eishun Ito* , Tadachika Ozono*, Toramatsu Shintani*

Abstract

An adaptable presentation is expanded an interactive presentation, that cognizes the presentation state, and that shows the materials, slides, and contents. An adaptable presentation requires the slide relationship that has the connection between slides. Therefore, an adaptable presentation requires too many objects. Too many objects in a canvas degrade rendering performance. We called this problem a canvas redrawing problem. We developed a layered canvas synchronization mechanism to solve the canvas redrawing problem for an adaptable presentation. A layered canvas synchronization mechanism improved the performance with layered canvases. Redistribution of drawing objects to each layered canvas suppresses the cost and realize the real-time rendering on a web application. The existing layered canvas algorithms remained a problem when we updated objects at the same time. We proposed a batch algorithm, that is a novel way of layered canvases. The experimental evaluation indicated the effectiveness of the batch algorithm and our system. Our contribution is that layered canvases solved the canvas redrawing problem and that the batch algorithm enabled us to update 100 objects at the same time.

Keywords: HTML5 canvas, real-time rendering, application, presentation

1 Introduction

We proposed a new presentation style called an adaptable presentation[1]. An adaptable presentation is expanded an interactive presentation, that cognizes the presentation state, and that shows the materials, slides, and contents. An adaptable presentation requires the slide relationship that has the connection between slides. Therefore, an adaptable presentation require too many objects.

An HTML5 canvas element is one of DOM elements to render graphics on a web page. We call an HTML5 canvas element canvas. We usually use a canvas for an interactive web application. Each canvas represents a bitmap image. An HTML5 canvas element has the fast rendering performance. We applied a canvas to an adaptable presentation system because an adaptable presentation requires the performance to render a lot of objects and the interaction with users. Too many objects in a canvas degrade rendering performance.

* Department of Computer Science Graduate School of Engineering Nagoya Institute of Technology

We called this problem a canvas redrawing problem. We developed a layered canvas synchronization mechanism to solve the canvas redrawing problem for an adaptable presentation. A layered canvas synchronization mechanism improved the performance with layered canvases. Redistribution of drawing objects to each layered canvas suppresses the cost and realize the real-time rendering on a web application.

The existing layered canvas algorithms remained a problem when we updated objects at the same time. We proposed a batch algorithm, that is a novel way of layered canvases. The batch algorithm enables us to update some objects at the same time. The batch algorithm is more effective in the practical use case than other algorithms. The batch algorithm reveals difference from other work about layered canvases. We confirmed the performance with the experimental evaluation and address the effectiveness. Additionally, we estimated and implemented a layered canvas mechanism. Our system estimated the drawing cost adaptively. In order to prove our system, we indicated three evaluations. Previous works had the focus on the drawing time to prove the performance. On the other hand, we investigated the memory usage of layered canvases. On the viewpoint of the memory usage, the result indicated the worth of the batch algorithm and our system. In this paper, we explain the details.

The remainder of this paper is organized as follows. In Section II, we discuss the requirement of an adaptable presentation with related works. We describe a layered canvas synchronization mechanism in Section III. Section IV addresses an application with a recommender mechanism for an adaptable presentation. Section V and Section VI shows the evaluation experiment and discussion. Finally, Section VII presents our conclusions.

2 Related Work

Previous works of Layered canvases had the focus on the frequency. Katayama et al.[2] proposed a drawing-frequency based layered canvas mechanism(DFLC). DFLC assigned objects to layered canvases based on the drawing-frequency of objects. The high frequency objects were assigned to upper layers. The low frequency objects were assigned to a bottom layer. DFLC was effective if the frequency of objects were known previously. For unknown drawing-frequency objects, Katayama et al.[3] proposed a generational layered canvas mechanism(GLC). GLC reassigned unknown drawing-frequency objects to layered canvases based on the frequency that means how many times an object was updated after its generation. A high frequency object was reassigned to a high frequency layer. GLC had the problem that a system had to generate layered canvases previously. It's difficult to know the number of layered canvases because the number of layered canvases depends on an application. Ozono et al.[4] proposed a stable layered canvas mechanism(SLC) in order to solve this problem. SLC showed the algorithm of dynamic layering a canvas. SLC dynamically generates a layered canvas based on its drawing cost. SLC can be applied to some kinds of applications.

We describe related works of presentation. A web presentation is implemented as a web application, that has advantages: introduction, sharing, or seamless. We applied a presentation. Moreover, we applied an interactive presentation, which improved discussion in the presentation between presenters and audiences. Prezi[1] put slides

[1] https://prezi.com/

together on one page, and expresses page transition by pan and zoom. The slides on one page shows us the structure, and the overview of the presentation. Edit page in the system is designed considering a view to make sense the structure of the presentation Slides[2] makes page transition in multiple directions of the slide. SlideShark[3] converts the existing presentation file so that it can be used among multiple devices such as iPad. SlideDogs[4] provides a seamless presentation corresponding to various forms of presentation. To correspond to a seamless presentation, this system converts some presentation format, pptx or pdf, into Slide Object format which adjust for rendering in the canvas.

Visualizing Real-Time Questionnaire Results to Promote Participation in Interactive Presentations[5] Interactive presentations intend to gather the real-time feedback from an audience while delivering the presentation. Inoue et.al a hybrid interactive presentation system that consists of the traditional slideware application, such of Microsoft Office PowerPoint or Apple Keynote, and a web application gathering feedback from an audience. A web application to gather feedback uses a real-time questionnaire whose result is shown on the presentation slide. To share the knowledge through the presentation, they developed SlideWiki[7] that is a cloud sourcing platform for a collaborative creation of the presentations. To share the knowledge is the concept of the SlideWiki. Not only to share, but to reuse, we developed a collaborative editing presentation mechanism. A Reactive Presentation Support System based on a Slide Object Manipulation Method[8] Reactive presentation can immediately react an unexpected context change: the state of audiences, unexpected questions and audience’s knowledge. NextSlidePlease[9] is a slideware application a directed graph structure approach for authoring and delivering multimedia presentations. HyperSlides[10] is a dynamic presentation prototyping, that uses a simple markup language for creation of hierarchically structured scenes, which are transformed into hyperlinked slides of a consistent and minimalist style. Chatplexer automatically estimates the important chat opinions and uses during a presentation[11]. Chatplexer uses a cross-channel reply(XCR). XCR has the function for a face to face reply and for a text-based reply about a slide content. A chat log is important for an adaptable presentation because it represents the feedback from audiences. Our system can’t rearrange objects in the slides, such as text, figures, or images. Niwa[12] at study is one of related works. In the presentation, the animation(added, deleted, or moved) of some contents improve audiences to understand, and enable presenters to expand the expression.

Considering a practical use case, we assumed the requirement: it allows many people to use the application, to share the materials, and to render fast as long as a real-time performance. Most existing presentation applications didn’t require a fast rendering because they supported a stereotype presentation. However, to support an adaptable presentation, it’s is essential to support the communication between presenters and audiences. The significant communication require a high performance, which is the reason why we aimed to a real-time performance. Our goal was that a drawing time was under 50ms for smooth communication.

An adaptable presentation requires a lot of objects. We set the components as following: slide contents, links, related materials or relationship. The evidence of the relation between slide materials are as following: a next page, a previous page, an important

[2]https://slides.com/
[3]https://www.slideshark.com/
[4]http://slidedog.com/

phase, a definition, a similar image or the content a presenter showed previously. We assumed that each presentation material consists of the average 30 slides pages, and that each page has the average 5 objects: text, image or other figures. Each presentation materials had 150 objects at least. Additionally, a slide material on an adaptable presentation has the complicated relationship that represents the candidates of page transition. We display related presentation materials and on the system for editing the slide architecture. If there are the relationship among 10 presentation materials and each slide page has 6 links at most, there are about 10,000 objects. Such a quantity of objects causes the canvas redrawing problem.

3 A Layered Canvas Synchronization Mechanism

We developed a layered canvas synchronization mechanism for a web collaborative works. A layered canvas synchronization mechanism is a rendering system which uses a canvas. We describe the details as follows.

3.1 Canvas Redrawing Problem

A canvas element(Canvas) defined on HTML5 is one of the most widely used standards for rendering 2D graphics on the Web. The canvas is widely used in complex visualizations such as graphs, game graphics or art. People on a collaborative works require the communication with a variety of tools. Canvas is scriptable with HTML or JavaScript, suits the application that use animation or update images frequently, such as a collaborative works. We point out some potential problem with the canvas. Generally, a web collaborative editing system renders the huge number of canvas objects. The huge number of objects worsens the performance of redrawing the canvas.

We need to solve this redrawing problem to achieve fast Web applications. We call this problem Canvas Redrawing Problem. One of the most severe problems on the canvas elements in collaborative web applications is that unnecessary redraw is required on updating or removing some drawings on the canvas. Because the canvas is a bitmap image and then the canvas cannot separate each drawn object. Fig.1 shows Canvas Redrawing Problem. When we move an object B on a canvas element, the redrawing caused by the moving propagates to the other object A and object C on the canvas. To move object B cause the lack of other objects since object B because of bitmap images. When an object is updated and redrawn, it has to clear a canvas and to redraw the others in order to update a canvas correctly. Thus, more objects are rendered on a canvas, more time costs to finish to render.

3.2 Approach

To solve a Canvas Redrawing Problem, we applied the method Ozono[4] proposed. The quantity of objects in the canvas causes of worsening the performance of rendering. Thus, we adapt the strategy to reduce objects in a canvas. We described the way to layer. HTML5 Canvas element can turn color into clear, so that the canvases overlap and appear to be one image. To distribute objects, the number of objects each layer has decrease, and it suppresses drawing time. We noted that redrawing objects correctly requires the compliance to keep the order of objects. The order means that objects are rendered from the upper to the bottom.

The way to layer canvases, we proposed two methods. The one is a Layered Canvas Architecture, that piled up the group of layered canvases. Fig.2 shows a

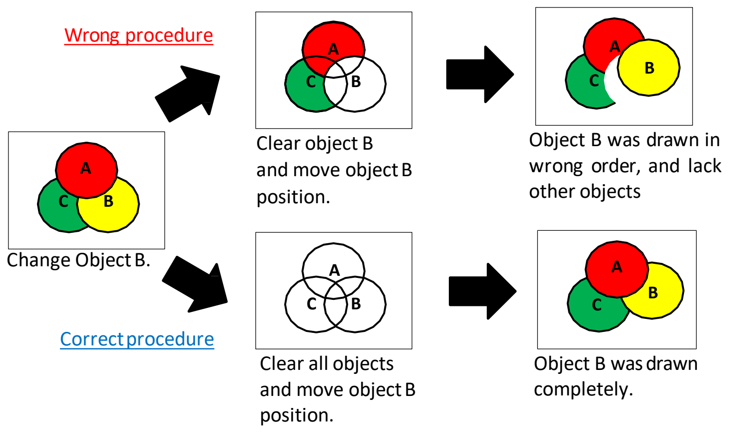
Wrong procedure
A
C
B
Change Object B.
Clear object B and move object B position.
Object B was drawn in wrong order, and lack other objects
Correct procedure
Clear all objects and move object B position.
Object B was drawn completely.

Figure 1: Canvas Redrawing Problem.

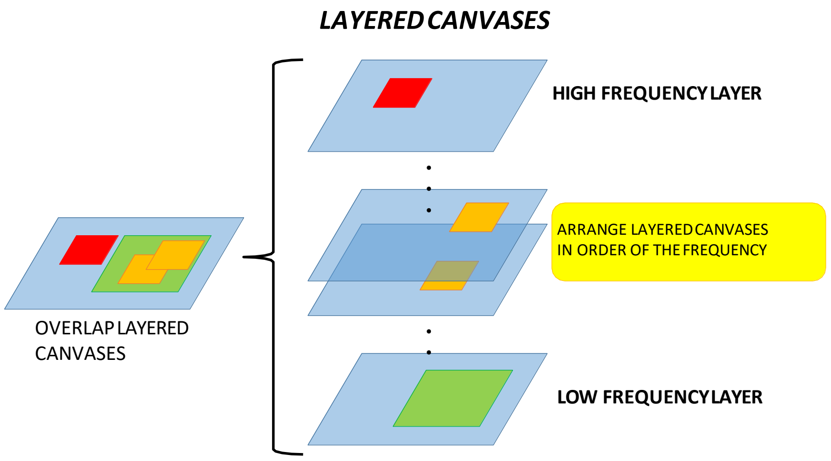

Figure 2: A Layered Canvas Architecture.

Layered Canvas Architecture. Fig.2 shows three groups of layered canvas and its update frequency. There are many kinds of objects rendered in the Canvas. For example, animation or effects are generated because of change of the object's status. Thus, its frequency is higher than other objects. Besides, the background or the graphical area of the group seldom update. When a layer updates, every object on the layer updates. Hence, objects should be separated on the update frequency. A layered canvas synchronization mechanism can set the threshold of separating, and allows users to set the threshold as their aim. If we know the update frequency of objects, we generate groups of layered canvases, and assign them on their frequency.

Another is the method to layer canvases dynamically. A layered canvas architecture is static and execute before the start step. To adapt the application status, we considered the cases. Fig.3 shows the cases, cost, and solutions. The cases are time to add a new object, and to update or delete an object.

First, we describe when we add an object. A layer management module, that is part of a layered canvas synchronization mechanism, controls when to divide a layer. A layer management module divides a layer according to the quantity of objects on the layer and the canvas size, as cost of redrawing objects. We assumed that it's high probability the latest object will update. If we add a new object to a layer, we should assign it to the latest layer.

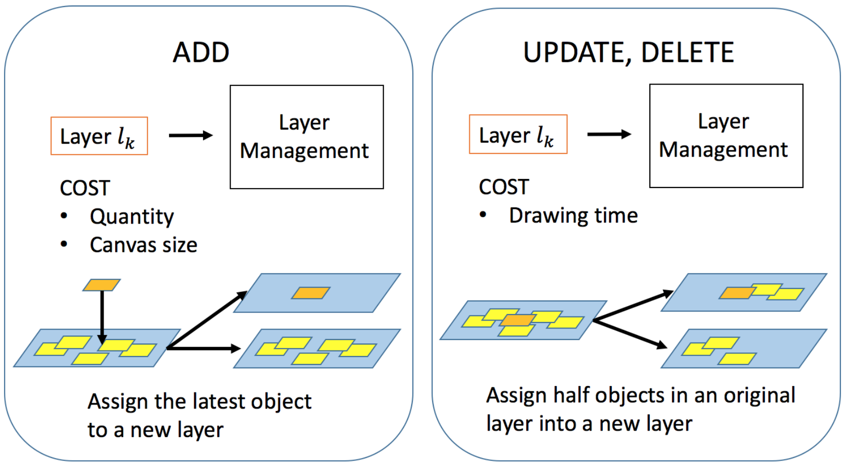

Figure 3: How to divide a layer.

A layer management module estimates the cost after a new object is added. If the cost is over the threshold of dividing, the system generates a new layer, arranges it on the top of layers, and assigns a new object into a new layer.

Second, we describe when we update or delete an object. The difference with adding is that the object to be operated is on a layer. The strategy to assign is splitting objects in half, and our system assigns them into layers. It may be better to assign a group including the updated object into a layer which has less objects. However, we don't know the order of the updated object. We have to search the updated object, and it additional cost of dividing a layer. Therefore, we adapted the way to split half. It's brought to divide a layer by the performance of drawing objects on a layer. A layer management module measures the drawing time, if it costs over, generates a new layer, and assign objects in half.

A layered canvas synchronization mechanism defined a Card Object that's drawn in the Canvas. An HTML5 Canvas element renders bitmap images and doesn't have objects. An HTML5 Canvas element follows the script written in JavaScript. Hence, we defined objects that has the script to render on an HTML5 Canvas element.

Card objects have many advantages. First, identification. Assigned id, every object is distinct by hitting, such as mouse click, a touch event, or collision between object and object. Second, event handling. We can design different events on each object. For example, if the object hit, change a different color. If we drag the object on its corner, it expands as the movement. Finally, arrangement. We can set mouse event on objects.

In the collaborative application, a lot of objects update at the same time. When an objects update, we have to redraw other objects. Hence, it costs less to gather updated objects and render them altogether. We proposed Batch algorithm that shows Algorithm1. Batch algorithm enable us to render objects altogether. We describe the flow. Objects to update are stored in Object Stack(*Stack*). Each object in Stack is resisted in *Map* with the id of a layer the object belongs to. *Map* is a key-value store, and defines the id of the layer as the key, and store objects as the value. First, storing objects for a while. Stored objects are assigned into Map following a layer. After assigning, objects are updated by a layer.

Algorithm 1 Batch Algorithm.

```
 1: for $o \in Stack$ do
 2:     $l \leftarrow Layer(o)$
 3:     $key \leftarrow l.id$
 4:     if $map[key].isEmpty()$ then
 5:         $map[key] \leftarrow newArray()$
 6:     end if
 7:     $map[key].add(o)$
 8:     $update(o)$
 9: end for
10: for $key$ in $map.keys()$ do
11:     $l \leftarrow getLayer(key)$
12:     $update(l)$
13: end for
```

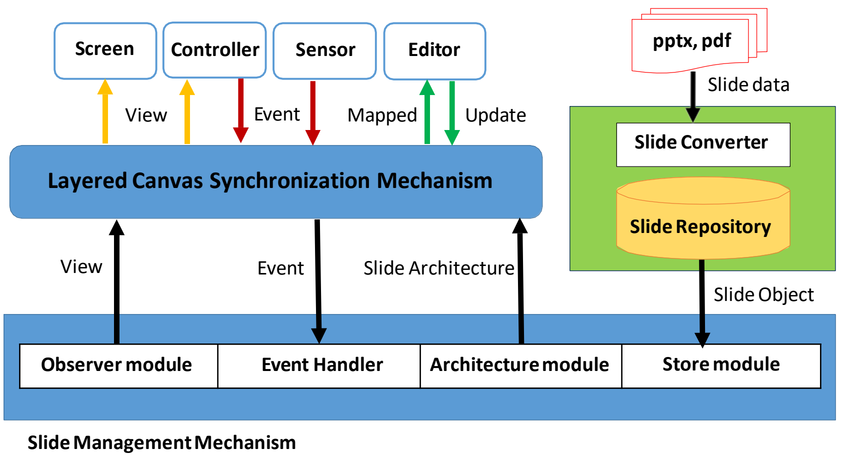

Figure 4: Architecture.

4 Application

We developed an application for an adaptable presentation using the layered canvas synchronization mechanism. The application had a slide management mechanism and a recommender mechanism. First, we describe a slide management mechanism with its architecture. Next, we describe a recommender mechanism.

4.1 Architecture

Fig.4 shows a system architecture of a slide management system. The slide management mechanism consists of following components: an edit mechanism, a layered canvas synchronization mechanism, an editor, a screen, a controller, sensors, a slide converter and a slide repository. Moreover, the slide management mechanism consists of a store module, an observer module, an event handler and an architecture module. We describe its details.

Slide source files such as pptx or pdf are stored in the slide repository. The files are converted into slide objects by a slide converter. The slide object consists of images and texts

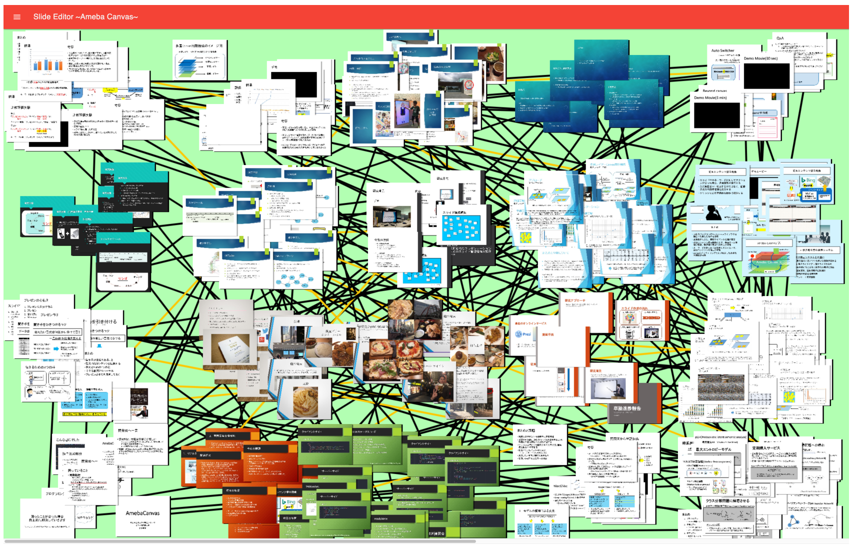

Figure 5: Edit Screen.

in order to present, and is a minimum configuration. Slide objects in the slide repository is conducted to the store module when editing. The store module provides objects in an editor, that is consists of an edit screen. Slide objects are converted into Card objects that is for a layered canvas synchronization mechanism. A layered canvas synchronization mechanism premises an HTML5 canvas element. Card objects can be drawn in an HTML5 canvas element. Fig.5 shows the edit screen. Editor screen shows card objects that represents the slide architecture. Update data in the editor are conducted to an architecture module. The architecture produces Editor's results.

When we make a presentation, we use an observer module and an event handler. The observer module is a slide provider. An architecture module sends a slide architecture to the observer module. According to a slide architecture, an observer module provides slides. The observer module also provides candidates of slide page transition on the controller of the presentation. The controller shows the candidates, and presenter can select the next slide. Selected result is sent to the event handler.

The event handler is the module that receive the events data from controller or Sensor in order to verify the permission which can occur to the assigned action. Fig.6 shows a screenshot of the controller. On this screen, the controller shows the candidates: in the Fig.6, an additional slide, a next slide, and a previous slide. Each candidate consists of its title, description, and thumbnail. If we select, we touch the part of the slide. The slide candidate has the notification function. A controller notifies the event defined for the particular slide. For example, a presentation overran the schedule, controller generate notification. The event handler verified the notification. If allowed, the event handler generated the event, and sent to Observer module. Sensors also generate notification. In this paper, Sensors include cameras, microphones, Leap Motion and so on.

4.2 Recommender Mechanism

In this section, we describe a recommender mechanism for an adaptable presentation. There are two cases to recommend. First, presenters edit a slide architecture. When we edit a slide architecture, editors demand the candidates of the destination of a page transition.

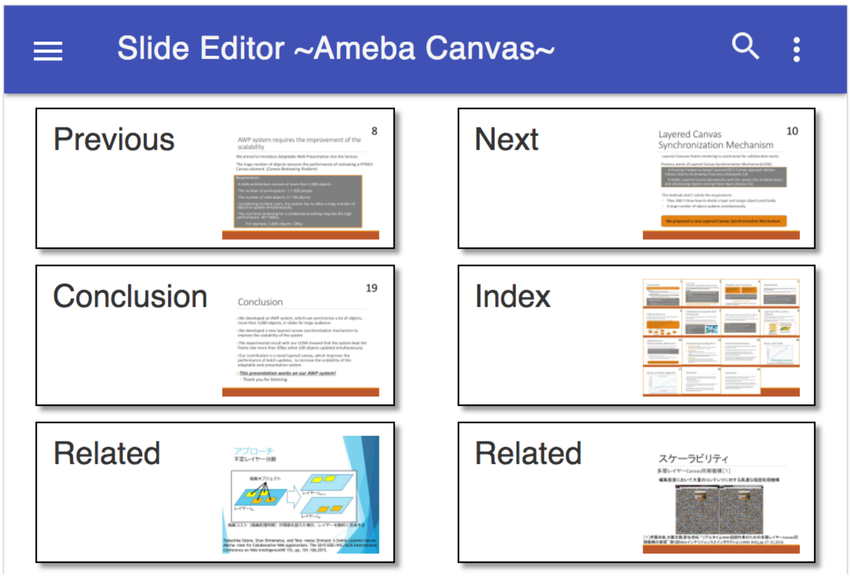

Figure 6: Controller.

When editing, it's enough time to consider which to select. Second, presenters make a presentation. During a presentation, presenters have little time to consider. It should be filter slides and narrow candidates.

The recommender mechanism need to refer to a slide architecture as evidence. We assumed the similarity of slides from slide contents. We describe the procedure. First, we extracted important phases using a feature vector constructed of term frequency and inverse document frequency(TF-IDF) metrics for each term that appears in the text. We set the top 10 words as important phases. Namely, the top 10 words as important phases as keywords as referred. We recommended slides which had the keywords. If no slide had the keywords, we generated a slide vector. In the recent study, it's popular to generate the vector with word embedded, such as word2vec[13] or pragraph2vec[14]. To calculate the vector of keywords, we used word2vec. We had generated the model from the articles of Wikipedia. We generated the vector of each word using the model. We summed vectors of the keywords and normalized it because the normalization is valid if the number of important phases are less than 10. We defined the normalized vector as a slide vector. As first step, we will design for the generation of the vector expression from only text in the slides. The next step, we will consider styles or images in a slide material.

5 Evaluation

In this section, we describe our experimental evaluation about a layered canvas synchronization mechanism and present its results. We evaluated three experiments. (I) We evaluated the memory usage of layered canvases. (II) We compared the drawing time between layered canvases and not layered. (III) We evaluated the sequential algorithm and the batch algorithm. First, we explain our experimental setup simulating the editing of objects in our editing part of the system. Second, we show the experimental results.

The experimental environment consisted of Google Chrome version 54.0.02840.71(64-bit), OS X 10.11, 2.7 GHz Intel Core i5 CPU, and 8GB 1,867 MHz DDR3 RAM.

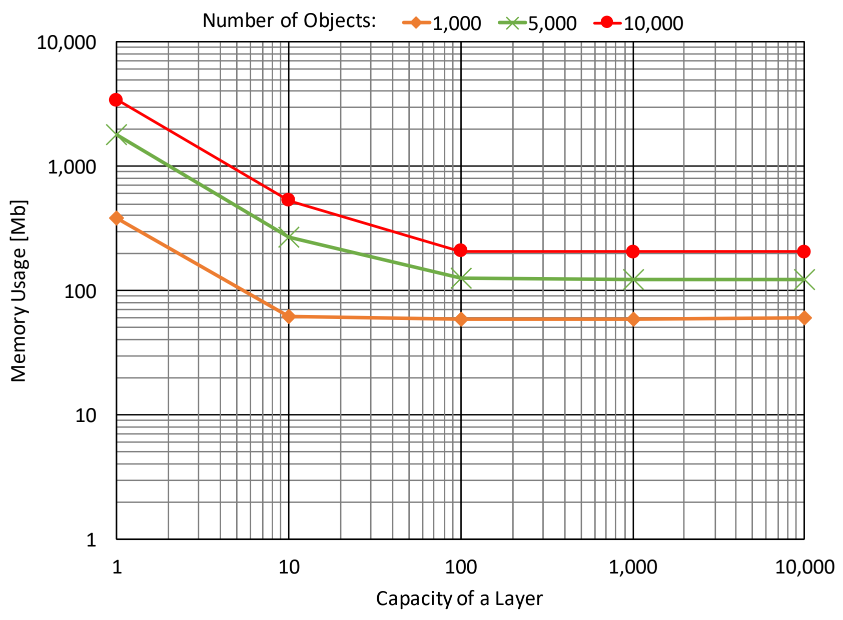

Figure 7: Memory usage.

5.1 Memory Usage

5.1.1 Experimental Setup

The number of objects on a layer correlate to a drawing time. If the only object is drawn on each layer, a drawing time would be best. However, we assumed that a lot of layers used too much memory. We evaluated the memory usage of layered canvases. We generated many objects on layered canvases and measured the number of layers and the memory at the time. We set the threshold for the capacity of layered canvases. The capacity represents how many objects a layered canvas retains. We set the capacity of a layer as following: 1 object, 10 objects, 100 objects, 1,000 objects, and 10,000 objects.

5.1.2 Experimental Results

We show the results on Fig.7. The vertical axis represents the memory usage(Mb). The horizontal axis represents the capacity of a layer. The capacity means how many objects one layer has at most. The investigation indicated that the small capacity used much memory. A new layer is the same size of an original layer.

5.2 Drawing time

5.2.1 Experimental Setup

We compared the performance of a normal HTML5 Canvas element and a layered canvas element. We investigated the edit time of an object while configuring drawing frequency. The experiments were designed to evaluate the performance of editing of the objects. A Layered Canvas Synchronization System can set the capacity of real-time rendering. We compared cases: 100 objects, 1,000 objects, and unlimited. We call these cases Layered.

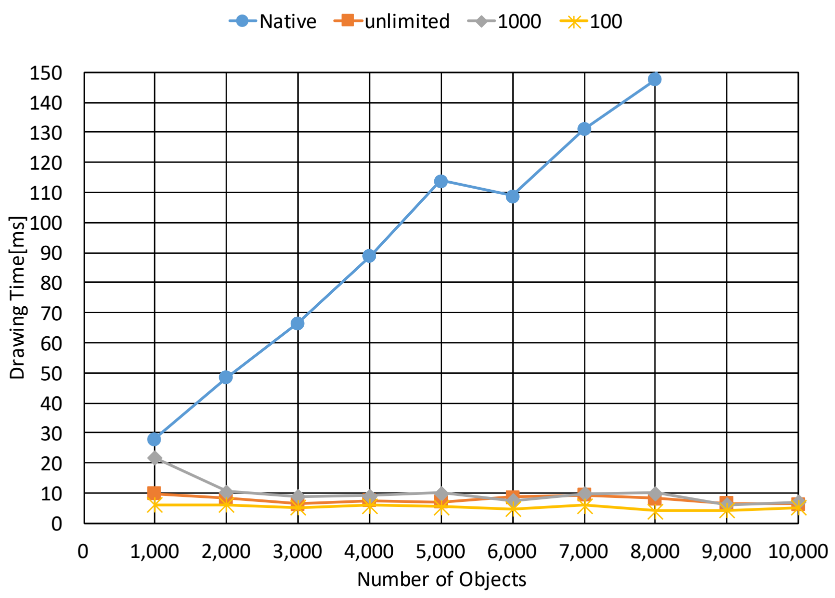

Figure 8: Uniform Condition(drawing time)

In the experiments, we generated 10,000 objects on an HTML Canvas, Layered. Then, a randomly selected object was moved to random coordinates 10,000 times and we measured how long it took to move all the objects. The selection of objects randomly used the hitting test. We determined an object which had the collision with the point. The point was determined randomly in the range of the whole canvas. If the point didn't have the collision with any objects, we selected again.

We also evaluated the population of objects in each layer in order to investigate user's behavior for selection of updated objects. For example, the upper objects have the high probability of selection. An updated object would be assigned to the divided layer when the capacity of the original layer was over the threshold of dividing. We set the two conditions of the distribution of the updating object frequency. The uniform condition simulated that the dispersion of an updated layer was large. The exponential condition simulated the localization of updated layer. The result shows the exponential

5.2.2 Experimental Results

The experimental results are shown in Fig.8 and Fig.9. Fig.8 shows the result of uniform distribution. Fig.9 shows the result of exponential distribution. The horizontal axis and vertical axis represent the number of objects and the drawing time (ms), respectively. The results show that Layered is faster than the HTML5 Canvas. We assumed that to layer a canvas reduced a redrawing cost per a layered canvas.

The result of the evaluation experiments shows that a layered canvas synchronization mechanism suppresses a drawing time, and keep the real-time rendering. A layered canvas synchronization mechanism is a generic system, thus, support a web collaborative works. For example, we show an application of editing album or documents.

Fig.10 and Fig.11 shows the population of the uniform distribution and the population of the exponential distribution. The horizontal axis and vertical axis

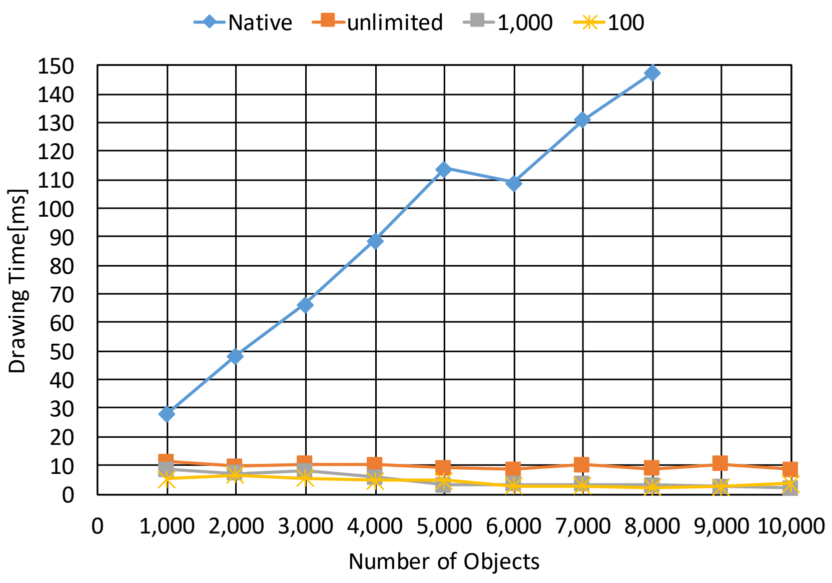

Figure 9: Exponential Condition(drawing time)

represent the sum of objects and population of objects. The result of the exponential distribution shows that updated layers were more local than the result of the uniform distribution.

5.3 Batch Algorithm

5.3.1 Experimental Setup

We proposed the batch algorithm in order to update the multi objects at the same time, and evaluated the performance. There will be tens of people and hundreds of objects for practical use of a presentation or a collaborative editing. We compared the sequential procedure that updates objects one by one. In this experiment, we generated many objects, and updated some of them at the same time. The updated objects were selected according to the hitting test. We investigated the influence on a drawing time with the number of updated objects. Under the sequential algorithm, we measured the time of finishing updating from the first to the end. We generated 100 objects at most considering the number of the participant in the group work.

5.3.2 Experimental Results

We show the results on Fig.12. The horizontal axis and vertical axis represent the number of objects at the same time and the drawing time (ms), respectively. We compared the batch algorithm and the sequential algorithm. The batch algorithm is faster than the sequential algorithm.

The result indicated that the batch algorithm enabled us to update objects at the same time. The batch algorithm targeted the 100 objects at most. It means that small number of objects should be updated altogether. Fig. 8 and Fig. 9 indicated that the large number of objects caused the canvas redrawing problem. Hence, the expected scale was adequate. Fig. 7 also indicated that a layered canvas should have 100 objects.

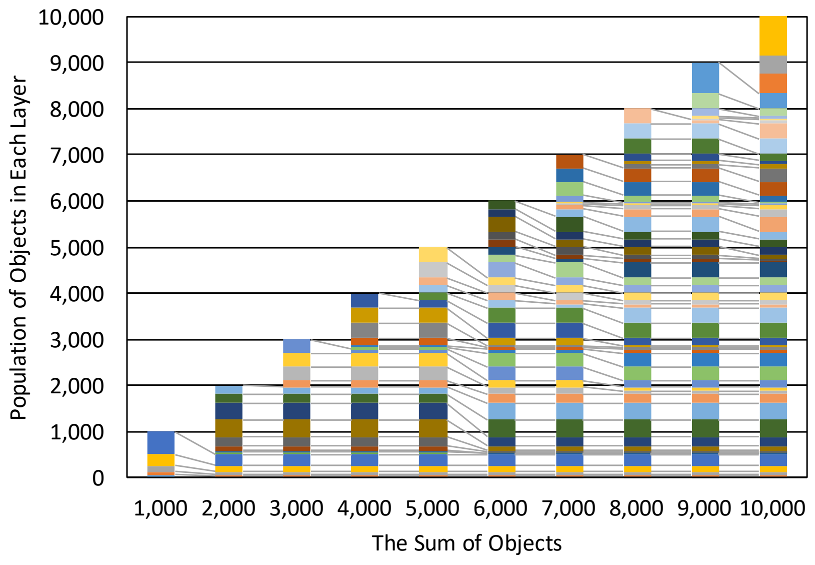

Figure 10: Uniform Condition(population).

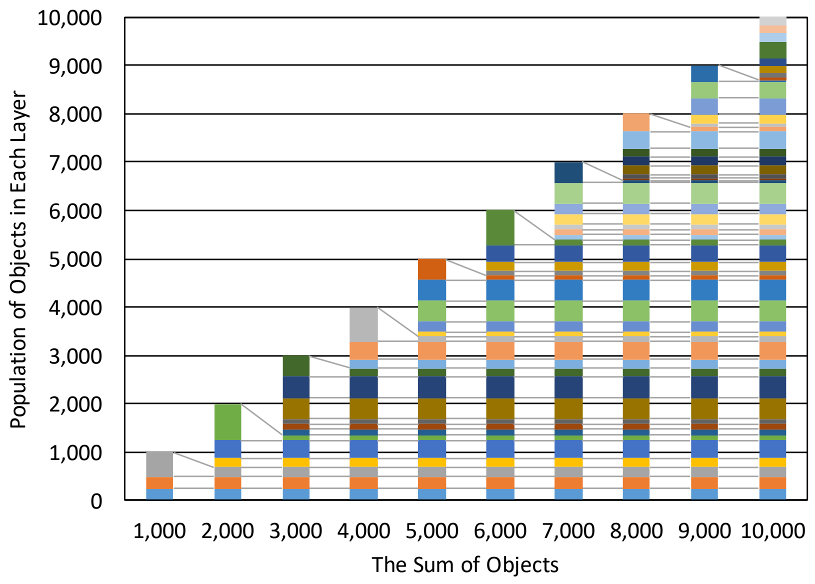

Figure 11: Exponential Condition(population).

6 Discussion

For an adaptable presentation, we expected that the number of objects on an adaptable is 10,000 objects at most. The result of (II) shows that layered canvases suppress the increase of the drawing time if 10,000 objects are rendered. That indicates that a layered canvas synchronization mechanism is effective for the canvas redrawing problem. Our contribution is that layered canvases solved the canvas redrawing problem and that the batch algorithm enabled us to update 100 objects at the same time.

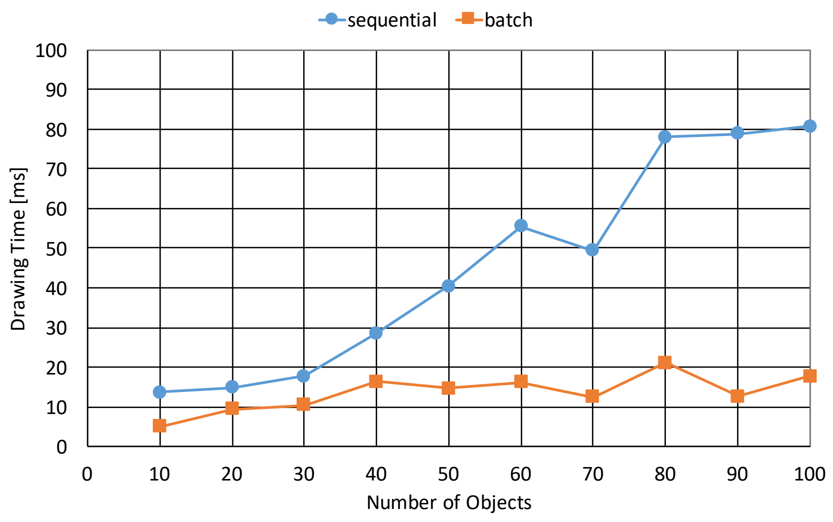

Figure 12: Simulation Results.

We proposed the batch algorithm. Following the batch algorithm, before we update some objects, we store them by a layer, and then update the layers at the same time. The batch algorithm reduces the frequency of updating layers and realize faster rendering. The worst case of the batch algorithm is 1 layer 1 object. However, it hardly occurs because of the result of the memory usage. We discuss the memory usage. If the number of updated objects were at least, the rendering performance would be best. However, the investigation of the memory usage indicates the less objects in a layer used the more memory than the case that all objects were drawn in one layer. A canvas represents a bitmap image data. When the capacity of a layer reduces, the number of layers will increase. Hence, on the one side, the cost of updating objects increases with the number of objects in each layer. However, suppress the memory usage. The result indicated that we can use 1,000 layered canvases at most, namely each layer should hold 10～100 objects. If a system uses more than 1.0 GB, it seems not enough performance for the real-time rendering. Considering the image data size, in practical use case the application used much more memory. We concluded the capacity of objects in a layer is about 100 at least. The optimum cost can’t be defined because the capacity depends on an application. If an application usually uses large image objects, a system should save the capacity.

We assessed the object population in each layer for the selection bias. If the object selection has the selection bias, we can reduce the frequency of the dynamic layering. Dy-namic layering cost much time. It would be danger of the bottle neck. Therefore, we tried to reduce the frequency of the dynamic layering. That is if we know the frequency of all objects in the future, we should assign objects to layers in order to the frequency. The low frequency of objects should be assigned to layers that has a large capacity. The high frequency should be assigned to layers that has a small capacity. The result shows the dynamic layering worked in a few upper layers under the exponential condition. Considering the selection bias supports the batch algorithm. If the updated layers are limited, the batch algorithm costs little.

7 Conclusion

For an adaptable presentation, we expected that the number of objects on an adaptable presentation is 10,000 objects at most. The results show that layered canvases suppress the increase of the drawing time if 10,000 objects are rendered. That indicates that a layered canvas synchronization mechanism is effective for the canvas redrawing problem. Our contribution is that layered canvases solved the canvas redrawing problem and that the batch algorithm enabled us to update 100 objects at the same time.

Acknowledgments

This work was supported in part by JSPS KAKENHI Grant Number JP15K00422, JP16K00420.

References

[1] Eishun Ito, Ozono Tadachika, Toramatsu Shintani, "Adaptable Web Presentation System with Layered Canvas Synchronization Mechanism for Scalability," 8th International Conference on E-Service and Knowledge Management, pp.1–6, 2017.

[2] Shinya Katayama, Takushi Goda, Shun Shiramatsu, Tadachika Ozono, Toramatsu Shintani, "On a Drawing-Frequency based Layered Canvas Mechanism for Collaborative Paper Editing Support Systems," International Journal of Networked and Distributed Computing(IJNDC), Vol.2, No.2, pp.91–99, 2014.

[3] Shinya Katayama, Shun Shiramatsu, Tadachika Ozono, Toramatsu Shintani, "Generational Layered Canvas Mechanism for Collaborative Web Applications," IIAI 3rd International Conference on Advanced Applied Informatics(IIAI-AAI 2014), pp.70–75, 2014.

[4] Tadachika Ozono, Shun Shiramatsu, Toramatsu Shintani, "A Stable Layered Canvas Mechanism for Collaborative Web Applications," The 2015 IEEE/WIC/ACM International Conference on Web Intelligence(WI'15), pp.101–106, 2015.

[5] Ryota Inoue, Shun Shiramatsu, Tadachika Ozono, Toramatsu Shintani, " Visualizing Real-Time Questionnaire Results to Promote Participation in Interactive Presentations, " Proceedings of 5th International Conference on E-Service and Knowledge Management, pp-64–69, 2014.

[6] V. Triglianos and C. Pautasso, "ASQ: Interactive Web Presentations for Hybrid MOOCs," Proceedings of the 22nd international conference on World Wide Web companion, pp.209–210 2013.

[7] Ali Khalili, Soren Auer, Darya Tarasowa and Ivan Ermilov, "SlideWiki: Elicitation and Sharing of Corporate Knowledge Using Presentations, " Knowledge Engineering and Knowledge Management Lecture Notes in Computer Science, val.7603, pp.302–316, 2012.

[8] Hiroyuki Yamada, Yusuke Niwa, Shun Shiramatsu, Tadachika Ozono, Toramatsu Shintani, "A Reactive Presentation Support System based on a Slide Object Manipulation

Method," Proceedings of the 2014 International Conference on Computational Science and Computational Intelligence(CSCI 2014), Vol.2, pp.46–51, 2014.

[9] Ryan Spicer, Yu-Ru Lin, Aisling Kelliher, Hari Sundaram, "NextSlidePlease: Authoring and delivering agile multimedia presentations," ACM Transactions on Multimedia Computing, Communications, and Applications(TOMM), Vol.8, Issue.4, 2012.

[10] Darren Edge, Joan M. Savage, Yatani Koji, "HyperSlides: Dynamic Presentation Prototyping," Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp.671–680. 2013

[11] Tomoya Kobayashi, Kazushi Nishimoto, "Chatplexer: Supporting Extraction of Important Opinions for a Presenter in an Oral Presentation where a Text-chat is Concurrently Used," IPSJ Journal, Vol.53, pp.12–21, 2012.

[12] Yusuke Niwa, Shun Shiramatsu, Tadachika Ozono, Toramatsu Shintani, "A Collaborative Web Presentation Support System Using an Existing Presentation Software," Proceedings of the 2014 IIAI 3rd International Conference on Advanced Applied Informatics(IIAI-AAI 2014), pp.80–85, 2014.

[13] Tomas Mikolov, Scott Wen-tau Yih, Geoffrey Zweig, "Linguistic Regularities in Continuous Space Word Representations," Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies(NAACL-HLT-2013), pp.746–751, 2013.

[14] Quoc V. Le, Tomas Mikolov, "Distributed Representations of Sentences and Documents," Proceedings of the 31st International Conference on Machine Learning(ICML-14), pp.1188–1196, 2014.