Assessing Lag Perception in Electronic Sketching

Ugo Braga Sangiorgi, Vivian Genaro Motti, François Beuvens, Jean Vanderdonckt
Louvain Interaction Laboratory, Université catholique de Louvain
Pl. Place des Doyens, 1 - B-1348 Louvain-la-Neuve (Belgium)
{ugo.sangiorgi, vivian.genaromotti, francois.beuvens, jean.vanderdonckt}@uclouvain.be

ABSTRACT
Electronic sketching has received a recurrence of interest over the years and again nowadays within the mobile web context, where there are diverse devices, operating systems and browsers to be considered. Multi-platform (e.g. web-based) sketching systems can be constructed to allow users to sketch on their device of preference. However, web applications do not always perform equally on all devices, and this is a critical issue, especially for applications that require instant visual feedback such as sketch-based systems. This paper describes a user study conducted to identify the most appropriate response rates (expressed in frames per second) for end users while sketching. The results are expected to guide stakeholders in defining response parameters for sketching applications on the web by showing intervals that are accepted, tolerated, and rejected by end users.

Author Keywords
Electronic Sketching; Latency; Lag perception; HTML5.

ACM Classification Keywords
H.5.2 User Interfaces: Benchmarking, Input devices and strategies.

INTRODUCTION
Sketching is an important – perhaps necessary – tool for design and it is usually performed with pencil and paper. For over 45 years since the first sketch-based computer systems were proposed [3, 2] there has been recurring interest in supporting sketching with computation.

Electronic sketching has some important disadvantages when compared with the classic pencil and paper approach. The most basic is *latency* – the visual feedback of what is being sketched is not immediate, compared to paper, as the drawing "lags" behind the pen.

Latency is an issue composed of both the device – related to refresh rate of the display; and the system – developers need to address how often to refresh the screen. One could argue the system should refresh as fast as possible, which would unavoidably increase the consumption of resources such as battery. Besides that, the increasing availability of low-end smartphones [10] makes performance a critical issue for applications designed to run on mobile devices.

Current research on electronic sketching needs to consider the fact that nowadays many devices are available for users to sketch upon [7]. Moreover, in the future, the difference between device's screen and paper is likely to be reduced (i.e with flexible paper-like displays [16]). We argue that in order to start an investigation of electronic sketching in that diverse context, an acceptable **lower limit** for sketching rendering needs to be properly investigated.

There are many differences in visual rendering according to the device type. Figure 1 illustrates a comparison of HTML5 rendering performance among different platforms according to the amount of strokes (ranging from 0 to 1000). The applications implemented to run in these different devices should be able to adapt their behavior, in order to avoid significant performance decays and to provide a better user experience. However, there are no guidelines and standards that can be used by developers to implement this kind of adaptation.

We have designed and conducted an experiment with 35 users to assess their perception of different response rates measured in frames per second. The users were asked to draw and rate the speed in which the sketch was displayed while being drawn. The results show intervals that are accepted and rejected by the users in terms of FPS. Throughout the paper, we will refer to FPS as the amount of *frames per second* a display is capable of exhibiting. Usually in technical specifications one will find that a device's refresh rate is expressed in terms of Hz (e.g. a screen monitor with a refresh rate of 60 Hz is capable of redrawing the screen at a rate of 60 FPS).

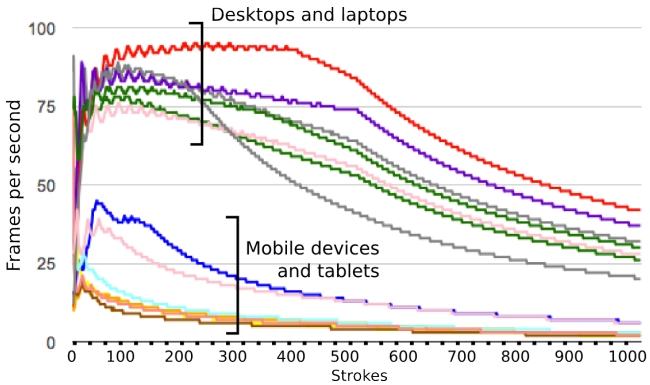

Figure 1. Benchmark performed on different devices for an HTML5 application that measures the refresh rate for 1000 strokes.

NordiCHI '12, October 14-17, 2012 Copenhagen, Denmark

This paper is organized as follows: in the next section we discuss the motivation for electronic sketching, contextualizing the latency problem. Section 3 presents the benchmark constructed to compare desktop and mobile devices in terms of HTML5 performance. Section 4 describes the methodology used on the experiment. In Section 5 we present the results followed by a discussion and then we conclude with final remarks.

MOTIVATION

Despite sketching recognition field being fairly well addressed, electronic sketching as a tool has still a long way ahead [4]. The goal of the study reported in this paper is to further develop the research on sketch-based interaction. From a software development point of view, we expect to provide an acceptable range of screen refreshing rates for developers who want to build sketching applications. Therefore, developers can be sure about how fast the sketches need to be rendered, which might significantly improve battery life, for instance, of mobile and tablet devices.

The phenomenon behind human frame rate perception is called *Flicker fusion* [18], i.e. the point in which a flickering image appears as stationary. This concept relies on different criteria such as stimulus luminance, size, position, color, etc. Early silent film projections had frame rates varying from 16 to 24 FPS, and nowadays the standard rate for movies is at least 24 FPS.

Within the research domain of human vision, there are works addressing a possible frame rate limit that human beings are able to perceive. It is estimated that the human visual system can perceive from 10 to 12 frames per second individually, above this limit the illusion of motion would take place, and humans would not be able to perceive individual frames [11, 17, 19].

However, the considered limit applies when no interaction is required. This work is interested in determining whether this limit can also be applied for the specific case of sketching, which, to our knowledge, no study has already targeted.

We refer to sketch as described in [4]: quickly made depictions that facilitate visual thinking, which may include everything from abstract doodles to a roughly drawn interface. Sketching is a ubiquitous activity among humans, and people engage in a sort of 'conversation' with their sketches in a tight cycle of drawing, understanding, and interpreting [15].

As Van der Lugt [20] identified, sketching serves three primary functions. It stimulates a re-interpretive cycle in the individual designer's idea generation process; it provotes the designers to re-interpret each other's ideas and finally it stimulates the use of earlier ideas by enhancing accessibility. Weiser in [21] argues that rather than being a tool through which we work, by disappearing from our awareness, the computer too often remains the focus of attention.

Therefore, electronic sketching, primarily as a design activity, should be as close as possible to the paper-based interaction in terms of performance, otherwise the system underneath becomes too evident, hindering the designer's 'conversation'.

Electronic sketching can be depicted as Figure 2 illustrates. While components 1 and 2 may widely vary across the many existing devices currently available, and they are likely to achieve a lower latency of both input and output in the coming decades, the software feedback on 3 still needs to be addressed, since it is ultimately a developer's decision.

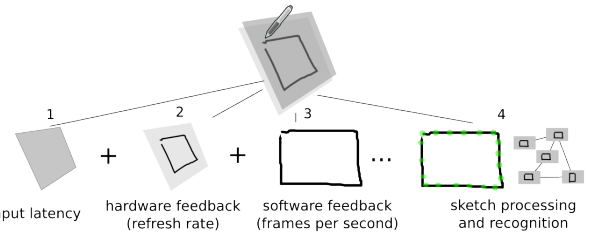

Figure 2. Depiction of electronic sketching. Perceived latency is composed of 1, 2 and 3.

Systems that use electronic sketching span a wide range of domains, from software engineering [14] and user interface design to architecture and engineering [9] or visual thinking and general brainstorming, as in collaborative workplaces [8].

Due to the fact that sketching on browsers using HTML and Javascript is not essentially new, we believe that an efficient multi-platform sketching interface would benefit both developers – for whom it would ideally minimize the code fragmentation across platforms; and end-users – who would be able to perform sketching activities on their preferred device.

However, nowadays developers cannot find accepted standards that guide the implementation of sketch-based applications. Such guidance is relevant to assuring the adaptation of systems on multiple platforms, according to the context of use and user profile.

Adjusting the visual feedback response according to the context of use, e.g. to the users' profiles, their impairments, and device constraints, has many advantages, among which we can highlight: an optimized resource usage, concerning, for instance, battery level, processing capabilities, and internet connection bandwidth, and a higher usability level, concerning the user satisfaction.

Although some works have approached human perception of lag in various domains [1, 5, 6, 13, 17], we have not found references in the literature reporting performance tests with users in design activities such as sketching.

In order to be able to identify optimal FPS rates, user studies are necessary. Defining adaptation parameters and associating them with context information is not an easy task, mainly because several influencing factors are involved. We intend, however, to find patterns in the perception of users regarding sketching activities, and associate these patterns with some characteristics of the user profile, such as expertise level, domain of experience, and interaction speed. Once the users' perception is known, we can establish parameters that support the dynamic adaptation of the behavior of applications, and consequently provide users with a better interaction.

ASSESSING HTML5 PERFORMANCE FOR SKETCHING

The preliminary phase of this experiment consisted in assessing and comparing the rendering performance on different devices. In order to reduce a potential bias in the results' interpretation (mainly caused by intrinsec differences of each device type), we planned and conducted a benchmark, by constructing a simple webpage using HTML5's *canvas* element and Javascript.

The test consisted in displaying a 640x480 pixel canvas in which 1000 lines (or strokes) were sequentially drawn in random directions. The canvas was cleared and then the same process was repeated two more times, in order to calculate an average of the three rounds.

Device	Browser
MacBook Pro MacOS	Firefox, Chrome
MacBook Pro Windows	Internet Explorer, Firefox, Chrome
IPad OSX	Safari, iChromy
Samsung Galaxy Tab 10.1 Android	Firefox, Dolphin
Android G1	Gecko
IPhone	Safari, iChromy
TabletPC Windows 7	Firefox, Internet Explorer

Table 1. Devices and browsers used for the benchmark

We performed the test on the devices listed in Table 1, however the goal was not to compare operating systems and browsers, but device types. The specific values obtained by each device are not to be considered, but the overall FPS rates achieved by each type of device reveals the rendering differences across devices.

This test allowed us to perceive firstly that the HTML5 rendering performs differently on each device type (mobile and desktop), probably because of both hardware and software differences, and secondly that the performance on mobile devices decays much faster than on desktops.

For instance, Figure 1 shows that more powerful devices, such as laptops, are in the range of 80 FPS, while tablets, such as an iPad and an Android one, are in the range of 20 FPS. This represents a significant difference of performance, and developers of multi-platform sketching applications would benefit from the results about users perception of speed in order to adapt their applications.

Application

In order to perform the experiment we developed a web application with Javascript to capture input events such as mouse/pen movements and clicks and an HTML5 *canvas* element that allows the drawing of graphics.

Although Javascript does not currently support true multithreading, it is possible to make separated functions to run concurrently on a time-slice fashion, using the *setInterval* function. In this way, one can specify in terms of miliseconds how often a given function should repeat its execution. A simplified function for drawing a sketch based on a set of points is presented below.

```
function draw(){
    clearScreen();
    for(int i=1;i<points.length;i++){
      drawLine(canvas.context,
      points[i-1],
      points[i]);
    }
    frames++;
}
setInterval('draw()', 33);
```

Figure 3. An example of function for drawing a sketch based on a set of points, at a specific refresh rate.

In this way, we were able to set a target FPS rate and refresh the screen accordingly (e.g. a rate of 30 FPS would require the screen to be updated each 1000/30 = 33 milliseconds). We implemented a parallel function to simply count the frames being displayed each second, measuring the actual FPS rate.

For the benchmarking application we used the same system, with the addition of a third function to automatically draw 1000 lines in random directions onto a canvas of 640x480 pixels, repeating it three times to calculate the average at the end. We set the target refresh rate to 100 FPS (therefore refreshing the screen every 10 miliseconds) and we measured the actual rate on different devices using the aforementioned parallel function. The number of applications running on the device was kept to a minimum in order to reduce interference on the benchmark.

EXPERIMENT

We designed the experiment to observe the following aspects regarding user's perception:

1. Due to other processes running on a system, the rendering might not always be the same. Therefore, are users able to identify the difference when the refresh rate changes? How does the difference needs to be in order to be perceived?
2. What is the practical limit of FPS rate at which the users start to perceive the system as 'too slow'?

In order to minimize the variance of speed and make the subjects focus on the response time rather than on the drawing itself, we asked them to draw squares, given that squares do not require too much effort to be drawn (in terms of skill and time), and also they permit subjects to evaluate the rendering response in four different directions.

We will refer to *rate* as the FPS rate in which the application performed and to *grade* as the subject's evaluation for a given rate.

Subjects and physical setup

35 subjects from different domains, such as Economics, Biology, Psychology and Computer Science, performed the experiment, 16 being women, and 19 being men. The average age was 28 years old with standard deviation of 7.5. 14 users reported that they had never used pen-based interaction. 51%

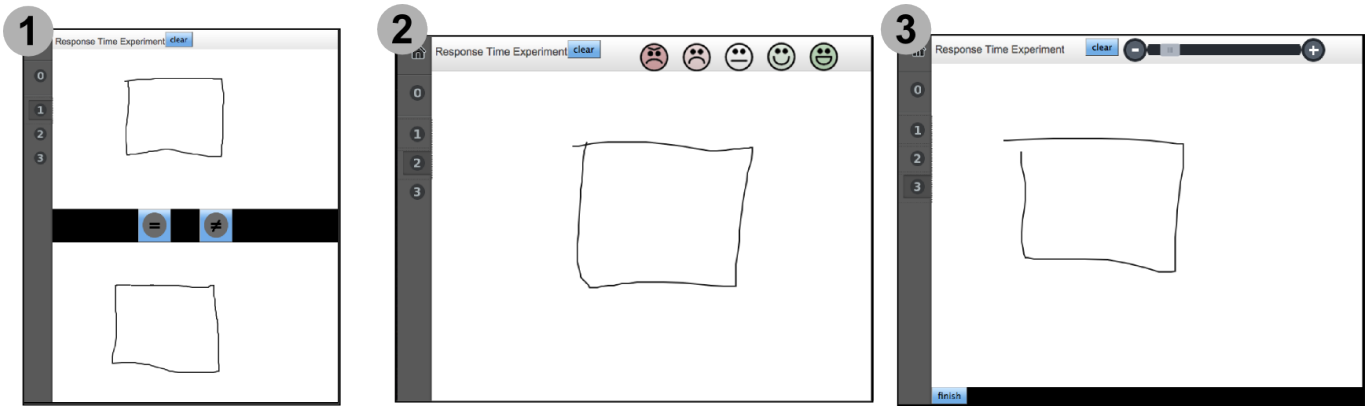

Figure 4. Three phases of the experiment: (1) A vs. B comparison of FPS rates; (2) FPS rating with Likert scale; and (3) Subjective selection of lowest FPS limit.

of the users (18) reported having a computer, a laptop and a feature phone, 6 users out of 35 have a smartphone, and only 1 user reported also having a tablet.

We recruited subjects throughout the campus in different departments. The relative simplicity of the experiment ease the process of recruitment, since any person was eligible to participate, and we remarked that no drawing skills were necessary. All the subjects were rewarded with a simple gratification (e.g. a chocolate bar, a cup of coffee or a cookie) after concluding the experiment.

The device used was a 12-inch Wacom Cintiq tablet with a resolution of 1280x960 and a refresh rate of 75 HZ (therefore with a maximum of 75 FPS), connected to a MacBook Pro with an Intel processor running Firefox 7. The stylus used was a standard Intuos4 Grip Pen with a standard plastic nib.

We conducted 4 rounds of pilot experiments to adjust the experiment parameters, such as time, ordering and tasks to be performed by the subjects. The total experiment time was adjusted to 15 minutes.

The experiment was preceded by a warm-up step in which the subject could freely draw and change the FPS rate (varying from 1 to 80) using the slider bar located on the top of the window. This step was essential since not all subjects were familiar with tablet devices and pen-based interaction. Besides, it was important to assure that all subjects noticed what an FPS rate is and how different speeds are rendered.

First phase - FPS Pair Comparison

The first step aimed at verifying whether the subjects could differentiate the visual feedback response considering intervals of 2, 5 and 10 FPS in a scale ranging from 2 to 80. Subjects were asked to draw one square in the upper canvas and one in the lower canvas (Figure 4.1) and then to classify the response rate as equal or different. Subjects were told to clear the screen and to repeat the drawings as much as they wanted, in case of uncertainty.

The sampling used for this phase consisted of the pairs depicted in Table 2, tested in a randomized way:

The standard explanation used was: "*The system will display your drawing in different pairs of speed and then request you to classify them. Some of them will be equal, some of them will be a lot or a bit different*".

The initial hypothesis was that the perception would vary according to how large the range was, e.g. differences of 2 FPS would be less perceived than the ones of 5 or 10; and also according to how high the FPS range was located, i.e. for values that are higher than a specific FPS, the users would not clearly perceive differences, independent of how large they were.

Second phase - FPS rating with Likert scale

The second step of the experiment consisted in asking users to classify on a five-point Likert scale different FPS rates. Icons associated with the semantic differential of *really bad*, *bad*, *neutral*, *good* and *really good*, represented the scores. In case of uncertainty, users could clear the canvas and repeat the drawing as many times as necessary.

The sampling for this step was a double set from 2 to 50 FPS, with increments of 2, in a random order. The goal was to make each subject test the same FPS value twice and then calculate the average. The subjects did not know which value of FPS was being used at each time, so they did not have any immediate reference of "good" or "bad" FPS rates.

This step aims at associating the FPS values with the subjects' perception, we were expecting to obtain values ranging proportionally to the FPS used (for instance, very low FPS rates such as 2, 4, 8 would be rated as *bad* and *very bad* more

FPS range	Pairs used
2 FPS	2vs4 4vs6 6vs8 8vs10 10vs12 12vs14 14vs16 16vs18 18vs20 20vs22 22vs24 24vs26 26vs28 28vs30 30vs32 32vs34 34vs36 36vs38 38vs40
5 FPS	5vs10 10vs15 15vs20 20vs25 25vs30 30vs35 35vs40 40vs45 45vs50 50vs55 55vs60 60vs65 65vs70 70vs75 75vs80
10 FPS	10vs20 20vs30 30vs40 40vs50 50vs60 60vs70 70vs80

Table 2. Devices and browsers used on the benchmark

frequently), as an indication that the different FPS were perceived as intended.

In this step we were especially interested in rates that received the scores *neutral*, *good* and *very good*, since values below these would indicate rejection by the subjects and would therefore represent a minimum FPS to be provided by applications intended to do sketching. The values below this range should then be considered as rejected by users.

Third phase - Active selection of lower limit

For the third step of the experiment, the subjects were asked to select the lowest value of FPS that they considered as acceptable. The standard explanation used was: "*Suppose that you have a device that is slow to draw, what would be the lowest acceptable response speed in your opinion?*".

A slider bar located on the top part of the screen enabled users to vary and set different FPS rates to test them. Once the subject decided the lowest acceptable value, he or she could submit the FPS rate and conclude then the experiment.

The goal of this step was to achieve a minimum threshold that is accepted by users in general and to cross-validate the results of the previous phase.

RESULTS

Concerning the frequency of drawing (in general) the majority of the users (80%) informed that they never (or almost never) draw, 8% (3) informed that they do it sometimes, and 11% (4) informed to draw often.

In order to ease the analysis of the results and to search for a correlation, the subjects were separated into two groups (A and B) based on their sketching speed measured in pixels per second (pps). Figure 5 illustrates the speeds of the two groups above and below the median value of 135 pps.

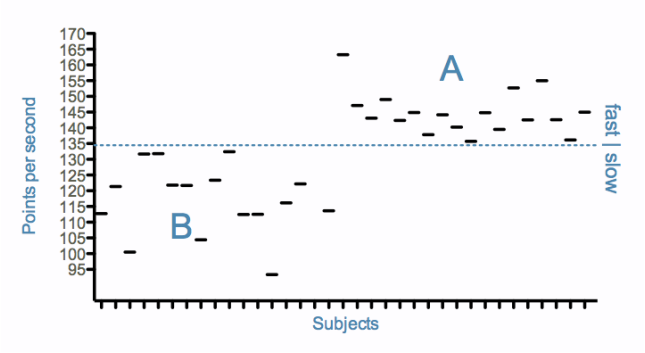

Figure 5. Sketching speeds of subjects.

With the separation between groups A and B, a two-way ANOVA could be applied, resulting in a very significant variation between the two groups ($P<0{,}0001$, $F=49.27$, $DFn=1$, $DFd=17$).

First phase results

The semantic values of *EQUAL* and *DIFFERENT* were the grades associated to each button. We summed the number of *DIFFERENT* grades given for each pair of FPS to analyse the subjects perception. Figure 6 illustrates a comparison between the sum of *DIFFERENT* values for 2FPS, 5FPS and 10FPS ranges, also with a separation of groups A and B.

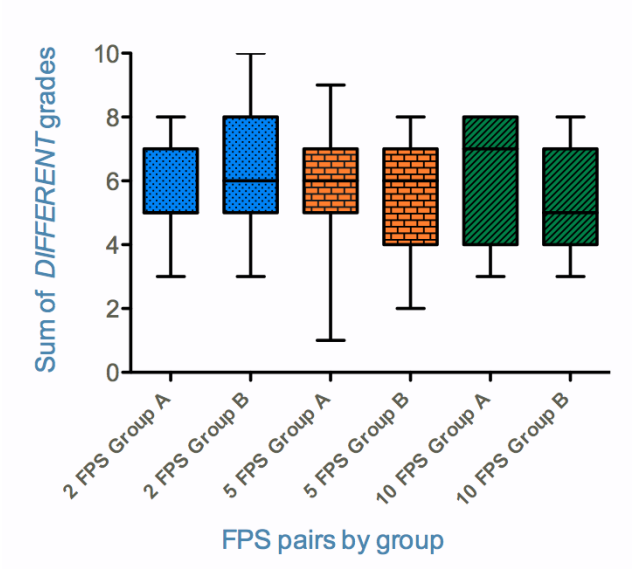

Figure 6. Comparison between the amount of 'different' grades given by subjects in ranges of 2, 5 and 10 FPS.

In the first phase we were expecting subjects to perceive differences according to how large the range of FPS pair was (i.e. pairs that differ by 2 FPS would receive more *EQUAL* grades than pairs of 5 or 10 FPS).

However, neither the one-way ANOVA nor a Bonferroni's Multiple Comparison analyses showed any significant variance between the ratings for the sum of *DIFFERENT* grades on 2, 5 or 10 FPS groups ($p=0.6383$; $F=0.4543$; $R=0.0233$). Besides this, there was no significant difference between grades given by groups A and B ($p=0.7967$; $F=0.6584$; $R=0.04152$).

Furthermore, we were expecting to obtain more significant results regarding the higher and lower ranges of FPS within the same FPS group. For instance, we expected to see more FALSE grades in lower pairs (8vs10, 15vs20, etc) than in higher pairs (42vs45, 45vs50, etc).

When splitting the measured pairs into low and high FPS ranges, the two-way ANOVA analysis showed no variation between ratings of low and high FPS with differences of 2 FPS or 5 FPS ($p=0.564$ for pairs of 2 FPS, $p=0.2746$ for pairs of 5 FPS). In other words, we cannot confidently state that differences of performance of 2 and 5 FPS are perceived. However, the analysis for 10 FPS showed a significant difference between grades for low and high FPS pairs ($p=0.0136$).

Summary of first phase

More data would have to be gathered to confidently state that subjects perceived differences of 2,5 and 10 FPS. Therefore, the first phase of the experiment did not produce sound results. One possible explanation is that, due to the short time available for the experiment, the window chosen had to be

narrowed (more pairs needed to be tested) in addition to the single-testing of pairs – each subject tested each pair of FPS just once, and not twice like in the second phase.

Second phase results

For the second phase of the experiment we obtained the results as presented in Figure 7. The graph shows the average scores (1 to 5 respectively represent the grades of *very bad*, *bad*, *neutral*, *good* and *very good*) on the different FPS rates from 2 to 50. The graph shows a consistent result – subjects provided low grades where low FPS rates were used and high grades where high FPS rates were used.

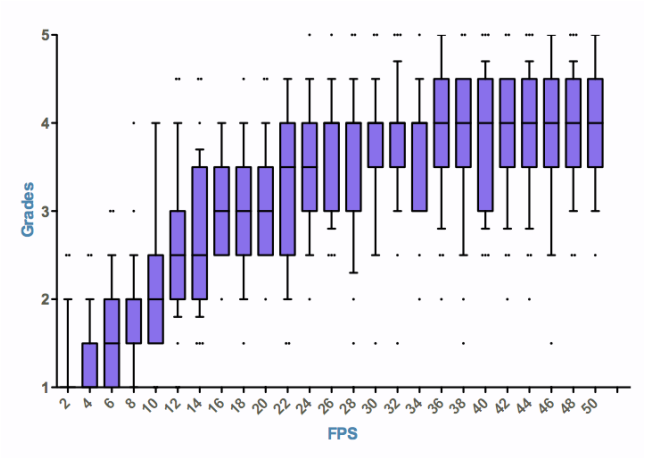

Figure 7. Results for the second phase of the experiment: average grades from from 1 to 5. The dots represent the 10% outliers.

However, the experiment was designed to outline a range of rejected and accepted values. We were especially interested in the turning point, i.e. for which range of FPS the subjects graded *bad* and *very bad*, which represents the range of rejection.

We isolated and summed up the results of *neutral*, *good* and *really good* (respectively 3, 4 and 5 on the Likert scale). In Figure 8 we summed, for each FPS value, the amount of grades higher than the average of 2.5. In this way, we obtained a graph that is consistent with the previous one, revealing a pattern with FPS rates higher than around 24.

After performing a linear regression analysis for both ranges: from 2 to 24 and from 24 to 50, we obtained the values as presented in Figure 8. There is a high variance in FPS rates in the range of 2 to 24 ($p<0.001$, slope deviates significantly from zero), but there is a significant stabilization on the range above 24 FPS ($p=0.1569$, slope does not deviate significantly from zero).

Another interesting observation is related to how groups A and B graded the FPS values. The group of subjects who drew faster (group A) tended to give lower rates than the group of subjects who drew slower (group B) as Figure 9 illustrates. A two-way ANOVA analysis demonstrated that this difference is highly relevant ($p<0,0001$; $F=56.58$; $DFn=1$; $DFd=24$).

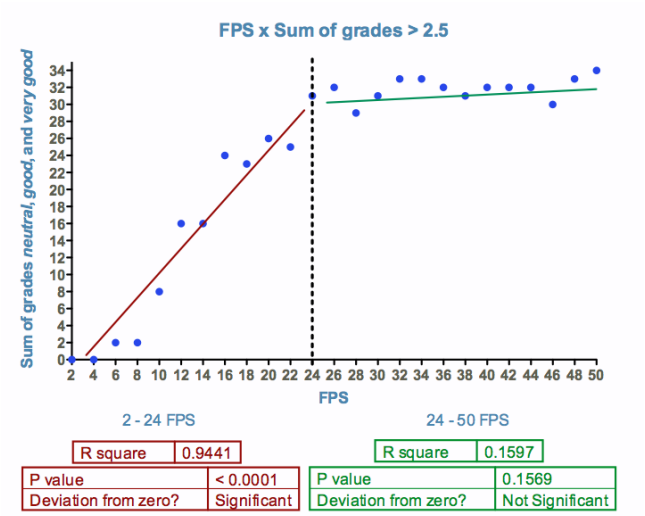

Figure 8. Analysis of grades higher than 2.5. While from 2 to 24 FPS there is a high variance of grades, within the range of 24 to 50 the grades tend to stabilize.

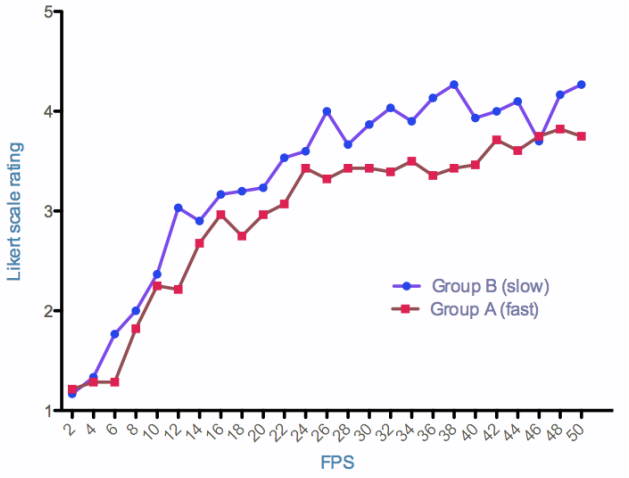

Figure 9. Average ratings given by the two groups A and B.

Summary of second phase

The second phase outlined two distinct observations: one about the lowest FPS limit for sketching and another about the relation between sketching speed and speed perception.

The result of the second phase shows that subjects graded the FPS rates above 24 as *neutral*, *good* and *really good*, more consistently than the grades below 24. This means that the minimum acceptable rate is 24 FPS for that phase of the experiment, posing an important limit to consider. In other words, developers would have to carefully consider this range in their sketching applications, allowing their applications to adapt when/if the refresh rate of the application becomes slower than this limit.

The second observation is about users' profiles, since subjects in group A assigned more grades *bad* and *really bad* than group B. This is another factor to be considered by developers

– by simply calculating the speed at which the user sketches it would be possible to relax or not the rejection limit rate.

Third phase results

For the third phase of the experiment, we grouped the results of the minimum FPS acceptable to subjects for each of the two groups. Figure 10 shows a range from 20 to 39 FPS with an average of 32 FPS as the minimum acceptable rate.

Subjects in group A graded higher than the ones of group B. This is consistent with the results from the previous phase, since subjects that drew in a faster pace seemed to be less satisfied with slow FPS rates.

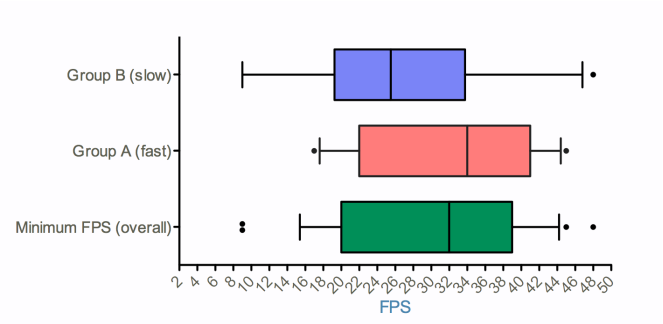

Figure 10. Grouped results of the subjective choice of a minimum FPS rate, showing an average of 32 FPS.

Summary of the third phase

The difference between the second and third phases is that in the latter subjects did not know which FPS rate was being used, while in the former subjects could actively choose higher and lower FPS rates.

It is noticeable that most of the range of the actively chosen values is above the 24 FPS limit assessed at the second phase.

Discussion

Lag has been already acknowleged as a degrading factor of human performance in motor-sensory tasks on interactive systems [6]. We started to assess the lag perception while sketching electronically in order to provide acceptable guidelines for developers. With this experiment, we are aiming at providing objective (rather than subjective) means of answering the question "is this device suitable for sketching?".

The experiment was made using HTML5, but despite its focus on a web technology, we believe the results presented in this paper to scale onto electronic sketching in general, regardless of technology.

Experiment limitations

Some remarks can be made about the experiment. One threat to establishing validity that we recognize is that we asked subjects to draw only squares. However, we chose to reduce the experiment's scope and make the subjects to focus solely on the rendering speed.

Another remark is about the pen nib – a standard plastic one was used, while there are other kinds such as rubber, felt and spring nibs. The friction between the pen and the tablet screen is very different between different nibs. In the same way, finger interaction is something relevant to be considered both for the friction and activity type. Therefore other experiments will be done for sake of comparison.

The benchmark application enabled us to perceive firstly that the HTML5 rendering performs in different ways on mobile and on desktops and that the performance on mobile devices decays much faster than on desktops. However, the arbitrary amount of 1000 strokes chosen for the test has to be refined for different domains. For instance, a graphic artist can make more than 1000 strokes on a drawing while an interface designer might need a lot less.

Overall Results

About the results, when observing the results for phases 2 and 3, there is a clear distinction between grades given by groups A and B. We believe this phenomenon to be directly related to perception of speed in the two groups. Subjects that drew at a faster pace tended to be more sensitive to latency since the system was not able to respond at the speed demanded by them; for subjects on the other group, the system's response was considered as adequate with a relative lower FPS rate.

We expect the difference between A and B to be even larger for groups of designers. For the current experiment we recruited subjects from very diverse backgrounds, and mostly from outside of the design community (since 80% informed that they never or almost never draw in their work practices).

Despite the first phase's results not being conclusive, the results of the second and third phases are meaningful: the range of minimum acceptable values in phase 3 matches the range in which the scores start to stabilize in phase 2 (around 24 FPS). Therefore, it was possible to outline a range of rejection and acceptance by combining the results of the second and third steps.

An important observation can be made by joining the results of the experiments and the benchmark presented on the first section (Figure 11). It allows us to see that most mobile devices and tablets would not be suitable for sketching geometrical forms in HTML5 applications, since all of the tested devices are in the range of rejection after about 300 strokes.

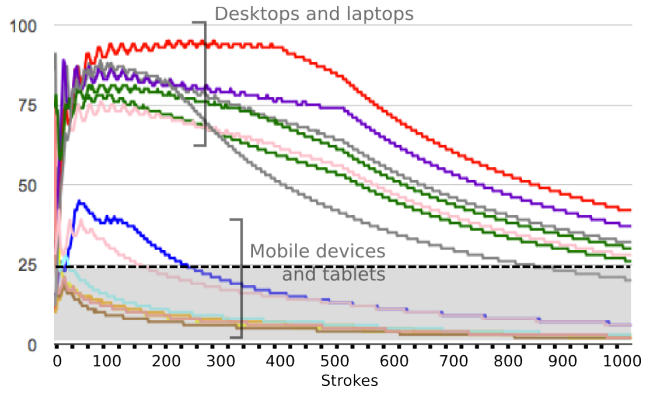

Figure 11. Grouped results of the benchmark and assessed rejection range (highlighted).

Within interactive applications the assessed rejection range

can be considered in the possibility of a performance drop during its use, or as a deciding factor for discarding a device type, brand or operating system, in case the device cannot overcome the limit.

There are other factors that contribute to degrading performance during usage of interactive applications, and it is valid to note that the benchmark was conducted under controlled conditions: no other activities were being performed at the same time, and the number of applications running on the device was kept to a minimum.

Therefore, despite *Desktops and laptops* being far from the rejection range on the benchmark, it is not safe to assume that they will never reach the rejection range under real usage situations. That means that the results described in this paper are applicable not only to *Mobile devices and tablets*, therefore developers would have to address the problem whenever their applications reach the rejection range, by warning the user and offering alternatives when possible.

Application on Multi-platform Sketching

The experiment was made using a HTML5 application we constructed as the first step for a larger multi-platform, web-based sketching system called GAMBIT [12]. Figure 12 shows the two main interfaces of the system, which allows users to sketch and share their sketches on an interactive shared workspace. The tool runs on many devices through a browser or as an embedded website (i.e. through a 'wrapper' application).

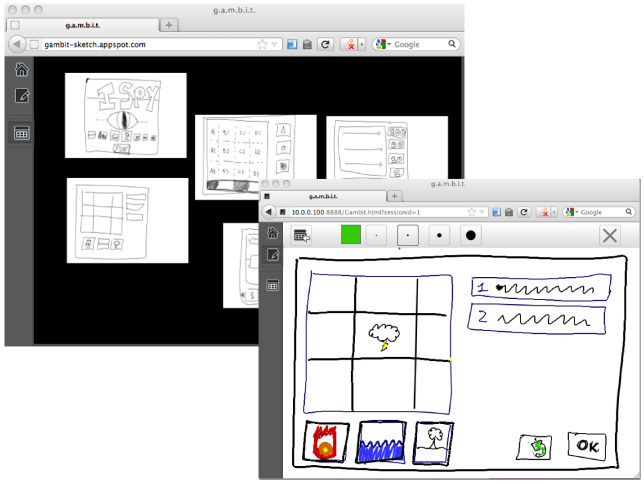

Figure 12. GAMBIT interfaces for sketch production (front) and sketch sharing (back).

This software is an essential part of a broader research on sketching, whose goal is to investigate electronic sketching usage in current UI design practices for producing and validating interactive prototypes.

The main investigation to be made with this tool is related the impact of producing possibly more complete prototypes, since GAMBIT allows the use of the same device for producing and testing an interactive prototype by using sketches.

CONCLUSION

In the electronic sketching domain, now so more than before, developers can concentrate on the sketching interface and on providing a better interaction experience relying on application libraries to provide the back-end recognition or other services [4]. In this context, a multi-platform sketching interface constructed with HTML5 would benefit both developers – for it would ideally minimize the code fragmentation for many platforms; and end-users – that would be able to perform sketching activities on the device of their preference.

The study conducted in this paper is the first of a series on sketching for multi-platform sketching systems. We have conducted an experiment to assess the perception of users regarding different response rates measured in frames per second in order to provide intervals that are accepted and rejected for sketching activity.

Among our findings, we can highlight that:

1. The range below 20 FPS was rejected by users;
2. The difference between grades is not significant above 24 FPS (receiving grades 3, 4 and 5 on the Likert scale), which can be considered as accepted;
3. We have found no conclusive evidence that subjects perceived differences of 2, 5 and 10 FPS when testing pairs of rates;

Overall, the results show that users are comfortable with a refresh rate of between 21 and 24 FPS. Following the code example in section 3, the *canvas* element should be refreshed at an interval between 41 and 43 milliseconds according to this result.

The assessed rejection range can be used on applications that allow sketching of diagrams, for instance in architecture, user interface design or software domains, for which users need mainly geometric forms.

It is possible that lag perception may vary for other tasks such as writing and playing, since they would possibly require a different kind of attention to what is being *produced*. Therefore, further experiments are necessary to investigate whether the FPS rates obtained be different for other activities, and also to precisely identify the correlation between user profiles and FPS preferences. Furthermore, it would be worth to investigate if the same results can be obtained for sketching activities with pen-based interaction in different platforms and devices (e.g. large interactive whiteboards and tablets).

ACKNOWLEDGEMENTS

The authors acknowledge the support of the ITEA2-Call3-2008026 UsiXML (User Interface extensible Markup Language) European project and its support by Région Wallonne, Direction générale opérationnelle de l'Economie, de l'Emploi et de la Recherche (DGO6) and Serenoa project that is funded by the European Union through its Seventh Framework Programme as a STREP Project no. FP7-ICT-258030. Also, the authors thank Bruno Sangiorgi, Sophie Dupuis and Brigitte Ryan for their help and all the subjects for their participation on the experiment.

REFERENCES

1. Dabrowski, J., and Munson, E. Is 100 milliseconds too fast? In *CHI'01 extended abstracts on Human factors in computing systems* (2001), 317–318.
2. Ellis, T., Heafner, J., Sibley, W., and Corporation, R. The GRAIL Project: An experiment in man-machine communications.
3. Ivan, S. Sketchpad: A man-machine graphical communication system. In *Proceedings-Spring Joint Computer Conference, Michigan*, vol. 23 (1963), 329–346.
4. Johnson, G., Gross, M. D., Hong, J., and Yi-Luen Do, E. Computational Support for Sketching in Design: A Review. *Foundations and Trends in Human-Computer Interaction 2*, 1 (2008), 1–93.
5. Keval, H., and Sasse, M. To catch a thief - you need at least 8 frames per second: the impact of frame rates on user performance in a CCTV detection task. In *Proceeding of the 16th ACM international conference on Multimedia*, ACM (2008), 941–944.
6. MacKenzie, I. Lag as a determinant of human performance in interactive systems. *on Human factors in computing systems* (1993), 488–493.
7. MacLean, S., Tausky, D., Labahn, G., Lank, E., and Marzouk, M. Is the iPad useful for sketch input? A comparison with the Tablet PC. *EUROGRAPHICS Symposium on Sketch-Based Interfaces and Modeling* (2011).
8. Mangano, N., Baker, A., and van der Hoek, A. Calico: a prototype sketching tool for modeling in early design. In *MiSE '08: Proceedings of the 2008 international workshop on Models in software engineering*, ACM (New York, NY, USA, 2008), 63–68.
9. Masry, M., and Lipson, H. A sketch-based interface for iterative design and analysis of 3d objects. In *ACM SIGGRAPH 2007 courses*, ACM (2007), 31.
10. Nogee, A. Less is More: The Worldwide Emergence of Low-Cost Android Smartphones. Tech. rep., In-Stat, 2011.
11. Read, P., and Meyer, M.-P. *Restoration of motion picture film*, 1 ed. Butterworth-Heinemann, 2000.
12. Sangiorgi, U. B., Beuvens, F., and Vanderdonckt, J. User Interface Design by Collaborative Sketching. In *Proceedings of the Designing Interactive Systems Conference on - DIS '12* (2012), 378–387.
13. Savery, C., and Graham, T. It's About Time : Confronting Latency in the Development of Groupware Systems. In *Proceedings of the ACM 2011 conference on Computer supported cooperative work*, ACM (2011), 177–186.
14. Schmieder, P., Plimmer, B., and Vanderdonckt, J. Generating systems from multiple sketched models. *Journal of Visual Languages and Computing 21*, 2 (2010), 98–108.
15. Schon, D. A., and Wiggins, G. Kinds of seeing and their functions in designing. *Design Studies 13*, 2 (1992), 135–156.
16. Shah, J., and Brown, R. M. Towards electronic paper displays made from microbial cellulose. *Applied microbiology and biotechnology 66*, 4 (Jan. 2005), 352–5.
17. Shen, H., Shimodaira, Y., and Ohashi, G. Speed-tuned mechanism and speed perception in human vision. *Systems and Computers in Japan 36*, 13 (2005).
18. Simonson, E., and Brožek, J. Flicker fusion frequency: background and applications. *Physiological Reviews 32* (1952), 349–378.
19. van Boxtel, J. J. a., van Ee, R., and Erkelens, C. J. A single system explains human speed perception. *Journal of cognitive neuroscience 18*, 11 (Nov. 2006), 1808–19.
20. van der Lugt, R. Functions of sketching in design idea generation meetings. *Proceedings of the fourth conference on Creativity cognition – CC '02* (2002), 72–79.
21. Weiser, M. Hot topics: Ubiquitous computing. *IEEE Computer 26*, 10 (1993), 71–72.